Bachelor of Science

IoT Network

AUBERT Guillaume

HESVIK Brede

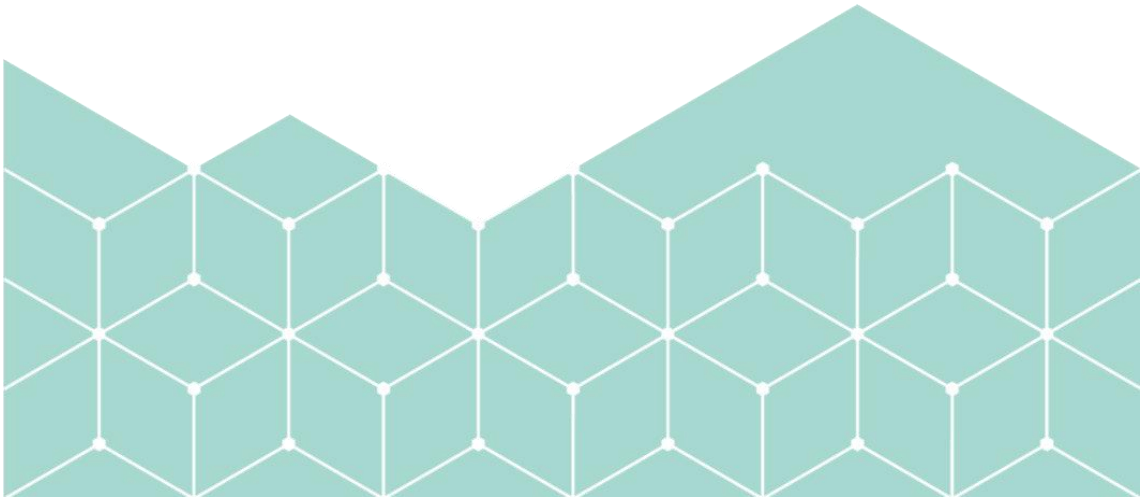
MOURIER Antoine

THOMAS Adrien

06.06.2018

Electrical engineering

IR



PREFACE

We would like to thank Høgskolen i Østfold for giving us the possibility to work on this project and Reidar Nordby for guiding and helping us throughout. We feel that it has given us the opportunity to acquire new exciting knowledge. The supervisors for this project has been Reidar Nordby and Hong Wu.

Guillaume AUBERT

Brede HESVIK

Antoine MOURIER

Adrien THOMAS

TABLE OF CONTENTS

List of Figures

LIST OF TABLES

LIST OF APPENDICES

Appendix A: *Datasheets* :

SX1272.pdf (SX1272 LoRa shield for the node)

SX1272MB2DAS_HM.pdf (SX1272 LoRa shield for the node)

iC880A-SPI_QuickStartGuide.pdf (iC880A LoRa concentrator for the gateway)

TC74.pdf (Thermal sensor)

RG058.pdf (Coaxial cable)

SX1272_schematic.pdf

SX1272_Hardware_Description.pdf

Appendix B: *Codes* :

Measure current on MCU.cpp (Mbed)

Norway.ino (Arduino Uno: sensor management and data emission)

util_pkt_logger (Raspberry Pi: data reception)

client.py (Raspberry Pi: data transmission to the server)

iC880-SPI_reset.sh (Raspberry Pi: iC880A SPI pins reset)

lorawan.sh (Raspberry Pi: to launch the 3 previous codes at once)

server.py (Server: data reception)

Norway (Website: database reading)

Appendix C *: Tutorials :*

Tutorial "How to install the Arduino Node"

Tutorial "How to install the Raspberry Pi Gateway"

Tutorial "How to install the Apache2 Server and the Website"

Appendix D *: Process documents :*

Email from Klimavakten

- *Contacted the Norwegian weather institute to ask if there is any chance to predict clouds*

Meeting request email on status update(31.05)

Meeting report

- *A report from the two meetings we have had with Reidar.*

Press release

Approvement press release email

- *Email from Reidar to verify that press release document is approved*

Measuring current on MCU

- Describe why and how we measure and the results of it.

Gantt project

- Gantt project included and describing how it differentiate from the original plan.

ACRONYMS

Table 1: Acronyms

Acronym	Definition	Page
AJAX	Asynchronous Javascript And XML	53
ASK	Amplitude Shift Keying	22
CAN	Converter Analog to Digital	45
CAD	Computer Aided Design	59
CDMA	Code Division Multi Access	25
CRC	Cyclic Redundancy Check	34
CSS	Chirp Spread Spectrum	20
CSS	Cascading Style Sheets	53
EEPROM	Electrically-Erasable Programmable Read-Only Memory	43

FM	Frequency Modulation	20
FSK	Frequency Shift Keying	22
GSM	Global System for Mobile communication	14
HDMI	High Definition Multimedia Interface	42
HTML	HyperText Markup Language	53
I2C	Inter Integrated Circuit	43
ICSP	In Circuit Serial Programming	43
IDE	Integrated Development Environment	43
IoT	Internet of Things	14
IP	Internet Protocol	74
ITU	International Telecommunication Union	47
LDR	Light Dependent Resistor	49
LPWAN	Low Power Wide Area Network	14

LoRa	Low power long Range	14
LoRaWAN	Low power long Range Wide Area Network	14
MCU	MiCrocontroller Unit	41
MSB	Most Significant Bit	39
OOK	On Off Keying	46
OS	Operating System	41
PCB	Printed Circuit Board	51
PHP	Hypertext PreProcessor	53
PLA	PoLylactic Acid	55
PWM	Pulse Width Modulation	43
RAM	Random Access Memory	42
SD	Secure Digital	42
SPI	Serial Peripheral Interface	16

SQL	Structured Query Language	52
SRAM	Static Random Access Memory	43
UART	Universal Asynchronous Receiver Transmitter	45
USB	Universal Serial Bus	43

SUMMARY

Along the course of this project, we researched the specific technology that is LoRaWAN and inserted it in a realistic application. The report addresses how to set up your own LoRaWAN system with tutorials, system costs, and explanations of the modulation technique involved, to the extent possible. The project application is a solar panels field: we need to monitor brightness and temperature changes in its surroundings in order to predict its power output.

Throughout this report we have talked about LoRa range, power consumption and cost: three essential subjects when it comes to IoT networks. For small packets transmissions, at the moment there is no better option than LPWAN to establish long-range communications while only consuming a low amount of power. GSM is one of the few long-range alternative, assuming an already deployed network, but it is expensive and consume a significant amount of power. Therefore LPWAN is the only viable technology for our project, and its most promising candidate is currently LoRa

1. Introduction

1.1 About this report

The report addresses how to set up your own LoRaWAN system with tutorials, system costs, and explanations of the modulation technique involved, to the extent possible. There have also been different attempts to reduce power consumption which have been successful. To fully understand this report, the reader should have basic knowledge about signal modulation techniques and different programming languages.

1.2 Background

The project application is a solar panels field: we need to monitor brightness and temperature changes in its surroundings in order to predict its power output, so power generation can be adapted elsewhere to balance the grid accordingly. Data from sensors are sent across several hundreds of meters using a wireless method called LoRa, and then stored on a database available on the internet via a website.

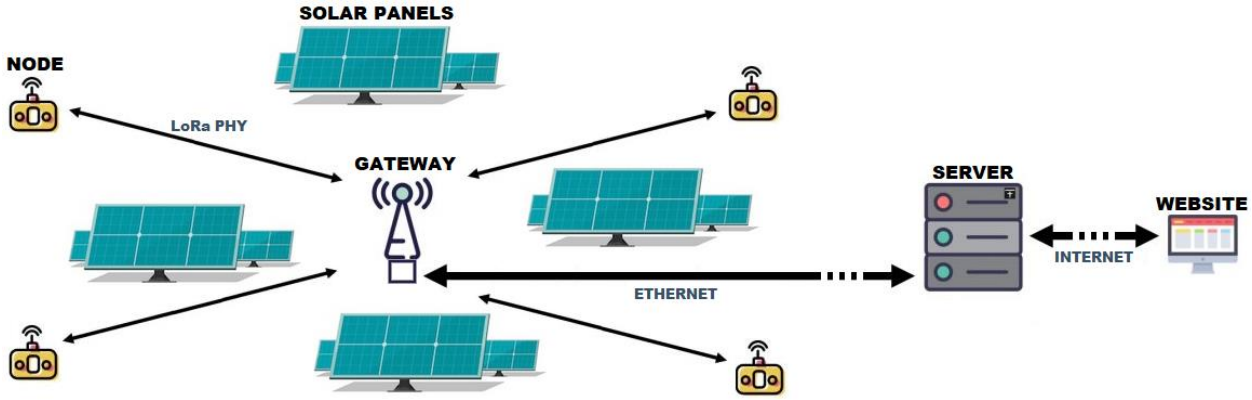


Figure 1: Field application schematic

1.3 PROBLEM

The usual problem with IoT's cellular connectivity is the price: it is quite expensive. They also have a rather short range, and batteries are running out quickly. Wifi, Bluetooth or mobile phones do not offer the abilities that we need for some specific IoT applications.

Our goal is to demonstrate how to benefit from LoRaWAN radio link technology in a scenario where it is needed to wirelessly transmit small packets of data from off-grid locations and over long distances, over the course of a few years and without having to intervene after implementation.

1.4 SOLVING STRATEGY

Our strategy was to split the technical part of our project into four consecutive stages, to simplify resolving issues as we encountered them. To get a better understanding of how we proceeded, we suggest reading these definitions specific to our project:

Node: assembly of an SX1272 shield on top of an Arduino Uno Rev 3, which permits to get data from sensors and then wirelessly send them via the LoRa communication protocol. Being at one extremity of the communication chain, the node is an "end device". Its emission is an "uplink".

Gateway: assembly of an IMST iC880A-SPI shield on top of a Raspberry Pi 3, capable of receiving data from several nodes simultaneously using multiple channels, then possibly decode it and send it to a server. The gateway act as a bridge between the front ends (the nodes) and the back end (the internet). Its emission to a node is a "downlink". Its connection to the other extreme, toward where the server is located, is a "backhaul".

1. *Node to node communication*

At first, we simply set up two identical nodes, so that one was transmitting data packets to the other. Having the same hardware on both ends simplified bugs chasing, and validated a first successful LoRaWAN connection.

2. *Node to Gateway communication*

Knowing how to correctly setup the node's emission, it was time to manage to receive the data using instead a Gateway, which is a more dedicated device for this task.

3. *Gateway to Server communication*

The following task has been to bring data from the Gateway to an Ubuntu server, to store it in a database, and to be able to read it from a website.

4. *Data encryption*

Finally, we tried to implement an encryption method on top of it all, in order further to secure node to server data transmission.

1.5 TECHNICAL BOUNDARIES / MILESTONES

- Install and set up a node using a Semtech SX1272 Embedded shield transmitter and fit it on an Arduino Uno Rev 3.
- Install and set up a gateway using an IMST iC880A-SPI receiver with a Raspberry PI 3 model B.
- Install Ubuntu on a computer, then add a server and database software to receive and stock data from the gateway.
- Reverse engineer LoRaWAN to learn as much as possible and go in depth of it.

2. THEORY

2.1. GENERAL

2.1.1 REGULATIONS OF USE OF FREQUENCY

Norwegians and Europeans telecommunication laws have elaborated regulations that deal with wireless communication systems, frequencies and bandwidth restrictions and broadcasting energy limits. The Norwegian law, Chapter 8 on short-range communication, dictates that the maximum allowed radiated power is 25 mW (14 dBm), and that the maximum transmission time (duty cycle) has to be less than one percent.

2.1.2 Chirp Spread Spectrum

Pioneered in the early twentieth century, Chirp Spread Spectrum systems were primarily developed for military applications by the US army during the mid-sixties, to provide radars with anti-jamming and low probability of interception features. Spread spectrum techniques are digital transmission methods by which a sinus signal is deliberately spread in the frequency domain, resulting in a signal using a wider bandwidth. They are used for a variety of reasons: establishment of secure communications, increase resistance to interferences such as natural noise or jamming, it is hard to detect and limits power flux density. Frequency-hopping spread spectrum may date back to radio pioneer Jonathan Zenneck´s 1908 German book *Wireless Telegraphy* (source: Spread spectrum Wiki and *principles of mobile communication* by Gordon Stuber).

Sidney Darlington patented the use of chirp modulation in digital communications in 1954, but M. R. Winkler has done significant later work in 1962.

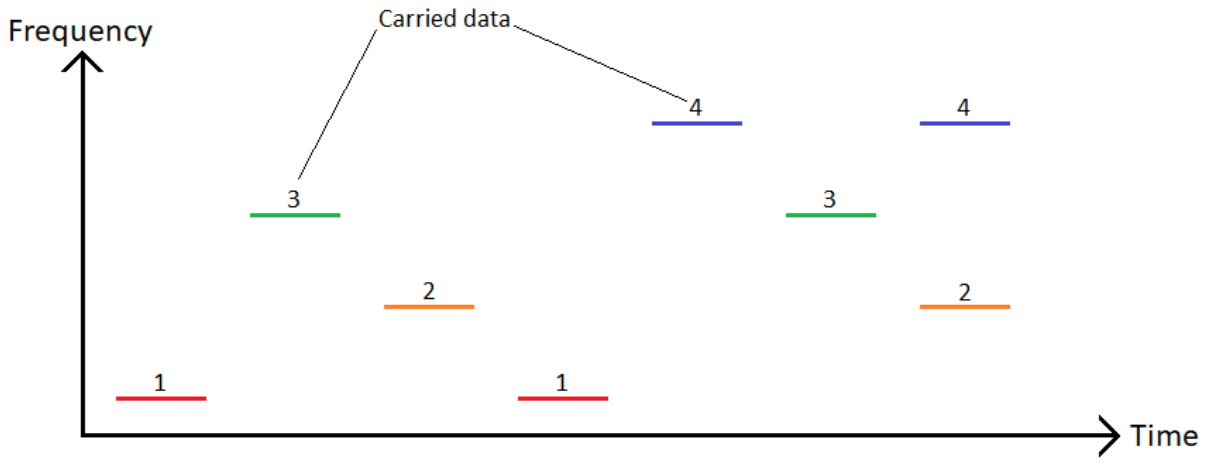


Figure 2:Example of frequency hopping

In this picture, we can see an example of frequency-hopping: different frequencies correspond to different data.

The general class of linear chirp signals is described by :

$$ci(t) = \sqrt{\frac{2Ei}{Ti}}\, cos(2\pi fit + \pi a_i t^2)\square \quad 0 \leq t \leq Ti$$

where

$$a_i = \frac{W_i}{T_i}$$

T_i : i^{th} *signal duration* W_i : i^{th} *signal bandwidth*

F_i : i^{th} *carrier frequency* E_i : i^{th}*signal energy*

T_i : i^{th}*signal duration*

[21]

Linear Frequency-Modulated (FM) signals, or equivalently linear chirp signals, have been used extensively in radar technology. Because they are Doppler resilient, they allow accurate range and velocity measurements. The precision depends on the parameters of the linear chirp signals, which includes chirp rate: the first derivative of chirp frequency, used to balance signal robustness and data rate.

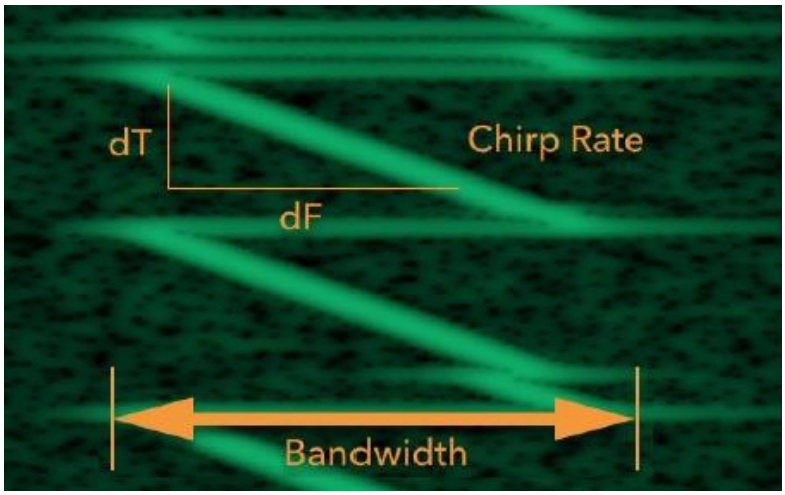


Figure 3: Chirp rate[1]

Chirp signals occupy a wide transmission bandwidth where its instantaneous frequency increases or decreases linearly over time. To represent different data symbols, CSS uses linear FM signals of distinct length by going from one end of the bandwidth to the other coupled with the use of jumps back in frequency.

CSS is more robust to interferences than classic FM modulation techniques such as direct frequency-hopping, although it often implies lower data rates or the use of a broader bandwidth.

2.1.3 Comparison to other modulation techniques

Spread spectrum systems are methods by which a signal is generated with a predetermined bandwidth spread in the frequency domain.

Interference and jammer rejection:

Interference is usually caused by transmission signals of other users or systems, or by our own dephased signal (destructive interference) when it has been reflected by obstacles (walls, ground, trees, etc). Usually, the interference has a distinct

frequency. The reason why LoRa is so great at interference rejection is that its signal stretches over a defined bandwidth. After demodulation, a distinct spurious frequency will have almost no impact on LoRa data readability.

LoRa demodulation ("de-chirping") consists of multiplying the signal with chirps of opposite direction. By defining the center of the used bandwidth as zero, we can use the following properties to transform the chirps into a more familiar form similar to frequency hopping: $f_0 \times f_1 = f_o + f_1$, and thus $f_o \times (-f_0) = 0$

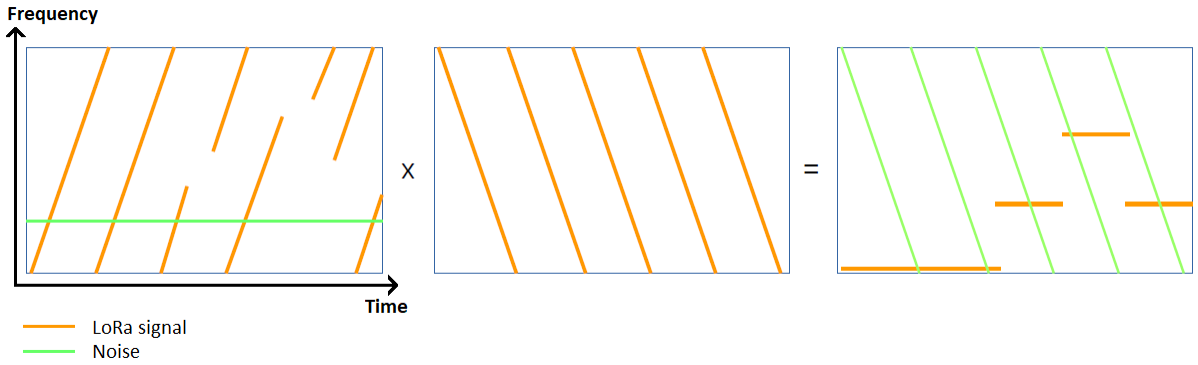


Figure 4: Showing LoRa demodulation and interference rejection[2]

We can see that even polluted by noise; the signal is still easily identifiable.

Jamming is generating a signal to disturb a transmission channel intentionally. For the same reason mentioned above, a LoRa signal can sustain a significantly amount of jamming before seeing its bit error rate increase.

LoRa's CSS (a form of Frequency Modulation) has a high interference rejection compared to other modulation techniques, like for example ASK (a form of Amplitude Modulation) which is very susceptible to noise interference.

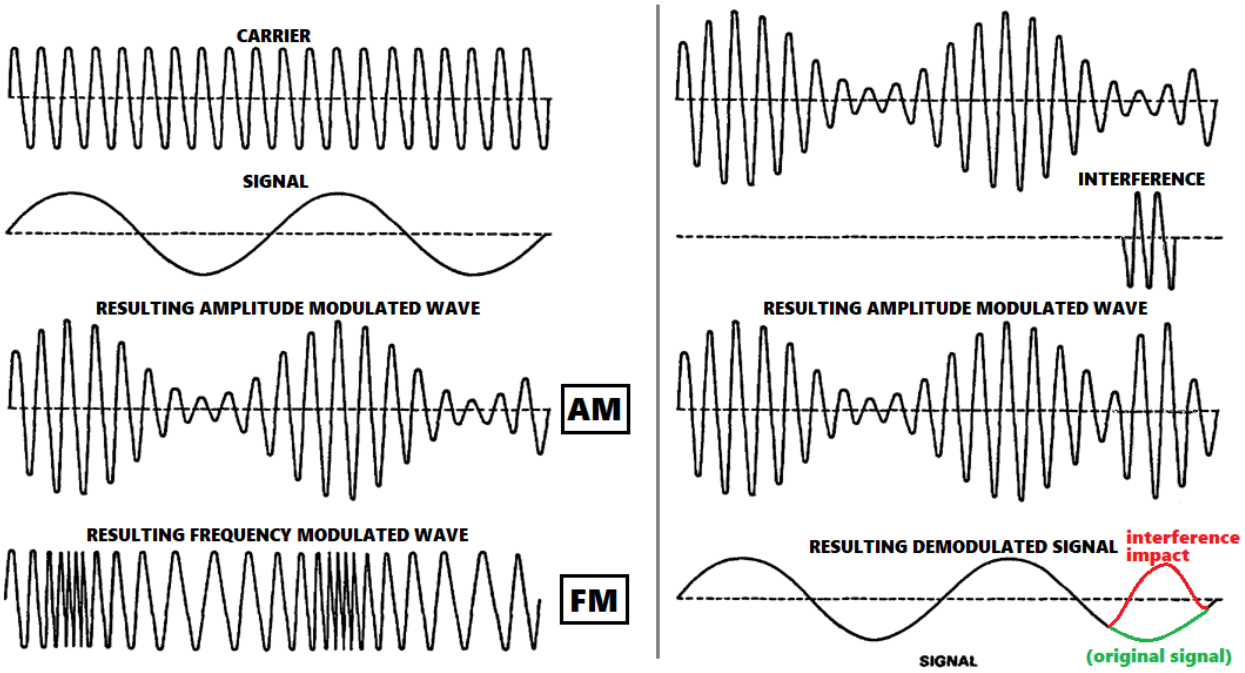


Figure 5: Amplitude modulation compared to frequency modulation

In the picture above, we can see that an AM signal is dramatically impacted by amplitude spikes, which are the most common kind of interferences. While the FM signal would also see its amplitude modified by these interferences, it would have no significant impact since the data is only carried by the change in frequency. That said, FM has the disadvantage of needing a broader bandwidth, and AM has a extended range because it can use lower frequencies more efficiently.

FSK and OOK have issues with interferences of nearby frequencies, disrupting the original signal by overlaying it, possibly making data recovery impossible:

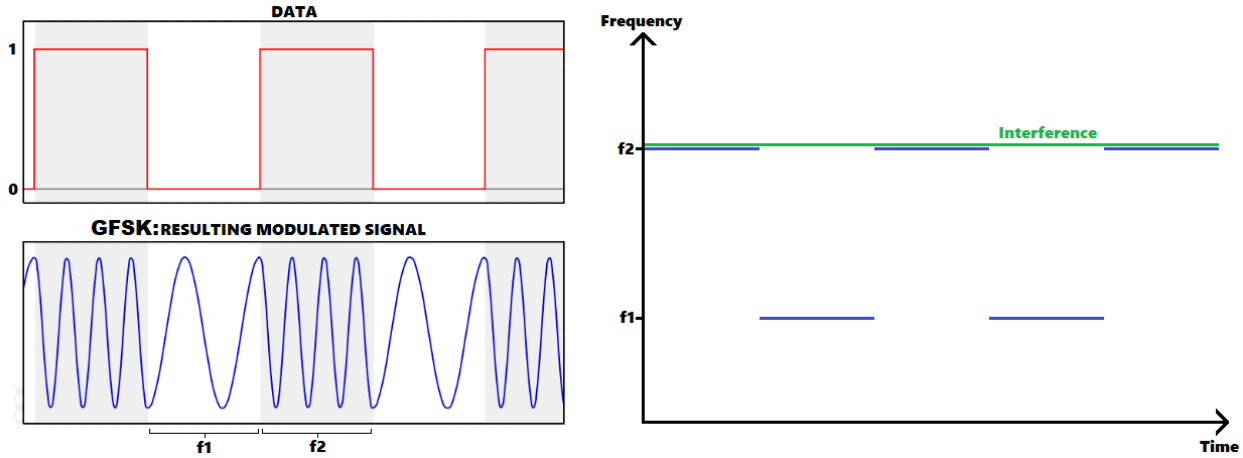


Figure 6: FSK signal and interference

In the Gaussian FSK (used in GSM) example above, it will not be possible to determine when the f2 frequency emits if the receiver is not extremely sensitive.

Multipath Suppression:

When a signal or packet reaches the receiver input from more than one path, it is called multipath. These signals are added together, and depending on the phase difference between them it can produce destructive interferences, according to the Huygens principle. However, if a signal is delayed by more than what a certain threshold is, the delayed signal is then identified and suppressed.

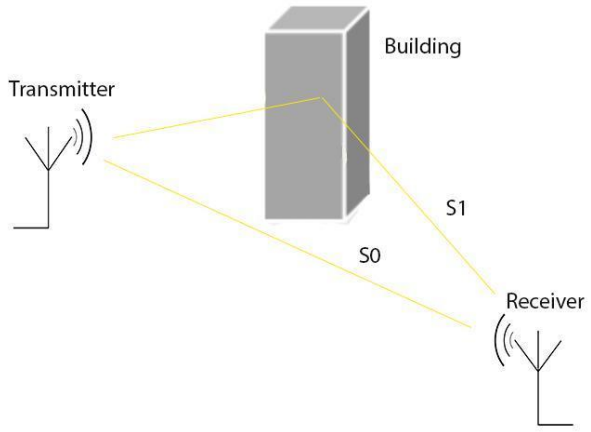


Figure 7: Example of multipath

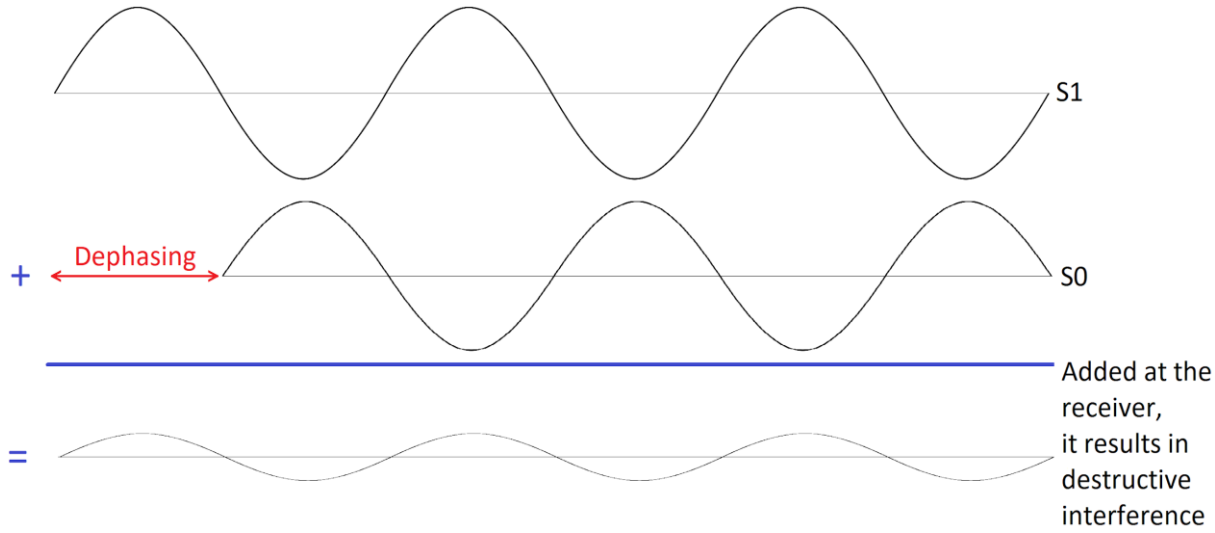


Figure 8: Example of destructive interference

There is at least one source of reflection that is impossible to avoid in the ground to ground telecommunication: it is the ground itself. These communications can be optimized for modifying the heights of the transmitter and the receiver according to the distance separating them, using the principles of the Fresnel Zone.

Code Division Multiple Access:

CDMA allows several transmitters to communicate simultaneously to the same receiver while using the same frequency band, as long as they each have their distinct code sequence.

Each transmitter gets a unique bit combination called Spreading Code, which will be used to transform their messages. The amplitude of these coded messages can then be added, producing a single signal. If the receiver has the corresponding spreading codes, it will be able to separate each message and decode them through computation. If the key used by the receiver is not the same as the one used by the transmitted signal, it will result in a null signal.

The next picture is an example of two data signals each composed of two bits, which are then coded with different four bits spreading codes:

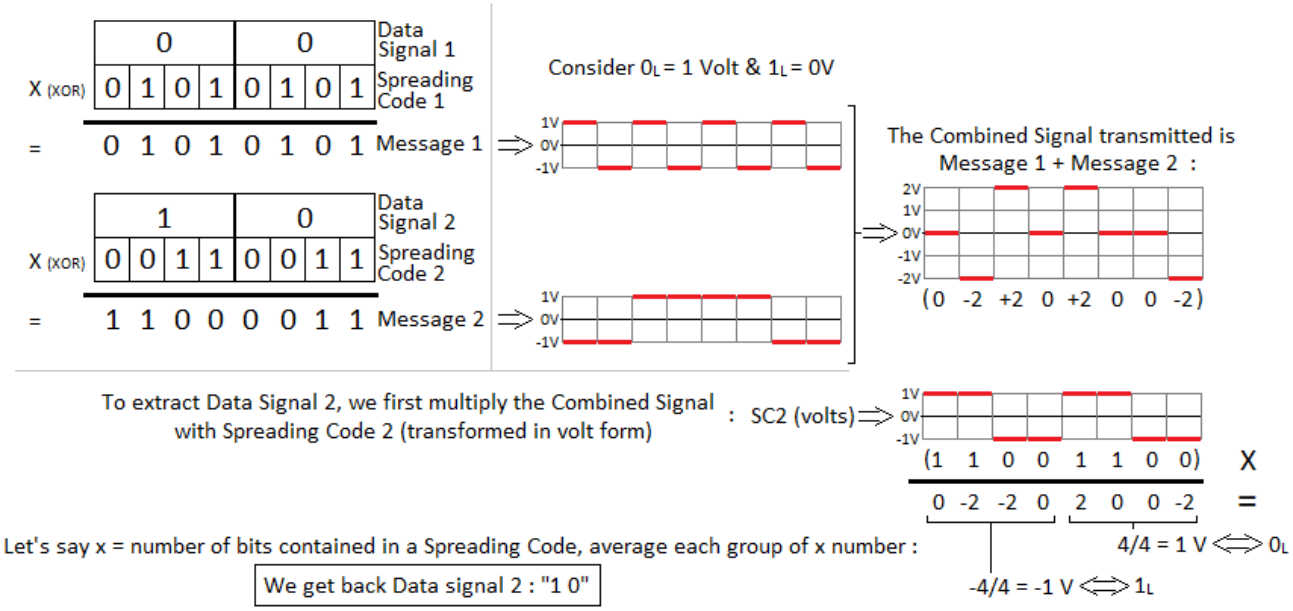


Figure 9: Example of CDMA

Spreading codes have to be orthogonal to each other. Orthogonal means that their product will result in a null signal. According to the Walsh matrix, *n* bits can produce *n* orthogonal spreading codes. In other words, while Data Signals

processed with CDMA get their length multiplied by *n*, a single Combined Signal could carry up to *n* different messages. CDMA's cost mainly lies in the computation needed to code and decode signals. N.B. while in the previous example a modified amplitude is used, in the case of LoRaWAN the data signal of different data rates is spread in the frequency domain.

