Université Libre de Bruxelles

Projet Multidisciplinaire

Rapport de Projet

Groupe : 03

Chef de projet : Luke Burke

Le Pari Bacar

Simon Yousfi

Nicolas Wallemacq

Maxime Thonar

Hugo Sorvillo

Florian Gilbert

Hippolyte Simonart

16 Mars 2018

Résumé

Le projet Bacar Zéro consiste en la conception d'un prototype de voiture autonome pouvant rouler sans intervention directe de l'homme. Ce prototype doit donc être capable d'analyser les différentes situations de routes et de les interpréter. Pour cela, plusieurs technologies ont été mises à notre disposition : l'Orange Pi (unité principale recevant et retournant les commandes), l'Arduino nano (contrôle des capteurs et des moteurs) et la caméra (permettant la visualisation de la route par le prototype). Une machine virtuelle permettant de faire des simulations informatiques du prototype a également été mise à la disposition de chaque étudiant.

La réalisation du projet comprend principalement la conception et la construction du prototype et la programmation des différents modules permettant le fonctionnement de la voiture autonome. Les travaux de programmation importants qui ont été réalisés sont un path-detector qui analyse la situation de la route, un sign-detector qui analyse les panneaux de circulation pour les identifier, une state machine qui informe sur l'état de mobilité de la voiture ainsi que des programmes Arduino qui permettent de gérer la tension à appliquer dans le moteur (par l'intermédiaire d'une loi tension/vitesse) et à contrôler les capteurs. Les programmes "path-detector", "sign-detector" et "state machine" ont été développés et testés au moyen de simulations sur ordinateur.

Le prototype est donc capable de reconnaître les différentes situations de routes qui se présentent à lui ainsi que les panneaux de circulation "stop, tourner à gauche et tourner à droite". Ensuite, il sera capable de réagir en fonction des différents éléments analysés. La bonne coordination de toutes les parties de la voiture (programmes, capteurs de bord, caméra, moteurs, ...) tient surtout en la communication des différentes parties et en la programmation de ces commandes afin de réaliser des manœuvres concrètes.

Ce rapport compte 8922 mots.

Table des matières

1 Introduction

De la marche à pied à l'avion supersonique en passant par la calèche, la mobilité a toujours été un véritable défi pour l'homme. C'est d'autant plus vrai aujourd'hui, du fait que notre civilisation a été habituée à parcourir de nombreux kilomètres en peu de temps. Mais comment les gens réagiront-ils le jour où les voitures n'auront plus de place sur les routes ? Parce que c'est déjà le cas dans de nombreux endroits, à Bruxelles notamment. La mobilité est donc devenue un défi à relever non seulement par la société mais surtout par le monde de l'ingénierie. Car la solution existe ! Elle a été imaginée et des appareils sont maintenant déjà en état de marche. Cette solution, c'est la voiture intelligente. Le projet des étudiants de première année bachelier en école polytechnique de Bruxelles est de relever ce défi en construisant ce type de véhicule.

L'objectif des organisateurs était de mettre le groupe dans la peau d'une équipe de développement de la société automobile nommée "Bacar". Ce projet, le pari bacar, porte d'ailleurs bien son nom. En effet, à cause de la concurrence, l'équipe se trouve dans l'obligation de concevoir un prototype de voiture intelligente. Quel est le pari ? Ce défi doit être relevé en un laps de temps très court !

Le véhicule devra être amené à réaliser un maximum de manœuvres tout en respectant les contraintes de sécurité. Ces manœuvres, toutes définies dans le cahier des charges (Annexe A), sont réparties selon quatre catégories différentes. Toutes ces catégories seront expliquées mais seules les manœuvres que le groupe est parvenu à réaliser seront énoncées.

Afin de comprendre l'objectif de chaque catégorie, il est essentiel de définir brièvement les composants majeurs du prototype. Ce dernier est équipé de deux micro-ordinateurs : l'Arduino, ordinateur qui gère la partie physique du véhicule, et l'Orange Pi, le cerveau de la voiture. Le véhicule est également équipé de moteurs et de capteurs, tous deux pris en charge par l'Arduino, ainsi qu'une caméra.
Pour la première catégorie, seul l'emploi de l'Arduino et des capteurs est nécessaire. La manœuvre la plus intéressante qui s'y trouve est celle proposant de faire avancer le véhicule sur un chemin sinueux uniquement à l'aide des capteurs de bord.
La seconde catégorie nécessite l'utilisation d'une loi tension/vitesse (donnant la vitesse de la voiture en fonction des tensions envoyées dans les moteurs), en plus de l'Arduino et des capteurs de bord. En effet, la manœuvre que le groupe a réalisée consiste à faire rouler la voiture selon une trajectoire circulaire de 50cm de rayon et de s'arrêter à son point de départ. Il s'agira alors, à l'aide de calculs basés sur la loi tension/vitesse, d'envoyer la bonne tension à chaque moteur afin d'obtenir la trajectoire souhaitée. Cette manœuvre sera expliquée plus en détails dans la partie "Arduino".

Les deux dernières catégories font toutes deux appel à l'unité centrale de la voiture,

l'Orange Pi. Elles sont principalement focalisées sur la reconnaissance de route et de panneaux, avec un accent sur la prise de décision dans la quatrième catégorie.
Le groupe a réussi à mettre au point les programmes de reconnaissance de la route et des panneaux (qui ont été testés avec succès en simulation) mais n'a pas disposé de suffisamment de temps en ce qui concerne la prise de décision.

La structure de ce rapport sera analogue à la manière dont le groupe a attaqué le problème. Ainsi, il traitera de deux sujets principaux que sont la partie hardware et la partie software. Au milieu de ces deux sections se trouve une recherche expérimentale de la loi tension/vitesse, indispensable pour obtenir la trajectoire souhaitée du véhicule.

2 Hardware

2.1 Architecture physique et interaction entre les modules

Dans le cadre du projet bacar, certains composants tels que des micro-ordinateurs ainsi qu'une caméra ont été mis à la disposition du groupe. Cette partie a pour but d'expliquer brièvement leur fonctionnement ainsi que la manière dont ils interagissent.

Le prototype est organisé tout autour d'un micro-ordinateur, l'Orange PI PC. Grâce à son Wi-Fi intégré, il est possible d'envoyer des commandes depuis un ordinateur extérieur. De plus, il est connecté à une caméra avec un objectif grand angle, lui permettant, après un traitement au sein du micro-ordinateur, une représentation de son environnement routier. Pour avoir la possibilité d'interagir avec les deux moteurs, il est nécessaire que l'Orange PI soit aidé d'un micro-contrôleur : l'Arduino Nano. Ce micro-contrôleur offre la possibilité d'envoyer des tensions indépendamment dans chaque moteur, en fonction des besoins, par exemple tourner ou bien s'arrêter. Mais pour pouvoir contrôler la tension aussi bien dans son intensité que dans son sens, un pont en H est nécessaire [1]. En effet, le pont en H permet d'inverser le sens du courant, mais aussi de modifier l'intensité des tensions envoyées par l'arduino pour que celles-ci soient proportionnelles à la demande des moteurs. L'arduino a une autre fonction que l'envoi des tensions, il est connecté à des capteurs de bord, qui détectent le changement de couleur lié au bord de la route, et à un capteur de distance, qui détecte s'il y a un obstacle, à quelques centimètres, devant le véhicule. Ces détecteurs fonctionnent de la même manière, ils envoient un signal électromagnétique d'une fréquence bien précise et analysent la réflection de ces ondes [2] pour ensuite communiquer les informations récoltées à l'Arduino.
L'alimentation se divise en deux, 4 piles AA en série alimentent les deux moteurs via le pont en H, et la batterie externe de 2 A alimente l'Orange PI, l'Arduino et tout ce qui est en communication avec ces deux modules.
Tous ces composants font donc partie d'un même circuit électrique, qui a été grandement simplifié grâce à l'utilisation de la breadboard qui est un support destiné à concevoir des circuits électriques bien organisés et facilement modifiables.

1. Toshiba, H-bridge datasheet *Toshiba Bi-CD Integrated Circuit Silicon Monolithic TB6612FNG*, 2007
2. IR sensor, *IR Sensor for Obstacle Avoidance KY-032*, Site web sur INTERNET `http://irsensor.wizecode.com/`, 2016

2.2 Prototype

Lors de la construction du prototype, il y eut des choix à faire concernant le modèle et la position de ses composants. Cette partie a pour but de justifier ces choix ainsi que de décrire brièvement les modèles choisis. Il est important de noter que chaque groupe s'est vu attribuer un budget maximal de 100€ (cf. annexe B pour le budget du groupe), sans compter les composants les plus importants (micro-ordinateurs, capteurs, breadboard, pont H et caméra) offerts par D'Ieteren, sponsor du projet.

1. Châssis, moteurs et roues

   Le kit, représenté qui a été choisi était proposé par l'équipe organisatrice sur le forum du cours TRAN-H101 [3]. Il est constitué d'un châssis en plexiglas, de deux moteurs électriques à courant continu et d'un support pour piles muni d'un interrupteur (Figure 1)

   Il a été sélectionné pour les dimensions de son châssis qui sont de 20 cm × 14 cm (et donc conformes au cahier des charges) ainsi que pour son boîtier pour piles, muni d'un interrupteur et qui répond donc à un critère de sécurité du cahier des charges concernant l'alimentation, qui spécifie que le prototype doit être équipé d'un interrupteur permettant de couper l'alimentation des moteurs sans perturber l'alimentation de l'unité centrale. (Figure 2)

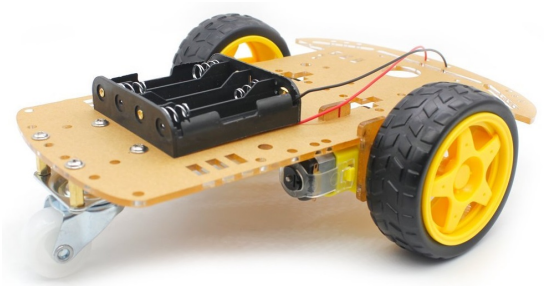

FIGURE 1 – Châssis du kit

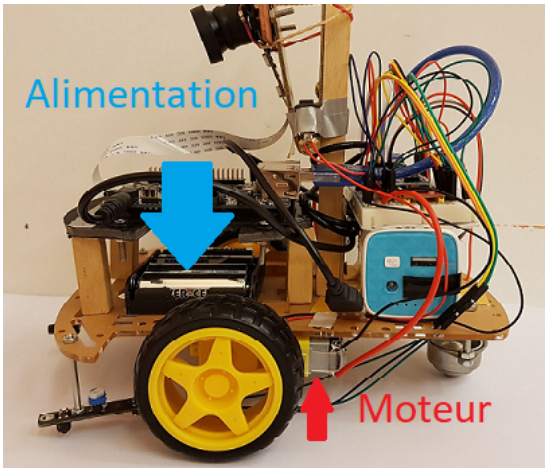

FIGURE 2 – Moteurs et boîtier pour piles

3. Université Virtuelle, *FAQ*, `https://uv.ulb.ac.be/mod/forum/view.php?id=303348`, 2017/2018

2. La batterie

Afin d'alimenter l'Orange-pi et l'Arduino, il était nécessaire d'équiper le prototype d'une batterie externe pouvant délivrer au minimum 2A. Au premier quadrimestre, le choix du groupe s'était porté sur la batterie Anker PowerCore 10000. En effet elle était peu cher, légère et compacte. Cependant, elle a très vite posé problème. En effet, même quand elle était chargée au maximum, elle ne délivrait plus assez de courant pour maintenir la fonctionnalité wifi de l'orange pi allumée. Il est donc intéressant de se pencher sur le fonctionnement d'une telle batterie, dite "batterie au lithium".

Une batterie se compose d'électrodes séparés par un électrolyte mis en contact avec des composants chimiques et permettant une réaction dite d'oxydoréduction, c'est à dire une réaction au cours de laquelle se produit un échange d'électrons et qui conduit donc à la formation de courant. Ce déplacement prend place de l'anode (-) vers la cathode (+) pour une décharge (Figure 5) et inversement lors d'une recharge. (Figure 6) [4].

