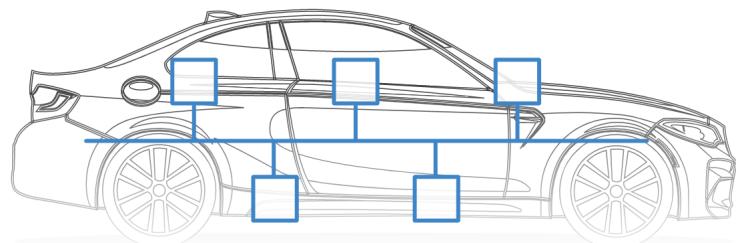


MASTER 1 SME

RAPPORT TER

Bus CAN et apprentissage – CAN bus hacking



Réalisé par :
BOUSADIA Ramzi
DOUKI Thiziri
ZIANE Mouloud

Encadré par :
Mr. BERTHOU Pascal

Table des matières

Introduction	1
Chapitre 1 Contexte théorique	2
1.1 Control Area Network :	2
1.1.1 Historique :	2
1.1.2 Domaines d'application :	2
1.1.3 Le bus CAN :	3
1.1.4 Le Protocole CAN et les couches OSI :	4
1.1.5 Support de transmission :	5
1.1.6 Caractéristiques du Bus CAN :	7
1.1.7 Les informations sur le bus :	8
1.1.8 Rôle des bits dans le champ d'arbitrage :	9
1.2 L'apprentissage Automatique :	9
1.2.1 Méthodes d'apprentissage automatique :	9
1.2.2 Catégorisation sur la base de la sortie requise :	10
Chapitre 2 Projet : Analyse de la manière de conduite	11
2.1 Cahier des charges :	11
2.2 Présentation du projet :	11
2.3 Présentation du système support et du matériels utilisés :	12
Chapitre 3 Étude et conception du système	13
3.1 Phase d'analyse :	13
3.1.1 Identification des acteurs :	13
3.1.2 Les Cas d'utilisations :	13
3.1.3 Diagramme des Cas d'utilisation :	14
3.1.4 Diagramme de séquence :	15

3.1.5	Diagramme de collaboration :	17
3.1.6	Diagramme de Classes :	18
Chapitre 4	Étude économique et gestion du projet	19
4.1	Méthodologie de travail :	19
4.2	Estimation du coût du projet :	21
4.3	Fonctionnement des composants :	21
Chapitre 5	la mise en oeuvre matérielle	25
5.1	Schéma fonctionnel :	25
5.2	Désigne 3D et impression de boîtier	26
Chapitre 6	La mise en oeuvre software	27
6.1	Extraction des signaux utiles d'un message CAN	27
6.1.1	Initialisation du bouclier de bus CAN	28
6.1.2	Récupération des données :	29
6.2	Apprentissage automatique :	30
6.3	Page Web :	32
6.3.1	Environnement de développement :	32
6.3.2	Languages utilisés :	32
6.3.3	Conception de la base de données :	33
Conclusion		35
Bibliographie		37
Annexe A	Annexe	38
A.0.1	Programme principal	38
A.0.2	Désigne 3D	41

Introduction

Dans notre temps actuel on constate une expansion accélérée du secteur automobile associée à l'augmentation du tourisme (véhicules non lourds) et de la quantité de camions sur les routes. Dans la plupart des cas, les voitures de ces deux divisions ont des caractéristiques technologiques avancées, qui incluent des systèmes intelligents embarqués, des systèmes de détection des failles et d'analyse des performances incluant l'efficacité énergétique et les effets environnementaux. Ces avancées ont permis de simplifier les tâches de réparation du véhicule, en cherchant à analyser la manière de conduite, et de centraliser la visualisation des variables en temps réel ; de nos jours, ces valeurs peuvent être lues à partir de l'ECU [Electronic Control Unit].

Dans le cadre de notre première année du cycle master en systèmes et micro-systèmes embarqués, nous implémentons un projet qui consiste en un système basé sur le protocole OBD-II sur un bus CAN [Controller Area Network], qui permet la collection des données en temps réel dans le but de détecter la manière de conduite du conducteur. Le système nous permet de récupérer en temps réel un grand nombre de variables - importantes pour la détection de la manière de conduite - telles que la vitesse, le niveau de carburant et les tours par minutes. Nous avons implanté un système OBD-II avec une carte de développement Arduino Uno connectée à une interface de connexion avec OBD-II. Les données récupérées seront utilisées pour la détection de la manière de conduite en utilisant le Machine Learning.

1

Contexte théorique

1.1 Control Area Network :

1.1.1 Historique :

Depuis les années 1960 la longueur de câble utilisée dans l'automobile ne cesse de croître pour dépasser 2000 m en 1995. Le nombre des connexions atteint 1800 à cette même date. La fiabilité et la sécurité sont menacés. Les normes en matière de pollution et de consommation d'énergie obligent les constructeurs à multiplier les capteurs et actionneurs intelligents dans leur véhicules accélérant ce processus de multiplication des câbles et connexion depuis une vingtaine d'années. Le besoin de sécurité accrue (ABS, ESP, AIR-BAG...) et la demande de confort (mémorisation des réglages de conduite, climatisation régulée par passager, système de navigation...) ne font que renforcer cette tendance. La société BOSCH développe dès le début des années 1980 une solution de multiplexage des informations circulant à bord de la voiture. Le bus CAN apparaîtra et sera normalisé dans les années qui suivent (dès 1983).

Les composants CAN se démocratisent et investissent d'autres secteurs de l'électronique embarqué (médical, produits numériques, systèmes électrotechnique...). [1]

1.1.2 Domaines d'application :

Le CAN est un réseau de terrain (ou réseau embarqué) qui occupe aujourd'hui une position de leader sur le marché automobile. D'autres secteurs utilisent également le bus CAN : véhicules industriels, matériel agricole, bateaux, avions, production industrielle, automatismes, la domotique ... etc



FIGURE 1.1 – Domaines d’application du Bus CAN.

1.1.3 Le bus CAN :

Le bus CAN (Controller Area Network) est un bus série de terrain car il doit fonctionner dans un environnement limité et sévère (milieu industriel, atelier, voiture...) permettant la transmission asynchrone de données numériques. Il s’agit d’un bus multiplexé.

Les particularités de ce bus sont :

- Bus multi maîtres où tous les participants ont les mêmes droits.
- Les nœuds (hôtes) de ce bus ne sont pas adressés seuls les messages transmis possèdent un identificateur. Chaque nœud scrute en permanence le bus et décide selon l’identificateur si le message lui est destiné ou pas.
- Fiabilité élevée des mécanismes de protection du protocole.

Il existe 2 protocoles CAN :

- Le protocole CAN 2.0 A dit CAN standard qui possède un identificateur de 11 bits ce qui permet de délivrer 2048 messages différents.
- Le protocole CAN 2.0 B ou CAN étendu qui possède un identificateur de 29 bits soit plus de 536 millions de messages différents.

Pour chacun des 2 protocoles il existe 2 types d’interfaces (normes de transmission) :

- Le CAN Low Speed ou CAN-LS (norme ISO 11519). Ce bus présente un débit max de 125 kbits/set peut raccorder jusqu’à 20 nœuds. Dans l’automobile, il est appelé

bus confort et utilisé pour la climatisation, la radio, le tableau de bord ... etc.

- Le CAN High speed ou CAN-HS. Ce bus présente un débit max de 1 Mbits/set peut raccorder jusqu'à 30 nœuds. Dans l'automobile, il est utilisé pour les équipements de sécurité (freinage, moteur ...etc.).

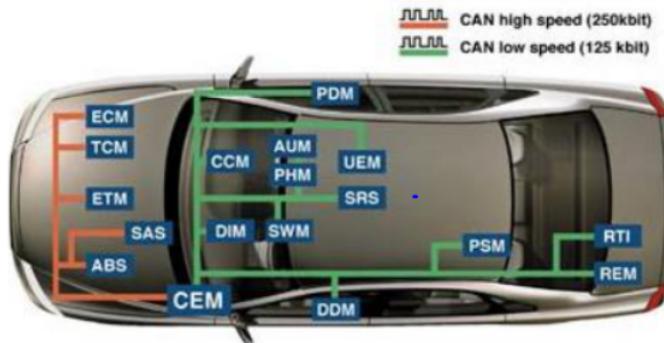


FIGURE 1.2 – Réseau CAN du véhicule

1.1.4 Le Protocole CAN et les couches OSI :

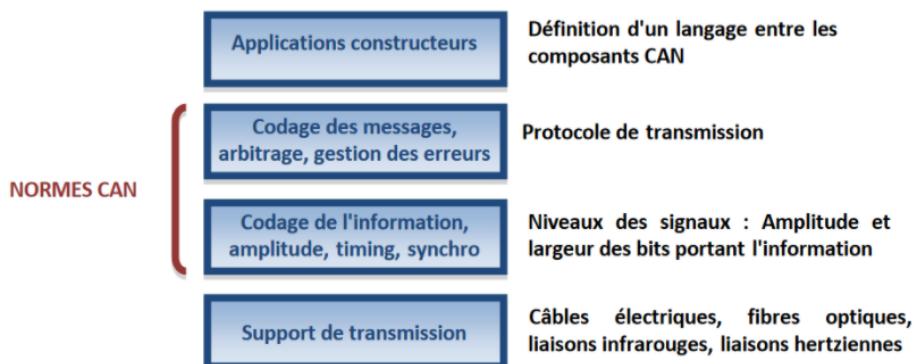


FIGURE 1.3 – Les couches CAN

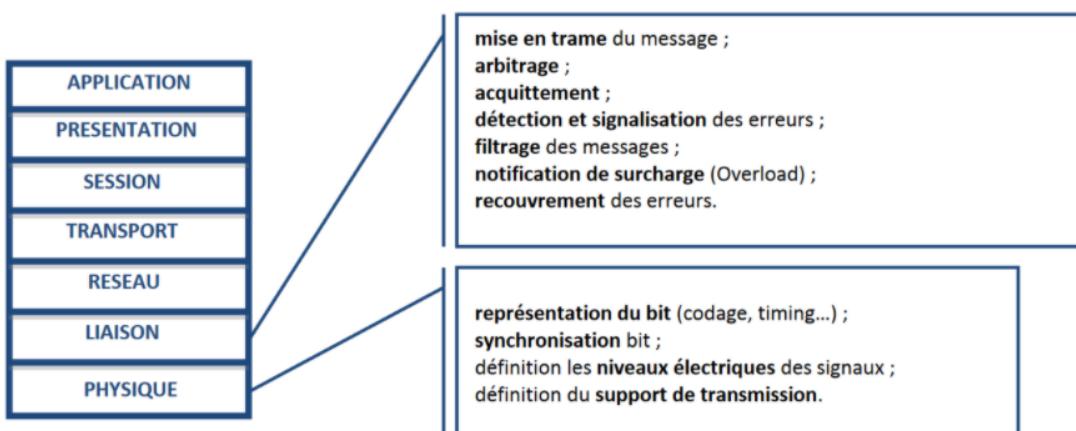
La structure du protocole du bus CAN possède implicitement les principales propriétés suivantes :