Selective addressing:

Allows one transmitter to communicate with one particular receiver out of several receivers in a network, just by selecting the spreading code used by that receiver.

Low-Detectability Signal Transmission:

The signals may be transmitted at a very low power level relative to the background channel noise, which not only prevents interference to other systems but also provides a lower probability of being intercepted. The sensibility of the gateway's comparator/integrator is then the primary factor for the separation of multiple channels.

2.2. LoRa modulation technique

2.2.1 What is LoRa

LoRa is a rapidly expanding Low Power and Long Range signal modulation technique which is based on Chirp Spread Spectrum. It is an excellent choice in an IoT network, as it offers outstanding range compared to its low power consumption. By changing a variety of properties, you can choose between more extended range or higher data speed. LoRa is divided into two parts: *LoRa PHY* is the physical layer, and this technique is patented by and property of Semtech. *LoRaWAN* is the MAC layer protocol used to manage communication between gateways and nodes, and to manage the communication frequencies,

data rate, and power. It is maintained by the LoRa Alliance, which Semtech is part of.

On the picture below you can see LoRa compared to other wireless technologies. The choice depends on whether you need more range, more bandwidth or lower power consumption. Higher bandwidth or higher range means higher power consumption.

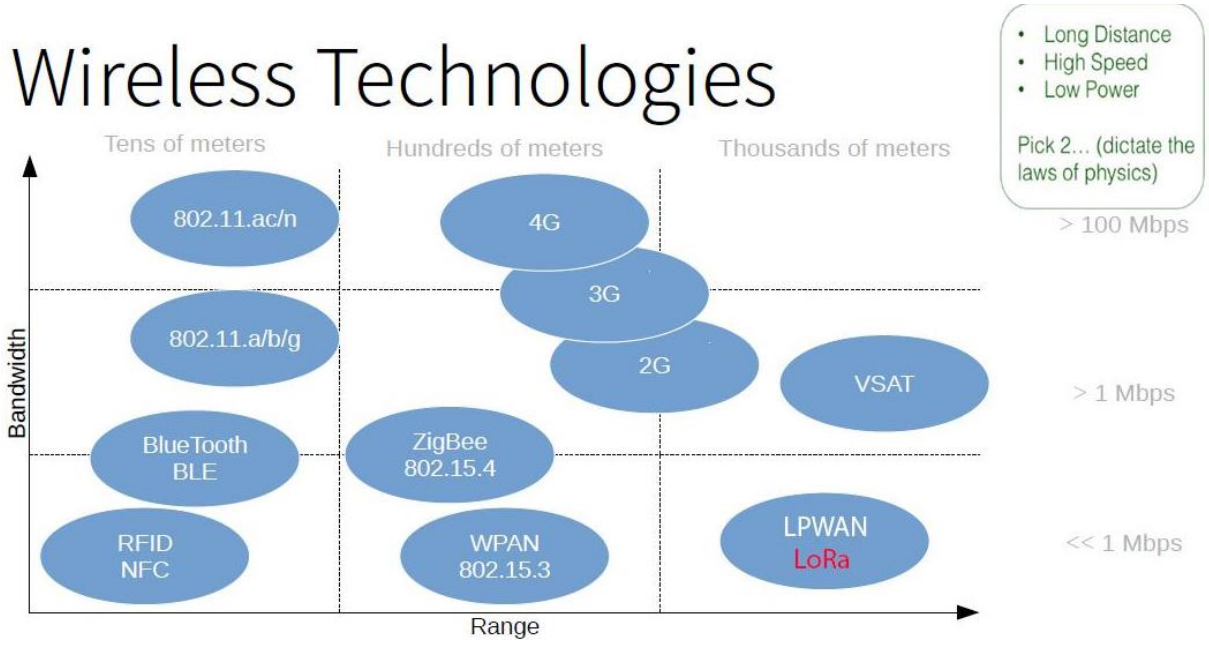


Figure 10: Wireless technologies available[3]

As its name states it, LPWAN, which includes LoRa, is the best option for low power consumption and long-range applications, if a low bandwidth capability is sufficient.

2.2.2 LINK BUDGET

The Link budget is summing all the signal strength gains and losses in a telecommunication system: from the transmitter, through the medium, to the receiver. It accounts for the attenuation of the transmitted signal due to propagation, as well as antenna gains, and miscellaneous losses.

To moderate the manipulation of large numbers, a logarithmic scale is used to measure the signal strength. Power is therefore expressed in dBm (the "m" standing for the power reference : 1mW) : dB = 10 log(measured power/1mW)

Example of link budget equation:

PRX = PTX – LTX + GTX – LFS – LM + GRX - LRX

where:

PRX : received power (dBm)

PTX : transmitter output power (dBm)

LTX : transmitter losses (coax, connectors...) (dBm)

GTX : transmitter antenna gain (dBi)

LFS : path loss, usually free space loss (dBm)

LM : miscellaneous losses (destructive interferences, fading margin, body loss, polarization mismatch, etc) (dBm)

GRX = receiver antenna gain (dBi)

LRX : receiver losses (coax, connectors...) (dBm)

The picture below shows a simplified transmission link budget:

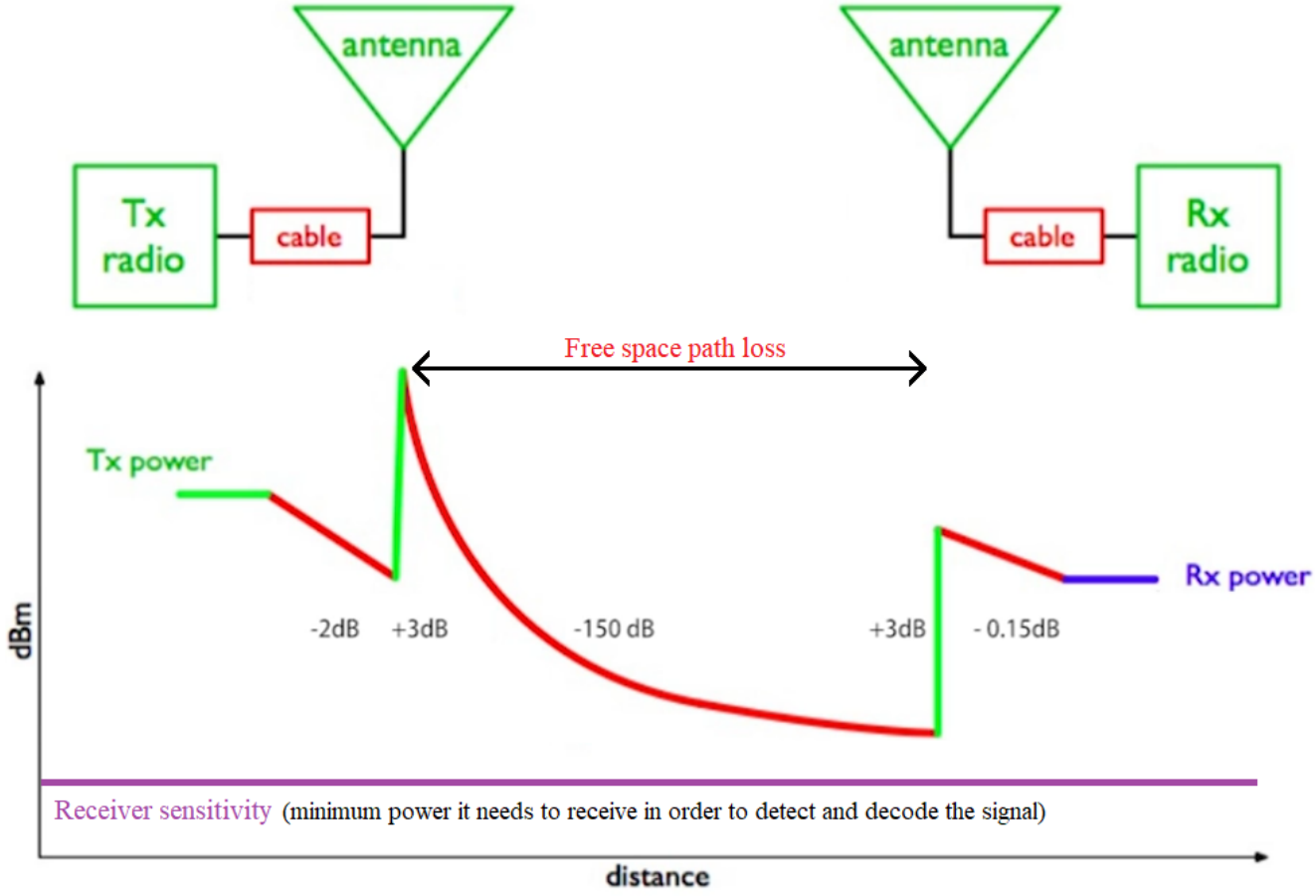


Figure 11: Simplified link budget schematic[22]

Note that the antenna gain is not due to energy being added in the system since antennas are passive components, but it is because they are directive and concentrate the power generated by the transmitter in specific directions. The same process occurs at the receiving point.

The free space path loss is proportional to the square of the distance: doubling the distance results in a 6 dBm loss. In Europe, the maximum transmitting power allowed in the band used by LoRa is 14 dBm, equivalent to 25 mW.

[4]

2.2.3 LoRa technical explanation

Most LoRa devices work in an asymmetric mode. LoRaWAN has a star network topology, where the nodes communicate with a gateway, and not between each other.

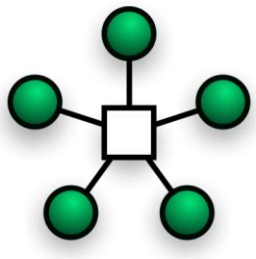


Figure 12: Star network example. Green circles are nodes, the white box is a gateway[5]

The gateway can then pass on the information received, for example to a database or a website, using ethernet, wifi or cellular. It can receive multiple parallel signals of different central frequencies. Its capabilities to receive different messages at the same time can also be dramatically increased using CDMA algorithms; the limitation would then be the computational power needed to do so.

Explaining the LoRa signal:

LoRa uses chirps symbols to carry information. A chirp is a linear frequency modulation starting at frequency f1 and ending at frequency f2. Let's call a straight chirp without a frequency shift, a *base chirp*.

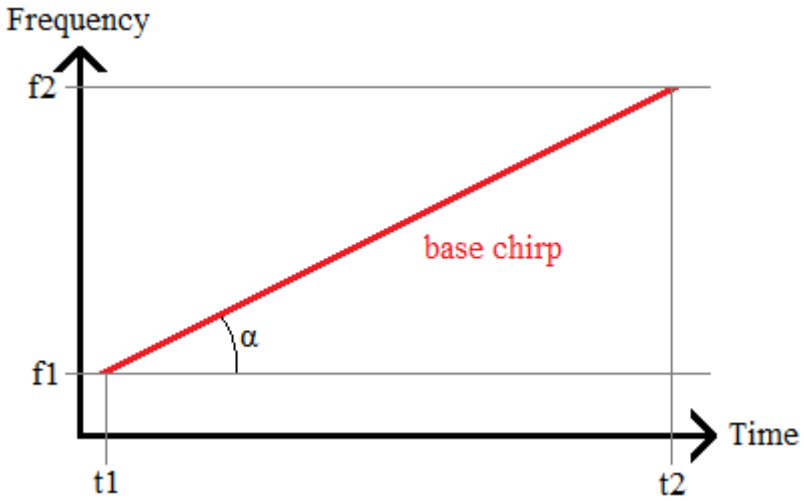


Figure 13: LoRa signal base chirp

Contrary to e.g. OOK, the CSS technique is capable of conveying more than one bit of data per symbol generated. It is achieved through instantaneous frequency shift, thus delaying the time at which f2 is reached. Two factors are used to differentiate one symbol from another: the moment t2 when the shift occurs, and the scope of the jump back in frequency *fs* inducing the length of the chirp t3-t1.

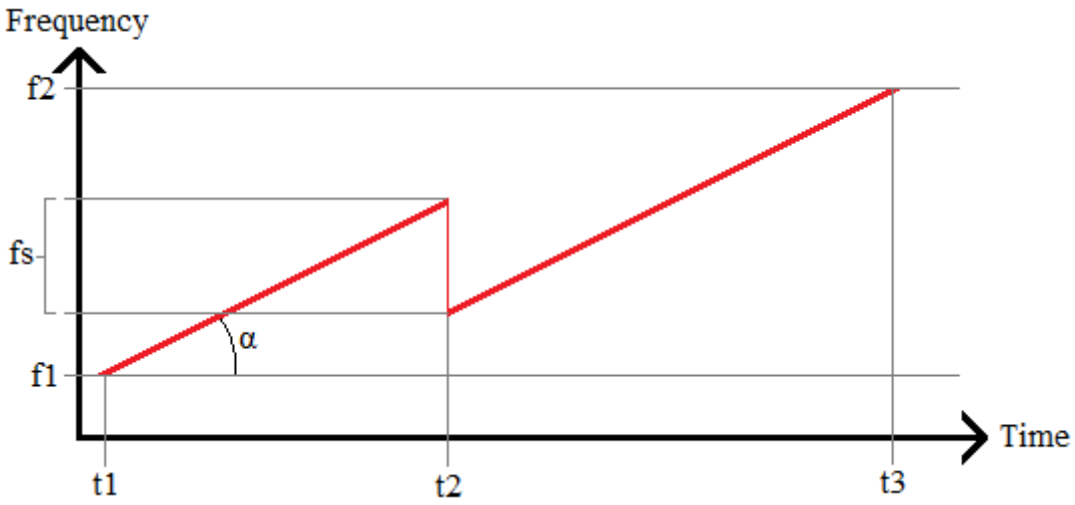


Figure 14: Example of frequency shift to represent a different symbol

Since there are physical limitations about the precision at which these changes can be made and detected, there are limitations to the number of different symbols possibly created. Nonetheless, it is possible to extend this number of

available symbols by modifying the alpha angle: Spreading Factors are multiples of the shortest base chirp rate.

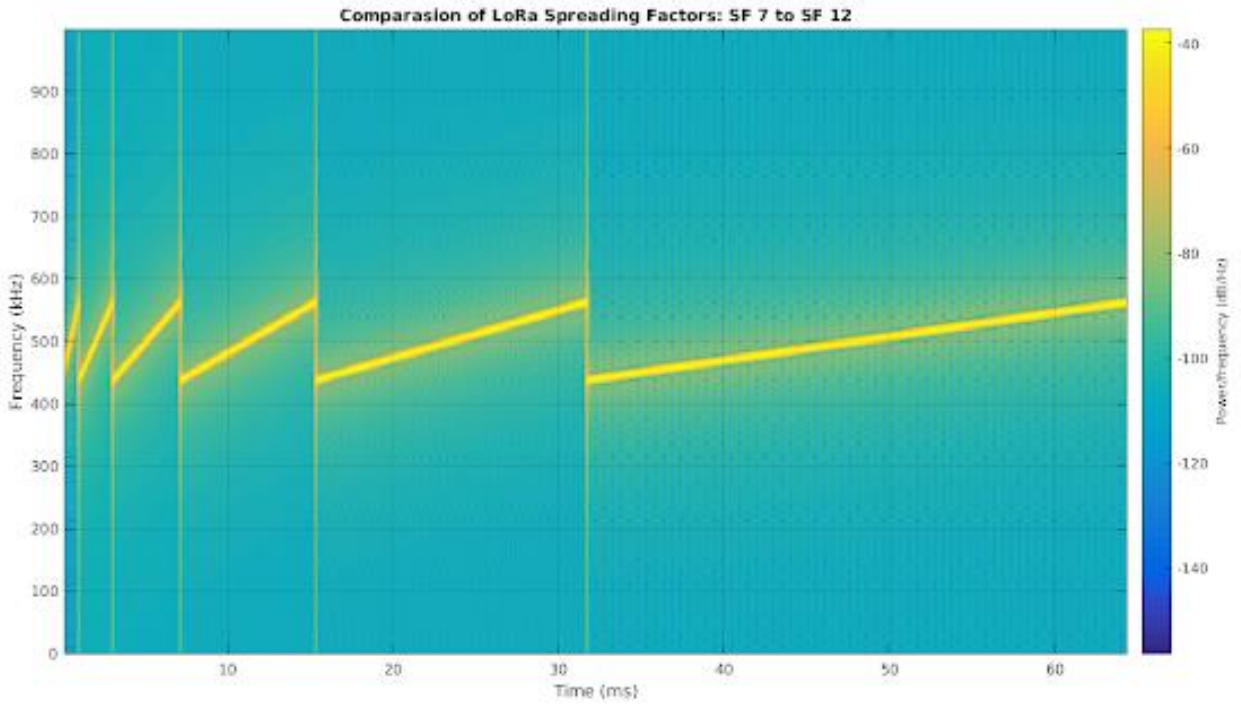


Figure 15: Comparison of the different LoRa spreading factors [18]

Incrementing the spreading factor by one is equivalent to doubling the time length of a base chirp, and therefore doubling the number available symbols. Semtech standard is defined like this: an SF7 chirp carries seven bits of data (128 available symbols), an SF8 one carries eight bits of data (256 available symbols), and its base chirp is twice as long, and so on… Similarly, doubling the spread of the frequency spectrum (f1 to f2) will potentially double the number of available symbols, e.g. 125 kHz to 250 kHz. Semtech standard only proposes to choose between 125, 250 and 500 kHz, the latest one being unauthorized by law in Europe.

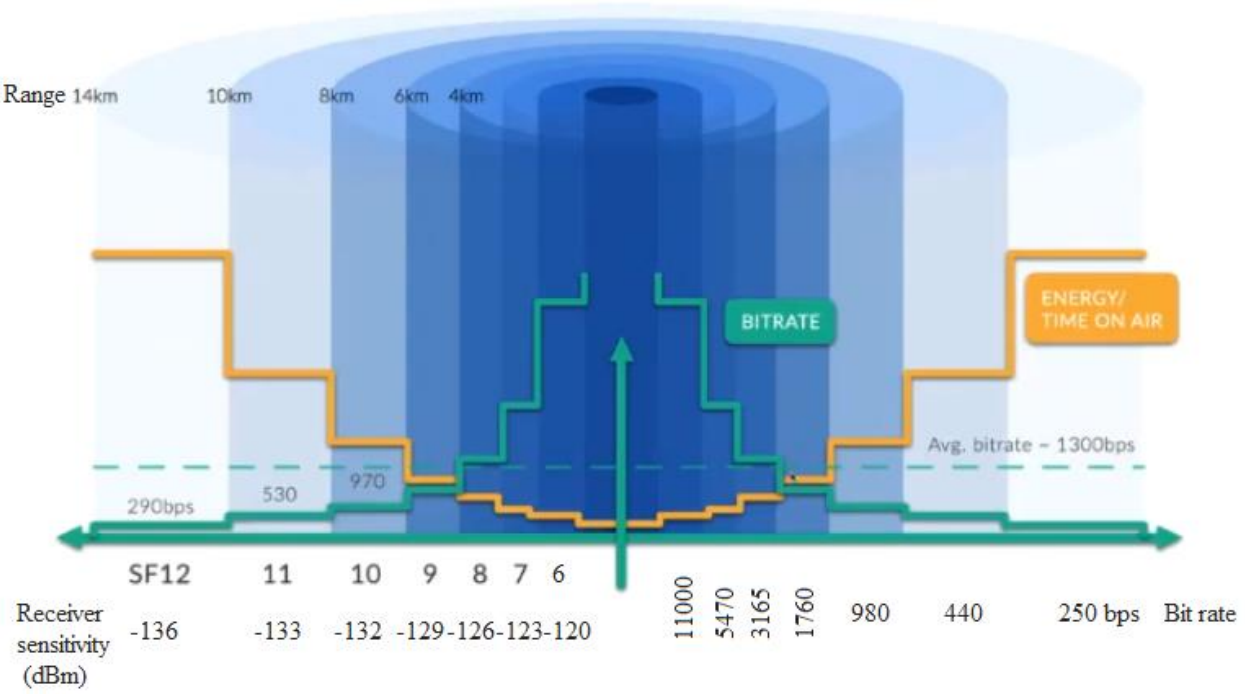


Figure 16: Spreading factors effect on range and bandwidth [23]

A LoRa signal is composed of severals chirps:

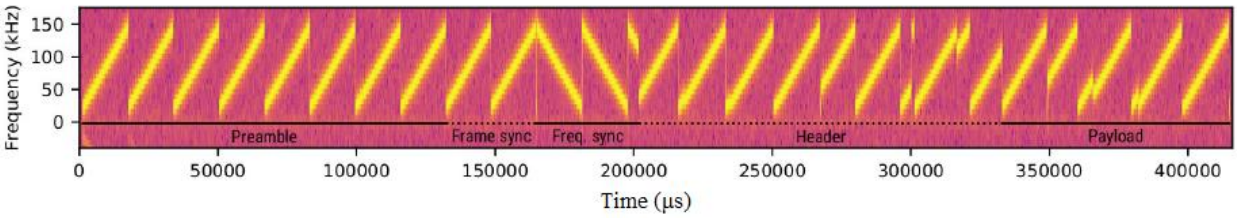


Figure 17: Example of LoRa signal [24]

Preamble and frame synchronization:

it informs the receiver of the start of a LoRa signal and permits it to measure the bandwidth and the spreading factor utilized. It consists of several base up-chirps, whose number can be modified (minimum: 2).

Frequency synchronization:

it is a sync word composed of two and a quarter base down-chirps, to inform the receiver when the data packet begins.

Header:

It is optional. It contains information about the message: if there is a CRC, and what is the of payload length and its Coding Rate. The header itself is written with a CR of 4/8.

Payload:

It contains the actual data.

2.2.4 LoRa communication classes

LoRa has a three classes system: A, B, and C. All LoRaWAN transmitters are at least class A capable. It is set by the user depending on his needs, if his device's hardware allows it: for example if the system runs on battery, the most economical class may be preferable. Or on the contrary if it is plugged, the class that has the most transmission options may be the most suitable.

2.2.4.1 CLASS A

It is a pure ALOHA system. This means that nodes send whenever they want. Nonetheless, two short windows are opened afterward to receive an acknowledgment from the gateway. If no acknowledgment is received and the node is set to expect it, it will then resend the data. These response windows are predetermined, usually placed one and two seconds after sending data. Being the most energy efficient class, devices using it are usually battery powered.

2.2.4.2 CLASS B

Class B devices are capable of receiving GPS signals, and so they can synchronize: every 128 seconds all gateways transmit beacons specifying to the nodes the moment and time duration of an additional listening window. They can then use these windows to trigger emissions from the nodes.

2.2.4.3 CLASS C

Its hardware allows the node to listen constantly; therefore the gateway can initiate a node emission at any time. By the induced shortened latency, it is ideal for communications requiring many downlinks. Because it drains much of energy, class C devices are preferably plugged into the power grid.

2.2.5 LORA SECURITY

LoRa has a few different methods to increase security. Although it is more difficult to intercept, it has nonetheless some encryption keys that are predetermined.

LoRaWan network requires two keys to operate. Keys can be pre-configured or Over The Air, where one goes from node to gateway and another one from node to the application, All using 128 bit AES encryption based on 802.15.4 security.

The different keys have different tasks:

- *Network session key* (NwkSKey) is a key used by the network server and end device to calculate and verify message integrity code for all data messages to ensure data integrity, i.e. secure data to be unscathed. Unique per end-device

- *Application Session Key* (AppSKey), the key used by the network server and the end device to encrypt and decrypt the data tax payload field. Unique per end-device.

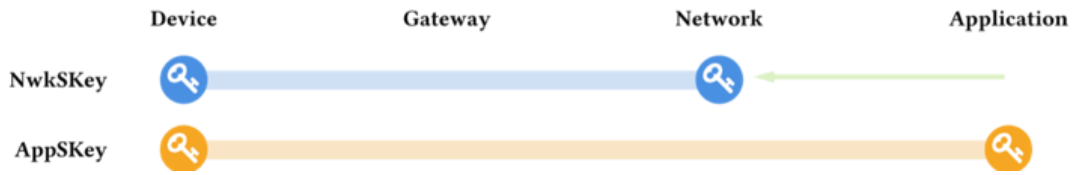


Figure 18: LoRa keys[6]

Beside encrypting, LoRa also adds layers of encoding along its process, in that order:

- *Data whitening* of the payload, to prevent long strings of 0's or 1's which could be interpreted as DC bias by the receiver.

- *Hamming algorithm* is introducing parity bits for Forward Error Correction.

- *Interleaving* to scrambles the output of the FEC algorithm, so the code gets more resistant to burst errors, i.e. corrupted packet which has more than one erroneous bit. That is because it makes bit errors easier to correct by the FEC probabilistic mechanism by spacing them out.

- *Gray indexing*, so that two successive values differ in only one bit, adding error tolerance.

Trying to retrieve the data mischievously would require knowing the exact settings of all these steps. While some of them are partially described in Semtech patents, some other are more tricky to reverse engineer according to the conferences about this particular subject that we studied.

2.2.6 Error detection, encryption & Forward Error Correction

2.2.6.1 CRC

Because transmission bitrate and data processing is not an issue in our case, we chose to activate CRC algorithm proposed by our LoRa device. CRC is an error-detecting code, working by comparing data obtained through a hash procedure. It is an evolution of checksum: the transmitter sends with the packet a number obtained using a mathematical formula (most often based on polynomials) on the payload. The receiver, knowing the formula used, compares this number to the payload received to see if they are coherent and thus deducing if the payload is valid. It exists a probability of having both the CRC number and the payload corrupted but yet giving out a coherent comparison, but it is diminished by using more complex formulas, at the cost of additional computational power needs.

2.2.6.2 Convolutional encoder

To ensure correct data transmission and to improve security, we coded a single parity bits convolutional encoder software based on the Viterbi algorithm. Using a logic chain of bit creation, doubling the length of the message in the process,

it enables the recovering on the receiving end of bit errors occurring during the transmission (due to, e.g., interferences). The process also adds a degree of encryption, making it difficult for hackers to deduce the meaning of the raw data: to decipher the message, it is needed to know how the logic gates (XOR in our case) are placed. The model we used is shown below:

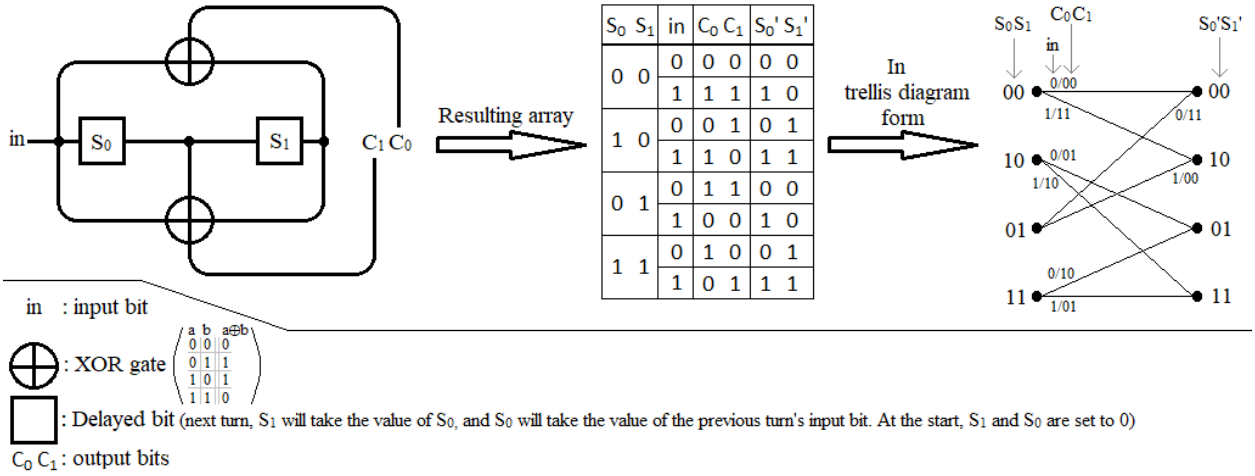


Figure 19: Model of the convolution encoder

While the encoding process is straightforward, to get a better understanding of the decoder, we will use an example: let us suppose it receives the encoded binary message "0001011010111". With a graphical representation utilizing the trellis pattern deduced earlier, the decoding process would look like this:

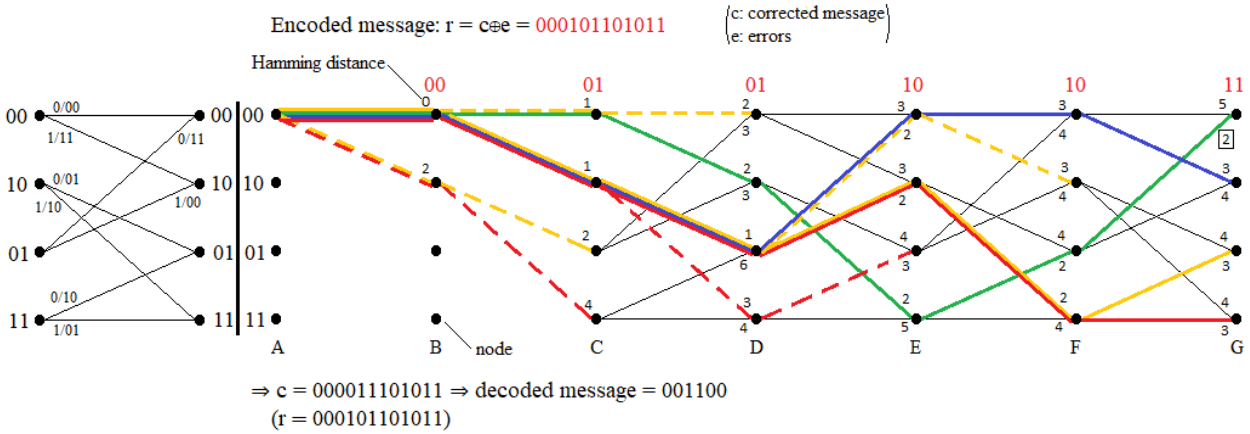


Figure 20: Decoding example

Commencing at position A, up to position G, the goal is to reproduce the encoded message as close as possible. At each position and for each possible "path", we will compare the relevant portion of the received message to the result produced. Each bit difference will then be summed along the way: it is the *Hamming distance*. The principle of the convolutional encoder is that at the end, it will choose the path with the shortest Hamming distance. That means that it will bet that the received message has not strayed away much from the intended one.

