

Département : Technologies industrielles (TIN)

Filière : Génie électrique

Orientation : Electronique et Automatisation industrielle (EAI)

Année d’étude : 2019-2020

Yverdon-les-Bains, le 24.07.2020

Par

Maxime CHARRIÈRE

Professeur responsable

Pierre BRESSY

Travail de Bachelor

Voiture RC pilotée par IA

Préambule

Ce travail de Bachelor (ci-après TB) est réalisé en fin de cursus d’études, en vue de l’obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'Ecole.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d’auteur.

HEIG-VD

Le Chef du Département

Yverdon-les-Bains, le 24.07.2020

Résumé

*Find more on*



[github.com/maximecharriere/AutonomousRcCar](https://github.com/maximecharriere/AutonomousRcCar)

Table des matières

[1. Glossaire 6](#_Toc46056948)

[2. introduction générale 7](#_Toc46056949)

[2.1. Plan du document 7](#_Toc46056950)

[2.2. Une voiture pilotée par IA, c’est quoi ? 7](#_Toc46056951)

[3. Pré-étude 8](#_Toc46056952)

[3.1. Choix de la voiture 8](#_Toc46056953)

[3.1.1. Caractéristiques techniques 9](#_Toc46056954)

[3.2. Choix de l’ordinateur de bord 10](#_Toc46056955)

[3.2.1. Calcul interne ou streaming 10](#_Toc46056956)

[3.2.2. Choix du SBC 10](#_Toc46056957)

[3.2.3. Refroidissement 14](#_Toc46056958)

[3.3. Choix des capteurs 15](#_Toc46056959)

[3.3.1. Détection de route 15](#_Toc46056960)

[3.3.2. Détection de panneaux 15](#_Toc46056961)

[3.3.3. Détection d’obstacles 15](#_Toc46056962)

[3.4. Choix du système d’alimentation 17](#_Toc46056963)

[3.4.1. PiJuice 17](#_Toc46056964)

[3.4.2. Servo et moteur 18](#_Toc46056965)

[3.5. Choix du système de communication 19](#_Toc46056966)

[3.6. Choix du langage de programmation 19](#_Toc46056967)

[4. Conception mécanique 20](#_Toc46056968)

[4.1. Voiture 20](#_Toc46056969)

[4.2. Route 21](#_Toc46056970)

[4.3. Signalisation 22](#_Toc46056971)

[5. Conception électronique 23](#_Toc46056972)

[5.1. Electronique de la voiture 23](#_Toc46056973)

[5.2. Electronique du feu bicolor 25](#_Toc46056974)

[6. Installation RPi 26](#_Toc46056975)

[7. Architecture logicielle 27](#_Toc46056976)

[8. Suivit de ligne 28](#_Toc46056977)

[9. Detection de panneaux 29](#_Toc46056978)

[9.1.1. MobilNet 29](#_Toc46056979)

[9.1.2. TensorfFlow vs Tensorflow Lite 29](#_Toc46056980)

[10. Detection d’obstacles 30](#_Toc46056981)

[11. Problèmes rencontrés 31](#_Toc46056982)

[11.1. PWM pas correcte 31](#_Toc46056983)

[11.2. Le RPi s’arrête 31](#_Toc46056984)

[12. Améliorations 32](#_Toc46056985)

[12.1. Plaque de support du RPi 32](#_Toc46056986)

[13. Planning 33](#_Toc46056987)

[14. Remerciements 34](#_Toc46056988)

[15. Bibliographie 35](#_Toc46056989)

[16. non-plagiat 36](#_Toc46056990)

[17. Annexes 37](#_Toc46056991)

[17.1. Énoncé du TB 37](#_Toc46056992)

[17.2. Clause de confidentialité 37](#_Toc46056993)

# Glossaire

|  |  |
| --- | --- |
| **TB** | Travail de Bachelor |
| **RPi** | Raspberry Pi |
| **SBC** | Single-board computer (ici l’ordinateur de bord) |
| **RC** | Radio commandé |
| **PWM** | Pulse Width Modulation |
| **BT** | BlueTooth |
| **ML** | Machine Learning |
| **CPU** | Central Processing Unit |
| **GPU** | Graphic Processing Unit |
| **NPU** | Network Processing Unit |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# introduction générale

## Plan du document

## Une voiture pilotée par IA, c’est quoi ?

Doit être indépendant d’aide externe -> pas de GPS

# Pré-étude

## Choix de la voiture

Malgré le fait que le choix d’une voiture RC soit spécifié dans l’énoncé du TB (17.1), une étude des choix possibles a été effectué.

Il est possible :

* De construire sa propre voiture miniature
* D’utiliser une vraie voiture
* D’utiliser un karting
* D’acheter une voiture RC déjà fonctionnelle

Compte tenu des raisons évidente de sécurité et de moyen, l’utilisation d’une vraie voiture ou d’un karting est impossible.

Il serait idéal de confectionner soi-même la voiture afin de métriser les caractéristiques de chaque composant, mais le TB ne portant pas sur ce sujet et une telle conception prenant beaucoup de temps, il a été décider d’acquérir une voiture déjà montée.

Afin de choisir une voiture RC, les points suivants ont été pris en compte :

* Prix (-300fr)
* Conception simple à comprendre et modifier (pour commander le guidage et le moteur par un SBC et non par radiofréquences)
* Batterie (autonomie et pouvoir éventuellement alimenter le SBC) / recharge facile
* Taille (pour y installer batterie, SBC et capteurs)

Après recherche de ce qu’il existait sur le marché, il a été choisi d’acheter la voiture :

* T2M Mad Pirate Brushless 1/10e

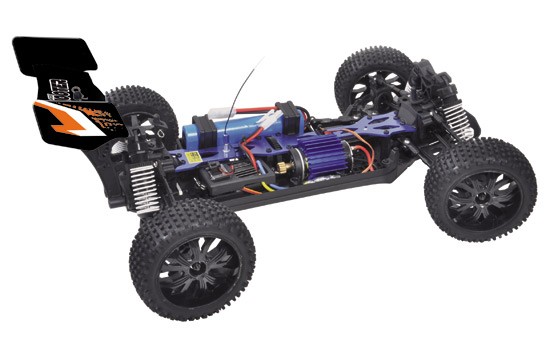


Figure 1 T2M Mad Pirate Brushless 1/10e

### Caractéristiques techniques

Il a été primordial de pouvoir guider la voiture simplement, et sur ce type de voiture l’accélération et le guidage se fait grace à un simple PWM.

De base, la manette envoie à la voiture la position de ses potentiomètre de direction et gaz par onde radio, et un récepteur radio se trouvant dans la voiture converti ces informations en PWM qui commande un servo-moteur pour la direction et un controleur de moteur brushless pour l’accélération.

Après plusieurs mesures effectuées à l’oscilloscope sur les sorties PWM du récepteur radio, les caractéristiques suivantes ont été montrés :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Freq** | **Duty cycle** | | |
|  | **Low** | **Middle** | **High** |
| **Steering** | | 50Hz | 1.2ms (Left) (6%) | 1.5ms (7.5%) | 1.8ms (Right) (9%) |
| **Speed** | | 50Hz | 1ms (Back) (5%) | 1.5ms (7.5%) | 2ms (Forward) (10%) |

Tableau 1 Mesures PWM des actuateurs

La voiture est alimentée par une batterie :

* NiMh 7.2V 1800mAh

##### Arrêt d’urgence de la voiture

Le contrôleur du moteur a une fonction spéciale d’arrêt d’urgence de la voiture.

