Social Network for Pollution

T.Chen, A.Alami, C.Bravo, D.Laille, T.Teiti

January 19, 2019

Contents

Abstract

Nowadays, air pollution has become a common phenomenon in big cities. Gases and particles can be dangerous for people's health, especially for pregnant women or young people. Moreover, it impacts our outdoor exercises too. The university of Wollongong, based in Australia, has proposed a project about the air pollution monitoring. We created the connected air quality sensor assembled in a package designed by them. Our goal is to detect the air quality and develop a website to inform people about the air pollution in real-time. And we need to use open-source resources because we want to develop an accessible product. We can already detect the 2.5 µm, 10 µm, NO_2, CO, NH_3, temperature, humidity and pressure by our fixed sensors, send the data to The Things Network (TTN) and retrieve the data to show them in a map by website. We use the LoRa technology to make the communication between the sensors and the user interface which is not expensive and can transmit in long distance. And we use the Angular Framework for the website to make the user interface friendly. In the future, we will work on the mobile sensors and deploy our air quality monitor in Occitanie and in France.

We would like to thank Thierry Monteil, Nicolas Verstaevel and Hugh Forehead, our tutors for this project. We thank Joseph Shea, our english teacher, and Sebastian Di Mercurio, for helping us on our project. Lastly, we want to thank INSA which provide us work rooms and materials for our project.

1 Introduction

Nowadays, big cities have various nuisances which impacts the health and the comfort of citizen. The University of Wollongong, based in Australia, has proposed a project about designing an embedded sensor platform to be deployed by non-scientific people.

The objective of this platform is to collect data about the air quality in a big city, analyze these data and display them on a user interface such as a website.

This project is the main project for the Innovative Smart System training as it combines electronic and computer science skills. As the Internet of Things (IoT) is growing exponentially, we think that doing the project about IoT is essential and crucial for a student studying the Innovative Smart System. And this project will improve our skills and knowledge in the IoT domain.

2 Context and Objectives

It is reported that 50% of the population live in cities. The air pollution is really serious in big cities. People with bad health and pregnant women care more about the pollution because it impacts on their health. If we want to run outside, we need to know about the air pollution today. So being informed about the air pollution is really important for everyone and we need to know it in real-time.

We want to do this project to make people aware about the air pollution and more active in measuring it by providing them an easy access to data.It is also important for users to have a product not so expensive. The objective of our project is to detect the air quality and to show it graphically to people by the website. Also, our website can be accessed freely.

We have divided our project into two parts: a hardware part and a software part. The hardware part works mainly on detecting the 2.5 µm, 10 µm, NO_2, CO, NH_3, temperature, humidity and pressure in the air by sensors and on sending the detected data to The Things Network using a LoPy microprocessor. While the software part works mainly on retrieving the data from The Things Network for data analysis and showing the results through a map on the user interface.

The first targeted users are the people in INSA Toulouse, such as students, teachers, members of staff or anyone who wants to know about the air quality in INSA Toulouse in real-time. We test our prototype in INSA firstly, so it is easiest for the people in INSA to use it. After that, we will test our project in Toulouse, for example the Capitole, where we can compare with INSA to see if the air pollution is heaviest in the downtown.

We also have a long-term perspective for our project where we would use the mobile sensors in INSA and University of Paul Sabatier, because during our test we always use the fixed sensors. We put the fixed sensors on bikes, cars and buses which will move around the INSA and University of Paul Sabatier regularly. Moreover, we want to deploy our air quality monitor in Occitanie and France to gain more market if we can.

3 Need Analysis and Competition

1. **Deploying a platform of air quality for non-scientific people**
   There are already some platforms of air quality, but most of them are made for scientific people because there are much technical information, maybe we couldn't understand them. So we want to create a air quality monitor for non-scientific people which will give the clear information for people and people can get the air quality easily. We need to develop this platform to make people informed about air quality when they enter in our website, especially for non-scientific people.

2. **Low cost**
   Many resources and platforms are not open-source, even if we can use them freely at the beginning but after that, we need to pay for it if we want to continue to use the resource. For our project, we want to make cost as low as possible. Thus, we need to choose all the open-source for developing our air monitor project, such as The Things Network.