Since S0 and S1 are both set to 0 at the beginning, that is the starting point of any possible encoding. From this configuration, only 00 or 10 can be produced as S0 'S1', and respectively 00 and 11 as outputs. 00 and 11, when compared to this part of the received message (its two MSB: the bit couple 00), produce respectively a Hamming distance of 0 and 2. The same process is then repeated at each position.

When two paths collide at the same node, the one with the shortest resulting Hamming distance will go on, and the other one can be terminated since we are looking for the least different encoding from the received message. It is possible that a path is terminated on both of his available paths. That is what is represented

by dashed lines. It will then copy up to this point the behavior of the surviving path it was fighting, and separate from it right after.

From position C and on, there is four surviving paths at each step. It implies that we will obtain four candidates to choose from for the decoded message at the end, thus why we represented this graphic using only four different colors.

For our example, the green path is the one producing the shortest Hamming distance, or in other words the one having the least amount of corrected incoherence. Therefore the machine bets that this is what the received message should have been. Finally, the only part left is to decode it, which is trivial.

Notice that if the received message finished at position F, there would have been two paths having the shortest Hamming distance, but each giving different decoded messages. So, there are limitations to repairing damaged transmissions using a convolutional encoder, which involves a ratio equal to the data length multiplied by the number of convolutional bits then divided by the number of errors. To recover the correct data, this ratio has to be above a certain threshold. There is also the probability that the decoder will make a wrong estimation of the original symbol: it is the pairwise error probability. Discussing these topics further requires advanced mathematics that are beyond the spectrum of this report.

2.2.7 LoRa competitors

To this day, LoRaWAN main competitor is Sigfox. While it is also an LPWAN, they differ in some respects: it is doesn't use Chirp Spread Spectrum frequency modulation, but Differential Binary Phase-Shift Keying for uplink and Gaussian Frequency Shift Keying for downlink. Link budget and bit rate are not as good on the downlink than on the uplink, contrary to LoRa which has symmetric transmissions abilities. It operates on an Ultra NarrowBand, thus mitigating the

misfortune of encountering interferences better, but it is not as robust as CSS when it does happen. All in all, they have quite similar performances.

While Semtech business model is to sell chips or license the LoRa technology to constructors, Sigfox shares for free their designs. Another difference is that Sigfox is deploying the network itself, using expensive gateways, and is acting as an operator: to use a Sigfox device you have to pay monthly a low subscription fee. On the other hand Semtech relies on the community to build the network: notably, The Thing Network is a community-based initiative to build a global LoRaWAN by centralizing gateways.

Another promising IoT LPWAN actor is NB-IOT. It is a narrowband radio technology standardized by a collaboration of telecommunications standards associations called 3GPP. It is deployed in the LTE bands. While still in its infancy stage, the weight of the companies involved in this project makes it a notable future competitor to LoRa and Sigfox.

3. TESTING FOR DEVELOPMENT

Power consumption measurement tool:

See Appendix D "Measure current on MCU".

Voltage drop tests for sensors:

We plotted the data given by the light and temperature sensors in a constant environment while lowering the voltage powering them, to simulate an eventual voltage drop from batteries powering the node as they empty:

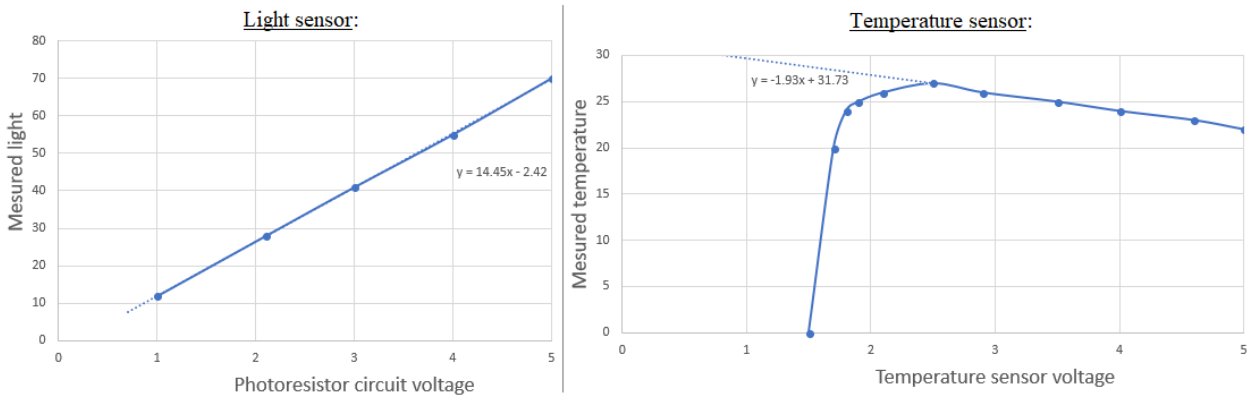


Figure 21: Sensors data with respect to voltage powering them

The Arduino code has then been adapted to adjust the sensors values according to these voltages.

4. IMPLEMENTATION

4.1 EQUIPMENT

4.1.1 RASPBERRY PI 3 MODEL B

The Raspberry Pi 3 Model B is a mono card nano-computer. It has a 1.2 GHz 64 bits ARM architecture quad-core processor, 1 GB of RAM, four USB ports, one SD card port, one ethernet port, and one HDMI port. It also has an integrated Bluetooth and WiFi node.

The reason we are using a Raspberry PI 3 is that it is a low cost, low power device with all the necessary equipment we need for our gateway.

It allows the execution of several variants of the GNU/Linux-Debian free Operating System, as well as some Microsoft Windows ones. We chose to install Raspbian because it is the most used OS on the Raspberry Pi; therefore it allows us to access to many documentations and thus to solve problems more easily. We also chose

it because we already worked with it, and also because using a Linux distribution gives us the most control over the OS.

We used two programs for the Raspberry Pi. The first one, created by the manufacturer of the LoRa concentrator, is written in C language and drives the concentrator: it allows it to receive the data from the node, which is then stocked on a .csv file. We wrote the second one in Python: it reads the data from the .csv file, sorts it out, and then sends the right information to the server. We used the Python language because it is a high-level language which allowed us to manage networking directly than if we had used C.

Figure 22: Raspberry Pi 3 Model B [7]

4.1.2 Arduino Uno Rev 3

The Arduino Uno Rev 3 is a programmable board build around the ATmega328P, a microcontroller of the AVR's family from Atmel company. It has fourteen digital input/output pins (whose six can be used as PWM outputs), six analog input/output pins, a 16 MHz quartz crystal used in a Colpitts oscillator in order to create a clock for the microcontroller, a USB connection, a DC barrel jack power connection, an ICSP header and a reset button. It is able to use I2C and SPI. It has a 32 Ko of flash memory, 2 Ko of SRAM memory and 1 Ko of EEPROM.

This board can be programmed with different languages (C/C++, Java, Assembly..) or directly in compiled code (binary or hexadecimal) and with different software (Arduino IDE, Atmel Studio, AVR Studio, etc).

We chose to use the Arduino Uno Rev 3 because there was an available library to drive the LoRa emitter (SX1272), and furthermore we were already familiar with it.

It is the heart of the node: it gathers the sensors data and drives the LoRa emitter. We chose to program it in Arduino language (a mix of C, C++, and Assembler), using the Arduino IDE, because it is accessible and thus facilitate coding. However, if we had needed to optimize the code further, we would have used another environment such as Atmel Studio, better for debugging and quicker once mastered. Unfortunately, its learning curve is such that it would have slowed down the project too much.

The code uploaded into the Arduino allows it to read the external luminosity and temperature, and to measure the actual voltage of its 5V line. It then sends these data to the LoRa shield, which encrypts it before sending it wirelessly in packets. To decrease the node power consumption to a minimum, we also implemented a *sleep mode*: set between each packet transmission, which occurs every 5 minutes and during a few seconds. It allows only to keep the bare minimum functionalities needed by the Arduino to be able to reboot the parts required to perform its primary tasks in due time. We have also set up a watchdog timer to reboot the board if the program was to bug and get stuck at one step.

Figure 23: Arduino Uno Rev 3 [8]

4.1.3 MBED NXP LPC1768

The NXP LPC1768 Mbed development board is based on the ARM Cortex-M3 32-bit processor, which runs at 96 MHz and includes 512 KB of flash memory and 64 KB of RAM. The NXP LPC1768 Mbed development board has serial (UART), SPI, I2C and CAN bus interfaces. This board can be programmed with C/C++ with the online software Mbed. [9]

We used this board in order to test power consumptions of different materials (Arduino, sensors, Raspberry Pi...). It is not an integral part of the finished product, but it allowed us to study in details how the devices behave, and so we were able to find solutions and improvements for the project.

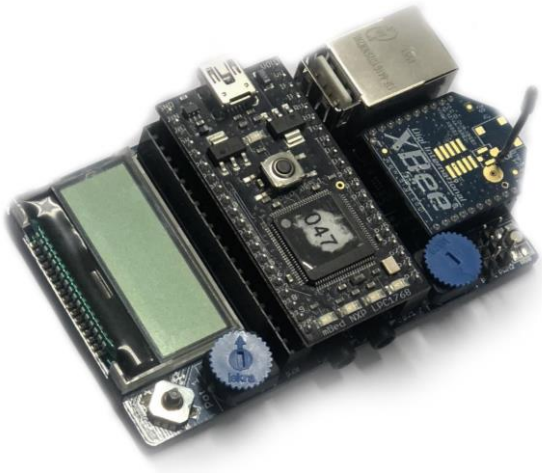


Figure 24: NXP LPC1768 MBED board and shield

4.1.4 IMST IC880A SPI

The concentrator board, assisted by a Raspberry PI 3, serves as our gateway for receiving sensor data from our nodes.

The IC880A SPI is a class A, B and C capable LoRa concentrator board able to receive and to send packets, working in the European ISM 868 MHz band, and featuring an SPI bus. Capable of receiving up to eight different frequencies in parallel, it is supposed to serve as a gateway.

The reason why we chose this board is because it is one of the most commonly used therefore there are many guides, and different libraries that suit well our Raspberry PI 3 already exist.

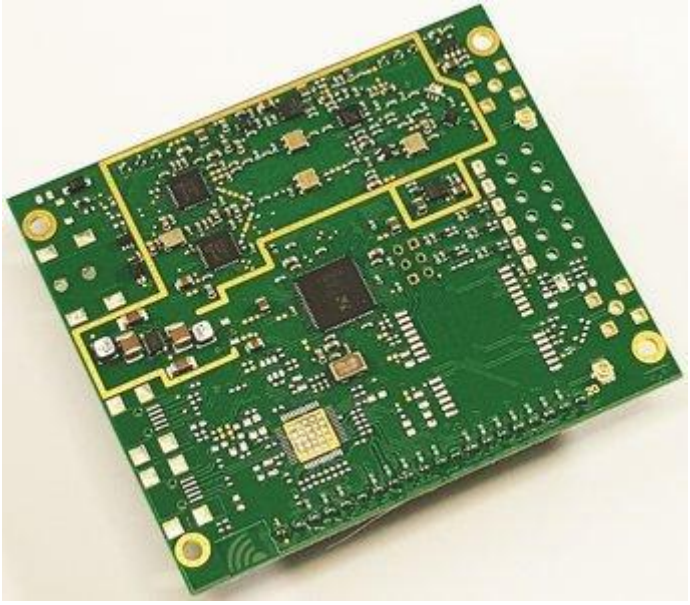


Figure 25: IMST IC880A SPI [10]

4.1.5 SEMTECH SX1272 SHIELD

The Semtech SX1272 LoRa Mbed Enabled shield combines two different parts: the Semtech SX1272 Low Power Long Range Transceiver and the Arm Mbed Internet of Things device platform. The SX1272 transceiver mounted on the shield is LoRa class A and C capable, and provides as well FSK and OOK modulations.

The SX1272 shield is controlled through the SPI bus at a maximum speed of 10 Mbps. The board comes with standard pass-through pins to connect to a variety of boards. Many Arduino ports are still accessible thanks to its grove connectors, so a variety of sensors can be connected on the shield. The SX1272 Shield also includes a detachable antenna with a coaxial plug. [11] It can be coded using various microcontroller development platforms such as Arduino IDE or Nucleo.

When using an Arduino Uno Rev3, a small hardware modification of the shield is required (this modification is mentioned in the tutorial "How to install the Arduino node").

The SX1272 has high efficiency and thus can receive and send data over long distances using very low power. It is allowed a maximum output power of 14 dBm in Europe, but it can be set lower to save energy if the transmission does not require that much. Using ISM bands, It can be set on these central frequencies: 433 MHz and 868 MHz (authorized in the ITU region number one, which includes Europe) and 915 MHz (ITU region 2), with a bandwidth of 125, 250 or 500 kHz (although this last one is not allowed in Europe). It is capable of spreading factors ranging from six to twelve.

To ease the use of the antenna in our box, we have connected an extended antenna wire that will be placed outside the box. To be able to use this antenna, its length has to be half or full wavelength long. This is because we do not know the impedance of transmitter or antenna. We have used this formula for calculating the length of the antenna :

$$\lambda = \frac{V}{f}$$

where:

λ: Length of the wire

V: Velocity of the signal in the wire. The ratio for the RG58 wire is 0.66 (coaxial wire)[12] , which makes the velocity approximately $3 \times 10^8 \times 0.66 = 2 \times 10^8$m/s.

f: Working frequency of the node. Ours is set at 868 MHz.

Therefore:

$$\lambda = \frac{2x10^8}{868x10^8} = 0.23m$$

This means that the cable full wavelength is 0.23 m and its half-length 0.115 m. We are using an RG 058 cable which characteristic impedance is 50 Ω, and we

need to choose either half wavelength or full wavelength. So, we have made a wire 23 cm long because this length suits our project.

Figure 26: Mbed Shield SX1272 and its antenna [13]

4.1.6 ADALM - PLUTO

The ADALM - PLUTO Active Learning Module is a Software-Defined Radio based on an embedded board AD9363 from Analog Devices. This electronic board allows to study radio frequencies and can transmit and receive signals from 325 MHz to 3.8 GHz with up to 20 MHz of instantaneous bandwidth.

Figure 27: ADALM-PLUTO [14]

4.1.7 SENSORS

For measuring the luminosity and temperature values, we used two sensors:

To measure brightness, we used a photoresistor made of expanded polycrystalline semiconductor material. We coupled it with a 10 k**Ω** resistor to limit the current to save energy and protect the device. The working principle of a Light Dependent Resistor is that its resistance varies with light intensity: the resistance decreases when the luminosity increases, therefore the voltage at its terminals decreases, and reciprocally. We measured this voltage with the Arduino Uno, using one of its analog input pins, so we were able to deduce the brightness. Another positive point is that an LDR reacts slow enough so that irrelevant fast light changes will not be recorded (e.g. a bird passing by).

Figure 28: Photoresistor

The photoresistor has been installed on a PCB so that it can be easily integrated in the container we will create later on. Along with the photoresistor, our constraints were also to integrate a 1x3 pin generic connector, a 10 kΩ axial resistor on a 35 mm by 35 mm square surface, in which four 3.5 mm diameter holes will be drilled to receive screws. A board has therefore been drawn using KiCad, a free Electronic Design Automation software geared toward creating and designing printed circuits:

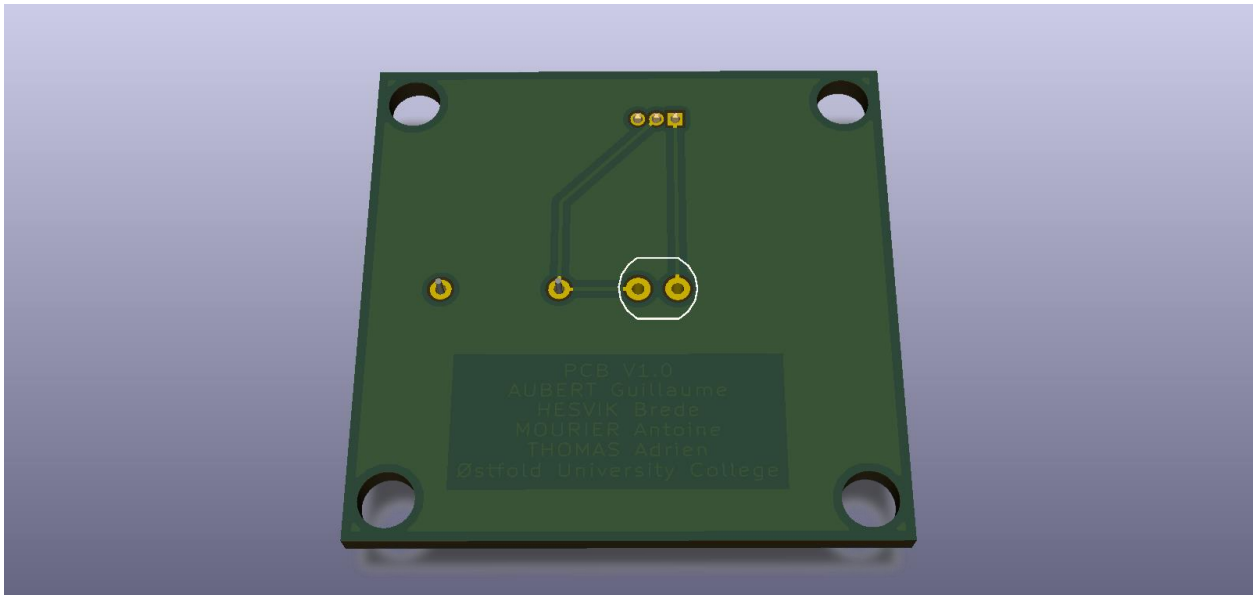


Figure 29: Front of the PCB

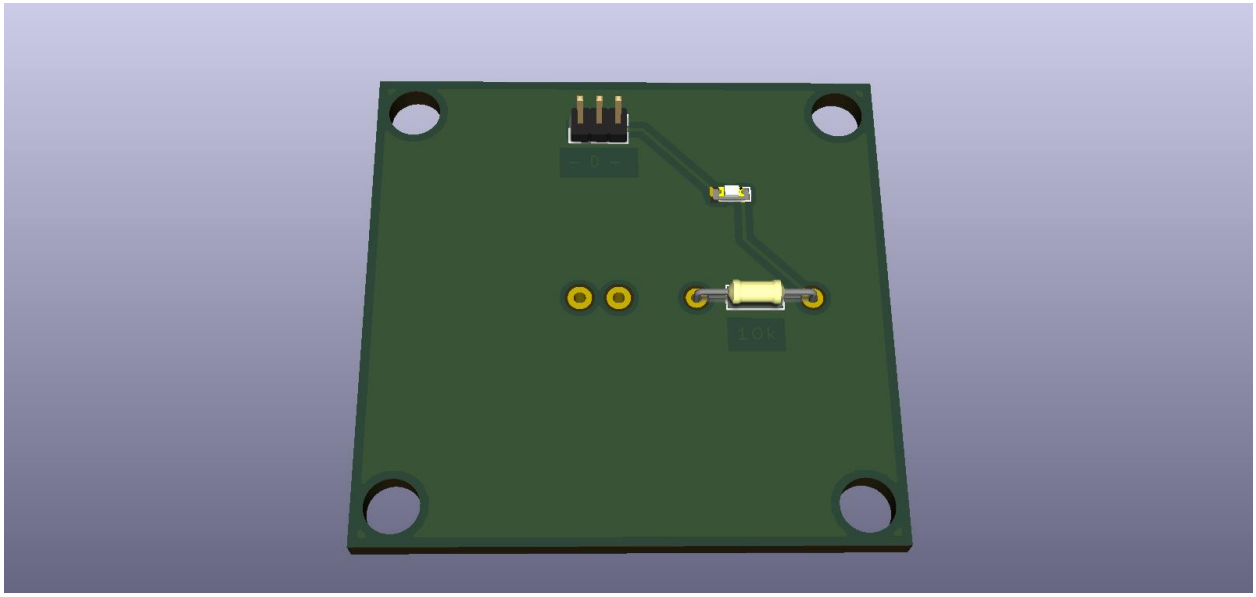


Figure 30: Back of the PCB

We ended not used it because it takes too long to command a PCB, and it was easier and faster to use a generic board than printing one ourselves.

To monitor the outside temperature, we used the TC74A0: it is a small digital sensor from Microchip, in which the thermal sensor element and output convert temperature as an 8-bit digital word. Communication with the thermistor is achieved via a simple two-wire I2C serial synchronous port.

This sensor was installed in a probe, remote from the main box and connected by a wire. We worried if the added resistance induced by the length of the wire (50 cm) could have a detrimental influence on the data transmission and thus the normal operation of the sensor, so we calculated the wire's resistance and made the following table using this formula:

$$R_{wire} = \rho \frac{l}{S}$$

where

R_{wire}: Resistance of the wire (Ω)

ρ: Resistivity of the material ($\Omega.m$)

l: Length of the wire (m)

S: Diameter of the wire (m²)

$$R_{Cu} = 1{,}7.10^{-8} \times \frac{0.5}{0{,}75.10^{-6}} = 0{,}0113\ \Omega$$

$$R_{Al} = 2{,}7.10^{-8} \times \frac{0.5}{0{,}75.10^{-6}} = 0{,}018\ \Omega$$

Copper wire (0.75 mm²) for 50 cm	Aluminium wire (0.75 mm²) for 50 cm
$0.0113\ \Omega$	$0.018\ \Omega$

So for a 50 cm long wire with a 0.75 mm² section, and regardless if it is copper or aluminium, the resistance is negligible: it will have no significant impact on the data values.

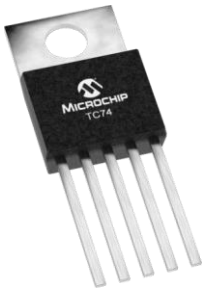


Figure 31: TC74A0 Thermal sensor [15]

4.1.8 SERVER

We installed Ubuntu on a desktop as the operating system for the server.

We chose Ubuntu because it is a fully configurable free open source Linux distribution that runs well on low-end PC.

We then installed XAMPP, a package comprised of an Apache2 Server, different kinds of SQL databases and some web languages (PHP, CSS, HTML, and Javascript). This allowed us to build a server, to receive data, store it in a database, and host a website to display dynamic data in real time.

The website has been written with different programming languages: PHP for the internal parts (to create a link with the database, to create animations, to use mathematical formulas, etc), HTML to create texts, CSS to design the website, SQL to code the database, and Javascript in order to use the AJAX method to refresh the website pages automatically each minute (to update sensors values). To secure the website we used a login system on the main page, so it is impossible to access the other pages without a valid identification. A download button has been added to the website to get all system programs, tutorials, and datasheets

in passworded zip form. We also implemented a language selector on the website, allowing to switch between Norwegian and English.

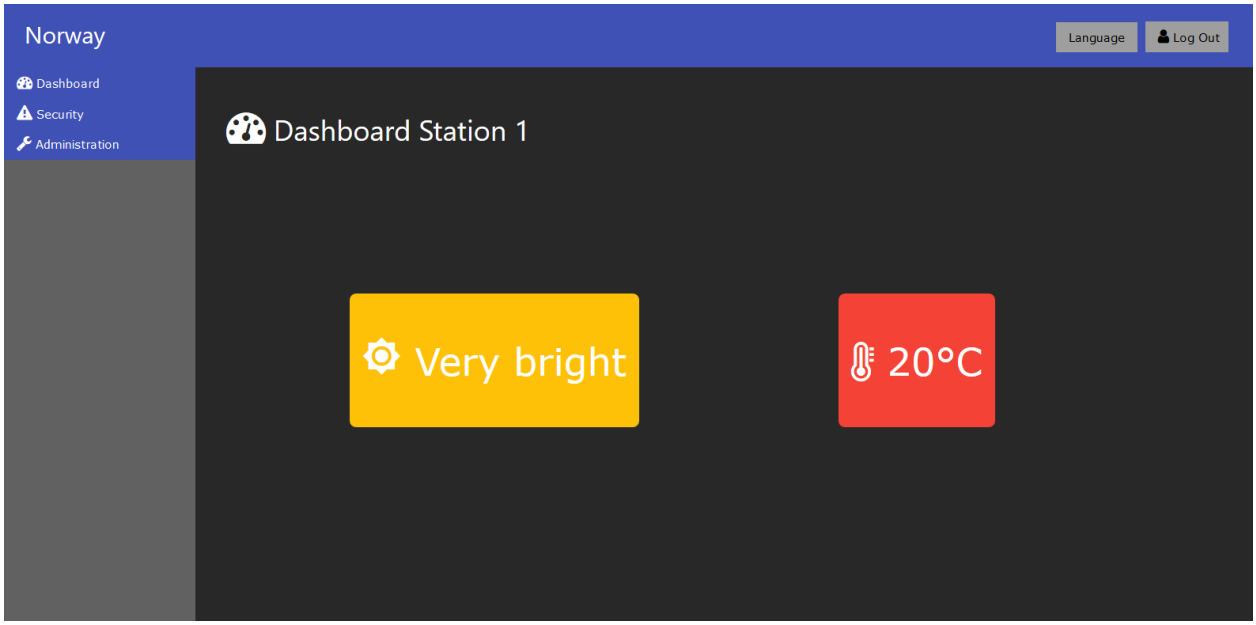


Figure 32: Website dashboard

4.1.9 MEASURING TOOL

We have made a tool that can measure the current going into our MCU. It can be a very beneficial to determine the power output of an MCU and then find out how long it can last on batteries. It can also be used to find the lowest amount of current consumption by experimenting with different situations like including a sleep code or turning off various power draining devices on the MCU. This measuring tool is therefore particularly helpful for IoT devices implementation. This can be done with any MCU, and with some small modifications it can be applied to anything that consumes electricity. How it works is described in appendix D: *measuring current on MCU*.

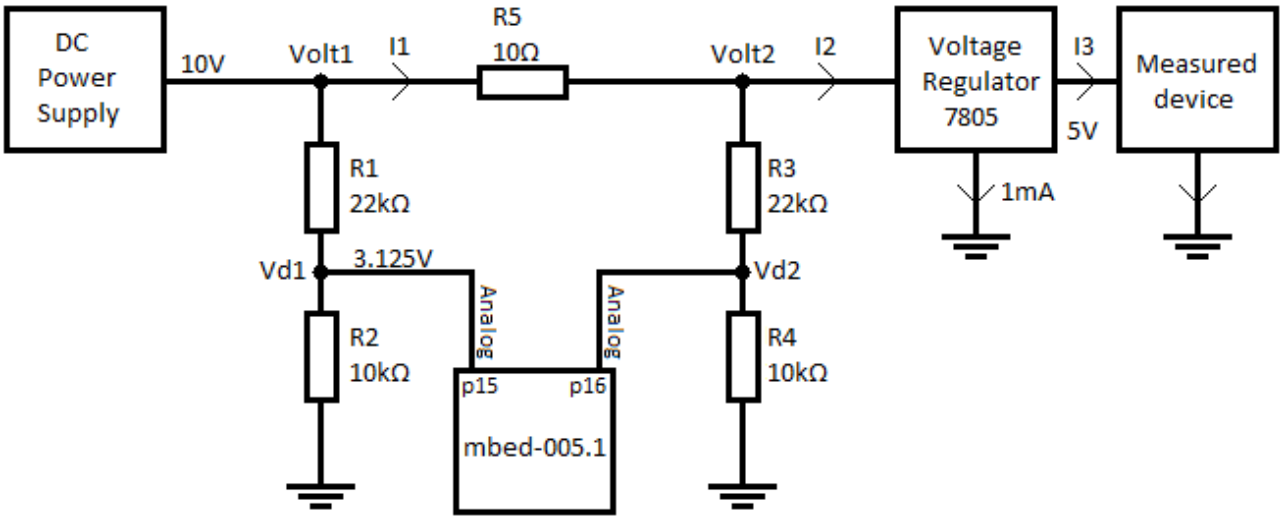


Figure 33: Schematic of the circuit used to measure current

4.1.10 Design of equipment

For our node, we have designed a waterproof container that should be able to last years. We 3D printed the parts, and put a plexiglass window for the luminosity sensor. Some problems need to be addressed such as condensation, water leaks, the porosity of the PLA plastic, and the inside temperature:

Condensation is a worrisome problem: when condensed, water is collecting on the internal surface of the container, and the risk of malfunction is increased. This may result in rusting and corrosion, short-circuiting and electrical component breakdowns.

To protect against this phenomenon, we built the box big enough so that it could also contain a bag of silica gel. We could also have used a small hygrometry sensor such as the DHT22/AM2302, to monitor the humidity level inside the box and send an alert if it goes above a certain threshold.

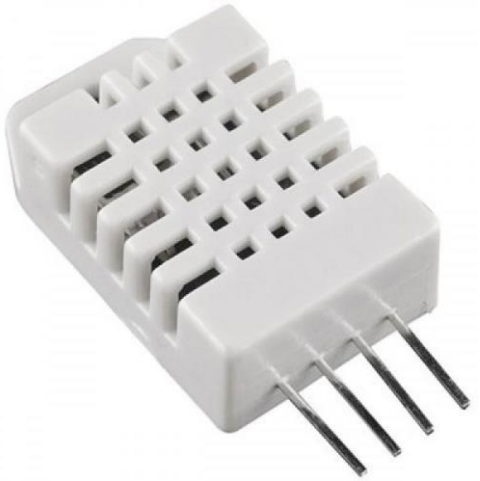


Figure 34: DHT22 humidity sensor [16]

Another major issue is water leaks, which causes the same type of degradations as condensation. To prevent it, the container should be sealed with acrylic gel.

Also, PLA plastic is porous enough to let water pass through it. To overcome this problem, we plan to immerse the box's pieces in an acetone bath for a short time to bind layers of plastics, thus making it waterproof.