Effectivement, si le duty-cycle correspondant à la position arrêté de la voiture (1.5ms) est appliqué, le moteur ne sera simplement plus alimenté et la voiture continuera à avancer grâce à son énergie synectique.

Le contrôleur du moteur comporte donc un système de frein. Ce frein s’active si un duty-cycle plus petit que la position neutre (ici ) est directement appliqué alors que la voiture avançait.

Alors que pour aller en marche arrière, une pause (temps non spécifié) doit être fait en position neutre avant d’appliquer le duty-cycle désiré servant à reculer. (1)

## Choix de l’ordinateur de bord

Etant donné que le but du projet est d’agir sur le système de guidage de la voiture en fonction de son environnement, il faut que la voiture comporte des capteurs, des actuateurs ainsi qu’un ordinateur de bord (SBC) afin de prendre les décisions.

### Calcul interne ou streaming

Il est possible d’effectuer le traitement des informations et la prise de décision directement avec le SBC se trouvant dans la voiture, mais une autre solution très intéressante doit être prise en compte.

Nous pourrions uniquement acquérir les données des différents capteurs sur la voiture, puis envoyer par onde électromagnétique (WiFi / Radio / Bluetooth) ses informations à une machine externe plus puissante, qui exécutera les différents algorithmes et renverra uniquement l’angle des roues et la vitesse à appliquer à la voiture.

Grace à cette technique dite « de streaming », le SBC à bord de la voiture pourra être très rudimentaire étant donné qu’il ne devra seulement lire et écrire les entrées/sorties.

L’avantage est qu’un tel SBC consommera très peu (autonomie de la voiture accru), et la puissance de calcul de la machine externe peut être très grande et est facilement adaptable.

De plus, la rapidité de transfert par onde est de nos jours bien assez grande pour transmettre les quelques informations et images que nous aurons à traiter.

Cependant, un tel système signifie que la voiture ne peut pas fonctionner sans être connecté à la machine externe, ce qui ne la rend plus intégralement autonome. De plus une perte de connexion peut vite arriver. Cette solution n’a donc pas été retenu.

### Choix du SBC

Le SBC doit donc pouvoir gérer la lecture des différents capteurs, commander les actionneurs et exécuter l’intégralité du code de traitement. Il serait préférable que celui-ci ai aussi une carte réseau afin de pouvoir le programmer à distance sans devoir si connecter par câble.

Les points suivants ont donc été pris en compte dans le choix du SBC :

* Puissance de calculs (CPU / GPU / NPU)
* Facilité d’utilisation
* Communauté l’utilisant
* Prix
* Consommation électrique
* Entrée / sorties (PWM hardware, gestion camera)
* Temps de démarrage
* Connection (WiFi / Bluetooth)

##### CPU, GPU, NPU. Quelle différence ?

Les opérations d’exécution d’un algorithme sont effectuées dans l’unité de traitement de l’ordinateur, qui est généralement une CPU. Cependant, certains types de calculs ne sont pas adapter à l’architecture d’une CPU.

Effectivement, une CPU ne pourra exécuter en parallèle qu’un nombre de calcul correspondant à son nombre de cœur, allant généralement de 2 à 8.

Ce type d’architecture est excellente pour exécuter des opérations séquentielles, car la fréquence des CPU peut monter très haut, cependant dans le cas de traitement d’images ou d’exécution d’un réseau de neurone, une GPU ou NPU est plus efficace.

Ces domaines font appels à des millions d’opérations simples pouvant s’exécuter en parallèle, et l’architecture des GPU et NPU est justement optimisé afin de faire du parallélisme.

Les NPU ont encore l’avantage d’accélérer l'usage des réseaux de neurones en implémentant directement dans ses circuits certaines fonctions normalement réalisées par du code machine.

##### Table comparative

* Pour plus d’infos sur les comparatifs des SBC, voir (2) et (3)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Positif** | **Négatif** |
| Raspberry Pi 4B 4Gb | Image de l'en-tête de Raspberry Pi 4 : toutes les infos sur ce nouveau micro-ordinateur ! | 63fr  2 PWM  WiFi / BT  Camera  Forte communauté | Pas de GPU / TPU pour du ML  Pas de refroidisseur |
| Arduino Mega |  | 38fr  15 PWM  Forte communauté  Faible consommation | Pas un processeur mais un microcontrôleur  Pas de WiFi / BT |
| NVIDIA Jetson Nano |  | Comporte une GPU  Camera | 159fr  1 PWM  WiFi / BT en option |
| Google Coral Dev Board | 102110260 | Buy Coral Dev Board | Seeed Studio | Distrelec | Comporte une NPU  3 PWM  WiFi / BT  Camera | 153fr |
| Rock Pi N10 | ROCK PI N10 Model B - RK3399Pro 6GB LPDDR3 32GB eMMC - Seeed Studio | Comporte une NPU  2 PWM  Camera | 148fr  Pas de WiFi / BT |

Tableau 2 Compartif des SBC

L’Arduino a tout de suite été retiré car le multithreading y est impossible, la puissance de calcul est très faible et il y est impossible d’y coder en Python (pour le choix du Python, voir. 3.6).

La carte Nvidia et Rock Pi ont été retiré par leur manque de PWM et de WiFi/BT. Il serait totalement possible d’y ajouter des modules externes afin de remédier à ce problème, mais il était préférable d’avoir tout intégré à la carte de base afin d’éviter d’éventuels problèmes. De plus très peu de tutoriels et support concernant ces cartes ont été trouvé sur internet.

Le problème du RPi est qu’il n’est pas fait pour faire tourner des réseaux de neurones.

Cependant une NPU externe, la « **Google Coral Accelerator** » qui est largement utilisé par la communauté peut résoudre ce problème. Cette puce est la même que celle se trouvant dans le « Google Coral Dev Board » vu dans le précédent comparatif.

Le benchmark suivant montre bien le gain qu’apporte cet accélérateur lorsqu’il est connecté au RPi.

Note : Le benchmark a été effectué sur le réseau de neurone MobilNet V2 utilisant des images de 300x300 pixels (voir 9.1.1)

|  |  |
| --- | --- |
| **Board** | **Time (ms)** |
| Raspberry Pi 3B+ | 654,0 |
| Raspberry Pi 4B | 483,5 |
| NVIDIA Jetson Nano | 309,3 |
| NVIDIA Jetson Nano (optimisation GPU) | 72,3 |
| **Google Coral dev board** | **20,9** |
| Raspberry Pi 4B + Google Coral Accelerator (USB2) | 102,3 |
| **Raspberry Pi 4B + Google Coral Accelerator (USB3)** | **18,2** |

Tableau 3 SBC Benchmarking for Machine Learning

Source : (4)

Le choix entre le « Google Coral dev board » et le duo « Raspberry Pi 4B » + « Google Coral Accelerator » a été fait en **prenant en compte la communauté très présente du coté du RPi**.

Les différents problèmes auxquels nous pouvons faire face avec le RPi ont pour la plupart déjà été discuté sur des forums, et la majorité des tutoriels sont fait sur un RPi.

