

# **FLOATING WATER QUALITY STATION PROJECT 10 - INFO 4**



Cette photo par Auteur inconnu est soumise à la licence CC BY-NC

Supervising teachers :  
Mr. DONSEZ - Mr. PALIX

## TABLE OF CONTENTS

INTRODUCTION.....	3
▪ Context : .....	3
▪ Objective : .....	3
ORGANIZATION OF THE PROJECT .....	4
ANALYSIS OF THE EXISTING .....	6
▪ Type of measurement and sensors used : .....	6
TOOLS USED.....	7
▪ Equipment available : .....	7
▪ Technologies : .....	7
▪ Connecting the board : .....	8
DESIGN.....	9
PROGRAMMING .....	10
▪ Sending LoRa packets continuously.....	10
▪ Temperature sensor reading.....	11
▪ Reading of other analog sensors .....	12
▪ Sending and processing data from TTN/CampusIoT to MyDevices .....	12
TESTS .....	14
▪ Temperature sensor test.....	14
▪ pH sensor test.....	15
▪ Turbidity sensor test.....	15
▪ Review .....	17
ADVANCEMENT .....	18
CONCLUSION.....	19

# INTRODUCTION

---

- Context :

Nowadays, due to pollution, the problems related to maintaining good water quality in large areas have greatly increased. With this new difficulty, it becomes necessary to be able to measure the various parameters related to water quality. Several questions may then arise: What is necessary to measure to ensure good water quality? How to set up the sensors so that they continuously transmit the required results?

- Objective :

To answer these questions, it is necessary to look at the way the project works as a whole. Indeed, we have at our disposal sensors, a LoRa-E5 mini card allowing us to send data via LoRa packets, as well as an application to format the results. In order to set up a water quality treatment station, it is necessary to create a complete physical solution that will connect all the sensors to the LoRa-E5 card. These sensors, immersed in the water, will retrieve values that will be recovered by the card. Then, the card will apply calculations to transform these analog data into digital data. The data will then be sent, thanks to LoRaWan technology, to the shaping application via an antenna specially used for LoRa packets. It will then be possible to process the data provided and display the information needed to verify the quality of the water, based on the various measurements made.

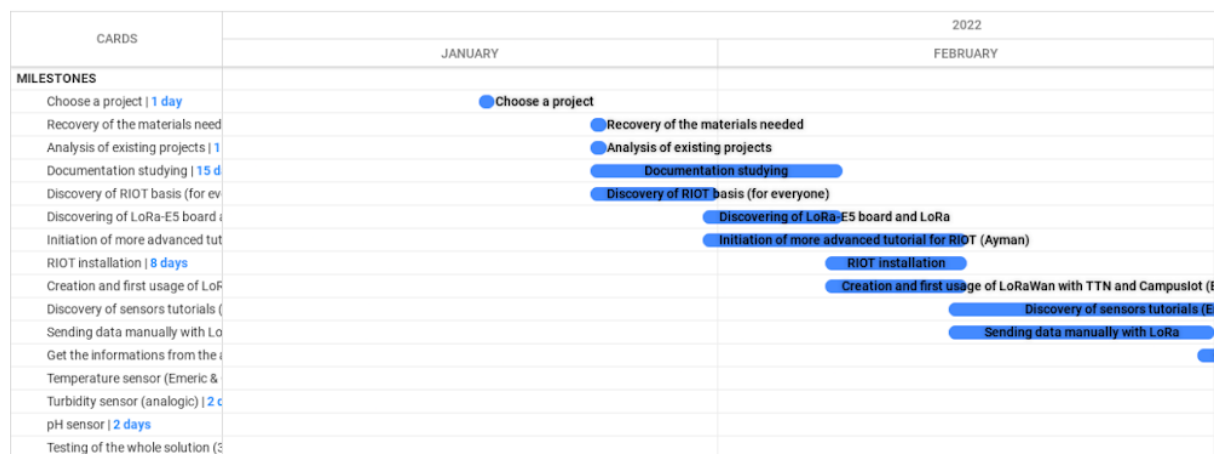
# ORGANIZATION OF THE PROJECT

As the project is wide and there are many steps to be done, we decided to divide the tasks and schedule them according to their priority, but also according to the prerequisites. We each started by analyzing the existing situation, looking for similar projects in the use of sensors for water quality treatment. Then we searched and trained on technologies and tools such as Riot OS and LoRaWan.

