NXP CUP
INTELLIGENT
CAR RACING

2ème année Informatique

Rapport du Projet au Fil de l'Année
NXP Cup

Ouedraogo Sonia
Bahrami Tara
Leguay-Lenormand Carla
Jacobi Otavio
Laaroussi Abdelouahab
Despretz Valérian
Venot Alexandre

Encadrant :
M. Rollet Antoine
Client :
M. Barthou Denis

Table des matières

1 Introduction

Dans le cadre d'un projet de deuxième année nous avions à implémenter une voiture autonome pour participer à la NXP Cup.

L'équipe actuelle est composée de :
- Alexandre Venot
- Ouedraogo Sonia
- Bahrami Tara
- Leguay-Lenormand Carla
- Jacobi Otavio
- Laaroussi Abdelouahab
- Despretz Valérian

Cette compétition a pour but de concevoir une voiture qui puisse suivre une piste inconnue de manière autonome le plus rapidement possible. Elle est également composée d'épreuves optionnelles :
- Le parcours en 8
- L'évitement d'obstacle
- La zone de vitesse limite
- L'arrêt d'urgence

Ces quatre épreuves seront précisées plus tard. Les modifications de la voiture nécessitent l'application d'outils provenant de différents domaines dont l'informatique, l'électronique et la modélisation 3D.

Ce projet a également pour objectif de nous mettre en situation réelle de travail. Nous avions en effet un client Monsieur Denis Barthou, professeur à l'ENSEIRB-MATMECA, pour lequel nous produisions cette voiture autonome et nous étions aussi supervisés par un responsable pédagogique Monsieur Antoine Rollet, également professeur à l'ENSEIRB-MATMECA. Ce projet s'est déroulé sur six mois avec une première partie de génie logiciel consacrée à la spécification des besoins et une deuxième partie pour le développement.

Dans ce rapport nous présentons les différentes démarches pour la résolution du problème mais également notre analyse et nos réflexions sur le travail réalisé. Ainsi, nous pourrons comparer ce qui était prévu dans le cahier des charges et le produit final.

Ce rapport est composé de trois grandes parties :

— La première décrivant nos démarches afin de participer à la NXPCup.
— La deuxième permettant d'adapter le projet pour être réalisable à distance dû à la situation sanitaire de la première moitié de 2020.
— La dernière expliquant nos problèmes rencontrés et nos pistes d'amélioration.

Nous remercions nos deux professeurs cités ci-dessus pour leur disponibilité et soutien durant tout le projet.

2 Première version : Version Physique

2.1 Analyse du problème et organisation du travail

2.1.1 Analyse du problème

Le but de cette première version est de réaliser une voiture pouvant suivre un circuit non prédéfini de manière autonome, et d'effectuer différentes épreuves annexes telles qu'un évitement d'obstacle par exemple, et ce avant la date limite du 2 avril 2020 qui correspond aux premières qualifications pour la NXPCup.
Les domaines d'ingénierie impliqués étaient aussi bien informatiques que électroniques, car chaque équipe participant à la compétition était libre (sous réserve de quelques contraintes) quant à la manière dont il conçoit son véhicule autonome.
Les deux grandes tâches du problème sont la création d'un ou plusieurs programmes à téléverser sur des cartes électroniques embarquées, et la conception du modèle physique de notre voiture. L'ensemble du problème est décrit de façon détaillée dans le Document de Spécifications joint en annexe.

2.1.2 Organisation des tâches

Après avoir consacré trois mois à la spécification des besoins, à l'aide du chef de projet qui était d'abord Théo Matricon puis Alexandre Venot mais également de notre encadrant pédagogique Antoine Rollet, nous étions en capacité de diviser les tâches de la manière suivante :

- Valérian Depretz s'est chargé en particulier des problèmes techniques comme l'électronique et la conception de la voiture mais également de la documentation sur le SLAM.
- Sonia Ouedraogo et Tara Bahrami se sont occupées de l'épreuve optionnelle sur la zone de vitesse limitée par des passages pietons mais également sur la détection de la ligne d'arrivée.
- Abdelouahab Laaroussi, Alexandre Venot et Otavio Jacobi ont travaillé sur le code de l'épreuve principale
- Enfin, Carla Leguay-Lenormand s'est intéressée au LIDAR plus particulièrement à l'épreuve optionnelle de l'arrêt d'urgence.

Un diagramme de Gantt avec les tâches prévisionnelles est présent dans le document de spécification. Cependant, durant l'avancement du projet, beaucoup de problèmes techniques nous ont empêchés d'avancer aussi rapidement qu'on l'aurait voulu. Ainsi, nous avons abandonné l'idée du SLAM, tâche présente dans la phase de spécification.

La figure 1 est un diagramme de Gantt qui présente plus précisément les tâches réalisées par chacun pendant la première phase du projet.

Auteur	Tâche	Début	Fin
Théo & Abdelouahab	Module vitesse 1/10	22/11/2019	06/12/2019
Alexandre	Freinage 1/10	06/12/2019	13/12/2019
Otavio	Module Caméra	22/11/2019	06/12/2019
Carla & Sonia	Module Lidar 1/10	22/11/2019	13/12/2019
Tara & Valérian	Calcul Trajectoire 1/10	13/12/2019	10/01/2020
Abdelouahab	Module Vitesse 1/18	17/01/2019	24/01/2020
Alexandre & Otavio	Freinage 1/18	24/01/2020	14/02/2020
Valérian & Carla	Module Lidar 1/18	24/01/2020	14/02/2020
Otavio & Alexandre	Calcul Trajectoire 1/18	28/02/2020	06/03/2020
Valérian	Création Plateforme Caméra 1/18	14/02/2020	21/02/2020
Sonia & Tara	Passage Piéton	14/02/2020	28/02/2020
Sonia & Tara	Ligne d'arrivée	28/02/2020	06/03/2020
Carla	Arrêt d'urgence	14/02/2020	06/03/2020
Sonia & Carla	Evitement d'obstacle	14/02/2020	Non fini

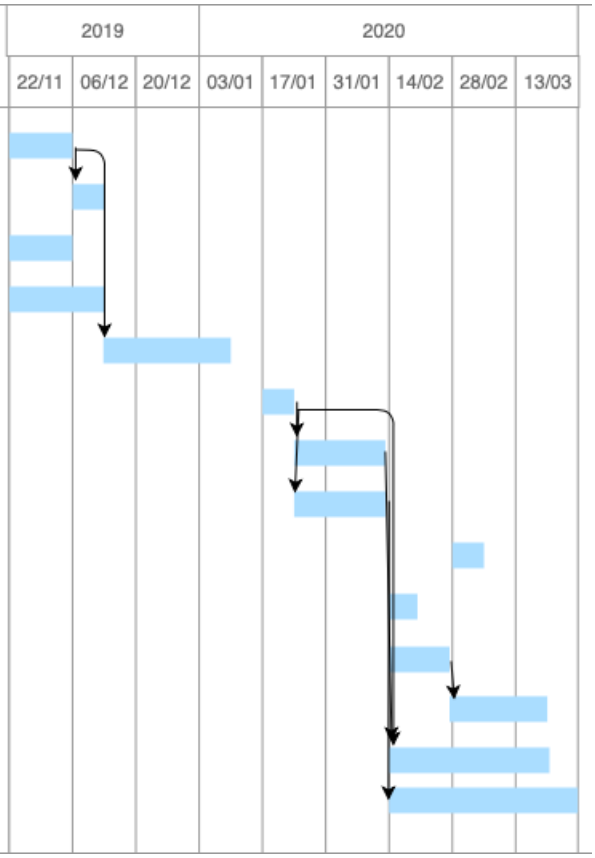

FIGURE 1 – Diagramme de Gantt de la première version

2.2 Résolution

2.2.1 Construction du modèle physique

La construction du modèle physique nous a pris beaucoup de temps. Nous avons commencé par utiliser un modèle 1/10 (peu adapté pour la piste prévue par les réglementations de la NXPCup donc inutilisable pour la compétition), puis parce que nous n'avons pas eu les modèles définitifs avant janvier 2020.
Pour le modèle 1/10, que nous savions comme n'étant pas notre modèle définitif, nous avons placé la caméra sur une tige à l'aide de ruban adhésif, tige fournie dans le kit de construction d'un autre modèle de voitures, les 1/16 (nous expliquerons plus tard pourquoi nous ne les avons pas utilisées). Nous avons fixé le tout au châssis de la voiture à l'aide de ruban adhésif et de fils.

Cette voiture nous a servi jusqu'à l'interview avec France 3, et à partir de ce moment là nous avons consacré à nouveau beaucoup de temps à essayer de retrouver le comportement que nous avions sur la 1/10 mais cette fois sur la 1/18. Il a fallu d'abord reconstruire un modèle physique entier. Cependant, la tige de métal semblait un peu lourde pour ce modèle plus petit, d'autant plus que les modèles 1/18 possédaient des amortisseurs peu rigides, ce qui avait comme effet de rendre la voiture assez instable en raison du centre de gravité assez haut. Cette fois la caméra est en revanche fixée sur la tige à l'aide de bois et de colle à bois, même si l'ensemble n'est pas toujours très stable. La base de la tige est également collée au châssis de la voiture.

Au niveau des connexions électriques, nous avons utilisé des breadboards au début, et simplement connecté les fils. Avec le temps, nous avons constaté que les fils avaient tendance à se détacher de la breadboard et nous avons eu l'idée de les souder afin de palier ce problème. Mais nous avons fini par griller la carte Teensy4 sur laquelle les soudures ont été effectuées. Cela finalement nous a ramené à l'utilisation des breadboards jusqu'au dernier moment avant la compétition.

2.2.2 Parcours sur circuit inconnu

Le programme permettant à notre voiture de réaliser le parcours sur circuit inconnu, qui est l'épreuve centrale de la compétition, a été pensé de la manière suivante : la voiture adopte un nombre fini de comportements différents, en fonction de la situation dans laquelle elle se trouve. En pratique, cela donne un switch sur une valeur que l'on appelle un état. Dans la version fonctionnelle la plus avancée, on trouve trois états différents : la ligne droite, le virage et le pré-virage. Ces trois états sont détectés en fonction de patterns aux paramètres précis sur les vecteurs que renvoie notre caméra Pixy2.
Ce qui suit sert à décrire chacun des trois états. On rappelle que la caméra renvoie de manière périodique les coordonnées d'un nombre fini de vecteurs dans l'espace.

Ligne droite
La ligne droite est l'état par défaut, elle n'est donc pas détectée à partir de patterns spéciaux. En revanche, lorsque l'on se trouve en ligne droite, pour parvenir à rester sur la piste, la voiture cherche à rendre égale la norme des deux plus grands vecteurs. Comme le montrent les schémas sur la figure 2, lorsque les deux vecteurs sont de taille égale, alors la voiture peut mettre ses roues droites. Mais lorsque qu'un vecteur est plus petit que l'autre, cela signifie que la voiture est en train de sortir de la piste.

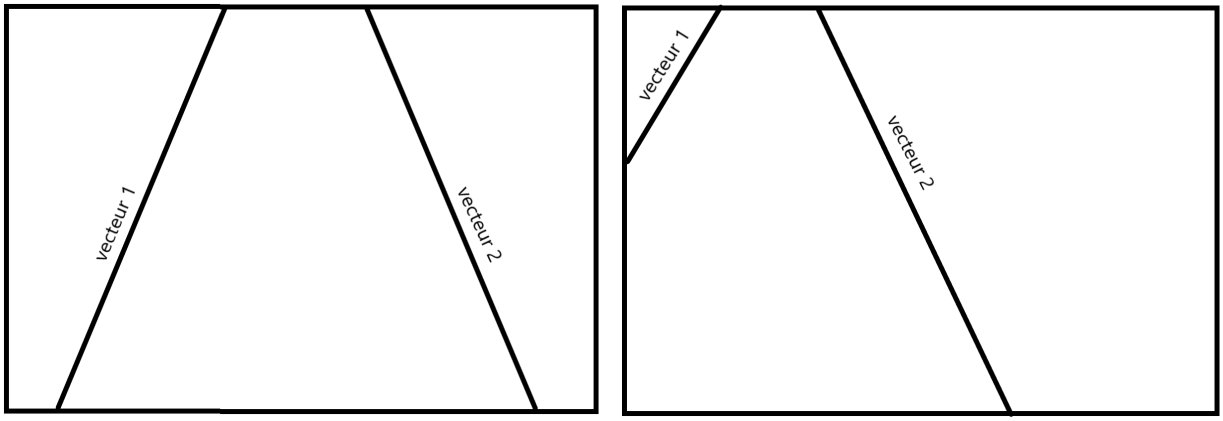

FIGURE 2 – Illustration de ce qu'observe la caméra sur une ligne droite

Nous avons donc mis en place un système asservi (en boucle fermée) pour maintenir la voiture au centre de la route. Dans le cas de l'image de droite au-dessus, il faut naturellement tourner les roues un peu à gauche.

