Innovative Project report

Save Water and Energy

Consumption awareness tool

INSA students

Karim Aldandachi
Mickaël Commerot
Vincent Laurens
Lorine Pose
Hamza Safri

ILYA founders

Simon Buoro
Antoine Escande

INSA Tutor

Thierry Monteil

Acknowledgments

We would like to thank first of all Simon Buero and Antoine Estande, ILYA's co-founders, for trusting us with their project and for giving us the opportunity to work on it.

We would like to also express our gratitude to our INSA tutor, Thierry Monteil, who accompanied us during this project, and we thank him for all the advice and help he provided us with.

We really appreciate the help given to us by Gauchard David, mainly on the ESP8266 development. His contribution was essential to the completion of this project.

Finally, we would like to thank our English teacher, Joseph Shea, for helping us with our written reports and oral presentations, and for always making sure our English is correct.

We dedicate the result of our hard work to our beloved classmate, Lorine Pose, who we know did the best she could to be a part of this project. We send her our best wishes.

Abstract

Nowadays, people are becoming increasingly concerned about the future of our planet. The consequences of climate change are worse every year. Fighting against it has become a source of inspiration for many young entrepreneurs. ILYA is a young startup that is developing two products: a cyclic shower to reduce water consumption and an awareness tool to sensitive people to save water, energy and therefore money when taking a shower. In this report, we present in detail the second product, a tool that allows measuring in real-time the water and energy consumption during a shower. The prototype we developed consists of a sensor system that collects data on the water flow and quantity, stores it in a database, and then display an analysis of this data on a web interface. The data is captured by an autonomous and energy-efficient node and is then sent via WiFi to a local server to be stored in a database and then on the cloud. The final prototype is not fully optimized but both simple and realistic.

Table of contents

I. Introduction

ILYA is a french start-up that aims to decrease the environmental impact of everyday actions by offering innovative and durable solutions to assist companies in their responsible approaches. Their first project is a sensibilization tool that provides an energy and water consumption diagnostic service to its clients.

Today, particularly in hotels, people tend to have an irresponsible consumption of water and energy. This phenomenon is mostly due to the fact that since they paid for the room, they feel the need to make the stay as profitable as possible.

To tackle this issue, ILYA wants to develop a solution that sensitizes people about their water and energy consumption in order to decrease it. This tool will be installed first in public places like hotels and camps, measuring the consumption, analyzing the collected data and then displaying it on a screen. Therefore, the person will be able to know how much he or she is consuming, while in the shower!

II. State of Art

Before starting to design the prototype, we needed to research the tools and technologies we were going to use, mainly the circuit board and the type of database. Below is the result of our research.

A. Hardware

Before we started to design our prototype, we wrote down a list of requirements that the prototype had to respect. For example, it has to be autonomous and self-dependant on energy. In fact, the client will add that module in a bathroom and the system must be able to stand-alone for as long as possible. Therefore, we needed a module that is able to fetch data from sensors and to send them to a cloud using a Wi-Fi protocol, all while consuming the least amount of energy possible.

We chose to use ESP boards rather than other constructor boards like STM because we wanted a board that can connect to WiFi. Furthermore, the development kit proposed by ESP relies on Arduino's development IDE[1], which is very easy to use. More importantly, ESP cards and Arduino IDE had already been used by all team members in previous projects. For example, most team members used the ESP8266 board during INSA's Hackathon back in December of last year.

[1] https://www.arduino.cc/en/Main/Software

We narrowed the choices down to two ESP boards: ESP8266 and ESP32. The following paragraph will describe studies on the energy consumption of those two modules. In the first part, we will compare the energy consumption of ESP8266 and ESP32. In the second part, we will present the development environments used to develop the software of these boards.

To measure the energy consumption of both boards, we used a USB Safety Tester, which is a basic plug-in USB tester that displays voltage and current values.

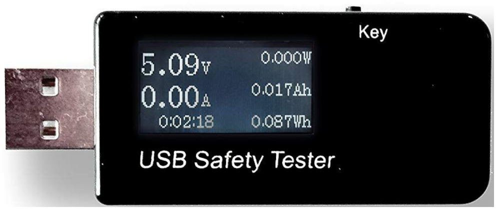


Figure 1: The USB Safety Tester

Below is a table presenting in more detail the different characteristics of each board.

	ESP32	ESP8266
Microprocessor	Xtensa Dual-Core 32-bit LX6 with 600 DMIPS	Xtensa Single-core 32-bit L106
Number of cores	2	1
Wi-Fi	Yes	Yes
Bluetooth	Yes	No
Supply needed in Run mode	110 mA	80 mA
Supply needed in Sleep mode	50 mA	20 mA
Energy consumption per min (I/P=UI/E=PI) Accuracy 1A	110mA/0.05Watts/33 Joules	80mA/0,04 Watts/24 Joules
Energy Management integrated	Yes	No

Table 1: Comparison between ESP32 and ESP8266 board

The results above show that an ESP8266 board consumes less energy than an ESP32 board with the same test conditions (program, temperature, sensors connected...). But if we look at the performance aspect, an ESP32 board with two cores is more efficient than an ESP8266, which uses one core for Wi-Fi connection and transmission, and the other one to process data collection from sensors.

The development environment is something that must be taken into consideration because it is used to design the "brain" of the board. One thing that is important to mention is that an ESP board has a RISC microcontroller, which means that it can be programmed in C++ using Arduino IDE. This criterion is another reason, after the ability to connect to Wi-Fi, that comforted our choice of an ESP board rather than an STM board.

In the end, we chose ESP8266. In fact, our system doesn't necessarily need a high performing board because it will only collect and send data via Wi-Fi. The energy consumption criteria was a more critical point for our choice.

Complementary to the hardware part, we know that traditional grooves are very big so we had to think of a way to reduce the size, by having fewer inputs for example. We designed a PCB groove shield that is the size of the circuit board. It can be found in the appendix at the end of this document (Appendix 3).

B. Database

For our database, we had two options, SQL and NoSQL, the most common types of databases. We decided to compare them based on attributes like consistency, availability, and speed and then choose depending on the needs of our application.

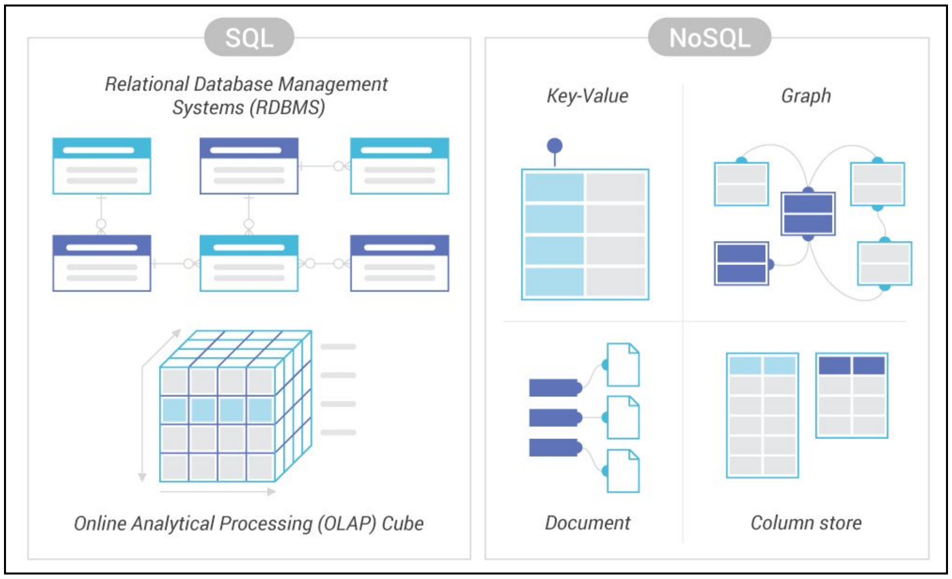

SQL
Relational Database Management Systems (RDBMS)
Online Analytical Processing (OLAP) Cube
NoSQL
Key-Value
Graph
Document
Column store

Figure 2: A representation of SQL and NoSQL databases

Source: https://www.scylladb.com/resources/nosql-vs-sql/

SQL stands for Structured Query Language and is the traditional and most popular database system. It is based on the relational model, which means that data is stored in interrelated tables, with rows and columns. An indexing mechanism supported by SQL is used to speed up various operations like reading or creating views, which can join data from multiple tables. Examples of relational databases available are MySQL, Oracle, and SQLServer.

However, in the last couple of years, another database system has grown in popularity, due to its ease of access, speed, high scalability, and distributed computing. NoSQL is a non-relational database management system that supports flexible schema to store the data. NoSQL databases support a large scale of data storing by providing distributed data stores. Most of them are based on storing key-value pairs. Hence, they are suitable to store data generated by IoT applications. An example of a NoSQL database is MongoDB.

