Òmnia
smart solutions only.

NOVIA
UNIVERSITY OF APPLIED SCIENCES

Development of sensors using Internet of Things

Connecting citizens through sensors

created by a multidisciplinary team

Supervisor

Lindén Hans

Authors

Elegeert Jeroen
Driout Eléa
Garcia Celia
López Eudald
Verbiest Annelien
Yerpes Borja

ACKNOWLEDGEMENTS

First of all, our team wants to thank the Novia University of Applied Sciences for giving students the opportunity to attend a European Project Semester. To start with the people involved in making this a successfully Erasmus semester abroad. Starting from the beginning course, the inspiring teambuilding sessions given by Roger Nylund, We also would like to thank him for arranging an introduction course about Additive Manufacturing and 3D printing. Besides, we would like to thank Mikael Ehrs for the Project Management courses, that helped us to manage holding a clear vision on the project together as a team, as well as our English teacher Hanna Kuusisto who gave us advice to write papers in a more academic way. Last but not least, our multidisciplinary team would like to thank Hans Lindén, who gave us useful and constructive feedback each week and supported us to persevere and keep improving and searching for solutions. And thanks again for the lessons our supervisor gave us and additional information about the possibilities in the IoT world, the Internet of Things.

ABSTRACT

The University of Novia receives foreign students every year in the EPS program, with the aim of integrating them in experimental and concrete project groups. The aim of these projects is to train students in teamwork on long-term projects, but above all to respond to a given problem. This may concern the physical, natural or health field. The aim is to change our daily lives through our work: we must be able to demonstrate the benefits of our project when it is completed.

In a rapidly evolving world, technology is playing an increasingly important role in our daily lives. We delve into the internet of things and look for solutions to certain problems.

The two months following the research phase were devoted to the concrete part of our project. That is to say, the programming of our 4 sensors, namely the water level sensor, the sea level sensor and the vibration and movement sensors.

We also worked on the marketing and design of our brand and our sensors, to have a complete and concrete project. All those points will be further developed thereafter.

In this final report, the IoT team will present these results to you, going through each step of the progress and any difficulties encountered.

TABLE OF CONTENTS

1 THE EPS PROJECT

The European Project Semester is a multicultural European Project that is formed for nineteen countries through Europe (Austria, Belgium, Finland, France, Spain, Germany, The Netherlands, Romania...)

There are some universities that bring the opportunity to do this program and get the credits, living an incredible experience in another country with different students that are living the same as you.

This project is focused on the students that have coursed 2 years of any type of engineering but there are some exceptions, likewise if you can support the project with some knowledge, you are always welcome.

Sweden
Finland
Norway
Estonia
Baltic Sea
Latvia
North Sea
Lithuania
United Kingdom
Denmark
Ireland
London
Berlin
Belarus
Poland
Warsaw
Germany
Belgium
Kyiv
Paris
Czechia
Ukraine
Slovakia
Austria
Moldova
France
Hungary
Romania
Croatia
Serbia
Italy
Rome
Bulgaria
Black Sea
Portugal
Madrid
Spain
Lisbon
Tyrrhenian Sea
Greece
Istanbul
Ankara
Turkey

The purpose of this project is to bring together engineers from various ethnicities and cultures in order to achieve common goals. Furthermore, this is a mix of academics and project development, all of which are aimed at providing us with the knowledge we need to better our talents and apply these strategies to our project.

In this program, between 3 and 6 people form a group consisting of a minimum of 3 nationalities. In some cases, you do some interdisciplinary tasks but also work with some companies that bring you some keys to get what you are trying to do, or this is made just with academic objectives.

The EPS helps engineers that are developing their skills to improve individually and in collective. The engineers that have participated in always get new keys and techniques to be more productive and perfectionist but also change the mind of all the students and they learn about human beings and cultures.

2 NOVIA UNIVERSITY

Novia University is the largest Swedish-speaking UAS in Finland. With traditions dating back to 1813, Novia offers multidisciplinary higher education with practical education. It's one of the best University of Finland and it works focused on training students to become experts and developers based on requirements of working life.

Novia's area of activity is located along the Finnish coast and the Universities are in Vaasa, Turku, Raasepori and Pietarsaari. In this case, our university is in Vaasa, located exactly at Palosaari, Wolffintie 33, 65200 Vaasa. It is the largest Swedish-speaking University of Applied Sciences in Finland with more than 4000 students and offers programs related to subjects as Health and Welfare, Technology and Seafaring, Culture, Bioeconomy and Business.

In addition, it has an extend network institutions inside the country and outside and their goal is to continue expand it, trying to work with the best and most recognized organizations to keep climbing to the top of the best universities. This activity brings the opportunity to all the students of Novia to become more competitive and prepared for the requirements of working life.

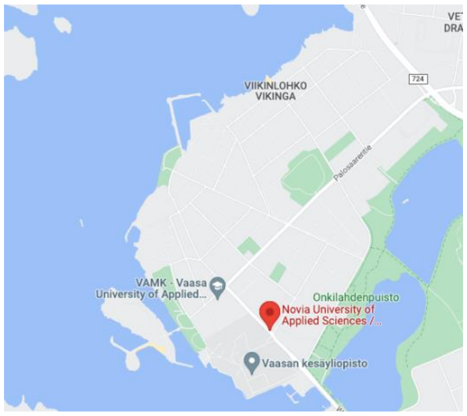

VIIKINLOHKO
VIKINGA
724
Palosaarentie
VAMK - Vaasa University of Applied...
Onkilahdenpuisto
Novia University of Applied Sciences /...
Vaasan kesäyliopisto

Figure 1.0 -Location Novia Google Maps

Figure 1.1 -Novia UAS Building

2.1 TECHNOBOTHNIA

Technobothnia has been the place where our team has really performed all the necessary tasks and expertise. Technobothnia is a sector of Novia University of Applied Sciences that focuses more on science & technology. Was founded in 1996, and its building is a renovated old cotton mill factory.

Among the objectives for which it was established is to increase cooperation between Vaasa technical education and local industries in the Vaasa region. It also aims to promote the technical education that schools can offer.

Figure 1- Photo of Technobothnia entry

These are the three universities that co-owned the resources and facilities provided by this building:

VAMK VAASAN AMMATTIKORKEAKOULU UNIVERSITY OF APPLIED SCIENCES Vaasan yliopisto UNIVERSITY OF VAASA YRKESHÖGSKOLAN NOVIA

Figure 2- Logo of Universities

In total, approximately 2,400 technical students from the three schools make use of this building, where research work and close interaction with local industries are also carried out.

With the help of the 3D printer, we were able to create and test the many cases of both the frequency sensor and the traffic counter sensor.

The use of the IoT lab was also really helpful for creating and soldering the wiring of the sensors. Also, Hans Linden’s desk was in this IoT Lab. We could always go to him with our questions and problems, and he would put us on the right track.

In the following image we can find a map of Technobothnia to be aware of the big dimensions and all the resources it has. The areas marked in red are those that have been particularly important for the development of our project.

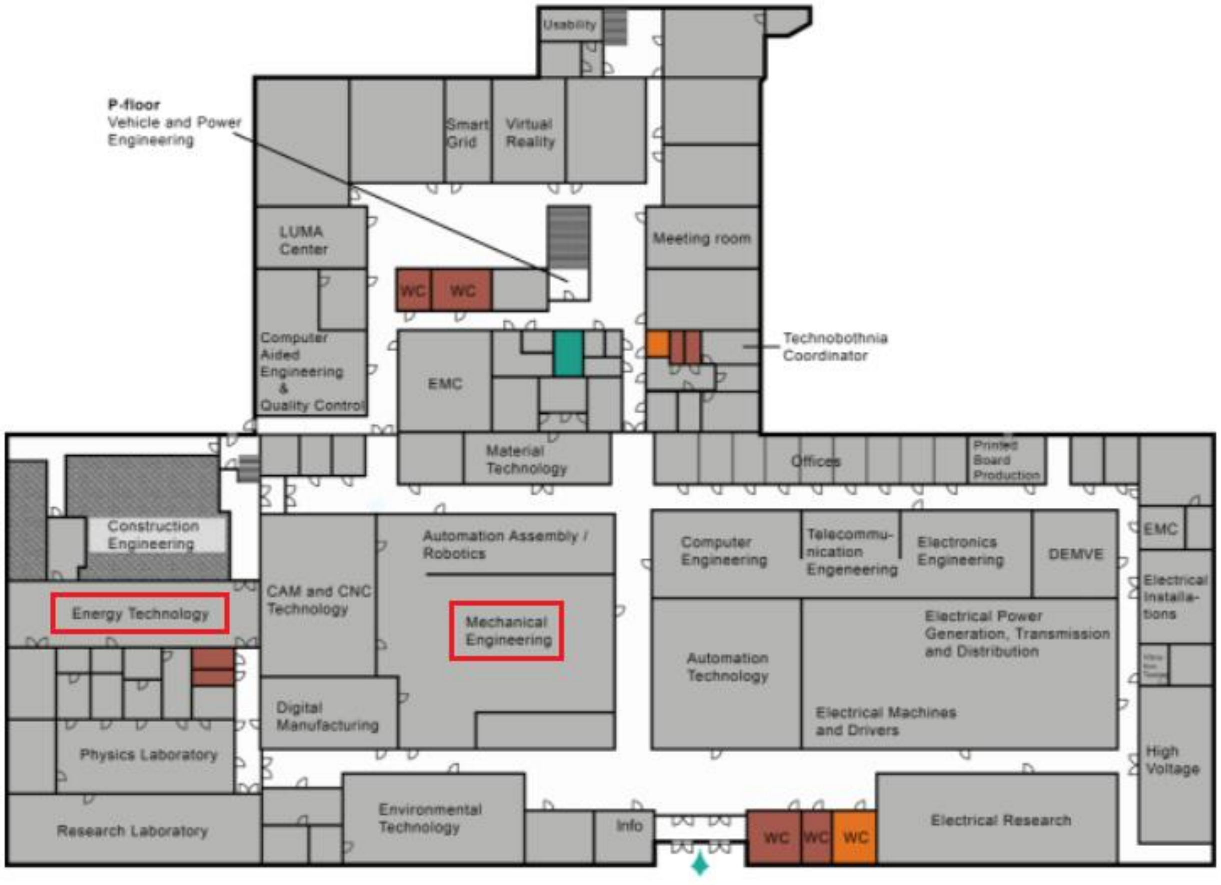


Figure 3- Map of Technobothnia

Generally, our project has been carried out in three areas:

- **The EPS room**

Here is where all the different lessons of EPS take place. This classroom is always booked for EPS students, to do what we need, all the teams are always working here writing the report or making some changes in the prototype that they are developing. The full equipment that this zone reunites, makes it the perfect place to work.

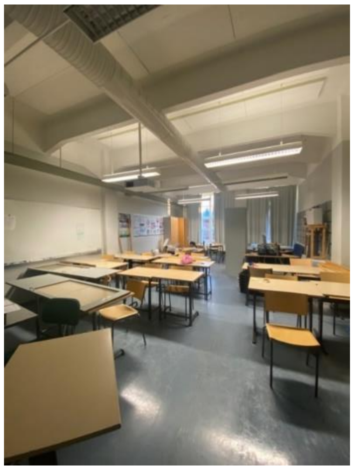


Figure 4- EPS Room

- **The Digital Manufacturing room**

The Digital Manufacturing room has been one of the key places for our prototype development. This is where all the 3D printers can be found. We used this when we needed to create a prototype that we've designed previously in some 3D software. For achieving our goal, we needed many chances to get all the cases fitting perfectly with the components that we wanted to implement in our devices and also for all the aesthetical changes.

This Laboratory has different 3D printers, and you can use the one that fits better with the material that you want to print. In our case, we developed everything with *Ultimaker 3D* printers in particular s3 and s5.

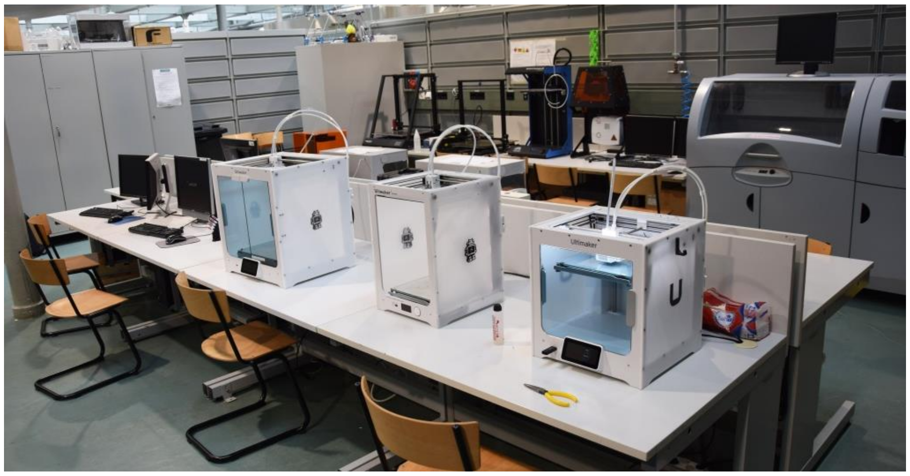


Figure 5- Digital manufacturing room

- **The Energy Technology Laboratory**

All the electric measurements that you need to do take place here. In this Laboratory you can find all the tools for develop the electrical devices. There are many computers to succeed with you coding or wire welders for create different electrical circuits.

3 INTRODUCTION OF THE TEAM

This multicultural team is formed by 6 members:

Eudald López Esteve – *Barcelona, Spain*

Student of Mechanical Engineering at UPC EPSEVG.

“I am Eudald, from Barcelona. I’m studying my last year of mechanical engineering. The reason that I’m doing this EPS is that I really like to challenge myself, and I thought that this would be one of the best ways to get knowledge and complete some personal goals. Also, I really enjoy meeting new people and discovering unfamiliar places and cultures and the EPS has all what I was looking for.

Figure 6- Image of Eudald

Otherwise, I am really interested in all the development of a new company and what steps you must follow to be more competitive in the market sector that you have chosen. Besides all the mechanical knowledge I also want to add some new background of different subjects that are not directly connected with my degree to be more complete in the future.”

Annelien Verbiest – *Antwerp, Belgium*

Student Product Development, Design Sciences at the university of Antwerp.

“Hi, my name is Annelien Verbiest. I’m studying Product design at the university of Antwerp in Belgium. I choose an EPS in Vaasa, because I think improving multidisciplinary teamwork skills are crucial in a rapidly changing diverse world.

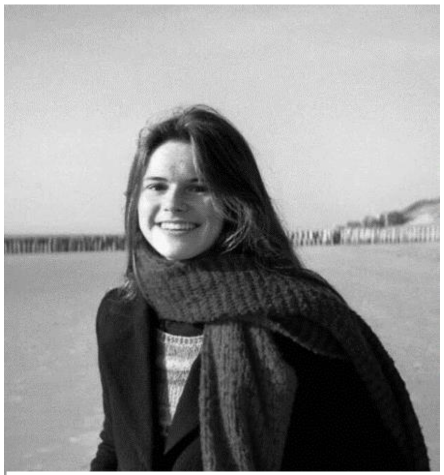


Figure 7- Image of Annelien

I strongly believes that in this current world intercultural cooperation and innovation are the keys to successfully create long-term solutions for the future. My passions are engineering and the combination of science and art.”

Celia García Freile – *Valladolid, Spain*

Student of Industrial Electronics and Automation Engineering at the University of Valladolid.

Figure 8- Image of Celia

"I am Celia García and I come from Spain. I study at the Technical School of Industrial Engineering in Valladolid. My field of study is based on designing electronic and automation equipment, developing control systems and optimizing industrial processes.

I applied for the EPS because I want to finish my degree setting myself a challenge and immerse myself in very different environment to the one I am used to. I think there is no better way to do it than in the EPS because in the work teams not only several fields of study are mixed, but also different nationalities and hence different cultures, backgrounds and ways of seeing things. I hope this EPS project will be a very enriching experience, which will make me to grow professionally and personally".

Jeroen Elegeert – *Antwerp, Belgium*

Student technical-commercial advisor in the field of electromechanics at the AP university of Antwerp.

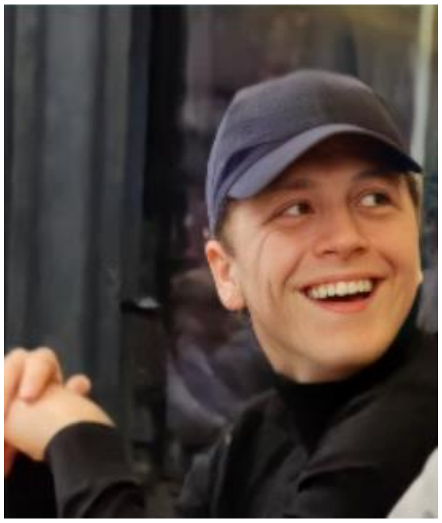


Figure 9- Image of Jeroen

"My name is Jeroen Elegeert and I am from Belgium. I study at the Artesis Plantijn University in Antwerp. The reason I wanted to do this EPS project is because I wanted to go out of my comfort zone. Every time I have done something out of my comfort zone in the past, it has been a great experience where I have learned a lot, both educationally and personally. Going to another country to work on a project with students from all over the world seemed like a wonderful challenge.

By carrying out this EPS project, I hope to gain more experience and knowledge both in the technical-commercial field as well as in other fields. In this way, I will increase my general knowledge. Which is very valuable on the labor market."

Borja Yerpes López – *Valencia, Spain*

Student of Industrial Electronics and Automation Engineering at the Valencia Polytechnic University.

"Hi, my name is Borja, and I am form Spain. I study electronics and automation in Valencia. The reason I wanted to do an EPS is because I think that it is a very interesting way of working. As **teams** are formed by students from different fields and countries, therefore from different cultures, it is a very broaden experience in different ways.

I believe that this EPS project is the best way to finish my bachelor, because I will be able work in the same way that it is done in the industry. So that, it will give me a new perspective within my field. Furthermore, I hope to develop my soft skills through team working, because working in a team requires more than just technical knowledge."

Figure 10- Image of Borja

Eléa Driout – *Reims, France*

Student in packaging engineering at the university of ESI in the city of Reims.

"I am Eléa Driout, a second-year engineering student at ESI Reims, France. I study packaging in all its forms: material science, design, machine operation, etc.

I decided to do a semester abroad because I think it is very important to open to the world. What better destination than Finland to get a change of scenery? This is how I arrived in Vaasa at the beginning of February 2022.

The IOT project is a brand-new field of application for me, and I hope to learn a lot because I have never had the opportunity to work on sensors. It promises to be a rewarding experience both personally and professionally."

Figure 11- Image of Eléa

4 PROJECT DEFINITION AND PLANNING

4.1 PROJECT DESCRIPTION

Novia University, in cooperation with Vaasa, wants to place several measuring sensors in Vaasa for different purposes. For example, they want a measuring sensor that can constantly display the water level, a measuring sensor that can measure the snow depth, a vibration sensor for detecting vibrations and a motion sensor for counting the cars coming in and going out of the city of Vaasa. Our task in this project is to compose/design the most suitable sensors that can successfully perform these measurements all year round. Our team carries out investigations, tests, meetings, and experiments to find the best solution to deliver the requested product.

4.2 OBJECTIVES

Our objective in this project is to successfully design and create all requested sensors so that these can be used to carry out the necessary measurements. In particular, the sensors to be developed are as follows:

- Sea water level sensor.
- Snow depth sensor
- Vibration and acoustic sensor with Arduino Nicla Sense ME.
- Car counter sensor.

By doing so, we have absorbed a lot of knowledge, and we have also helped the city of Vaasa with these measurement solutions.

Our main stakeholders are the university of NOVIA, the local companies here in Vaasa and other AI companies. With our solutions and gained knowledge they are able to do further research about the topic.

4.3 MISSION

The “mission” of Òmnia is to find the correct measurement methods for the requested measurements and in this way the right sensors for the given materials with the focus on IoT.

4.4 VISION

Our vision is to gain new insight in the IoT world for local companies for research on AI and IoT and this way help with delivering a final product that can be used for local companies.

4.5 VALUES

Our values are first of all working in a multidisciplinary teamwork with transparency and open communication. Thereby, empathy and curiosity are important values to improve team atmosphere. Finally, passion is a motivation to take on all the obstacles or the challenges this semester.

4.6 ORGANIZATION OF THE PROJECT TEAM

For the organization of the project team, we divide the workload and divide our team in two smaller groups, each contains three persons. That’s the reason that clear communication and being transparent is important. In the weekly meetings we have a chairman who is the main leader of the meeting to make it more efficient, then there is also a secretary who writes the important things and keeps a logbook. After the meeting are questions and issues are solved and our supervisor gives us some feedback to improve our report each time. By working on achievable deadlines, we keep structure during the progress of the project. Step by step and together we improve every week.

5 RESEARCH

5.1 DEFINITION AND OPERATION OF SENSORS

This project is all about researching and finding the right measurement methods by using sensors. In order for this research to run smoothly, we must of course first understand what a sensor is and how it works.
A sensor is a device which can detect or measure a physical property and record, indicate, or respond to it. The sensing element reacts to the force or pressure of the process. This creates an output signal that can be interpreted by a readout device or a data- collection device.

5.2 CLASSIFICATION OF SENSORS

Our world is full of sensors. In houses, offices, cars, on the street the collected data of sensors gives us information to improving our quality of life, making our lives easier or the surroundings safer. Sensors are applied in different surroundings. To make it more easy sensors can be ranked in diverse classification systems. Here is a way to classify the variety of sensors.

- ***Active and Passive sensors***: Active or parametric sensors, these sensors need an external power source in order to operate. (f.e. GPS and radar sensors). The passive sensors are known as self-generated sensors, they can provide or generate their own electric signal. (Examples are thermal sensors, electric field sensing and metal detecting)
- ***Contact and Non-Contact Sensors:*** on one hand there is the contact sensor, that require contact and a stimulus. (f.e. temperature and strain gauge sensors). On the other hand, the non-contact sensors, they require no physical contact (optical, magnetic sensors and infrared thermometers).
- ***Absolute and Relative Sensors:*** The first one reads the stimulus in an absolute way. An example is the thermistor, which measures the exact or absolute temperature. The second one reads the measurements in a fixed or relative way. The measurement of the temperature difference measured by the thermocouple is an example.
- ***Analog and Digital Sensors:*** In the case of the analog sensor, it produces analog output. (Accelerometers, pressure sensors, light and sound sensors). By contrast,

the digital, electronic or electrochemical sensor transmits data digitally. (f.e. digital accelerometer, pressure and temperature sensors).

- ***Miscellaneous Sensors:*** This category include all the other types of sensors, like electric, biological, chemical, radioactive and more.

In conclusion, the classification of sensors makes it more clear what kind of sensors already exist. In general, the classification of sensors add structure to the list of sensors to classify them in logical groups.

5.3 APPLICATIONS

Research firm IDC, in its "Worldwide Internet of Things Spending Guide", predicted double-digit growth digit rate in 2021, with a compound annual growth rate of 11.3% from 2020 to 2024.

This means that day by day all the IOT devices are substituting all the market of the automatic devices, because of the pros that it includes. Statista predicted there will be 75 billion IoT devices in use by 2023.

Currently, we can divide all the IoT devices in four big groups of uses:

- ***Healthcare, manufacturing, retail***: The automatic devices here, can define some information about tracking valuable equipment and tools remotely via sensors and let the company know all the available stock of specific product.
- ***Cities, construction, manufacturing, oil and gas:*** Monitor conditions in facilities, cities on worksites using different technologies like drones we can know exactly how everything is going in different sites at the same time. It can be really helpful to know the condition of some changes for example, monitoring a different method of work for the employees.
- ***Logistics, retail and transportation:*** Supply chain and logistic efficiencies using different IoT devices like robots or drones to be able to do different tasks at the same time, optimizing the timing for each product.
- ***Healthcare, manufacturing, telecoms… :*** With the remote support, setting up different sensors in the critical parts of the engines and machines these devices are able to call the specialists and technicians when a revision is needed to augment the quality and prevent machine breakdowns.

As we said before, we can find this device almost everywhere some specific examples are: Industrial, agricultural and commercial management, smart cities, smart buildings/homes, traffic management, connected vehicles, hospitals…

5.4 INTRODUCTION TO THE INTERNET OF THINGS

Internet of Things is a technology that has been developed during the last years. This is a system of interrelated computing devices, mechanical and digital machines, objects, people and so on, who have the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction, although people can interact with these devices for example for setting them up. The goal of this technology is to improve the wireless communication between humans and the objects that we use every day, with the goal of creating a more connected society in which people can access to every data they want.

An IoT ecosystem consists of web-enabled smart devices which use embedded system (processors, sensors, communication hardware) to collect, send and act on the data they acquire from their environments. IoT technology has been used in many fields as smart homes, smart cities but also in others as health, logistics and industry, and almost everywhere. Some examples of internet of things are a farm animal with biochip transponder, a person with a heart monitor implant or just a man-made object which is able to transfer data over a network.

As the world continues to become a more connected place, this technology still growing up and developing each year, being a part of the 17 technologies explored in the Industry 4.0, at the same level as other as 5G, Artificial Intelligence or Blockchain among others. In fact, the Internet of things is the key enabling technology in the Industry 4.0, because the application of broadly used IoT technologies will enable a system more integrated and will bring down the hardware, software and labor cost.

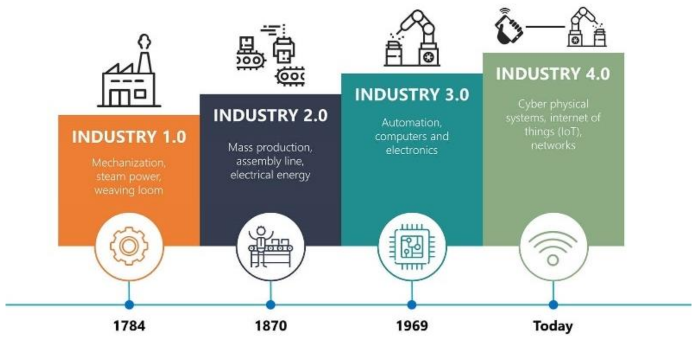

INDUSTRY 1.0
Mechanization, steam power, weaving loom
INDUSTRY 2.0
Mass production, assembly line, electrical energy
INDUSTRY 3.0
Automation, computers and electronics
INDUSTRY 4.0
Cyber physical systems, internet of things (IoT), networks
1784
1870
1969
Today

Figure 12- Summary types of industries

5.5 IOT IN SENSORS

A new era has arrived with the arrival of Industry 4.0 and the Internet of Things. Devices may now communicate with one another, allowing us to work with them remotely. This can be applied to a variety of industries, as well as the building of smart cities and even smart homes. Smartwatches, which have become increasingly popular in recent years, are one example of IoT devices. Thanks to devices like these, we can collect data that can be helpful in many ways, and that is why sensors are an important part of the IoT.

As there are many kinds of sensors and they can be used in several ways, there are different options to connect these devices to the internet that varies in some of their characteristics.

First, we have some well-known options of IoT connectivity: Bluetooth and Wi-Fi. These options are characterized by having a high-speed signal of some Mbps ~ Gbps but a short distance range between 10- and 1500-meters Bluetooth and between 15- and 100-meters Wi-Fi. Both Bluetooth and Wi-Fi are the most used options in IoT by far, and are mainly used in computers and smartphones, but also in smart home’s environments.

Figure 13- Types of connection

In second place, there are some options that also have a short distance range, but with a medium-speed signal. However, these networks count with the advantage of having a low power consumption. Some examples are Z-Wave and ZigBee and are mainly used in smart homes.

Then, we can find options with a longer distance range up to 10 kilometers, and a signal around 200 kbps to 1 Mbps. These are known as the cellular networks and the main technologies are LTE-M and NB-IoT. The main difference between them is that NB-IoT is a bit slower, but it also has a lower consumption level that can make it more interesting in some cases. These networks are often used in the industry, logistics and smart cities.

Finally, there are also some long-range networks with low consumption power but low-speed signal. Some examples can be Lora-Wan and Sigfox and are mainly used in smart cities and for rural porpoises as it is needed this long range of many kilometers due to the large distances.

Figure 14 Types of connection

As this project consists in the development of different sensors, we will use several IoT technologies based on the properties of the sensors we are developing for this project. First, two of our sensors will be used in the Vaasa area, so we'll need a wireless alternative with at least a few kilometers of range, and because the sensors will be placed outdoors, we'll need to make them battery powered. As a result, it's also critical to adopt a low-consumption technology because we'll be extending the battery's loading cycle. We chose the NB-IoT technology to develop a solution based on these standards since it meets all of our requirements: the distance range is up to 10 kilometers, it has a low consumption level, and it is fast enough to retrieve the data we require from the sensors.

The other two sensors will be placed on campus afterwards. One will be battery-powered yet easy to charge, while the other will be tethered. We will utilize Wi-Fi technology to implement the communication because the battery will not be a problem and they are close. There is already a Wi-Fi network that is operational on campus, it is the fastest alternative, and the battery will not be a problem in this situation.

5.6 SENSORS INVOLVED IN THIS PROJECT

In this part of our report, we will discuss the research, findings and final conclusions of all our sensors.

First, the vibration sensor, what the exact purpose of this sensor is and what the team's final idea is to realize this sensor.

Next, we discuss the Car Counter. We explain the purpose and function of the car counter. After this, we will discuss and compare all possible ways of traffic counting and make our final decision which method to use.

We also discuss the Sea Water Level sensor. We will discuss and compare all possible methods that are used today to measure water level. After this we make a decision which sensor is the most suitable for our project.

Finally, we will discuss the snow depth sensor. We will give general information about the purpose of this sensor and which components we will use to realize this sensor.

Vibration sensor

What was proposed to us was to develop an acoustic and vibration sensor, which is a sensor that can detect and measure frequencies. It can be used, for example, to measure the speed at which a particular machine is running, or whether the machine is properly balanced. A vibration sensor can also be used to pick up sound (e.g., a microphone) and record it.

The reason why the development this type of sensor is very interesting is because of the usefulness of vibration analysis. This consists in the process of measuring the vibration levels and frequencies of industrial machinery and using that information to determine the "health" of the machine and its components. By doing this process it is possible to identify, predict and prevent failures in machinery, which improves its efficiency and reduces downtime by eliminating mechanical or electrical failures. For example, some of the most typical failures identified by vibration analysis are improper installation, machining errors, insufficient lubrication, misalignment of shafts or pulleys, loose bolts, bent shafts and so on. In most cases, it can detect these problems long before the damage can be seen by the maintenance technician, and long before it damages other machine components.

In this way, vibration analysis procedures are now used in all parts of industry to identify machinery faults, plan machinery repairs and keep machinery running for as long as possible without failure. These procedures can also be applied to personal machinery to ensure that the service life of our machines and appliances is as long as possible.

Generally, the devices used to measure vibrations are accelerometers. These are devices that provide the ability to measure and analyze linear and angular acceleration. Accelerometers are used in static gravitational acceleration measurements, allowing to determine the angle of deviation of the measured object from vertical, as well as in dynamic acceleration measurements due to shock, motion, impact or vibration. Its operation is based on the fact that it generates a electric signal proportional to the acceleration of the vibrating device to which the accelerometer is connected.

The next step is transmitting the acceleration signal produced by the accelerometer to the vibration analyzer which, in turn, converts the signal into a velocity signal. Depending on the intended use of this signal, it can be displayed as a velocity waveform or a velocity spectrum, which is derived from a velocity waveform by a mathematical calculation known as the Fast Fourier Transform or FFT.

In the following diagram is a very simple graphical representation of how vibration data is acquired:

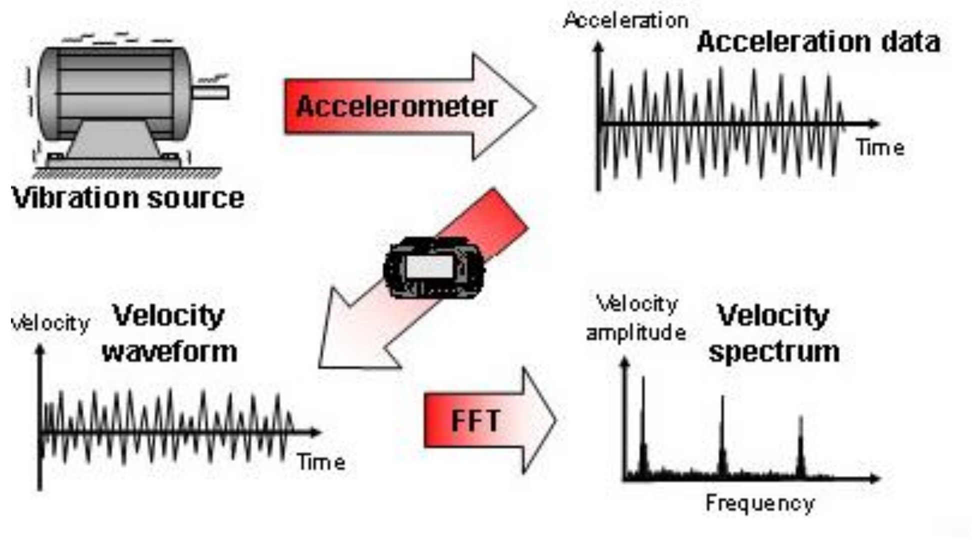


Figure 15- How vibration data is acquired

In our case, it will suffice to take the acceleration and time measurements and send them. The subsequent transformation to velocity, either as a waveform or as a spectrum, is left as a possible continuation of this project.

One of the options for developing the vibration sensor was to do it with the **Arduino Nicla Sense ME** which has the BHI260AP sensor created by Bosch Sensortec. This sensor is a self-learning AI smart sensor with integrated accelerometer and gyroscope which will allow us to measure vibrations through acceleration.

After doing further research on this sensor, we concluded, thanks to the potential qualities of this device, that it would be interesting to build a more complex device. This consists of a device with two functionalities. On the one hand, its normal function is as an environmental sensor which allows us to obtain information about some aspects such as temperature, humidity or air quality. These values will be updated every 5 minutes as they are values that vary very little over time, it is not necessary to carry out a continuous study of them, it can be done periodically. The other mode of operation is that by pressing a button, this device collects for about 10 seconds the vibrations of any machine on which it is placed. This information will be sent to a server for further processing and use.

Car Counter

Vehicle counters are a device that is used in generally urban areas to monitor the traffic volume of cars, vans and trucks and thus provide concrete data, for example, on the use of secondary roads to avoid tolls or on the effectiveness of traffic restrictions.

They are situated mostly along a particular road, but also can be installed on a path or in an intersection. Generally, the vehicle counters are automatic, which means that it doesn't need to be constantly operated by a person; it is, nevertheless, also possible to find manually counters which consist of observes who record traffic with a hand-held electronic device.

Vehicle or traffic counts can be quite useful for the authorities, as it allows them to control the people arriving and leaving the city as well as which routes are the busiest ones. This has been one of the reasons behind the idea of creating a vehicle counter.

The first step was to investigate the different types of technologies used in these devices. The technology used can be classified into two main groups, and two further groups can be also considered. This classification is made based on the impact on the road pavement during installation, use and maintenance. We can then distinguish between **intrusive sensors**, which are required to be installed directly on the pavement surface, in cuts or holes of the road surface, by tunneling under the surface or anchoring directly to the pavement surface, and the non-intrusive sensor which are mounted either above the road or on one of the sides.

The following diagram shows a classification of the different systems that can be found today in vehicle counters. Later we will look in more detail at some of the most commonly used ones.

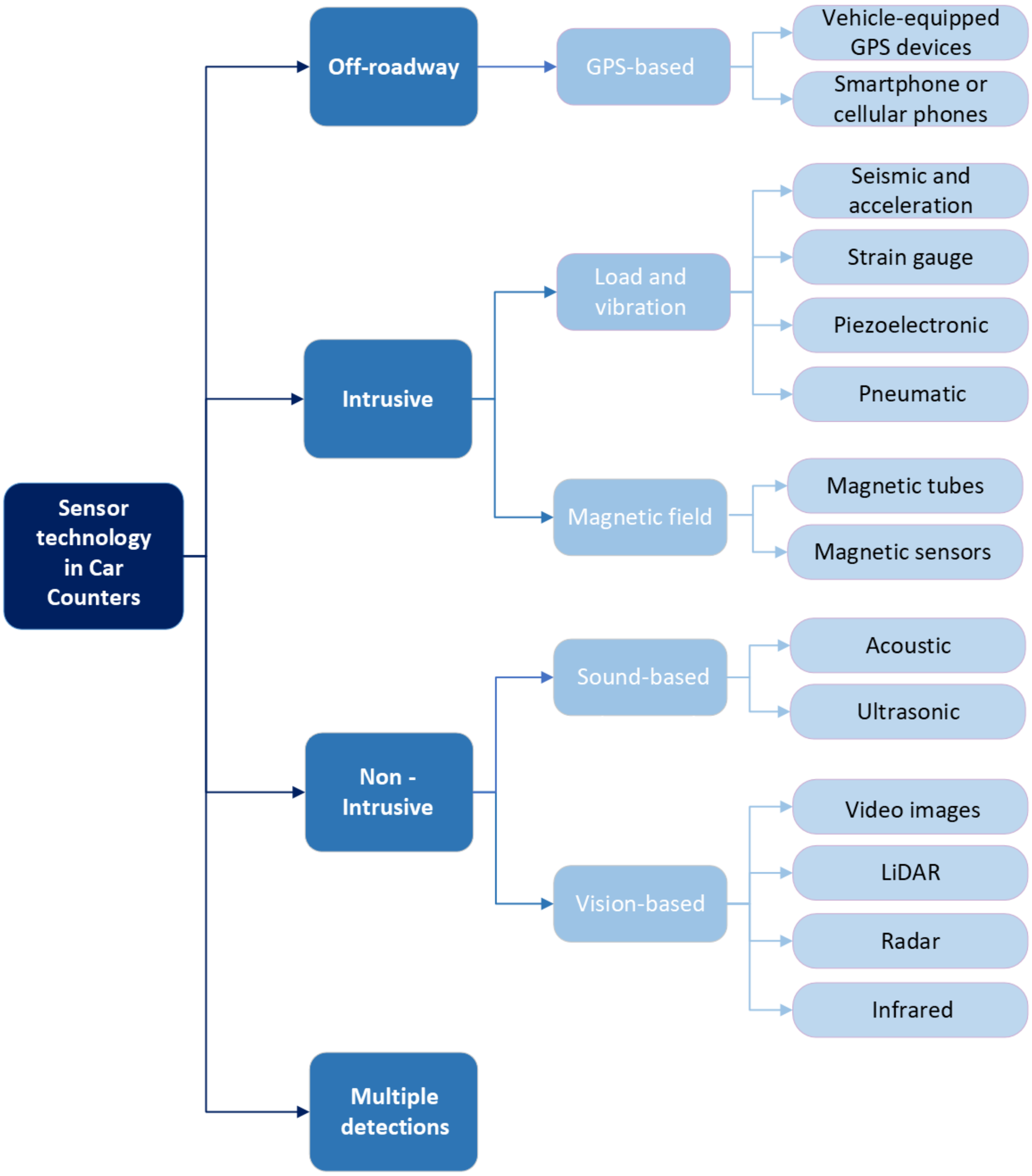


Table 1- Types of sensors technology in car counters

Intrusive sensors:

As mentioned above, this type of sensors is installed on the road, either burying them or anchoring them to the asphalt. Their measurements are precise and generally provide better-quality data, but the cost of their installation is quite high and their working life depends on the surface they are installed. Traditional intrusive sensors include the inductive loop, magnetic, magnetometer, pneumatic and pressure sensitive.

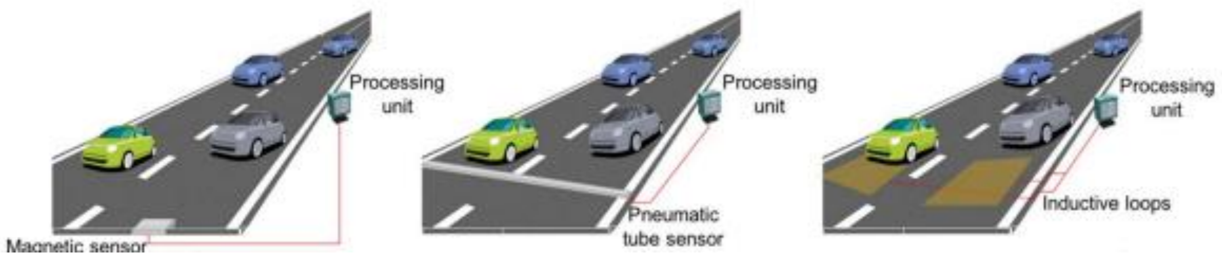


Figure 16- Example of intrusive sensors

▪ Inductive detector Loops (IDL)

An inductive loop consists of a coated wire coils buried in grooves cut in the road surface. It works with the principle of magnetic field introduced near an electrical conductor that causes an electrical current to be induced so it detects the metal mass of the vehicle as it approaches the sensor.

The large metal vehicle acts as the magnetic field and the inductive loop as the electric conductor. The data Collector is the one who records all the electric impulses and recognizes as a vehicle passing through.

This type of technology allows also to measure the speed and the length of a vehicle by calculating the time when the first loop is activated, and the second loop is disabled. We know the distance between those two points, meaning we also know the speed. Speed and time give us the length of the vehicle.

It is a technology that has been in operation for many years and is quite inexpensive considering the installation costs. In fact, they form the backbone detector network for most traffic control systems.

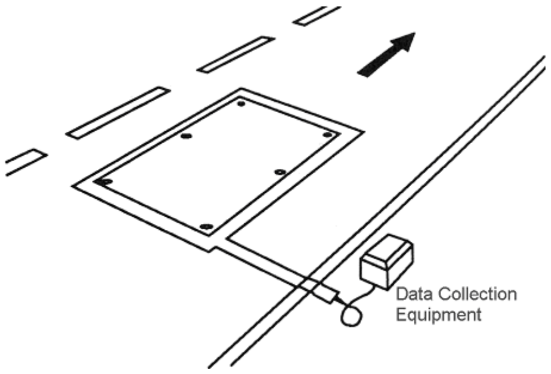


Figure 17- Inductive detector loops design

▪ Pneumatic tubes

This method consists of a pressure sensor that measures the difference between two points of a rubber tube. Once the vehicle passes through this bar, it pulses some air that must be converted into an electrical sign. The pneumatic road tube sensor is portable, using lead-acid, gel, or other rechargeable batteries as a power source.

If only one road tube is used only the vehicle count can be performed, however if a second tube is added then it is possible to know exactly the velocity, the volume, or the type of vehicle.

Road Tube holds up well in sunlight and in all weather conditions except they have a particular aversion to street sweepers and snowplows. The life span of road tube varies depending on the location, installation, and volume of traffic. However, these systems are deployed generally on a temporary basis due to the fragile nature of tubes, which are easily damaged or torn up by heavy or fast-moving vehicles therefore is commonly used for short-term traffic counting.

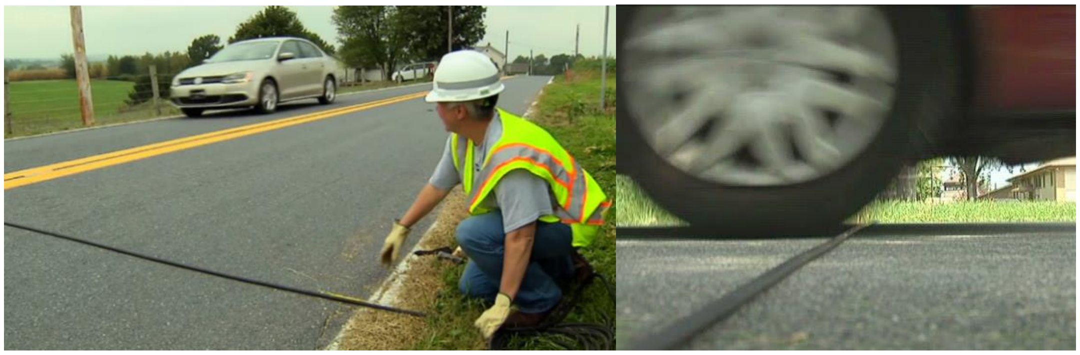


Figure 18- Image of pneumatic tube sensor working

▪ Piezoelectric Sensor (Weigh-In-Motion)

Piezoelectric sensors collect data by converting mechanical energy into electrical energy. When used to count vehicles, the sensor is mounted in the groove cut of the road's surface. When a car drives over the piezoelectric sensor, it squeezes the piezoelectric wire and causes an electric potential – a voltage signal. The size of the signal is proportional to the degree of deformation. When the car moves off, the voltage reverses (negative signal).

The counting device itself, which is connected to the sensors, is housed in an enclosure by the side of the road. Data may be collected locally via an Ethernet or RS232 connection to a laptop, or it can be sent over a mobile phone network.

Counting vehicles with piezoelectric sensors has the disadvantage that the road needs to be closed during the installation. The units can generally only monitor one or two lanes of traffic at a time. In fact, these systems are relatively rare and are used in specific locations for enforcement or access control. They are usually coupled with other systems, either intrusive or non-intrusive, to provide additional cross-checks on collected data.

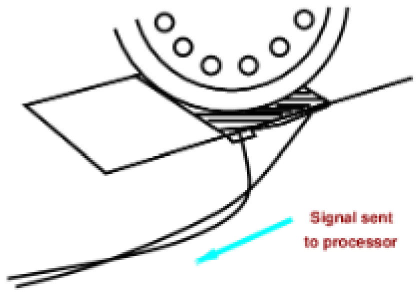


Figure 19- Example of piezoelectric sensor

Figure 20- image of how piezoelectric sensor works

▪ Magnetometers

Is a device that usually is buried on the road or at one side and it monitors the fluctuations in the relative strength of the Earth's magnetic field. The passage of a metal object, in this case a car, will cause a change in the magnetic field that will be detected by the magnetometer.

As in the previous case, the use of a single magnetometer only allows the counting of vehicles, but the use of two magnetometers at the same time allows measure flow, occupancy, vehicle length, and speed

One of the worst things about this method is that it has some difficulties when two vehicles pass the magnetometers together. Because both vehicles pass on the same time, the magnetic field will interpret it as only one vehicle passing.

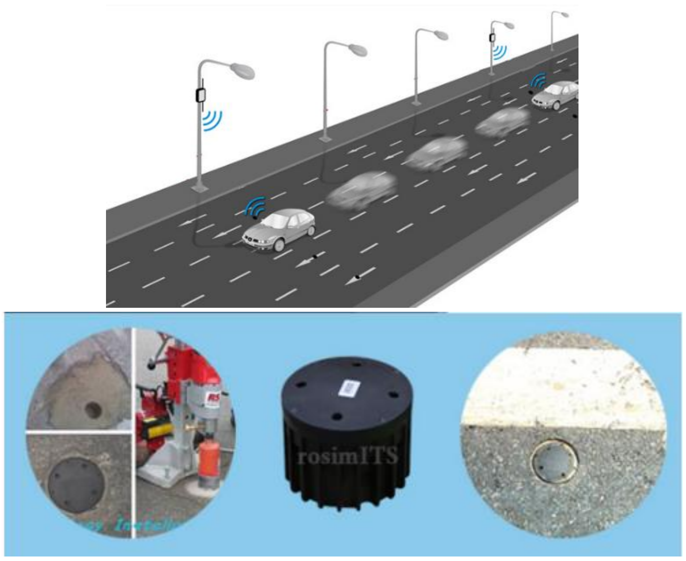


Figure 21- Example of magnetometers

Non-intrusive sensors:

Although intrusive technologies are still more widely used today because non-intrusive technologies are not yet well established, non-intrusive technologies are rapidly evolving and becoming more reliable and easier to deploy and use.

They have a clear advantage compared to intrusive one and it is that they minimize the traffic interference and reduce the safety risk. In addition, in some places such as bridges or tunnels where pavement cutting and boring are undesirable, non-intrusive sensors present a suitable alternative.

They can supply a wider set of information, but they are also more complex, have high power requirements and their performance may deteriorate in adverse weather conditions. Some examples: video cameras, microwave radar, laser or ultrasound.

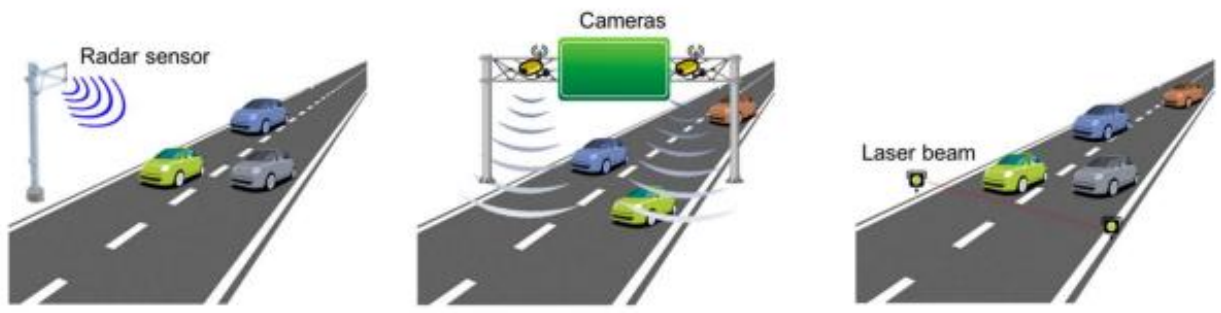


Figure 22- Differences between sensors

▪ Video vehicle detection

This process works on the principle of manual counting, just as the human eye can see the cars, so can the video sensor. The software analyses the video recorded by cameras specifically set up for the purpose.

This method is a good alternative where the number of vehicles is light and maximum accuracy in counting is not required. Unlike the intrusive methods, which are known for their good accuracy, it is different here, as the software can fail, but also other external conditions affect it. For example, the angle of the camera is important, if it is too excessive the perspective distortions can cause the software to interpret the image of two short vehicles as a long vehicle. Another external fact that can affect is the weather conditions because for example raindrops, in our case snow or condensation on the lens would affect the quality of the images, and video detection is only possible if the images are clear.

Despite all this, it also has many advantages over the traditional methods. It is much more cost effective, not only because it is much cheaper to install, but also because it can count in many directions at once and only one camera is needed for several lanes or exits at a junction. It is also easy to add or change the zones where vehicles are counted from a laptop computer.

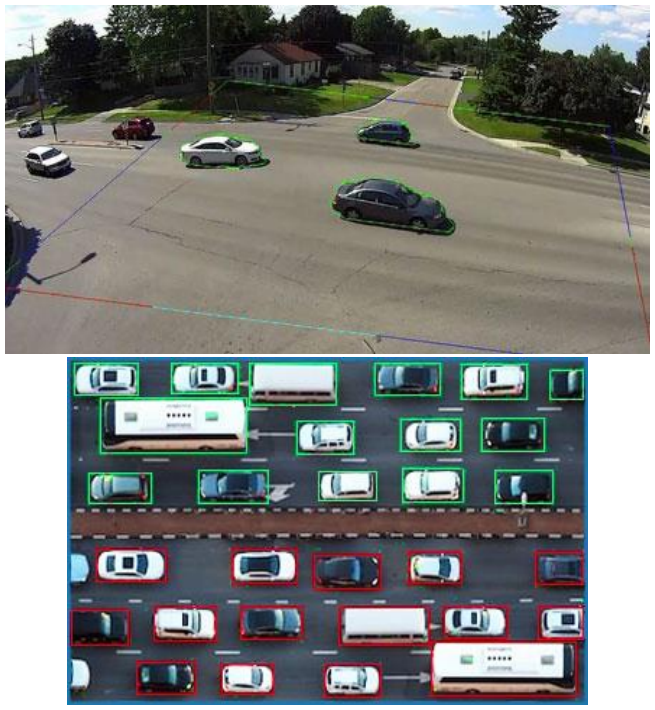


Figure 23- View of car counter sensor

▪ Ultrasound sensor

It can detect the number of vehicles passing through its wave front. For instance, one possible way of using it, is counting the vehicles that pass through its wave for 8 seconds. The following situations should be considered because they can lead to failures using this sensor:

- **Overlap between vehicles:** This effect happens when two vehicles pass at the same time in two different lanes of the same road which is controlled by a single sensor. So, the length of one vehicle hides the other vehicle.
- **No detection or overlap due to vehicle speed**: If the vehicle is going at a low speed or is stopped, it can be counted multiple times. The opposite can also be the case, if it is going too fast it may not be counted.