Virage

Les virages sont détectés lorsque le plus grand des vecteurs est suffisamment horizontal, et lorsque son extrémité la plus haute, est suffisamment basse par comparaison à un seuil.
Quand la voiture se trouve dans cette position, c'est qu'elle est presque en train de sortir du virage, l'angle de braquage maximal est donc appliqué aux roues. Le schéma de la figure 3 représente le cas d'un virage à gauche.

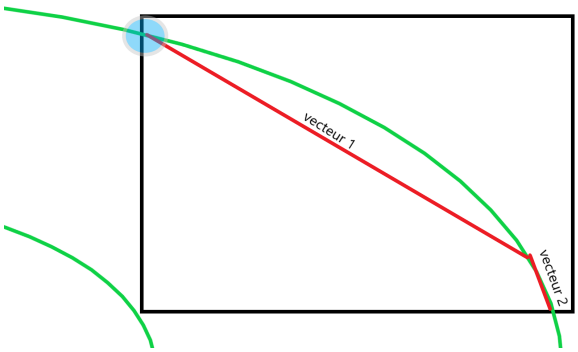

FIGURE 3 – Schéma d'un virage

Le vecteur 1 est naturellement le plus grand des deux, et si les coordonnées du point au centre du cercle bleu sur l'image, sont suffisamment basses, alors l'état virage est activé.

Pré-virage

Le pré-virage est un état que nous avons créé en raison des contraintes liées à la caméra. Cette dernière captait les informations un peu trop loin, et avait donc tendance à indiquer un état virage alors que la voiture n'était pas encore dans le virage. La voiture tournait donc trop tôt. Le test de l'état pré-virage est fait après celui de l'état virage, et est donc activé lorsqu'au moins deux vecteurs sont détectés par la caméra, et que le plus incliné de tous l'est suffisamment. La situation décrite par le schéma sur la figure 4 est une situation de pré-virage : la caméra a détecté un virage mais il est trop tôt pour tourner. Six vecteurs sont détectés, mais le vecteur 1, avec l'angle d'inclinaison le plus grand, n'est pas assez grand pour passer la voiture dans l'état virage. C'est donc un pré-virage.

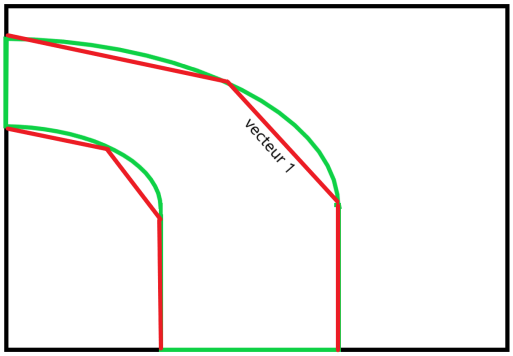

FIGURE 4 – Schéma d'un pré-virage

Objectif attendu

Nous souhaitions pour la compétition que la voiture adopte un comportement optimal vis à vis de la piste, c'est à dire qu'elle ralentisse avant les virages et accélère dès l'arrivée dans une ligne droite par exemple. Elle devait également parcourir la piste en suivant une trajectoire qui se rapprochait au maximum de la trajectoire optimale pour le circuit.

Objectif atteint

En pratique, nous avons réussi à faire en sorte que la voiture ne sorte pas de la piste, et ce sur des circuits variés. Mais la voiture n'adopte pas des trajectoires optimales, et ce même si nous varions les vitesses utilisées en fonction de l'état de la voiture. Ceci premièrement parce que la voiture peut théoriquement parcourir la piste plus rapidement, mais que notre algorithme ne le permet pas. Secondement parce qu'en pratique la voiture se trouve

souvent en état de pré-virage, car la caméra voit loin devant elle et voit donc le bout du virage alors qu'elle est encore dans une ligne droite.

Autres tentatives

Nos premiers algorithmes tentaient d'aligner les roues avec la moyenne de l'angle des vecteurs que la caméra détectait (donc pour un vecteur vertical les roues seraient droites, et pour un vecteur en diagonale, les roues seraient alignées selon cette diagonale) ; mais le fonctionnement n'était pas du tout approprié, et ce parce que la caméra détectait les virages encore une fois trop tôt.

Nous avons également tenté d'implémenter un algorithme qui faisait en sorte que la voiture reste au centre de la piste, mais cela n'a pas abouti.

2.2.3 Détection des passages piéton

Objectif attendu

Cette épreuve avait pour principal objectif de détecter des marquages au sol du type passage piéton. Il y en avait de deux sortes comme l'illustre la figure 5. La voiture devait donc détecter ces bandes noires placées n'importe où sur une portion de ligne droite du circuit. A la détection des quatre bandes elle rentre dans une zone de réduction de vitesse et reprend sa vitesse initiale lorsqu'elle détecte les trois bandes. Nous avons implémenté deux approches différentes pour détecter ces bandes noires.

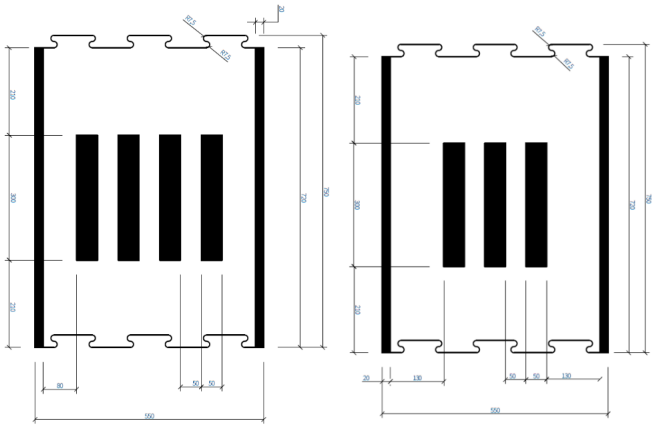

FIGURE 5 – Marquages au sol à détecter

Première approche

Nous avons choisi dans un premier temps de détecter les bandes noires par la même méthode que la détection des bordures : en utilisant le mode vecteur fourni par la Pixy2. Étant donné que celle-ci est sensible au contraste de couleur elle renvoie donc des vecteurs associés au bandes noires sur la piste blanche. Une principale difficulté est que la Pixy2 ne nous renvoie pas que les vecteurs associés aux bandes du centre, ce qui était prévisible car elle détecte aussi les bordures de la piste. Il nous a fallu donc tout d'abord extraire dans le tableau de points que fournit la Pixy2 ceux correspondant aux bandes du centre.

Nous avons l'assurance, d'après les règles de la NXP, que les pièces de la figure 5 ne pourraient être placées que sur une ligne droite et pas en plein milieu d'un virage. Cela nous assure donc que les vecteurs de bordures sont deux vecteurs droits aux extrémités (et pas plusieurs morceaux de vecteurs dû aux virages). Ainsi pour traiter les bandes de passages piéton nous ne considérons pas les vecteurs de plus grands et plus petits abscisses, ceux ci sont récupérés à travers la fonction *extrem_vectors*. Une fois les bordures écartées nous ne pouvions pas simplement dénombrer les vecteurs restants et affirmer la détection des trois ou quatre bandes. En effet la caméra pourrait détecter des imperfections sur la piste ou autour et les considérer comme un vecteur, s'en tenir au nombre de vecteurs n'est donc pas approprié. Nous avons donc choisi d'ajouter une autre condition à savoir que les vecteurs restants (après extraction des bordures) doivent avoir les mêmes normes et mêmes angles de déviation à des coefficients près. Ceci est illustré par l'extrait de code en figure 6. Les valeurs 0.2 et 20 sont empiriques, elles ont simplement été choisies à la suite de différents tests que nous avons pu réaliser. Une fois que nous détectons ces passages piétons nous changeons la valeur d'une variable globale qui nous sert à savoir s'il faut ou pas réduire la vitesse de la voiture.

```
328
329     for (uint8_t i = 1; i < n; i++) {
330       double diff_norm = abs(vectors_norm[0] - vectors_norm[i]) / vectors_norm[0];
331       double diff_angle = abs(vectors_angles[0] - vectors_angles[i]);
332       if(diff_norm > 0.2 || diff_angle > 20) //0.2 and 20 are chosen values, have to be tested
333         return false;
334     }
335     return true;
336   }
```

FIGURE 6 – Extrait de code pour la détection des bandes de passages piéton.

Tests réalisés avec cette première approche

Nous avons premièrement réaliser la majorité des tests avec la voiture à l'arrêt posé sur la piste. Ces tests ont permis de vérifier que la caméra détecte bien les bandes de passage piéton placées sur la piste. Cela fonctionnait bien pour la plupart du temps mais nous avons constaté qu'avec une légère déviation la caméra ne détecte pas certaines bandes noires de passage piéton. Cela allait être problématique pour une voiture en mouvement car même si la pièce de passage piéton est sur une ligne droite, rien ne garantit que la voiture viendra à coup sûr en ligne droite. Notre code ne fonctionne que dans des conditions idéales, qui sont improbables dans la pratique. Il fallait donc reconsidérer l'implémentation ce qui nous a amener à une deuxième approche.

Deuxième approche

Au lieu donc de chercher à détecter entièrement les bandes de passage piéton par le mode vecteur de la Pixy2 nous nous sommes orientés sur les **barcodes** (figure 7).
En effet c'est un autre mode de détection de formes que possède la Pixy2. Comme on peut le constater elle possède seize formes prédéfinies qu'elle sait reconnaître pour effectuer un traitement, également prédéfini, sur chacun. Comme on peut le constater le passage piéton avec les quatre bandes se rapproche bien des codes barre 0, 14 et 15 et effectivement la caméra la détecte comme tels. Cependant le même problème que précédemment s'est posé. Suite à une légère déviation de la voiture, elle associe le symbole à un autre code barre comme le numéro 8. Pour régler ce soucis nous n'avons donc pas cherché un code barre en particulier. Dès que la voiture nous renvoie un code barre détecté nous considérons qu'il s'agit des quatre bandes. Nous avions la possibilité de faire ce choix car en ce qui concerne les trois bandes la caméra ne détectait jamais un code barre, on n'avait donc pas à s'interroger quand on recevait un code barre.

L'implémentation en utilisant les codes barres nous a permis de résoudre une partie du problème à savoir la détection des quatre bandes. Mais pour les trois bandes, il n'y avait de code barre associé. Nous avons émis l'hypothèse d'ajouter de nous même la pièce des trois bandes comme code barre à détecter par la Pixy2. Cependant, cette piste n'a pas pu être creusée plus en profondeur au vu des évènements qui ont suivis. Avant de passer à la version virtuelle du projet, nous avons une détection imparfaite des passages piétons. Les quatre bandes sont bien détectées en utilisant les codes barre mais en ce concerne les trois bandes nous sommes restés sur la première approche d'implémentation qui, comme expliqué plus haut, ne les détecte pas à coup sûr.

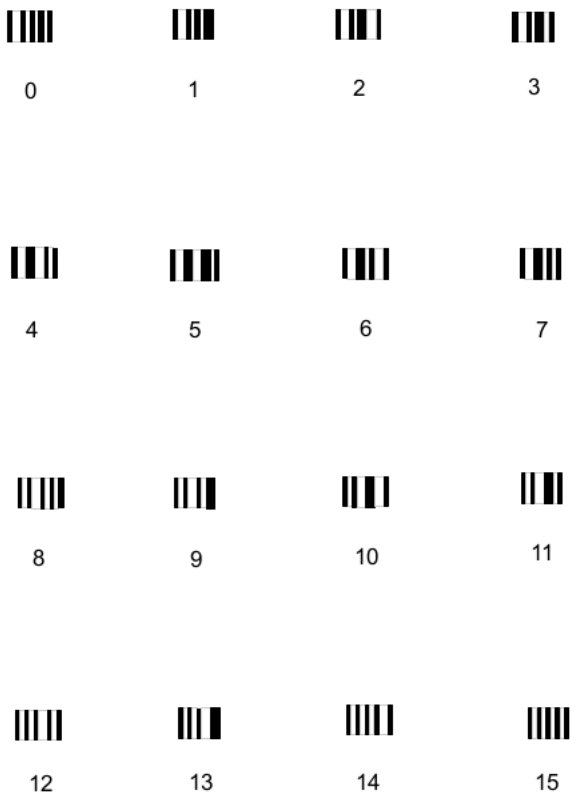

FIGURE 7 – Ensemble des *barcodes* que détecte la pixy2

2.2.4 Détection de la ligne d'arrivée

Un autre marquage au sol important à prendre en compte pour la compétition est la ligne d'arrivée (figue 8). Pour l'épreuve principale, l'équipe à le droit à trois essais. Le premier essai valide est celui qui arrive à franchir la ligne d'arrivée sans commettre de fautes.

