



ECOLE
POLYTECHNIQUE
DE BRUXELLES

RAPPORT DE PROJET MULTIDISCIPLINAIRE BA1

Pari Tesla Bacar

Groupe 30 :

Nina MANIVONG
Maxime PUISSANT
Théo SACLIER
Mohamed SY
Anthony TERROIR
Julien VAN DELFT
Andrei Robert VIDAN
Manon WASTIAUX

Chef d'équipe :

Billy TRAN

Superviseur :

Philippe BOUILLARD

Lecteur :

Peter BERKE

Année 2017-2018

Abstract

Ce document contient 8507 mots.

Dans le domaine de l'automobile, la conception d'une voiture autonome est l'un des plus grands challenges pour les ingénieurs. En effet, de nos jours, un véhicule "robotisé" révolutionnerait le milieu de la technologie et de l'automobile. C'est pourquoi cette année, notre équipe a pour but de créer un véhicule fiable, autonome et sécurisé respectant un cahier des charges bien précis : il sera capable, grâce à des programmes, de réaliser différents types de manœuvres, suivre la route et ses panneaux de signalisation, et éviter les obstacles. Notre travail se concentrera principalement sur la conception d'un modèle réduit et sur la réalisation de programmes informatiques pour contrôler celui-ci, plus précisément, un code pour détecter la trajectoire que le véhicule doit emprunter ainsi que la reconnaissance des panneaux stop et sens obligatoire. Ces derniers, couplés, assureront une pleine autonomie de la voiture.

Mots clés : voiture autonome, programmes Python, algorithme de détection, lecture de panneaux.

Nombre de mots : 140

Table des matières

1	Introduction	5
1.1	Contextualisation	5
1.2	Annonce du contenu	5
1.3	Objectifs	6
2	Rappel de la problématique	7
2.1	Tâches	7
2.2	Contraintes	8
3	Construction du prototype	9
3.1	Mécanismes de déplacement	9
3.2	Description des composantes	10
3.3	Agencement des composantes	11
4	Fonctionnement du véhicule	14
4.1	Reconnaissance des panneaux : Sign Detector	14
4.2	Reconnaissance des zones accessibles : Path Detector	16
4.3	State Machine	18
4.4	Communication Arduino/Orange Pi	19
4.5	Manoeuvres	20
4.5.1	Implémentation des manoeuvres à l'Arduino : Langage C	20
4.5.2	Manoeuvres choisies	20
5	Bilan global	23
5.1	Simulation	23
5.1.1	Sign Detector	23
5.1.2	Path Detector	24
5.2	Passage de la simulation à la pratique	26
5.2.1	Modélisation	26
6	Fonctionnement du groupe	31
6.1	Organisation du groupe	31
6.2	Canaux de communication	32
6.3	SWOT	32
7	Conclusion	35

Table des figures

3.1	Orange Pi	10
3.2	Caméra	10
3.3	Optique	10
3.4	H-Bridge	11
3.5	Arduino NANO	11
3.6	Capteur de proximité	11
3.7	Capteur de bord	11
3.8	Prototype 3D latéral	12
3.9	Légende des schémas du prototype 3D	12
3.10	Prototype 3D oblique	12
3.11	Prototype 3D haut	13
3.12	Prototype 3D de l'arrière	13
4.1	Organigramme des liens entre les programmes	14
4.2	Image vue en matrice	15
4.3	Détection du panneau stop	15
4.4	Détection du panneau sens obligatoire à gauche	16
4.5	Le Path Detector selon la deuxième méthode	17
4.6	Organigramme du code de la State Machine	18
4.7	Organigramme de la communication	19
5.1	Simulation détectant les éléments du décor et les panneaux	23
5.2	Simulation ne détectant que les panneaux	24
5.3	Arène Manhattan	24
5.4	Arène Moustache	25
5.5	Arène Infinite	25
5.6	Roulement d'une roue	26
5.7	Diagramme du corps libre de la roue en statique	28
5.8	Représentation des vitesses du roulement	29
8.1	Cahier des comptes	37
8.2	Courbe de caractérisation tension/vitesse	38
8.3	Contraintes de sécurité	39
8.4	Organigramme du code du Path Detector	40
8.5	Exemple de code en langage C	41
8.6	Exemple de code du "sign detector"	42

Chapitre 1

Introduction

1.1 Contextualisation

Dans le cadre de la première année de bachelier au sein de l'Ecole polytechnique de Bruxelles, un certain nombre d'unités d'enseignement a été dédié à la réalisation d'un projet multidisciplinaire. Ce projet correspond, en l'occurrence pour l'année 2017-2018, au développement d'un véhicule autonome à échelle réduite (le chapitre 2 section I. « Rappel de la problématique » présentera plus en détail en quoi consiste le projet). Pour ce faire, le corps enseignant a réuni 6 à 8 étudiants sous la forme de groupe de travail, chaque groupe étant dirigé par un étudiant en master, lui-même ayant un superviseur. Cette introduction a pour but d'annoncer brièvement, à la fois, le contenu et les objectifs du rapport.

1.2 Annonce du contenu

Rappel de la problématique : cette partie du rapport reprend et décortique l'énoncé du projet, permettant ainsi de correctement évaluer la difficulté du projet et de visualiser la façon dont celui-ci peut être scindé en sous-problèmes de complexité moindre. On y parle également des contraintes de sécurité et de budget pour correspondre aux exigences et pour assurer aux utilisateurs une sécurité maximale.

Construction du prototype : les différents mécanismes de déplacement y sont décrits ainsi que les choix effectués concernant la réalisation du prototype y sont présentés et justifiés. Plusieurs points y seront considérés, à savoir : la description des composantes nécessaires au bon fonctionnement du véhicule, ainsi que l'agencement de ces composantes.

Fonctionnement du véhicule : cette partie reprend la description théorique des modules permettant au prototype d'être autonome, c'est-à-dire être capable de lire les panneaux de signalisation et de reconnaître les zones du sol accessibles, tout en respectant les contraintes que cela représente. De plus, c'est également dans cette partie que sera expliqué, d'une part, la technique mise en œuvre pour introduire certaines manœuvres pratiques au prototype et d'autre part, l'articulation entre ces différents modules assurant le bon fonctionnement du véhicule.

Bilan global : Dans le cadre du projet, des simulations vidéos et schématiques sont mises à la disposition des étudiants, afin que ceux-ci puissent tester si les résultats théoriques qu'ils obtiennent correspondent à leurs attentes. Une fois cela fait, il s'agit de soumettre ces résultats à un environnement bien moins idéal : la réalité. Cette section du rapport englobe alors les différents résultats opérationnels et les obstacles rencontrés lors de cette transition. De plus, la **modélisation** va permettre, grâce aux théories scientifiques de la mécanique rationnelle, de comprendre le fonctionnement du véhicule, en particulier, quand celui-ci se met en mouvement.

Fonctionnement du groupe : Il est primordial de donner des informations précises relatives à l'organisation et aux méthodes de communication du groupe. En l'occurrence , cette partie comportera deux points supplémentaires correspondant respectivement à une série d'auto-évaluation du groupe et au déroulement du projet, tout au long de l'année, dans lequel certaines pistes d'améliorations seront envisagées.

1.3 Objectifs

Ce rapport a pour objectif premier de décrire et justifier la réalisation, qui est propre à notre groupe, du projet dans son intégralité sans omettre les erreurs ou les idées auxquelles le groupe a renoncé afin d'avoir une trace écrite la plus objective possible de cette expérience.

Chapitre 2

Rappel de la problématique

2.1 Tâches

Au fur et à mesure que l'on avance dans l'Histoire de l'humanité, l'Homme développe et perfectionne de nouvelles techniques qui lui permettent de faire voir le jour à une multitude d'inventions. En effet, chaque nouvelle avancée, peu importe le domaine, amène à de nouveaux enjeux et de nouveaux challenges.

En ce qui concerne le domaine de l'automobile, les grandes firmes se concentrent depuis peu, sur deux principales recherches de développement, à savoir : la voiture électrique et la voiture autonome. C'est cette dernière qui capte toute l'attention de notre groupe de travail. Effectivement, à la suite de la présentation d'un prototype de voiture intelligente par une entreprise en concurrence directe avec la grande société automobile qu'est BACAR, il a été demandé à l'équipe de développement de cette dernière de mettre au point d'urgence un modèle autonome similaire, la « *BACAR ZEN* ». C'est dans le cadre de cette mission contre-la-montre nommée « *LE pari BACAR* » que le staff met tout en œuvre pour répondre rapidement à cette demande tout en respectant les contraintes imposées par l'environnement hyper concurrentiel de ce secteur d'activité.

Avant toute chose, il est important de conceptualiser un modèle réduit de ce véhicule intelligent capable de réaliser un certain nombre de manœuvres, tout en répondant à un ensemble de contraintes de sécurité et de résistance¹. Dans un premier temps, les résultats obtenus devront être présentés dans un simulateur mis à disposition, puis devront être concrétisés dans la réalité. Pour réaliser tout cela, il est primordial de penser à la construction et le développement du prototype ; c'est-à-dire, connaître l'utilité des différents types de composantes qui constituent le véhicule et le rendent autonome, ce qui fera donc l'objet du prochain chapitre.

1. Bureau d'Appui Pédagogique en Polytechnique (BAPP). 2017. *Le pari Bacar*. Document PDF sur INTERNET. <https://uv.ulb.ac.be/pluginfile.php/1036832/mod_ressource/content/1/Guide_BACAR_2017.pdf>.

2.2 Contraintes

Dans tout projet, il est toujours important de prendre connaissance des différents aspects qui pourraient mettre en danger les concepteurs et/ou les clients de celui-ci. Dans notre cas, un cahier des charges à respecter, repris en annexe, concernant la construction du prototype nous a été allégué. Il concerne les consignes de sécurité et d'homologation au niveau du câblage entre les différentes composantes du modèle et de la qualité du prototype principalement. En effet, ce projet nécessite l'utilisation de courant et par conséquent, il sera indispensable de manipuler avec attention les sources d'alimentation pour la sécurité de tous. De plus, pour que le véhicule soit fonctionnel et fiable, ses éléments doivent être solidaires. Il doit également respecter certaines dimensions pour être facilement manipulable.

Par ailleurs, il est nécessaire de respecter un certain budget : cela permet de cibler ses priorités, d'améliorer les prises de décision en prenant du recul et finalement d'anticiper les problèmes. Cette année, le budget est fixé à 100 euros. C'est pourquoi il faut redoubler de prudence et rester concentrer tout le long du projet.²

2. Pour plus de précision sur nos dépenses, un cahier des comptes est repris dans les annexes.

Chapitre 3

Construction du prototype

3.1 Mécanismes de déplacement

Avant de construire notre prototype, il faut d'abord penser à une structure mécanique de déplacement adaptée de manière à la rendre le plus optimale possible¹.

Pour assurer le déplacement de la voiture, cette dernière est munie de quatre roues motrices, obtenues avec le châssis du prototype commandé en ligne, chacune raccordée directement à un moteur. Généralement, les deux roues qui se trouvent à l'avant d'un véhicule sont celles qui sont motrices, mais un châssis à quatre roues motrices paraissait plus judicieux pour mieux contrôler les virages. En effet, le choix s'est porté sur ce châssis pour sa facilité à le manipuler puisqu'il est muni de quatre moteurs à engrenages recevant du courant continu.

Afin de bien contrôler le déplacement du prototype, la vitesse des roues est ajustée de la manière la plus adaptée. La vitesse normale théorique du véhicule est de 0.65 m/s et peut être améliorée par la tension de fonctionnement du moteur qui se situe entre 5 et 10V.

