Une image contenant flèche

Description générée automatiquement

DOSSIER TECHNIQUE



Club mécatronique ENSET-M 3

e-mobility : Smart Drive

2ème édition de la compétition inter-écoles

Table des matières

[**I.** **Introduction :** 2](#_Toc133620275)

[**II.** **Le planning de GANT** 3](#_Toc133620276)

[**III.** **L’analyse fonctionnelle du projet** 4](#_Toc133620277)

[1. Chart de projet 4](#_Toc133620278)

[2. Bête à corne : 5](#_Toc133620279)

[3. Diagramme pieuvre : 5](#_Toc133620280)

[4. Diagramme de SADT 6](#_Toc133620281)

[**V.** **Les caractéristiques du véhicule électrique choisie : dimensions, puissance et le type d’alimentation** 8](#_Toc133620282)

[1. Les dimensions du véhicule et conception 8](#_Toc133620283)

[2. Les caractéristiques des composantes 9](#_Toc133620284)

[3. Les puissances et vérification de choix de la batterie 10](#_Toc133620286)

[**VI.** **La simulation des fonctions sur CARLA : Fonctions use-case, circuit de trajectoire ;** 10](#_Toc133620287)

[**VII.** **Le lien du Dataset utilisé (kaggel).** 11](#_Toc133620288)

[**VIII.** **La justification de la méthode d'apprentissage choisi et les outils.** 12](#_Toc133620289)

[**IX.** **Le programme python relative à l’use-case doit être rédigé et compilé** 13](#_Toc133620290)

[X. **Conclusion** 29](#_Toc133620291)

1. **Introduction :**

Nous avons préparé un dossier technique pour présenter notre participation à la compétition inter-écoles e-mobility: Smart Drive organisée par Capgemini Engineering. Notre objectif est de construire un prototype de voiture autonome équipé d'un système d'assistance à la conduite avancé (ADAS), afin de mettre en pratique nos compétences en mécatronique, robotique et ingénierie automobile, tout en nous permettant d'acquérir de nouvelles compétences et de renforcer notre esprit d'équipe à travers une compétition de haut niveau.

Le dossier technique explique en détail chaque étape de notre projet, de la conception à la réalisation, y compris les choix technologiques, les aspects mécaniques, électroniques et informatiques, ainsi que les tests et la validation. Nous avons également pris en compte les exigences et les contraintes du système ADAS, en veillant à la sécurité, à la performance, à l'ergonomie et à la convivialité.

En somme, le dossier technique présente notre prototype de voiture autonome équipé d'un système ADAS de manière complète et détaillée, dans le but de convaincre le jury de la compétition de la qualité et de la pertinence de notre travail.

1. **Le planning de GANT**

Une image contenant table

Description générée automatiquement

Une image contenant graphique

Description générée automatiquement

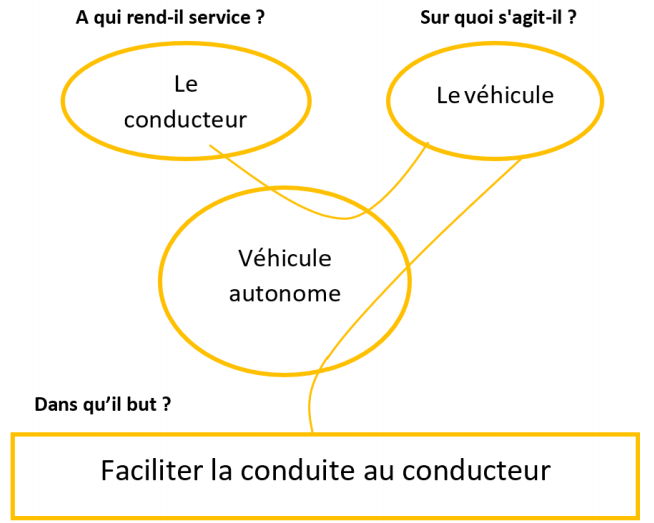
1. **L’analyse fonctionnelle du projet**

Nous avons réalisé une analyse fonctionnelle pour décrire les fonctions et les interactions de chaque composant de notre système ADAS pour notre prototype de voiture autonome. Cette analyse nous a permis de mieux comprendre le fonctionnement global de notre système, ainsi que les exigences et les contraintes qui y sont associées. Nous avons identifié les fonctions principales du système ADAS et détaillé les interactions entre chaque fonction et les différents composants. En somme, cette analyse nous a permis d'optimiser la performance de notre système en veillant à ce que chaque fonction remplisse son objectif de manière efficace et sécurisée.

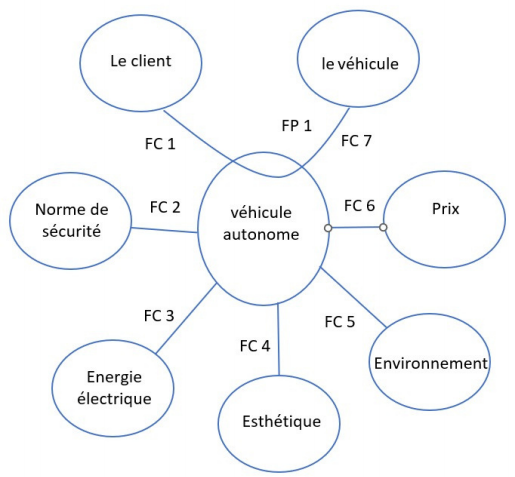
1. Chart de projet

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Intitulé du projet** | La Conception et la réalisation d'une voiture autonome |  |
| **Sociétés** | Capgemini |  |
| **Objéctifs du projet** | stimuler l'intérêt et la passion des étudiants pour la mobilité intelligente, en particulier la conduite autonome et les systèmes ADAS, tout en renforçant leurs compétences techniques dans un cadre compétitif. |  |
| **Impact du projet** | Cette compétition peut avoir un impact positif sur l'industrie automobile, les étudiants participants, ainsi que sur la réputation de Capgemini engineering. |  |
| **Contrainte du projet** |  |  |
| **Acteurs du projet** | |  |
|  |  |  |
| **Maître d'ouvrage** | ENSET M |  |
| **Constitution de l 'équipe du projet** | |  |
| **Encadrants** | RAFIK Mohamed |  |

