

# 2015

## Voiture avec anti-patinage télécommandé par un smartphone



Blazevic Mehmed

CFPT

01/06/2015

Professeur : M. Thierry Forestier

Expert : M. Christophe Perriard

## **Table des matières**

<b>1.</b>	<b>Sujet.....</b>	<b>3</b>
<b>2.</b>	<b>Introduction.....</b>	<b>3</b>
<b>3.</b>	<b>Enoncé.....</b>	<b>4</b>
3.1.	Photos du produit fini.....	5
<b>4.</b>	<b>Analyse et pré-étude.....</b>	<b>6</b>
4.1.	Analyse fonctionnelle de la voiture .....	6
4.2.	Analyse fonctionnelle smartphone .....	7
<b>5.</b>	<b>Etude de la voiture.....</b>	<b>8</b>
5.1.	La mécanique.....	9
5.1.1.	<i>Transmission</i> .....	9
5.1.2.	<i>Direction</i> .....	10
5.2.	Le récepteur .....	11
5.3.	Les moteurs.....	12
5.3.1.	<i>Introduction aux moteurs</i> .....	12
5.3.2.	<i>Le servomoteur</i> .....	12
5.3.3.	<i>Le moteur principale</i> .....	14
5.3.4.	<i>La batterie</i> .....	18
<b>6.</b>	<b>Etude des différents éléments .....</b>	<b>19</b>
6.1.	Schéma bloc matériel d'origine de la voiture .....	19
6.2.	Schéma bloc matériel modifié.....	19
6.3.	Introduction aux éléments.....	20
6.3.1.	<i>Le microcontrôleur</i> .....	21
6.3.2.	<i>Le module Bluetooth</i> .....	22
6.3.3.	<i>Le capteur de vitesse</i> .....	30
6.3.4.	<i>Accéléromètre</i> .....	33
6.3.5.	<i>Le capteur de distance</i> .....	35
6.3.6.	<i>Real Time Clock</i> .....	38
6.3.7.	<i>Carte SD</i> .....	40
<b>7.</b>	<b>Conception et développement .....</b>	<b>41</b>
7.1.	Conception et développement matériel .....	41
7.1.1.	<i>Schéma bloc fonctionnel</i> .....	41
7.1.2.	<i>Alimentation</i> .....	42
7.1.3.	<i>Mise en forme vitesse</i> .....	44
7.1.4.	<i>Protection tension basse</i> .....	45
7.1.5.	<i>Schéma complet</i> .....	48
7.2.	Conception et développement logiciel uC.....	52
7.2.1.	<i>Conception microcontrôleur</i> .....	52
7.2.2.	<i>UART</i> .....	52

7.2.3.	<i>Les timers</i> .....	54
7.2.4.	<i>PCAS</i> .....	54
7.2.5.	<i>ADC</i> .....	54
7.2.6.	<i>Les ports</i> .....	55
7.2.7.	<i>Le code : les fonctions</i> .....	56
7.2.8.	<i>Le code : fonctionnement</i> .....	60
7.2.9.	<i>Protocole de communication</i> .....	64
7.3.	<b>Conception et développement logiciel Android</b> .....	68
7.3.1.	<i>L'inclinaison du smartphone</i> .....	68
7.3.2.	<i>La communication Bluetooth</i> .....	69
7.3.3.	<i>Les différentes class</i> .....	69
7.3.4.	<i>MainActivity</i> .....	70
7.3.5.	<i>ConnectThread</i> .....	70
7.3.6.	<i>ConnectedThread</i> .....	71
7.3.7.	<i>Principe de fonctionnement</i> .....	71
7.3.8.	<i>L'application</i> .....	72
7.4.	<b>Test Fonctionnel</b> .....	73
8.	<b>Conclusion</b> .....	74
8.1.	Aspects techniques sur le projet.....	74
8.2.	Améliorations possibles.....	74
8.3.	Aspects gestions du projet.....	75
8.4.	Aspects personnels .....	75
9.	<b>Remerciements</b> .....	76
10.	<b>Outils de développement utilisés</b> .....	76
11.	<b>Sources et bibliographies</b> .....	76
12.	<b>Annexes</b> .....	77
12.1.	Répartition heures .....	77
12.2.	Planning .....	78
12.3.	Dossier de fabrication .....	79
12.3.1.	<i>Plan d'implantation</i> .....	79
12.3.2.	<i>Dimensions</i> .....	79
12.3.3.	<i>Liste des composants</i> .....	80
12.3.4.	<i>Implantation dessus</i> .....	81
12.3.5.	<i>Implantation dessous</i> .....	81
12.4.	<b>Codes Android</b> .....	82
12.4.1.	<i>MainActivity</i> .....	82
12.4.2.	<i>GestionTrames</i> .....	94
12.4.3.	<i>ConnectThread</i> .....	97
12.4.4.	<i>ConnectedThread</i> .....	99
12.5.	<b>Code uC</b> .....	101

## **1. Sujet**

Voiture avec anti patinage télécommandé par un smartphone

## **2. Introduction**

Lors de la deuxième et dernière année de technicien en électronique, il nous est demandé de réaliser un projet de diplôme. Celui-ci est évalué par les professeurs de l'école et par des experts venant d'entreprises ou écoles externes au CFPT. Il consiste en la réalisation d'un appareil ; d'une étude sur différents aspect de l'électronique, afin que les étudiants démontrent leur compétences acquises au cours de la formation.

Le but de ce projet est de réaliser un appareil complet qui prend en compte différents aspects de l'électronique comme l'analogique, le numérique et l'électronique de puissance pour l'alimentation. Ce travail de diplôme consiste à reprendre un modèle de voiture RC et modifier l'électronique, afin qu'elle puisse fonctionner selon les critères sélectionnés par le professeur qui a proposé le sujet. La voiture a été acheté complète, afin de ne pas avoir à se soucier de la mécanique. Elle est étonnement complète et efficace pour une voiture de 1/10. La mécanique est prévue pour faire du drift. C'est une discipline sportive qui consiste à faire glisser les voitures dans les virages, en travers. Avec des roues suffisamment dures, un moteur qui a du couple et quatre roues motrices, la voiture est quasiment impossible à faire avancer sans patiner avec les roues.

La grosse difficulté de ce projet consiste à créer un système anti-patinage au moyen de plusieurs capteurs. Deux capteurs nous permettent cela : le premier est un accéléromètre et le deuxième est un capteur de vitesse. Si on calcule la dérivée de cette vitesse, nous obtenons l'accélération. Grâce à cela nous avons deux grandeurs physiques qu'il ne reste plus qu'à comparer. Il y a beaucoup de calcul et d'élément à prendre en considération pour la réalisation de ce projet.

Etant donné que la voiture est un modèle réduit, il faut pouvoir la piloter à distance. C'est pour cela que l'on a décidé d'utiliser un smartphone qui fonctionne sous le système d'exploitation Android, pour piloter la voiture à distance. La communication se fait via Bluetooth et le pilotage se fait au moyen du gyroscope du smartphone. C'est-à-dire qu'il suffit de l'incliner pour contrôler la voiture. De plus, il peut afficher différentes données comme la vitesse, la charge de la batterie, etc.

La dernière fonction de la voiture est le stockage de données sur une carte SD. La carte électronique est prévue pour accueillir une carte SD. S'il y a suffisamment de temps, des fonctions d'enregistrement de données seront développées.

Il n'y a pas de réel contrainte pour la réalisation de ce projet, à part éventuellement la mécanique. Une carte électronique doit être réalisée, du coup il faut qu'elle puisse rentrer dans le modèle réduit. C'est la seule grosse contrainte de ce sujet.

Au final, le produit est destiné au grand public désireux d'avoir une petite voiture télécommandé, afin de pouvoir se divertir durant son temps libre.

### 3. Enoncé



REPUBLIQUE ET CANTON DE GENEVE  
Département de l'instruction publique, de la culture et du sport  
Enseignement secondaire II postobligatoire  
Centre de formation professionnelle technique

les écoles supérieures

#### Diplôme de technicien ES en électronique - 2015

Candidat : **Mehmed BLAZEVIC**

**Sujet** Voiture avec antipatinage télécommandé par un smartphone



**Description** Développement de l'électronique pour le contrôle d'une voiture télécommandé avec un smartphone "Android" à travers une liaison Bluetooth.  
La fonction d'anti-patinage qui sera développée, pourra être activée ou désactivée depuis le smartphone.  
L'interface homme-machine IHM permettra le contrôle de la voiture et l'affichage des informations comme la vitesse, l'accélération, le niveau de charge de la batterie, etc...  
La carte électronique de la voiture devra être équipé de :  
- un microcontrôleur C8051F320  
- un capteur d'accélération 3 axes (Analog Devices ADXL335)  
- un ou plusieurs capteurs de vitesse (à définir)  
- une horloge en temps réel RTC (Maxim DS3234S interface SPI)  
- un dispositif de sauvegarde sur carte SD

**Travail** Gestion du projet (planning, ressources)  
Pré-étude du système (analyse du besoin, analyse fonctionnelle, FAST)  
Etude et réalisation  
Développement des logiciels (micro C8051F320 et smartphone)  
Validation du dispositif complet  
Documentation

**A remettre** Un mémoire dactylographié en 3 exemplaires, couverture verte, comprenant:  

- Une description technique complète, avec la motivation des choix effectués, les schémas et les calculs des composants.
- Une description des structures de données et du protocole de communication
- Les descriptions, organigrammes et « listings » des programmes.
- Les résultats des essais et mesures effectuées.
- Les caractéristiques des composants spéciaux.

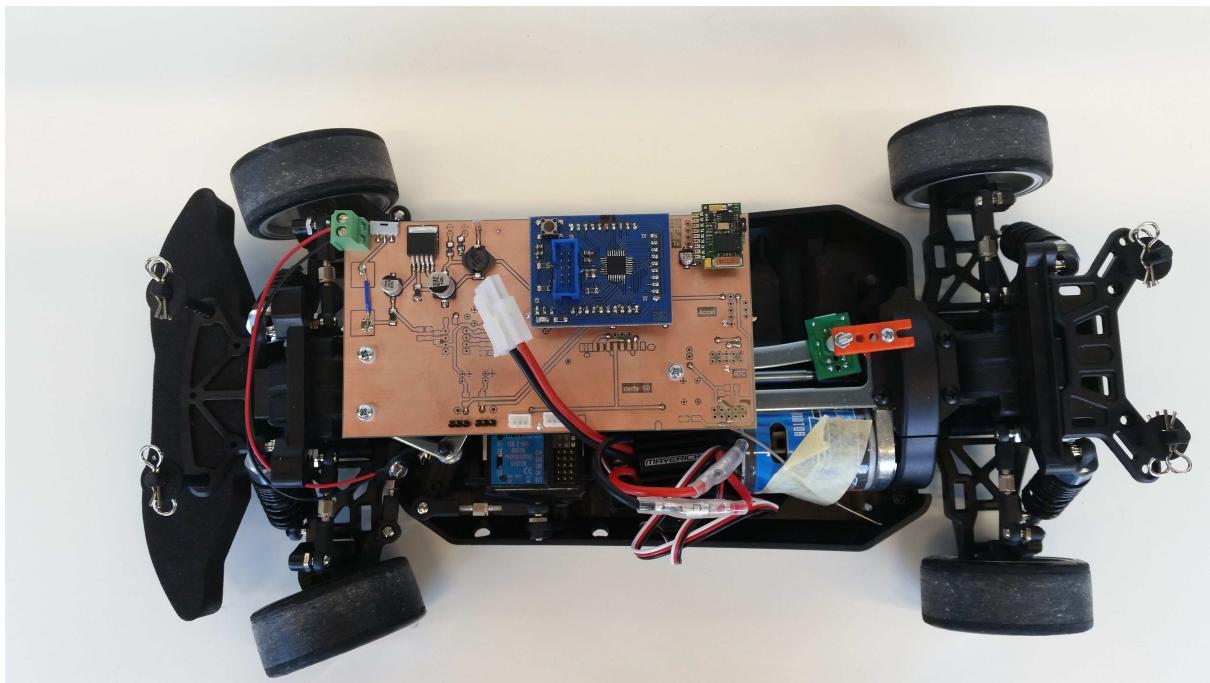
Le montage du dispositif, mécanique et électronique.

Les logiciels, schémas, organigrammes, etc., sur support informatique.

Thierry FORESTIER



### **3.1. Photos du produit fini**



## **5. Etude de la voiture**

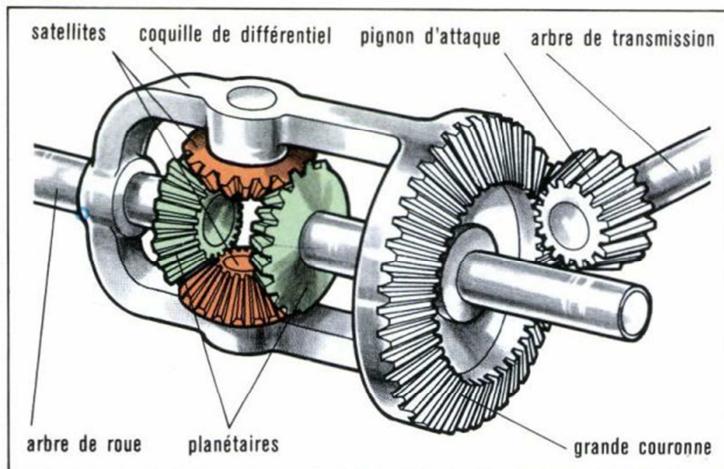
Une grosse partie de ce projet aura consisté à étudier la voiture. Il y a une partie mécanique à comprendre et naturellement, une partie électronique. Différentes mesures ont étées faites : consommation générale de la voiture, puissance du moteur, signaux qui permettent le contrôle des moteurs, angle de rotation des roues, etc. C'est l'une des parties du projet qui aura pris le plus de temps et qui a été très intéressante à réaliser. La voiture a été suffisamment étudiée, afin de pouvoir par la suite être reprise par n'importe qui et pouvoir être exploitée rapidement. La voiture est une « maverick strada dc evo » et voici à quoi elle ressemble :



## 5.1. La mécanique

### 5.1.1. Transmission

Les premiers paramètres qui ont été analysés sont la manière dont la transmission intégrale est faite. C'est ce qui a été le plus intéressant, car on constate que même sur une voiture aussi petite, la mécanique est précise et la transmission se fait au moyen d'un arbre et de différentiels de transmission. C'est sensiblement le même système que l'on retrouve dans les automobiles dotés de 4 roues motrices. Voici à quoi ressemble un différentiel :



Eléments constitutifs d'un différentiel.

Le différentiel va donc servir à transmettre la force du moteur sur les roues. Deux différentiels sont utilisés : l'un se trouve au milieu à l'avant de la voiture et le deuxième au milieu à l'arrière. Cette paire sera reliée par un arbre qui va parcourir toute la longueur de la voiture. Ainsi, lorsqu'on va appliquer une force sur l'arbre de transmission pour le faire tourner, les 4 roues seront entraînées simultanément, à vitesse égale. C'est ainsi que la majorité des voitures à quatre roues motrices fonctionnent, avec des éléments supplémentaires qui servent à réguler et à transmettre plus ou moins de forces à une paire de roue, afin d'améliorer les performances de la voiture.

C'est d'ailleurs l'arbre de transmission qui est utilisé pour mesurer la vitesse de rotation des roues et ainsi connaître la vitesse de déplacement de la voiture. Pour s'y faire, un capteur a été placé au-dessus de la transmission et mesure sa vitesse. Voici l'arbre avec le capteur de vitesse en vert :



### 5.1.2. Direction

Comme dit précédemment, la direction est assurée par un servomoteur et tout un jeu de barre d'accouplement. Le fonctionnement n'est pas très compliqué. Le servomoteur fournit un mouvement rotatif que l'on va coupler avec des barres pour y convertir en mouvement de vas et viens. Nous pourrons ainsi diriger les roues en utilisant ces barres. Ceci étant compliqué à expliquer, voici une image qui montre comment le fabricant a réalisé le système de direction :



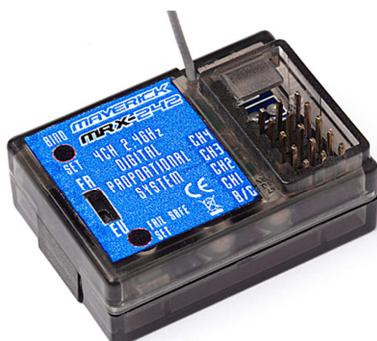
Des mesures ont été faites, afin de connaître l'angle d'inclinaison des roues par rapport à l'axe de la voiture. L'angle maximum est de 20 degrés à gauche et à droite. Ce qui nous fait un total de 40 degrés maximum pour l'ouverture des roues. Grâce à cela nous pourrons diriger la voiture avec précision. Les mesures de l'inclinaison des roues seront détaillées dans la section qui décrit le fonctionnement du servomoteur, avec un graphique de mesures.

Avec l'étude de la mécanique de la voiture, on se rend compte que la voiture est relativement bien construite et bien réfléchie. On peut faire beaucoup de liens entre le module réduit et les vrais véhicules qui jonchent les rues. Le modèle utilisé dans ce projet possède en plus des amortisseurs et des cardans de direction comme on peut le voir sur les diverses photos. Ces deux parties de la mécanique ne sont pas détaillées, car elles ne sont pas utiles à la suite du projet. Cela dit, il est quand même intéressant de constater que la mécanique est très complète et possède tous les atouts à l'élaboration du projet et à d'éventuelles modifications par la suite. Tous les composants sont disponibles en magasin, ce qui permet de personnaliser la voiture à notre guise.

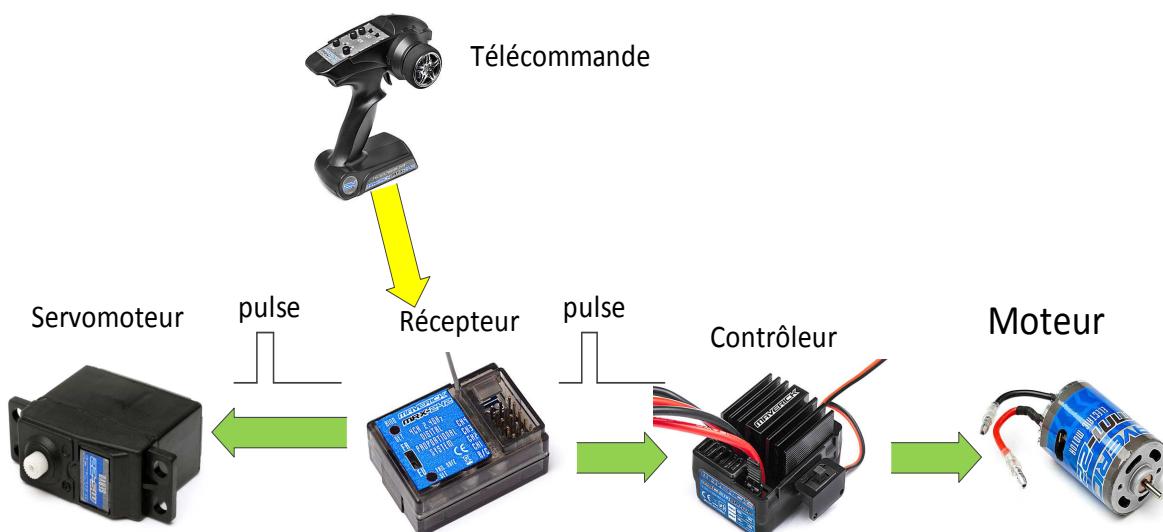
## 5.2. Le récepteur

La commande des différents éléments de la voiture passe par le récepteur. C'est un module 4 canaux fonctionnant en 2.4GHz. C'est le module qui va communiquer avec la télécommande de la voiture et interpréter les données envoyées, afin de commander les deux moteurs de la voiture. C'est l'élément qui doit être remplacé par la carte électronique fabriquée pour pouvoir diriger la voiture avec le smartphone.

