INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Innovative Project

Smart Connected Car

BERTA Pauline - pauline.berta@yahoo.fr
BERRADA El Ghali – ghaliinsa@gmail.com
CONCEICAO NUNES Joao -nunesjoao98@outlook.fr
POTIERS Léo – leo.potiers@hotmail.fr
SKIKER Hicham – skikerhicham@gmail.com

January 2021

MONTEIL Thierry
SCANLAN Paul

Abstract

Smart connected cars are becoming a common thing in our lives with the evolution of autonomous driving technology. Cars are becoming smarter and smarter with the progress made in technology and in legislation.

The objective of our project is to develop a system that provides smart features to an old car, helping the owner keep his car longer. We aim to use external sensors, cameras, and motherboards to develop an add-on solution providing some security and comfort smart functionalities.

In response to our goal, we are going to create a physical system using a 3D-printed box containing a Jetson Nano board, air quality sensors and an ESP32 board to communicate with a mobile application.

TABLE OF CONTENTS

I - Introduction

The ISS Innovative Project is the opportunity to work on exciting and complex problems with a group of 6 people, where each one has a different background and different skills. In our case we had a very diverse group of students combining physics and electronic design along with software development and business techniques.

This project was done alongside with INSAs infrastructure and aims to become a part of a bigger project with companies working on the automobile industry, in the future. The goal, as you will learn, is to transform *old* and *basic* cars into smart and connected ones. We have purposely designed our system to help people keep their car and improve their comfort and security without having to buy a new one and multiply their expenses.

In this report you will find every information about our project and our solution as well as a detailed outline of each step in the conception and development of our product.

Our solution uses two distinct image processing algorithms returning the type of traffic sign and a line assist indication. The same system also is equipped with two sensors, a temperature/humidity sensor, and a gas sensor analyzing the environment inside the case. Every information is displayed to the driver through a mobile application connected to the system with a BLE connection.

II. Objective & Specifications

II.1. Context

Today, when the environmental and energy transitions have never before been such a priority for all of us and for the next generations, it is our responsibility to reconsider the way we consume, and benefit from the growth of technologies to facilitate this transition.

When we began to think about cars and reducing the CO2 impact, we automatically think of fuel, and we forget about the conception and the creation of a car. This is what we call the "grey energy". In fact, manufacturing and releasing a new car corresponds to 500 kg of released CO2, which represents one quarter of the pollutant emissions of a car's lifetime.

Thus, making a car costs so much energy, that the most ecological car is the one we already have. On the other hand, budget wise, this solution is also the best option.

Now, we assume that we try to last the life of our car to its maximum, what will be our next issue? Obviously, make sure that our car stays safe for us and our passengers, while interacting with the new generation of autonomous cars. How could we benefit from the new technologies of automobiles, and enhance the comfort of driving, being sure that our vehicle is smart and secure?

Therefore, we decided to design an ergonomic and adaptable kit to transform old cars into smart connected cars. The kit will be composed of a variety of sensors deployed in the car and a software platform such as a phone to manage them and to visualize basic information about the car.

II.2. Specifications

Because the Smart Car kit project was not directly linked with a company it was not provided with precise specifications. It was our responsibility to bring out ideas and to design the kit from scratch. We knew that the kit was designed for old cars, but this was very vast, so we started by setting two conditions for the user of the kit. We ask the user to have a smartphone to run the application that will be the interface with the user.

In order to plug the kit to a USB port, we also chose to target the low range cars that are not obsolete but not well equipped. The public we are targeting are students and their first car, or any person that wants to update their vehicle without having to buy a new one. At this point, we started to ask ourselves what were the main features that were missing in that category of car.

The first priority we fixed was security. To enhance security, we have been inspired by autonomous cars. We thought about an intelligent computer vision system that would be able to recognize the traffic signs or to detect lines. Moreover, this would enhance driving comfort for the driver and its passengers.

The second priority we fixed was the price of the kit. Designed to be deployed on low budget cars, the price of the kit needs to be consistent. We will take this into account in the choice of types and models of sensors.

The next priority is to develop the ecological sensitivity of the car and of the user. By giving them access to fuel consumption information or to a driving style evaluation gauge. Because the kit box will be installed on the front desk of the car under the windshield, it must be small to be adaptable to every car.

Finally, because we wanted to get some out of the box perspectives, we conducted a survey. This market study has allowed us to refine our research and work for our customers.

II.3. Team Organization

Our team is composed of five students, each with their own specific mission, according to their previous experiences and skills:

Berrada El Ghali : Developed a market study to understand our client and "what he needs", his profile, his opinion on the project, and his budget. He prepared an analysis to conclude the real customer needs to translate these needs into an action plan in our project. He also worked on designing the case of our product.

Conceicao Nunes Joao : Developed a mobile application for the driver and the BLE link between the motherboard and the smartphone. The driver would be able to benefit from many features via his phone and thanks to the motherboard (Jetson Nano): the data sent by the image processing algorithms (recognition of traffic signs) and the sensors, features such as GPS, location, consumption management etc.

Berta Pauline : Developed a line assistant algorithm, this line algorithm works via the motherboard (jetson nano), which will be measured in the space between the car and the white lines that surround it. This algorithm will give more control and security to the driver.

Potiers Léo : Developed a code for the processional image part (traffic sign), this code will run on the motherboard (jetson nano). This feature will show the driver traffic signs in his car, in other words alerting the driver to pay attention to the road example: speed limit, turn etc.

Skiker Hicham : Administrative tasks, and also developed a code “Driving Environment Analysis” for the gas, temperature, and humidity sensors, via an Arduino uno card then I used an ESP32 Our code was script by android mobile application via BLE. Then deployed a solution about All the sensor’s data was record in a microSD card module to be saved in txt file (datalog) .

Amour Redouane : A physics department student which started developing the line-assist code with Pauline, but had to leave the project due to medical problems.

II.4. Project Timeline

For the time organization we used some of the tools given by the project management course, like the Gantts diagram. The concept behind the Gantts diagram is to have an overview of the project and the time each main axis will take to develop.

So, in our Gantts diagram we have the red areas that are basically the guideline of our project and the blue areas that are tasks that can be done at the same time.
See below the details of the tasks of our project for five months.

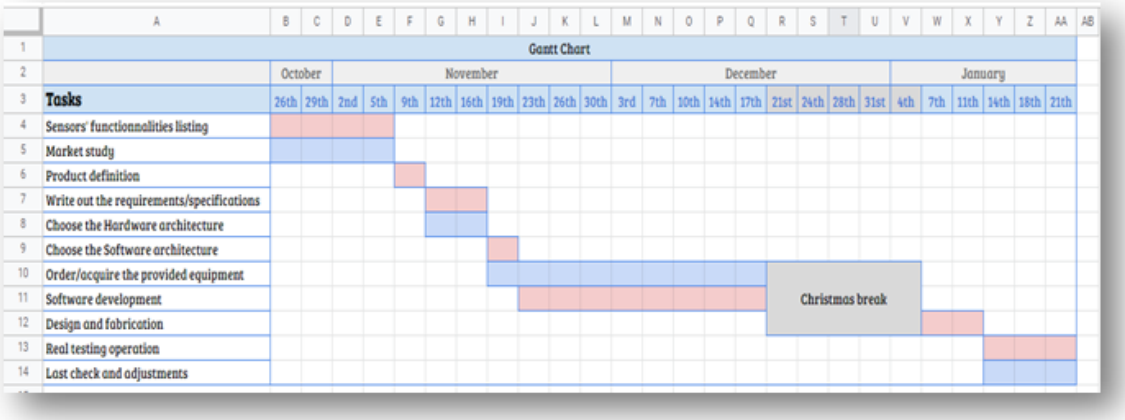


Figure 1: Gants Diagram for our project

Our project takes place between October 2020 and January 2021. We organized our work in four major phases during those five months as shown in the diagram below:

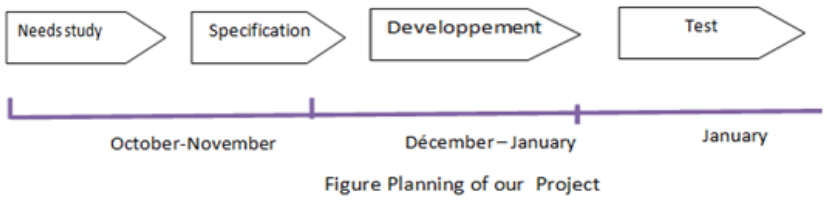


Figure Planning of our Project

Figure 2: Basic timeline for our project

N°1: October – November

Needs study and specification

During this phase, our team must focus on the following steps. This start-up step is a very important step in the project management component. Indeed, the planning for our team is a critical phase especially based on the management of priorities.
I will mention a few tasks that our team has to focus on as a priority for example:

- Understanding the project in terms of mission and objectives.
- Conducting market research to list customer needs.
- Developing a schedule for our project.