Il existe également d'autres types de structures mécaniques assurant la motricité des robots mobiles. Par exemple, les robots à chenilles qui présentent une bonne adhérence au sol et une facilité pour franchir des obstacles mais ne s'adapte pas à tous les chemins imposés. Ou encore les structures composées de trois roues dont une roue folle qui permettent une bonne rotation du véhicule, mais manque de précision pour les trajets rectilignes puisque composé de seulement deux moteurs, il suffit d'un léger déséquilibre de puissance entre ces deux derniers pour que la voiture tourne légèrement.

En définitive, le choix s'est porté sur le système de quatre roues motrices puisque c'est la technique la plus communément employée et bien qu'elle peut ne pas paraître être la meilleure, celle-ci assure tout de même un déplacement dans toutes les directions, et à grande vitesse si le besoin se présente.

1. Instructables. Mise à jour : 25/10/2016. *Smartphone controlled Arduino Rover*. Site web sur INTERNET.
<<http://www.instructables.com/id/Smartphone-Controlled-Arduino-Rover/>>

Voiture à 4 roues motrices	Voiture à chenille	Voiture à 3 roues
<ul style="list-style-type: none"> - <i>Avantages</i> : flexibilité de la rotation, courant sur le marché. - <i>Inconvénients</i> : les 4 roues non synchronisées = glissements et perte de puissance due aux frottements. 	<ul style="list-style-type: none"> - <i>Avantages</i> : bonne adhérence au sol et facilité de franchissement d'obstacles. - <i>Inconvénients</i> : pas en mesure de s'adapter à un chemin sinueux. 	<p><i>Avantage</i> : permet une bonne rotation du véhicule.</p> <p><i>Inconvénients</i> : n'effectue pas de trajets rectilignes avec grande précision.</p>

3.2 Description des composantes

Une des premières étapes du développement du prototype consiste bien évidemment en l'installation des différentes composantes utiles à son bon fonctionnement. Celles-ci sont fournies par l'Université Libre de Bruxelles et présentées ci-dessous.²

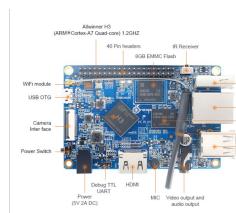


FIGURE 3.1 – Orange Pi

Unité central Orange PIPC PLUS : Cette composante complexe effectue les calculs telle une partie du processeur d'un ordinateur. Elle est munie d'une connectivité wifi et est programmée en langage Python.



FIGURE 3.2 – Caméra



FIGURE 3.3 – Optique

Caméra et optique : Une fois l'optique de grand angle montée sur la caméra, celle-ci renvoie les images qu'elle reçoit à l'unité centrale, nécessairement compatible avec cette dernière.

2. Bureau d'Appui Pédagogique en Polytechnique (BAPP). Octobre 2017. *Description des éléments matériels de la BACAR Zero*. Image sur PDF sur INTERNET. <https://uv.ulb.ac.be/pluginfile.php/1041623/mod_resource/content/1/Annexe%20-%20Hardware.pdf>.



FIGURE 3.4 – H-Bridge

H-Bridge : Cette composante est l'unité de contrôle de la puissance des moteurs, elle sert donc à commander la tension électrique aux moteurs via l'Arduino.



FIGURE 3.5 – Arduino NANO

Arduino NANO 3.0 : C'est un micro-contrôleur qui, relié à l'unité centrale et au pont H-Bridge, permet de commander les moteurs et les capteurs grâce à ce dernier.



FIGURE 3.6 – Capteur de proximité



FIGURE 3.7 – Capteur de bord

Capteurs de bord et de proximité : Ils servent à capter respectivement la présence d'un bord de route autour du véhicule et les obstacles situés dans un rayon de 10 cm du véhicule.

Maintenant que nous savons à quoi servent ces composantes, il faut penser à agencer celles-ci de manière qualitative en tenant compte, bien sûr, de la sécurité ; en raison du fait que certaines de ces composantes soient sensibles.

3.3 Agencement des composantes

Une fois le prototype construit sur base de deux châssis et de quatre roues, il a donc fallu discuter, de manière concise, sur la mise en place des différentes composantes^{3.2} à ajouter³.

3. GIAMARCHI, Frédéric, Laurent FLORES. 2007. *Construisons nos robots mobiles*. Paris : ETSF. 160 p.

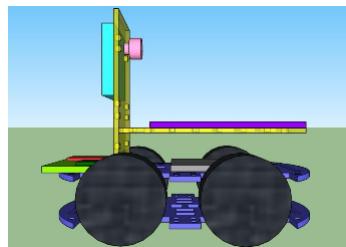


FIGURE 3.8 – Prototype 3D latéral



FIGURE 3.9 – Légende des schémas du prototype 3D

Dès le début, l'idée d'incorporer un étage supplémentaire aux deux premiers châssis du prototype s'est présentée naturellement, afin d'y placer la batterie externe assez imposante et d'éviter son échauffement. Il a d'abord été imaginé d'imprimer cet étage en 3D. Le manque de temps a finalement eu raison de cette idée. En effet, le prototype devait être monté au plus vite afin de commencer les tests. Deux plaques faites de bois contre-plaqués ont finalement été utilisées, l'une placée verticalement et l'autre horizontalement, et sont maintenues, via des équerres, par des vis et des écrous de manière à ce qu'elles s'emboîtent solidement. D'autres équerres devaient maintenir en place la plaque verticale sur le châssis supérieur mais la disposition de ces dernières, étant mal élaborée, n'était pas en concordance avec les trous déjà existants du prototype : l'utilisation d'adhésif double-face a parfaitement convenu à la situation.

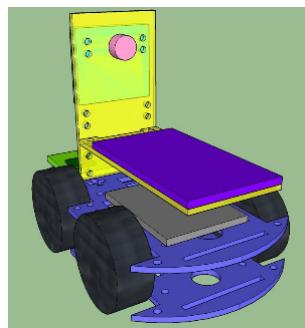


FIGURE 3.10 – Prototype 3D oblique

Ainsi, la *batterie* est installée sur la plaque horizontale au moyen de velcro. La *caméra* et le *boîtier à piles*, qui alimente les roues, sont fixés sur la plaque verticale. De plus, une fente a

étée découpée dans la plaque horizontale afin que la languette de la caméra puisse directement se brancher sur l'unité centrale, située à l'étage inférieur.

L'*Orange Pi* et le *breadboard*, contenant l'*Arduino Nano* et le *H-Bridge* (décris dans la section

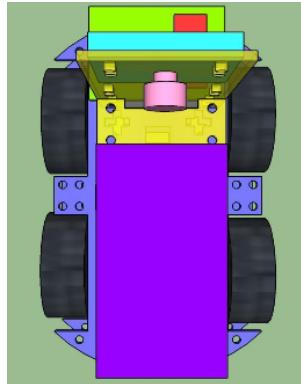
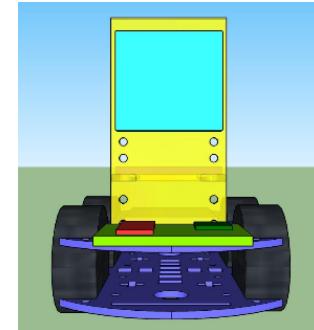


FIGURE 3.11 – Prototype 3D haut FIGURE 3.12 – Prototype 3D de l’arrière



3.2), sont alors placés sur le châssis supérieur. Pour être précis, le premier se trouve au centre du prototype. Sa position a été sujette au débat : il fallait le disposer soit dans la longueur du châssis afin de ne pas abîmer la languette de la caméra assez courte, soit dans la largeur pour avoir facilement accès aux ports USB de l'*Arduino Nano*. Le premier choix fût finalement celui appliqué puisque le câble USB est suffisamment long contrairement à la languette de la caméra. Et le second est, quant à lui, installé derrière la plaque verticale à l'aide d'adhésif double-face. Son câblage s'est d'ailleurs déroulé sans encombre.

L'*Arduino Nano*, le *H-Bridge* et les moteurs ont d'abord été reliés par de simples fils de cuivre, mais ceux-ci, étant trop fragiles, ont fini par causer une perte de tension et ont finalement été remplacés par des câbles de pontage mâle/mâle, plus résistants.

Un *interrupteur*, actionnant le prototype, a ensuite été installé entre le boîtier à piles et le *breadboard*.

Les *capteurs de bords* ainsi que le *capteur frontal* d'obstacles sont fixés à l'avant du prototype sur le châssis inférieur. Ils sont solidement maintenus en place par des colliers de serrage en plastique. Des câbles de pontage mâle/femelle sont utilisés pour les relier au *breadboard*.

En fin de compte, le prototype respecte les contraintes de sécurité et de résistance (décris en 2.2), avec une masse de 1,168 kg, 25,6 cm de longueur, 14,8 cm de largeur et 21 cm de hauteur.

Chapitre 4

Fonctionnement du véhicule

Il est important de stipuler que l'autonomie du véhicule est le but principal à atteindre. En effet, le véhicule doit pouvoir respecter un bon nombre de contraintes de sécurité (2.2) ainsi que de catégories de manœuvres grâce à des programmes tels que la reconnaissance des panneaux (*Sign Detector*) et la détection des bords de routes et des obstacles (*Path Detector*). Pour ce faire, il est primordial de passer par des simulations via la programmation de certaines composantes, décrites en 3.2, l'*Orange Pi* et l'*Arduino*.

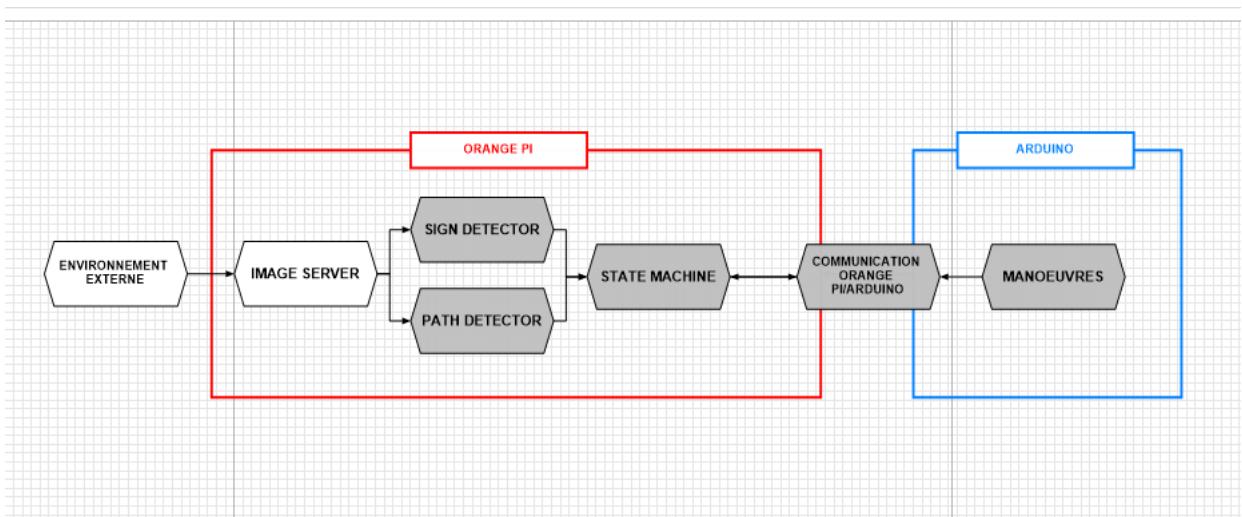


FIGURE 4.1 – Organigramme des liens entre les programmes

4.1 Reconnaissance des panneaux : Sign Detector

Tout d'abord, les images analysées par la caméra sont envoyées à l'*Orange Pi*. Ce dernier va les analyser à l'aide de l'*Image Server* puis du **Sign Detector** sous forme de matrices à trois dimensions correspondantes chacune à une couleur, c'est-à-dire, le rouge, le vert et le bleu¹.