Avant de pouvoir remplacer le récepteur, il faut étudier son fonctionnement. Pour commencer, en voici une image du récepteur :



Le récepteur est relié au servomoteur et au contrôleur du moteur principal. C'est le contrôleur qui va alimenter le récepteur avec une tension continue de 5 volts. Une fois celui-ci alimenté, il peut communiquer avec la télécommande de la voiture. La télécommande envoie des valeurs de consignes et ensuite le récepteur pilote les moteurs avec des pulses. C'est la largeur de ces pulses qui vont dire aux moteurs comment fonctionner. La largeur de pulse va varier de 1 milliseconde à 2 millisecondes. En fonction de cette valeur, le servomoteur va se positionner à un certain angle et le contrôleur du moteur principal interprétera le signal et pilotera ce dernier. Voici un schéma synoptique qui montre le lien entre les différents composants :



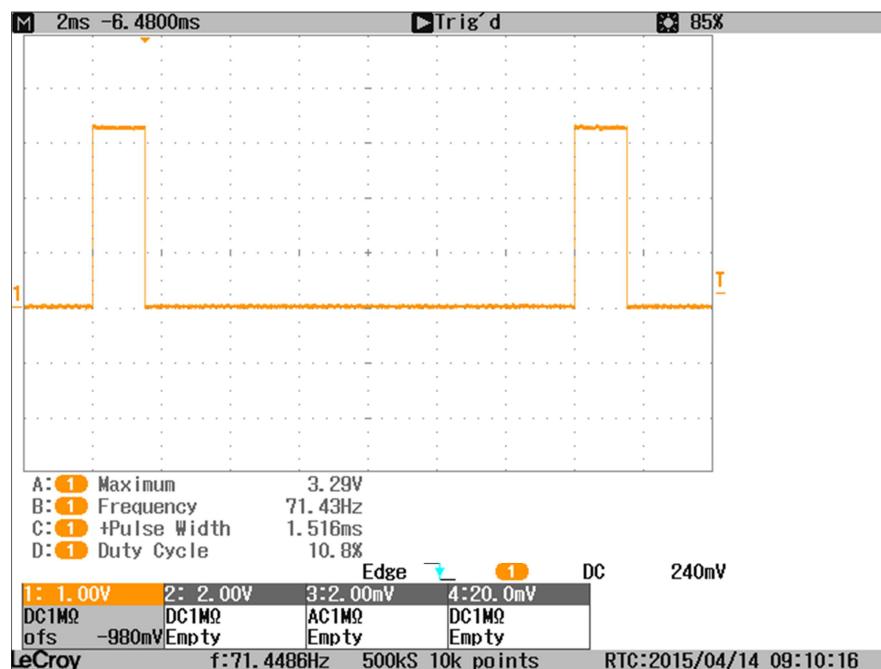
## 5.3. Les moteurs

### 5.3.1. Introduction aux moteurs

Pour la suite de l'étude, il faut faire des mesures sur les moteurs : le servomoteur et le moteur principal. Le premier va servir à diriger la voiture, au moyen d'une transmission et le deuxième est le moteur qui va tout simplement servir à faire avancer la voiture. Plusieurs mesures ont été effectuées, afin de connaître la consommation des moteurs et surtout, comment les contrôler. Pour ce dernier point, il faut débrancher le récepteur du reste de la voiture et procéder aux mesures avec l'oscilloscope.

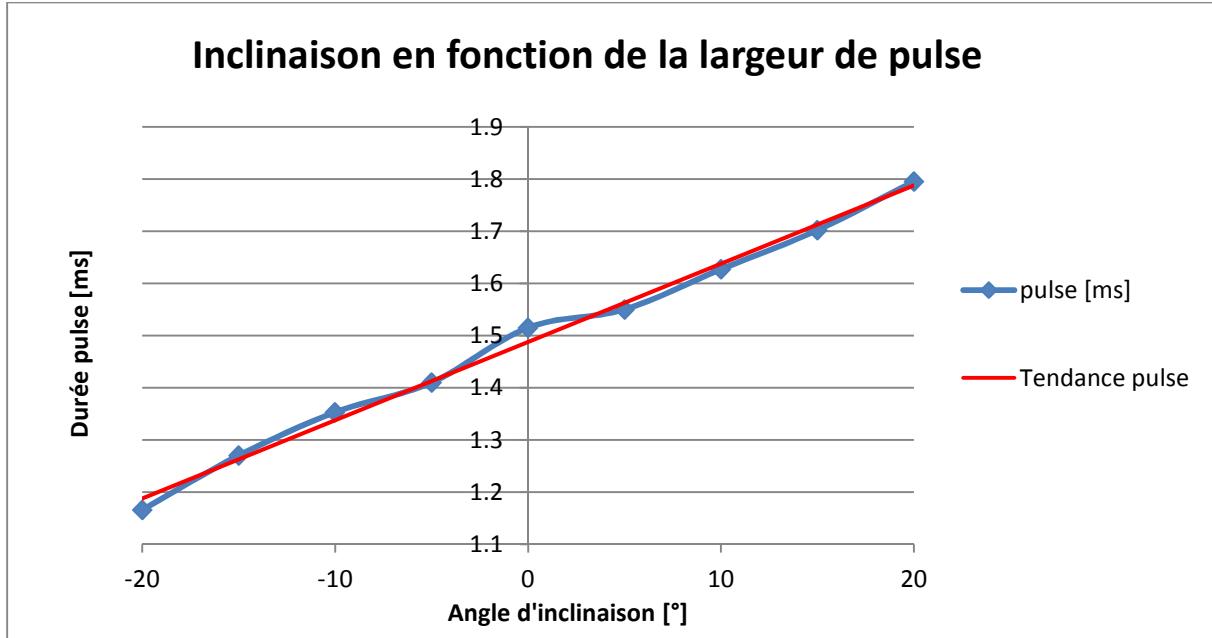
### 5.3.2. Le servomoteur

Le fonctionnement du servomoteur est relativement facile. Des mesures des signaux qui permettent de contrôler le servomoteur ont été faites avec un oscilloscope. On constate que pour contrôler ce dernier, il faut envoyer des largeurs de pulses qui varient entre 1 et 2 millisecondes, comme dit précédemment. Voici un oscillogramme qui montre ces pulses :



Sur ce relevé d'oscilloscope, on voit les signaux de commandes qui sont envoyés sur le servomoteur. Ces signaux sont envoyés depuis le récepteur de la voiture. La largeur de la pulse va déterminer la position du servomoteur et ainsi, la position des roues avant. Comme on peut le voir, la tension de commande du servomoteur est de 3.3 Volts, donc compatible avec le microcontrôleur C8051F320. C'est parfait car cet uC est connu et il n'y a pas besoin de l'interfacer avec le servomoteur avec un élément externe.

Comme dit précédemment, la largeur de pulse va déterminer la direction dans laquelle on veut tourner. Des mesures ont été effectuées afin de connaître exactement l'angle de rotation des roues en fonction de l'axe de la voiture. Ce qui nous permet de déterminer la trajectoire de la voiture et ainsi d'être précis dans la gestion de cette dernière. Voici un graphique qui nous montre le degré d'inclinaison des roues par rapport à l'axe de la voiture :



Ces données ont été relevées sur la roue droite de la voiture. On retrouve les mêmes données sur la roue gauche. C'est une très bonne chose, car notre voiture est relativement précise dans la direction. On constate que plus la largeur de la pulse est petite, plus on va tourner à droite et inversement, à gauche.

Nous pouvons en conclure que l'on va avoir un pas de **15.371 microsecondes par degrés** de rotation. Ainsi, nous pourrons contrôler la voiture de façon très précise.

Le servomoteur a une consommation électrique de 10 milliampères à l'arrêt. Si on lui donne une consigne, il consommera jusqu'à **300mA** pour se déplacer. Une fois l'angle voulu atteint, il va redescendre à une consommation de 10mA. Si on essaie de déplacer les roues, le servomoteur va de nouveau consommer le maximum de courant possible, afin de maintenir la position de consigne.

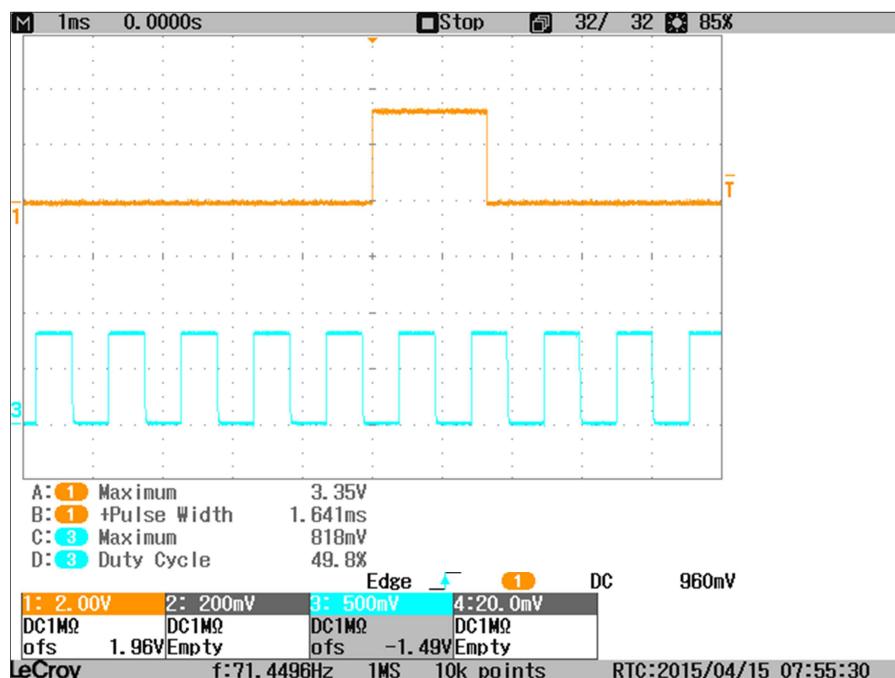
### 5.3.3. Le moteur principale

L'élément clé de la voiture est le moteur principal car c'est celui-ci qui permet à notre voiture de se déplacer. Il est assez difficile de trouver des données techniques le concernant. On sait que c'est un moteur puissant. Sa consommation, son couple, etc. ne sont pas donnés par le fabricant. Au moins, le moteur peut être étudié, afin de déterminer ses caractéristiques globales. Pour commencer, on sait que la batterie est de 7.2 volts. Cela laisse supposer que le moteur sera alimenté à environ 7 volts. Le moteur possède un contrôleur qui permet le pilotage de celui-ci en fonction de la largeur de pulse envoyée par le récepteur.

Etant donné que le moteur a son propre contrôleur, il n'est pas nécessaire de le caractériser, car on peut garder son contrôleur. Cela dit, cela ne serait pas digne d'un technicien. Donc, le contrôleur sera gardé pour piloter le moteur, mais il faut tout de même le caractériser, afin de connaître notre produit sur le bout des doigts.

La première étape est de déterminer comment le contrôleur est piloté. Les mesures nous montrent que le récepteur pilote les deux moteurs de la même manière : en utilisant des pulses qui varient de 1 à 2ms. C'est une excellente chose, car cela nous permettra de piloter les deux moteurs de la même manière, sans utiliser d'éléments complexes.

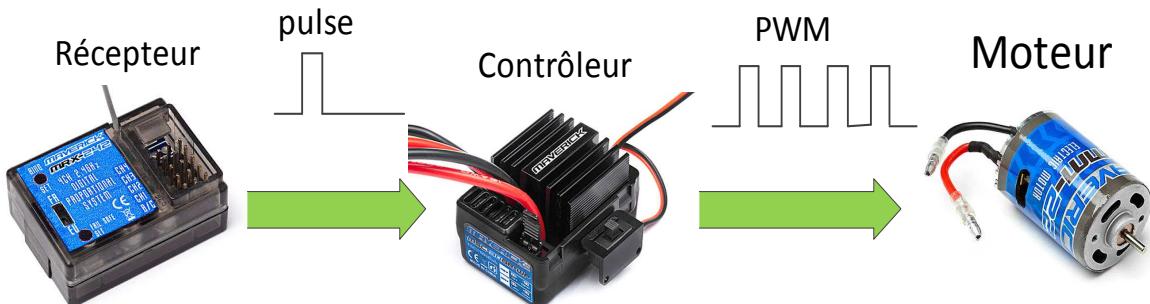
Une mesure est faite entre la commande du récepteur et la sortie du contrôleur du moteur :



Canal 1 : La commande du récepteur

Canal2 : La commande du contrôleur

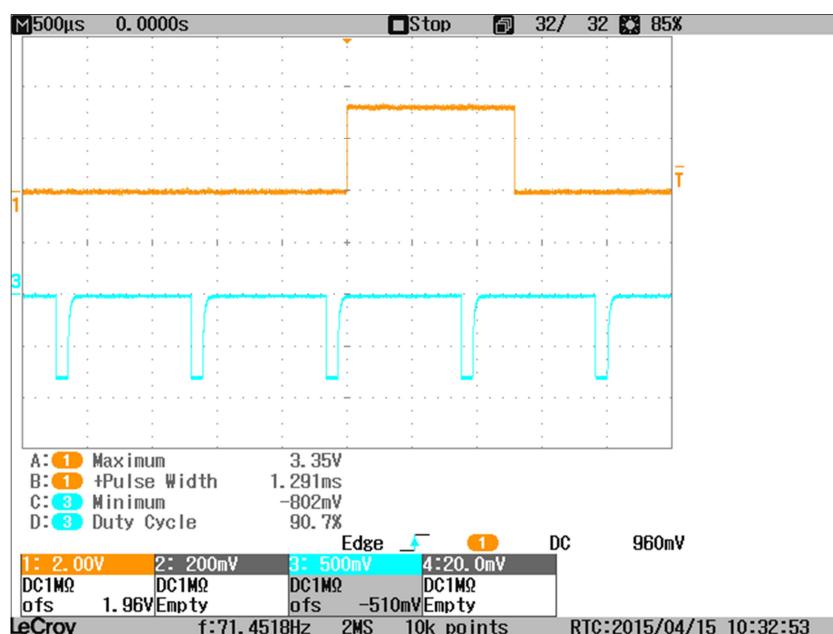
Pour mieux se situer à quelle niveau les mesures sont faites, voici un petit schéma synoptique :



Cela nous permet de savoir que le moteur principal de la voiture est piloté par PWM, comme la pulse qui est envoyée par le récepteur. La différence est que la pulse doit avoir une largeur qui varie de 1 à 2ms et le PWM du moteur va de 0 à 100%.

Si on regarde les données fournies par le fabricant concernant le contrôleur, on apprend que la fréquence du PWM du moteur est fixée à 1.5kHz. Cela est intéressant s'il faut changer le contrôleur. Etant donné que dans ce projet nous gardons celui déjà fourni, cela ne nous concerne pas. La fréquence de la pulse est quant à elle mesuré à environ 70Hz.

Le moteur peut avancer, ce qui est plutôt une bonne chose. Il peut aussi reculer, ce qui est encore mieux. Pour faire reculer le moteur, il faut envoyer sur le contrôleur une pulse qui va de 1.5ms à 1ms. La plage de 1.5ms à 2ms sert à faire avancer la voiture. Voici l'oscilloscopogramme qui nous montre comment cela fonctionne :