Pour détecter la ligne d'arrivée, nous avons décidé de réutiliser la première approche de la détection des passages piéton. En effet, la Pixy 2 détecte assez bien les deux lignes horizontales de la ligne d'arrivée. Ainsi, il suffit de reprendre le code en vérifiant cette fois que l'extrémité de chaque vecteur se trouve sur la même abscisse et que ces derniers ont la même norme à une constante près.

Cependant, par faute de temps nous n'avons pas pu tester correctement notre code avant de passer à la version virtuelle.

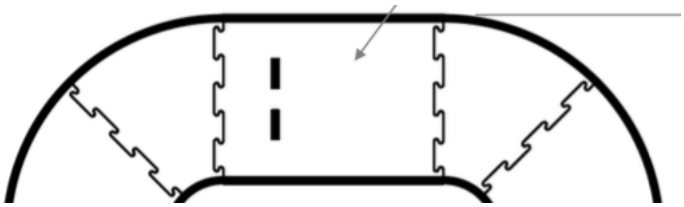

FIGURE 8 – Marquage au sol désignant la ligne d'arrivée

2.2.5 Détection et évitement d'obstacle

Cette partie se concentre sur deux épreuves optionnelles : l'arrêt d'urgence et l'évitement d'un obstacle. Dans ces épreuves, un cube blanc, dont on connaît les dimensions, est placé sur une ligne droite de la piste de manière aléatoire. Le cube étant blanc il n'est pas détectable avec la Pixy2. C'est pourquoi nous avons eu recours à un capteur de distance Lidar.

Détection du cube :
Pour l'arrêt face à un obstacle, le cube est placé au milieu de la piste et donc un unique Lidar est suffisant pour le détecter. Néanmoins, pour l'évitement de l'obstacle, le cube est placé sur un bord quelconque de la piste. Un Lidar à balayage était envisagé, ainsi que l'utilisation de deux Lidars simples à placer de chaque côté de la voiture afin de détecter de chaque côté. Notre choix de capteur de distance s'est porté sur le TF-Mini qui mesure des distances de 0,3 à 12 mètres. Les données du capteur sont codées sur 9 octets. Les deux premiers octets sont réservés à l'en-tête. Les 3e et 4e octets sont pour la distance, les 5e et 6e octets sont pour l'intensité, et les 3 derniers octets sont réservés. La distance et l'intensité sont chacun deux octets car le premier est un octet de poids faible et le second un octet de poids fort. L'algorithme pour utiliser le capteur de distance fonctionne de cette manière :

- Dès que 9 octets sont disponibles, on vérifie que les deux premiers correspondent à l'en tête 0x59.
- On lit les octets de distance et d'intensité, en décalant à gauche de 1 octet celui de poids fort, et on les additionne afin d'avoir un seul entier pour la distance et pour l'intensité.
- On lit les 3 derniers octets mais on ne les utilise pas.

De cette manière on obtient la distance en centimètre. L'algorithme calcule également l'intensité mais nous ne l'avons pas exploité car nous voulions déjà nous concentrer sur l'arrêt et l'évitement avant de se pencher sur cette valeur.

Freinage :
Pour le freinage on vérifie la distance renvoyée par le Lidar. Il faut que cette dernière soit comprise entre la distance minimum renvoyée par le Lidar et une distance de sécurité afin de laisser le temps de freiner. Cet intervalle de distance est nécessaire car en testant le Lidar nous avons remarqué qu'il peut parfois renvoyer une distance nulle alors qu'elle n'est pas dans sa plage de mesure et qu'il n'y a pas d'obstacle. Ainsi, on utilise un intervalle avec comme distance minimum, celle mesurable par le capteur. Ensuite, si la distance mesurée par le capteur est comprise dans cet intervalle défini, il faut envoyer une vitesse nulle afin de s'arrêter.

Évitement :

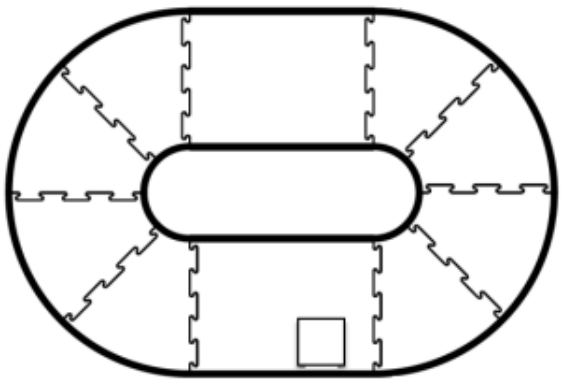

FIGURE 9 – Cas d'un évitement d'obstacle

L'évitement d'un obstacle est plus complexe que le freinage car il faut détecter de quel côté l'obstacle se situe par rapport à la piste, afin de savoir de quel côté tourner. Par ailleurs, comme le cube blanc est au bord de la piste, il cache un bord noir ce qui peut être exploité grâce à la caméra. En effet, puisque la caméra ne renverra pas de vecteur là où le cube cache le bord noir car il y aura une continuité de blanc à cet endroit. Notre piste de recherche était d'associer la caméra et le capteur de distance : le capteur permet de détecter un obstacle et la caméra permet de savoir quel bord de la piste est libre.

Par ailleurs, le fait de connaître les dimensions est utile car cela permet de savoir quelle proportion du circuit le cube occupe. Ainsi, on peut utiliser ces informations afin de limiter le risque que la voiture percute l'obstacle. Nous n'avons pas pu implémenter un algorithme d'évitement d'obstacle par manque de temps mais nous avons travaillé sur cette épreuve lorsque nous sommes passés sur une modélisation virtuelle. Notre recherche sur cette modélisation sera expliquée dans la partie 3.

2.3 Étude de validité

2.3.1 Outil de débogue mis en place

L'un des grands problèmes rencontrés dans ce projet est de savoir ce que la voiture voitlorsqu'elle roule. Une solution a été établie pour voir en temps réel le flux d'images récupéré par la Pixy2 en utilisant une Raspberry Pi 3.
Après avoir connecté la Pixy2 à la Raspberry qui est alimentée par une batterie, on peut se connecter sur la Raspberry et la contrôler à distance. Pour cela, on connecte la Raspberry sur le réseau WIFI d'un ordinateur. Pour visionner, nous avons décidé de créer un point d'accès sur l'ordinateur et de connecter la Raspberry sur ce point. Ainsi la récupération du flux sera plus rapide grâce à l'absence d'un routeur entre l'ordinateur et la Raspberry.
Il y a beaucoup de logiciels qu'on peut utiliser pour se connecter sur la Raspberry (Remote desktop sur Windows par exemple), puisqu'on utilise Linux comme système d'exploitation on a utilisé "rdesktop" pour cette tâche. Après être passé par les étapes précédentes, on peut se connecter en exécutant la commande suivante :

```
rdesktop -u pi <adresse IP de la raspberry>
```

où pi représente le nom d'utilisateur par défaut sur la Raspberry, un mot de passe sera demandé après cette étape qui est par défaut "raspberry" [1]. Finalement on lance le logiciel "Pixymon" pour récupérer le flux de la caméra.
Cette méthode aide à voir la piste du point de vue de la voiture et avoir une bonne idée pour raisonner, comme elle aide à détecter les bruits qui peuvent fausser les résultats de nos algorithmes.

2.3.2 Tests réalisés

2.3.3 France 3

Une première démonstration officielle de notre voiture autonome a eu lieu le 13 janvier 2020. En effet, dans le cadre de l'inauguration de la nouvelle ligne de tramway à

Bordeaux, l'équipe de France 3 Aquitaine est venue faire un reportage à l'ENSEIRB-MATMECA afin de présenter les nouvelles technologies liées au déplacement. Notre projet a donc été présenté à la télévision par Théo Matricon avant son départ.

Pour la présentation, nous avons utilisé la voiture 1/10. Cette dernière était capable de suivre de manière autonome n'importe quelle piste à une allure lente. Cependant, cette piste ne devait pas contenir d'intersections ou de marquages au sol.

Nous avons beaucoup appris de cette expérience. En effet, nous avons été prévenus du passage de France 3 peu avant le Jour J. En quelques jours, nous avons été capables de rendre la voiture autonome. Pour atteindre cet objectif, nous avons dû travailler dans l'urgence et de manière efficace ce qui a été concluant.

2.3.4 Journée à NXP Toulouse

Autre événement lié au PFA, la dry run à Toulouse. En effet, le jeudi 27 février 2020, l'entreprise NXP organisait une petite réunion dans ses locaux pour permettre aux équipes de tester leurs voitures, poser des questions, se rencontrer.

— Organisation : Nous sommes partis en équipe réduite dans un unique véhicule (celui de monsieur Rollet) au matin du 27 direction Toulouse. Arrivés en fin de matinée, nous découvrons les locaux et quelques autres équipes.

— Problèmes rencontrés sur place : En arrivant nous découvrons que plusieurs câbles se sont dessoudés durant le transport. Ils sont ressoudés dans les locaux de NxP. Mais cela n'est pas concluant car la voiture ne réagit aux images reçues. Après analyse sur place nous en déduisons que la carte a grillé et n'ayant pas de carte de remplacement, nous ne pouvons faire rouler la voiture. Mais cette journée aura tout de même eux de nombreux points positifs.

— Déduction de cette journée : Cette journée a pu amener à plusieurs conclusions.

1. Le transport doit être pris en compte pour la compétition afin de ne pas tout perdre avant que la course débute pour des problèmes de matériel. Bien que l'idée ai été évoquée avant, le simple carton ne permettait pas de stocker la voiture convenablement.

2. Nous avons pu observer le niveau et les stratégies des autres équipes. Dans une utilisation ultérieure, cela aurait pu être utile si le projet ne s'était pas interrompu brusquement (Expliqué dans la suite).

3. L'équipe d'électronique a pu observer notre voiture et nous signaler que nos branchements n'étaient pas propres et non sécurisés. Encore une fois, une collaboration avec la filière électronique a alors été évoqué par la suite pour imprimer des cartes et parler des stratégies mais cela ne sera pas mis en place.

3 Deuxième version : Version Virtuelle

3.1 Changement de contexte

A partir du mois de mars, suite à la situation sanitaire actuelle, le projet se met en pause pour une reprise fin mars sur une nouvelle version. Nous passons à une version simulée sur ordinateur.

3.1.1 Spécification du nouveau besoin

Après réunion avec le client, nous avons désormais un nouvel objectif. L'idée générale est de continuer le projet sous Unity mais après quelques semaines, l'équipe se scinde en deux pour gérer deux aspects : certains membres étant grandement intéressés par l'IA, ils essayent des pistes d'intelligence artificielle et l'autre partie se charge d'implémenter l'évitement d'obstacle.

3.1.2 Nouvel environnement de travail

Pour ce nouveau projet, un module est fourni par notre client, monsieur Barthou, sous Unity. Le code est donc développé sous forme de scripts C sharp qui sont ensuite importés dans Unity. Après une période d'adaptation aux nouveaux logiciels et langages, la programmation a pu débuter.

3.2 Développement réalisés

La figure 10 est un diagramme de Gantt qui présente les tâches réalisées par chacun pendant la deuxième phase du projet. Comme nous pouvons le remarquer, les tâches sont beaucoup moins nominatives. En effet, nous avons scindé l'équipe en deux : Sonia, Valérian et Carla ont travaillé sur l'évitement d'obstacle, tandis que Otavio, Abdelouahab, Alexandre et Tara ont travaillé sur l'intelligence artificielle. De plus, il a fallu passer du temps sur le rapport que vous lisez actuellement.