1. The MathWorks. 2017. *colormap matrix*. Image sur INTERNET. <https://nl.mathworks.com/help/matlab/creating_image-types.html?requestedDomain=www.mathworks.com&requestedDomain=true>.

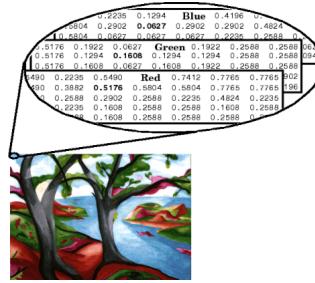




FIGURE 4.4 – Détection du panneau sens obligatoire à gauche

Dans le cas du panneau bleu, il faut également déterminer le sens de la flèche. Pour cela, il faut répéter la même opération mais avec la matrice bleue. Une fois le panneau assurément bleu, il faut diviser la matrice bleue en deux en suivant la colonne du milieu. Cela afin de comparer la partie gauche et la partie droite. En effet, si les valeurs de bleu sont plus grandes à gauche, la flèche pointe vers la gauche, l'ordinateur voyant le blanc plus bleu que le bleu lui-même. Pour vérifier cela, tous les éléments de la partie gauche sont sommés et il en est de même pour la partie droite. Ces deux sommes sont comparées et si la valeur venant de la matrice gauche est plus élevée, la flèche pointe vers la gauche. Et inversement pour la partie droite.

Les différents panneaux sont dès lors bien détectés et différenciés. Mais afin de supprimer toutes les détections parasites, le **Sign detector** ignore toutes les formes ne ressemblant pas à un panneau. En d'autres termes, si la forme du panneau ne se rapproche pas d'un carré, le panneau est ignoré. Cela est effectué via un rapport entre la longueur et la hauteur de la zone détectée par l'image server qui doit rester entre deux valeurs étalons (le code précis se trouve en annexe).

4.2 Reconnaissance des zones accessibles : Path Detector

Le **Path Detector**⁴ est une section cruciale de l'obtention du caractère autonome de la voiture. En effet, c'est la partie de la problématique qui se charge de reconnaître les zones auxquelles le prototype a accès et d'optimiser son déplacement. En d'autres termes, c'est lui qui garantit le fait que le prototype suive la route, tout en adoptant les trajectoires les plus dégagées.

Il est à noter que cette branche de code a été réalisée par le biais de deux méthodes différentes. En effet, la version finale a été élaborée lors de l'optimisation de la première. Les principes de fonctionnement distincts de ces deux approches sont détaillés ci-dessous.

La première ébauche du **Path Detector** considère que le modèle réduit évolue dans l'espace au sein d'une matrice binarisée exclusivement composée de 255 et de 0 correspondant respectivement aux zones inaccessibles et accessibles par le prototype; et ceci tout en étant

4. MASSART, Thierry. 2016. *Syllabus INFO-H-100 Informatique*. Bruxelles : Presses Universitaires de Bruxelles a.s.b.l. 217 p.

entouré d'un cercle imaginaire, centré sur la voiture⁵. En évaluant un à un les points du cercle, on détermine les bords de la route lorsque deux points consécutifs sont égaux à deux nombres différents. Une fois les deux bas-côtés relevés, leurs coordonnées sont enregistrées dans de nouvelles variables sous la forme de vecteurs. La moyenne arithmétique des coordonnées des deux vecteurs donne finalement accès au centre de la route.

En ce qui concerne l'orientation de la voiture, une fonction pré-définie fournie sur l'Université Virtuelle (plateforme informatique mise à disposition par L'Université Libre de Bruxelles pour les étudiants) *model_to_heading* a été utilisée afin d'éviter les obstacles. Cette dernière procède à un calcul d'angle entre la droite verticale passant par le centre de la voiture et la droite reliant le centre de la voiture au centre de l'obstacle. Si l'angle est négatif, le prototype tournera vers la droite tandis que si l'angle est positif, il tournera vers la gauche.

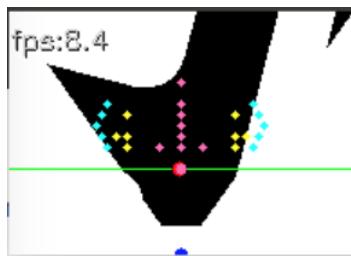


FIGURE 4.5 – Le Path Detector selon la deuxième méthode

Cette première méthode n'assurant pas parfaitement les trajectoires curvilignes, le code a été réinitialisé pour en adopter une autre. Le **Path Detector** a été reprogrammé aux moyens des fonctions données dans le fichier exemple sur la machine virtuelle, donc la méthode de détection a complètement changé. En effet, l'idée du cercle trigonométrique a été complètement abandonnée. À la place, nous avons utilisé la fonction *profile* mise à notre disposition. Cette dernière prend un nombre prédéfini de segments de longueur de plus en plus grande à gauche et à droite d'un point de référence. Ainsi, un point repère a été pris plus ou moins au centre de la matrice ($x=img_width/2$ et $y=img_height*.65$) et la fonction *profile* prend des vecteurs à gauche et à droite de ce point pour trouver le premier vecteur pointant sur un pixel «non nul» correspondant aux bords. Une fois les bords trouvés (qui sont donc bien des vecteurs), on peut faire la moyenne pour trouver le vecteur qui pointe vers le centre de la route. C'est ce dernier qui servira à orienter la voiture.

Pour que le **Path Detector** soit complet et satisfaisant, il faut s'assurer que la voiture ne cogne pas un obstacle. Pour cela, plusieurs points ont été pris et marqués en rose (voir image ci-dessus) et la voiture vérifie constamment si ces points sont bien nuls, c'est-à-dire correspondant à la route. Ils sont essentiels pour assurer le bon fonctionnement de la voiture. Par exemple, en cas de "bruits" sur les matrices, la voiture continue à chercher le centre de la route tant qu'au moins 5 points sur 7 sont nuls (correspondant à la route). Sinon, d'autres points sont pris à gauche et à droite de la voiture (marqués en jaune sur l'image). Ces points sont utilisés si la voiture détecte un obstacle devant elle et déterminent si dans un croisement en T il est possible de tourner à gauche ou à droite. Sinon, il s'agit d'une voie sans issue et la

5. L'idée de définir un environnement circulaire autour de la voiture provient du groupe 26.

voiture fait demi-tour. Ensuite, les points bleus latéraux, sous forme de flèche, symbolisent également leur usage au sein du Path Detector. Ceux-ci sont vérifiés à tout moment pour déduire si la voiture se rapproche d'un croisement où elle peut tourner à gauche ou à droite, selon l'orientation de ces points. Pour ce faire, le Path Detector vérifie pour chaque nouvelle image reçue si les points situés à gauche ou à droite de la voiture sont nuls et donc se trouvent entièrement sur la route pour créer des variables à valeur booléenne *clear_left* et *clear_right* qu'elle transmettra ensuite avec le *heading* à la State Machine. (Pour comprendre comment la voiture réagit au code, un organigramme du **Path Detector** est repris dans les annexes.)

4.3 State Machine

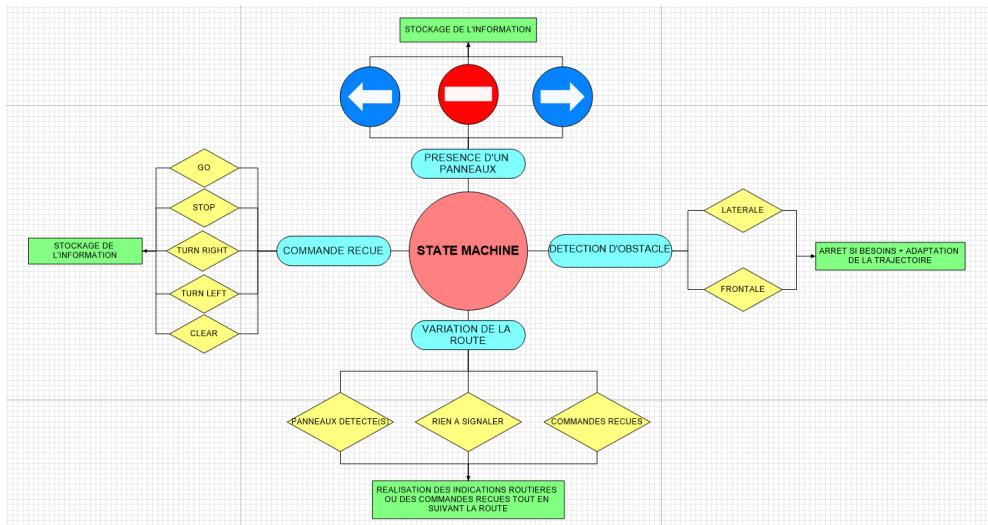


FIGURE 4.6 – Organigramme du code de la State Machine

Désormais, la gestion des indications routières et du suivi de la route étant abouties, il s'agit d'imbriquer ces deux codes. C'est là qu'intervient la **State Machine**. En effet, elle est l'unité de code qui fusionne le **Sign Detector** et le **Path Detector** de manière à ce que ceux-ci puissent être en fonction simultanément. La stratégie mise en place pour atteindre cet objectif est la suivante.

Tout d'abord, il a été nécessaire de développer un système de priorité entre la réalisation des différentes informations reçues en fonction de leur nature. Effectivement, les instructions que reçoit la **State Machine** peuvent provenir soit du **Sign**, soit du **Path** ou encore de **Commandes** à distance. L'exécution des signalisations routières prévaut sur le reste durant un temps pré-défini. En l'absence de celles-ci, ce sont les commandes qui prennent l'avantage sur la modification de la trajectoire par le **Path**.

Il s'agit dès à présent de gérer la bonne circulation des informations. En ce qui concerne la signalisation, un panneau rencontré est stocké dans une liste si et seulement si celui-ci diffère du dernier ajouté, le tout devant se faire dans un laps de temps assez court. Les commandes,

"TURN RIGHT" et "TURN LEFT", sont également gardées en mémoire dans une liste mais cette fois-ci sans condition. Il est à préciser qu'il existe une commande "CLEAR" qui permet d'effacer les éléments de la liste de commandes. Pour finir, l'angle de direction de la voiture et la possibilité pour celle-ci de tourner à droite ou à gauche sont renvoyés dans un dictionnaire par le **Path**.

En définitive, les deux listes sont lues et effectuées en respectant l'ordre des priorités. Il est important de préciser que les changements de direction issus des parties **Sign** et **Commandes** s'effectuent dès que possible. La ré-initialisation de la liste servant d'espace de stockage des panneaux s'opère une fois que l'instruction correspondant au(x) panneaux a été effectuée ; même principe pour la ré-initialisation de la liste des commandes.