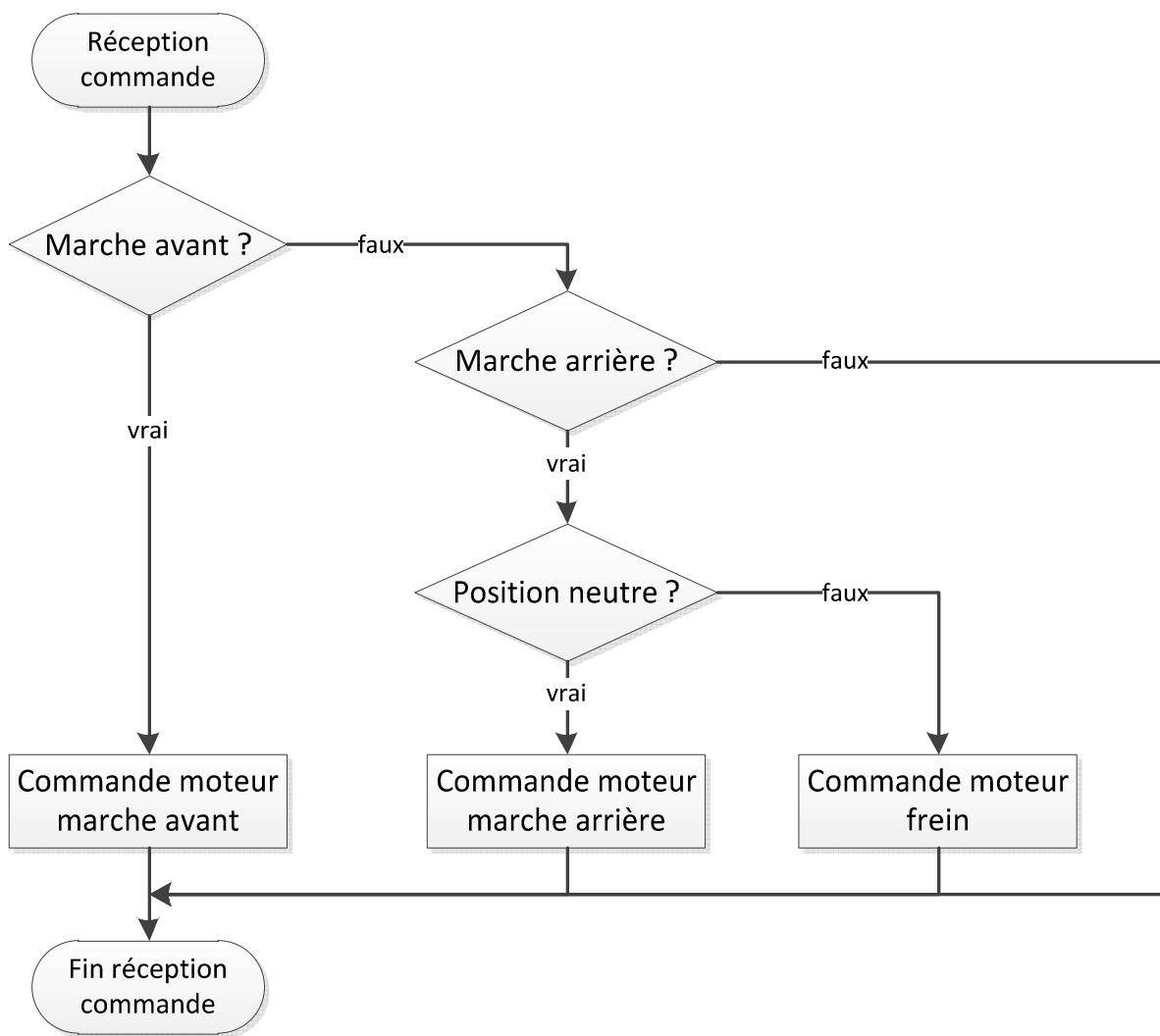


Canal 1 : La commande du récepteur

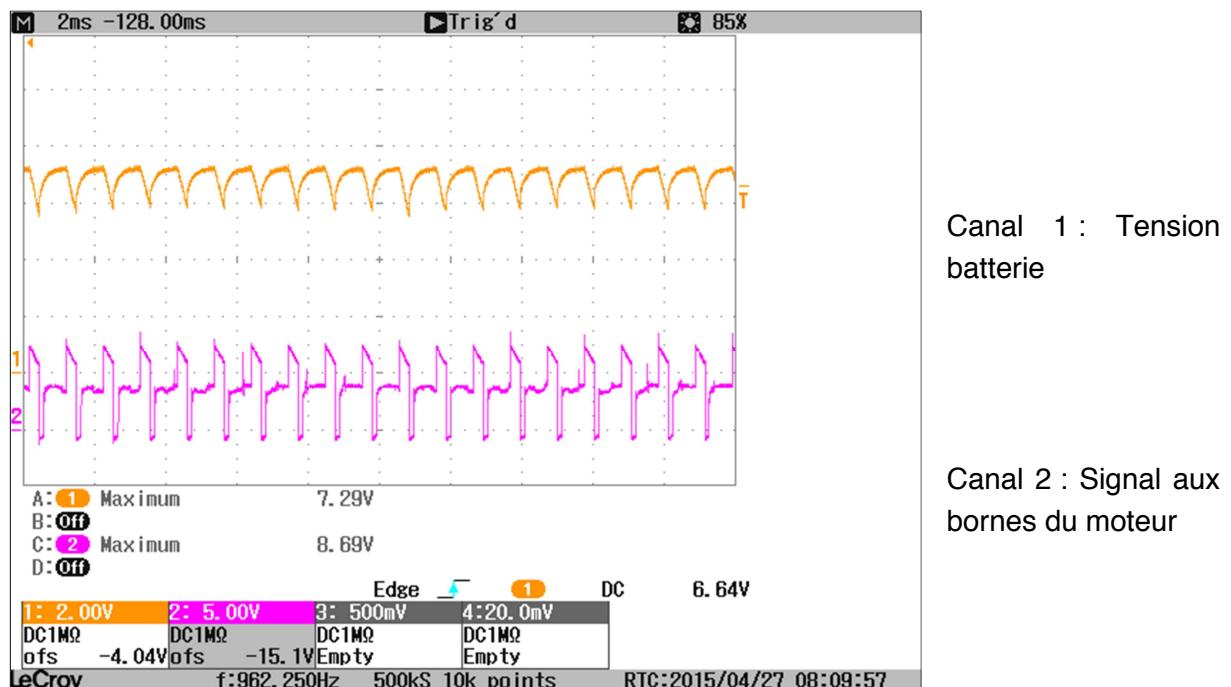
Canal 2 : La commande du contrôleur

Sur ce relevé, nous voyons bien que le moteur va en marche arrière, car la tension est négative. Le rapport cyclique est de 90%, mais en réalité il ne faut pas le regarder dans ce sens. C'est la tension négative qui pilote le moteur cette fois-ci. Ça veut dire que notre rapport cyclique est de 10% en réalité.

Toutefois, il y a une petite particularité avec le contrôleur : si le moteur est en marche avant et que l'on veut reculer, le moteur va d'abord freiner. Ensuite, il faut revenir à la position neutre de la télécommande, c'est-à-dire générer une pulse de 1.5ms qui correspond au point « neutre ». Une fois que le point neutre est passé, on peut remettre la marche arrière pour quitter le mode « frein » de la voiture. Voici un petit schéma explicatif :



Les deux mesures précédentes sont réalisées sans le moteur, à vide. Le moteur se comporte comme un circuit RL et sa consommation électrique est de quelques ampères. Cela a comme conséquence de provoquer beaucoup de bruit sur le moteur et sur la batterie comme le montre l'oscilloscopogramme suivant :

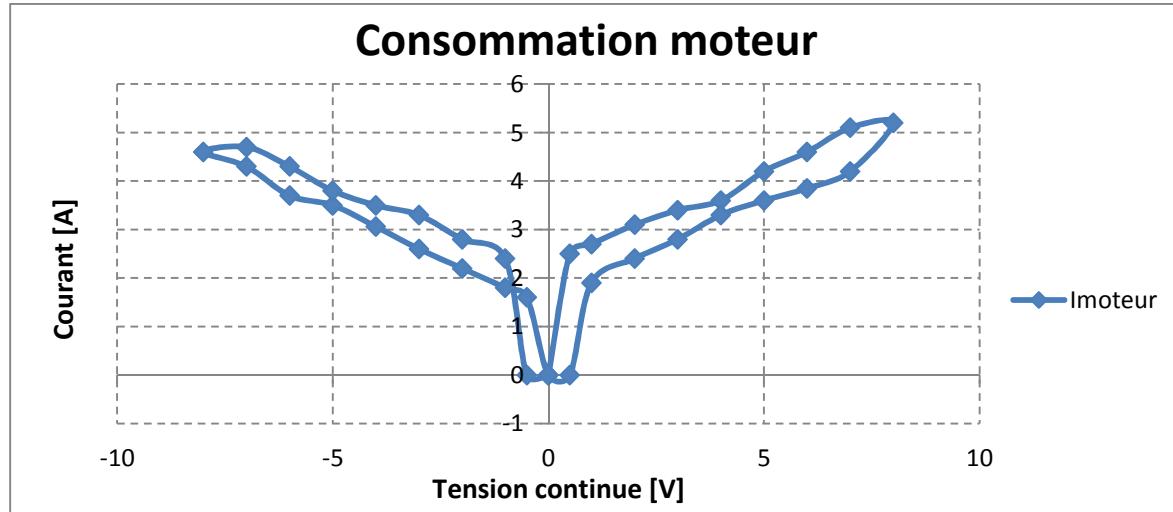


Comme on peut le voir sur cette mesure, le moteur provoque énormément de bruit sur la batterie. Il faudra faire attention à ce phénomène, car si on veut alimenter la carte fabriquée avec la batterie, il risque d'y avoir des problèmes.

Tout ceci nous prouve que le moteur est un élément relativement critique. Malgré le fait qu'on utilise un driver pour le faire fonctionner, il faut vraiment surveiller son fonctionnement, afin qu'il ne vienne pas perturber le circuit.

La dernière caractéristique qu'il faut relever sur le moteur est sa consommation. Pour s'y faire, on le branche sur une alimentation de laboratoire directement en DC et on monte sa tension, afin de voir le courant qui est débité. Cela nous donnera une idée de sa consommation en marche. Le problème étant que son contrôleur le pilote en PWM, du coup on n'aura pas exactement les mêmes caractéristiques en tension continu et en marche avec la voiture.

Voici une idée de ce que donne la consommation du moteur principale en courant continu (sur alimentation de laboratoire) :



Comme on peut le voir, la consommation du moteur est très variable. Il y a certainement des petites déviations sur les mesures. Cela dit, il ne faut pas oublier que c'est un moteur qui se comporte comme un circuit RL. En plus, il est relié à la mécanique ce qui provoque différentes frictions mécanique. Tous ces éléments mis ensemble provoquent forcément des variations de courant.

#### 5.3.4. La batterie

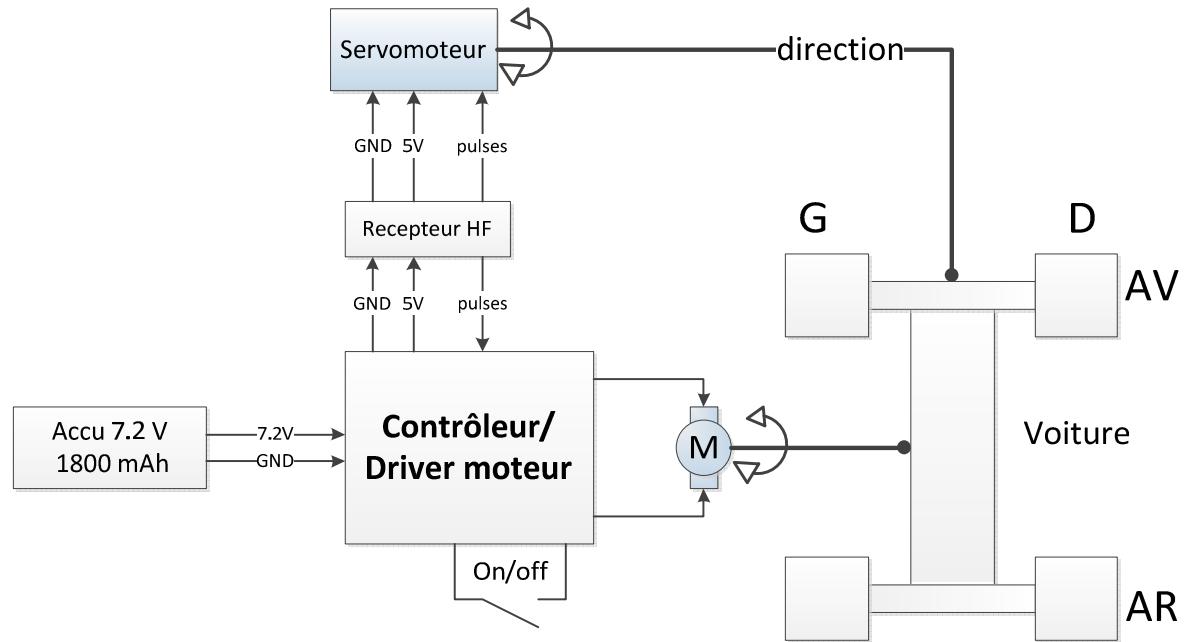
La batterie est fournie avec la voiture. C'est une batterie standard que l'on retrouve dans la plupart des modèles réduit de voitures. Elle est suffisamment puissante pour alimenter les deux moteurs et les différents éléments de la voiture. Elle possède une tension de 7.2 volts. Elle est branchée au driver du moteur qui à son tour fourni une tension de 5 volts pour le récepteur de la voiture. Voici la batterie :



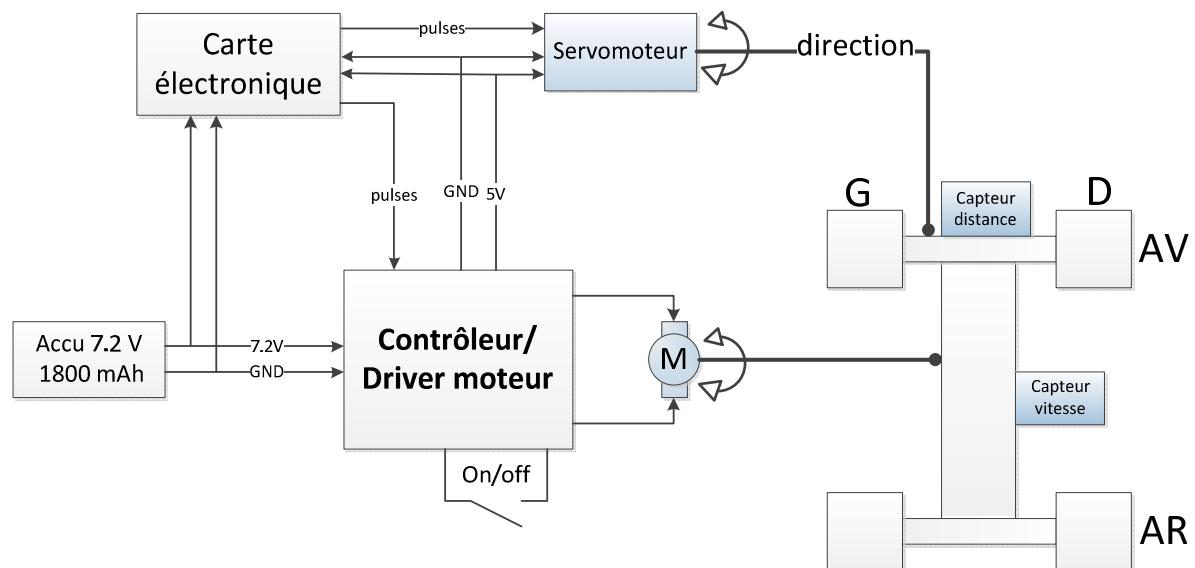
Etant donné qu'il y a énormément de bruit généré à cause des moteurs sur la batterie, nous allons certainement utiliser une alimentation différente pour la carte de commande. Ainsi, la carte ne sera pas affectée par tous les bruits générés par les composants de la voiture elle-même. Il faut tout de même effectuer des essais, afin de voir comment le système réagit.

## 6. Etude des différents éléments

### 6.1. Schéma bloc matériel d'origine de la voiture



### 6.2. Schéma bloc matériel modifié



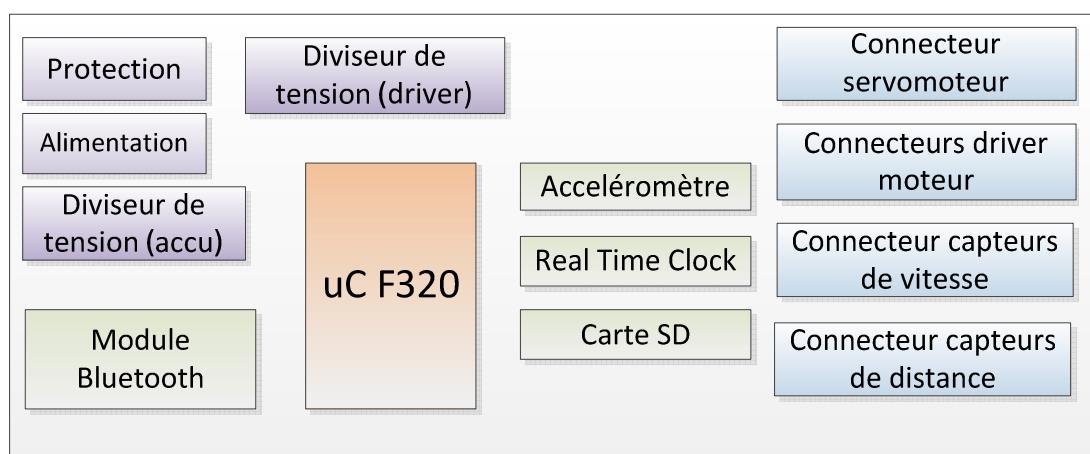
### 6.3. Introduction aux éléments

Comme dit précédemment, le but de ce projet est de remplacer le récepteur d'origine de la voiture par une carte électronique. Cette dernière sera dotée de différents éléments. Notamment, d'un microcontrôleur dont le rôle est de gérer toute la voiture. Le but étant de fabriquer un système anti-patinage, la voiture se verra munir d'un accéléromètre et d'un capteur de vitesse. Ces deux éléments sont indispensables, afin de pouvoir créer un système fiable qui empêche la voiture de patiner. Afin d'éviter l'endommagement du véhicule, on va lui greffer un capteur de distance qui permettra d'éviter les collisions avec les objets se trouvant devant la voiture. Au final, il faut pouvoir contrôler la voiture à distance, c'est pour cela qu'un module Bluetooth sera intégré à la carte électronique.

Deux éléments seront en options, c'est une carte SD et un Real Time Clock. Le premier servira au stockage de données et le deuxième de base de temps qui peut être exploité pour divers usage. Ces deux composants sont prévus sur la carte électronique et leur mise en place est incertaine, à cause du temps réduit qui est alloué au projet.

Enfin, il ne faut pas oublier le smartphone. C'est lui qui nous servira de télécommande. Une application Android dédiée au contrôle de la voiture sera réalisée.

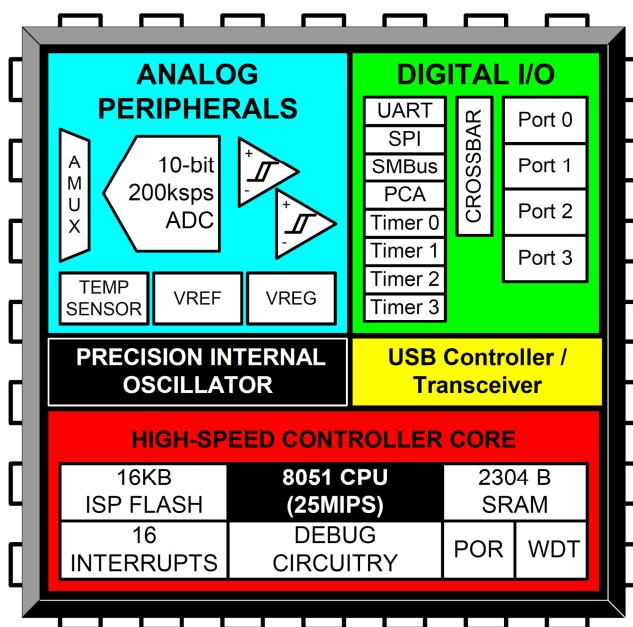
Voici le schéma-bloc matériel de la carte :



Le capteur de distance et de vitesse ne sont pas sur la carte, car l'un doit être placé devant le véhicule et le deuxième est situé au-dessus de l'arbre de transmission. Ils sont reliés tout simplement avec des connecteurs sur la carte. Les deux diviseurs de tension peuvent être considérés comme capteur de tension. Ils servent juste à réduire la tension de la batterie et la tension du driver moteur, afin qu'elles puissent être lues par le microcontrôleur.