- **Initial recalibration:** The ultrasound sensor is calibrated regarding the surface that is in front of it. If this is modified, for example if a car is parked, sensor measurements will not be useful.

In addition, if the monitored road contains two or more lanes it will be necessary to add one sensor per lane. Despite their drawbacks, there are several workable solutions, which can make them an incredibly good option. For instance, it could be possible to place it instead of at one side, at the top. Or it could also be combined with a laser sensor.

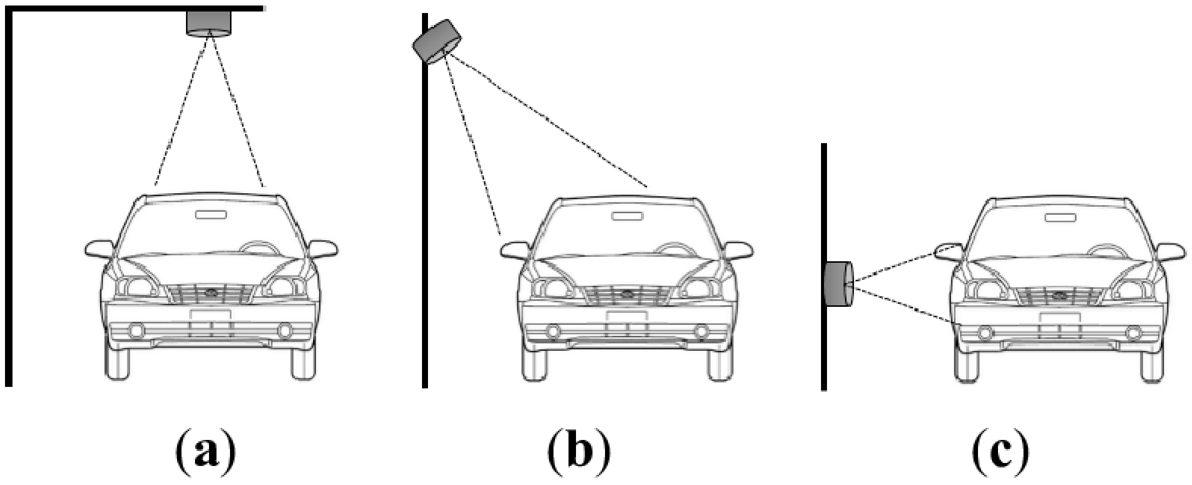


Figure 24 Mount positions of an ultrasonic sensor

Acoustic sensor

Acoustic sensors are one of the least used in terms of car counters, however, they are a very interesting alternative that we wanted to add to the most commonly used ones. This method is based on the analysis of the signals collected by two microphones placed on the road. That is to say, these sensors do not emit signals but analyses the sound waves emitted by the vehicles, mostly from the tires and the engine.

There are many drawbacks to this method, and its inaccuracy is quite considerable. However, as work continues to develop and improve detection algorithms, it is likely that these types of sensors will become one of the most widely used in the future.

Reference	Intrusive/ non-intrusive	Price	Advantage	disadvantage
Pneumatic/Road tubes	Intrusive	€ 6.000	Good accuracy and precision	Vulnerable for streetsweepers, snowplows
Piezoelectric Sensor	Intrusive	€ 40.000	Wireless data transfer	Road needs to be closed during installation/maintenance
Inductive Loop	Intrusive	€ 33.600	Accurate measurements	Road need to be closed during installation/maintenance
Magnetometers	Intrusive	€ 3.080	Easy installation	Difficulties when 2 vehicles pass together
Video vehicle detection	Non- Intrusive	€ 230	Clear overview and accurate counting	difficulties with authority over camera placement
Ultrasound sensor	Non- Intrusive	€ 180	Low energy consumption	Not accurate on its own
Acoustic sensor	Non- Intrusive		simple	inaccuracy

Conclusion:

After all the information gathered from the study on existing traffic counting technologies and given our economic and temporal situation, it has been decided to use one of the non-intrusive methods.

The next question was which of them might be the most interest to us in order to build our objective. The decision was between using either an ultrasonic sensor or video detection. Both have various advantages and disadvantages. On one hand the technology using ultrasound is quite simple while the technology using the video detector is quite complex and any member of the team has worked with it before. Also, the installation and powering of the ultrasound is much more complex as it would have to be placed on a road, whereas the camera could be placed from a side of the road, for example from a window of a building.

Finally, the fact that this sensor, unlike the other three, has been proposed to us more for experimental purposes than to build a functional sensor, we have decided to use video detection is a great learning opportunity for us because it will allow us to get started in the world of artificial vision and object detection.

Snow depth sensor

Finland is one of the coldest countries in the world and for this very reason it is also one of the countries with more snow. It can be very beautiful, but it can also be disturbing as it can cause some troubles in the city. Therefore, it can be interesting to know the snow level to know when to do a snow removal. Our goal is to measure the snow level by using data of sensors and transmit them wireless via IoT to Novia database or other devices that could need it.

There are more reasons why snow depth sensors are necessary, first the human safety, second to do (weather) predictions and third to gain data to gain more insights about the collected information.

There are lots of commercially available sensors for measuring the snow depth though most of them are based mainly on two concrete technologies, by using laser or ultrasonic sensors. These two types of sensors work in a similar way as they both emit a wave (a light wave in laser sensors and a sound wave in ultrasound sensors) and by calculating the time it takes the wave to rebound and come back to the sensor it is possible to know what the distance is. For this reason, we are going to test both laser and ultrasonic sensors and determine with this testing which of these sensors fits better for us.

▪ Laser: TF-Luna LiDAR Module

The TF-Luna LiDAR Module is a low-cost laser sensor with a high-performance level. It has a high-resolution level, with a high sensitivity and accuracy as well. It is also very light and small, so it is easy to work with. His operating range measures up to 8 meters, so it will cover our needs. Besides this, it is very easy to use, so it is a good option to solve this problem.

Operating range	0.2~8 m
Accuracy	±6 cm @ 0.2-3 m
	±2 % @ 3-8 m
Range resolution	1 cm
FOV	2°
Frame rate	1~250 Hz
Communication	UART / SPI

Figure 25- TF- Luna LiDAR Module

- ### Ultrasound: ANGEEK JSN-SR04T Ultrasonic Range Module

This ANGEEK ultrasonic device is an easy-to-use sensor that is ready to work with Arduino, so it is very comfortable to work with it. It measures up to 4.5 meters with a high resolution. It also has a good price-quality ratio, so it can also be a good choice.

Operating range	0.25~4.5 m
Range resolution	2 mm
FOV	45º~75º
Frame rate	40 kHz
Communication	I2C

Figure 26- ANGEEK JSN-SR04T Ultrasonic Range Module

Sea water level sensor

Contrary to the countries that are closer to the equator, the sea level in Finland depends mostly on the winds and not that much on the moon's effect on tides. In order to study the wind effect on the sea level we will design a battery-powered water level sensor that uses IoT. This will be helpful as will help to predict the level of the sea in the future and enabling anybody to know the real time water level and see the records of measurements. As in the previous case, our goal is to take measurements of the water level by using data of sensors and transmit them wireless via IoT to Novia database or other devices that could need it.

We can find many types of sensors for measuring the water level. However, most of them are designed to work in a water tank and not in moving water as the sea. For this reason, we are going to explain how some of these sensor works and if they will perform correctly in the sea.

Table 2- Types of water level sensor

Optical level switches This sensor works using a LED and a phototransistor. The LED emits an infrared light that is reflected and come back to the phototransistor. When the sensor is inside the water, a part of the infrared light is refracted, so the energy that comes back to the phototransistor is less than when in air, so it is possible to calculate the amount of water. When used in the sea, it could be affected by reflections.	
Capacitance They work by measuring the change in capacitance between two plates when the water level changes. It needs to be calibrated and it would be too big to measure what we want, so we will not use it.	
Ultrasonic sensor This sensor measures the level by calculating the duration of sound waves that are reflected off the surface of the liquid and return to the sensor. As with the optical level switches, it could be reflected by the waves in the sea.	
Microwave/Radar It is very similar to the ultrasonic sensor, but now the waves travel at the speed of light. Radars are very precisely but they can also be very expensive as well.	
Tuning fork sensor As they cannot measure continuously, we will not consider them for this project. However, they can be a good system being used as an alarm with some other sensor always measuring the current level.	
Conductivity or resistance As the tuning fork sensor, it can only be used to know if the level is above or under a certain point. In addition, it also emits a low current signal that should not be dangerous, but as we are working in the sea, we would like to avoid this.	

Float switch In the same way as the previous two, it can only measure if a specific point has been surpassed or not, so it is not a good option for our project.	
Pressure sensor As the pressure at the bottom of a liquid is directly related the height of the liquid, we can calculate the height by knowing the pressure at this point. There are many types of pressure sensors, but in this case the best option is to use a differential pressure sensor. This is because it is possible to measure the pressure in some point and compare it with another one, as the atmosphere pressure, so you can find out the real pressure the water exerts.	

By comparing the different sensors in a table, we gained a better perspective on the pros and cons of the sensors. By visualizing the characteristics, we get a better view of the features. We found that the pressure sensor is the best for measuring the water level.

However, we are also testing the ultrasonic and laser sensors that we are comparing for the snow sensor, as they can also be a good option as shown in the table below.

First, we immediately see that the conductivity is not a right option in terms of safety, it contains the characteristics that the sensor can create a current in water, so it means that electricity is generated. For that reason, electricity and water is not a good match.

Second if look at the critical level the kind of data working with the tuning fork, conductivity sensor and float switch is less accurate than the others. Also, the depth of the measure is limited for the capacitive sensors and the radar. The latter sensor is together with the ultrasonic sensor more expensive.

Third, the typical errors, for some reason the optical level switches, the ultrasonics sensors and the radar are more reflected by the waves. This can lead to inaccurate capturing of data.

In short, the sensors with the most benefits for the use in our experiment is the pressure sensor. In the conclusion, we dive deeper into the properties of the pressure sensors.

Table 3- Pros and Cons of Water level sensors

Reference	Depth of measure	Price	Advantage	Disadvantage
Optical Level Switches	Enough	Average	Easy to use/install	May be reflected by the waves
Capacitive sensors	Limited	Average	Small and simple	Cannot measure large values
Ultrasonic sensors	Enough	Expensive	Easy to use/install	May be reflected by the waves
Radar	Limited	Expensive	High accuracy	May be reflected by the waves
Pressure sensor	Enough	Average	Small, high precession and accuracy	Needs to be placed in the water
Tuning fork	Enough	Inexpensive	Very cheap	Binary output (1 or 0)
Conductivity sensor	Enough	Low-cost but requires maintenance	Easy to use	Binary output (1 or 0)
Float Switch	Enough	Inexpensive	Very easy to use	Binary output (1 or 0)

Conclusion: ***pressure sensor***

We decided to orient our research to the pressure sensor according to the table we made. This one answers the best to all the requirements we're looking for. Of course, we will try other sensors to be sure of our choice.

Some of the applications of the sea level measuring can be navigation and dredging purposes, pollution studies, tide analysis and prediction, extreme events (Surge and Tsunami) studies and Secular changes.

A level sensor is an electronic device that measures the height of material, usually liquid, in a tank or other container.

An integral part of process control in many industries, level sensors fall into two main types. The point level sensor is used to mark a single discrete liquid height - a predefined level condition.

Typically, this type of level sensor functions as a high alarm, to signal an overflow condition, or as an indicator for a low alarm condition. The continuous level probe is more sophisticated and can provide level monitoring of an entire system.

It measures liquid level over a range, rather than at a single point, producing an analogy output that is directly correlated to tank level. To create a level management system, the output signal is connected to a process control loop and a visual indicator.

6 CORPORATE IDENTITY

6.1 BRAND NAME

During all our initial steps, we want to decide on a name that has enough engage for take the attention of the possible customers. Also, we were looking for short and empowered brand names. Furthermore, we tried to search all words that are involved into technological fields or points of view of work, but we wanted to find a more aesthetic vibe just getting an "IOT" letter (in this order o reverse) to have the opportunity to create a logo.

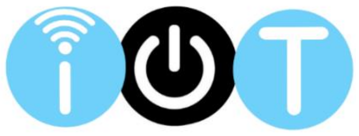
In our process of picking the best option for the name, first we made a brainstorm of some examples and good options:

- **Riotous:** Very loud and uncontrolled, and full of energy
- **Inzwa:** Sensor in Zulu language
- **Òmnia:** "Things, everything, every thing" in Latin
- **Connexa:** "Connected" in Latin
- **Metimur:** "Measure" in Latin
- **Tortoise:** has iot inverse
- **Étoile:** "star" in French and has iot inverse
- **Iota:** Letter of a Greek alphabet

Finally, we decided to get ÓMNIA as our name, because we thought it has engaged, is so powerful and plenty of energy. Words in Latin seems formal, and we think that "things" are also a word that can be used for everything, but we can play with this in our logo using: "Òmnia" -" Things" - "Internet of things". There is also a famous saying that 'Amor vincit omnia', which means that love conquers everything. We can say that in these days the technology and AI is already everywhere and transforms everything in a smart device.

6.2 LOGO

First, we tried to satisfy our logo without our name company, thinking to not relation our logo with our brand name, or just using it as our name: IOT. That's why we created the following logo:



Secondly, when we decided to get our name, we thought about it and saw that this doesn't make any sense because it seems that we are two different brands that don't have any type of relationship. Knowing a brand as: ÒMNIA with a logo that just have three letters: IOT was a non-sense. That's why we decided to change it, trying to find a minimalistic logo that has the perfect proportions because we wanted to add this one to our devices and very complex lines would make difficulties with our task.

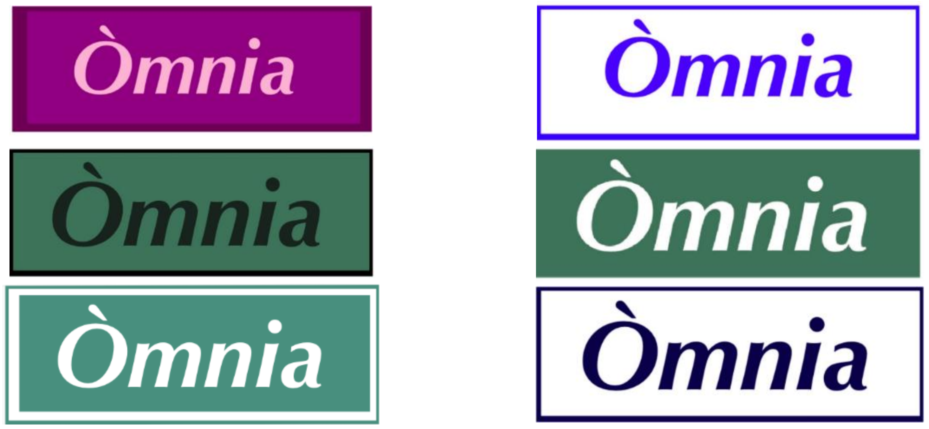
Our choice was to take the best font of letters for our design and try to keep a relation between beauty and formality in our logo, the one that can be accepted on a device or on a paper without difficulties on the dimension of it.

Òmnia

We decided our logo and then we had to pick the different combinations of colors with different priorities (in order):

- Easy to read
- Not have aggressive colors to get good harmony
- Can easily combine with different colors (like white, black, grey...)
- You can recognize our brand just for the color, this must be a very characteristic part of our company

These are some of the best results:



Finally, we decided that the logo of our company will be:

The official logo of our company is the last version is the black and white version, that makes it perfect to print on all the developed devices.

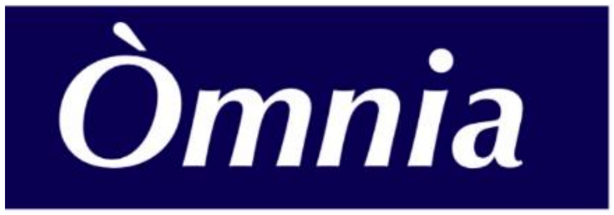


Figure 27- ÒMNIA logo

6.3 COMMUNICATION AND PROMOTION

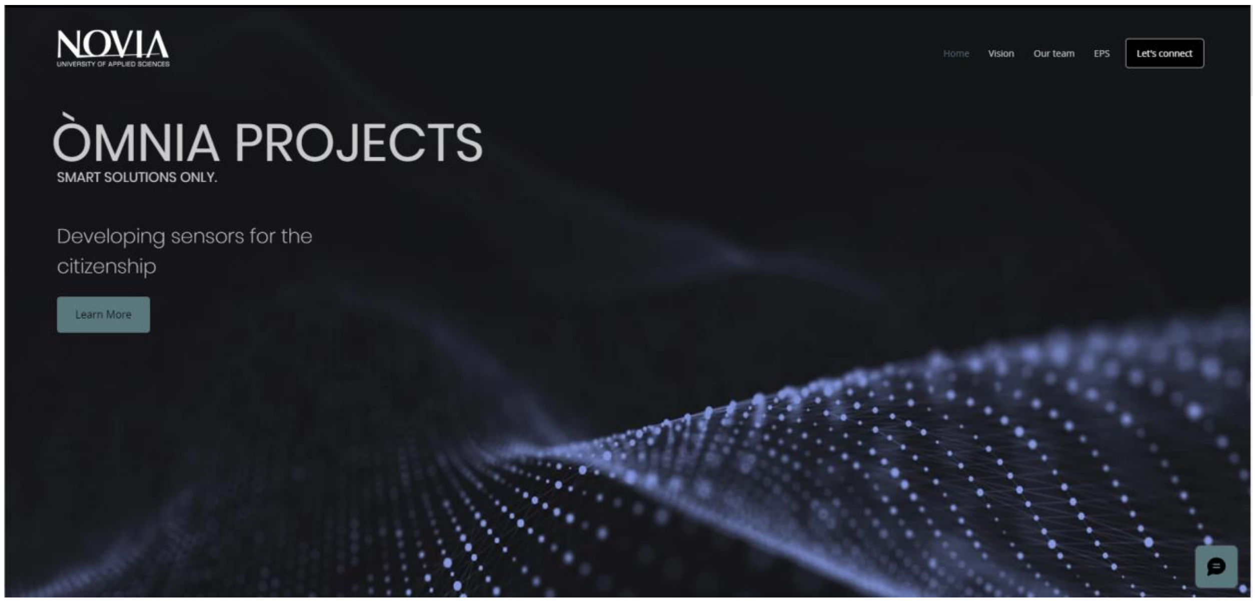
Website

For our website, since we do not have any website designer/ programmer we used a web creator called "*WIX*".

"Wix" is a platform where you can create your own web page without the need of having a background in html coding. The best part about this tool is the number of pre-sets that it contains, you can create easily an attractive interface for your webpage. In addition, you can add some links to other pages like your social media or another company's website.

Although it has the premium option that costs 15€/month, we use the free version because we are a group that is in development and in this way, we can reduce our budget. In the free version you cannot change the domain and you are limited in the GB that you can broadcast.

Our website is more like a minimalistic interface, with dark, pleasing colors for the eyes. Following these steps, we achieve a technology website aura.

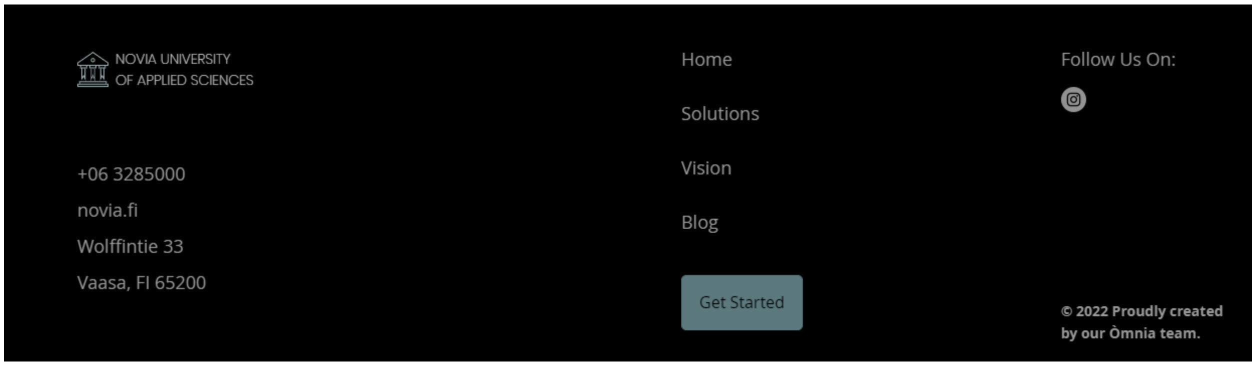


This website contains different pages…

- Home: Here you can find a short description of who we are and what we are doing with a brief explanation of each sensor. Besides, you can find some brands that we are working together with.

- Vision: In this page the customer can find an overview of our mission, the goal and values of OMNIA company.
- Our team: This is the page where the consumer has the opportunity of knowing about us member by member. There is a brief description of the persons that form this group with each nationality and some details about the careers.
- EPS: The last information page is for let the viewer know what EPS it is about and how works. Here we explain what this program is, who is involved and what is the purpose.
- Let´s connect: Finally, we add an easy way of direct contact between the customer and the company where anyone who wants to talk with us can send us a message just writing the name and the text. For us, this is a good method to get external feedback and we can improve hearing what other people think about us.

On the bottom part of all the pages, there is a post where customer can find all the information about the University (address, phone number, website…) and our social media.


NOVIA UNIVERSITY OF APPLIED SCIENCES

+06 3285000
novia.fi
Wolffintie 33
Vaasa, FI 65200

Home
Solutions
Vision
Blog
Get Started

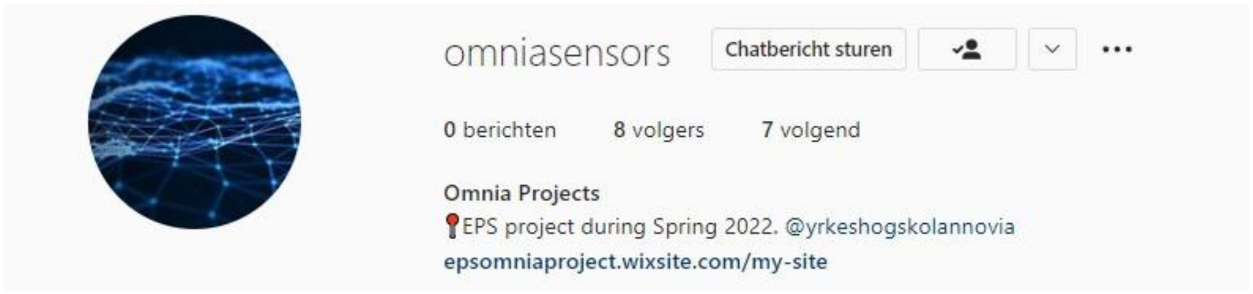
Follow Us On:

© 2022 Proudly created by our Òmnia team.

The link for our website the following:

https://epsomniaproject.wixsite.com/my-site

Social Media

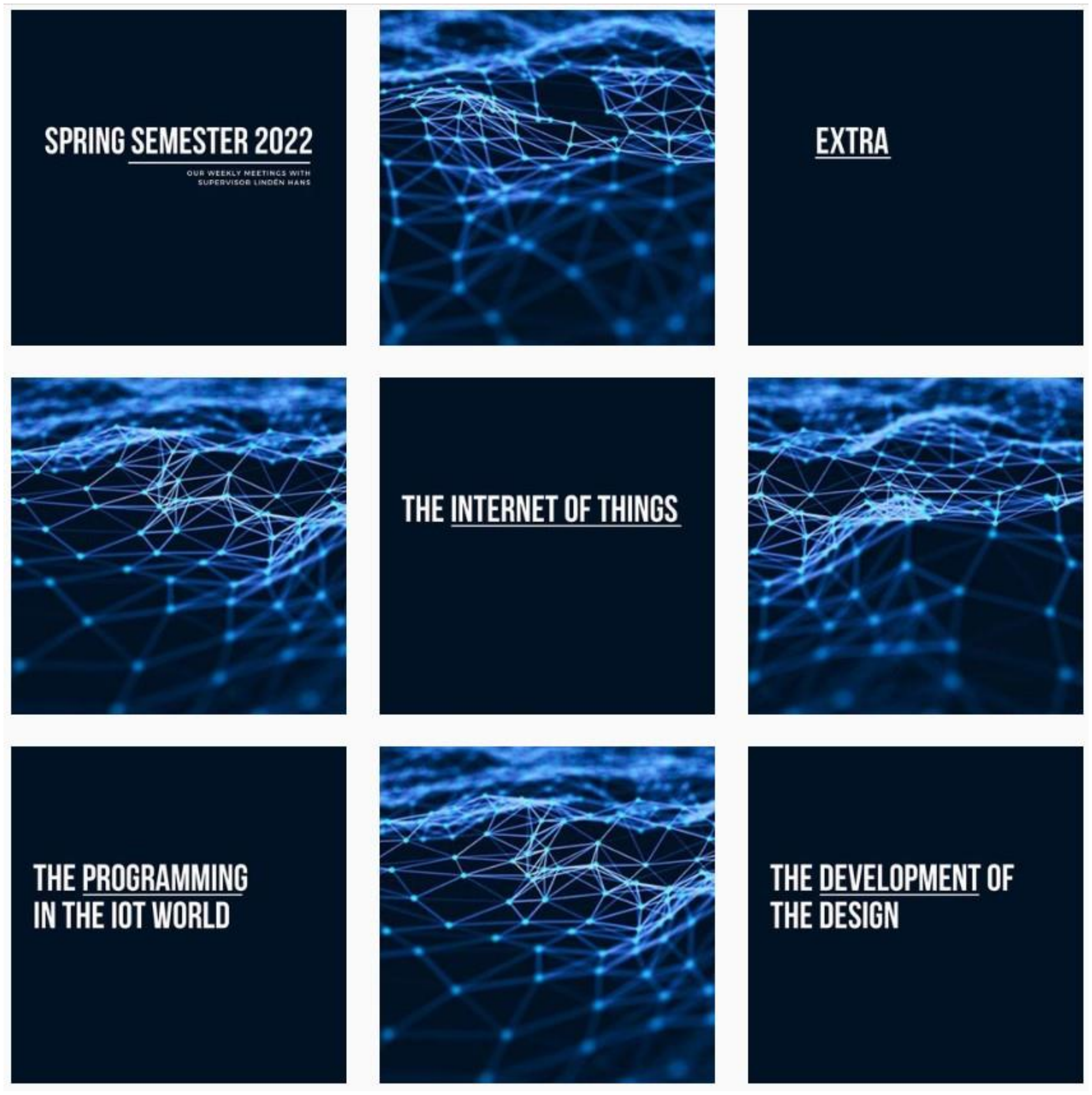

omniasensors
Chatbericht sturen

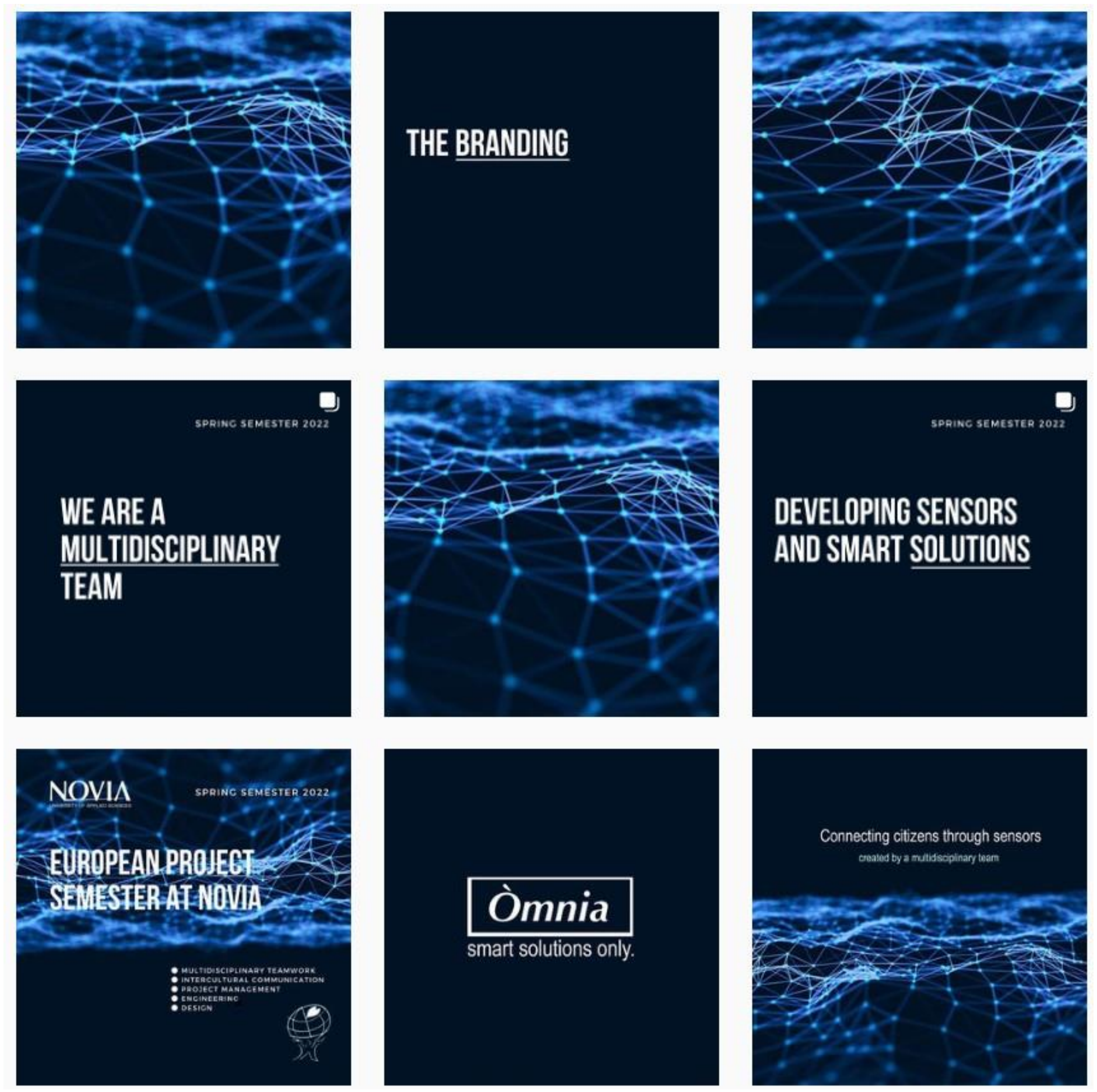
0 berichten 8 volgers 7 volgend

Omnia Projects
📍EPS project during Spring 2022. @yrkeshogskolannovia
epsomniaproject.wixsite.com/my-site

Figure 28 -Instagram Account of our project: Omnia Projects

Next step is developing a proper social media account, to inform people about what our project during the Spring Semester 2022 at the Novia University was about. Again, it is important for a company to keep a same brand colors and style that characterize the professionality. With the dark blue colors and the clean design exudes an atmosphere of future technology, innovation and connectivity.


SPRING SEMESTER 2022
OUR WEEKLY MEETINGS WITH SUPERVISOR LINDÉN HANS
EXTRA
THE INTERNET OF THINGS
THE PROGRAMMING IN THE IOT WORLD
THE DEVELOPMENT OF THE DESIGN


THE BRANDING
SPRING SEMESTER 2022
WE ARE A MULTIDISCIPLINARY TEAM
SPRING SEMESTER 2022
DEVELOPING SENSORS AND SMART SOLUTIONS
NOVIA
SPRING SEMESTER 2022
EUROPEAN PROJECT SEMESTER AT NOVIA
MULTIDISCIPLINARY TEAMWORK
INTERCULTURAL COMMUNICATION
PROJECT MANAGEMENT
ENGINEERING
DESIGN
Òmnia
smart solutions only.
Connecting citizens through sensors
created by a multidisciplinary team

The design of our account is made in Photoshop Creative Cloud and on the design website Canva.

The link for our social media account on Instagram, where you find more information about the project. https://www.instagram.com/omniasensors/

Posters

Here is the process of the development of the design of the poster captured in some pictures. Below you will find one of the brainstorming tools to efficiently visualize ideas together in an overview by making a Pinterest Moodboard. In the search to find the right words, colors, symbols and shapes in order to keep in line with the complete branding.

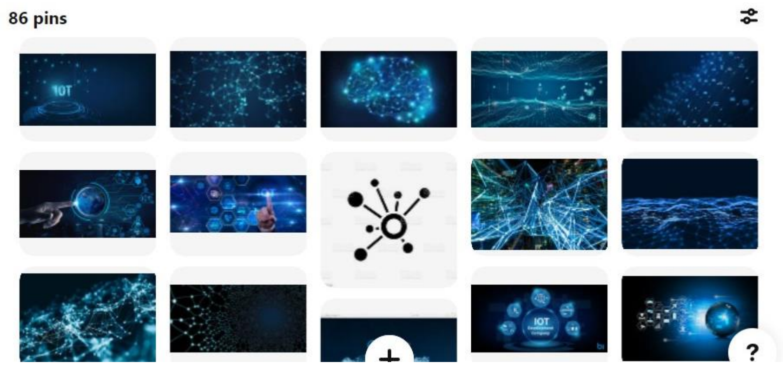


Figure 29 -Inspiration on Pinterest platform: Moodboard and first ideas for the design of the poster

Designing means iterating and a process of contemplating and seeing the details in order to achieve the desired goal and that is that your message and values of your companies are seen in giving a poster one look. The branding part is important in order to keep your company alive and attract the right customers. The design is made in Photoshop Adobe Creative Cloud.

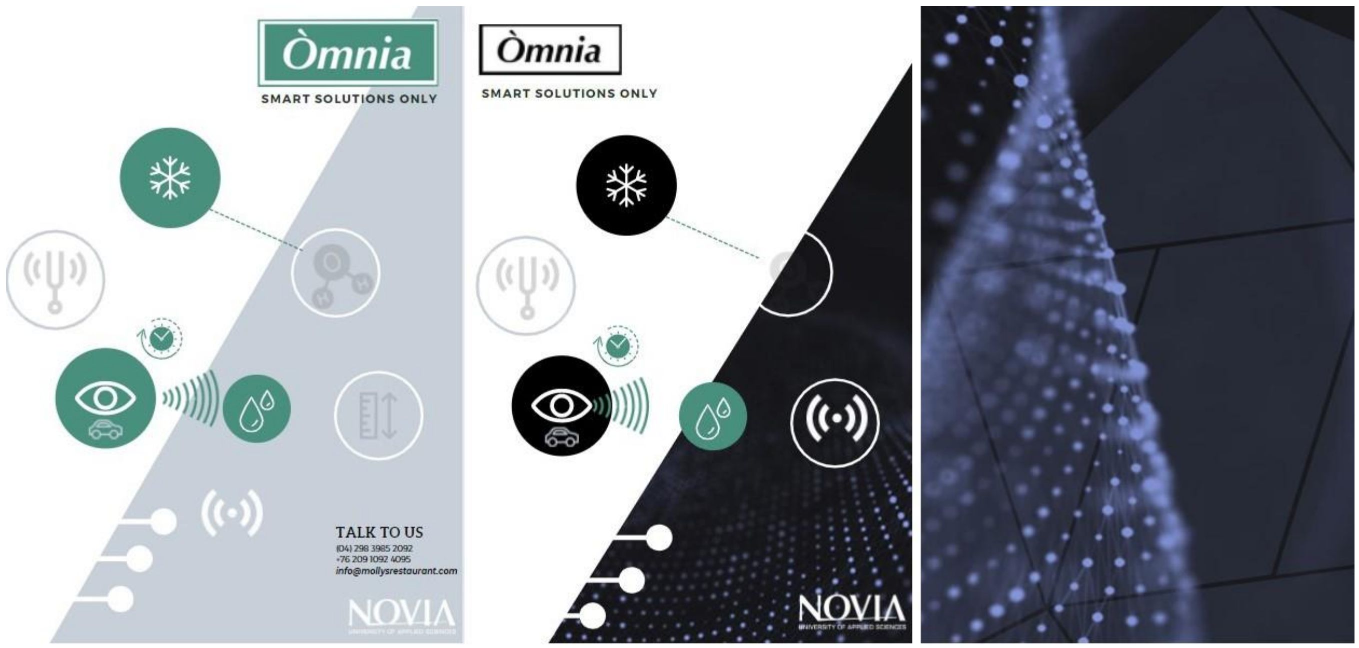


Figure 30 -The first prototypes of the Omnia poster. The visualization of the first ideas

First, we combined and compared some icons in order to make the right symbols for each project…

Second, it is important to choose the right shapes. Inspired by some shapes to visualize the connectivity aspect in our company. A poster needs to be minimalistic but with enough information or visualizations with icons in order to make it a proper poster.

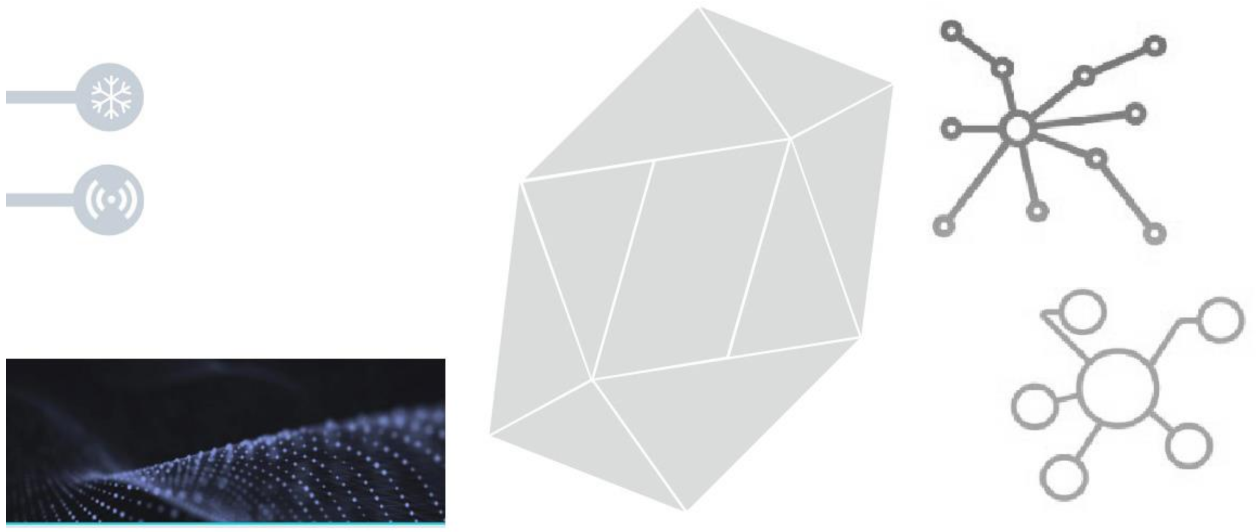


Figure 31 -Inspirational illustrations, icons, symbols and shapes

In general, our Omnia Team created an overall branding. The futuristic design describes our values of our brand. An innovative, fresh technology look. Our website, social media and poster are similar. Here is our innovative poster to attract interested people. By making clear that Omnia is a technology company that is open for future collaboration and other project ideas. The poster contains the logo of the European Project Semester, the Novia University and our Omnia logo. Thereby, all the current projects, we worked on this spring 2022, are mentioned and the Omnia projects for other future project ideas. Then there is the QR code with the link to the website, where we share more information. The connection lines are as already said inspired by some shapes and it appears as a network that the project connected together. The main goal of the poster is to get the attention of people, so that people will check what our project is about.

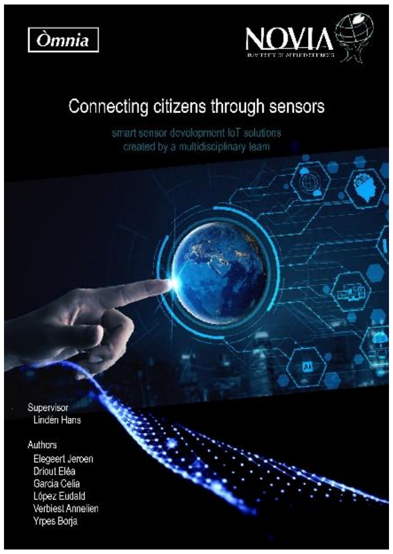


Figure 33 -Another idea with an futuristic look

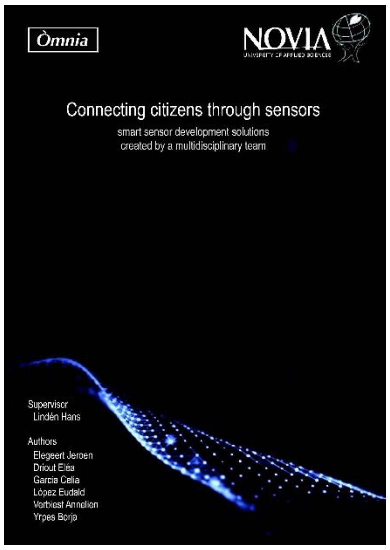
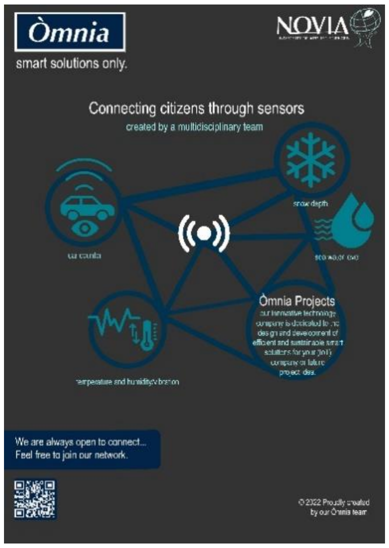


Figure 32 -Simple idea for the poster



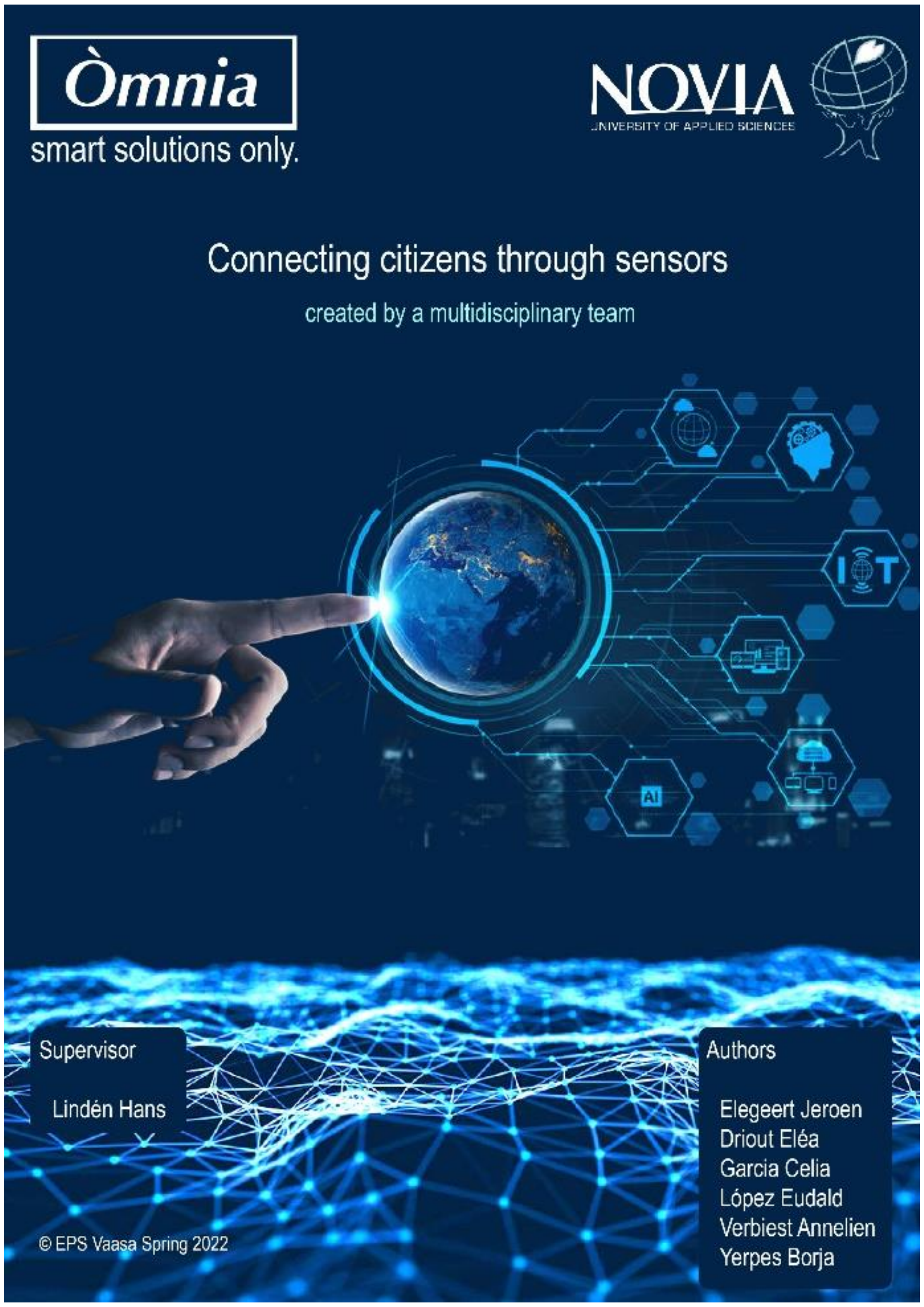
Òmnia
smart solutions only.
NOVIA
Connecting citizens through sensors
created by a multidisciplinary team
snow depth
car counts
Òmnia Projects
temperature and humidity/vibration
We are always open to connect...
Feel free to join our network.
© 2022 Proudly created by our Òmnia team

Figure 34 -Trial poster with grey background

Here is the lay-out already the final version of the poster, before we dive deeper into the last details to optimize the design. On the left the blue version of the poster. On the right the front page, that contains the same lay-out such as the poster.

Furthermore, here are some shapes and colors eventually transformed in the final version. From black, to grey, to blue to the choice dark blue, in order to add more contrast between colors. In the middle, in the heart of the poster, there is the typical symbol for vibration and connectivity.

In conclusion, our final Omnia poster is innovative, futuristic and with the ideal amount of information.


Òmnia

smart solutions only.

NOVIA

UNIVERSITY OF APPLIED SCIENCES

Connecting citizens through sensors

created by a multidisciplinary team

Supervisor

Lindén Hans

© EPS Vaasa Spring 2022

Authors

Elegeert Jeroen
Driout Eléa
Garcia Celia
López Eudald
Verbiest Annelien
Yerpes Borja

Figure 35 -The final version for the front page

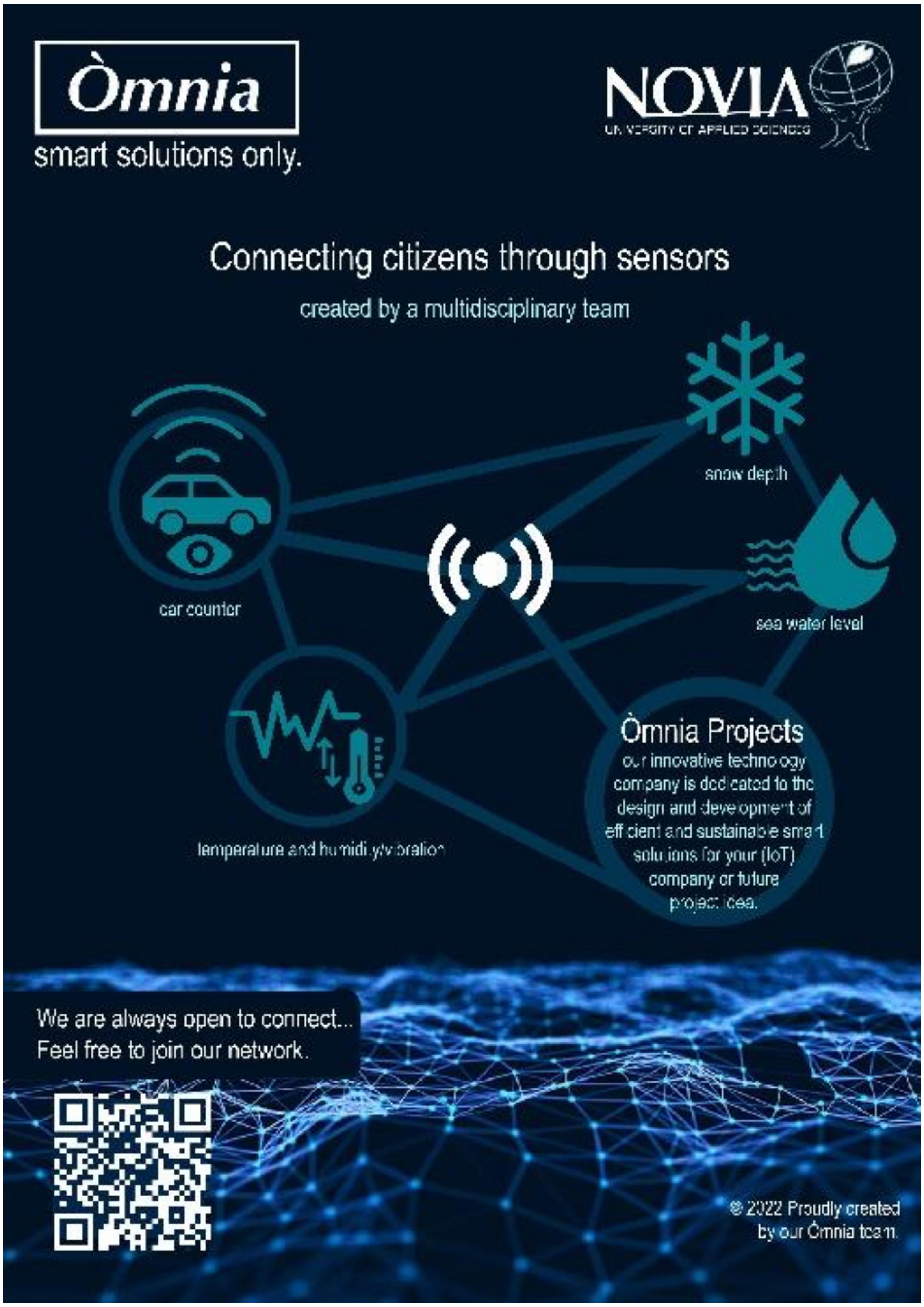

Òmnia
smart solutions only.
NOVIA
UNIVERSITY OF APPLIED SCIENCES
Connecting citizens through sensors
created by a multidisciplinary team
snow depth
car counter
sea water level
temperature and humidity/vibration
Òmnia Projects
our innovative technology company is dedicated to the design and development of efficient and sustainable smart solutions for your (IoT) company or future project idea.
We are always open to connect...
Feel free to join our network.
© 2022 Proudly created by our Òmnia team.