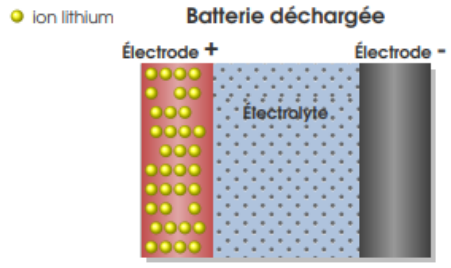

FIGURE 3

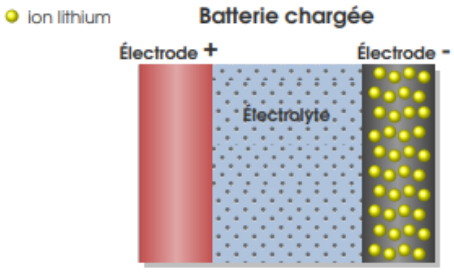

FIGURE 4

Lors des cycles de charge et de décharge, une fine couche d'ions vient se déposer sur l'électrode, empêchant la circulation des ions et entraînant ainsi une diminution de la capacité de la batterie par dégradation des électrodes. Ce phénomène est surtout observable lorsque la batterie est entièrement déchargée avant d'être rechargée (chose que l'équipe a malencontreusement fait). Il est donc préférable de recharger la batterie plus régulièrement après de petites sessions d'utilisation afin de la préserver au mieux.

4. VALLVERDU, Germain,Université de Pau, *Principe de fonctionnement des batteries au lithium*,pages 12-16, 22 juin 2011

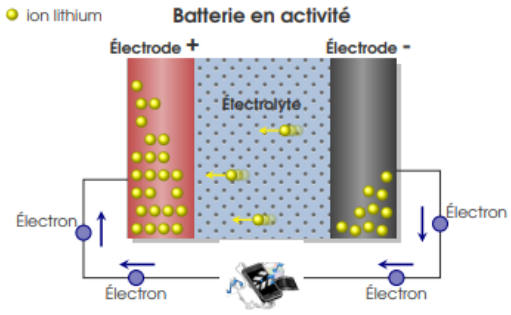

FIGURE 5

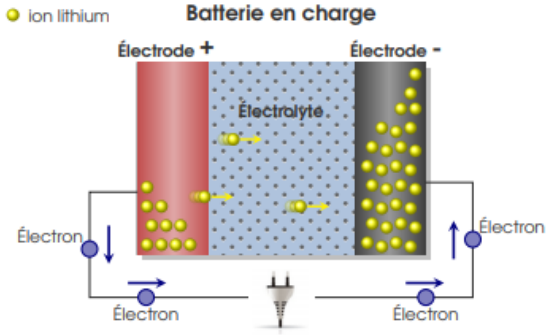

FIGURE 6

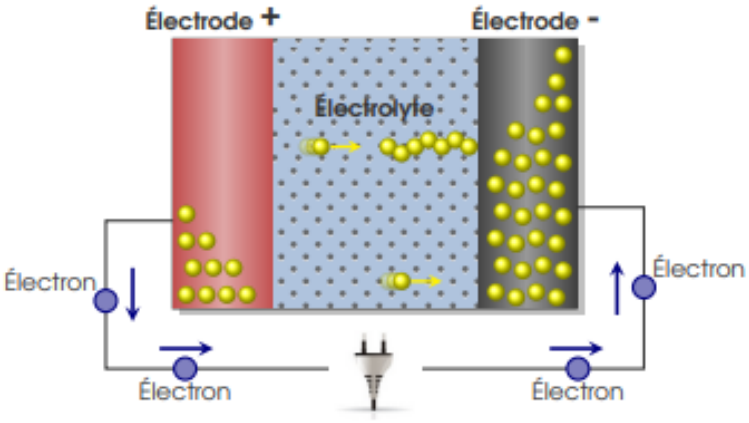

FIGURE 7

Le lithium étant le métal le plus léger et possédant la plus grande énergie par densité de masse, c'est le composant à privilégier lorsque la masse de la batterie est contraignante. En effet, ce type de batterie possède une grande capacité de stockage ainsi qu'un potentiel électrique assez haut comparé aux batteries au plomb ou au nickel [5]. Une telle batterie était incontournable pour ce projet : étant légère et capable de délivrer les 2A requis pour l'alimentation du véhicule, ce choix n'était pas foncièrement mauvais. Il semblerait juste que l'équipe l'ai mal utilisée.

La nouvelle batterie TP-lINK [6] pesant un peu plus lourd que la première, il a été décidé qu'elle serait placée à l'arrière du prototype afin d'assurer une répartition homogène des masses.

5. BUSHMANN, Isidor, *Batteries in a portable world*, pages 11,18-21, 1997
6. TP-Link, *10400mAh Power Bank TL-PB10400*, `https://www.tp-link.com/us/products/details/cat-5524_TL-PB10400.html`, 2017

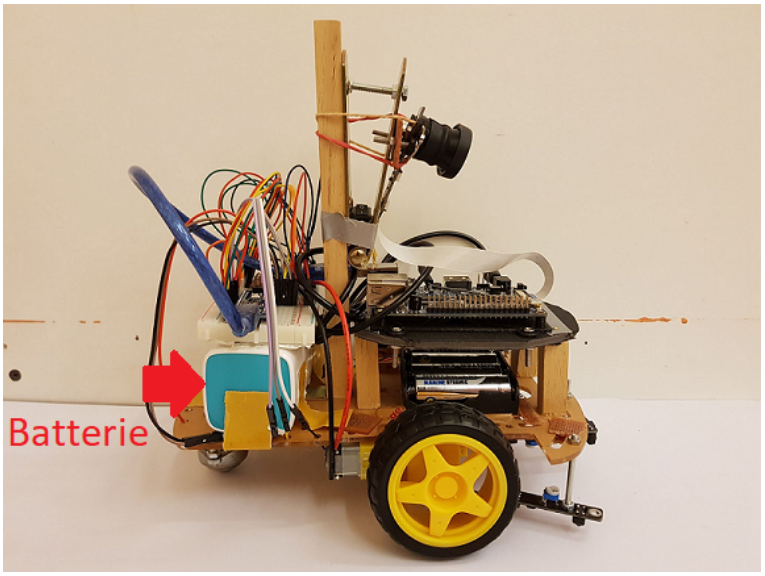

FIGURE 8 – Prototype avec batterie légendée

3. La Ball-caster

   La roue libre qui était fournie avec le kit, en tournant sur elle-même de manière imprévisible, perturbait la trajectoire de la voiture. Il est fort probable que ce problème soit dû au fait que le point de contact de la roue avec le sol ne se trouve pas dans l'axe de rotation. Le groupe 4 ayant rencontré ce problème un peu plus tôt, il a acheté un pack de 6 ball-casters et a accepté de nous en céder une, après en avoir testé les bienfaits. Il a été décidé que le prototype fonctionnerait par traction, c'est à dire que les roues motrices se trouvent à l'avant du véhicule et par conséquent, la Ball-caster se situe à l'arrière.

4. Le mât et la charnière

   À l'aide d'un angle métallique, un mât en bois a été fixé au châssis. À ce mât vient s'ajouter une charnière à laquelle se fixera la caméra. L'utilité du mat est de surélever la caméra alors que la charnière sert à l'incliner vers l'avant pour avoir un meilleur angle de vue (elle permet donc de régler cet angle). Cette astuce fut appliquée en prévision de l'utilisation de la caméra qui a besoin d'une vue assez dégagée en fonction de l'angle choisi (qui était encore inconnu à ce moment).

5. Les câbles et la breadboard

   Il y a deux types de câbles différents. La plupart sont des mâle-mâle et ont été utilisés pour connecter l'Arduino au pont H par l'intermédiaire de la breadboard. Les autres câbles sont des mâle-femelle et ont été utilisés pour connecter les capteurs à l'Arduino, toujours par l'intermédiaire de la breadboard. Comme le montre la Figure 9, il a été décidé de placer la breadboard derrière la caméra afin que les câbles n'obstruent pas son champ de vision.

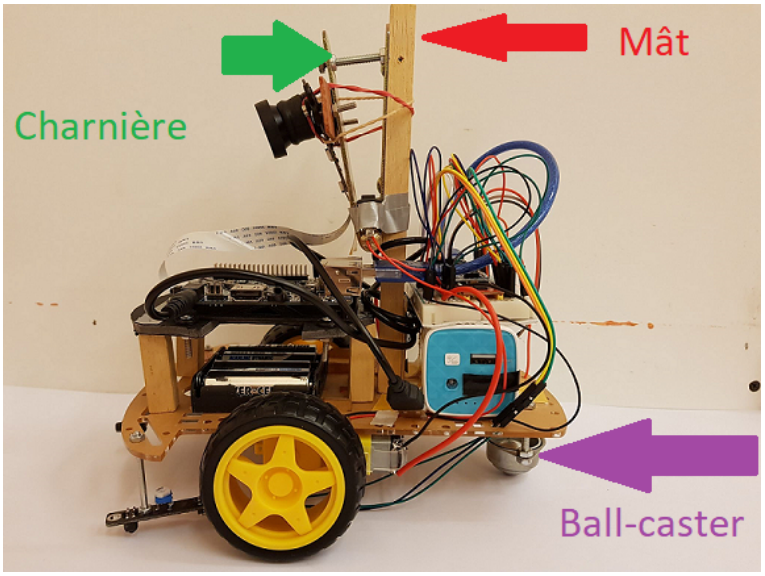

FIGURE 9 – Prototype avec charnière, mât et ball-caster légendés

Avant de parler du software, qui constitue la partie centrale du projet, il faut se pencher sur la loi tension/vitesse qui est un véritable outil permettant de faire le lien entre la physique de la voiture et les programmes qui la pilotent. En effet, cette loi diffère pour chaque prototype, voire même pour chaque moteur.

3 Courbe Tension/Vitesse

L'expérience décrite ci-après a pour but de définir une loi permettant de donner la vitesse en fonction de la tension envoyée dans les moteurs. Cette loi constitue un des piliers du projet, puisqu'elle permet à la voiture d'adopter une trajectoire bien précise, telle qu'avancer tout droit ou tourner de 10° vers la gauche, par exemple. Deux moteurs n'étant jamais parfaitement identiques, ils réagiront différemment à une même tension. Ceci est vrai en particulier pour ce que l'on appelle "la tension seuil". C'est la tension minimale nécessaire pour faire tourner le moteur. C'est donc pour ces raisons qu'il a été décidé de définir deux lois différentes. Ces lois fournissent la vitesse de la voiture, en fonction de la tension dans le moteur de droite pour l'une et en fonction de la tension dans le moteur de gauche pour l'autre. Du faite de la différence significative qu'il peut y avoir entre chaque prototype il faut bien se rappeler que ces courbes ne sont valables que pour une charge donnée (ici la masse du prototype de l'équipe).

1. Matériel utilisé
   - Le prototype
   - Un PC portable
   - Un mètre
2. Protocole
   - A partir d'un programme Arduino, nommé "test motors" (Annexe D), faire varier le pourcentage de tension maximale, x, dans la ligne de code motorA.actuate(x) ainsi que le pourcentage de tension maximale, y, dans la ligne de code motorB.actuate(y) (le côté de motorA dépend du câblage, il faut donc avant tout définir de quel côté il se trouve). Une attention toute particulière sera apportée dans le choix des tensions à faire varier afin que la voiture avance en ligne droite. Il est en effet beaucoup plus facile de calculer la vitesse d'un objet lorsque sa trajectoire est linéaire.
   - Connaissant les tensions envoyées dans chacun des moteurs, mesurer la distance parcourue sur un temps connu (ce temps peut être modifié dans la ligne de code delay(z), où z est en millisecondes. Voir annexe D). Répéter l'opération trois fois pour les mêmes tensions.
   - Recommencer l'opération plusieurs fois pour des tensions différentes (tout en s'assurant que la trajectoire de la voiture reste linéaire).

3. Mesures expérimentales

Tension motorA (V)	Tension motorB (V)	Distance sur 5 secondes (cm)	Vitesse (cm/s)
0.96	1.26	0	0
1.68	1.8	68	13.6
2.34	2.4	97	19.4
2.94	3.0	130.5	26.1
3.558	3.6	164.5	32.9
4.17	4.2	194.8	39
4.8	4.8	225	45

4. Lois et graphiques

   Voici les graphiques de la vitesse en fonction de la tension dans le moteur de droite/gauche.