About the temperature: its inertia inside a black plastic box is such that we would get erroneous measures if we had kept the temperature sensor inside it. That is why the sensor has been installed into a small waterproof 3D printed probe, which is wired to the box. With a 50 cm long cable, the technician installing the node will be able to easily find a location for the probe unexposed to the sun.

During the first stages of design, the container was looking like this:

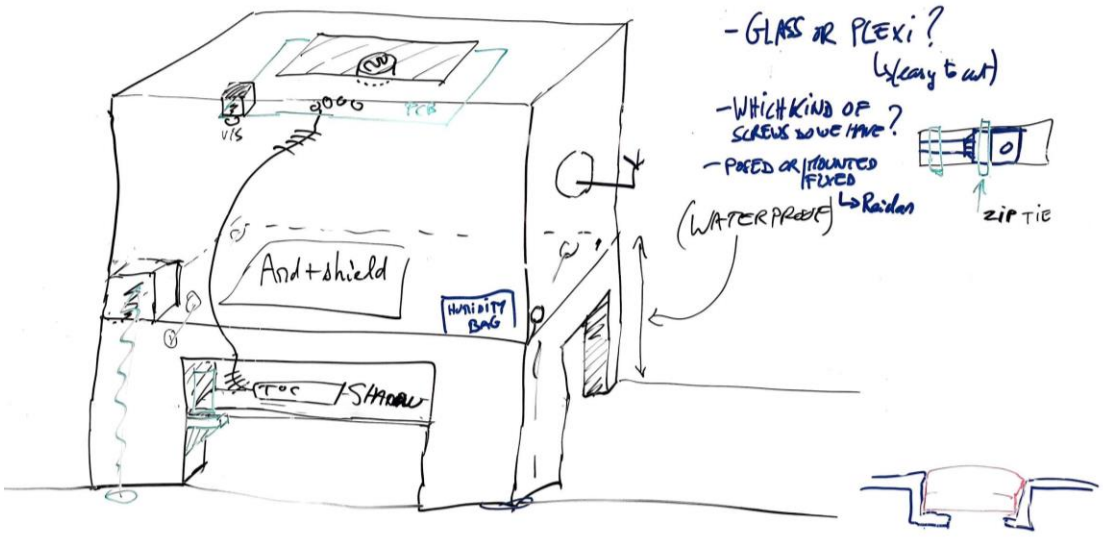


Figure 35: First draft of the node box

Throughout researches it evolved, and finally these are the components that have been printed:

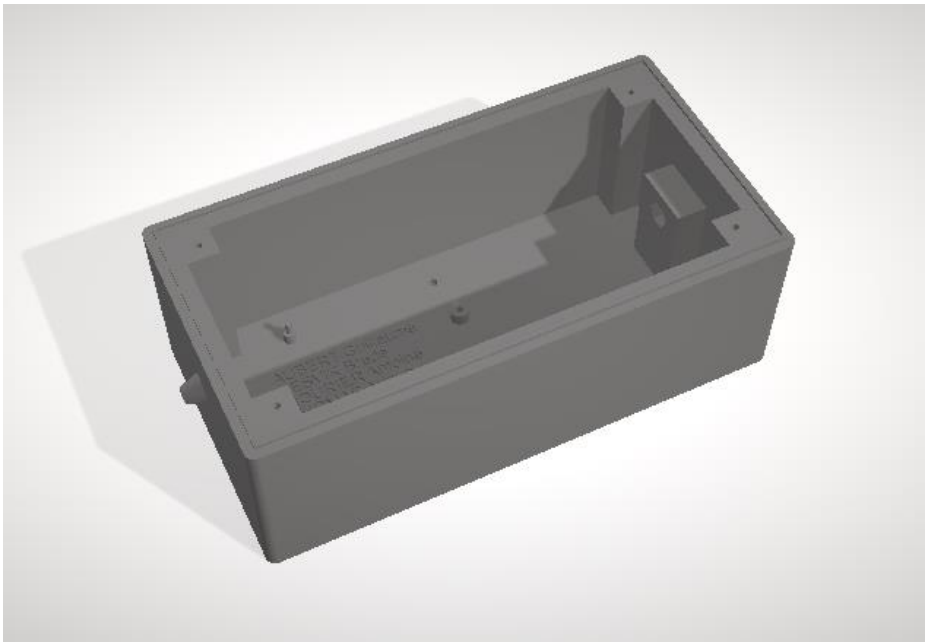


Figure 36: The core of the box

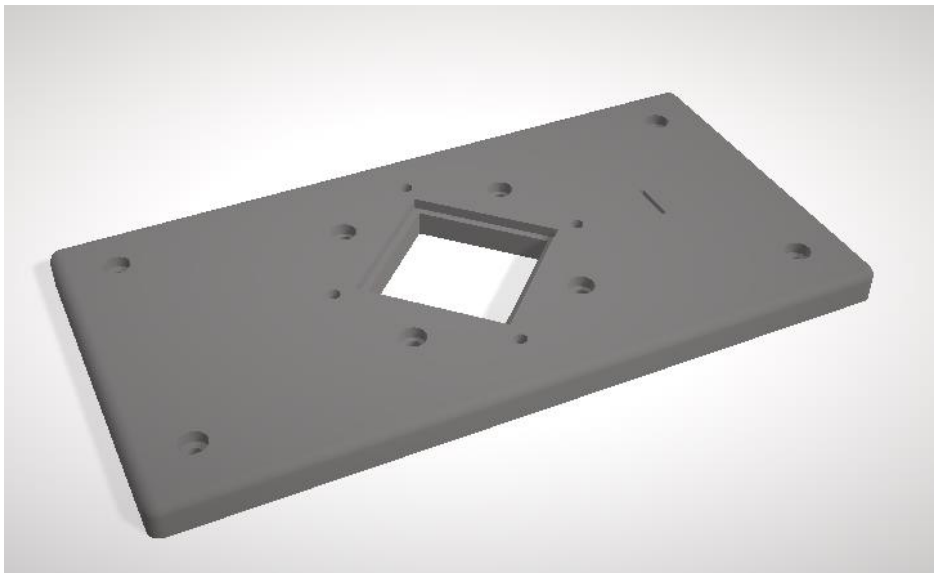


Figure 37: The lid

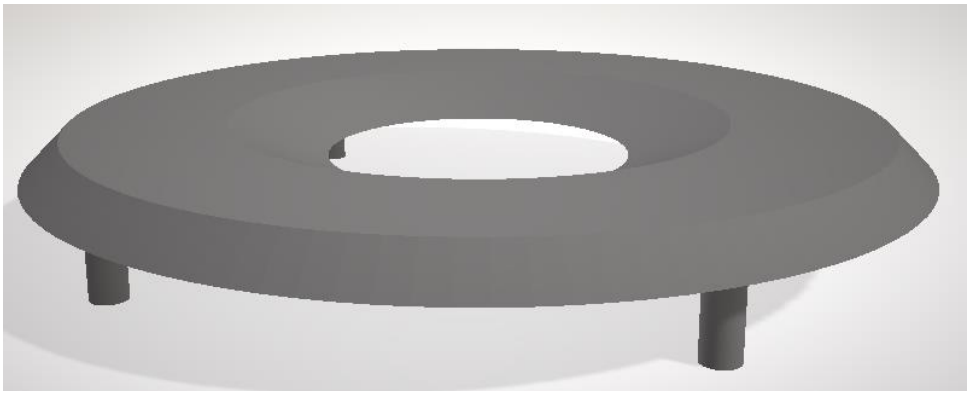


Figure 38: The cover

Figure 39: Probe´s upper part

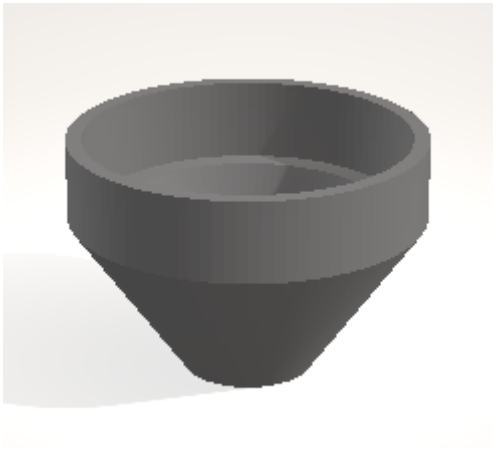


Figure 40: Probe´s bottom part

The pieces have been modeled with OpenSCAD, a free parametric open-source software in which solid 3D CAD objects can be designed using lines of code. The created templates can then be exported as an STL file and used by a 3D printing machine.

Figure 41: The resulting printed pieces

4.2 DEVICES SETUP

4.2.1 NODE SETUP

See the in the appendix C the tutorial named "*How to install the Arduino node*".

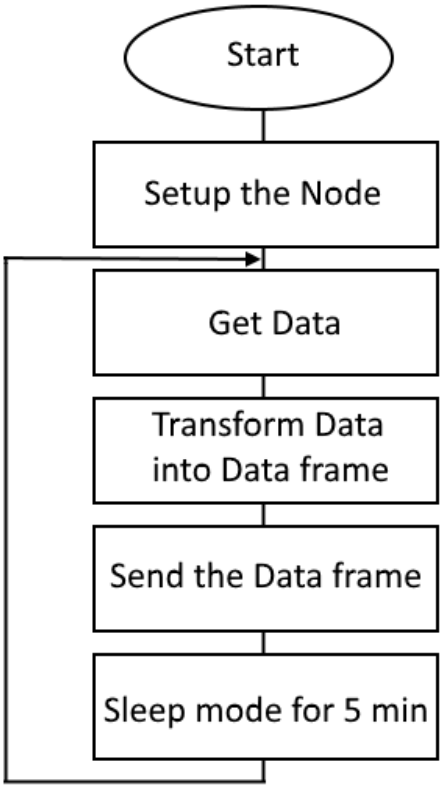


Figure 42: How the node function

The Node code is based on this algorithm.

4.2.2 GATEWAY SETUP

See the in the appendix C the tutorial named "*How to install the Raspberry Pi Gateway*".

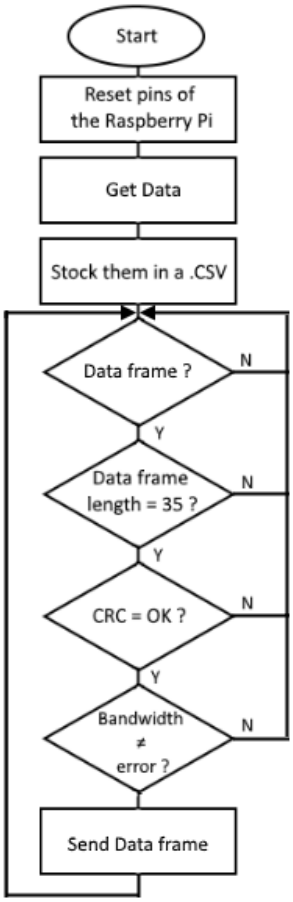


Figure 43: How the gateway function

The Gateway code is based on this algorithm.

4.2.3 SERVER SETUP

See the in the appendix C the tutorial named "*How to install the Apache2 Server and the Website*".

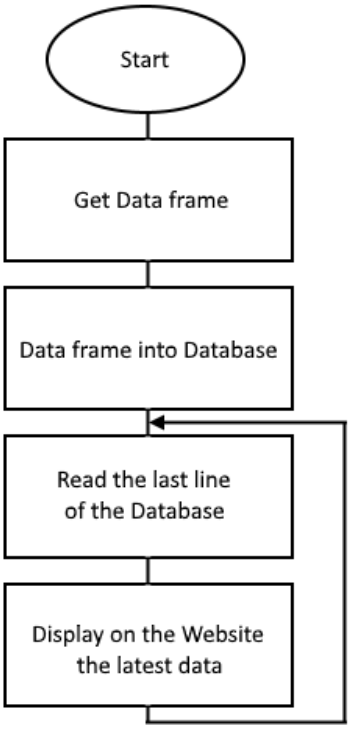


Figure 44: How the server function

The Server code is based on this algorithm.

4.3 ECONOMY

From an economic perspective, the system is not particularly expensive. Below is a budget displaying the different figures for each component:

Table 2: Cost for a node

Name	Price	Number	From
SX1272 Mbed Shield	20.29 €	1	Digi-key.com
Arduino Uno Rev 3	24.00 €	1	kjell.com
Various other costs (screws, wires, etc)	around 5.00 €		
Total cost	49.29 €		

Table 3: Cost for a gateway

Name	Price	Number	From
IMST IC880A SPI	119.00 €	1	shop.imst.de
Pigtail wire for IMST 880A SPI	6.50 €	1	shop.imst.de
SMA antenna for IMST 880A SPI	6.50 €	1	shop.imst.de
Raspberry Pi 3 Model B	39.00 €	1	komplet.no
SD Card	9.26 €	1	biltema.dk
Other costs	around 5 €		
Total cost	185.26 €		

5. TESTING FOR END REPORT

5.1 SOFTWARE TESTING

Arduino IDE serial monitor allowed us to check the different steps in our code involved in sending data to the gateway:

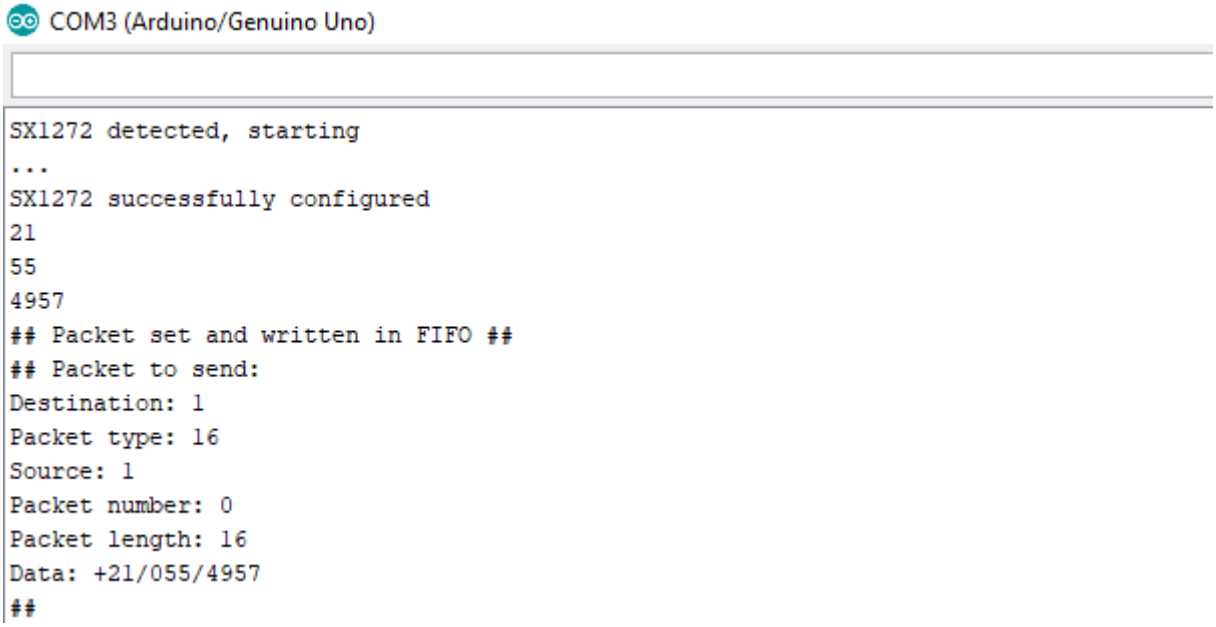


Figure 45: Serial monitor on Arduino IDE showing sending operations results

In the above screenshot, the line *data: +21/055/4957* means that the sensors read 21°C in temperature and 55 in luminosity, and that the Arduino 5V line is currently at 4.957V.

In the same fashion, Raspbian's terminal window on the Raspberry Pi permitted us to verify the gateway setup and its proper operation:

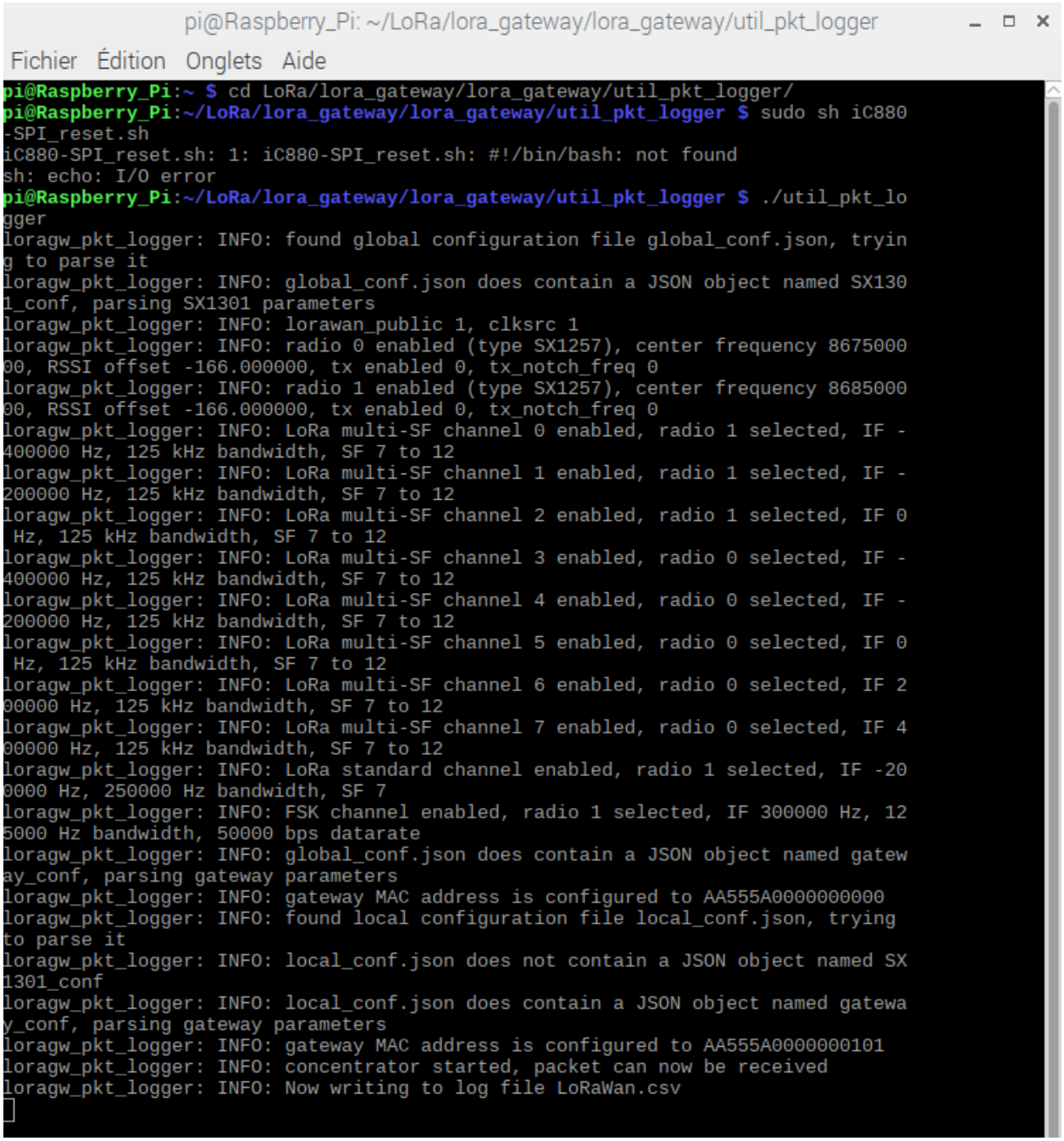


Figure 46: Raspbian terminal window showing the gateway booting up and its configuration

This next picture shows the spreadsheet file we used to store the data on the Raspberry Pi, that we used to be assured that the reception was working as intended before going further:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	gateway ID	node MAC	UTC timestamp	us count	frequency	RF chain	RX chain	status	size	modulation	bandwidth	datarate	coderate	RSSI	SNR	payload
2	AA555A0000000101		2018-05-23 16:52:00.272Z	23718603	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-35	+8.8	011001B7-2B32312F-3035362F-34393739
3	AA555A0000000101		2018-05-23 16:52:24.114Z	47561243	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-34	+8.8	011001BE-2B32312F-3035322F-34393537
4	AA555A0000000101		2018-05-23 16:52:27.520Z	50967339	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-34	+8.0	011001BF-2B32312F-3035352F-34393739
5	AA555A0000000101		2018-05-23 16:52:37.741Z	61185627	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-35	+9.0	011001C2-2B32312F-3035352F-34393537
6	AA555A0000000101		2018-05-23 16:52:54.771Z	78216123	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-36	+9.5	011001C7-2B32312F-3036302F-34393739
7	AA555A0000000101		2018-05-23 16:52:55.078Z	78521796	867900000	0	7	CRC_BAD	138	LORA	125000	SF7	1/2	-85	-11.5	80282FA4-09F44168-9312B1A7-BF758087-17
8	AA555A0000000101		2018-05-23 16:52:58.178Z	81622235	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-34	+9.8	011001C8-2B32312F-3035392F-34393537
9	AA555A0000000101		2018-05-23 16:53:08.394Z	91840531	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-33	+9.5	011001CB-2B32312F-3035382F-34393739
10	AA555A0000000101		2018-05-23 16:53:28.832Z	112277123	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-35	+9.5	011001D1-2B32312F-3035352F-34393537
11	AA555A0000000101		2018-05-23 16:53:39.667Z	123112652	867500000	0	5	CRC_BAD	10	LORA	125000	SF7	2/3	-97	-11.0	529FB5BF-2268666D-1CB6
12	AA555A0000000101		2018-05-23 16:53:42.455Z	125901515	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-36	+8.8	011001D5-2B32312F-3035342F-34393537
13	AA555A0000000101		2018-05-23 16:53:59.487Z	142932027	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-31	+9.0	011001DA-2B32312F-3035352F-34393537
14	AA555A0000000101		2018-05-23 16:54:04.112Z	147556644	867700000	0	6	CRC_BAD	182	LORA	125000	SF7	4/7	-99	-11.5	7C4F4E00-E1FAE05E-DE161539-B68E0A7D
15	AA555A0000000101		2018-05-23 16:54:30.142Z	173586995	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-36	+10.0	011001E3-2B32312F-3035352F-34393537
16	AA555A0000000101		2018-05-23 16:54:40.361Z	183805227	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-34	+8.8	011001E6-2B32312F-3035342F-34393537
17	AA555A0000000101		2018-05-23 16:54:43.768Z	187211323	868100000	1	0	CRC_OK	16	LORA	125000	SF7		-33	+9.2	011001E7-2B32312F-3035372F-34393537

Figure 47: Stored information on the Raspberry Pi SD card´s csv file

This Raspbian terminal window was used for debugging, showing which data were sent to the server:

Figure 48: Sending data from the Raspberry to the server

In the same way as the previous picture, Ubuntu's terminal window was used to monitor sending data to the SQLite 3 database:

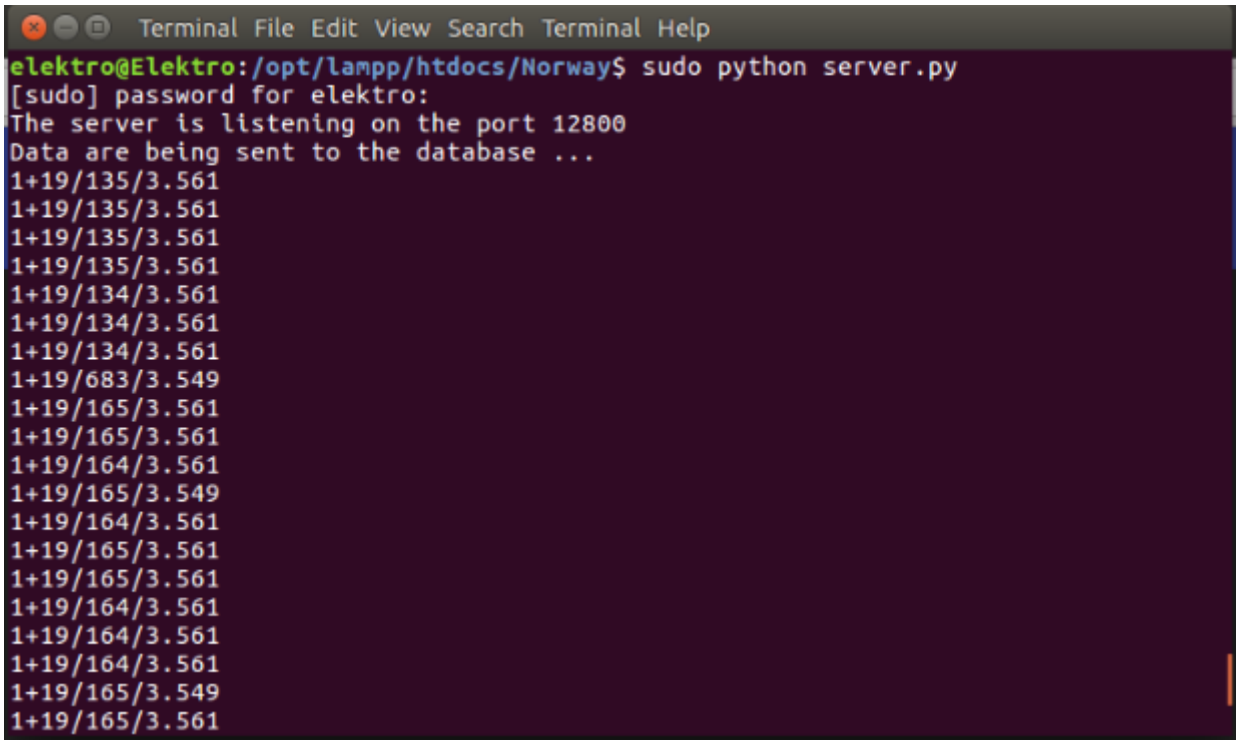


Figure 49: Data reception on the server

This last picture show our website, where we could monitor that the sensors data were being displayed, validating the whole communication chain:

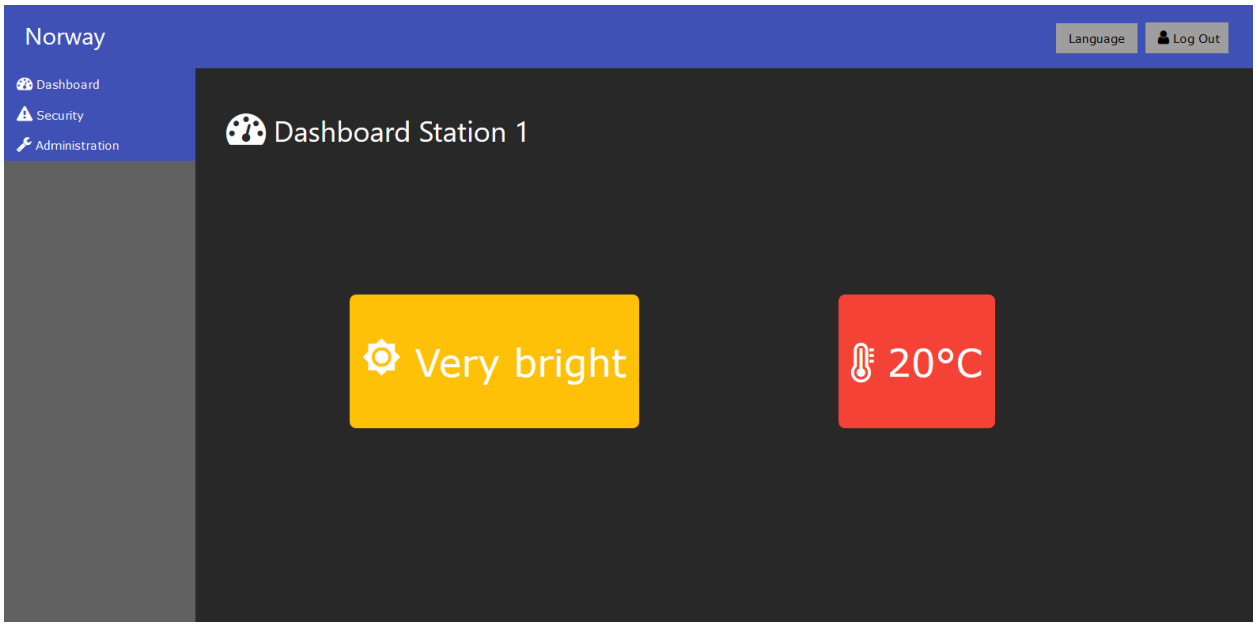


Figure 50: Data read and displaed by the website

5.2 LoRa parameters

To establish a proper communication link between a node and the gateway, a few parameters had to be set up: the central frequency of the communication channel, the CSS bandwidth, the spreading factor, the transmission power, the coding rate, the choice of using a CRC algorithm, the choice of using a header, and the address of the node.

First and foremost, the mode of transmission had to be set up to *LoRa* (CSS), because the SX1272 also provides FSK and OOK.

The central frequency has been set to the 868 MHz band, which is part of the license free bands in Europe, because it is the only one that can be received by the concentrator of our gateway.

Among the two spread spectrum bandwidths allowed in europe, we chose to use 125 kHz instead of 250 kHz to save energy, since the packets sent by the node are quite small and thus don't require more.

To determine the best Spreading factor and the best Emission power, we realized a test in real conditions as you can see on the picture below.