N°2: December - January

Development

- Setting up a mobile application to interface all the functionalities integrated into our project (sensors, road signs, GPS, etc.).
- Implementation of a line assistance algorithm.
- Setting up a code for the sensors to send them to our mobile application.

N°3: January

Deployment test

For the traffic sign part, this part will be recommended to test via the Jetson nano board, and the same for the line assist. The application will then be connected via the BLE communication protocol because our client is very close to the mobile phone. Finally, we will have to integrate the code related to the sensors aiming to save data via a micro-SD recorder in a file (.Txt). This will allow our application to capture this data and display it in the mobile application.

III - Market Research

In each project and especially during the development of a product or service proposition, a market study is necessary. It helps to understand the consumer, his profile, the main problems he faces every day, his opinion about the project, and his budget. Over all, a market study allows us to :

- Understand the client's needs.
- Design the most desirable product, with the most important features.
- Anticipate our costs considering the client's budget.

To realize our market study, we decided to put out a survey containing selected and relevant questions:

- Profile: Age, Profession, ...
- The cars age.
- Reason why to change a car.
- Evaluate the importance of each proposed feature.
- The budget.
- 2 or 3 technical questions required to design our product.

Thus, our market study gave us fundamental information:

- Average age of the respondent's car : 10 years. It represents a sizable age for a car, as well as options and features (smart connected cars) are limited and not common for 10-years cars or more. This confirms our initial hypothesis: the vehicle fleet is poorly connected.
- Reasons why to change a car:

Figure 3: Survey results, INSA students/teachers

1. For more comfort.
2. To buy an electric or hybrid car.
3. Excessive consumption.
4. For more safety.

In our kit, we must consider options that could improve the consumer's comfort, safety, and consumption.

- Most requested features: Detection of collision and Quality of air index.
- Average price: 330€. This price gives us a threshold that is important to respect otherwise we will not have any clients. This means that the material and development costs must be lower.

All that information allows us to elaborate the specifications of our product. However, we discovered that all the sensors and elements must be inside the car in order to be supplied. This forces us to remove the detection of collision features as it works with a Lidar that necessarily needs to be outside the car. So, to stay in line with the consumer's needs for more safety, we decided to replace this feature with two new ones: road signs recognition and line detection. Also, we decided to work on other comfort options such as temperature, humidity, and power supply. Finally, we decided to develop a functionality related to the ecological impact which seems to be an important aspect for the consumers.

IV - Driving Environment Analysis

IV.1. Context & Objectives

Throughout our market study, we realized that comfort and safety are one of the most important aspects for a driver, and what motivates him to change his car.
We decided then to work on features that would improve the driver's comfort and safety within the car: Index quality of the air inside and outside the car, temperature, and humidity measurements. These features will be based on sensors, and their data will be used for two different purposes.

The first and most important is to send this information to the mobile application and help the driver of the car understand the environment around him. The second purpose of the data is to be saved on a memory card and used on future applications. Because we were uncertain of the integration part and how everything would work together, we decided to explore this second solution to ensure that the sensors would be useful at the end.

IV.2. The hardware chosen for our system

At the start of the project, we started testing and developing everything with a Arduino Uno board, because it was the fastest way to test the sensors and understand if the functionalities we wanted to develop with these sensors were going to work properly or not. We started testing with a DHT11 sensor for measuring temperature and humidity, and with a Groove Multichannel sensor for the gas detection.

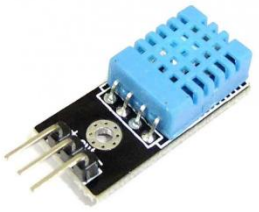


Figure 4: DHT11, temperature and humidity sensor

The DHT11 sensor is a single wire sensor, measuring temperature from 0 to 50 °C with an accuracy of +- 2°C, and measuring relative humidity from 20 to 90% with an accuracy of +- 5%. The sensor is powered from 3 to 5.5V DC, so it allowed us to connect it to the Arduino Uno board, the Jetson Nano, or as we will see later the ESP32 board.

The Groove multichannel sensor is a I2C gas sensor with programmable addresses and low power management. This sensor can be powered between 3.1 to 5.5V DC, which means we could use it with the Arduino board but also with the Jetson and ESP32 boards. This sensor detects a wide range gases, such as:

- Carbon monoxyde CO 1 – 1000ppm
- Nitrogen dioxyde NO2 0.05 – 10ppm
- Ethanol C2H6OH 10 – 500ppm
- Hydrogen H2 1 – 1000ppm
- Ammonia NH3 1 – 500ppm
- Methane CH4 >1000ppm
- Propane C3H8 >1000ppm
- Isobutane C4H10 >1000ppm

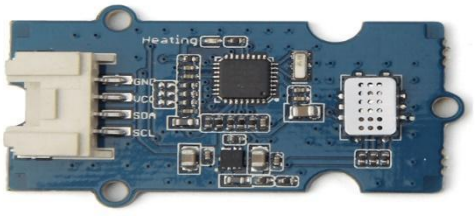


Figure 5: Groove multi-channel gas sensor

IV.3. Solutions Deployed

IV.3.1. Saving data on a SD card

As explained before we developed two different solutions using the data sent by the two sensors. The first one to be developed was the save feature. This feature was developed using an Arduino Uno board and was later adapted to a ESP32 board.

The objective behind the save feature is to provide the user of the system a safe with all the data captured inside the car. The mobile application being just a displaying tool, if the driver wants to check older records, he can by using the SD-card available inside our system.

For this feature we used a SD-card adapter, initially on an Arduino Uno, and finally with a ESP32 board. As we can see in the following picture the connections to this adapter can be used both with Arduino and ESP32 boards.

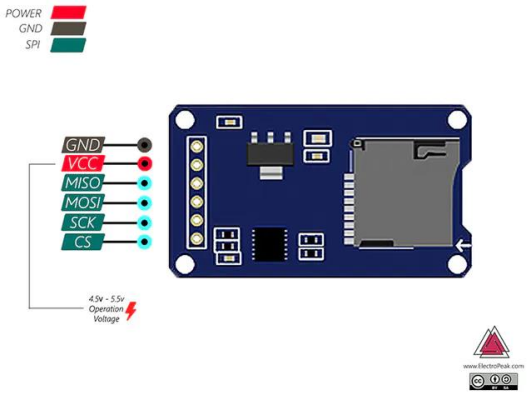


Figure 6: Micro SD-adapter for Arduino boards

We have tested this feature with the DHT11 sensor and a MQ2 gas sensor, and the code is available on the GitHub repository. This functionality will be extremely useful in the future if the next teams want to add an internal battery to the system and keep track of everything that happens inside the car. With the SD-card module and saving functionality the system could easily evolve to a permanent system, being ON even when the car is not being used.