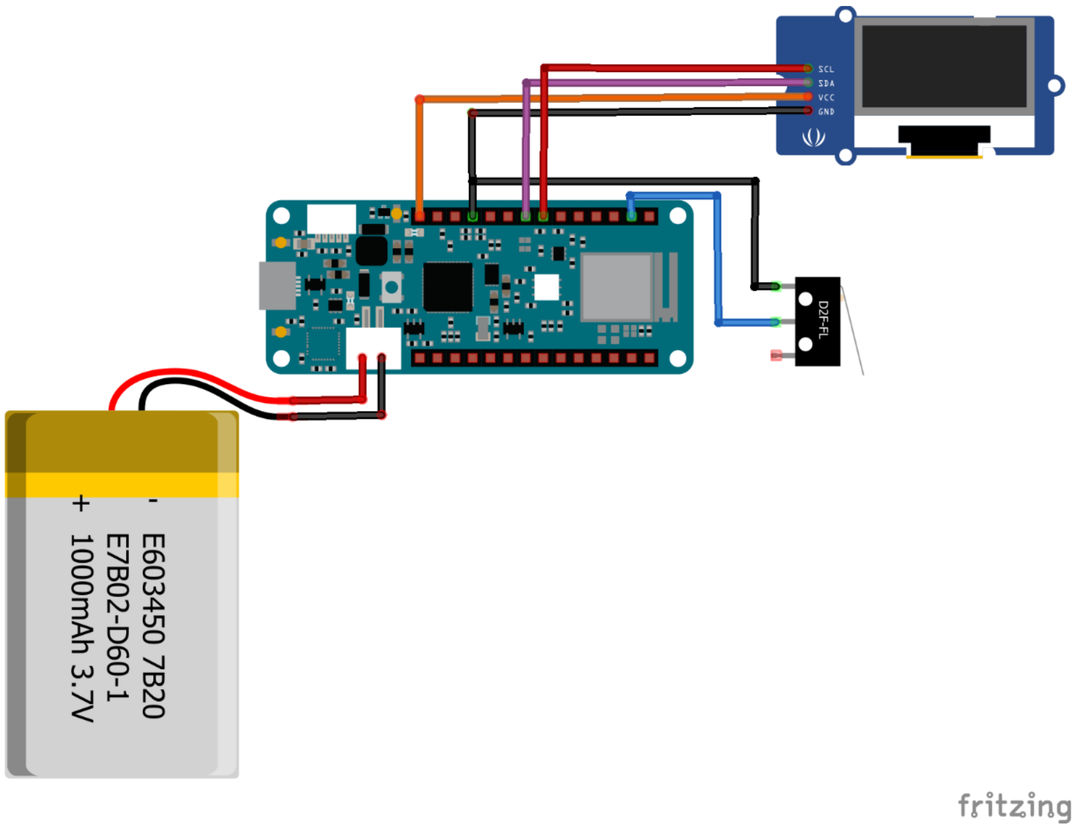
Figure 36 -Final version of our Omnia poster

7 PRACTICALITIES

7.1 VIBRATION SENSOR

Hardware

This device will consist of four elements. The main device is the Arduino Nicla Sense ME, which has the multiple sensors that will obtain data from both the environment and vibration. This device will be connected to the Arduino MKR Wifi 1010 board that will allow the transmission of these values via Wifi to be accessible from any computer. It will also include a screen where the values can be seen, a button that will allow the vibration sensor mode to be initiated and a battery that will power the device so that it can be used wirelessly in any location. Below is a diagram showing the connections of all components:



Arduino Nicla Sense ME

The Nicla Sense ME is a small, low-power usage tool that sets a new standard that enables users to develop smart sensing applications. It is designed to easily analyze motion and the surrounding environment – Hence the “M” and “E” in the name.

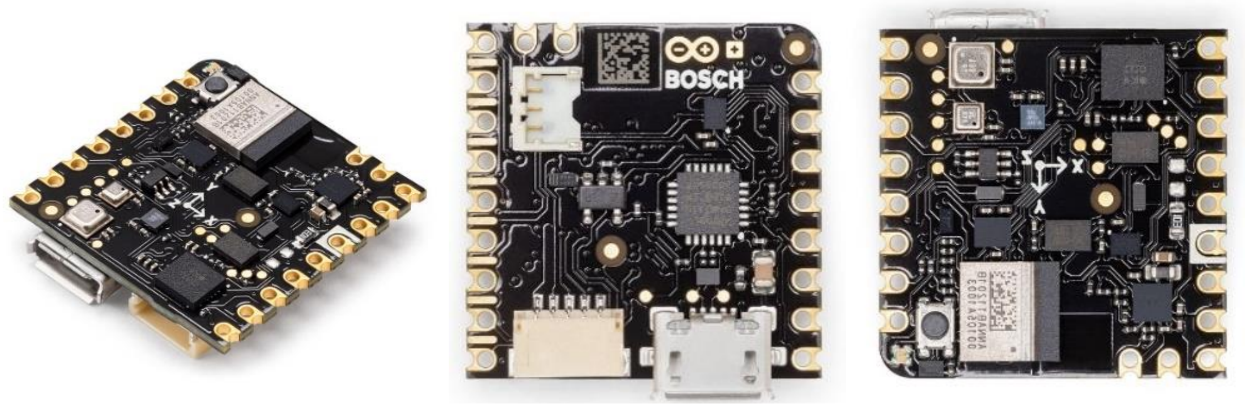


Figure 37- Arduino Nicla Sense ME

In addition, this tiny device is suitable for projects which combine sensors an AI due to its strong computational power and low consumption. This can even lead to standalone applications when it is battery operated. The key benefits taken from the official Arduino Website are as follows:

- Tiny size, packed with features
- Low power consumption
- Add sensing capabilities to existing projects
- When battery-powered, it becomes a complete standalone board
- Powerful processor, capable of hosting intelligence on the Edge
- Measures motion and environmental parameters
- Robust hardware including industrial-grade sensors with embedded AI
- BLE connectivity maximizes compatibility with professional and consumer equipment
- 24/7 always-on sensor data processing at ultra-low power consumption

The board Arduino Nicla Sense ME combined four state-of-the-art sensors from BOSH Sensor Tec:

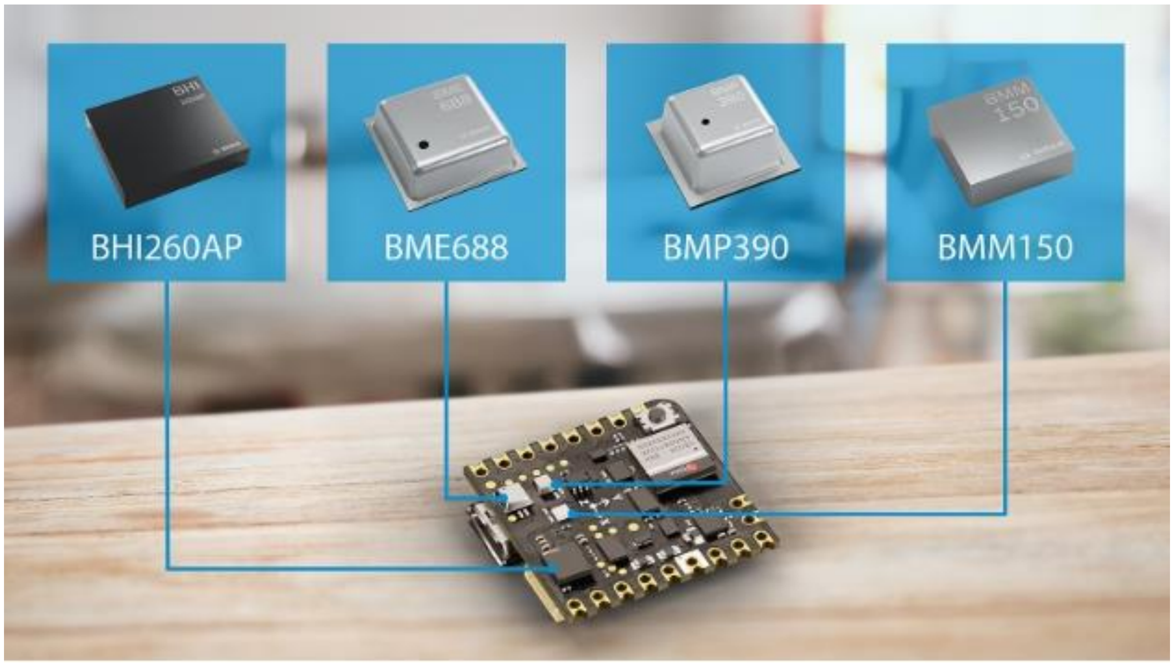


Figure 38- Nicla SENSE ME sensors

BHI260AP

Is an AI smart sensor hub with integrated 6 axis IMU (3-Axis Accelerometer + 3-Axis Gyroscope) for activity detection, powered by a 32-Bit Synopsys DesignWare ARC™ EM4™ CPU.

- Self-learning AI software
- Low power pedestrian position tracking
- Personalized fitness tracking and swim analytics

BMP390

A high-performance pressure sensor operating between 300 - 1250 hPa with low drift. It calculates the pressure into altitude and vice versa so it is ideally suited for a wide range of altitude tracking applications, such as smartphones, GPS modules, wearables, hearables and drones. For example, to maintain a quadcopter drone in a specific altitude during a flight. Also, it is able to forecast when rain can be expected.

- Best performance in market
- Unique accuracy and stability
- Lowest noise

BMM150

is a 3-axis digital geomagnetic sensor. It provides absolute spatial orientation and motion vectors with high accuracy and dynamics. Useful if you are using magnetic encoders or if you just need to know the orientation of the device.

- Outdoor/indoor navigation
- Head movement tracking

BME688

Is the Environmental sensor which integrates pressure, humidity and temperature sensors. In addition to being the first gas sensor with Artificial Intelligence (AI). It can detect Volatile Organic Compounds (VOCs) and other gases such as carbon monoxide and hydrogen in the part per billion (ppb) range

- Specific detection of VSCs
- Application-specific gas scanner
- BME AI-Studio

Back to talking about the Arduino Nicla Sense ME as a whole, the following image shows the pinout diagram, which can summarize the usefulness of the board:

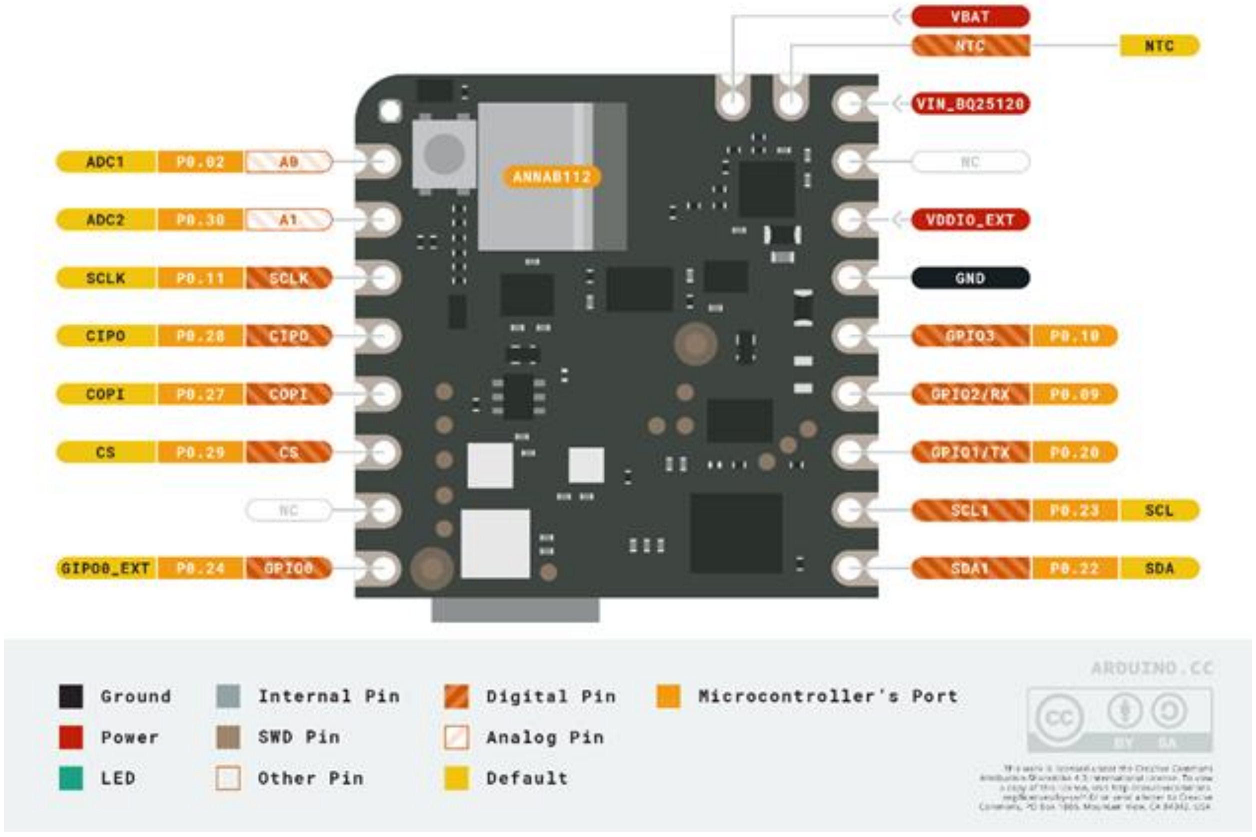


Figure 39- Input Arduino Nicla SENSE ME

As it can be seen, and like many of the Arduino products, it has multiple pins that allow it to have several communication protocols. In our device we will use **I2C**, even so, we wanted to include a short explanation of all the types of communication supported by this board.

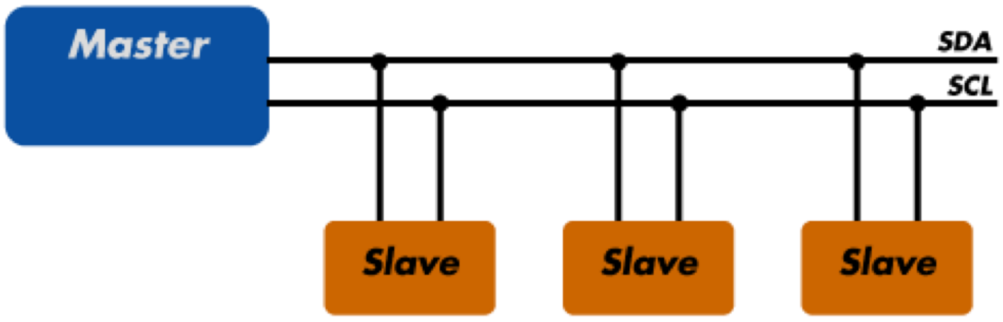
- **I2C**

  This is the protocol we will use in our device. The pins used for I2C (Inter-Integrated Circuit) on the Nicla Sense ME are the following: SDA1 and SCL1.

  It is a serial communication protocol, defining the data frame and physical connections to transfer bits between 2 or more digital devices. It is one of the most widely used for communicating with digital sensors because unlike the SPI, its architecture allows confirmation of the data received, within the same frame, among other advantages.

  It is based on a MASTER - SLAVE system, but now, instead of being connected by four lines, they are connected by two: **S**erial **DA**ta – **SDA** y **S**erial **CL**ock – **SCL**. The first one is the line on which data moves among devices and the second one is the line of the clock pulses that synchronize the system. As shown in the following picture multiple sensors and/or actuators can be connected to both.

Table 4 - SDA and SCL connections



There are major differences between the two devices:

▪ **Master or Controller:** is the one in charge of initiating and controlling the communication. n our case, the Arduino MKR Wifi 1010 will play this role.
▪ **Slaves or Peripherals**: these are devices that will be continuously waiting for a master to communicate with them. Common cases are sensors and actuators that support this protocol, although it is also possible, and sometimes necessary, for a microcontroller to behave as a slave. In our case, the Arduino Nicla Sense ME will behave as a slave to the Arduino MKR 1010.

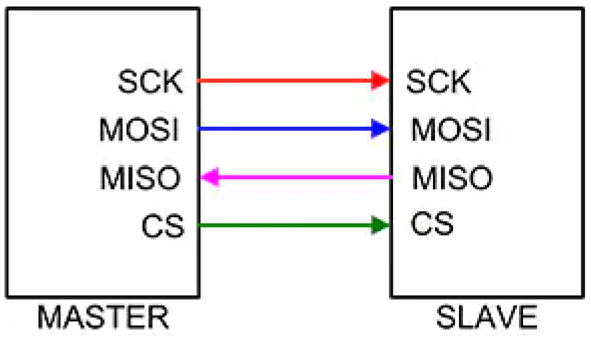
▪ SPI

The pins used for SPI (Serial Peripheral Interface) on the Nicla Sense ME are the following: CS, CIPO, COPI and SCLK.

This type of communication is a simple hardware addressed interface that offers great flexibility in terms of the size of data sent. It is a very good choice for transferring long data streams, as it does not use a standard protocol and transfers only data packets.

As I2C, it is based on a master-slave model and can handle multiple secondary devices using duplex communications operating at clock speeds up to 50 MHz

Table 5- Communication between Master and Slave



This communication uses a maximum of four signal lines on which the master device supplies and controls the clock (SCK) and chip select lines (CS). The complete multiplexer operation is handled via the Master Out Slave In (MOSI) and Master in Slave Out (MISO) data lines. The data is outputted via the clock signal in such a way that the data transfer resembles a shift register with one bit shifted for each clock.

- **UART**

The pins used for UART (Universal asynchronous receiver-transmitter) are the following: Rx (GPIO2) and Tx (GPIO1).

UART (universal asynchronous transceiver) is a protocol for exchanging serial data between two devices. UART is very simple and like I2C, only uses two wires between the transmitter and receiver to transmit and receive in both directions. Communication on the UART can be simplex (data is sent in one direction only), half-duplex (each side transmits but only one at a time), or full duplex (both sides can transmit simultaneously)

One of the great advantages of the UART is that it is asynchronous: the transmitter and receiver do not share a common clock signal. While this greatly simplifies the protocol, it places certain requirements on the transmitter and receiver. Since they do not share a clock, both terminals must transmit at the same preset rate to have the same bit synchronization. In addition to having the same baud rate, both sides of a UART connection must also use the same parameters and frame structure.

In recent years, the popularity of the UART has declined, and it has been replaced by SPI and I2C protocols, although it is still used for lower speed and lower performance applications, as it is very simple, low cost and easy to implement.

▪ **Bluetooth® Low Energy**

This way of communication is quite interesting as it is not so common in the usual sensors. It is provided by the ANNA- B112 Bluetooth Module with is incorporated on the Nicla board.

BLE is a type of Bluetooth that consumes much less energy than the classic Bluetooth, approximately 10% less. Was develop for data transmission, but with a strong bias to maximize battery savings, precisely to ensure a long battery life.

It is a very useful technology and is an excellent choice for those who need more continuous monitoring, with high data processing and accuracy. The following image shows a possible use that could be given to our Arduino Nicla using this type of communication.

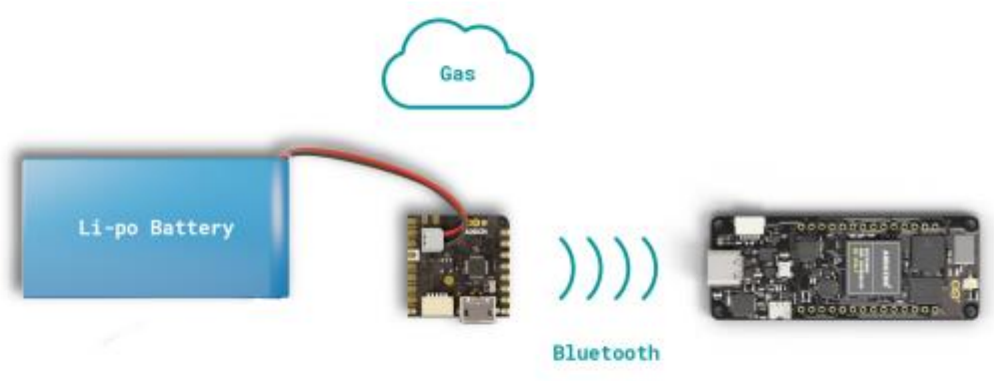


Figure 40- Bluetooth explanation

Another important aspect of this board is how it will be powered, the following image shows the technical characteristics of the board

Table 6- Operation conditions of Nicla Sense ME

Symbol	Description	Min	Typ	Max	Unit
V_{IN}	Input voltage from VIN pad	3.5	5.0	5.5	V
V_{USB}	Input voltage from USB connector	4.8	5.0	5.5	V
V_{DDIO_EXT}	Level Translator Voltage	1.8	3.3	3.3	V
V_{IH}	Input high-level voltage	$0.7V_{DDIO\_EXT}$[1]		V_{DDIO_EXT}	V
V_{IL}	Input low-level voltage	0		$0.3V_{DDIO\_EXT}$[2]	V
T_{OP}	Operating Temperature	-40	25	85	°C

In our device it will be powered at 5V from the Vin pin, which is connected to the %V pin of the Arduino MKR Wifi 1010.

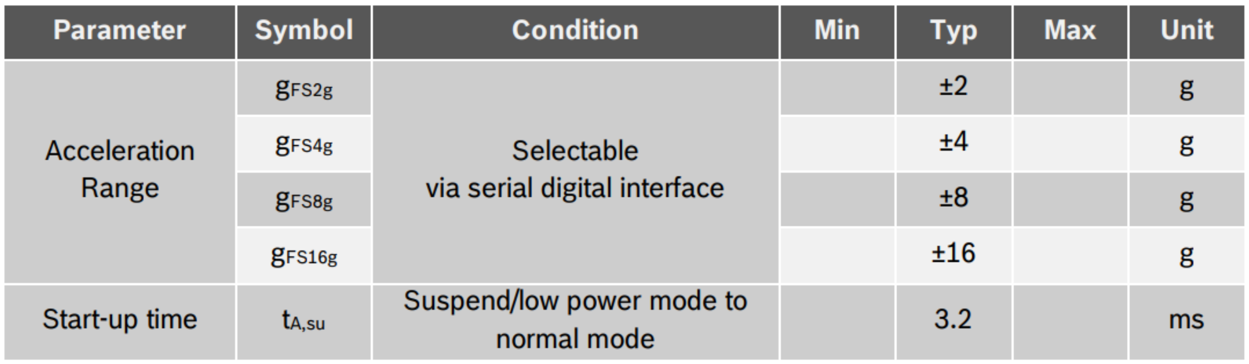
To finish with the explanation of this device, and although all the sensors of which this card is composed have already been explained in a general manner, the one that is most important since the main functionality of this device is to measure vibration is the accelerometer which is integrated in the **BHI 260 AP** sensor. Another less conventional sensor that we are using is the **BME688**, which, as explained above, is the "environmental" sensor. Both reasons lead us to explain each of these sensors in more detail.

Accelerometer BHI 260 AP

The BHI 260 AP integrates a low power and low noise inertial measurement unit (6DoF IMU) consisting in a 16 bit digital triaxial gyroscope and a 16 bit digital triaxial accelerometer. It supports data rates up to 1600Hz in the accelerometer and 6400Hz in the Gyroscope.

Below is a table showing the operating conditions of the accelerometer. And the table 4shows the output signal accelerometer,

Table 7- Operation conditions of accelerometer



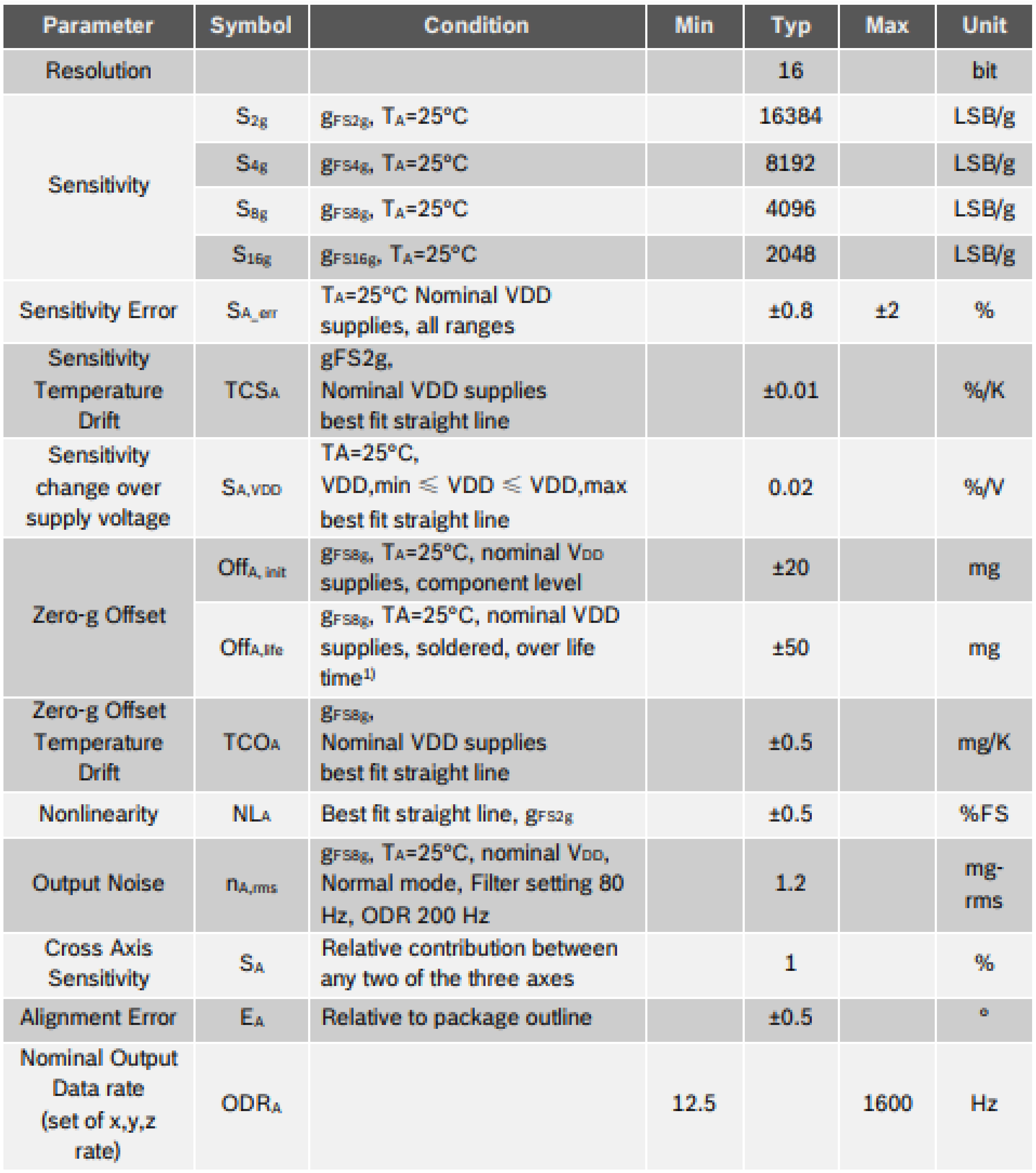
Parameter	Symbol	Condition	Min	Typ	Max	Unit
Acceleration Range	g_{FS2g}	Selectable via serial digital interface		±2		g
	g_{FS4g}			±4		g
	g_{FS8g}			±8		g
	g_{FS16g}			±16		g
Start-up time	$t_{A,su}$	Suspend/low power mode to normal mode		3.2		ms

As shown in the table above, this sensor can work in different acceleration ranges ±2g, ±4g, ±8g and ±16g. With the default configuration of the sensor the sensor works in a range of ±8g, and as we can see in the following table the resolution is 16 bits. Thus, to know to which value 1 g is equivalent, it will be necessary to perform this calculation, which divides the total range of possible g values by the total numbers to represent them:

$$\frac{acceleration\ range}{2^{resolution}} = \frac{\pm 8\ \mathrm{g}}{2^{16-1}} = 4096$$

We conclude then that 1g equals 4096m so we should scale down the output data by 4096 whenever we get our values of acceleration on any of the three axes.

Table 8 - Operation conditions of accelerometer



Parameter	Symbol	Condition	Min	Typ	Max	Unit
Resolution				16		bit
Sensitivity	S_{2g}	g_{FS2g}, T_A=25°C		16384		LSB/g
	S_{4g}	g_{FS4g}, T_A=25°C		8192		LSB/g
	S_{8g}	g_{FS8g}, T_A=25°C		4096		LSB/g
	S_{16g}	g_{FS16g}, T_A=25°C		2048		LSB/g
Sensitivity Error	S_{A_err}	T_A=25°C Nominal VDD supplies, all ranges		±0.8	±2	%
Sensitivity Temperature Drift	TCS_A	gFS2g, Nominal VDD supplies best fit straight line		±0.01		%/K
Sensitivity change over supply voltage	$S_{A,VDD}$	TA=25°C, VDD,min ≤ VDD ≤ VDD,max best fit straight line		0.02		%/V
Zero-g Offset	$Off_{A,\ init}$	g_{FS8g}, T_A=25°C, nominal V_{DD} supplies, component level		±20		mg
	$Off_{A,life}$	g_{FS8g}, TA=25°C, nominal VDD supplies, soldered, over life time[1)]		±50		mg
Zero-g Offset Temperature Drift	TCO_A	g_{FS8g}, Nominal VDD supplies best fit straight line		±0.5		mg/K
Nonlinearity	NL_A	Best fit straight line, g_{FS2g}		±0.5		%FS
Output Noise	$n_{A,rms}$	g_{FS8g}, T_A=25°C, nominal V_{DD}, Normal mode, Filter setting 80 Hz, ODR 200 Hz		1.2		mg-rms
Cross Axis Sensitivity	S_A	Relative contribution between any two of the three axes		1		%
Alignment Error	E_A	Relative to package outline		±0.5		°
Nominal Output Data rate (set of x,y,z rate)	ODR_A		12.5		1600	Hz

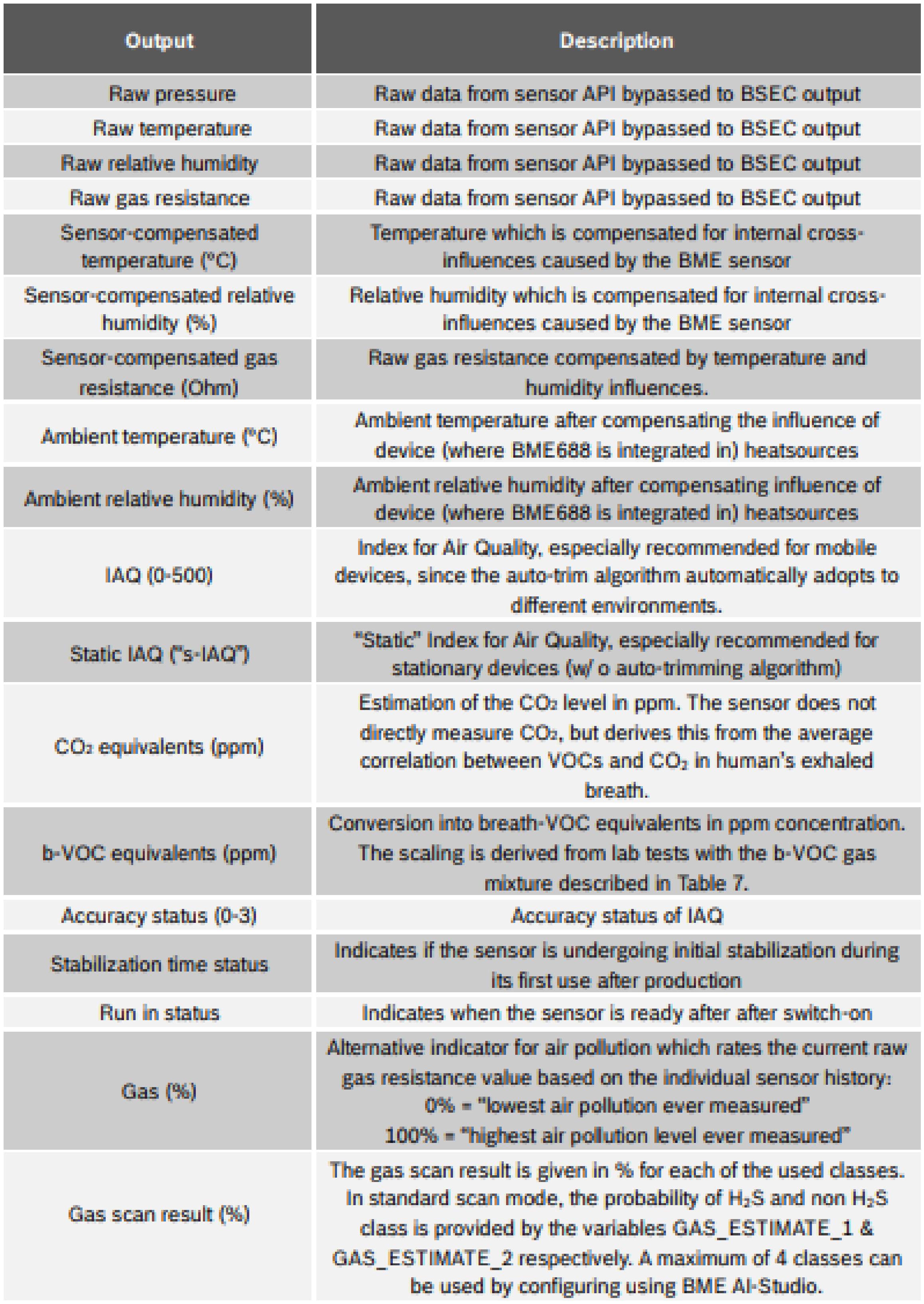
BMW688 sensor

The BME is a low-power gas, pressure, temperature, and humidity sensor that includes artificial intelligence. The gas sensor we'll use can detect volatile organic compounds (VOCs), volatile sulfur compounds (VSCs), and other gases in the parts per billion range, including carbon monoxide and hydrogen.

- The following are some examples of typical applications:
- Measurement of indoor and outdoor air quality
- The analysis of volatile sulfur compounds, which are a sign of bacteria development, is used to detect bad breath or ruined food.
- Unusual gases and odors detected, which could signal a fire for example.

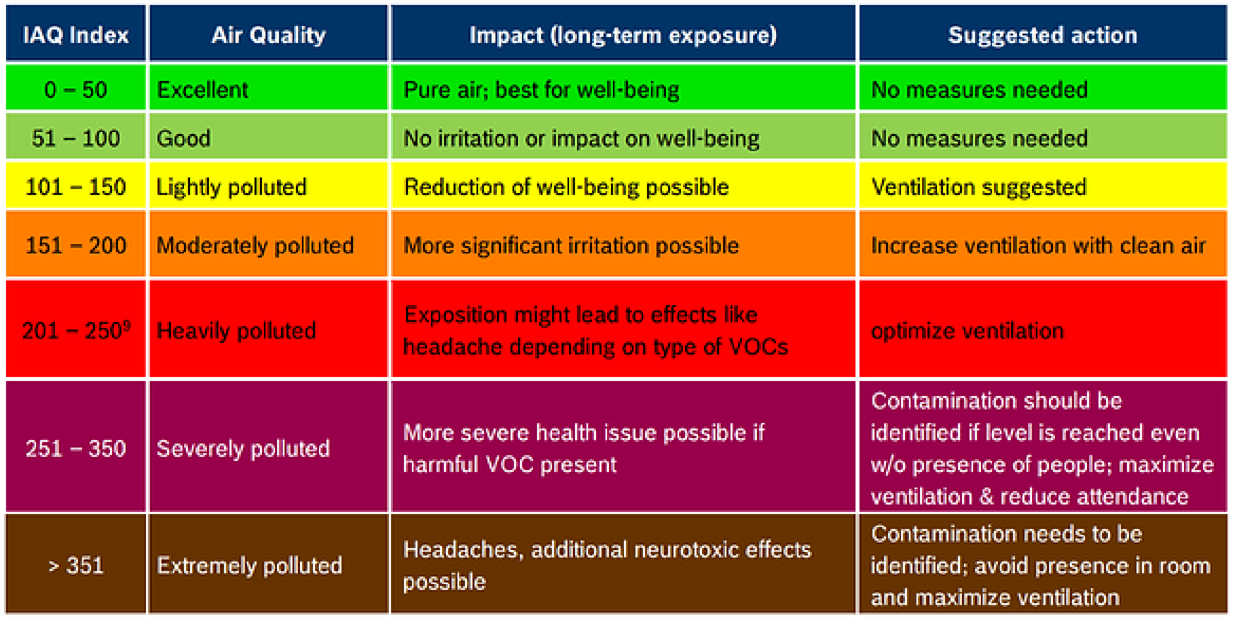
For our project, the most interesting application of this sensor is as an indoor air quality sensor. In fact, the BM688 is a gas sensor made of metal oxide that detects gases through adsorption on its sensitive layer. The signal's intensity is usually related to the chemical reactivity of the gases. The BME988 reacts to most volatile chemicals as well as a variety of other indoor air pollutants and unlike sensors that only measure one specific component, the BME988 can measure the total amount of VOCs/contaminants in the air. The BME688 can detect outgasing from paint, furniture, and/or rubbish, as well as high VOC levels from cooking food or exhaled breath.

The BME688 will output the gas sensor resistance values and variations owing to varying VOC concentrations as a raw signal (the higher the concentration of reducing VOCs, the lower the resistance and vice versa). Because this raw signal is impacted by factors other than VOC content (such as humidity), it is provided a software solution (BSEC: Bosch Software Environmental Cluster). The software has been meticulously designed to work in tandem with the BME688's four inbuilt sensors. The BSEC generates a variety of relevant outputs, as can be seen in the following image. In our project we are only measuring IAQ and CO2 equivalents as they have been considered among the most suitable parameters for measuring environmental quality in both daily and industrial environments.



Output	Description
Raw pressure	Raw data from sensor API bypassed to BSEC output
Raw temperature	Raw data from sensor API bypassed to BSEC output
Raw relative humidity	Raw data from sensor API bypassed to BSEC output
Raw gas resistance	Raw data from sensor API bypassed to BSEC output
Sensor-compensated temperature (°C)	Temperature which is compensated for internal cross-influences caused by the BME sensor
Sensor-compensated relative humidity (%)	Relative humidity which is compensated for internal cross-influences caused by the BME sensor
Sensor-compensated gas resistance (Ohm)	Raw gas resistance compensated by temperature and humidity influences.
Ambient temperature (°C)	Ambient temperature after compensating the influence of device (where BME688 is integrated in) heatsources
Ambient relative humidity (%)	Ambient relative humidity after compensating influence of device (where BME688 is integrated in) heatsources
IAQ (0-500)	Index for Air Quality, especially recommended for mobile devices, since the auto-trim algorithm automatically adopts to different environments.
Static IAQ ("s-IAQ")	"Static" Index for Air Quality, especially recommended for stationary devices (w/ o auto-trimming algorithm)
CO_2 equivalents (ppm)	Estimation of the CO_2 level in ppm. The sensor does not directly measure CO_2, but derives this from the average correlation between VOCs and CO_2 in human's exhaled breath.
b-VOC equivalents (ppm)	Conversion into breath-VOC equivalents in ppm concentration. The scaling is derived from lab tests with the b-VOC gas mixture described in Table 7.
Accuracy status (0-3)	Accuracy status of IAQ
Stabilization time status	Indicates if the sensor is undergoing initial stabilization during its first use after production
Run in status	Indicates when the sensor is ready after after switch-on
Gas (%)	Alternative indicator for air pollution which rates the current raw gas resistance value based on the individual sensor history: 0% = "lowest air pollution ever measured" 100% = "highest air pollution level ever measured"
Gas scan result (%)	The gas scan result is given in % for each of the used classes. In standard scan mode, the probability of H_2S and non H_2S class is provided by the variables GAS_ESTIMATE_1 & GAS_ESTIMATE_2 respectively. A maximum of 4 classes can be used by configuring using BME AI-Studio.

It should be noted that, the IAQ scale goes from 0 (clean air) to 500 (heavily polluted air). The following image shows the classification of the different ranges of air quality indicated by this index.



IAQ Index	Air Quality	Impact (long-term exposure)	Suggested action
0 – 50	Excellent	Pure air; best for well-being	No measures needed
51 – 100	Good	No irritation or impact on well-being	No measures needed
101 – 150	Lightly polluted	Reduction of well-being possible	Ventilation suggested
151 – 200	Moderately polluted	More significant irritation possible	Increase ventilation with clean air
201 – 250[9]	Heavily polluted	Exposition might lead to effects like headache depending on type of VOCs	optimize ventilation
251 – 350	Severely polluted	More severe health issue possible if harmful VOC present	Contamination should be identified if level is reached even w/o presence of people; maximize ventilation & reduce attendance
> 351	Extremely polluted	Headaches, additional neurotoxic effects possible	Contamination needs to be identified; avoid presence in room and maximize ventilation

The algorithms automatically calibrate and adapt to the common situations where the sensor is used during operation (e.g., home, workplace, inside a car, etc.). This automatic backdrop calibration guarantees that IAQ performance is consistent.

The calibration method takes into account the most recent measurement history (often up to four days, customizable) to ensure that IAO - 25 represents "usually excellent" air and IN) - 250 represents "typical polluted" air. Although the time the boards demand to do this process is not clearly indicated in the documentation, the conclusion obtained in practice has been that the duration of the calibration is around 5 minutes.

Arduino MKR WIFI 1010

The companion board we have chosen to work as a gateway to the Nicla is the Arduino MKR Wifi 1010. One of the main reasons we have chosen these boards is because of their perfect compatibility with the Arduino Nicla Sense ME. That is, the Nicla Sense card can be used as MKR shield which enables to extend some of the functionalities of the MKR board. Both devices communicate over I2C.

The following picture shows how to connect the two boards. As we can see, there is full compatibility between them.

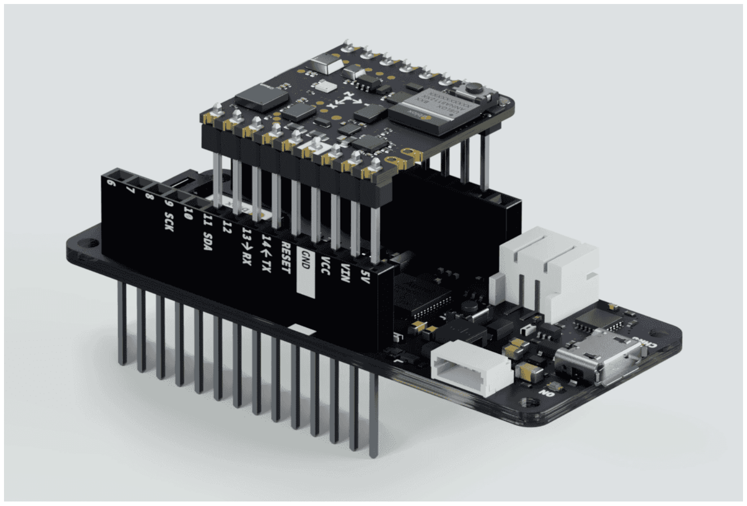


Figure 41- Connection Arduino Wifi MKR1010 and Nicla Sense ME

Later, in the code section, we will detail how the communication between the two boards is carried out.

The Arduino MKR WiFi 1010 is a tiny board form de Arduino MKR Family released in June 2018, as an improvement of the previous models MKR 1000 and Uno Wifi.
One of the main changes was that it adds the uBlox NINA/ESP32 Module, one of the smallest and most powerful industrial Wi-Fi and Bluetooth modules. Its three main features, which make it a much more complete product than some of its peers, are as follows:

u-blox NINA-W102 Module
This module support Wi-Fi and conform to IEEE 802.11b/g/n single-band 2.4 GHz operation, Bluetooth BR/EDR, and Bluetooth low energy

Cortex-M0 32-bit SAMD21
The powerful, low-power processor that is used in all MKR Family boards.

ATECC508 crypto chip

The ATECC508A device is a member of the Microchip CryptoAuthentication™ family of crypto engine authentication devices with highly secure hardware-based key storage. The ECC508 crypto chip makes sure your data remains secure and private and can store up to 16 keys in an EEPROM array.

In addition, the following image shows the pinout diagram to get a better idea of the functionalities of this board:

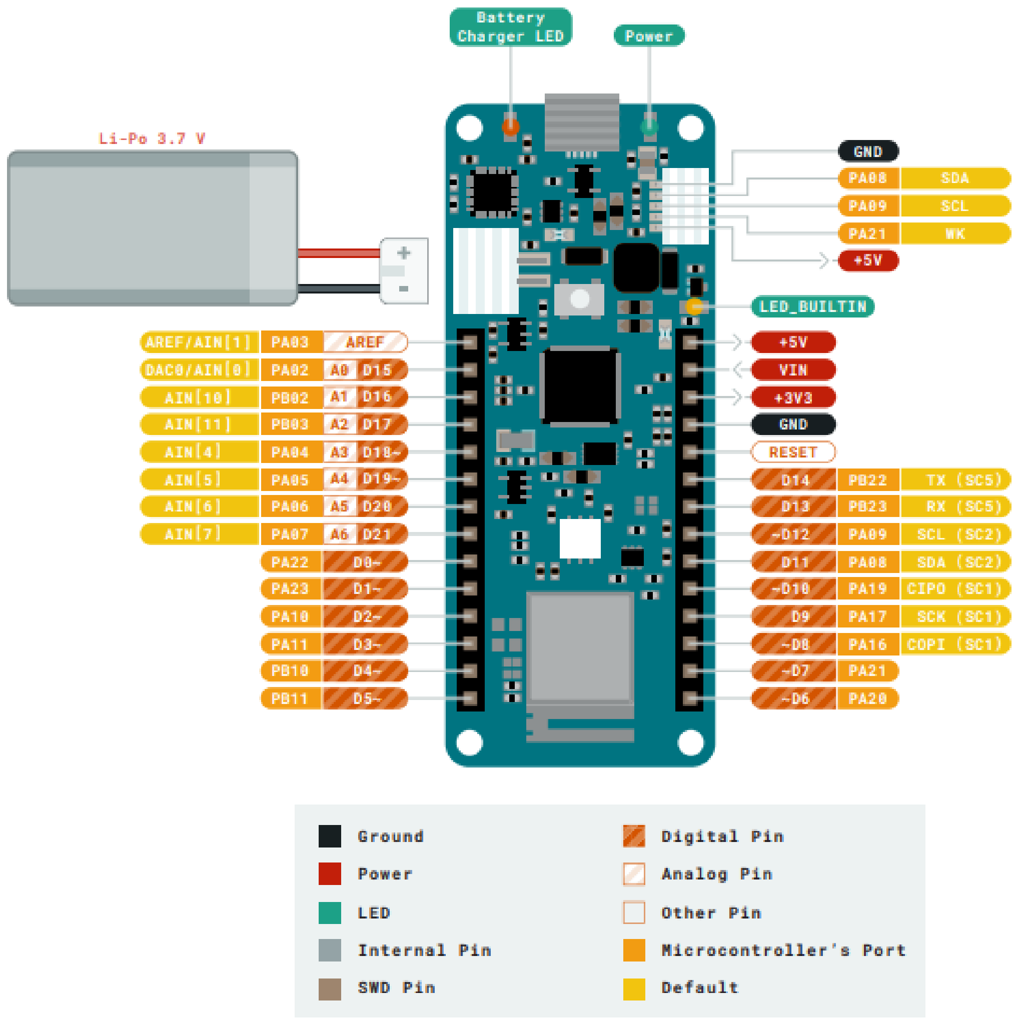
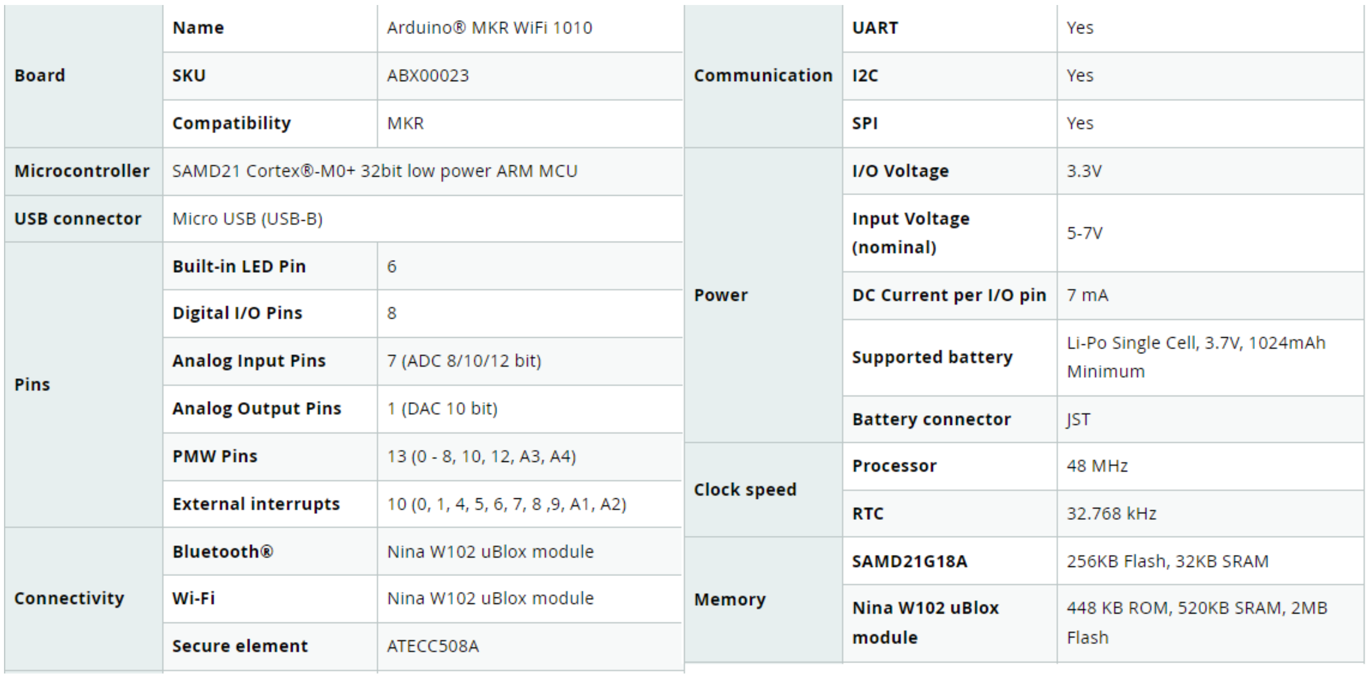


Figure 42- Inputs Arduino Wifi MKR 1010

All these features make the Arduino MKR Wifi 1010 board an excellent for many of the basic IoT application scenarios. It is a great choice for any beginner, maker or professional to get started with Internet of Things (IoT). Its wireless connectivity ability makes this board suitable for use in projects that will communicate with the cloud, such as collecting data from the sensors and uploading this data to cloud services.

The following table summarises the technical aspects of this board:

Table 9- Technical Aspects of the board



Board	**Name**	Arduino® MKR WiFi 1010
	SKU	ABX00023
	Compatibility	MKR
Microcontroller	SAMD21 Cortex®-M0+ 32bit low power ARM MCU	
USB connector	Micro USB (USB-B)	
Pins	**Built-in LED Pin**	6
	Digital I/O Pins	8
	Analog Input Pins	7 (ADC 8/10/12 bit)
	Analog Output Pins	1 (DAC 10 bit)
	PMW Pins	13 (0 - 8, 10, 12, A3, A4)
	External interrupts	10 (0, 1, 4, 5, 6, 7, 8 ,9, A1, A2)
Connectivity	**Bluetooth®**	Nina W102 uBlox module
	Wi-Fi	Nina W102 uBlox module
	Secure element	ATECC508A
Communication	**UART**	Yes
	I2C	Yes
	SPI	Yes
Power	**I/O Voltage**	3.3V
	Input Voltage (nominal)	5-7V
	DC Current per I/O pin	7 mA
	Supported battery	Li-Po Single Cell, 3.7V, 1024mAh Minimum
	Battery connector	JST
Clock speed	**Processor**	48 MHz
	RTC	32.768 kHz
Memory	**SAMD21G18A**	256KB Flash, 32KB SRAM
	Nina W102 uBlox module	448 KB ROM, 520KB SRAM, 2MB Flash

Display OLED

In order to find the best component that would allow us to clearly visualise the measured values of the frequency sensor, we started looking for a display large enough to comfortably read the results. This display also had to meet our case dimensions.