Regarding Riot OS, we performed multiple tutorials on GitHub. After we managed to use the basics of the OS, we started to really split the tasks. We then distinguished three main tasks, respectively the more advanced use of Riot OS (done by Ayman) using new tutorials in order to better understand the use of the technology in our context, the discovery of the sending of LoRa packets to the dedicated antenna (which was done by Camille) and finally the use of temperature and turbidity sensors (done by Emeric), via tutorials provided and code examples usable via an Arduino board.

The GANTT diagram of our project, used to show the scheduling of the tasks we have done, is the following:

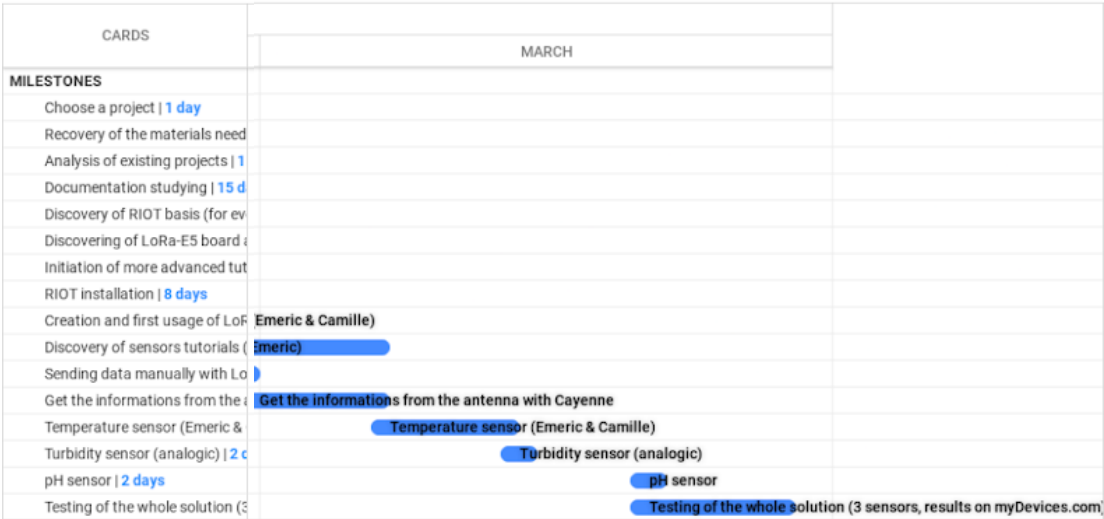
## FloatingWater



This document is created with GanttProject - https://ganttproject.com

Figure 1 - Diagramme de Gantt

FloatingWater



This document is created with Ahtahle library

Figure 2 - Suite du Diagramme de Gantt

# ANALYSIS OF THE EXISTING

In order to get more inspiration and to start our project, we first inventoried several projects related to water quality measurement in lakes, rivers or ponds to determine the types of measurements performed, the sensors used as well as the main technologies integrated in these projects.

Examples of projects related to ours:

- IOT solution for water management.
- Smart Water solutions for better quality mussels and salmon in New Zealand.
- Libelium takes care of river health with water IoT technology.
- RC IOT boat monitoring water pollution.