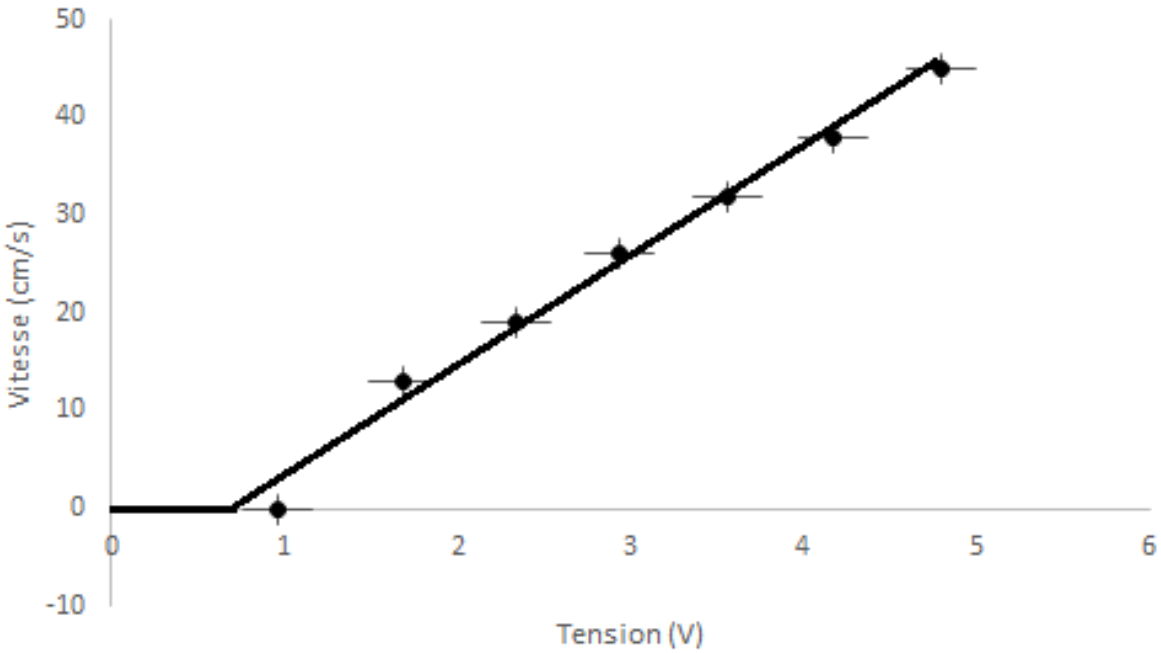

FIGURE 10 – Graphique de la vitesse en fonction de la tension dans le moteur droit

Le graphique de la Figure 10 a pour équation :

$$\text{v (cm/s)} = 11,2 \text{ (cm/Vs)} \times \text{T (V)} - 8 \text{ (cm/s)}$$

Où v représente la vitesse de la voiture et T représente la tension dans le moteur droit. Il est important de noter la vitesse nulle pour une tension comprise entre 0 et 0.96V, cette dernière valeur correspond à la tension seuil.

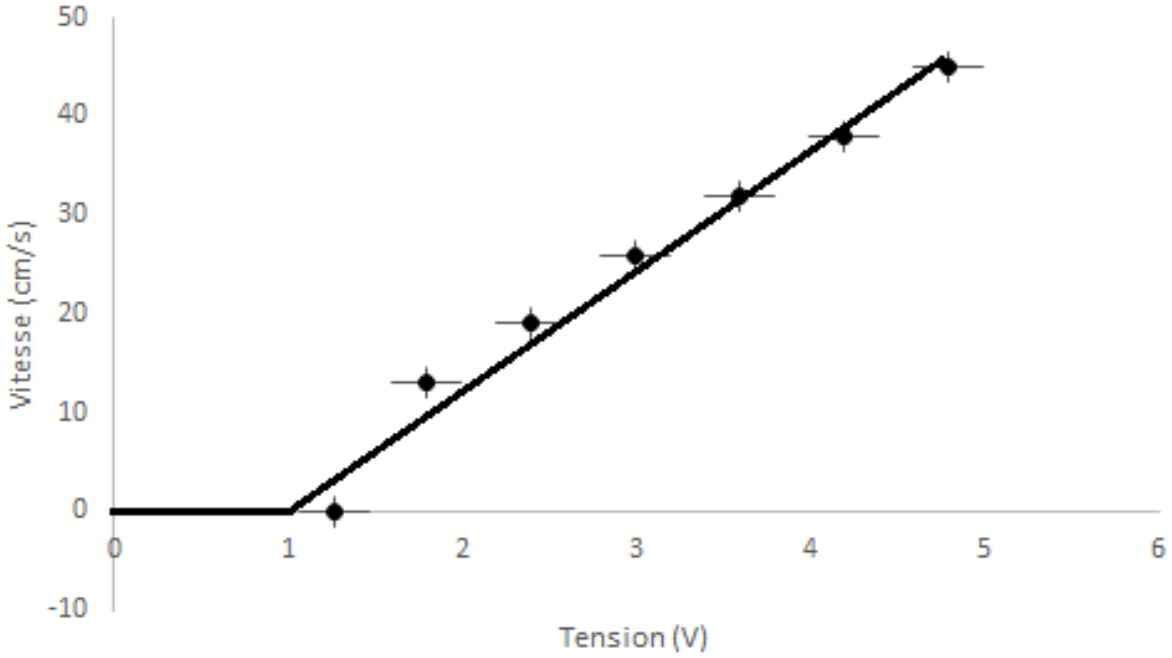

FIGURE 11 – Graphique de la vitesse en fonction de la tension dans le moteur gauche

Le graphique de la Figure 11 a pour équation :

$$\text{v (cm/s)} = 11,9 \text{ (cm/Vs)} \times \text{T (V)} - 11 \text{ (cm/s)}$$

Où v représente la vitesse de la voiture et T représente la tension dans le moteur gauche. La vitesse est nulle pour une tension allant de 0 à 1.26V, cette dernière valeur correspond à la tension seuil.

En conclusion, ces lois sont indispensables pour faire avancer le véhicule en ligne droite et pour lui définir une trajectoire particulière.

4 Software

4.1 Architecture logicielle

Dans cette partie, la composante logicielle de ce projet va être abordée. Les différentes interactions entre les programmes au sein du prototype vont être expliquées dans un premier temps. L'architecture logicielle, visible sur la Figure 12, s'organise en deux modules : l'unité centrale et l'arduino. Ces deux modules utilisent un langage de programmation différent mais leur interaction est nécessaire au bon fonctionnement du prototype.

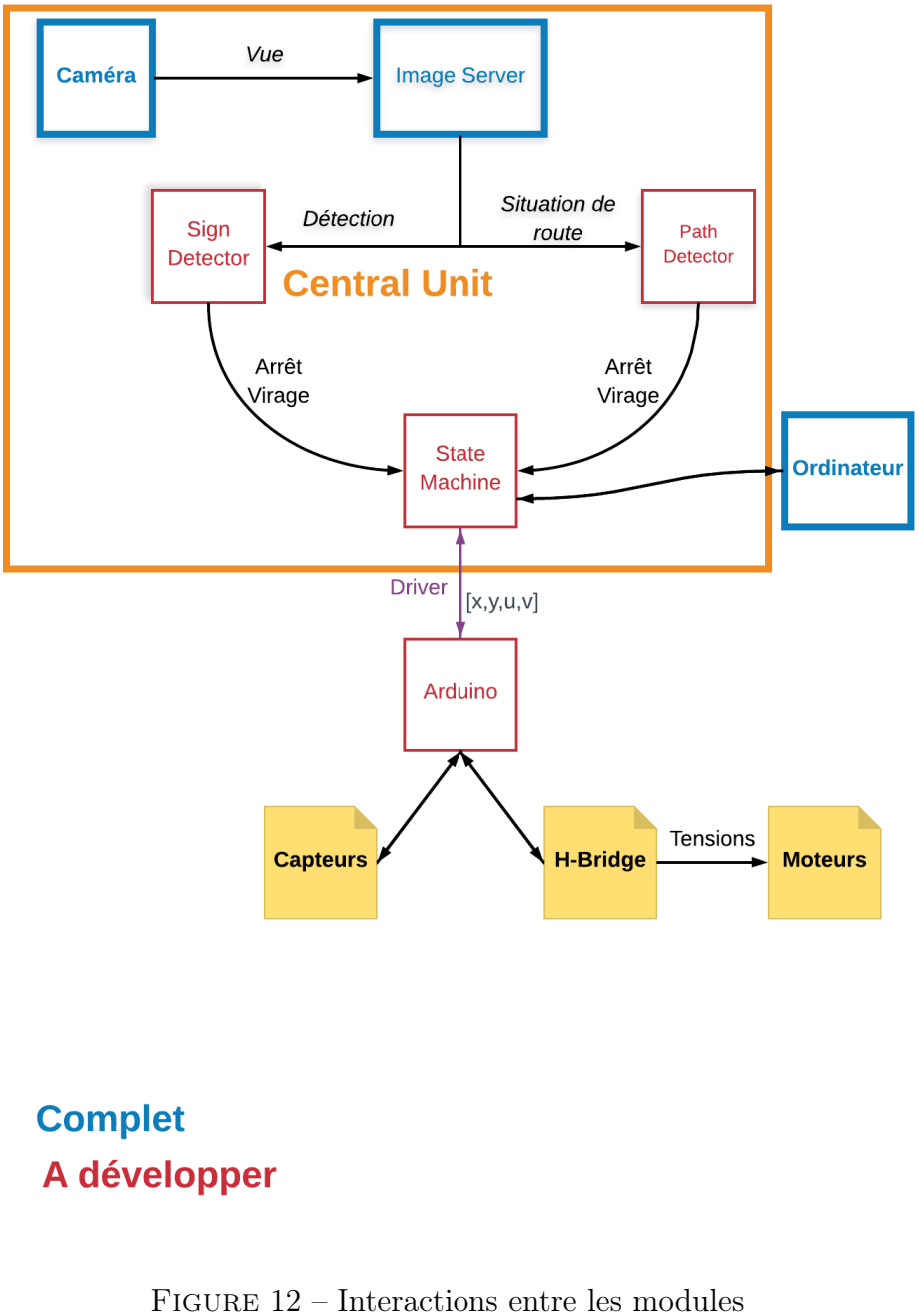

FIGURE 12 – Interactions entre les modules

L'unité centrale est composée de quatre programmes principaux : la state machine, l'image server, le path detector et le sign detector. Lorsque la caméra capte une image de la route, l'image server va renvoyer une image en vue verticale (une image redressée ou encore bird's view) et en noir et blanc de la route et il est muni d'une fonction qui détecte les panneaux (avec un certain ratio d'erreur,détaillé dans la partie 4.4.2 Sign detector). Il va ensuite envoyer la vue de la route au path detector qui va alors analyser cette image. Une fois l'analyse terminée, les informations sont transmises à la state machine. Quant au sign detector, il analyse ce que l'image server a détecté comme panneau, pour savoir si c'est un panneau stop, de direction ou bien une fausse détection. Il transmet lui aussi les informations à la state machine. Une fois toutes les informations récoltées, la state machine calcule les modifications à effectuer dans les moteurs.
Pour finir le travail de l'unité centrale, la state machine envoie un quadruplet contenant les informations nécessaires à la bonne modification de la trajectoire composé de deux entiers (x, y) et de deux nombres flottants (u, v). Cet envoi se fait via le driver qui est un programme au sein de l'unité centrale et qui permet d'établir la communication avec l'Arduino. Cette communication fera l'objet d'une section de ce rapport.

Parallèlement à l'analyse de la situation de route par l'unité centrale, l'Arduino va, à l'aide de ses capteurs de distance et de bord, s'assurer que le prototype ne se trouve pas en dehors de la route et qu'il n'y a pas d'obstacles devant le véhicule. Mais quand l'Arduino reçoit un message du driver (et donc indirectement de l'unité centrale), il va interpréter ce message et effectuer les modifications motrices adéquates. Pour finir la boucle et recommencer la démarche dans une autre situation, l'Arduino renvoie le message reçu vers le driver, qui lui va le retransmettre à l'unité centrale [7].

Comme le montre la Figure 12, certains de ces modules ont déjà été donnés complets, d'autres ont dû être développés. Ces développements ont été faits via l'ajustement et l'ajout de programmes (en python pour la partie path et sign detector ainsi que state machine et en langage se rapprochant du C++ pour l'Arduino) qui, une fois finalisés, permettent le déplacement de la voiture uniquement grâce aux images reçues ainsi qu'aux panneaux détectés plutôt qu'à l'aide de capteurs servant principalement d'arrêt d'urgence (Chacun des modules énoncés va bien entendu être expliqué dans la section le concernant).

Afin de détailler le fonctionnement de tout ces modules il faut avant tout partir des bases et de l'objectif à atteindre ainsi les manœuvres Arduino qui furent plus "basiques" serviront d'introduction aux manœuvres plus corsées nécessitant un code plus complexe et donc une plus grande réflexion derrière leur fonctionnement.