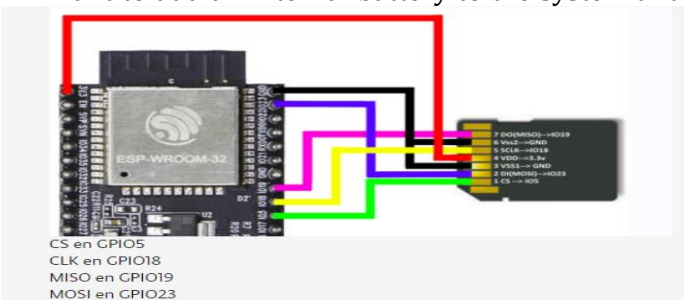


Figure 7: ESP32 connection with SD card

IV.3.2. Mobile Application Display

The main purpose of the sensors is to show the driver the temperature, humidity, and air quality index inside the car, for its own comfort and security. To extract the sensor data, we developed an algorithm that managed the heating periods of the gas sensor and the delays with the DHT11 sensor. For example, the DHT11 sensor being a single wire sensor could not send both humidity and temperature data, so we had to introduce some delays for it to properly work when the user needed both temperature and humidity data. For the gas sensor we also had to manage the heating period, because to measure the gas rates present in the air, the sensor must heat its chamber.

Every sensor information is then sent to the mobile application through the BLE module, and it is explained on the integration part of this report. All this data is used to calculate the air quality

index and to understand the current situation inside the car regarding gases, temperature, and humidity.

Air quality index and analysis

The Air Quality Index[1], or AQI, is an environmental tool. Its purpose is to communicate the air quality daily. It is color coded and each category provides information on the air quality in your place and it is used to prevent pollution damage.
The air quality index is based on five major pollutants, which are dangerous for the health of citizens and for the environment: suspended particles (PM2, 5 and PM10), ground-level ozone (O3), nitrogen dioxide (NO2) and sulfur dioxide (SO2).

Principes de base de l'AQI pour la pollution par l'ozone et les particules

Couleur AQI quotidienne	Niveaux de préoccupation	Valeurs de l'indice	Description de la qualité de l'air
vert	Bien	0 à 50	La qualité de l'air est satisfaisante et la pollution atmosphérique pose peu ou pas de risque.
Jaune	Modérer	51 à 100	La qualité de l'air est acceptable. Cependant, il peut y avoir un risque pour certaines personnes, en particulier celles qui sont inhabituellement sensibles à la pollution atmosphérique.
Orange	Malsain pour les groupes sensibles	101 à 150	Les membres de groupes sensibles peuvent subir des effets sur la santé. Le grand public est moins susceptible d'être affecté.
rouge	Mauvais pour la santé	151 à 200	Certains membres du grand public peuvent subir des effets sur la santé; les membres de groupes sensibles peuvent subir des effets plus graves sur la santé.
Violet	Très malsain	201 à 300	Alerte santé: le risque d'effets sur la santé est augmenté pour tous.
Bordeaux	Dangereux	301 et plus	Avertissement sanitaire des conditions d'urgence: tout le monde est plus susceptible d'être affecté.

Figure 8: Air Quality Index table

This table shows how we can judge whether our air quality presents a risk or not. For example, the green color represents no risk, while the purple and burgundy represent dangerous levels of bad gases.
In our mobile application, we will try to calculate the Air quality index based on the information we have from our sensors, and we will also display for the user the AQI outside the vehicle, using the GPS coordinates from the smartphone.

In our case, our sensors can detect some dangerous gases that could be present inside the car. For example, NO2, nitrogen dioxide, is emitted during combustions and its main sources are vehicles (nearly 60%) and combustion installations (thermal power stations, heaters, etc.). It is a non-flammable gas and even if it is not considered to be toxic, it oxidizes easily with air which then becomes very toxic. Nitrogen dioxide along with nitric oxide are two important air pollutants, referred to as NOx.

Another example is CO, carbon monoxide. It is a colorless, odorless, flammable gas that is very toxic even at low doses. It quickly binds to hemoglobin causing decreased cellular respiration and making the driver lose its conscience while driving. Finally, another dangerous gas, NH_3, ammonia, is also a colorless, poisonous gas which can cause health problems such as cough, dyspnea, respiratory distress, or pulmonary edema.

[1] « AQI Basics | AirNow.Gov ».

Thus, two information will be displayed for the driver: air quality index inside the car thanks to sensors, and air quality index outside the car thanks to data given by the GPS.

IV.4. Recommended Temperature and Humidity rates

The Humidity rate inside a home or a close environment as a car, is recommended to be between 40 and 60%, when the temperature oscillates between 23 ° and 25 °C.

If the temperature is too high the driver can fall asleep or it can cause disturbances to the level of concentration. It also can change children's and animals behavior inside the car.

In the case of an unusual humidity rate, the environment inside the car can be compromised. A high humidity rate will promote the presence and development of mites and particles of all kinds which are the cause of many respiratory diseases, while a low humidity rate promotes the release of dust into the air and can cause risks of asthma. In addition, the absence of humidity favors allergies, and rhinitis or bronchitis which are the result of viral or bacterial infections. Although it is most often linked to problems of overly high humidity, asthma is also provoked by overly dry room air.

V. Computer Vision

V.1. Traffic Sign Detection Algorithm

V.1.1. Interest of such feature

The detection of traffic signs [2] will be deployed to make the drive smoother and safer. The neural network can recognize upcoming traffic signs faster and with more reliability than a human that could be tired or distracted. The detected information will be rapidly shared to the driver so that he can anticipate road dangers or regulations and so adopt the appropriate driving style. Moreover, we can imagine that once deployed on enough cars, this tool will help to reduce traffic congestion and to significantly reduce the crash risk.

V.1.2. Choice of Language and Framework

To implant this function in our kit, we will use python 3. To display images, we will use OpenCV, which is a software specialized in real-time image processing. To produce a neural network and to make it learn to detect objects, we also need a framework. The framework is a set of functions and libraries that permits the learning process that is essential to implement recognition features based on artificial intelligence. We chose to use TensorFlow because it has great results, it has a large OS compatibility, and it is perfectly adapted to do visual recognition. Moreover, TensorFlow enables accelerators like XLA that will be more efficient than CPU for some specific computation. Due to these advantages, TensorFlow is mainly used by the community, and so codes or help can be found more easily on the web which is not negligible.

V.1.3. Presentation of the work of Álvaro Arcos

Alvaro Arcos is a doctor in computer science at the Seville University in Spain. He is working in deep learning and software development. In August 2018, assisted by Juan A. Álvarez-García and Luis M. Soria-Morillo, he published an interesting work about traffic sign detection using Python and TensorFlow. In this part we will present the work he did and implement his open-source code to test traffic sign detection.

Presentation of the training dataset

When we think about machine learning, we think about big data. Before expecting the neural network to learn how to recognize features on an image, it must be explained what the characteristic to be recognized looks like.
To do this, we must gather a huge number of images, which must be as representative as possible of the diversity of the characteristic to be recognized. In our use case, it would be a matter of representing each recognized traffic sign from different angles, at different distances or in different weather conditions. Still in our case, it would also be relevant to group panels from various countries, as their geometry, size or style may vary. The purpose of this diversity in the dataset is to make the recognition tool as robust, adaptable, and reliable as possible.

[2] Shustanov et Yakimov, « CNN Design for Real-Time Traffic Sign Recognition ».

For the reasons mentioned above, the choice of the dataset is crucial. In addition, the training phase has a significant cost in time and computing power. It would be expensive to realize after training, that the dataset was not exhaustive.
The work on which we rely on is based on the German Traffic Sign Detection Benchmark (GTSDB). This dataset is composed of 900 images divided in 600 training images and 300 evaluation images. Images display up to 6 traffic signs that may appear in every perspective and under every lighting condition. The size of the images varies from 16x16 to 128x128 and they are stored in ppm format. In the dataset, each image is associated with a .CSV file, which contains the image labeling data. This file contains the coordinates of each feature to be recognized. It was generated after framing each road sign on each image, by entering the category to which the feature belongs. This phase is very long and tedious but has a crucial impact on the accuracy of future detection. Each image and its associated labeling file will constitute a training model for the network.