## ▪ Type of measurement and sensors used :

Type of measurement	Sensors used
Identification of pollution's causes	HD camera
pH, turbidity, dissolved oxygen sensor	pH sensor and turbidity sensor
GPS location	a GPS system
Bacteria colony count	Colour strips
Oxidation Reduction Potential, Conductivity, Humidity, Pressure	Different sensors with the same name
Turbidity and Temperature	NTU sensors
Smart Water Ions measurement	
General monitoring (temperature + pH....)	Smart Water Xtreme Sensors

# TOOLS USED

---

Once it was done, we needed to adapt the material and technologies of these projects with what we had at our disposal in the Laboratory (FabLab) and the technological solutions that fit the project like LORA-E5 and Riot.

- Equipment available :

- pH-meter
- Turbidity sensor
- Temperature sensor
- Lora-e5-mini board
- Nucléo STM32 board

- Technologies :

## **RIOT OS :**

An OS that provides the main classes for detecting sensors and sending LoRa packets via the card. Allows the compilation of the project.



## **STM32 Cube Programmer :**

An IDE for connecting and flashing the board (with the help of the STM32 Nucléo board).



## **LoRa/LoRaWan:**

Allows sending packets from the board to a LoRa antenna.



## **CampusIoT / TTN (The Things Network) :**

Allows you to enter the identifiers of a card, and to retrieve LoRa packets sent to an antenna from this card.



## **Cayenne:**

Sends data from CampusIoT/TTN to MyDevices and transforms the resulting data into something interpretable.



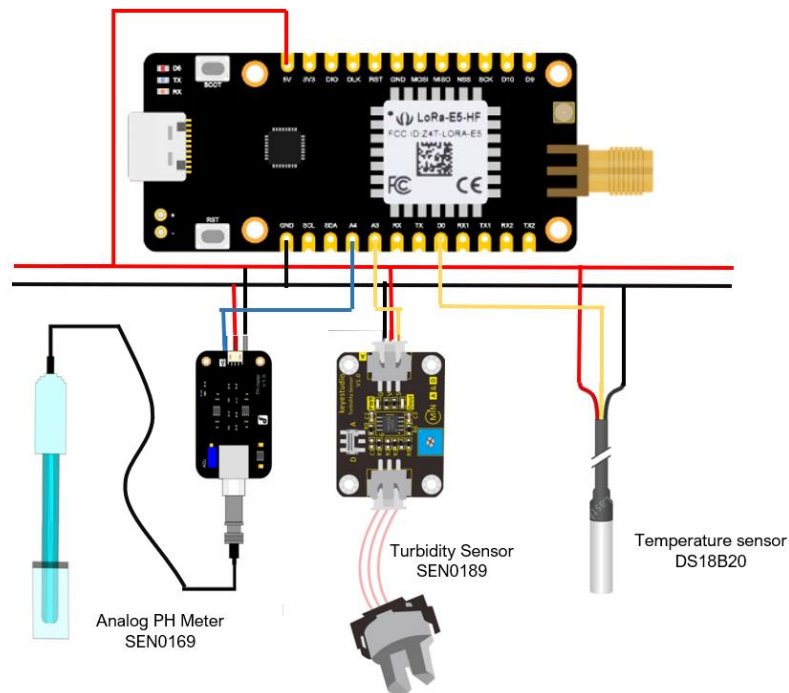
## **MyDevices:**

Application to display the different results.



- Connecting the board :

Our LoRa-E5 card will be connected to the different sensors necessary for the functioning of our global solution. We will then keep the following connection :



*Figure 3 - Connection diagram of the different sensors to the Lora E5 mini board*

The turbidity sensor will be connected to pin D0, and the turbidity and pH sensors to pins A3 and A4 respectively.