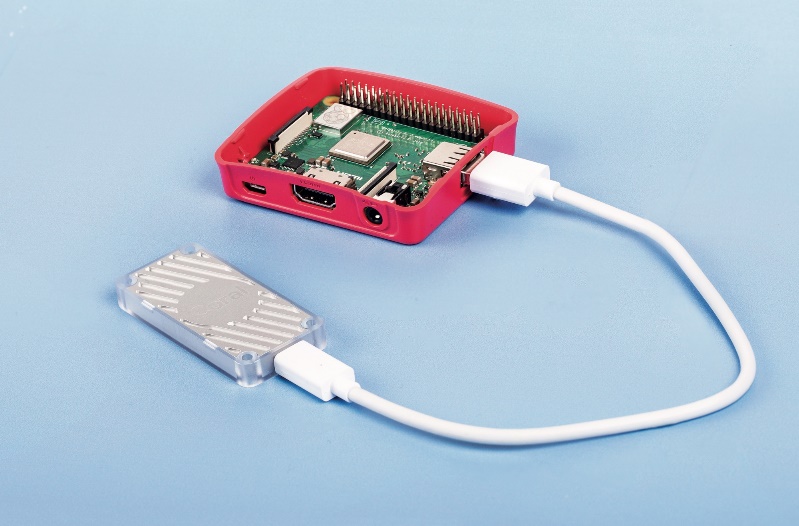


Figure 2 RPi et Coral Accelerator

Il m’a été conseillé de cependant faire attention au temps de démarrage de certains SBC, et le RPi met environ de 7 à 25s à démarrer, ce qui est acceptable. (5)

### Refroidissement

Le RPi ne comporte aucun système de refroidissement, ce qui peut poser problème lors d’une utilisation intensive et prolongée du processeur. De plus, comme nous le verrons au point 0 la batterie est situé juste en dessus du processeur. Par sécurité, un système de refroidissement actif comportant deux ventilateurs a donc été ajouté.

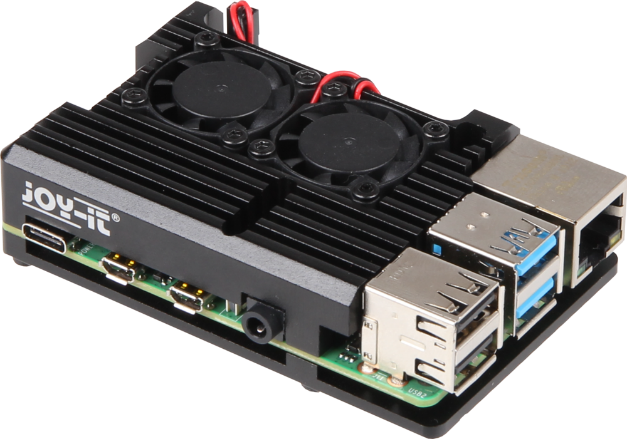


Figure 3 Joy-It Block Active

## Choix des capteurs

La voiture doit répondre à trois problématiques. Chacune a des besoins spécifiques afin d’être résolus. Il sera discuté ici des différents capteurs utilisables pour chaque problématique.

### Détection de route

Afin de détecter la route, deux technologies ont été retenu :

* Une caméra
* Un lidar

Le lidar envoie un laser et mesure le temps que le rayon met à revenir afin de mesurer des distances, dans le but par exemple de constituer un nuage de point représentant son environnement 3D.

Dans notre cas où des lignes peintes doivent être détectés, nous pourrions penser qu’un tel capteur n’est pas utilisable.

Cependant, certains lidars envoient des rayons sur plusieurs longueurs d’ondes (plusieurs couleurs) et mesure l’intensité lumineuse du rebond afin de déterminer les propriétés lumineuses de la surface touchée. (6)

Par contre, un tel système est très lourd et onéreux. **La caméra sera donc retenue.**

### Détection de panneaux

Ici, uniquement l’utilisation d’une **caméra** a été trouvé.

Il serait possible d’intégrer des puces RFID dans les panneaux et de détecter que l’on s’en approche, mais ceci signifierait que l’environnement externe doit s’adapter à la voiture, alors que c’est à la voiture autonome de s’adapter intégralement à son environnement.

### Détection d’obstacles

De nombreuses technologies peuvent être utilisés dans ce but, comme :

* Une **camera**
  + Doit utiliser un algorithme d’analyse d’image, très gourmand en puissance de calcul et en temps
* Un **laser** (modèle considéré : *VL53L0X*)
  + Distance : 3 – 120cm
  + Résolution : 1mm
  + Angle : <3°
* Des **infrarouge** (modèle considéré : *TF Mini LiDAR*)
  + Distance : 30 – 1200cm
  + Résolution : 1cm
  + Angle : 2.3°
* Des **ultrasons** (modèle considéré : *HC-SR04*)
  + Distance : 2 – 400cm
  + Résolution : 3mm
  + Angle : 30°
  + Peut être perturbé par le bruit ambiant
* Un **lidar 360**° (modèle considéré : *RPLIDAR A1*)
  + Permet de détecter des obstacles sur 360°
  + Permet de connaitre précisément la position de l’obstacle, afin de par exemple le contourner
  + Détecte uniquement sur un plan (pas en 3D)
  + Consomme beaucoup de courant
  + Information compliquée à traiter

Etant donné que les obstacles ne sont pas forcément exactement devant le capteur mais doivent être détecté sur toute la largeur de la voiture, l’angle de détection est important.

* Obstacle
* Laser
* Ultrason

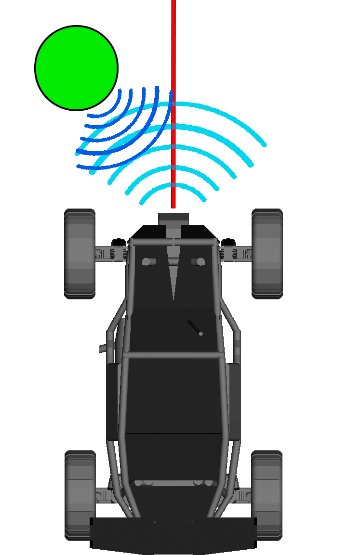


Figure 4 Angle de détection laser / ultrason

C’est pour ceci que le **capteur à ultrason a été choisi**. De plus les objets se trouvant de 5 à 200cm doivent être détecté, ce qui correspond à la distance de détection du capteur.

##### Choix de la camera

## Choix du système d’alimentation

La voiture ne doit pas être connecté à une quelconque source d’alimentation externe. Il faut donc la faire fonctionner sur batteries.

Les composants devant être alimenté sont les suivants et avec la tension spécifiée :

* Le contrôleur du moteur brushless (1)
  + Batterie NiMh 5 à 9 cellules (6 - 10.8v)
  + Batterie LiPo 2 à 3 cellules (7.2 - 11.1v)
* Le servo-moteur de direction
  + 4 – 6V
* Le RPi (celui-ci alimentera ensuite directement la caméra, le capteur à ultrason, le Coral Accelerator ainsi que les ventilateurs de refroidissement)
  + 5V régulé

Le contrôleur du moteur fourni également une tension régulée à 6V.

### PiJuice

Nous pourrions imaginer utiliser uniquement la batterie de la voiture afin de tout alimenter, en utilisant par exemple un régulateur de tension à 5V pour alimenter le RPi.