Comparison of different models

In his work[3], Alvaro Arcos has trained 7 different networks to recognize three categories of traffic signs. The categories to be recognized are "Danger", "Mandatory" or "Prohibitory".
The following figure shows the set of recognized signs by the trained models as well as their classification.

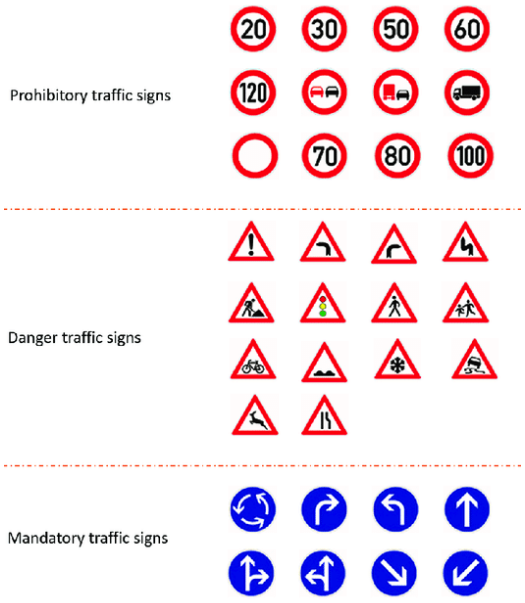


Figure 9: Traffic signs in the GTSDB dataset and their classification

[3] Arcos-García, Alvarez-Garcia, et Soria Morillo, « Evaluation of Deep Neural Networks for traffic sign detection systems ».

The interest of training different models with the same dataset, and for the same purpose, is to be able to compare them to choose which one is best suited for our use, in terms of reliability, energy consumption, and speed. The following figure compares the characteristics measured for each of the trained networks:

Model	mAP (%)	Parameters	Memory (mb)	Total execution time (ms)	Acceleration execution time (ms)	CPU execution time (ms)
Faster R-CNN Resnet 50	91.52	43337242	5256.454	104.036	75.933	28.102
Faster R-CNN Resnet 101	95.08	62381593	6134.705	123.272	90.337	32.935
Faster R-CNN Inception V2	90.62	12891249	2175.206	58.533	38.768	19.765
Faster R-CNN Inception Resnet V2	95.77	59412281	18250.446	442.220	366.158	76062
R-FCN Resnet 101	95.15	64594585	3509.751	85.452	52.403	33.048
SSD Mobilenet	61.64	5572809	94.696	15.145	4.0212	11.123
SSD Inception V2	66.10	13474849	284.512	23.744	9.393	14.350
YOLO V2	78.83	50588958	1318.108	21.481	18.139	3.341

Figure 10: Comparison table of various trained models for traffic sign detection

The main criteria for our study are the following: mAP, memory, and the total execution time.
The mean average precision (mAP) is calculated during the evaluation phase of the trained model, from the difference between the ground truth bounding box and the detected bounding box. This

criterion is then calculated from the average of this measurement on each image and for each category. The higher the mAP, the more accurate the model is in its detections.

The memory is the size of the model after training. Since we want to deploy this model on an embedded system, it is important to consider the memory space it occupies in order to minimize it. By taking up as little memory space as possible, a program will leave space for others and be executed more easily.

Finally, the total execution time which is the sum of the CPU execution time and the acceleration execution time. The CPU execution time is the time between the start and the end of execution of a given program. This time accounts for the time the CPU is computing the given program, including operating system routines executed on the program's behalf, and it does not include the time waiting for I/O and running other programs. Hardware acceleration is the use of computer hardware specially made to perform some functions more efficiently than is possible in software running on a general-purpose central processing unit (CPU).

By comparing the results of each model trained for these three criteria, it appears that the RFCN-Resnet 101 model would be the most efficient for our use. It has one of the best reliabilities (95.15%), a middle-range memory (3509,751 mb) and a satisfactory total execution time (85,552 ms).

V.1.4. Results

Testing the code on real images

Now that we have selected the model, we were able to test it on images that did not appear in the training dataset. To carry out an exhaustive test about the reliability of the model, we will try to test it in difficult visibility conditions, where traffic signs will be truncated, partially masked or weather conditions will be bad.
Following the explanation of the choice of model, the examples below will all have been made with the RFCN-Resnet 101 model.

Figure 11: Traffic sign detection and classification using RFCN-Resnet 101

Here is a first example of detection. The bounding box frames the panel and adopts a color according to the category to which the model matches it. Above the box you can read the associated category, as well as the reliability percentage of the model for this detection.
We will now test the robustness of our model using less obvious images.

Figure 12: Traffic sign detection and classification in bad visibility conditions (night, snow, truncation)

We note that even in difficult visibility conditions, recognition results are satisfactory. In addition, it is important to note that the model, which had been trained with a German image base, recognizes well the French signs that look similar.
We also tested the model over digital signs like the ones that we can usually see before or inside tunnels, and it works well too:

Figure 13: Digital traffic sign detection and classification

It seems obvious to point out that this sign recognition tool is a driving assistance tool and is not sufficient on its own, a vigilance of the driver is of course always necessary.

We also implemented a function to return the position of the sign on the image. This function returns a vector like [ymin, xmin, ymax, xmax, % of reliability].
Here are the coordinates that were returned for the for the figure 11:

```
[[99, 268, 1503, 1650, 99.9985933303833]]
```

Figure 14: Example of the bounding box coordinates vector

Of course, if a single image shows various traffic signs, the function will return as many vectors as signs are detected. Subsequently, assuming that the signs all have the same regulated dimensions, these coordinates could be used to warn the user of the signs by classifying them in order of proximity to the car.

Deployment of this technology in our project

We chose to deploy our code on a Jetson Nano card. This kit from NVIDIA is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification. In addition, this card has a very affordable price which will make it possible to sell the final kit at an accessible price.
Even if the detection code works, there is still worked to be done at the deployment level. Before implementing the code on the board, it is necessary to configure it, that is to say to import the library, framework and software necessary for the code to work properly.

At the time of writing this report, we are still working on the board configuration. In addition to losing time because of the Covid-19 sanitary measurements, to share the hardware components, we have encountered difficulties with the Tensorflow and OpenCV installations. We are currently studying how to finalize the installation of the necessary software and libraries in order to deploy the code and to show it working for the final presentation.

V.2. Line Assist Algorithm

V.2.1. Interest of this feature

This part will be dedicated to the presentation of our functionality concerning line detection. The main objective is to detect and inform the driver about the position of his car in his lane.

The presence of this option in our kit adds a certain comfort to the driver.
Indeed, informing the driver can make it possible to notify him in case of temporary tiredness, drowsiness, or bad concentration on his driving, or if the car drifts out of its lane.
Moreover, this option can simply be a driving aid since it notifies the driver when there is a slight shift to inform him to refocus consistently.
Its implementation will be detailed later, and we will present the limits of the code that has been put in place. In our opinion, the use of such a functionality seems necessary especially on high-speed roads, such as highways, national or departmental roads.

V.2.2. Algorithm Analysis

V.2.2.1. Choice of Language and Interface

First, we will talk about the code that has been put in place. This feature has been coded under Python with the Spyder version (Python 3.8).

The objective is to take advantage of libraries that allow image processing work, such as the OpenCV library with the Canny function, the operation of which we will detail in the following.
The full script of the program is available in the attached folder under the name "Franchissement.py".

V.2.2.2. Mathematical Concepts

Detecting lines on the road requires the use of image processing and mathematical algorithms. The "Gradient" method is the one we have decided to use, since it allows us to determine the contours of an image by detecting a sudden change in brightness. All this with the aim of diverting the detected object.

The first step of our code is to measure these sudden changes in brightness and for this we have carried out a convolution treatment using Sobel's filters.
The Sobel filter is an operator used in image processing for the detection of contours. It is one of the simplest operators giving sufficiently satisfactory results for the use we are going to make of it.