4.4 Communication Arduino/Orange Pi

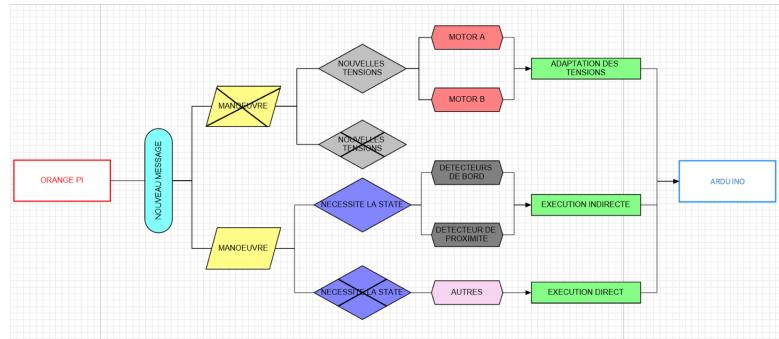


FIGURE 4.7 – Organigramme de la communication

A ce stade du développement, l'étape suivante logique correspond à la transformation des instructions transmises par la **State Machine** sous forme de tensions dans les moteurs de façon à ce que le déplacement du modèle réduit satisfasse nos attentes en ce qui concerne l'obtention du caractère autonome. Le **Code Arduino** assure parfaitement la bonne communication entre la **State Machine** et l'**Arduino** qui est le centre de contrôle des moteurs. En effet, il transmet directement les directives de la **State** sous forme de 4 variables : x,u,v,y. Chacune de ces variables représente respectivement la manœuvre envoyé par l'utilisateur via le *terminal remote* de l'**Orange Pi**, la tension à appliquer aux moteurs A et la tension à appliquer aux moteurs B (à savoir que le prototype constitue un véhicule quatre roues motrices dont les moteurs sont couplés deux à deux), et la dernière n'est pas utilisée.

Ici aussi un ordre de priorité a été établi. En premier lieu, ce sont les manœuvres qui sont effectués sans prendre en compte les valeurs de u et de v. Il est par ailleurs important de préciser que la réalisation de certaines manœuvres nécessitent non seulement l'utilisation du **Code Arduino** mais également celui de la **State Machine** et dure donc un temps qui n'est pas défini à l'avance : l'emploi des détecteurs de bords et de proximité. Chacune des autres manœuvres ne nécessite que de l'Arduino Nano et ne dure donc qu'un temps limité défini. En second lieu vient l'adaptation des tensions dans les moteurs qui permet à la voiture de suivre le choix directionnel envoyé par le **Path Detector** par le biais de la **State**. Effectivement, les

variables u et v font correspondre des tensions qui sont directement calculées à partir de l'amplitude de l'angle du *Heading*.

4.5 Manoeuvres

4.5.1 Implémentation des manoeuvres à l'Arduino : Langage C

L'arduino est une composante qui permet particulièrement de commander les moteurs et les capteurs ; les commandes étant codées uniquement en **langage C**⁶.

Les manoeuvres seront donc particulièrement implémentées dans l'environnement de compilation de l'arduino. De ce fait, lors des tests de manoeuvres, les codes doivent être implémentés dans la commande *setup()*, ainsi l'exécution du programme se fera qu'une seule fois. Mais pour la version finale du prototype, les commandes de type « avancer » doivent se trouver dans la commande *loop()* pour s'effectuer de manière continue.

Des fonctions BacarMotor particulières sont fournies afin de faciliter l'implémentation des manœuvres en langage C⁷. Donc, après avoir initialisé les moteurs à l'aide de ces fonctions ainsi que la LED, il est aisément, par exemple, d'alimenter les moteurs pour faire avancer le véhicule. Pour cela, il faut utiliser la fonction *motorX.actuate(fraction de tension)*, X étant le nom du moteur.

La commande pour avancer étant mise en place, les manoeuvres de **catégorie I** sont codées de sortes que La durée d'action soit bien déterminée à l'aide de la fonction *delay(temps voulu en seconde)*. De plus la vitesse du véhicule a été mesurée afin de pouvoir le faire avancer sur un mètre. Ainsi, le véhicule va à 0.430 m/s lorsque la tension est maximale. Mais il est à noter que le modèle de voiture à quatre roues provoque un glissement qui cause une variation de vitesse. Cela signifie que la vitesse peut varier autour de 0.430 m/s.

4.5.2 Manoeuvres choisies

Les manoeuvres ci-dessous sont reprises du cahier des charges et représentent ce que notre prototype est capable de faire grâce aux codes informatiques décrits jusqu'ici.

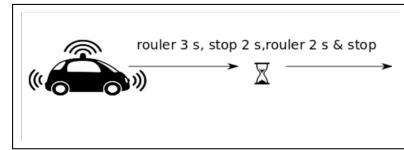
Catégorie I

Dans cette première catégorie, les manoeuvres ne nécessitent que de simples codes directement reçus de l'Arduino sans aucune interaction entre la voiture et son environnement.

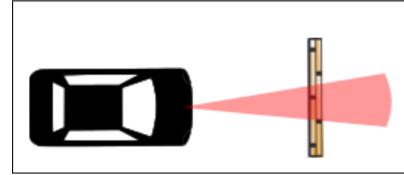
6. TAVERNIER, Christian. 2005. *Programmation en C des PIC*. Collection Technique et ingénierie - EEA. s.l : Edition Dunod, L'usine Nouvelle. 215 p.

7. NEBRA, Mathieu. Mise à jour : 23/11/2017. *Apprenez à programmer en C!*. Site web surINTERNET. <<https://openclassrooms.com/courses/apprenez-a-programmer-en-c>>.

I.3 : Le véhicule réalise une séquence d'avance de 3 sec suivie d'une pause de 2 sec puis d'une deuxième séquence d'avance de 2 sec puis d'un arrêt automatique.⁸



I.4 : Le véhicule démarre, avance et s'arrête dès qu'il détecte un obstacle à l'aide du détecteur frontal.



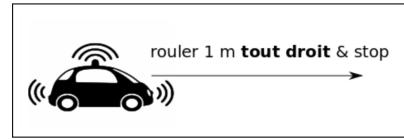
I.5 : Le véhicule démarre, avance et s'arrête automatiquement si un bord est détecté à l'aide des capteurs de bord.



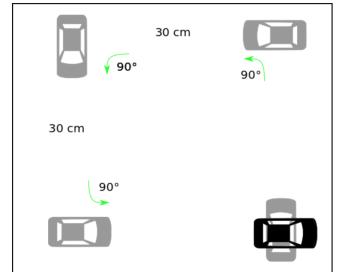
Catégorie II

Cette catégorie de manœuvres nécessite de recevoir des informations concernant la route. Cela signifie que l'utilisation des capteurs est sollicitée : ceux-ci envoient une certaine information à l'Arduino qui va les interpréter et diriger la voiture.

II.1 : Le véhicule avance sur 1 m en ligne droite et s'arrête automatiquement.



II.3 : Le véhicule suit une trajectoire carrée en avançant successivement de 30 cm et en tournant sur place de 90°. Le véhicule s'arrête automatiquement quand il revient à son point de départ (orientation non comprise).



Catégorie III

Cette catégorie met en évidence l'utilisation des différents programmes de détection de panneaux et du chemin mais de manière séparée.

8. Bureau d'Appui Pédagogique en Polytechnique (BAPP). 2017. *Le pari Bacar*. Image sur INTERNET.<https://uv.ulb.ac.be/pluginfile.php/1036832/mod_resource/content/1/Guide_BACAR_2017.pdf>.

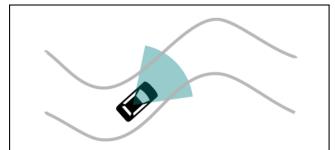
III.1a : La voiture reconnaît le panneau stop à l'aide du code du Sign Detector.



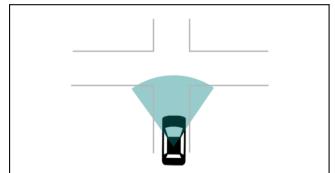
III.2a : La voiture reconnaît les panneaux de sens obligatoires à l'aide du code du Sign Detector.



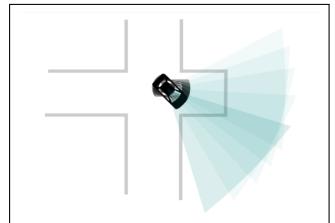
III.1b : La voiture suit le chemin à l'aide de sa caméra sans utiliser les capteurs.



III.3b : Dans un carrefour, s'il n'y a pas de panneau sens obligatoire, la voiture traverse la chaussée.



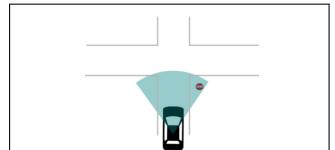
III.6b : Dans une voie sans issue, la voiture fait un tour sur elle-même.



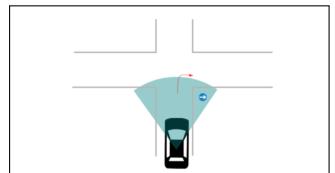
Catégorie IV

Cette catégorie nécessite l'utilisation du Path Detector couplé avec le Sign Detector.

IV.1 : La voiture détecte un panneau, marque un arrêt puis traverse la chaussée.



IV.2 : La voiture détecte un panneau sens obligatoire à droite et tourne dans cette direction.



Chapitre 5

Bilan global

5.1 Simulation

5.1.1 Sign Detector

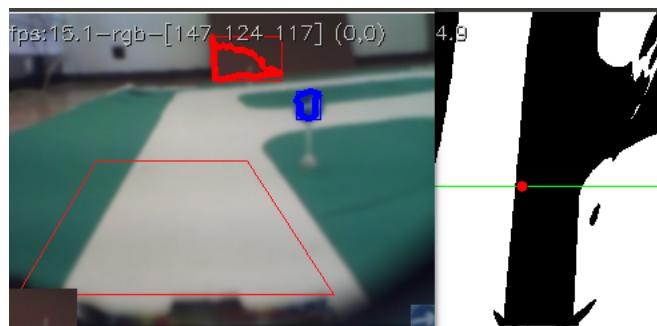


FIGURE 5.1 – Simulation détectant les éléments du décor et les panneaux

La simulation du **Sign Detector** se fait via des vidéos pré-enregistrées qui sont au nombre de 6. Ainsi comme expliqué dans le point du Sign detector, le premier code mis en place ne marchait pas toujours sur ces vidéos. De plus, dans ces vidéos, la caméra détecte volontairement des éléments du décor afin de voir si le programme parvient à gérer les détections parasites. Il était donc nécessaire de changer le code afin de régler ce souci. Comme montré sur l'image ci-joint où en plus du panneau, la voiture détecte une partie du mur.

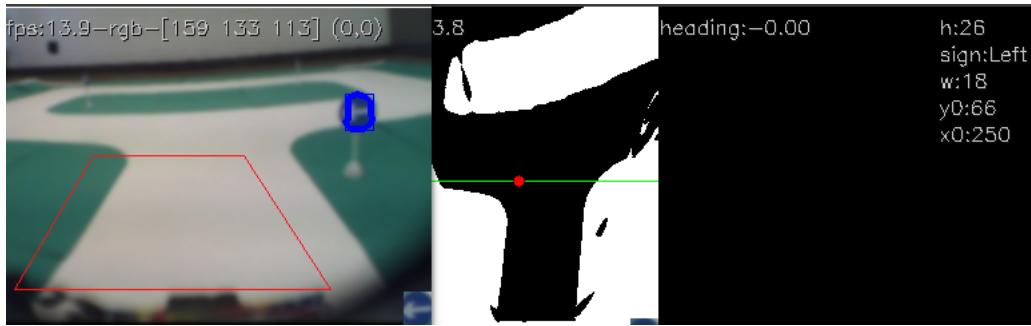


FIGURE 5.2 – Simulation ne détectant que les panneaux

Le nouveau code fonctionne à la perfection sur les vidéos pré-enregistrés. De plus, les détections parasites sont ignorées. Ceci vaut pour les simulations vidéo. Il y a un autre type de simulation développée dans le point suivant. Cette simulation est schématique mais il y a aussi des panneaux disposés sur les routes. Sur ce type de simulation le Sign Detector fonctionne également.