After looking at all the different types of screens that are currently available on the market, we decided the 1,3 inch OLED I2C 128 x 64 pixel diplay would be the best option for us since it fits perfectly with the specs we are looking for. Furthermore, since its pixels are blue, we can show the message using just one colour, otherwise adding more colours will increase the price. Another interesting aspect is that because the self-luminescent pixels do not require backlighting, the display performs better than a conventional LCD display and consumes little power, which saves battery life and will extend battery life.

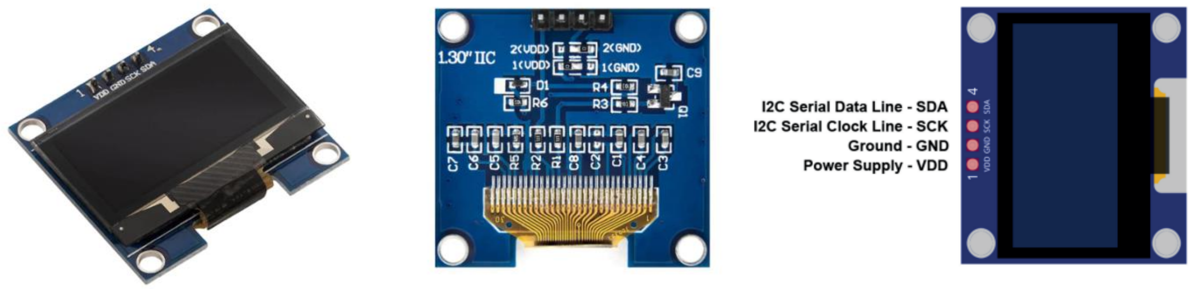
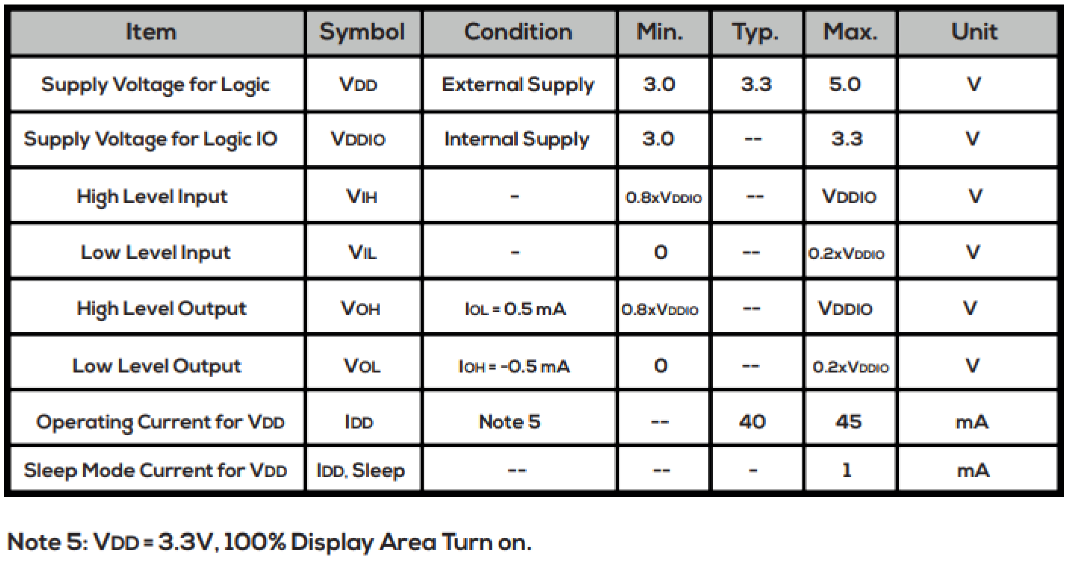


Figure 43- Views of display OLED

The communication protocol used is **I2C**, so it is compatible with our Arduino MKR Wifi 1010.

Finally, the following table summarises the electrical characteristics of this display.

Table 10- Specifications of Display OLED



Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Supply Voltage for Logic	V_{DD}	External Supply	3.0	3.3	5.0	V
Supply Voltage for Logic IO	V_{DDIO}	Internal Supply	3.0	--	3.3	V
High Level Input	V_{IH}	-	$0.8xV_{DDIO}$	--	V_{DDIO}	V
Low Level Input	V_{IL}	-	0	--	$0.2xV_{DDIO}$	V
High Level Output	V_{OH}	I_{OL} = 0.5 mA	$0.8xV_{DDIO}$	--	V_{DDIO}	V
Low Level Output	V_{OL}	I_{OH} = -0.5 mA	0	--	$0.2xV_{DDIO}$	V
Operating Current for V_{DD}	I_{DD}	Note 5	--	40	45	mA
Sleep Mode Current for V_{DD}	$I_{DD, Sleep}$	--	--	-	1	mA

Note 5: V_{DD} = 3.3V, 100% Display Area Turn on.

Limit Switch

The last element that will compose our device is the button that will allow us to start the vibration sensor mode in our device. In our case, instead of the classic button we have decided to place a limit switch as it simplified the design of the housing in which the device will be placed.

Figure 44 - Limit Switch

When a limit switch is pressed, the movement of the actuator plunger causes the electrical contacts inside the switch to close (for a normally open circuit) or open (for a normally closed circuit) its electrical connection. In this way the microcontroller can detect whether or not it has been pressed.

Due to the way this button has been programmed in the code, we have configured it as normally open, so we must connect the NO pin to the corresponding pin from which we read the signal in the Arduino board and the ground pin to the ground provided by the Arduino board. The following diagram shows how to connect a NO Limit Switch to an Arduino board, although it is not the Arduino MKR Wifi 1010 model, this configuration is equivalent in all boards.

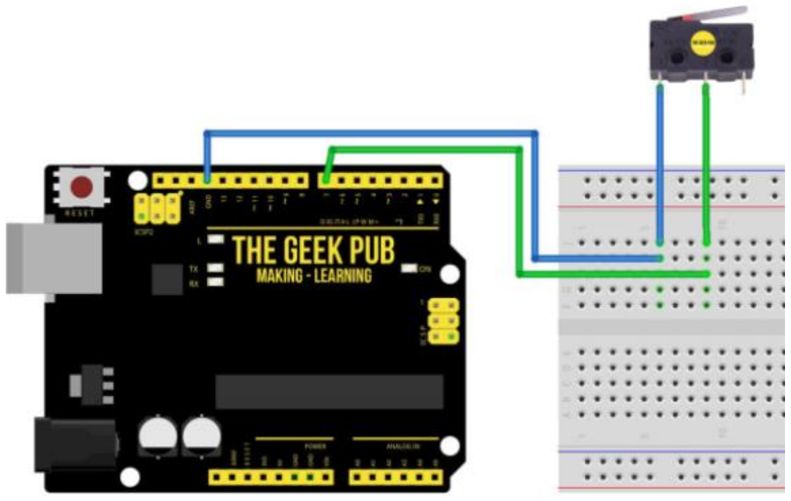
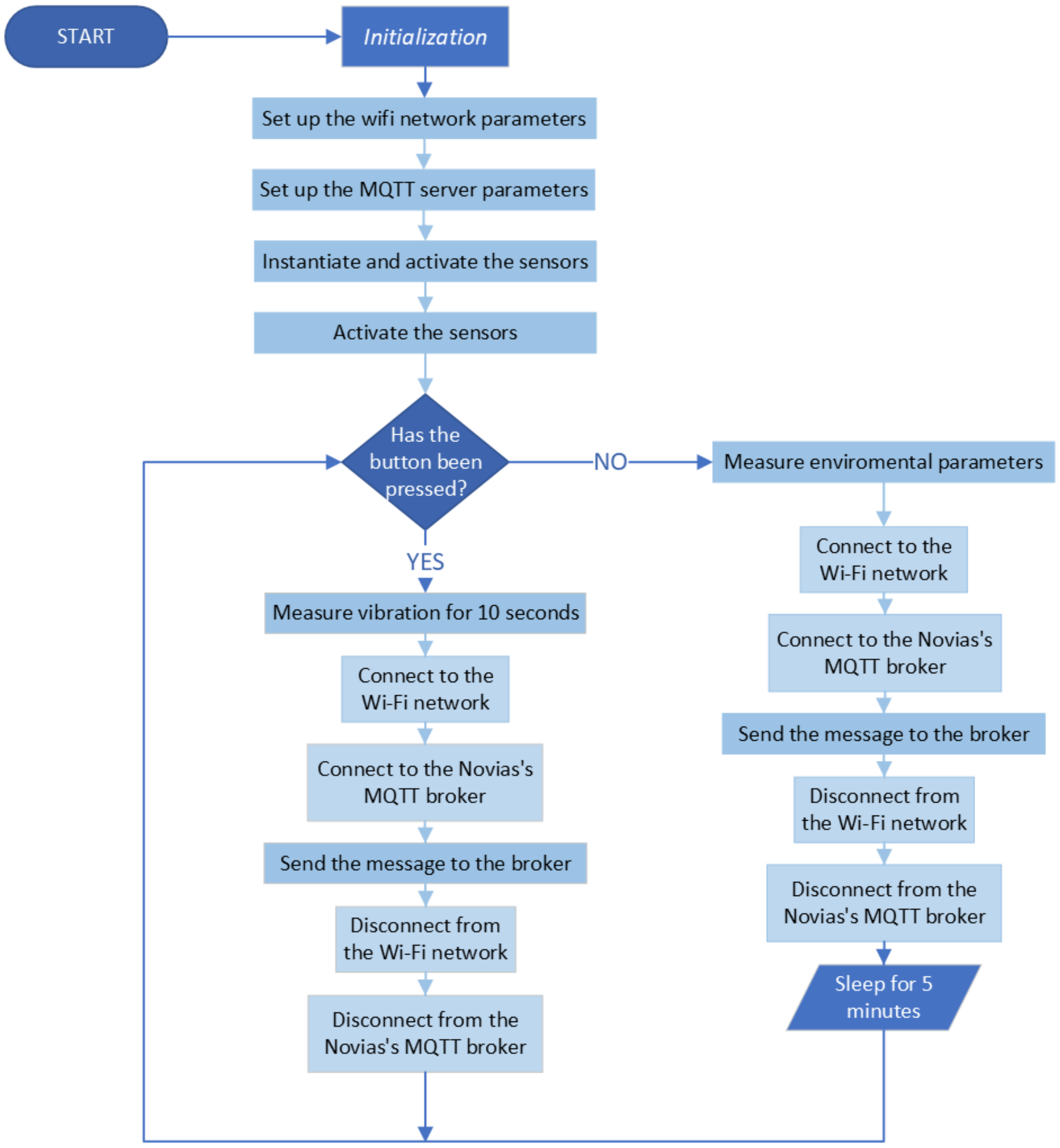


Figure 45- Connection of limit switch

Coding - MQTT Broker

As for the code, as explained above, we are looking for this device to update every five minutes the data of temperature, humidity, barometric pressure, IAQ and co2 level, and when a button is pressed, it will take vibration measurements for 10 seconds. All this information will be sent over Wi-Fi to the Novia's MQTT server so that anyone can access this information. Below is a flowchart summarising how the developed code works. It is quite simple so that it is not too complicated to understand, it should be noted that as the button is programmed for interruptions, as soon as it is pressed the device will start to measure the vibrations, it is not necessary to wait for any process to finish.


START
Initialization
Set up the wifi network parameters
Set up the MQTT server parameters
Instantiate and activate the sensors
Activate the sensors
Has the button been pressed?
NO
Measure enviromental parameters
YES
Measure vibration for 10 seconds
Connect to the Wi-Fi network
Connect to the Novias's MQTT broker
Send the message to the broker
Disconnect from the Wi-Fi network
Disconnect from the Novias's MQTT broker
Connect to the Wi-Fi network
Connect to the Novias's MQTT broker
Send the message to the broker
Disconnect from the Wi-Fi network
Disconnect from the Novias's MQTT broker
Sleep for 5 minutes

Communication between the device and the server will be via MQTT, which will be explained in more detail later. To begin with the explanation of the code, we will first explain how the two devices communicate with each other.

Communication between both boards

Both boards communicate over I2C, in order to communicate through the BHY2Host library. This library provided by Arduino is for Host boards, as in our case the Arduino MKR Wifi 1010, to interact with the BHY2 chip on Nicla Sense.

The first step to establish communication, is to upload the App.ino sketch provided by the BHY2 library to the Nicla board. In this code we should modify a parameter in the setup function, specifically we should add NICLA_BLE_AND_I2C and NICLA_AS_SHIELD to the BHY2.begin(). Function. The sketch is as following:

```
#include "Arduino.h"
#include "Arduino_BHY2.h"
// Set DEBUG to true in order to enable debug print
#define DEBUG false

void setup()
{
#if DEBUG
  Serial.begin(115200);
  BHY2.debug(Serial);
#endif

  BHY2.begin(NICLA_BLE_AND_I2C, NICLA_AS_SHIELD);
}
void loop()
{
  // Update and then sleep
  BHY2.update(100);
}
```

The first parameter is to enable both BLE and I2C communication, and the second parameter is to configure the Arduino Nicla as a shield. These settings will internally adjust the communication parameters so it can communicate through the headers.

The next step is to upload the sketch Passthrough.ino sketch provided by the BHY2Host library to the Arduino MKR Wifi 1010. Again, we must modify some parameters, in particular, we must modify the function BHY2Host.begin() by adding true and NICLA_AS_SHIELD. This is how the code should look like:

```
#include "Arduino.h"
#include "Arduino_BHY2Host.h"
#ifdef ARDUINO_ARCH_MBED
#include "USB/PluggableUSBSerial.h"
arduino::USBSerial SerialUSB2(false);
#else
Serial_ SerialUSB2(USBDevice);
#endif
// Set DEBUG to true in order to enable debug print
#define DEBUG false
void setup()
{
  Serial.begin(115200);
  BHY2Host.beginBHY2Host.begin(true, NICLA_AS_SHIELD);
}

void loop()
{
  BHY2Host.update();
}
```

Once it has been checked that the communication between the two boards is correct, the following step is the developing the code that will be uploaded to both processors once they are connected, i.e. in total three sketches will be running on these boards.

The first point will be to select which measures we are interested in. As we have already explained, we will take measurements of six variables in total: temperature, humidity, barometric pressure, IAQ, CO2 level and acceleration, this last one is separated in turn into three: the one in the x-axis, the one in the y-axis and the one in the z-axis.

In order to do this, we must first set the variables where we are going to store their values, and then define the sensors object or, in other words, instantiate all the sensors we want to use. The syntax required this is: *SensorClass variablename(SENSOR_ID MACRO)*. As the following image shows:

```
Sensor tempSensor(SENSOR_ID_TEMP);
Sensor barSensor(SENSOR_ID_BARO);
Sensor humSensor(SENSOR_ID_HUM);
Sensor gasSensor(SENSOR_ID_GAS);
SensorBSEC bsec(SENSOR_ID_BSEC_LEGACY);
SensorXYZ accelerometer(SENSOR_ID_ACC);
```

Once we have done this, it will be possible to activate them. We will use the function sensor.begin(), which will be included in the setup. Then, in order to obtain the values, we will use the following commands depending on the sensor type:

- **Temperature, barometric pressure, and humidity:**

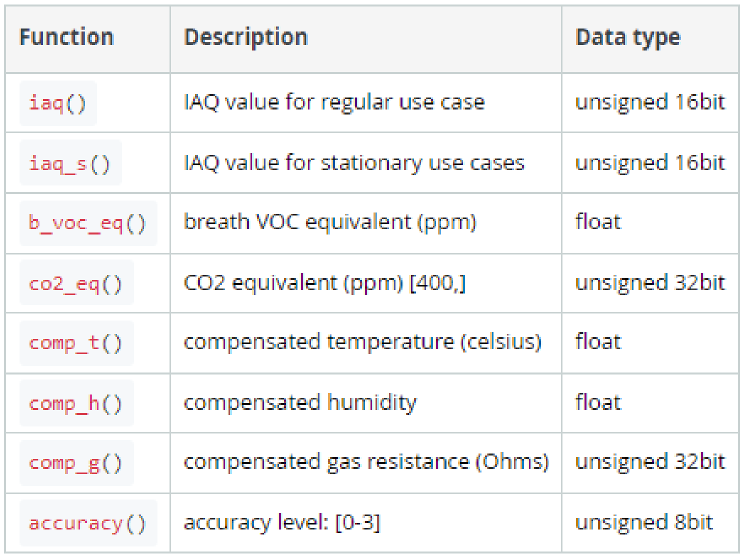
For these sensors the command we will use is sensor.value(), which will return a float with the sensor reading.

```
temp = ((int)(tempSensor.value()*10))/10.0;
bar = ((int)(barSensor.value()*10))/10.0;
hum = humSensor.value();
```

In addition, with the intention of not displaying a value with numerous decimal places we have decided to reduce the number of decimal places to one.

- **IAQ and CO2:**

It works in a different way to the previous ones The following table shows how we can obtain the different values depending on the environmental sensor parameter we are interested in.

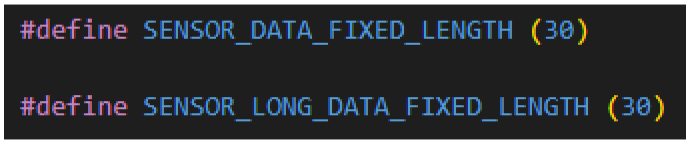


Function	Description	Data type
iaq()	IAQ value for regular use case	unsigned 16bit
iaq_s()	IAQ value for stationary use cases	unsigned 16bit
b_voc_eq()	breath VOC equivalent (ppm)	float
co2_eq()	CO2 equivalent (ppm) [400,]	unsigned 32bit
comp_t()	compensated temperature (celsius)	float
comp_h()	compensated humidity	float
comp_g()	compensated gas resistance (Ohms)	unsigned 32bit
accuracy()	accuracy level: [0-3]	unsigned 8bit

Since we are interested in knowing the value of the IAQ and the CO2 equivalence, these are the functions we should include:

```
iaq_= bsec.iaq();
co2 = bsec.co2_eq();
```

A note worth mentioning is that in order to see these values properly. It is necessary to modify two parameters in the file **SensorTypes.h** of the library **BHY2Host: t**he SENSOR_DATA_FIXED_LENGTH and the SENSOR_LONG_DATA_FIXED_LENGTH should be change to a larger value such as 30, because it needs to be greater than the frame size of 29 for the sensor bsec. This is due to the fact that most sensor frame size is less than 10, so these parameters default set to 10 to save space.



```
#define SENSOR_DATA_FIXED_LENGTH (30)

#define SENSOR_LONG_DATA_FIXED_LENGTH (30)
```

- **Acceleration:**

As we have already explained, this parameter is divided into three, one for each axis, to read each axis we must use the functions .x(), .y() and .z().

```
accX = accelerometer.x();
accY = accelerometer.y();
accZ = accelerometer.z();
```

After checking that all sensor readings are correct, the next step is to establish a connection to the broker and the Wi-Fi.

MQTT Broker

We will first look at what an MQTT broker is and how it fits into the publish subscriber architecture.

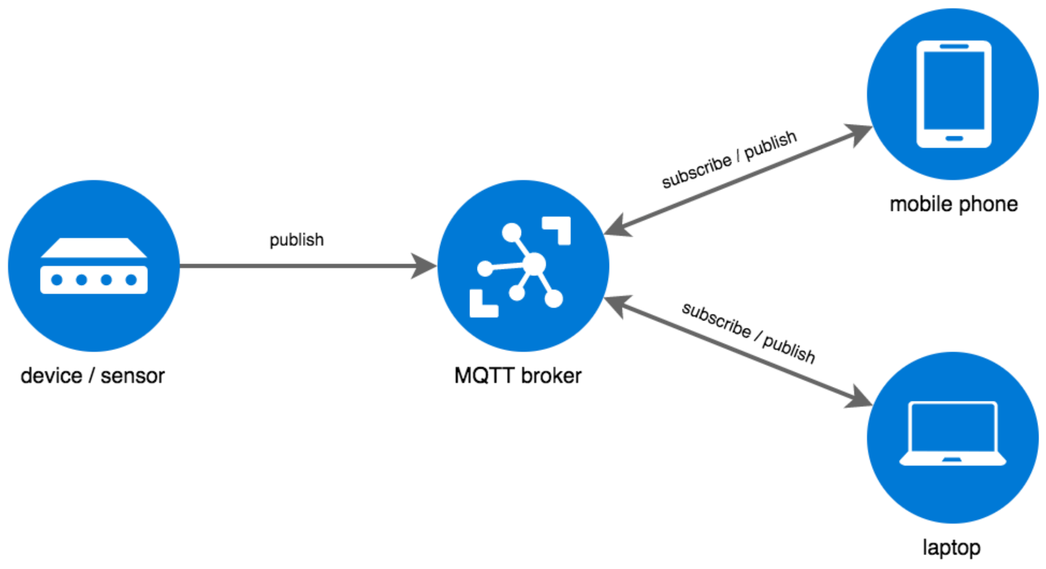
In MQTT protocol there are two main entities, one is the broker and the other one is the clients. A broker is simply software running on a computer which enables transaction between MQTT clients. These are the steps it follows to enable the transactions:

1. Allow clients to make a connection request
2. Authenticate the devices based on the connection information shared by the connecting clients.
3. Once authenticated, make sure that the device can send or receive messages to or from other devices securely using generally Transport Layer Security (TLS) encryption.
4. Stores messages within the server so that they could be re-sent in the case of unwanted connection loss, on client-connect, on client-disconnect, etc.

It can be of several types: on premise or in the cloud, self-built or 3rd party hosted. In our case this broker is provided by Novia University. The broker's name is **iot.novia.fi.**

Regarding clients, they are any type of device that is able to run an MQTT library and be connected to an MQTT broker over a network. Therefore, its functions are limited to connect themselves to a broker by using a username and a password and either subscribe or publish to a topic.

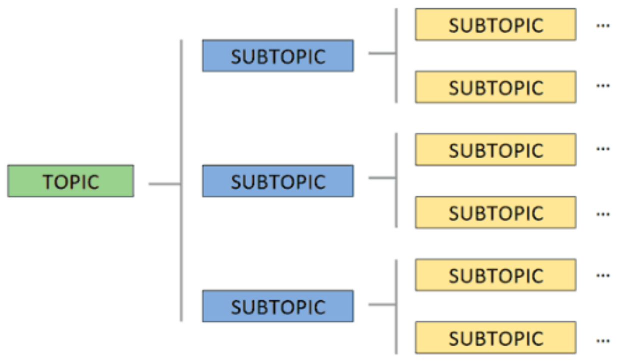
The following image shows a diagram of how this broker works:



As shown in the image, the sensor publishes to the broker while the mobile device or laptop can both publish and subscribe. This is what is known as the **publish-subscribe** model.

This model is different from the traditional client-server model used on HTTP, because now it works over TCP/IP. It distinguishes between the client (publisher) who transmits the message and the client (subscriber) who receives it. The publisher and subscriber do not need to communicate directly. A user (client) can post in a publication, and another user (client) who has subscribed to that publication (topic) will be notified that a new post is available for reading. We can allow many publishers to send messages to the same subscriber at the same time, or we can allow numerous subscribers to receive messages from the same publisher at the same time. The basic idea is that a middle role known as a broker is in charge of all message routing and delivery.

A fundamental component of the MQTT protocol is the **Topic**. This consists of a filter that the broker applies to the messages that are received by the broker from the publishers, and that is used to distinguish to which successful agents the message should be delivered. The topic simply consists of a UTF-8 text string, and a maximum length of 65536 characters. It may consist of one or more "levels" separated by a slash '/'. Each level must consist of one or more characters.



For example, examples of valid Topic could be:

House/Kitchen/Temperature

House /Livingroom/Temperature

The Broker accepts all Topics, even if not explicitly created before publishing or subscribing to the Broker. Clients, on the other hand, publish messages indicating a single Topic. The Broker receives the message and, if it finds a subscription that meets the Topic filter, it transmits the message to the subscribed clients.

An interesting tool of the MQTT is that a customer can subscribe to multiple Topic using Wildcards. We can use two of them:

- **Single-level wildcard:** The + character can be used to replace a single level anywhere in the Topic. For example, *House/+/Temperature* can be used for *House/Kitchen/Temperature* and *House /Livingroom/Temperature.*
- **Multi-level wildcard:** The # character can be used to replace any number of levels and can only be used at the end of the Topic. For instance, with the topic house/# we will receive all the topics starting with house/

In our device we will publish the following topics:

- **eps/environment:** In this topic we publish values of temperature, humidity, barometric pressure, IAQ and co2.
- **eps/Acceleration X:** In this topic we publish values of acceleration collected for 10 seconds in the x-axis
- **eps/Acceleration Y:** In this topic we publish values of acceleration collected for 10 seconds in the y-axis
- **eps/Acceleration Z:** In this topic we publish values of acceleration collected for 10 seconds in the z-axis

The library that we are going to use to MQTT messaging is the **PubSubClient** library. This library contains a client for handling simple publish/subscribe communications with a MQTT server. The library interacts with the underlying network hardware via the Arduino Ethernet Client API. This means it only works with an increasing number of boards and shields, such as the Arduino MKR Wifi 1010.

To send this MQTT data it is necessary that the device is connected to a Wi-Fi network over which it can send the information, for this we will make use of the **WIFININA** library. It is a very useful library because it enables network connections (both local and Internet) with the Arduino MKR WiFi 1010. It can act as both a server or a client, taking incoming connections or making outgoing ones and it supports WEP, WPA2 Personal, and WPA2 Enterprise encryptions.

Once both libraries are downloaded, they must be introduced in code.

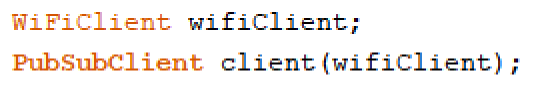
```
#include <WiFiNINA.h>
#include <PubSubClient.h>
```

Next, we define some constants to specify which server and wifi network we want to connect to.

```
char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;
char ssid_server[] = SECRET_SSID_MQTTserver;
char pass_server[] = SECRET_PASS_MQTTserver;

const char broker[] = "iot.novia.fi";
int port = 1883;
```

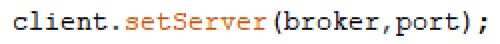
It is necessary to include both the name and password of the wifi network and the server, these data are included in the arduino_secrets.h folder as they are private data. It is also defined the name of the broker to which we want to connect, which, as we have already mentioned, is the one provided by the university.

After this, the next step is to define the MQTT client and the WIFI client as objects of the WiFIClient and PubSubClient classes. As we can see in the picture the MQTT client requires the network client as argument.



```
WiFiClient wifiClient;
PubSubClient client(wifiClient);
```

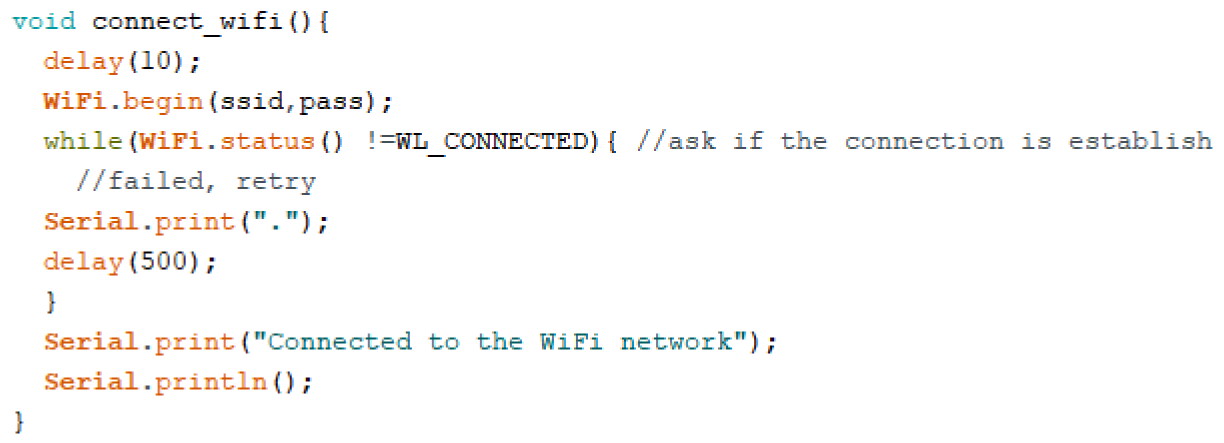
It is also necessary to set the server and port using the following function in the set up:



```
client.setServer(broker,port);
```

After all these steps, the only thing left to do is to perform the two functions that will allow us to connect to both the wifi and the server once they are called.

The following image shows the function we will use to connect to the Wi-Fi network. It is worth noting that in the images of the code the messages are included as if they were shown on the serial port, although in the final code they are shown on the display. This has been done to simplify the code shown in the screenshots.



```
void connect_wifi(){
  delay(10);
  WiFi.begin(ssid,pass);
  while(WiFi.status() !=WL_CONNECTED){ //ask if the connection is establish
    //failed, retry
  Serial.print(".");
  delay(500);
  }
  Serial.print("Connected to the WiFi network");
  Serial.println();
}
```

It tries to connect to the wifi through the WiFI.begin() function, then checks if it is connected or not, and if it is not, it tries again.

Next, the following image shows the function for connecting to the MQTT server.

```
void reconnect(){
  while(!client.connected()){
    Serial.println("Attempting to connect to the MQTT broker... ");
    if(client.connect("Wifi1010Client",ssid_server,pass_server)){
      Serial.println("Connected");
    }else{
      Serial.print("Failed to connect with state : ");
      Serial.print(client.state());
      Serial.println(" Attempting to connect again to the MQTT broker.");
      delay(5000);
    }
  }
}
```

It works in the same way as the Wi-Fi function as it tries to connect and if it does not succeed it tries again, with the addition that it indicates the reason why it has not been possible to establish the connection thanks to the function client.state(). It returns the state of the client. This can be used to receive further information about a connection attempt that fails. These are the following values that can be returned:

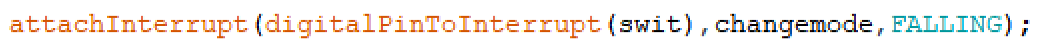
- -4: MQTT_CONNECTION_TIMEOUT - the server didn't respond within the keepalive time
- -3: MQTT_CONNECTION_LOST - the network connection was broken
- -2: MQTT_CONNECT_FAILED - the network connection failed
- -1: MQTT_DISCONNECTED - the client is disconnected cleanly
- 0: MQTT_CONNECTED - the client is connected
- 1: MQTT_CONNECT_BAD_PROTOCOL - the server doesn't support the requested version of MQTT
- 2: MQTT_CONNECT_BAD_CLIENT_ID - the server rejected the client identifier
- 3: MQTT_CONNECT_UNAVAILABLE - the server was unable to accept the connection
- 4: MQTT_CONNECT_BAD_CREDENTIALS - the username/password were rejected
- 5: MQTT_CONNECT_UNAUTHORIZED - the client was not authorized to connect

Button, interruptions and mode switching

Another essential part of the code is the button and how it switches from environmental sensor mode to vibration sensor mode. This button has been programmed with interrupts, allowing us to react to events external to the board, such as in this case pressing the button, in a fast way. Using interruptions we can quickly execute a piece of code that we are interested in after a specific event while stopping the execution of the code that was being executed.

By working in this way we can free up the Arduino processing and simplify the code as it doesn't have to constantly check if the pin connected to the button has been pressed.

It is not necessary to add any library for the use of interrupts, but just use the following function:



```
attachInterrupt(digitalPinToInterrupt(swit),changemode,FALLING);
```

This corresponds to the function **attachInterrupt(pin,ISR,mode),** which allows one of the pins to be configured as an interrupt port. The parameters of which it is composed are:

- **Pin:** It specifies which pin will be used as an interrupt. We must pass the ordinal, not the pin number. But in case of doubt, digitalPinToInterrupt(pin), returns the ordinal of the pin we want to utilize. In our case the pin we will use is 7, as it is the one to which we have connected the limit switch.

- **ISR:** it stands for Interrupt Service Routine and refers to the function that is called when an interrupt occurs. It's a special type because it doesn't take any parameters and does not return any value. In general, we should make ISR methods or functions as short and fast as possible as this can lead to deadlocks, in this way, functions such as delay() do not work. So, we have to adapt our code to these features.
  When an interrupt occurs in our device, it will call the changemode() function, which is as follows:

```
void changemode(){
  if (millis() - startTime > timeThreshold)
  {
    modevibration = !modevibration;
    startTime = millis();
    tiempol = millis();
  }
}
```

What this function does is to change the state of a boolean variable. This variable is called modevibration and has been declared as volatile, which is a reserved word that tells the compiler that these variables can change value at any time and, therefore, the compiler must reload the variable every time it is referenced somewhere. It must do this because what happens with other variables is that the compiler relies on some copy it may have in some register in the processor.

In this way, when this variable is false, the device will be working as an environmental sensor, and when the button is pressed, this variable will change to true and will start working as a vibration sensor until the 10 seconds of taking measurements are over, then it changes back to false and the device return to the environmental sensor.

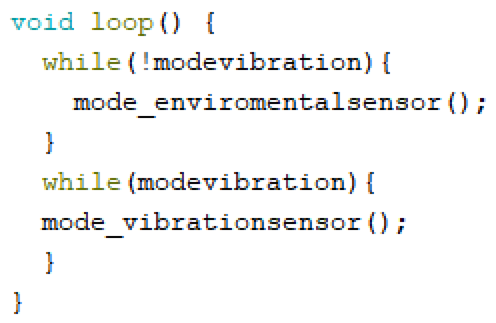
Some lines of code have also been added to eliminate the debounce caused by noise generated on the signal edges that can cause multiple triggers of an interrupt. To do this, we check the time between triggers of the interrupt. If the time is less than a certain time called timeThreshold we simply ignore the interrupt.

- **Mode:** defines when the interrupt should be triggered. It can take four constant values depending on what we want to do.
  - LOW: the interrupt will be triggered when the pin is in a low state.
  - CHANGE: the interrupt will be triggered when the pin changes from high to low, or from low to high.
  - RISING: the interrupt shall be thrown when the pin changes state from low to high.
  - FALLING: the interrupt will be thrown when the pin changes state from high to low.

Due to the operation we were interested in, we have chosen the falling mode.

Loop

The function that will be running continuously in our cards is very simple, it just checks the state of the boolean variable modevibration that we have explained before and depending on its state it decides whether to run the vibration mode or the environmental mode.



```
void loop() {
  while(!modevibration){
    mode_enviromentalsensor();
  }
  while(modevibration){
  mode_vibrationsensor();
  }
}
```

Reading and sending the data with ArduionJSON

To finish with the explanation of the code, we will see how the functions that define each of the modes work

Mode environmental sensor

The function that the device executes when it enters this mode is in the following image. Its operation is quite intuitive, first run BHY2.update() that will update the sensors value. It is necessary to call it continuously, because otherwise the reading of the IAQ and CO2 sensors will take too long to adjust themselves to the current environment.

The next is to set a stopwatch that will allow you to perform this cycle only once every five minutes, which will allow us to save battery.

The next thing is to establish a stopwatch that will allow us to do this cycle only once every five minutes, which will allow us to save battery. For them we save the milliseconds in which the previous cycle ended and if the difference of this time and the current time is greater than 5 minutes (3000000 milliseconds) then a new cycle runs.

```
void mode_enviromentalsensor(){
  BHY2Host.update();
  if (millis()-previousTime > 300000){
  temp = ((int)(tempSensor.value()*10))/10.0;
  bar = ((int)(barSensor.value()*10))/10.0;
  hum = humSensor.value();
  iaq_= bsec.iaq();
  co2 = bsec.co2_eq();

  update_Display();

  connect_wifi();
  if(!client.connected()) reconnect();
  Jsonmessageenvironement();
  client.loop();
  WiFi.disconnect();
  WiFi.end();
  client.disconnect();
  Serial.println("Wifi disconnected");

  previousTime=millis();
  }
```

In this cycle you do is first read the values of temperature, barometric pressure, humidity, IAQ and co2. Then update these new values on the display.

Once this is done, the following is to send the data to the MQTT server, to do this the board must connect first to the Wi-Fi network and then connect to the broker. It will send the data and then disconnect from the Wi-Fi and broker. Finally, it will store this time to be able to repeat the percentage in five minutes

The Jsonmessageenviroment() function is called to send the data. Before explaining it in more detail let's see what JSON and the Arduinojson library consists of.

JSON is a simple text-based open standard for data interchange. JSON is generally used to serialize and transport structured data across a network connection — data between a server and a client. It is frequently used in public data services like APIs (Application Programming Interfaces) and web services that provide public data.

Interchanging information as a plain text may appear unusual, as it necessitates more work in data interpretation (parsing) prior to receiving a binary stream, for example. However, it has the advantage of being compatible with a variety of systems.

Data is formatted in a specific way in JSON. JSON has the following syntax and uses symbols such as {}.: " " []. To express data in JSON the following syntax must be followed:

- Key/value pairs are used to represent data.
- The colon (:) gives hey a value.
- Commas (,) are used to separate key7value pairs.
- Objects are held in place by curly brackets ({ })
- Arrays are held in square brackets ([ ])

To express data in JSON, for example, the key/value pairs are as follows:

{"key1":"value1", "key2":"value2", "key3":"value3}

But we can also apply this in the actual world, for example you could want to organize data on a user as follows:

{"name":"Celia", "country": "Spain", "age":21}

The issue now is how to operate (serialize and deserialize) with Json files using an Arduino-compatible CPU. This is when the excellent **Arduino Json** library comes in handy. Json files can be written (serialized) or read/parsed (deserialized) in almost all languages. The Arduino environment is no exception, and we have the Arduino Json library, which includes routines for quickly serializing and deserializing objects.

Multiple development boards (Arduino Uno, Nano, Mega, Micro, Leonardo, Due, ESP8266, ESP32...) and our Arduino MKR Wifi 1010 are compatible with Arduino Json.

Once explained this, we can already see what the Jsonmessageenviroment() function consists of. First, we must define which json document we want to send. We want to send the data from the five sensors together, so should have five keys:

{"Temperature: “: temp, "Humidity: ": hum, "Barometic pressure":bar, "IAQ: ": iaq_, "CO2":co2}

```
void Jsonmessageenvironement(){
  const size_t CAPACITY = JSON_OBJECT_SIZE(10);
  StaticJsonDocument<CAPACITY> doc;
  Serial.println(CAPACITY);
  JsonObject object = doc.to<JsonObject>();
  object["Temperature"] = String(temp+String("C"));
  object["Humidity"] = String(hum+String("%"));
  object["Barometic pressure"] = String(bar+String("hPa"));
  object["IAQ"] = String(iaq_);
  object["CO2"] = String(co2+String("ppm"));

  char bufferE[256];
  serializeJson(doc, bufferE);
  if (client.publish("eps/environement", bufferE)==true){
     Serial.println("Success sending message");
    } else {
      Serial.println("Error sending message");
    }
  delay(1000);
}
```

First, the Json document that we want to send is built with a capacity to accommodate 8 elements, for this we will use the macro JSON_ARRAY_SIZE(n), which returns the number os bytes requires to store an array that contains n elements.

After this, it is necessary to serialize it in a temporary buffer before publishing it to a topic mqtt. Once this is done, it is published it to the topic eps/environement and check whether or not the shipment has been successful.

Mode vibration sensor

Finally, we will see how it works the function, which is running on the device when it works as a vibration sensor. It also has an intuitive operation. First it notifies by screen that it is measuring the vibration, then it will call the function Jsonmessagevibration() that we will explain later and finally it changes of state the variable boolean modevibration to false in order to return to the mode of environmental sensor.

```
void mode_vibrationsensor(){
  Serial.println("Measuring vibration");
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_5x8_tf);
  u8g2.drawStr(3,10,"EPS IoT Team");
  u8g2.drawLine(0,12,128,12);
  u8g2.setFont(u8g2_font_9x18_mf);
  u8g2.drawStr(20,35,"MEASURING");
  u8g2.drawStr(20,55,"VIBRATION");
  u8g2.setBitmapMode(false /*solid*/);
  u8g2.sendBuffer();

  Jsonmessagevibration();

  Serial.println("Return to the enviromental sensor");
  modevibration = !modevibration;
}
```

The following image shows how does the Jsonmessagevibration() works. First create 3 json documents to save the acceleration data for each of the axes. It has been decided to separate each axis in a document to reduce the size of these and not pose a problem when sending them. Then with JsonArray we create a reference to an array that reside in their respective JsonDocument. Later it is set a for which allows to make a total of 75 measurements of vibration, which is equivalent to about ten seconds. In this for loop we first call BHY2Host.update(), later we save these values in each of the variables and then we make an adjustment so that these values are in the unit g and with a limit of 4 decimal places.

After finishing this for loop we connect to both the wifi and the mqtt broker, later we create the temporary buffer necessary to serialize the documents. Here we must take into account that the PubSubClient library limits the size to 256, so we must modify this value to 1300 in the document **PubSubClient.h** so that it has enough capacity to send 76 documents

```
#ifndef MQTT_MAX_PACKET_SIZE
#define MQTT_MAX_PACKET_SIZE 1300
#endif
```

After this we will serialize each json document in a different buffer and send it to mqtt broker each under a different topic. Once this is done just disconnect the device from t the wifi and server.

```
void Jsonmessagevibration(){

  const size_t CAPACITY = JSON_ARRAY_SIZE(76);
  StaticJsonDocument<CAPACITY> AXdoc;
  StaticJsonDocument<CAPACITY> AYdoc;
  StaticJsonDocument<CAPACITY> AZdoc;

  JsonArray aX = AXdoc.to<JsonArray>();
  JsonArray aY = AYdoc.to<JsonArray>();
  JsonArray aZ = AZdoc.to<JsonArray>();

  for (int i=0;i<=75;i++){
  BHY2Host.update();
  accX = accelerometer.x();
  accY = accelerometer.y();
  accZ = accelerometer.z();
  aX.add(((int)(accX/65536.0*16*1000))/1000.0);
  aY.add(((int)(accY/65536.0*16*1000))/1000.0);
  aZ.add(((int)(accZ/65536.0*16*1000))/1000.0);
  }
  serializeJson(AXdoc, Serial);
  Serial.println("");
  serializeJson(AYdoc, Serial);
  Serial.println("");
  serializeJson(AZdoc, Serial);
  Serial.println("");

  connect_wifi();
  delay(100);
  if(!client.connected()) reconnect();

  char bufferAX[1500];
  char bufferAY[1500];
  char bufferAZ[1500];

  serializeJson(AXdoc, bufferAX);
  serializeJson(AYdoc, bufferAY);
  serializeJson(AZdoc, bufferAZ);
```

```
  delay(2000);
  if (client.publish("eps/Acceleration X", bufferAX)==true){
   Serial.println("Success sending Acceleration X");
  } else {
    Serial.println("Error sending Acceleration X");
  }
  delay(2000);
   if (client.publish("eps/Acceleration Y", bufferAY)==true){
   Serial.println("Success sending Acceleration Y");
  } else {
    Serial.println("Error sending Acceleration Y");
  }
  delay(2000);
   if (client.publish("eps/Acceleration Z", bufferAZ)==true){
   Serial.println("Success sending Acceleration Z");
  } else {
    Serial.println("Error sending Acceleration Z");
  }
  delay(1000);
  client.loop();
  WiFi.disconnect();
  WiFi.end();
  client.disconnect();
}
```

```
void Jsonmessagevibration(){

  const size_t CAPACITY = JSON_ARRAY_SIZE(76);
  StaticJsonDocument<CAPACITY> AXdoc;
  StaticJsonDocument<CAPACITY> AYdoc;
  StaticJsonDocument<CAPACITY> AZdoc;

  JsonArray aX = AXdoc.to<JsonArray>();
  JsonArray aY = AYdoc.to<JsonArray>();
  JsonArray aZ = AZdoc.to<JsonArray>();

  for (int i=0;i<=75;i++){
  BHY2Host.update();
  accX = accelerometer.x();
  accY = accelerometer.y();
  accZ = accelerometer.z();
  aX.add(((int)(accX/65536.0*16*1000))/1000.0);
  aY.add(((int)(accY/65536.0*16*1000))/1000.0);
  aZ.add(((int)(accZ/65536.0*16*1000))/1000.0);
  }
  serializeJson(AXdoc, Serial);
  Serial.println("");
  serializeJson(AYdoc, Serial);
  Serial.println("");
  serializeJson(AZdoc, Serial);
  Serial.println("");

  connect_wifi();
  delay(100);
  if(!client.connected()) reconnect();

  char bufferAX[1500];
  char bufferAY[1500];
  char bufferAZ[1500];

  serializeJson(AXdoc, bufferAX);
  serializeJson(AYdoc, bufferAY);
  serializeJson(AZdoc, bufferAZ);
  delay(2000);
  if (client.publish("eps/Acceleration X", bufferAX)==true){
   Serial.println("Success sending Acceleration X");
  } else {
    Serial.println("Error sending Acceleration X");
  }
  delay(2000);
   if (client.publish("eps/Acceleration Y", bufferAY)==true){
   Serial.println("Success sending Acceleration Y");
  } else {
    Serial.println("Error sending Acceleration Y");
  }
  delay(2000);
   if (client.publish("eps/Acceleration Z", bufferAZ)==true){
   Serial.println("Success sending Acceleration Z");
  } else {
    Serial.println("Error sending Acceleration Z");
  }
  delay(1000);
  client.loop();
  WiFi.disconnect();
  WiFi.end();
  client.disconnect();
}
```

MQTT Explorer

Turning to the issue of how we will check that the data is actually being sent to the server we have decided to use MQTT Explorer. MQTT Explorer is a powerful MQTT client that gives you a logical overview of your MQTT topics and makes interacting with devices and services on your broker simple. Here are some of its main features:

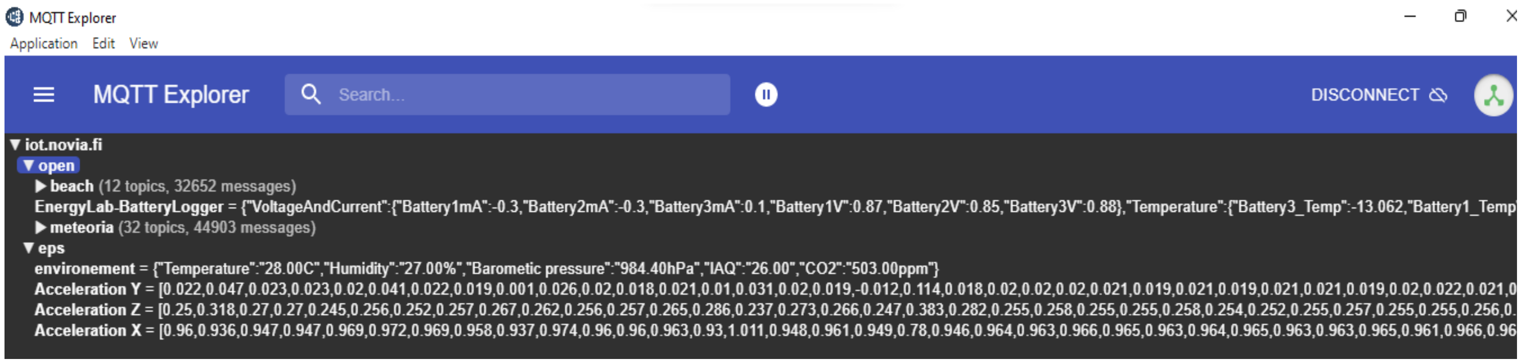
- Visualize topics and topic activity.
- Remove any topics that have been saved.
- Topics can be searched and filtered.
- Recursively delete topics
- A comparison of the current and previous messages received
- Publish topics.

- Plot numeric topics
- Keep track of each topic's history.

Compared to others MQTT clients, the MQTT explorer stands out from the rest MQTT clients like MQTTLens, MQTTBox or MQTT.fx due to his hierarchical view.

Despite MQTT Explorer can meet most development needs, but it also has some disadvantages such as the fact that connection can exist at a time, which is not convenient for multiple-connections debugging.

Below is an example of what you would see when this device is running:



```
▼ iot.novia.fi
 ▼ open
   ▶ beach (12 topics, 32652 messages)
   EnergyLab-BatteryLogger = {"VoltageAndCurrent":{"Battery1mA":-0.3,"Battery2mA":-0.3,"Battery3mA":0.1,"Battery1V":0.87,"Battery2V":0.85,"Battery3V":0.88},"Temperature":{"Battery3_Temp":-13.062,"Battery1_Temp
   ▶ meteoria (32 topics, 44903 messages)
 ▼ eps
   environement = {"Temperature":"28.00C","Humidity":"27.00%","Barometic pressure":"984.40hPa","IAQ":"26.00","CO2":"503.00ppm"}
   Acceleration Y = [0.022,0.047,0.023,0.023,0.02,0.041,0.022,0.019,0.001,0.026,0.02,0.018,0.021,0.01,0.031,0.02,0.019,-0.012,0.114,0.018,0.02,0.02,0.02,0.021,0.019,0.021,0.019,0.021,0.021,0.019,0.02,0.022,0.021,0
   Acceleration Z = [0.25,0.318,0.27,0.27,0.245,0.256,0.252,0.257,0.267,0.262,0.256,0.257,0.265,0.286,0.237,0.273,0.266,0.247,0.383,0.282,0.255,0.258,0.255,0.255,0.258,0.254,0.252,0.255,0.257,0.255,0.255,0.256,0.
   Acceleration X = [0.96,0.936,0.947,0.947,0.969,0.972,0.969,0.958,0.937,0.974,0.96,0.96,0.963,0.93,1.011,0.948,0.961,0.949,0.78,0.946,0.964,0.963,0.966,0.965,0.963,0.964,0.965,0.963,0.963,0.965,0.961,0.966,0.96
```

Coding - Arduino IoT Cloud

Introduction to the Arduino IoT Cloud

In a search of the different possibilities that the Arduino MKR Wifi 1010 has, it was found that it was a board compatible with the Arduino IOT cloud. So, we decided to take the opportunity to work with this service provided by Arduino, as it can be very useful for IoT applications.

Specifically, it is a cloud compatible board that can be connected to the Arduino IOT Cloud via Wi-Fi. This type of connection is a simple alternative in which credentials can be entered securely during project setup thanks to the fact that all Arduino cloud compatible boards come with a hardware security element (ECC508 cryptochip). It is a very suitable connection for short-range projects where the board is connected to the cloud via your home or work router, for example.

First, we will explain what the Arduino IoT Cloud is all about. It is a service provided directly by Arduino that allows you to configure, program and deploy your devices. It simplifies IoT development, allows you to build visual dashboards to monitor and control your devices, integrate with other services and much more. Below is a list of Arduino IoT Cloud features, taken directly from the official arduino website:

- **Data Monitoring** - learn how to easily monitor your Arduino's sensor values through a dashboard.
- **Variable Synchronisation** - variable synchronisation allows you to sync variables across devices, enabling communication between devices with minimal coding.
- **Scheduler** - schedule jobs to go on/off for a specific amount of time (seconds, minutes, hours).
- **Over-The-Air (OTA) Uploads** - upload code to devices not connected to your computer.
- **Webhooks** - integrate your project with another service, such as IFTTT.
- **Amazon Alexa Support** - make your project voice controlled with the Amazon Alexa integration.
- **Dashboard Sharing** - share your data with other people around the world.