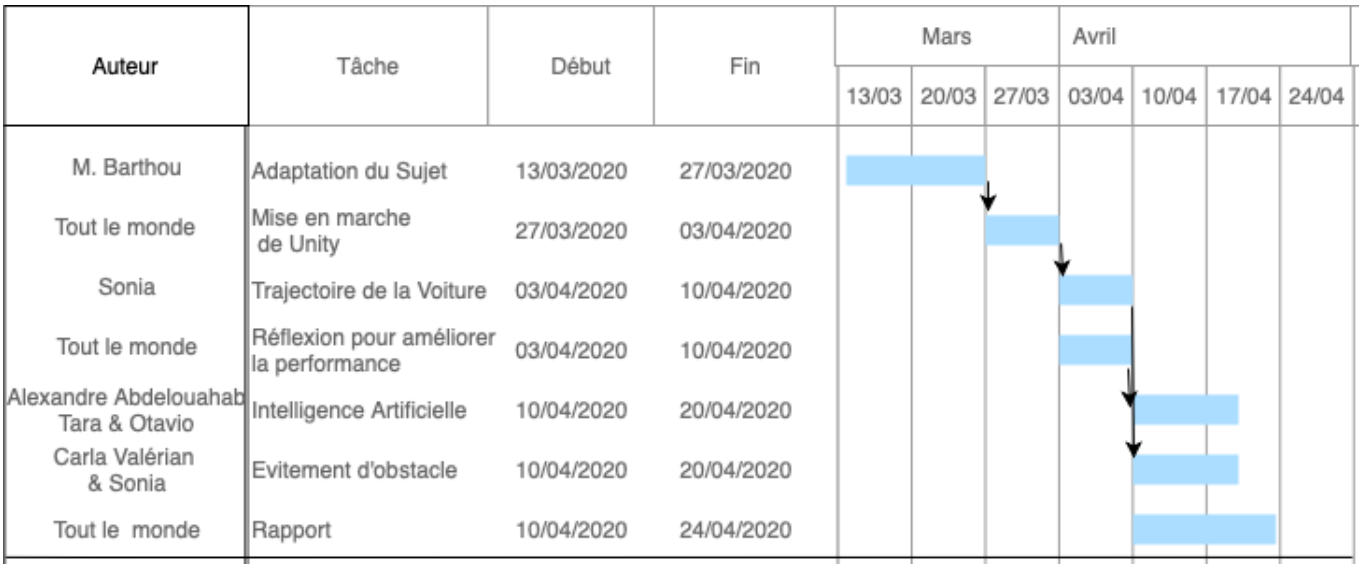

FIGURE 10 – Diagramme de Gantt de la deuxième version

3.2.1 Amélioration du code de suivi de piste

Afin de tester la détection d'obstacle, il fallait déjà que la voiture reste sur la piste. Nous expliquons dans cette partie notre démarche pour faire cela. **Situation de départ** Nous avions déjà un script fourni par le client qui permettait de suivre la piste mais la voiture déviait parfois et sortait de la piste. Nous avons essayé donc d'améliorer ce comportement. La stratégie implémentée était de se placer toujours au centre de la piste en regardant la position de la ligne gauche.
Contrairement au modèle physique où nous étions dans l'espace et que la Pixy2 détectait les bordures par des vecteurs, dans ce modèle virtuel, la caméra nous retourne une image autrement dit une matrice de taille $SIZEVIEW \times SIZEVIEW$ (avec $SIZEVIEW =$

128 dans notre cas) contenant des valeurs de pixels. Chaque ligne de cette matrice contient les valeurs de 128 pixels d'une ligne horizontale dans le champ de vision de la caméra (128×128). Si la voiture est placée bien au milieu des deux bordures, on sait que la bordure gauche est au pixel 17 et la droite au pixel 102 de la ligne horizontale au centre de l'image donnée par la caméra. La stratégie dans le code fournit par le client consiste à parcourir la ligne horizontale du milieu de l'image. On s'arrête dès qu'on a un pixel sombre et on en déduit la position de la ligne gauche. L'angle de correction est donc calculé en fonction de l'écart entre la position de ligne gauche détectée et celle où elle devrait être (c'est à dire au pixel 17). Si la ligne gauche est bien à la position 17 on continue tout droit.

Problème constaté

Avec cette implémentation les tests ont montré que la voiture reste bien au milieu de la piste en ligne droite mais pas dans les virages. De plus, lorsque la voiture est dans d'une intersection et qu'elle a une bordure noire en face elle continue tout droit et sort donc de la piste. Comme on peut le constater sur la figure 11, la voiture a en face d'elle une bordure noire. En parcourant la ligne au milieu comme précédemment elle trouvera à coup sûr une valeur sombre dans les premiers pixels disons par exemple 5. L'écart entre 17 et 5 étant faible l'angle de correction est très faible et la voiture ne va tourner que faiblement (voir pas du tout quand on a égalité). Ce qui fait qu'elle poursuit son chemin et sort de la piste.

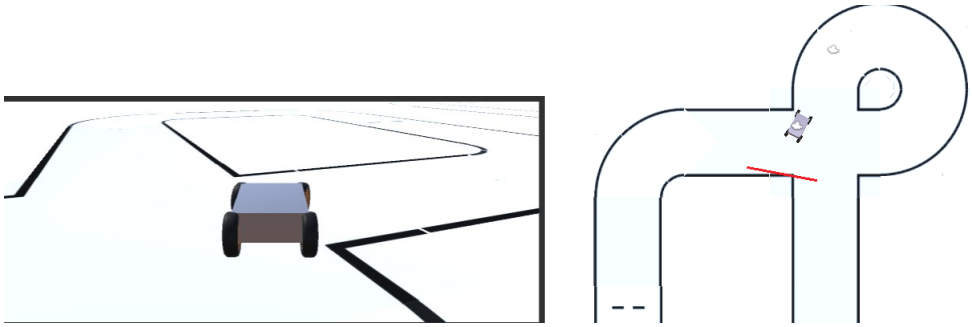

FIGURE 11 – Illustration du cas problématique

Piste de résolution

Nous avions pensé à deux solutions pour résoudre ce problème. La première était de modifier la stratégie du code pour permettre que la voiture soit toujours au centre de la piste même dans les virages. Nous pensons en effet que si elle reste au milieu de la piste

dans le virage, en sortant de celui-ci elle continuera tout droit sans croiser la bordure noire en face. Cela est illustré sur la figure 12, en bleu c'est le comportement actuel de la voiture et en rouge ce qu'elle ferait si elle prenait le virage en restant au centre. Nous n'avons pas réussi à mettre en place un code pour réaliser cela.

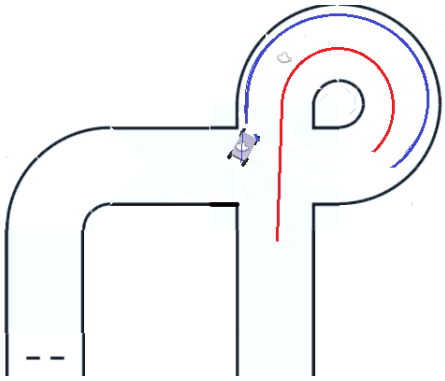

FIGURE 12 – Illustration du comportement de la voiture dans le virage

Nous nous sommes penchés sur la deuxième solution qui ne résout pas totalement le problème mais qui permet pour ce circuit en particulier de terminer la course sans en sortir. Le but étant de détecter lorsque la voiture a exactement en face d'elle la bordure. Pour cela nous parcourons une deuxième ligne horizontale de l'image que donne la caméra. Nous avons choisi cette nouvelle ligne proche du milieu de l'image, autrement dit si la ligne du milieu de l'image est la ligne numéro n, nous choisissons la ligne numéro $n+2$. la stratégie étant de voir si on trouve la bordure noire à la même position de pixel que pour la ligne du milieu. Il convient de ne pas choisir la ligne numéro $n+1$ car est trop proche de la ligne du milieu. En effet nous avons constaté que même lorsque la voiture est au centre de la piste ces deux lignes, n et $n+1$, pouvaient avoir la bordure noire sur la même position de pixel. Le choix de deux rangs d'écart nous a permis donc d'isoler ce cas de figure. Une fois qu'on détecte une bordure noire en face, le traitement était donc de faire tourner la voiture pour la replacer sur la piste.

Cette solution n'attaque pas le problème à la racine puisque nous n'avons pas réussi à éviter que la voiture soit dans le cas illustré sur la figure 11. Nous avons néanmoins cherché à apporter une légère amélioration, bien qu'imparfaite, qui permet de rester sur la piste.

3.2.2 Intégration IA

En utilisant un environnement virtuel, il nous est permis de simuler rapidement les performances de différents algorithmes, c'est pourquoi nous avons pensé à un contrôleur de voiture qui pourrait utiliser des techniques d'apprentissage automatique pour contrôler la voiture.

Au début, comme nous pouvions simuler différentes courses dans la piste, notre première idée a été d'utiliser les techniques de Q-Learning ou Deep Q-Learning tel qu'introduites dans la référence [3]. Quelques premières tentatives ont été faites, mais le temps de simulation était très long (nous n'avons pas trouvé le moyen de faire fonctionner Unity *headless*, c'est-à-dire de faire tourner l'engin sans vraiment afficher (*render*) l'interface). Nous avons alors décidé de changer d'approche (même si nous pensons toujours que la précédente pourrait être celle qui donne les meilleurs résultats) pour une approche plus viable dans le temps dont nous disposions.

La nouvelle approche consiste, au lieu de faire rouler la voiture plusieurs fois automatiquement a essayer d'apprendre comment conduire la voiture nous-mêmes puis à former un algorithme d'apprentissage automatique pour nous imiter. C'est ce qu'on appelle l'apprentissage par imitation et cette idée est basée sur le travail développé dans la référence [5]. Même si le problème à résoudre dans cet article est complètement différent, l'idée d'apprendre du comportement humain est toujours la même.

Cela dit, notre premier objectif est de générer le jeu de données (*dataset*). En regardant simplement comment le projet a été structuré, nous avons défini que nous possédons en entrée l'image de la caméra et en sortie les valeurs de l'angle des roues, le couple moteur et le couple de freinage qui sont les trois valeurs utilisées pour conduire la voiture.

Avec cela, nous avons roulé avec la voiture pour un tour complet et généré un simple jeu de données (*dataset*) qui contient 1242 images et ses valeurs de couple et d'angle respectives. Chaque image est un RGB au format 128x128. Avec ce jeu de données, nous avons essayé différentes approches pour prédire les valeurs des couples et des angles d'une image donnée.

Notre première tentative a été un réseau de neurones convolutifs (CNN) [4] dans l'image directe (sans prétraitement) avec 4 couches convolutionnelles, une couche entièrement connectée avec 128 neurones, puis une autre entièrement connectée avec 256 et une dernière couche de 3 neurones. Nous avons utilisé la fonction d'activation ReLU et comme il s'agit d'un problème de régression et non de classification, nous avons également opté pour la perte d'erreur quadratique moyenne (MSE).

Nous avons essayé plusieurs architectures pour CNN (différentes quantités de couches, différentes quantités de neurone par couche) et celle-ci a les meilleurs résultats bien que restant assez mauvais.

Nous pouvons voir le résultat de l'entraînement dans la trame 13 suivante :

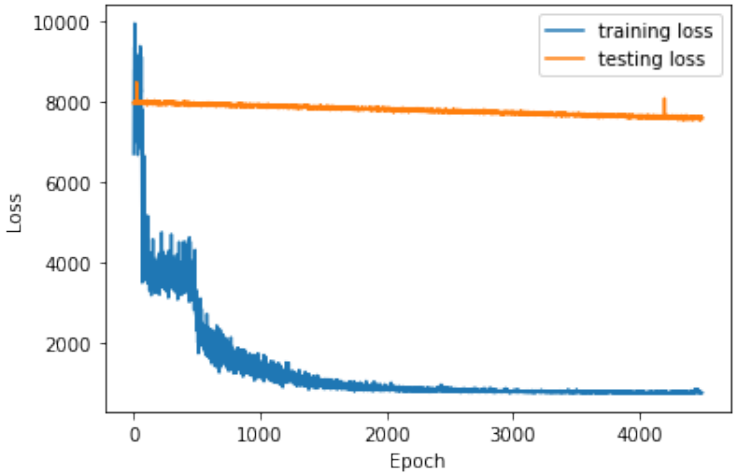

FIGURE 13 – Résultats des premiers CNN

Ce résultat n'est pas bon, car même si l'erreur d'entraînement (training) diminue, l'erreur de test est fondamentalement stable, ce qui signifie que notre réseau n'apprend pas à faire la régression jusqu'à la valeur correcte, mais se contente en fait, de mémoriser quelques valeurs. Le partie plus difficile dans la version IA est de surmonter ce problème et pour trouver une solution nous avons dû faire quelques pas en arrière. Inspiré par le travail original de CNNs [2] nous avons décidé d'utiliser la même architecture 14 que celle de ce document en modifiant la quantité de neurones dans la dernière couche à 3 et la fonction de perte pour MSE (comme suggéré précédemment étant donné que nous avons un problème de régression).