### 6.3.1. Le microcontrôleur

La tête pensante de la voiture est le microcontrôleur. C'est un C8051F320 et il est fabriqué par Silicon Laboratories. C'est l'uC utilisé au CFPT. C'est celui-ci que l'on utilise lorsqu'on fait une production nécessitant un microcontrôleur. La tension d'alimentation est de 3.3 Volts, mais il y'a une entrée dédiée pour l'alimenter en 5 volts. Si on lui fournit cette tension, un régulateur interne pour ensuite créer une tension stable de 3.3 V et qu'on peut exploiter pour alimenter d'autres éléments. Il possède différernt périphériques internes permettant de faire des communications, des commandes, des signaux PWM, etc. Voici une image qui montre comment il est constitué :



Le but n'étant pas de faire une explication complète sur le microcontrôleur, son utilisation est expliquée brièvement. Certains périphériques internes ont des contraintes. Par exemple, la communication série peut être exploité uniquement sur le port 0. Il est suffisamment puissant pour être exploité et il peut fonctionner jusqu'à 24MHz. Voici un tableau récapitulatif des périphériques utilisés :

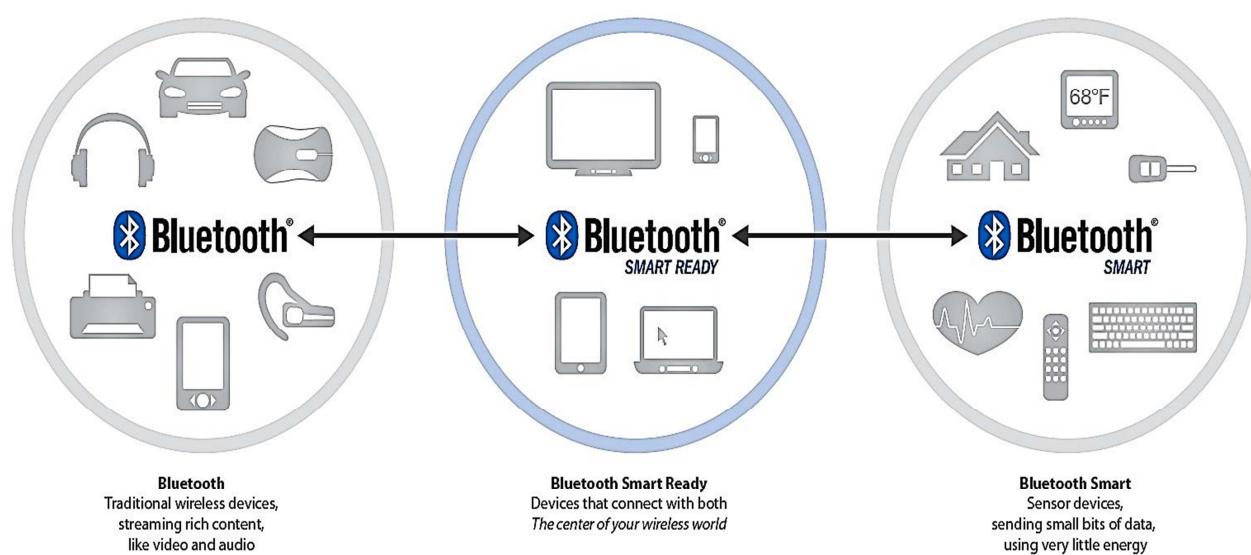
Périphérique interne	Utilisation	fondction
UART	oui	communication bluetooth
SPI	oui	communication carte SD et RTC
TIMER 0	oui	mesure vitesse
TIMER 1	oui	horloge UART (115'200bauds)
TIMER 2	oui	base de temps
TIMER 3	oui	Timeout UART
PCA_CEX0	oui	commande servomoteur
PCA_CEX1	oui	commande moteur principale
ADC0	oui	mesure accélération, distance, tension batterie et tension driver

### 6.3.2. Le module Bluetooth

La communication Bluetooth est un protocole de communication radio. Il permet d'échanger des données de façon bidirectionnelle. La bande utilisée est l'UHF. C'est-à-dire les Ultras Hautes Fréquences qui sont dans une bande comprise entre 300 MHz et 3GHz. Le but de cette liaison est de simplifier les liaisons filaires et de s'en passer. En effet, les appareils exploitant une liaison filaire, peuvent depuis l'apparition du Bluetooth utiliser une bande radio. Ceci permet de simplifier les communications et bien sûr, de ne plus utiliser de câbles.

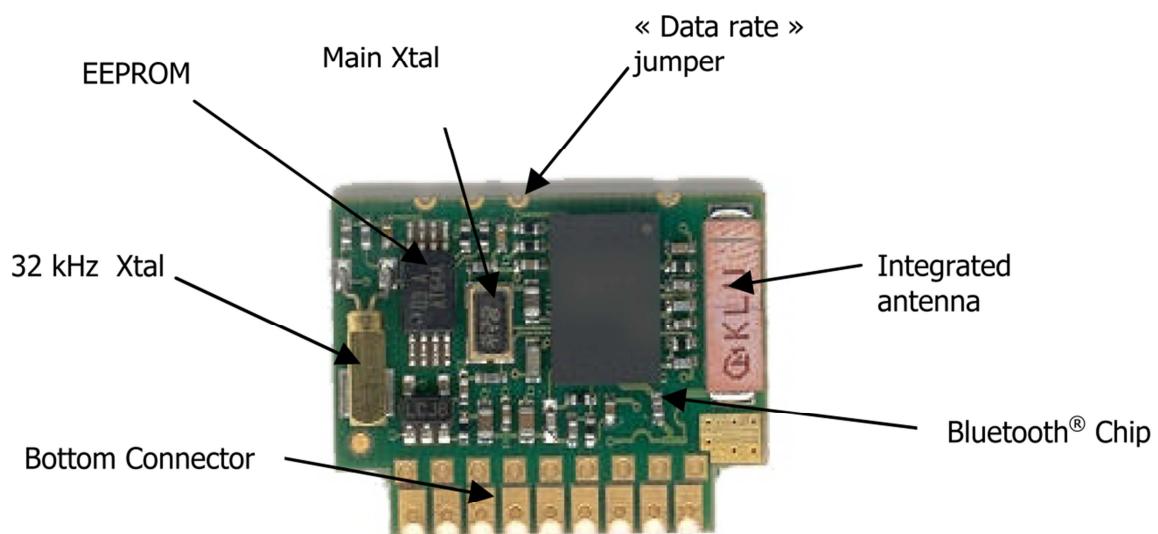
La spécification Bluetooth 1.0 est sortie en 1999. Cela dit, la communication a été créé en 1994 par la société Ericsson. Ce système radio peut transmettre des données à courte distance. C'est-à-dire que les données peuvent être échangées jusqu'à environ 20 mètres en extérieures. Il paraît même qu'en théorie, ça peut monter jusqu'à 100 mètres sans obstacles. Depuis, plusieurs versions sont sortis augmentant ainsi débit, portée, consommation, sécurité, etc. Les éléments fondamentaux de la communication sont définis dans les deux premières couches protocolaires : la couche radio et la couche bande de base. Certaines bandes radio sont partagées avec le protocole Wifi. Ce protocole a été rendu populaire notamment grâce aux téléphones mobiles.

Depuis l'avènement de la spécification 4.0, le Bluetooth se généralise notamment dans les objets connectés. Voici une image qui résume l'utilisation de cette spécification :



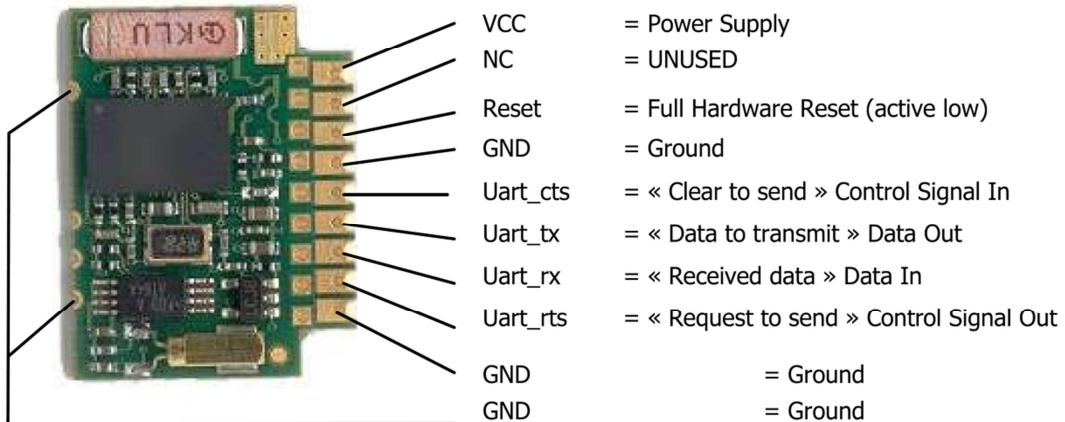
Voici le module : ARF7044. Ce dernier est fabriqué par Adeunis RF, une société française. Le module exploite la spécification 2.0 du protocole Bluetooth et possède un composant UART. Le module permet l'interfaçage entre une communication Bluetooth et une communication série. C'est exactement pour cela qu'il a été choisi, car cela permettra au microcontrôleur et au smartphone de communiquer. Le module a été proposé par le professeur accompagnant et validé par l'élève, car il correspond tout à fait à l'usage désiré. Il faut juste faire attention à une chose : le module est obsolète. C'est assez compliqué de trouver de la documentation pertinente.

L'ARF7044 est prévu pour être directement utilisable. Il suffit de l'alimenter et se connecter dessus via Bluetooth. Une fois que la connexion à lieu, on peut communiquer sans autres. Le module devient « transparent » aux yeux des périphériques qui sont connectés dessus. La vitesse de transmission sans fil peut atteindre les 723 kbps. Une antenne est intégrée au module. Ce dernier possède aussi une EEPROM et deux sources de clocks sont disponibles :



Il faut cependant faire attention à une chose, la connectique. On peut l'utiliser en SMD, mais ce n'est pas très pratique, si on souhaite le démonter et le réutiliser. Il convient de mettre une barrette, afin de pouvoir le monter et le démonter facilement. La barrette doit avoir un espacement de 2 mm. Malheureusement, le module ne possède pas une connectique très répandue, donc attention à cela.

Voici un résumé des caractéristiques du module Bluetooth :



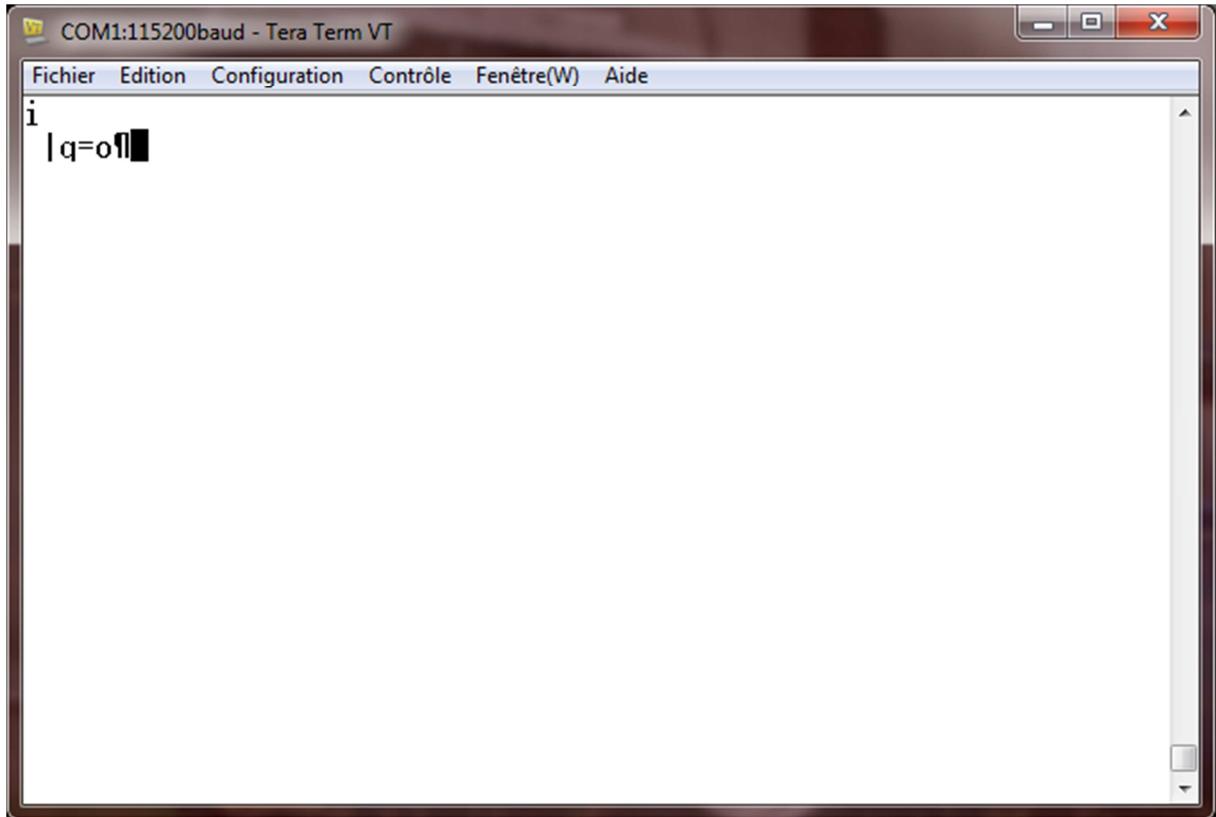
## Interface

### Pin description

Signal	I/O	Designation	Comment
VCC	I	Main power supply	2.85 < VCC < 3.6 V and I < 65 mA
NC	-	Not Connected	NOT TO BE USED
Reset	I	Hardware reset	A <sup>RF32</sup> reset when Low
Uart_cts	I	Clear to send Signal	Serial port Flow control Input <b>(MUST BE USED)</b>
Uart_tx	O	Data to transmit	Serial port Data Output (0/Vcc level)
Uart_rx	I	Received data	Serial port Data Input (0/Vcc level)
Uart_rts	O	Request to send Signal	Serial port Flow control Output <b>(MUST BE USED)</b>
GND	-	Common Ground	Connected to motherboard ground

Au niveau de la communication entre le module et le microcontrôleur, on passe par un UART. Celui-ci nous permet de mettre en place une communication série. Une trame UART est constituée d'au moins un bit de <START>, de <DATA> et d'un bit de <STOP>. Un contrôle de flux peut être utilisé au besoin, grâce aux lignes CTS et RTS. Le niveau logique de repos est à 1 et la vitesse de transmission est exprimée en bauds. C'est-à-dire en bits par seconde. Cette vitesse peut varier en fonction des besoins et elle correspond toujours à un multiple ou un sous-multiple de 9600 bauds. Cette dernière est la vitesse standard, couramment utilisée.

Dans la datasheet, on nous dit que le contrôle de flux doit être utilisé pour la communication série. Lorsqu'on teste la communication entre le PC, le module et une application de type terminal sur smartphone, voici ce qu'on obtient lors de la connexion :



Du côté du terminal sur ordinateur, on reçoit ces caractères lorsqu'un périphérique Bluetooth se connecte. C'est une bonne chose, car on pourra détecter la connexion et la déconnexion, la voiture pourra donc s'arrêter en fonction de la situation. Le module a déjà une configuration d'usine qui nous permet une connexion sur celui-ci et une utilisation dès son déballage. La configuration d'usine pour la communication série est avec un contrôle de flux matériel et une vitesse de 9600 bauds.

Pour la connexion entre le PC et le module, il faut passer par un convertisseur TTL-to-RS232, afin que la communication puisse fonctionner. À l'école nous possédons différents éléments, afin de réaliser une petite platine de test. C'est très pratique pour faire des tests rapidement.

Voici l'application de type terminal qui permet de faire un peu de débogage et surtout de tester la connexion Bluetooth :

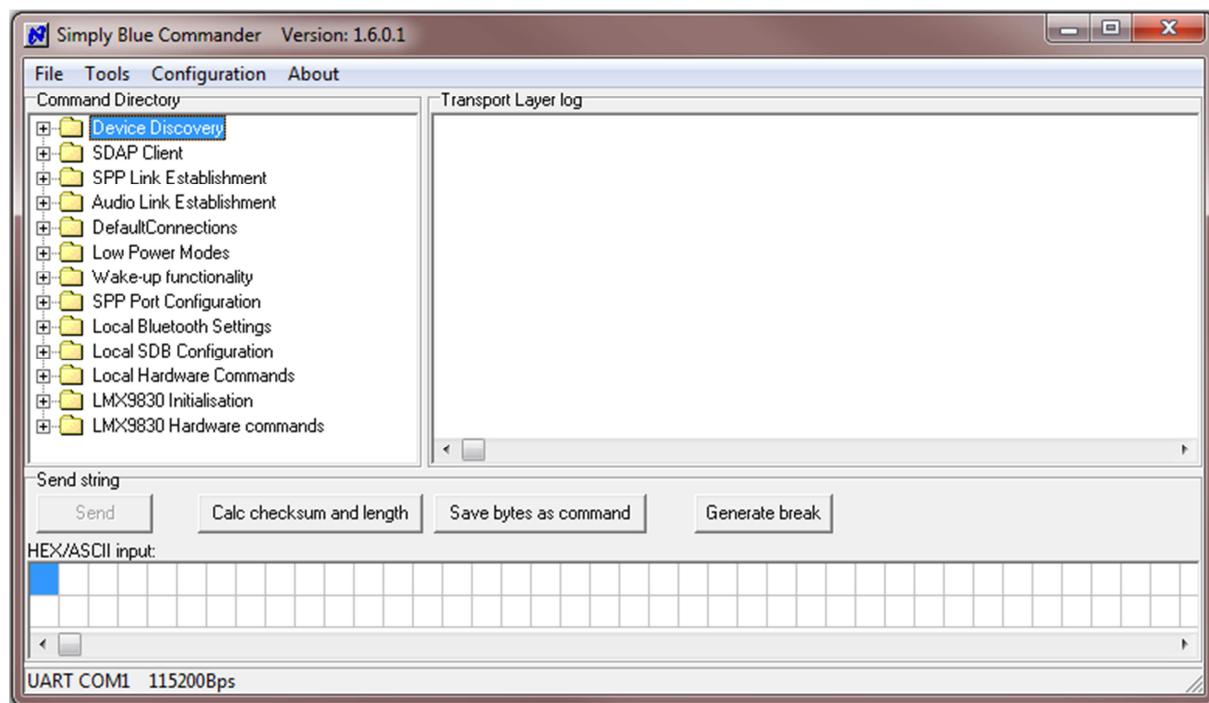


La connexion passe parfaitement et on peut communiquer entre cette application et l'ordinateur. Il faut juste que le module détecte les caractères '\r' et '\n', ce qui correspond à la touche « Enter » du clavier, pour envoyer la trame d'un point à l'autre.