- Hiérarchisation des messages.
- Garantie des temps de latence.
- Souplesse de configuration.
- Réception de multiples sources avec synchronisation temporelle.

- Fonctionnement multimètre.
- Détections et signalisations d'erreurs..
- Retransmission automatique des messages altérés dès que le bus est de nouveau au repos.
- Distinction d'erreurs (temporaire ou non-fonctionnalité permanente au niveau d'un nœud).
- Déconnexion automatique des nœuds défectueux.

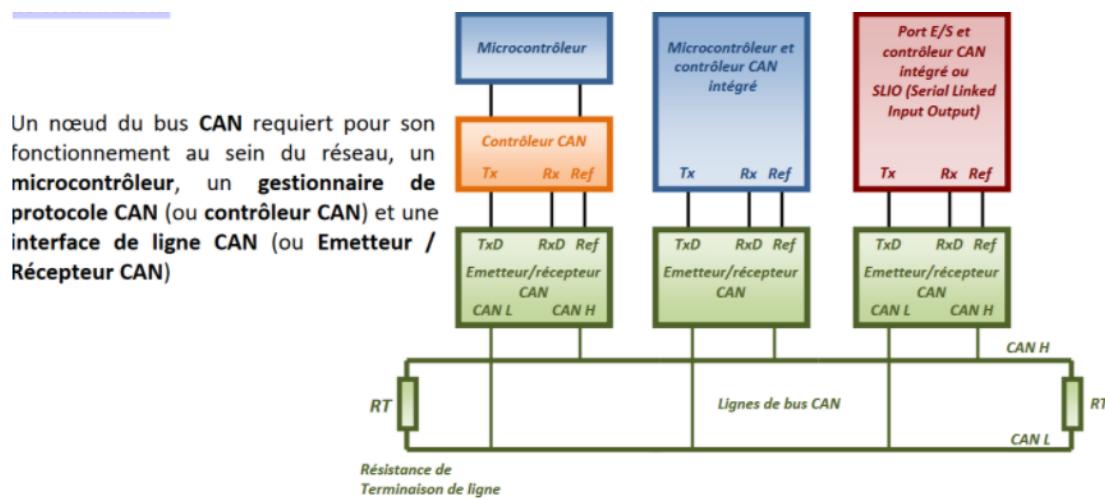
Le protocole CAN ne couvre seulement que deux des sept couches du modèle d'interconnexion des systèmes ouverts OSI.

On retrouve ainsi dans le protocole CAN, la couche liaison de données (couche 2) et la couche physique (couche 1).



1.1.5 Support de transmission :

La transmission des données est effectuée sur une paire filaire différentielle torsadée. La ligne est donc constituée de deux fils : CAN L (CAN Low) et CAN H (CAN High). Le Signal CAN transmis est donc obtenu par la différence de tension entre les deux lignes. La ligne du bus doit se terminer par des résistances de terminaison.[\[2\]](#)



Paramètres	CAN LS	CAN HS
Débit	125 kb/s	125 kb/s à 1 Mb/s
Nombre de nœuds sur le bus	2 à 20	2 à 30
Courant de sortie (mode émission)	> 1 mA sur 2,2 kΩ	25 à 50 mA sur 60Ω
Niveau dominant (NL0)	CAN H = 4V CAN L = 1V	CAN H = 3,5 V CAN L = 1,5 V
Niveau récessif (NL1)	CAN H = 1,75V CAN L = 3,25V	CAN H = 2,5 V CAN L = 2,5 V
Caractéristique du câble	30 pF entre les câbles de ligne	2 × 120 Ω
Tensions d'alimentation	5 V	5 V

FIGURE 1.4 – Comparaison entre CAN_L et CAN_H.

Les noeuds sont câblés sur le bus de telle manière qu'en cas d'émission simultanée de deux noeuds, le NL0 s'impose par rapport au NL1 : Le NL0 est donc appelé état dominant et le NL1, état récessif.

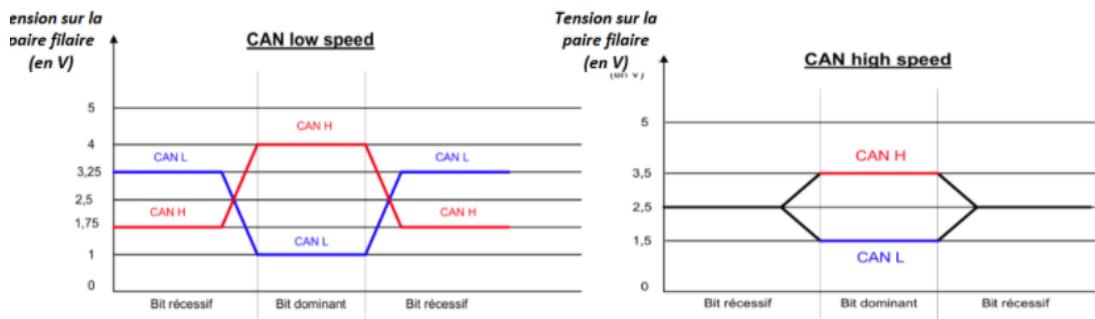


FIGURE 1.5 – Comparaison de vitesse de transmission entre CAN_L et CAN_H .

La transmission différentielle du signal sur le bus CAN assure l'immunité électromagnétique car les deux lignes du bus sont affectées de la même manière par un signal

perturbateur. La longueur du bus et le débit maximum sont liés par la courbe suivante :

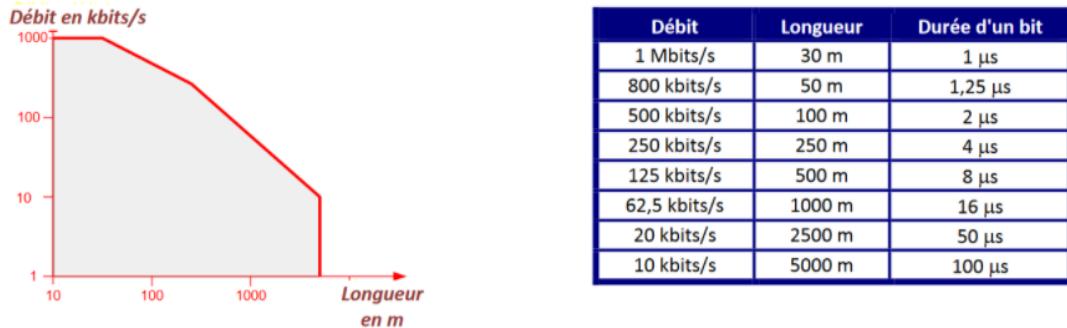
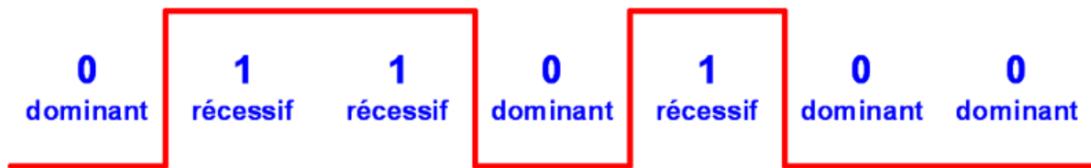


FIGURE 1.6 – Débit de transmission en fonction de la longueur .

1.1.6 Caractéristiques du Bus CAN :

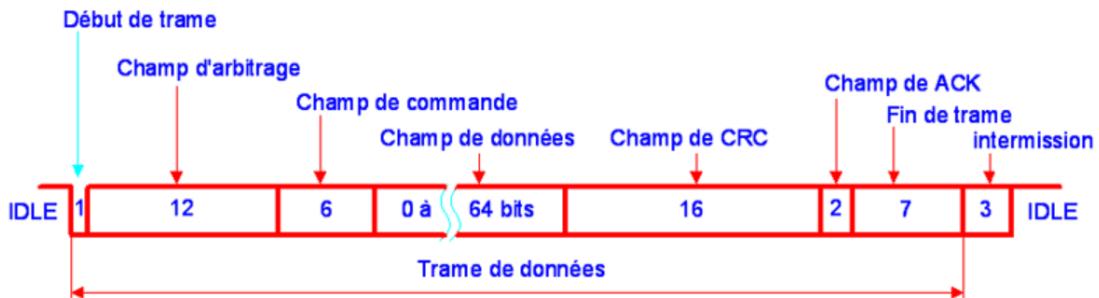
Le codage NRZ : bits dominants et récessifs :

La succession de bits transitant sur le bus est codé avec la méthode du NRZ (Non Return To Zero). Pendant la durée totale du bit, le niveau de tension de la ligne est maintenu, c'est à dire que pendant toute la durée durant laquelle un bit est généré, sa valeur reste constante qu'elle soit dominante ou récessive.