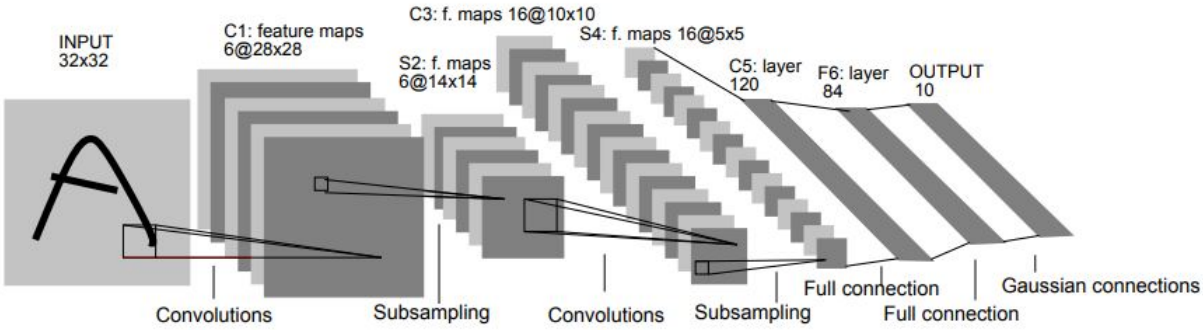

FIGURE 14 – Architecture de CNN LeNet5 Original

Pour s'adapter à cette architecture, nous avons dû pré-traiter les images de notre jeu de données (*dataset*). Nous avons adopté trois étapes simples pour ce pré-traitement : tout d'abord, les images doivent être en niveaux de gris. Ensuite, nous les dimensionnons en images 28x28 et enfin nous mettons en blanc les quatre premières rangées de pixels (pour effacer la skybox de Unity). Il en résulte une transformation, comme celui dans l'image15 suivante :

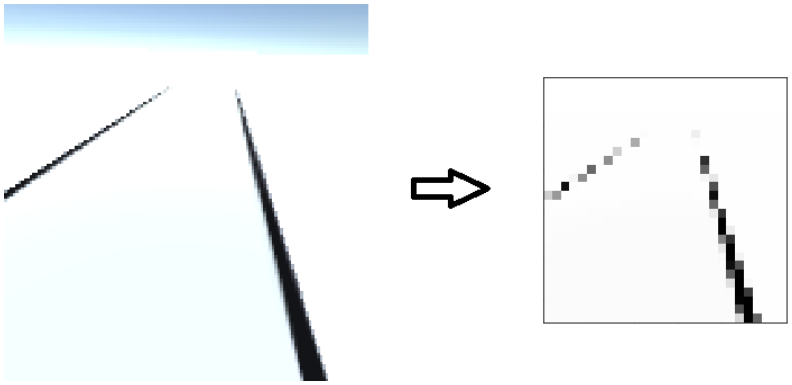

FIGURE 15 – Résultats du pré-traitement de l'image

Grâce à tout cela, nous sommes finalement parvenus à un meilleur résultat. Notre erreur de test a commencé à diminuer et nous avons quelque chose qui semble bien meilleur, comme indiqué dans l'image suivante :

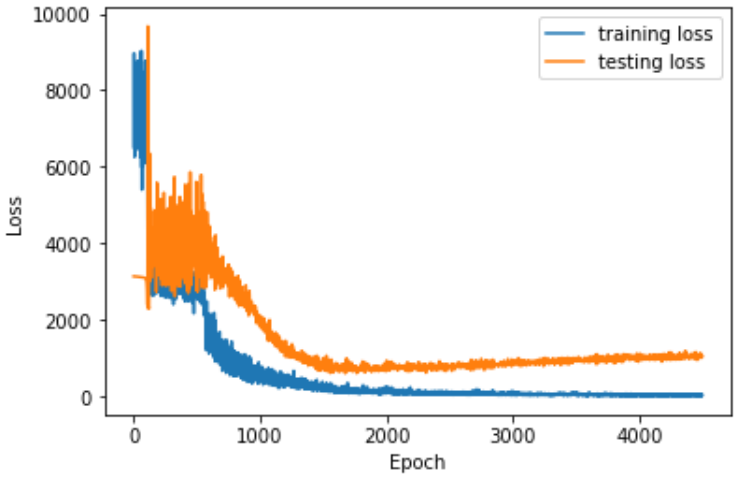

FIGURE 16 – Résultats de le training

Comme on peut le voir sur l'image 16, c'est vers 1700 que nous obtenons les meilleurs résultats sur jeu de test (*testing set*). Après cela, c'est essentiellement du sur-entrainement (*overfitting*) sur le jeu d'entrainement (*training set*), donc nous arrêtons l'entraînement et obtenons les paramètres à environ 1700. Maintenant, même si l'erreur est faible, le comportement de la voiture avec ces valeurs n'est pas encore idéal notamment car l'espace de recherche (les valeurs que les sorties peuvent avoir) est trop large.

Ceci est notre mise en œuvre finale, mais nous avons discuté de quelques idées d'améliorations futures. La première, afin de réduire l'espace de recherche, consiste à utiliser des options binaires telles que "accélérer", "tourner à gauche", "tourner à droite", au lieu d'utiliser les valeurs d'angle et de couple comme sortie, et, de ce fait, à changer également la fonction de perte en cross-entropie binaire ou quelque chose de similaire. Une autre possibilité discutée a été d'ajouter la vitesse actuelle de la voiture en entrée, ce qui peut se faire soit en l'ajoutant à l'image (par exemple, plus la voiture est rapide, plus l'image est "rouge") ou éventuellement aussi en ajoutant simplement un nouveau neurone à la première couche entièrement connectée du réseau et en alimentant directement la valeur.

3.2.3 Évitement d'un obstacle

Nous nous sommes également penchés sur l'évitement d'un obstacle puisqu'il s'agit d'une épreuve facultative qui se prête à la modélisation 3D. Pour ce faire, nous avons créé un cube 3D en guise d'obstacle, similaire aux conditions réelles. La procédure de détection d'obstacle diffère du modèle réel puisque nous n'utilisons pas de capteurs de distance. Il existe des modules Lidar sur Unity mais les versions ne sont pas compatibles avec celle que nous utilisons et nous avons jugé plus simple de détecter l'obstacle grâce aux coordonnées dans l'espace.

Le principe est de récupérer les vecteurs positions de la voiture et de l'obstacle et de les exploiter en calculant d'autres vecteurs. Bien que seules deux coordonnées nous intéressent, nous utilisons des vecteurs à 3 coordonnées pour que ce soit plus compréhensible et pour faciliter les opérations entre vecteurs. De ce fait, les coordonnées nécessaires pour l'évitement d'obstacle sont celles sur les axe x et z, y étant la hauteur.

A partir des coordonnées des deux objets, on peut calculer :

— le vecteur distance voiture/obstacle (vecCarCube)

— le vecteur vitesse de la voiture obtenue grâce à la différence des vecteurs positions de la voiture à t et $t + 1$ (speedCar)

— l'angle entre le vecteur vitesse et le vecteur distance

La figure 17 permet d'illustrer les deux vecteurs et l'angle entre les deux. Le vecteur rouge speedCar est obtenue grâce à la dérivée de la position de la voiture dans le temps, c'est pourquoi il y a deux voitures sur cette figure : celle aux temps t et $t+1$. Le vecteur vert verCarCube est le vecteur directeur de la distance entre la voiture à t et l'obstacle.

Ensuite, il faut vérifier que le vecteur vitesse de la voiture est proche du vecteur entre la voiture et l'obstacle vecCarCube. Dans ce cas, la voiture peut se diriger vers l'obstacle, il faut alors regarder l'angle entre ces deux vecteurs et faire tourner la voiture dans le cas où l'angle entre les deux est trop faible. Un produit scalaire des deux vecteurs donne le cosinus de l'angle entre ces deux vecteurs, ce qui nous permet de déduire cet angle. Par ailleurs, il faut également comparer les angles avec la normale des deux vecteurs afin d'obtenir le sens dans lequel la voiture doit tourner. Cependant lors des tests, les roues ne semblent pas tourner bien que les valeurs sont modifiées.

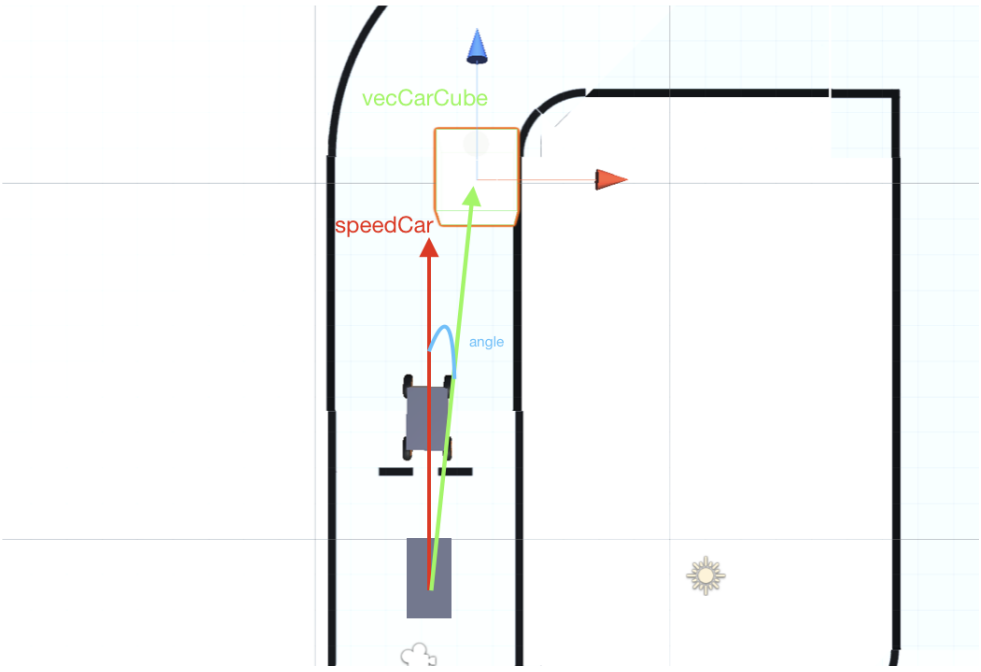

FIGURE 17 – Vecteurs dans le cas d'un évitement d'obstacle

4 Problèmes rencontrés - Réflexion

4.1 Électronique

La gestion du matériel électronique nous a grandement ralenti pendant ce projet, premièrement parce nous utilisions pour la première fois la majorité des composants qui nous ont été utiles, et deuxièmement parce nous avons cassé du matériel en raison de notre manque de connaissances.
Nous avons par exemple grillé deux cartes Teensy4 en raison de l'absence de régulateur de tension entre l'ESC et cette dernière, mais également abîmé une batterie en raison de notre ignorance de la procédure d'utilisation qu'il faut connaître pour la garder en bon état.
A propos justement des ESC, des outils dont nous n'avions pas du tout l'habitude d'utiliser, il nous fallait prévoir 1 à 2 minutes de calibrage avant de pouvoir tester le code avec de nouveaux paramètres. Cela fut vraiment contraignant.

4.2 Matériel Commandé

Nous avons eu des problèmes au niveau du matériel notamment avec les voitures échelles 1/16 que nous avons commandées. Elles étaient en théorie les modèles officiels de la NXPCup, mais à la réception nous nous sommes rendus compte qu'il manquait des pièces, et ce dans chacune des trois boîtes que nous avons reçues. De plus elles n'étaient ni équipées d'ESC ni de batteries, nous demandant plus d'investissements que d'autres modèles.
Nous avons finalement décidé de ne pas les utiliser pour la compétition et nous sommes plutôt penchés vers les modèles 1/18 qui eux étaient complets, fonctionnels et mieux construits (suspension, arbre de transmission, différentiel).

4.3 Pistes d'amélioration

L'une des améliorations sur notre voiture autonome serait d'utiliser le SLAM, c'est-à-dire la localisation et cartographie simultanées afin d'améliorer la trajectoire mais également la rapidité de ce dernier. Cela consiste, pour notre véhicule autonome à simultanément construire une carte de son environnement et de s'y localiser. La méthode pour utiliser cette technique est expliquée plus précisément dans le document de spécification.

Nous aurions également aimé améliorer l'algorithme de parcours sur un circuit afin de lisser la trajectoire de la voiture en y rajoutant plus d'états que ligne droite, pré-virage

et virage.

Il y a également les intersections que nous n'avons pas eu le temps d'implémenter, elles auraient probablement fait l'objet d'un "état" en plus des trois existants.
L'idée globale de détection des intersections se baserait sur les quatre vecteurs que détecte la caméra lorsqu'elle est au milieu d'un intersection, qui sont illustrés sur la figure 18

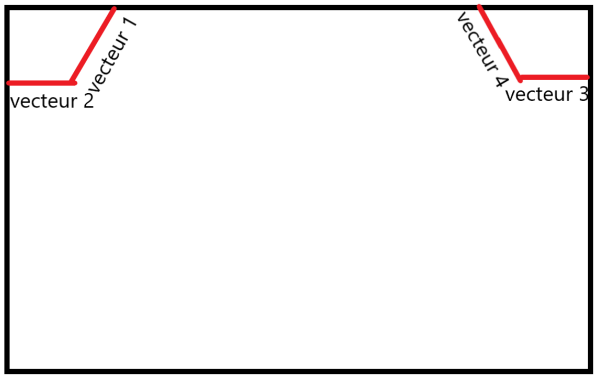

FIGURE 18 – Schéma des vecteurs dans une intersection

5 Conclusion

5.1 Vis à vis du cahier des charges

Nous avions scindé notre projet en deux phases à savoir une phase opérationnelle et une phase avancée. Concernant la phase opérationnelle notre objectif était d'avoir une voiture prête pour la course et toutes les épreuves facultatives en début mars. À cette période ci, la voiture avait encore des problèmes matériels ce qui a ralenti les recherches et tests pour les épreuves. Au niveau des épreuves facultatives, nous avons trouvé des stratégies d'implémentation pour toutes mais aucune n'a eu le temps d'être testée et validée correctement. Par ailleurs, la phase avancée se concentrait sur la stratégie du SLAM qui aurait pu être un moyen de nous démarquer des autres équipes mais que nous n'avons pas pu implémenter. Bien que nous n'avions pas prévu de passer sur une version simulée sur ordinateur, nous avons pu définir des objectifs à savoir explorer des pistes d'intelligence artificielle et implémenter l'épreuve facultative d'évitement d'obstacle. L'intégration IA a permis d'améliorer le suivi de la piste bien qu'il reste des améliorations. Concernant l'évitement d'obstacle, nous avons trouvé une alternative au capteur du distance mais les tests ne sont pas concluants sur la simulation Unity.