De cette manière on peut communiquer parfaitement avec tous les éléments et de plus au besoin, on peut détecter d'éventuelles erreurs avec l'application terminale de l'ordinateur.

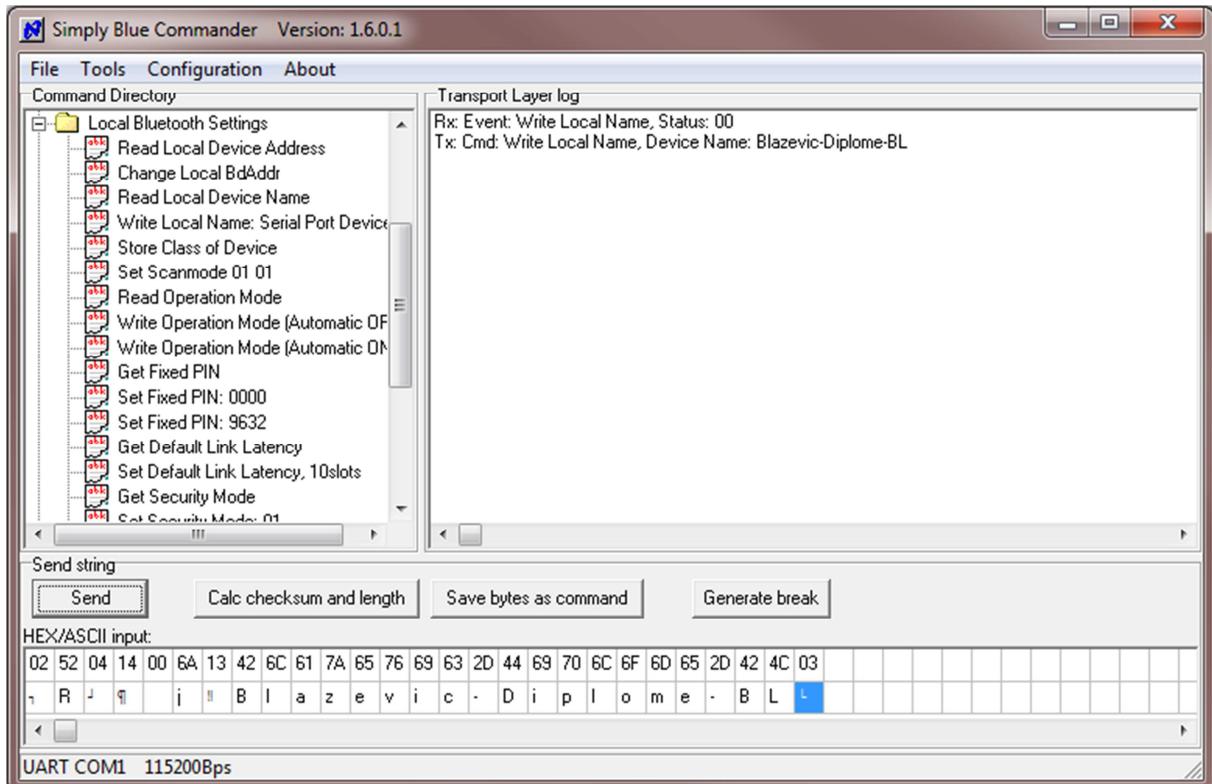
Si on configure le terminal sur l'ordinateur avec un contrôle de flux, on s'attend à voir les lignes CTS et RTS bouger. En réalité ce n'est pas le cas. Le module Bluetooth attend bien l'activation de sa ligne CTS, mais lorsqu'il veut envoyer des données, il n'active pas sa ligne RTS. Après avoir mesuré ces lignes, il s'avère que le fonctionnement est assez étrange. Etant donné que le temps alloué au projet était assez court, il a fallu prendre une décision rapidement, afin de continuer le projet. C'est pour cela qu'il faut faire un « by-pass » sur la ligne CTS du module Bluetooth. Cela suffira à faire fonctionner le système correctement. Cependant ce n'est pas très propre et le manque de temps ne permet pas une étude approfondie du problème. Pour du prototypage cela suffit. Pour une éventuelle commercialisation, il faut évidemment régler ce problème.

Le dernier point à soulever est la configuration du module. La datasheet n'est pas très explicite malheureusement. Après plusieurs recherches, il s'avère qu'un logiciel permet de configurer le module : « Simply Blue Commander Software ». C'est une petite application permettant la configuration de modules Bluetooth. Etant donné que les informations sont dures à trouver, c'est compliqué d'expliquer la provenance et l'utilité de ce programme. Cela dit, la datasheet du module ARF7044 en fait référence et après quelques recherches, le logiciel semble venir de National Semiconductor. Voici à quoi ressemble le logiciel :



Comme on peut le voir, il nous permet de faire diverses configurations de notre module Bluetooth. On peut configurer par exemple, la vitesse de l'UART, le nom du module, etc.

Le logiciel nous permet des réglages plus ou moins approfondis du module. Dans notre cas, on ne veut rien faire de particulier, mise à part changer le nom du module, afin qu'on le reconnaisse facilement lors d'une recherche Bluetooth comme suit :



Pour s'y faire, il suffit de cliquer dans le menu à gauche « Write Local Name : Serial Port Device » et changer le nom dans les carrés se trouvant en bas, en changeant les caractères un à un. Une fois que la modification est faite, il faut d'appuyer sur « Send » en bas à gauche et le module nous confirme le changement.

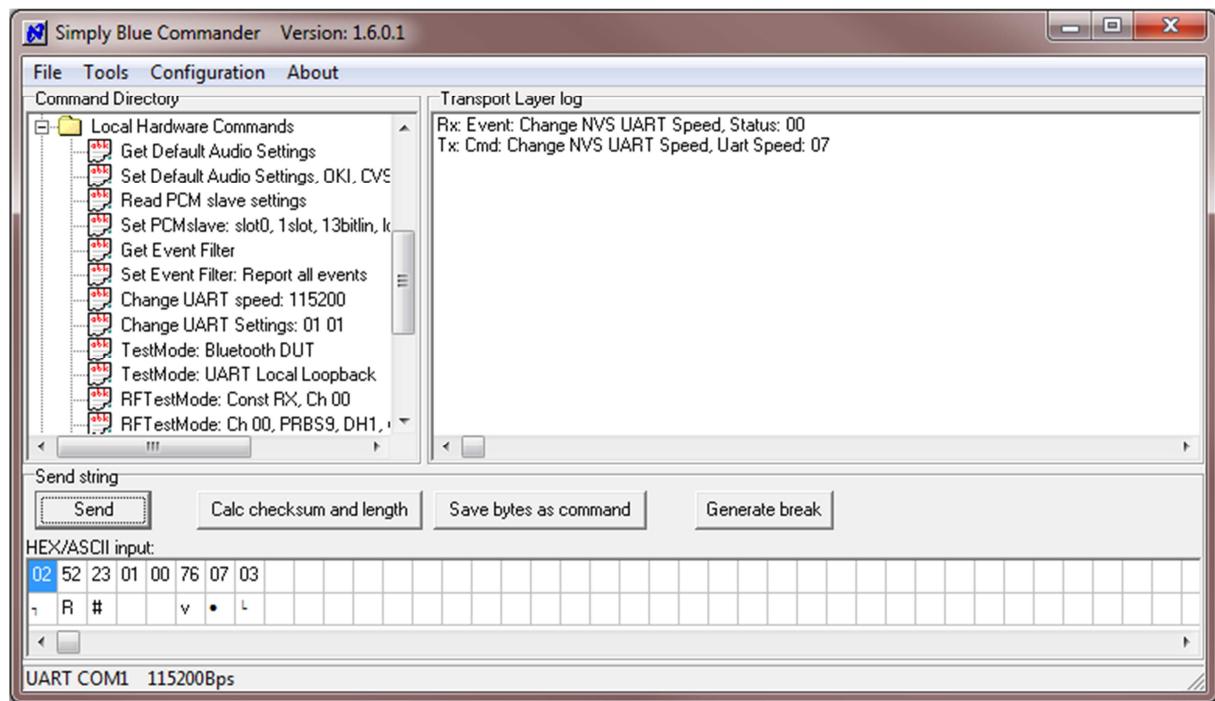
Pour trouver la documentation du logiciel, il faut chercher un peu sur internet. Pas très facile à trouver, car certainement obsolète. Quoi qu'il en soit, il existe un document : « Simply Blue Software User Guide » qui explique les différentes commandes et les différentes configurations que l'on peut faire de notre périphérique. De nouveau, il n'y a pas suffisamment de temps pour étudier la chose, il faudra passer quelques semaines dessus, afin de comprendre exactement ces divers éléments.

Le but étant de créer un lien direct entre le microcontrôleur et le smartphone. Le périphérique Bluetooth nous permet cela directement, sans chercher des réglages approfondis.

Une chose intéressante tout de même, serait de pouvoir changer la vitesse de communication de l'UART qui est un peu limitée, car elle est de 9600 bauds. Le logiciel nous permet de changer cette valeur directement, en la mettant à 115'200 bauds. La vitesse peut certainement être modifiée en mettant d'autres valeurs, mais celle-là a été choisie arbitrairement et est suffisamment rapide, afin de subvenir aux besoins de la voiture.

Avec cette vitesse, l'envoi d'une trame complète possédant 11 caractères se fait en moins d'une milliseconde. C'est suffisant pour contrôler la voiture.

Voici comment on configure la vitesse de l'UART :



Il suffit de cliquer dans le menu déroulant sur « Change UART speed : 115'200 ». Le module nous renvoi une confirmation et c'est fini.

Pour conclure sur le module, nous avons vu comment configurer le nom et la vitesse. Ce sont les deux éléments qui nous intéressent principalement, car le reste n'a pas été étudié et n'est pas forcément utile pour le déroulement du projet. Maintenant que la configuration est faite, le module est totalement exploitable. Il faut se connecter dessus et une fois que c'est fait, la communication est totalement transparente que ce soit du coter du smartphone ou du coter du microcontrôleur. Ce dernier n'a rien besoin de faire. C'est la communication sans fil qui doit être établie, alors que la liaison série est totalement libre d'accès.

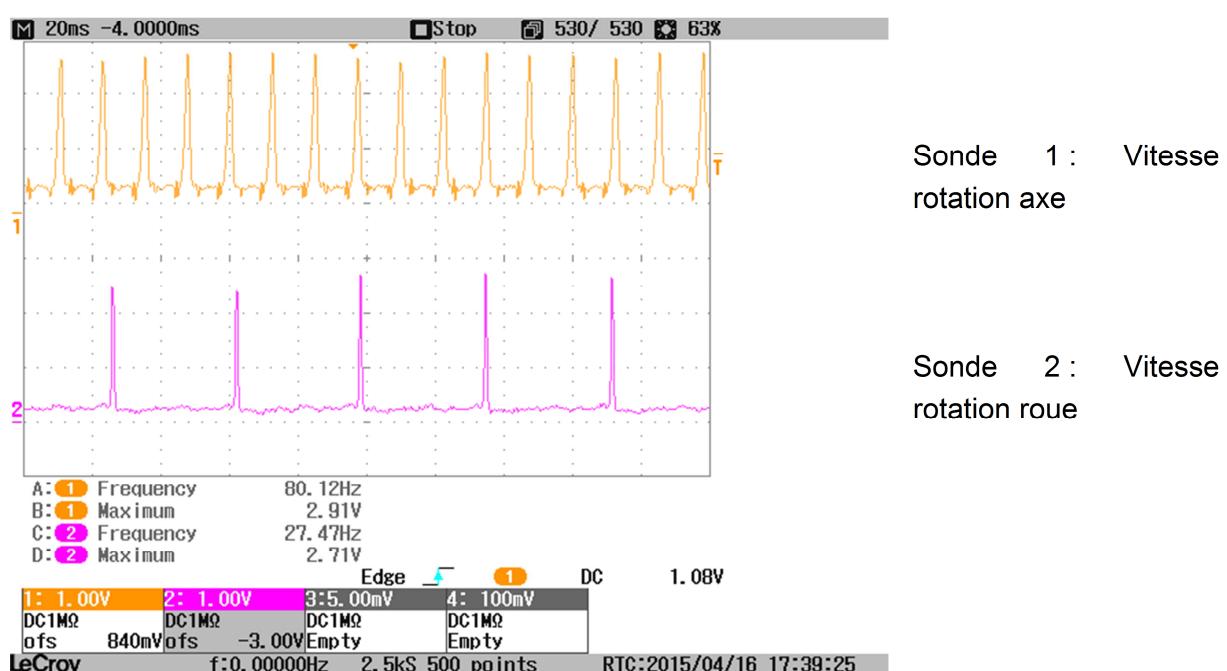
### 6.3.3. Le capteur de vitesse

Pour pouvoir empêcher le patinage de la voiture, il faut connaître la vitesse de rotation des roues. Pour cela, nous avons impérativement besoin d'un capteur de vitesse. En première année de technicien, nous avions utilisé un capteur de couleur, afin qu'un robot puisse suivre une ligne tracée au sol. Le capteur est un zx-03 reflector. Etant donné qu'il a déjà été utilisé et qu'il est connu, c'est une bonne idée de l'utiliser.

Pour commencer, il faut trouver l'endroit où la vitesse sera mesurée. Le meilleur positionnement est au-dessus de l'axe de transmission comme suit :



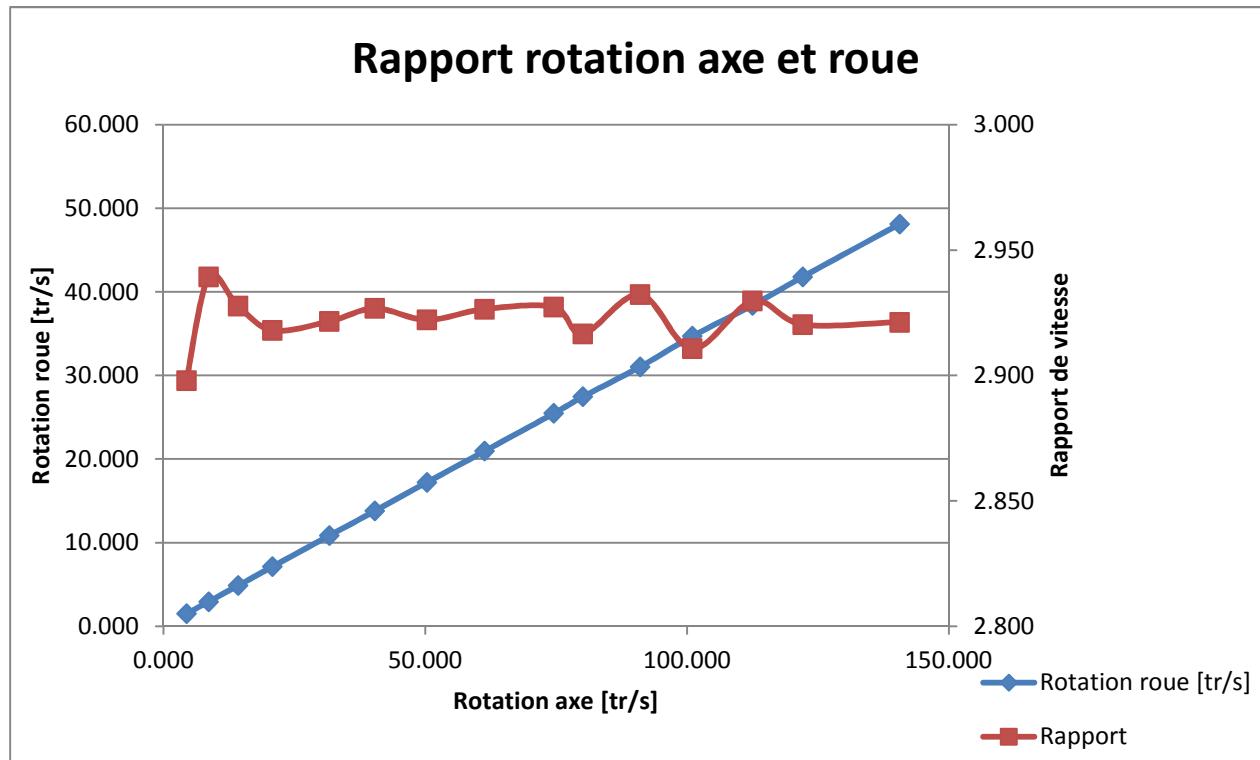
L'endroit où le capteur est positionné est noir. Il suffit de faire une marque blanche dessus, afin de voir apparaître un signal en sortie du capteur. Chaque fois que l'arbre de transmission fait un tour sur lui-même, une pulsation apparaît à l'oscilloscope. Pour connaître la vitesse exacte de la voiture, il faut mesurer le rapport de vitesse qu'il y a entre une roue et l'axe de rotation. Une fois que le rapport de rotation est connu, la vitesse de la voiture est connue. Voici un relevé d'oscilloscope du rapport de rotation



La mesure de la vitesse de rotation de la roue se fait avec le même capteur que pour la rotation de l'axe. Il suffit de faire une marque blanche sur la roue, afin qu'une pulse soit générée par le capteur à chaque passage de la marque devant celui-ci. Ainsi, nous connaissons avec exactitude la vitesse de rotation de la roue. Lorsqu'on enclenche la voiture et qu'on fait tourner les roues, on peut voir directement le rapport qu'il y a entre la rotation de l'axe et la rotation de la roue.

Les signaux ne sont pas très pertinents pour être envoyés directement dans le microcontrôleur. Il faut les traiter, afin d'obtenir de belles pulses bien carrés. Ce n'est pas forcément nécessaire, car l'uC peut normalement détecter les pulses (ceci n'a pas été testé).

Pour être sûr que l'on connaisse parfaitement le rapport de vitesse, il faut faire plusieurs mesures, ce qui nous permettra de calculer la moyenne et ainsi connaître la rotation des roues de la voiture en fonction de la rotation de l'arbre de transmission de celle-ci :



Une fois qu'on a relevé le rapport de vitesse plusieurs fois, on peut en déterminer la moyenne qui est de 2.923. Maintenant, on sait que lorsque la roue fait un tour sur elle-même, l'axe de transmission de la voiture en fait 2.923.