# DESIGN

To explain the interactions between the different elements of our solution, it is important to have a clear view on the existing links, and the order of the actions performed :

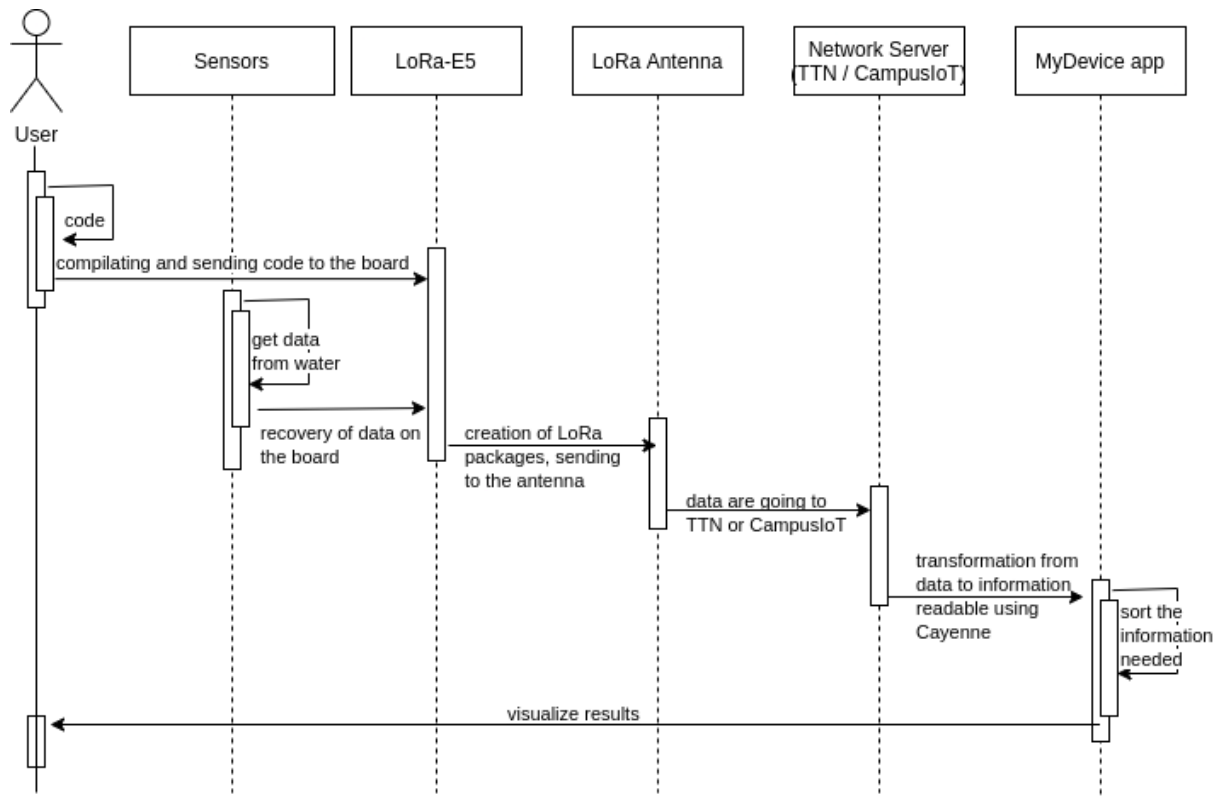


Figure 4 - Sequence diagram of the application, in the case of the connection to the Antenna succeed

Thus, we see that the user must code classes to read the data provided by the sensor, and to send the LoRa packets continuously. When this code is created, it is compiled via Riot OS and flashed onto the LoRa-E5 card. Then, sensors placed in the water will be able to retrieve data, which will be sent to the card in the same way. These received values, combined with the existing code, will create LoRa packets that will be sent to the antenna and then received by TTN or CampusIoT. Finally, it is possible thanks to Cayenne to process the data obtained on one of the network servers, in order to send it directly to a formatting application such as MyDevices, which will then sort the necessary information. The user can then modify the values he wants to display on his results application as he wishes (in our case, he can display temperature, turbidity and pH).