5.2 Ressenti général

Ce projet, même si quelque peu perturbé par le covid-19, aura tout de même été enrichissant pour l'ensemble des membres de l'équipe.
Pour la première partie de notre projet, nous pouvons globalement tous nous accorder sur l'impression que nous avons eue que l'électronique et la conception 3D a pris une place trop importante pourtant hors de notre champ de compétence. Nous nous attendions à développer plus de code et chaque tâche concernant l'électronique semblait être un frein à notre avancement. Nous savions que nous allions consacrer un peu de temps à notre modèle physique de voiture mais avoir un code fonctionnel et ne pas pouvoir faire fonctionner notre voiture était assez frustrant.
Cela dit il fut grandement satisfaisant de voir enfin notre modèle fonctionner convenablement, comme c'était le cas pour l'interview avec France3.

6 Bibliographie

Références

[1] https://www.raspberrypi.org/documentation/linux/usage/users.md.

[2] Yann LeCun et al. "Gradient-Based Learning Applied to Document Recognition". In : *Proceedings of the IEEE*. T. 86. 11. 1998, p. 2278-2324. URL : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665.

[3] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv : 1312.5602 [cs.LG].

[4] Keiron O'Shea et Ryan Nash. "An Introduction to Convolutional Neural Networks". In : *CoRR* abs/1511.08458 (2015). arXiv : 1511.08458. URL : http://arxiv.org/abs/1511.08458.

[5] Eike Rehder, Jannik Quehl et Christoph Stiller. "Driving Like a Human : Imitation Learning for Path Planning using Convolutional Neural Networks". In : 2017.

ANNEXES

NXP CUP
INTELLIGENT
CAR RACING

2ème année Informatique

Document de Spécification NXP Cup

Ouedraogo Sonia
Bahrami Tara
Leguay-Lenormand Carla
Jacobi Otavio
Laaroussi Abdelouahab
Despretz Valérian
Venot Alexandre
Matricon Théo

Encadrant :
M. Rollet Antoine
Client :
M. Barthou Denis

Table des matières

1 Introduction

Le présent document a pour objectif de définir l'ensemble des spécifications qui vont détailler le projet de notre équipe d'étudiants, qui est de participer à l'édition 2020 de la NXP Cup.
Notre équipe d'étudiants est composée de 8 personnes :

- Sonia Ouedraogo
- Carla Leguay-Lenormand
- Tara Bahrami
- Otavio Jacobi
- Abdelouahab Laaroussi
- Valérian Despretz
- Alexandre Venot
- Théo Matricon

Nous sommes tous étudiants en 2ème année d'école d'ingénieurs en informatique à l'ENSEIRB-MATMECA. Nous répondons à un sujet de PFA proposé par notre client Denis Barthou, professeur à l'ENSEIRB-MATMECA. Nous sommes encadrés par le responsable pédagogique Antoine Rollet, professeur à l'ENSEIRB-MATMECA.

La NXP Cup oppose plusieurs équipes autour du thème de la voiture autonome. Ces équipes ont en commun la taille de la voiture, une échelle 1/16 imposée par les organisateurs de la compétition. L'enjeu principal de la compétition est de construire une voiture qui puisse suivre une piste inconnue de manière autonome tout en ayant à l'esprit de finir la course en record de temps le plus bas. C'est une compétition internationale qui regroupe des étudiants des quatre coins du globe. Les premiers repartiront avec un prix de 3000 euros. Une première phase nationale aura d'abord lieu, à Paris notamment concernant la France, et permettra de sélectionner les équipes qualifiées pour la finale à Bucarest. Les modifications de la voiture nécessiteront l'application d'outils provenant de différents domaines dont l'informatique, l'électronique et la modélisation 3D.
Ce document détaillera donc les besoins de ce projet et la manière dont nous allons y répondre.

2 Cadre du Projet

2.1 Enjeux et objectifs

Les épreuves se déroulent sur un circuit blanc de largeur constante 55cm qui possède des bordures noires de chaque côté de largeur constante de 2cm (inclus dans les 55cm). Il y a un total de 6 épreuves : seule l'épreuve de course est obligatoire, les autres sont facultatives. Les épreuves facultatives rapportent des points. Ces épreuves seront détaillées dans ce document.

La voiture sera inspectée avant sa participation aux épreuves par les juges. Elle sera ensuite placée dans la zone d'inspection. L'équipe ne peut plus toucher à la voiture sans autorisation préalable d'un arbitre, et dans ce cas les modifications autorisées sont limitées.

Lors de l'intégralité de l'évènement des fautes peuvent être commises par l'équipe, elle peuvent aller de l'ajout de temps additionnel sur les courses à l'élimination de l'équipe dans l'intégralité des épreuves.

Les circuits sont construits à partir d'une liste exhaustive d'éléments de circuits : ligne droite, virage à 90 degrés, 45 degrés, intersection de deux lignes droites à 90 degrés, 45 degrés, un élément bosse, un élément zig-zag, un élément tunnel.

Notre objectif est de réaliser la course et toutes les épreuves facultatives.

2.1.1 Course

C'est l'épreuve qui rapporte le plus de points, les 10 premiers gagnent des points de 650 à 100 points. L'ordre de passage est aléatoire et annoncé quelques minutes avant la course. Tout le monde participe sur le même circuit inconnu à l'avance. Lorsqu'une équipe est appelée, un membre peut prendre la voiture de la zone d'inspection pour l'ajuster pendant 2 minutes. Il peut exclusivement : utiliser les boutons, changer l'angle de la caméra, changer la batterie, nettoyer les roues. Il place ensuite la voiture avant la ligne de départ, une fois que les juges confirment le "Ready" la voiture a 30 secondes pour dépasser la ligne de départ. Le temps est enregistré à partir du moment où la voiture franchit la ligne de départ.

Chaque équipe a 3 essais. Le premier essai valide est celui qui sera gardé. Un essai est valide si la voiture arrive à franchir la ligne d'arrivée sans commettre de fautes disqualifiant l'essai. Après chaque essai, le membre de l'équipe peut à nouveau l'ajuster pendant 2 minutes.

2.1.2 Vitesse sur un 8

Cette épreuve rapporte des points en fonction du classement de 200 à 10 du premier au dixième. Le but est de réaliser sur le circuit de la figure 1 le plus de tours en 60 secondes. Le système d'essais est le même que pour la course de vitesse, sauf que le délai de préparation est de 30 secondes.

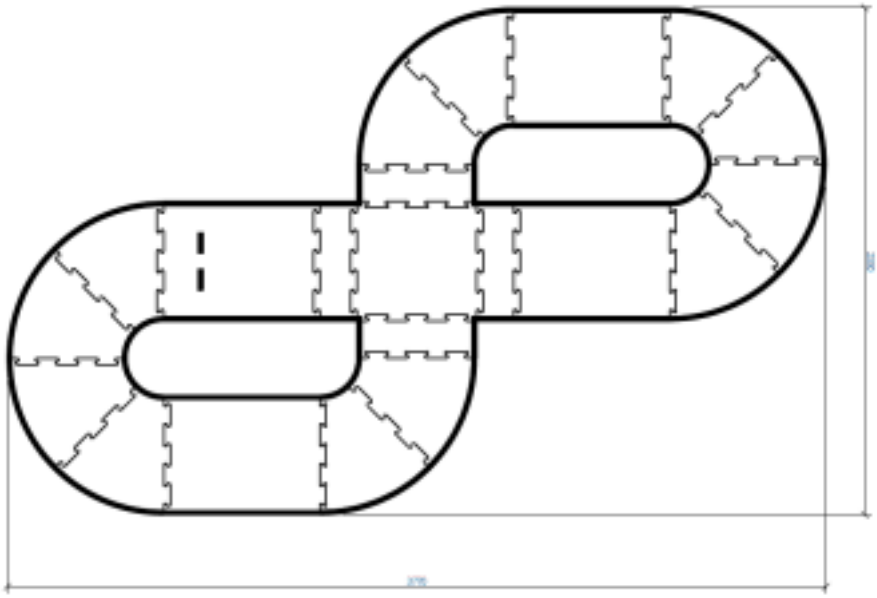

FIGURE 1 – Circuit utilisé pour la course de vitesse sur le 8

2.1.3 Freinage d'urgence

Cette épreuve rapporte 150 points si elle est réussie. Elle se déroule sur un petit circuit comme sur la figure 3. La voiture est positionnée sur une ligne droite, elle doit effectuer au moins 2 tours et prendre de la vitesse. Au même moment que la voiture roule, un membre du jury va poser un bloc, en noir sur le circuit de la figure 2, bloquant l'intégralité de la largeur du circuit. La voiture doit s'arrêter sans toucher le bloc. Un seul essai possible.

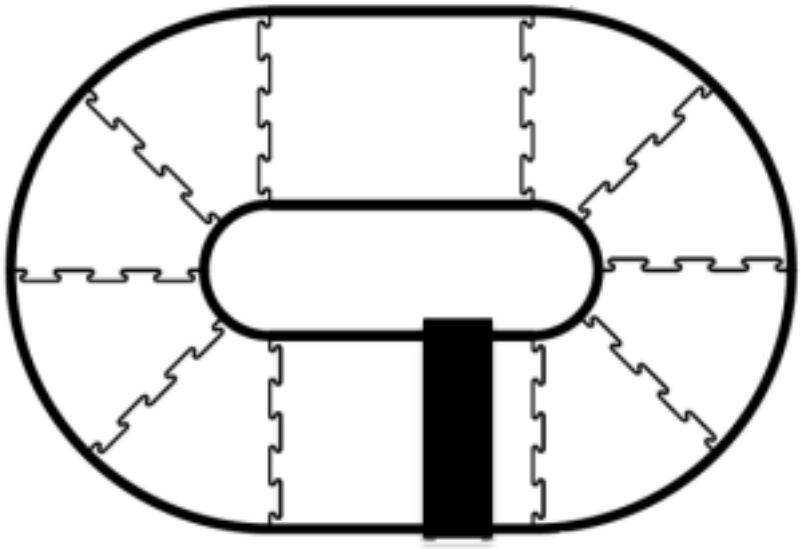

FIGURE 2 – Circuit utilisé pour le freinage d'urgence

2.1.4 Éviter un obstacle

Cette épreuve rapporte 150 points si elle est réussie. Elle se déroule sur un petit circuit comme celui de la figure 3. La voiture est positionnée sur une ligne droite. Un premier tour sera effectué sur le circuit sans obstacle, puis le jury placera l'obstacle sur une ligne droite. L'obstacle est un cube 20x20x20cm. Seul 1 essai est autorisé. Aucune partie de la voiture ne doit toucher l'obstacle. Il convient de préciser que la voiture doit passer le niveau où l'obstacle se situe sans s'arrêter, ainsi simplement s'arrêter face à l'obstacle ne compte pas comme un essai réussi. Le circuit illustré dans la figure 3 est un exemple.

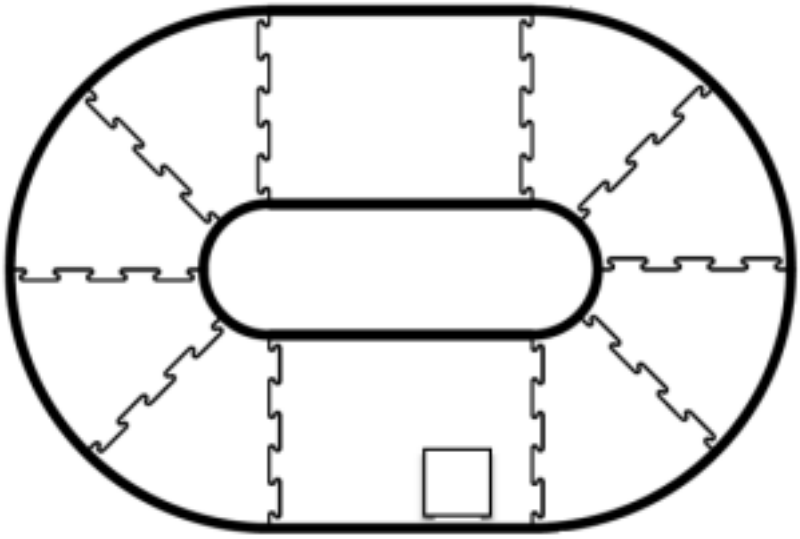

FIGURE 3 – Circuit avec un obstacle

2.1.5 Contrôle de la vitesse

Cette épreuve rapporte 150 points si elle est réussie. Elle se déroule sur un petit circuit similaire à celui de la figure 3. Seul 1 essai est autorisé. La voiture roule à une vitesse constante sur le circuit, puis lorsqu'elle détecte une marque au sol : trois traits comme la ligne de départ, elle entre dans une zone basse vitesse. La voiture doit donc réduire sa vitesse de moitié tant qu'elle est dans la zone puis reprendre sa vitesse normale à la fin de la zone.