This operator uses two convolution matrix, considering A the source image and two images (Gx and Gy) which in each point contain approximations of the horizontal (Gx) and vertical (Gy) derivative of each point.

Subsequently, the approximations of the horizontal and vertical gradients in each point will be combined to obtain an approximation of the norm of the gradient (G), allowing us to obtain variations of brightness according to x (with Gx) and according to y (with Gy). The combination of the two images thus obtained will finally allow us to observe the contours of our original image.

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{A} \quad \mathbf{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \mathbf{A} \quad \mathbf{G} = \sqrt{\mathbf{G_x}^2 + \mathbf{G_y}^2}$$

Figure 15: Sobel's filter Matrix

It is necessary to determine a threshold, above which we can consider the real presence of a contour. With the OpenCV library we have used the Canny function to perform this task. This function needs not one, but two thresholds (S1 and S2) for more precision.

```
capteur1=cv2.Canny(gray1, th1, th2)
```

Figure 16: Canny Function

A first pass over the image is performed and all pixels with a brightness higher than threshold number 2 (S2) will be considered to be contours and those with a brightness lower than threshold number 1 (S1) will be considered to be non-contours.
During the second pass, the pixels with a brightness between the two thresholds will be considered on a case-by-case basis.

The algorithm will check whether the pixel in question is connected to a pixel that has been previously considered as a contour, so that the pixel string connected to the contour pixel will be considered as a contour too.

Other pixels may have the same brightness but not be connected to a pixel representing a contour, so they will be considered as pixels representing a non-contour.

V.2.3. Visual of the Results Obtained

V.2.3.1. Demonstration and Test of correct operation of the Detection Areas

When the contours of each image in our video stream are detected, we try to focus on the contours of the lines surrounding the lane in which the vehicle is moving. To do this, we decided to target two detection areas. One of them targeting the area of the white line on the right of the windshield and the other targeting the area on the left.

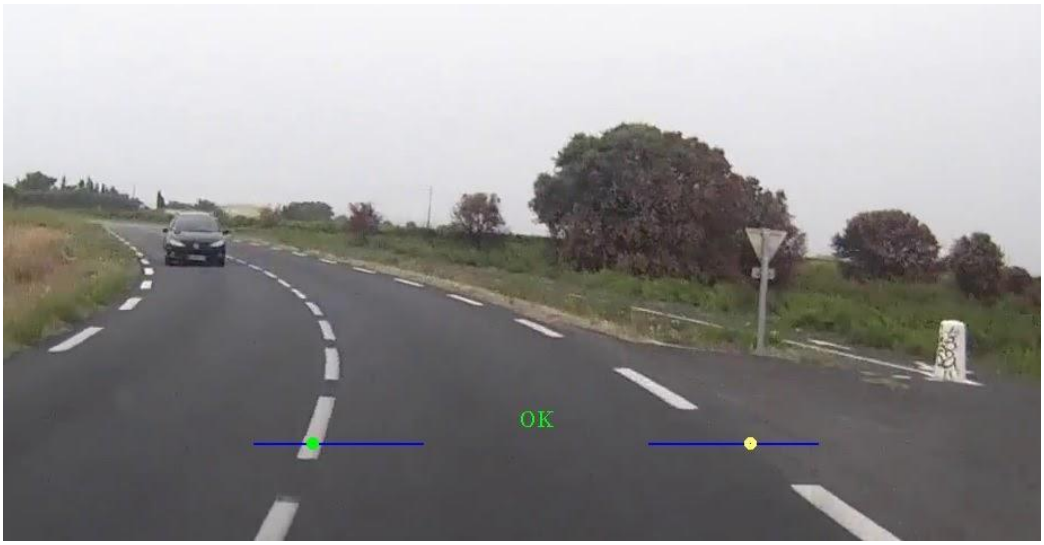


Figure 17: Detection Areas on the road

Let us consider the detection zone located on the right of the windshield (the operation of the left one remains identical), this one obviously aims at detecting the presence of the line delimiting the lane by the right.
The program detects if a contour, representing the line, is well perceived within the zone and keeps the information in memory for one second so as not to notify the driver of bad information when passing in a lane delimited by a discontinuous line.

If a contour is not detected for more than one second, the driver is informed that he is out of his lane or that his offset on the lane is too great and dangerous.

On the other hand, the program notifies the driver to refocus to a greater or lesser extent on his lane when there are slight shifts within the lane.

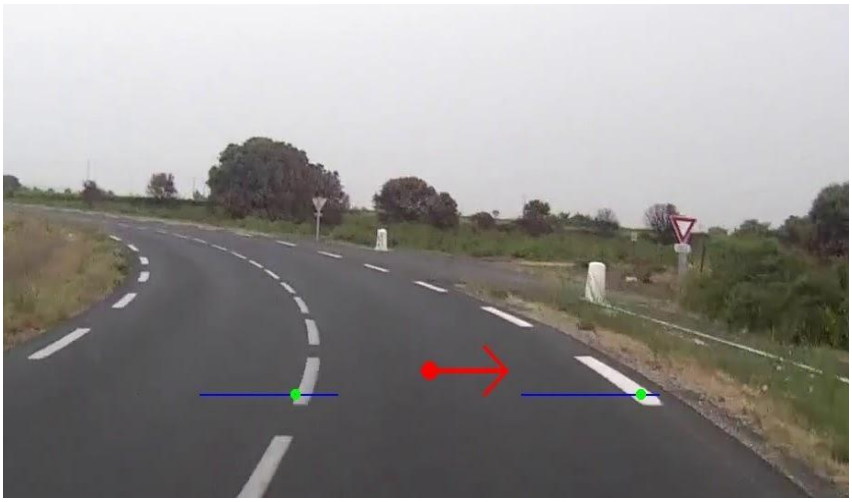


Figure 18: Arrow which informs the driver to refocus on the road

The results are conclusive, tests were carried out on video streams during good weather, rainy weather and at night. The operation of the program was in no way altered during these tests.

Figure 19: Tests with bad weather and during the night

V.2.3.2. Calibration of the Camera integrated in the Kit

In the current state, the program concerning the line detection could not be integrated into the housing, the integration part of the code and calibration is to be considered in a theoretical way. The objective, if the project is continued, is to achieve the result that we present below.

However, this code development imposes several constraints since the program requires calibration to correctly target the lines of the lane used by the vehicle.

The two detection zones must be pre-calibrated during the test phases. The camera has a fixed position within our kit, which will be accompanied by precise instructions as to the correct placement of the box within the passenger compartment by the driver.

VI. Mobile Application

VI.1. Context

In our project the main goal is to provide a cheap but effective tool to the user if he wants to add security and comfort features to its car. Our two guidelines were to develop the most efficient system so that we could decrease the ecological and financial impact of our product.
To do so, we decided to develop a mobile app that would connect to the system and serve as a screen to display the information about the traffic signs recognition, line assist and sensors.

While developing this mobile application we envisioned that the user would launch the app every time he would use the car, so that at every time, the information gathered by our system would be available, and displayed to him. The communication between the mobile application and the Jetson motherboard is a BLE communication, and we will explain all this in the next part of the report.

In addition to displaying the information gathered by the motherboard, we decided the mobile application could also serve as a helper to the user on some specific moments. For example, being that the mobile application is launched in the phone, that meant we could use the sensors and features available on smartphones and add some special functionalities to our application. As we will explain further down, we used the GPS, the accelerometer, and the gyroscope built-in every phone to add some extra functionalities.

VI.2. Development Environment

VI.2.1. Existing solutions for Mobile Applications Development

Smartphones are sometimes the only thing people carry around. It has replaced the photo and video camera, the computer, the notebook and sometimes even the gaming console. With the growth of smartphone users, the number of applications has also grown exponentially. There are three type of mobile applications.