Le guide d'utilisation de la voiture nous donne comme diamètre de roue 62.1. Grâce à cela on peut calculer le périmètre :

$$\text{Périmètre} = \pi * D = 3.14 * 62.1 = 195.1 \text{ mm}$$

Maintenant, nous savons que lorsque l'axe a fait 2.923 tours, la voiture a avancée de 195.1mm. Pour déterminer la vitesse de la voiture, il suffit de multiplier le périmètre de la roue avec sa vitesse de rotation. Ainsi, pour une vitesse de l'axe de 100 tours par seconde par exemple, nous obtenons une vitesse de rotation de la roue de 34.21 tours/seconde. Ainsi la vitesse de rotation de la voiture se calcule comme suit :

$$\text{Vitesse voiture} = \frac{\text{Rotation axe}}{\text{rapport de vitesse}} * \text{Diamètre roue} = \frac{100}{2.923} * 195.1 = 6674.65 \text{ mm/s}$$

Ensuite, il ne reste plus qu'à convertir cette valeur en kilomètres par heure :

$$\text{Vitesse voiture} = \frac{6674.65}{1000} * 3.6 = 24 \text{ km/h}$$

Si on veut faire un anti-patinage suffisamment précis, il faut rajouter des coches sur l'axe de transmission. Si on ne le fait pas, on perd en précision et les roues risquent de patiner légèrement au démarrage de la voiture. Le temps qu'une coche soit détectée, la roue a déjà avancée de 66.75 millimètres. Si on fait 4 coches sur l'axe de rotation, d'une coche à l'autre, la roue aura parcouru 16.7mm. Comme cela nous pourrons plus facilement mettre en place un système anti-patinage. De plus le système sera bien plus réactif.

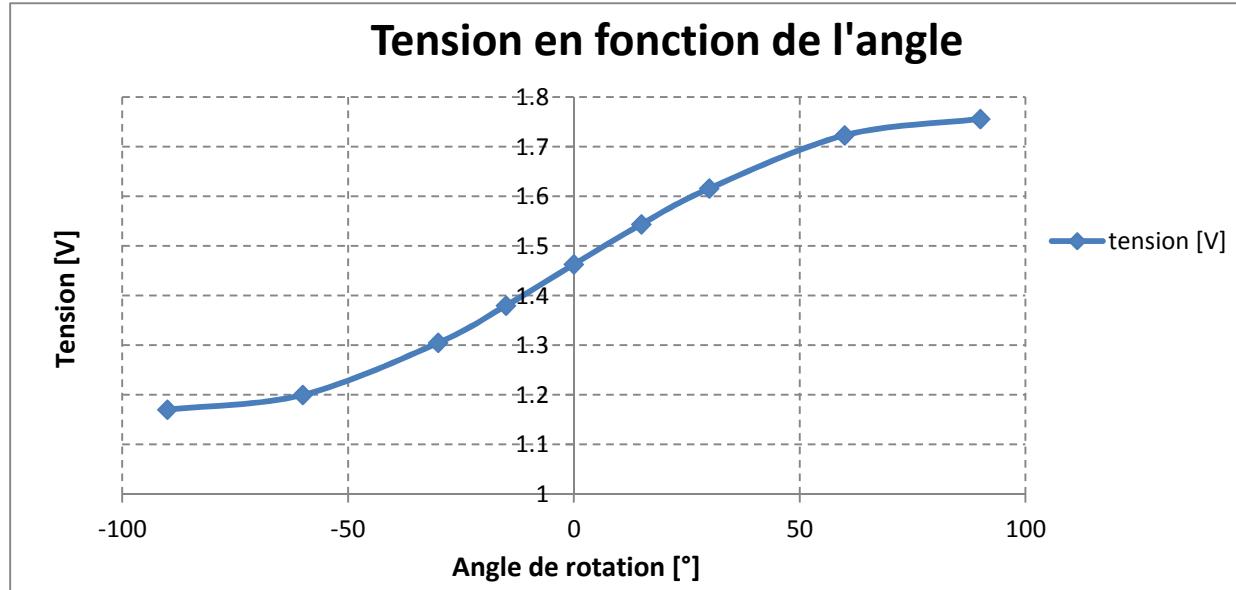
Maintenant que nous connaissons la vitesse de rotation de l'axe, des roues et de la voiture, nous pourrons attaquer l'étude de l'accéléromètre. C'est grâce aux données recueillies par le capteur de vitesse de l'accéléromètre que nous pourrons déterminer si la voiture est entrain de patiner ou non.

### 6.3.4. Accéléromètre

L'accéléromètre est le composant qui doit être utilisé pour mettre en place le système d'anti-patinage. Le but de celui-ci est de mesurer une certaine accélération sur un axe. On peut mesurer l'accélération tout objet qui va d'un endroit à un autre. Le capteur utilisé dans ce projet est un ADXL335. Voici ces caractéristiques principales :

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		g
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity <sup>1</sup>			±1		%
SENSITIVITY (RATIO METRIC) <sup>2</sup>	Each axis				
Sensitivity at X <sub>OUT</sub> , Y <sub>OUT</sub> , Z <sub>OUT</sub>	V <sub>S</sub> = 3 V	270	300	330	mV/g
Sensitivity Change Due to Temperature <sup>3</sup>	V <sub>S</sub> = 3 V		±0.01		%/°C
ZERO g BIAS LEVEL (RATIO METRIC)					
0 g Voltage at X <sub>OUT</sub> , Y <sub>OUT</sub>	V <sub>S</sub> = 3 V	1.35	1.5	1.65	V
0 g Voltage at Z <sub>OUT</sub>	V <sub>S</sub> = 3 V	1.2	1.5	1.8	V
0 g Offset vs. Temperature			±1		mg/°C
NOISE PERFORMANCE					
Noise Density X <sub>OUT</sub> , Y <sub>OUT</sub>			150		µg/√Hz rms
Noise Density Z <sub>OUT</sub>			300		µg/√Hz rms
FREQUENCY RESPONSE <sup>4</sup>					
Bandwidth X <sub>OUT</sub> , Y <sub>OUT</sub> <sup>5</sup>	No external filter		1600		Hz
Bandwidth Z <sub>OUT</sub> <sup>5</sup>	No external filter		550		Hz
R <sub>FILT</sub> Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz
SELF-TEST <sup>6</sup>					
Logic Input Low			+0.6		V
Logic Input High			+2.4		V
ST Actuation Current			+60		µA
Output Change at X <sub>OUT</sub>	Self-Test 0 to Self-Test 1	-150	-325	-600	mV
Output Change at Y <sub>OUT</sub>	Self-Test 0 to Self-Test 1	+150	+325	+600	mV
Output Change at Z <sub>OUT</sub>	Self-Test 0 to Self-Test 1	+150	+550	+1000	mV
OUTPUT AMPLIFIER					
Output Swing Low	No load		0.1		V
Output Swing High	No load		2.8		V
POWER SUPPLY					
Operating Voltage Range		1.8		3.6	V
Supply Current	V <sub>S</sub> = 3 V		350		µA
Turn-On Time <sup>7</sup>	No external filter		1		ms
TEMPERATURE					
Operating Temperature Range			-40	+85	°C

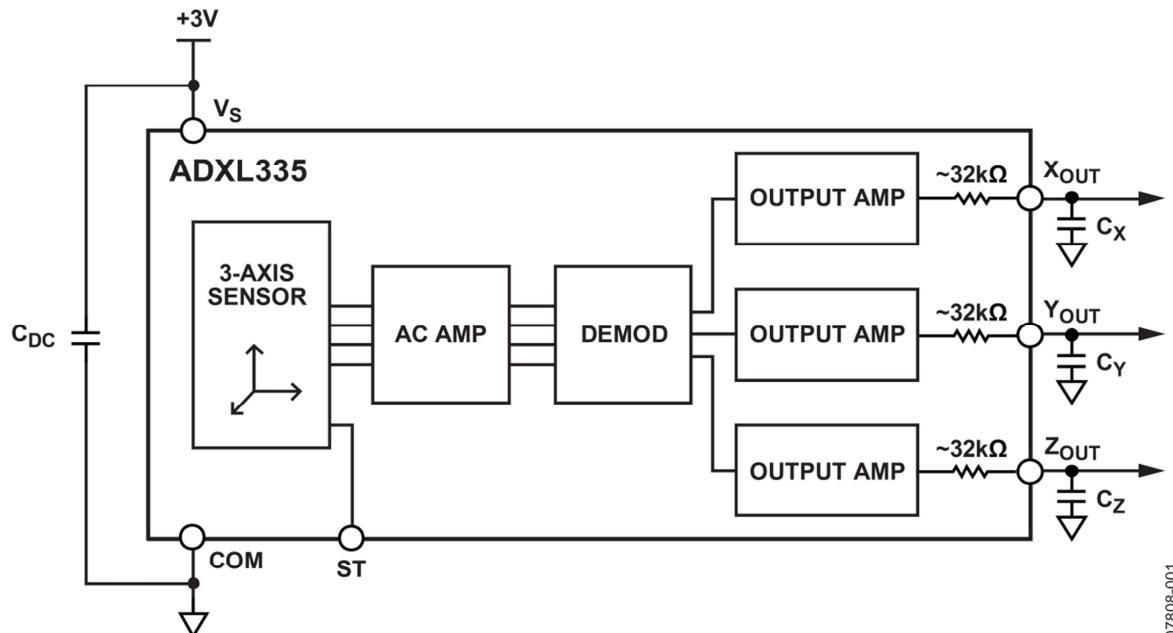
Des mesures d'angles sont faites sur l'accéléromètre, afin qu'on puisse se rendre compte de l'impact de l'accélération terrestre sur notre capteur. Voici ce qu'on retrouve à la sortie de l'accéléromètre en fonction de la valeur de l'angle :



Comme on peut le constater, la gravité terrestre à bel et bien un impact sur notre accéléromètre. Il faudra faire attention aux mesures que l'on veut réaliser. Il faut faire des mesures d'accélération sur l'accéléromètre. Quelques tests sont réalisés à main levée, afin de voir quel signal est fourni par l'accéléromètre.

Il faut faire une étude complète de l'accéléromètre, afin de pouvoir l'exploiter. Il faut mettre aux sorties de celui-ci des condensateurs pour filtrer les signaux. Ensuite, il faut échantillonner les signaux.

## FUNCTIONAL BLOCK DIAGRAM



07808-001

### 6.3.5. Le capteur de distance

Faire rouler une voiture c'est très sympathique. Pouvoir l'arrêter si on n'y prête pas attention, c'est mieux. C'est pour cela que pour sécuriser la voiture, il faut mettre en place un petit capteur de distance qui va empêcher la voiture de foncer droit dans un mur. Le véhicule possède déjà une petite mousse à l'avant pour amortir les chocs. La mise en place d'un capteur de distance peut avoir plusieurs utilités. Premièrement arrêter la voiture. Une autre fonctionnalité serait d'éviter les obstacles, par exemple. Le capteur retenu pour ce projet est un GP2D120, car il a déjà été utilisé auparavant. Voici un aperçu de ses caractéristiques principales :

#### ELECTRICAL SPECIFICATIONS

##### Absolute Maximum Ratings

$T_a = 25^\circ\text{C}$ ,  $V_{CC} = 5 \text{ VDC}$

PARAMETER	SYMBOL	RATING	UNIT
Supply Voltage	$V_{CC}$	-0.3 to +7	V
Output Terminal Voltage	$V_O$	-0.3 to ( $V_{CC} + 0.3$ )	V
Operating Temperature	$T_{opr}$	-10 to +60	$^\circ\text{C}$
Storage Temperature	$T_{stg}$	-40 to +70	$^\circ\text{C}$

##### Operating Supply Voltage

PARAMETER	SYMBOL	RATING	UNIT
Operating Supply Voltage	$V_{CC}$	4.5 to 5.5	V

#### Electro-optical Characteristics

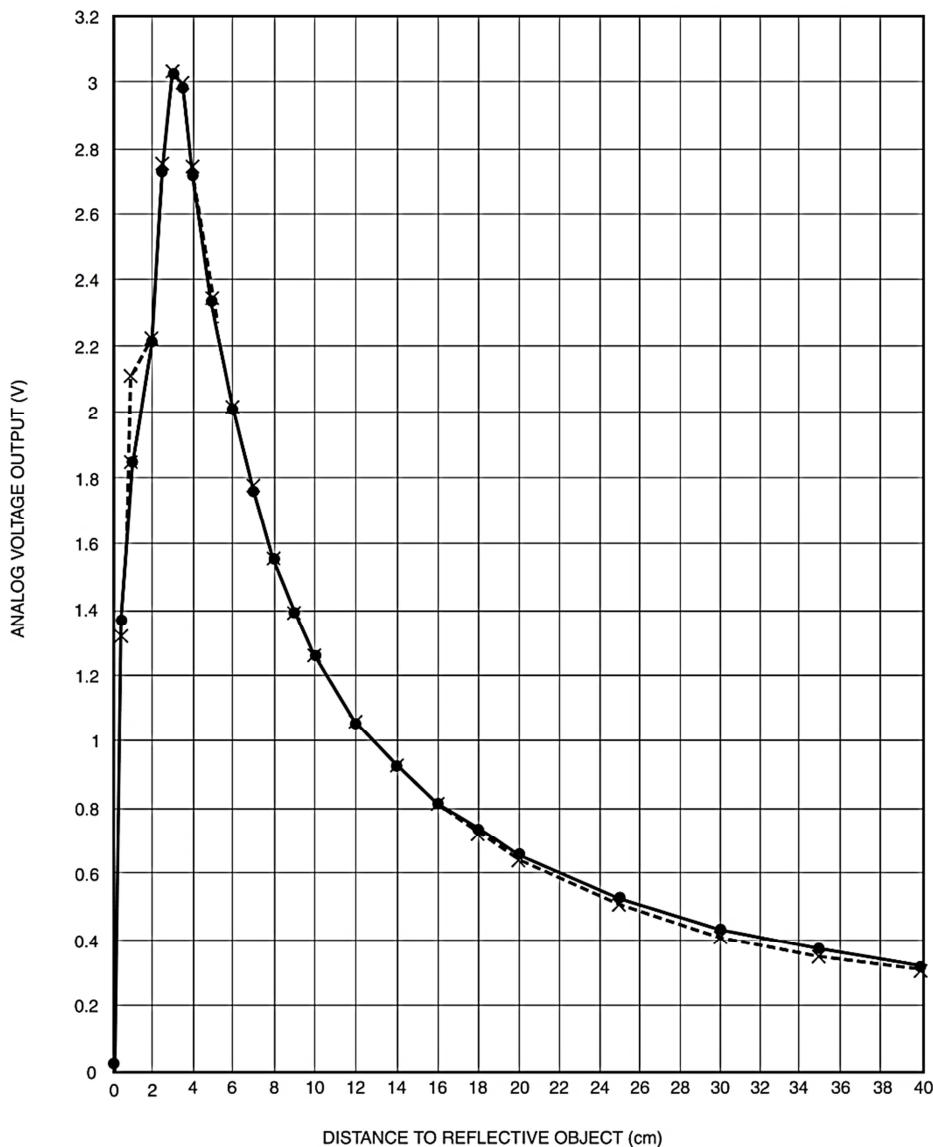
$T_a = 25^\circ\text{C}$ ,  $V_{CC} = 5 \text{ VDC}$

PARAMETER	SYMBOL	CONDITIONS	MIN.	TYP.	MAX.	UNIT	NOTES
Measuring Distance Range	$\Delta L$		4	—	30	cm	1, 2
Output Terminal Voltage	$V_O$	$L = 30 \text{ cm}$	0.25	0.4	0.55	V	1, 2
Output Voltage Difference	$\Delta V_O$	Output change at $\Delta L$ (30 cm – 4 cm)	1.95	2.25	2.55	V	1, 2
Average Supply Current	$I_{CC}$	$L = 30 \text{ cm}$	—	33	50	mA	1, 2

##### NOTES:

1. Measurements made with Kodak R-27 Gray Card, using the white side, (90% reflectivity).
2.  $L$  = Distance to reflective object.

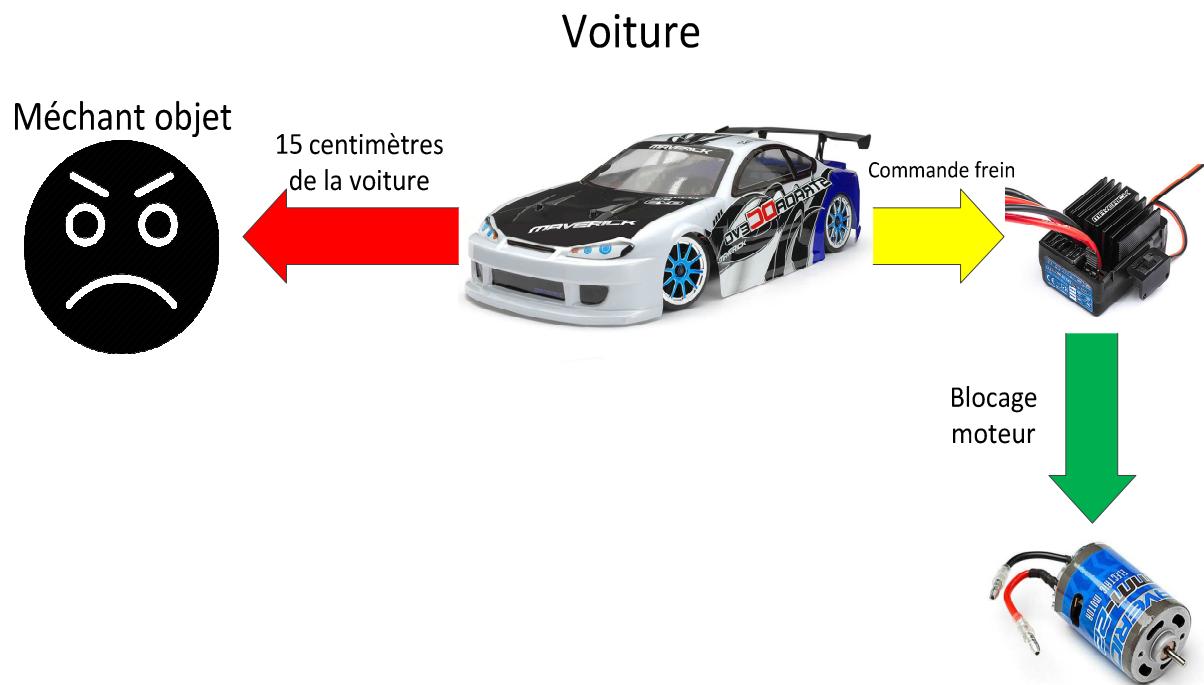
La mise en place du capteur de distance est relativement simple. On va se concentrer sur le fait de bloquer tout simplement les roues si un obstacle se trouve à moins de 15 centimètres de la voiture. Pour cela, voici la courbe de réponse du capteur :



La courbe de réponse est relativement simple, il n'y a pas besoin de faire de filtrage. Il faut tout de même faire attention si un objet se trouve sous la barre des 3 centimètres, car la courbe s'inverse. Une possibilité pour éviter un disfonctionnement est d'enregistrer dans le microcontrôleur l'ancienne valeur de la distance. A chaque fois qu'on vient faire une conversion AD, on compare la nouvelle valeur de la conversion avec l'ancienne. Ceci nous permettra de savoir si on se trouve dans la plage normale d'utilisation.