2.1.6 Fautes

Des fautes peuvent êtres commises par l'équipe lords de la compétition, il existe différents degrés de fautes.

2.1.7 Pendant l'épreuve obligatoire

Ces fautes s'appliquent pendant l'épreuve principale de la course.
Les actions suivantes ajouteront une pénalité de temps mais n'invalident pas l'essai :

- La voiture ne démarre pas dans les 30 secondes. [+1 seconde].
- La voiture ne s'arrête pas dans les 2 mètres qui suivent la ligne d'arrivée [+1 seconde]

— la voiture sort du circuit après avoir franchi la ligne d'arrivée [+1 seconde]

Les actions suivantes rendront l'essai comme nul :

— 3 roues ou plus quittent le circuit avant de franchir la ligne d'arrivée
— L'équipe n'est pas prête dans le temps imparti
— Le membre de l'équipe touche la voiture sans consentement de l'arbitre.
— La voiture ne franchit pas la ligne d'arrivée dans les deux minutes suivant son départ.
— Le membre de l'équipe touche la voiture pendant la course.

2.1.8 Pendant les épreuves facultatives

Ces fautes s'appliquent pendant toutes les épreuves sauf celle de la course.

Les actions suivantes rendront l'essai comme nul :

— La voiture ne démarre pas dans les 30 secondes.
— La voiture sort du circuit
— Une partie quelconque des roues ne touche plus le sol
— L'équipe n'est pas prête dans le temps imparti

Les actions suivantes rendront tout les scores comme nuls, l'équipe est éliminée :

— Un équipement qui détériore les autres voitures ou le circuit
— Pas de Team Log Book (décrit plus tard)
— Modifications non autorisées après l'inspection
— Plus d'une personne sur le circuit
— Triche
— Inspection technique ratée, c'est à dire non respect du cadre technique imposé expliqué ci-après.

2.2 Cadre technique

La NXP Cup possède néanmoins certaines contraintes techniques au niveau du matériel utilisé pour réaliser la voiture, une marge de liberté est toutefois disponible.

— MPU et MCUs NXP seulement, sans limite de nombre
— Pas plus de 4 roues
— Pas plus de 2 moteurs avec ou sans broches
— La vision doit se faire principalement à l'aide d'une caméra (consigne officielle aussi vague), d'autres capteurs peuvent être ajoutés
— Des capteurs additionnels peuvent être ajoutés afin d'améliorer la navigation, si un modèle NXP existe il doit être utilisé.
— La voiture doit être autonome et ne doit pas être contrôlée à distance.
— Toute connexion sans fil est interdite

— L'utilisation d'un ESC (Electronical Speed Control) est autorisée
— Batterie : une seule, rechargeable NiCd, NiMH or Li-ION avec au maximum 5300mAh ou LiPo mais limité aux modèles 2s 7.4 avec au maximum 5500mAh (Attention au voyage en avion avec des batteries LiPo -¿ compliqué)
— Un logbook (Team log book) doit être fourni, il contient pour chaque épreuve :
  — Chassis, taille, informations sur la/les batterie(s) et sur le(s) moteur(s)
  — Les informations sur l'électronique avec les numéros de référence
  — Schémas électriques des MPU et MCU si ils ne sont pas de NXP
  — La facture du matériel
  — Les paramètres spécifiques pour l'électronique
— Pour différentes épreuves, on peut changer l'électronique : ajouter/enlever des capteurs et/ou des cartes mais cela doit se faire dans le temps imparti.

3 Spécifications fonctionnelles

Les besoins fonctionnels pour ce projet peuvent être répartis sur deux phases. La première constitue en quelque sorte le MVP, le produit minimum viable. Ce sont toutes les fonctionnalités de base nécessaires pour la mise en place de la voiture autonome. La seconde étape quant à elle constitue les fonctionnalités poussées qui vont venir améliorer la voiture autonome.

3.1 Phase Opérationnelle : Suivre simplement le circuit

3.1.1 Réaliser une course sur circuit inconnu

— **La voiture se localise** La voiture doit être capable de se localiser sur la piste par rapport aux lignes situées sur les bords de pistes.
— **Démarrer dans les 30 premières secondes après que la voiture soit déposée sur le circuit** Pour la NXP Cup il est imposé que les voitures franchissent la ligne de départ dans les 30 secondes qui suivent le lancement du programme par l'utilisateur grâce à un bouton connecté au processeur de la voiture. Il est donc indispensable que notre voiture prenne en compte cette contrainte dans le programme. Connaissant à l'avance à quelle distance d la voiture est placée par rapport à la ligne de départ, il suffit simplement de lui fournir une vitesse initiale v_i de sorte de le rapport de d par v soit inférieur à 30s.

— **Suivre une ligne droite** Les pistes de la NXP Cup sont des bandes blanches avec des bandes noires sur les côtés spécifiant la fin de la piste. Étant donné déjà

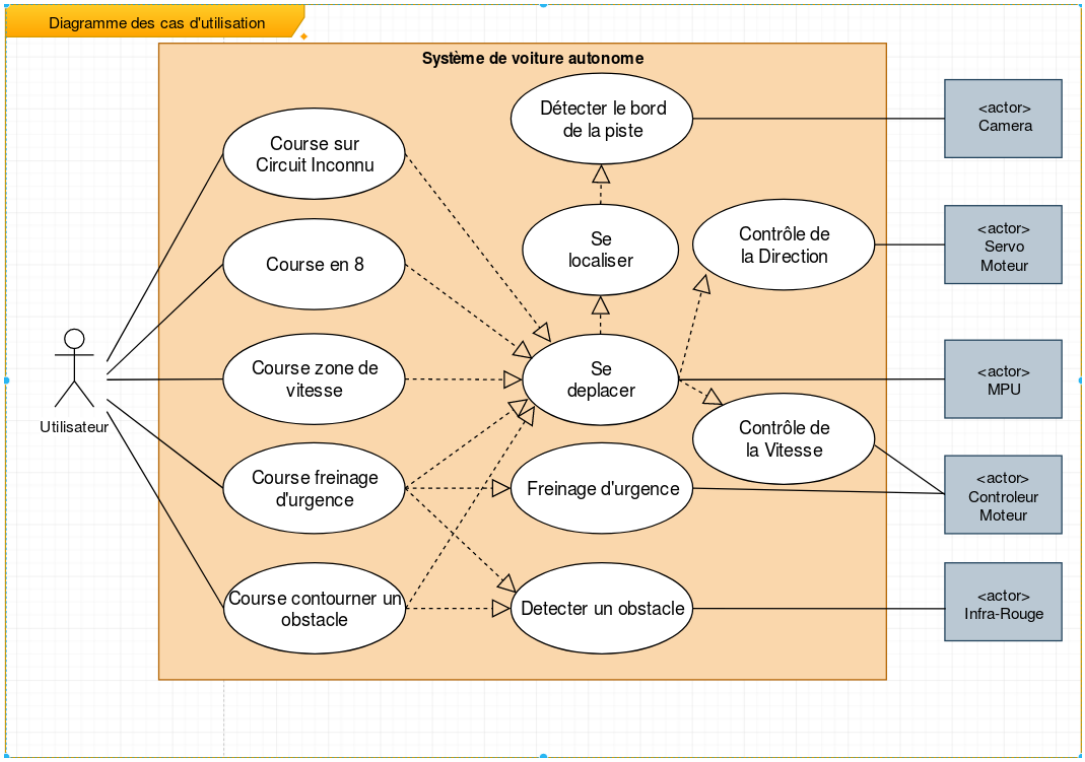

FIGURE 4 – Diagramme des cas d'utilisations

une ligne droite, la voiture doit être capable de rouler correctement sans sortir de la piste. Cela impose qu'elle puisse détecter les bandes noires sur le circuit avec la caméra et ainsi se positionner au centre de la piste en changeant sa direction si besoin. La détection doit se faire avec des capteurs (au minimum une caméra) installés sur la voiture puisque la piste n'est pas connue à l'avance. Sur une ligne droite on peut choisir de changer la vitesse (en l'augmentant par exemple).

— **Emprunter les virages sans sortir de la piste** Les voitures ne circuleront pas que sur des pistes droites, il est à prévoir de nombreux virages. La voiture doit être en mesure de détecter un virage et modifier la direction de ses roues. Une réduction de la vitesse à l'approche ou dans un virage est à envisager. Dans une situation de virage, la caméra fournie naturellement de vecteurs parallèles mais déviés par rapport à la normale. La valeur de cette déviation constitue justement la valeur de l'angle à envoyer au servomoteur pour changer la direction de la voiture.

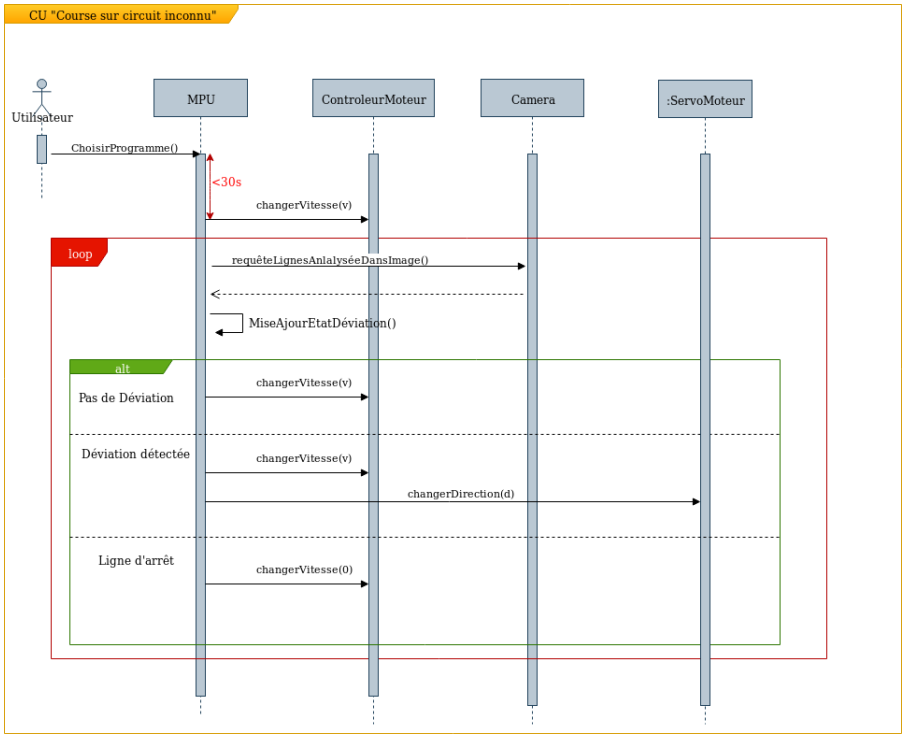

FIGURE 5 – Diagramme de séquence pour le cas d'utilisation "Réaliser la course du circuit inconnu"

— **S'arrêter dans les 2 mètres après la ligne d'arrivée** Une fois la ligne d'arrivée détectée, la voiture doit pouvoir s'arrêter obligatoirement dans les deux mètres qui suivent. Pour cela il faut que pouvoir détecter la ligne d'arrivée avec la caméra. La ligne d'arrivée est connue à l'avance. Il s'agit d'une bande noire détectable donc avec la caméra comme la détection des bords. Une fois la ligne dépassée il faut arrêter la voiture en envoyant la valeur 0 au contrôleur ce qui stoppe immédiatement la voiture.
— **Le système doit pouvoir détecter l'élément de circuit bosse** Une bosse de hauteur supérieur à vingt centimètres peut faire partie du circuit. Nous devons pouvoir détecter cela afin de ne pas sortir du circuit. Pour cela, nous utiliserons le capteur infrarouge qui détectera la bosse qui sera suffisamment haute.
— **Gérer plusieurs programmes** Un programme correspond à une épreuve, il doit être possible de détecter quel programme a été choisi par l'utilisateur par l'intermédiaire de boutons sur la voiture.