5.1.2 Path Detector

La simulation du **Path Detector** permet de tester l'algorithme développé dans un environnement idéalisé où le prototype ne subit aucune interférence et où la matrice image reçue est parfaite, sans aucun bruit. La simulation peut être faite sur différentes arènes, correspondant chacune à des situations que le prototype pourrait rencontrer sur un circuit de test réel, telles que *Manhattan*, qui est un circuit basique avec des virages de 90 degrés et des panneaux routiers, ou *Moustache*, qui teste la capacité de la voiture de suivre une route sinuuse et de faire demi-tour à la fin d'un chemin sans issue.

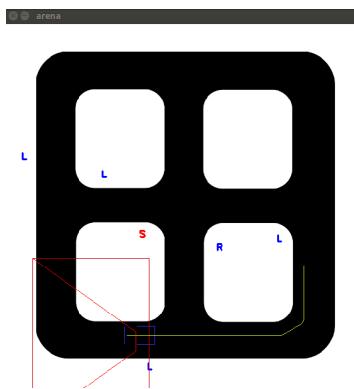


FIGURE 5.3 – Arène Manhattan

Lors des simulations, *Manhattan* est un succès pour le code du Path Detector grâce auquel la voiture reste toujours sur la route. Le code vérifie si la route est dégagée devant le prototype, qui avance de préférence tant que c'est le cas. Si la voiture se rapproche d'un obstacle (voie sans issue ou virage très brusque), alors le code vérifie l'environnement de la voiture pour voir s'il peut continuer vers la gauche ou vers la droite, comme expliqué dans la partie consacrée

au code **Path Detector**. Au tout début, Même si les panneaux routiers apparaissaient sur le circuit, le prototype ne suivait pas leurs indications car la State Machine, qui est le couplage du **Sign Detector** et **Path Detector**, n'était pas encore en cours de développement. C'est au fur et à mesure de l'avancement du projet que la state machine a pu être programmé. Ainsi, lors des simulations, la voiture sait détecter aussi bien les obstacles et les bords que les panneaux routiers.

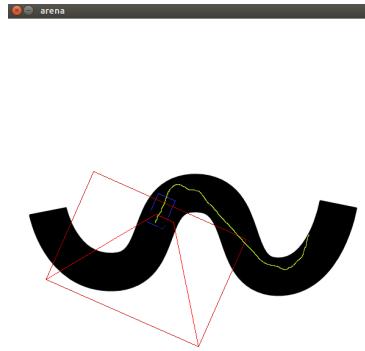


FIGURE 5.4 – Arène *Moustache*

Cependant, l'arène *Moustache* s'avérait, au premier temps, plus difficile pour l'algorithme. En effet, il y avait des hésitations par moments quand l'obstacle se trouvait à droite du prototype. Bien qu'il suivait la route et qu'il n'en sortait jamais, le prototype touchait les bords de la route lors de ces hésitations. Cette partie a donc été sujette à des améliorations ultérieures au fil de l'avancement du projet. Finalement, l'algorithme actuel permet bien au prototype de faire demi-tour à chaque fois que la route se termine.

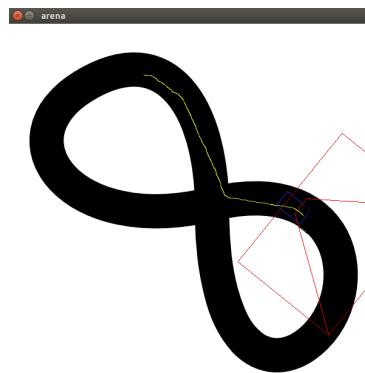


FIGURE 5.5 – Arène *Infinite*

Enfin, grâce au nouveau code (en 4.2), l'arène *Infinite* est un succès. La voiture suit toujours la route et n'a plus d'hésitations aux virages.

5.2 Passage de la simulation à la pratique

Jusqu'ici, nous n'avions considéré que les simulations mais il faut dès lors se lancer dans la réalité. Pour cela, comme dans tout projet, il est nécessaire de mettre en place des théories et de trouver une explication rationnelle aux comportements du produit à conceptualiser.

5.2.1 Modélisation

Rôle des pneumatiques

Le déplacement d'une voiture dépend de quatre forces : la force de traction, la force de retenue, la force de freinage et la force de guidage. Si elles n'existaient pas, la voiture serait immobile et incontrôlable.

Ces forces naissent au contact du sol et s'exercent sur les pneumatiques pour ensuite se transmettre aux roues, au châssis et à l'ensemble des éléments qui constituent la voiture. C'est un organe essentiel de la transmission du mouvement¹.

Les forces en présence

Comme décrit ci-dessus, nous pouvons voir la voiture comme un système de solides sur lequel s'exercent des forces.

Tout d'abord, notons qu'un roulement est la combinaison d'une translation et d'une rotation, c'est-à-dire qu'il y a translation du centre de la roue et rotation de celle-ci autour de ce centre.

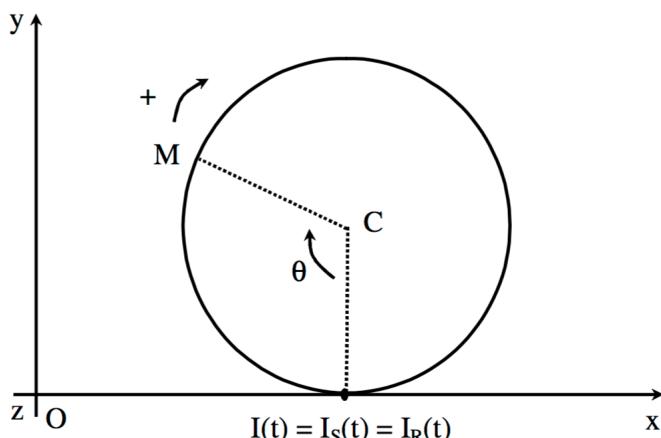


FIGURE 5.6 – Roulement d'une roue

Sur le repère ci-dessus, on définit un cercle, de rayon r , de centre C , se déplaçant sur le sol horizontal fixe en restant sur le même plan vertical. On appelle I le point de contact de la roue et du sol à l'instant t .

- le point I_s du sol qui est fixe dans (R).

1. Association Adilca. s.d. *Adhérence et Glissement de pneumatiques*. Document PDF sur INTERNET. <http://adilca.com/ADHERENCE_ET_GLISSEMENT_DES_PNEUMATIQUES.pdf>.

- le point I_R de la roue qui, lorsqu'elle roule, ne se trouve plus au contact du sol à un instant ultérieur.
- le point géométrique I qui localise le contact.

Nous supposerons que toutes les roues de la voiture ont même poids et mêmes dimensions. Ainsi, l'étude des forces et du mouvement est identique pour toutes les roues.

Considérons la statique du système²...

D'après la formule de Coulomb :

Soient deux solides S1 et S2 en contact ponctuel. On note \vec{v}_g la vitesse de glissement de S2 par rapport à S1.

- Si

$$\vec{v}_g \neq \vec{0} \text{ (il y a glissement)}$$

Alors

$$\begin{aligned} \vec{T} &\parallel \vec{v}_g; \\ \vec{T} \cdot \vec{v}_g &< 0; \\ \|\vec{T}\| &= f_0 \|\vec{N}\| \end{aligned} \tag{5.1}$$

où

f_0 est le coefficient de frottement statique de glissement

\vec{T} est la composante tangentielle au point de contact

\vec{N} est la composante normale au point de contact

- Si

$$\vec{v}_g = \vec{0} \text{ (il n'y a pas glissement)}$$

Alors

$$\|\vec{T}\| \leq f_0 \|\vec{N}\| \tag{5.2}$$

De plus,

- Il y a roulement si et seulement si

$$\|\vec{\Gamma}\| = k_o \|\vec{N}\|^3 \tag{5.3}$$

où

$\vec{\Gamma}$ est la torsion qui s'oppose au roulement

k_o est le coefficient de frottement statique de roulement

- Il n'y a pas roulement si et seulement si

$$\|\vec{\Gamma}\| \leq k_o \|\vec{N}\| \tag{5.4}$$

2. Auteur inconnu. s.d. Mécanique du solide. Document PDF sur INTERNET.
http://olivier.granier.free.fr/cariboo_files/PC-meca_solide.pdf.

3. DELCHAMBRE, Alain. 2010. Mécanique rationnelle I. Bruxelles : Presses Universitaires de Bruxelles a.s.b.l. 167p.

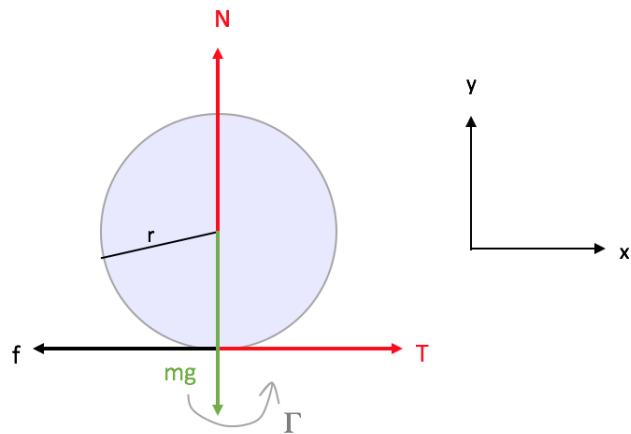


FIGURE 5.7 – Diagramme du corps libre de la roue en statique

Nous savons que

$$\|\vec{T}\| = ma^4 \quad (5.5)$$

$$\|\vec{N}\| = mg \quad (5.6)$$

$$\|\vec{\Gamma}\| = r\|\vec{T}\| = rma \quad (5.7)$$

où

m est la masse de la roue

g est la constante gravitationnelle

a est l'accélération

r est le rayon de la roue

Donc pour avoir roulement sans glissement :

$$\frac{k_0}{r} \leq \frac{a}{g} \leq f_0 \quad (5.8)$$

En conclusion, nous remarquerons qu'il sera nécessaire que la voiture ait une certaine accélération pour qu'elle puisse rouler sans glisser.

Si nous allons plus loin, il est possible de trouver un lien entre la tension fournie aux moteurs et la vitesse du véhicule. Pour cela, représentons la rotation plus en détails...

4. Deuxième loi de Newton stipule que l'accélération subie par un objet est proportionnelle à la force qu'il subit et inversement proportionnelle à sa masse.

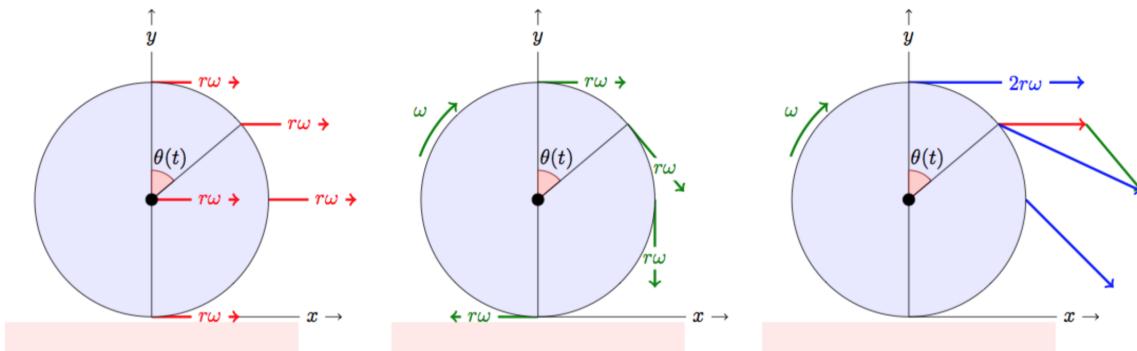


FIGURE 5.8 – Représentation des vitesses du roulement

Lorsque la roue effectue un tour, elle couvre une distance égale à sa circonférence pendant un temps égal à sa période T^5 . Le module de la vitesse du centre de la roue est donc :

$$v_c = \frac{2\pi r}{T} = r\omega \quad (5.9)$$

où

ω est la vitesse de rotation angulaire définie par $\omega = \frac{2\pi}{T}$

r est le rayon de la roue

Pour qu'il y ait roulement sans glissement, cette vitesse doit être égale à la vitesse tangentielle de la rotation. Comme le roulement est une combinaison d'une translation du centre et d'une rotation autour de ce centre, la vitesse d'un point quelconque de la roue est donc :

$$\vec{v}(t) = \vec{v}_c + \vec{v}_t(t)$$

où

\vec{v}_c est vitesse de la translation du centre

\vec{v}_t est la vitesse de la rotation autour du centre

Au point le plus haut de la roue, ces deux vecteurs sont de même sens si bien que $v = 2\pi r$.