1. Bête à corne :



1. Diagramme pieuvre :



1. Diagramme de SADT

**Réglage**

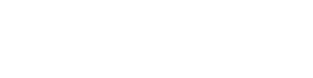
**Exploitation**

**configuration**

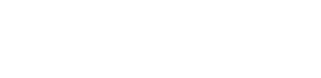
**Energie**

**Voiture autonome**

Faciliter la conduite au conducteur en toute sécurité



**Conduite non autonome et non sécurisé**



**Conduite non autonome et non sécurisé**



1. **Matériels Utilisés**

Nous avons sélectionné avec soin les matériels nécessaires pour la réalisation de notre prototype en prenant en compte les exigences de chaque composant et les contraintes liées à notre projet. Nous avons choisi des capteurs de haute qualité, des calculateurs puissants, des actionneurs précis et fiables, des dispositifs de communication performants ainsi que des batteries adaptées pour alimenter le système. Nous avons veillé à sélectionner des composants de haute qualité pour garantir le bon fonctionnement et la sécurité de notre prototype.

|  |  |  |  |
| --- | --- | --- | --- |
| Composent | Description | Nombre | Prix (MAD) |
| Raspberry pi 4 | Le Raspberry Pi 4 est un ordinateur monocarte, également appelé nano-ordinateur, qui fait partie de la série de mini-ordinateurs | 1 | 2 500 |
| Webcamera | webcam 1080p | 1 | 250 |
| L298N V3 FOUR DC MOTOR DRIVER | Le L298N V3 est capable de contrôler des moteurs DC jusqu'à 2A par canal, avec une tension d'alimentation allant jusqu'à 46V. Cela en fait un choix approprié pour les projets nécessitant le contrôle de moteurs puissants, tels que les robots, les voitures RC, les imprimantes 3D et autres applications nécessitant une puissance de moteur élevée. | 1 | 370 |
| DC Motor  Motor réducteur avec roue 65mm, DC 3-5v Tunisie | Moteur réducteur 6 V 1 : 48 pour Arduino Robot intelligent voiture.  Moteur réducteur 6 V 1 : 48 pour Arduino Robot intelligent voiture.  – Tension de fonctionnement : 3 V – 6 V  – Taille de roue : 65 x 26 mm (approx.). | 4 | 240 |
| [batterie-lithium-sony-18650-vtc6 3000mah 15](https://www.arduiplanet.ma/produit/batterie-lithium-sony-18650-vtc6-3000mah-15a/)A | une capacité de 3000mAh, cette batterie Li-Po offre une grande capacité de stockage d'énergie, ce qui peut être avantageux pour les projets nécessitant une alimentation prolongée, tels que les drones, les voitures télécommandées, | 3 | 300 |
| Arduino |  | 1 | 130 |
| Capteur ultrason | Ils peuvent être utilisés pour mesurer la distance, la présence ou la détection d'objets, ce qui les rend polyvalents et adaptés à différents besoins. | 5 | 40 |
| VENTILATEUR RACEPPERY |  | 1 | 30 |
| Power bank  Batterie externe Mi 50 W | Xiaomi France | On va utiliser le power Bank pour alimenter le Raspberry car il est besoin d’une alimentation stable, **20800mAh 5.8A 3-Port** | 1 | 70 |
| Module d’alimentation Lm2696 | Tension d’entrée : 4.5 – 40V  Tension de sortie : 1,5 – 35V (réglable) avec 3A max | 1 | 30 |
| LES VIS ET L'ECROS |  | -- |  |
| CABLES |  | 30 | 20 |
| Switch |  | 1 | 5 |

1. **Les caractéristiques du véhicule électrique choisie : dimensions, puissance et le type d’alimentation**
2. Les dimensions du véhicule et conception

Nous avons réussi à réaliser la conception de notre prototype de voiture autonome équipé d'un système ADAS grâce à notre expertise en mécatronique, en robotique et en ingénierie automobile.

En utilisant des outils de CAO tels que SolidWorks, nous avons pu concevoir avec précision les différentes parties de notre prototype, en prenant en compte les contraintes et les exigences du système ADAS. Nous avons également intégré les différents composants électroniques tels que les capteurs, la caméra, la batterie et les cartes électroniques dans la conception globale.

Une image contenant diagramme

Description générée automatiquement

1. Les caractéristiques des composantes

**Raspberry pi**

Le Raspberry pi 4 nécessite une alimentation USB-C de bonne qualité capable de fournir 5V à 3A. Si attaché

Les périphériques USB en aval consomment moins de 500mA, une alimentation 5V, 2.5A peut être utilisée

**Arduino uno**

tension d'alimentation : 5V

Tension d’entrée recommandée pour la broche Vin : 7-12V

Courant CC sur les broches I/O : 40 mA

**Capture ultrasonique HC-SR04**

Working Voltage: DC 5V · Working Current: 15mA · Working Frequency: 40Hz

**Power bank 20800 mAh**

# 20800mAh 5.8A 3-Port Power Bank Charger

**Moteur réducteur :**

Rapport de réduction : 1/48  
Tension de fonctionnement: 3V – 6V  
Couple: 0.8 kg/cm  
110 tr / min  
Diamètre de Roue : 70mm  
Épaisseur de la roue: 26.5mm  
Dimensions du moteur: 70 x 36,5 x 24,5 mm

**Batteries:**

**Turnigy 3000mAh 3S 20C Lipo Pack w/XT-60:**

Minimum Capacity: **3000mAh**  
Configuration: **4S1P / 14.8v / 4Cell**  
Constant Discharge: **20C**  
Peak Discharge (10sec): **30C**  
Pack Weight: **238g**  
Pack Size: **137x 43x 19**  
Charge Plug: **JST-XH**  
Discharge Plug: **XT-60**

1. Les puissances et vérification de choix de la batterie

|  |  |
| --- | --- |
| Raspberry Pi | 5 W |
| l'Arduino | 0,5 W |
| les capteurs | 5x 0,5 W |
| caméra | 15W |
| L298N | 2.5 W |
| Moteurs | 4x3W |

Nous avons estimé la consommation électrique de notre système à 30W on ajoute le facteur de sécurité de chaque composant.