3. **Open source**
   Making our resources open-source is a step forward democratization. While a lot of people do not trust governmental or industrial data, an independent and easy-to-use air quality monitor should be a tool to build a citizen strength and awareness about climate change. It can modify behaviours and also help to adjust policies and laws in this field. Today, it already exists some associations working on this topic, like ATMO, which gather a lot of collaborators to monitor air quality in France, or the world air quality index, provided by a group of people all around the world, and which gives an indicator about level of pollution.

4. **Low energy consumption**
   We want to make low energy consumption for the communication between the sensors device and the user interface. For that, we need to choose a

technology which has low consumption and can transmit data in long distance, such as LoRa network.

5. **Easy to use**
For people to use it easily, we need to create a website which it can be accessible to anyone at anytime. After that, the website should inform the air quality information clearly, user can choose whatever the city and then the website will show the air quality at real-time and give some suggestions to user.

4 Design and Specification

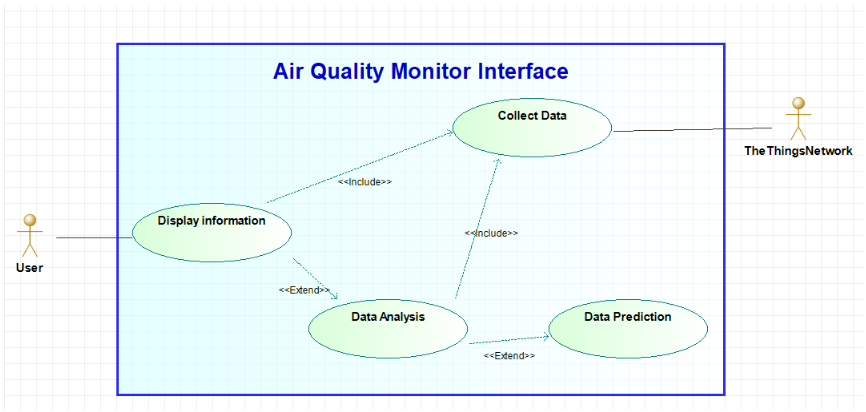


Figure 1: Use Case Diagram on the user interface

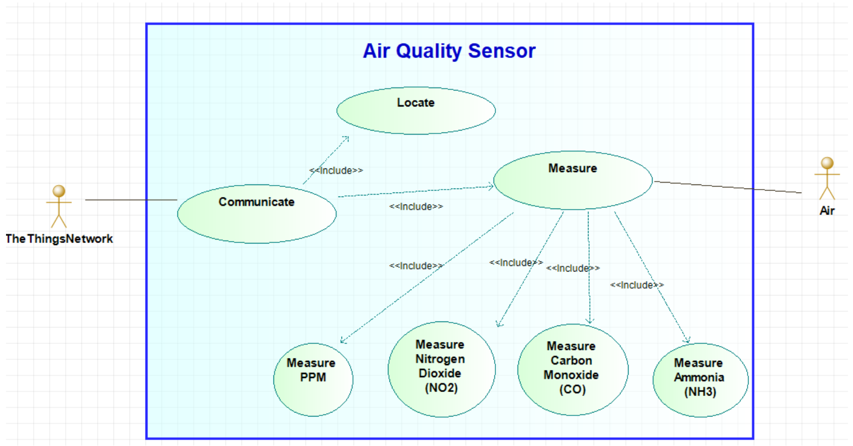


Figure 2: Use Case Diagram on air quality sensor

4.1 Hardware

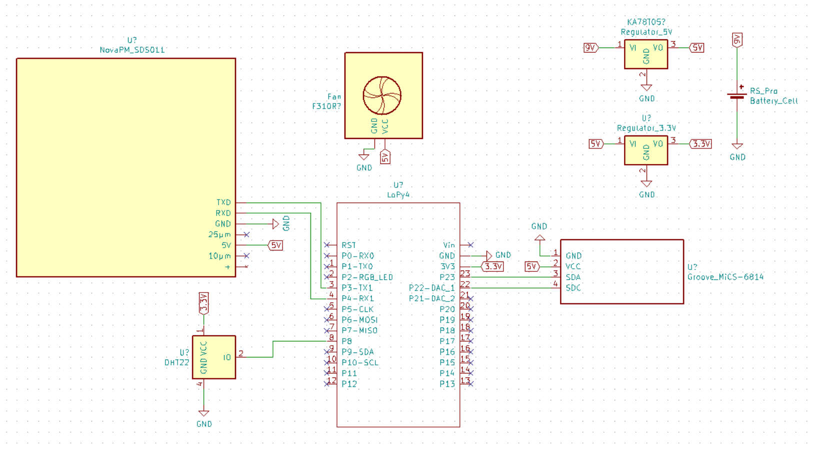


Figure 3: Overview of the connections

4.2 Software

The software part of this project intends to provide a user interface easily accessible for citizens in order to visualize efficiently the level of the air quality of their city. Data transmitted by our air quality sensors have to be available whenever. The software specifications are focused on the recovery of data from the sensors, the storage of these data and the visualization on a user interface.