7. Université Virtuelle, *Architecture logicielle*, `https://uv.ulb.ac.be/pluginfile.php/1041625/mod_resource/content/1/Annexe_software_architecture.pdf`, 2017/2018

4.2 Arduino

Toutes les manœuvres ne nécessitent pas l'entièreté de l'architecture logicielle, c'est le cas de cette partie qui est consacrée à des manœuvres dites de base et de précision. Les manœuvres de base ne font appel qu'à l'utilisation du temps ainsi que des capteurs et celles de précision utilisent principalement la courbe tension/vitesse.

On peut donc diviser les manœuvres de base en deux parties : celles qui utilisent la notion de temps et celles qui ont besoin des capteurs.
Les trois manœuvres liées au temps sont : "avancer pendant 5 secondes et puis s'arrêter", "avancer d'un mètre et puis s'arrêter" et "rouler trois secondes, s'arrêter deux secondes, recommencer à avancer pendant deux secondes et puis arrêter les moteurs". Ces trois manœuvres ont été réalisées, la plus compliquée étant celle où il faut avancer d'un mètre car cela requiert une prise de données expérimentales, il fallait trouver la tension (et donc la vitesse) adéquate pendant une durée précise pour faire avancer le prototype d'un mètre.
Les trois autres manœuvres sont liées aux deux types de capteurs (frontal et de bords). Pour deux de ces trois manœuvres, il suffit de faire arrêter le véhicule si un bord ou un obstacle est détecté. La troisième manœuvre n'exige pas un arrêt mais une correction de trajectoire, si un capteur de bord détecte un bord à gauche, il faut alors augmenter la tension dans le moteur de gauche pour faire tourner la voiture vers la droite. La difficulté de ces trois manœuvres réside dans la compréhension des capteurs et des informations qu'ils renvoient.

Seulement trois des manœuvres de précision ont été réalisées : "avancer d'un mètre tout droit et puis s'arrêter", "suivre une trajectoire carrée de 30 cm de côté (rotation de 90°) et puis s'arrêter au point de départ" et "suivre une trajectoire circulaire de 50 cm et puis s'arrêter à son point de départ". Cette dernière manœuvre a été réalisée à l'aide de calcul à partir de la courbe tension/vitesse. En effet, ce problème présente deux inconnues : la vitesse de la roue droite et la vitesse de la roue gauche.
En effet, supposons que la voiture tourne à gauche pour décrire ce cercle, on peut décrire le chemin parcouru par chacune des deux roues. Sachant que la distance entre les deux roues est d'environ 14cm, ces trajectoires décrivent un cercle de rayon $r \approx (50 \pm 7)$ cm. Sachant que le diamètre d'un cercle vaut $2 \times \pi \times r$, la distance parcourue par la roue gauche, $x \approx 270$ cm et la distance parcourue par la roue droite, $y \approx 358$ cm. Il suffira ensuite de fixer le temps que prendra le véhicule pour réaliser cette manœuvre pour connaître le vitesse de chacune des roues. Pour rappel, voici les deux lois tension/vitesse du prototype :

1. v (cm/s) $= 11,2$ (cm/Vs) $\times$ T (V) $- 8$ (cm/s) pour le moteur droit.
2. v (cm/s) $= 11,9$ (cm/Vs) $\times$ T (V) $- 11$ (cm/s) pour le moteur gauche.

Pour un temps de 10 secondes, on obtient :

$$35.8\ (\text{cm/s}) = 11,2\ (\text{cm/Vs}) \times \text{T}\ (\text{V}) - 8\ (\text{cm/s}) \tag{1}$$

soit une tension égale à 3.9V pour le moteur droit.

$$27\ (\text{cm/s}) = 11,9\ (\text{cm/Vs}) \times \text{T}\ (\text{V}) - 11\ (\text{cm/s}) \tag{2}$$

soit une tension égale à 3.2V pour le moteur gauche.

4.3 Fonctionnement de la simulation

Lors de l'installation de la machine virtuelle sur ordinateur, plusieurs fichiers y étaient déjà présents. Il y avait notamment un fichier "exemple" reprenant les fonctions nécessaires à une simulation sommaire mais extrêmement importante de par le faite qu'il est inutile d'avoir un arduino fonctionnel pour pouvoir tester ses programmes de path et sign detector (Pas besoin de se préoccuper des tensions et des communications, la simulation s'occupe de faire rouler la voiture à partir de la commande "GO").C'est donc un programme très pratique pour l'équipe, qui lui permettra de s'assurer qu'en cas de situation idéale, leur codes fonctionnent ou ne fonctionnent pas et pourront donc les modifier sans trop de pénibilité technique (Arrêter les moteurs,...). En effet, en simulation la vue redressée de la caméra (ou bird's view : figure **??** par exemple) est parfaitement au milieu de la route et ne tremble pas, les routes sont droites et sans aucun bruit (tests qui sont réalisés dans la partie 4.4.1 Path Detector),... il faut donc quand même relativiser il s'agit bien d'un programme de "tests" et non pas la réalité pure. Il est aussi possible de lancer des simulations vidéos qui remplacent la caméra en envoyant une vidéo déjà tournée sur un prototype appartenant aux organisateurs. Dans ce cas, l'image server et le viewer sont aussi lancés afin de justement vérifier sont programme dans une situation plus "réelle" (secousses,calibrage pas forcément parfait,...).

Ce programme, très complet, permet donc de tester ces codes dans des environnements prévus qui sont en réalités de simples arènes délimitant la route et les bords par du noir et blanc respectivement.

Sachant cela il reste maintenant à analyser ce que chacune de ces arènes ont à offrir (dans l'ordre de la figure 13) :

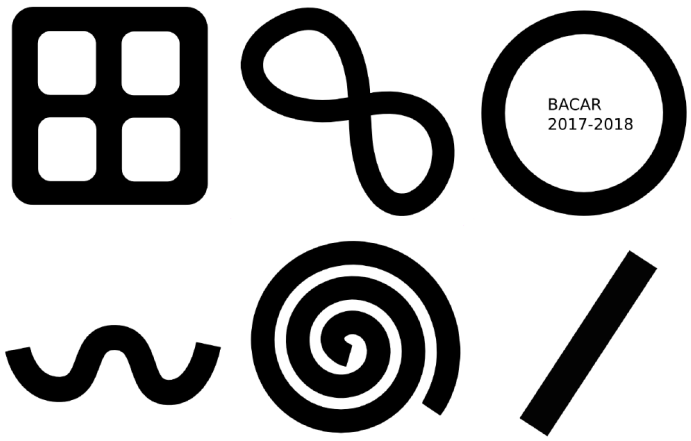

FIGURE 13 – Arènes de base (Manhattan, Infinite, Circle, Moustache, Spiral, Dead end)

1. Manhattan : Tout comme la ville du même nom, c'est une arène en carré lui-même découpé en quatre carrés. Des panneaux y sont disséminés afin de tester aussi le sign detector. Il est remarquable que certaines situations de routes particulières sont présentes : le T-turn, des tournants à gauche et à droite et un carrefour (Situations gérées dans la partie 4.4.1 Path Detector)". Le challenge provenant de ces situations à prendre en compte.
2. Infinite et Circle : Un circuit en infini et un cercle. Suivre une route circulaire, en arc constant. La route n'est plus juste une simple ligne droite à virages.

3. Moustache, Spiral et Dead end : Un circuit droit ou oscillant se terminant par un cul de sac où il faudra que le véhicule s'arrête (Particularité de spiral la route rétrécis de plus en plus et la route se rapprochant elle peut être détectée et confuse le véhicule)

4.4 Trois programmes indispensables

Afin de rendre le prototype de voiture autonome dans ses déplacements, il est nécessaire d'interpréter l'environnement et de doter la voiture d'une capacité de décision. Pour cela, trois programmes clefs ont été codés (cf. Annexes E, F, G). Le premier, le Path Detector, permet d'interpréter la situation de route et de calculer l'angle à prendre afin de rester au milieu de la route. Le second programme, le Sign Detector, interprète quant à lui le code de la route via la reconnaissance des panneaux de signalisation. Le dernier est le State Machine. Celui-ci reçoit les informations du Path Detector et Sign Detector et agit comme le cerveau de la voiture. Les décisions y sont prises et il envoie les informations nécessaires à l'Arduino.

4.4.1 Path Detector

La première étape vers l'automatisation d'une voiture et de doter celle-ci d'une capacité à reconnaître son environnement. Le Path Detector a donc été codé afin de traduire l'angle à entreprendre par la voiture ainsi que la situation de route à proximité. Ces informations seront transmises à la State Machine.

La Figure 14 montre le mode de fonctionnement du Path Detector.

L'Image Server reçoit l'image de la caméra et la convertit en représentation binaire. Une couleur noire est attribuée à la route et une couleur blanche aux bords de route. De plus, l'image est redressée de manière à avoir une vue du "dessus" en enlevant les effets de perspective. Cette nouvelle image est alors envoyée au Path Detector afin de déterminer quel type de croisement la voiture a en face d'elle.

Dans le cadre de nos tests, 3 types de croisements de route nécessitant une prise de décision sont à prendre en compte : le croisement en T, l'angle droit se dirigeant vers la gauche/vers la droite et la voie sans issue.

Lorsque rien de particulier n'est détecté, la voiture calcule le centre de la route en détectant plus loin devant elle le premier pixel blanc trouvé sur la droite et sur la gauche d'une même ligne verticale. Le milieu de cette droite est considéré comme étant le centre de la route. L'angle à emprunter par la voiture est l'angle entre la droite verticale passant par le centre de la voiture et la droite passant par le centre de route détecté. La valeur de l'angle est stockée dans la variable "heading" qui sera envoyée à la State Machine sous forme de dictionnaire : le "path dict". Une autre variable qui est stockée dans ce dictionnaire est la variable "event route" qui peut être un des strings suivants : "t turn", "angle going to right", "angle going to left", "carrefour", "deadend". Ce dictionnaire a été inséré dans le viewer pour pouvoir suivre plus simplement les différentes situations lors des simulations. Le viewer est une fenêtre lancée lors des simulations affichant le contenu des dictionnaires du Sign Detector et du Path Detector en temps réel.

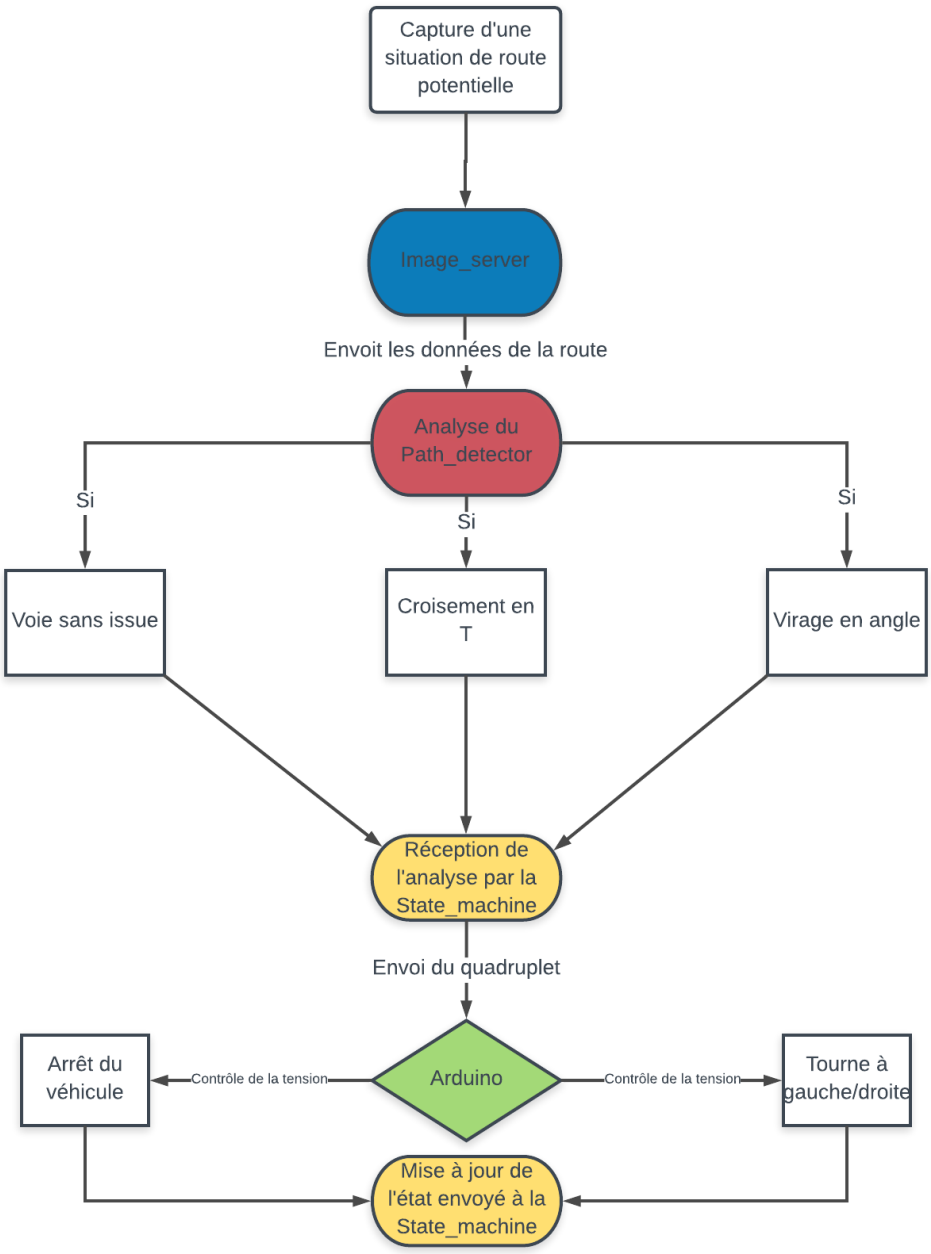

FIGURE 14 – Organigramme du fonctionnement du Path detector

Afin de détecter un croisement et éviter les interférences, un système d'interprétation de zones de couleurs a été instauré. Il a été délimité 4 zones visibles sur la Figure 16 : une zone à gauche, une zone à droite et deux zones avant. Chaque zone est un carré de 3 pixels sur 3 pixels dont la moyenne de couleur est calculée. Sur l'image, les boules rouges représentent la situation de ces carrés sur la route. Dans cette représentation binaire, la couleur de chaque pixel est représentée par un entier compris entre 0 (noir) et 255 (blanc). L'utilisation de la moyenne de couleur d'un ensemble de points d'une même zone permet d'éviter d'être trompé par le "bruit" et d'agir comme tampon si la couleur d'un des 9