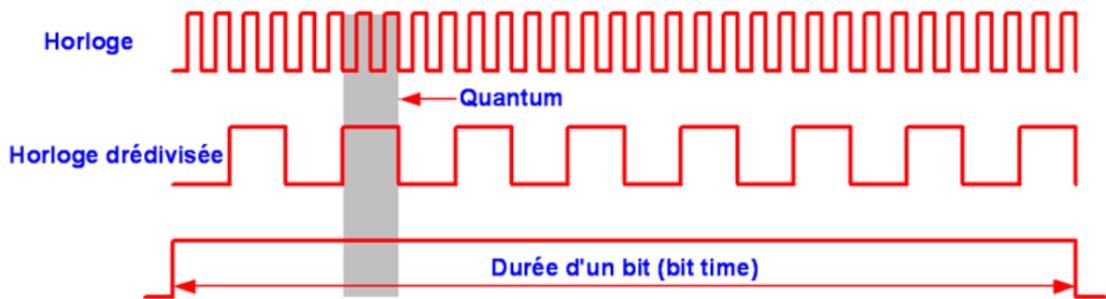
Le bit stuffing :

Une des caractéristiques du codage NRZ est que le niveau du bit est maintenu pendant toute sa durée. Cela pose des problèmes de fiabilité si un grand nombre de bits identiques se succèdent. La technique du Bit Stuffing impose au transmetteur d'ajouter automatiquement un bit de valeur opposée lorsqu'il détecte 5 bits consécutifs dans les valeurs à transmettre.



Le bit timing :

On définit la plus petite base de temps reconnue sur un bus CAN comme étant le Time Quantum. Cette base de temps est une fraction de l'horloge de l'oscillateur du bus. Un bit dure entre 8 et 25 quantum.



1.1.7 Les informations sur le bus :

Trame de données (data frame)

La trame de données standard CAN 2.0A (la plus utilisée) se décompose en 7 parties champs :

- Début de trame (1 bit dominant) : Start Of Frame (SOF).
- Champ d'arbitrage (12 bits) : arbitration field.
- Champ de commande (6 bits) : control field.
- Champ de données (0 à 64 bits) : data field.
- Champ de CRC (16 bits) : CRC séquence.
- Champ d'acquittement (2 bits) : ACKnowledgement field.
- Fin de trame (7 bits) : End of Frame (EOF).

On trouve ensuite une 8ème zone dite espace inter trame (interframe) qui appartient à la trame.



FIGURE 1.7 – La trame de données.

1.1.8 Rôle des bits dans le champ d'arbitrage :

Le bit SOF est dominant. Il signale à toutes les stations le début d'un échange. Cet échange ne peut démarrer que si le bus était précédemment au repos.

Identificateur La longueur de l'identificateur est de 11 bits, les bits sont transmis dans l'ordre de ID_10 à ID_0 (le moins significatif est ID_0).

Le bit RTR Lors d'une dataframe, le bit de remote transmission request (RTR) doit être dominant.

1.2 L'apprentissage Automatique :

L'apprentissage automatique est une application de l'intelligence artificielle qui offre aux systèmes la possibilité d'apprendre et de s'améliorer automatiquement à partir de l'expérience sans être explicitement programmés. L'apprentissage automatique se concentre sur le développement de programmes informatiques qui peuvent accéder aux données et les utiliser pour apprendre par eux-mêmes.

Le processus d'apprentissage commence par des observations ou des données, telles que des exemples, une expérience directe ou des instructions, afin de rechercher des modèles dans les données et de prendre de meilleures décisions à l'avenir sur la base des exemples que nous fournissons. L'objectif principal est de permettre aux ordinateurs d'apprendre automatiquement sans intervention humaine ou d'assistance et d'ajuster les actions en conséquence. [3]

1.2.1 Méthodes d'apprentissage automatique :

Les implémentations d'apprentissage automatique sont classées en plusieurs catégories suivant la nature du «signal» ou de la «réponse» d'apprentissage disponible :

- **Apprentissage supervisé** : Consiste à appliquer ce qui a été appris dans le passé à de nouvelles données en utilisant des exemples annotés pour prédire les événements futurs.
- **Apprentissage non supervisé** : Est utilisé quand les données ne sont pas étiquetées. Il étudie comment les systèmes peuvent déduire une fonction pour décrire une structure cachée à partir de données non étiquetées.
- **Apprentissage par renforcement** : Consiste à interagir avec son environnement en produisant des actions et en découvrant des erreurs ou des récompenses. Cette méthode permet aux machines de déterminer automatiquement le comportement idéal dans un contexte spécifique afin de maximiser ses performances.
- **Apprentissage semi-supervisé** : Est quelque part entre l'apprentissage supervisé et non supervisé, car ils utilisent à la fois des données étiquetées et non étiquetées pour la formation - généralement une petite quantité de données étiquetées et une grande quantité de données non étiquetées.

1.2.2 Catégorisation sur la base de la sortie requise :

Une autre catégorisation des tâches d'apprentissage automatique se produit lorsque l'on considère la sortie souhaitée d'un système appris par machine :

- **Classification** : Est lorsque les entrées sont divisées en deux classes ou plus, et que la machine doit produire un modèle qui attribue des entrées invisibles à une ou plusieurs de ces classes. Ceci est généralement abordé de manière supervisée.
- **Régression** : Est aussi supervisé, dans le cas où les sorties sont continues plutôt que discrètes.
- **Clustering** : Est lorsqu'un ensemble d'entrées doit être divisé en groupes. Contrairement à la classification, les groupes ne sont pas connus à l'avance, ce qui en fait généralement une tâche non supervisée.

2

Projet : Analyse de la manière de conduite

2.1 Cahier des charges :

Le projet consiste à réaliser un système pour déterminer la manière de conduite d'un conducteur, en prenant en compte quelques données qu'on récupère sur le calculateur du véhicule à l'aide de la prise OBD II. Notre système doit fonctionner de la manière suivante :

- Affichage, sur un écran LCD, si la connexion avec le véhicule est réussie, ainsi que l'initialisation de CAN.
- Récupérer les données (Vitesse, RPM, Niveau de carburant, et l'heure) à temps réel.
- Stocker les données dans l'ordinateur (dans une base de données)
- Affichage des données sur une page Web.
- Détection de la manière de conduite en utilisant le Machine Learning .
- Affichage de la manière de conduite sur un écran LCD.

2.2 Présentation du projet :

Le projet consiste en la récupération de données, en temps réel, à partir des calculateurs d'un véhicule. Celles-ci nous permettront par la suite de déterminer la manière de conduite du conducteur, par exemple : conduite économique, conduite souple ... etc. Cela étant dans le but d'aider le conducteur à faire des économies de carburant et à la protection du véhicule et de lui-même.

2.3 Présentation du système support et du matériels utilisés :

- Micro-contrôleur : Arduino UNO
- interface de connexion entre OBD II et la carte : Arduino Shield - CAN-BUS V2, MCP2515 / MCP2551
- écran LCD
- Cable OBDII vers DB9

3

Étude et conception du système

3.1 Phase d'analyse :

3.1.1 Identification des acteurs :

On peut identifier un acteur utilisateur du système :

- L'utilisateur : La personne qui conduit de véhicule.

3.1.2 Les Cas d'utilisations :

- Connexion Système :
 - Démarrage et arrêt
 - Connexion OBD
- Lecture des données :
 - Lecture de la vitesse.
 - Lecture des Tours par minute.
 - Lecture du niveau du carburant
- Calcule des données :
 - Calcule accélération
 - Calcule kilométrage
 - Calcule consommation du carburant
- Affichage des données :
 - Sauvegarde accélération
 - Sauvegarde vitesse
 - Sauvegarde tours par minute

- Sauvegarde kilométrage
- Sauvegarde consommation du carburant
- Détection de la façon de conduite
- Détection façon de conduite

3.1.3 Diagramme des Cas d'utilisation :

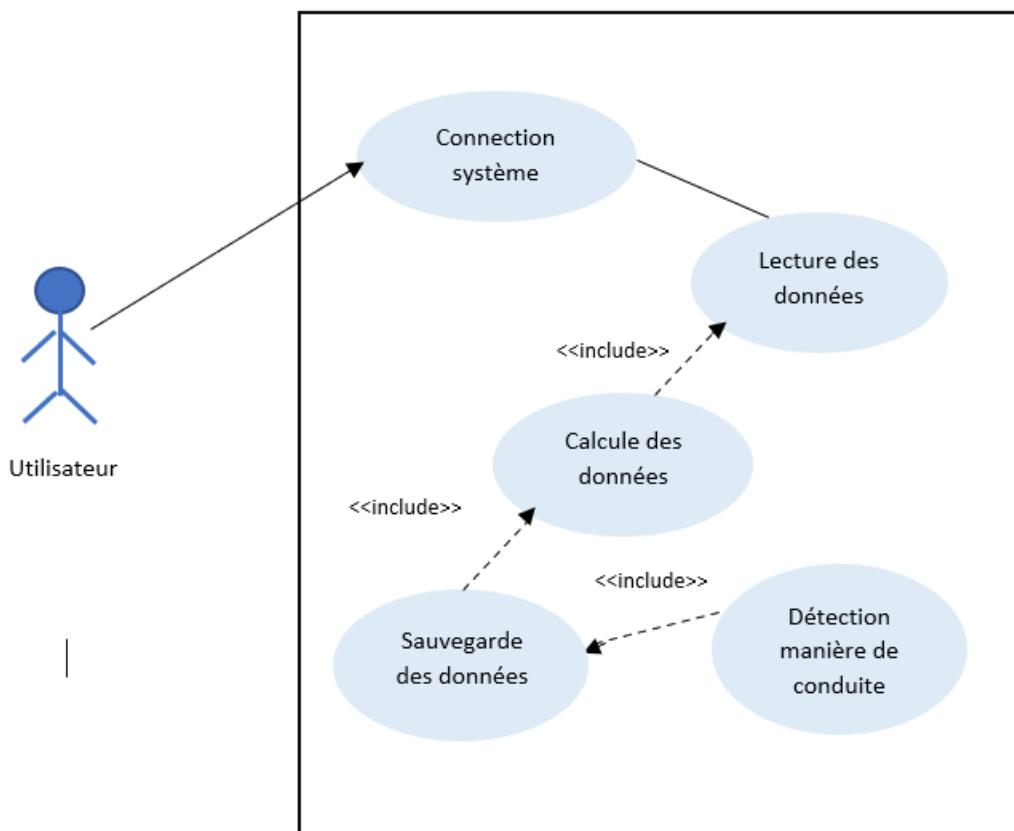


FIGURE 3.1 – Diagramme des Cas d'utilisation.