3.1.2 Effectuer la course sur un 8

Étant donné que la forme de la piste et ses dimensions sont connues à l'avance, ces données sont à intégrer dans le code pour améliorer les performances. On utilisera donc la caméra avec une fréquence faible par rapport aux autres courses.

3.1.3 Effectuer un freinage d'urgence

Il s'agit de faire, dans cette épreuve, deux tours obligatoires. Le premier est une tournée régulière. Cependant dans le second, un obstacle sera placé au milieu de la piste et tout l'objectif est que la voiture détecte cet obstacle et s'arrête tout juste avant de le percuter. La distance d'arrêt par rapport à l'obstacle n'est pas imposée. La vitesse n'a pas d'importance dans cette épreuve.

- **Détecter un obstacle : distance sur la piste**
  Un obstacle noir occupant l'intégralité de la largeur de la piste sera posé pendant que la voiture roule sur le circuit. L'obstacle sera posé sur une ligne droite. Nous devons à l'aide du capteur infrarouge détecté la présence de l'obstacle et sa distance par rapport à la voiture.
- **S'arrêter lorsqu'un obstacle bloque la route sur toute sa largeur** On envoie simplement la commande au moteur d'aller à une vitesse nulle.

3.1.4 Éviter un obstacle

.

- **Détecter un obstacle : sa distance par rapport à la voiture et sa largeur sur la piste** La détection d'obstacle utilisera le même principe que celle dans l'épreuve de de freinage. La différence cependant est que l'obstacle ne couvre pas la totalité de la largeur de la piste. Donc en plus de détecter la distance de la voiture par rapport à l'obstacle il faut aussi déterminer la largeur de l'obstacle et ainsi dévier la voiture pour éviter l'obstacle. Pour un obstacle qui ne couvre pas la largeur de la piste, le vecteur que fournira le capteur infra rouge sera constitué d'une section de distance par rapport à l'obstacle, et d'une section remplie de valeurs 'infinies' (maximale dans le domaine de valeurs) qui constitue la zone de la piste sans obstacle.
- **Contourner un obstacle** On connaît la partie de la piste occupée par l'obstacle grâce à la détection précédente. Pour contourner l'obstacle on projette comme les tentacules d'une pieuvre les trajectoires disponibles pour la voiture. Grâce aux données fournies par la caméra on peut éliminer l'ensemble des trajectoires menant à une collision, il ne nous reste plus qu'à prendre une trajectoire qui

fonctionne.

3.1.5 Zone de vitesse

— **Le système doit être capable de limiter sa vitesse.** Dans l'épreuve contrôler sa vitesse il faut dans une certaine zone limiter notre vitesse à la moitié de la vitesse avant l'arrivée sur zone.
— **Le système doit être capable de détecter un zone de limite de vitesse.** La zone de limite de vitesse est une zone identique à la ligne d'arrivée elle sera donc reconnue de la même manière.

3.2 Phase Avancée : SLAM et Calcul de Trajectoire optimale

La stratégie que nous allons aborder durant la course sera la suivante : Lors du premier essai, la voiture roulera à une allure modérée afin que l'on puisse enregistrer les données cartographiques à l'aide de la caméra. Une fois la ligne d'arrivée détectée notre voiture fera une sortie de route avant de franchir la ligne afin que le premier essai ne soit pas pris en compte par les jurys de la NXP Cup. Nous préciserons cette sortie par la suite. Nous serons donc amenés à faire un second essai en utilisant les données préalablement enregistrées. Ces données seront utilisées directement pour calculer la meilleure trajectoire (ce point sera décrit dans la partie suivante). On utilisera alors le flux d'informations des capteurs pour s'approcher au mieux de la trajectoire. Ainsi, les trajectoires pré-enregistrées nous permettront de gagner du temps lors du second essai en anticipant les actions.

3.2.1 Premier essai

— **Le système doit être en mesure de cartographier une piste inconnue.** L'une des plus grandes parties de notre projet concerne l'obtention d'informations à partir de la piste en utilisant obligatoirement une caméra et, si nous le souhaitons, d'autres capteurs. Pour cela, un de nos besoins fonctionnels (décrit précédemment) est de lire l'image courante de la caméra et d'extraire les vecteurs correspondant aux bords.
— **Le système doit être en mesure de garder en mémoire les données cartographiques.** Les cartes embarquées utilisées disposent d'un espace mémoire limité. Les données seront donc stockées en utilisant un système de liste d'entiers représentant les fragments de pistes. En effet chaque section du circuit est normalisée et est représentée par un entier. Finalement le système reconstitue le circuit à l'aide de ces entiers et de leurs sections liées.

— **Le système doit pouvoir détecter la ligne d'arrivée et faire une sortie de piste.** Une fois la ligne d'arrivée détectée le système doit être conscient que ce premier essai permet la cartographie de la piste, il devra donc faire sortir la voiture de la piste avant de franchir la ligne pour invalider le premier essai. Pour ce faire un interrupteur est présent sur la voiture afin de choisir le bon programme. Ainsi nous utilisons cet interrupteur entre les deux essais pour lancer le calcul de trajectoire et le second essai. Il aurait été possible de laisser la voiture sur un unique programme procédant successivement aux deux essais mais cela permet d'ajouter du contrôle et de la sécurisation sur le processus. Nous possédons avec la caméra la direction de la piste il suffit d'aller à 90 degrés à droite de la direction de la piste jusqu'à ce qu'on ne détecte plus de piste après quoi la voiture s'arrête.

3.2.2 Fonction de calcul de trajectoires

— **Le système calcul des trajectoires pour améliorer ses performances lors du deuxième essai.** Une fois la carte créée mais avant le second essai, le système doit faire tourner un algorithme de calcul de trajectoires. Pour cela, les capacités de la voiture sont analysées pour chaque fragment de circuit possible puis dans la mesure du possible, une trajectoire utilisant la meilleure solution pour chaque fragment en fonction de celui précédent et suivant est produite.
— **Le système dispose d'un temps limite pour le calcul.** Le programme doit tourner et calculer la trajectoire sur la carte définie par le SLAM en moins de 1min30 (temps disponible entre les deux courses).
— **Le système doit permettre de savoir quand le calcul est achevé.** Le système peut utiliser un système de voyants sur la voiture.
— **Le système utilise la trajectoire.** Le système va ensuite utiliser la trajectoire obtenue pour procéder au second essai en s'approchant le plus possible de cette trajectoire.

3.2.3 Deuxième essai

— **Le système doit pouvoir récupérer des données pour adapter sa trajectoire.** Lors du second essai, la voiture dispose et récupère un paquet de données pour procéder au meilleur temps possible. Le système repère les points caractéristiques (début ou fin de virages, intersections etc), se situe sur les fragments de la carte créée et récupère les données de vitesse et donc de distance parcourue.
— **Le système progresse vers la victoire** La voiture adapte sa trajectoire et sa vitesse à l'aide de ces données. L'idée est d'utiliser la carte pour se permettre

des vitesses excessives par rapport à une découverte en temps réel (et donc par rapport aux concurrents utilisant cette méthode).

4 Tests de validations

4.1 Phase Opérationnelle

— L'utilisation d'un système de log permettant d'afficher les vecteurs détectés par la caméra lorsqu'on lui fait parcourir des éléments de circuits vont permettre de vérifier que nous obtenons les bords du circuit correctement.
— L'utilisation d'un système de logs permettant d'afficher les données fournies par le capteur infrarouge lorsqu'il détecte les éléments suivants : un cube de 20x20x20cm, un bloc noir d'au moins 55cm de largeur, un élément bosse du circuit et rien, permettront de vérifier la bonne détection de ces éléments.
— La voiture doit pouvoir rouler sans sortir du circuit sur un circuit quelconque. Nous disposerons d'un circuit fournit par le client qui contient a priori tous les différents blocs de circuits.
— Chaque module programmé sera testé sur ordinateur en fournissant une simulation (des données fictives en entrée) et en observant les retours du module.

4.2 Phase Avancée

— On fait parcourir à la main le circuit par la caméra, puis on vérifie que la représentation du circuit construite par l'application correspond au circuit réel fournit par le client.
— La voiture doit pouvoir rouler plus vite au second tour que dans la phase opérationnelle. Nous devons donc comparer les temps sur le circuit fournit par le client.
— On vérifie la pertinence de nos algorithmes de calcul de trajectoire grâce à un simulateur de circuit. On observera alors la pertinence des trajectoires.

5 Spécifications non fonctionnelles (ou techniques)

Le but de cette section est de lister de manière exhaustive l'ensemble des matériels que nous utilisons et pensons utiliser pour le développement complet de notre prototype de voiture, tout en respectant les contraintes du Cadre Technique de la compétition défini en 2.3.

Concernant les châssis des voitures, nous allons travailler avec deux modèles principalement, tous deux proposés par DFR, et qui ont été imposés par le client :

— le DFROBOT ROB0157 que nous aurons en un seul exemplaire
— le DFROBOT ROB0165 que nous aurons en deux exemplaires

Les informations concernant le premier sont disponibles ici :
`https://www.mouser.co.uk/new/dfrobot/dfrobot-rob0165-brushless-motor-racing-car/`,
et celles concernant le second ici :
`https://www.mouser.co.uk/new/dfrobot/dfrobot-rob0157-brush-motor-racing-car/`

Un troisième modèle de type véritable voiture de modélisme sera utilisé. Le client nous a demandé de lui proposer un modèle dont l'échelle serait la même que celle réglementaire, c'est à dire 1/16. Nous avons donc trouvé un modèle qui vérifiait en plus de l'échelle les caractéristiques demandés par le client, soit un moteur brushless, une batterie qui ne pose pas de problème dans les aéroports en vue d'un potentiel trajet vers le lieu d'une compétition (en l'occurence une NiMH), et quatre roues motrices. C'est le modèle TrojanPro de chez HSP, à propos duquel des informations sont disponibles ici :
`https://modelisme-rc.net/voiture-rc-electrique-1-16-modelisme-rc/295-hsp-trojan-pro-b`
`html`.

À propos des cartes de programmation maintenant :

Nous débuterons notre projet avec une Teensy4.0 dont une bonne documentation est disponible avec ce lien : `https://www.pjrc.com/store/teensy40.html`. Ce support de programmation type Arduino nous a été imposé par le client, mais en prévision d'une nécessité de performances de calcul plus élevées, le client nous a également imposé de faire la compétition avec une carte plus puissante. A ce jour, il ne nous a pas encore précisé laquelle ce serait.

Au niveau des capteurs, nous utiliserons jusqu'au bout du projet une caméra Pixy2. Toutes les informations et documentations de Pixy2 sont disponibles sur : `https://pixycam.com/pixy2/`.

Nous utiliserons également un LIDAR afin de détecter aussi bien l'obstacle présent sur la piste mais aussi la bosse lors des épreuves associées. Le LIDAR de référénce 114991434 de la marque Seeed Studio convient parfaitement.

Des modifications sur le châssis sont à prévoir et à réaliser à l'imprimante 3D. Elles serviront par exemple à fixer les éléments additionnels au châssis, comme les capteurs et les cartes.

Nous disposons aussi d'un circuit imprimé conforme aux règles de la NXP Cup ainsi que des pièces de circuits joignables très similaires à ceux de la NXP Cup.

Toutes les cartes supportent le C, donc le code à produire sera en langage C (standard c99, celui utilisé usuellement par l'ensemble du groupe).

6 Livrables

Nous définissions les livrables suivant :
— Voitures basiques opérationnelles (la voiture roule) : Vendredi 20 décembre 2019
— Document individuel de synthèse : Vendredi 20 décembre 2019
— Voitures prêtes pour les épreuves facultatives : 17 février 2020
— Voitures prêtes pour la compétition : 8 mars 2020
— Rapport : Vendredi 10 avril 2020
— Soutenance : Lundi 20 avril 2020

7 Planning prévisionnel

Le planning prévisionnel a été développé dans le document `planning.pdf`. Voici les tâches et les dates clés :

Tache	Date début	Date fin
Module Vitesse	25/11/2019	08/12/2019
Adapter Modules pour différents modèles	09/12/2019	22/12/2019
Freinage	09/12/2019	15/12/2019
Module Camera	25/11/2019	08/12/2019
Module LIDAR	25/11/2019	15/12/2019
Détection obstacle	16/12/2019	19/01/2020
Gestion MultiProgramme	25/11/2019	01/12/2019
SLAM	09/12/2019	22/01/2020
Calcul trajectoire	23/01/2020	07/03/2020
Course 8	06/01/2020	19/01/2020
Freinage Urgence	20/01/2020	02/02/2020
Eviter Obstacles	20/01/2020	16/02/2020
Contrôle de la vitesse	06/01/2020	26/01/2020
Course de vitesse	08/03/2020	08/03/2020
Phase de retour pour les épreuves	17/02/2020	08/03/2020
Redaction du document personnel de synthèse	07/12/2019	20/12/2019