Web applications that are basically like web sites but responsive, so everything can adapt to the screen size, and display correctly on every smartphone. This solution is a very easy and quick way to produce a mobile application, but it comes with its down sides. Web apps can sometimes be longer to launch then the other types of mobile apps, depending on how big and complex the web application is. But more importantly, web applications cannot access the hardware of the phone, so no sensors or features on the phone can be used on the mobile application, which limits the result.

Native applications [4] that are developed for a specific platform, in our case Android or iOS. This type of application is the most optimized and efficient one because they are developed exactly according to the phone's OS and can exploit every bit of performance and usability of the phone. The big advantage of native applications is that the hardware can be used inside the application, so the only limits will be the phone's capabilities, and range of sensors. But on the other hand,

[4] « Understanding the 3 Types of Mobile Apps ».

native apps must be developed in the OS language, which means developing two different applications, one for android and another for iOS, and maintaining these two applications.
Hybrid applications that can be installed on every smartphone as native apps, but that will be run by a web browser. This means that every hybrid app is developed in HTML5 and does not need maintenance for separate platforms. This is a good option to save time and resources, but still deliver content.

In our project, as we described before we want to use the sensors built-into the phone, and we have a complex application, so performance must be correctly settled to have a good user experience when using the mobile app. With that in mind we knew we needed to develop a native application, but we would not have the time to develop a native application for both android and iOS, so we decided to use a tool called Flutter, that allows us to have the two native applications of just one single code base.

VI.2.2. Flutter, a Google Framework

Flutter is a free and open-source mobile UI framework, tool created by Google, that allows the developer to create a native mobile application with only one codebase. The flutter compiler will build two different apps from this one codebase, one for Android and another for iOS.

The codebase needed to develop a Flutter application is developed in Dart[5], a language also developed also by Google, and that is very close to C/C++, which is another advantage to us, being that C and C++ are the programming language we are the most familiar with.

To be more specific flutter consists of two major parts. The first one is the Software Development Kit (SDK), that is basically a big set of tools giving the developer everything he needs to develop his applications. For example, the SDK has the compiler that will compile the final code into two different projects, one being the android native app and the other the iOS native app. The other hugely important part of Flutter[6] is the collection of reusable UI (user interface) elements, like sliders, buttons etc. giving the developer basic elements ready to be used. Having this UI elements collection means that the developer saves a lot of time because he does not need to develop every single detail, some things are ready to be used and personalized.

As explained before, Flutter has a very large collection of UI elements already coded and ready to be used and personalized by the developer. But this collection is more than just some elements made to save time for the developer. These elements are the core of flutter and the way it works.

A flutter application is an application where the developer controls every pixel on the phone. For example, when we add a button to the screen, we are telling flutter to display a certain button, with a certain size, color, and proportion at a specific place of the screen, and that will be true in the two projects, android, and iOS. Everything displayed is the same on both applications (flutter

[5] « Dart (Programming Language) ».

[6] « Flutter - Beautiful Native Apps in Record Time ».

also can separate native code and make the two applications different depending on the OS, but we did not use this feature).

In flutter, everything is a widget. The developer combines different widgets provided by the framework, making his own personalized widget, or he can also make a widget from scratch.

In our application we used a widget called grid View where we added the button widget, so we have a button grid as the main page. By doing this association between the two widgets, we have created a new widget, personalized exactly as we wanted it to be.

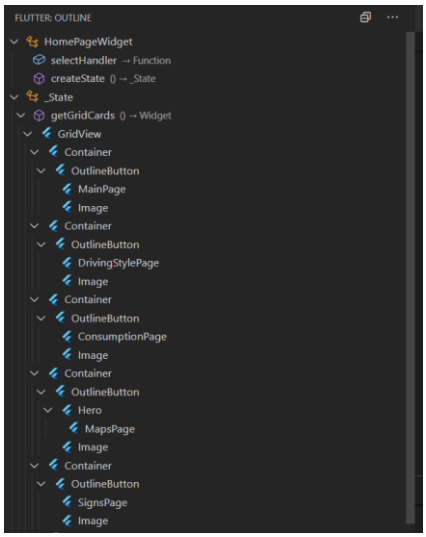


Figure 20: Widget tree for Home Page

At the end, the flutter app is the combination of hundreds of widgets, each one of them making or displaying something. Once everything is set up like the developer wants it to be, the SDK compiles it into the two different native apps ready to be posted on the AppStore and Play Store. Using Visual Studio as the development IDE, we could see the widget tree while coding.

VI.3. Our Mobile Application

As mentioned before, our mobile application has two main purposes. Firstly, it acts as a screen to the motherboard (Jetson Nano) to display the data sent from the Image processing algorithms and the sensors. Secondly, it has some helpful features aiming to make the users' life easier. The main page is a grid of big buttons, allowing the user to easily choose the information he wants displayed.

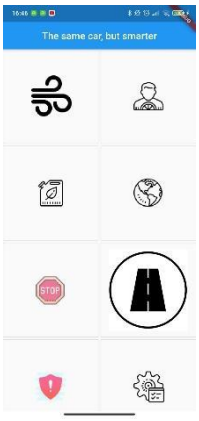
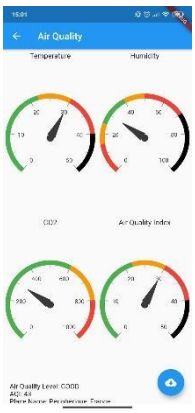


Figure 21: Result for the Home Page

Air Quality Monitoring

We will begin with the air quality page, providing the user key-information about the quality of the air he breathes inside and outside the car.
In this page the user can download the Air quality index for the city he is in, using the phone's GPS.



Our algorithm will download the information from reliable and official sources. Which source will depend on the GPS coordinates.
In addition to that, the motherboard will calculate an air quality index based on temperature, CO rates and humidity inside the car. This index will then be sent to the phone from via BLE and displayed by our app on the air quality page.

Alongside the two air indexes, the user will also have access to the CO rate, the temperature and the humidity percentage inside the car, and some advice depending on the situation. For example, if the temperature and humidity are too high inside the car, the phone will tell the user to open the windows for a while (if the exterior temperature is no higher) and to turn the AC on, because it purifies the air and will solve the humidity issue.

Figure 22: Result for the Air Quality Page

Traffic sign recognition

The traffic sign recognition is the algorithm explained above and it runs on the motherboard. When the algorithm captures a specific type of traffic sign, and after it has been through the priority check, it will send the type of traffic sign recognized, alerting the driver to pay attention to the road because it will have some important information displayed. If for example a speed limit sign is detected, the driver must be updated on this information so the motherboard will detect the sign and then the mobile application will alert the driver to the presence of a speed limit change.

With the evolution of the traffic sign recognition algorithm, the mobile application will be able, in the future, to tell exactly what traffic sign it has detected, and in the case of a speed change, it will be able to analyze it and compare it to the actual speed of the car.

Line Assist

As shown before, the line assist algorithm will also be running on the motherboard, and it will be looking to the road measuring the space between the car and the white lines spurring it, on the freeway for example. This analysis allows us to know if the driver is diverging too much to one of the sides, or if he continues to drive too close to a specific side of the road. The page to line assist displays a point representing the car inside two lines, and it moves according to the car's position on the road. If it gets too close to one side, the screen will flash red, and alert the driver he is driving too close to one of the sides. A sound alert will also be available if the user needs it.

For the moment, the Jetson Nano board is not yet able to run our image processing algorithms, and so this type of information is not displayed on the mobile app.

Driving style index

The driving style page aims to help the user be a better driver, along with the fuel economy page, that will be explained later.
Using the accelerometer and gyroscope built-in the phone, we decided to measure the number of abrupt changes in direction, strong accelerations, and deceleration, and calculate a driving quality index based on the number of these events.