3.1.4 Diagramme de séquence :

Diagramme de Séquence pour les cas "Connexion Système" :

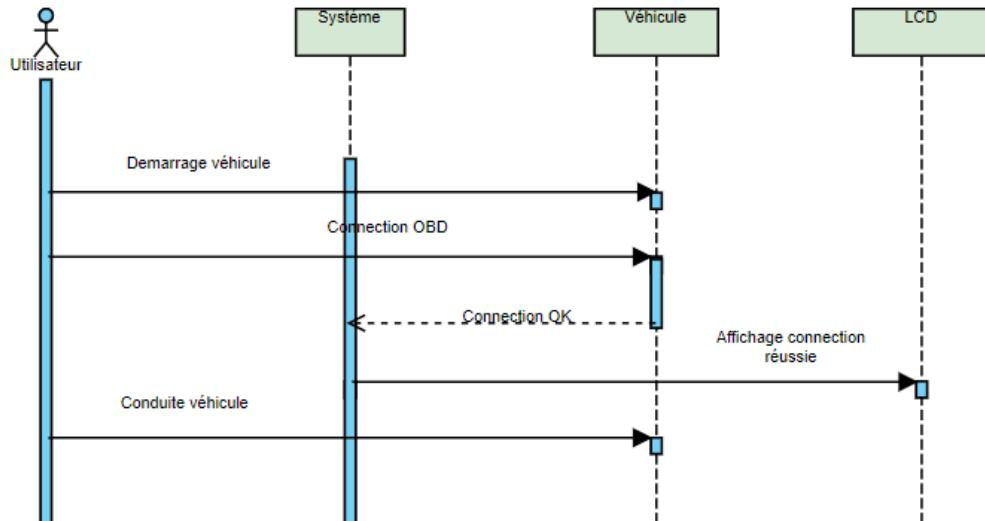


FIGURE 3.2 – Diagramme de séquence "Connexion Système"

Diagramme de Séquence pour le cas "Détection façon de conduire" :

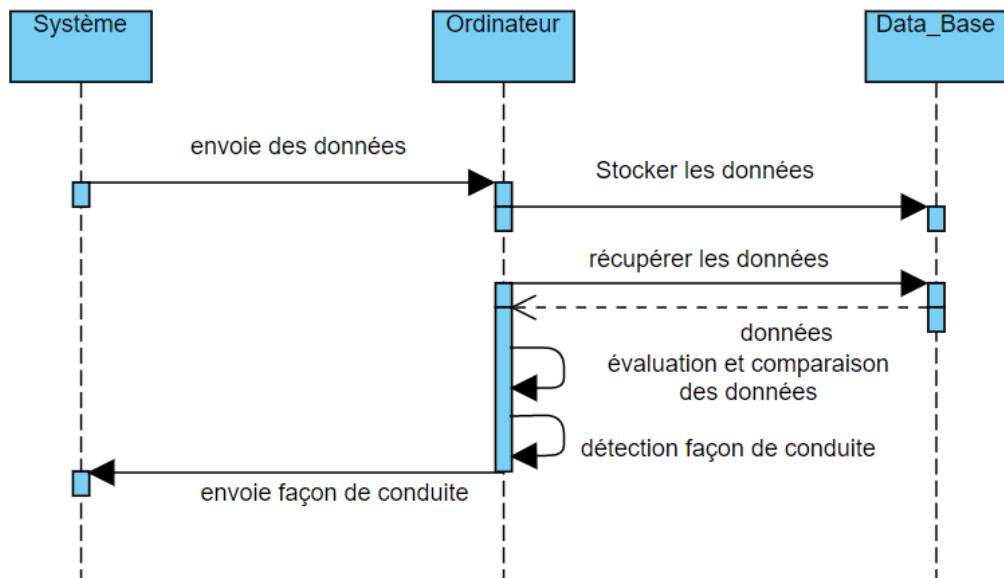


FIGURE 3.3 – Diagramme de séquence "Détection façon de conduire".

**Diagramme de Séquence pour les cas "Lecture, Calcule et Sauvegarde des données"
"Calcule des données" :**

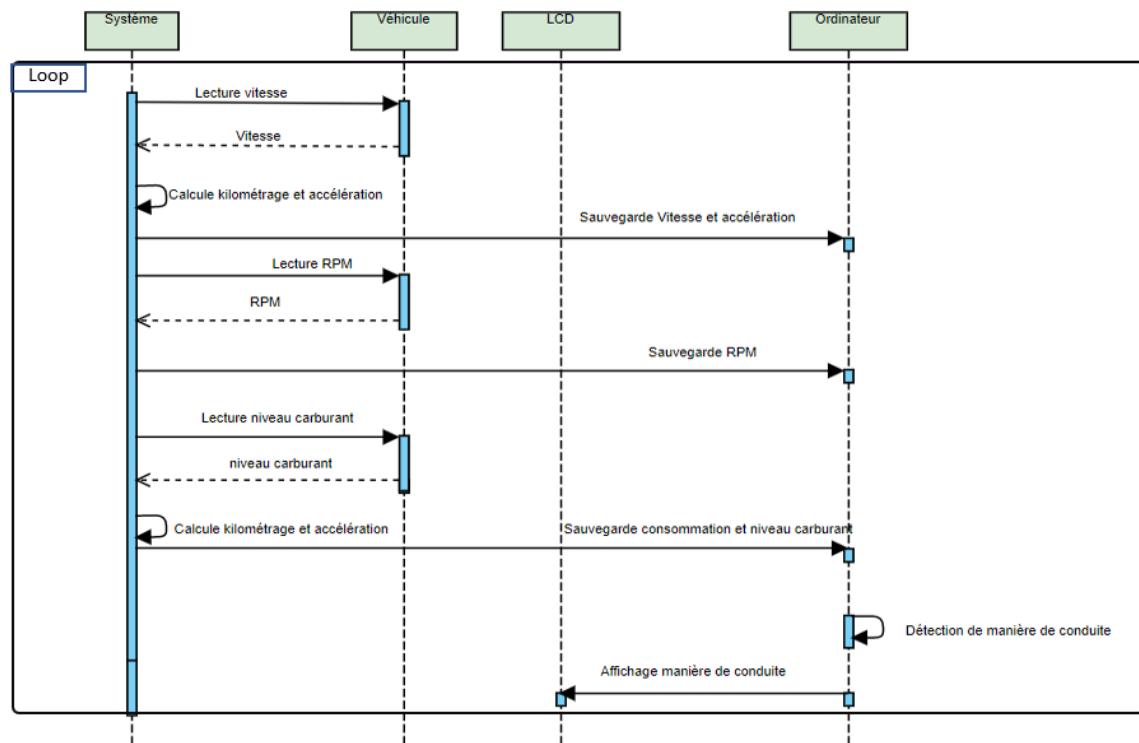


FIGURE 3.4 – Diagramme de séquence "Lecture, Calcule et Sauvegarde des données".