4.2.1 Mock-up

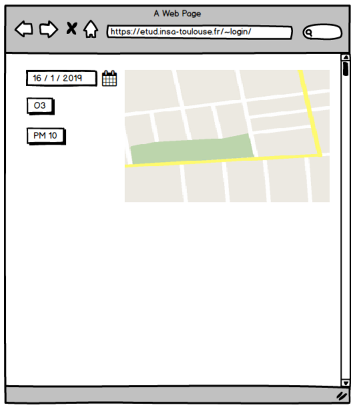
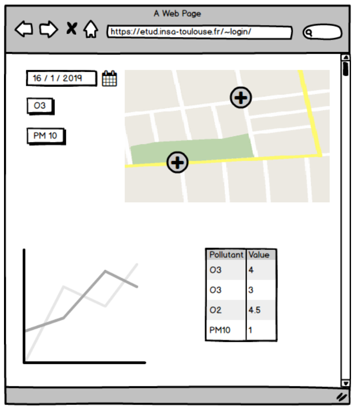


Figure 4: Mock-up of the website

This picture is user interface when a user enter our website. Actually, we need to input the url of our website, for example here the url is the service host of INSA. When we access the website, we can see there are a date chooser, two buttons for the type of pollutant and a map.



Pollutant	Value
O3	4
O3	3
O2	4.5
PM10	1

Figure 5: Mock-up of the website after clicking the city

We can choose the date which date we want and click in the map which place we want. After that, we can see a line chart and a table. While, the chart shows the value of pollutant since some months and the table has all the data of the pollutant in this place.

4.2.2 Architecture

Our architecture is based on the following components:

1. **the sensors**: our device measuring the air quality
2. **the gateways**: installation relaying data transmitted by near devices
3. **the cloud platform**: it retrieves data from gateways and provide them for client applications
4. **the server**: his role is to retrieve data from the cloud and to store them into a database
5. **the database**: it stores data and make them available for the server when needed
6. **the front-end application**: it manage the layout of the web page to display to the user

The figure 6 presents the interactions between the different components of the architecture and how the data is communicated. Also, we made choices about the technology used for each component that will be presented in the following part.

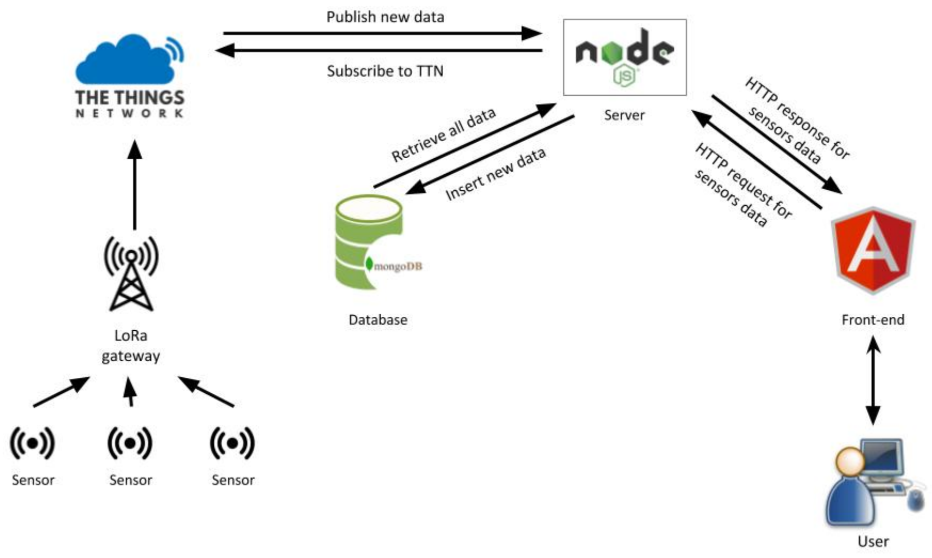


Figure 6: Overview of the global architecture

5 Technical Environment

After defining the specifications of the project, we made choices on the hardware and the software parts about the sensors, the data communication, the data storage and the web development. Several technologies are designed for IoT applications, so we focus on implementing a solution which is low cost, open, low power consumption and easy to deploy.