As we have mentioned, it allows you to create visual dashboards in a simple way, allowing you to effortlessly create an attractive template on which to display certain data of interest such as readings from various sensors, adjust a range of values with a slider or activate or deactivate things with buttons or switches.

On the dashboard, widgets can be added individually and linked to variables, and variables can be assigned to widgets automatically by adding a Thing.

These are visual representations of variables, for example you can set a Gauge Widget for a variable that stores temperature readings, or a Chart Widget to monitor energy consumption. Depending on the type of variable the widget can be of two types:

- Read-only cannot be interacted with (e.g. graphs).
- Read and write can be interacted with (e.g., sliders, speed controllers)

The following image shows all the widgets that can be added to our dashboards:

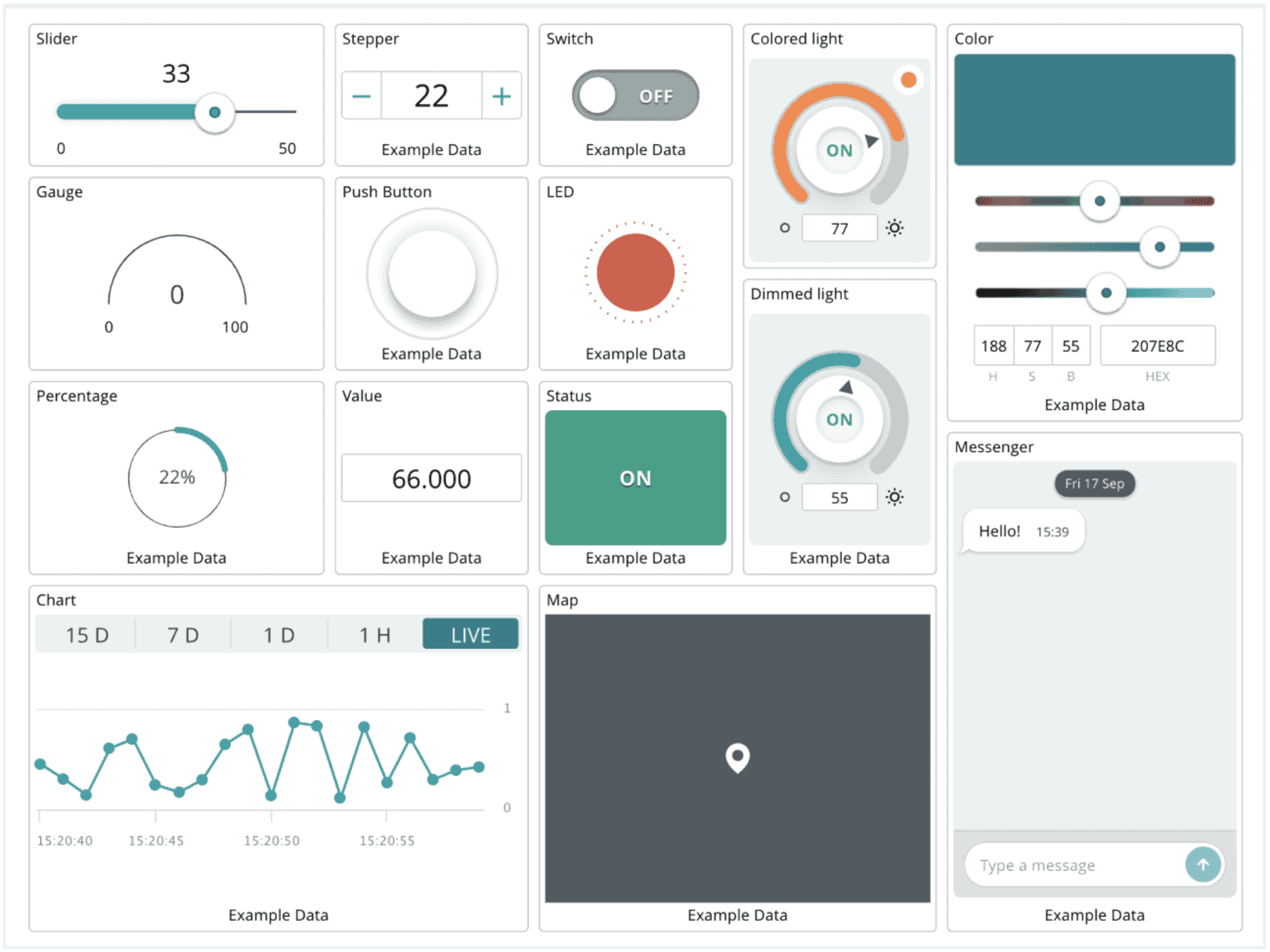


Figure 46- Example of possible widgets

An example of a project using this platform is the following, a soil moisture meter and an automated irrigation system.

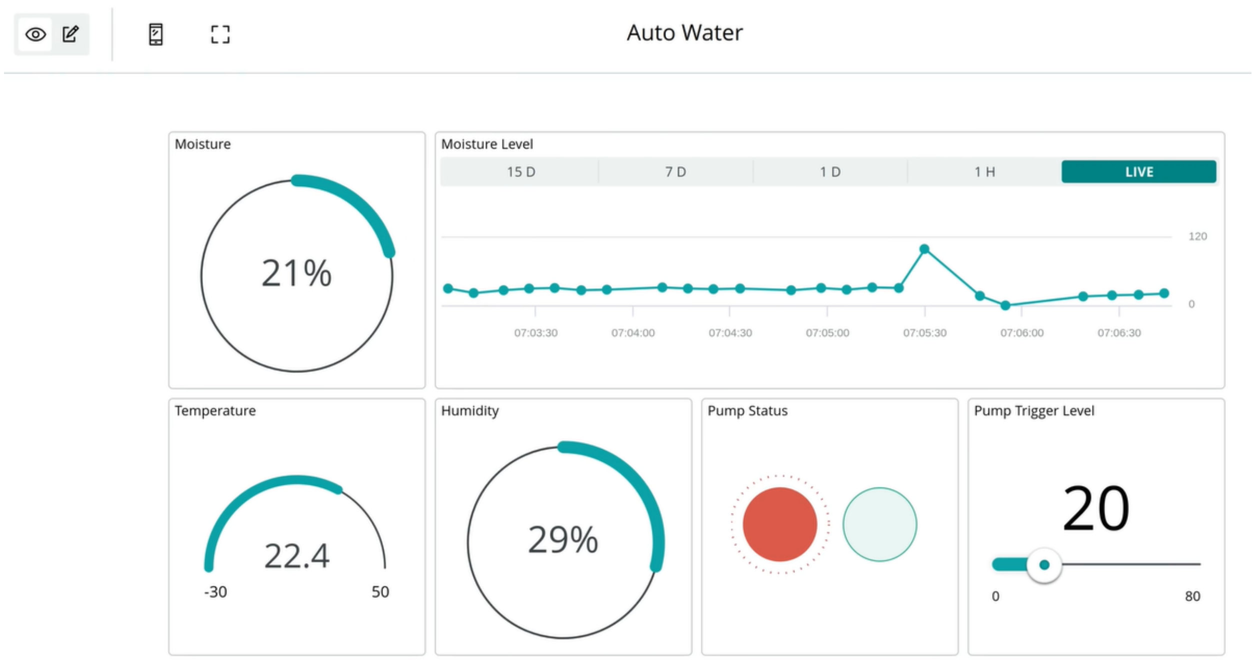
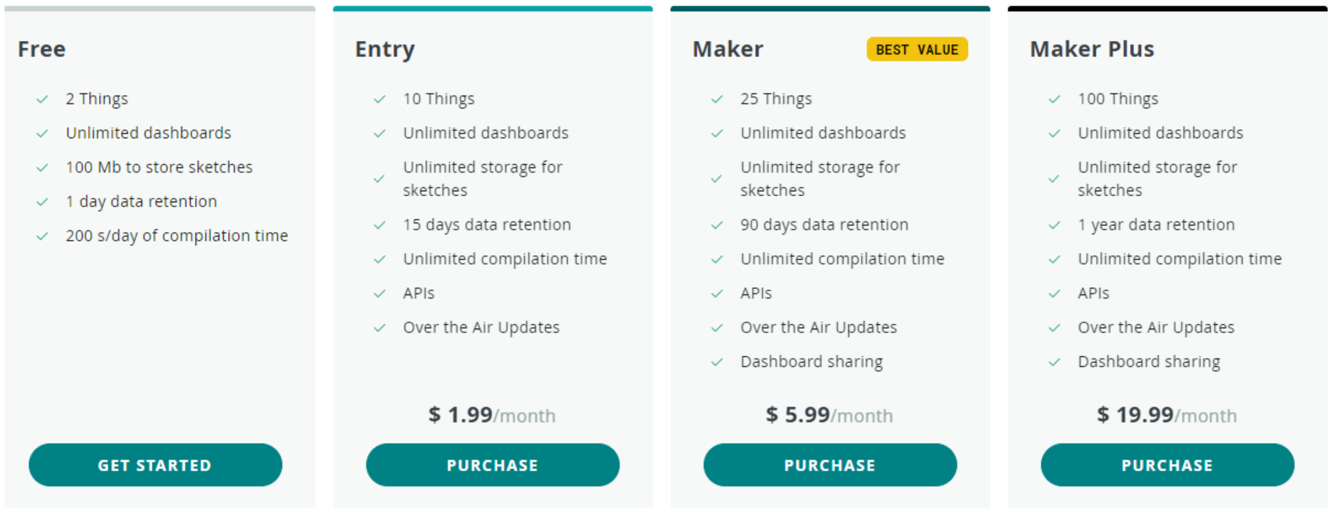


Figure 47- Example of widgets display

It should be noted that the free version of this platform is quite limited, and in case we would like to add more variables or more devices, we would have to pay for its use. Considering all the resources and possibilities at its disposal, it is not an expensive product, although it is an aspect to be taken into account.



Free

- 2 Things
- Unlimited dashboards
- 100 Mb to store sketches
- 1 day data retention
- 200 s/day of compilation time

GET STARTED

Entry

- 10 Things
- Unlimited dashboards
- Unlimited storage for sketches
- 15 days data retention
- Unlimited compilation time
- APIs
- Over the Air Updates

$ 1.99/month

PURCHASE

Maker BEST VALUE

- 25 Things
- Unlimited dashboards
- Unlimited storage for sketches
- 90 days data retention
- Unlimited compilation time
- APIs
- Over the Air Updates
- Dashboard sharing

$ 5.99/month

PURCHASE

Maker Plus

- 100 Things
- Unlimited dashboards
- Unlimited storage for sketches
- 1 year data retention
- Unlimited compilation time
- APIs
- Over the Air Updates
- Dashboard sharing

$ 19.99/month

PURCHASE

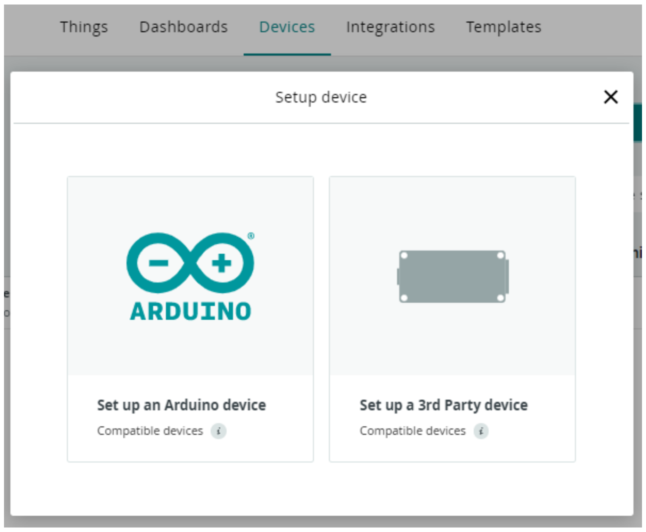
Figure 48- Subscription prices

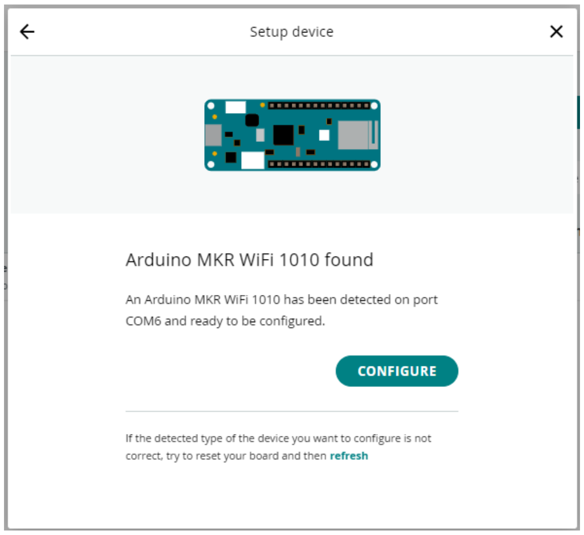
Using the Arduino IoT Cloud in our project

Due to the nature of vibration data capture, which requires sending a packet of data rather than a single value, it is not compatible with this platform, so this section is focused to the device in environmental sensor mode.

As in the previous case, the communication between the two boards is done through I2C, so both will communicate through the BHYHost library. Again, it will be necessary to upload the App.ino sketch to the Nicla Sense ME board and the Passtrough.ino sketch to the MKR Wifi 1010 board.

Once we have verified that both boards are communicating correctly, the next step is to start creating our dashboard on the Arduino Iot Cloud platform. Firstly, it is necessary to add the device with which we are going to communicate. In our case it will be the Arduino MKR Wifi 1010.

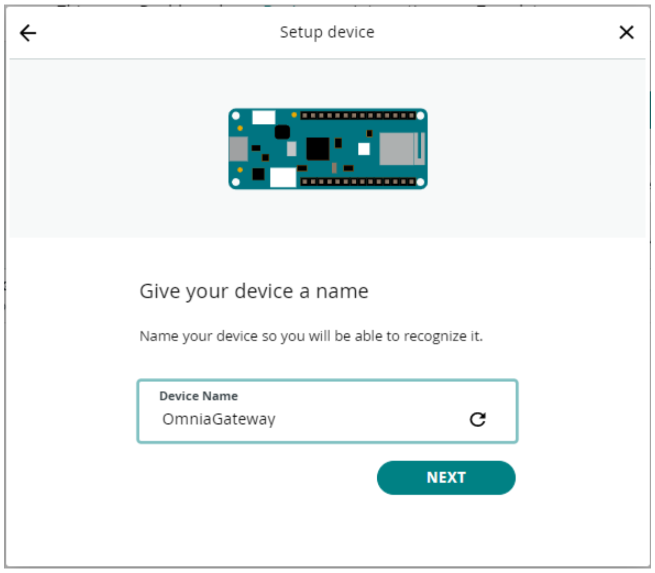




Arduino MKR WiFi 1010 found

An Arduino MKR WiFi 1010 has been detected on port COM6 and ready to be configured.

If the detected type of the device you want to configure is not correct, try to reset your board and then refresh



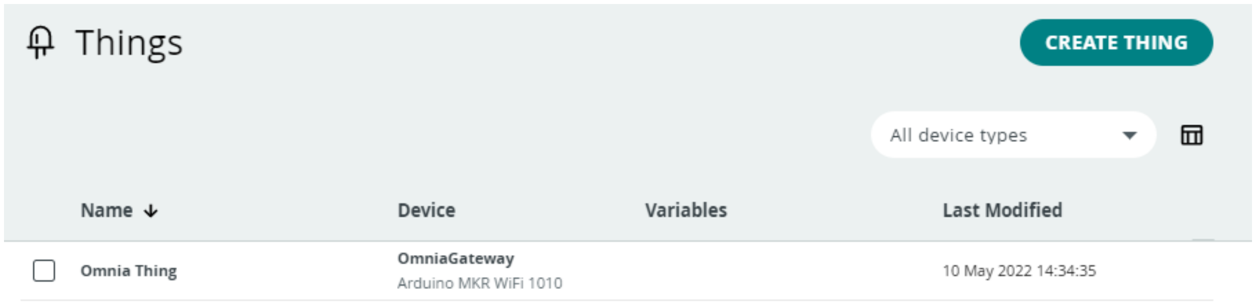
Give your device a name

Name your device so you will be able to recognize it.

Device Name
OmniaGateway

Figure 49 - Instruction to set up an Arduino in the Cloud

The next step is to add a Thing. It is an abstract concept coming from the Internet of things concept, it contains the configuration of the variables and other settings, as well as the history of the data collected for the variables. A Thing needs to be associated with each of the devices used in the Arduino IoT Cloud. In our case, as we will only have one device, it will only be necessary to create one Thing, we called it Omnia Thing.

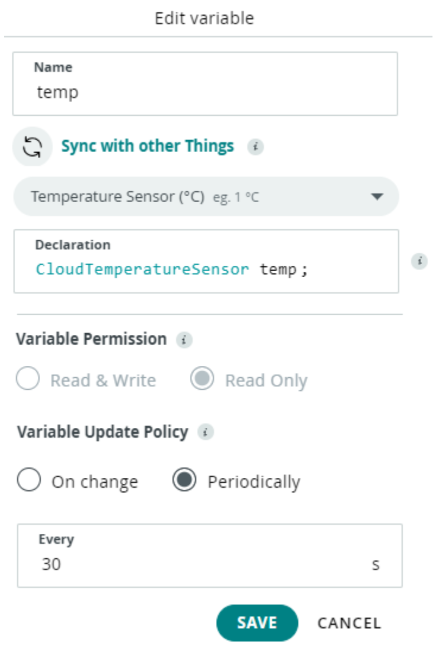


Name	Device	Variables	Last Modified
Omnia Thing	OmniaGateway Arduino MKR WiFi 1010		10 May 2022 14:34:35

Figure 50- Display of devices in Arduino Cloud

The next step is to declare all the variables that are of interest to the project, in our case they will be temperature, humidity, barometric pressure, air quality index and CO2 level.

The following images show how the configuration should be done in the case of temperature. As we can see in the image, we have selected that this variable is updated periodically every 30 seconds, so we avoid that it is constantly updated and with them we save battery. The same procedure is followed for the rest of the variables.



Name: temp

Temperature Sensor (°C) eg. 1 °C

Declaration
```
CloudTemperatureSensor temp;
```

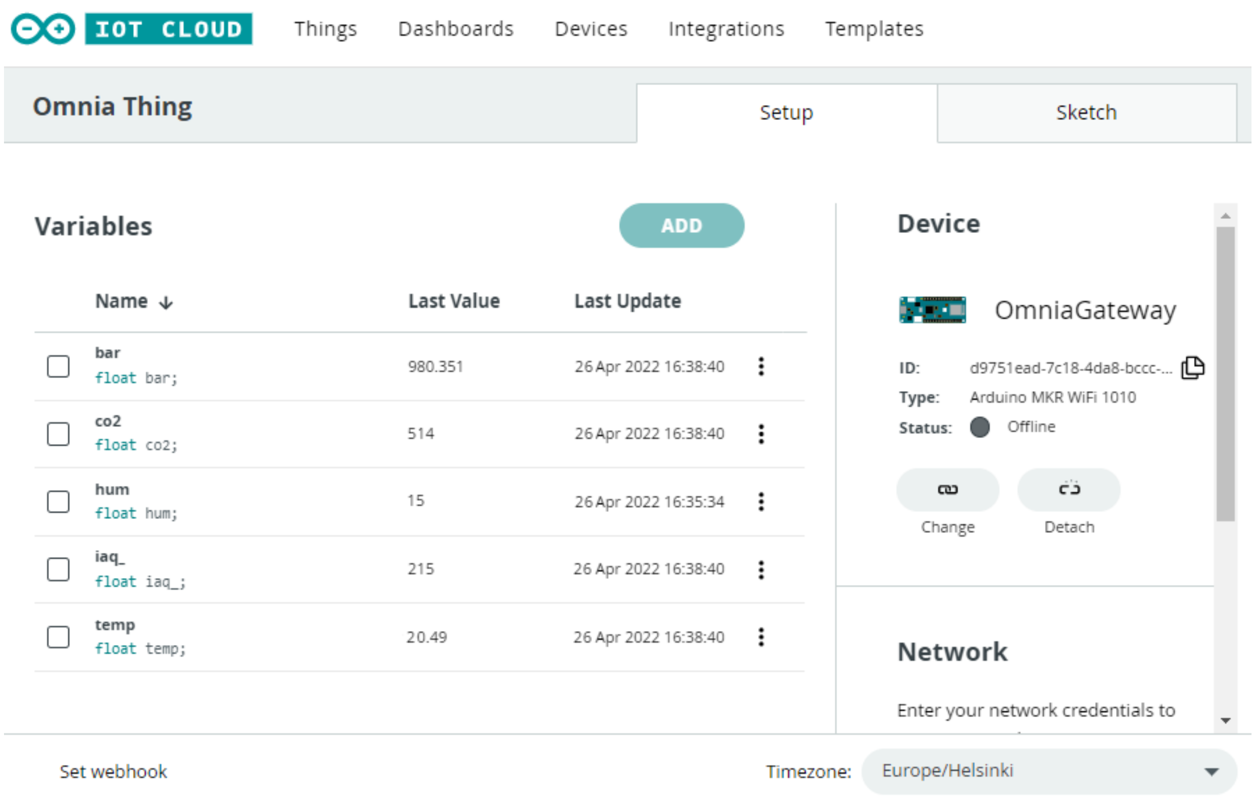
Variable Permission: Read & Write [✓ Read Only]

Variable Update Policy: On change [✓ Periodically]

Every: 30 s

Figure 51- configuration of Arduino cloud

All the variables must also be connected to a device, in our case the Arduino MKR Wifi 1010, in order to know this ID. This number is of great importance as it is the "name" that this variable receives, and it will be the one we will use when we want to refer to it in the Arduino sketch. Once configured and all the variables of our interest we see the following:



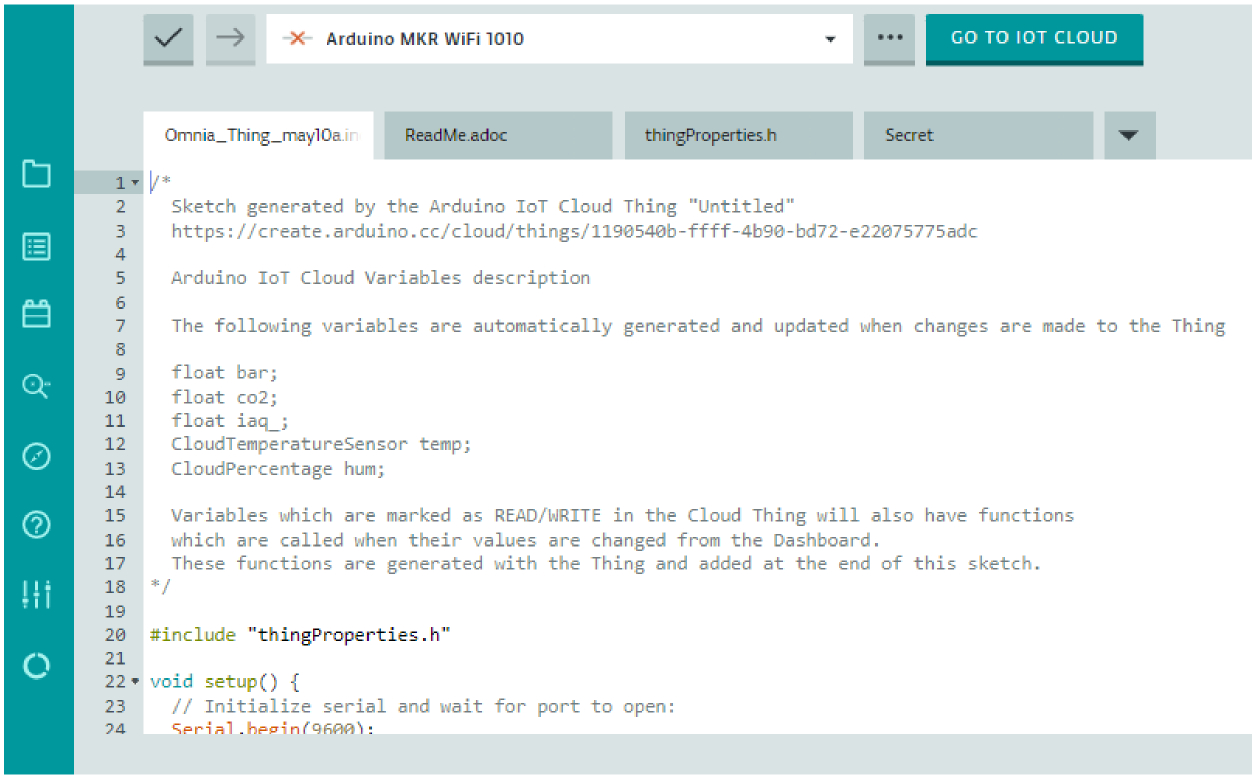
Name	Last Value	Last Update
bar `float bar;`	980.351	26 Apr 2022 16:38:40
co2 `float co2;`	514	26 Apr 2022 16:38:40
hum `float hum;`	15	26 Apr 2022 16:35:34
iaq_ `float iaq_;`	215	26 Apr 2022 16:38:40
temp `float temp;`	20.49	26 Apr 2022 16:38:40

OmniaGateway
ID: d9751ead-7c18-4da8-bccc-...
Type: Arduino MKR WiFi 1010
Status: ● Offline

Enter your network credentials to

Timezone: Europe/Helsinki

The next step is to start with the programming of the Arduino sketchOne of the best things about the Arduino IoT cloud is that it is able to autogenerate a code for your project based on your board type and your "thing". This would be what we would see from the IoT cloud with respect to the code.



```
/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/1190540b-ffff-4b90-bd72-e22075775adc

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes are made to the Thing

  float bar;
  float co2;
  float iaq_;
  CloudTemperatureSensor temp;
  CloudPercentage hum;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include "thingProperties.h"

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
```

In addition, we have also used the **Nicla_IoT_Bridge.ino** example provided by the Arduino library BHY2Host.h to develop our own code.

This code is based on three separate files Arduino_secrets.h, NiclaIoTCloud.ino and thingProperties.h.

We will now explain each of these separately:

- **Arduino_secrets.h**

This is the simplest file of all, it is very similar to the one explained in the previous case with the MQTT Broker, it is limited only to store the information about the name and password of our Wifi network.

```
#define SECRET_SSID "Wifi_name"
#define SECRET_PASS "Wifi_password"
```

- **thingProperties.h**

This file includes a fundamental library for this code, which is the ArduinoIoTCloud library. It will allow us to make this connection between the Nicla, the MKR Wifi board and the Arduino IoT Cloud. In fact, this library allows connecting to the Arduino IoT Cloud service.

In addition, the different variables that we are going to use are also declared, the declaration of each of them depends on their nature and how we have configured them in the cloud.

An important aspect is that here we must declare the ID of our variables that as we have indicated we get directly from our IoT Cloud, in this way we get to link our variable in the code with the variable in the cloud. The IDs of our variables are the following ones:

```
const char THING_ID_temp[] = "2cf5aca7-31c2-4274-99eb-568a35a82023";
const char THING_ID_hum[] = "feeb2c09-11f7-41c4-85f1-66ae35e2eb27";
const char THING_ID_iaq[] = "3904a5f7-d9c0-43d9-ad76-d791c1eeda37";
const char THING_ID_bar[] = "175536c1-4719-4860-9019-bfc0d88f149c";
const char THING_ID_co2[] = "d68cd91e-7b34-47f8-ac4f-db6849ec3c6d";
```

We also add the function that will relate the IDs declared before, with the variables declared in the code and with the properties chosen for them by us in their configuration within the cloud.

```
void initProperties(){

  ArduinoCloud.setThingId(THING_ID_temp);
  ArduinoCloud.setThingId(THING_ID_hum);
  ArduinoCloud.setThingId(THING_ID_iaq);
  ArduinoCloud.setThingId(THING_ID_bar);
  ArduinoCloud.setThingId(THING_ID_co2);
  ArduinoCloud.addProperty(bar, READ, 30 * SECONDS, NULL);
  ArduinoCloud.addProperty(co2, READWRITE, 30 * SECONDS, onCo2Change);
  ArduinoCloud.addProperty(iaq_, READWRITE, 30 * SECONDS, onIaqChange);
  ArduinoCloud.addProperty(temp, READ, 30 * SECONDS, NULL);
  ArduinoCloud.addProperty(hum, READWRITE, 30 * SECONDS, onHumChange);

}
```

Finishing with the key parts of this file, it remains to include the function that will enable communication via Wi-Fi between the MKR Wi-Fi 1010 board and the cloud. This function is part of the **<Arduino_ConnectionHandler.h>** library which is the Wi-Fi connection handler that is used to manage the Wi-Fi connection. Is the WiFiConnectionHandler and it oversees initialize the connection manager using the Wi-Fi Access Point.

```
WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

- **NiclaIoTCloud.ino**

This file is where we will place the main part of our code, that is to say, where we will make the setup and where will develop the loop that will be executing continuously our board.

First, we start by adding all the necessary libraries, where in addition to those already mentioned we include Arduino_BHY2Host.h, U8g2lib.h, U8x8lib.h and Wire.h, which were already explained in the previous section for the case with MQTT communication. The next step is to instantiate all the sensor object we are interested in, as we also show in the previous case the syntax used in order to do this is: *SensorClass variablename(SENSOR_ID MACRO)*. The ID of each sensor can be found in the Arduino documentation.

```
Sensor tempSensor(SENSOR_ID_TEMP);
Sensor barSensor(SENSOR_ID_BARO);
Sensor humSensor(SENSOR_ID_HUM);
Sensor gasSensor(SENSOR_ID_GAS);
SensorBSEC bsec(SENSOR_ID_BSEC_LEGACY);
```

The next thing we find in this code file is the **setup**. As already explained in the case with MQTT, in this section all sensors must be initialised using the *begin()* function. We also add the function explained in the previous section: *initProperties().* Finally, we must also include the function *ConnectCloud()*:

```
void ConnectCloud(){
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
  while(!ArduinoCloud.connected()) {
    ArduinoCloud.update();
    delay(10);
  }
}
```

In this function the connection to the Arduino Cloud is initialized using the connection manager defined in the file thingProperties.h. Once this connection is established, the connected() function is used to check whether the board is still connected and while this connection is still active through ArduinoCloud.update(). The following actions are carried out:

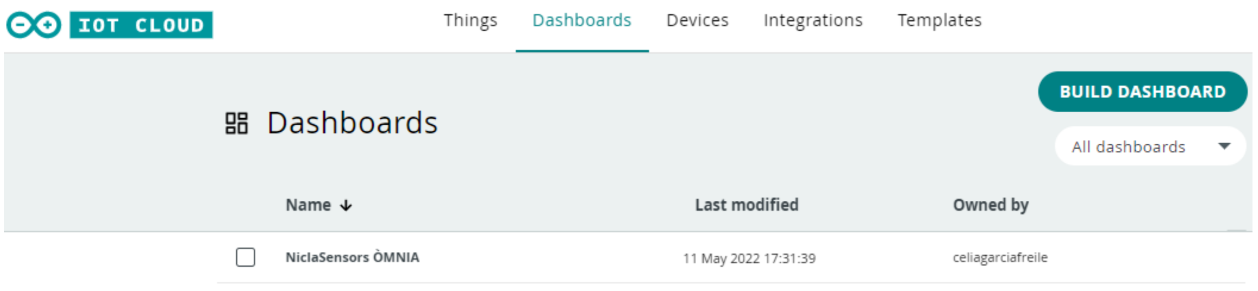
- Synchronization of property values between the Arduino and the Cloud
- Checks that the connection to the cloud is up and running
- Checks if a value has changed enough to be reported or if the time has lapsed to report a value again.

To finish with the code, it remains the loop part. In this we must include the *BHY2Host.update()* function already explained for the MQTT broker, as well as the different functions that will update the values of the different sensors and again the function ArduinoCloud.update(). This is what the loop function looks like:

```
void loop() {

    BHY2Host.update();

    temp = tempSensor.value();
    bar = barSensor.value();
    hum = humSensor.value();
    iaq_=bsec.iaq();
    co2 = bsec.co2_eq();

    ArduinoCloud.update();
    update_Serial();
    update_Display();