3.1.5 Diagramme de collaboration :

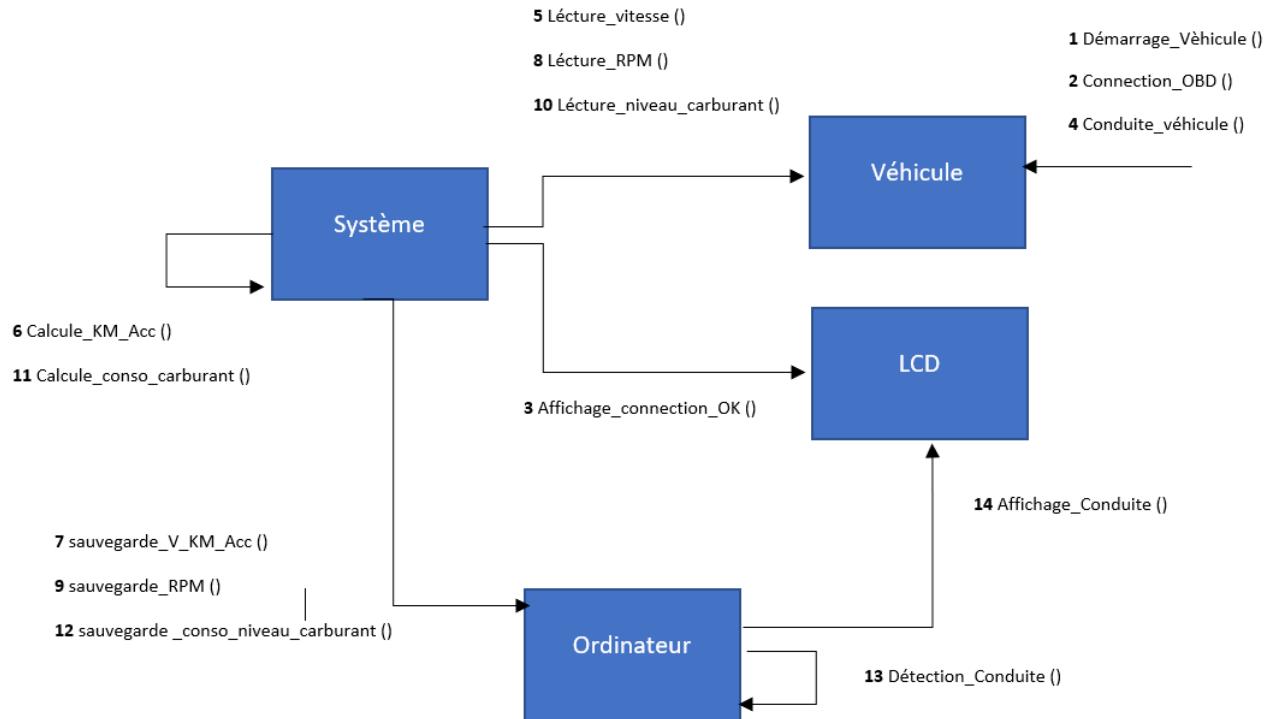


FIGURE 3.5 – Diagramme de collaboration .

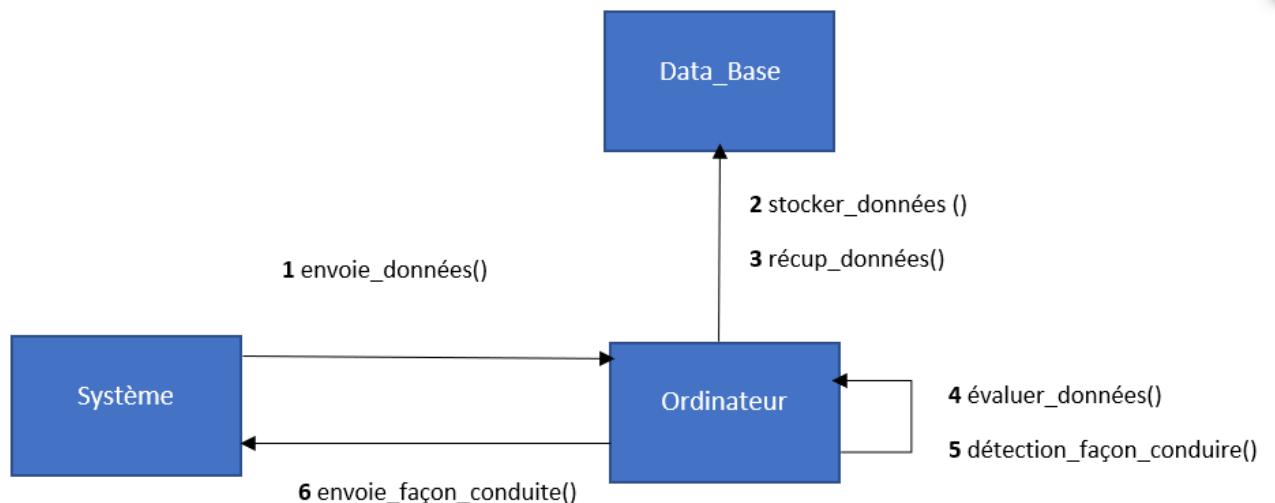


FIGURE 3.6 – Diagramme de collaboration "détection de la façon de conduire"

3.1.6 Diagramme de Classes :

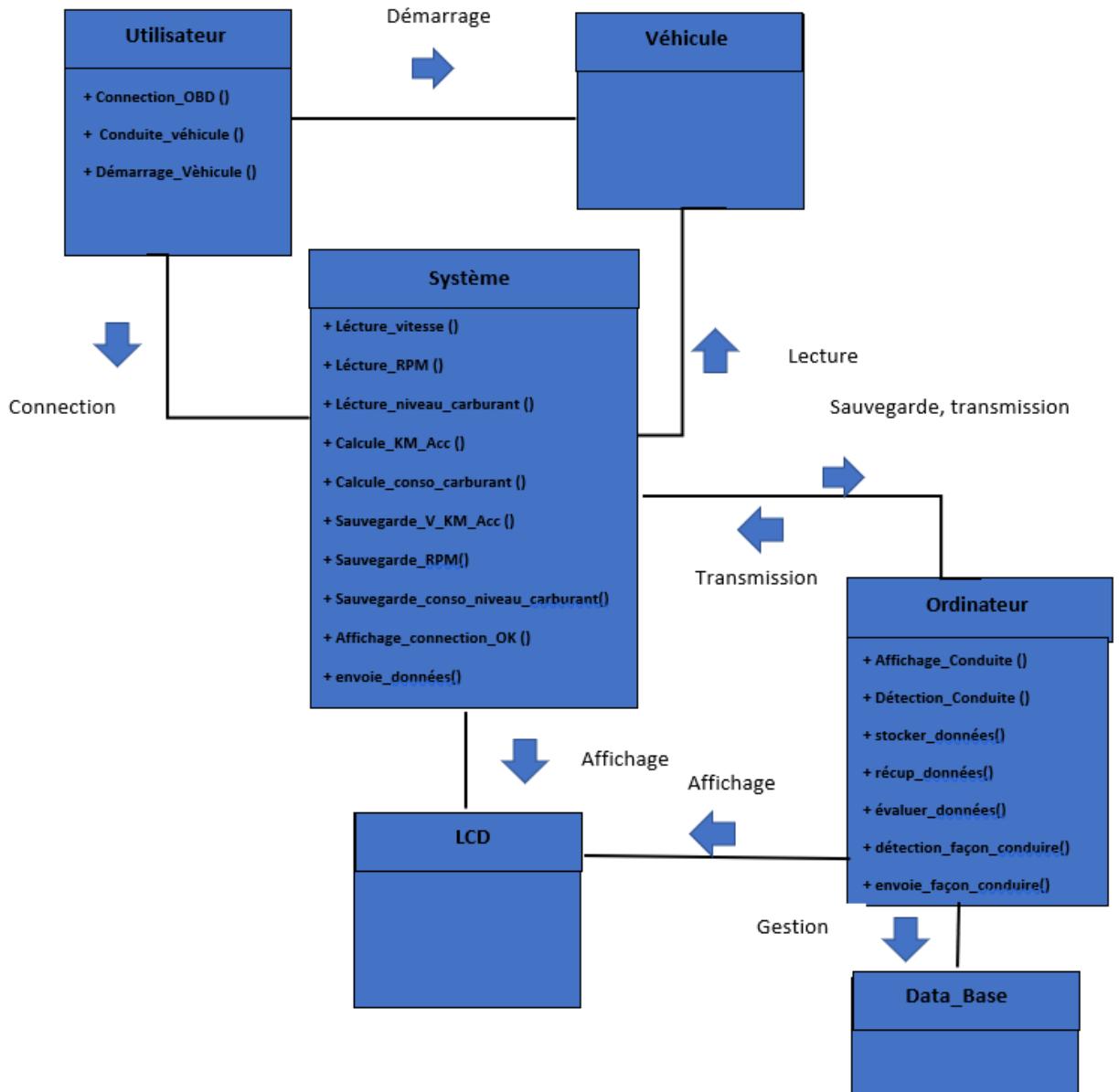


FIGURE 3.7 – Diagramme de Classes.

4

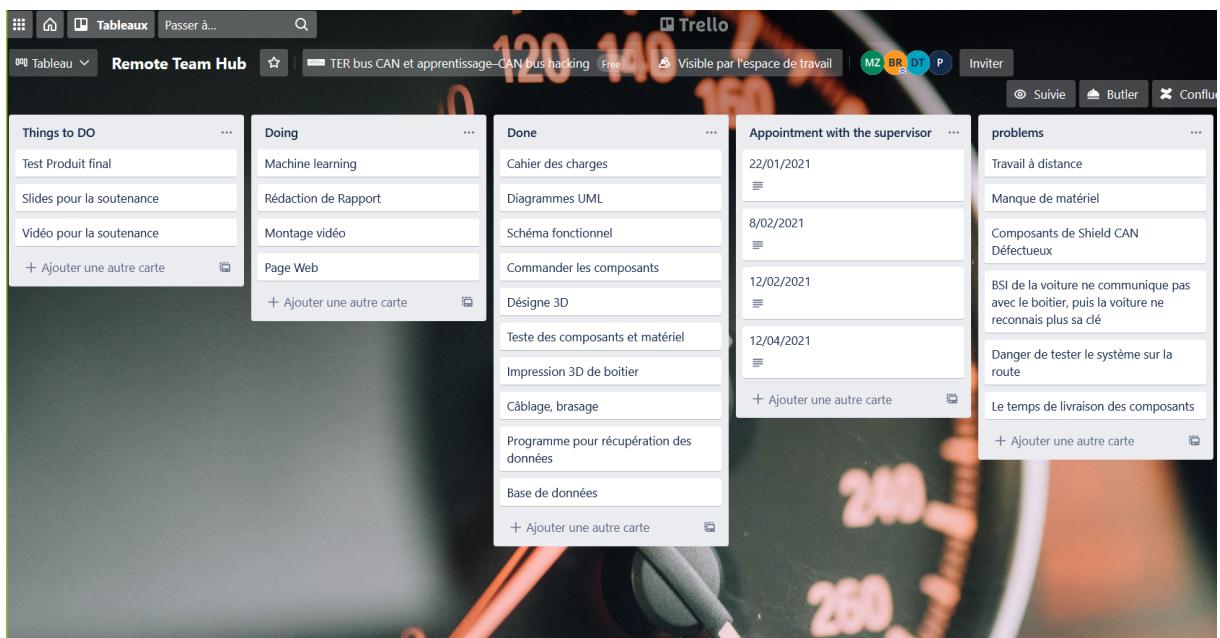
Étude économique et gestion du projet

4.1 Méthodologie de travail :

La réalisation du projet s'est faite en suivant les étapes suivantes :

- Mise en place de l'état de l'art.
- Commande du matériels nécessaires.
- Teste des composants.
- Câblage du système.
- Programmation du système.
- Création de la base de données et de la page web.