Il a cependant été préféré d’utiliser un système d’alimentation dédier et spécialement conçu pour le RPi, le **PiJuice HAT** (7) pour les raisons suivantes :

* Pouvoir rendre le PRi indépendant du reste de la voiture afin de le déplacer plus facilement durant la phase de conception
* Pouvoir utiliser le RPi tout en rechargeant les batteries (la batterie de la voiture doit être déconnecté afin de pouvoir la recharger)
* Pouvoir superviser l’état de la batterie depuis le RPi
* Pouvoir déconnecter le câble d’alimentation et passer sur la batterie sans interruption
* Pouvoir arrêter le RPi de façon contrôler et automatique lorsque le niveau de batterie est faible

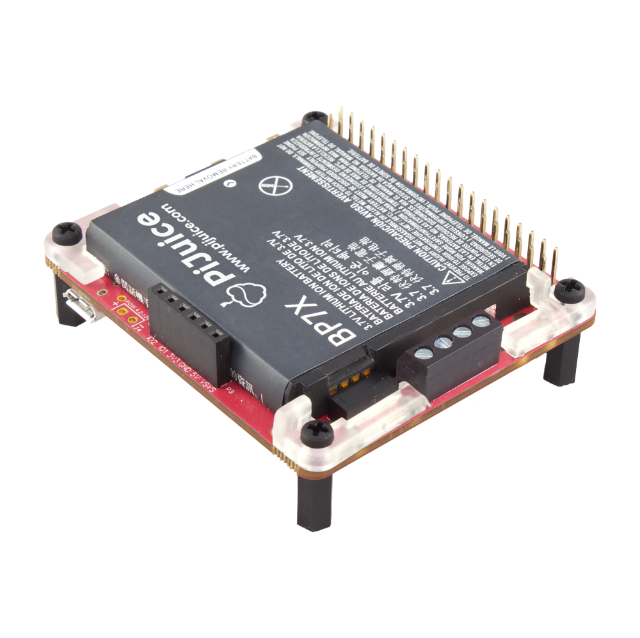


Figure 5 PiJuice HAT

### Servo et moteur

Le servo a tout d’abord été connecté sur le RPi car celui-ci consomme environ 1A et le PiJuice peut théoriquement fournir jusqu’à 2.5A (valeur non atteinte, voir 11.2). Cependant lors d’utilisation intensive du servo, le RPi s’arrêtait, étant sous-alimenté. Le servo a alors été connecté à la sortie VSYS du PiJuice. Cette sortie fournie la puissance restante, garantissant que le minimum nécessaire au bon fonctionnement du RPi soit fourni. Mais dans ce cas le servo fonctionnait moins bien et la batterie se vidait vite.

L’alimentation du servo a alors été fourni par la batterie de la voiture via le contrôleur du moteur fournissant une tension de 6V.

* Schéma électronique détaillé au point 4.

## Choix du système de communication

Etant donné que la voiture doit être autonome, elle ne devrait pas avoir besoin de communiquer avec l’extérieur. Cependant afin de pouvoir programmer la voiture, la débuguer, visualiser ce qu’elle fait et voit, remonter des informations sur cette dernière et la commander manuellement à distance, il est important d’avoir un système de communication sans fil à bord de la voiture.

Comme nous pourrons le voir dans le planning (chapitre 13), il a tout d’abord été prévu de mettre en place un système de communication par radiofréquence entre le SBC de la voiture et l’ordinateur (le choix du RPi n’était à ce moment pas encore fait).

Un tel système demande une installation électronique conséquente sur la voiture et sur l’ordinateur, ainsi que la mise en place d’un protocole de communication afin de transférer des informations.

Après avoir fait le choix du RPi, le transfert d’information par protocole TCP/IP (internet) entre la voiture et l’ordinateur est devenu évidente. La connexion peut se faire par câble ou par WiFi, partout dans le monde et est rapide. Toute l’électronique nécessaire à son fonctionnement est déjà intégrée dans le RPi et nos ordinateurs. De plus, tous les outils nécessaires afin de gérer le RPi à distance existent déjà :

* Shell à distance avec le protocole SSH
* Bureau à distance avec VNC
* Gestionnaire de fichier distant avec Samba
* Codage / débuggage à distance avec Visual Studio Code

Note : Différentes installations détaillées au point 6

Il a été décidé de pouvoir gérer la voiture manuellement si besoin. Une connexion à une manette de jeu a donc aussi été mis en place par Bluetooth.

## Choix du langage de programmation

Le **Python** a été choisi pour ce projet malgré le fait que mes connaissances en C / C++ / C# soient beaucoup plus grandes (33 crédits dédiés à ces langages, contre 0 pour le Python).

Ce choix a été pris pour les raisons suivantes :

* La quasi-totalité de la communauté RPi code en Python, donc les tutoriels, forums et librairies sont en Python
* Le Python permet une gestion simple des matrices (notamment pour les images)
* Les principales librairies de deep learning sont en Python
* Il m’intéressait d’apprendre ce langage

Il a donc fallu apprendre tout le fonctionnement de ce langage. Entre les packages, la gestions des classes et ce qui en découle (héritage, propriété, variable publique & privé, etc.), les threads, les allocations, les évènements, ou encore les fonctions et opérateurs de base du langage.

Bien que beaucoup de principes ressemblent au C++, le langage reste bien différent et avec ses particularités.