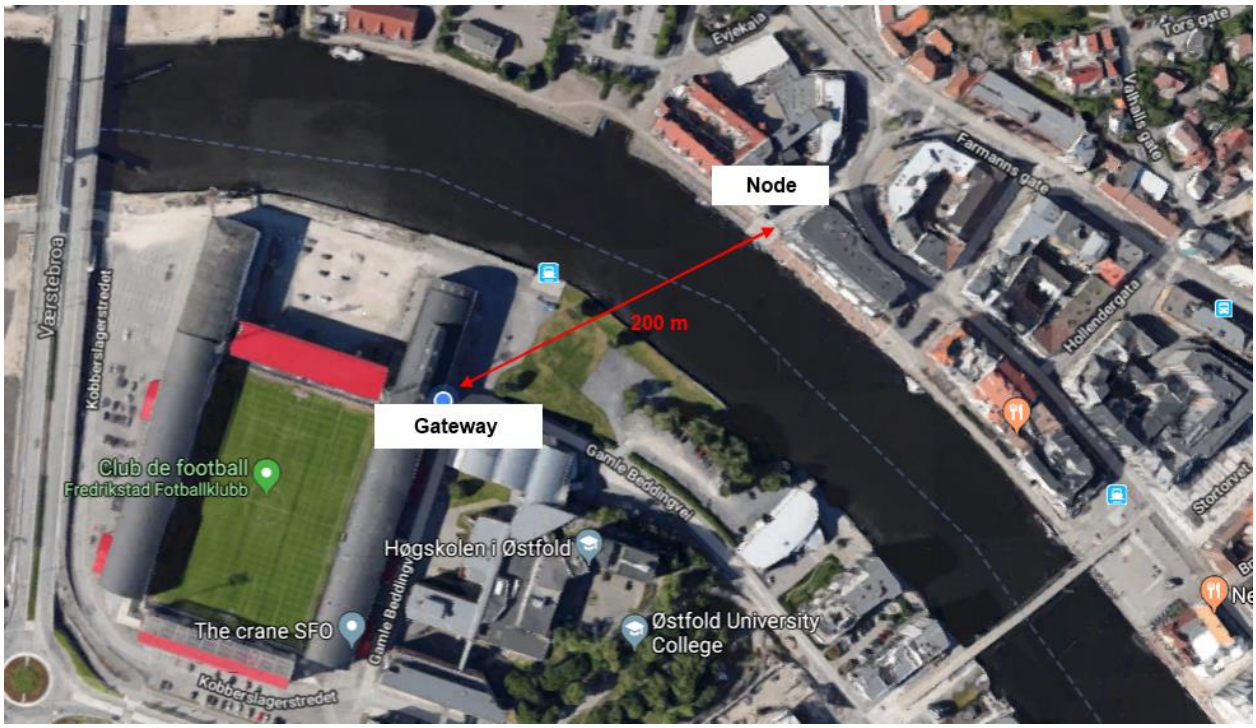


Figure 51: Overview of testing area

This test has permitted us to create this table :

Table 4: LoRa parameters

Power	Spreading factor	Data
Max	SF7	Received
Max	SF8	Received
Max	SF9	Received
Max	SF10	Received
Max	SF11	Received
Max	SF12	Not received

High	SF7	Received
High	SF8	Received
Low	SF7	Not received
Low	SF8	Received
Low	SF10	Received
Low	SF12	Not received

Low: 2 dBm, High: 6 dBm, Max: 14 dBm

We were able to deduce that the low power mode is not suitable for long-range external use because we received data correctly only a few times: it is not reliable enough. We had to put aside SF12 too because data takes too much time to arrive, even at maximum power. Finally, the best compromise between bit rate, low power consumption and signal reliability for transmitting at a distance of 200m, is to use SF7 with *High* power mode.

During most of the indoor tests we realized, at a distance of 1m, we used SF7 and *High* power mode too. It was the fastest and the most reliable, and we plan to use the same settings during the expo.

7. LoRa Reversed

Reversing LoRa can be a tedious task which can be quite time consuming.

We are using an ADALM-PLUTO coupled with MATLAB (a multipurpose mathematical and programming software) through its block diagram environment Simulink. Different blocks implemented in the software were used to filter out the noise and obtain the LoRaWAN signal.

A module is required to use ADALM-PLUTO in MATLAB. It can be found in the add-on store.

In Simulink we created the following spectral analysis program :

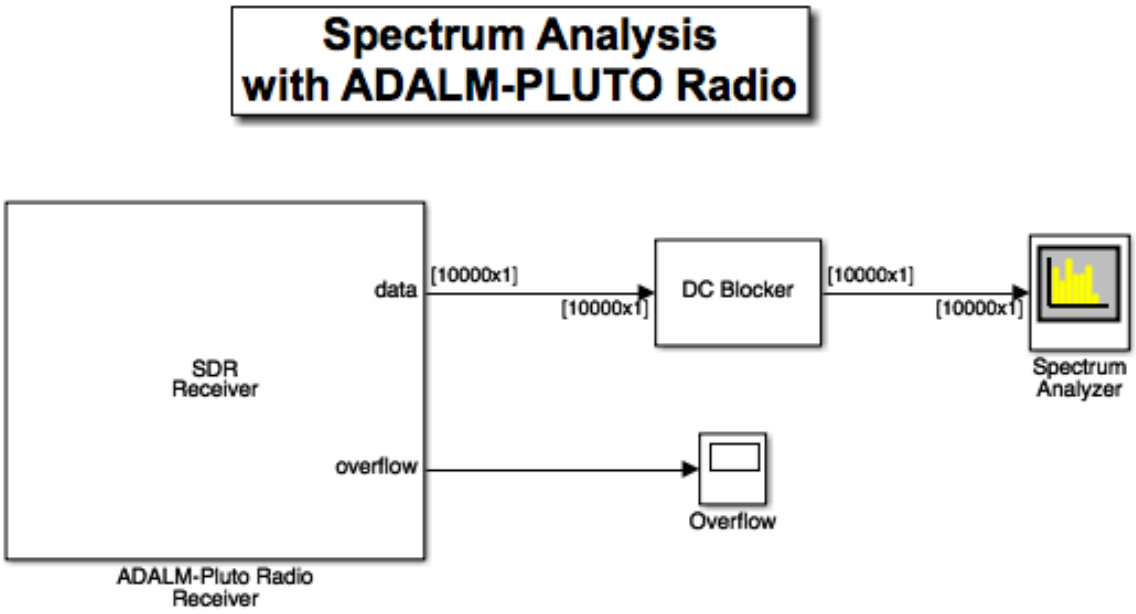


Figure 52: From spread spectrum analysis Simulink

N.B. It is possible to change the frequency the device is listening to by double clicking on SDR-Receiver to enter its properties.

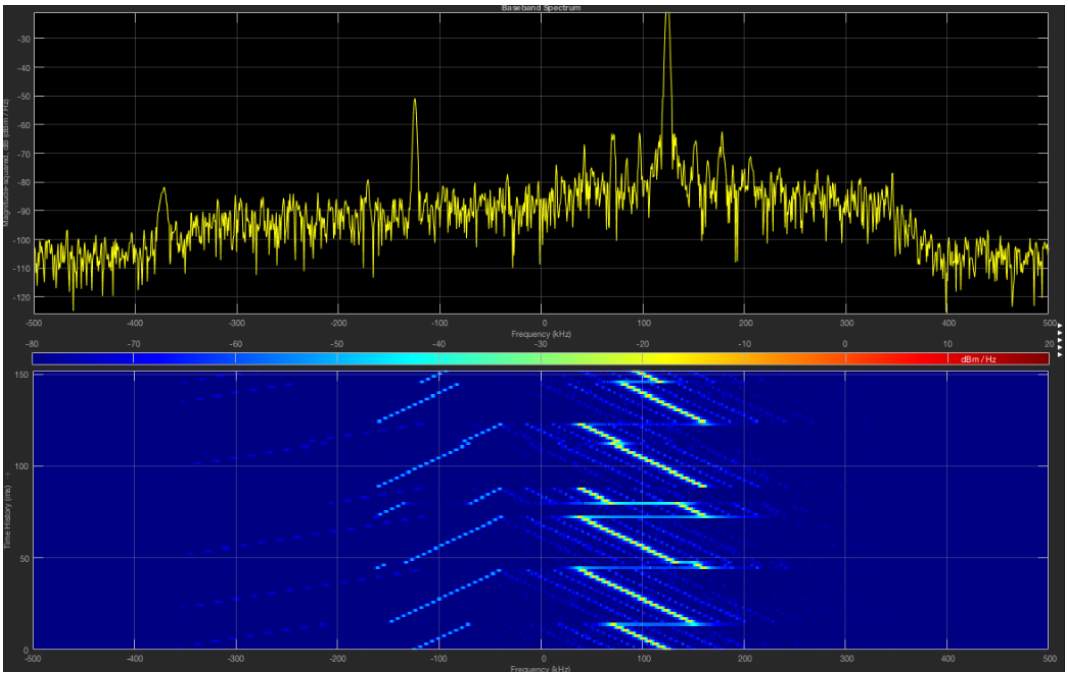


Figure 53: LoRa signal spectrum analyzed using ADALM-PLUTO through MATLAB

This picture from MATLAB spectrum analyzer shows the chirp modulated signal transmitted by our node. You can see on the spectrogram the up-chirps over time in a *waterfall* representation (horizontal axis: frequency, vertical axis: time).

8. Discussion

The Internet of Things is where objects composed of different types of electronics, sensors, and software are linked by a network. New and improved equipments are continuously being added to the pool of existing ones, becoming cheaper and smaller in the process. By developing automation at a rapid pace, it builds a smarter world where new possibilities arise day by day. The purpose of IoT is to make everyday life more comfortable for both businesses and individuals.

In this project, a group of students has gotten a task to set up a system able to record sensors information and send it over long distances to present it on an

interactive website. It has not been an easy road, and it has required a lot of time and effort.

8.1 CHALLENGES

We had some problems underway that were challenging to resolve:

- We struggled with the Ubuntu firewall which was prohibiting incoming connections. We first solved this by simply disabling the Ubuntu firewall altogether, so we could hastily continue to work on our main tasks. Later on, we found how to allow which IPs could go through which ports of the firewall.
- Another networking problem was connecting the Raspberry Pi 3 to the internet. We fixed it by setting a static IP.
- It has been challenging to find a lot of information regarding LoRaWAN. So far Semtech has been very shyly giving out information, compared to their huge ambitions for LoRaWAN. It is still rather poorly documented, and what was available was not easy to understand.
- We had to 3D print certain pieces of the node container several times to find the right settings for the printer and get what we desired.
- We had some mind-bending issues with the LoRa shield mounted on top of the Arduino, which were actually hardware compatibilities problems. We would have liked to discover this early in order to possibly switch to a better suited hardware.
- Another challenge has been to resolve conflicts between a program that could read the Raspberry Pi's csv file at the same time another one wrote in it, both written in Python. The fact that Python doesn't have a proper compiler and thus bugs have to be fixed one by one in their order of appearance, added to the complexity of the problem.

- There was also a conflict between the code measuring the Arduino 5V line voltage, and the code reading its CPU temperature. We lacked time to resolve it.

8.2 ECONOMY SAVINGS

One of the key difference between a LoRa system and e.g. LTE, beside the increased range between devices, is that we do not need to pay anything for the data bandwidth. There are also places which simply do not have LTE coverage yet.

operating costs are :

a Raspberry Pi 3 consumes 1.4 W [19]

To find out what the Raspberry PI 3 power consumption costs every year, we convert power into power per year, and then multiply it with the power cost in Norway (price taken from www.norgesenergi.no):

$$1.4W * 24\,hours * 365\,days = 12.265\,KWH\ every\ year$$

$$12.265 * 44.19\,NOK\ ØRE = 5.50\,NOK\ every\ year$$

Node average power consumption measured:

$$92\,mW * 24\,hours * 365\,days = 805.92\,Wh$$

$$805.92 * 44.19\,NOK\ ØRE = 0.3561\,NOK\ every\ year$$

you also have to pay 27 kroner every month for the monthly electric subscription, so its annual cost is:

$$27 * 12 = 324\,NOK$$

The total annual electric cost is:

$$Total = 324 + 0.3561 + 5.5 = 329.8\,NOK/year$$

The formula for electricity cost is:

*total electricity expense = (324 + 5.5 + 0.36x) * years in service*

where x equals number of nodes.

Note that supposing that you already pay for a monthly electric subscription, then the operating cost of a gateway and a few nodes is quite negligible.

8.3. FURTHER WORK

- We would like to program the Nucleo L74RG instead of the Arduino Uno, to get sensors information and send them with the LoRa shield. This is because the Nucleo uses a fraction of the power consumption that Arduino use and because it should not have the compatibility issues we encountered between the Arduino and the LoRa shield.
  According to the consumption calculator on CubeMX (a development interface for STMicroelectronics board), the Nucleo use at most 60 microamperes with the setting we would need.

  Nucleo uses 60 mikro Amper. So $\frac{3000\ mAh}{60\ \mu A}$ = 50 000 hours = 5.7 years

  of battery life expectancy, if we use one able to debit 3000 mAh.

- For further work we would have liked to expand the application possibilities beyond temperature and light sensors to improve the weather forecast, and for example use humidity or wind sensors.
- We would also have liked to reverse engineer the LoRa devices ourselves to learn more about it.
- To increase the professionalism of our system, futures tasks may be further to optimize all the programs used and do an in-depth debugging, to anticipate all possible problems and errors that may arise. We could also replace the photoresistor circuit board by the better one we realized with KiCad, and print an improved node container with upgraded size,

waterproofness, robustness, etc. Some other future tasks could be to host the website on the internet, resolve some minor graphics issues we are having with it at the moment with some internet browsers, integrate some data graphics, add a page where some potential actuators could be controlled, and enhance its overall design.

9. CONCLUSION

Along the course of this project, we researched the specific technology that is LoRaWAN and inserted it in a realistic application. We struggled from time to time on some diverse points, but the challenge gave us opportunities to learn a lot in different fields (theoretical, practical, organisation, communication, etc), including ourselves.

LoRa is a developing technology battling for monopoly in the emerging IoT field. While quite not mature yet, we demonstrated it is promising enough to potentially become a new standard for LPWAN applications: the goal from chapter 1.3 is reached.

10. REFERENCES

[1] Matt Knight, Reversing LoRa. [Online]. Available:

https://static1.squarespace.com/static/54cecce7e4b054df1848b5f9/t/57489e6e07eaa0105215dc6c/1464376943218/Reversing-Lora-Knight.pdf

[2] Romain Cambier, LoRaWAN and The Things Network. (11.2016) [Online]. Available:

http://public.jeudisdulibre.be/conferences/2016-11/JDL-20161117-LoRaWan-RCambier.pdf

[3]Romain Cambier, LoRaWAN and The Things Network. (11.2016) [Online]. Available:

http://public.jeudisdulibre.be/conferences/2016-11/JDL-20161117-LoRaWan-RCambier.pdf

[4]Wikipedia. Link Budget [Online]. Available:

https://en.wikipedia.org/wiki/Link_budget

[5] Wikipedia. Star Network [Online]. Available:

https://commons.wikimedia.org/wiki/File:Active_ring_topology.png

[6] The Things Network. Overview [Online]. Available:

https://www.thethingsnetwork.org/wiki/LoRaWAN/Overview

[7]Gear Best. Raspberry PI 3B [Online]. Available:

https://www.gearbest.com/raspberry-pi/pp_354347.html

[8] Richelt. Arduino UNO [Online]. Available:

https://www.reichelt.com/de/en/Single-Board-Microcontroller/ARDUINO-UNO-DIP/3/index.html?ACTION=3&GROUPID=8243&ARTICLE=154902

[9] MBED. LPC1768 [Online]. Available:

https://os.mbed.com/platforms/mbed-LPC1768/

[10]IMST. IC880A Concentrator board [Online]. Available:

https://shop.imst.de/wireless-modules/lora-products/8/ic880a-spi-lorawan-concentrator-868-mhz

[11] Mouser. Semtech SX1272 MBED shield [Online]. Available:

https://no.mouser.com/new/semtech/semtech-sx1272-mbed-shield/

[12] Belden. RG-58 coaxial cable techdata [Online]. Available:

https://catalog.belden.com/techdata/EN/9203_techdata.pdf?ip=false

[13] Digikey. SX1272[Online]. Available:

https://www.digikey.com/product-detail/en/semtech-corporation/SX1272MB2DAS/SX1272MB2DAS-ND/6099191

[14]Analog.ADALM-PLUTO[Online]. Available:

http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html

[15] Microship. TC74 termistor[Online]. Available:

https://www.microchip.com/wwwproducts/en/TC74

[16]Robotshop. DHT22 humidity sensor[Online]. Available:

https://www.robotshop.com/en/humidity-temperature-sensor-dht22.html

[17] Ali Benfattoum. De la technologie LoRa au rèsau (21.08.2016)[Online]. Available:

https://www.frugalprototype.com/technologie-lora-reseau-lorawan/

[18] Sghoslya. All about LoRa and LoRaWAN[Online]. Available:

http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html

[19] Pidramble. Power consumption[Online]. Available:

https://www.pidramble.com/wiki/benchmarks/power-consumption

[20] Analog. ADALM-PLUTO[Online]. Available:

http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html#eb-overview

[21] Stephen Hengstler, Chirp Spread Spectrum. Köln, Deutschland: Lambert Academic Publishing, 2009.

[22] Richard Wenner. LoRa Field Strength measurement. (21.11.2017) [Online]. Available:

https://www.youtube.com/watch?v=WizST6gighs

[23] Thomas Telkamp. *LoRa crash course. (02.11.2016) [Online]. Available:*

https://www.thethingsnetwork.org/forum/t/lora-crash-course-by-thomas-telkamp/3981

[24] Fosdem. SDR LoRa aes. (03.02.2018)[Online]. Available :

https://fosdem.org/2018/schedule/event/sdr_lora_aes/

Learning material not referred:

Wikipedia. Software-defined radio. Available:

https://en.wikipedia.org/wiki/Software-defined_radio

Paul Bjørn Andersen. LDR-mostand. (2009) [Online]. Available:

https://snl.no/LDR-motstand

Lovdata, 2012. Forskrift om generelle tillatelser til bruk av frekvenser (fribruksforskriften). Available:

https://lovdata.no/dokument/SF/forskrift/2012-01-19-77/KAPITTEL_13#KAPITTEL_13

Farnell. Thermal sensor TC74A0. [Online], Available:

http://uk.farnell.com/microchip/tc74a0-5-0vat/thermal-sensor-digital-to-220/dp/1332317

Belden Inc. 9203 techdata. (2018) [Online]. Available:

https://catalog.belden.com/techdata/EN/9203_techdata.pdf?ip=false

Semtech. LoRa [Online]. Available:

https://www.semtech.com/technology/lora

Lora-alliance. LoRaWAN specifications [Online]. Available:

https://lora-alliance.org/about-lorawan

Brian Ray. What is LoRaWAN (09.08.2015) [Online]. Available:

https://www.link-labs.com/blog/what-is-lorawan

The Things Network. LoRaWAN [Online]. Available:

https://www.thethingsnetwork.org/docs/lorawan/

Digi-Key's North American Editors. How to get 15 km Wireless(22.11.2016) [Online]. Available:

https://www.digikey.com/en/articles/techzone/2016/nov/lorawan-part-1-15-km-wireless-10-year-battery-life-iot

Sghoslya. Data rate configuration [Online]. Available:

http://www.sghoslya.com/p/table-01-data-rate-configuration.html

Semtech. LoRa modulation basic. (05.2015) [Online]. Available:

https://www.semtech.com/uploads/documents/an1200.22.pdf

RevSpace. DecodingLora. (17.02.2018) [Online]. Available:

https://revspace.nl/DecodingLora

Carl Oliver. CDMA Signal Spreading - The VERY basics of how it's done. (01.05.2013) [Online]. Available:

https://www.youtube.com/watch?v=XJ81CuujwYE

UConn HKN. Digital Communications: Convolutional Codes. (02.12.2017) [Online]. Available:

https://www.youtube.com/watch?v=kRIfpmiMCpU

UConn HKN. Digital Communications: Viterbi Algorithm. (03.12.2017) [Online]. Available:

https://www.youtube.com/watch?v=dKIf6mQUfnY

Electronics Protection Magazine. Condensation Inside Electrical Enclosures and How it Can be Prevented. (01.12.2017) [Online]. Available:

https://www.electronicsprotectionmagazine.com/main/articles/condensation-inside-electrical-enclosures-and-how-it-can-be-prevented/

APPENDICES

N.B. Nous n'avons pas joint les appendices A et B car le premier contient des fiches techniques faisant plusieurs centaines de pages, et le second des programmes informatiques ne pouvant donc pas être contenu dans un fichier PDF.

How to install the Arduino Node

An Arduino Node is a device which is used to send data, the technology used is LoRaWAN and to send data you can modify the settings as you want. This Arduino Node is used for a project in the Østfold University College. This one transmits data like temperature sensors and also light sensors. If you want to understand how we make this kind of device, you can follow this little guide.

First of all, to use this guide you must have these different components:

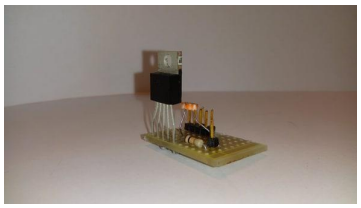
	Arduino Uno Rev 3
	SX1272 Embedded Shield
	Homemade electronic circuit
	Wire type A/B
	Wires

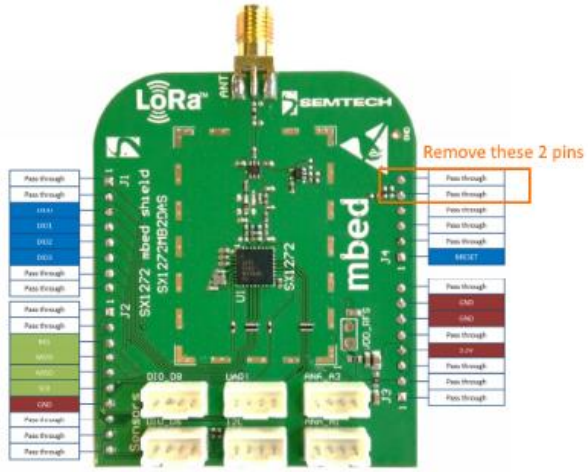
Table of contents

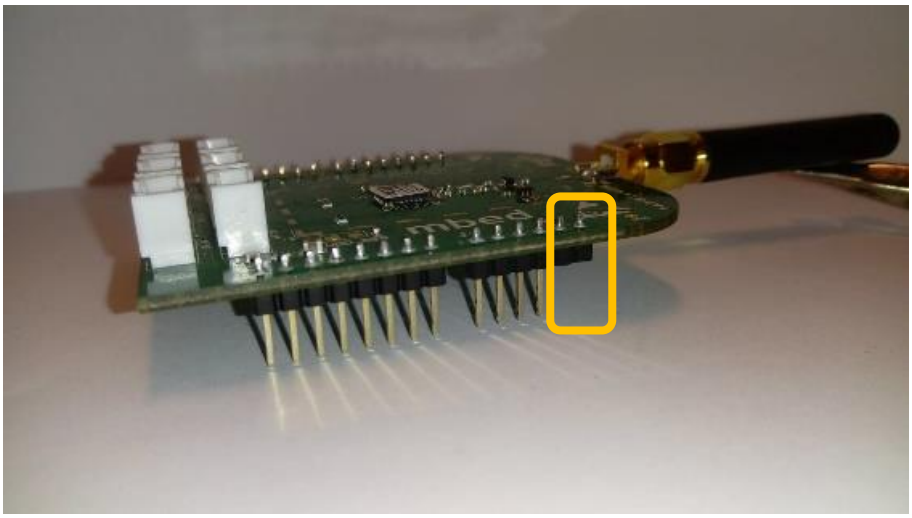
Now, if you have all these things you can start downloading the Arduino IDE and click on the OS that you selected and it will download it.

When this is done, run the file that you downloaded (.exe) and follow the instructions, it is very intuitive.
When the Arduino IDE is on your Desktop, you can follow the steps just below.

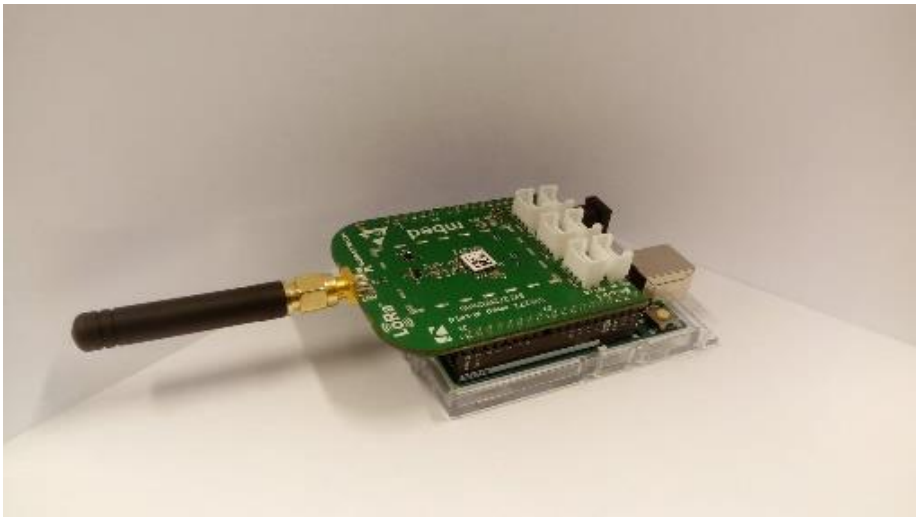
Step 1: Arduino + Shield

Unfortunately, two pins of the SX1272 Embedded shield have to be annulled in order to function with the Arduino Uno Rev 3. This is because the Arduino Uno Rev 3 can only use one I2C protocol at a time, but the SX1272 Embedded shield is expecting to use its two I2C protocols at the same time, thus separating the impulse which results in a voltage drop. This drop dips below what is needed to power the sensors plugged on the SX1272. Confronted to this compatibility issue between two pieces of hardware coming from different companies, we chose this straight and simple solution. So, in order to use our code please desolder or cut these two pins.



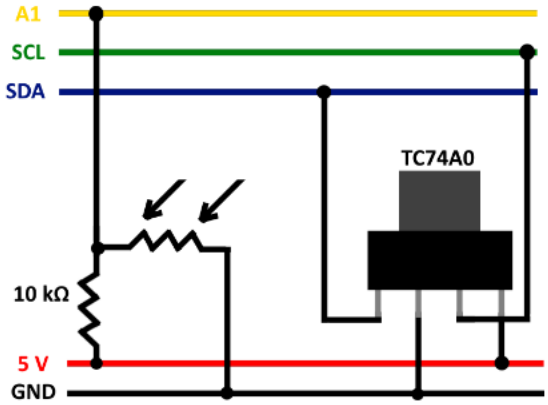


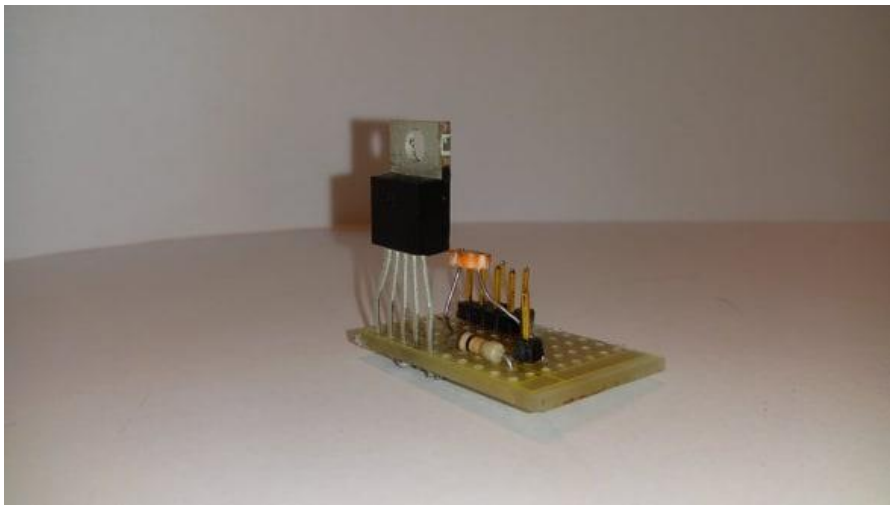
Fit together the Arduino Uno Rev 3 and the SX1272 embedded Shield.



Step 2: Homemade circuit

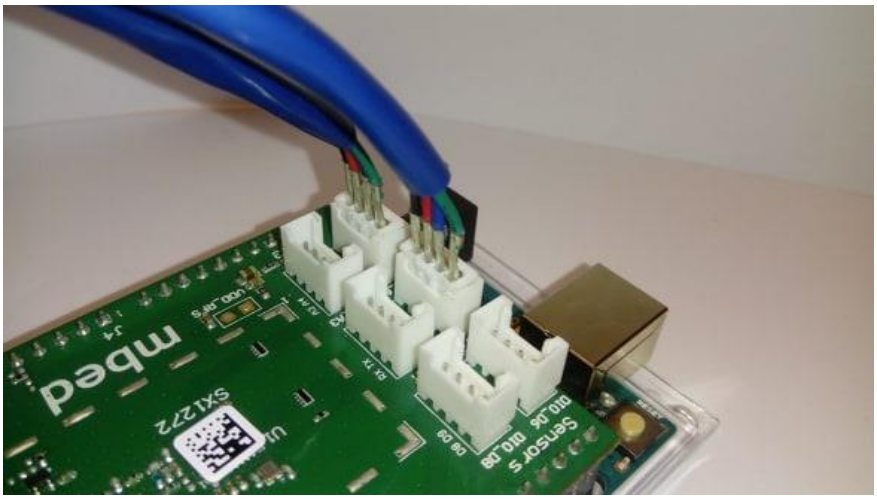
Create this sensors circuit in order to be able to measure brightness and temperature. Light sensor: LDR 12 mm and Thermal sensor: TC74A0.





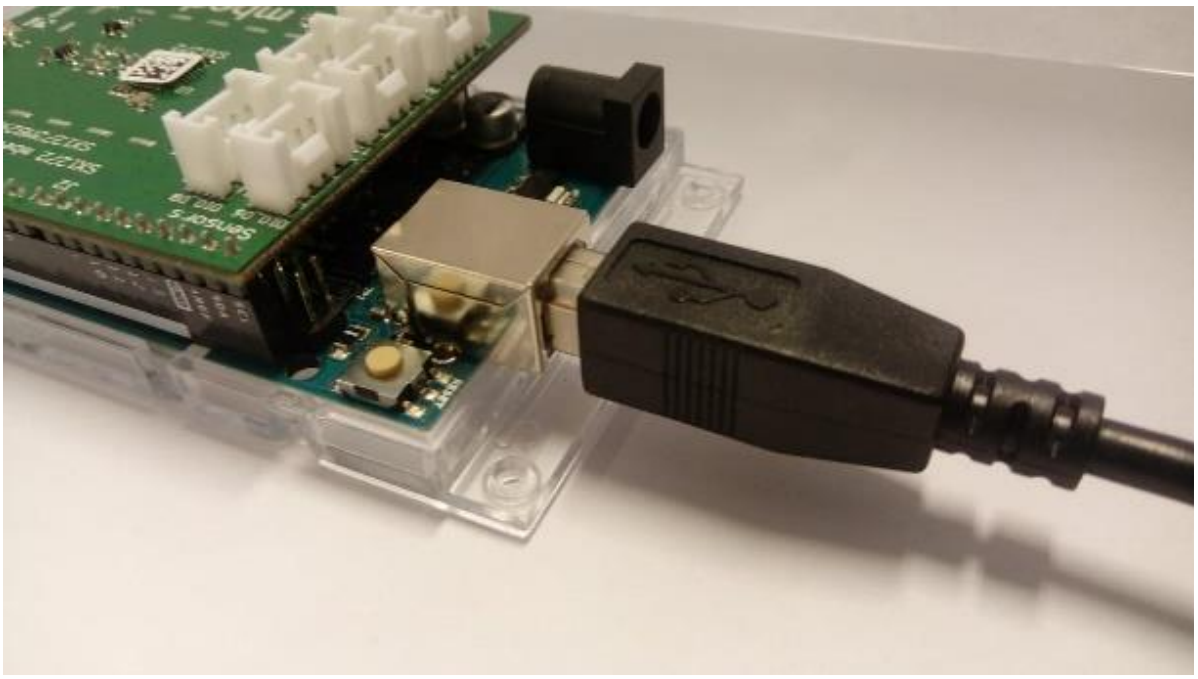
Step 3: Plug the Homemade circuit

Use wires to plug the sensors circuit board to the SX1272 Embedded shield.