}
```

All the functions necessary to connect this device to a display have also been added, as they have already been explained in the previous case, we have omitted their explanation here.

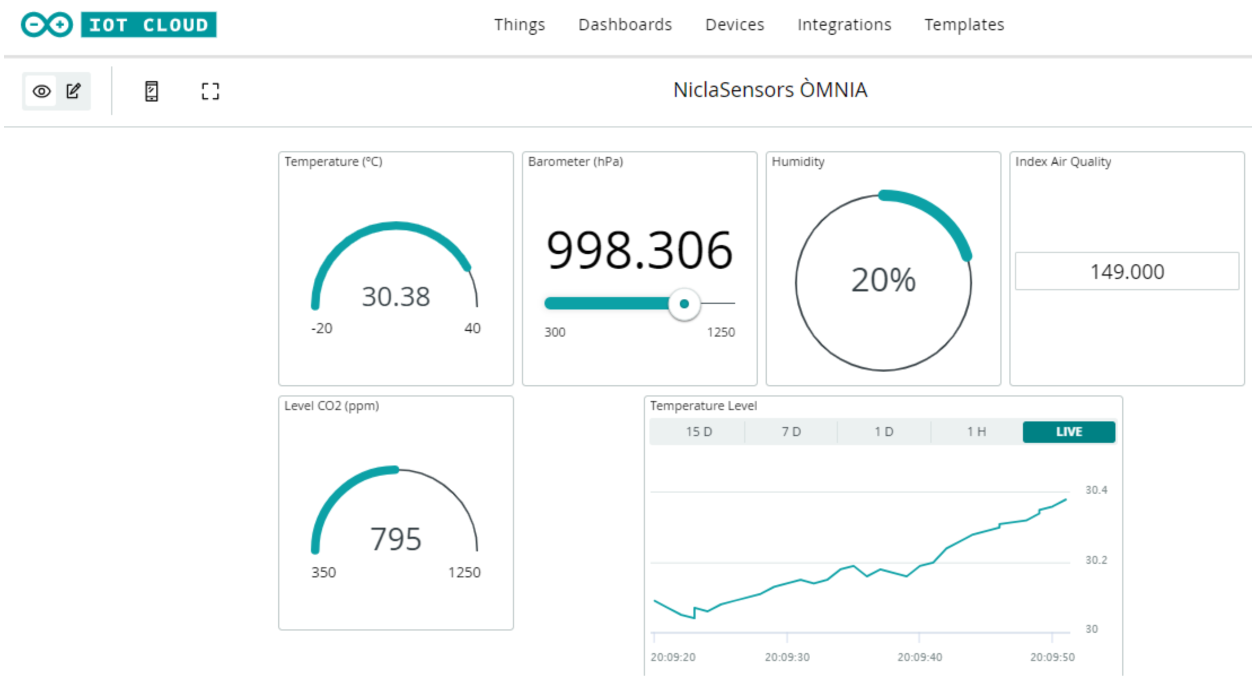
Finally, the last step is to create a dashboard in which the user can add all the widgets he wants to see the values of his variables in an attractive way.



Name	Last modified	Owned by
NiclaSensors ÒMNIA	11 May 2022 17:31:39	celiagarciafreile

In our case, we have called it NiclaSensors ÓMNIA. Then we select the widgets we want, in our case we have five values to read, so we will add five widgets: two gauges for the temperature and the level of CO2, one slider for the barometric pressure, one percentage for the humidity and a value for the IAQ. We have also added a graph where you can see how the temperature varies over the time.

This is how our final dashboard would look like in operation:



Housing design

In housing design, we want to create an ergonomic and minimalistic box just to fit perfectly with the daily technological environment of a modern house. Looking for our possibilities of marketplace we realize that we are going to develop a useful device for the home because it can be used as a thermometer and humidity measurement.

Deciding material is quite easier in this design than other sensors that we have developed because this is not going to be exposed in extreme weather conditions due to this is for indoor use. We determine that PLA is the best option, this is based on all the facilities and the balance of quality-price that we can find in this material. The black decision is just for aesthetics.

The first thing we did was to think about how the external box's shape would look like. From the beginning, we decided to design our case with rounded corners. This is because it is much more professional and aesthetically pleasing.

After that, once the shape has been determined, we go directly to the different compartments and the different options to distribute the different components that are required for our device. After a long search, we got our perfect organization for the case.

Finally, we work out the top cover design, it must include the button and the screen.

These components are the main objects to connect all the information with the customer. This must be clear and user-friendly. That is why our final solution was to create some sticks on the bottom of the cover to connect the screen to and cut one part with the "U" shape to make the pressing button action easier.

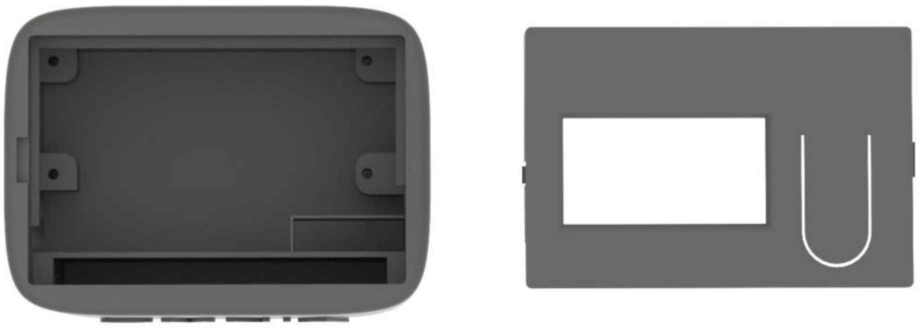


Figure 52- Views of the vibration sensor prototype (renders)

Finally, after achieving the perfect connecting mechanism between the case and the cover, we had our final design.

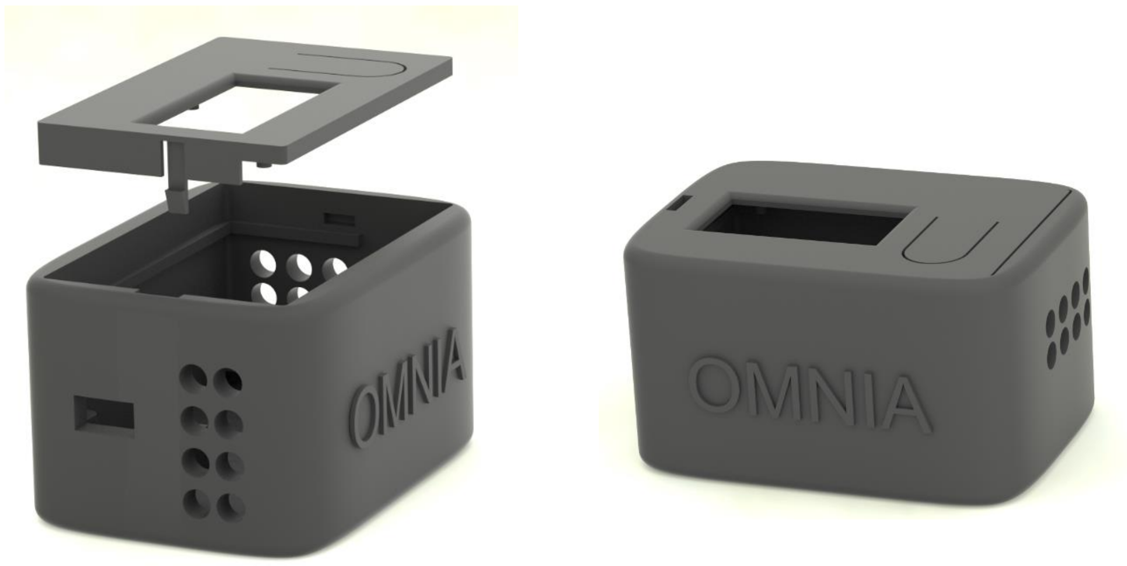


Figure 53 -Renders of the vibration sensor prototype

Supports

The goal of the frequency sensor in this situation is the measurement of indoors environmental factors. That is why we think that support for the wall is mandatory just to make the user enjoy more our sensor. We build a support so the case can hang freely in the middle of the environment instead of laying down on a table. This support is going to be fixed by screwing mechanism in the wall as a normal bookshelf. Also, another point is that it must follow the same design criteria as the product itself because it must fit perfectly with the other devices at home.

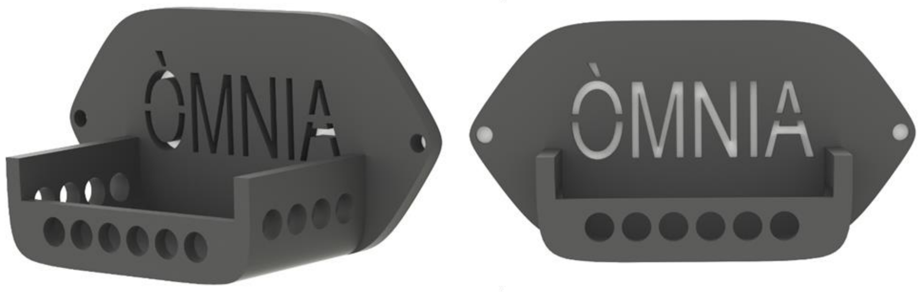


Figure 54 - Views of support for vibration sensor design (renders)

After this application we also decided to design another support that must be able to attach our sensor to another surface like a machine wall. The main problem is that we must decide which is the best material to stick our support to the machine that we want to measure because not all the machines are magnetic, so a magnet is not a viable choice. After some studies of the market, we chose adhesive stripes on the back of the support as our connecting method.

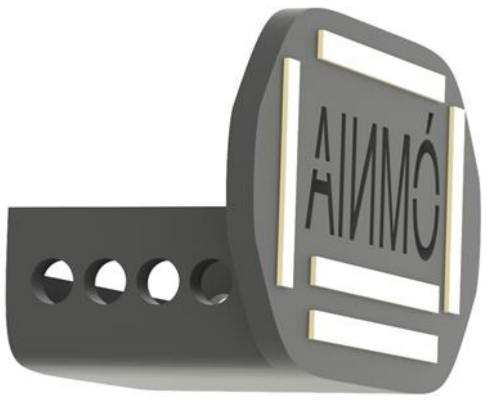
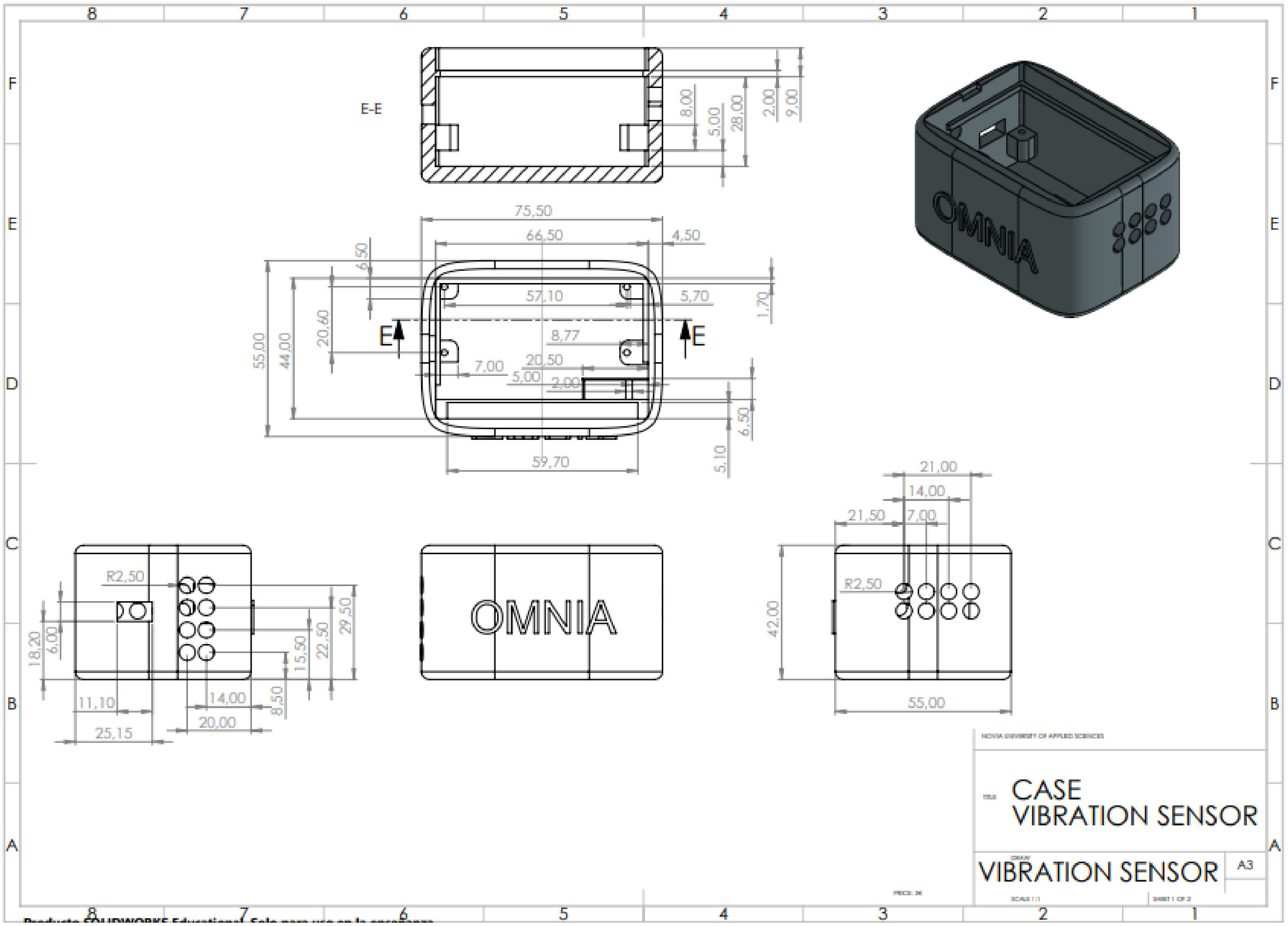
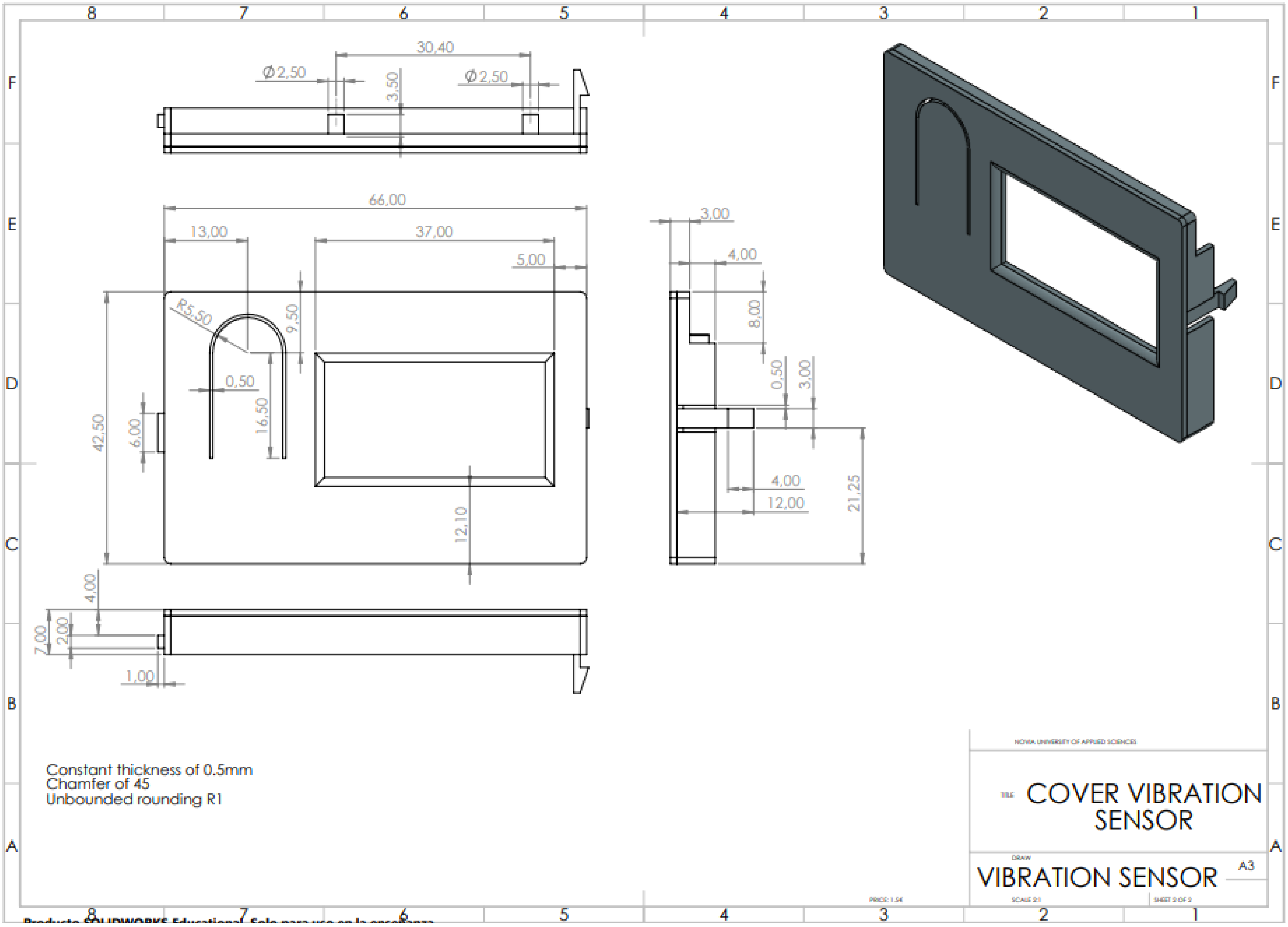


Figure 55 - Backside part of machine support


E-E
8,00
5,00
28,00
2,00
9,00
75,50
66,50
4,50
6,50
57,10
5,70
1,70
55,00
44,00
20,60
8,77
7,00
20,50
5,00
2,00
6,50
5,10
59,70
21,00
14,00
21,50
7,00
R2,50
18,20
6,00
11,10
25,15
14,00
20,00
8,50
15,50
22,50
29,50
OMNIA
42,00
55,00
NOVIA UNIVERSITY OF APPLIED SCIENCES
CASE
VIBRATION SENSOR
VIBRATION SENSOR
A3
Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.



30,40
Ø2,50
3,50
Ø2,50
66,00
13,00
37,00
5,00
R5,50
9,50
0,50
16,50
42,50
6,00
12,10
3,00
4,00
8,00
0,50
3,00
4,00
12,00
21,25
4,00
7,00
2,00
1,00
Constant thickness of 0.5mm
Chamfer of 45
Unbounded rounding R1
NOVIA UNIVERSITY OF APPLIED SCIENCES
TITLE COVER VIBRATION SENSOR
DRAW
VIBRATION SENSOR
A3
PRICE: 1.5€
SCALE 2:1
SHEET 2 OF 2
Producto SOLIDWORKS Educational. Solo para uso en la enseñanza.

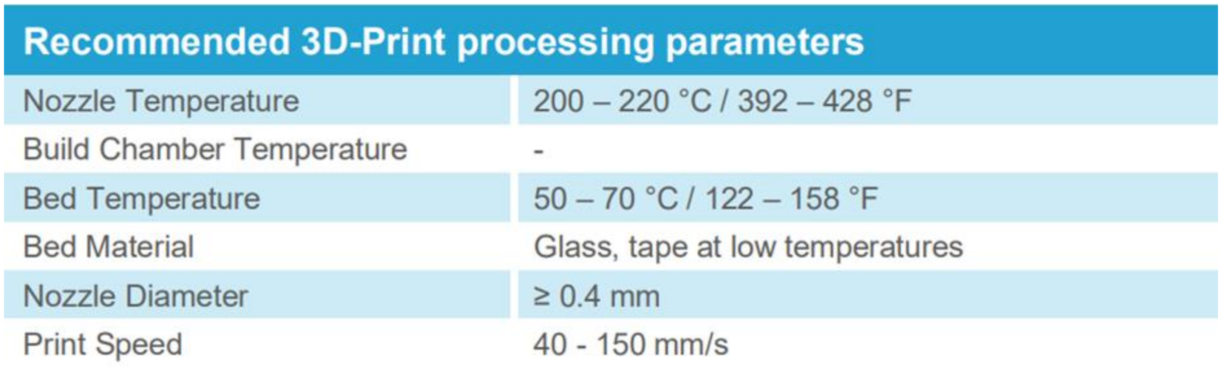
Material

The vibration sensor is not going to be exposed to extreme weather conditions, this makes the material choice easier. For our prototype we have been using Though PLA in the support and in the case, in the university we had so much Though PLA and that´s because we choose that one. Although the Though PLA is not good in some conditions, we chose it for the final product because its properties are so good and easy for printing it. The final results were very good.

About the aesthetics, this can be founded in so many different colors, but our choice is black on this component, we don´t have to worry about the overheating because of the sun because it is an indoor device, so we decided that black is the one that fits better with the company aesthetic.

In this university, the majority of roles Though PLA we used, are from BASF company. We wanted to still use this product because we had our own printing pre-set. This was done in base of the company requirements for this exact material. (ULTRAFUSE PLA PRO 1)

Table 11- Printing specifications for PRO1 PLA of BASF



Recommended 3D-Print processing parameters	
Nozzle Temperature	200 – 220 °C / 392 – 428 °F
Build Chamber Temperature	-
Bed Temperature	50 – 70 °C / 122 – 158 °F
Bed Material	Glass, tape at low temperatures
Nozzle Diameter	≥ 0.4 mm
Print Speed	40 - 150 mm/s

Finally, we should take care of the thermal properties like G*lass Transition Temperature* that must be around 63.0°C and *Melting Temperature* that must be between 170-180°C following the ISO 11357-2 and ISO 11357-3 respectively.

Battery problem

We discovered a problem with the power supply after everything was completed. When a battery is connected directly to the Arduino MKR WIFI 1010, it is unable to deliver 5V; this voltage can only be obtained by connecting the power supply via micro-USB. If only the battery is connected, 3V is produced, which is insufficient to turn on the Nicla Sense ME.

Our first and best solution was to create a wire that would send electricity from the battery to both the MKR WIFI 1010 and the SENSE ME, ensuring that both receive enough power and that we would not have any problems charging the battery because this solution would allow us to do so. The issue we faced with this approach was the battery to Arduino Nicla Sense ME connector; this device is new, and we couldn't find the connector to weld the cable for battery that would work with it. Although this connector exists, here in Finland is so difficult to find it because of the time that this device has been in the marketplace. We assumed that this would be simple to remedy in the near future because, once this connector is founded, it will simply be attached to the cable and plugged in.

On the other hand, we found a solution that fits and works perfectly with everything that we have already created. We need to power our Arduino MKR WIFI 1010 with micro-USB so we just change the power supply from PC to a power bank. If the Arduino takes this energy supply gives 5V to all the devices that are connected in. With this solution we should have solved every problem that we had but now we need to move the Power Bank with the device. We don´t think that this is a big problem because this is going to stay in the wall with the support the major part of uses, and you can always transport when is needed.

In addition, we made some changes to the wall support just to let a space for the power bank supply. The design is similar to the one is already done but with a “box” between the wall and the vibration sensor place where you can put the battery.

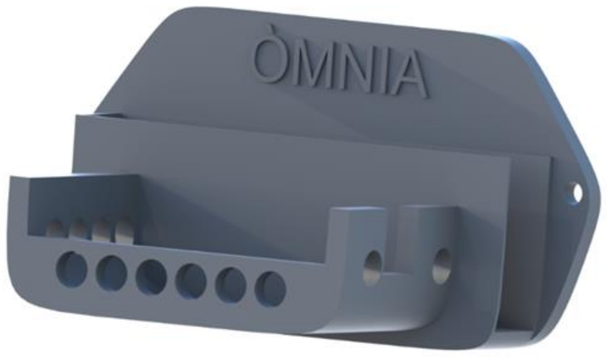


Figure 56- Support for the powerbank

User manual

In OMNIA we think that this device has to be the most user-friendly of all our sensors, this must be because of the purpose it is going to have. This device is focused on the measurement of Humidity, Temperature, barometric pressure and Quality of air. Although it can measure the vibration as it is the main objective of this device, this measurement is not that common in daily use.

This device has two different modes that we are going to talk about, the button that is on the left part of the screen is to change the mode. Just press whenever you want to get the vibration measurements.

Environmental mode:

This is the mode that is going to adopt by pre-set. From the screen we can read all the Humidity, Temperature and Quality of air measurements during all the time that the measurements are active. The measurements of this will be refreshed every 5 minutes, to make the duration of the battery more efficient.

Vibration mode:

When the customer wants to measure the vibration is needed to press the button. Once this action is done, the screen will show a “MEASURING VIBRATION” message for 10 seconds. We shall wait for the time needed and after that, we will see the results on the server. After this the vibration sensor will return to Normal mode.

7.2 CAR COUNTER

Hardware

As we have already mentioned in the research section, we have finally decided to use video detection for the car counter. The hardware we will use to carry out this will consist of a Raspberry Pi 4 Model B, the Raspberry Pi High Quality camera, its corresponding lens and the Raspberry Pi USB-C Power supply. Below we will discuss the Raspberry Pi components in depth.

Raspberry Pi 4 Model B

The Raspberry Pi is a low-cost computer the size of a credit card that connects to a computer monitor or television and utilizes a conventional keyboard and mouse. It's a capable small device that allows individuals of all ages to learn about computers and programming languages like Scratch and Python. It can do everything a desktop computer does, including accessing the internet and watching high-definition video, as well as spreadsheets, word processing, and gaming.
Aside from running Linux, it also includes a set of GPIO (general purpose input/output) pins for controlling electronic components and exploring the Internet of Things (IoT).

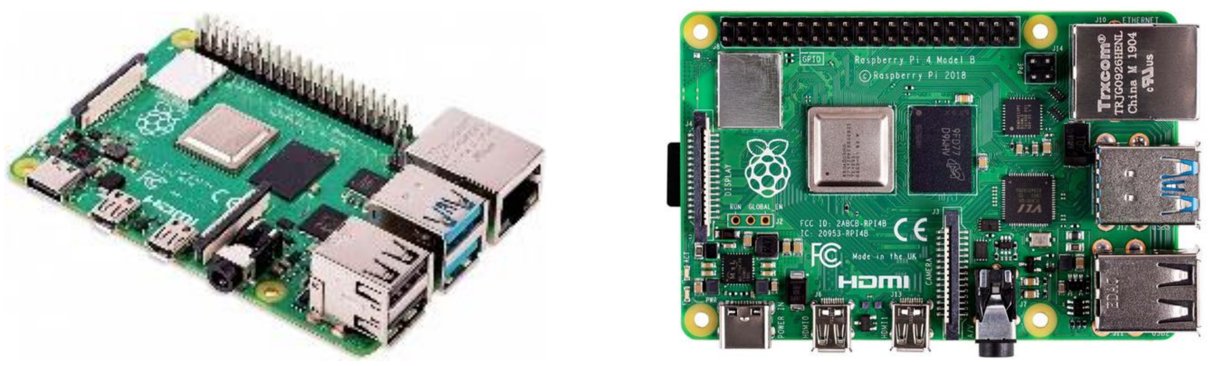


Figure 57- Raspberry Pi 4 Model B

Our Raspberry Pi model is the Raspberry Pi 4 Model B, which is the first of a new generation of Raspberry Pi computers with significantly improved CPU, GPU, and I/O performance while maintaining a same form factor, power envelope, and cost as the previous generation Raspberry Pi 3B+.

In the following graphic we show an image of the tech specs of this powerful board:

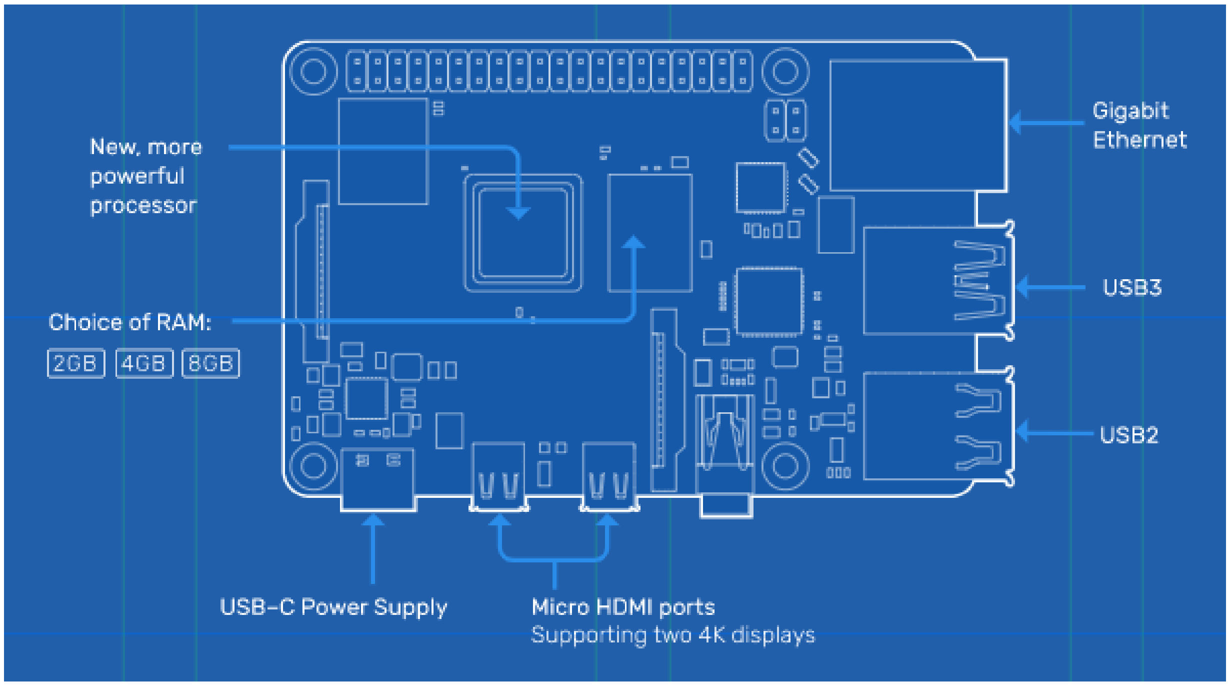


Figure 58-Tech specs of a Raspberry Pi 4 Model B

The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera peripherals, including our High-Quality Camera.

Despite its magnificent features and possibilities, it must be taken into account that its processor is still less powerful than that of a desktop computer, and this must be taken into account when programming.

Raspberry Pi High Quality Camera

This High-Quality Camera is the latest camera accessory that Raspberry Pi developed. It offers Higher resolution (12 megapixels) and sensitivity (±50% greater area per pixel for improved low-light performance) than the existing Camera module v2.

The camera is designed to work with CS-mount lenses as well as C-mount lenses with the included adaptor. Third-party goods compatible with the High Quality Camera include the CGL 6 mm CS-mount and 16 mm C-mount lenses.

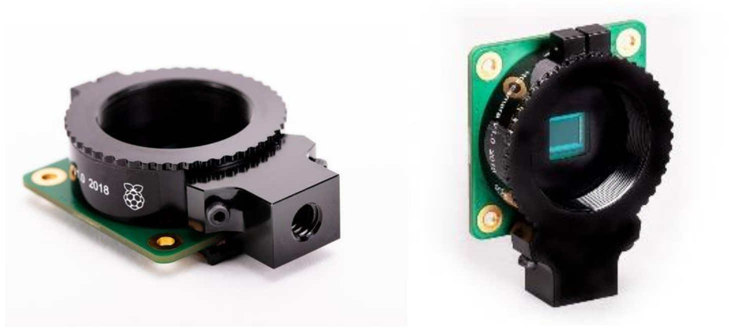
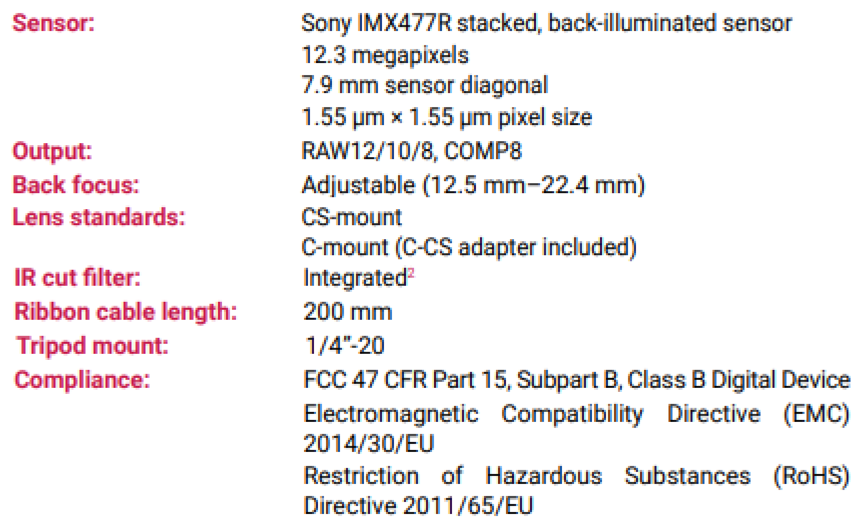


Figure 59- Raspberry Pi high quality camera

An overview of the specifications is shown in the following image:



Sensor:	Sony IMX477R stacked, back-illuminated sensor 12.3 megapixels 7.9 mm sensor diagonal 1.55 μm × 1.55 μm pixel size
Output:	RAW12/10/8, COMP8
Back focus:	Adjustable (12.5 mm–22.4 mm)
Lens standards:	CS-mount C-mount (C-CS adapter included)
IR cut filter:	Integrated[2]
Ribbon cable length:	200 mm
Tripod mount:	1/4"-20
Compliance:	FCC 47 CFR Part 15, Subpart B, Class B Digital Device Electromagnetic Compatibility Directive (EMC) 2014/30/EU Restriction of Hazardous Substances (RoHS) Directive 2011/65/EU

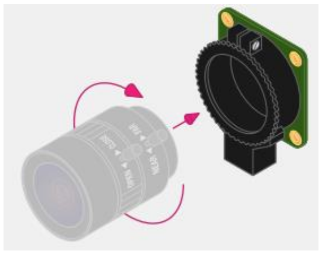
Figure 60- components of raspberry high-quality camera

Wide Angle 6mm CS-mount lens

The raspberry camera needs a lens to work. In our case we will use the Wide range CS-mount lens of 6mm. As it will allow us to see the road at a good angle and thus help us to detect vehicles in a better way.

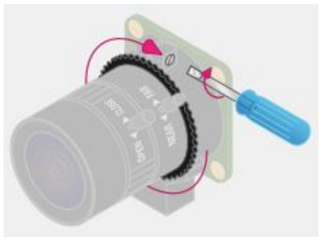
The steps needed to mount it on our Raspberry Pi camera are the following ones:

- **Fitting the lens to the camera**
  Because the lens is CS-mount, it has a short back focus and does not require the C-CS adapter that comes with the High Quality Camera. If the adapter ring is used, the lens will not focus properly.
  The next step is to move the lens clockwise until it reaches the back focus adjustment ring**.**



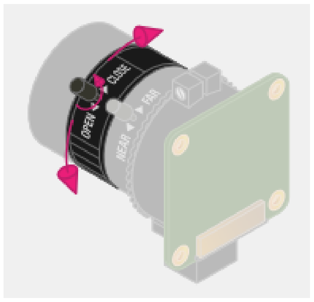
- **Back focus adjustment ring and lock screw**

  For the shortest possible rear-focal length, the back focus adjustment ring should be fully screwed in.



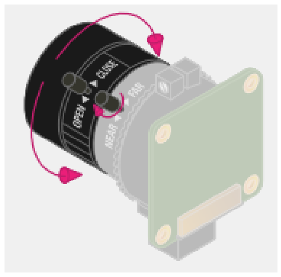
- **Aperture:**

  The aperture of a camera indicates the brightness of the lens. That is, the ability of more or less light to reach the sensor through the diaphragm. A larger aperture will let more light through than a smaller aperture. To adjust the aperture in this lens it is necessary to turn the middle ring while keeping the outer ring, which is the furthest away from the camera, steady. Then, close the aperture and reduce image brightness by turning the knob clockwise or open the aperture, turn the screw counterclockwise. Once satisfied with the light level, lock the aperture, tighten the screw on the side of the lens. This process will be carried out in the place where we want to place our car counter, in this way we ensure that the light level is adjusted to the future location of the camera.



- **Focus**

This is a very important step in our device, because without a clear image it is impossible to do video detection correctly. To adjust the focus, it is necessary to turn the two outer rings clockwise to focus on a close object and anti-clockwise to focus on a distant object.



To do this process, we will take several pictures of the place where we will place the camera and at the same time, we focus it to ensure the best possible focus.

Mounter

For the camera supporting, we should use a tripod that has to fits perfectly with the Raspberry Camera. The characteristics that we were looking for our tripod were the following:

- Water resistance: We need that all the parts that can possibly be oxide for be full of stainless steel.
- Small dimension: Our camera support has to be small because it has to be in the public zone and we cannot attract attention.
- Possibility of screwing: This device has to be placed in high that´s why we need to attach the support to a surface like streetlight or some building wall.

The Raspberry camera use standard screw so we can use a camera wall mount for our device.

We choose the CAMVATE video wall ceiling mount that counts with 1/4 – Inch 20 Thread and is attached to the surface with 4 screws, it´s going to be more than enough.

In addition, we optimize the relation quality price because this piece is founded in Amazon rounding 15e.

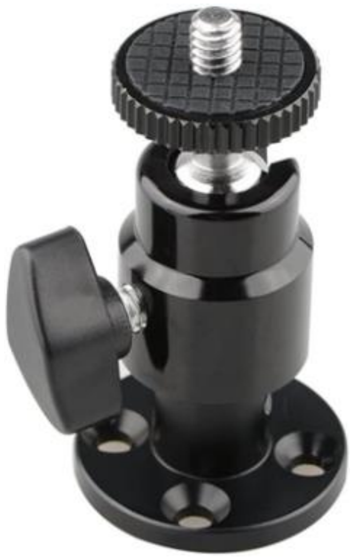


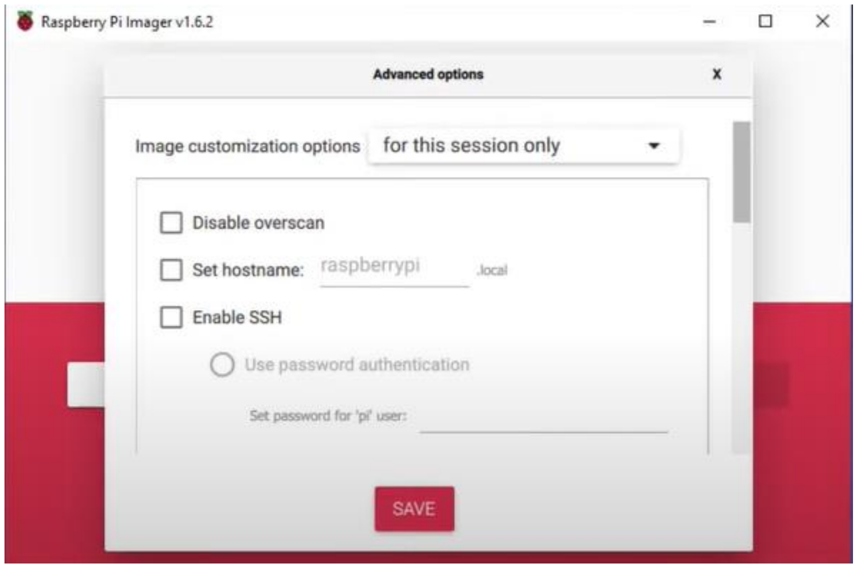
Figure 61- Support CAMVATE

Coding

After explaining the hardware that we will use in our device, the next step is to explain the code that will run our Raspberry.

The first step is to install the operating system. The official operating system of the Raspberry Pi Foundation is known as Raspbian or raspberry Pi OS, which is based on Linux and optimized for these microcomputers. The first thing we need is a Micro SD card, in our case, our SD car has a capacity of 32GB. Then, for installing Raspbian we have used Raspberry Pi Imager, because an easy way to install the operating system as it also formats the card at the same time as it installs it. We have installed the Raspberry Pi OS (32-bit), recommended by the Raspberry Pi Association.

On a side note, we have used a special command Ctrl+Shift+X, which allows us to open the advanced settings in the operating system installation process.


Raspberry Pi Imager v1.6.2

Advanced options

Image customization options: for this session only

Disable overscan

Set hostname: raspberrypi .local

Enable SSH

Use password authentication

Set password for 'pi' user:

SAVE

Figure 62- Advance option when installing Raspbian

One of the interesting options we find here is that it allows us to change the name of the raspberry, since by default all raspberries have the name raspberrypi.local. In our case the raspberry will be named omnia.local. But the most important thing is that it allows us to enable SSH in a very simple way, which although we will explain it in more detail later will allow us to manage our raspberry without a keyboard, mouse, or screen. Here we will also configure the credentials of our Wi-Fi network and the country we are in. After everything is done, we are ready to install this operating system on the SD card.

Protocol SSH

As this device is intended to be placed in a housing on the street, it forces us to be able to operate it without being able to place a keyboard or a monitor every time we have to make changes to the server settings.

SSH stands for Secure Shell or Secure Socket Shell and is nothing more than a network protocol that provides users, especially system administrators, with a secure way to access a computer such as the Raspberry over an open network. This protocol provides a mechanism that authenticates the remote user and transfers the input from the client to the host, relaying the output to the client.

The ssh protocol uses the client-server model, connecting a Secure Shell client application, which is where the session is displayed, to an SSH server, which is where the session is executed. Secure Shell provides secure password authentication and public key authentication, and encryption, to perform data communications between two connected computers.

This protocol is composed of three layers:

- **Transport layer:** responsible for establishing safe and secure communication between client and server, during and after authentication. This layer accelerates data exchange, compressing and caching data, maintaining a secure site. It also oversees encryption, decryption and data integrity protection.
- **Authentication layer:** responsible for communicating the supported authentication methods to the client. It also performs the authentication process of a user.
- **Connection layer:** responsible for managing communication between machines after successful authentication.

SSH is widely used for raspberry projects due to the facilities it provides. In the next point we will see how we are going to be able to work with SSH on our raspberry.

Visual Code Studio

Visual Studio Code, also known as VS Code, is a source-code editor for Windows, Linux, and macOS developed by Microsoft. Debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git are among the features. Users can customize the theme, keyboard shortcuts, and preferences, as well as install extensions that offer new features.

This tool is perfect for us as it has a Remote - SSH extension, that enables to open a remote folder on any remote system, virtual machine, or container (in our case it will be the raspberry) with a functioning SSH server and use all the features of VS Code. In this way it is possible to interact with files and folders on the remote filesystem once you've connected to a server.

Because the extension runs commands and other extensions directly on the remote system, no source code on your local workstation is required.

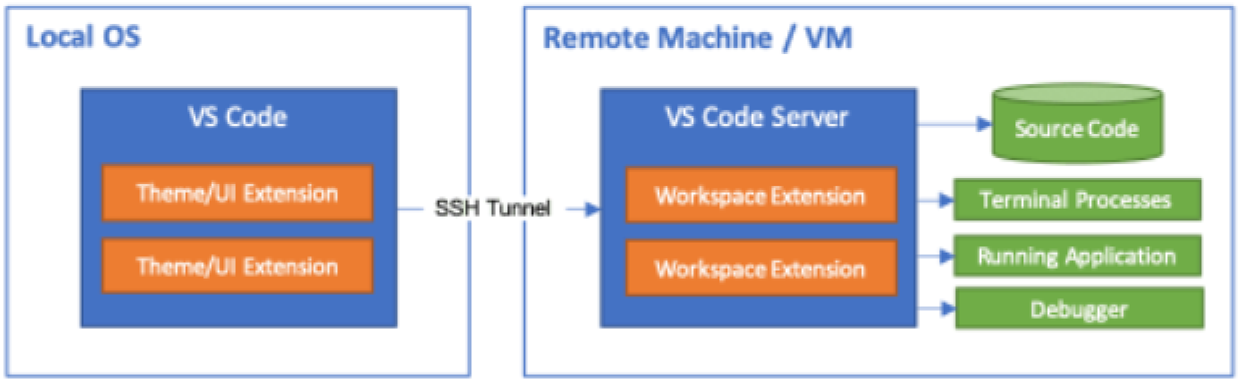


Figure 63-Representation of how it works the VS code

To connect with our raspberry, we just need to install the SSH extension, and then establish the connection with the Host. The first time we do this, we must indicate which host we want to connect to, i.e. which raspberry. To do this we will need to know the name and IP of the raspberry. Once this is done one time, it is no longer necessary to enter this data again on future occasions.

After verifying that we have been able to connect to our remote host, we are now working almost in the same way as in a local session and are ready to start programming the Raspberry.

It should be noted that to do all this process it is necessary that the computer in which we open this remote session is connected to the same Wi-Fi network as the Raspberry. If this is not possible, there are some applications such as ZeroTier that allow us to work from a different wifi network.

Object Detection

The next point is to explain what the object decion we are going to use to count cars consists of.

Computer vision is a field that falls within the broader spectrum of Artificial Intelligence studies. Computer vision consists of working with digital images and videos to deduce some understanding of the contents within these images and videos.

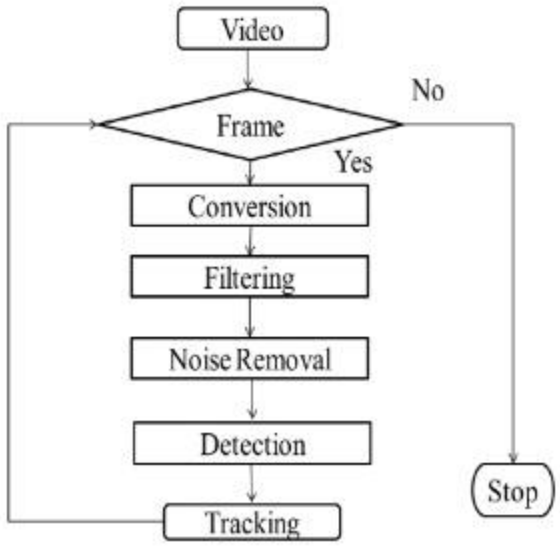
Object detection is a Computer Vision term that refers to a system that can detect the presence and placement of a desired object or body within an image, it may appear once or several times.

The best-known library that allows us to carry out this object detection is known as **Open CV** (Open Source Computer Vision Library). It is a free software library for computer vision and machine learning. OpenCV was created to provide a common infrastructure for computer vision applications and to let commercial goods incorporate machine perception faster.

The library contains over 2500 optimized algorithms, including a wide range of traditional and cutting-edge computer vision and machine learning techniques. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, remove red eyes from images taken with flash, follow eye movements, recognize scenery, and so on.

The question now is how we are going to count objects, in our case cars, from it. Well, this process involves several phases that will allow us to isolate the objects within an image.

The process is divided into 5 phases as shown in the following chart



1) **Convert the image to greyscale:**

This is necessary because when working with colour images, the computational cost grows exponentially.

For example, if you have three colour components as in RGB space (R for Red, G for Green and B for Blue) it is as if you were working with three different images, one for each component.

2) **Filter the image to remove noise**

Digital images are not without flaws. The inherent noise, as well as detrimental lighting effects, distort reality. This is not just true in terms of visual settings. Any digital or analogue signal is susceptible to many types of noise.

There are various approaches for removing noise in digital image processing, According to the official website of open cv we can find the following:

- Averaging
- Gasussian Blurring
- Median Blurring
- Bilateral Filtering

The common mathematical operation uses by all these methods is the convolution, which involves using a N × N mask or kernel to run through a picture pixel by pixel. The following image shows how this mask would work.

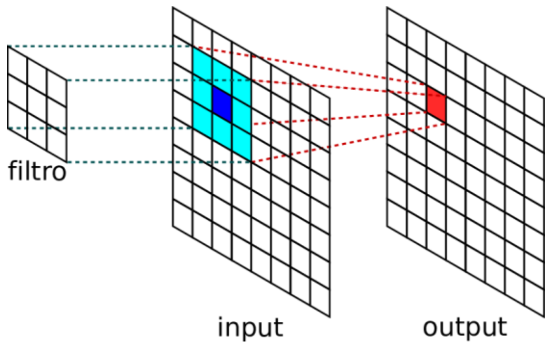


Figure 64- Ilustrative representation of a filter mask

3) **Apply Canny edge detector**

The process for edge detection with Canny is divided into 3 steps:

- **Edge detection with Sobel**
- **Suppressing pixels outside the edge**
- **Apply thresholding by hysteresis**

One of the major drawbacks of computer vision algorithms is parameterization. In many cases, these parameters are unique to each situation because they have a certain illumination or perspective. This fact makes it impossible to find a general method that works correctly for all situations. In the following we will see what each of these steps consists of.

- **Edge detection with Sobel**

The Sobel edge detector is based on the calculation of the first derivative. This mathematical operation measures the evolutions and changes of a variable. It basically focuses on detecting changes in intensity. When we talk about edges in an image, we are talking about pixels where there is a change in intensity. The aim is to be able to detect this edge through the first derivative.

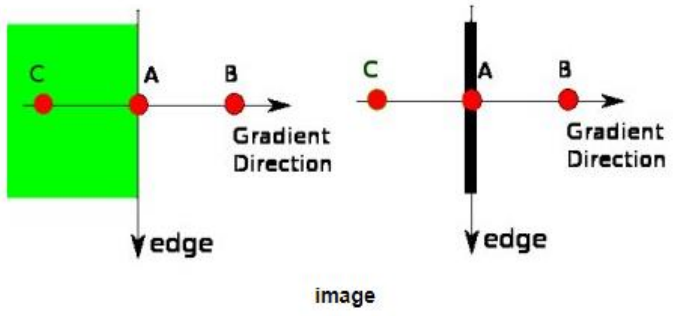
The smoothed picture is then filtered with a Sobel kernel in both the horizontal and vertical directions to obtain the first derivative in both directions (Gx) (Gy). We can find the edge gradient and direction for each pixel using:

$$Edge\_Gradient\;(G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\;(\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

- **Suppressing pixels outside the edge**

After obtaining the gradient magnitude and direction, the image is fully scanned to remove any undesired pixels that do not form the edge. For this, each pixel is tested to see if it is a local maximum in its neighborhood in the gradient direction. Look at the illustration below, taken directly from the opencv website:

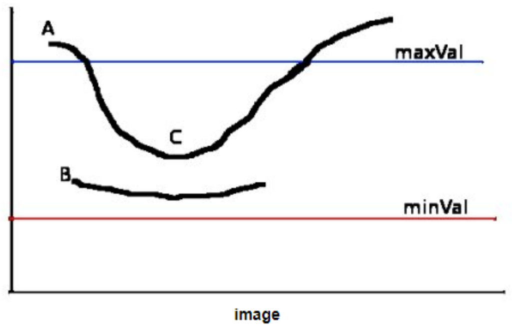


As they explain: “Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So, point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero). In short, the result you get is a binary image with "thin edges"”.

- **Apply thresholding by hysteresis**

This stage determines which edges are really edges and which are not. In fact, what it is done is determine if a pixel is a part of the background or a part of the object.

In other to do this, we'll need two threshold values, minVal and maxVal, for this. Any edges with an intensity gradient more than maxVal are certain to be edges, whereas those with an intensity gradient less than minVal are certain to be non-edges and should be rejected. Based on their connectedness, those who fall between these two thresholds are classed as edges or non-edges.



4) **Find contours within the detected edges**

The next thing is to distinguish between edge and contour. Edges, as we have seen above, are sharp changes in intensity. However, a contour is a curve of points without gaps or jumps, i.e. it has a beginning and the end of the curve ends at that beginning. If they are not contours, they are simply discarded.

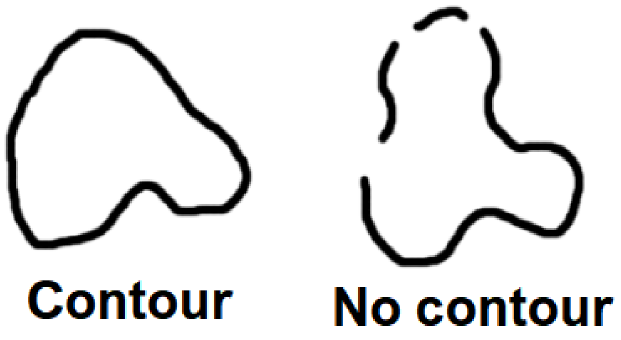


Figure 65- Differences between a countour and edge

5) **Draw these contours**

After all these steps, the last step would be to draw the contours that we have found in the image.

YOLOv4 Algorithm

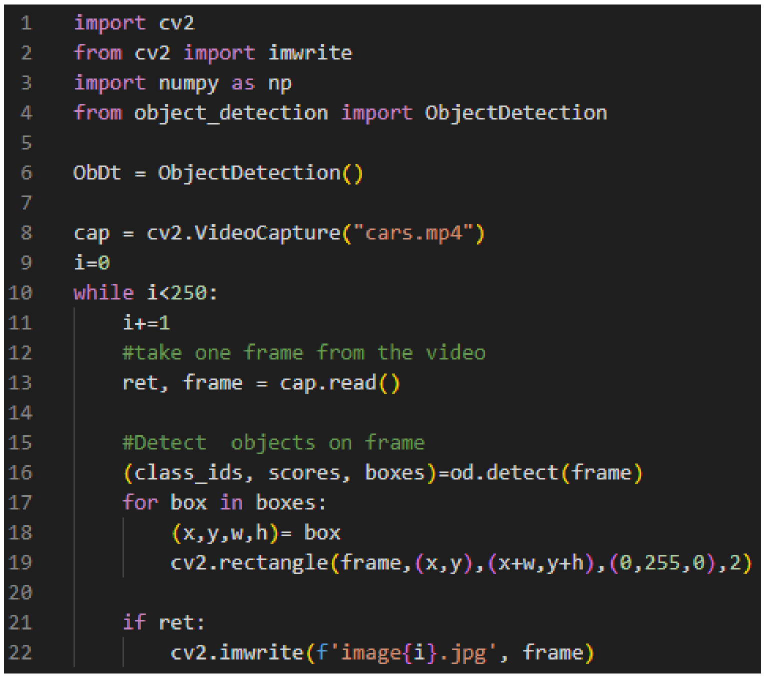
Finally, we will explain the code we have used to make our video detection. Since none of our team members had any experience with object detection using computer vision, we saw that the easiest solution was to use a ready-made algorithm, as we had neither the time or the knowledge to start from scratch.

Evidently, this is not an optimal solution, because it requires a high processing capacity and does not allow us to create a real-time car counter, but only allows us to do it from pre-recorded video.

A continuation of this project would be a very good idea, in which surely someone with more experience and background than us would be able to create a self-mad algorithm that could work in real time.

The already made algorithm that we will use is the yolov4 (You Only Look Once). It is a state-of-the-art open-source system for object detection, which makes use of a single convolutional neural network to detect objects in images. To operate, the neural network divides the image into regions, predicting identification boxes and probabilities for each region; the boxes are weighted based on the predicted probabilities. The result of applying this will be an image with the detected objects in a box. Some of the codes are able also to identify what there are detecting, and they include it in the image itself.

We will now explain the code that will run the raspberry:



```python
import cv2
from cv2 import imwrite
import numpy as np
from object_detection import ObjectDetection

ObDt = ObjectDetection()

cap = cv2.VideoCapture("cars.mp4")
i=0
while i<250:
    i+=1
    #take one frame from the video
    ret, frame = cap.read()

    #Detect  objects on frame
    (class_ids, scores, boxes)=od.detect(frame)
    for box in boxes:
        (x,y,w,h)= box
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

    if ret:
        cv2.imwrite(f'image{i}.jpg', frame)
```

It is quite simple; it starts by importing the libraries needed to run the code such as Open cv or Numpy. It also includes the object detection file, which is an already made python file in charge of running Open cv DNN with the algorithm YOLOv4.

Next the object detection is initialized using ObjectDetection(). Then we indicate from which video we want to obtain the frames, which in our case is a pre-recorded video with the Raspberry camera called cars.mp4.

Then, we establish a while loop that will end when the variable i exceeds the value of 250. In other words, a total of 250 frames will be studied, since this is the total number of frames that our recorded video of approximately 8 seconds has. After this, we will simply study the objects in each frame of the video and mark them with a rectangle. In the following images we can see how this code effectively detects all the cars passing along the road.

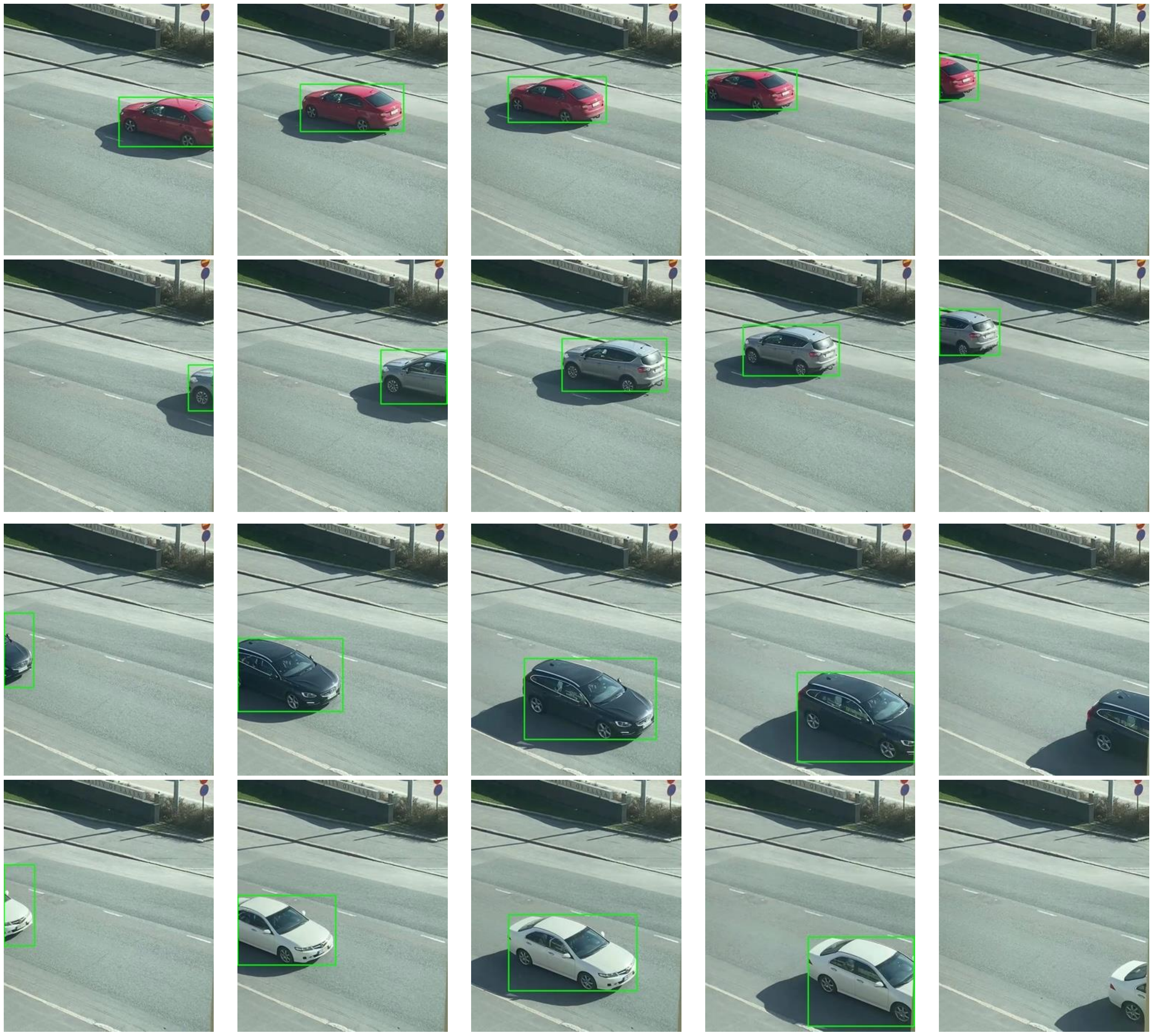
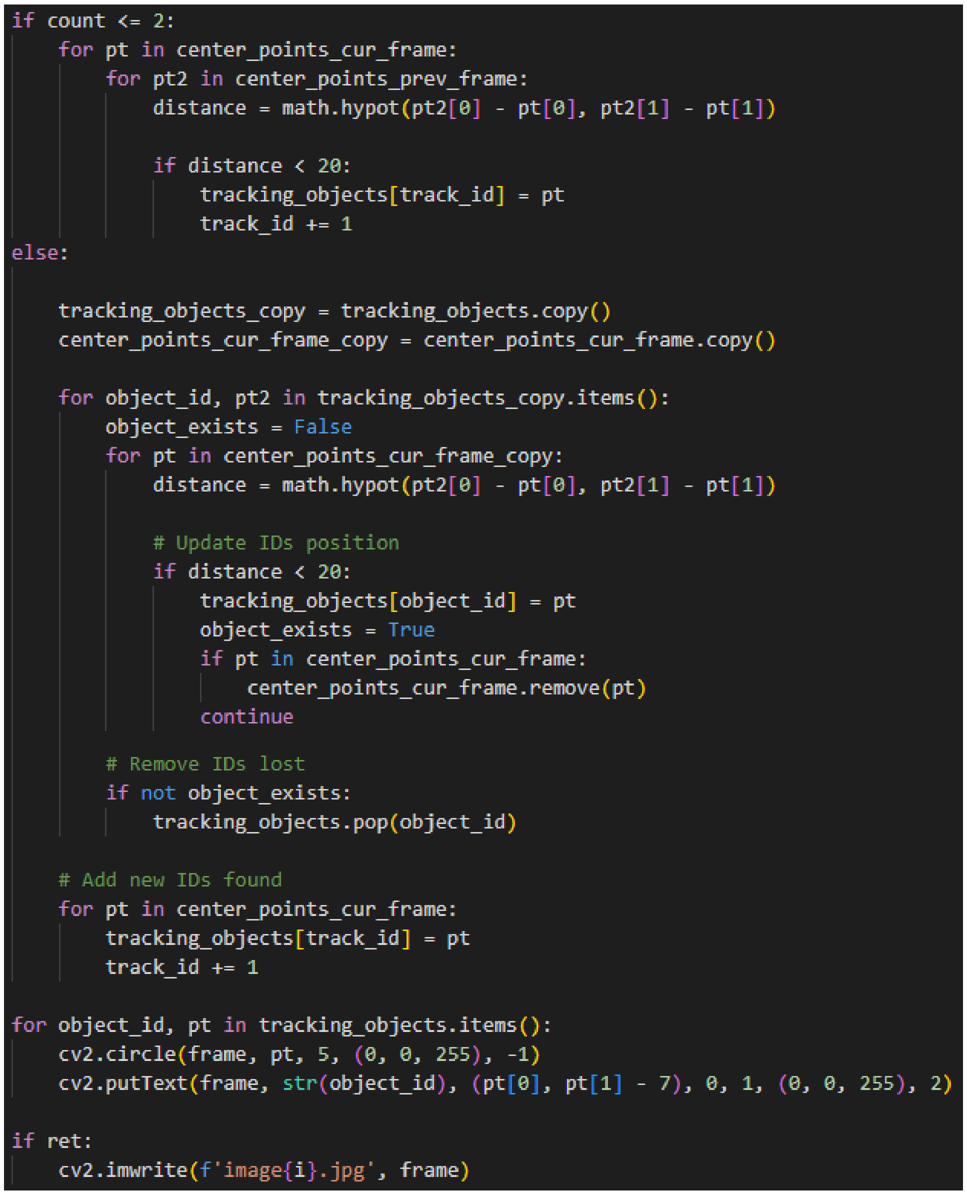


Figure 66- Car detection

The next step will be to start counting these cars. In other to do this, what this method carries out is calculate the center point of each the bounding box and store it. In the next frame it does exactly the same and then compare the new positions of the circles with the previous positions. When two circles are really closed one from the other, we can suppose that it is the same vehicle, so we do not count it as another one.