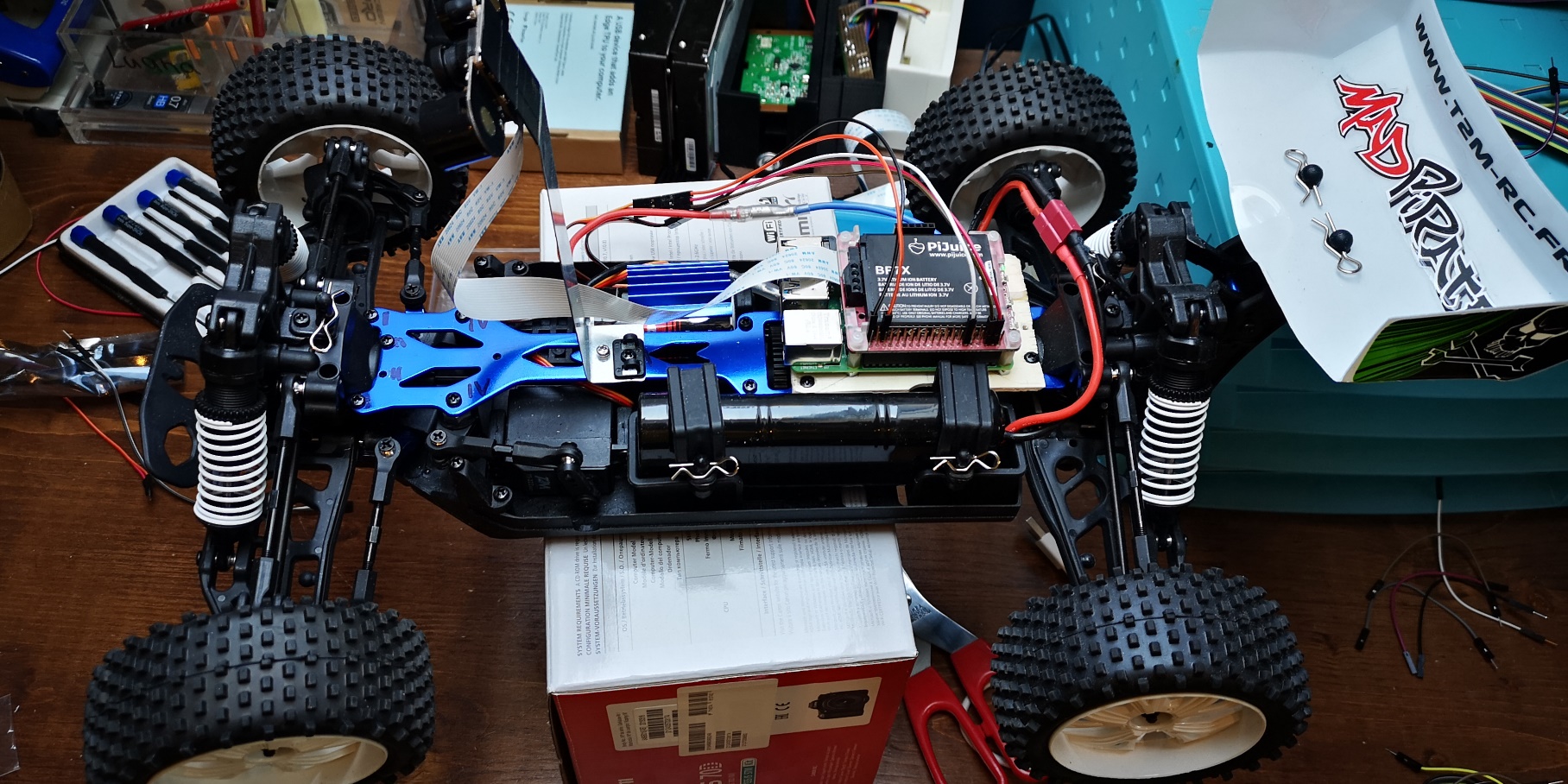
# Conception mécanique

## Voiture

Deux pièces ont dû être confectionné. Le support pour venir fixer le RPi et celui pour la caméra.

Ces supports ont tout d’abord été fait avec ce que j’avais chez moi, ne pouvant pas avoir accès aux locaux de l’école, et ont été fait en bois et en tôle.

Figure 6 Support RPi et camera, V1



Le problème de ce support de caméra est que la tôle étant fine, peu rigide et longue, le moindre mouvement de la voiture faisait beaucoup vibrer la caméra, ce qui rendait son utilisation impossible

Après avoir eux accès à une imprimante 3D, un nouveau supports bien plus rigide, avec une bonne hauteur et inclinaison a été conçu.

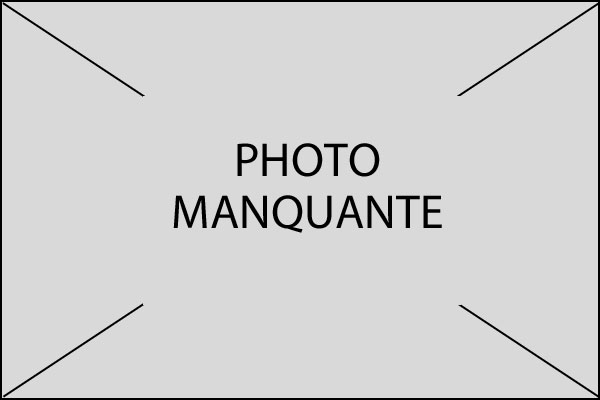


Figure 7 Support camera, V2

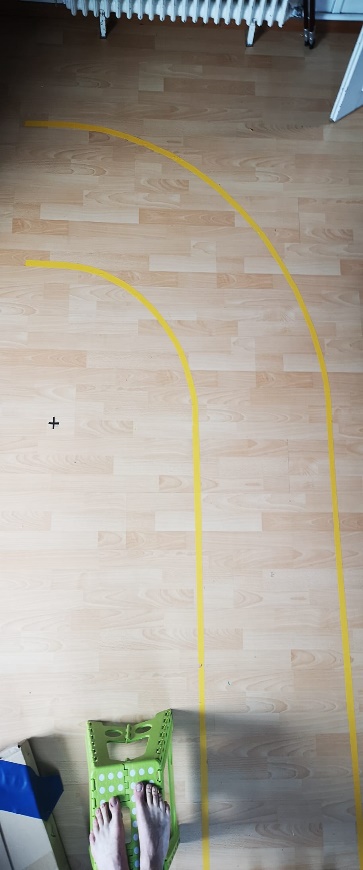
## Route

La largeur de la route a été choisi en se basant sur les vraies largeurs de route.

* Largeur des routes nationales française (8): 3.5m
* Largeur d’une voiture standard (9): 1.8m
* Largeur voiture RC : 30cm
* Largeur route miniature :

Ensuite pour les virages, un rayon de courbure moyen de 90cm a été choisi, afin que la route passe chez moi.

Figure 8 Rayon de courbure de la route et parties modulables



60cm

120cm



Pour finir, une route modulable a été conçu avec des parties de route en cartons de 1m50 se fixant les uns aux autres avec des scratchs afin de pouvoir modifier le parcoure et la déplacer plus facilement.

Le choix des couleurs des bordure sera discuté au point 8

## Signalisation

La voiture étant un modèle à 1/10eme, il a été choisi de garder la même proportion pour les panneaux et feux.

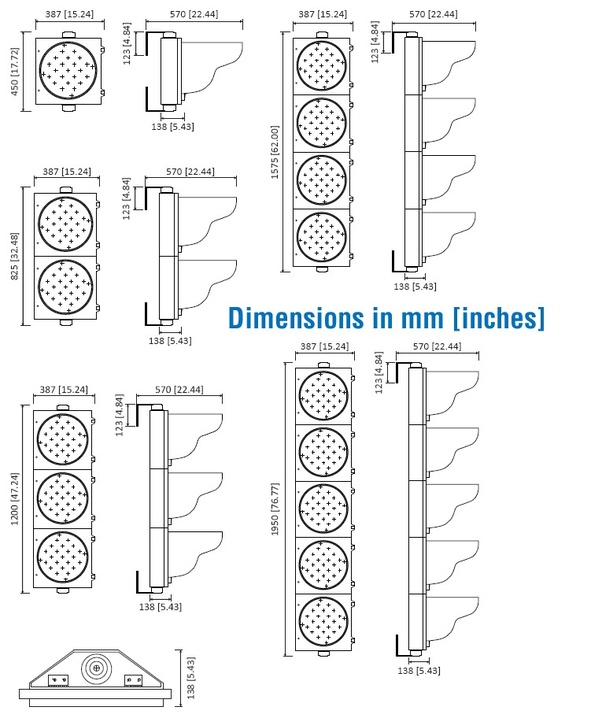
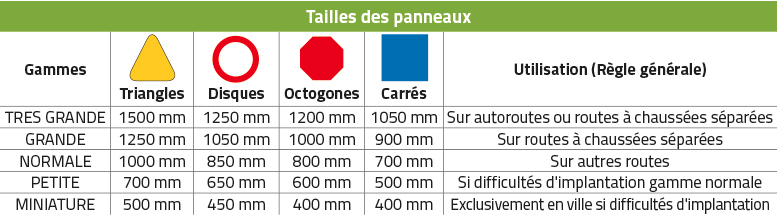


Figure 9 Taille des panneaux et feux Source : (11) et (12)

Le tout a été imprimé en 3D. Une taille « NORMALE » de panneau a été choisi et comme nous le verrons au point 5.2, de la place a été laissé dans le feu afin d’y mettre un Arduino Nano et une antenne BT.

# Conception électronique

## Electronique de la voiture

Le montage électronique de la voiture est relativement simple. La majorité des composants peuvent simplement être connecté les uns avec les autres.

Une carte des composants et de leurs connections suffit donc à reproduire le montage électronique.