pixels a été mal détectée. Chaque zone sera donc interprétée comme blanche (hors route) soit comme noire (route).

En utilisant des conditions "if" décrivant une certaine combinaison de zones blanches ou noires, il est possible de déterminer à quel croisement la voiture fait face. En guise d'illustration, voici l'exemple du croisement en T.

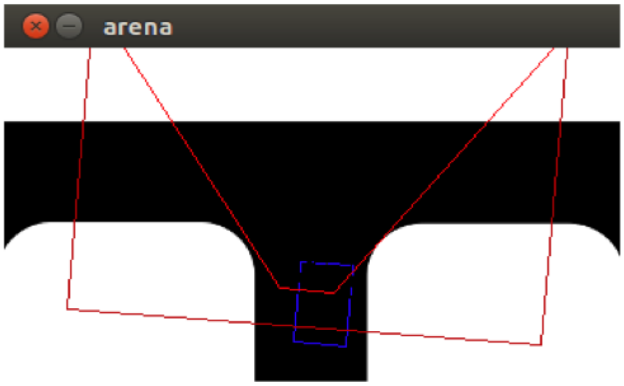

FIGURE 15 – Arena : Croisement en T

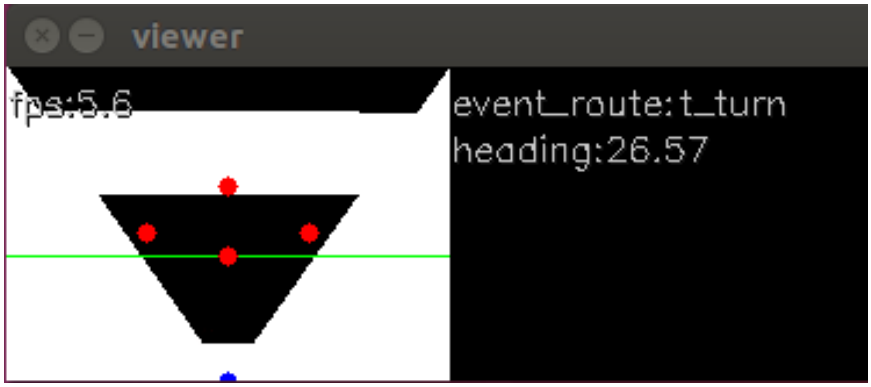

FIGURE 16 – Viewer : Croisement en T

Dans cette situation particulière, aucun bord n'est détecté ni à gauche ni à droite. Les conditions décrivent l'absence de route à l'avant mais la présence d'une route dégagée aussi bien à gauche qu'à droite. La situation de route est stockée dans la variable `event_route`.

C'est dans cette même démarche que les autres situations de route ont été codées.

Les résultats de ce code dans la simulation sont satisfaisants. La voiture se déplace correctement sur toutes les arènes mises à disposition à l'exception de l'arène "Spiral" dans laquelle la voiture sort de la route à la fin.

Dans la pratique, la voiture reconnaît correctement les situations de route.

Ce système pouvait être trompé par les "bruits". En effet, la simulation est une représentation idéale de l'image captée par la caméra. Dans la pratique, la caméra ne

capte pas parfaitement les couleurs et celles-ci sont alors mal interprétées, occasionnant certaines imperfections. Il faut donc un programme "résistant" face à ces imperfections. Il a fallu tester la fiabilité de celui-ci même lors de situations non optimales en créant des arènes avec des interférences, du bruit représenté par des petits points blancs. Ceux-ci seront lus comme étant des pixels blancs et donc interprétés comme des bords de route. Les arènes de base ont été modifiées grâce à Paint. Il a été décidé de faire 3 niveaux, tels que représentés sur la Figure 17 : le niveau 1 avec très peu de bruit réparti de façon égale dans l'arène ; le niveau 2 avec plus de bruit et des points blancs plus rapprochés les uns des autres ; finalement le niveau 3 avec un nombre d'interférences élevé pour tester les limites du système de zones indicatrices (cf Annexe C pour toutes les arènes).

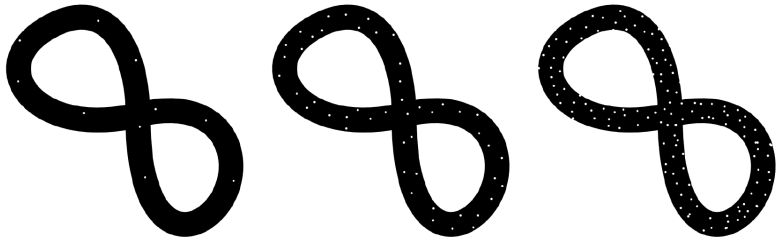

FIGURE 17 – Niveaux de bruit (1 ;2 ;3)

Ces trois niveaux ont été réalisés sur quatre arènes de test différentes qui sont : Manhattan, Circle, Spiral et Infinite.
En ce qui concerne le niveau 1, aucun problème n'est détecté.
Dans le niveau 2 par contre, un premier problème est rencontré lors du test de l'arène Infinite : la voiture opère un changement de direction alors qu'aucun croisement n'est réellement présent ce dernier étant un changement de direction. L'autre problème est dans l'arène de test Manhattan. En effet la voiture détecte un cul de sac en plein milieu d'un tournant. Aucun problème majeur n'est détecté dans les deux autres arènes (Spiral et Circle).
Lors des tests du niveau 3, des problèmes ont été rencontrés dans trois arènes. Dans Manhattan, la voiture sortait occasionnellement de la route ; dans Spiral, un dead end est détecté quasiment instantanément après le démarrage et dans Infinite, la voiture suit un chemin complètement différent de celui qu'elle est sensée suivre dans une arène sans bruit. Encore une fois, aucun problème détecté dans Circle.

On constate donc que le bruit a une grande influence sur le comportement de la voiture en fonction de sa densité (ex : Figure 17). La trajectoire de route est moins précise et dans certains cas, le Path Detector se trompe dans son interprétation de route et arrête ou fait sortir la voiture de la route par erreur.

4.4.2 Sign Detector

Une grande étape afin de rendre le véhicule autonome est de le doter non plus d'une capacité à interpréter la route qui se présente en face de lui mais d'une capacité à interpréter les règles de circulation qui s'y trouveraient via les panneaux de route. Un programme, le Sign Detector, a donc été écrit à cet effet. Son but est de détecter l'un des 3 panneaux suivants.

FIGURE 18 – Panneaux à détecter

Ces grandes étapes sont décrites sur la Figure 19.

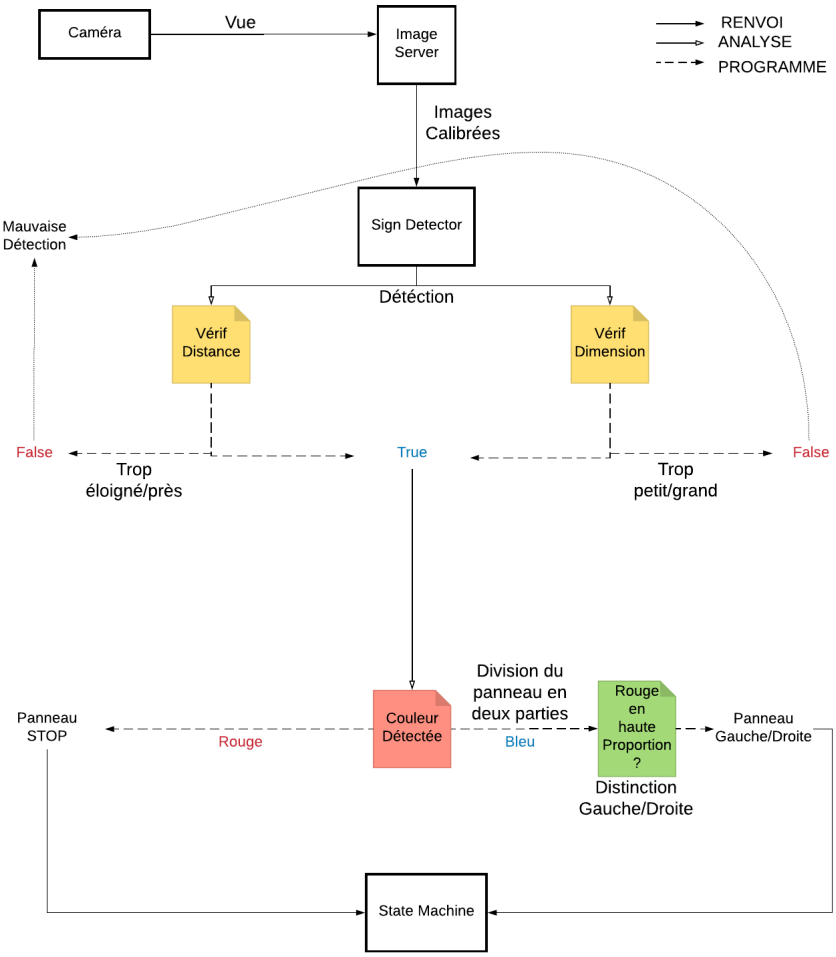

FIGURE 19 – Organigramme de fonctionnement du sign detector

L'Image Server, qui a préalablement analysé l'image reçue par la caméra et détecté une forme susceptible d'être un panneau, envoie deux informations au Sign Detector : bb et sign.

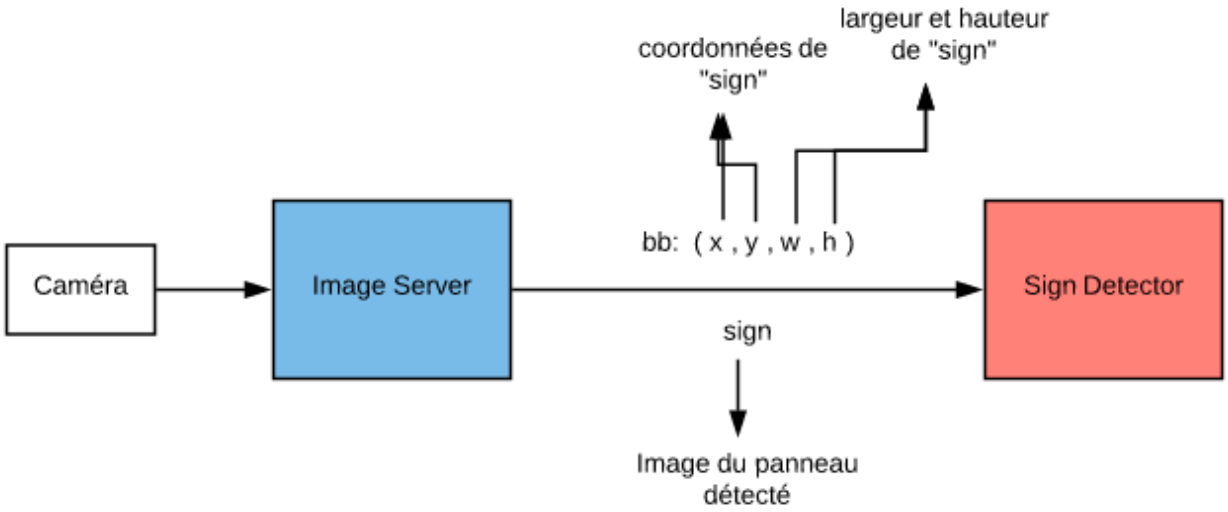

FIGURE 20 – Données reçues par le Sign Detector

Ce dernier est une image rectangulaire d'une partie d'image méritant d'être analysée. La bounding box, "bb", contient les coordonnées (x, y) de la détection du panneau sur la route. La route est représentée en 2 dimensions grâce à l'Image Server qui redresse l'image reçue par la caméra. Toute position sur la route peut donc être définie grâce à une coordonnée en X et en Y. Les axes sont tels que représentés sur la Figure 21.