5.1 Sensors

To measure pollution, it is interesting to have several sensors, to be able to have an overview of what impacts our health, in terms of atmospheric danger.

The most important is to measure the concentration of particulate matter in the air. To do that, we use a NovaPM sensor, which is able to sense 10 µm and 2.5 µm particles. With that sensor, we can detect smog, dust and other particles which can cause a problem. It is not the smallest one but it is one of the most accurate.

Then, it comes the measurement of some gases: carbon monoxide (CO), nitrogen dioxide (NO_2), ammonia (NH_3). To achieve this goal, we chose to use the MiCS-6814 sensor, coupled with a groove microelectronic board. The assembly of them form a device called multi-channel gas sensor.

Because the pollution of the air is also linked with temperature and pressure, we selected the BMP280 sensor. It is a small sensor integrated in a adafruit (designed by Bosch).

To add the possibility of mobility for the whole device, we also programmed a GPS, in order to be able to locate it everywhere. We do not consider to use the air quality monitor for a continuous mobility, because gas sensors needs time to take a valid measure. So the compromise is to have a semi-mobile device, that is to say a compact device, autonomous, and that we can move easily.

5.2 Micro-controllers

To make the calculations, collect and transmit data through wireless networks, we use a micro-controller. We decided to choose a compact one, which integrate wireless communication like WiFi, Bluetooth, SigFox and LoRa. This micro-controller, LoPy 4.0, is coupled with an extension board, and works under 3.3V. So it needs a quite low power supply. On the other hands, it is problematic because some gas sensors are not so efficient under 3.3V, they better function under 5V. The other drawback of this micro-controller is that programming is done under micropython, and there is not a large community in this language. The main consequence is that there is not so many libraries, so it takes more time to develop and to integrate new sensors.

5.3 Data communication

In order to transmit data from our sensor to the web, we chose to use the LoRa network, a popular digital wireless data communication technology largely used in the field of IoT. This technology was designed to allow data transmissions on a long range and with low power consumption. Also, the infrastructure is well deployed thanks to a large community installing public LoRa gateways.

"The Things Network" (TTN) is an open and collaborative IoT network using the LoRa technology, and provides tools to build IoT applications. It is possible to use the public network through public gateways already installed, so we do not need to deploy our own gateways to transmit data from our sensors. TTN provides "Node.js Application SDK", a module in Node.js using MQTT (Message Query Telemetry Transport) protocol based on publish-subscribe principle. In this way, when a new data is transmitted, TTN will publish this data to all subscribed applications.

5.4 Data storage

Data transmitted by sensors need to be stored in a database in order to retrieve them when needed and to make data analysis. Data sent from TTN are formatted in JSON (JavaScript Onject Notation) which is a format easy to manipulate in Javascript. NoSQL databases are suitable for IoT thanks to their flexibility and their scalability. Thus, they represent data in JSON format, which is appropriate with data sent from TTN. In this way, we chose to use MongoDB, a popular NoSQL cross-platform document-oriented database and Node.js libraries exist to interact with this database.

5.5 User Interface

5.5.1 Web interface

1. **Angular**
   Angular is an open source client-side framework. It uses the TypeScript programming language (TS), also open source and developed by Microsoft. It is based on a Model-View-Controller (MVC) architecture:

   (a) The Controller, which corresponds to the logic of the application concerning the actions performed by the user, is coded in TS calling web services to the server.

   (b) The View contains the presentation of the GUI and is coded in HTML / CSS. When a user performs an action, the Controller automatically updates the View.

   (c) Finally, the Models contain the data to be displayed in TS. They are updated by the Controller whom they notify in their turn.

   We use the latest version at the launch of our project, version 6.

   One of the main advantages of Angular is its modularity, thanks to installable modules allowing to easily customize the interface. It is based on TypeScript which is a superset of ECMAScript 6 and therefore backward compatible with JavaScript (ECMAScript 5, widely used in web development) to which it adds anonymous functions, iterators or for-of loops.