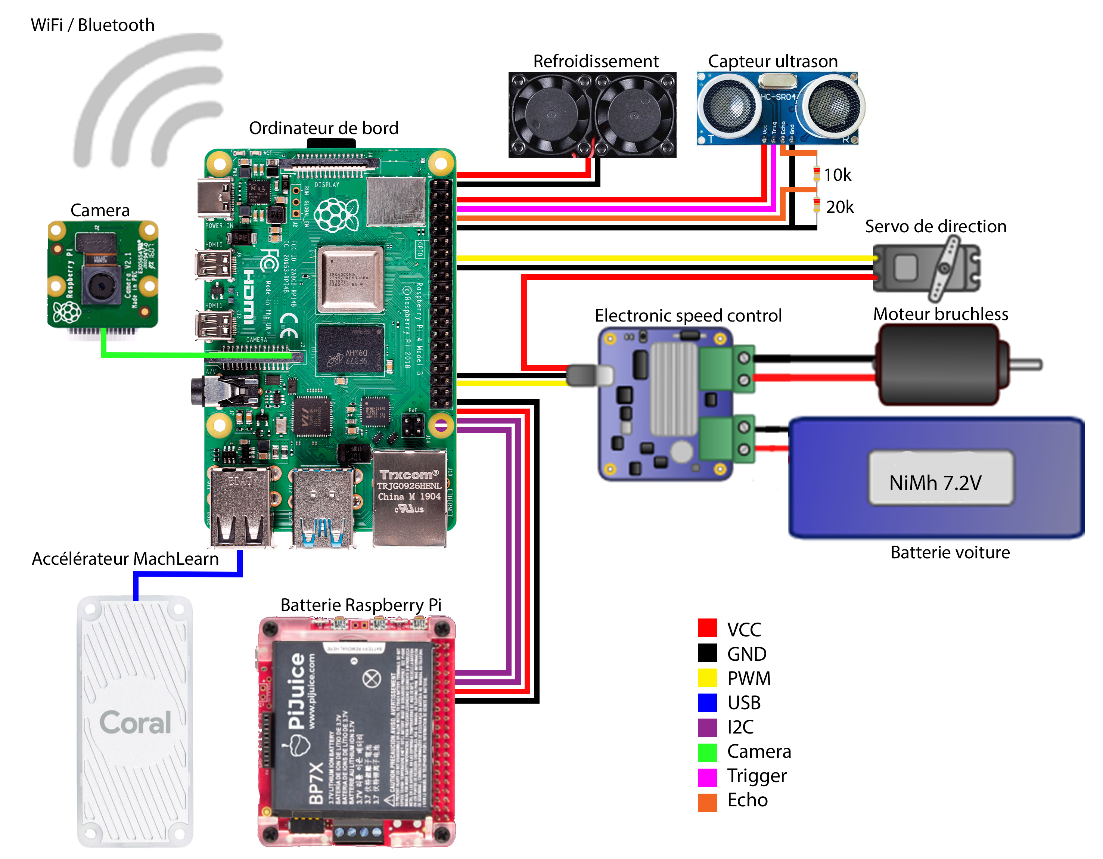
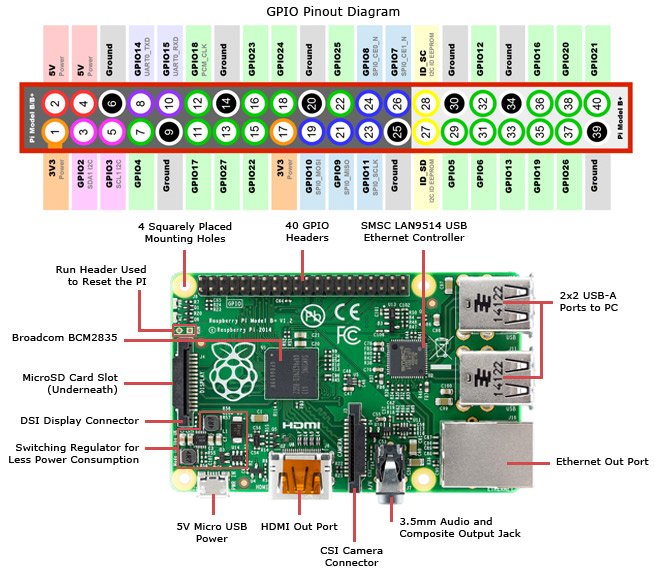


Figure 10 Montage électronique

* 2 (5V): HC-SR04 VCC
* 4 (5V): Fan VCC
* 6 (GND): Fan GND
* 9 (GND): HC-SR04 GND
* 12 (GPIO18): Speed controller PWM
* 14 (GND): Speed controller GND
* 16 (GPIO23): HC-SR04 Trigger
* 18 (GPIO24): HC-SR04 Echo
* 35 (GPIO19): Servo PWM
* 39 (GND): Servo GND

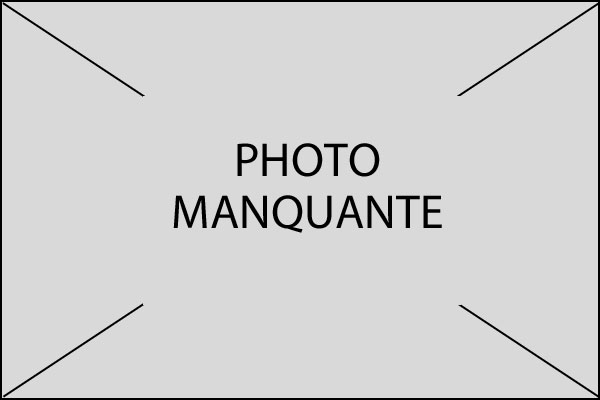
Le PiJuice vient simplement se plug via le connecteur 40pin et ne condamne aucune pin. Il faut juste noter que **l’adresse 14 du bus I2C (valeur modifiable) est utilisé par le PiJuice.**

Un diviseur de tension 5V 🡪 3.3V a tout de même dû être soudé à la sortie du capteur de distance.

Les GND des deux batteries, du RPi et du servo sont bien tous connectés.

Les PWM ont par ailleurs poser passablement de problèmes, qui seront discutés au point 11.1.

Le RPi et la batterie ont tout d’abord été placé céparément afin de prendre moins de hauteur et ainsi permettre de remettre la carosserie de la voiture. Cepandant une telle connexion entre les deux élements avec des jumpers (tout ce que j’avais chez moi en période de confinement) menaient à des problèmes de communication sur le bus I2C dû à des perturbations. Les fonctionnalités de la batterie étaient inutilisables. Il a donc fallu tout de même connecté la batterie au RPi directement via le connecteur 40 pins.