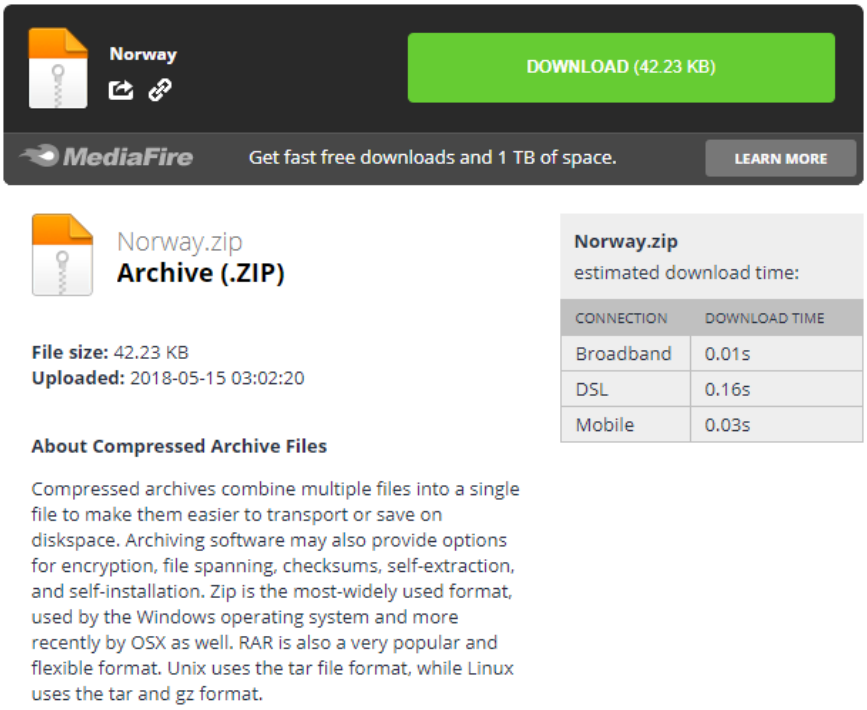
Step 4: Connection between Arduino and a PC

Plug the Wire type A/B into the Arduino Uno Rev 3 and also plug the Arduino Uno Rev 3 into an USB port of your computer.



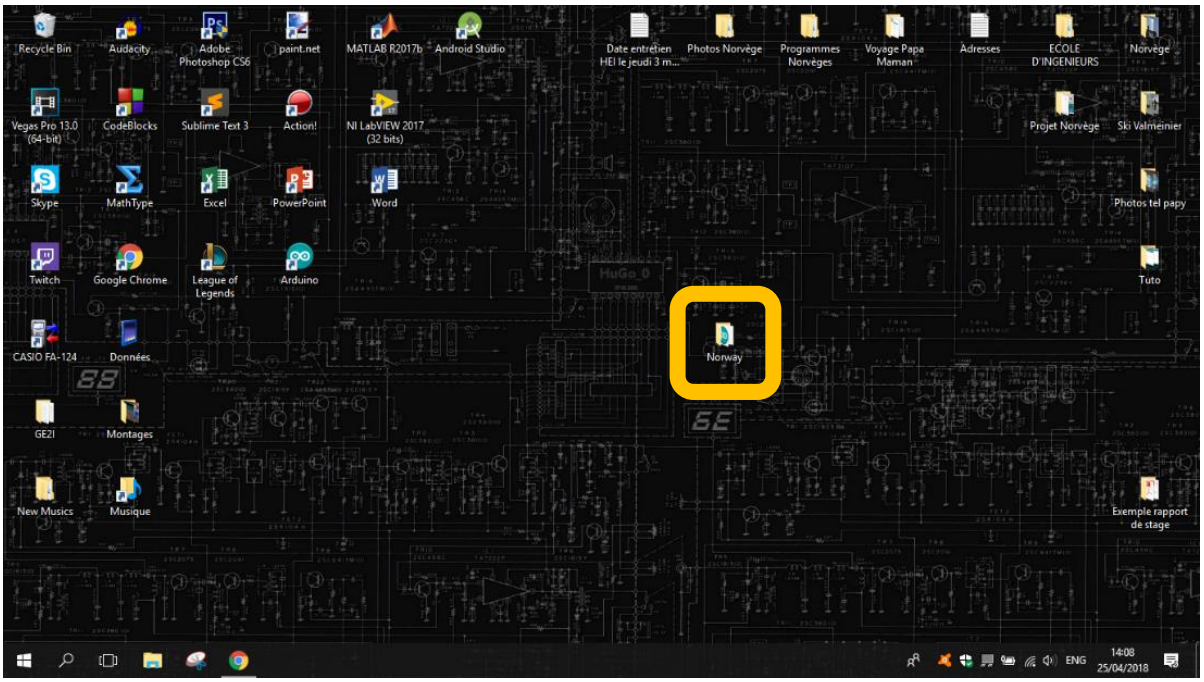
Step 5: Download the code

Download the Arduino called Norway.

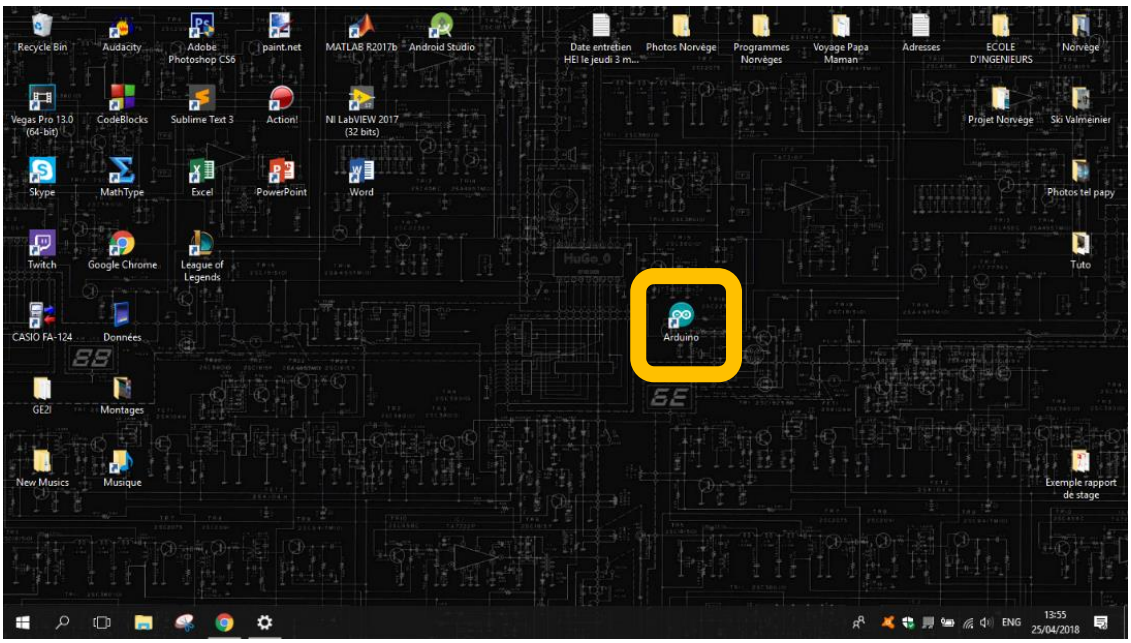


Click here to download

It is a ZIP file so you must extract the folder on your Desktop.

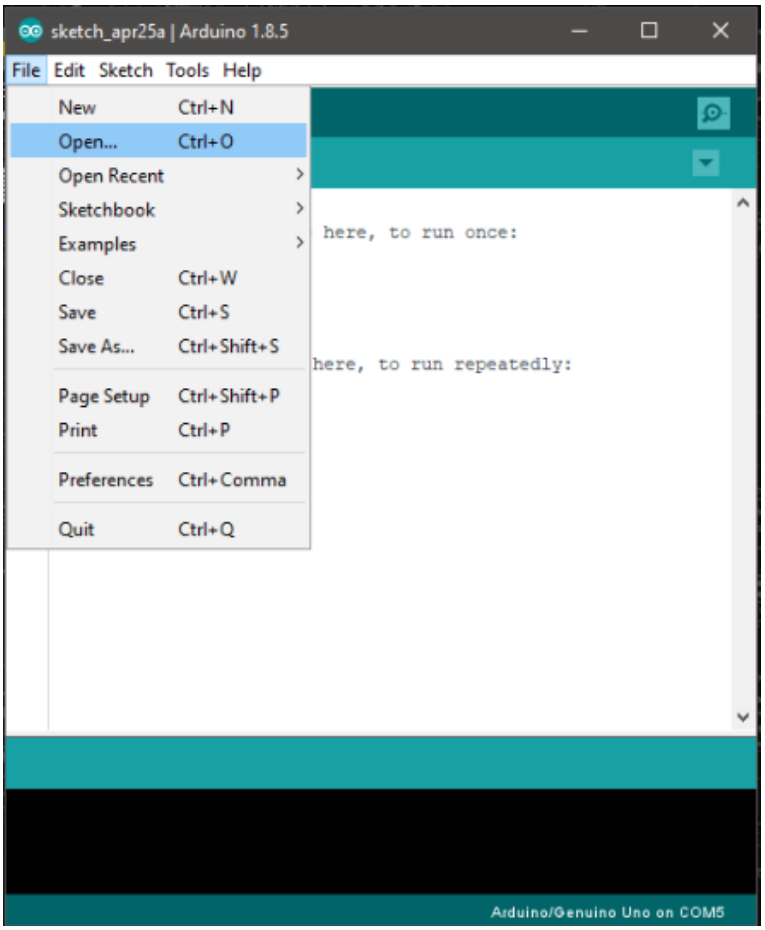


Next, you need to turn on the Arduino IDE.

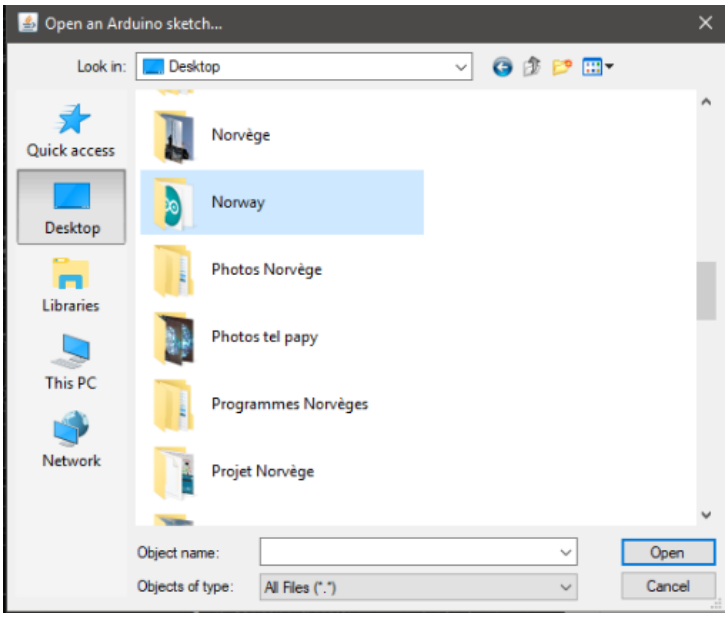


Step 6: Insert the code into Arduino IDE

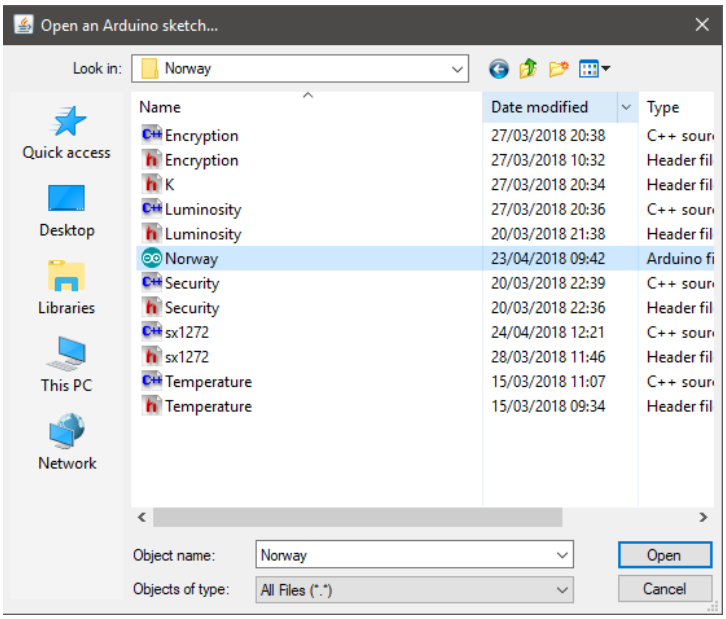
Once in Arduino IDE, go in File and select Open (shortcut: Ctrl + O).



Now select the file on the Desktop named *Norway*.

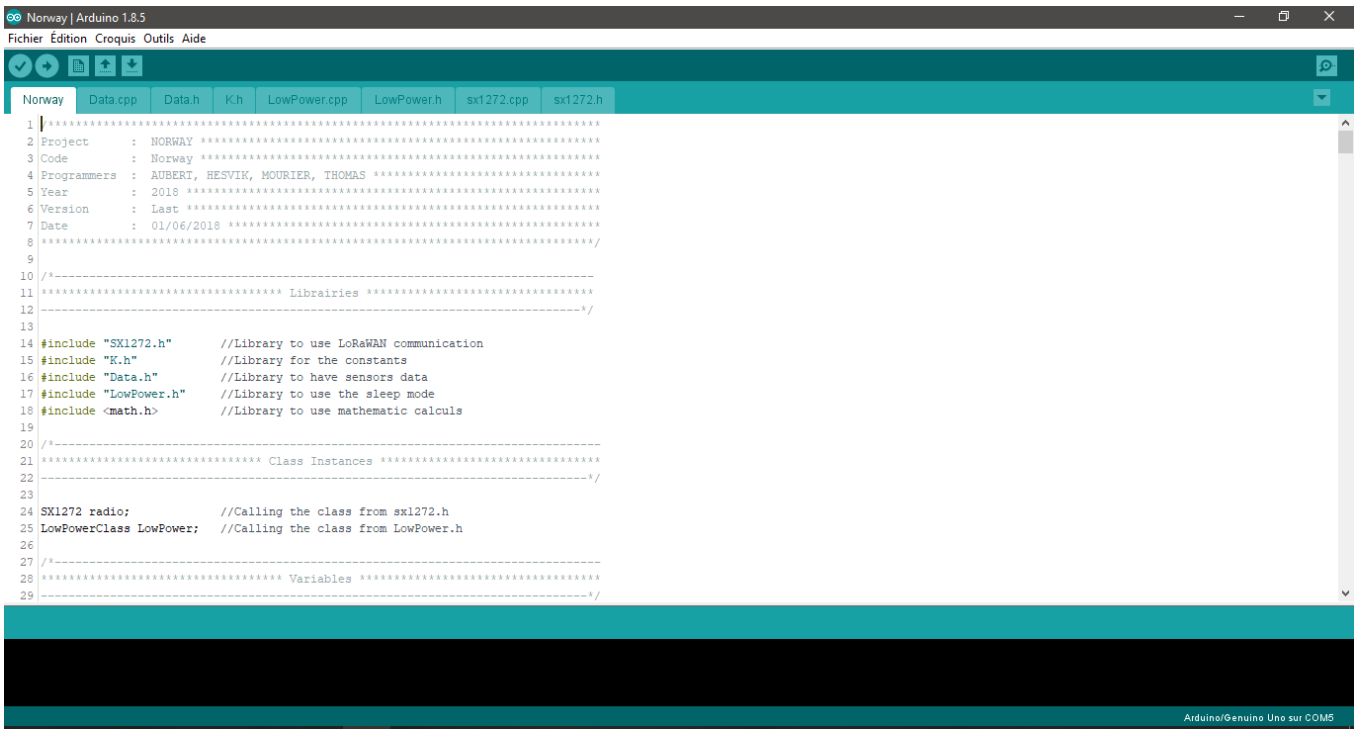


Inside, choose the file *Norway.ino* and click on Open.

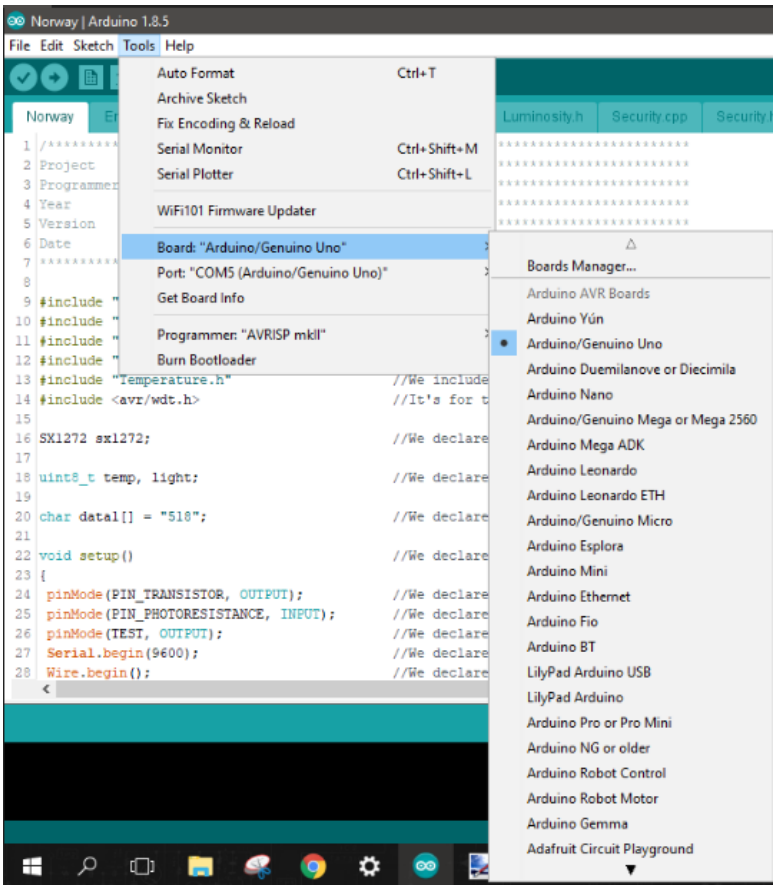


Step 7: Configurations

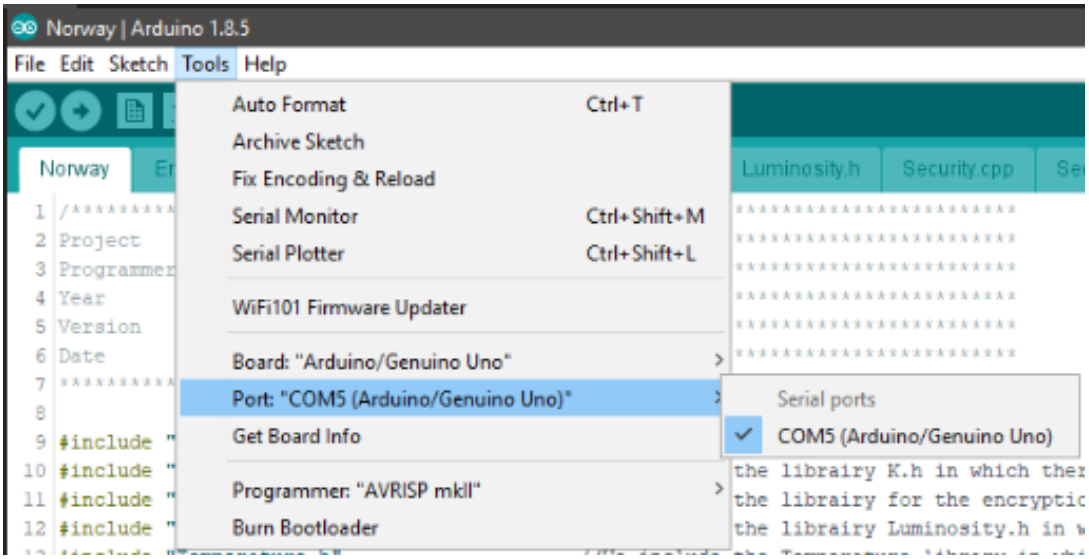
You should get this page:



Now, go in Tool, select Board, and chose the type of card: it is an *Arduino/Genuino Uno*.



Then, indicate the USB port used by the Arduino: Tool -> Port and select the adequate COM. Most often, it is the first one on the list.

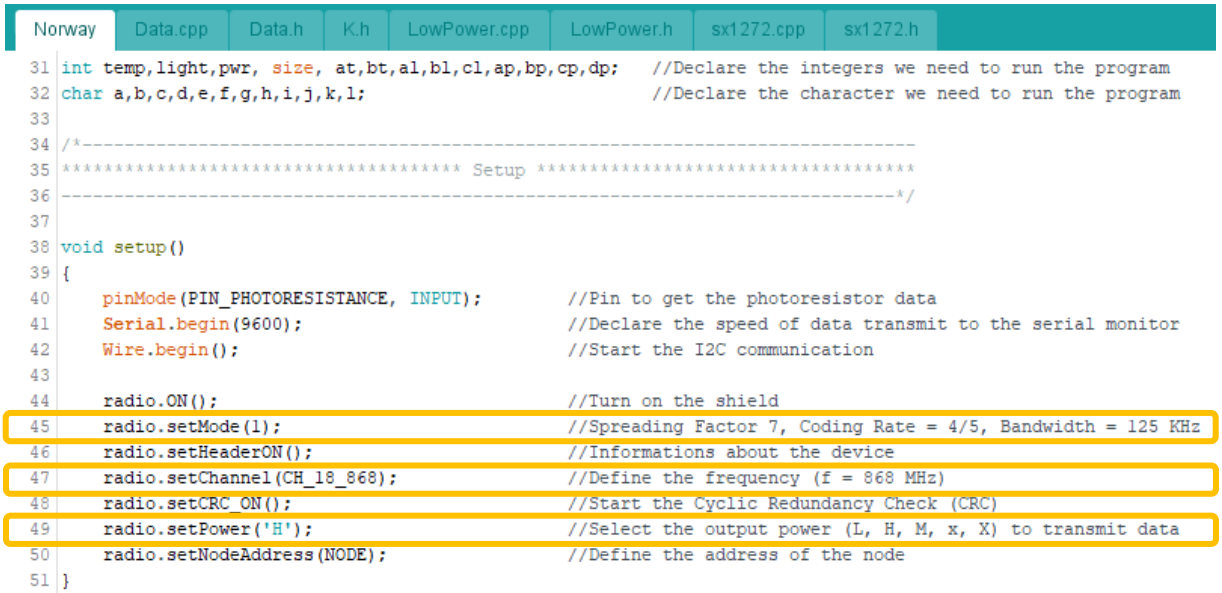


Step 8: Uploading

Now, just upload the code to the Arduino: select Sketch, then Upload, or use the icon surrounded in the picture below.

When the upload is done you should have this message “Done uploading”:

If you want to change things about the LoRaWan transmission, you can modify few things in these yellow rectangles:



If, you want more precisions, find these elements in the code:

Norway | Data.cpp | Data.h | K.h | LowPower.cpp | LowPower.h | sx1272.cpp | sx1272.h

```
1216    }
1217
1218    writeRegister(REG_OP_MODE, LORA_STANDBY_MODE);  // LoRa standby mode
1219
1220
1221
1222    switch (mode)
1223
1224    {
1225
1226    // mode 1 (better reach, medium time on air)
1227
1228    case 1:
1229
1230        setCR(CR_5);         // CR = 4/5
1231
1232        setSF(SF_7);        // SF = 7
1233
1234        setBW(BW_125);       // BW = 125 KHz
1235
1236        break;
```

The picture above is for the Coding rate, the Spreading Factor and also the Bandwidth.

Norway | Data.cpp | Data.h | K.h | LowPower.cpp | LowPower.h | sx1272.cpp § | sx1272.h

```
4784    case CH_10_868:
4785
4786    case CH_11_868:
4787
4788    case CH_12_868:
4789
4790    case CH_13_868:
4791
4792    case CH_14_868:
4793
4794    case CH_15_868:
4795
4796    case CH_16_868:
4797
4798    case CH_17_868:
4799
4800        //added by C. Pham
4801
4802    case CH_18_868:
```

The picture above is for selecting the Frequency.

Norway | Data.cpp | Data.h | K.h | LowPower.cpp | LowPower.h | sx1272.cpp § | sx1272.h

```
5320    case 'X':
5321
5322    case 'M':  value = 0x0F;
5323
5324        // SX1272/76: 14dBm
5325
5326        break;
5327
5328
5329
5330    // modified by C. Pham, set to 0x03 instead of 0x00
5331
5332    case 'L':  value = 0x03;
5333
5334        // SX1272/76: 2dBm
5335
5336        break;
5337
5338
5339
5340    case 'H':  value = 0x07;
5341
5342        // SX1272/76: 6dBm
5343
5344        break;
```

The picture above is for selecting the power of the signal.

Now that the program is loaded, as it is automated you just need to plug the device (Arduino Uno + the SX1272 Embedded shield + sensor circuit board) to any USB port to power it in order to gather and transmit data.

This program will send data to the Lora gateway.

This tutorial is finished. You should now go read the tutorial about How to install the Raspberry Pi Gateway, in order to receive the LoRa transmitted data.

Thank you for reading.

Aurora project members

How to install the Raspberry Pi Gateway

A Gateway is a device which is used to receive data from a Node (in this example, it is an Arduino Node), the technology used is LoRaWAN. This Gateway is used for a project in the Østfold University College. This one gets data like temperature sensors and also light sensors. If you want to understand how we make this kind of device, you can follow this little guide.

First of all, to use this guide you must have these different components:

	Raspberry Pi 3 model B
	iC880-SPI Concentrator module
	Micro SD card 16 GB minimum
	USB Keyboard
	USB Mouse

	Screen
	HDMI wire
	Power supply (5V, 2A) micro USB
	Ethernet wire type RJ45
	Wires

The Gateway allows to gather data from the Arduino node in order to send it to the Ubuntu server where they'll be stocked.

Table of contents

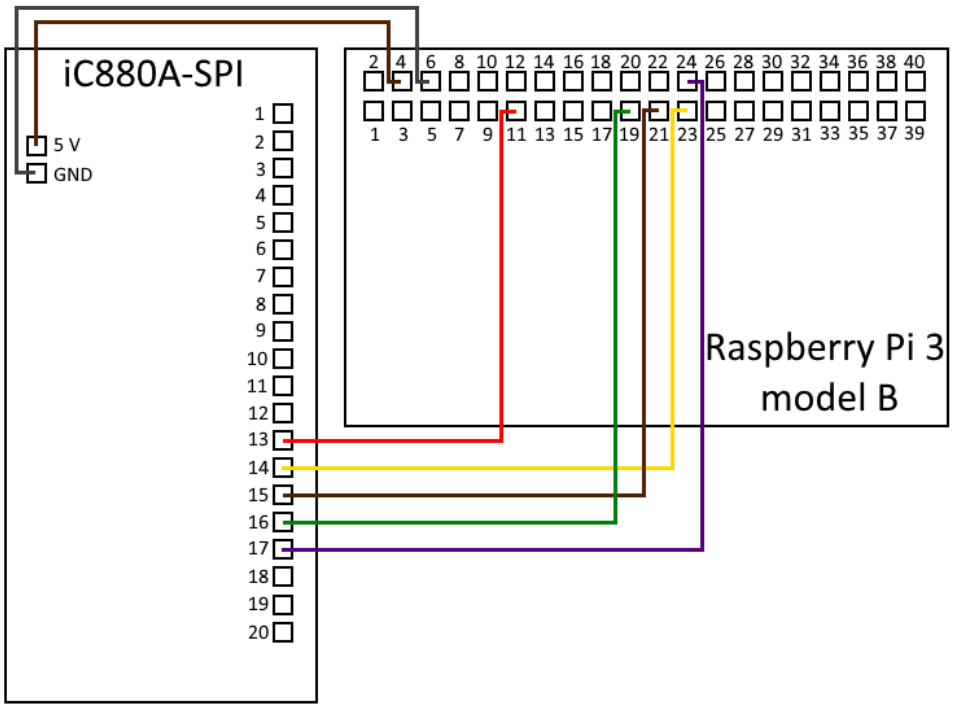
Step 1: Installing the Raspberry Pi

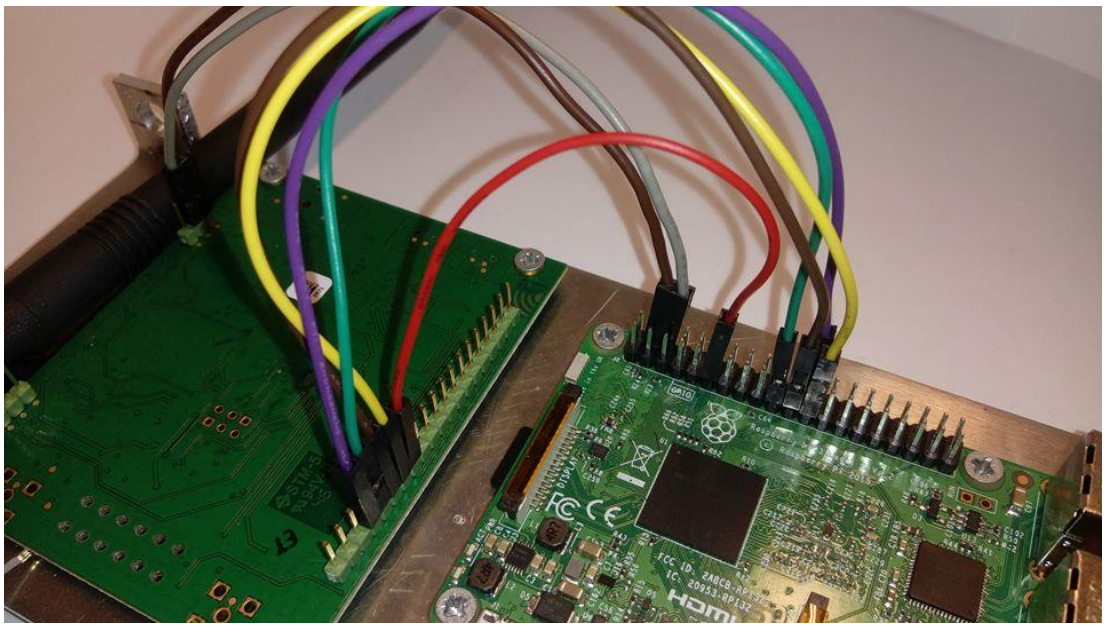
Configure the Raspberry Pi 3 model B by following the instructions using the video below:

Getting Started With The Raspberry Pi 3 by element 14

Step 2: Cabling

Now, you need to connect the iC880A-SPI to the Raspberry Pi 3 model B with the wires, just below, you can see how is cabled the module and the Raspberry Pi.