	SQL	**NoSQL**
Model	Relational database system	Non-relational database system
Data storage	Tables (rows and columns)	Documents, column stores, graphs, key-value pairs
Schema	Fixed, very hard to modify	Schema-free database, easy to modify
Data structure	Structured data	Semi-structured or unstructured data
New fields	Adding new fields in the table may require altering the schema	New fields can be added with much more ease
Scalability	Scales well vertically	More suited for horizontal scaling
Querying language	Use of powerful declarative language	Use of JSON data objects
Transactions	Supports ACID transactions	ACID transactions support varies per solution
Consistency	Strong	Eventual

Table 2: Comparison between SQL and NoSQL databases

The following paragraphs are based on the paper written by Rautmare, S. and Bhalerao, D.

- Data model:

SQL is a relational database where data is stored in the form of rows and columns in tables that are linked together. In the table, columns represent different fields, while rows represent the records. This data model gives a well organized but rigid structure of data. For example, the same table cannot be used to store different kinds of information. You also can't insert a number where a string is expected. This strict data template enables a low probability of a user making mistakes.

On the other hand, NoSQL is a non-relational database that uses data modeling types like document stores, key-values, and graphs. Similar documents can be stored in a collection, which is analogous to an SQL table. You can store any type of data in any document. This flexibility can lead to consistency issues. Nonetheless, this feature proves to be efficient when handling unstructured data like Word, PDF or multimedia files.

In IoT applications where data structure cannot be predefined properly or data is heterogeneous, NoSQL definitely has a higher edge. But for simpler applications, a fixed data structure is enough and accessing data is faster.

- Scalability:

Vertical scalability is the ability to scale up or to add resources to a single node in a system, typically involving the addition of CPUs or memories to a single computer. Traditional databases like SQL were designed primarily to scale vertically by adding more power to a single node. Horizontal scalability is the ability to scale out or to add more nodes (servers) to the system so its load can be shared. It is one of the main characteristics of NoSQL.

For IoT applications where the distribution of data amongst several servers is required, the use of NoSQL is a better option because it allows the expansion of the hardware system in a stepwise manner, without needing to invest in high-end modules of hardware.

- Data Retrieval:

To look up data in SQL databases, the user needs to collect it from interlinked tables using JOIN statements and then combine the results. The process is the same when writing.

In NoSQL, each piece of data represents an object which contains all its related data. This allows a much faster Read and Write operations. This feature is mainly required when analyzing the data stored in the database or when taking any action on a specific record.

- Transactions:

Relational databases like SQL support ACID (Atomicity, Consistency, Isolation, Durability) properties and hence guarantee the validity of all transactions. NoSQL supports BASE (Basically Available, Soft state, Eventual consistency) properties, focusing more on eventual consistency, providing availability and on being partition tolerant.

For IoT applications in their developing stage, data generation increases day by day. So, a database system that is partition tolerant is more suitable. Also, in IoT applications where data is collected first and then analyzed, later on, the database can be eventually consistent. In this case, strong consistency is not a mandatory characteristic. It is better suited in application scenarios where an immediate update of data is necessary (online reservation systems, inventory systems, retail stores, banking).

- Maturity of the system:

Finally, one of the most important issues in a database is security. As NoSQL is a recent technology and hence is less mature, it is more likely to generate new issues, due to a lack of knowledge. Developers have less experience with newer database systems, so mistakes are made. On the other hand, in SQL, most of the issues are all addressed as it has been used for several years now. Security concerns like authentication, data integrity, confidentiality, auditing, and client communication are addressed in SQL. In NoSQL, they are yet to be integrated.

In the IoT ecosystem, sensors transfer a lot of sensitive information. So, it is necessary to have a secure channel for information exchange and a secure system to store this sensitive data. For applications where data security and integrity is of high priority, it is better to go for a secured data storage like SQL.

After studying all these different attributes, we opted for a NoSQL database because its structure is more suited to the evolving data schemas of our project.

III. The implementation

a) A general overview of the tool

The tool consists of sensors, a circuit board, and a cloud. The sensors are placed on the showerhead and measure the temperature as well as the water flow. The circuit board then receives the data and sends it to the cloud, where a database is stored.

Each sensor is linked to a specific hotel room when configured. In fact, we need to be able to attribute each data to the corresponding shower head.

b) <u>The architecture of the tool</u>

To design our tool's architecture, we got the inspiration from the M2M standard developed by the European Telecommunication Standard Institute (ETSI), which is used in many markets, such as Connected Vehicles, Smart Metering, and e-Health.

ETSI identifies three domains: Application, Network, and Device. The Device layer consists of sensors, actuators, and any other type of device found in a wireless sensor network. It also contains a gateway to be used by devices that can't connect by themselves to the M2M core. The Application layer, as its name suggests, contains all the different applications. Finally, the Network layer provides a communication network between the Application and the Device domains. [1]

For our specific tool, here is how we designed the architecture:

- The device layer contains the different sensors (flow and temperature) which are connected to an ESP8266 circuit board. These sensors send the collected data using the Wi-Fi module of the board to a broker, which in turn sends them to M2M applications.

- The network layer is responsible for the communication between the circuit board and the database. This is done thanks to a broker. A broker is something or someone that acts as an intermediary third party, managing transactions between two other entities. In the case of our tool, the broker is an intermediary computer program module that publishes a message for all subscribed applications.

- The application layer provides 2 services. The first consists of a MongoDB database that receives and stores the collected data. The second is an interface used by the supplier (Ilya) and the clients (hotel managers) to configure sensors and visualize the collected data in the form of graphs.

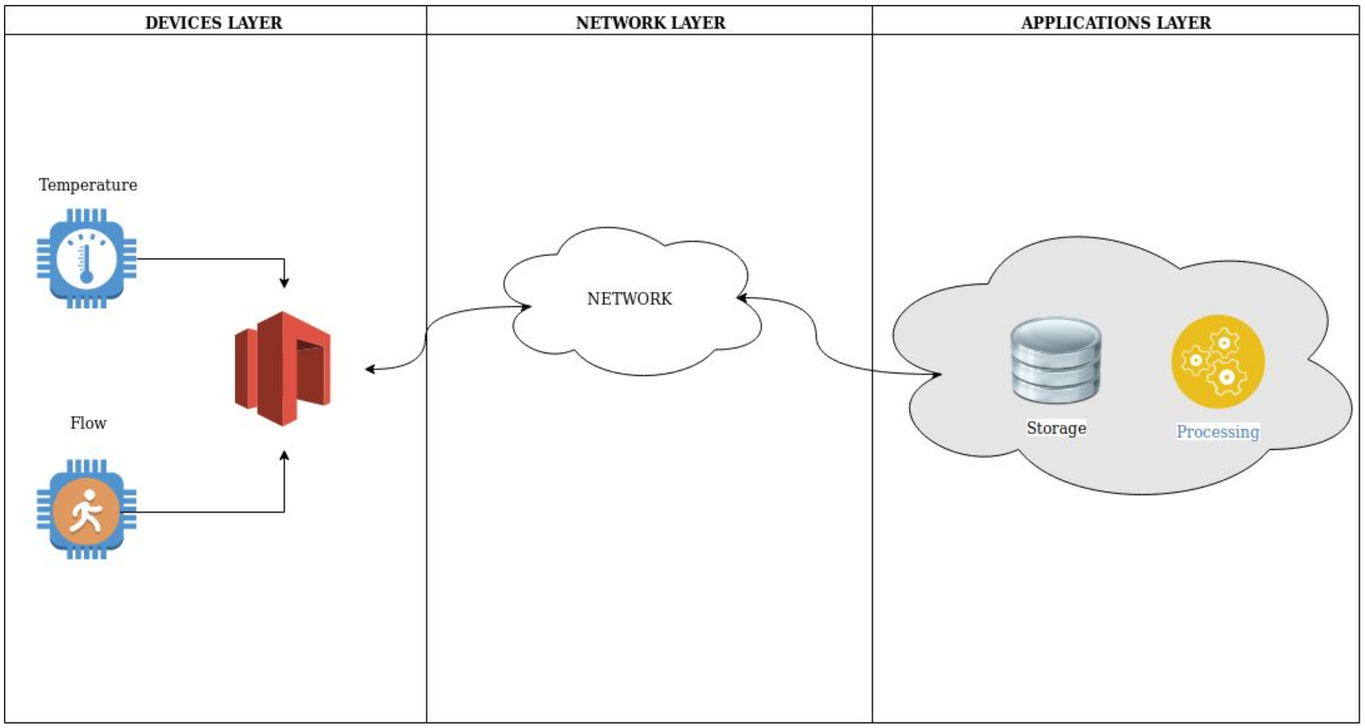


Figure 3: The architecture of our system

c) The Device Layer

In the Hardware state of the art, we explained the thought process that led us to choose an ESP8266 circuit board instead of an ESP32 or any board sold by STM (less energy consumption, Wi-Fi module available). Regarding the sensors, we needed them to give us information on the water flow as well as its temperature. Therefore, we used two types of sensors: a temperature sensor and a flow sensor.

For the temperature sensor, we used one called The Grove. It detects the ambient temperature thanks to a thermistor whose resistance increases when the temperature decreases. It can detect any temperature ranging from -40° to 125° with an accuracy of +/- 1,5°.