Pour faciliter, on peut tout simplement placer le capteur un peu en retrait sur la voiture, afin de ne jamais se trouver à une valeur de moins de 4 centimètres. C'est une méthode un peu archaïque, mais elle est fonctionnelle et on garantit qu'il n'y aura pas d'erreur dans la mesure. La coque de la voiture a dû être un peu percée pour accueillir le capteur, comme on peut le voir sur les images du produit finit. Ce n'est pas très élégant, mais il ne faut pas oublier que c'est un prototype, pour une éventuelle commercialisation, il faudra bien entendu faire les choses beaucoup plus proprement.

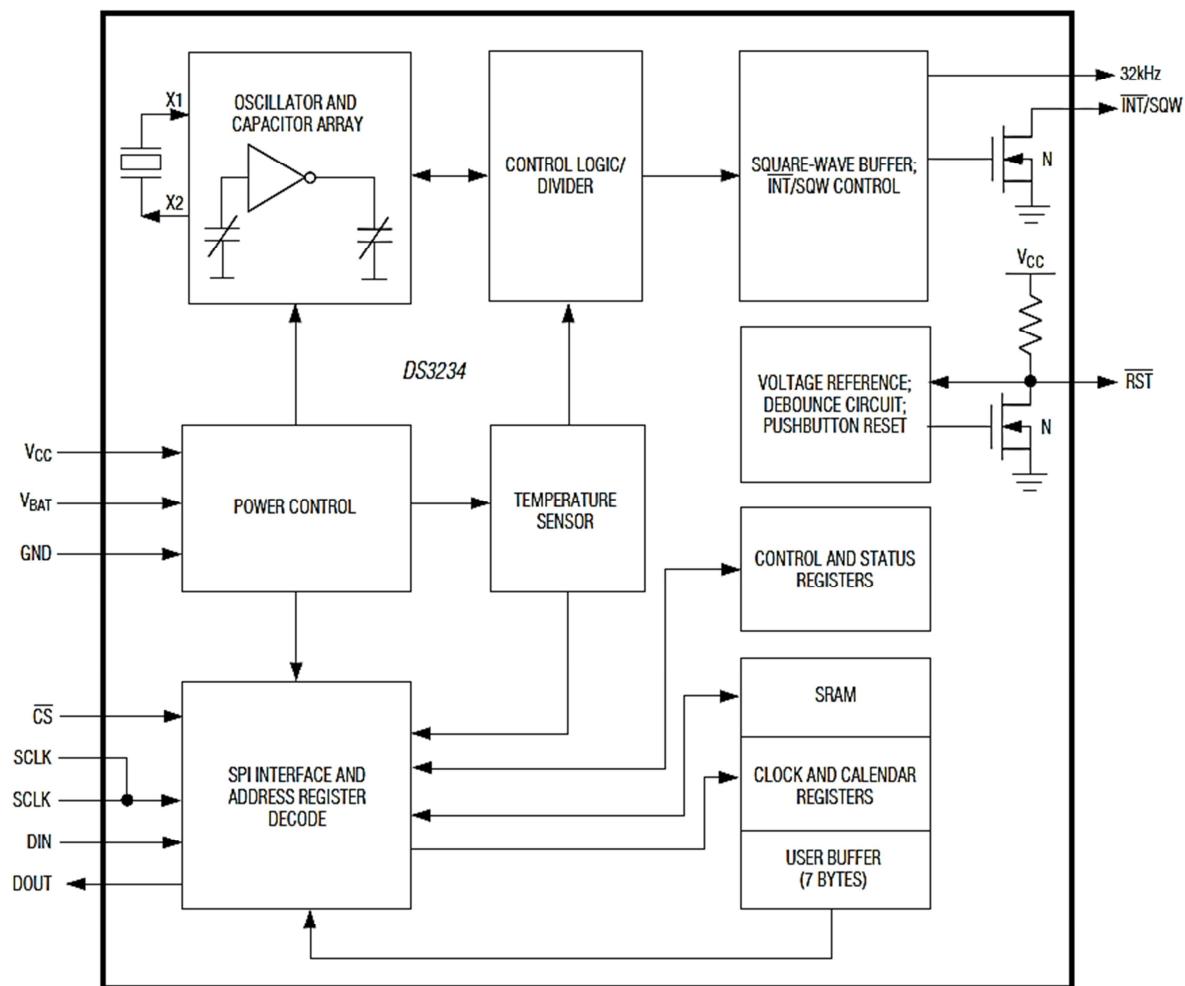
Le capteur se trouve à environ 7 centimètre de l'avant de la voiture. Donc si on veut que la voiture bloque les roues à 15 centimètres de distance, il faut rajouter ces 7 centimètres. Au final, la voiture s'arrêtera si elle détecte un objet à moins de 22 centimètres de la distance du capteur. On peut même ajouter de la distance par sécurité. Pour l'instant on laisse ces valeurs-là.



### 6.3.6. Real Time Clock

L'horloge est prévue sur la carte finale de la voiture, mais elle n'est pas utilisé. Elle est là pour ajouter des fonctionnalités éventuelles à la carte. Le RTC utilisé est un DS3234. Il est basique et il faut passer par une communication SPI pour communiquer avec lui. Il possède quelques fonctionnalités utiles comme une alarme ou bien un capteur de température incorporé.

Le but du Real Time Clock est d'avoir une base de temps pour diverses utilisations. On peut s'en servir pour connaitre le temps d'utilisation de la voiture, la date, la température de la carte électronique ou encore mettre en place une alarme pour diverses utilisations. Le but principal est de connaitre la date et l'heure auxquels on a stocké les données sur la carte SD. Voici le schéma interne du RTC :



Et voici les caractéristiques principales de ce composant :

## Recommended Operating Conditions

( $T_A = -40^\circ\text{C}$  to  $+85^\circ\text{C}$ , unless otherwise noted.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V <sub>CC</sub>		2.0	3.3	5.5	V
	V <sub>BAT</sub>		2.0	3.0	3.8	
Logic 1 Input CS̄, SCLK, DIN	V <sub>IH</sub>		0.7 x V <sub>CC</sub>	V <sub>CC</sub> + 0.3		V
Logic 0 Input CS̄, SCLK, DIN, RST	V <sub>IL</sub>	2.0V ≤ V <sub>CC</sub> ≤ 3.63V	-0.3	+0.2 x V <sub>CC</sub>		V
		3.63V < V <sub>CC</sub> ≤ 5.5V	-0.3	+0.7		

## Electrical Characteristics

(V<sub>CC</sub> = 2.0V to 5.5V, V<sub>CC</sub> = active supply (see Table 1), T<sub>A</sub> = -40°C to +85°C, unless otherwise noted.) (Typical values are at V<sub>CC</sub> = 3.3V, V<sub>BAT</sub> = 3.0V, and T<sub>A</sub> = +25°C, unless otherwise noted. TCXO operation guaranteed from 2.3V to 5.5V on V<sub>CC</sub> and 2.3V to 3.8V on V<sub>BAT</sub>.) (Notes 2, 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Active Supply Current	I <sub>CCA</sub>	SCLK = 4MHz, BSY = 0 (Notes 4, 5)	V <sub>CC</sub> = 3.63V		400	μA
			V <sub>CC</sub> = 5.5V		700	
Standby Supply Current	I <sub>CCS</sub>	CS̄ = V <sub>IH</sub> , 32kHz output off, SQW output off (Note 5)	V <sub>CC</sub> = 3.63V		120	μA
			V <sub>CC</sub> = 5.5V		160	
Temperature Conversion Current	I <sub>CCSConv</sub>	SPI bus inactive, 32kHz output off, SQW output off	V <sub>CC</sub> = 3.63V		500	μA
			V <sub>CC</sub> = 5.5V		600	
Power-Fail Voltage	V <sub>PF</sub>		2.45	2.575	2.70	V
V <sub>BAT</sub> Leakage Current	I <sub>BATLKG</sub>		25	100		nA
(V <sub>CC</sub> = 2.0V to 5.5V, T <sub>A</sub> = -40°C to +85°C, unless otherwise noted.) (Notes 2 and 3)						
Logic 1 Output, 32kHz I <sub>OH</sub> = -500μA I <sub>OH</sub> = -250μA I <sub>OH</sub> = -125μA	V <sub>OH</sub>	V <sub>CC</sub> > 3.63V, 3.63V > V <sub>CC</sub> > 2.7V, 2.7V > (V <sub>CC</sub> or V <sub>BAT</sub> ) > 2.0V (BB32kHz = 1)	0.85 x V <sub>CC</sub>			V

### **6.3.7. Carte SD**

Les cartes SD sont des dispositifs de stockage utilisé notamment dans les appareils mobiles comme des téléphones, des appareils photo, etc. Voici à quoi elles ressemblent :



Leur capacité peut varier de 2 à 64 Gb pour les cartes les plus couramment utilisées. Certaines cartes ont une capacité qui peut monter jusqu'à 512 Gb. Il existe trois formats de carte qui sont :

- Carte SD
- Carte miniSD
- Carte microSD

La différence se trouve notamment dans la taille de la carte.

Dans le cadre de ce projet, la carte SD n'est pas étudiée dû au manque de temps. Cependant, on sait que pour communiquer entre la carte et le microcontrôleur, on utilise une interface SPI, tout comme le Real Time Clock. Dans la carte de la voiture, un support pour carte SD est prévu et peut être exploité.

Le but de la carte est de stocker surtout les données d'utilisations de la voiture. Les données stockées seraient reprises sur un ordinateur et être interprétées. Les données stockées sont surtout les accélérations de la voiture, éventuellement les trajectoires ou encore niveau de la batterie.

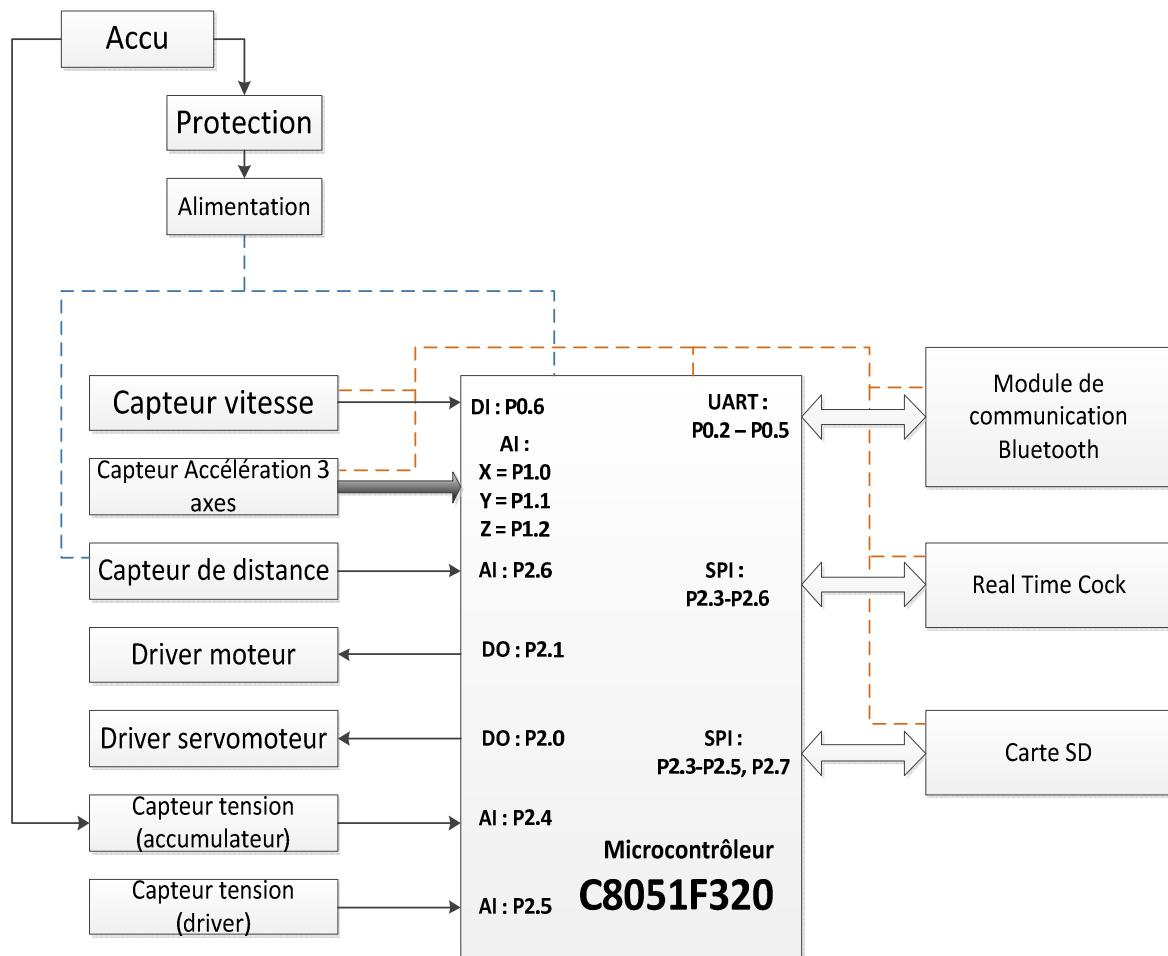
## 7. Conception et développement

### 7.1. Conception et développement matériel

Après avoir étudié tous les différents composants et les éventuelles stratégies à employer, il faut commencer à faire le développement de la carte avec les diverses contraintes physique, électrique, etc.

#### 7.1.1. Schéma bloc fonctionnel

Maintenant que nous connaissons le fonctionnement des composants et que le schéma bloc matériel est fait, on peut dresser un schéma bloc fonctionnel :



Le schéma bloc de la voiture prend en compte les diverses contraintes du microcontrôleur, notamment au niveau de ses ports.

### 7.1.2. Alimentation

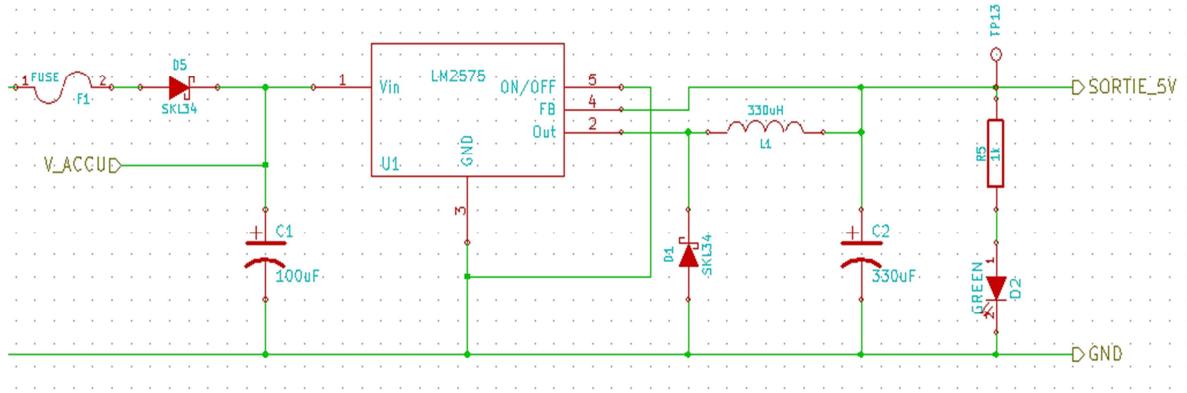
Avant de pouvoir faire le schéma complet, il faut mettre en place certaines parties du schéma qui sont délicates. Pour commencer : l'alimentation. Il faut prévoir une alimentation en fonction des besoins des divers modules que l'on utilise. Voici un récapitulatif des besoins de ceux-ci :

Composants	Tension d'alimentation [V]	Consommation max de courant [mA]
Microcontrôleur C8051F320	3,3 - 5	10
Module Bluetooth ARF7044	3.3	65
Accéléromètre ADXL335	3.3	0.35
RTC DS3234	3,3 - 5	0.4
Capt. de vitesse	3,3	-
LM393	3,3	1
Capt. de distance GP2D120	5	50
Carte SD	3,3	-
<b>Total :</b>		126.75

Pour la carte SD et le capteur de vitesse, la consommation est indéterminée, mais elle est probablement faible. Etant donné que la majorité des circuits sont compatible 3.3V et que le microcontrôleur possède un régulateur interne, il est préférable de faire une alimentation de 5 volts et d'alimenter les modules qui en ont besoin en 3.3V via le F320. L'alimentation de ce dernier peut fournir jusqu'à 100 mA qui sont suffisant pour alimenter le reste des modules et en plus, elle est très stable. Il n'y a ainsi pas de risque d'avoir des parasites.

Tous les modules sont alimentés par le régulateur interne du microcontrôleur, mise à part le capteur de distance qui est le plus énergivore d'entre tous et qui nécessite une alimentation de 5 volts. Ainsi, on protège l'uC d'une éventuelle surconsommation et au besoin, si des modifications ou bien des rajouts sont prévues, il reste encore de la marge pour alimenter d'autres modules.

Voici l'alimentation générale de la carte :



La sortie 5 volts va donc alimenter directement le capteur de distance et le microcontrôleur. Ce dernier, à l'aide de son régulateur interne va alimenter les autres éléments de la carte à une tension de 3.3 volts. La diode en sortie n'est qu'un simple témoin lumineux qui nous montre que le circuit est bel et bien alimenté.

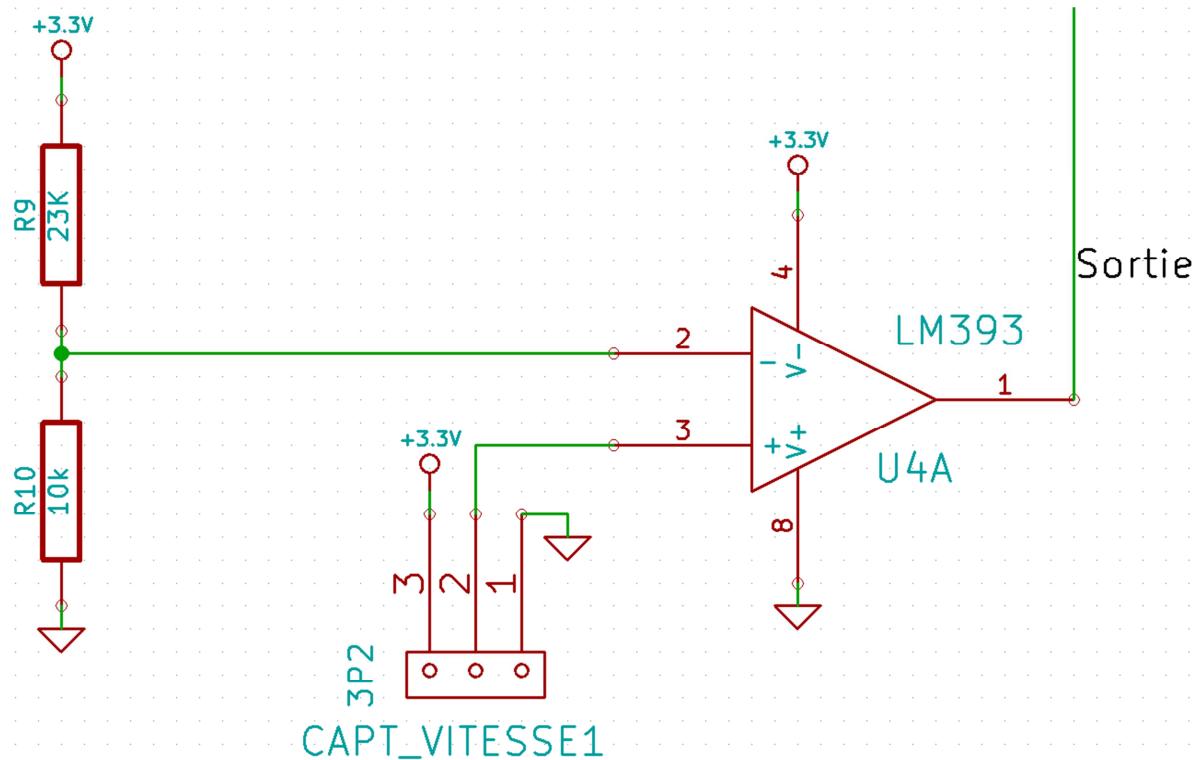
Comme on peut le voir, le composant utilisé pour l'alimentation est un LM2775. C'est donc une alimentation à découpage et ce choix est tout à fait arbitraire. Durant la formation de technicien, ce même composant avait été utilisé pour alimenter diverses cartes. Étant donné qu'il est relativement stable et efficace, il est repris dans le cadre de ce projet. De plus, il est connu. La tension de sortie peut être différente : elle n'est pas fixe. La datasheet apporte plus de détails à ce sujet et on trouve un schéma permettant de fournir une tension de 5 volts.