1. **La simulation des fonctions sur CARLA : Fonctions use-case, circuit de trajectoire ;**

Nous avons installé et testé le simulateur de conduite CARLA, qui nous a permis de simuler des scénarios de conduite complexes et d'analyser les performances de notre prototype et notre modèle dans des conditions réalistes. Les résultats de nos tests ont été prometteurs et nous avons pu identifier des points d'amélioration pour réaliser des simulations dans ce simulateur.



Une image contenant diagramme

Description générée automatiquement

1. **Le lien du Dataset utilisé (kaggel).**

[**https://www.kaggle.com/datasets/andrewmvd/road-sign-detection**](https://www.kaggle.com/datasets/andrewmvd/road-sign-detection)

****

Ce Dataset contient 877 images de 4 classes distinctes pour l'objectif de détection des panneaux de signalisation.

Les annotations de la boîte englobante sont fournies au format PASCAL VOC les classes sont :

* Feu de circulation
* Stop
* Limitation de vitesse
* Passage clouté

Le choix de ce dataset est liée à plusieurs raisons :

* **Pertinence :** La détection des panneaux de signalisation routière est une tâche importante en vision par ordinateur qui a des applications pratiques pour la conduite autonome, la sécurité routière et la navigation. En outre, ce dataset contient un grand nombre de catégories de panneaux différents, ce qui le rend intéressant pour la reconnaissance de la diversité des panneaux de signalisation.
* **Qualité :** Ce dataset contient 877 images avec des annotations précises des panneaux de signalisation routière, ce qui le rend riche en informations. Les images ont été collectées dans différentes conditions d'éclairage, de temps et d'angles de vue, ce qui rend le dataset diversifié et adapté à l'entraînement de modèles de vision par ordinateur pour la détection des panneaux de signalisation routière dans des situations réelles.
* Accessibilité : Le dataset est disponible gratuitement sur Kaggle et peut être téléchargé facilement. En outre, il est accompagné d'une documentation détaillée qui explique la structure des données et les annotations des images, ce qui facilite l'utilisation du dataset pour la formation des modèles.

Le choix de ce dataset est justifié par sa pertinence, sa qualité et son accessibilité pour l'entraînement de modèles de détection de panneaux de signalisation routière.

1. **La justification de la méthode d'apprentissage choisi et les outils.**

Le choix du meilleur modèle ou méthode d’apprentissage dépend de la tâche que nous souhaitons accomplir avec le dataset. Si notre objectif est de détecter la présence de panneaux de signalisation routière dans une image, nous pouvons utiliser une SVM ou un réseau de neurones convolutifs (CNN) pour classer chaque image comme étant soit une image contenant un panneau de signalisation routière, soit une image ne contenant pas de panneau de signalisation routière.

Cependant, si nous avons besoin de prédire la localisation précise des panneaux de signalisation routière dans l'image, nous devons utiliser une approche de détection d'objets telle que YOLO ou SSD, qui sont spécifiquement conçues pour la détection d'objets dans les images.

En fin de compte, le choix du meilleur modèle dépendra de notre objectif spécifique et de la complexité de la tâche. Il peut être nécessaire d'expérimenter avec plusieurs modèles différents et de mesurer leur performance pour déterminer celui qui convient le mieux à notre tâche.

Pour la classification des panneaux de signalisation routière dans les images, un réseau de neurones convolutifs (CNN) est l'un des meilleurs modèles à utiliser. Les CNN ont été largement utilisés avec succès dans de nombreuses tâches de vision par ordinateur, y compris la classification d'images, la détection d'objets, la segmentation d'images et bien plus encore.



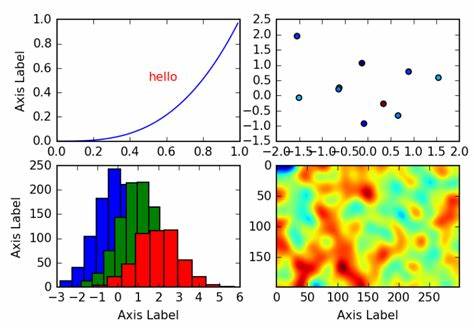
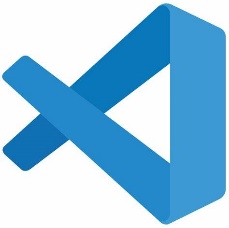
Pour entraîner un CNN, nous devons prétraiter les images, les redimensionner en une taille uniforme, et les diviser en ensembles d'entraînement, de validation et de test. Ensuite, nous pouvons entraîner le CNN en utilisant l'ensemble d'entraînement, en ajustant les poids des neurones du réseau pour minimiser l'erreur de classification. Nous pouvons ensuite évaluer la performance du modèle en utilisant l'ensemble de validation, en ajustant les hyperparamètres du modèle si nécessaire.

**Outilles :**

* **TensorFlow**: est une bibliothèque open source populaire pour l'apprentissage automatique et le calcul numérique développée par Google. Elle est utilisée pour la création et le déploiement de modèles d'apprentissage profond tels que les réseaux de neurones. TensorFlow est conçue pour fonctionner efficacement sur des systèmes distribués et permet ainsi de traiter de grands ensembles de données à une échelle industrielle. En utilisant TensorFlow, les développeurs peuvent créer des modèles d'apprentissage automatique pour une variété d'applications telles que la reconnaissance d'images, la traduction de langues, la prédiction de séries temporelles et bien d'autres encore. TensorFlow est largement utilisée dans l'industrie et la recherche en raison de sa facilité d'utilisation, de sa grande flexibilité et de sa grande échelle.
* **Scikit-learn**: est également connu sous le nom de sklearn, est une bibliothèque Python open source pour l'apprentissage automatique. Elle est conçue pour fonctionner avec d'autres bibliothèques Python telles que NumPy et SciPy. Scikit-learn propose une large gamme d'algorithmes d'apprentissage automatique pour la classification, la régression, le clustering et la réduction de dimensionnalité, ainsi que des utilitaires pour le prétraitement des données, la sélection de modèles et l'évaluation de performances.