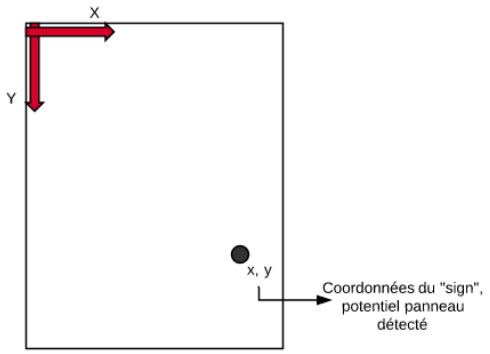

FIGURE 21 – Axes X et Y formant le repère de la route

La bounding box contient également la largeur et la hauteur (w, h) de l'image "sign" envoyée (voir Figure 22).

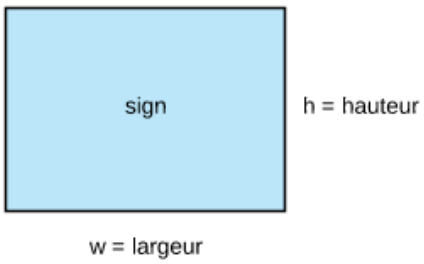

FIGURE 22 – Largeur et hauteur de l'image "sign" reçue

Le programme incrémenté est divisé en 3 modules.
Le premier vérifie les dimensions du panneau reçu. Il utilise pour cela la largeur et la hauteur reçues dans la bounding box. Si l'image est trop grande ou trop petite, de dimension peu semblable à celle d'un panneau, il est fort probable qu'il s'agisse d'une fausse détection. La fonction `verif_dimension` renvoie donc un booléen : True pour une dimension correcte, False pour une dimension incorrecte. Lors de plusieurs tests, il a été estimé par le groupe qu'un panneau perçu par la caméra devait avoir une hauteur et une largeur comprise entre 10 et 38 pixels. Ceci a été rendu possible en faisant un "print" des valeurs w et h dans le terminal.

Le second module, la fonction `verif_distance`, récupère la coordonnée dans l'espace du panneau contenue dans la bounding box et renvoie True ou False selon que la position du panneau par rapport à la voiture est plausible ou non. Il est attendu que le panneau soit détecté sur le côté droit de la route à une distance ni trop proche ni trop éloignée de la voiture. Par exemple, une détection située dans le fond ou sur le côté gauche de la route peut directement être considérée comme un bruit et ne pas être prise en compte par le programme car on sait à priori qu'un panneau de signalisation routière ne se trouve pas dans cette zone-là. La Figure 23 illustre ce propos.

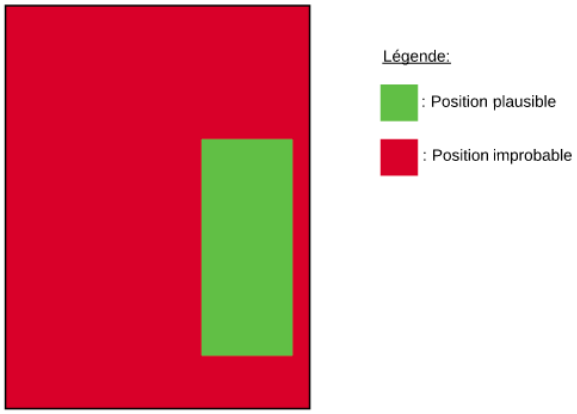

FIGURE 23 – Position probable/improbable d'un panneau de signalisation sur le plan de la route

Le dernier module traite l'image contenue dans l'argument "sign". Celui-ci est une image et toute image est en réalité une matrice de pixels. Chaque pixel est défini par une couleur qui elle même est une combinaison des 3 couleurs primaires : le bleu, le vert et le rouge. Chaque pixel renferme donc un triple d'entiers variant de 0 à 255. Dans notre cas, l'ordre des couleurs dans le triple est bleu - vert - rouge. La Figure 24 reprend ce concept ainsi que quelques exemples de valeurs BGR (blue - green - red) correspondant à des couleurs pures.

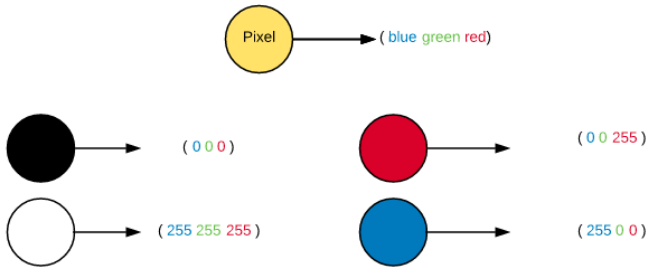

FIGURE 24 – Triples de certaines couleurs pures : noir pur, blanc pur, rouge pur, bleu pur

Les panneaux "Tourner à droite", "Tourner à gauche" et "Stop" sont distingués en regardant la couleur dominante de l'image reçue. La fonction `determiner_couleur_dom` reçoit les arguments "sign" et "bb" et compare la moyenne de teneur en bleu et de rouge de l'image contenue dans "sign". Si la moyenne de rouge est plus grande que la moyenne de bleu, l'image est susceptible d'être un panneau "Stop". Au contraire, si le bleu est le plus dominant, elle est susceptible d'être un panneau "Tourner à gauche" ou "Tourner à droite".

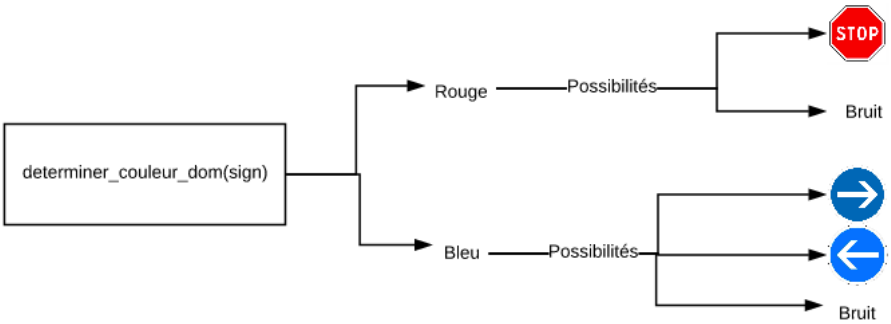

FIGURE 25 – Schéma descriptif de la fonction déterminer_couleur_dim

La combinaison de ces trois modules permet de déterminer avec fiabilité si l'image détectée est un panneau et, le cas échéant, lequel. Cette combinaison est faite grâce à des conditions if. 3 conditions doivent être respectées pour qu'un panneau soit envoyé à

la state machine. Les fonctions `verif_dimension` et `verif_distance` doivent toutes les deux renvoyer True et l'image doit avoir une couleur dominante rouge ou bleue. Le cas d'un panneau "Stop" est le plus simple car une fois les deux premières conditions respectées, il suffit de capter une image de couleur dominante rouge pour pouvoir confirmer qu'il s'agit de ce panneau. Lorsque l'image est de couleur dominante bleue, il peut s'agir soit d'un panneau "Tourner à gauche", soit d'un panneau "Tourner à droite". Il reste à différencier ces deux derniers. Pour cela, on peut ne s'intéresser qu'à la teneur en rouge d'une moitié d'image. Cela peut sembler contre-intuitif mais en analysant le problème de plus près, on se rend compte que si on divise le panneau en deux parties au niveau de sa largeur, on s'attend à trouver une couleur dominante se rapprochant plus du blanc dans la moitié de l'image contenant la pointe de la flèche car plus de pixels blancs y sont présents. Or, à une couleur se rapprochant du blanc est associée un triple d'entiers élevés pouvant aller jusqu'à maximum (255 255 255), blanc pur. La couleur dominante dans le côté de la flèche aura donc une valeur élevée pour le bleu mais également un valeur relativement élevée en vert et rouge (du moins plus que l'autre moitié). Dès lors, on peut comparer la teneur en rouge de chaque moitié de panneau et celui ayant le plus de rouge contiendra en fait une couleur globale se rapprochant plus du blanc que l'autre moitié.

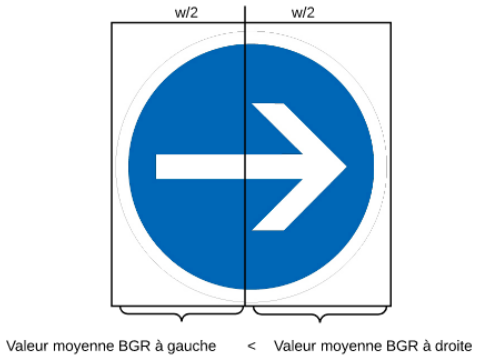

FIGURE 26 – Cas d'un panneau "Tourner à droite" : la teneur en rouge est plus élevée du côté de la flèche

Dans cette même logique, il aurait été possible de faire la même chose en comparant la teneur en vert. Cependant, l'analyse du rouge a été préférée car la couleur du bord de la route étant verte, les résultats auraient pu être faussés si l'image "sign" détectée comportait une fraction bord de route.

Voyons sur la Figure 27 l'exemple de la détection d'un panneau "Tourner à droite" ou "Tourner à gauche".

```
    elif couleur_dom == "bleu" and verif_distance(x0, y0) == True and
verif_dimensions(w, h)==True:
        print (np.sum(sign[:,:int(w/2),2])/(h*(w/2)), np.sum(sign[:,int
(w/2):,2])/(h*(w/2)))
        if np.sum(sign[:,:int(w/2),2])/(h*(w/2)) > np.sum(sign[:,int
(w/2):,2])/(h*(w/2)):
            sign_detected = "TURN LEFT"
        else:
            sign_detected = "TURN RIGHT"
```

FIGURE 27 – Code : détection des panneaux "Tourner à droite" et "Tourner à gauche"

Lorsqu'un panneau est confirmé, une valeur de type string est attribuée à la variable `sign_detected` et sera retournée sous forme de dictionnaire et pourra être lue par la State Machine. Dans les différentes arènes et vidéos de simulation, les panneaux sont détectés par défaut à la coordonnée (x, y) = (100, 100). La fonction `verif_distance` a donc été ignorée lors des tests en simulation. Dans le cas contraire, les panneaux sont systématiquement ignorés. Ceci étant fait, les panneaux étaient correctement détectés dans les différentes arènes.

4.4.3 State Machine

La voiture détecte désormais les situations de route ainsi que les panneaux de signalisation. Il reste à la doter d'un centre de décision. C'est l'objet de cette section. Le programme State Machine a été rédigé dans l'optique de décider quel trajectoire entreprendre après avoir traité les différentes informations reçues par le Path Detector et le Sign Detector. Ce programme envoie ensuite les informations nécessaires à l'Arduino par l'intermédiaire du Driver. La Figure 28 montre les grandes étapes de fonctionnement de la version de la State Machine que le groupe est parvenu à coder. Après explication de celle-ci, il sera ensuite fait mention d'une meilleure version de ce programme imaginée par le groupe mais qui n'a pas encore réussi à être codée.