Step 3: Commands for configuration

Open the terminal window and close it just when it is written "Close the terminal window" because it is important not to close it.
Now type these commands:

Little tip: If you want to copy these commands you can just select the commands with your cursor and do Ctrl + C and after if you want to paste them in the terminal window, you should press Ctrl + Shift + V.

You should download the folder called LoRa, extract it on the Raspberry Pi desktop and finally type the following commands in a terminal window:

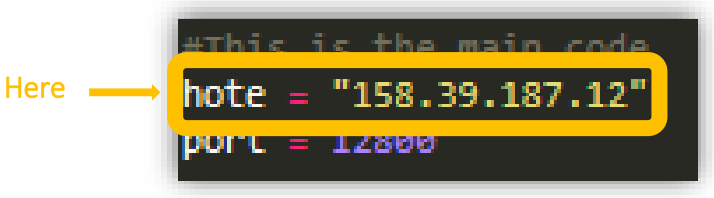
```
cd Desktop
```

```
mv LoRa /home/pi
```

In order to transmit data from the Gateway to the Server, you now need to indicate its IP address. To obtain it you need to write this command on the server's terminal window and after you can close the terminal window:

```
hostname -I
```

Once you have obtained it, go in *LoRa/lora_gateway/lora_gateway/util_pkt_logger* and open the file called client.py and write the server's IP address at the *hote* line, between the double quotes (it is at the end of the code):



Now, you can save the file by pressing Ctrl + S and close it.

Step 5: Compile the codes

In order to use our programs, open a terminal window and type these commands:

Type that to go in the *LoRa* repertory :

```
cd LoRa/lora_gateway/lora_gateway
```

To compile our programs and make them functional, enter this on the terminal window:

```
make
```

You can now close the terminal window.

Step 6: Run the programs

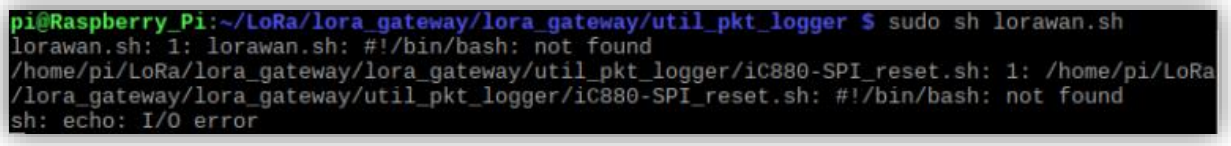
Open a terminal window and type the first command, it permits to locate the file we will use to run our program.

```
cd LoRa/lora_gateway/lora_gateway/util_pkt_logger
```

The second one executes the program to reset the pins of the Raspberry Pi:

```
sudo sh iC880-SPI_reset.sh
```

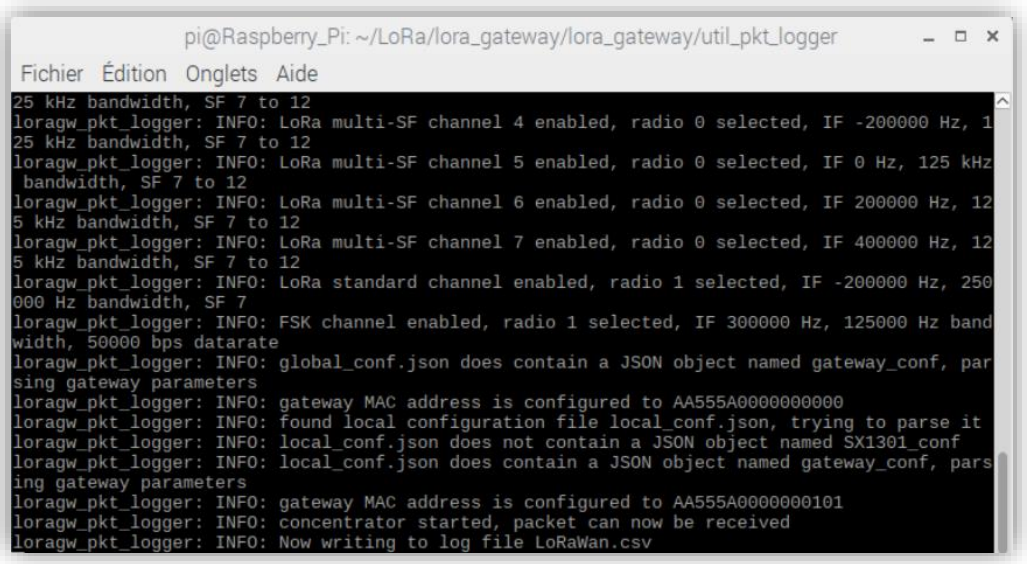
You should obtain this result:



Next, this command allows to get the data from the Arduino Uno Rev 3, and to stock them in the Raspberry SD card:

```
./util_pkt_logger
```

And after this you should obtain this result:



Open another terminal window to write this command. The command will execute a program which sends all the data to the server:

```
sudo python client.py
```

Finally, you should obtain this result:

This tutorial is finished. You should now go read the tutorial about How to install the Apache2 Server and the Website.

Thank you for reading.

Aurora project members

How to install the Apache2 Server and the Website

A Xampp server is a server for Linux which can receive data from a client (in this example, it is the Raspberry Pi Gateway) and can be used for a Website. This server is used for a project in the Østfold University College. This server gets data frame which is composed of different things like the temperature, the brightness and a voltage. If you want to understand how to implement this, you can follow this little guide.

First of all, to use this guide you must have these different components:

	Desktop
	Ethernet wire type RJ45

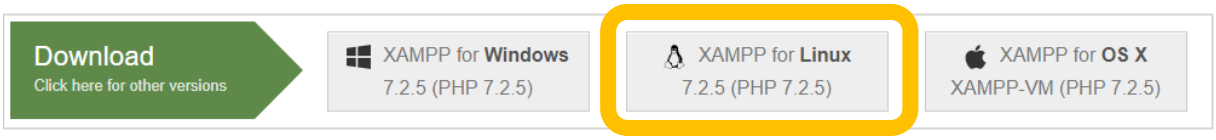
Table of contents

Step 1: Installing Ubuntu

You must have Ubuntu installed on your computer because you need to have a Linux OS, if it isn't done yet you can follow the instructions on this simple tutorial: Install Ubuntu desktop

Step 2: Installing Xampp

We used XAMPP 7.2.5 because it is a simple server which work with our Website. To install it, click on this link and chose *XAMPP for Linux*.



Once the file is downloaded, install XAMPP using the terminal window, first type this command to target the folder where the file you downloaded is located:

```
cd Skrivebord
```

When it is done, you need to type the command which allows to get into the *Administrator mode* and therefore have the necessary rights in order to install the server:

```
sudo su
```

This previous command will request the Ubuntu administrator password. You have to type this password for ours:

```
elektro
```

The next command will grant you the permission for the executions rights, in order to finish the installation. Type this (replace the * by your XAMPP's version number. In our case it was 7.2.5):

```
chmod 755 xampp-linux-*-installer.run
```

Now, run the program by typing this next command (once again, replace the * by your XAMPP's version number):

```
./xampp-linux-*-installer.run
```

The XAMPP server should be installed. Close the terminal window and read how to use it in the next step.

Step 3: Configurations

If you have another server running at the moment, let's say an Apache2, you must stop it beforehand to avoid conflicts with the XAMPP server. To do so, open a new terminal window and type this command:

```
sudo systemctl stop apache2
```

To prevent its possible automatic reactivation each time you boot up the computer, input this command:

```
sudo systemctl disable apache2
```

Step 4: Program to get data from the Gateway

To use the data from the Gateway, download Norway2.zip and unzip it on the Desktop. To do so, open a terminal window and write these commands:

```
cd Skrivebord
```

```
mv Norway /opt/lamp/htdocs/
```

After, write this command to go in the repertory called *opt/lamp/htdocs/Norway*.

```
cd
```

```
cd opt/lampp/htdocs/Norway
```

When this is done, you can now type this command to stop the firewall:

```
sudo ufw disable
```

Launch the program by typing:

```
sudo python server.py
```

Step 5: Starting the server

To run the server, you should type these commands in another the terminal window:

```
cd /opt/lampp/htdocs/Norway
```

```
sudo su
```

Normally it will ask you the password and it is again:

```
elektro
```

To start the server please, write it on the terminal window:

```
/opt/lampp/lampp start
```

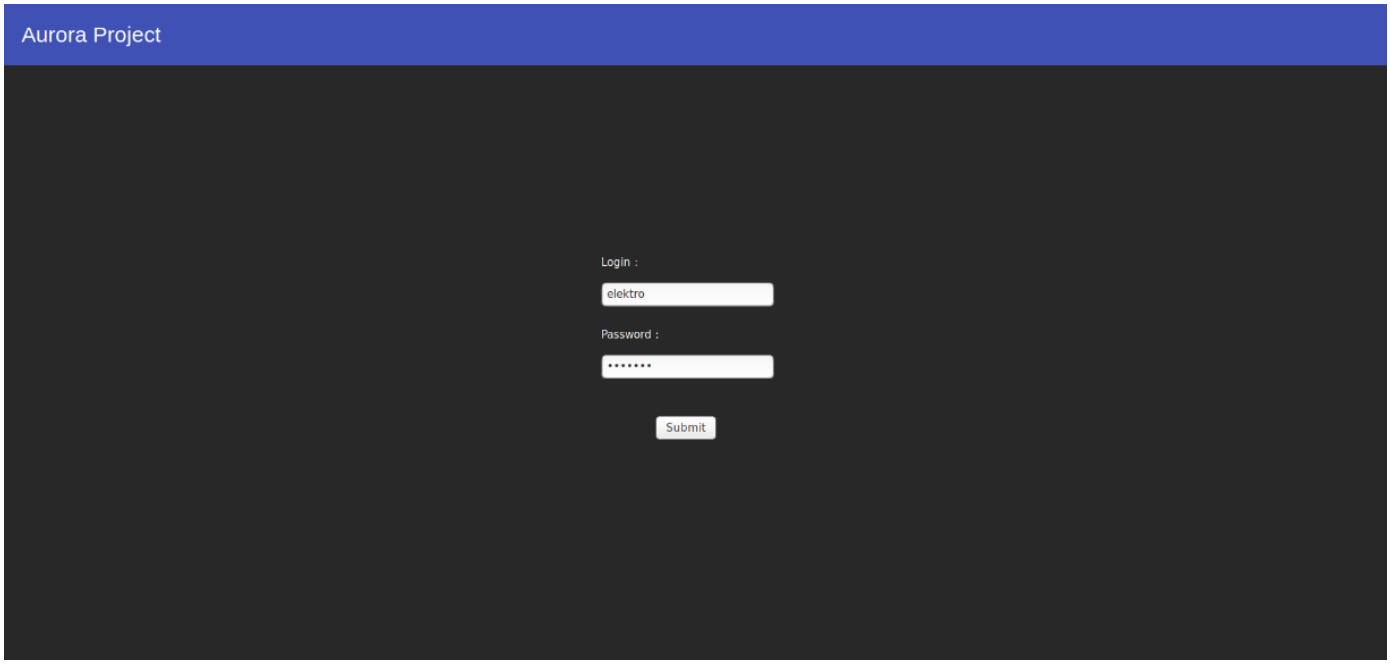
The server is now running and you can close the terminal window.

Step 6: Website

If you want to know the sensors values you can actually read them on a Website. To do that, you have to open Mozilla Firefox and type on the search bar the following thing:

http://localhost/Norway

You will now arrive on this Web page:



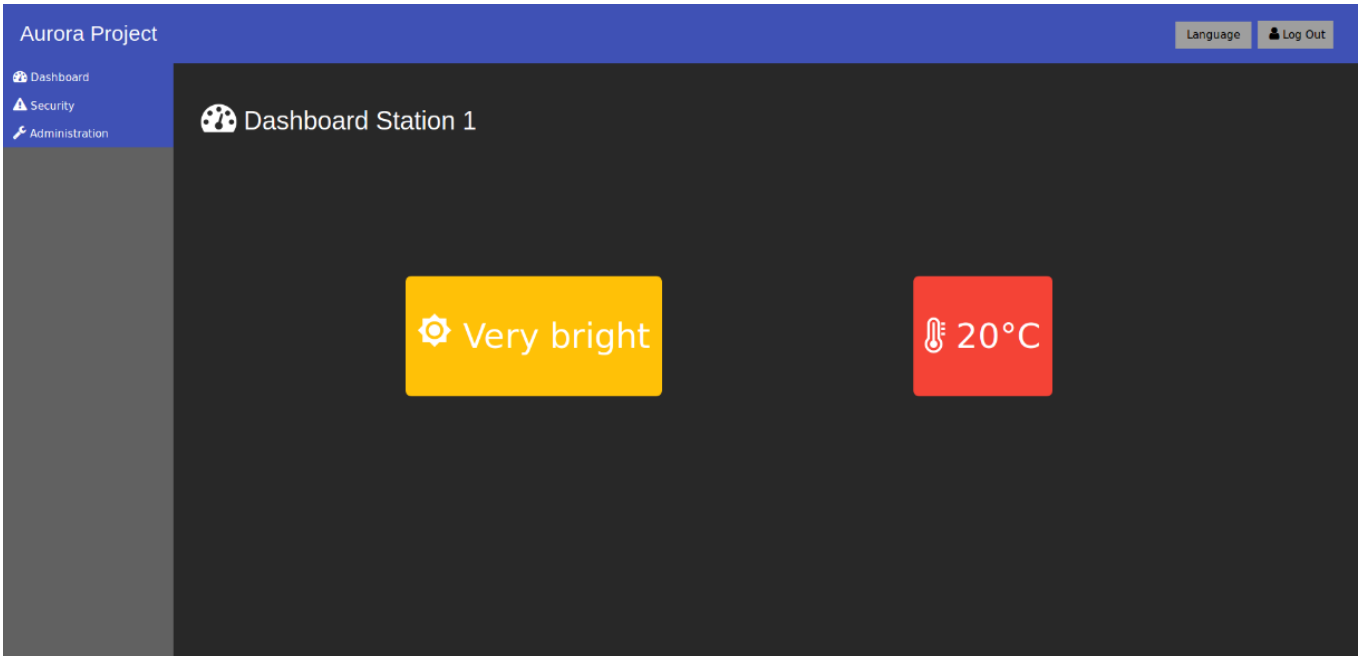
Use this login:

```
elektro
```

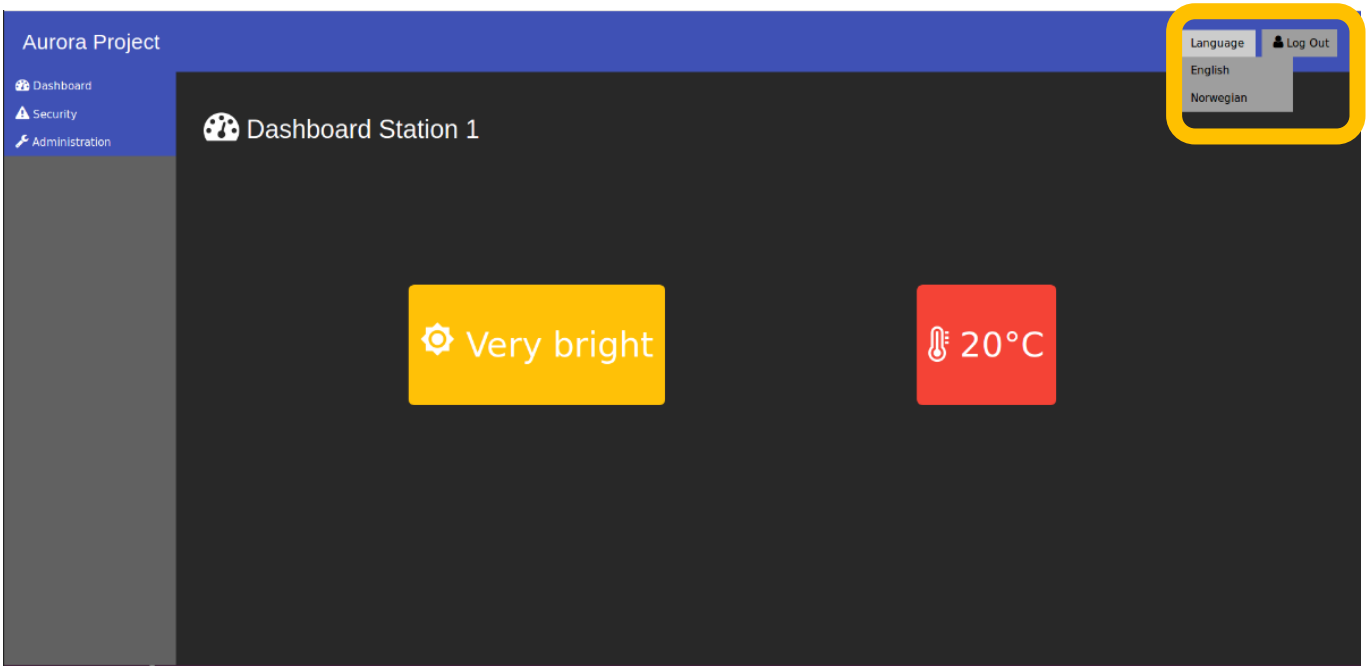
Use this password:

```
elektro
```

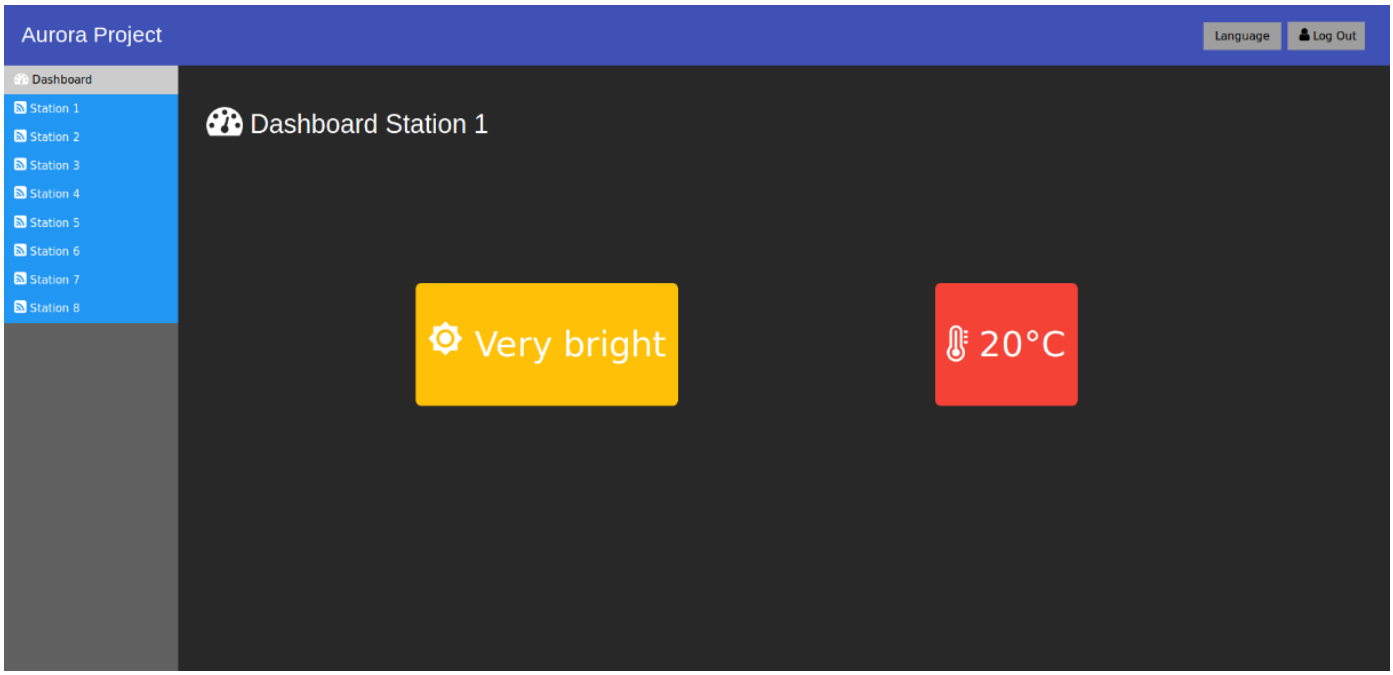
After your login you will arrive on this next page, it is the main page:



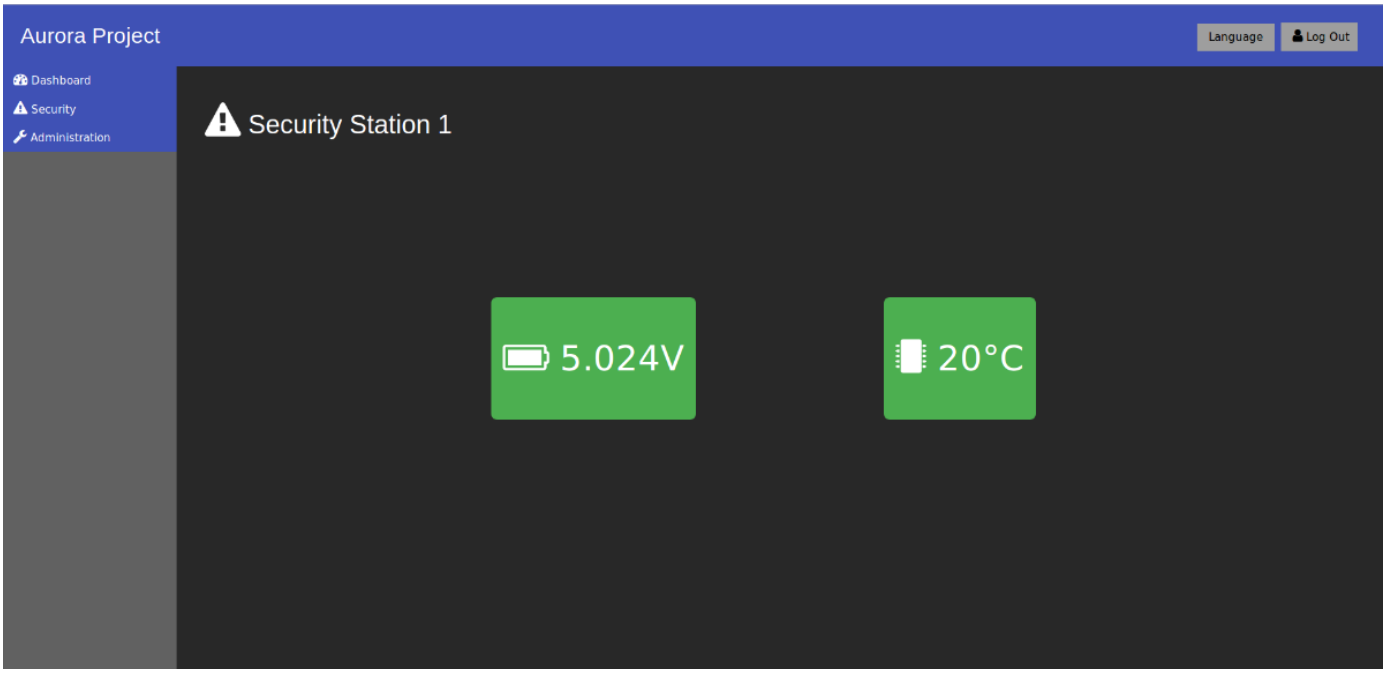
On the top right-hand corner, you click on *Language* to switch between Norwegian and English



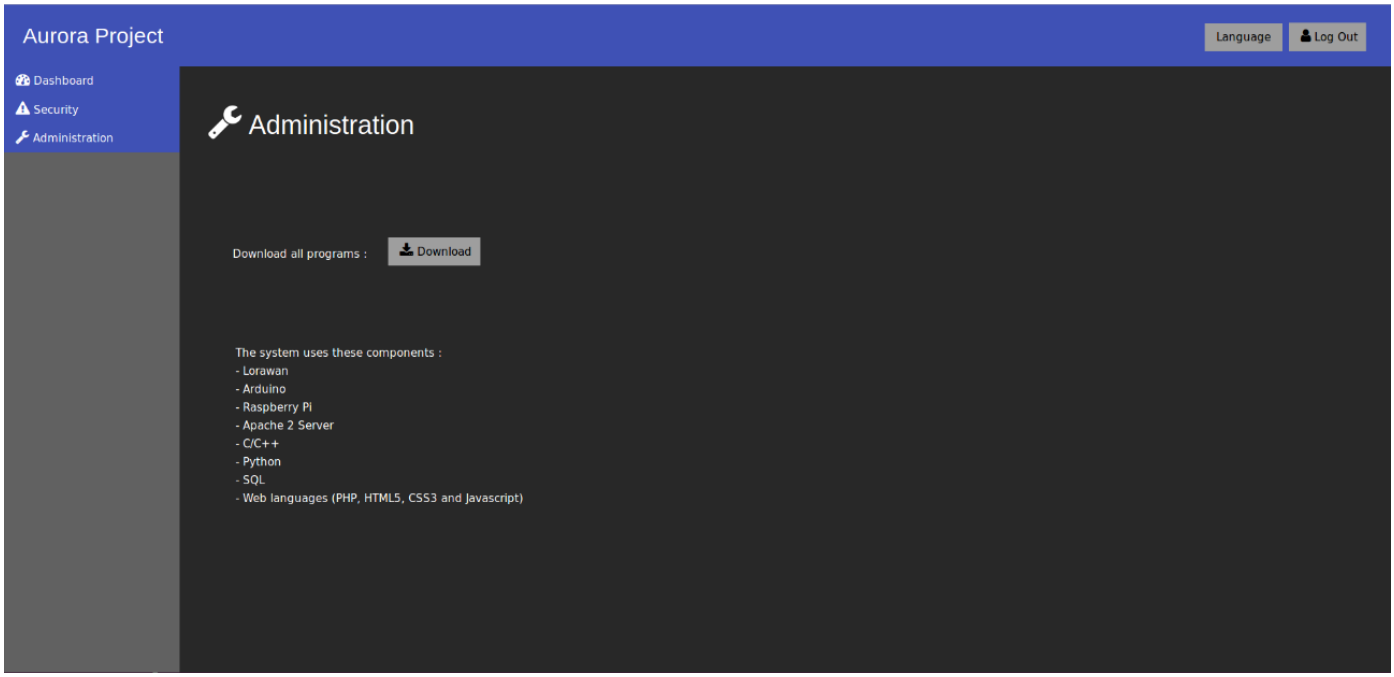
You can also choose your station by moving your mouse over *Dashboard*, and then selecting the one you would like to consult:



You can also check the Arduino 5V voltage line measurement and its CPU temperature: enter the Security Stations by moving your mouse over it



And finally, you can view the page called Administration by clicking on it:



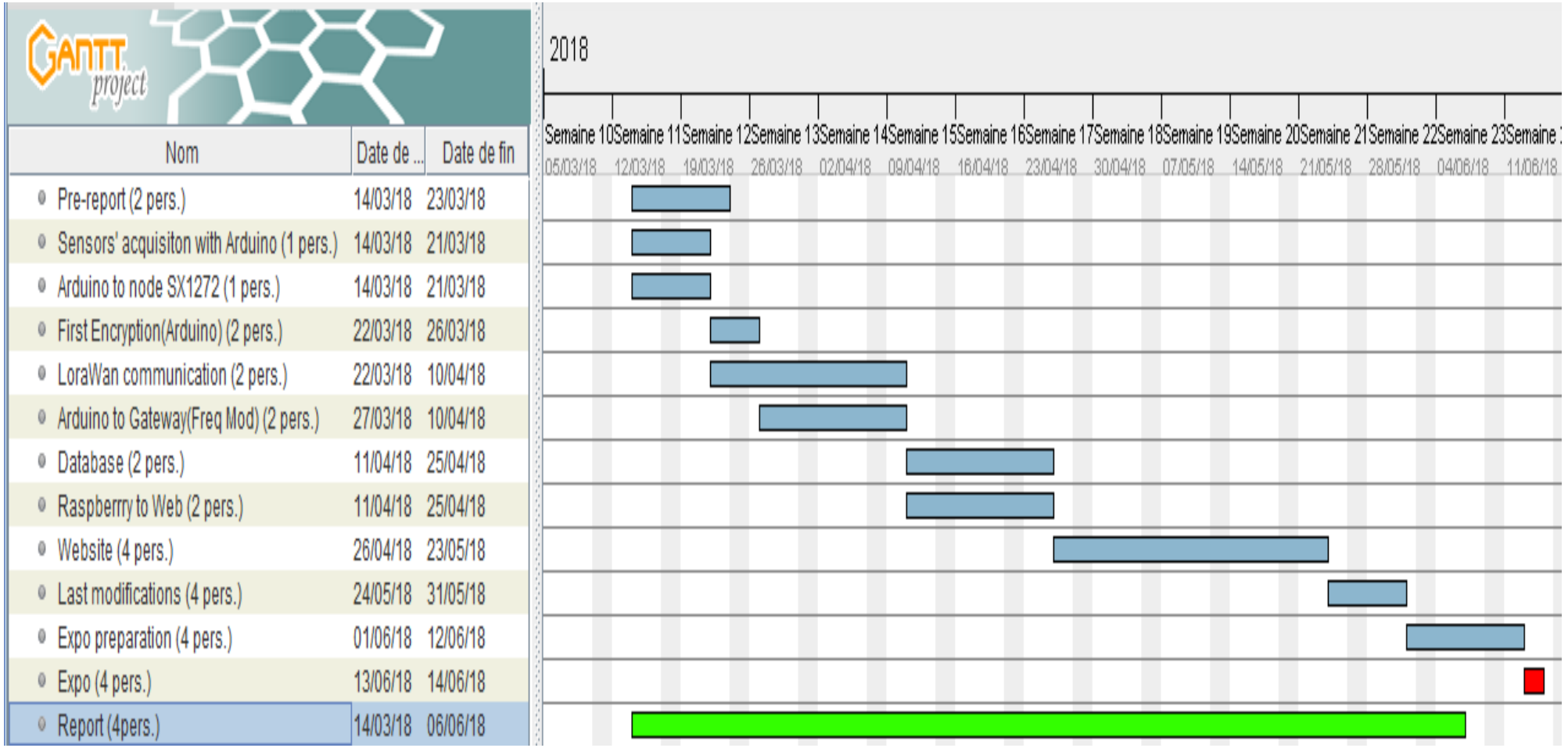
Now the tutorial is finished.