# PROGRAMMING

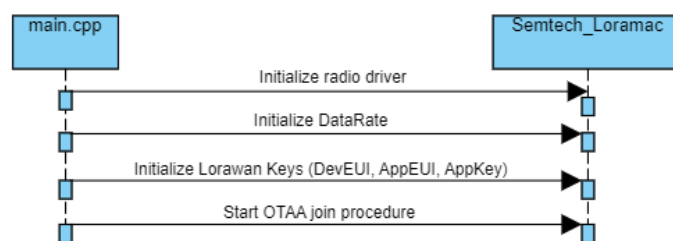
First we followed different tutorials to better understand RIOT and LoraWan. We then started to go deeper into the project, trying to implement the first step of our water quality treatment process. We defined, in agreement with each other, that this first step consisted of sending LoRa packets to an antenna, and receiving this packet with TTN or CampusIoT, as desired. We used both tools in order to have a wider variety of antennas, so that we could use our programme everywhere (knowing that in some places there are only antennas for one of the two tools).

- Sending LoRa packets continuously

Let's take TTN as an example. In order for the messages sent by the card to be received by the tool, it needs to be provided with certain information in order to initialise a connection such as the DevEui, AppEui and AppKey. This data can be found with the basic program by typing simple commands such as "AT+ID=DevEui" to get the DevEui.

Once this information is built into TTN, we can start writing the code to send LoRa packets. RIOT provides a public API such as Semtech LoRaMAC to use Lora and various useful drivers such as the sx126x driver to use the antenna. So this part consists in using the elements at our disposal.

As we can see in the sequence diagram below, the part prior to sending the data in LoraWan consists of a sum of several initializations:



*Figure 5 - Sequence diagram of the initialisation for sending packets via Lora*

We then try to launch the OTAA (Over-The-Air Activation) procedure to establish the connection between the card and the antenna. This launch will return an error if the DevEui, AppEui or AppKey do not match those of TTN (because the connection cannot be established), if the card has tried to send too many messages (duty cycle restriction, so you must wait before trying to connect again), or if the card is already connected. If the launch is successful, the sending of packets is launched.

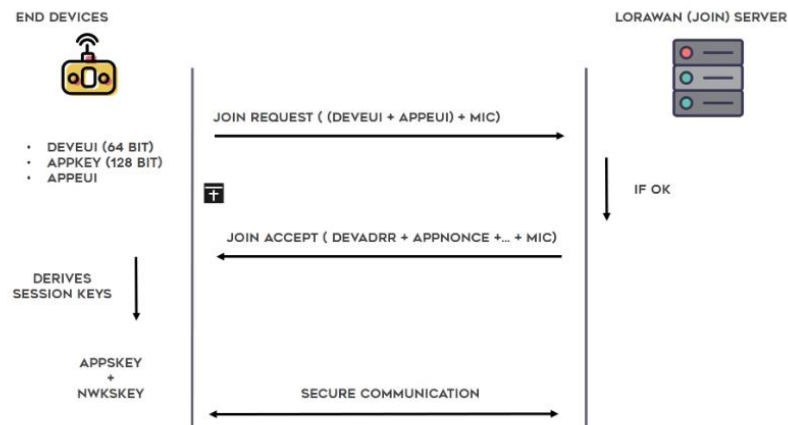


Figure 6 - OTAA procedure

Extract from : <https://riot-os.github.io/riot-course/slides/05-lorawan-with-riot/#13>

In the sending loop, all you have to do is send, via the loramac connection created previously, the data you want to pass and the length of the data. This created message will be sent to the antenna and retrieved via TTN or CampusIoT. To avoid an overload of messages, a timer can be added to make the execution "sleep" for a given time.

Finally, in the Makefile, we must add the board we are using, the access path to RIOT (in order to be able to retrieve the different OS modules), but also the modules and packages called in our main program. We also call the Makefile at the base of the RIOT folder, which will compile the whole RIOT project before compiling our project.

## ■ Temperature sensor reading

As the card is now able to send continuous messages, it is necessary that these messages contain real data, usable in our project. Thus, we need to use sensors, especially the temperature sensor. RIOT OS is able to read data from some sensors thanks to modules contained in the "drivers" folder, especially the one corresponding to our temperature sensor.

To be able to read the values from our card, we add the "ds18" module in the Makefile and in our program. Thus, we can create a global variable which will represent our temperature sensor, and which will benefit from all the functions already implemented by the module.