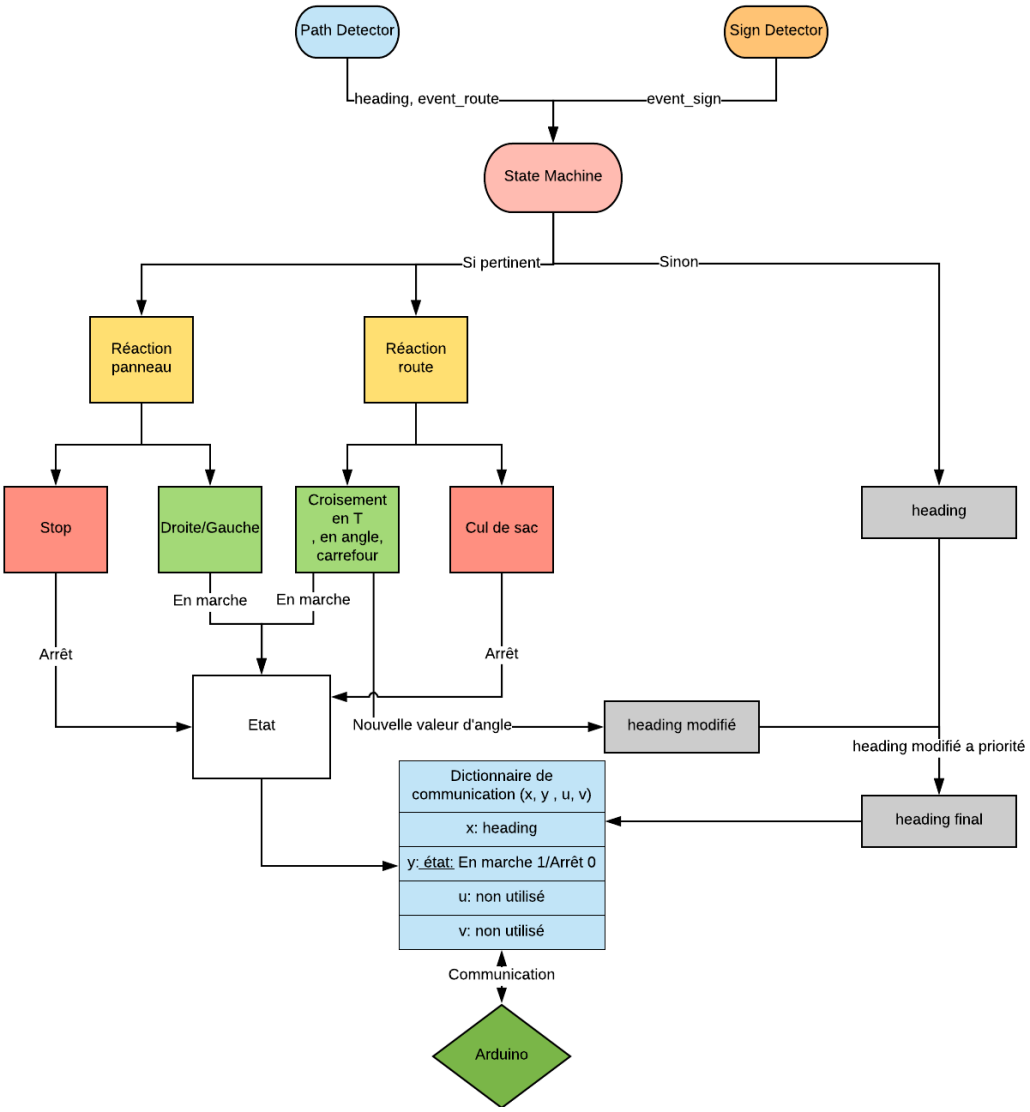

FIGURE 28 – Organigramme du programme State Machine actuel

Tout d'abord, un état de marche/d'arrêt a été implémenté dans le code afin de savoir en permanence comment se comporte le véhicule. Si le Sign Detector détecte un panneau "Stop" ou que le Path Detector détecte un cul de sac, l'information est directement transmise aux moteurs contrôlés par l'Arduino grâce à l'envoi d'un quadruplet et l'état devient "Arrêt". Toute autre situation conserve l'état comme étant "En marche". La communication entre l'Orange Pi et l'Arduino sera détaillée dans le sous-chapitre suivant.

Dans sa version actuelle, la State Machine récupère les informations du Path Detector et du Sign Detector. Elle reçoit aussi un angle (heading) à entreprendre de la part du Path Detector. Si aucune situation de route particulière n'est détectée, cet angle est correct et est communiqué à l'Arduino. Cependant, il est possible que la voiture doive ignorer cette valeur d'angle lors de situations de route et/ou de signalisation particulière. En effet, la voiture doit par exemple être capable de tourner à droite lorsqu'elle arrive à

un virage en angle vers la droite et non de continuer tout droit vers le bord de route. La State Machine modifie alors la valeur du heading de manière à ce que la voiture tourne afin de suivre la route. Ce nouvel angle remplace l'angle reçu du Path Detector et est envoyé à l'Arduino.

En simulation, avec ce programme, la voiture tourne aux croisements sans toutefois obéir au code de la route. Cependant, elle s'arrête quand même lorsqu'un cul de sac ou un panneau "Stop" est détecté.

Cette version est incomplète. En effet, elle ne prend pas en compte les instructions des panneaux de signalisation "Tourner à droite" et "Tourner à gauche". Le groupe a manqué de temps pour incrémenter cela. Voici néanmoins l'explication de comment celui-ci s'y serait pris. La Figure 29 reprend une proposition de fonctionnement pour la State Machine.

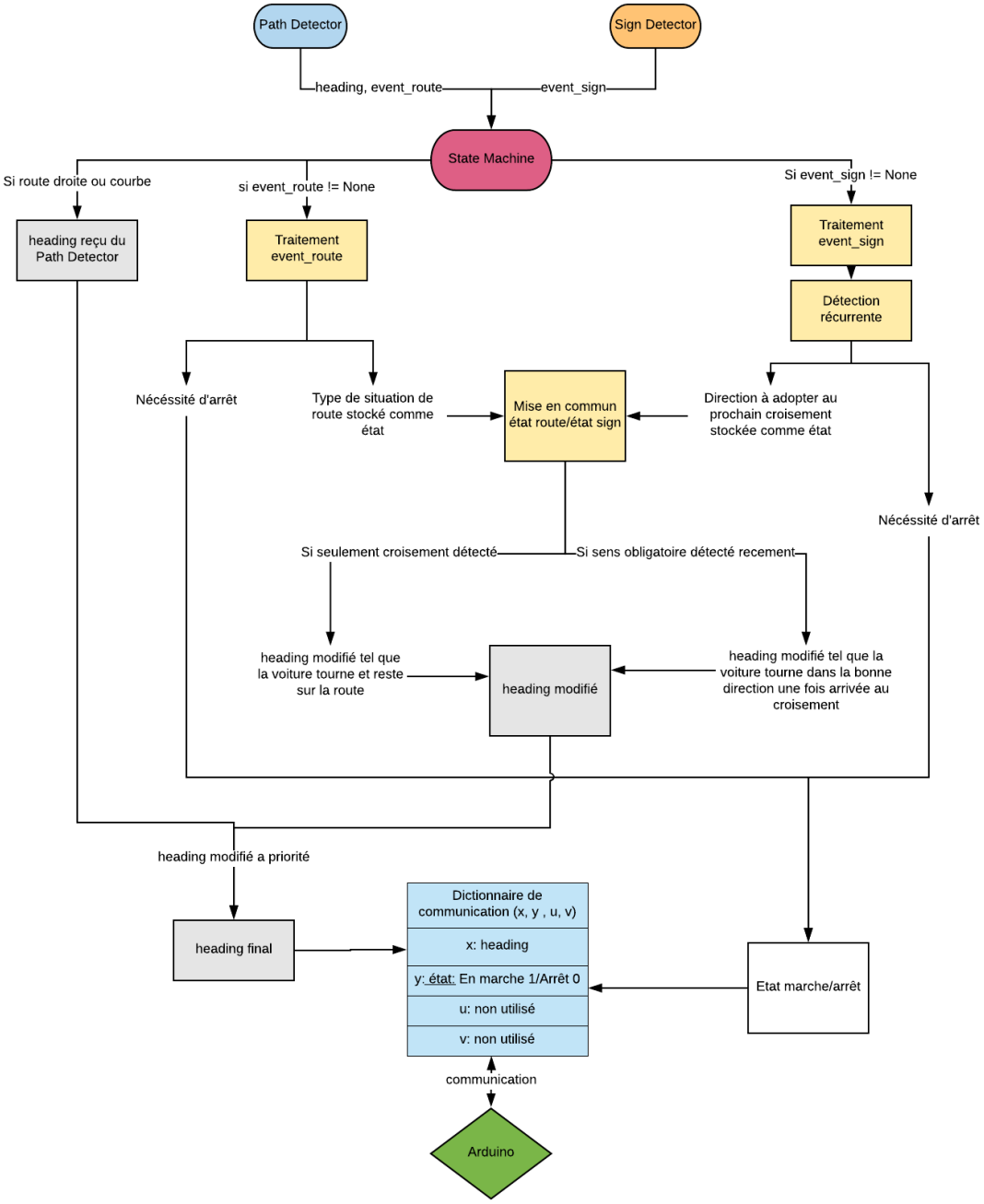

FIGURE 29 – Proposition de fonctionnement de la State Machine

Cette proposition reprend évidemment les différents points discutés ci-dessus (état marche/arrêt, angle modifié en cas de situation de route particulière). Elle se distingue à deux niveaux.

Premièrement, une fonction vérifie que la détection d'un panneau de signalisation est récurrente. En effet, si le Sign Detector renvoie plusieurs fois d'affilée une même détection, celle-ci peut être jugée très fiable et sera retenue. Dans le cas contraire, elle sera ignorée.

Ensuite, l'application du code de la route se fait comme suit. Les différentes situations de route tout comme les détection des panneaux de signalisation sont stockés dans des variables d'état. Ceci permet la mise en commun des situations et des détections. L'exemple décrit sur la Figure 30 décrit bien cela.

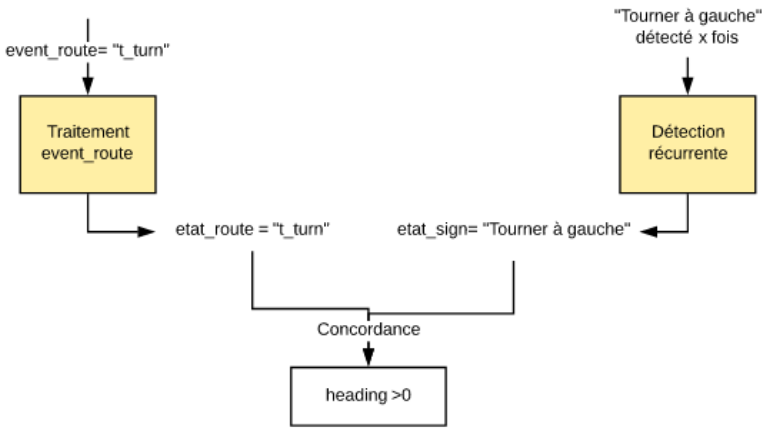

FIGURE 30 – Exemple de mise en commun des états de route et de sign

Le Sign Detector a détecté plusieurs fois un panneau de sens obligatoire "Tourner à gauche". La State Machine enregistre donc cela dans une variable d'état, appelée `etat_sign`. De cette façon, l'information n'est pas perdue lorsque la voiture continue son chemin. Il est évident que celle-ci doit attendre d'avoir un croisement valide, c'est à dire avec une voie dégagée sur la gauche dans ce cas-ci, avant de tourner. Lorsque le Path Detector détecte un croisement, dans ce cas-ci en T, l'information est gardée dans une autre variable d'état, appelée `etat_route`. L'instruction de tourner, via une modification de l'angle heading, n'est donnée que lorsque `etat_route` et `etat_sign` concordent. Ceci peut se faire grâce à une condition if. Une fois le virage effectué, les états `etat_route` et `etat_sign` sont remis à une valeur None afin de retrouver un mode de conduite "normal". L'angle heading modifié est envoyé à l'Arduino et a priorité sur le heading récupéré du Path Detector. Cette même logique peut-être appliquée dans le cas d'un panneau "Tourner à droite".

4.5 Communication Orange Pi/Arduino

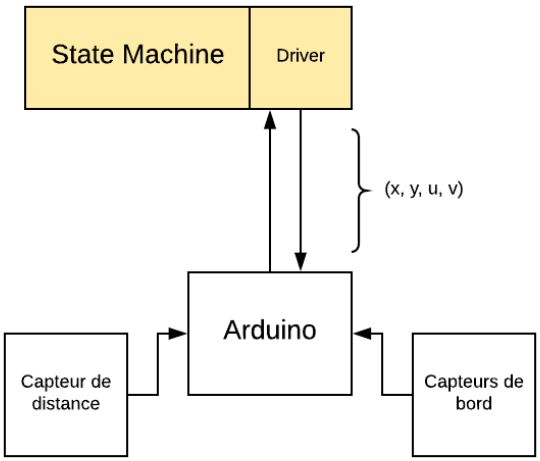

FIGURE 31 – Communication entre l'Orange PI et l'Arduino via le Driver

Après toutes ces explications sur la manière dont la state machine fonctionne, il faut que celle ci puisse transmettre les informations importantes à l'Arduino pour qu'elle effectue les changements moteurs nécessaires. Pour cela, il a été mis à la disposition de tous, un programme au sein de l'unité centrale, appelé Driver. Ce programme permet l'envoi et la réception, entre l'Orange PI et l'Arduino, de tuples de 4 éléments dont les deux premiers éléments sont des nombres de type entier et les deux suivants sont des nombres de type float. Ce quadruplet est libre d'être utilisé comme le groupe le souhaite.