Scikit-learn est très populaire dans la communauté de l'apprentissage automatique en raison de sa facilité d'utilisation, de sa flexibilité et de ses performances. Il est utilisé dans une grande variété de domaines tels que la science des données, la biologie, la finance, la reconnaissance vocale et d'image, ainsi que dans de nombreuses applications industrielles.

* **OpenCV (Open Source Computer Vision Library)** : est une bibliothèque open-source de vision par ordinateur et de traitement d'image. Elle fournit des fonctions pour l'acquisition, la manipulation, le traitement et l'analyse d'images et de vidéos en temps réel. OpenCV a été initialement développé en C++, mais il est également disponible dans de nombreux autres langages de programmation tels que Python, Java et MATLAB. La bibliothèque est utilisée dans une variété d'applications telles que la reconnaissance faciale, la détection d'objets, la vision industrielle, la robotique, la réalité augmentée et bien d'autres encore.
* **NumPy (Numerical Python)** : est une bibliothèque open source de calcul numérique en Python. Elle fournit des structures de données pour stocker des tableaux multidimensionnels et des matrices, ainsi que des fonctions pour effectuer des opérations mathématiques sur ces tableaux tels que l'algèbre linéaire, la statistique, les transformations de Fourier, etc.
* **Matplotlib** : est une bibliothèque de visualisation de données en Python. Elle permet de créer des graphiques, des histogrammes, des diagrammes de dispersion et bien plus encore. Matplotlib est utilisé pour générer des visualisations de données dans divers domaines tels que les sciences, l'ingénierie, les statistiques, la finance et l'informatique. La bibliothèque est hautement personnalisable, ce qui permet de créer des visualisations adaptées aux besoins spécifiques de l'utilisateur. Matplotlib est largement utilisé dans la communauté scientifique et est une bibliothèque essentielle pour tout scientifique de données en Python.
* **VSCode** :est un éditeur de code source développé par Microsoft. Il est disponible sur plusieurs plateformes, notamment Windows, macOS et Linux. VSCode est populaire auprès des développeurs en raison de sa rapidité, de sa facilité d'utilisation et de son extensibilité. Il prend en charge plusieurs langages de programmation et fournit des fonctionnalités avancées telles que la coloration syntaxique, la complétion de code, la détection d'erreurs et le débogage. VSCode est également livré avec un certain nombre d'extensions qui peuvent être installées pour améliorer ses fonctionnalités et répondre aux besoins spécifiques des développeurs.
* **Google Colaboratory (Colab en abrégé)** :est un service cloud de Google qui permet d'exécuter du code Python en utilisant un navigateur web. Il est conçu pour faciliter l'apprentissage en ligne de la science des données, de l'apprentissage automatique et de l'intelligence artificielle en fournissant un environnement de développement gratuit, en ligne et prêt à l'emploi.

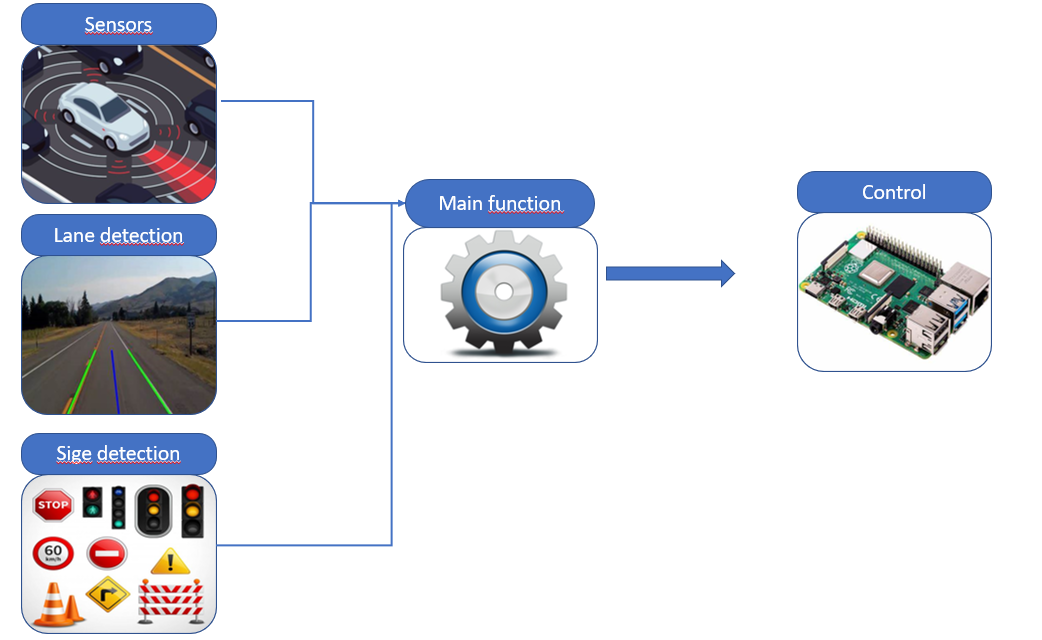
Colab est basé sur le projet Jupyter Notebook, qui permet de créer et de partager des documents interactifs contenant du code, du texte, des graphiques et des images. Avec Colab, vous pouvez exécuter du code Python dans le cloud sans avoir à installer et à configurer un environnement de développement sur votre propre ordinateur. Vous pouvez également collaborer avec d'autres utilisateurs en partageant votre notebook avec eux et en travaillant ensemble en temps réel.

Colab est également livré avec une gamme d'outils et de bibliothèques de traitement des données et d'apprentissage automatique, y compris TensorFlow, PyTorch, Scikit-Learn et Pandas, ce qui en fait un excellent choix pour l'expérimentation et l'apprentissage de ces technologies.