Connaissant cela, nous pouvons introduire la notion de puissance qu'il faudrait mettre aux moteurs pour que la voiture roule. Pour cela, nous avons considéré un couple de moteur tel que :

$$C = r.F \quad (5.10)$$

où

C le couple moteur (en Nm)

r le rayon de la roue (en m)

F la force appliquée aux moteurs

5. Université catholique de Louvain (UCL). s.d. *Roulement*. Image sur INTERNET. <<https://perso.uclouvain.be/vincent.legat/documents/iepr1011/iepr1011-cours11.pdf>>.

Le couple correspond à la force angulaire d'un élément moteur, c'est-à-dire l'effort immédiat que cet élément moteur est capable de produire pour une vitesse donnée. Connaissant cela, nous pouvons calculer la puissance par la relation :

$$P = C \cdot \omega \quad (5.11)$$

où

P est la puissance (en Watt)

C le couple (en Nm)

ω la vitesse de rotation (en m/s)

Il est maintenant facile de retrouver la force motrice en connaissant la puissance et le couple moteur⁶. En effet, par la deuxième loi de Newton $F = m \cdot a$ et l'accélération intervient dans une équation fondamentale⁷ qui met en relation la distance, l'accélération, le temps, la vitesse initiale et la distance initiale telle que $d = \frac{1}{2}at^2 + V_0t = d_0$. Comme le point de départ est la voiture à l'arrêt, $V_0 = 0$ et $d_0 = 0$ et donc

$$a = \frac{2d}{t^2} \quad (5.12)$$

Il suffit de remplacer l'équation du couple par les variables trouvées et nous obtenons :

$$C = f \cdot r = m \cdot a \cdot r = m \cdot \frac{2d}{t^2} \cdot r \cdot P = C \cdot \omega = m \cdot \frac{2d}{t^2} \cdot r \cdot \omega \quad (5.13)$$

Nous retrouvons la vitesse par (5.9) et donc

$$P = \frac{md}{t^2} \cdot v \quad (5.14)$$

Empiriquement, nous remarquons que lorsque la puissance est nulle, la vitesse l'est aussi mais en réalité, il faut une grande puissance au démarrage appelée "pic de démarrage" dû au fait que cette dernière doit combattre toute l'inertie du système pour ensuite entretenir le mouvement. En fait, dans un moteur, la tension V est à l'image de la vitesse : plus la tension est grande, plus le moteur va vite. De plus, l'intensité I est à l'image du couple : plus l'intensité est grande, plus le moteur aura de couples mécaniques. Donc plus on élève les deux, plus le moteur est puissant par la relation $P = V \cdot I$.

Finalement, la relation entre l'intensité et le couple moteur, linéaire, permet d'introduire une constante de vitesse. Elle apparaît car le moteur possède une vitesse maximale qui tend à "ralentir" l'accélération, c'est-à-dire que plus on tourne vite, plus on s'approchera de la vitesse maximale. C'est cette constante qui explique le fait que sur la courbe de tension reprise en annexe, la voiture n'avancera qu'à partir d'un certain seuil de tension (dans notre cas 20%).

6. Auteur inconnu. s.d. *Couple, puissance, vitesse.* Site Web sur INTERNET. <<http://calibra-classic.org/pages/powerNTorque.htm>>.

7. Robot Maker. Mise à jour : 23/01/2015. *Choisir et simuler un moteur pour votre robot.* Site web sur INTERNET. <<http://www.robot-maker.com/forum/tutorials/article/50-choisir-et-simuler-un-moteur-pour-votre-robot/>>

Chapitre 6

Fonctionnement du groupe

6.1 Organisation du groupe

Au commencement, le groupe avait peu de connaissances requises pour apporter des progrès concrets au projet. Les premières répartitions de tâches se sont donc faites en fonction des notions que chacun possédait. Ensuite, pour que les membres puissent se familiariser avec toutes les tâches, un système de pair programming¹ a été instauré. Chaque sous-groupe rend alors un rapport hebdomadaire sur l'avancement de leur travail, avant la prochaine réunion. Lors de ces réunions, tous les sous-groupes doivent être capable d'expliquer le travail effectué pour chacune de ces tâches. De là, les membres envisagent des solutions aux éventuels problèmes rencontrés, discutent et décident des prochaines étapes à réaliser.

Un ordre du jour, concernant la réunion à venir, est rédigé par le chef d'équipe et lu par l'ensemble du groupe. Ceci permet d'assurer un certain professionnalisme lors des réunions et de ne pas perdre de vue les objectifs en cours. Un secrétaire est également désigné avant chaque réunion, celui-ci étant chargé de prendre des notes concernant les points discutés lors de ces réunions et de les partager aux autres membres le plus rapidement possible.

Durant le premier quadrimestre, le groupe se rencontrait deux fois par semaine. Le lundi, pour les réunions assistées par le chef d'équipe, et le vendredi étant une séance de travail durant laquelle la présence du chef d'équipe n'est pas nécessaire.

Une fois le prototype construit et muni de toutes ses composantes, il n'était plus indispensable d'organiser des séances de travail puisque les tâches restantes ne consistaient qu'en la rédaction du rapport, et en le codage. Le groupe se rejoignait, dès lors, tous les jeudis, en concordance avec les horaires de chacun.

Une charte a été établi entre le chef de projet et le groupe afin de définir quelles raisons pouvaient engendrer une sanction ludique. Cette dernière est donnée afin d'éviter les comportements parasites pour le bon déroulement du projet tels que les retards, le non-respect d'autrui, le manque d'attention lors des réunions, l'incapacité d'expliquer l'avancement du projet,

1. méthode de travail dans laquelle deux développeurs travaillent ensemble sur un même poste de travail. Le premier rédige le code et le deuxième est observateur.

Wikipédia. Mise à jour :03/03/2018. *Programmation en binôme*. Site Web sur INTERNET. <https://fr.wikipedia.org/wiki/Programmation_en_binôme>.

etc. Cette sanction, définie par l'ensemble du groupe, consiste en le paiement d'une tournée de bières. Cette méthode de pénalité s'est d'ailleurs avérée efficace, que ce soit pour l'amélioration du comportement en général ou de la bonne entente entre les différents membres.

6.2 Canaux de communication

Les canaux de communication favorisés sont un groupe Facebook et le Google Drive. Les communications de dernières minutes ou les petits débats concernant certaines décisions à prendre en dehors des réunions sont publiés sur le groupe Facebook. Tandis que tous les documents partagés tels que les ordres du jour, les PV, les rapports hebdomadaires, et autres sont mis sur le Drive.

Ensuite, le groupe a commencé à utiliser Discord, une application de discussion en ligne multi-plateforme. Celle-ci permet à tous les membres de travailler sur leur tâche, ou débriefer sur l'avancement du projet sans pour autant devoir se rencontrer.

6.3 SWOT

L'analyse **SWOT**, acronyme pour Strengths - Weaknesses - Opportunities - Threats, est un outil de stratégie d'entreprise. Concrètement, les forces et faiblesses correspondent à l'environnement interne du groupe, c'est-à-dire à son cadre de travail, son organisation, son avancement, etc. Tandis que les opportunités et les menaces se rapportent à son environnement extérieur, donc les facteurs sur lesquels le groupe ne peut intervenir et qui influencent son travail.

Cet outil d'évaluation permet d'obtenir un diagnostic pertinent sur le fonctionnement du groupe, et ainsi de l'améliorer.

Un premier SWOT est réalisé le 23 octobre.

Forces	Faiblesses
- Propositions de plusieurs idées.	- Beaucoup de discussions sans progrès concrets ; - Manque de communication.
Opportunités	Menaces
- Possibilités d'utilisation de l'imprimante 3D de la faculté d'Architecture.	- Réception d'un mauvais châssis ou d'un châssis endommagé commandé en ligne ; - Problèmes d'installation ou de fonctionnement de la machine virtuelle.

Au début du projet, les membres du groupe ne se connaissaient que très peu, ce qui renvoyait une collaboration plutôt floue, sans grande productivité. Au fur et à mesure des réunions, le groupe apprend à se connaître et finit par trouver un bon équilibre tenant compte des faiblesses et des forces de chacun, apportant un meilleur travail d'équipe.

Des idées concrètes sont finalement apportées grâce à une bonne visualisation des tâches à effectuer à la suite des séances de travail en groupe. Quelques progrès sont réalisés, ce qui motive l'ensemble du groupe et renforce la bonne entente.

Les plaques à ajouter au châssis pouvaient être obtenues via imprimante 3D, malheureusement cela n'a pas pu se faire par manque de temps. L'équipe a donc du trouver un autre moyen d'acquérir celles-ci.

Il y avait certaines craintes concernant la bonne réception du châssis du prototype mais aucun problème n'est finalement survenu. Des membres du groupe ont rencontré des difficultés pour l'installation de la machine virtuelle, la plateforme de simulation des programmes. En effet, certains avaient un ordinateur incompatible avec cette dernière, d'autres avaient le leur qui beugait. Mais au final, les sous-groupes étaient organisés pour que chacun ait accès à au moins ordinateur portable fonctionnel avec la machine virtuelle.

Un second SWOT est effectué le 04 décembre.

Forces	Faiblesses
<ul style="list-style-type: none"> - Groupe motivé et progrès concrets apportés en fin de sprint ; - Bonne ambiance = envie de se réunir ; - Bonne répartition du travail. 	<ul style="list-style-type: none"> - Bonne ambiance néfaste = réunions moins efficaces ; - Manque de connaissances pour certaines thématiques du projet.
Opportunités	Menaces
<ul style="list-style-type: none"> - Plaques en bois contreplaqué offertes par le vendeur ; - Conservateurs de collections du Cercle Polytechnique ; - Entre-aide entre les groupes. 	<ul style="list-style-type: none"> - Problème de câblage = perte de tension et/ou cours-circuits

La bonne ambiance acquise au sein du groupe s'avère être à double tranchant, puisque le groupe finit distract lors des réunions et s'écarte du sujet supposé être abordé ainsi que des objectifs en cours. Les sanctions sont alors plus strictes afin d'être le plus efficaces possible durant les réunions.

Certains sous-groupes ont eu du mal à avancer sur leurs tâches à cause du manque de connaissances requises, pour le codage des programme par exemple, mais ont fini, grâce à l'aide des autres groupes de projet, par apporter de gros progrès.

Un autre moyen d'obtenir les plaques à ajouter au châssis devait être trouvé. Qu'elles soient en bois contreplaqué paraissait idéal et puisque le vendeur proposait de les offrir, cette idée s'est finalement imposée. Pour qu'elles puissent être montées et maintenues correctement, des trous ont été faits avec le matériel disponible chez les Conservateurs de collections du Cercle Polytechnique de l'Université Libre de Bruxelles, ne disposant pas du matériel nécessaire.