This page is simply a gage representing the driving index at the current position according to the information recorded and analyzed, but it also contains advice to the driver. The advice shown differs depending on the current index value and will congratulate the driver if he has a good and comfortable driving style or alert him to a dangerous and uncomfortable driving style. It will show alert signs and advise him to avoid abrupt changes of direction and to use the accelerator or brake pedal with a softer touch.

Because our system is not connected to the car's controller, the motherboard cannot know if the driver uses his turn signals or not before changing trajectory, but a future evolution of the system could be installing some sensors allowing the system to have this information and use it to alert and advise the driver.

For the moment, the alert system is not yet available, and so the mobile app will show the number of abrupt changes in direction and advise the driver accordingly.

Fuel Economy

The fuel economy page is a very simple but effective way of helping the driver and owner of the car to understand if the car consumes too much fuel, or it starts consuming too much fuel at some point. We decided to include this feature to help people searching to save money but also reduce their impact on the planet.

To use this tool, the driver must fill a form each time he goes to the fuel station, entering the number of liters he put into the car, the current miles on the car, the total he paid at the gas station, and which gas station it was. This way, our application will calculate the average fuel consumption of the car, it will alert the driver if some high consumption is detected and will also calculate and show the driver which gas station was the cheapest to fill the car.

Another feature on this page is the possibility to add other expenses made with a car. The driver can enter the expenses made with oil changes and other repairs on the car and keep an eye on the total spent on the car each year.

For the moment, this feature does not save data locally on the phone. We ran out of time to develop a local database and were not able to finish the fuel economy page.

Parking helper

The parking page is an extremely helpful feature if the driver keeps forgetting where he parked the car, whether it was when going shopping or just going somewhere in town. This kept happening to some members of our team so we decided to add it to the mobile application, because it would truly help their everyday life, when using the car.
This page does not aim to replace a navigation app or to give the driver any indication on which road to follow, its only job is to have a map and to save the position of the car when parked, using the GPS built-in phone.

The parking page has a button that the user can press when he parks his car, and this saves the position of the car into the map. Later when he wants to get back to the car, the exact location of the vehicle and the current position of the user will be displayed on the map, so the user can see exactly where his car is and in which direction to move.

In addition to the position marker, the user can also add other markers on the map if for some reason he wants to save a particular position.

To conclude on the mobile application, we have developed and tested only on android phones, because to be able to compile the iOS version of the app, we would have to have an Apple computer set up with flutter. For the moment, a Windows computer was used for the development, but no further development should be needed to compile the mobile app, being that everything we used, every widget and function is iOS compatible. You will at the end of this report the GitHub link for the entirety of the project, including the source code of the mobile app. **VII. System Integration**

VII. System Integration

VII.1. Case Manufacturing

To offer a product based on our operational features to drivers, we have decided to design a 3D printed case in which we could insert our Jetson Nano and camera and then place it easily on the windshield. This aims to offer an attractive product but also a practical and convenient one, easy to place and deploy.

Specification :

- Size suitable for Jetson Nano Nvidia (70*45mm)
- Antenna mounting points for Intel 8265NGW Wireless-AC 8265 NGFF 802.11ac 867Mbps 2x2 MU-MIMO WIFI BT 4.2 Card
- Lightpipe hole for status LED
- 2 buttons for reset and power
- Space for fan cable
- Space/Holes for cable connections (for sensors)
- CSI cable hole
- Raspberry camera frame with GoPro compatible mounting
- 4 legs to insert suction cups

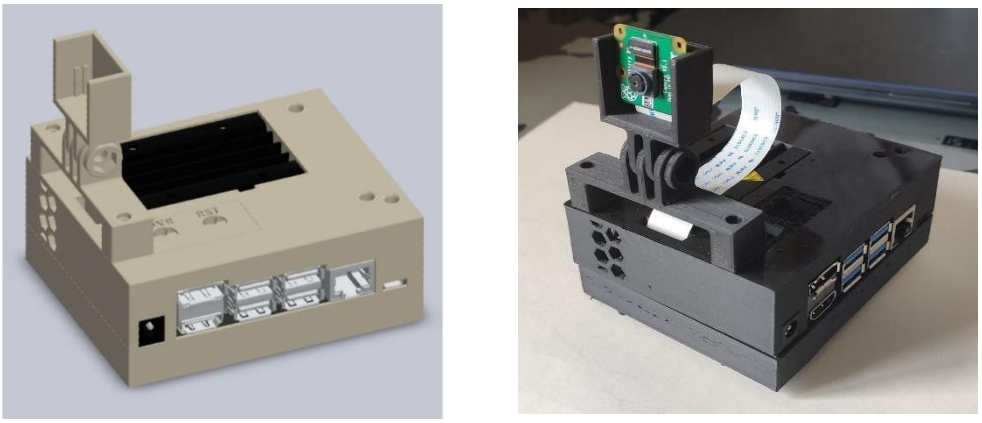


Figure 23: 3D model case for Jetson Nano and real printed case

VII.2. Motherboard to Smartphone Communication

Context

At the beginning of our project, we had planned to have the Jetson Nano board as our systems motherboard, to attach to it the sensors as well as the BLE module providing the Bluetooth communication with the mobile phone, the other piece of the puzzle. But due to delays on the delivery of the Bluetooth module for the Jetson Nano we decided to change our plans and use a ESP32 board, which was already available at INSA. This meant we were adding another piece to this puzzle, and we could face power issues in the future, but it also meant that we could start the integration processes, and that seemed more important.

What is Bluetooth, and BLE?

Bluetooth technology is a wireless networking protocol designed to connect devices quickly and remotely to computers or to each other. It can be used to connect cell phones to each other to exchange photos, to connect peripherals to a computer (mouse, keyboard, printer, etc.) or to connect headsets to cell phones to make hands-free calls (Bluetooth car kit). Bluetooth devices automatically detect and connect to each other, making communication much easier. There are two forms of Bluetooth: Bluetooth Classic and Bluetooth Low Energy, or BLE.

BLE is widely used as a protocol for the IoT. What makes BLE so interesting compared to other wireless protocols is that it is the easiest way to design a product able to communicate with any modern mobile platform (iOS, Android, etc.). Bluetooth Low Energy technology is a very low-power alternative for sensors and accessories. It is ideal for applications that do not require a continuous connection but require a long-life battery. It is not yet compatible with audio broadcasting, but it is compatible with remote controls, for example.

The use of ESP32 board to communicate with the mobile Application

The ESP32 board is basically the same as an Arduino board but without the ability to use analog inputs and, most importantly, the ESP32 as a BLE module built into it. And it has SPI, I2C, UART buses just like the Arduino Uno board used to develop the sensor algorithm, so it was easy to migrate the Arduino script to the ESP32.

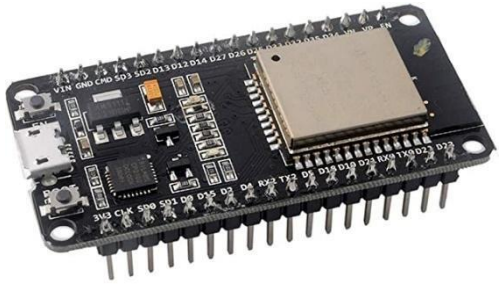


Figure 24: ES32 WROOM board

On the mobile application side, we had to develop a new script to connect to allow the user to connect to the system using his Bluetooth module built-in his phone. To do so, we added a connection page inside the air quality feature. This connection page allows the user to scan and find every Bluetooth device in the surroundings and ask permission for connection. Once everything is connected the phone receives all the sensor information from the ESP32, analyses it, and determines if the message received is the temperature, humidity, or gas rates, and displays it on the correct gauge.

For the moment, the connection must be confirmed each time the user uses the air quality page. The connection and the BLE device are saved, but in the future this last step will be automatic.

Communication between the Jetson Nano and ESP32

Because we have now a three-piece system, and the ESP32 is the one communicating with the phone, we must establish a connection between the Jetson Nano and the ESP32.