Note bien : Nous avons utilisée google colaboratory pour travailler ensemble dans le même projet

1. **Le programme python relative à l’use-case doit être rédigé et compilé**

Architecture de notre programme



**La fonction Main.py**

from sign\_detection\_module import sign\_detection

import lane\_detection\_module as ld

import numpy as np

import threading

import Control

import cv2

import sys

import time

class lane\_detection(object):

    def \_\_init\_\_(self):

        """ Variables """

        self.latestImage = None

        self.cap = cv2.VideoCapture("http://192.168.137.19:8080/video")

        #self.cap = cv2.VideoCapture(0)

        cv2.namedWindow('Lane detection', cv2.WINDOW\_NORMAL)

        cv2.resizeWindow('Lane detection', 640, 480)

        self.cap.set(cv2.CAP\_PROP\_AUTOFOCUS, 0)

        self.cap.set(3, 640)  # width=1920

        self.cap.set(4, 480)  # height=1080

        self.kernel\_size = 19

        self.low\_threshold = 0

        self.high\_threshold = 20

        self.fps = 0

        self.height = self.cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)

        self.width = self.cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)

        print((self.height, self.width))

        self.Sd = sign\_detection(self.width, self.height)

    def run(self):

        start\_height = 280 # Scan index row 235

        # Run sign detection thread

        x = threading.Thread(target=self.Sd.run)

        x.start()

        Auto = False

        while cv2.getWindowProperty('Lane detection', 0) >= 0:

            # Start time

            start = time.time()

            # Only run loop if we have an image

            OK, self.latestImage = self.cap.read()

            if OK:

                # Rotate frame

                self.latestImage = cv2.rotate(self.latestImage, cv2.ROTATE\_90\_CLOCKWISE)

                # step 1: detect Sign

                self.Sd.IN\_image = self.latestImage

                if self.Sd.sign\_detected:

                    self.latestImage = self.Sd.OUT\_image

                    Control.Stop()

                    Auto = False

                # step 2: detect binary lane markings

                self.blurImage = ld.gaussian\_blur(self.latestImage, self.kernel\_size)

                light\_black = (100, 100, 100)

                dark\_black = (0, 0, 0)

                self.blurImage = cv2.inRange(self.blurImage, dark\_black, light\_black)

                #self.edgeImage = ld.canny(self.blurImage, self.low\_threshold, self.high\_threshold)

                self.blurImage = 255 - self.blurImage

                # step 3: Define region of interest for cropping

                height = self.latestImage.shape[0]

                width = self.latestImage.shape[1]

                vertices = np.array( [[

                        [5\*width/6, 2\*height/6],

                        [width/6, 2\*height/6],

                        [20, height-10],

                        [width-20, height-10]

                ]], dtype=np.int32 )

                self.maskedImage = ld.region\_of\_interest(self.blurImage, vertices)

                ret, thresh = cv2.threshold(self.maskedImage, 220, 255, cv2.THRESH\_BINARY\_INV)

                signed\_thresh = thresh[start\_height].astype(np.int16)  # select only one row

                diff = np.diff(signed\_thresh)  # The derivative of the start\_height line

                points = np.where(np.logical\_or(diff > 200, diff < -200)) #maximums and minimums of derivative

                if len(points) > 0 and len(points[0]) > 1: # if finds something like a black line

                    middle = int(np.mean(points[0]))

                    #print("Mean point : " + str(middle))

                    cv2.circle(self.latestImage, (points[0][0], start\_height), 5, (255,0,0), -1)

                    cv2.circle(self.latestImage, (points[0][-1], start\_height), 5, (255,0,0), -1)

                    cv2.circle(self.latestImage, (int(middle), int(start\_height)), 10, (0,0,255), -1)

                    cv2.putText(self.latestImage, text= "FPS : " + str(self.fps), org=(10,20),

                    fontFace= cv2.FONT\_HERSHEY\_SIMPLEX, fontScale=1/2, color=(0,0,0),

                    thickness=1, lineType=cv2.LINE\_AA)

                    if Auto:

                        thr = 80

                        Control.Move(middle, width, thr)

                    else:

                        Control.Stop()

                        pass

                # End time

                end = time.time()

                if end != start:

                    self.fps = int(1/(end-start))

                cv2.imshow('Lane detection', self.latestImage)

            # press 'Q' if you want to exit

            if cv2.waitKey(1) & 0xFF == ord('q'):

                print("Exiting......")

                print("Good By !!")

                self.cap.release()

                break

            elif cv2.waitKey(1) & 0xFF == ord('a'):

                print("Mode changed")

                Auto = not Auto

if \_\_name\_\_ == '\_\_main\_\_':

    Ld = lane\_detection()

    Ld.run()

    cv2.destroyAllWindows()

La fonction main implémente les autres fichier qui contient le système de détection de ligne et de panneaux de signalisation routière en utilisant la bibliothèque OpenCV et la méthode de transformation de seuillage pour détecter les pixels noirs. La détection de ligne est réalisée en trouvant la position du milieu de la ligne sur une seule ligne d'indice de la matrice transformée. Si vous souhaitez améliorer la précision du système, vous pouvez envisager d'utiliser des méthodes de détection de ligne plus sophistiquées telles que la transformée de Hough pour détecter des lignes droites dans l'image.

**Lane\_detection\_modele.py**

import cv2

import numpy as np

import math

def gaussian\_blur(img, kernel\_size):

    """Applies a Gaussian Noise kernel"""

    return cv2.GaussianBlur(img, (kernel\_size, kernel\_size), 0)

def canny(img, low\_threshold, high\_threshold):

    """Applies the Canny transform"""

    return cv2.Canny(img, low\_threshold, high\_threshold)

def region\_of\_interest(img, vertices):

    """

    Applies an image mask.

    Only keeps the region of the image defined by the polygon

    formed from `vertices`. The rest of the image is set to black.