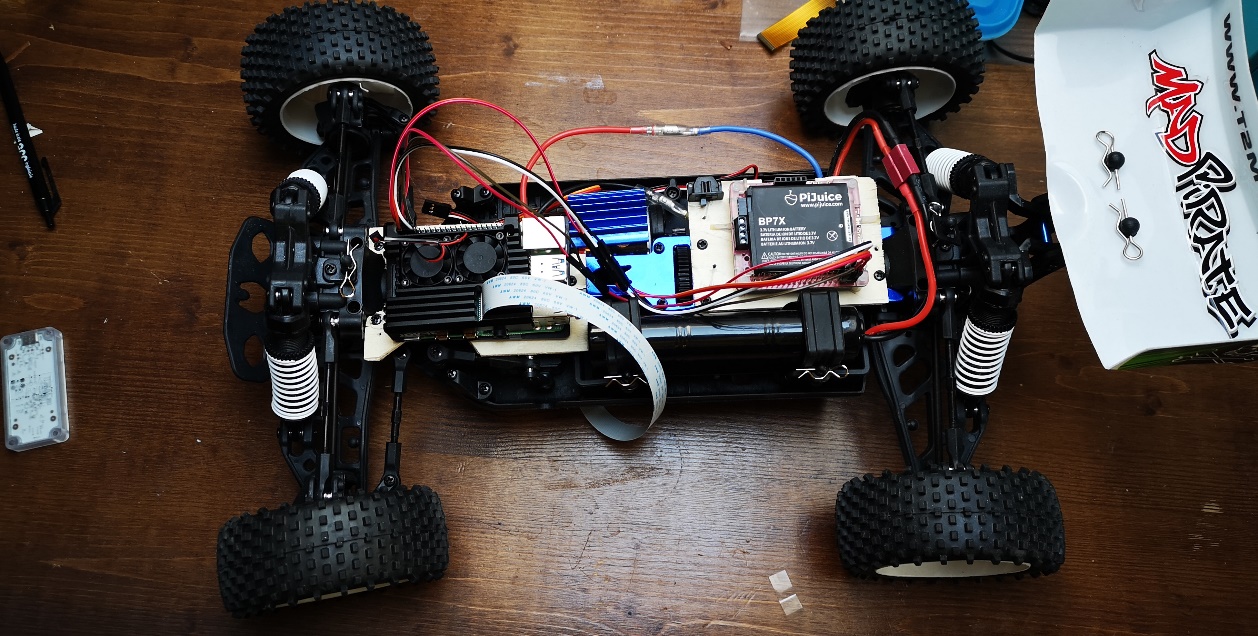


Figure 11 Montage batterie séparée et ensemble

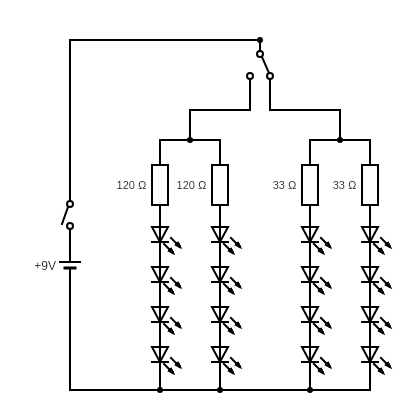
Par contre, étant donné que le système de refroidissement du RPi (voir 3.2.3) prend trop de place, il n’a pas été possible de fixer la batterie comme prévu.

Il aurait été possible de le fixer sous le RPi, mais ce dernier n’a pas un connecteur 40pin prévu pour fixer des modules par-dessous. Il aurait fallu remplacer celui d’origine, au risque d’abimer le RPi.

Il a donc été choisi de monter une rallonge sur le connecteur du RPi.

## Electronique du feu bicolor

Le feu devait à la base contenir un Arduino Nano ainsi qu’une antenne BT afin de pouvoir le guider à distance depuis un téléphone ou la voiture. Cependant la décision a été prise de simplifier sa conception et de le commander manuellement via un switch afin d’y consacrer moins de temps et pouvoir l’investir sur d’autres parties du travail.



Rouge Vert

Figure 12 Circuit électronique d'un feu

Le feu est alimenté par une pile 9V, contient un interrupteur principal et un switch permettant de passer du rouge au vert.

Afin d’avoir une source lumineuse bien répartie derrière la vitre du feu, au minimum 8 LED sont nécessaires.

Par soucis de simplicité de conception, les LED ont été mis en série. Cependant les LED rouges ayant une chute de tension () de 1.7V et les verte de 2.1V, uniquement 4 LED pouvaient être mis en série ( et ). Deux blocs de 4 LED ont donc été faits.

Le calcul des résistances découle du courant voulu dans les LED. Ce dernier à été choisi afin d’avoir un compromis entre intensité lumineuse et consommation du feu et a été choisi à 20mA.

Les résistances se calculs de cette façon :

Et valent donc ( de la série e12) pour le rouge et () pour le vert.

# Installation RPi

Script Pi Juice

# Architecture logicielle

Une première architecture logicielle a été pensé au début du projet, mais a beaucoup évolué lors du développement et l’acquisition de nouvelles connaissances. La version finale est présentée ici.

## La classe “Car”

Cette classe est une représentation de la voiture dans le monde réel et est à la base de toute communication entre le soft et les différentes capteurs et actuateurs.

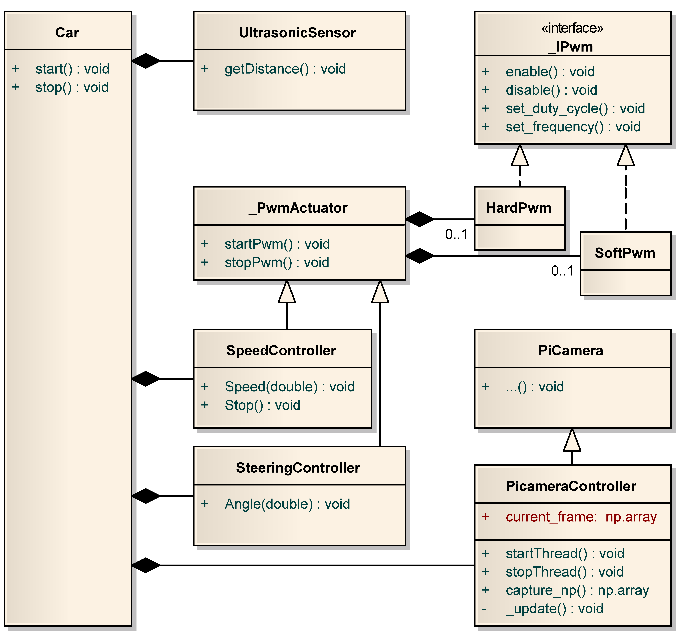


Figure 13 UML de la voiture

Nous pouvons voir que la voiture est constituée de :

* Un capteur de distance (UltrasonicSensor)
* Un moteur bushless (SpeedController)
* Un servo de guidage (SteeringController)
* Une caméra (PicameraController)

Le contrôleur du moteur et le servo étant les deux piloté grâce à un PWM, leurs classes correspondantes héritent toutes les deux de la classe \_PwmActuator.

Cette dernière va permettre de généraliser l’initialisation et le démarrage des sorties PWM.

Après avoir rencontré un problème de précision des PWM (voir point 11.1), il a dû être nécessaire d’implémenter un PWM hardware et de ne plus utiliser la classe de base « RPi.GPIO.PWM » qui utilise un PWM software.

Afin de permettre de passer facilement d’un PWM software à hardware, les classes SoftPwm et HardPwm réalisent le même interface (\_IPwm). Ainsi leurs utilisations est parfaitement identique et les différences d’implémentation sont gérées en interne.

Lors de la création d’un objet \_PwmActuator, son paramètre « hardware :bool » définira quelle sorte de PWM sera créé.

La classe PicameraController hérite de la classe PiCamera (provenant du package portant le même nom) et lui ajoute deux fonctionnalités non présentes de base.

* L’implémentation d’une fonction de capture d’image directement au format Numpy
* La gestion d’un thread dédié à la capture d’image, capturant à la vitesse spécifié dans le paramètre « framerate » de la caméra et plaçant le résultat dans la variable current\_frame afin que celui-ci puisse être utilisé par d’autres threads.