La câblage du breadboard s'est fait à l'aide de simples fils de cuivre, ce qui est assez fragile et peut causer des pertes de tension voire des court-circuits. Il a donc fallu remplacer ces fils par des câbles de pontage afin d'optimiser le câblage.

Un dernier SWOT s'est fait le 13 mars.

Forces	Faiblesses
- Bonne cohésion de l'équipe ; - Meilleure autonomie du groupe.	- Répartition et communication des tâches.
Opportunités	Menaces
- Aire de test ; - Entre-aide des groupes.	- Communication <i>Arduino Nano-Orange Pi</i> ; - Connection WIFI lente.

Arrivé à la fin du projet, le groupe a acquis de bonnes habitudes et est devenu plus autonome, les réunions sont donc plus productives.

Certains membres ont fini par se spécialiser dans certaines tâches, en particulier dans le codage, ce qui fait que d'autres n'ont presque pas ou plus travailler sur celles-ci, ils avaient donc des lacunes dans ces domaines. Le groupe lisait donc les rapports hebdomadaires réalisés, histoire de rester à jour au niveau des progrès effectués et une séance de travail a été organisé avant les évaluations afin de prendre le temps d'expliquer précisément les différentes thématiques du projet.

Un problème est survenu au niveau de la communication *Arduino Nano-Orange Pi*. Bien que tout se déroulait parfaitement en simulation, l'*Arduino* n'interprétait plus correctement les données reçues une fois dans la réalité. Grâce à l'aide des autres groupes, le problème a été réglé et les manoeuvres expérimentées aux aires de tests.

Certaines manoeuvres ont été codées mais n'ont pu être testées à cause d'une mauvaise connectivité WIFI. De plus, il n'était désormais plus possible de réaliser les autres manoeuvres déjà fonctionnelles sans cette connection WIFI. Heureusement, un nouveau câble assurant cette connection, et reliant la batterie à l'unité centrale, a été prêté à l'équipe afin de pouvoir présenter les manoeuvres disponibles lors de l'évaluation du prototype.

Désormais, l'équipe travaille de manière harmonieuse et apporte des progrès concrets tout en profitant des atouts qui s'offrent à elle. Les seuls obstacles rencontrés sont dus à des problèmes techniques mais ceux-ci finissent toujours par être résolus de manière efficace.

Chapitre 7

Conclusion

En conclusion, le prototype « Bacar Zen » est suffisant pour commencer la guerre concurrentielle dans le monde des voitures autonomes. La voiture construite dispose de quatre roues motrices et d'un châssis résistant et sûr permettant d'encaisser les chocs et d'assurer la sécurité d'éventuels passagers. Les différents programmes nécessaires à l'autonomie du véhicule ont été codés avec succès. Plus précisément, le Sign detector permet de détecter et différencier les différents panneaux, en estimant la couleur dominante du panneau, et donc de savoir quelle instruction est donnée par le panneau. Ensuite, le Path detector garde la voiture sur la route en vérifiant à chaque instant que la voiture suive bien la route en ligne droite. Si ce n'est pas le cas, la voiture est redirigée à l'aide d'un heading dépendant de l'angle entre la droite verticale et la direction que prend la voiture.

Seulement ces deux programmes seuls sont insuffisants, il faut un troisième programme couplant les deux afin que la voiture sache comment suivre la route tout en suivant les panneaux. Ce programme est la State Machine. Cette dernière prend les informations données par chaque code et les organise. Ainsi, la State Machine effectuera d'abord les ordres donnés par les panneaux car le Path Detector pourrait empêcher la voiture de tourner dans la direction désirée. Donc la State Machine permet de mettre un ordre de priorité entre les différentes informations. En plus de cela, l'envoi de commande par l'utilisateur à la voiture passe par la State Machine car cette dernière constitue le lien principal entre l'Arduino, qui contrôle les moteurs, et les codes Python de l'Orange Pi. Enfin, c'est grâce à l'Arduino et au code C implémenté dans ce dernier que l'on peut concrètement effectuer chaque manœuvre en commandant directement les moteurs de droite et de gauche.

Les simulations furent un succès. Le véhicule suit la route et obéit à chaque panneau qu'il voit. Le passage à la réalité fut plus compliqué que prévu car, pour supporter la réalité, il faut que le code soit souple et qu'il permette d'ignorer les nombreuses interférences, que ce soit pour la caméra ou pour les détecteurs (par exemple les différences de luminosité, les reflets, ...). De plus, l'équipe fut ralentie par un problème de câblage. Néanmoins, le véhicule, à l'heure actuelle, correspond aux exigences de départ. En définitif, la voiture a toujours du mal à évoluer dans la réalité mais l'équipe va rapidement optimiser les codes et plus particulièrement la State Machine afin de pouvoir se lancer définitivement sur des prototypes plus imposants de voitures. Avec les différents prototypes qui sont à venir, il est certain que l'entreprise BACAR se lancera dans très peu de temps sur le marché et a toutes les chances de dépasser chacun de

ses concurrents !

Chapitre 8

Annexes

Cahier des comptes

Dates :	Descriptif :	Dépenses :	Rentrées :	Solde :
	Budget initial accordé.	/	/	100,00+
25/10/2017	Kit assemblage, batterie externe.	27,98-	/	72,02+
14/11/2017	Vis, écrous, équerres.	2,90-	/	69,12+
21/11/2017	Têtes pins mâle.	1,50-	/	67,62+
25/11/2017	Vis, écrous.	2,30-	/	65,32+
04/12/2017	Adhésif double-face.	2,55-	/	62,77+
04/12/2017	Vis, écrous.	1,45-	/	61,32+
11/12/2017	Câbles pontage mâle/femelle.	5,50-	/	55,82+
12/12/2017	Colsons, glue.	5,40-	/	50,42+
19/02/2018	Câbles pontage mâle/mâle.	5,10-	/	45,32+
12/03/2018	Vis, écrous.	4,15-	/	41,17+
13/03/2018	Piles.	6,50-	/	34,67+

FIGURE 8.1 – Cahier des comptes

Courbe de caractérisation de la vitesse en fonction de la tension

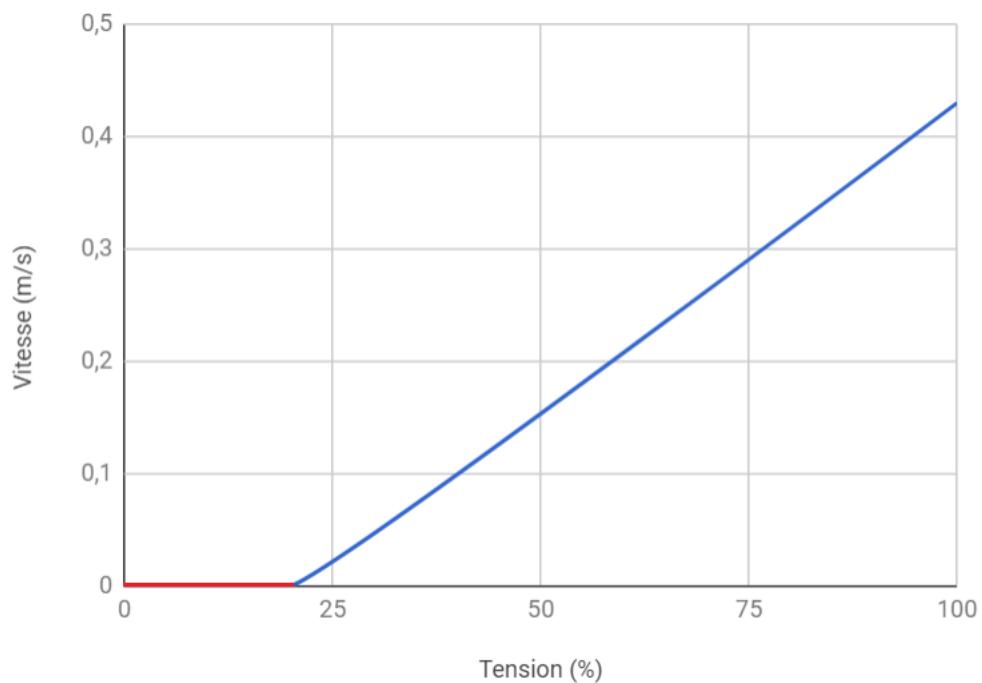


FIGURE 8.2 – Courbe de caractérisation tension/vitesse

Contraintes

Catégorie S (sécurité et homologation)		
S.1.	Isolation	Le dispositif mettant en œuvre des circuits électroniques fragiles, un soin tout particulier sera accordé à l'isolation des différentes sources d'alimentation électrique. De même, il faut veiller à éviter les contacts entre les cartes électroniques et les éléments conducteurs (métal, utilisateur,...).
S.2	Alimentation	Deux alimentations distinctes doivent être prévues pour l'unité centrale d'une part et pour les moteurs d'autre part. Un contrôle du montage sera effectué lors des permanences techniques prévues lors des semaines 8 et 9, avant la mise sous tension des circuits. L'alimentation des moteurs doit être munie d'un interrupteur (non fourni) permettant de facilement couper l'alimentation de ceux-ci, indépendamment de l'alimentation de l'unité centrale.
S.3	Batteries	Les batteries sont des éléments pouvant présenter un danger si elles sont mal utilisées. Leur branchement devra se faire à l'aide de connecteurs adéquats et leur recharge ne pourra se faire qu'à l'aide de chargeurs prévus à cet effet (éviter les montages "maison").
S.4	Dimensions	Largeur maximum ³ : 21 cm Longueur maximum ⁴ : 30 cm Un dépassement raisonnable(<3cm) peut être toléré pour les câbles. ⁵ Hauteur maximum: 25 cm
S.5	Masse	Le véhicule, batteries comprises ne doit pas excéder 2kg Une charge utile supplémentaire de 0 à 200g doit pouvoir être supportée par le prototype lors de la phase de test.
S.6	Robustesse	Tous les éléments composant le prototype doivent être solidaires. Le prototype doit pouvoir être manipulé sans risque pour son intégrité structurelle.
S.7	Ergonomie	Le prototype ne devra pas présenter de pièces tranchantes/salissantes pouvant causer des dommages aux personnes ou aux autres véhicules ainsi qu'à l'aire de test.
S.8	Environnement WIFI	L'unité centrale étant munie d'une connectivité WIFI, il est strictement interdit de tenter de perturber le bon fonctionnement de l'environnement WIFI et/ou du véhicule des autres groupes.
S.9	Respect matériel du	De manière générale, le matériel mis à disposition sera utilisé "en bon père de famille". En cas de casse, l'équipe sera solidairement responsable du remplacement des éléments endommagés.