    """

    #defining a blank mask to start with

    mask = np.zeros\_like(img)

    #defining a 3 channel or 1 channel color to fill the mask with depending on the input image

    if len(img.shape) > 2:

        channel\_count = img.shape[2]  # i.e. 3 or 4 depending on your image

        ignore\_mask\_color = (255,) \* channel\_count

    else:

        ignore\_mask\_color = 255

    #filling pixels inside the polygon defined by "vertices" with the fill color

    cv2.fillPoly(mask, vertices, ignore\_mask\_color)

    #returning the image only where mask pixels are nonzero

    masked\_image = cv2.bitwise\_and(img, mask)

    return masked\_image

Ce code est une collection de fonctions utilisées pour traiter des images en utilisant la bibliothèque OpenCV en Python. Les fonctions incluent le flou gaussien, la transformation de Canny et la sélection de région d'intérêt.

La fonction "**gaussian\_blur**" applique un noyau de bruit gaussien à l'image d'entrée pour réduire le bruit.

La fonction "**canny**" applique la transformation de Canny à l'image d'entrée pour détecter les bords de l'image.

La fonction "**region\_of\_interest**" permet de sélectionner une région spécifique de l'image en masquant le reste de l'image. Elle prend en entrée les coordonnées des points qui définissent la région à garder et les utilise pour créer un masque qui ne garde que les pixels à l'intérieur de la région définie.

**Simple\_Sign\_Detection\_Modele.py**

import cv2

import numpy as np

from time import sleep

class sign\_detection(object):

        def \_\_init\_\_(self, width, height):

                self.scale\_percent = 100 # percent of original size

                self.width = int(width \* self.scale\_percent / 100)

                self.height = int(height \*self. scale\_percent / 100)

                self.dim = (self.width, self.height)

                self.sign\_detected = False

                # load classifier

                self.stop\_sign\_ENG\_cascade = cv2.CascadeClassifier('stopsign\_classifier.xml')

                self.IN\_image = None

                self.OUT\_image = None

                self.is\_running = True

        def run(self):

                while self.is\_running:

                        sleep(0.001)

                        try:

                                if not self.IN\_image.any():

                                        continue

                        except:

                                continue

                        #image = cv2.resize(self.IN\_image, self.dim)

                        img\_filter = cv2.GaussianBlur(self.IN\_image, (5, 5), 0)

                        gray\_filered = cv2.cvtColor(img\_filter, cv2.COLOR\_BGR2GRAY)

                        stop\_signs\_ENG = self.stop\_sign\_ENG\_cascade.detectMultiScale(gray\_filered, scaleFactor=1.1, minNeighbors=30, minSize=(60, 60))

                        if len(stop\_signs\_ENG):

                                print("STOP sign Detected")

                                for (x,y,w,h) in stop\_signs\_ENG:

                                        x \*= int(1 / (self.scale\_percent / 100))

                                        y \*= int(1 / (self.scale\_percent / 100))

                                        w \*= int(1 / (self.scale\_percent / 100))

                                        h \*= int(1 / (self.scale\_percent / 100))

                                        self.IN\_image = cv2.rectangle(self.IN\_image, (x, y), (x+w, y+h), (255, 255, 0), 3)

                                self.sign\_detected = True

                        else:

                                self.sign\_detected = False

                        self.OUT\_image = self.IN\_image

                        self.IN\_image = None

def detect\_sign(ORimage):

        scale\_percent = 30 # percent of original size

        width = int(ORimage.shape[1] \* scale\_percent / 100)

        height = int(ORimage.shape[0] \* scale\_percent / 100)

        dim = (width, height)

        image = cv2.resize(ORimage, dim)

        sign\_detected = False

        # load classifier

        stop\_sign\_ENG\_cascade = cv2.CascadeClassifier('stop\_sign\_classifier\_ENG.xml')

        img\_filter = cv2.GaussianBlur(image, (5, 5), 0)

        gray\_filered = cv2.cvtColor(img\_filter, cv2.COLOR\_BGR2GRAY)

        stop\_signs\_ENG = stop\_sign\_ENG\_cascade.detectMultiScale(gray\_filered, scaleFactor=1.05, minNeighbors=20, minSize=(20, 20))

        if len(stop\_signs\_ENG):

                print("STOP sign Detected")

                for (x,y,w,h) in stop\_signs\_ENG:

                        x \*= int(1 / (scale\_percent / 100))

                        y \*= int(1 / (scale\_percent / 100))

                        w \*= int(1.2 / (scale\_percent / 100))

                        h \*= int(1.2 / (scale\_percent / 100))

                        ORimage = cv2.rectangle(ORimage, (x, y), (x+w, y+h), (255, 255, 0), 3)

                sign\_detected = True

        return sign\_detected, ORimage

Le code ci-dessus contient deux classes:

La classe **sign\_detection** est une classe qui peut détecter les panneaux d'arrêt dans une image. Elle utilise un classificateur en cascade pour détecter les panneaux d'arrêt dans l'image. Lorsqu'un panneau d'arrêt est détecté, un rectangle est dessiné autour du panneau d'arrêt dans l'image d'entrée.

La fonction **detect\_sign** prend une image en entrée et utilise également un classificateur en cascade pour détecter les panneaux d'arrêt dans l'image. Si un panneau d'arrêt est détecté, un rectangle est dessiné autour du panneau d'arrêt dans l'image d'entrée et la fonction retourne True. Si aucun panneau d'arrêt n'est détecté, la fonction retourne False et l'image d'entrée n'est pas modifiée.

Les deux utilisent la méthode de flou gaussien pour supprimer les bruits dans l'image, convertissent l'image en niveaux de gris, puis appliquent un détecteur de panneaux d'arrêt en cascade sur l'image.