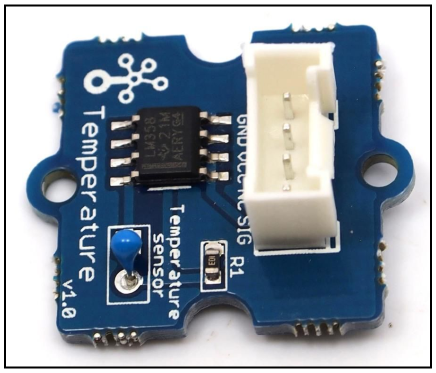


Figure 4: The Grove temperature sensor

As for the flow sensor, we used a YSF 201 Hall Effect Water Flow Meter. It contains a pinwheel sensor to measure how much liquid has moved through it. An integrated magnetic hall effect sensor outputs an electrical pulse with every revolution of the wheel. The water flow can be deduced from the frequency of the generated output signal.

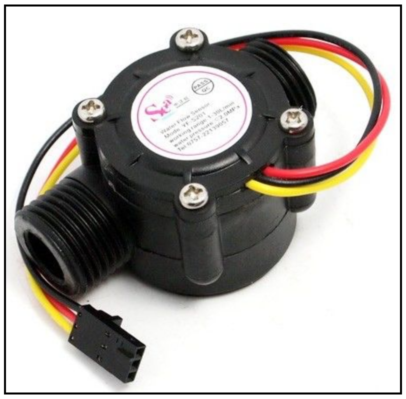
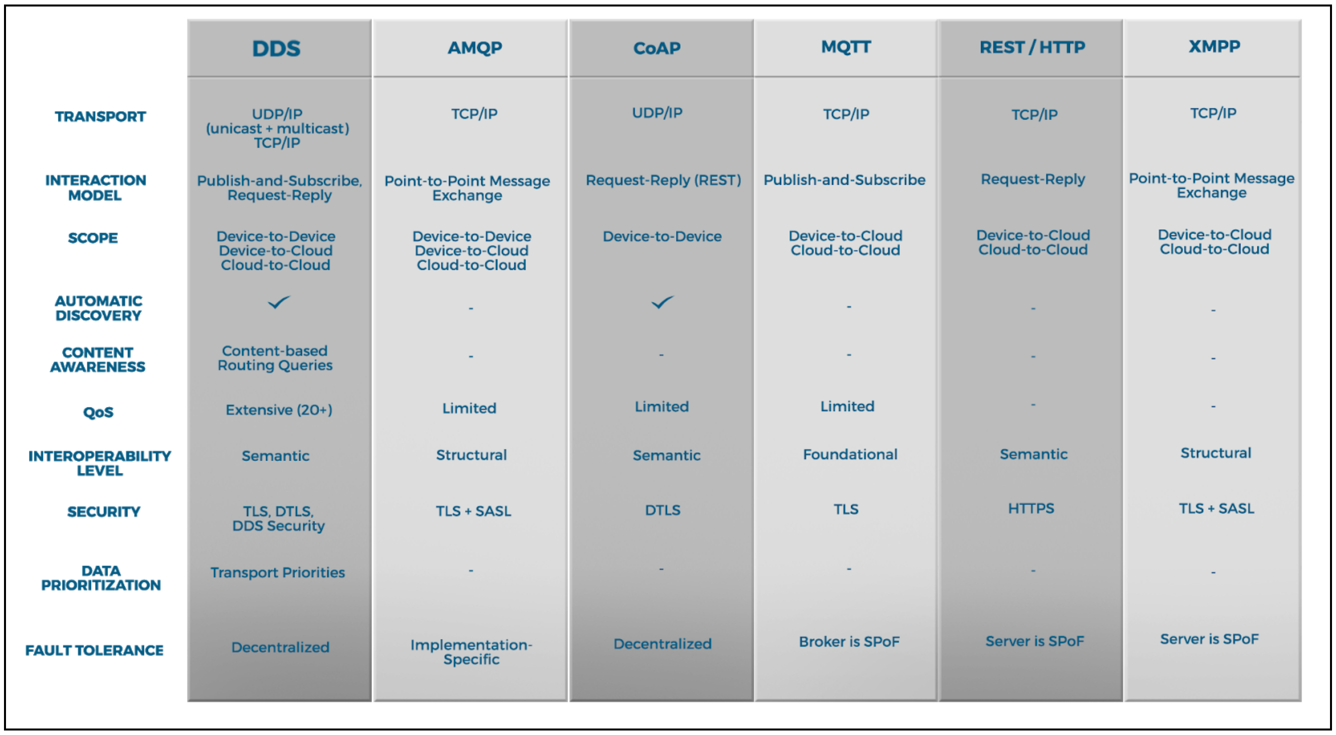


Figure 5: The YSF 201 sensor

d) The Network layer

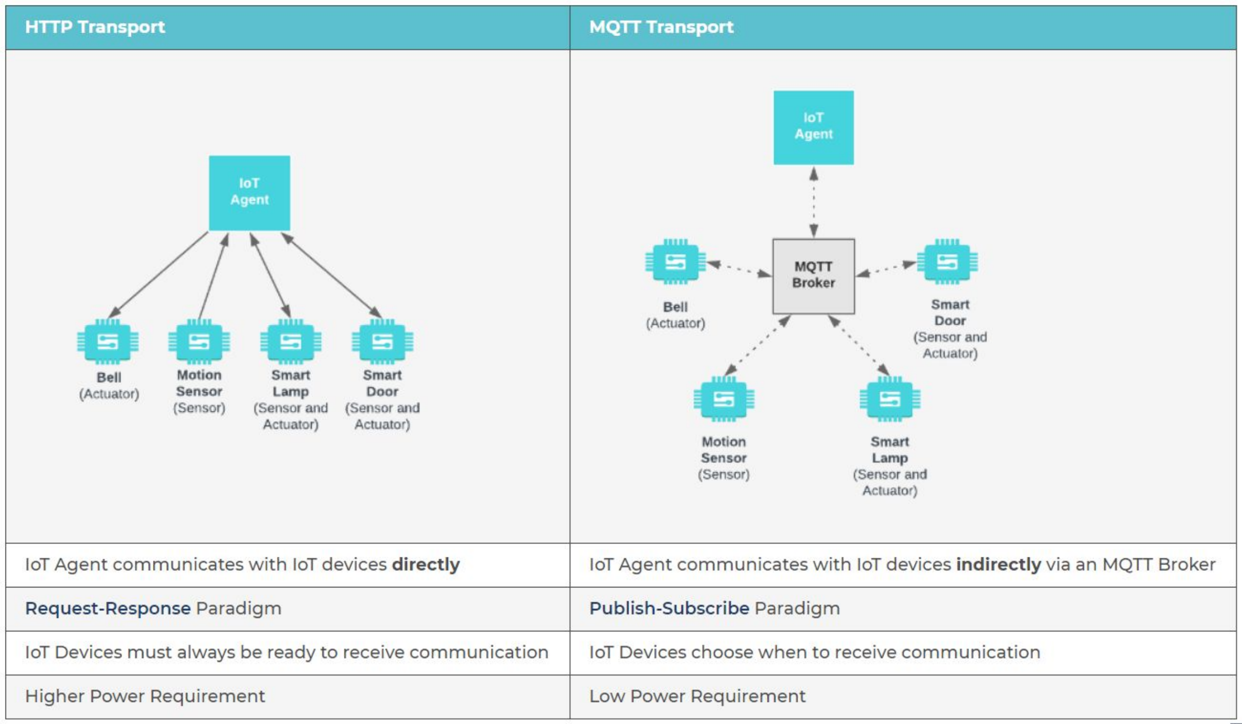
The Network layer is the interface between the Device layer and the Application layer, enabling the communication between them. On one hand, the monitoring application must be able to get the data from the sensors. On the other hand, the application server must be able to receive a request containing data sensors inside the payload. To send a request, there are many different network protocols to choose from.