In our program, we initialize the sensor with the corresponding parameters (the parameter class for ds18 also exists in RIOT, so we only need to include it in our project to use it). If the sensor is badly initialized, an error message will be returned, otherwise the initialization of the LoRa card will be done as seen previously.

In the execution loop, we will retrieve at each loop the temperature value, which we will display and store in a "temperature" variable. An error message can be returned if no value

is found (if the sensor is not connected for example). The value is then sent to TTN/CampusIoT.

## ▪ Reading of other analog sensors

Other sensors exist, and even if they are not already implemented in RIOT OS, they can be read. As our LoRa-E5 board has two analogue inputs, we are able to connect two more sensors for which the value will be obtained analogue and then converted according to a formula specific to the data reading of this information.

In RIOT there is a feature, which allow you to read the analogue values sent to a given analogue input. This feature is "adc", contained in the drivers folder (part of the peripherals). From "adc" it is possible to create new classes that can read our turbidity and pH sensors.

Thus, we have created two classes with their corresponding headers, each of which contains a sensor initialization function, in order to give the corresponding analog port, which is already stored in a list "ADC\_LINE" in the "adc" feature. We just have to take the right value for each of our sensors. We also declare a function to read the data, where we retrieve the value on the analog port assigned to our sensor, at a specific time, and then we check its veracity. The value obtained is between 0 and 4096, so it must be converted by a cross product to obtain the corresponding voltage. If the value does not exist (sensor not connected), an error is displayed, otherwise we use calculation formulas to obtain the turbidity or pH value corresponding to the given voltage.

These formulas, found in the documentation of our two sensors are:

"pH = 14 - (3.5 \* voltage) + Offset;" where voltage is the value of our voltage and Offset equals 1.39 (for pH)

"turbidity = -1120.4 \* pow(value, 2.0) + 5742.3 \* value - 4352.9;" where value is our voltage value (for turbidity)

After adding these classes to the Makefile and to our main program, we are able to initialize and read their values as explained earlier with the temperature sensor (we just need to call the functions created in the ph and turbidity classes), and then send the values to TTN/CampusIoT.

## ▪ Sending and processing data from TTN/CampusIoT to MyDevices

To finish processing the solution, we want to be able to send the retrieved data to a processing application such as MyDevice, which we have chosen here for its ease of implementation with TTN and CampusIoT, since it is possible to process data and send it directly with CayenneLPP, which is a library capable of creating LoRaWan packets for processing applications like MyDevice.