## L’application

L’application contient :

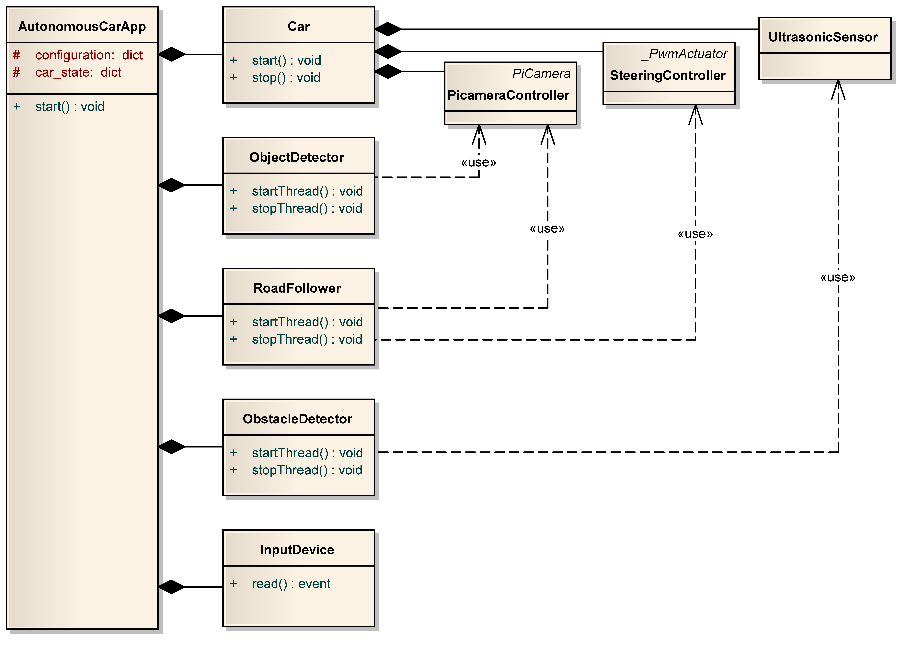


Figure 14 UML de l’application

* La représentation de la voiture
* L’algorithme de détection de panneaux
* L’algorithme de suivit de route
* L’algorithme de détection d’obstacles
* Et un gestionnaire d’input qui est dans notre cas la manette de jeux.

Les différents algorithmes utilisent les capteurs de la voiture afin d’acquérir l’image courante (mis à jour par le thread de la caméra) ou la distance fournie par le capteur à ultrason.

##### Un ou plusieurs thread?

L’application a tout d’abord été conçu de façon séquentielle. L’image était d’abord capturée, traitée, puis chaque algorithme analysait ce qu’il fallait faire un après l’autre, puis pour finir la vitesse et la direction étaient modifiés.

Le problème est que chaque partie possède des phases d’attente du matériel externe.

* La capture d’image n’est pas immédiate et attend sur la caméra
* La détection d’obstacle doit attendre que l’écho revienne

La direction des roues est directement modifié par le thread de suivit de route parce que c’est le seul qui modifie cette valeur.

## Fichier de configuration

# Suivit de ligne

# Detection de panneaux

### MobilNet

### TensorfFlow vs Tensorflow Lite

(10)

# Detection d’obstacles

# Problèmes rencontrés

Servo de guidage alimenté par le RPi -> Il s’arrétait

Camera IR pose problème au niveau des couleurs

## PWM pas correcte

## Le RPi s’arrête

# Améliorations

## Plaque de support du RPi

# Planning

# Remerciements

Je tiens à exprimer mes sentiments de gratitude et de reconnaissance à Monsieur Pierre BRESSY, d’abord pour ses conseils et ses encouragements et ensuite pour m’avoir suivi durant toute la durée de ce travail de diplôme en mettant à ma disposition ses connaissances dans la gestion d’un tel projet, sa disponibilité, sa patiente ainsi que son expertise technique. Un remerciement particulier aussi à Madame Claire MEYER et Monsieur Guillaume BANGALA, Monsieur Matthias HUSER pour leurs disponibilités, leurs conseils, et encouragements. Ensuite, j’exprime aussi mes remerciements à mes collègues Vincent BERGER, Rémy PAPAUX, Thomas JOLIAT, Michael HUBER, pour leurs coopérations pendant la validation et les tests de l’application. En fin, je remercie toute ma famille en particulier mon frère Monsieur Martial MBEFO et son épouse Tatiana MBEFO, pour m’avoir leurs encouragements et soutiens tout au long de ce projet.

# Bibliographie

1. **Brushed Speed controller WP40 - Instructions.** [Online] [Cited: 07 16, 2020.] <https://asset.conrad.com/media10/add/160267/c1/-/gl/001399921ML02/mode-demploi-1399921-regulateur-de-vitesse-brushed-de-voiture-reely-wp-1040-brushed-charge-admissible-max-180-a-limite-moteur-nombre-de.pdf>.

2. **Best Single Board Computers.** [Online] [Cited: 07 16, 2020.] <https://www.seeedstudio.com/blog/2019/11/20/best-single-board-computers-of-2019/>.

3. **Rock Pi N10 vs Raspberry Pi 4 vs Jetson Nano.** [Online] [Cited: 07 16, 2020.] <https://www.seeedstudio.com/blog/2019/12/05/rk3399pro-vs-raspberry-pi-4-vs-jetson-nano-ai-and-deep-learning-capabilities/>.

4. **Benchmarking Machine Learning.** [Online] [Cited: 07 17, 2020.] <https://www.hackster.io/news/benchmarking-machine-learning-on-the-new-raspberry-pi-4-model-b-88db9304ce4>.

5. **Debian Boot-time on Raspberry PI.** [Online] [Cited: 07 16, 2020.] <https://raspberrypi.stackexchange.com/questions/320/what-is-a-typical-boot-time-for-the-standard-debian-distribution-on-a-typical-sd>.

6. **Bringing Colour to Point Clouds.** [Online] [Cited: 07 17, 2020.] <https://www.gim-international.com/content/news/bringing-colour-to-point-clouds-2>.

7. **PiJuice HAT.** [Online] <https://uk.pi-supply.com/products/pijuice-standard>.

8. **Surfaces occupées par les infrastructures routières.** [Online] [Cited: 07 19, 2020.] <http://piece-jointe-carto.developpement-durable.gouv.fr/REG074B/FONCIER_SOL/N_OCCUPATION_SOL/L_EMPRISE_ROUTE_R74/Fiche1-7-1.pdf>.

9. **Cas de croisements et largeur de chaussée.** [Online] [Cited: 07 19, 2020.] <https://mobilitepietonne.ch/wordpress/wp-content/uploads/2017/07/06_2017_Fiche-info_Cas_de_croisement.pdf>.

10. **TensorFlow and TensorFlow Lite on the Raspberry Pi.** [Online] [Cited: 07 17, 2020.] <https://www.hackster.io/news/benchmarking-tensorflow-and-tensorflow-lite-on-the-raspberry-pi-43f51b796796>.

11. **Reglementation signalisation routiere.** [Online] [Cited: 07 19, 2020.] <https://www.signals.fr/cms/reglementation-signalisation-routiere.html>.

12. **How big is a traffic light ?** [Online] [Cited: 07 19, 2020.] <https://www.quora.com/How-big-is-a-traffic-light>.

# non-plagiat

Je soussigné, Maxime CHARRIÈRE, atteste par la présente avoir réalisé ce travail et n’avoir utilisé aucune autre source que celles expressément mentionnées.

Yverdon-les-Bains, le 24.07.2020

Maxime CHARRIÈRE

# Annexes

## Énoncé du TB

## Clause de confidentialité