```python
if count <= 2:
    for pt in center_points_cur_frame:
        for pt2 in center_points_prev_frame:
            distance = math.hypot(pt2[0] - pt[0], pt2[1] - pt[1])

            if distance < 20:
                tracking_objects[track_id] = pt
                track_id += 1
else:

    tracking_objects_copy = tracking_objects.copy()
    center_points_cur_frame_copy = center_points_cur_frame.copy()

    for object_id, pt2 in tracking_objects_copy.items():
        object_exists = False
        for pt in center_points_cur_frame_copy:
            distance = math.hypot(pt2[0] - pt[0], pt2[1] - pt[1])

            # Update IDs position
            if distance < 20:
                tracking_objects[object_id] = pt
                object_exists = True
                if pt in center_points_cur_frame:
                    center_points_cur_frame.remove(pt)
                continue

        # Remove IDs lost
        if not object_exists:
            tracking_objects.pop(object_id)

    # Add new IDs found
    for pt in center_points_cur_frame:
        tracking_objects[track_id] = pt
        track_id += 1

for object_id, pt in tracking_objects.items():
    cv2.circle(frame, pt, 5, (0, 0, 255), -1)
    cv2.putText(frame, str(object_id), (pt[0], pt[1] - 7), 0, 1, (0, 0, 255), 2)

if ret:
    cv2.imwrite(f'image{i}.jpg', frame)
```

After testing this, we concluded that although the idea is good there are many drawbacks. It only worked for the first two vehicles, however from the third it was not able to distinguish it as a single car, so it lost the count. This may be due to several reasons, for example that the speed was higher than the rest, but it can be due also to the angle from which it was recorded since unlike the other two previous cars it was circulating in the lane closest to the camera. In fact, excessive angles generate perspective distortions, therefore, the best possible position for traffic counting by video analysis is one that takes images from overhead on the road surface.

Despite not having achieved a fully functional code in real time, this project can be taken as a base point to develop a car counter that works to perfection, at the same time it has been useful as a first contact for the group with raspberry, python and the world of machine vision and object detection.

Housing design

For the housing design, we wanted to create a minimalist, waterproof and shock-resistant case. When we were brainstorming the possibilities, we came up with the idea of a case that consists of two parts, an inner shell and an outer shell.

The inner shell is mainly meant to keep all the components in the designated location. It also provides basic protection for the components.

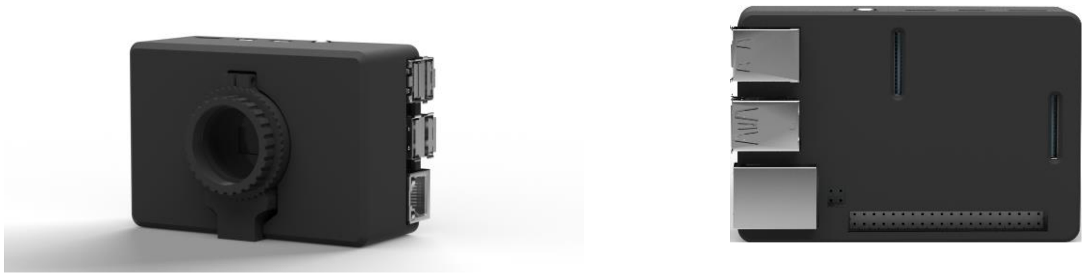


Figure 67- Create the camera from a Raspberry

The outer shell is more for protection against all kinds of weather conditions in Vaasa. We designed this housing in NX. The outer shell needed to be as water resistant as possible, but also remain minimalistic.

We have designed a case that slides together in a vertical way. The outer layer of this case overlaps the inner layer. This gives us a good seal against both water and unwanted dirt. We have also created a protection for the lens of the camera against these hazardous weather conditions.

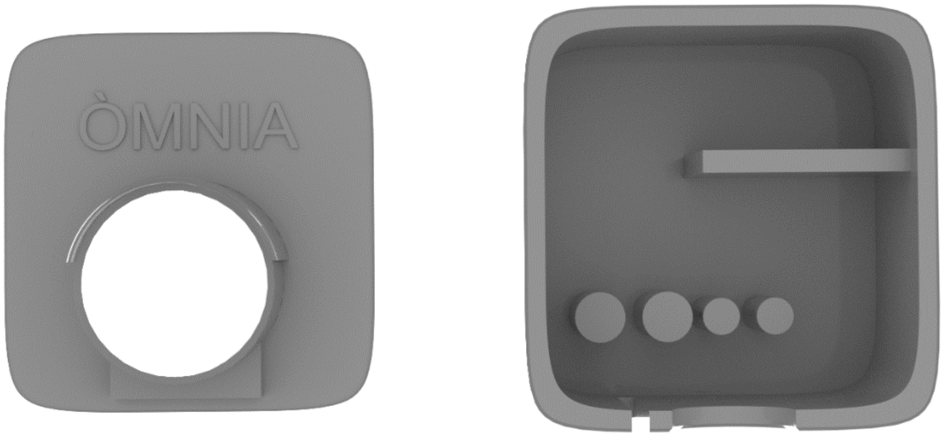


Figure 68- Views for the Raspberry camera (Renders)

Furthermore, we also wanted to be sure that if some water comes in the outer case, the case has some holes where the water can exit the case and not create humidity or get to our Raspberry. These holes also help to avoid condensation of the protective glass and the camera lens. As condensation makes visibility almost impossible, these holes were of great importance.

Determining the right material was very important for the sensor case. This is because the sensor will be constantly outside in all kinds of weather that can harm the sensor and the other components within the case.

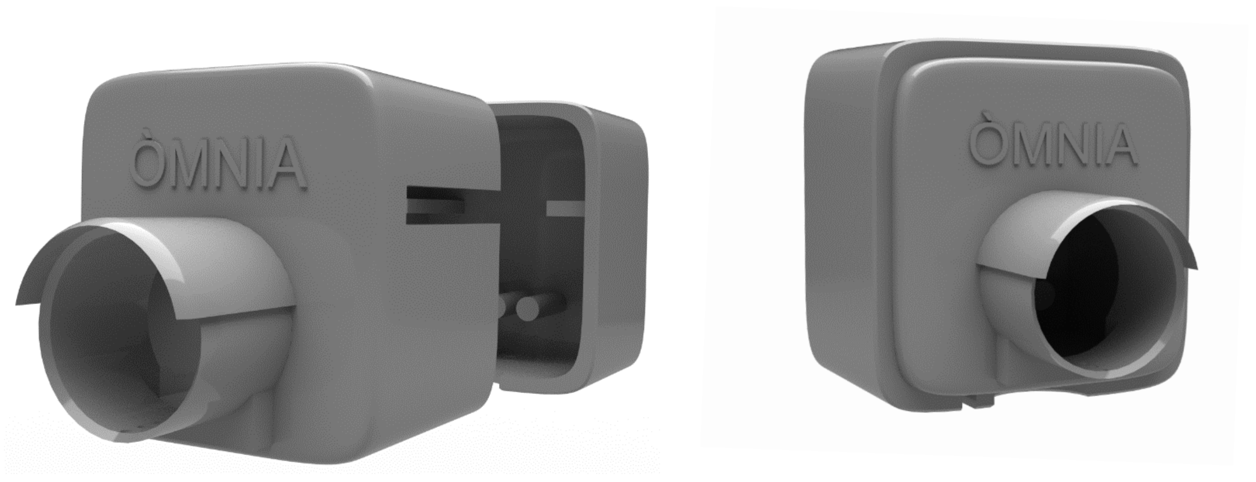
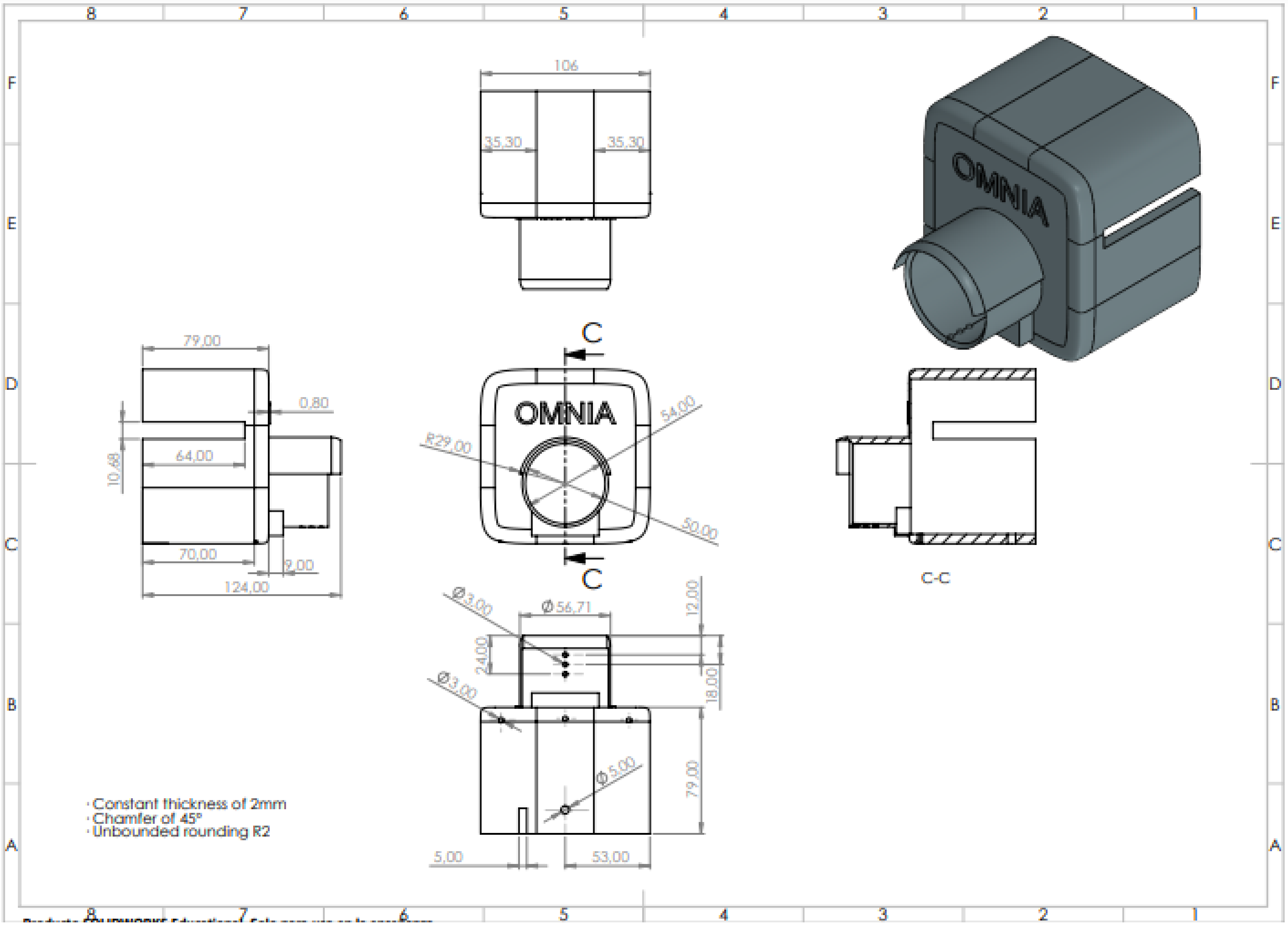


Figure 69- Case for Raspberry Camera


106
35,30
35,30
79,00
0,80
64,00
10,68
70,00
9,00
124,00
C
C
OMNIA
R29,00
54,00
50,00
C-C
Ø3,00
Ø56,71
12,00
24,00
18,00
Ø3,00
Ø5,00
79,00
5,00
53,00
· Constant thickness of 2mm
· Chamfer of 45º
· Unbounded rounding R2

Materials

This material selection is based on the idea that it needs to anticipate all the extreme weather conditions that this device can suffer during this measurement. This device is going to take all the information about the cars in Finland, that is why we have to be accurate at the material selection because of the conditions of this country with lower temperatures and meters of snow in winter and almost 24h of sun in the summer.

All of our prototypes are built of PLA for the reasons stated above, but after a thorough investigation into which material is appropriate for this purpose, we discovered that ASA is the best of our options.

ASA is an acronym of *Acrylonitrile Styrene Acrylate*. This material is an amorphous thermoplastic developed as an alternative of ABS because this has low impact resistance, toughness or weather resistance… Looking for our necessities we find that the only pro that ABS has against ASA is the price.

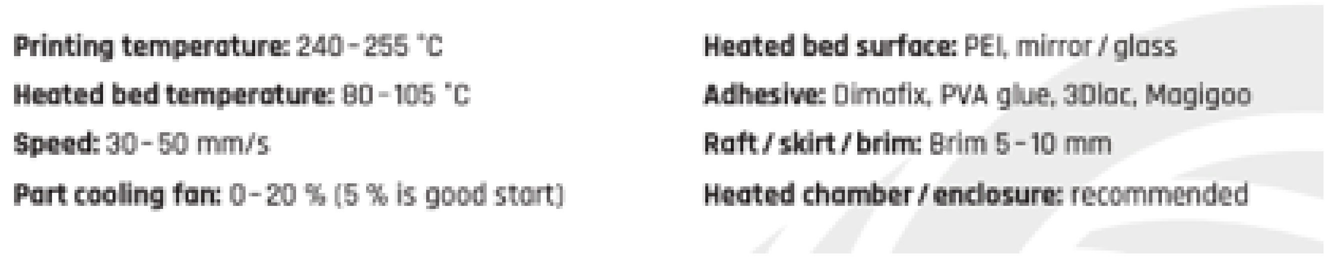
This material is currently one of the most well-known in the 3D printing industry due to its excellent characteristics and malleability. This element's excellent outside qualities make it the greatest choice for practically all enterprises that require outdoor machines or devices, with the automotive industry being the largest consumer.

The properties that make ASA interesting are following:

- Excellent chemical resistance
- Resistance to extreme weather conditions
- UV Resistance
- Resistance to impact

In our case, for the 3d printers we have been using the ASA material from 'Fillamentum'. This is the brand that "Novia" uses for its ASA material. We set up the settings for the 3d printer following the company specifications:

Table 12- Printing specifications for ASA from Fillamentum



Printing temperature:	240 - 255 °C	**Heated bed surface:**	PEI, mirror / glass
Heated bed temperature:	80 - 105 °C	**Adhesive:**	Dimafix, PVA glue, 3Dlac, Magigoo
Speed:	30 - 50 mm/s	**Raft / skirt / brim:**	Brim 5 - 10 mm
Part cooling fan:	0 - 20 % (5 % is good start)	**Heated chamber / enclosure:**	recommended

User manual

The goal we are trying to achieve is to get a complete overview of the amount of car traffic entering and leaving Vaasa. Good location is needed for get a good quality image to be sure that all the cars that the device is counting are real and not any fail of the system (like passerby or sun rays going directly to the lens).

The function of the car counter is to detect these cars and determine whether they enter or leave Vaasa. We do this by using a Raspberry Pi 4 and a Raspberry High-Quality Camera. Using a program on the Raspberry Pi 4 computer in combination with the Raspberry High-Quality Camera, we can detect the cars and determine whether they enter or leave Vaasa.

Knowing that this measurement is something specific for a determinate group of people (for example students or statisticians) the use of this is not user-friendly.

This code can take a video and divide into a picture just to make it easy to determine the number of cars that are going one way or another. It doesn't work in real time because it is a thing that we must fix but can detect the traffic in a proper way.

For using this the customer may have connected his computer to the Raspberry (this should be programmed with the Wifi) and you could have the different videos and a green square for each vehicle that is passing on that images.

Otherwise, there is another option to plug in some peripherals directly to the Raspberry and get all the information. This alternative is not viable because of the location and casing of the device but is always good to have other options to find the same results.

7.3 SNOW DEPTH SENSOR

Hardware

TF-Luna Lidar sensor

For the development of the snow depth sensor, we used some different hardware components. First, we have the TF-Luna lidar sensor. As we already explained, this device measures the time that a light beam takes to go from the sensor to the surface of the snow and back to the sensor.

Operating range	0.2~8 m
Accuracy	±6 cm @ 0.2-3 m
	±2 % @ 3-8 m
Range resolution	1 cm
FOV	2°
Frame rate	1~250 Hz
Communication	UART / I2C

Figure 70- TF-Luna Lidar sensor

Arduino MKR NB 1500

Then, we used an Arduino MKR NB 1500 board. This board has several input interfaces: an UART port, an I2C port and a SPI port. These features provide us some tools to establish a communication between the sensor and the board. This board has also 8 digital I/O pins, 7 analog inputs and 1 analog output, as well as an antenna and a LI-PO battery inputs.

As this board is designed to bring connectivity, and especially IoT connectivity, we can identify that some of its main features are related with the IoT technology:

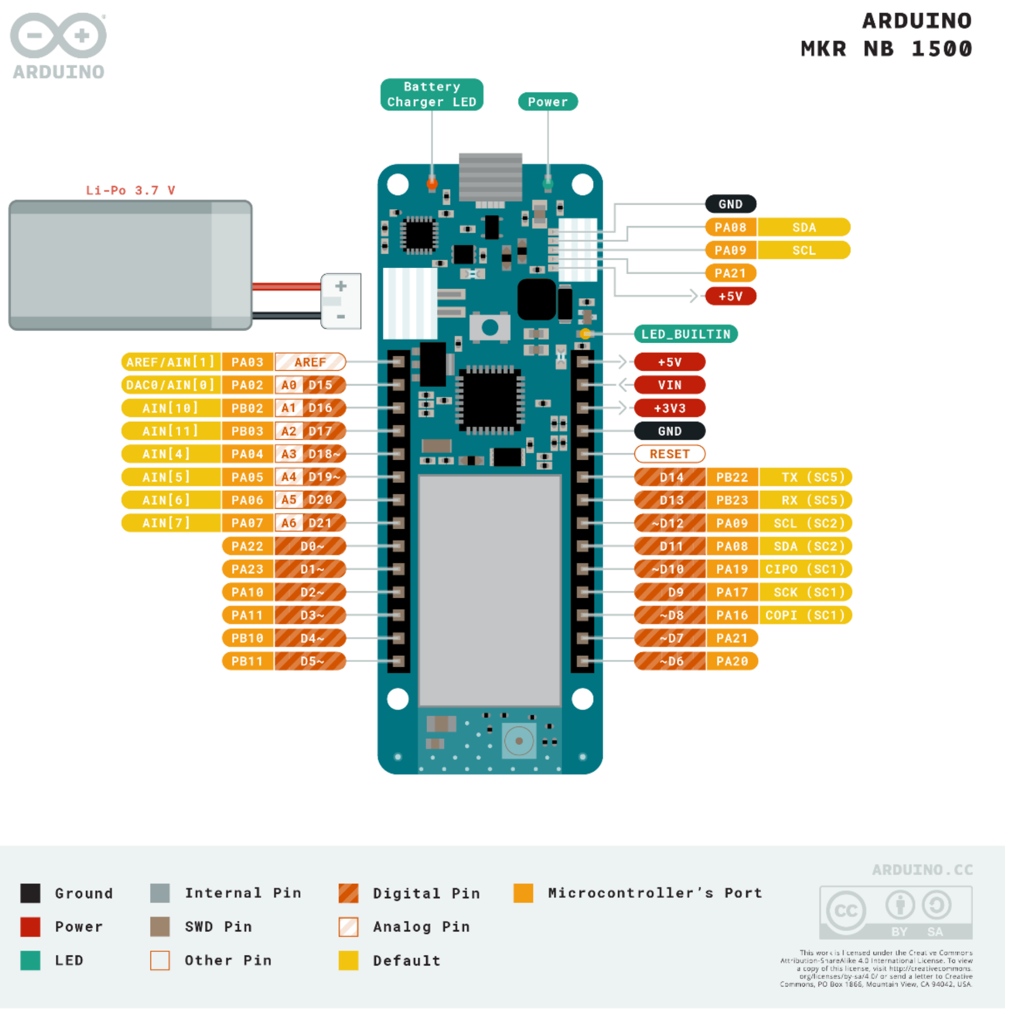


Figure 71- Inputs Arduino MKR NB 1500

uBlox SARA-R410M-02B

This module can communicate over the LTE-M, NB-IoT and EGPRS networks. It is possible to send and receive SMS, and supports several protocols for sending data over the Internet, including TCP, UDP and HTTPS.

Coverage

NB-IoT and LTE-M coverage includes many regions, such as Europe, North & South America and large parts of Asia.

Cortex-M0 32-bit SAMD21

The powerful, low-power processor that is used in all MKR Family boards.

ATECC508 crypto chip

The ECC508 crypto chip makes sure your data remains secure and private, and can store up to 16 keys in an EEPROM array.

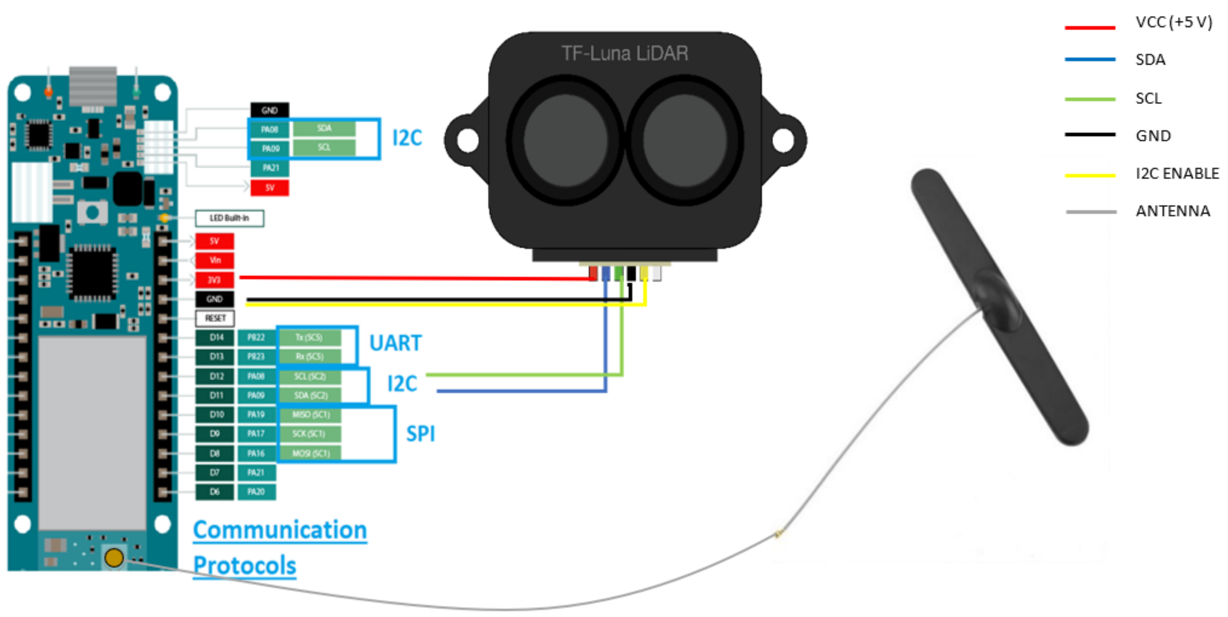


Figure 72- Explanation of the electric connections

In order to establish communication between the sensor and the board we can use either I2C or UART ports of the Arduino, as the sensor can be programmed to be connected in both ways, but we finally decided to use the I2C. Then, for connecting the board to the internet, we need to plug in an antenna, as it will enable the board to transmit and receive signals. We will use an antenna that works in the same frequencies as the NB-IoT works.

Battery

Finally, and as we already mentioned, the sensor must be battery powered. For this reason, we will also use a battery with enough power to leave the sensor working for some months without the need to charge it. The battery model we are using is LIPO-903450, with a voltage of 3.7 V and a capacity of 1600 mAh that is enough for powering the sensor for some months.

Figure 73- Battery PL 903450

Coding Software

Arduino IDE

Arduino is an important open-source software and hardware company that designs microcontrollers. These microcontrollers or boards are equipped with many input/output digital and analog pins, as well as some communication ports and other features.

For the programming of the boards, Arduino provides his own integrated development environment, the Arduino IDE. This is a cross-platform application that supports both C and C++ languages using a standard API, known as Arduino language. This IDE also supplies a wide software library that contains examples of using for many different projects, so you don't need to start from the bottom. It also includes a serial monitor that allows the user to read the messages sent by the device.

One of the most interesting things of using Arduino is that is easy to start with and the user can find some free codes available for the project they want to develop as there is many people updating the libraries with new projects, resolving doubts in the internet forums and providing tutorials on certain tasks.

In this project, we will implement the programming of the boards by using the Arduino IDE. We can take profit from it by using some libraries, so we can focus more on communication, making the programming of the sensor itself easier. It is also a great help when testing the program as we can see the outputs in the serial monitor.

MQTT

MQTT is a network protocol used in IoT communication. It works by using a publish-subscribe method that transports messages between devices. It is designed for being used in remote locations where the bandwidth is limited.

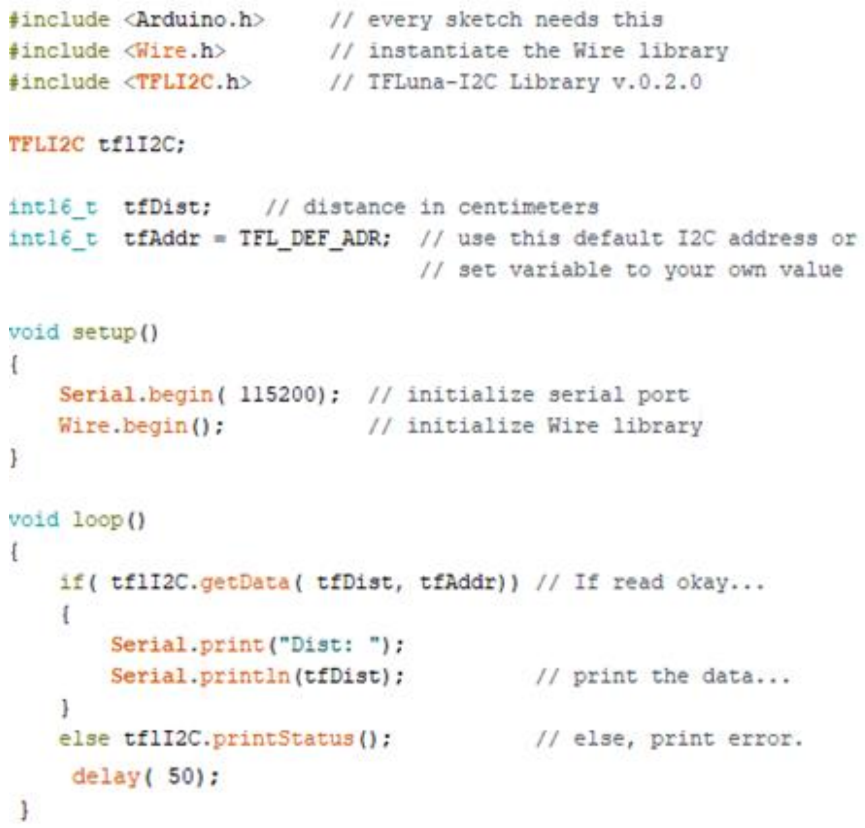
There are two different parts involved in MQTT communication: the broker and the clients.

A broker is a server that receives messages from all the clients and sends them only to the clients that requested it, while the clients are every device that is connected to this MQTT broker, and they can send and receive information.

When a client wants to transmit some data, they can publish a message. This publication consists of a topic that acts like an identifier of the message, and the data, which is the actual message. Once a message is published, other clients can ask for this data by subscribing to this topic, so every time a message from this topic is published, the client will receive it. A single client can be published and subscribed at the same time, so it can be using the data from another client before publishing his message.

Coding

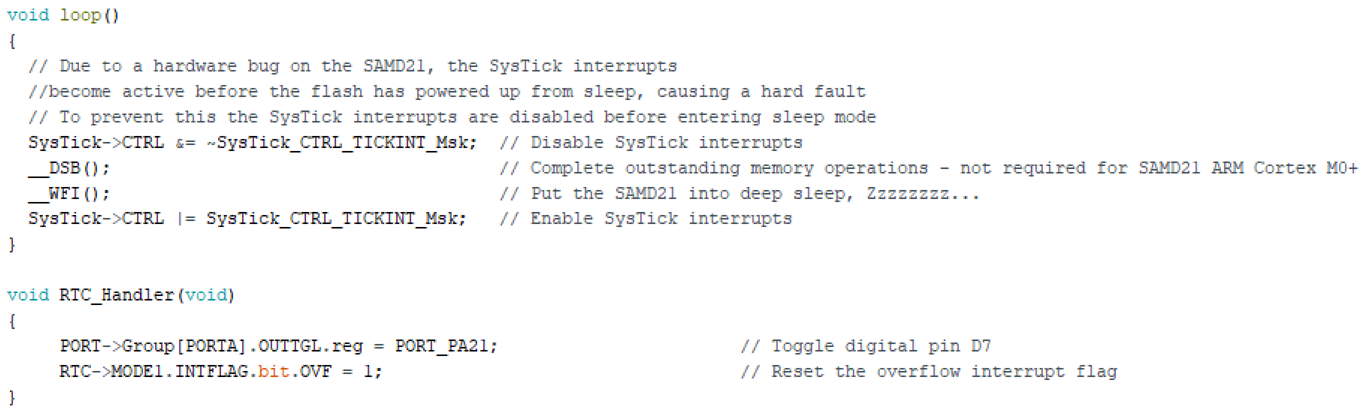
The first step was to research about this concrete Arduino board as we never worked with it before but also about the lidar sensor. There were some libraries available for using the sensor, but we had some issues because this Arduino board did not support several features that are available in other boards that are more common, such as the Arduino Uno. This library was communicating the device by using two serial connections, but our board only supported one serial connection as it only has one serial port. For this reason, we searched for another library that could be used in our board and we found one that worked correctly.



```
#include <Arduino.h>     // every sketch needs this
#include <Wire.h>        // instantiate the Wire library
#include <TFLI2C.h>      // TFLuna-I2C Library v.0.2.0

TFLI2C tflI2C;

int16_t  tfDist;    // distance in centimeters
int16_t  tfAddr = TFL_DEF_ADR;  // use this default I2C address or
                                // set variable to your own value

void setup()
{
    Serial.begin( 115200);  // initialize serial port
    Wire.begin();           // initialize Wire library
}

void loop()
{
    if( tflI2C.getData( tfDist, tfAddr)) // If read okay...
    {
        Serial.print("Dist: ");
        Serial.println(tfDist);          // print the data...
    }
    else tflI2C.printStatus();           // else, print error.
     delay( 50);
 }
```

We tested it in diverse ways, such as measuring the distance between the sensor and a wall ore measuring other objects as a person. Then finally we tried to measure it with snow, and it worked precisely, even if the snow was not flat.

We also implemented a sleeping mode to save energy and allowing it to work for more time, as it is going to be battery powered, but we encountered a problem. Apparently, there is a bug in the hardware of the board that enables the SysTick interrupts before the flash has powered up from sleep, and this caused a higher consumption rate that expected when using Arduino Low Power library, but we managed to find a code that configures the board to solve this problem and reduces the power consumption of the board. The code consists in the configuration of the RTC and NVIC in the setup to obtain the correct sleeping time and interruption to wake it up. Then, we have the loop, when it goes to sleep and the RTC handler, who manages the interruption:



```
void loop()
{
  // Due to a hardware bug on the SAMD21, the SysTick interrupts
  //become active before the flash has powered up from sleep, causing a hard fault
  // To prevent this the SysTick interrupts are disabled before entering sleep mode
  SysTick->CTRL &= ~SysTick_CTRL_TICKINT_Msk;  // Disable SysTick interrupts
  __DSB();                                     // Complete outstanding memory operations - not required for SAMD21 ARM Cortex M0+
  __WFI();                                     // Put the SAMD21 into deep sleep, Zzzzzzzz...
  SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk;   // Enable SysTick interrupts
}

void RTC_Handler(void)
{
     PORT->Group[PORTA].OUTTGL.reg = PORT_PA21;                        // Toggle digital pin D7
     RTC->MODE1.INTFLAG.bit.OVF = 1;                                   // Reset the overflow interrupt flag
}
```

Finally, we implemented the communication part using the NB-IoT technology. In this process, we had some complications, as we had some troubles with the SIM card we used in the Arduino, but when we tried to communicate with the internet provider, their answer was too slow, so we started to work in this task several weeks later than we wanted. When we managed to start with the programming, we found another issue, as the board was not able to connect to Novia's server and it took some time to us to find out that we were encrypting the messages, but Novia's server does not support encryption, so it always failed to connect. Despite this, the rest of the task was developed without problems using the MQTT communication, that is simple and intuitive to use.

The first part of the code was declaring the SIM card, broker, and device information. We declared them in another document called arduino_secrets.h to keep this information out of the code and making it simple to change the information if wanted.

```
#include <ArduinoMqttClient.h>
#include <MKRNB.h>
#include "arduino_secrets.h"

/////// Enter your sensitive data in arduino_secrets.h
const char    pinnumber[]   = SECRET_PINNUMBER;
const char    broker[]      = SECRET_BROKER;
const char    username[]    = SECRET_USERNAME;
const char    password[]    = SECRET_PASSWORD;
String        deviceId      = SECRET_DEVICE_ID;
```

After this, we have the setup and the loop, when the communication is done. In the loop, the code first tries to connect to the cellular network. Once that is done, it tries to connect to the broker, and later it sends a message and disconnects in order to save power.

```
void setup{
// Define MQTT parameters-------------------------
  mqttClient.setUsernamePassword(username, password);
  mqttClient.setId(deviceId);
}
void loop{
  // IoT communication-----------------------------------------------
  if (nbAccess.status() != NB_READY || gprs.status() != GPRS_READY) {
    connectNB();   // Connect to the cellular network
  }

  if (!mqttClient.connected()) {
    connectMQTT(); // Connect to the MQTT broker
  }

  mqttClient.poll();     // poll for new MQTT messages and send keep alives
  publishMessage();      // publish a message
  delay(100);
  nbAccess.shutdown();   // disconnects from the cellular network to save battery
  delay(100);
}
```

Finally, there are the specific functions for connecting to the cellular network, to the MQTT broker and for sending a message. We can see that for the first two functions, there is a loop where the code keeps trying to connect until it is finally done, because if we lost the connection, it will be impossible for us to collect the data. In the last function, it sends a message to the desire direction. In order to read this message, another device needs to subscribe to that topic.

```
void connectNB() {

  while ((nbAccess.begin(pinnumber) != NB_READY) ||
         (gprs.attachGPRS() != GPRS_READY)) {
    // failed, retry
    delay(1000);
  }
}

void connectMQTT() {

  while (!mqttClient.connect(broker, 1883)) {
    // failed, retry
    mqttClient.connectError();
    delay(5000);
  }
}

void publishMessage() {

  // send message, the Print interface can be used to set the message contents
  mqttClient.beginMessage("eps/Òmnia");
  mqttClient.print("Hello, we are Òmnia");
  mqttClient.endMessage();
  delay(1000);
}
```

Housing design

In addition to programming, our sensors need to be protected and contained in a package that can bring together all the components necessary for their proper functioning.

To develop the protection for our sensors, we needed the material that can fits as well as possible all the requirements. It means, a shape which fits good all the components, an easy-to-use tool but also an important point that is the material used for it. The material needs to be strong, to not break and to be resistant to the environment.

To achieve this, we decided to compare some common materials used for 3D printing.

Table 13- Comparative table for 3D printing materials

Materials	PLA (Polylactic Acid)	ABS (Acrylonitrile Butadiene Tyrene)	PC (Polycarbonate)	ASA (Acrylonitrile Styrène Acrylate)
Print speed	30 to 90 mm/s	30 à 70 mm/s		
Extrusion temperature range	180 à 220°C	230 à 250°C		80 à 100°C
Pros	- Easy to use - Biomaterial - Low temperatures - No warping* - Good bridging* - Compatibility with different types of 3D printers	-Impact resistant -Durable -Heat resistant -More likely to bend than break - Can be painted - Good mechanical strength	- Easy to post-process - Can be an interesting alternative to ABS as the properties are quite similar	- No warping - UV-resistant - Easy to use for 3D printing - High dimensional stability - High glass transition temperature - Chemical resistance
Cons	- Very poor mechanical strength - Not very heat resistant (softens around 50°) - Sensitive to moisture (store in a dry place)	- Not biodegradable - Requires heating tray - Complicated to print because of warping - Doesn't resist to hard environmental conditions	- UV sensitive	

Warping*: Good adherence

Bridging*: fast cooling capacity in the cantilever zone

Conclusion: After making the comparison with the most common materials that are used for 3D design, we can conclude that ASA seems to be the better choice we have. Of course, it retains its appearance and impact resistance even under adverse conditions, such as prolonged exposure to air, rain, cold, heat, etc.

Figure 74- ASA from the company Zortrax

ASA is the material that can be used for the time of the real application of our sensors. For all our prototypes made with the 3D printer, we will use the better material available to us here in the university: PLA.

The snow sensor works with other components that need to be included in the housing design: one antenna, one battery, one Arduino board and of course some cables for the connection between all of them.

Designing the case

We started of course with some prototypes. First, the case that will contain all the components.

The first prototype was, of course, a start, to better perceive where to place each component, without forgetting the necessary connections between them. Indeed, the antenna, the battery and the sensor will all 3 be connected to the Arduino board. It is therefore necessary to take into consideration the connection space between these elements.

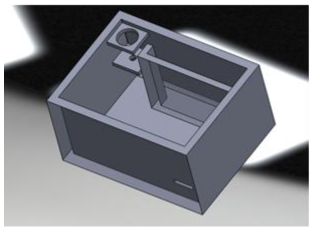


Figure 69- Prototype 1 housing design

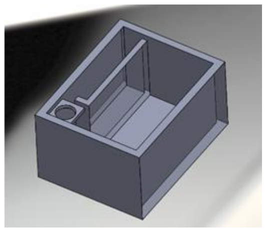


Figure 70- Prototype 2 housing design

On these first prototypes, we first thought about the position of the antenna and the battery. The two holes on the left are used to guide and hold the antenna and the separation on the right is used to hold the battery.

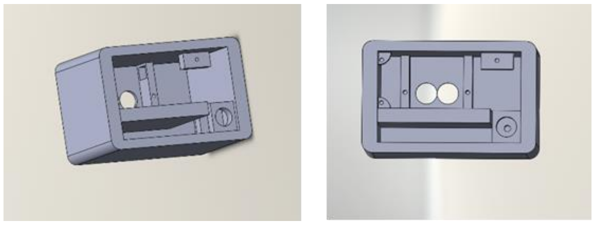


Figure 71- Prototype 2 housing design

On the following prototypes, the sensor and the Arduino board have been added.

For the snow depth measurement, the sensor lenses must be oriented towards the ground. It was therefore necessary to find a way to position it in this way. These elements will be held in place with screws.

We have also added fillets at the corners for a more aesthetic look.

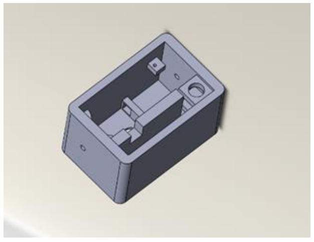


Figure 72- Prototype 4 housing design

On this prototype, several elements have been modified/added.

First of all, in order to guide the many cables of the sensor, a recess has been added so that all the wires can pass through more easily.

Then, to allow the connection between the battery and the arduino board, a cut has been made in the separation. The space is necessary to let the cables pass through and join the two elements.

In addition, with a view to a cover to close and protect the set, holes have been created to screw the set.

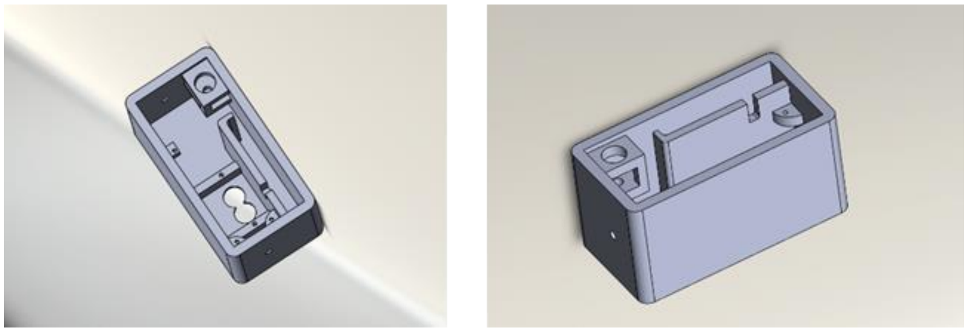


Figure 73- Prototype 5 housing design- final one

For the result, adjustments were made to some of the distances, including the holes for Arduino and the antenna. In addition, we realized that there was not enough free space for all the cables to pass through, so it was necessary to enlarge the box.

Fillets were also added for a more aesthetic look once again.

<u>The cap:</u>

Due to the climatic conditions in Finland and the sensitivity of the material used, it is important to protect the box from all possible weather conditions and of course snow. Therefore, it was necessary to think of a cover with a shape that would prevent snow from piling up on the box and risking freezing the system components.

So, the idea came to us to make a lid with an oval shape so that the snow could slide off and not stay on top. In addition, there is a hole for the antenna to pass through, as well as holes to close it.

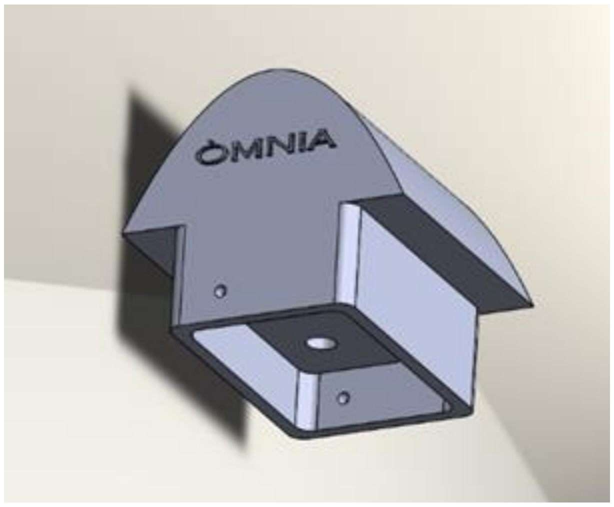


Figure 74- Cap for housing design

In order to understand the realization of the box, this explanatory diagram provides details of the measurements of the piece.

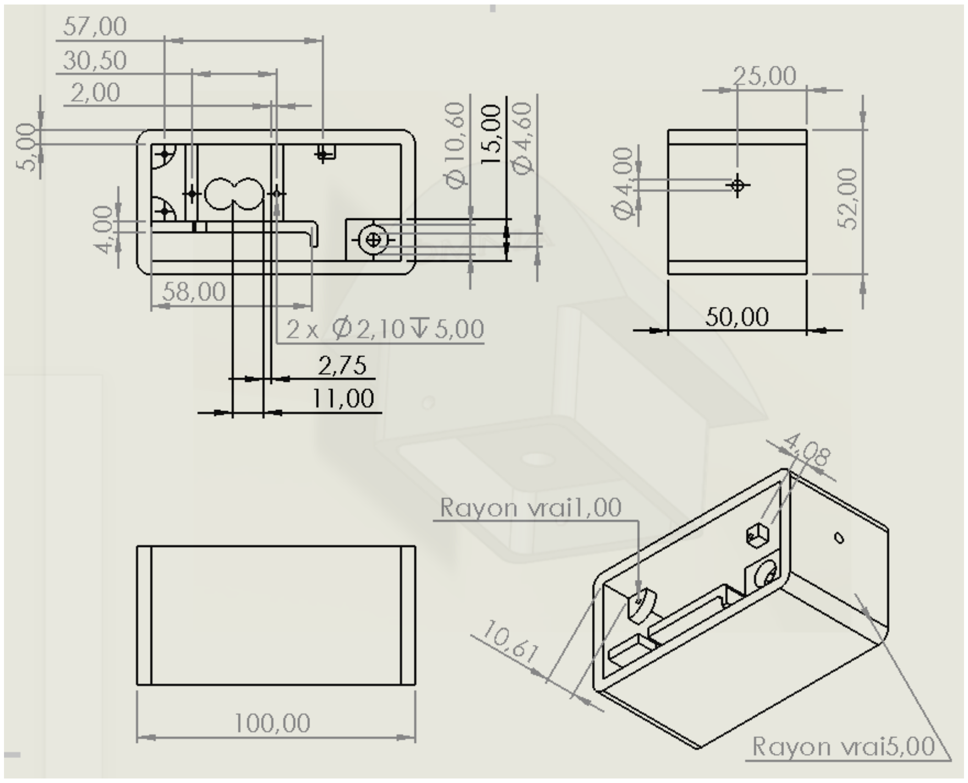


Figure 75- Schematic part housing case

User manual

The Òmnia snow depth sensor is a device able to measure the snow depth and send the data remotely using cellular networks and IoT.

This device must be located outdoors and in an open area in order to let snow pile up and take a more accurate measurement, and at a fixed height, as it is important to know the distance between the device and the floor to calculate the snow depth. The height of the sensor must be defined in the code changing the value of the height variable:

```
uint16_t      height         = 300;   // sensor height in centimeters
```

By default, the height of the sensor in the code will be 300 centimeters, but it can be defined up to 800, as it is the maximum range of the sensor.

For placing the sensor, the side with the lens of the laser must be pointing to the floor in such a way that that face is completely parallel to the floor and the beam hits it perpendicularly.

Once the device is placed and working, the data will be sent to the Novia's MQTT broker with the topic "eps/Òmnia/SnowDepth" every 30 minutes, and the message will be the value in centimeters. For reading the data, another device must subscribe to the topic and will receive the messages.

7.4 SEA WATER LEVEL SENSOR

Hardware

ABP series pressure sensor

To develop this sensor, we are going to use a differential pressure sensor from the Honeywell ABP-series. As we already explained, after comparing all possible sensors, this kind of sensor seems to be the best option for measuring in the sea.

Another advantage of these sensors is that they are calibrated, and temperature compensated, so water temperature changes will affect less to the measurement. They are also small, allowing us to assemble more easily in any location.

The sensor that we are going to use is the ABPDJJT005PDSA3. This sensor has an approximately range of ±3.5 meters of water, a high resolution of 12 bits and an accuracy of ± 0.25 %FSS BFSL.

Operating range	±5 psi (±3.5 m)
Accuracy	± 0.25 %FSS BFSL
Range resolution	12 bits
Response time	0.46 ms
Communication	SPI

Figure 75- ABPDJJT005PDSA3

Arduino MKR NB 1500

In the same way as for the snow sensor, we are also going to use the same board, an Arduino MKR NB 1500 with the same specification.

For the communication between the sensor and the board we are going to use the SPI port. We will also need the antenna for the IoT communication of the board.

Battery

For the battery, we will use the same model as for the snow sensor, the LIPO-903450.

Coding Software

The coding software used for developing the water level sensor is the same that for the snow sensor: the Arduino IDE and an MQTT broker.

Coding

For this sensor, there is less to do, as the IoT communication and the sleeping mode that we developed for the last sensor are also useful for this one. We found a library for the same sensor we were using, that communicates the sensor and the board using the SPI port. The code is quite simple:

```
#define slaveAPin A1
#include <Arduino.h>
#include <SPI.h>

double        tranferFuntion(int16_t data);
short         data;
// set up the speed, data order and data mode
SPISettings settingsA(425000, MSBFIRST, SPI_MODE0);

void setup() {
  //Configure slaveAPin as an OUTPUT
  pinMode (slaveAPin, OUTPUT);
}

void loop(){
  // Read the data from the sensor--
  SPI.begin();
  SPI.beginTransaction(settingsA);
  digitalWrite (slaveAPin, LOW);
  data = SPI.transfer16(0x00);
  digitalWrite (slaveAPin, HIGH);
  SPI.endTransaction();
  SPI.end();
}