	DDS	AMQP	CoAP	MQTT	REST / HTTP	XMPP
TRANSPORT	UDP/IP (unicast + multicast) TCP/IP	TCP/IP	UDP/IP	TCP/IP	TCP/IP	TCP/IP
INTERACTION MODEL	Publish-and-Subscribe, Request-Reply	Point-to-Point Message Exchange	Request-Reply (REST)	Publish-and-Subscribe	Request-Reply	Point-to-Point Message Exchange
SCOPE	Device-to-Device Device-to-Cloud Cloud-to-Cloud	Device-to-Device Device-to-Cloud Cloud-to-Cloud	Device-to-Device	Device-to-Cloud Cloud-to-Cloud	Device-to-Cloud Cloud-to-Cloud	Device-to-Cloud Cloud-to-Cloud
AUTOMATIC DISCOVERY	✓	-	✓	-	-	-
CONTENT AWARENESS	Content-based Routing Queries	-	-	-	-	-
QoS	Extensive (20+)	Limited	Limited	Limited	-	-
INTEROPERABILITY LEVEL	Semantic	Structural	Semantic	Foundational	Semantic	Structural
SECURITY	TLS, DTLS, DDS Security	TLS + SASL	DTLS	TLS	HTTPS	TLS + SASL
DATA PRIORITIZATION	Transport Priorities	-	-	-	-	-
FAULT TOLERANCE	Decentralized	Implementation-Specific	Decentralized	Broker is SPoF	Server is SPoF	Server is SPoF

Figure 6: A comparison of the different existing network protocols

Source: *https://istblog.adlinktech.com/tag/messaging-protocol/*


IoT Agent
Bell (Actuator)
Motion Sensor (Sensor)
Smart Lamp (Sensor and Actuator)
Smart Door (Sensor and Actuator)
IoT Agent
MQTT Broker
Bell (Actuator)
Smart Door (Sensor and Actuator)
Motion Sensor (Sensor)
Smart Lamp (Sensor and Actuator)

HTTP Transport	MQTT Transport
IoT Agent communicates with IoT devices **directly**	IoT Agent communicates with IoT devices **indirectly** via an MQTT Broker
Request-Response Paradigm	**Publish-Subscribe** Paradigm
IoT Devices must always be ready to receive communication	IoT Devices choose when to receive communication
Higher Power Requirement	Low Power Requirement

Figure 7: A visual comparison between HTTP and MQTT
Source: *https://fiware-tutorials.readthedocs.io/en/latest/iot-over-mqtt/index.html*

In the context of IoT, 3 main protocols can be used: CoAP, MQTT or HTTP.

- CoAP is based on the UDP protocol, which means that it sends data without getting feedback once the receiver has received the message. Therefore, data can get lost.
- HTTP is based on the TCP/IP protocol, which means that it gets a feedback message when the message is received. This requires more energy.
- MQTT uses a server broker which is the intermediate between the two ends. There are fewer requests and thus less energy consumption.

With this comparison, we can easily see that MQTT is the best solution for our tool. In fact, we cannot afford to lose information from the sensors (that would alter the analysis), and the low energy consumption, as we mentioned before, is a very important criterion.

For the initial testing phase, we only needed an already existing MQTT broker. After researching for a free and online MQTT broker, we found one called MaQiaTTo. It was interesting for us to use it because it hosts an online MQTT broker and allows us to easily create topics. However, there was a slight disadvantage of limiting the number of users to only one. Nonetheless, we still decided to use it for our tests because it is available everywhere, as long as there is an Internet connection.

Once we validated the tests, we needed to have a private MQTT broker for Ilya, instead of relying on an online version. So, we implemented our own Mosquitto broker and hosted it on a virtual machine (VM). Mosquitto is an open-source, configurable and fully implementable MQTT broker. Not only it doesn't limit the number of users like MaQiqTTo, but it allows the administrator to define the ACLs (Access Control List) for each user. That way, a user will only be able to publish on topics where he or she has the right to. This translates into extra security to protect each hotel's data. However, it is less transportable during tests because it requires the VM to be connected through the web, which is why we had to use MaQiaTTo first for the tests.

e) <u>The Application layer</u>

i) Data structure

The data structure is an essential phase in the age of data. As mentioned before, we used a NoSQL database. In this project, we can classify the data into two groups:

- Indexing data: an ID field created by Ilya to identify a device in a hotel
- Collected data: the data sent by the device to the database via the broker

Therefore, we created two databases. The first one is an indexing database that contains mapping data between the device id and the path to the database storing the collected data. The second one stores the data collected from the shower sensors. In order to properly structure the data, we programmed a Rest service that plays the role of an intermediary between the broker and the database, as well as between the database and the web interface.

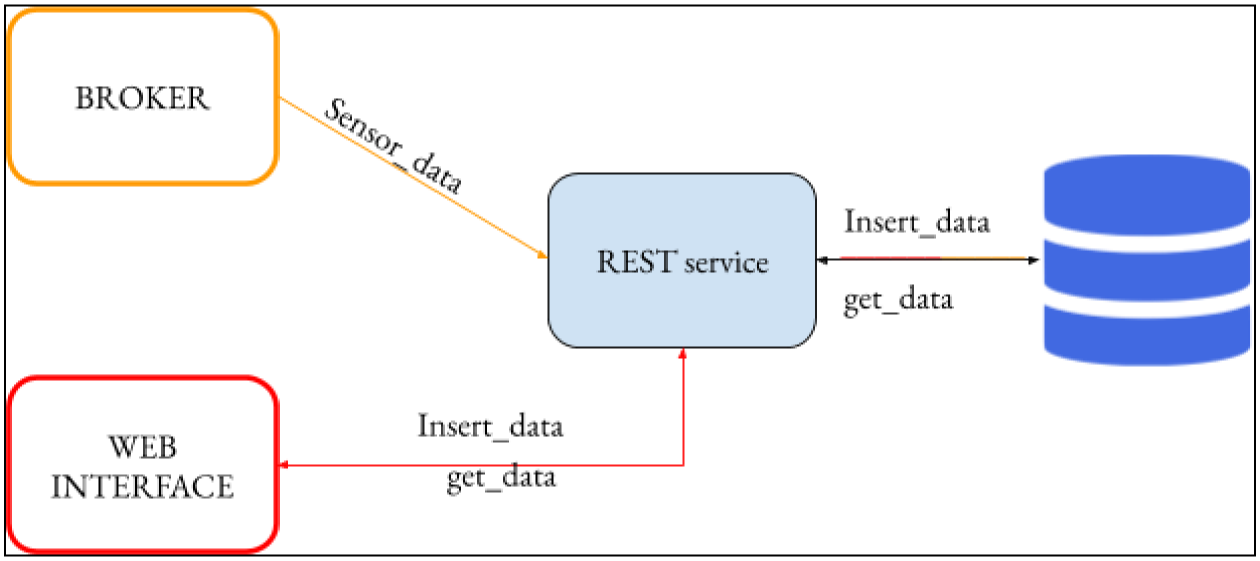


Figure 8: The architecture of the REST service

ii) Web interface

The interface's purpose is to visualize the data given by the sensors and to plot charts and dashboards. We used Node-RED, a programming tool that enables us to wire together hardware devices, APIs and online services, using a wide range of nodes. It is very practical to use because it is a browser-based editor, which means that it can be run on any operating system without too many dependencies. The tool offers as well a code interface that is directly integrated into the browser. Furthermore, a deep knowledge of programming languages and algorithms isn't necessary to build an application on it.

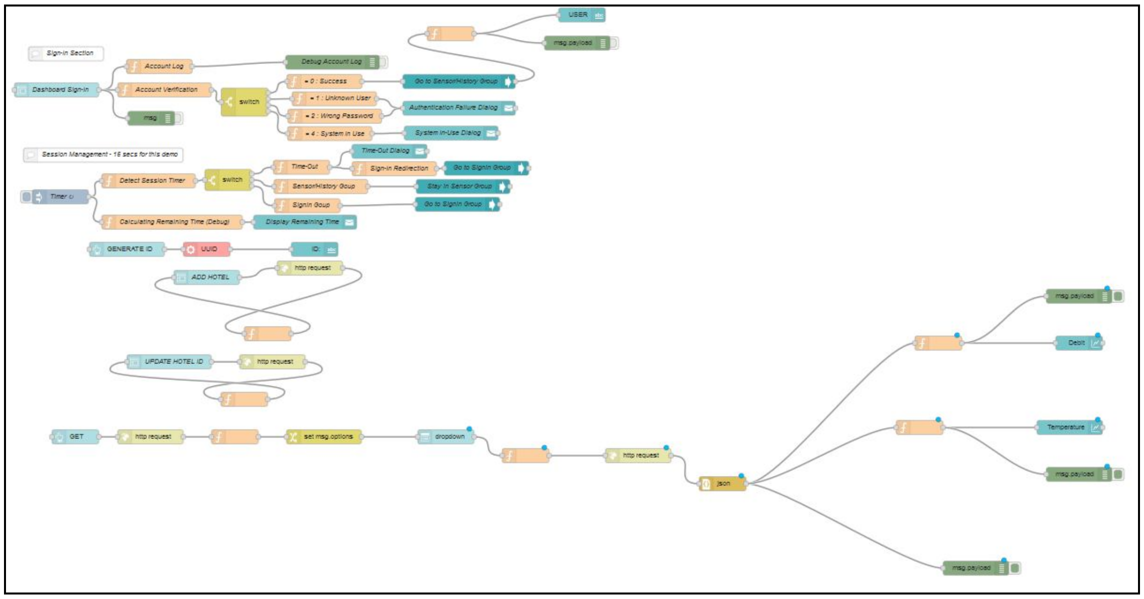
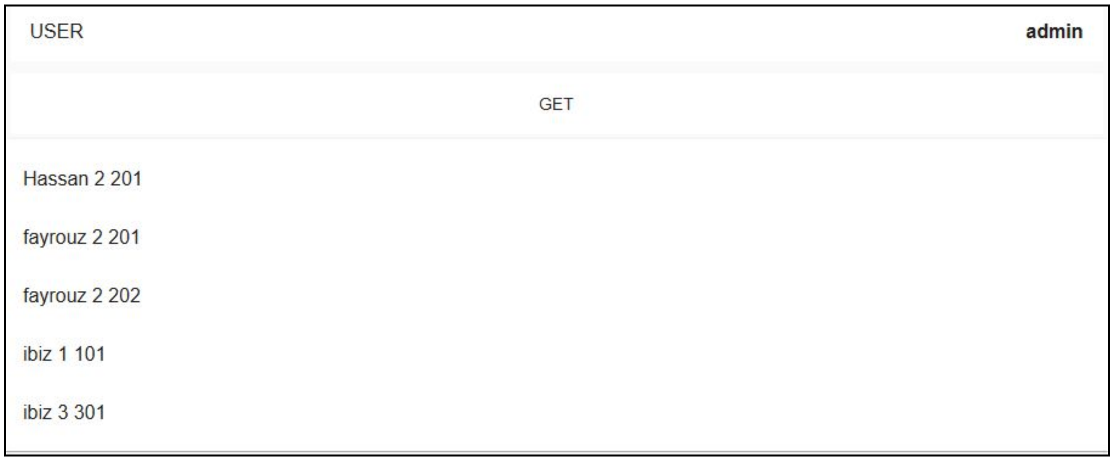