   All this allows a dynamic loading of the page and an asynchronous compilation of the different models.

   For us the interest is to be able to create a page as easily as possible. For example, to generate an HTML table, we just have to note on the template a loop (ng-for) traversing an array that has been declared in the part in TypeScript.

This simplifies our work in the sense that we do not have to write all the JavaScript functions for the dynamic display of the page as it is managed by the functions provided by Angular.

2. **JSON**
   JSON stands for JavaScript Object Notation. This is indeed a textual data format derived from the notation of JavaScript objects. It thus makes it possible to represent the structured information (like XML), accompanied by labels to interpret it (without restriction on their number). It is very simple to read, to learn and to implement but also complete and not verbose. It is now widely used as a data transport language by AJAX and web services.

   In our case, we recover the data stored on The Things Network by our sensors. Its data is converted to JSON which will be interpreted by Angular and assigned to a data table. This will be detailed in the next part of the report.

3. **Leaflet**
   Leaflet is an online mapping JavaScript library that is used by OpenStreetMap, a free and open source mapping project in the same vein as Google Maps.

   The open source character of OpenStreetMap is its main advantage for us since it allows us to freely use the API unlike Google Maps which requires a key and to pay a subscription outside the trial version.

   In addition, Leaflet offers interesting features compared to previously identified needs. It allows simple creation of clickable markers, triggering actions on the page. For example, directly from the TypeScript code and the recovered data array (in JSON), one can generate points on the map that can be clicked to display details.

4. **CanvasJS**
   Finally, CanvasJS is an easy-to-use HTML5 and JavaScript graphics library. It works on any device (mobile, desktop, etc.) and allows you to create dashboards. It is implemented in the JavaScript code called when clicking on a map marker and automatically generates a graph from the data stored in our program.

5.5.2 Server

The role of the server is to make the connection between TTN, MongoDB and Angular. Thanks to libraries available to interact with TTN and MongoDB, Node.js was naturally chosen to develop the server. It is also an asynchronous event driven Javascript runtime, allowing the execution of instructions when receiving events which is useful for the data communication. Indeed, the server sends a request to an other service, like a database or TTN, it will not wait the response and will continue its execution. When the server receives the response, it will stop his current execution, execute the instructions dedicated to handle this response, and after this, it will go back to the normal execution. In this way, Node.js can handle several activities in ”parallel” without it seems to slow down the other applications. When the server starts, it subscribes to TTN in order to wait new data. When the server receives a new data, it stores this data in MongoDB.

For enabling Angular to retrieve data from the database, the Node.js server has to provide a REST (Representational state transfer) service. Each time a user ask for a page, Angular sends an HTTP request to the server in order to retrieve all data stored in the database. When the server receives the request, it will interact with the database to get data and send them into an HTTP response to Angular. The figure 7 shows the web architecture taking into account the interaction between TTN, the Node.js server, MongoDB and Angular:

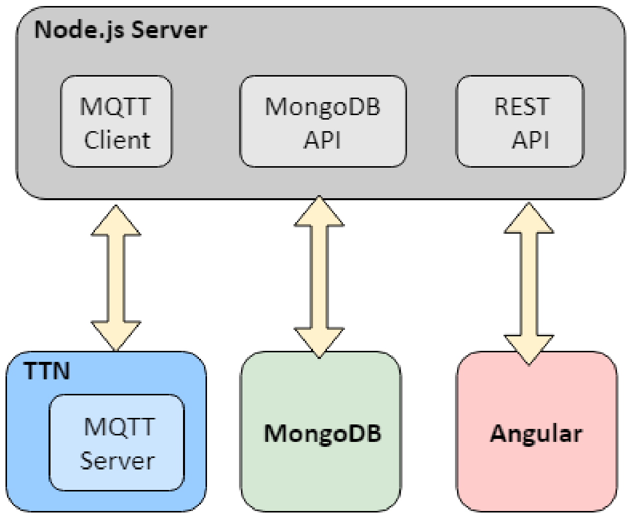
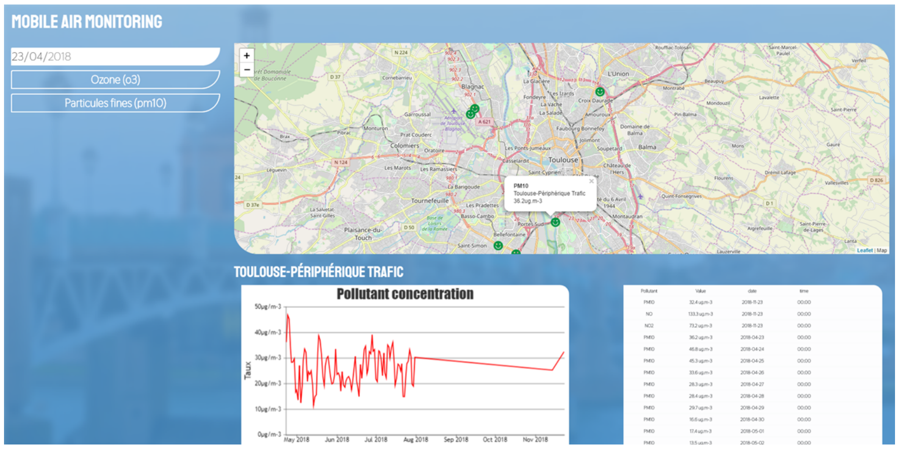