Pour l'organisation et la communication avec l'encadrant on a utilisé l'outil Trello. On a établi, dessus, la liste des rendez-vous ainsi que l'ensemble des taches qu'on a réalisé ou qu'il nous reste à réaliser.



Autres outils utilisés :



4.2 Estimation du coût du projet :

Composant	Référence	Prix
Arduino	Arduino Uno Rev.3 ATmega328 USB	22.90€
LCD	Grove - 16x2 LCD (White on Blue)	5,95 €
Arduino Shield CAN V2	Arduino Shield - CAN-BUS V2 MCP2515/MCP2551	21.07€
Cable OBDII vers DB9		15€
	Totale :	68,92€

TABLE 4.1 – Cout des composants.

4.3 Fonctionnement des composants :

Arduino Uno :

Une carte microcontrôleur basée sur ATMega328 de Atmel, c'est la plus simple et la plus économique carte à microcontrôleur d'Arduino.



FIGURE 4.1 – Arduino Uno.

La carte dispose de :

- Microcontrôleur ATMega328
- 16 MHz - Oscillateur à quartz
- Port USB
- Variante DIP/DIL
- Une fonction de réinitialisation
- Un en-tête ICSP (programmation in-situ)

Données techniques :

- Tension de fonctionnement : 5 V
- Tension d'entrée : de 7 à 12 V
- Tension d'entrée (limite) : de 6 à 20 V
- Entrées / sorties numériques : 14 (dont 6 peuvent être utilisées comme sortie PWM)
- Entrées analogiques : 6
- I c.c. 40 mA (E/S) / 50 mA (3,3 V)
- Mémoire flash : 32 Ko
- SRAM : 2 Ko
- EEPROM : 1 Ko
- Bus série (CAN), I2C et SPI

Arduino Shield - CAN-BUS V2, MCP2515 / MCP2551

Permet de connecter de multiples capteurs, calculateurs et autres modules, sans multiplier les câbles de connexion grâce à la technologie du multiplexage. Il est particulièrement utile en tant qu'outil diagnostique dans la conception automobile.



FIGURE 4.2 – Arduino Shield.

Le shield dispose de :

- Un contrôleur CAN-BUS MCP2515
- Emetteur-récepteur CAN MCP2551

Données techniques :

- Implémente CAN V2.0B avec jusqu'à 1 Mb/s
- Connecteur Sub-D à 9 broches selon la norme de l'industrie

- Affectation des broches standard OBD-II et CAN sélectionnable
- Goupille CS interchangeable pour slot pour carte TF
- Bornier à vis facile à connecter CAN_H et CAN_L
- 2 connecteurs (I2C et UART)
- Interface SPI jusqu'à 10 MHz

Remarque : pendant la mise en œuvre le MCP2515 est grillé, donc un ami qui travaille à Continental nous a fourni les composants nécessaires pour continuer la réalisation de notre projet.

Grove - 16x2 LCD

Affiche 2 lignes de 16 caractères et est compatible Arduino ou Seeeduino



FIGURE 4.3 – Ecran LCD

Le LCD dispose de :

- Une interface I2C
- Supporte les polices anglaises et japonaises

Données techniques :

- Tension de fonctionnement : 3.3V / 5V
- I2C Adresse : 0X3E
- Méthode de conduite : 1/16 duty, 1/5 bias
- Dimensions : 84 x 45 x 13 mm
- Indications par leds
- Affichage : LCD 2 x 16 caractères

Câble OBD2 vers DB9

Permet de récupérer les signaux d'un port ODB-II afin de les restituer sur un connecteur SUB-D 9 broches femelle.



FIGURE 4.4 – Câble OBD2 vers DB9

Le Cable OBD2 vers DB9 dispose des broches suivantes :

Description des broches	OBDII	DB9
J1850 BUS+	2	7
Chassis Ground	4	2
Signal Ground	5	1
CAN High J-2284	6	3
ISO 9141-2 K Line	7	4
J1850 BUS-	10	6
CAN Low J-2284	14	15
ISO 9141-2 L Line	15	8
Battery Power	16	9

TABLE 4.2 – Tableau des broches.

5

la mise en oeuvre matérielle

5.1 Schéma fonctionnel :

Après l'étude détaillée du système avec le langage UML, nous proposons le schéma fonctionnel suivant qui regroupe la partie électronique et informatique de notre réalisation.

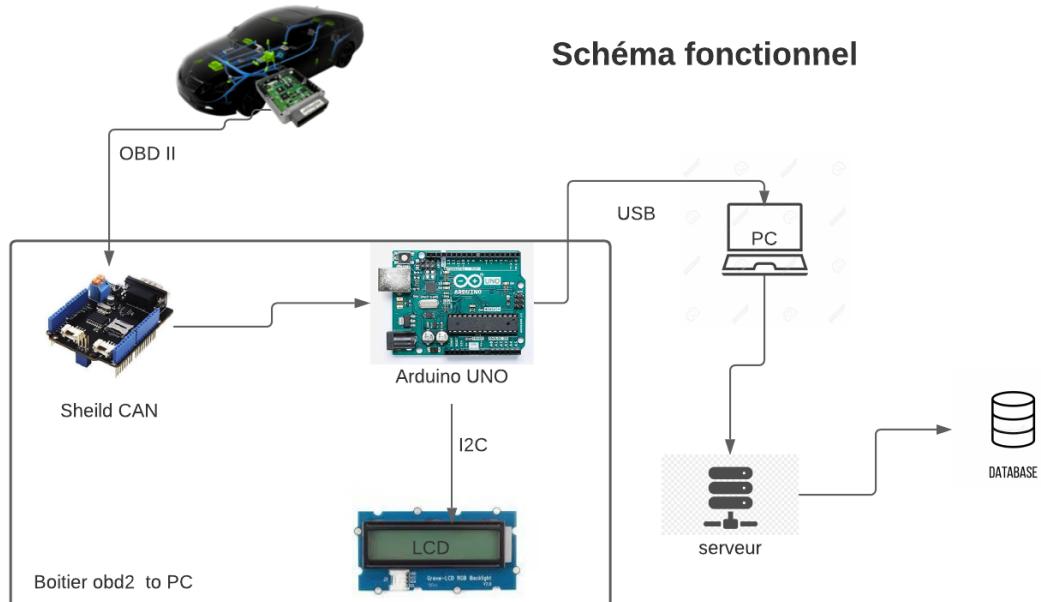


FIGURE 5.1 – Schéma fonctionnel

5.2 Désigne 3D et impression de boîtier

Afin d'assembler l'ensemble des composants, nous avons décidé d'imprimer un boîtier. En premier temps nous avons crée un modèle 3D qui a passé par plusieurs versions dans le but d'avoir un boîtier le plus compacte possible à 1 mm près, En fin nous avons arrivé au modèle ci-dessous :

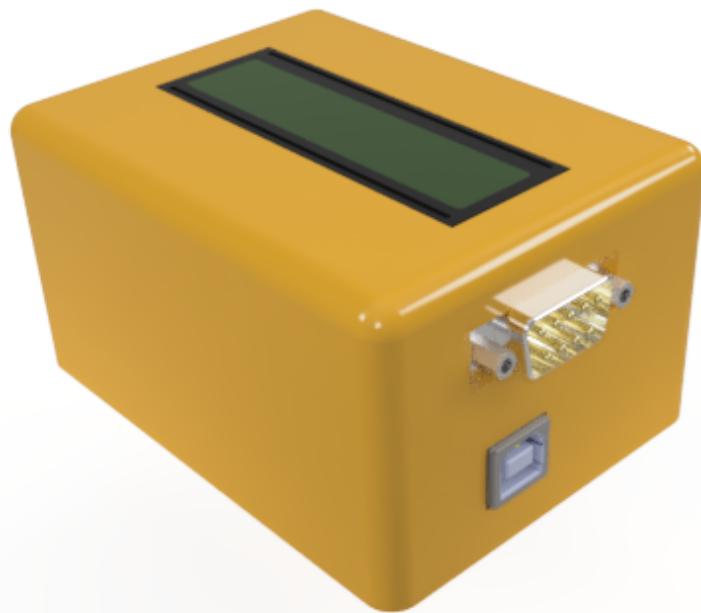


FIGURE 5.2 – Prototype version final

Après la conception, nous avons imprimé notre prototype à l'aide d'une imprimante 3D.

6

La mise en oeuvre software

6.1 Extraction des signaux utiles d'un message CAN

CAN est un protocole basé sur les messages, ce qui signifie que les messages et leur contenu sont plus importants que l'ECU de l'expéditeur lui-même. Ainsi, les ECU ne reçoivent pas d'identifiant, mais chaque message a un identifiant unique dans un bus spécifique. Ces ID sont responsables de la définition de la priorité des messages dans le cas où deux ou plusieurs ECU essaient de mettre leurs messages sur le bus. L'ID de valeur décimale.