Figure 9: A visual of our application services on Node-RED

In our scenario, we divided the interface into two parts. The first is a service that allows the user (ILYA) to select and retrieve data from the database, as well as print charts and statistics on a dashboard. The second part is for the configuration of sensors. The user can generate a unique id for a newly-installed sensor, add it to the corresponding hotel or update the sensor's id.


USER **admin**

GET

Hassan 2 201

fayrouz 2 201

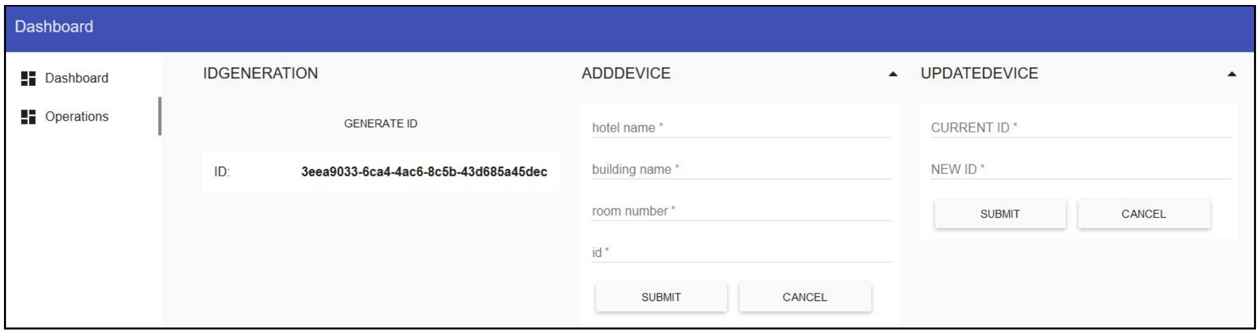
fayrouz 2 202

ibiz 1 101

ibiz 3 301

Figure 10: The first service of the interface

In the picture above, the administrator can retrieve data from sensors situated in room 201 in Hassan hotel, rooms 201 and 202 on the second floor of Fayrouz hotel and rooms 101 and 301 in Ibis hotel.



ID: 3eea9033-6ca4-4ac6-8c5b-43d685a45dec

Figure 11: The second service of the interface

In the picture above, the user has generated a new id (*3eea9033-6ca4-4ac6-8c5b-43d685a45dec*) for a sensor using the 'IDGENERATION" button. This id is then used when configuring the new sensor in a specific hotel room.

f) The final implementation

At the end of the project, we packed our prototype to make sure that it is functioning well. We also wanted a turnkey solution. So, we gathered all the application layer entities (Web interface and Data structure) on a single virtual machine. The objective is to have an application layer that can be easily deployed on a cloud provider, but also to be able to deploy and use it without having to install anything (an OS or a hypervisor).

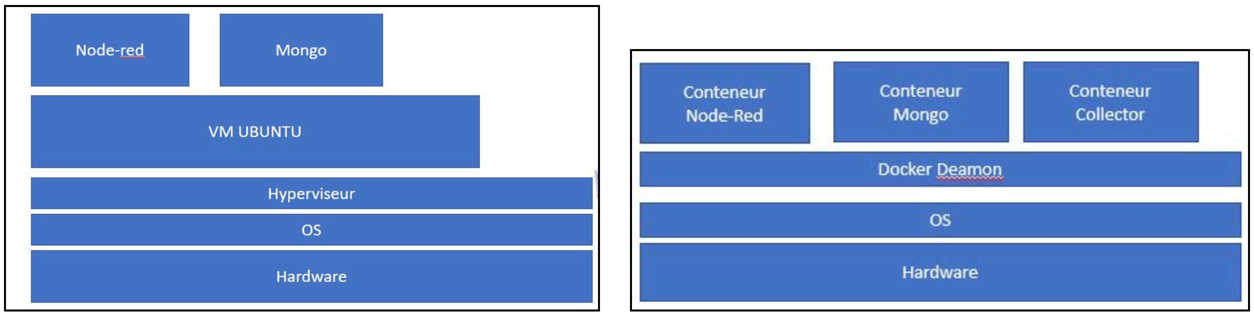


Figure 12: Description of virtualization (POC package on the left, industrialization version on the right)

On the figures above, we can observe that the first solution we proposed consisted of deploying services on a virtual machine (VM). It that case, the VM containing all services (database, UI, collector python) can be deployed on any cloud (OpenStack, Azure, AWS, Google…). Indeed, we want our application to be deployed without any complementary configuration.

Once we finalized our VM, we started to imagine a solution that would be more scalable, based on containerization. Indeed, we know that using containers allows the solution to be serverless which helps to scale the application's storage capacity up or down depending on the demand. The more requests there are from clients, the more we will deploy container instances to load and balance the demand, instead of overloading the servers. Another advantage of using containers rather than a VM is the fact that containers can be installed very fast.

To implement our VM, we used an Ubuntu distribution where we installed Node-Red (npm, nodejs), git, python3, and MongoDB. That way, to start the services, we just needed to start the VM. Once we validated the implementation of the VM, we implemented a container solution based on DOCKER technology. We prepared this solution to introduce our services in an industrial environment. From a technical aspect, containers can be directly started on a cloud (OpenStack, AWS, Azure, etc.) using a recipe we wrote in a DockerFile stored on the "serviceDocker" branch.

IV. Project management

a) Team members

Our team is composed of five INSA students from varying academic backgrounds.

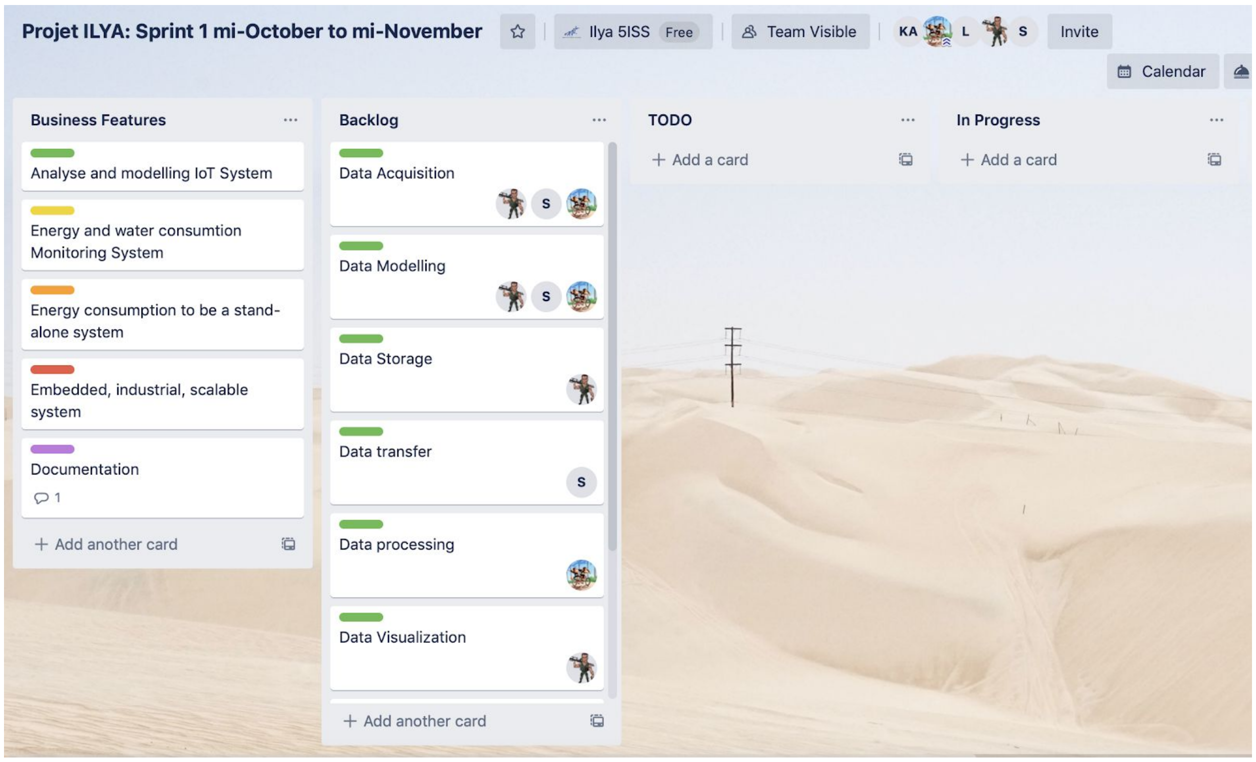
- Karim Aldandachi: specialized in Embedded Systems, double degree program at TSM in International Management.
- Mickaël Commérot: specialized in Computer Science, Master REOC (Réseaux Embarqués, Objets Connectés) at INP.
- Vincent Laurens: computer branch, specialized in Networks and Telecommunication.
- Lorine Pose: specialized in Embedded Systems.
- Hamza Safri: specialized in Computing, Networks, and Telecommunication, Master REOC (Réseaux Embarqués, Objets Connectés) at INP.