Figure 7: Overview of the web architecture

6 Work and Test Result

6.1 User Interface



Pollutant	Value	date	time
PM10	32.4 ug.m-3	2018-11-23	00:00
NO	133.3 ug.m-3	2018-11-23	00:00
NO2	73.2 ug.m-3	2018-11-23	00:00
PM10	36.2 ug.m-3	2018-04-23	00:00
PM10	46.8 ug.m-3	2018-04-24	00:00
PM10	45.3 ug.m-3	2018-04-25	00:00
PM10	33.6 ug.m-3	2018-04-26	00:00
PM10	28.3 ug.m-3	2018-04-27	00:00
PM10	28.4 ug.m-3	2018-04-28	00:00
PM10	29.7 ug.m-3	2018-04-29	00:00
PM10	16.6 ug.m-3	2018-04-30	00:00
PM10	17.4 ug.m-3	2018-05-01	00:00
PM10	13.5 ug.m-3	2018-05-02	00:00

Figure 8: Overview of the user interface

We wanted a light and easy-to-use user interface. It is based on a single page representing a map of the region on which, for a given date, different points corresponding to the concentration of various pollutants.

For our demonstration, we relied on open data of ATMO Occitanie [1] to then adapt it to our own data. Note that when writing this report, we could not recover enough data from our own sensors so all this part is based on these open data.

The map is centered by default on Toulouse (we manually entered the GPS coordinates of the city) and on the scale of Occitanie (zoom level of the map established in the same way). Of course all this can be automated by a selector of different cities or regions attached to GPS coordinates and a zoom level, allowing a dynamic display but this was not necessary for our demonstration.

On the side of the map, there is a date picker that displays a calendar. If we collect more data in the future, we can consider adding an hours selector. Then, it is possible to select the pollutant whose concentration is to be displayed. Here we have limited ourselves to ozone and fine particulate matter (10 µm) because they are the ones for which we had the most data.

[1]http://data-atmo-occitanie.opendata.arcgis.com/datasets?t=Mesures

6.1.1 Map and colour code

Whenever the date or pollutant is selected, the map will automatically update displaying for each measuring station a colored icon indicating the different levels of pollutants (4 levels from excellent to critical). The concentrations are in µg.m-3.

To determine this scale, we relied on scientific documentation indicating from what concentration at constant exposure, the pollutant became dangerous (last level). For the other levels, we were inspired by what is done on other maps such as ATMO Occitanie.

Therefore, for a given moment, we get a map of the region to quickly know the state of the air with a simple color code to understand.

These icons, generated through Leaflet, are clickable and call a JavaScript function that will perform automatically on the web page. With this, we display details for a given station at the bottom of the page.

6.1.2 Details display for a station

In our demo version, we display both a graph with ozone concentration (blue curve) and PM10 (red) for the past year. By passing the cursor on these curves, one can know more about a specific point (date, precise value). These values can be found in the complete table of data displayed next to it.

These graphics are made thanks to CanvasJS library.