When we decided to use the ESP32 we also decided that the communication protocol between the Jetson Nano and the ESP32 would be UART. This way the Jetson Nano would simply send messages via the UART pins and the ESP32 would transmit them to the mobile app.

For the moment this link between the two boards is not ready because we ran out of time for the development of this part. In the future version of the product, this solution could be developed or replaced by the original plan of having the Jetson Nano controlling everything.

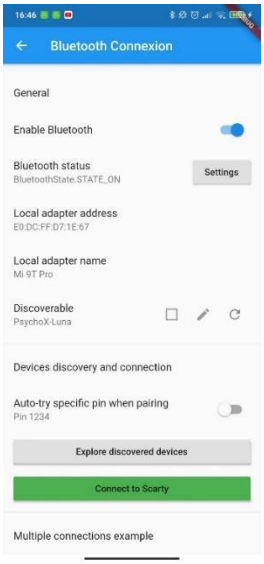


Figure 25: Bluetooth connection page

VIII. Project Management:

*"**Project management** is the application of processes, methods, skills, knowledge and experience to achieve specific project **objectives** according to the project acceptance criteria within agreed parameters."* (APM Body of knowledge 7th edition).

Project management covers all the aspects of a project, from the beginning to the end. It must consider objectives, time management, task repartition, management of the progress, the changes, and the risks but it could also consider any critical parameter such as communication within the team, or the Covid-19.

In our case, we decided to focus on these aspects: the process elaborated for achieving the project with its various steps, our time and team organization, change management, progress management, problem management. We will also try to have a critical point of view for each aspect.

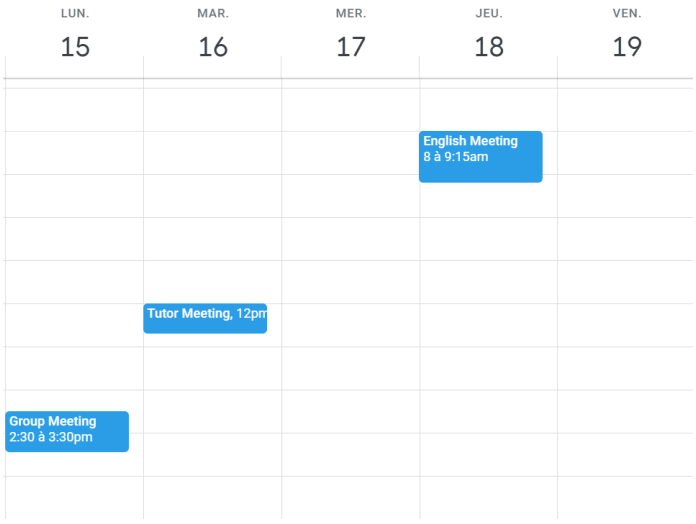
N°1:

Throughout our degree at INSA Toulouse, we only had few projects where we had to manage all the aspects (technical, financial, human…), and in which we were totally free to establish our own methodology.

During the first month, we realized a benchmarking of the existing solution in France and all around the world to have an overview of the product, the main features, but also identify the defects, and possible improvements. Then, we elaborated a survey based on our hypothesis, doubts but also our will to understand what the client needs and wants. Based on the results of the survey, we studied with our tutor the feasibility of the product in terms of material, technology, and time. Both of those steps gave us a guideline, and allowed us to establish objectives, specification, and a calendar.

We ordered the missing material. Then, we divided the project in various technical blocks and attributed each one to a member of the team, according to their skills and willingness to work on this aspect. For example, SKIKER had experience in project management, risks management to prepare meeting, planning, report, needs of each member of our group. El Ghali had a strong business background through his entrepreneurial experiences, that is why he oversaw the market research. Léo had a strong experience on image processing through his last internship, so he oversaw the panel recognition feature.

From the end of October to December, each one or group was working on the development of his specific part. Through weekly meetings, we presented our progress, and mainly our problems to benefit from the experience and the skills of the other members of the teams. We also identified in those meetings the main important points to report to our tutor to benefit from his advices and help. Every Thursday, we also realized meetings with our English professor to present our progress, and benefit from his critiques.



In January, we entered the integration part, which is the most complicated one, and that forces us to review our objectives. Finally, we did not have the time to test correctly and entirely our product.

Figure 26: General meetings schedule

N°2:

Beyond our task repartition, each one has a different role in the group to ensure good communication, progress but also a friendly atmosphere. We can say that those roles were naturally allocated, but we also tried to base those roles on each one's personality, and abilities.

For example, Skiker was in charge of all the administrative tasks (such as establishing meetings, updating the calendar, sending documents etc.…) as he is very used to dealing with this aspect. El Ghali was in charge of facilitating communication within the group. His personality and character gave him the ability to face people.

N°3:

Each member of the group has faced multiple problems on his technical component, due to configurations or lack of material for example. Of course, the first reflex is trying to solve the problem especially for an engineer. We all tried to solve our problems alone, then with the group, then with the tutor, then we tried harder.

As engineers, we wanted to get it solved, and we did not sufficiently have the business vision, to reevaluate our objectives, our product's specification. And so, we realized at the end that our product will be incomplete and that it will not respect the specification; we should have realized that beforehand in order to prevent our tutor and present our new objectives.

However, overall, we were able to bypass our technical problems and to find alternatives in order to present a working but improbable product with useful features.

N°4:

During this project, we were also confronted with human problems, and we had to adapt.
First, we have lost one of our teammate Redouane due to a health situation. In response, we had to review objectives relative to the functionality he was working on. We did not want to burden the workload of one of our teammates, and we decided instead to scale down our project.

VIII. Conclusion

Despite the various problems we encountered, we were able to adapt and develop a physical prototype that meets several functionalities initially planned.

However, the project is not finished, and can be continued by another group. Nevertheless, we have a case that can contain a motherboard, various sensors, communicating with an application we have developed, and a camera, which will be useful for the use of the functionalities performing image processing.
Some steps are missing, such as the integration on the Jetson board of the two codes concerning the image processing that could be carried out if the project has a sequel.

On the other hand, we have identified some points of improvement that would be interesting to implement to optimize our overall achievement.
An improvement of the program concerning the detection of the traffic signs would be judicious, to allow it to recognize the traffic sign and not only the type to which it belongs. The detection program could also be optimized to perhaps find another detection technique and allow us to avoid having to calibrate the camera and to simplify its use.
The addition of a battery in the case could also allow to store data and keep the values recovered by the gas sensors at any time.

You can find all our source code on this link: https://gitlab-rech.insa-toulouse.fr/monteil/box-for-smart-connected-car.git

Table of illustrations

Bibliography

« AQI Basics | AirNow.Gov ». AirNow.gov, U.S. EPA. Consulté le 17 janvier 2021. https://www.airnow.gov/aqi/aqi-basics.

Arcos-García, Álvaro, Juan Alvarez-Garcia, et Luis Soria Morillo. « Evaluation of Deep Neural Networks for traffic sign detection systems ». *Neurocomputing* 316 (1 août 2018). https://doi.org/10.1016/j.neucom.2018.08.009.

« Dart (Programming Language) ». In *Wikipedia*, 20 décembre 2020. https://en.wikipedia.org/w/index.php?title=Dart_(programming_language)&oldid=995355576.

« Flutter - Beautiful Native Apps in Record Time ». Consulté le 31 décembre 2020. https://flutter.dev/.

Shustanov, Alexander, et Pavel Yakimov. « CNN Design for Real-Time Traffic Sign Recognition ». *Procedia Engineering*, 3rd International Conference "Information Technology and Nanotechnology", ITNT-2017, 25-27 April 2017, Samara, Russia, 201 (1 janvier 2017): 718-25. https://doi.org/10.1016/j.proeng.2017.09.594.

Software Development & IT Staffing Company. « Understanding the 3 Types of Mobile Apps: Native, Mobile, and Hybrid | Charter Global », 23 mars 2020. https://www.charterglobal.com/understanding-the-3-types-of-mobile-apps-development-services/.