The variety of academic backgrounds amongst team members was a good thing because it allowed us to learn from each other's way of working and knowledge. This also helped us to cover all aspects of the project (Networking, database, electronics, team management...).

b) The organization of our team

We chose to undergo this project using the Agile method. First, we divided the overall project into different sub-projects and tasks and assigned each one to a team member. We also divided it into different sprints that lasted 3 weeks each. We agreed to meet once every week to keep each other in touch with the latest developments and difficulties. We also tried to meet the client every week to keep them updated on our progress. But that wasn't always possible because of scheduling issues.

We knew right from the start that this wasn't like any of our previous projects done at INSA. It was bigger, more demanding and the stakes were higher. Therefore, we decided to search for a software or website that would allow us to better manage the team as well as the project, and most importantly to keep track of our progress throughout the weeks. The software we chose is a task management app called Trello. It is used by millions of people and organizations all around the world to manage and organize projects. What's good about it is that it gives a visual overview of what is being worked on and who is working on it. This fits our decision to assign tasks to each member, as mentioned above.



Projet ILYA: Sprint 1 mi-October to mi-November

Ilya 5ISS Free · Team Visible · KA L S · Invite · Calendar

Business Features
- Analyse and modelling IoT System
- Energy and water consumtion Monitoring System
- Energy consumption to be a stand-alone system
- Embedded, industrial, scalable system
- Documentation (1)
- + Add another card

Backlog
- Data Acquisition (S)
- Data Modelling (S)
- Data Storage
- Data transfer (S)
- Data processing
- Data Visualization
- + Add another card

TODO
- + Add a card

In Progress
- + Add a card

Figure 13: Our first sprint detailed on Trello

c) Difficulties faced

The main difficulty our team had to face was setting up a meeting with each other, as well as with ILYA's founders. Indeed, as mentioned above, we all had varying backgrounds, with some in a double degree program or in professional training. Therefore, we all had different and heavy schedules. Finding a timeslot that suited everyone's schedule was challenging every time we needed to meet. As one would expect, not everyone was able to come to every meeting. But to solve this issue, someone simply took care of taking notes and then wrote a report that was later sent to every team member. This not only allowed any absentee to keep track of what was discussed, and therefore not miss out on any new developments but also to keep a record of what was said exactly.

Another difficulty was the fact that one of our members, Lorine, left the project mid-way for personal reasons. Therefore, we had to re-assign her part of the project between us, on top of what we each already had to do. This was definitely unexpected and it provoked a slight moment of panic in the team. We also had to adjust the sprints and the corresponding deadlines.

Another challenge was finding a balance between working on this project and still meeting all the other deadlines we had for other subjects, the extra double-degree work for some of us and the professional training for others. This wasn't an easy task.

V. Conclusion

At the beginning of this year when we were assigned to this project, we saw an opportunity to use our years of learning and technical experience to develop an innovative solution that would help the planet. The requirements for this project were the following: provide a functioning Proof of Concept of the tool, with the Network communication, sensor data, database management, and visual interface all taken care of. ILYA expected to have a functioning prototype where we can connect sensors they have designed and which are able to retrieve values and store them on a database.

We are happy to say that we successfully met the challenge. Throughout this report, we have presented the findings from our preliminary research to explain why we chose an ESP8266 circuit board and a MongoDB database system. We have also described how we designed the architecture of the tool and then went into more detail for each of its layers. We followed that up with an evaluation of the way we handled the project and the difficulties we faced.

However, we weren't able to tackle the energy consumption of the tool very well, due to a lack of knowledge as well as a lack of time. The founders wished to use BLE but we used Wi-Fi.

Bibliography/Webography

[1] A Feasibility Study on Developing IoT/M2M Applications over ETSI M2M Architecture
[2] Expressif, Datasheet of ESP8266 (2018). [online] Available at:
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwjA5bzHuv7mAhVSqxoKHeulAiwQFjAAegQIAhAC&url=https%3A%2F%2Fespressif.com%2Fsites%2Fdefault%2Ffiles%2Fdocumentation%2F0a-esp8266ex_datasheet_en.pdf&usg=AOvVaw0teH8Qd8wMW9ICXxdEvzq-
[3] Expressif, Datasheet of ESP32 (2019), [online] Available at :
https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
[4] Docker website, [online] Available at :
https://www.docker.com

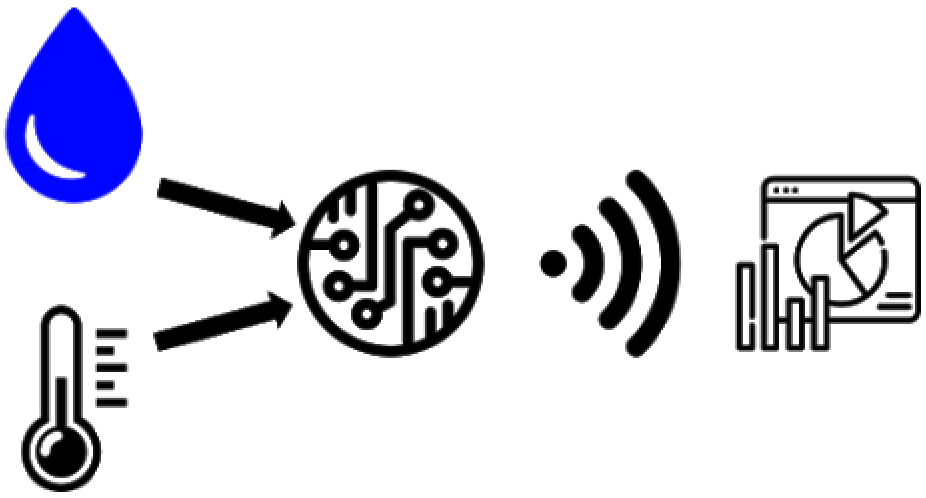
Rautmare, S. and Bhalerao, D. (2016). SQL & NoSQL Database Study for Internet of Things. [online] Available at: http://www.ijirset.com/upload/2016/may/350_Sharvari%20Paper.pdf [Accessed 12 Jan. 2020].

Appendix

1) Requirements document

Requirements document

Sensibilization Tool



Prepared by: ILYA Team
Prepared for: ILYA founders
Date Submitted: Nov 07, 2019

Project Sponsor: Thierry Monteil
Client Acceptor: ILYA
Project Manager: Vincent Laurens
Team members: Mickaël Comerot, Hamza Safri, Lorine Pose, Karim Aldandachi

Document Number: 1111-11/Project Number /BRD
Security Classification: Low
Version: 0.1

Creation Date: Nov 07, 2019
Last Updated: Nov 07, 2019

Project Overview

The main idea of the project is to provide an Energy and Water consumption diagnostic service to clients. Today, particularly in hotels, people tend to have an irresponsible consumption of water and energy. This phenomenon is probably due to the fact that since they paid for the room, they need to make the stay as profitable as possible.

ILYA is a company that proposes innovative and durable solutions in order to decrease the environmental impact of everyday actions and to assist companies in their responsible approaches.

In this case, the purpose of the project is to sensitize people in order to decrease their water consumption at first in public places such as hotels. To do so, the company is looking for a tool that can be used to do consumption measurements for several weeks and then to analyze the collected data.

A sensors system collector in the Device layer will be added to the client's bathroom appliances (shower, sink, toilet...). This collector is equipped with a flow and temperature sensor that collects data and sends it to a gateway connected to the cloud, which will then store it in a database hosted on the cloud.