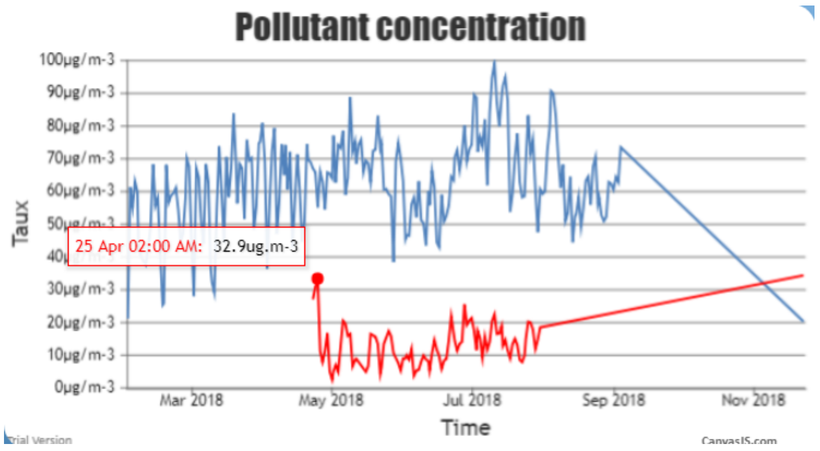


Figure 9: graphics display

7 Project Management

7.1 Definition of project scope and projected Gantt

We relied on an approach that could be described as agile in the sense that our work was articulated by sprints completed every 1 or 2 weeks depending on the team meetings and sometimes, exchanges with the tutors of INSA or Wollongong University. The idea of agile methods is to put the product at the center of the project and that goes through a realization of incremental functions, sprints. At each sprint start, we define realistic objectives to achieve, the functions to develop. At the end, a point is made on the progress of the work to redefine new objectives.

These project management methods are therefore based on the idea of adaptation and, as a result, organizational projection, such as the implementation of schedules, are made in the short term. However, at the beginning of the project, it seemed wise to develop a Gantt chart to plan future tasks. This diagram can be consulted in appendices.

To establish this Gantt chart, we applied the process of "Design Thinking", a synthesis between analytical and intuitive thinking. This method is based on 5 steps:

1. "Empathize": what do users do? what do they think? what do they feel? what do they say ?
2. "Define": define the framework of the problem
3. "Ideate": to produce solution ideas, typically brainstorming
4. "Prototype": refine solutions, have a user feedback
5. "Test": Test the prototype satisfies the user or not

In summary, the idea is to rigorously define the context and framework of the project, define what we will do and make decisions about the equipment and technologies used.

Having framed the project, we were able to define a set of tasks (listed in the Gantt) and assign positions based on skills. Overall, the project was divided into two main areas of expertise: electronics (David and Amine specialty) and IT (Ting, Tehema and Clément specialty). The communication part with the LoRa module represented the interface between the two trades between sending data (electronics) and retrieval (informatics).

The forecast Gantt chart could not be observed as expected. It still allowed us to define the tasks to be achieved, this being reported in our project management tracking tool on Trello (see appendices).

7.2 Teamwork and communication

The methodology used in this project involved iterative work cycles. These iterations were articulated around team meetings and sometimes with our university tutor, Thierry Monteil. Each of the team meetings allowed us to analyze what was done and to update the progress of the project through the Trello Task Table.

Trello is an online project management tool. It is presented as a table where you can list the tasks, the people to assign to them or the deadlines of these. The first tasks were defined from the Gantt chart but these were very general. The advantage of the agile method was to be able to refine them to arrive at a level of granularity the largest possible and get the most simple and precise tasks to help us in the progress of the project.

Externally, we were able to communicate with the sponsor of the University of Wollongong, Nicolas Verstaevel, by email or by meeting him in January. This allowed to reframe the objectives of the project.

7.3 Pooling resources

To facilitate teamwork, all our resources were shared within a cloud, a storage and file sharing service, Google Drive. It is on file online that are put all the documentation and some parts of the code necessary for understanding the project. This folder is only accessible to project members.

To share the code and allow it to be updated over the versions, a repository, also shared within the project team, was created on GitHub. This tool allows the versioning specific to our application and to observe the evolutions made in the code as a function of time, allowing us to recover old versions in case of errors for example.

8 Conclusion

Due to time constraints and lack of knowledge, we did not succeed in reaching our goal in time and in implementing all features that we wanted. However, this project was a good experience for us, mixing hardware and software skills and learning us more about the IoT field. The communication between the TTN and our web server is implemented, and the user interface includes a map and graphs to monitor air quality as we planned. The biggest challenge was the data extraction from sensors using LoPy micro-controller: there was few libraries for exploiting gas sensors.