**Advance\_Sign\_Detection\_Modele.py**

import sys

import os

import cv2

import datetime

# import imutils

import numpy as np

from pathlib import Path

import random

import xml.etree.ElementTree as ET

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Input

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.preprocessing.image import img\_to\_array

from tensorflow.keras.preprocessing.image import load\_img

from tensorflow.keras.models import load\_model

from tensorflow.keras.applications import VGG16

from tensorflow.keras.layers import Dropout

from tensorflow.keras.utils import to\_categorical

from sklearn.preprocessing import LabelBinarizer

import pickle

data = []

labels = []

bboxes = []

imagePaths = []

# classes = ["trafficlight", "speedlimit", "crosswalk", "stop"]

annot\_dir  = "./annotations"

images\_dir = "./images"

for filename in os.listdir(annot\_dir):

    f = os.path.join(annot\_dir, filename)

    tree = ET.parse(f)

    root = tree.getroot()

    w = int(root.find('./size/width').text)

    h = int(root.find('./size/height').text)

    for box in root.findall('./bndbox'):

        xmin = int(box.find('xmin').text) / w

        ymin = int(box.find('ymin').text) / h

        xmax = int(box.find('xmax').text) / w

        ymax = int(box.find('ymax').text) / h

    label = root.find('./object/name').text

    imname = root.find('./filename').text

    impath = os.path.join(images\_dir, imname)

    image = load\_img(impath, target\_size=(224,224))

    image = img\_to\_array(image)

    data.append(image)

    labels.append(label)

    bboxes.append((xmin,ymin,xmax,ymax))

    imagePaths.append(impath)

# normalize -> from [0-255] to [0-1]

data = np.array(data, dtype="float32") / 255.0

# convert to np arrays

labels = np.array(labels)

bboxes = np.array(bboxes, dtype="float32")

imagePaths = np.array(imagePaths)

# one-hot encoding

lb = LabelBinarizer()

labels = lb.fit\_transform(labels)

# test-train split 20%,80%

split = train\_test\_split(data,

                         labels,

                         bboxes,

                         imagePaths,

                         test\_size=0.20,

                         random\_state=12)

(trainImages, testImages) = split[:2]

(trainLabels, testLabels) = split[2:4]

(trainBBoxes, testBBoxes) = split[4:6]

(trainPaths,  testPaths)  = split[6:]

# saving test files for later use

with open("testing\_multiclass.txt", "w") as f:

    f.write("\n".join(testPaths))

# The Neural Net : Architecture

vgg = VGG16(weights="imagenet",

            include\_top=False,

            input\_tensor=Input(shape=(224, 224, 3)))

# freeze training any of the layers of VGGNet

vgg.trainable = False

# max-pooling is output of VGG, flattening it further

flatten = vgg.output

flatten = Flatten()(flatten)

bboxHead = Dense(128, activation="relu")(flatten)

bboxHead = Dense(64, activation="relu")(bboxHead)

bboxHead = Dense(32, activation="relu")(bboxHead)

bboxHead = Dense(4, activation="sigmoid", name="bounding\_box")(bboxHead)

# 4 neurons correspond to 4 co-ords in output bbox

softmaxHead = Dense(512, activation="relu")(flatten)

softmaxHead = Dropout(0.5)(softmaxHead)

softmaxHead = Dense(512, activation="relu")(softmaxHead)

softmaxHead = Dropout(0.5)(softmaxHead)

softmaxHead = Dense(len(lb.classes\_), activation="softmax", name="class\_label")(softmaxHead)

model = Model(

    inputs=vgg.input,

    outputs=(bboxHead, softmaxHead))

INIT\_LR = 1e-4

NUM\_EPOCHS = 25

BATCH\_SIZE = 16

losses = {

    "class\_label": "categorical\_crossentropy",

    "bounding\_box": "mean\_squared\_error",

}

lossWeights = {

    "class\_label": 1.0,

    "bounding\_box": 1.0

}

trainTargets = {

    "class\_label": trainLabels,

    "bounding\_box": trainBBoxes

}

testTargets = {

    "class\_label": testLabels,

    "bounding\_box": testBBoxes

}

opt = Adam(INIT\_LR)

model.compile(loss=losses,

              optimizer=opt,

              metrics=["accuracy"],

              loss\_weights=lossWeights)

#print(model.summary())

# Training

H = model.fit(

    trainImages, trainTargets,

    validation\_data=(testImages, testTargets),

    batch\_size=BATCH\_SIZE,

    epochs=NUM\_EPOCHS,

    verbose=1)

model.save("model\_bbox\_regression\_and\_classification", save\_format="h5")

f = open("lb.pickle", "wb")

f.write(pickle.dumps(lb))

f.close()

# Testing the model

def predict(imagePath):

     # loading input image

    image = load\_img(imagePath, target\_size=(224, 224))

    image = img\_to\_array(image) / 255.0

    image = np.expand\_dims(image, axis=0)

    # predicting bbox and label

    model = load\_model("./model\_bbox\_regression\_and\_classification")

    lb = pickle.loads(open("./lb.pickle", "rb").read())

    (boxPreds, labelPreds) = model.predict(image)

    (startX, startY, endX, endY) = boxPreds[0]

    # finding class label with highest pred. probability

    i = np.argmax(labelPreds, axis=1)

    label = lb.classes\_[i][0]

    return label

def main():

    path = "testing\_multiclass.txt"

    filenames = open(path).read().strip().split("\n")

    for f in filenames:

        lablePredict=predict(f)

        print(lablePredict)

if \_\_name\_\_=="\_\_main\_":

    main()

Le code ci-dessus permet de construire un modèle de réseau de neurones capable de détecter et de classifier des objets dans des images en utilisant l'apprentissage profond. Il utilise le dataset "annotations" et "images" pour entraîner le modèle.