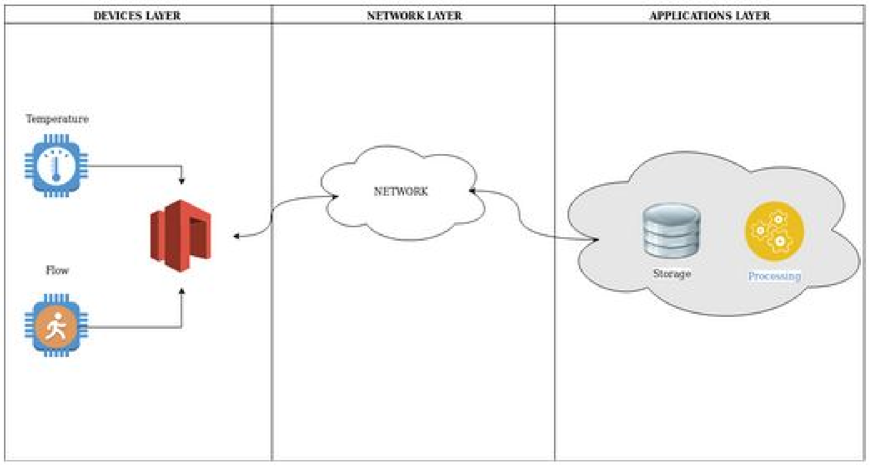


Figure 1: Architecture design overview

An application layer will be needed to establish statistics on data values and print resulted in charts of flow, volume of water, shower temperature, and energy consumption analysis.

Statement of functionality

An electronic acquisition system will be able to fetch the temperature and debit sensors' values. Then, it will model the data and store it in a database hosted on the cloud. This software will allow users to have a real-time look on their energy and water consumption when they have a shower, just by using a web application. The web interface has a Statistics dashboard of the flow, time and heat for each shower or sink in the bathroom.

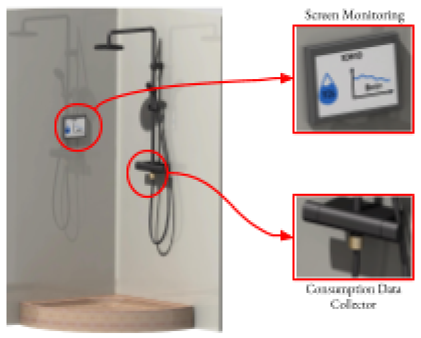


Figure 2: Sensors on a shower head

It also needs an Administrator access for the service providers, a Configuration access for the people who will install the sensors, an unmodifiable access to the data for the clients, and eventually a real-time display of the information for the users (the clients of the clients), as shown in the figure above.

Expected Results

- ❏ A functional Proof of Concept with flow and temperature sensors;
- ❏ A database with tools to display the data on dashboards;
- ❏ Documentation about the work done in order for the project to be continued later on.

Performance requirements

- We want the user to access statistics in real-time on a dashboard.
- We have to fix a data loss tolerance to 20% during transfer from sensor to cloud data storage.

Usability requirements

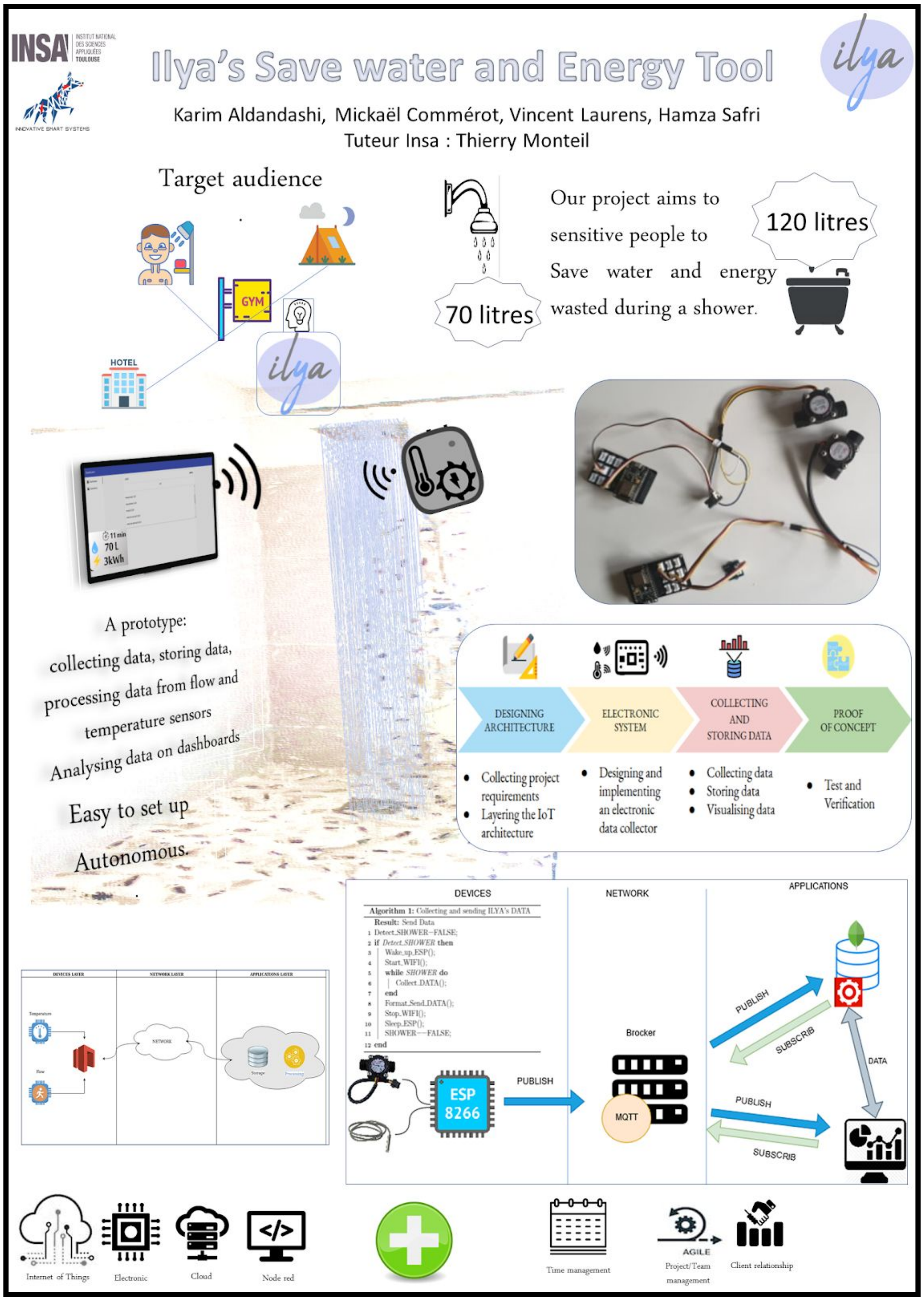
- The system must be easy to deploy or to use in a bathroom.
- Users must have an easy access to data via a user-friendly dashboard web application.
- Users can log in on an interface with their credentials, and then have access to the statistics.
- Users shouldn't be able to modify the configuration of the dashboard.
- Users can change the Installation and Deployment configurations, via an Administration view.

Out of Scope

We don’t take into consideration neither the “Network Layer” issues described in Figure 1, the Industrialization process, nor the Integration process.

2) **Poster**

This poster summarizes the scope of this project and the solution.



3) PCB groove shield design

While working on this project, we faced a crucial problem of space because our prototype as of now is not minimised. So, we thought about a solution which can be improved to start reducing the size. We designed a PCB groove which contains a connector for bot sensors and another one for power supply. In this document, we will describe the design of our shield.

We used a software called Kicad. It links several design tools together: EESCHEMA to design an electronic circuit and verify if the fundamental electric rules are respected, PCBnew to design the PCB component form, the final shape of the board and to make routing connexion of components by defining tracks.

The first step was to design the electronic components and the connections between them.