Thank you for reading

Aurora project members

Appendix D: GANTT project

We followed the Gantt project plan close, but there have been some small alterations: the website took a bit less time than we expected since we partially adapted one that we already had. Also, the connection from the Raspberry Pi to the website took us more time than envisioned, because it has partially been coded using Python, a programming language we were not familiar with and so it slowed us down. Nonetheless, the project is completed according to the plan.


GANTT project
2018
Nom
Date de ...
Date de fin
Pre-report (2 pers.) 14/03/18 23/03/18
Sensors' acquisiton with Arduino (1 pers.) 14/03/18 21/03/18
Arduino to node SX1272 (1 pers.) 14/03/18 21/03/18
First Encryption(Arduino) (2 pers.) 22/03/18 26/03/18
LoraWan communication (2 pers.) 22/03/18 10/04/18
Arduino to Gateway(Freq Mod) (2 pers.) 27/03/18 10/04/18
Database (2 pers.) 11/04/18 25/04/18
Raspberrry to Web (2 pers.) 11/04/18 25/04/18
Website (4 pers.) 26/04/18 23/05/18
Last modifications (4 pers.) 24/05/18 31/05/18
Expo preparation (4 pers.) 01/06/18 12/06/18
Expo (4 pers.) 13/06/18 14/06/18
Report (4pers.) 14/03/18 06/06/18
Semaine 10 Semaine 11 Semaine 12 Semaine 13 Semaine 14 Semaine 15 Semaine 16 Semaine 17 Semaine 18 Semaine 19 Semaine 20 Semaine 21 Semaine 22 Semaine 23
05/03/18 12/03/18 19/03/18 26/03/18 02/04/18 09/04/18 16/04/18 23/04/18 30/04/18 07/05/18 14/05/18 21/05/18 28/05/18 04/06/18 11/06/18

Appendix D: Measure current on MCU

D. MEASUREMENTS

D.1 ARDUINO POWER CONSUMPTION

D.1.1 WHY WE MEASURE

We want to measure the power consumption to figure out if and how long we can sustain the node on a solar panels battery that is charging when it is sunny outside. It could be difficult to reach the node location, therefore we want to make it last as long as possible on batteries without having to change them.

D.1.2. HOW WE MEASURE

We used another microcontroller to measure the voltage going into the node, applying the Kirchoff's law. We used two voltage dividers, with 22k ohm and 10k ohm resistors, to scale down 10 V to 3.3 V since that is what Mbed maximum *analog in* input can sustain. We found a scaling value from the 10 V to 3.3 V by using the formula:

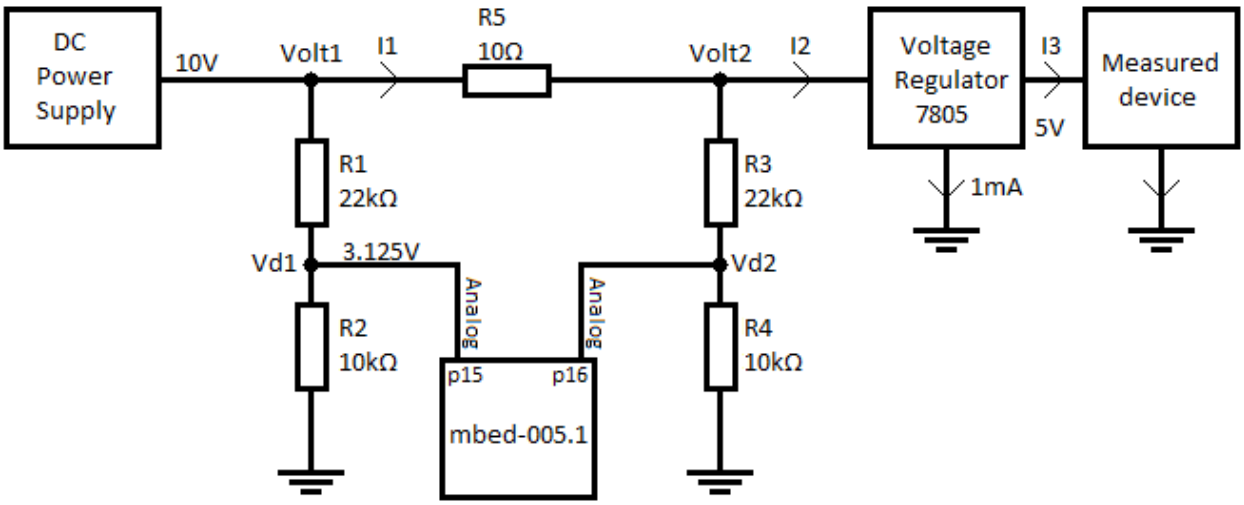
V= M*SF

Volt = DC power supply

M = the value measured on analog in on MBED that varies from 0-1.

SF = Scaling factor we want to find.

With a voltage of 10 volts and a value read on analog in at 0.956777, we found out that the scaling factor is 10.4525.



The 7805 voltage regulator only allows 5 volts into the node, independently of which voltage it receives. To do that, it uses 1 mA.

By summarizing all currents going into the node, we found out what one node consumes.

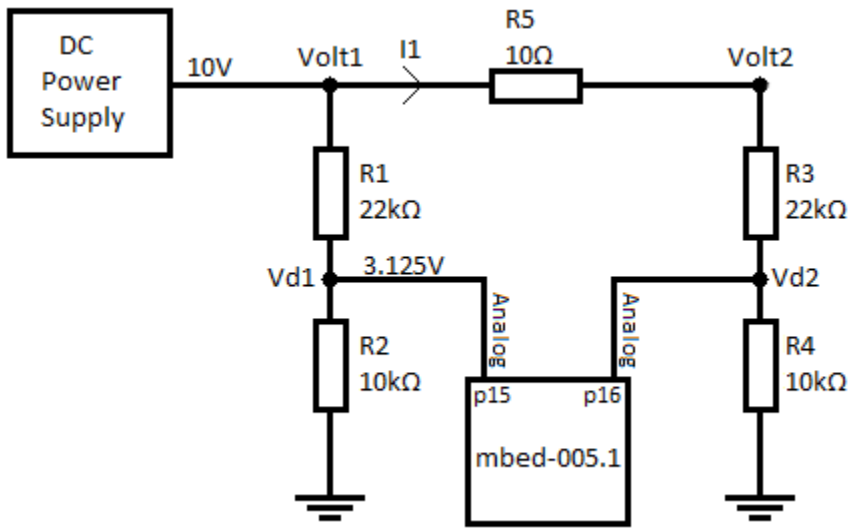
We programmed the following formula, deduced using Kirchoff's law which states that the current entering any junction is equal to the current leaving that junction:

$$\frac{Voltage\ 1 * SF - Voltage\ 2 * SF}{10.9\ Ohms} - \frac{Voltage\ 2 * SF}{32000\ Ohms} - 0.001\ mA = \text{Current going into the node}$$

N.B. The 10 ohm resistor in the circuit actually is a 10.9 Ohm one after measurement.

The average consumption over a period of time is deduced by averaging the obtained sample values.

D.1.3. TESTING THE MEASURING TOOL



Measurements gives values on vd1 and vd2 that are about the same, this tells us that the values we get are correct. This device can therefore be used for testing all sorts of MCU, and units that drain current.

D.1.4. RESULTS

For our system, we want to be able to reduce the power consumption as much as possible, and therefore we want to look into how we can improve the system in such a way that we can make it sustainable on a solar panels battery.

We measured the current consumption on the Arduino Uno node while it was using the different sensors and the LoRa sending device. We tested its current consumption with and without a sleep code, and also reduced the voltage input to the Arduino to check what the outcome would be, and the results are as following:

Arduino power consumption

Test number	Information	sample time	Number of samples	Current	Voltage	Power
1	No code	30 seconds	182	53.34 mA	5 V	266.7 mW
2	Sleep code nr 1	30 seconds	182	40.1 mA	5 V	200.5 mW
3	with sensor information and no TX	300 seconds	1809	55.898 mA	5 V	279.49 mW
4	with sensor information and no TX	1800 seconds	10411	55.498 mA	5 V	277.49 mW
5	only LoRa send constantly	300 seconds	1809	65.496 mA	5 V	327.48 mW
6	Lora send 8 sec sleep	300 seconds	1809	43.992 mA	5 V	219.96 mW
7	Lora send 8 sec sleep	60 Seconds	363	20.9 mA	4.8 V	100.32 mW
8	lORa send 8 sec sleep	60 Seconds	363	22.3 mA	4.7 v	104.8 mW
9	Lora send 8 sec sleep	60 Seconds	363	21.757 mA	4.8109 V	104.67 mW
10	Lora send 64 sec sleep	60 Seconds	363	20.003 mA	4.6 V	92 mW
11	Lora send 64 sec sleep	1800 seconds	10411	20.142 mA	4.6 V	92.6532 mW

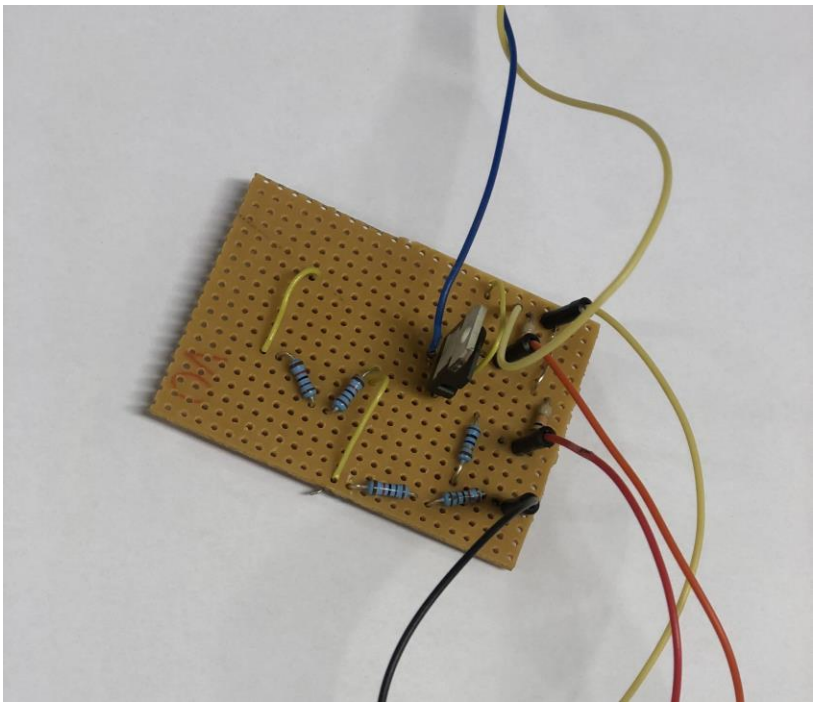
* Sampling 6 times every second.

We first thought of reducing the current the sensors are consuming by controlling a transistor used as an ON/OFF gate. After measuring when the transistor was OFF and when it was ON, we concluded that using it was not relevant enough. Also we made our sleep code disabling all ADC functions for a certain period of time to make it consume less.

Arduino transistor power consumption

Test number	Information	sample time	Number of samples	Current	Voltage	Power
1	Let power through	60 seconds	363	55.45 mA	5 V	277.25 mW
2	Dont let power through	60 seconds	363	55.89	5 V	279.45 mW

* Sampling 6 times every second.



We have also made a program on Mbed able to sample both the Vd1 and Vd2 values every 0.2 seconds approximately, then it converts them to Volt 1 and Volt 2, after it uses the formula to convert them to current, and finally makes an average of a set period of time. When the time expires, it gives out the average current, and the number of samples it used.

D.2. RASPBERRY PI 3 POWER CONSUMPTION

We want to make an expense report and see how much the whole system costs per year. For this, we need to know what all entities consume. The Raspberry PI 3 is a very energy efficient minicomputer. Because of this, it suits very well with our setup. When idle, it consumes 1.4 W.

Pidramble. Power Consumption Benchmark [Online]. Available:

https://www.pidramble.com/wiki/benchmarks/power-consumption

Appendix D: Meeting request email status update (31.05.18)

The meeting will be held at S-504 and must be finished before 13:45

The agenda:
- You tell me the status of the project
- You ask, -I try to ansver.
- What to do next.

Best Reidar

_____
Fra: brede hesvik <brede_hesvik@hotmail.com>
Sendt: 29. mai 2018 14:49
Til: Reidar Johannes Nordby; Adrien Thomas; antoine.mourier@etu.univ-orleans.fr; guillaume.aubert@etu.univ-orleans.fr
Emne: Formal meeting on Thursday

Hello,

Calling in a formal status meeting at 12:30 on Thursday.

Regards

- Brede

Appendix D: Meeting report

04/04/18:
We gave a status update on the following:

- Node
- Gateway
- Server
- Encryption
- Pre report
- Main report
- Evolution of Gantt

We talked about adding a solar panel and batteries to our project if we have time.

We talked about theoretical stuff:

- Reidar mentioned CDMA is used in the LoRa chirp modulation
- Reidar mentioned that he will most likely buy an SDR, which we could use to analyze LoRa and produce similar signals.

We talked about how our grades are set:

- Presentation
- Report
- Process underway (this log is part of the process, and how structured we are)

We asked for some equipment:

- Power supply for raspberry pi 3
- Grove connector for sx1272

31/05/18:
We reviewed the actual state of the report.

☆ **Reidar Johannes Nordby** Innboks - Hotmail 14:03

SV: Pressemelding

Til: brede hesvik

Hei Brede

Denne pressemeldingen er godkjent!

mvh
Reidar Nordby

Ta vare på denne meldingen og legg den i prosesssvedlegget til rapporten sammen med pressemeldingen.

Fra: brede hesvik <brede_hesvik@hotmail.com>
Sendt: 30. mai 2018 13:26
Til: Reidar Johannes Nordby
Emne: Pressemelding

Hei,

står ikke noe sted hvor pressemeldingen skal sendes, om den skal leveres på fronter eller ikke. Holder det med å levere den til deg?

- Brede

Appendix D: Email to klima vakten

Svar fra instrumentkyndige.

---------- Videresendt e-post ----------
Fra: **Hildegunn V. Dyngeseth Nygård** <hildegunnd@met.no>
Dato: 26. mars 2018 kl. 14:52
Emne: Re: Hjelp til bacheloroppgave
Til: Felleskonto Klimavakten <klimavakten@met.no>

Det han ønsker seg er altså sensorar som skal varsle om det kjem skyer som fører til dårlig lading på solcellepanelet? Det fins jo ingen sensorar som kan varsle at skyer er på vei. Då er det varslingsmodell han må koble dette til. Og så er det jo forskjell på skyene. Tynne høge skyer gir jo ikkje så stor reduksjon i solcelle-lading som nedbørskyer. Vil kanskje anbefale at han heller konsentrerar seg om nedbørstilfeller. Då kan han bruke radarbildene (nåvarselet).
Du kan jo sende til SUV, men eg er usikker på kven som kan svare.

Hildegunn

26. mars 2018 kl. 14:19 skrev Felleskonto Klimavakten <klimavakten@met.no>:
Hei Hildegunn,

Vet du om noen som kan svare på dette hos dere?

Hilsen Elin

---------- Videresendt e-post ----------
Fra: **Felleskonto Klimavakten** <klimavakten@met.no>
Dato: 26. mars 2018 kl. 14:18
Emne: Re: Hjelp til bacheloroppgave
Til: brede hesvik <brede_hesvik@hotmail.com>

Hei igjen Brede,

Dette høres litt for avansert ut for oss. Vi har ikke noe med slik solcelleparker å gjøre. Du kan ikke ta kontakt med noen som har driver eller har drevet med det. Vi driver jo med temperaturmålinger, men har du tenkt å koble dette til skyer?

Jeg sender over forespørselen din til noen som jobber med meteorologiske instrumenter.

Hilsen Klimavakten

26. mars 2018 kl. 14:07 skrev brede hesvik <brede_hesvik@hotmail.com>:
Takker for svar.

Ja, det kan være det blir litt for avansert og mye å sette seg inn i. Mulig min forklaring var litt vag, men skal prøve å gi et litt mer forklarende spørsmål.

Vår oppgave er å finne ut når en solcelle park må hente inn strøm fra en batteribank så kunder ikke mister strømmen når solcelle parken ikke får inn tilstrekkelig kraft. Planen er derfor å kunne så tidlig som mulig ved hjelp av sensorer tilsi når dette kan skje, ved eksempelvis at skyer vil komme og redusere effekten. Vi programmerer mikrokontrollere selv til å gjøre dette for oss som sender informasjonen over langdistanse signaler.

Om det ikke blir en altfor komplisert oppgave å kunne få indikasjoner på skyer ved hjelp av temperatur sensor, lyssmålere, trykkmålere og diverse andre sensorer, så kunne dette vært aktuelt for oss.
Er det altså mulig ved hjelp av noen få sensorer å kunne få indikasjoner på skyer? Eller blir dette som du sier litt for avansert?

Det er ikke i selve beskrivelsen av bacheloroppgaven at vi skal kunne utføre dette, men noe vi har tenkt som tilleggsoppgave.
Skal prøve å se litt på deres verktøy, men i verste tilfellet kan jeg se på deres værdata. Det er mulig at det finnes en del artikler på dette, men tenkte det var raskere å sende dere en email om hvor aktuelt det kan være for min del før jeg begynner å sitte meg inn i dette.

Mvh

Brede Hesvik

26. mar. 2018 kl. 13:38 skrev Felleskonto Klimavakten <klimavakten@met.no>:

Hei Brede,

Dette høres jo svært interssant ut. Jeg er litt usikker på vinklingen her. Skal du kanskje følge med når lavtrykk er på gang eller fronter er på vei innover Norgee.l. Så kan du jo kanskje bruke noen av våre verktøy, men det er litt komplisert å lære. Har du hørt om Diana? Det finnes ikke artikler på dette? Mulig at undersøkelsene du skal gjøre her blir litt for store. Så du burde ta en snarvei om et litteraturstudiet først.

Det høres ut i hvert fall ut som ganske avansert forskning. Mulig du burder diskutere litt mer med veileder. Her er det mye å ta tak i.

Trenger du å finne mer ut om værdata/klimadata?
så logg deg inn på vår tjeneste:
http://eklima.met.no og registrerer deg som bruker.
- og ta ut de værdataene du ønsker

Hilsen
Klimavakten

v/Elin Lundstad
Klimaforsker

Mobil 48 06 63 58
Sentralbord 22 96 30 00

Meteorologisk institutt
Henrik Mohns plass 1,
Postboks 43 Blindern,
0313 Oslo
www.met.no

yr.no

---------- Videresendt e-post ----------
Fra: **Meteorologisk institutt** <metinst@met.no>
Dato: 26. mars 2018 kl. 10:41
Emne: Fwd: Hjelp til bacheloroppgave
Til: Felleskonto Klimavakten <klimavakten@met.no>

Skal denne til dere?

Mvh Trine

---------- Videresendt e-post ----------
Fra: **brede hesvik** <brede_hesvik@hotmail.com>
Dato: 26. mars 2018 kl. 10:32
Emne: Hjelp til bacheloroppgave
Til: "post@met.no" <post@met.no>

Hei. Jeg holder på å skrive en bacheloroppgave der vi skal kunne forutse hvor mye effekt en solcelle park leverer. Vi lager sensorer selv som skal måle blant annet lysmengde og temperatur. Håper dere kan hjelpe meg litt med følgende spørsmål: er det mulig å forutse når det er stor sjanse for at skyer kan komme, utifra sensor informasjon som vi har muligheter til å benytte? Eventuelt andre metoder vi kan bruke til å finne ut når solcelle effekt vil gå ned?

Mvh

Brede Hesvik

Appendix D: project daily logs

Week 11

12/03
Visited the university with Andreas and Malin.

13/03
Met with Brede and Reidar. Project presentation : what it is about and technical details.

14/03
Created a Google Drive.
Did the Gantt diagram with all the tasks.
Started getting some informations about the LoraWan network.
Plugged the temp & light sensors on the Arduino board. Adapted the voltage from 3.3 to 2.2v for the light sensor. Wrote a program (with *Arduino IDE*) to process the sensors data.
Started the pre-projet paper.

15/03
Wrote the part for the report about the voltage divider for the light sensor, and gathered documentation.
Improved the voltage divider to get 1.3v, as to drain less battery.
Finished to do the last modifications for the code, and we did the commentary of the code too, by the way we reuploaded it on the Google Drive.
Worked on the encryption and made a simple encryption for the first step, we'll code the program this week-end.
Wrote more on the pre-project.

16/03
Used a transistor to cut power to the sensors when the Arduino isn't reading them. Adapted the resistor employed to divide the current for the light sensor, since it's now plugged on the same output than the temp sensor (5v instead of 3.3v).
Wrote more on the pre-project.

Week 12

19/03
Researches about the LoRaWAN : theory (freq modulation (video : Decoding the LoRa PHY (33c3)) and code (datasheet for set up, trying to find already done code, etc)
Created a schematic for the arduino/sensors routing (to be used for others, and for us if we unplugged it).

20/03
Booted up the Raspberry, installed Raspbian. Managed to lit some LEDs with it.
Further researches on LoRaWAN. Managed to turn on/off the LoRaWAN shield using Arduino code.

21/03
Managed to transmit a message using LoRaWAN with a SX1272 (on an Arduino), then received by another SX1272 (on an Arduino).

22/03
Further testing of the LoRa connection between the 2 Arduino.

Researches on how to modify a code found on the internet, to send data without using The Things Networks servers.
Commenting the .h Arduino code for the SX1272.

23/03
Finishing commenting the .h Arduino code for the SX1272.
Continuing research for the gateway.
Research about encrypting.
Got some information about error correction, from Reidar : convolution encoder (which can include encryption), and block encoder.

Week 13

26/03
Brede is back from his trip.
Work on the pre-report. Started the final report (structure).
Work on our own encryption method, and coding it.
Further research on the LoRa theory.
Research about setting up an Apache server on an Ubuntu OS. Trying to connect to the one already set up.
Measured a linear change in the temp sensor when dropping it's voltage from 5V (until ~2.5v).

27/03
Measured a linear change in the temp sensor when dropping it's voltage from 5V.
Continuing encryption.
Continuing testing the server previously installed on the computer we'll use for that as well.
Managed to send data between the Raspberry and the server's console.

28/03
Further refinements about the report structure. First draft of theoretical explanations.
More research about LoRa tech.
Set up the gateway to receive data.
Maybe (?) managed to send data between the Arduino and the Raspberry (received a lot of different data, probably from other devices not belonging to us too)

29/03
Managed to send data from the node (Arduino) to the gateway (Raspberry), and read the data (it's in hexa, but translated using the ASCII table give the original message).
Managed to isolate data received by the gateway to only ours, by setting the Tx and the Rx with the same values (spreading factor, coding rate, bandwidth, central frequency).

30/03
Same as yesterday.
Preparing travel to Tromso.

Week 14

02/04
Managing to send data from the Raspberry (.csv on its SD card) to the server's console again (was a firewall problem, and connection problem).
Working on translating hexa data into decimal (in the Raspberry).
Redacting report.
Research about asymmetric communication, LoRa devices classes.

03/04
Started adding the code to correct the temp read by the sensor if the Arduino voltage drops.
Meeting Reidar (shown pre-report). Prepared the meeting with him tomorrow.
Managed to pick up any char from the string of data (from the .csv in the Raspberry SD card) to the server's console.
Managed to translate hexa to decimal (in the Raspberry).

04/04
Meeting with Reidar Nordby. Details in main report/meeting logs.

05/04
Researches about CDMA (how encode data to transmit several data on the same channel/frequency at the same time).
Managed to transmit data only if their state is correct ("ok", not "bad"). Trying to implement a sleep mode for the Raspberry code so that it wait for the next data, instead of shutting down.

06/04
We planned our trip to Tromso.

Week 15
Frenchies in Tromso. "Bon courage Brede =)"

09/04
Brede: Gone through layout of the setup, and then had a meeting with reidar and done changes.
Writing on report.

10/04
Brede: Made accounting sheet for all the parts what each cost, where to find and total cost.
Writing on report.

11/04
Some small writing on the report.

13/04
Researched CDMA
Updated blog
Writing on report.

Week16

16/04
Started designing of the box housing the node.
Started proofreading the final report.
Resolved the writing/reading conflicts on the .csv (raspberry file).
Researches about how to send the data from the server to the database.
Made cables (arduino shield to sensors)
Researched stuff.

17/04
Trying to write data on the database files (.db) coming from the Raspberry. SQL language.
Trying to measure sensors power consumption, using two method (one not finished yet).

Continuing proofreading the main report (which triggered research about AM/FM modulation).
Made an antenna for the node (a new one for when it will be inside its box).

18/04
Continuation of yesterday's work.
Team meeting to address work issues (pace, goals, work distribution) and redraw a roadmap accordingly.

19/04
Continuing proofreading and editing the main report.
Continuing measuring node's power consumption (with/without shield, with/without sensors, etc).
Managed a first reading of the database from a website.

20/04
Designing website.
Continuing measuring Node power consumption.
Continuing research about LoRa.
Draw the current measurement circuit.
Checked if we could actually save power using a transistor to cut off current in the sensors circuit : even off, the transistor let enough current pass through to get a reading from the sensors. Probably useless to use a transistor, unless we find a more sensitive one (even then, power consumption is so low it could anyway be irrelevant to use one to save power).
Rewatch of "decoding LoRa" video. Way better understanding.

Week 17

23/04
First drafts for the Expo catalog (deadline thursday).
Researches about convolutional encoders, and other form of encryption, tried to find examples.
Managed to get the last data on the website (from the database) + refresh every 30s.
Managed to create login/password on the website.
Created a executable file launching 3 programs at once.
Configured the server's firewall to authorize the Raspberry (IP+port).

24/04
Improving the Expo catalog text. Discussing about the 2 pictures.
Began coding in C the convolutional encoder.
Continuing Arduino power consumption measurements.
Renamed the .csv. It doesn't change name anymore.
Checked the whole chain (arduino to … to database) again.
Improved website security (all pages need the login/PW).

25/04
Continuing coding the convolutional encoder : got 3 convoluted bits for 1 bit input.
Took pictures for the 1st picture for the Expo catalog.
Continuing writing the report.
Arduino tutorial.
Review of the Arduino code.
Trying to get the sensors working through the shield, to the Arduino.

26/04
Tried to find out why the Arduino pin through the shield loses voltage.

Tried to find another temp sensor, which would not be plugged through the I2C shield protocol, to bypass the previous problem.
Continuing coding the convolutional encoder : got to decode and correct 4 groups of 3 bits.
Worked on the second picture for the Expo catalog.

27/04
Resolved and understood why the I2C of the shield and the Arduino weren't compatible.
Made a small PCB with the thermal sensor and with the light sensor.
Researches about the error correction algorithm in the convolutional encoder.

Week 18

30/04
(no Brede)
Continuing convolutional encoder (Viterbi algorithm)
Finishing tutorial for Arduino
Improving Arduino code
Refining website design

31/04
Finished the Raspberry tutorial.
Commented all the code for the power consumption circuit program.
Continuing writing about the power consumption circuit on the report.
Continued power consumption researches and how to lower it.
Continuing convolutional encoder (Viterbi algorithm).

01/05
Trying to code the viterbi algorithm.
Writing on the report.
Research how to reduce power consumption.
Continuing devices tutorials.
Continuing refining code.

02/05
Trying to code the viterbi algorithm.
Writing on the report.
Continuing devices tutorials.
Continuing refining code.

03/05
(Antoine & Guillaume day off with visiting friend)
Proof-reading, writing and adding pictures to the report.

04/05
(Antoine & Guillaume day off with visiting friend)
Proof-reading, writing and adding pictures to the report.

Week 19

07/05
Writing report.
Convolutional encoder continued.

Researches about the node's box.
Creating the box which receive the arduino card and the shield for the arduino and also all the sensors.

08/05
Writing report.
Tried to finish creating the box which receive the arduino card and the shield for the arduino and also all the sensors.
Convolutional encoder continued.
Received the thingy to detect and recreate radio waves.

09/05
Managed to receive LoRa signal on the thingy.
Convolutional encoder continued.
Printed the first element of the node box.
Writing report.

10/05 (Brede cuve sa vinasse)
Convolutional encoder continued (debugging the decoder).
Box work.
Writing report.

11/05
Convolutional encoder debugged.
Printed a second piece for the Arduino box. The 3D printer needs some adjustments.
Writing report.

Week 20

14/05
Writing report.
Proofreading tutorials.
Changed PHPMyAdmin for Xampp on the server (newer, updated and more stable).
Loaded the website on the server.

15/05
Proofreading tutorials.
Writing report and tutorials.
Thinking of how to print correctly the box lid.

16/05
Proofreading tutorials.
Writing report and tutorials.
Designing new pieces for the node box.

17/05
Day off: Norway's national day.

18/05
Proofreading report & tutorials.
Writing report.
Setting up the whole chain of devices for incoming full-scale tests (outside).
Debating unfinished details and creating a todo-list about them.

Week 21

21/05
Proofreading report & tutorials.
Writing report.
Python coding to get the data to the database.

22/05
Proofreading report & tutorials.
Writing report.
Node configuration tests.

23/05
Proofreading report.
Writing report.
Bug chasing in the chains code.
Printing pieces, then adjusting them.
Removing cut pins from the Arduinos.

24/05
Proofreading report.
Writing report.
Finding out the problem in the chain: it’s probably the SD card on the RP3 dying.

25/05
Proofreading report.
Writing report.
Field test: connection between the Raspberry and the node in the streets of Fredrikstad.

Week 22

28/05
(Lunch w/ Hong)
Writing international blog.
Writing report.

29/05
Writing report.
Proofreading report, tutorials.
Reinstalling Raspbian on a new SD card (Raspberry pi).
Searching a solution to the mystery .csv bug

30/05
Writing report.
Proofreading report.
Searching a solution to the mystery .csv bug.

31/05
Meeting with Reidar about the report.
Writing report.
Proofreading report.
Finding solutions to the mystery .csv bug. Debugging the Python codes writing/reading in the .csv file.

01/06
Writing report.
Proofreading report.
Testing the new found solution for the .csv file.

Week 23

04/06
Outdoor field tests with the node.
Set the chain of communication to use WIFI instead of ethernet, for the Expo.
Finished writing the report and the tutorials in a rush to meet unforeseen french deadlines.