double tranferFuntion(int16_t data) {
    return 70.307*5*((data - 0.1 * 16384 ) / (0.4 * 16384) - 1);
}
```

First, there is a configuration of some settings as the speed, mode, data order and defining the slave pin as an output. Then, in the loop, the sensor reads the data and sends it to the board via SPI. Finally, there is the transfer function, that transforms the data received in bits to centimeters.

Housing design

The next step in our progress in order to manufacture a quality product is to develop a case for the components. First of all, we define as well the aesthetics as the specifications. About the style, words as minimalism, simple but functional are relevant. The ultimate goal is to protect the different components inside. In addition, the product consists of two parts, a bottom and top connected with each other. The final step to make it a real product produced by our company, is the logo added on the product.

Here is the development progress without the final version that is printed in black PLA. The first three prototypes are printed in grey PLA, then one in PET to test that material and finally the last one in PLA. These versions are prototypes, and the design is flexible for other innovations.

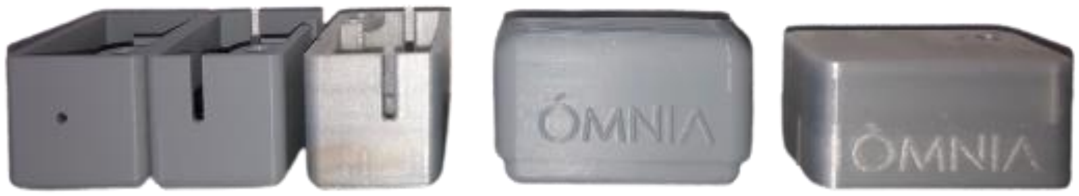


Figure 77-Prototypes made in Ultimaker, the 3D printers of the Novia University

Prototyping and iterating the design is the way to find the ultimate solutions for products.

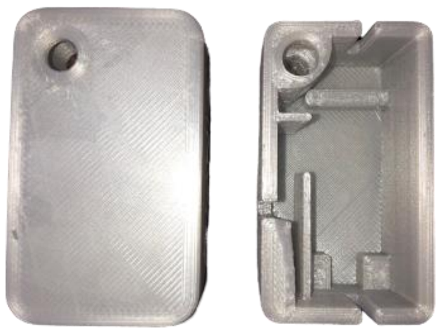


Figure 77 -front view of cover and case in PET

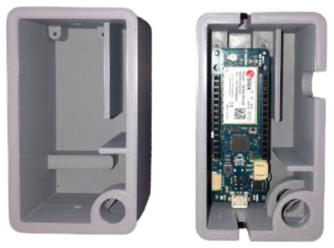


Figure 76 - front view of case without and with Arduino.

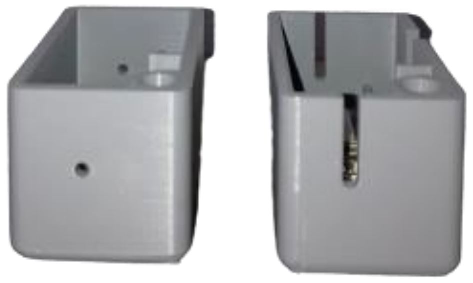


Figure 78 -Prototypes to solve the problem of the connection between inner and outer world

Clearly, the outside is for protecting Inside components against different weather conditions. Since it will be located in areas close to water, it is important to make it as waterproof as possible, fot instance by using the right material. Working with a double layer, the cover and the product, makes it more resistant against extremely weather conditions or water surroundings. This product is designed in SolidWorks Simulation Student Engineering Kit (SEK) 2021-2022.

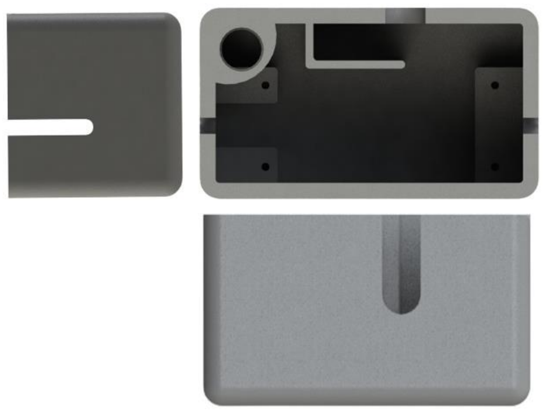


Figure 79- The two side views and top view of the water design

Additionally, something more about the connections and components inside. On the top left there are holes in order to connect the antenna, that communicates between the inner technological world and the outer real world. The four holes are there to connect the Arduino with the product. Inside there is a longer hole in order to make it simple to connect the two parts, and to leave some space to give the components enough place. Then the battery is placed next to the antenna, which is also indicated with a L-shape boundary.

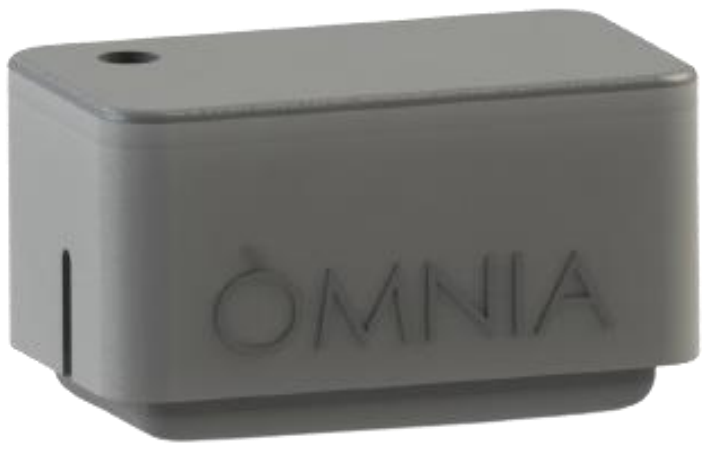
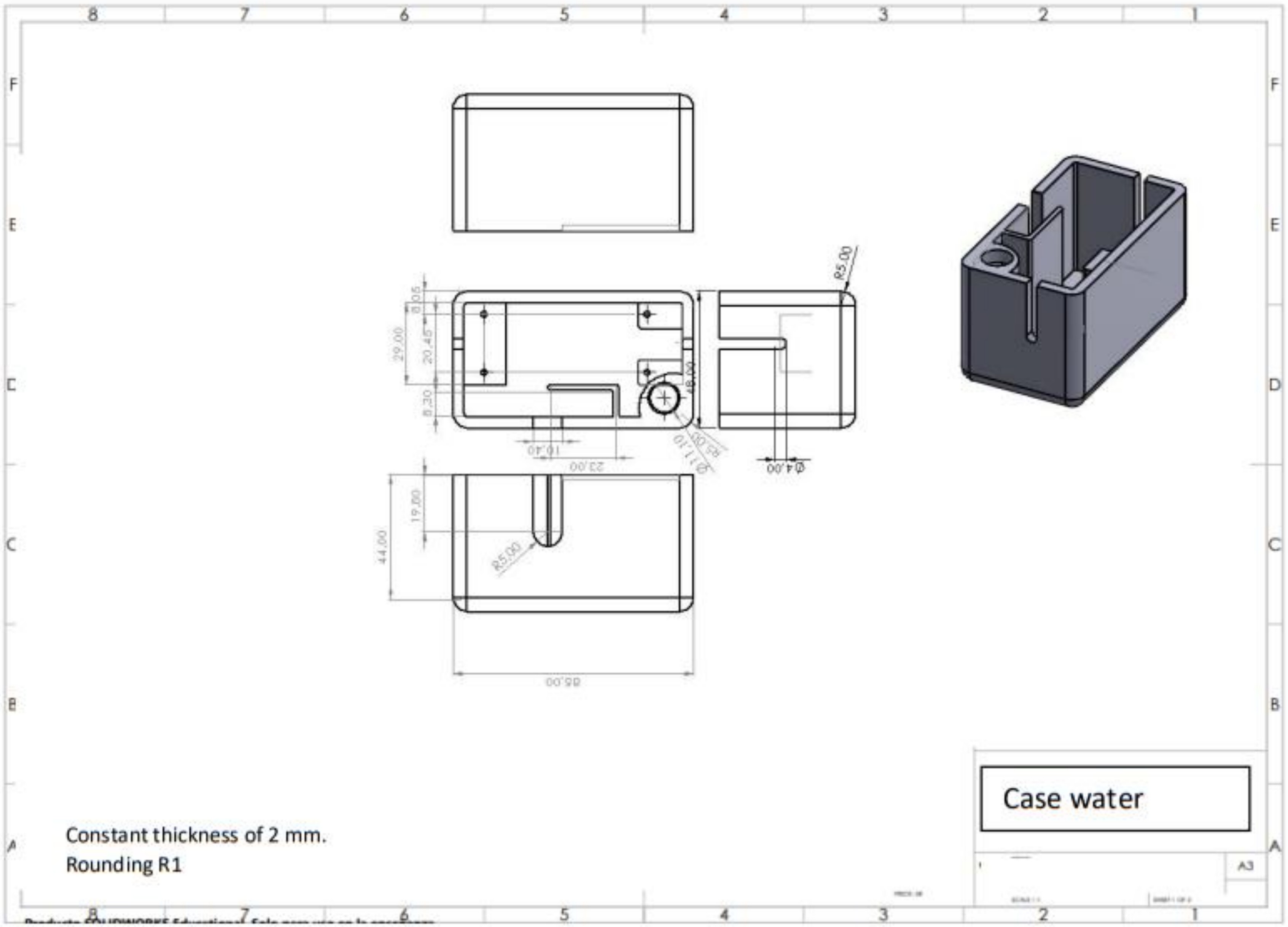


Figure 80- the design of the water sensor

Here is how it looks like in the real world. With both the antennae and the hose connected to the outside world. About the material, here on the pictures is still a prototype and it is printed in though PLA, but for real life situations we choose ASA, a strong material, good for a lot of weather conditions. The cable is connected with the inner technological world and the outside world where there is water in the surrounding to make accurate measurements.

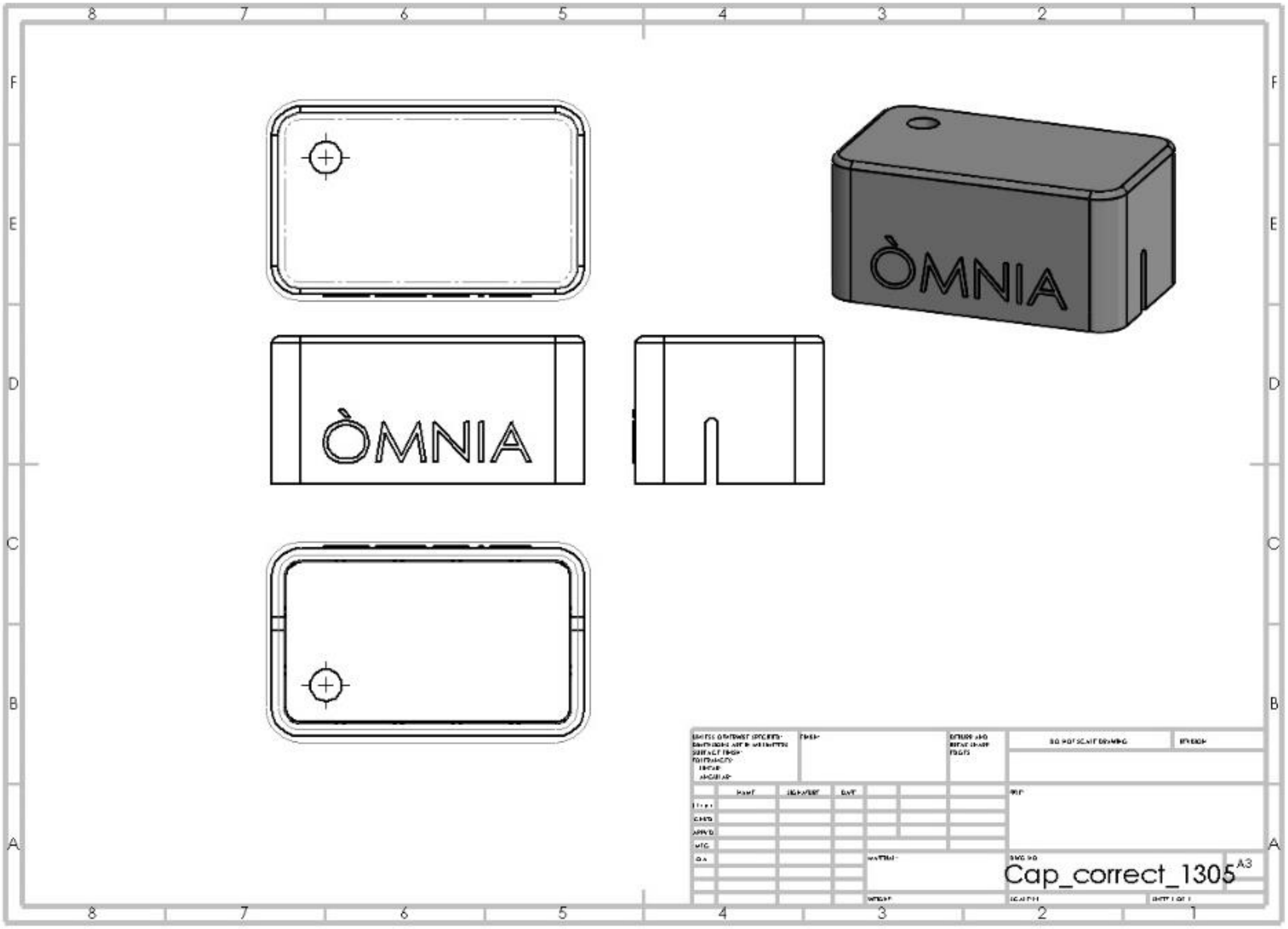
Here are the technical drawings made in SolidWorks. This is the final important step to manufacture the design.

Next, it is a final example of a technical drawing of the cap of the case, which is the same sizes based on the case of the water.



Case water

Constant thickness of 2 mm.
Rounding R1


ÒMNIA
ÒMNIA
Cap_correct_1305
A3

Finally, here is an example how it looks like in the real world. With both the antennae and the hose connected to the outside world. About the material, here on the pictures is still a prototype and it is printed in though PLA, but for real life situations we choose ASA, a strong material, good for a lot of weather conditions. The cable is connected with the inner technological world and the outside world where there is water in the surrounding to make accurate measurements.

User manual

The Òmnia water level sensor is a device able to measure the sea water level and send the data remotely using cellular networks and IoT.

This device must be located in the water. It is designed specifically to work in the sea, but it can also be used for measuring different water levels, as in a lake or a river.

When the sensor is placed, the end of one of the lateral tubes must reach the surface, while the other one will go to the bottom, in such a way that the sensor stays near to the average water level.

Once the device is placed and working, the data will be sent to the Novia’s MQTT broker with the topic “eps/Òmnia/WaterLevel” every 60 minutes, and the message will be the value in centimeters. For reading the data, another device must subscribe to the topic and will receive the messages.

8 SUGGESTIONS

Here is some advice for the engineers that we will pass the torch to. Here are some gained insights this semester or things that we want to tell the next IoT group. Sometimes there are component missing, or sometimes it is difficult to order special devices in the scheduled time and perhaps also budget. For the programming part it is important to start as soon as possible as the semester start to make sure that you calculate enough time to deal with the different issues you get and in order to overcome the challenges.

For the snow depth and water level sensors, the battery duration is an important issue, as those sensors are going to be placed in the surroundings of the city, but we had some troubles to optimize this as the Arduino MKR NB 1500 board has some bugs when using the Arduino Low Power library and this makes the average consumption in sleep mode of around 20 mA. The solution we found is an upgrade to this as we managed to obtain a consumption near to 1 mA. There is also a problem in the time the loop takes to execute, as the average time is around 20 seconds per cycle and reducing it can also help to the battery saving. However, there is still quite difference with other boards, so it would be interesting to go deeper into this to expand the battery cycle.

Finally, for the vibration sensor, we already talked about the battery problem. It would be interesting to replace the actual battery system, that is a temporary solution, with the connector of the NICLA SENSE ME, as it will be a better option in many ways.

9 CONCLUSION

In conclusion, our multidisciplinary team started almost fully from scratch with the project, that is now developed in an innovative, technology start-up ‘company’, from nothing to everything, Òmnia. During the spring semester 2022 we had weekly meetings with our supervisor Lindén Hans in order to achieve the vision created in particular in the Project Management courses. Our vision was to deliver final products, our team accomplished this by starting with profound research, then programming and dive in the world of Internet Of Things and how data is connected, then the case development and how to develop products that in our case are a product that measures the snow depth or sea water level, that competent to count cars and a device for vibrations. As in our branding, the website, poster and social media is stated that we are a business open for new project and ideas, our main goal is still connecting citizens through sensors. After all, our team is proud of our challenges and how we tackled them, and our final deliverables. Our company omnia is curious about where the future of Internet of Things will lead us.

Here is the link to our EPS spring 2022 aftermovie:

https://youtu.be/Ld7kxWP6Fbw

10 APPENDIX

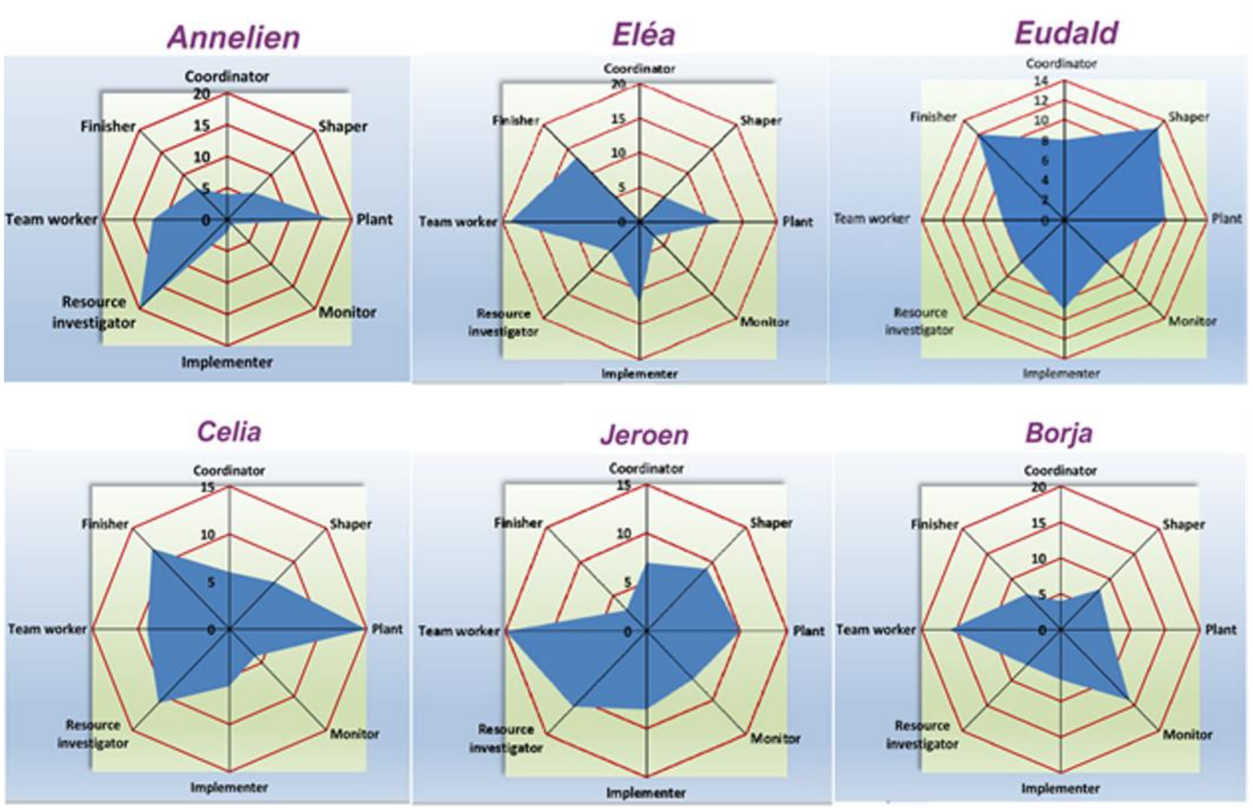
10.1 BELBIN TEST

Dr Meredith Belvin researched for many years trying to understand why some teams succeed while others fail in his work. This research led Belvin to discover his Nine Belvin Team Roles. This study explains how there are 9 different roles when working as a team and that a good team is a team in which these 9 roles are represented. Usually, every person has a dominant role and one or two secondary roles in a way that one person can adapt more than one role. In addition, Belvin explained that for a good functioning of a team, every person needs to focus on the positive points of his roles, because if the team is well balanced, the weaknesses of a person will be cleared by the strengths of another one. For that reason, is very important that everybody knows his roles and his teammate's roles to ensure a good performance.

Which are this Nine Belbin Team Roles?

Resource Investigator	**Always bring back ideas to the team thanks to their inquisitive nature**
Team worker	Creates a good environment to work, helping the team to come together as a team, identifying the work to do and finishing it
Coordinator	The one who helps the team to find their objectives, knows what each person should do and is able to delegate the work effectively
Plant	Very creative and imaginative, always find good solutions thanks to his skills in an unconventional way
Monitor	Helps the team by filtering ideas in an impartial way, using his logic and good judgement
Specialist	An expert of a concrete field of knowledge
Shaper	Makes sure than the team reaches his objectives driving them by the good way and making them not lose their focus
Implementer	Finds a good strategy to follow and carry it out in an efficient and successful way
Finisher	Always checks the work done seeking for work errors to fix, making sure that the final product has the best possible quality

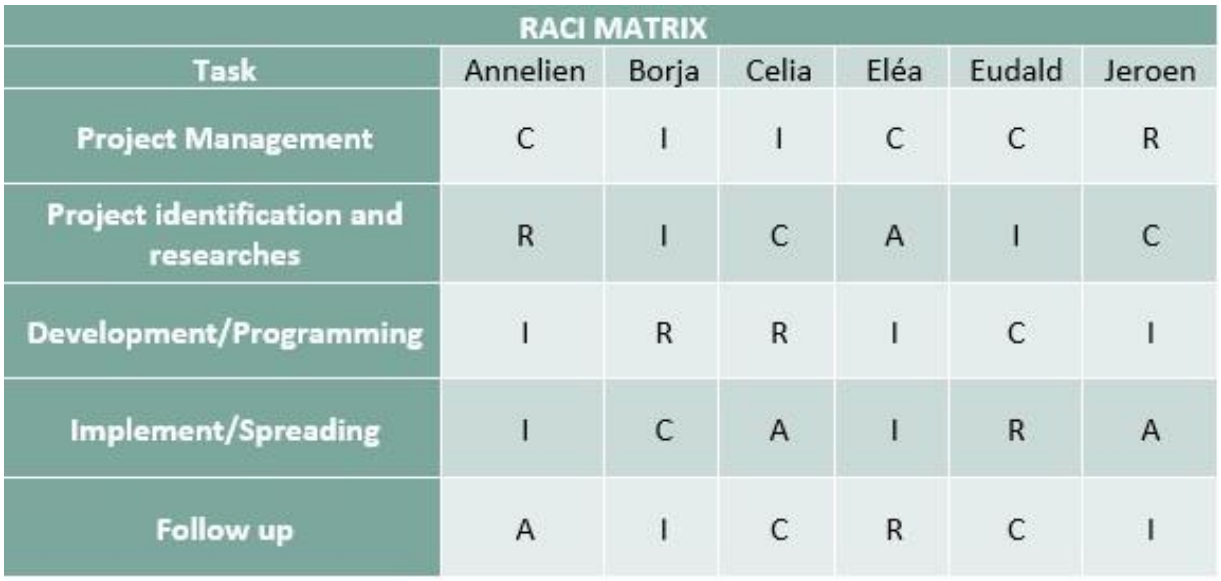
The Belbin Team Roles in our team:



We can find many different roles in our team. Half of the team has Team worker as their main role, so it will be easier for us to create this friendly environment that will make the team perform better. The other three main roles are Resource Investigator, Finisher and Shaper. As a secondary role, we can also find in our team several Plant roles, a monitor, an Implementer, and some other roles that are already represented in a main role. Our team is quite balanced when we refer to team roles as we have almost every role present in the team. It is true that there is a lack of the coordinator role, but despite this, we think we could overcome to this and that our team will be able to perform in a good way.

10.2 THE RESPONSIBILITY MATRIX (RACI MATRIX)

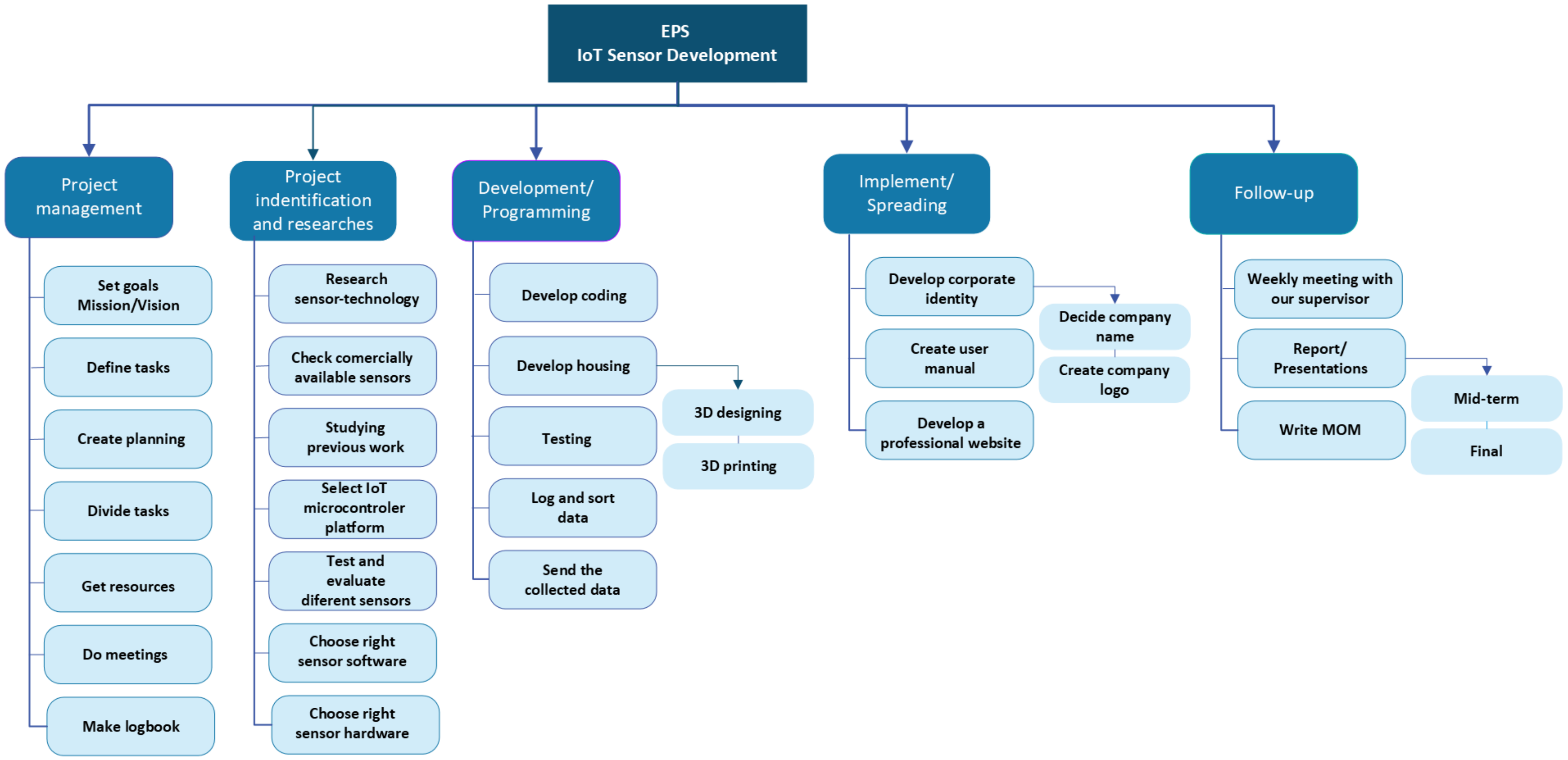
Together with the team, we created the responsibility matrix. Here we defined the roles of each team member in each task. This ensures that someone always has a responsibility for a task and will therefore always strive for progress within this task. We have placed our focus on ensuring that everyone has an equal amount of responsibility (equally R, A, C and I). This way everyone will be equally involved in the project



RACI MATRIX						
Task	Annelien	Borja	Celia	Eléa	Eudald	Jeroen
Project Management	C	I	I	C	C	R
Project identification and researches	R	I	C	A	I	C
Development/Programming	I	R	R	I	C	I
Implement/Spreading	I	C	A	I	R	A
Follow up	A	I	C	R	C	I

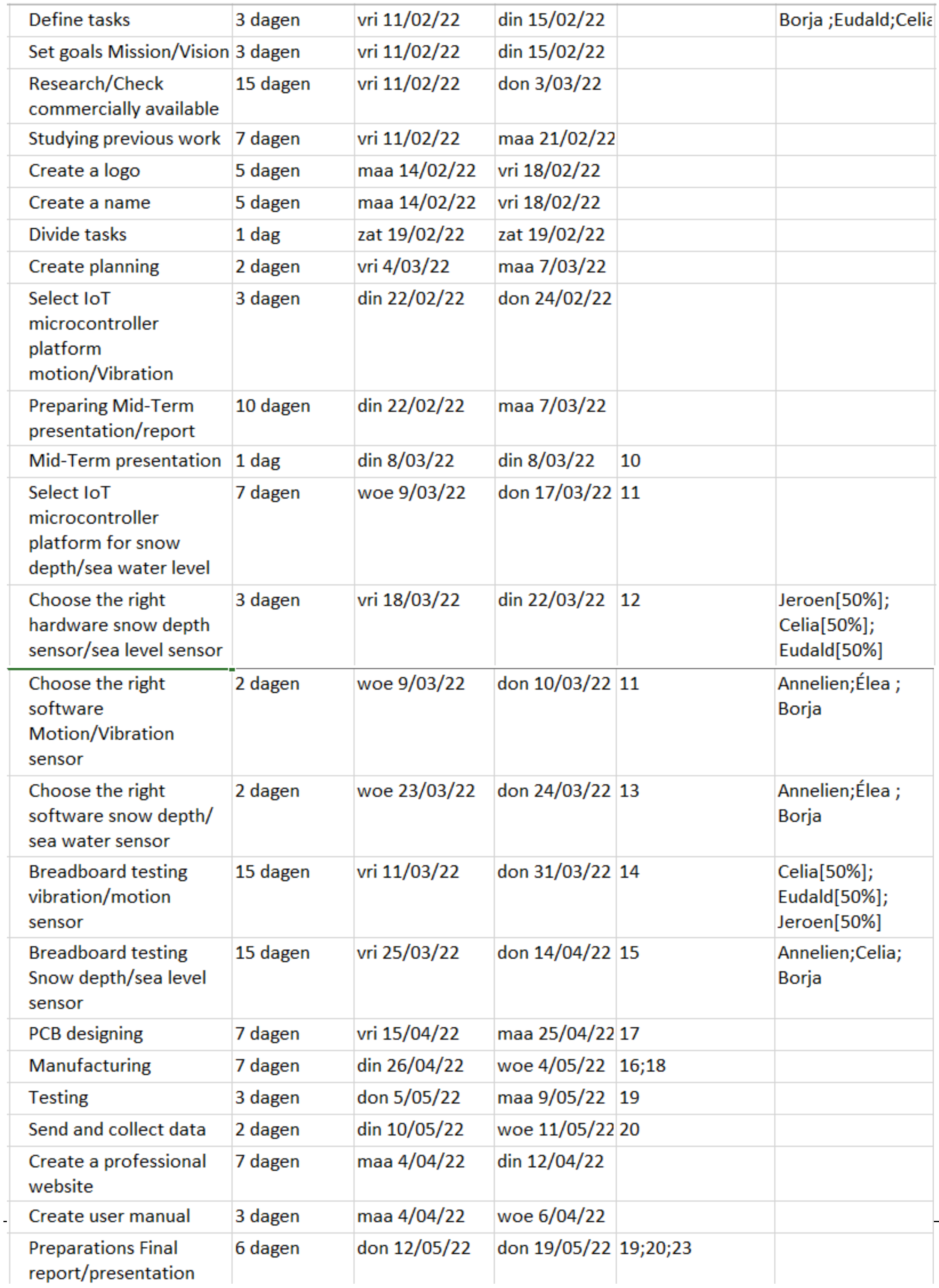
10.3 WORK BREAKDOWN STRUCTURE (WBS)

This is the work distribution structure that we, as a team, have established for the project "IoT Sensor Development". Here we have divided all the tasks that need to be performed by the team into different categories. This way we create a clear overview of exactly what tasks will have to be performed throughout this project.

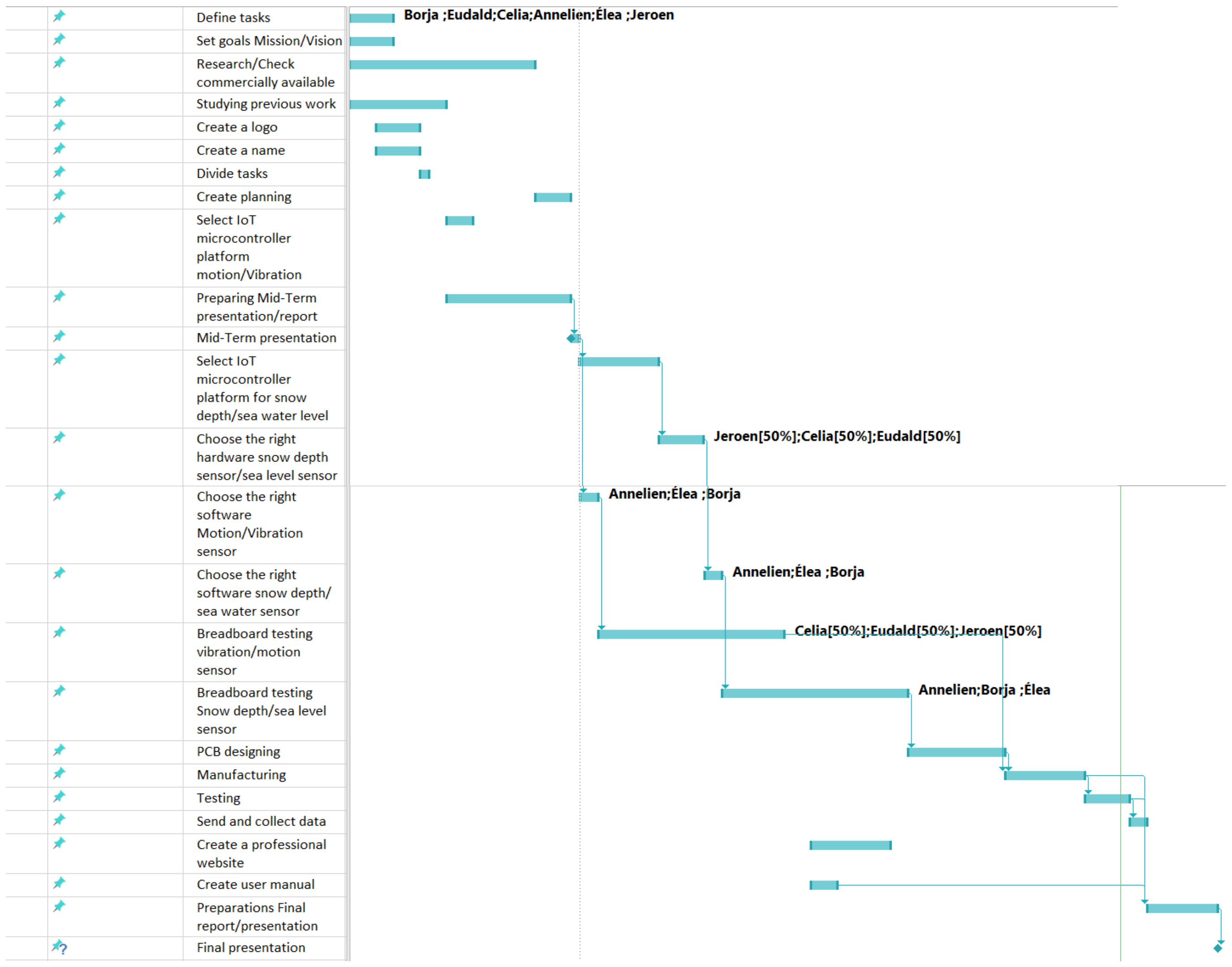

EPS
IoT Sensor Development
Project management
Set goals Mission/Vision
Define tasks
Create planning
Divide tasks
Get resources
Do meetings
Make logbook
Project indentification and researches
Research sensor-technology
Check comercially available sensors
Studying previous work
Select IoT microcontroler platform
Test and evaluate diferent sensors
Choose right sensor software
Choose right sensor hardware
Development/ Programming
Develop coding
Develop housing
3D designing
3D printing
Testing
Log and sort data
Send the collected data
Implement/ Spreading
Develop corporate identity
Decide company name
Create company logo
Create user manual
Develop a professional website
Follow-up
Weekly meeting with our supervisor
Report/ Presentations
Mid-term
Final
Write MOM

10.4 GANTT CHART

The Gantt Chart is a useful project management tool to manage the tasks, while keeping a view on the planning. It is a clear visual presentation of activity and time. We divide the big project in different smaller tasks. By working with different colors, we get a better overview to see what kind of tasks belongs together.



Define tasks	3 dagen	vri 11/02/22	din 15/02/22		Borja ;Eudald;Celia
Set goals Mission/Vision	3 dagen	vri 11/02/22	din 15/02/22		
Research/Check commercially available	15 dagen	vri 11/02/22	don 3/03/22		
Studying previous work	7 dagen	vri 11/02/22	maa 21/02/22		
Create a logo	5 dagen	maa 14/02/22	vri 18/02/22		
Create a name	5 dagen	maa 14/02/22	vri 18/02/22		
Divide tasks	1 dag	zat 19/02/22	zat 19/02/22		
Create planning	2 dagen	vri 4/03/22	maa 7/03/22		
Select IoT microcontroller platform motion/Vibration	3 dagen	din 22/02/22	don 24/02/22		
Preparing Mid-Term presentation/report	10 dagen	din 22/02/22	maa 7/03/22		
Mid-Term presentation	1 dag	din 8/03/22	din 8/03/22	10	
Select IoT microcontroller platform for snow depth/sea water level	7 dagen	woe 9/03/22	don 17/03/22	11	
Choose the right hardware snow depth sensor/sea level sensor	3 dagen	vri 18/03/22	din 22/03/22	12	Jeroen[50%]; Celia[50%]; Eudald[50%]
Choose the right software Motion/Vibration sensor	2 dagen	woe 9/03/22	don 10/03/22	11	Annelien;Élea ; Borja
Choose the right software snow depth/ sea water sensor	2 dagen	woe 23/03/22	don 24/03/22	13	Annelien;Élea ; Borja
Breadboard testing vibration/motion sensor	15 dagen	vri 11/03/22	don 31/03/22	14	Celia[50%]; Eudald[50%]; Jeroen[50%]
Breadboard testing Snow depth/sea level sensor	15 dagen	vri 25/03/22	don 14/04/22	15	Annelien;Celia; Borja
PCB designing	7 dagen	vri 15/04/22	maa 25/04/22	17	
Manufacturing	7 dagen	din 26/04/22	woe 4/05/22	16;18	
Testing	3 dagen	don 5/05/22	maa 9/05/22	19	
Send and collect data	2 dagen	din 10/05/22	woe 11/05/22	20	
Create a professional website	7 dagen	maa 4/04/22	din 12/04/22		
Create user manual	3 dagen	maa 4/04/22	woe 6/04/22		
Preparations Final report/presentation	6 dagen	don 12/05/22	don 19/05/22	19;20;23	



Define tasks
Set goals Mission/Vision
Research/Check commercially available
Studying previous work
Create a logo
Create a name
Divide tasks
Create planning
Select IoT microcontroller platform motion/Vibration
Preparing Mid-Term presentation/report
Mid-Term presentation
Select IoT microcontroller platform for snow depth/sea water level
Choose the right hardware snow depth sensor/sea level sensor
Choose the right software Motion/Vibration sensor
Choose the right software snow depth/ sea water sensor
Breadboard testing vibration/motion sensor
Breadboard testing Snow depth/sea level sensor
PCB designing
Manufacturing
Testing
Send and collect data
Create a professional website
Create user manual
Preparations Final report/presentation
Final presentation

10.5 PROTOTYPE AND FEEDBACK

During a project, each member of the group or the client's side is sometimes difficult to make themselves understood. Indeed, each one will use his own words and have a precise idea in mind while everyone else will understand something different. The perfect example is the swing set. To avoid this kind of problem, it is important to make a point during the project, by expressing in a concrete way, by images, diagrams, what is expected.

Here are the questions we asked to our stakeholders followed by his answers:

• **What things do the stakeholders like about your proposal/mock-up?**

You have looked at commercially available sensors and different technology that they use. You describe well different sensors to use.

• **What things do they like less - if you force them to name a few things?**

Not much you know quite well how the sensors should be constructed. Power supply to the sensors could be looked into also. Snow depth and water level sensors should be battery powered. Maybe a small solar panel could be added for charging (?)

• **Is any part of what you discussed at the beginning of the project missing?**

The only thing I found out is the Nicla sensors two function modes. One is every 5 minutes. The other is for measuring very fast for 10 seconds. Not that you do one measure every 10 seconds. You do hundreds of measures per second with a specific interval. When you use the device, you can run all the samples in FFT analysis on a computer to find out frequencies in the sampled data.

• **Can you jointly come up with a couple of things that should be added / changed to make your vision of the end product closer to what they had hoped for?**

Not really. Maybe the 3d printing of enclosure for the sensors. What they could look like. Water resistant for rain, UV resistant. How the finished device could look like and how to mount it when it is used.

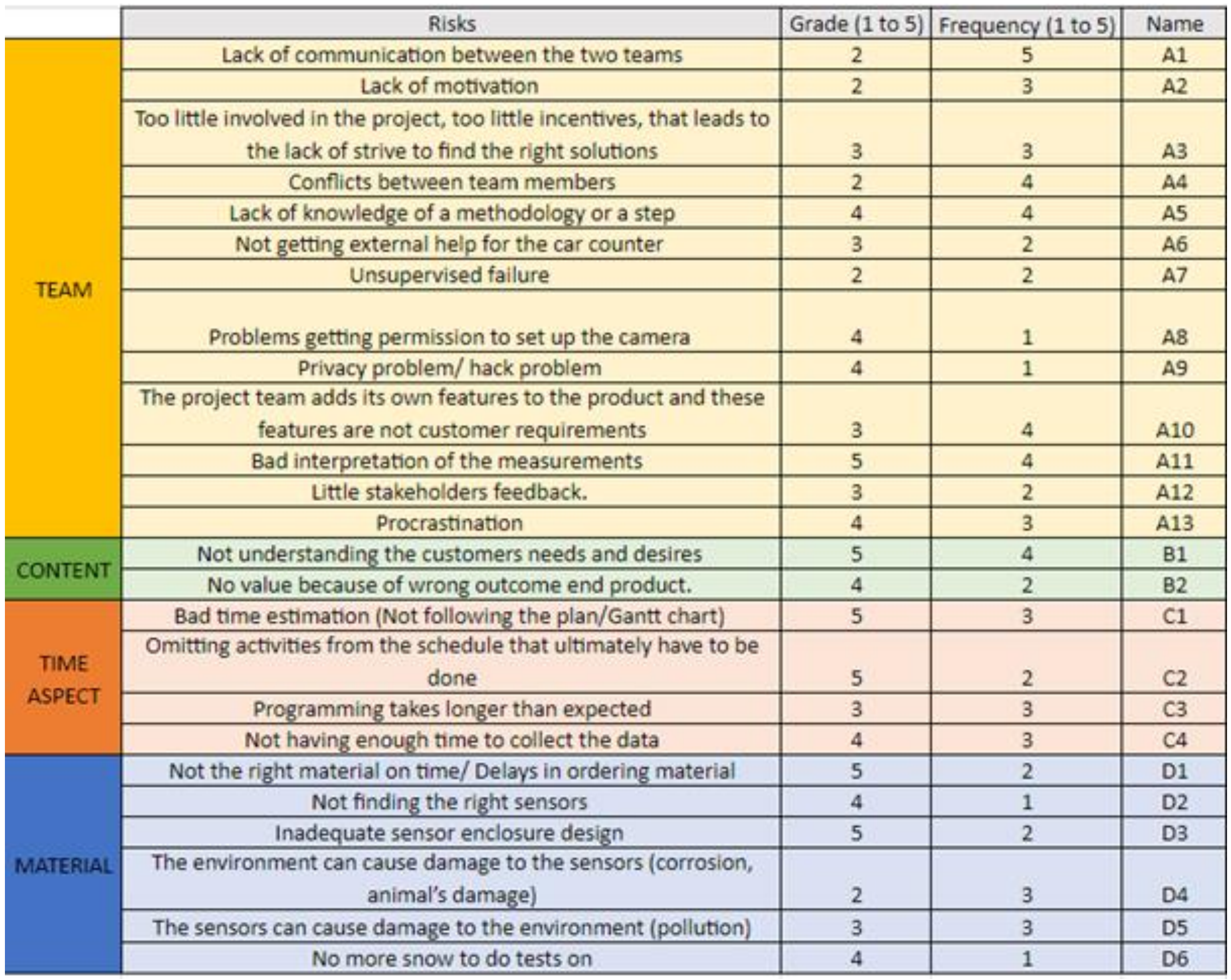
• **After reviewing the prototype, what are the most important reflections stakeholders can do their part – are there things they should contribute (information, data, contacts) that can**

Project Management – help, are there parts of the project that they absolutely want to include and parts that are only nice-to-have?

What could also be added is main components and how they schematically are connected. Component types to use and cost. What way to transfer of data to and from the sensor.

10.6 RISK ASSESMENT AND MANAGEMENT IN PROJECTS

Here you can find the list of risks, rank in a scale 1-5 in two ways. Firstly, according to how much destruction will this risk cause is it actually occurs and secondly, according with the frequency these this risk could occur. Finally, with all this information it is possible to make the table and the risk matrix



	Risks	Grade (1 to 5)	Frequency (1 to 5)	Name
TEAM	Lack of communication between the two teams	2	5	A1
	Lack of motivation	2	3	A2
	Too little involved in the project, too little incentives, that leads to the lack of strive to find the right solutions	3	3	A3
	Conflicts between team members	2	4	A4
	Lack of knowledge of a methodology or a step	4	4	A5
	Not getting external help for the car counter	3	2	A6
	Unsupervised failure	2	2	A7
	Problems getting permission to set up the camera	4	1	A8
	Privacy problem/ hack problem	4	1	A9
	The project team adds its own features to the product and these features are not customer requirements	3	4	A10
	Bad interpretation of the measurements	5	4	A11
	Little stakeholders feedback.	3	2	A12
	Procrastination	4	3	A13
CONTENT	Not understanding the customers needs and desires	5	4	B1
	No value because of wrong outcome end product.	4	2	B2
TIME ASPECT	Bad time estimation (Not following the plan/Gantt chart)	5	3	C1
	Omitting activities from the schedule that ultimately have to be done	5	2	C2
	Programming takes longer than expected	3	3	C3
	Not having enough time to collect the data	4	3	C4
MATERIAL	Not the right material on time/ Delays in ordering material	5	2	D1
	Not finding the right sensors	4	1	D2
	Inadequate sensor enclosure design	5	2	D3
	The environment can cause damage to the sensors (corrosion, animal's damage)	2	3	D4
	The sensors can cause damage to the environment (pollution)	3	3	D5
	No more snow to do tests on	4	1	D6

Now, we will go into more detail on each of these risks, what they are and how we can reduce them.:

General Risk/Team

- **Lack of communication between the two teams**

  We are split into two groups within the team itself, so a lack of communication between the two groups can lead to half of the group not being aware of whether there has been a problem or if they need help. To reduce this risk, we can plan weekly meetings in which we communicate to the other team what progress we have made and where we are struggling.

- **Lack of motivation**

  It is possible that if we see that the project is not going as well as we would like or if we are struggling more than we though, we may lose our motivation. For this not to happen, it is important that if any of the members of our team islosing motimation, he/she should communicate it to the rest of the team so that they can motivate him/her.

- **Too little involved in the project, too little incentives, that leads to the lack of strive to find the right solutions**

  Quite related to the point above, it is possible that if some people do not feel fully involved in the project, this may result in a lose of motivation.

- **Procrastination**

  The fact that there is still some time left until the final delivery may cause us to delay some tasks, and in the end we may not have enough time to do them well. One way to reduce this risk is to have periodic deadlines.

- **Conflicts between team members**

  If two people have different opinions on an issue, this may lead to a conflict between team members. However, this is a problem that can be solved with a good comunication.

- **Lack of knowledge of a methodology or a step necessary for the implementation of the project (car counter)**

  In fact, I am in charge of the programming part of the car counter, and I have never worked before with the technology we are going to use, and this may be a problem in order to finish the project. One solution could be to receive some external help or lessons about it.

- **Not getting external help for the car counter**

  Related to the point above, maybe if we don't get external help from either Hans or someone from the IoT, we may not be able to solve some problems. To reduce this risk, we should show clearly that we need help.

• **Problems getting permission to set up the camera**

We need to place the camera on the outside of the building, so we need permission from the local council. If we have any problems, it would delay the work a lot. The only way to reduce this risk is to ask for permission as soon as possible, in order to have enough time left.

• **Little stakeholders feedback.**

One risk can be having too little feedback from our stakeholders and not knowing clearly if what we are doing is right or if we are going in the wrong direction. One solution may be to try to have as many meetings as possible and to keep the communication flowing.

• **The project team adds its own features to the product and these features are not customer requirements**

It may happen that our team add some things that in our opinion add value to the final result, but nevertheless we move away from what the client really wants. Here too, good communication is the only solution.

• **Bad interpretation of the measurements**

It can also happen that we are not able to interpret the values given by the sensors correctly, for example because we do not know in which units is the sensor measuring. In order to reduce this effect, we should carry out a good technical research.

Content

• **Not understanding the customers needs and desires**

There may be problems of misunderstanding with the customer, for example because of the language barrier. To reduce these problems we should periodically ask the customer if he thinks we are going in the right direction.

• **No value because of wrong outcome end product.**

It is also possible that the product we will develop does not work properly due to small problems that we have not realised during the course os the project. For this not to happen, we should do periodic test.

Time aspect

- **Bad time estimation (Not following the plan/Gantt chart)**

Bad planning can cause us to spend too much time doing some activities and end up running out of time to do others. To reduce this risk it is important to make a good chart.

- **Omitting activities from the schedule that ultimately have to be done**

Related to the above, some activities may arise during the course of the project that we did not take into account at the beginning of it. It is difficult to reduce this risk, but good initial research can help.

- **Programming takes longer than expected**

This is a typical risk in all projects that involve programming, as many unexpected problems or difficulties can arise. There is no way to reduce them before they happen, the only way to make them affect less is to ask for help from people who know more about it.

- **Not having enough time to collect data**

If we finish the programming too late, we may not be able to collect enough data to do a study. The only solution is to go as fast as possible with the programming part.

Material

- **Not the right material on time/ Delays in ordering material**

It may happen that we want a particular material but it takes a long time to arrive, and in the end we won't be able to get it. In fact this has already happened to us. To reduce its effect, it is always a good idea to have several options.

- **Not finding the right sensors**

This refers primarily to the water and snow sensors. It may happen that after testing several sensors, we have not been able to find sensors that really work well. The only option to reduce this risk is to do good initial research.

- **Inadequate sensor enclosure design**

It is important that the enclosure adequately protects the sensors. If we choose the wrong material or shape, all the previous work will be lost. One way to reduce this risk may be to conduct several test before making a choice.

- **The environment can cause damage to the sensors (corrosion)**

External causes can also affect our project, related to the point above, in order to reduce this risk it will be important to design a good enclosure.

- **The sensors can cause damage to the environment (pollution, animal's damage)**

Our project can also affect the environment, for example the seawater sensor can affect the ecosystem. It will be important to make a good study of what can happen and how we can avoid it.

- **No more snow to do the tests**

It may happen that the snow has completely de-iced before having the right sensor. One solution is to start earlier with the snow sensor.

From this we have made the Risk Assessment Graph, on the one hand the risks, on the other hand the grades and frequencies, that describe the probability that the risk will occur. Besides that, it also describes how much destruction the risk will cause, and how much impact it has on the global task. The risk rating is a way to describe the highest chance of happening. So, the ultimate goal is that the risk eventually will disappear, that the risk factors will reduce.

		Impact				
		Very low	Low	Medium	High	Very high
Probability	Very likely	🟨	🟧 A1	🟥	🟥	🟥
	Likely	🟩	🟨 A4	🟧 A10	🟥 A5	🟥 A11, B1
	Possible	🟩	🟨 A2, D4	🟨 A3, C3, D5	🟧 A13, C4	🟥 C1
	Unlikely	🟩	🟩 A7	🟨 A6, A12	🟨 B2	🟧 C2, D1, D3
	Rare	🟩	🟩	🟩	🟩 A8, A9, D2, D6	🟨

11 BIBLIOGRAPHY

General sources

Ericsson. (2020). Technology Trends 2020.
[online] Available at:
https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/technology-trends-2020#trend1
This article gave the team a more in depth sight of what Internet of Things really is about. From this point we had more of an idea about what IoT is all about and had a direction to go to.

Water and snow sensors

Types of level sensors:

Proces Sensing Technology. (s.d) The 7 Main Types of Level Sensing Methods – How do they differ?.
[online] Available at:
https://sstsensing.com/7-main-types-of-level-sensors/
On this page we found out more about all the kinds of level sensing methods. We learned how they work and how they differ. From this we could choose which kind of sensor is the best type for our project.

A Finnish project about sea level:

Finnish Meteorological institute. (s.d.) Sea level Measurements.
[online] Available at:
https://en.ilmatieteenlaitos.fi/sea-level?sealevel_station=-10022815&sealevel_graph=short&sealevel_mode=mwn2000.
We used this site to find out more about the sea level in Vaasa. This so we can get some background and expectations about the behavior of the sea and in what range we have to measure.

A similar project of sea level measuring in the USA:

Smart Sea Level Sensors. (s.d.) Smart Sea Sensors.
[online] Available at:
https://www.sealevelsensors.org/

A project that also works with smart sea level sensors. Our team tried to get some information and ideas from this site on how to begin designing and programming the sea level sensor.

Some info about pressure sensors:

Viatran, A Dynisco Company. (s.d.) Liquid Level Measurement.
[online] Available at:
https://www.viatran.com/static/media/uploads/technical_articles_notes/liquid_level_measurement_tech_note.pdf
A document that explains how the pressure sensor works, including all mechanical and physical details. The team knew how the pressure sensor worked and how we should interact with it thanks to this document.

SR04 Ultrasonic sensor Manual:

De Bakker. B. 2021. Waterproof JSN-SR04T Ultrasonic Distance Sensor with Arduino. [online] Available at:
https://saliterman.umn.edu/sites/saliterman.dl.umn.edu/files/general/arduino_waterproof_jsn-sr04t_ultrasonic_sensor_2_examples.pdf
This is an article about an ultrasonic distance sensor. This sensor could potentially be used as a water level sensor. We have been reading this article to get some more information about this concept and understand how it works.

Honeywell ABPDJJT005PDSA3 datasheet:

ABP SERIES. 2020. Basic Board Mount Pressure Sensors.
[online] Available at:
https://www.mouser.mx/datasheet/2/187/honeywell_sensing_basic_board_mount_pressure_abp_s-1662208.pdf.
A more detailed document from ABP series about their catalogue of pressure sensors
Where they talk about the specifications and how they function.

Arduino Tutorial–Securely Connecting an Arduino NB 1500 to Azure IoT Hub. [online] Available at:
https://create.arduino.cc/projecthub/Arduino_Genuino/securely-connecting-an-arduino-nb-1500-to-azure-iot-hub-af6470
A tutorial about how to connect the Arduino MKR NB 1500 board to Azure IoT Hub or another MQTT broker.

Arduino Forum–MKR NB High Power Consumption. [online] Available at:
https://forum.arduino.cc/t/mkr-nb1500-high-power-consumption/597574
In this forum we found the solution to the high power consumption of the board.

<u>Housing design – material research:</u>

MARKERSHOP 3D
[online] Available at:
https://www.makershop.fr/content/43-guide-achat-filament-resine-impression3d#:~:text=Le%20filament%203D%20PLA%20(Polylactic,est%20sa%20simplicit%C3%A9%20d'utilisation

FILIMPRIMANTE3D
[online] Available at:
https://www.filimprimante3d.fr/content/47-pla-abs-petg-choisir-son-materiau-pour-imprimante-3d-en-fonction-de-l-application

<u>EXTRA</u>

AXONIZE (s.d.) Classifying Different IoT Sensors & Their Uses.
[online] Available at:
https://axonize.com/blog/iot-platform/classifying-different-iot-sensors/
Basic information about sensors and what kind of sensors there are in the world.
Out of this document we learned a lot more about the working of sensors like the temperature sensor, the CO2 sensor, the accelerometer and the pressure sensor.

Vibration sensors

Reality AI – Vibration and Accelerometer Demo
[online] Available at:
https://www.youtube.com/watch?v=36WJVforG_k
A video that shows a really good example of the vibration sensor we need to develop. In this video they show you how the sensor can measure imbalances of certain moving object. This gave us a first view of what the idea behind this sensor is.

Hands-On with the Arduino Pro Nicla Sense ME!
[online] Available at:
https://www.youtube.com/watch?v=yVL7vXpvEog

In this video they are working with the exact sensor we also use in our vibration sensor.
This video gave us a lot of inspiration about the case development. It also gave a lot of basic information about the sensor that was really useful.

Store Arduino – Nicla Sense ME product
[online] Available at:
https://store.arduino.cc/products/nicla-sense-me
This is the store page of Arduino. Here you can buy the Nicla Sense ME. You can also find a lot of information and video’s that give good explanation about how to use the Nicla Sense ME and what it can do.

Store Arduino – Nicla Sense ME hardware
[online] Available at:
https://docs.arduino.cc/hardware/nicla-sense-me
This is the hardware page of the Nicla Sense ME. Here you can find information about which components and sensors the Nicla Sense ME consists of. You can find all the technical aspects of the Nicla Sense ME and the compatibilities it has.

BOSCH SENSORTEC – Arduino Nicla Sense ME
[online] Available at:
https://www.bosch-sensortec.com/software-tools/tools/arduino-nicla-sense-me/
This website gives us a in depth view of all the sensors the Nicla Sense ME has and gives detailed information about all of them. Technical date about the Nicla Sense ME can also be found on this website.

Arduino Nicla Sense ME|Tutorial 8|Sensor Data to Arduino IoT Cloud
[online] Available at:
https://www.youtube.com/watch?v=GxDCnnC6DZ4
This video explains us really detailed how we can transmit the data we measured with the Nicla Sense ME to the Arduino IoT Cloud. With this video we could successfully make a connection with the Nicla Sense ME and receive the measured data up to date.

Docs Arduino – Arduino Nicla Sense ME as a MKR Shield
[online] Available at:
https://docs.arduino.cc/tutorials/nicla-sense-me/use-as-mkr-shield

In this document it is explained how to connect the Nicla Sense ME with the MKR WiFi 1010. They explain how to solder the headers on the Nicla Sense ME and after how to successfully connect it with the MKR WiFi 1010.

Docs Arduino- Arduino Cloud IoT Cheat Sheet
[online] Available at:
https://docs.arduino.cc/cloud/iot-cloud/tutorials/technical-reference
In this document they explain us how to set up the Arduino Cloud IoT. There is also an overview with all the compatible boards, the API, configuration, Things, variables and dashboards.

Car counter

Libelium.com – Traffic and Road Conditions Monitoring in Malaga
[online] Available at:
https://www.libelium.com/libeliumworld/success-stories/traffic-and-road-conditions-monitoring-in-malaga/
This article is about traffic monitoring in Malaga. It gave us an idea about what traffic monitoring is and in what ways traffic monitoring can be done.

Centeye.com – Internet Of Things (IoT) Car traffic counter camera using an ArduEye and Xively
[online] Available at:
https://www.centeye.com/2013/02/internet-of-things-iot-car-traffic-counter-camera-using-an-ardueye-and-xively/
This article is about a similar project with car traffic counting. This article gave us more information of Internet of Things. The article also contains good examples on how to detect cars and how to save the data.

Diglab.technion.ac.il – IoT sensor for car count
[online] Available at:
https://diglab.technion.ac.il/projects/iot-sensor-for-car-count/
A short explanation of a similar project with car counting. This tiny article gave us a better idea about the car counting concept and how it exactly works.

Windmill.co.uk – Data acquisition – Monitor 211
[online] Available at:
https://www.windmill.co.uk/monitor211.html
A webpage with information about all the methods to perform traffic counting. With this information we could decide which method is the most useful for our project.

Hackaday.io – Traffic Counter – Road Tube
[online] Available at:
https://hackaday.io/project/4567-traffic-counter-road-tube
An in depth article and video about the Road tube traffic counting method. With this article we could understand this method for deciding if this is a good method for our project. This article was also useful for explaining how this method works on the mid-term presentation.

Create.arduino.cc – Arduino Car Counter
[online] Available at:
https://create.arduino.cc/projecthub/Aetos999/arduino-car-counter-4117e6
An article from Arduino on how to do car counting with their components by using the road tube concept. They give a detailed explanation on which hardware components you need, how to connect the electronics to each other and how to create the perfect code and upload it.

Civil.iitb.ac.in – Intrusive Technologies – Lecture notes in Transportation Systems Engineering
[online] Available at:
https://www.civil.iitb.ac.in/tvm/nptel/525_AutoLoop/web/web.html#x1-20001
An in depth explanation of every car counting method. This document goes really deep in detail. For the Dual-loop Detectors they give us the formulas to calculate a cars speed and give examples with it on how to use these formulas.

Mdpi.com – A Review on Vehicle Classification and Potential Use of Smart Vehicle-Assisted techniques
[online] Available at:
https://www.mdpi.com/1424-8220/20/11/3274/htm
This article talks about the pro's and cons of all the car counting methods. It also gives us a conclusion about the whole thinking behind the car counting methods and a view on the future of car counting.

Extra sources

Plm.automation.siemens.com – Siemens selects Tangent Works to democratize IoT data analytics for MindSphere
[online] Available at:
Siemens selects Tangent Works to democratize IoT data analytics for MindSphere | Siemens Software
Talks about the use of IoT, its benefits and the future of it.