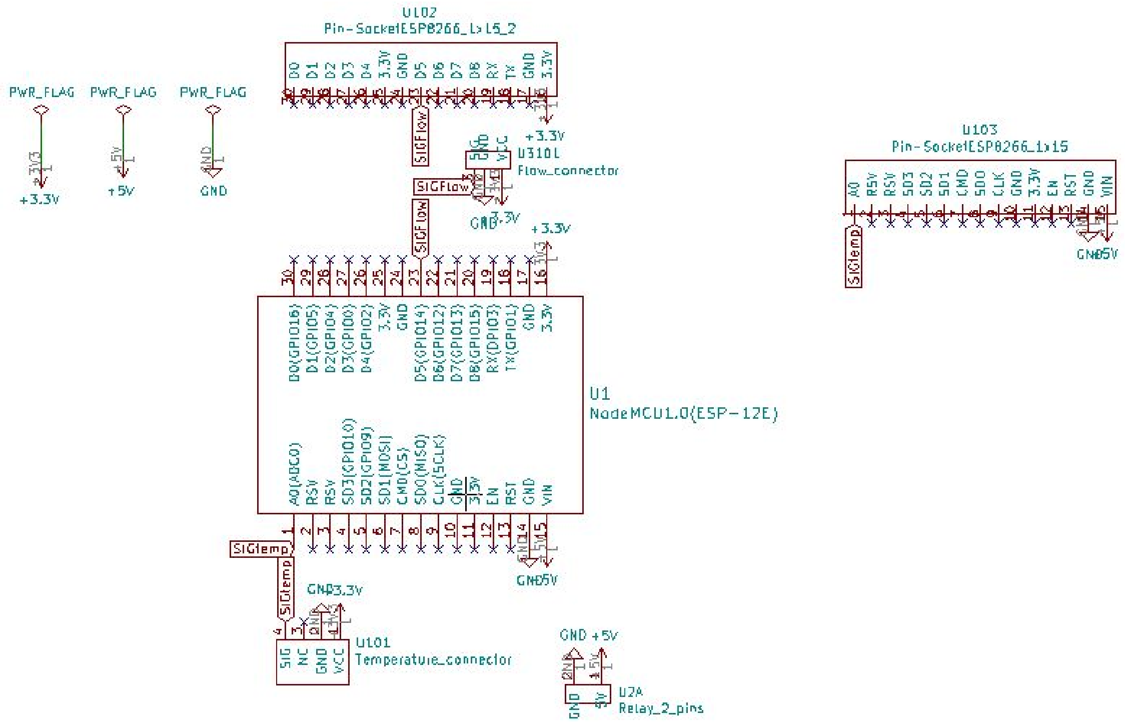

PWR_FLAG
PWR_FLAG
PWR_FLAG
+3.3V
+5V
GND
U102
Pin-SocketESP8266_1x15_2
D0 D1 D2 D3 D4 3.3V GND D5 D6 D7 D8 RX TX GND 3.3V
SIGFlow
U310L
Flow_connector
SIG GND VCC
+3.3V
GND
3.3V
U103
Pin-SocketESP8266_1x15
A0 RSV RSV SD3 SD2 SD1 CMD SD0 CLK GND 3.3V EN RST GND VIN
SIGtemp
GND
5V
U1
NodeMCU1.0(ESP-12E)
D0(GPIO16) D1(GPIO5) D2(GPIO4) D3(GPIO0) D4(GPIO2) 3.3V GND D5(GPIO14) D6(GPIO12) D7(GPIO13) D8(GPIO15) RX(GPIO3) TX(GPIO1) GND 3.3V
A0(ADC0) RSV RSV SD3(GPIO10) SD2(GPIO9) SD1(MOSI) CMD(CS) SD0(MISO) CLK(SCLK) GND 3.3V EN RST GND VIN
U101
Temperature_connector
SIG NC GND VCC
GND +5V
U2A
Relay_2_pins

Figure 1: Schema of the PCB

Once that was done, the next step consisted in designing the PCB footprints of components. We routed all the tracks to connect components on the board, following Kicad design rules. For example, the size of the tracks had to match the numbers shown on the table below.

	Isolation	Largeur Piste	Diamètre Via	Perçage Via	Diamètre µVia	Perçage µVia	Largeur Paire Diff	Dist Paire Diff
Default	0.4	0.8	1.6	0.8	0.3	0.1	0.2	0.25
Power	0.4	1	1.6	0.8	0.3	0.1	0.2	0.25

Furthermore, all the routing had to be concentrated on the same copper layer so that the PCB is easier and cheaper to make. To limit the track number on the board, we defined the board as the ground plan to connect all ground pads, i.e a copper layer which allows a component to be connected on the board.

On the figure below, we can see on the left, the routing schema and on the right, the routing showing the current and impact of the ground plan.

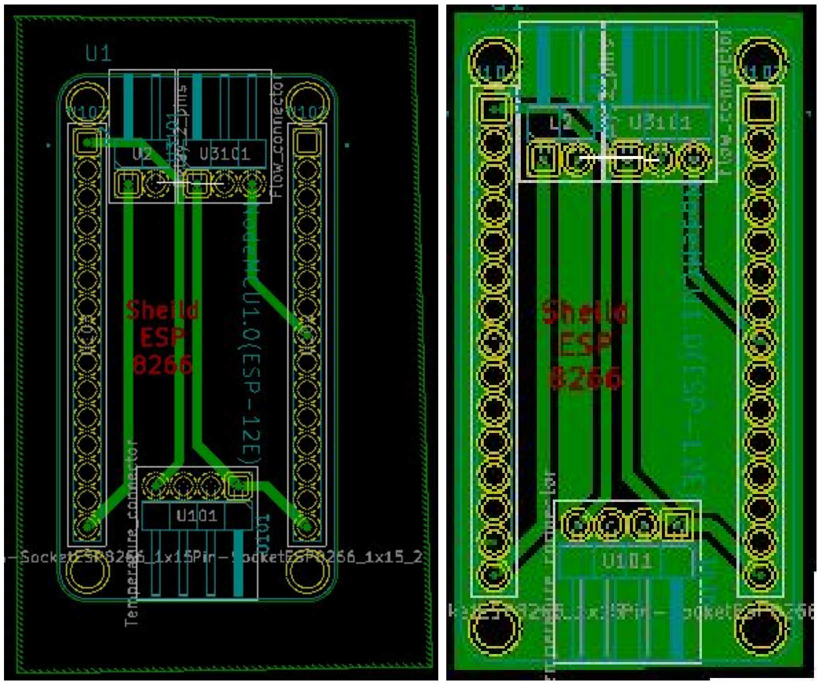


Figure 2: Description of the PCB routing

As a conclusion, here is a 3D view of the board. The Kicad project can be accessed on the "PCB" branch of our repository.

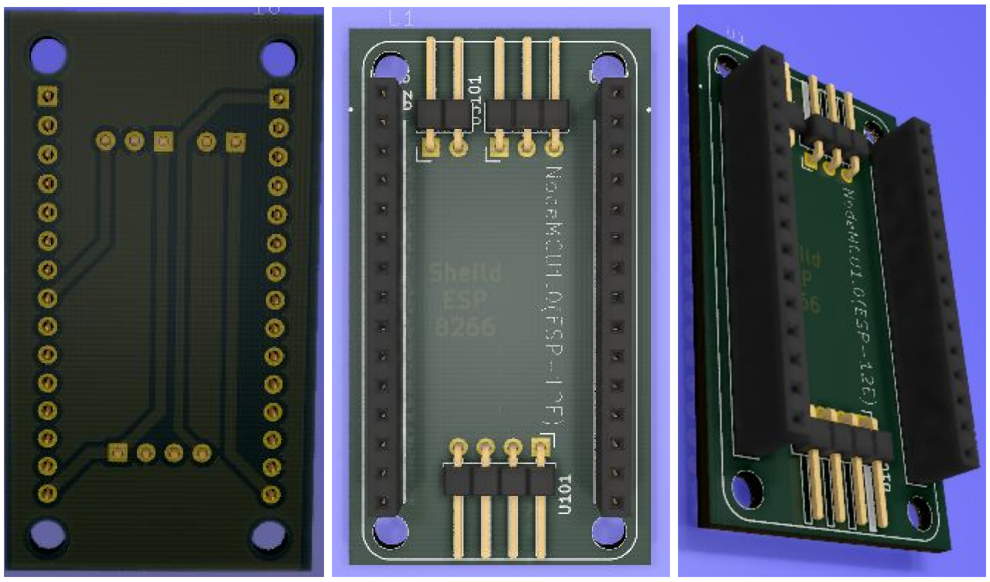


Figure 3: 3D view of the PCB groove

4) **Abstracts**

Abstract of Vincent Laurens

Lately, people have become increasingly concerned about the future of our planet. Ilya is a young company that is looking to develop two products that aim to save money and raise awareness about their water consumption mainly when they take a shower. We will present in detail a tool that allows people to measure the water and energy consumption of a shower in real time. We present a prototype based on a sensor system collecting data on water flow and quantity and storing it in a database in order to display it on a web interface. The data is captured by a node that must be autonomous and energy efficient. Data collected are then sent via Bluetooth Low Energy protocol to a local server to be stored in a database and then stored on the cloud. We expect to create a prototype that is not fully optimized but both simple and realistic that collects data in the shower and displays it on a web interface.

Abstract of Hamza Safri

Energy and water are intricately connected. This connection is obvious when using heated water for showers, washing hands, or doing the laundry. However, cold water also costs great energy consumption. In order to reduce risks associated with disruptions in water and energy, or increase of its cost, those elements should be monitored. This project presents a design of a low cost system for near real time monitoring of the water and energy using the potential of IoT (Internet of Things). The system has in one hand several sensors used to measure the water's Temperature and energy. On the other hand, it has a web interface to visualise the sensor data stored in a database.

Abstract of Mickaël Commérot

Nowadays, environmental issues are increasing more and more. The new trend is to recycle and reuse resources in order to avoid too much waste. Moreover, people tend to have an irresponsible use of water during showers especially in hotels or gyms. ILYA is a startup that aims to contribute to finding a solution to this environmental challenge. They propose an IoT solution that reduces energy and water consumption during a shower. The solution consists in measuring the water flow and temperature thanks to a connected device, and then in sending the collected data to an online application. This report presents this energy monitoring system, from its conception to its implementation and design.

Abstract of Karim Aldandachi

The effects of climate are more catastrophic every year. Fighting against it has become a common cause for all governments, organisations and populations. Companies