Dans l'état actuel de l'avancement de ce prototype, ce quadruplet est utilisé de la façon suivante :
Le premier élément entier, le x, indique l'état du véhicule : à l'arrêt/en marche. Donc quand la state machine reçoit un ordre de démarrage, celle ci envoie un chiffre (en l'occurrence 1) que l'arduino va interpréter comme la mise en marche des moteurs. A l'inverse, lorsque la state machine détecte ou reçoit une commande stop, elle envoie un autre chiffre (0 dans ce cas ci) et l'arduino interprète ce chiffre par l'arrêt des moteurs.
Le premier élément flottant, le u, transmet l'angle calculé par l'unité centrale. La communication est fonctionnelle, mais par manque de temps, il n'a pas été possible d'implémenter une fonction qui interprète cet angle en tension moteur dans l'arduino.

Dans l'idéal, une fonction dans l'arduino permettant d'effectuer les modifications moteurs en fonction de l'angle envoyé par la state machine, ainsi qu'un envoi par l'arduino des données issues des détecteurs de route grâce à l'utilisation d'un troisième élément du quadruplet aurait été fait. Mais par manque de temps, ces idées ne resteront que théorique.

4.6 Résultats

Les sections précédentes décrivent le prototype réalisé, l'interaction entre les différents modules et composants ainsi que les codes développés afin de faire rouler la voiture par elle même. Cette section décrit les résultats auxquels le groupe est arrivé.

En ce qui concerne les résultats en simulation, l'équipe est parvenue à avoir une voiture se déplaçant sans problème sur toutes les arènes mises à disposition, à l'exception de l'arène Spiral où la voiture sort de la route lors de virages trop serrés. La voiture avance tout en restant au milieu de la route, reconnaît les croisements vers lesquels elle se dirige et tourne pour continuer son chemin dans l'arène. Les panneaux y sont détectés avec succès et la voiture s'arrête lorsque celle-ci capte un panneau de signalisation "Stop". Les panneaux "Tourner à droite" et "Tourner à gauche", bien que détectés, ne sont pas pris en considération au croisement qui suit.

Dans l'application pratique, une fois la voiture actionnée à distance et l'image et les couleurs calibrées, la voiture est capable de détecter les panneaux de signalisation face à elle mais également de détecter les situations de route malgré les bruits. Elle avance tout droit et les moteurs s'arrêtent lorsqu'un panneau "Stop" ou une voie sans issue est détectée. On constate donc que les programmes Sign Detector et Path Detector sont fonctionnels.

5 Bilan sur le fonctionnement de l'équipe

Lors du premier quadrimestre, le groupe a pris du retard dû à une mauvaise organisation et à un problème de communication entre les membres de l'équipe. L'analyse du SWOT de mi-parcours a permis au groupe d'identifier ses principales forces et faiblesses ainsi que les menaces et les opportunités de son environnement.

1. Strengths
   - Bonne entente dans le groupe.
   - Volonté de travailler en équipe.
2. Weaknesses :
   - Problèmes d'organisation :
     - Procès-verbaux mal rédigés.
     - Manque de clarté dans la répartition des tâches.
   - Problèmes de communication :
     - Informations dispersées entre deux canaux de communication.
     - Parution tardive des procès-verbaux.
3. Opportunities :
   - Échange d'informations et de matériel avec un autre groupe.
   - Aide du site OpenClassRooms dans l'apprentissage du code C++ [8].
   - Local à la disposition du groupe.
4. Threats :
   - Problèmes de batterie.

La bonne entente entre les membres du groupe et la volonté de travailler ensemble sont les deux plus grandes forces de l'équipe. Il est beaucoup plus agréable de se retrouver et le travail est beaucoup plus efficace lorsqu'il n'y a pas de tension et que tout le monde s'entraide pour atteindre l'objectif commun.

8. ROUSSEAU, Jean-Noël, *Programmez vos premiers montages en Arduino dans OpenClassrooms.*, Site web sur INTERNET `https://openclassrooms.com/courses/programmez-vos-premiers-montages-avec-arduino`, 2016

Par contre, la communication était difficile entre les membres de l'équipe car deux canaux de communications étaient mis en place et certaines informations n'étaient pas présentes sur les deux canaux. Puisque chacun utilisait son moyen de communication préféré, tout le monde n'avait pas tous les renseignements et cela a causé des problèmes. En effet, certains n'étaient pas présents à toutes les réunions ou sessions de travail. Les procès-verbaux étaient aussi rarement mis à disposition dans les temps et n'étaient pas très utiles étant donné qu'ils n'étaient pas rédigés correctement. Les informations qui y étaient reprises n'étaient pas claires. Il n'y avait pas de description précise des tâches à réaliser par chacun. Rien n'était prévu non plus concernant l'approbation du PV précédent.
Concernant l'exécution des tâches, il a aussi fallu apporté une amélioration car au cours de certaines sessions de travail, il est arrivé que le groupe entier soit bloqué sur un problème matériel ou de programmation. Dans cette situation, une partie du groupe aurait pu réaliser une tâche différente pendant que les autres membres de l'équipe essayaient de résoudre le problème.

A la fin du premier quadrimestre, il est évident que le groupe avait plus de faiblesses que de forces. Cependant grâce à l'analyse du SWOT, les faiblesses ont été identifiées et tout a été mis en oeuvre pour y remédier. Le groupe a eu un dernier souci d'organisation lors du premier test dans les arènes. En effet, tout le monde ne pouvait pas travailler sur le prototype en même temps et de ce fait certains membres du groupe n'étaient pas productifs. Ce problème a été réglé dès le deuxième test : une partie du groupe rédigeait le rapport alors que les autres s'occupaient de la voiture. Pour remédier au problème de communication, les membres ont tout simplement vérifié régulièrement sur les deux canaux utilisés qu'aucune information importante n'avait été émise.

En ce qui concerne l'organisation, le groupe a décidé d'utiliser Trello [9] . Cet outil de gestion de projet permet de créer des tableaux reprenant les tâches à faire, les tâches en cours d'exécution, celles qu'il faut vérifier et les tâches terminées. Son utilisation permet également aux différents participants de communiquer à tout moment depuis un smartphone ou un PC. L'amélioration des procès-verbaux et leurs parutions plus rapides ont contribués à une meilleure organisation. Un bon procès-verbal a été utilisé comme exemple et le rédacteur était soumis à une contrainte de temps. En effet, le PC devait être rédigé au plus tard pour le vendredi suivant la réunion. En ce qui concerne les éléments matériels extérieurs à l'équipe, l'achat de la nouvelle batterie a été très utile.

9. Trello, *Tableaux d'organisation*, `https://trello.com/`

L'analyse du SWOT de fin de projet permet de comparer les forces et faiblesses du groupe avec celles identifiées à mi-parcours.

1. Strengths :
   - Bonne entente dans le groupe.
   - Plus de motivation (Approche des deadlines).
   - Meilleure organisation :
     - Procès-verbaux améliorés.
   - Meilleure communication :
     - Informations importantes concentrées dans un seul canal.
     - Parutions des procès-verbaux plus rapides.
2. Weaknesses :
   - Problème dans l'exécution des tâches lors des tests.
3. Opportunities :
   - Nouvelle batterie.
   - Local à la disposition du groupe.
   - Utilisation de Trello afin d'améliorer l'organisation.
   - Arènes de test très libres et donc possibilité de rester plus longtemps sans gêner d'autres groupes.
4. Threats :
   - Manque de temps.
   - Manque d'expérience en Latex.

L'analyse SWOT a été très utile. Elle a aidé l'équipe à se remettre en question et à corriger ses faiblesses. Au fil des semaines, l'évolution du groupe était très positive et lorsqu'un problème apparaissait, il était rapidement identifié et résolu.

6 Conclusion

La création d'une voiture "autonome" fut une première occasion de se comporter comme un réel "ingénieur" travaillant sur un projet au sein d'une équipe.

Vu la complexité du projet, les problèmes rencontrés nécessitaient de la rigueur pour être corrigés. Tout d'abord, identifier la source du problème, la comprendre pour ensuite pouvoir y trouver une solution.
Afin de mener à bien ce projet, il a fallu travailler par étapes. Premièrement prendre connaissance du cahier des charges et modéliser un premier jet du prototype afin que ses composants s'articulent entre eux de manière cohérente et sans se gêner, par exemple : pas de câbles dans le champ de vision de la caméra.
Ensuite seulement a commencé l'analyse des codes existants afin de pouvoir commencer à programmer les fonctionnalités manquantes nécessaires à l'automatisation de notre voiture : la reconnaissance du chemin et du code de la route et la prise de décision.

Ce projet est une introduction à certains outils utilisés par les ingénieur. D'abord les tests en simulation qui modélisent un cas "parfait" car il n'y pas d'interférences et les tests dans le monde physique, comme les arènes de test où le prototype évolue dans un environnent réel où tout ne se passe pas comme prévu. La simulation est un outil très utile car il permet de tester le bon fonctionnement d'un code sans avoir recours à un prototype. Les tests dans le monde physique servent à régler des problèmes inobservables dans la simulation car tout ne marche pas comme dans la théorie.

Au cours de ce projet, il a été remarqué que la qualité du matériel a de l'importance dans le bon fonctionnement du prototype : une mauvaise batterie peut mettre à mal la communication entre les composants, les moteurs peuvent être asynchrones, une caméra de basse résolution entraîne une moins bonne détection de la route.

Comme dans tout travail d'ingénieur, ce projet fut un travail d'équipe. Cela demande une certaine coordination, une bonne division du travail tout en exploitant au maximum le potentiel de chacun. Il a donc fallu s'améliorer et apprendre des erreurs passées pour former une équipe plus efficace. Le SWOT de chaque semaine a mis en évidence les faiblesses du groupe et a poussé ce dernier à les transformer en force pour la suite du travail.

Le groupe a répond à une partie du cahier des charges. Il a réussi à construire un prototype de voiture conforme aux normes de sécurité, capable de se mouvoir, d'utiliser des capteurs infrarouges, de reconnaître les panneaux de signalisation, de détecter la situation de route face à lui et de prendre des décisions. C'est dans la phase finale que le groupe a manqué de temps : communiquer les bonnes informations à l'arduino qui se chargerait d'appliquer les bonnes tensions aux moteurs. L'équipe a cependant émis des suggestions pour la résolution des problèmes restants.

7 Bibliographie

— BUSHMANN, Isidor, *Batteries in a portable world*, pages 11,18-21, 1997

— IR sensor, *IR Sensor for Obstacle Avoidance KY-032*, Site web sur INTERNET `http://irsensor.wizecode.com/`, 2016

— Lucid Software, *Tutoriel sur les diagrammes de séquence*, Site web sur INTERNET `https://www.lucidchart.com/pages/fr/tutoriel-sur-les-diagrammes-de-sÃl'quence`, 2018

— ROUSSEAU, Jean-Noël, *Programmez vos premiers montages en Arduino dans OpenClassrooms.*, Site web sur INTERNET `https://openclassrooms.com/courses/programmez-vos-premiers-montages-avec-arduino`, 2016

— Toshiba, H-bridge datasheet *Toshiba Bi-CD Integrated Circuit Silicon Monolithic TB6612FNG*, 2007

— TP-Link, Site web sur INTERNET *10400mAh Power Bank TL-PB10400*, `https://www.tp-link.com/us/products/details/cat-5524_TL-PB10400.html`, 2017

— Trello, *Tableaux d'organisation*, Site web sur INTERNET `https://trello.com/`,2017

— Université Virtuelle, *FAQ*, Site web sur INTERNET `https://uv.ulb.ac.be/mod/forum/view.php?id=303348`, 2017/2018

— Université Virtuelle, *Architecture logicielle*, Site web sur INTERNET `https://uv.ulb.ac.be/pluginfile.php/1041625/mod_resource/content/1/Annexe_software_architecture.pdf`, 2017/2018

— VALLVERDU, Germain,Université de Pau, *Principe de fonctionnement des batteries au lithium*,pages 12-16, 22 juin 2011