Compte tenu de cela, le message avec ID = 0x05 à plus de priorité que notre message avec ID = 0Xf1, nous pourrions supposer que le message qui informe d'un problème fatal dans le moteur aura la priorité la plus élevée. Ainsi, quel que soit l'ECU qui essaie d'envoyer ce message, gagnera le bus et continuera à envoyer son message pendant que tout le monde écoute juste jusqu'à ce qu'il se termine. À tout moment, chaque nœud CAN BUS voit le message envoyé via le bus. Mais tous ne le lisent pas et ne l'envoient pas à leurs ECU.

Nous filtrons le flux de messages et lisons uniquement ceux qui nous intéressent. Et si nous voulions échanger des signaux informatifs et simuler ce qui pourrait se passer dans bus CAN du point de vue de l'ECU du tableau de bord ! Parce que ce serait une perte de temps, et de bits en fait, si on utilise les 8 octets de données pour ne représenter qu'une seule information, les voitures peuvent contenir de nombreux signaux dans un seul message, comme le montre l'image ci - dessous , nous indiquant que pour le message de l'ID = 0xF1 :

Message ID 0xF1

7	6	5	4	3	2	1	0	bits bytes
1	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1	1
1	1	1	0	1	1	1	0	2
								3
								4
								5
								6
								7

	Speed
	RPM
	Check Engine Light
	Oil Indicator
	Not used

FIGURE 6.1 – Message Id 0xF1

-Le premier octet entier représente la vitesse de la voiture avec une valeur maximale de 255 mph.

-Les 14 bits suivants représentent le régime du moteur jusqu'à 16 383 tr / min. Juste pour pouvoir travailler avec une voiture de F1.

-Le bit suivant indique au tableau de bord d'allumer / d'éteindre l'indicateur de vérification du moteur.

-Le bit suivant indique si le niveau d'huile est en dessous d'un seuil ou non pour allumer / éteindre son indicateur. Le reste des bits ne sont pas utilisés dans notre message inventé.

6.1.1 Initialisation du bouclier de bus CAN

L'initialisation du bouclier sera requise pour toutes les tâches. Ici, nous définissons notre débit CAN . Chaque véhicule peut utiliser des vitesses de débit différentes. Pour notre cas, nous utilisons 500 kbps.

```
//tries to initialize, if failed --> it will loop here for ever
START_INIT:
{
    if(CAN_OK == CAN.begin(CAN_500Kbps))           //setting CAN baud rate to 500Kbps
    {
        Serial.println("CAN BUS Shield init ok!");
    }
    else
    {
        Serial.println("CAN BUS Shield init fail");
        Serial.println("Init CAN BUS Shield again");
        delay(100);
        goto START_INIT;
    }
}
```

FIGURE 6.2 – Initialisation du bouclier de bus CAN

6.1.2 Récupération des données :

Pour la première information, car elle tient dans un octet et notre code nous permet de traiter chaque octet du tampon séparément, nous devons extraire la vitesse de la voiture directement à partir du premier octet seulement.

```
//for the first signal: car speed
car_speed = buf[0];

Serial.print("Car speed = ");
Serial.println(car_speed);
```

FIGURE 6.3 – La base de données.

Pour les autres cas où les bits de signal prennent plus ou moins d'un octet, on aura besoin d'une manipulation de bits pour mettre ces bits dans la bonne configuration avant de les lire.

Pour lire le RPM qui se trouve dans deux octets différents (1 et 2), nous procéderons comme suit :

Tout d'abord, nous lisons l'octet supérieur (octet 1) dans un entier de deux octets, il se trouvera dans l'octet inférieur de engine-rpm.

Ensuite, nous décalons cette valeur de 8 bits vers la gauche pour mettre l'octet le plus élevé.

Puis, nous masquons tous les octets 2 bits à l'exception des six premiers bits utilisés pour le signal RPM.

Nous devons simplement ajouter l'octet le plus élevé à l'octet inférieur et obtenir notre valeur finale.

```
//for the second signal: engine RPM
engine_rpm = buf[1];
engine_rpm = engine_rpm << 8;
temp = buf[2] & 0x3F;
engine_rpm = engine_rpm + temp;

Serial.print("Engine RPM = ");
Serial.println(engine_rpm);
```

FIGURE 6.4 – Régime moteur

6.2 Apprentissage automatique :

Dans notre cas, nous faisons face à un problème de modélisation prédictive, on peut prédire la manière de conduire étant donné des exemples de données d'entrée. On conclue donc que la méthode d'apprentissage appropriée est **l'apprentissage supervisé**.

On utilisera le **Multi-Layer Perceptrons** qui est un réseau de neurones artificiels propagation avant (feedforward artificial neural network) qui mappe un ensemble de données d'entrée dans un ensemble de sortie appropriée. Il se compose de plusieurs couches de nœuds dans un graphe orienté avec chaque couche entièrement connectée à la suivante.

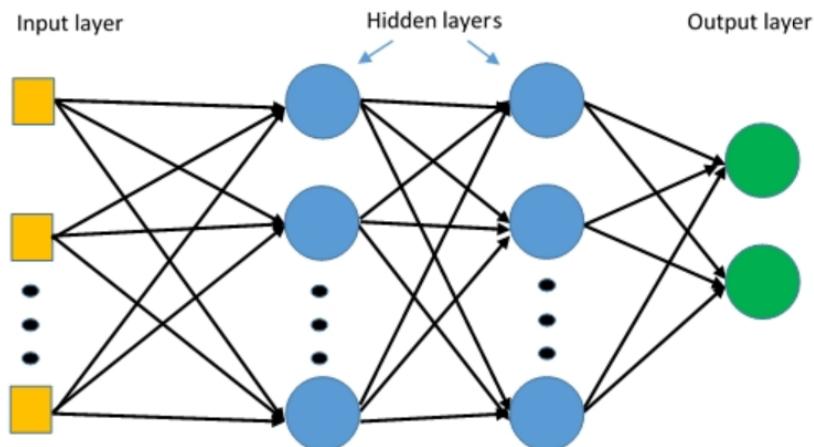


FIGURE 6.5 – Les couches dans le Multi-Layer Perceptrons.

Il apprend une fonction $F : R^n \rightarrow R^o$ en entraînant un jeu de données où m est le nombre de dimensions pour l'entrée et o le nombre de dimensions de la sortie

Dans notre cas nous avons 3 entrées à traiter (la vitesse, les tours par minute et la consommation du carburant) donc x_1 , x_2 et x_3 et une unité de polarisation (bias). Chaque neurone de la couche cachée transforme les valeurs de la couche précédente avec une sommation linéaire pondérée $w_1x_1 + w_2x_2 + w_3x_3 + bias$.

$$r = w_1x_1 + w_2x_2 + w_3x_3 + bias$$

En introduisant le r dans la fonction d'activation F (r), nous trouvons la sortie pour les couches cachées. Pour la première couche cachée h1, le neurone peut être calculé comme : $h_1^1 = F(r)$.

Pour toutes les couches cachées, on répète la même procédure jusqu'à atteindre le dernier ensemble de poids.

On commence donc par créer notre base de données où les données (les entrées) seront stockées

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	data_line	int(11)			Non	Aucune		AUTO_INCREMENT	Modifier Supprimer Primaire Unique Index Spatial plus
2	speed	float			Non	Aucune			Modifier Supprimer Primaire Unique Index Spatial plus
3	rpm	int(11)			Non	Aucune			Modifier Supprimer Primaire Unique Index Spatial plus
4	conso_carb	float			Non	Aucune			Modifier Supprimer Primaire Unique Index Spatial plus
5	heure	varchar(50)	latin1_general_ci		Non	Aucune			Modifier Supprimer Primaire Unique Index Spatial plus

FIGURE 6.6 – La base de données.

Remarque : Nous n'avons pas pu amener cette étape jusqu'à la fin du au manque de temps et au manque de connaissances dans le domaines de l'intelligence artificielle.

6.3 Page Web :

On réalise une page web qui affichera les données de l'utilisateur ainsi que ça façon de conduire, ce dernier pourra l'ouvrir sur son téléphone et ainsi vérifier sa conduite.

6.3.1 Environnement de développement :

Notre serveur Web est XAMPP qui est un acronyme pour le système d'exploitation, Apache, MySQL, PHP et PERL.

- **Apache** :Apache Web Server est un logiciel de création, de déploiement et de gestion de serveur Web open source. Apache Web Server est conçu pour créer les serveurs Web qui ont la capacité d'héberger un ou plusieurs sites Web basés sur HTTP, il a la capacité de prendre en charge de nombreux langages de programmation.
- **MySQL** :MySQL est un système de gestion de base de données relationnelle (SGBDR) basé sur SQL (Structured Query Language) .L'utilisation la plus courante de MySQL est destiné à une base de données Web.Pour gérer l'administration de MySQL sur le Web, nous utilisons phpMyAdmin qui est un outil logiciel écrit en PHP, utilisé pour effectuer des opérations de gestion de bases de données, de tables, de colonnes, de relations et l'instruction SQL peut également être utilisée directement.
- **PHP** :PHP est un langage de script côté serveur open source utilisé pour développer des sites Web statiques ou des sites Web dynamiques ou des applications Web.

6.3.2 Languages utilisés :

- **PHP** : PHP est un acronyme pour Hypertext Preprocessor, il permet la connexion aux bases de données telles que MySQL. PHP a la particularité d'être exécuté directement sur le serveur qui héberge les pages et non sur la machine de l'utilisateur
- **HTML** : HTML signifie Hypertext Markup Language. défini par le W3C (consortium World Wide Web), HTML fournit un moyen pour créer des documents structurés en indiquant la sémantique structurelle du texte comme en-têtes, paragraphes, listes, liens, citations et autres éléments
- **SQL** : SQL signifie Structured Query Language, c'est un langage standard pour les bases de données relationnelles utilisées pour stocker, manipuler et récupérer des données stockés en eux.