³ La largeur des routes du circuit de validation est égale à la largeur d'une feuille A4

⁴ Attention certaines manœuvres peuvent être problématiques avec un véhicule trop long

⁵ À condition que ce dépassement ne provoque pas de contact avec les éléments présents sur le circuit (par ex. la signalisation)

FIGURE 8.3 – Contraintes de sécurité

Path Detector

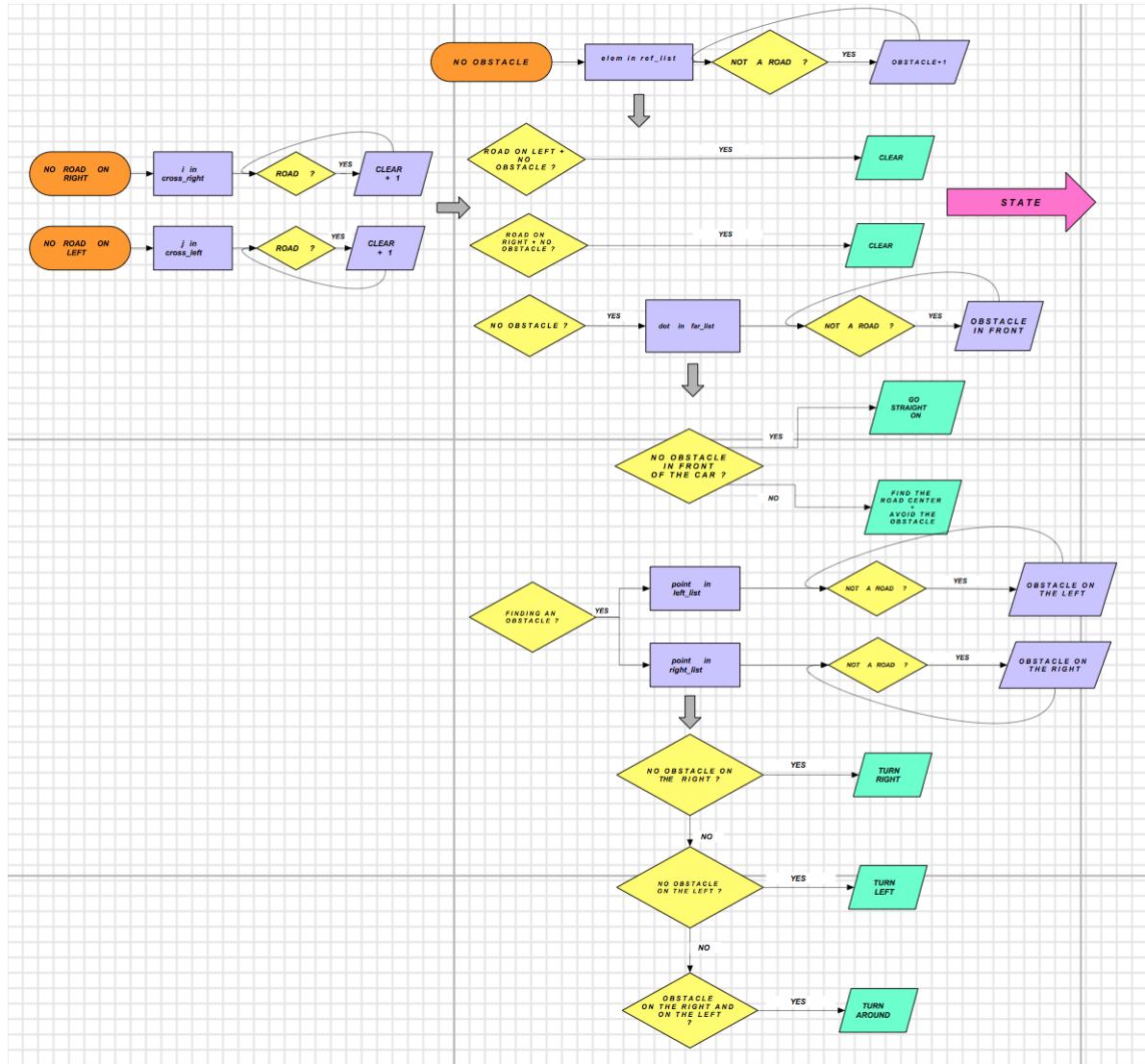


FIGURE 8.4 – Organigramme du code du Path Detector

Code en langage C

The image displays three separate windows of the Arduino IDE, each showing a different C program for controlling two motors (motorA and motorB) and a LED. The programs are as follows:

- avance_5_secondes_dans_le_setup**: This sketch moves the motors forward for 5 seconds. It includes setup and loop sections with pinMode, begin, actuate, and delay functions.
- avance_3_secondes_puis_pause_2_secondes**: This sketch moves the motors forward for 3 seconds, then pauses for 2 seconds. It includes setup and loop sections with similar logic.
- avance_sur_un_m_tre_dans_le_setup**: This sketch moves the motors forward until a button is pressed. It includes setup and loop sections with a while loop and a digitalRead function.

Each window shows the code, the 'Done compiling.' message at the bottom, and the memory usage information at the very bottom.

FIGURE 8.5 – Exemple de code en langage C

sign detector

```
import logging
import numpy as np
import cv2

logging.info('Template SignDetector has been initialized')

def detect(bb, sign):
    """This function receives:
    - sign: a color image (numpy array of shape (h,w,3))
    - bb which is the bounding box of the sign in the original camera frame
      bb = (x0,y0, w, h) where w and h are the width and height of the sign
      (can be used to determine e.g., whether the sign is to the left or
       right of the car's center)
    The goal of this method is to recognize which of the following signs it
    really is:
    - a stop sign
    - a turn left sign
    - a turn right sign
    - None, if the sign is determined to be none of the above

    Returns: a dictionary that contains information about the recognized
    sign. This dict is transmitted to the state machine it should contain
    all the information that the state machine to act upon the sign (e.g.,
    the type of sign, estimated distance).
    """
    (x0, y0, w, h) = bb
    b,g,r = cv2.split(sign)
    t = 0
    matrice_a_gauche, matrice_a_droite = np.array_split(b, 2, 1)
    somme_g = np.sum(matrice_a_gauche)
    somme_d = np.sum(matrice_a_droite)
    if w/h > 0.5 and w/h < 1.4:

        if len(r[r > 80]) > w*h//3 and len(b[b < 60]) > w*h//3 :
            t = "Stop"
        elif len(b[b > 81]) > w*h//3:
            if somme_g > somme_d + 100 :
                t = "Left"
            else :
                t = "Right"
    sign_dict = {'sign' : t, 'x0' : x0, 'y0' : y0, 'w' : w, 'h' : h}
    return sign_dict
```

FIGURE 8.6 – Exemple de code du "sign detector"

Bibliographie

Documents électroniques

Auteur inconnu. s.d. *Couple, puissance, vitesse*. Site Web sur INTERNET. <<http://calibra-classic.org/pages/powerNTorque.htm>>. Dernière consultation : 14/03/2018.

Auteur inconnu. s.d. *Mécanique du solide*. Document PDF sur INTERNET. <http://olivier.granier.free.fr/cariboost_files/PC-meca_solide.pdf>. Dernière consultation : 01/03/2018.

Auteur inconnu. s.d. *Rotation d'un corps rigide*. Document PDF sur INTERNET. <http://www.gfbienne.ch/physique/OS/OSm_Chap9.pdf>. Dernière consultation : 03/03/2018.

Association Adilca. s.d. *Adhérence et Glissement de pneumatiques*. Document PDF sur INTERNET. <http://adilca.com/ADHERENCE_ET_GLISSEMENT_DES_PNEUMATIQUES.pdf>. Dernière consultation : 28/02/2018.

Bureau d'Appui Pédagogique en Polytechnique (BAPP). 2017. *Le pari Bacar*. Document PDF sur INTERNET. <https://uv.ulb.ac.be/pluginfile.php/1036832/mod_resource/content/1/Guide_BACAR_2017.pdf>. Dernière consultation : 21/02/2018.

Bureau d'Appui Pédagogique en Polytechnique (BAPP). Octobre 2017. «Exemples de manipulation d'images» dans *Petit tutoriel Numpy*. Document sur INTERNET. <https://github.com/odebeir/numpy_tutorialBA1>. Dernière consultation : 21/02/2018.

Bureau d'Appui Pédagogique en Polytechnique (BAPP). Octobre 2017. *Lancer une simulation*. Vidéo sur INTERNET. <https://uv.ulb.ac.be/pluginfile.php/1041638/mod_resource/content/1/2-lancer%20une%20simulation.mp4>. Dernière consultation : 21/02/2018.

Bureau d'Appui Pédagogique en Polytechnique (BAPP). Octobre 2017. *Lancer une vidéo simulation*. Vidéo sur INTERNET. <https://uv.ulb.ac.be/pluginfile.php/1041639/mod_resource/content/1/3-lancer%20une%20video%20simulation.mp4>. Dernière consultation : 21/02/2018.

Instructables. Mise à jour : 25/10/2016. *Smartphone controlled Arduino Rover*. Site web sur INTERNET. <<http://www.instructables.com/id/Smartphone-Controlled-Arduino-Rover/>>. Dernière consultation : 14/03/2018.

NEBRA, Mathieu. Mise à jour : 23/11/2017. *Apprenez à programmer en C!* Site web sur INTERNET. <<https://openclassrooms.com/courses/apprenez-a-programmer-en-c>>. Dernière consultation : 21/02/2018.

Robot Maker. Mise à jour : 23/01/2015. *Choisir et simuler un moteur pour votre robot*. Site web sur INTERNET. <<http://www.robot-maker.com/forum/tutorials/article/50-choisir-et-simuler-un-moteur-pour-votre-robot/>>. Dernière consultation : 14/03/2018.

Université catholique de Louvain (UCL). s.d. *Roulement*. Document PDF sur INTERNET. <<https://perso.uclouvain.be/vincent.legat/documents/iepr1011/iepr1011-cours11.pdf>>. Dernière consultation : 03/03/2018.

Wikipédia. Mise à jour : 03/03/2018. *Programmation en binôme*. Site Web sur INTERNET. <https://fr.wikipedia.org/wiki/Programmation_en_binôme>. Dernière consultation : 15/03/2018.

Documents classiques

DELCHAMBRE, Alain. 2010. *Mécanique rationnelle I*. Bruxelles : Presses Universitaires de Bruxelles a.s.b.l. 167 p.

GIAMARCHI, Frédéric, Laurent FLORES. 2007. *Construisons nos robots mobiles*. Paris : ETSF. 160 p.

MASSART, Thierry. 2016. *Syllabus INFO-H-100 Informatique*. Bruxelles : Presses Universitaires de Bruxelles a.s.b.l. 217 p.

TAVERNIER, Christian. 2005. *Programmation en C des PIC*. Collection Technique et ingénierie - EEA. s.l : Edition Dunod, L'usine Nouvelle. 215 p.

Images

Auteur inconnu. s.d. *Mécanique du solide*. Image sur INTERNET. <http://olivier.granier.free.fr/cariboo_files/PC-meca_solid.pdf>.

Bureau d'Appui Pédagogique en Polytechnique (BAPP). Octobre 2017. *Description des éléments matériels de la BACAR Zero*. Image sur PDF sur INTERNET.
[<https://uv.ulb.ac.be/pluginfile.php/1041623/mod_resource/content/1/Annexe%20-%20Hardware.pdf>](https://uv.ulb.ac.be/pluginfile.php/1041623/mod_resource/content/1/Annexe%20-%20Hardware.pdf).

Bureau d'Appui Pédagogique en Polytechnique (BAPP). 2017. *Le pari Bacar*. Image sur INTERNET.
[<https://uv.ulb.ac.be/pluginfile.php/1036832/mod_resource/content/1/Guide_BACAR_2017.pdf>](https://uv.ulb.ac.be/pluginfile.php/1036832/mod_resource/content/1/Guide_BACAR_2017.pdf).

The MathWorks. 2017. *colormap matrix*. Image sur INTERNET. [<https://nl.mathworks.com/help/matlab/creating/_image-types.html?requestedDomain=www.mathworks.com&requestedDomain=true>](https://nl.mathworks.com/help/matlab/creating/_image-types.html?requestedDomain=www.mathworks.com&requestedDomain=true).

Université catholique de Louvain (UCL). s.d. *Roulement*. Image sur INTERNET.
[<https://perso.uclouvain.be/vincent.legat/documents/iepr1011/iepr1011-cours11.pdf>](https://perso.uclouvain.be/vincent.legat/documents/iepr1011/iepr1011-cours11.pdf).