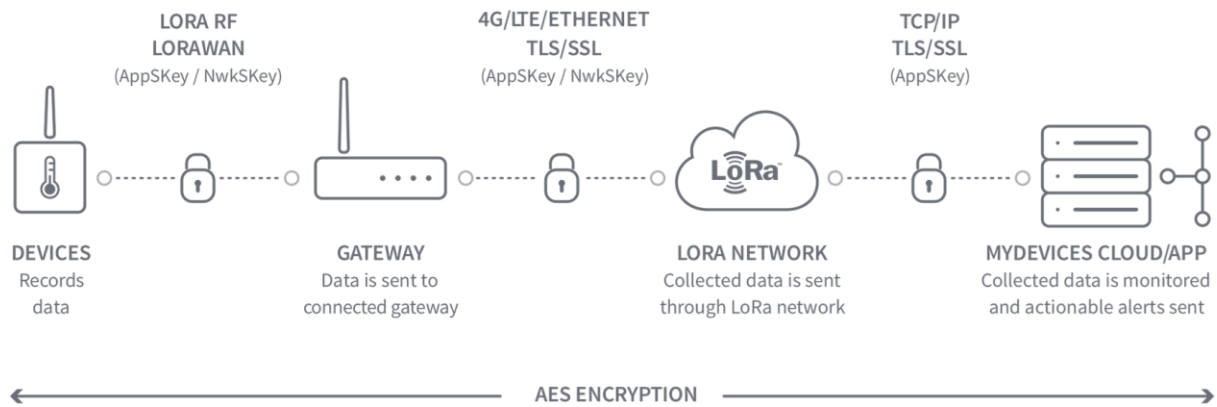


Figure 7- Diagram showing the transmission of data via the Lora network

To implement this in our program, we once again use the existing module in RIOT, which is called "cayenne\_lpp". This module implements a number of methods, including methods for adding temperature values, or analogue values. Thus, it is possible to retrieve the values obtained by the sensors, and to assign them directly to a specific parameter of the CayenneLPP packets that will be sent afterwards, which will make it easy to recognize which value corresponds to which sensor in MyDevice.

When the sensor values are added to the cayenne\_lpp object, the sending data of the LoRa packet can then be replaced by the data contained in Cayenne. In TTN, one will be able to observe the results, and on MyDevice, it will then be possible to create a new data reception page (which will be directly linked to the sending card by giving it the values DevEui, AppEui and AppKey as parameters), and to modify it as one pleases so as to display only the desired values. It is possible to display GPS coordinates or many other things, but we can also display only the values of the three sensors. MyDevice offers some freedom in the formatting, since it is possible to display the results in numerical, graphical or other forms.

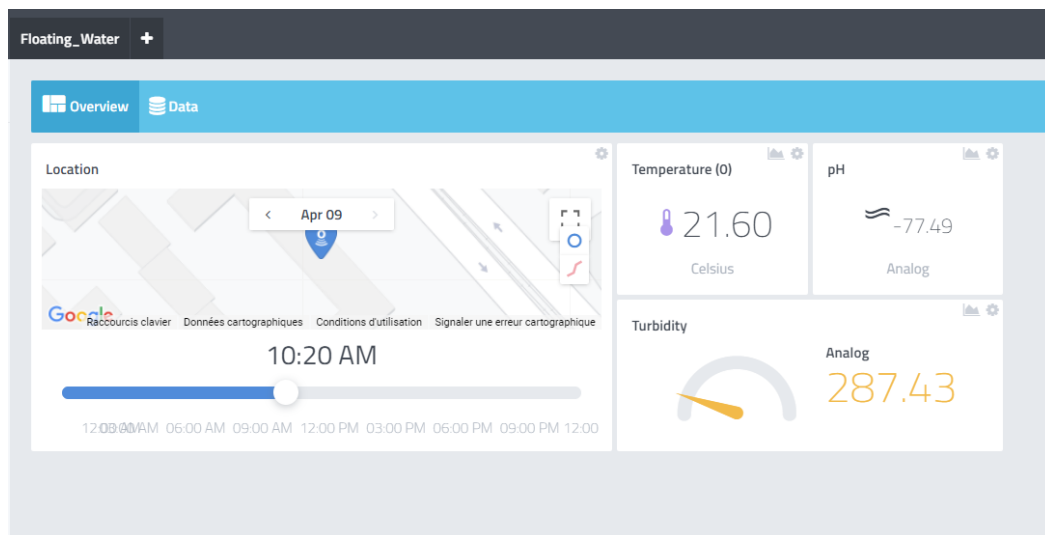


Figure 8 -Screenshot of the MyDevices interface with data from the sensors.

# TESTS

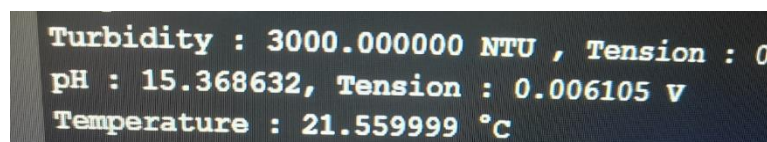
The aim of these tests is to check that all the sensors are functioning as expected. To do this, we sought to make measurements (displayed in a serial monitor) from known values.

## ■ Temperature sensor test

In order to test the temperature sensor, we used a thermometer to compare the two measurements.