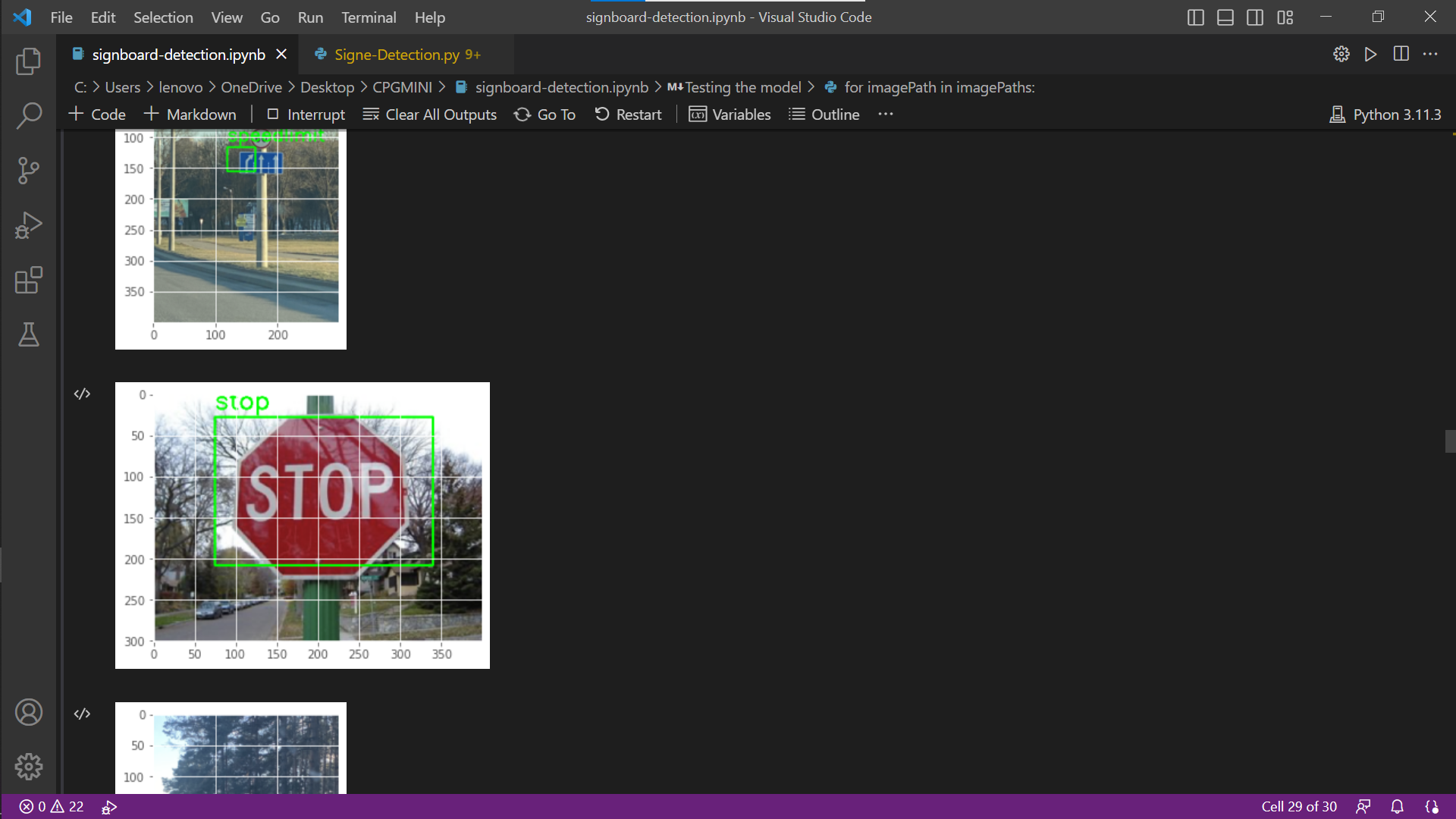
Le code commence par charger les images et les annotations, normaliser les pixels des images, convertir les étiquettes en format one-hot encoding et diviser les données en ensembles d'entraînement et de test.

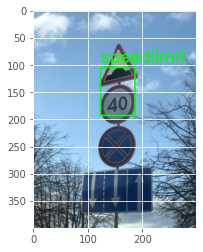
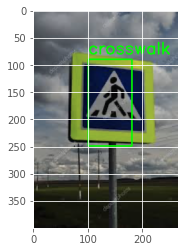
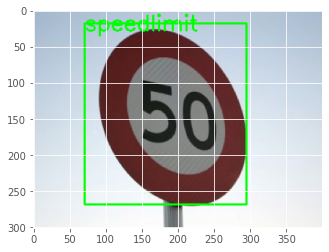
Le modèle utilisé est le VGG16, un réseau de neurones convolutionnel pré-entraîné sur des millions d'images. On congelé toutes les couches sauf la dernière couche de convolution, qu'on ajoute une tête de réseau composée de couches entièrement connectées pour classer et localiser les objets dans l'image.

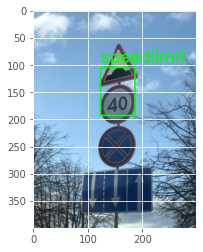
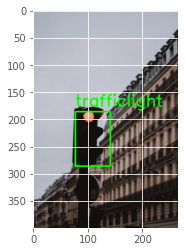
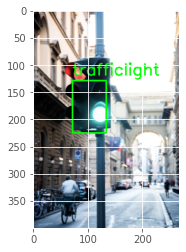
Le modèle est entraîné en utilisant la méthode de rétropropagation pour minimiser la perte, qui est calculée en utilisant à la fois la perte de classification et la perte de régression de boîte englobante. Les poids de ces pertes sont définis par l'utilisateur.

Le modèle est ensuite évalué sur l'ensemble de test et les résultats sont affichés en utilisant des graphiques de perte et de précision. Enfin, le modèle et l'encodage des étiquettes sont sauvegardés pour faire ensuite l’action associé à chaque étiquette.

Voici quelle que teste de modèle sur VS code :





1. **Conclusion**

En somme, notre dossier technique présente en détail notre projet de prototype de voiture autonome équipé d'un système ADAS, exposant les différentes fonctionnalités et technologies employées pour sa réalisation. Nous avons décrit les étapes de notre processus de conception, de modélisation et de fabrication, dans le but de participer à la compétition inter-écoles e-mobility: Smart Drive organisé par Capgemini engineering. Notre objectif est de présenter notre projet, démontrer notre expertise en mécatronique, et acquérir de nouvelles compétences en matière de conception, de modélisation et de programmation, tout en rencontrant d'autres équipes passionnées par la technologie. En fin de compte, notre participation à cette compétition est une occasion pour nous de mettre en valeur notre savoir-faire et de nous faire connaître dans le domaine de la conception et la fabrication de voitures autonomes.