- **Javascript** : JavaScript est un langage de script utilisé à la fois côté client côté client, il est utilisé pour créer des éléments pour les pages Web
- **CSS** : CSS signifie feuilles de style en cascade, il décrit comment les éléments HTML sont à afficher à l'écran. Il apporte du style aux pages Web en interagissant avec des éléments HTML.

6.3.3 Conception de la base de données :

On conçoit une base de donnée dans le but de stocker les données obtenues du véhicule.

`data(data_line,speed,rpm,conso_carb,heure)`

domaine	signification	type	clé
data_line	numéro de la donnée	int	primaire
speed	valeur de la vitesse	float	
rpm	tours par minute	int	
conso_carb	niveau du carburant	float	
heure	heure d'acquisition de la donnée	VARCHAR(50)	

TABLE 6.1 – tableau "data".

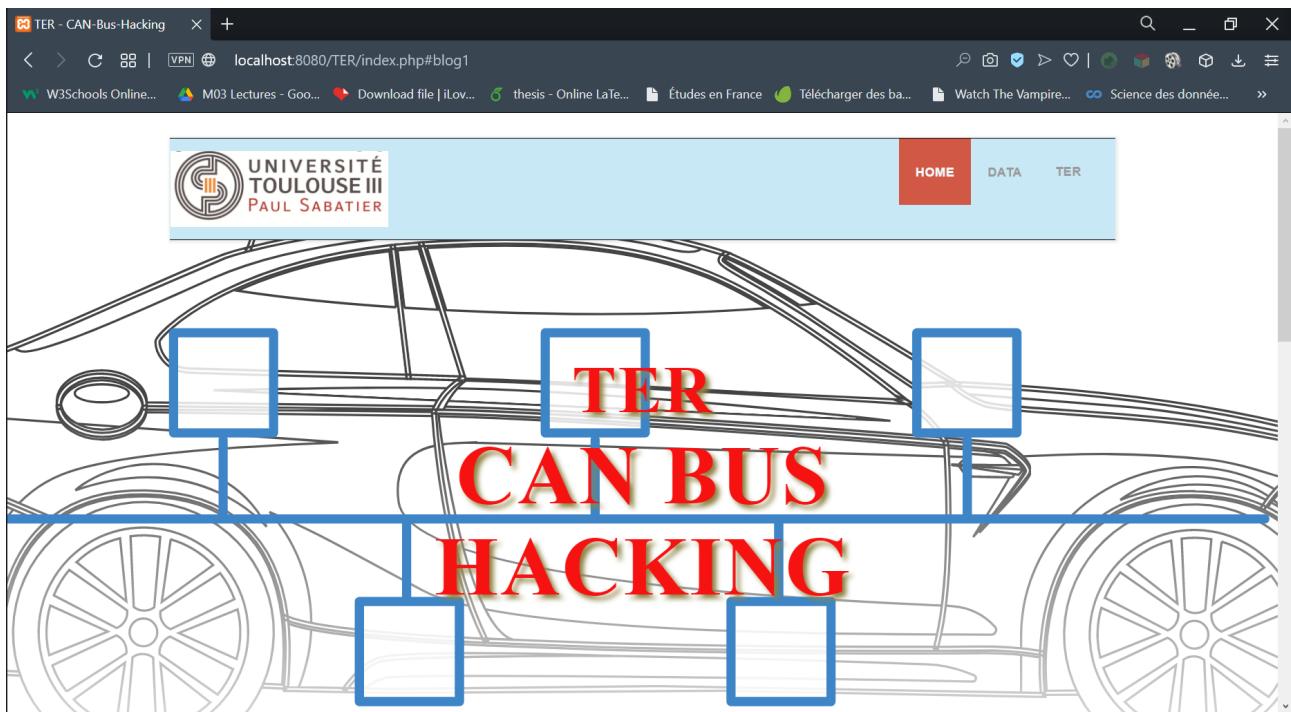


FIGURE 6.7 – La page web.

The screenshot shows a web browser window titled "TER - CAN-Bus-Hacking". The address bar indicates the URL is "localhost:8080/TER/index.php#blog1". The page content features the logo of Université Toulouse III Paul Sabatier and a navigation menu with "HOME", "DATA", and "TER" buttons. Below the menu is a section titled "LISTE DES DONNÉES". A table displays six rows of data:

numéro donnée	vitesse (Km/h)	tours par minute	niveau carburant (%)	heure
1	90	3000	0.25	12:45
2	0	887	0.25	12:45
3	0	887	0.25	12:45
4	0	887	0.25	12:45
5	0	887	0.25	12:45
6	15	1730	0.25	12:46

A "Read more" button is located at the bottom of the data list.

FIGURE 6.8 – La page web.

Conclusion

Contraintes :

Nous avons rencontré quelques contraintes au long du chemin, qui nous ont ralenti ou empêché de finaliser le projet et d'arriver au bout du cahier des charges

Parmis ces contraintes :

- Le travail à distance du à la situation sanitaire.
- Manque de temps et de connaissance sur l'intelligence artificielle.
- Le shield n'a pas été reconnu par la carte Arduino ; on tombe à chaque fois sur un problème d'initialisation du CAN.
- Après la connexion de l'OBD le véhicule ne s'allume pas.
- Le système ne communique pas avec les voitures avec un réseau CAN d'avant 2004.

Perspectives :

Différentes adaptations et améliorations ont été laissées pour l'avenir en raison du manque de moyens et de temps. Ces travaux futurs consistent à apporter des modifications et à l'ajout de certaines caractéristiques qui étendront les performances du système. Les idées suivantes pourraient être ajoutées :

- Une communication Wifi entre le système et le téléphone portable et affichage des données sur une application Android.
- Un système de diagnostic et de détection de panne en temps réel.

Bilans personnels :

BOUSADIA Ramzi

Pendant ce projet on a tracé une feuille de route qui nous a permis de satisfaire la majorité des exigences. On a appris des différentes méthodes d'analyse et de résolutions de problème software et hardware et aussi des connaissances de conception et réalisation, du cahier des charges proposé. Vu la situation actuelle on est obligé de s'adapter avec

le travail à distance et télétravail. Ce projet m'a permis d'approfondir mes connaissances et aussi c'est une opportunité pour appliquer plusieurs approches théoriques vues durant l'enseignement comme la conception système et la programmation en C et la collaboration entre les membres de groupe durant toutes la période de projet.

DOUKI Thiziri

Ce TER m'a permis d'apprendre la méthodologie à suivre pour la conception et la réalisation d'un projet ainsi que le genre de contraintes aux quelles on peut faire face. J'ai pu découvrir un nouveau domaine qui est le Machine Learning, et acquérir quelques connaissances dessus et aussi d'élargir mes connaissances techniques.

ZIANE Mouloud

Au tant que passionné de voitures, l'idée de créer un système d'analyse de la manière de conduite m'a poussé à aller plus loin dans ce concept vu que ce projet rassemble plusieurs approches et aussi c'est un système qui se développe avec les grandes innovations qui existent dans le domaine automobile, j'ai eu l'occasion d'utiliser des modules électroniques et de voir l'aspect mécanique et informatique ce qui m'a permis d'élargir mes connaissances techniques et cela en effectuant des tâches individuelles ou en équipe.

Bibliographie

- [1] technologuepro, <https://www.technologuepro.com/cours-systemes-embarques/cours-systemes-embarques-Bus-CAN.htm>.
- [2] technologuepro, <https://www.technologuepro.com/cours-systemes-embarques/cours-systemes-embarques-Bus-CAN.htm>.
- [3] Expert.ai, <https://www.expert.ai/blog/machine-learning-definition>.

A

Annexe

A.0.1 Programme principal

```
*****  
*          Authors          *  
*      BOUSADIA & DOUKI & ZIANE  *  
*      M1 SME                 *  
*      18/4/2021              *  
*****/*  
  
#include <SPI.h>           //SPI is used to talk to the CAN Controller  
#include <mcp_can.h>  
  
MCP_CAN CAN(10);           //set SPI Chip Select to pin 10  
  
unsigned char len = 0;  
unsigned char buf[8];  
unsigned int canID;  
  
unsigned char car_speed;  
unsigned int engine_rpm;  
unsigned char temp;  
  
void setup()  
{  
    Serial.begin(115200);    //to communicate with Serial monitor
```

```

//tries to initialize , if failed --> it will loop here for ever
START_INIT:
//setting CAN baud rate to 500Kbps
if(CAN_OK == CAN.begin(CAN_500KBPS))
{
    Serial.println("CAN_BUS_Shield_init_ok!");
}
else
{
    Serial.println("CAN_BUS_Shield_init_fail");
    Serial.println("Init_CAN_BUS_Shield_again");
    delay(100);
    goto START_INIT;
}
}

void loop()
{
    if(CAN_MSGAVAIL == CAN.checkReceive()) //check if data coming
    {
        CAN.readMsgBuf(&len , buf); //read data , len : data length ,
                                    //buf: data buffer
        canID = CAN.getCanId(); //getting the ID of the incoming message

        if (canID == 0xF1) //reading only our beloved 0xF1 message
        {
            Serial.print("ID_is:_");
            Serial.print(canID, HEX); //printing the ID in its standard form , HEX

            Serial.print("Length_is:_");
            Serial.println(len);

            Serial.println("\n\n");
        }
    }
}
//for the first signal: car speed

```

```
car_speed = buf[0];

Serial.print("Car_speed=");
Serial.println(car_speed);

//for the second signal: engine RPM
engine_rpm = buf[1];
engine_rpm = engine_rpm << 8;
temp = buf[2] & 0x3F;
engine_rpm = engine_rpm + temp;

Serial.print("Engine_RPM=");
Serial.println(engine_rpm);

Serial.println("\n\n\t*****");
}

}

}
```

A.0.2 Désigne 3D