*Figure 9 - Photo of the ambient temperature obtained with a thermometer*



*Figure 10 - Photo of the serial monitor with the temperature obtained with the sensor.*

As can be seen, the values are very close, so it can be concluded that the sensor works as expected. In addition, several other tests were carried out in different environments to verify the relevance of the data :

Type of test	Method	Measurement (approximately)
Wet environment	Measurement in a glass of cold water	11.2 °C
Cold environment	Measurement in a freezer	- 13 °C
Warm environment	Measurement with sensor clamped in one hand	30 °C

The data obtained is relevant, so we can conclude that the temperature sensor resists different environments over a wide range of temperatures. Moreover, it also seems to be quite accurate  $\pm 0.2^{\circ}\text{C}$  compared to the thermometer purchased in the shops (whose accuracy is unknown).

## ▪ pH sensor test

As we do not have a pH meter or pH measurement strips, we decided to test pH measurements from common products. After searching the internet, we were able to find theoretical pH values for these products. This test does not allow us to say whether the pH values obtained are accurate, but it does give us an idea of whether or not the product is working properly.

Product	Expected (approximately) pH	Measured pH
White vinegar	2.4	2.51
Coffee	5	4.99
Coca Cola	2.5	2.59
Grenoble Water	7.5	7.67
Cif crème	11	9.87 (diluted in water)

From the values obtained, it can be said that the test is validated and that the pH sensor is functioning correctly.

## ▪ Turbidity sensor test

In order to test the turbidity sensor, it would have been best to perform turbidity standards with formazin. In the absence of such a device, we decided to perform the test with milk diluted in water at different concentrations. It is not clear what values to expect, but it is known that the turbidity of clear water is very low and that turbidity increases as the water becomes cloudy.






*Figure 11 - Glasses with different milk concentrations - side view*





Figure 12 - Glasses with different milk concentrations - top view

By successively dipping the sensor in the different glasses, we obtain the following results:




Picture	Turbidity Measure	Voltage of the sensor
	0 NTU	4.21 V
	0 NTU	4.21 V
	0 NTU	4.21 V



	3 000 NTU	1.9 V
	3 000 NTU	0.02 V

It is clear that the values are not completely consistent. In the third image, the water is clearly turbid, although still slightly transparent, but the turbidity value remains at 0 NTU. On the last glasses, it would appear that the equation linking turbidity and voltage is not completely correct. In conclusion, the tests with the turbidity sensor are not validated.

## ■ Review

Temperature sensor test :   
pH sensor test :   
Turbidity sensor test : 

# ADVANCEMENT

---

We have managed to implement the main functionalities needed for water treatment, namely reading the data from each of the sensors provided and displaying them in a data editing application. However, the overall solution could be even more complete, and we have several points of improvement that would allow for a more efficient project for water quality treatment:

- The addition of new sensors, such as those measuring conductivity, suspended solids, the amount of chlorine in the water..... The main problem here is the lack of analogue or digital ports, which limits the number of sensors that can be used at the same time.
- Using MQTT to send the data to a mobile application, so that not only do we have the MyDevices display, but also a portable version of these results.
- Building a raft to test our solution over a larger area. Indeed, this project initially aims to measure the water quality of ponds or lakes, so it is necessary to have a support to carry our map and sensors on the water. In addition, it might be useful to remotely control the raft, so that it can test particular locations.
- More extensive testing of the sensors, as it would be preferable to have accurate and reliable tests to ensure that the sensors are working properly.
- When sending data to Lora, it would be important to create a function to avoid duty cycle restrictions.

# CONCLUSION

---

During this project, we had the opportunity to work on many different aspects, both in the development of the program and in the construction of the physical solution.

We discovered RIOT OS as well as LoRaWan technology, which required some time to adapt to their use. These are particular tools that were extremely interesting to use, and discovering recent tools meeting a specific need gives an additional interest to this work.

We would have liked to have been able to go further in the development of our solution by doing the different points explained above, but we still achieved the objectives we had set ourselves to have a functional water quality treatment giving the expected results.