L'alimentation est un peu surdimensionnée, car elle est sensée pouvoir fournir jusqu'à 1 ampère en théorie. Cela dit, l'inductance ne tient que 600mA. Le choix n'est certainement pas très judicieux, car notre circuit consomme au maximum 126mA. Quoiqu'il en soit, malgré que l'alimentation soit légèrement surdimensionnée, le circuit fonctionne parfaitement. De plus, si un jour une modification doit être faite, afin d'ajouter des composants par exemple, l'alimentation pourra tenir une plus grande charge. Par contre, il faut faire attention à l'apport en énergie. Si la tension de batterie qu'on connecte à notre carte est inférieure à 7 volts, le fonctionnement du circuit n'est pas garanti. Il faut cette valeur minimum de tension pour que le régulateur puisse fonctionner.

Pour finir, le montage est protégé par un fusible et par une petite diode schotky. Le fusible est là pour prévenir d'éventuels courts-circuits et la diode si on branche l'accumulateur ou la batterie à l'envers.

### 7.1.3. Mise en forme vitesse

Comme vu précédemment, les signaux fournis par le capteur de vitesse ne sont pas très exploitable. Il faut les mettre en forme, afin qu'ils puissent être lus par le microcontrôleur correctement. Pour cela, plusieurs méthodes sont envisageables. Dans le cadre de ce projet, on va se servir d'un comparateur, afin de mettre en forme ces signaux. On pourrait faire un montage de type trigger, mais ce n'est pas nécessaire car le signal que nous retourné le capteur de vitesse est stable. Voici le montage avec le comparateur :

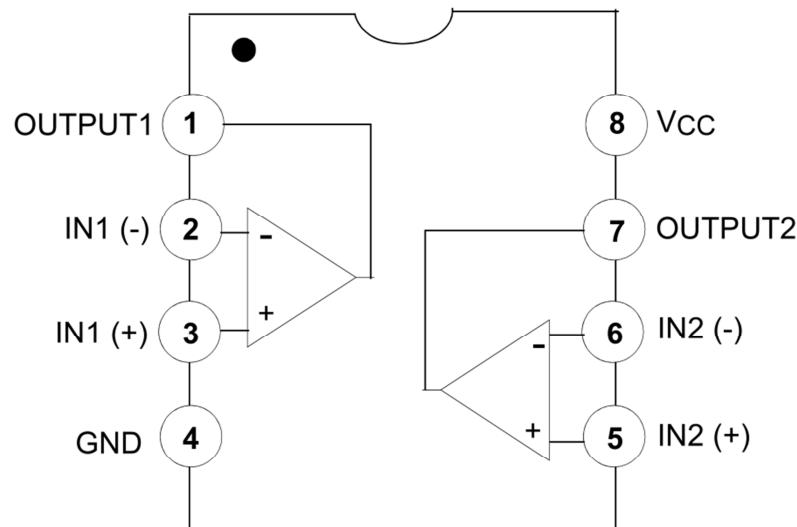


Le capteur de vitesse est branché au connecteur et le signal arrive directement sur le comparateur. Il n'y a rien de particulier sur ce montage. Les résistances R9 et R10 servent seulement à fixer le point de basculement à 1 volt. Ainsi, à la sortie du comparateur, on aura des pulses carrés qui sont bien plus pertinentes à traiter. Pour le calcul du point de basculement, on peut admettre que  $R10 = 10\text{k}\Omega$ . Si on veut une tension de basculement de 1 volt, on admet que  $UR1 = 1$  volt. La formule pour calculer R9 :

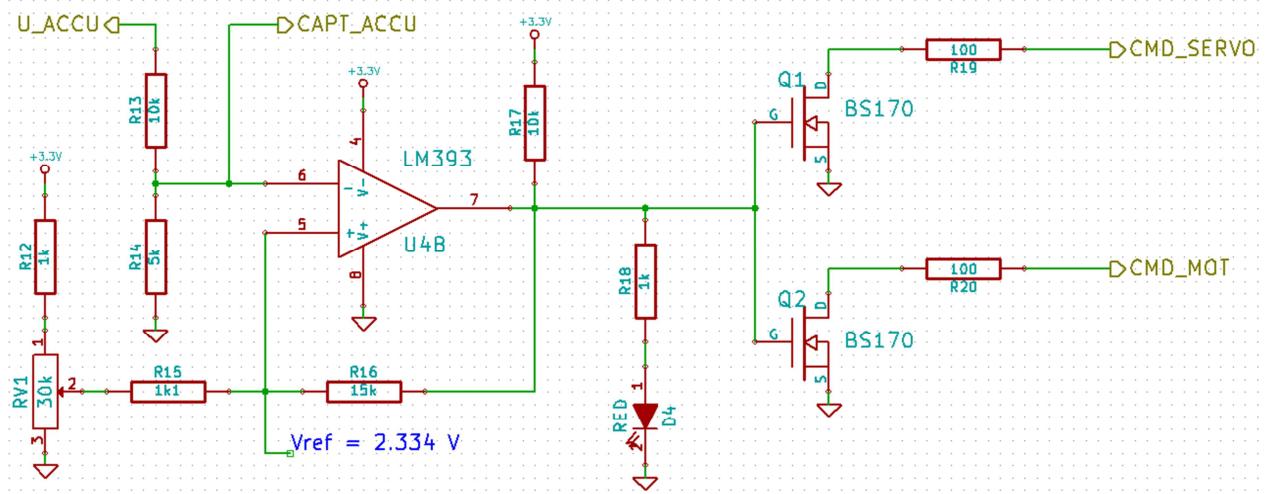
$$R9 = \frac{(Vcc - UR1)}{\frac{UR1}{R1}} = \frac{3.3 - 1}{\frac{1}{10000}} = \frac{2.3}{0.0001} = 23\text{k}\Omega$$

#### 7.1.4. Protection tension basse

Si la batterie de la voiture se vide, nous voyons tout de suite que le véhicule n'avance plus. Forcément on va arrêter de l'utiliser et mettre en charge la batterie. Il peut être utile de réaliser un petit circuit dédié qui empêche l'utilisation de la voiture, afin d'éviter l'endommagement de la batterie. Pour le capteur de vitesse, on utilise déjà un comparateur : le LM393. Le boîtier du composant contient deux comparateurs internes :



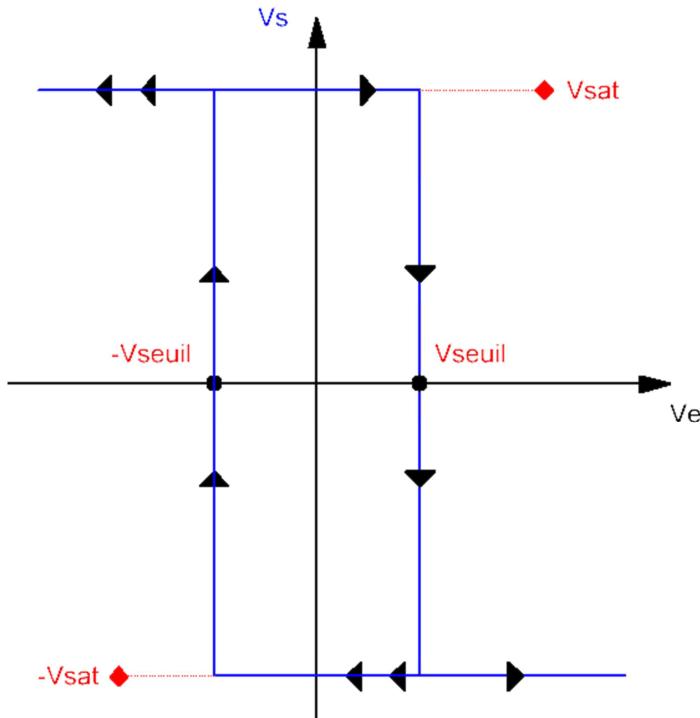
On peut se servir de ce deuxième comparateur, afin de faire un petit circuit empêchant l'utilisation des moteurs, dans le cas où la batterie aurait atteint une tension trop basse. Pour s'y faire, on doit réaliser un trigger de schmitt qui aura deux points de basculement. Ainsi, si la batterie est vide, on force les moteurs à zéro et ils deviennent inutilisables tant que la batterie n'a pas été rechargée ou changée. Voici le schéma du trigger de schmitt :



Il y a plusieurs éléments sur ce montage, mais son fonctionnement reste assez simple. On peut voir le diviseur de tension avec les résistances R13 et R14. Le diviseur est là car la tension de l'accumulateur est 7.2 volts, donc trop haute pour être lue. A l'aide de ces résistances, on va diviser la tension par 3 et ainsi obtenir une tension d'approximativement 2.3 volts.

Grâce à cela, la tension peut être exploitable par la « protection tension basse » et par le microcontrôleur qui peut réagir en fonction de la charge de la batterie. De plus, en utilisant ce procédé, on pourra même déterminer l'autonomie de la voiture, en fonction de la tension de la batterie.

Maintenant, on passe aux points de basculements. Un trigger de schmitt à la particularité de posséder deux points de basculement, contrairement à un, comme c'est le cas pour le capteur de vitesse. Voici ce que donne le signal à la sortie d'un trigger de schmitt habituel :



Comme on peut le voir, la sortie va dépendre de deux points de basculement. C'est exactement le comportement que l'on souhaite. Bien sûr, ce n'est qu'une illustration pour comprendre le fonctionnement. Le trigger de ce projet aura des points de basculement différent. Si la batterie est vide, la sortie du comparateur sera active et les deux transistors vont se mettre à conduire. Ce qui provoquera un blocage des moteurs, car leur commandes seront directement mises à la masse et la LED nous indiquera que la batterie est vide.

Voici les deux formules nécessaires au calcul des points de basculement :

$$PSB = Vref + (VoH - Vref) * \frac{R15}{R15 + R16}$$

$$PIB = VoL + (Vref - VoL) * \frac{R15}{R15 + R16}$$

$VoH$  = Tension de sortie haute = 3.3 [V]

$VoL$  = Tension de sortie basse = 0 [V]

PSB = Point supérieur de basculement

PIB = Point inférieur de basculement

La batterie ne peut plus faire tourner les moteurs à partir d'environ 6.5 volts de charge. Donc le PIB se trouve à 6.5/3, donc à 2.16 volts, car il ne faut pas oublier que le comparateur et le microcontrôleur sont alimentés en 3.3 volts. Leurs entrées ne peuvent directement lire une tension de 6.5V. Pour protéger la batterie, nous pouvons mettre de la marge en la fixant à 2.23 volts, ce qui correspond à une tension de batterie de 6.7 volts. Le PSB est fixé en fonction de la pleine charge de la batterie, donc 7.2 volts. Ce qui correspond à une valeur de 2.4 volts pour le PSB. Pour récapituler :

PSB = 2.4 volts

PIB = 2.16 volts

Une fois que nous avons ces données, il faut transformer la formule, afin d'obtenir la tension de référence. Voici comment la calculer :

$$Vref = \frac{PSB * (R15 + R16) - VoH * R15}{R16}$$

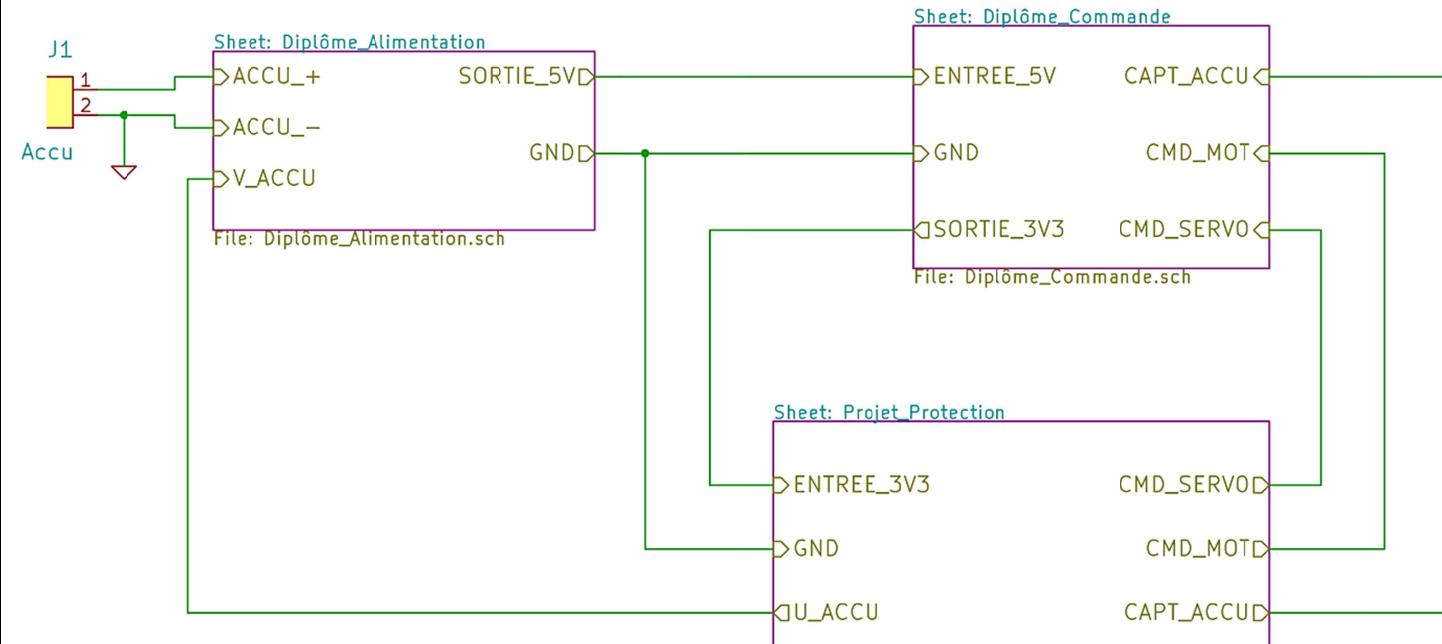
$$Vref = \frac{2.4 * (1k1 + 15k) - 3.3 * 1k1}{15k} = 2.334 [V]$$

Une fois que l'on connaît la tension de référence, il ne reste plus qu'à la fixer avec le potentiomètre. Ensuite, à chaque fois que notre batterie descend sous la tension minimum de 6.7 volts, la protection se déclenche et les moteurs ne peuvent être contrôlés. Lorsque la tension remonte à 7.2 volts, la protection de tension basse relâche les moteurs et on peut ainsi les utiliser de nouveau.

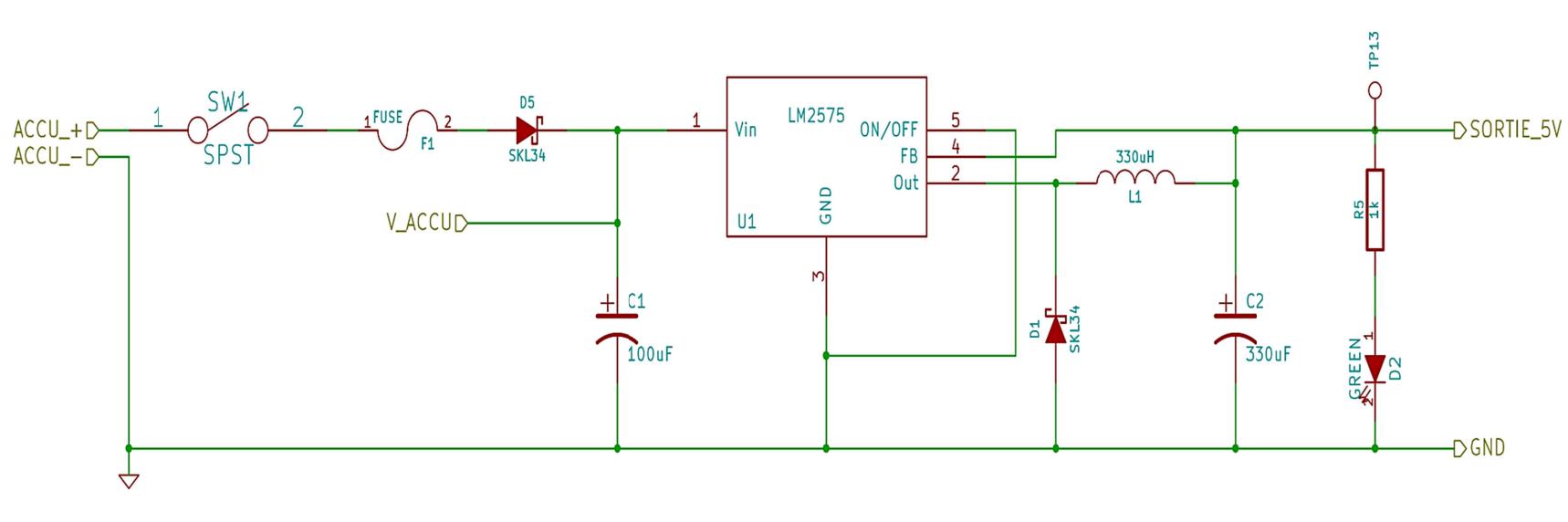
### 7.1.5. Schéma complet

Le schéma étant grand, il est présenté sous quatre blocs :

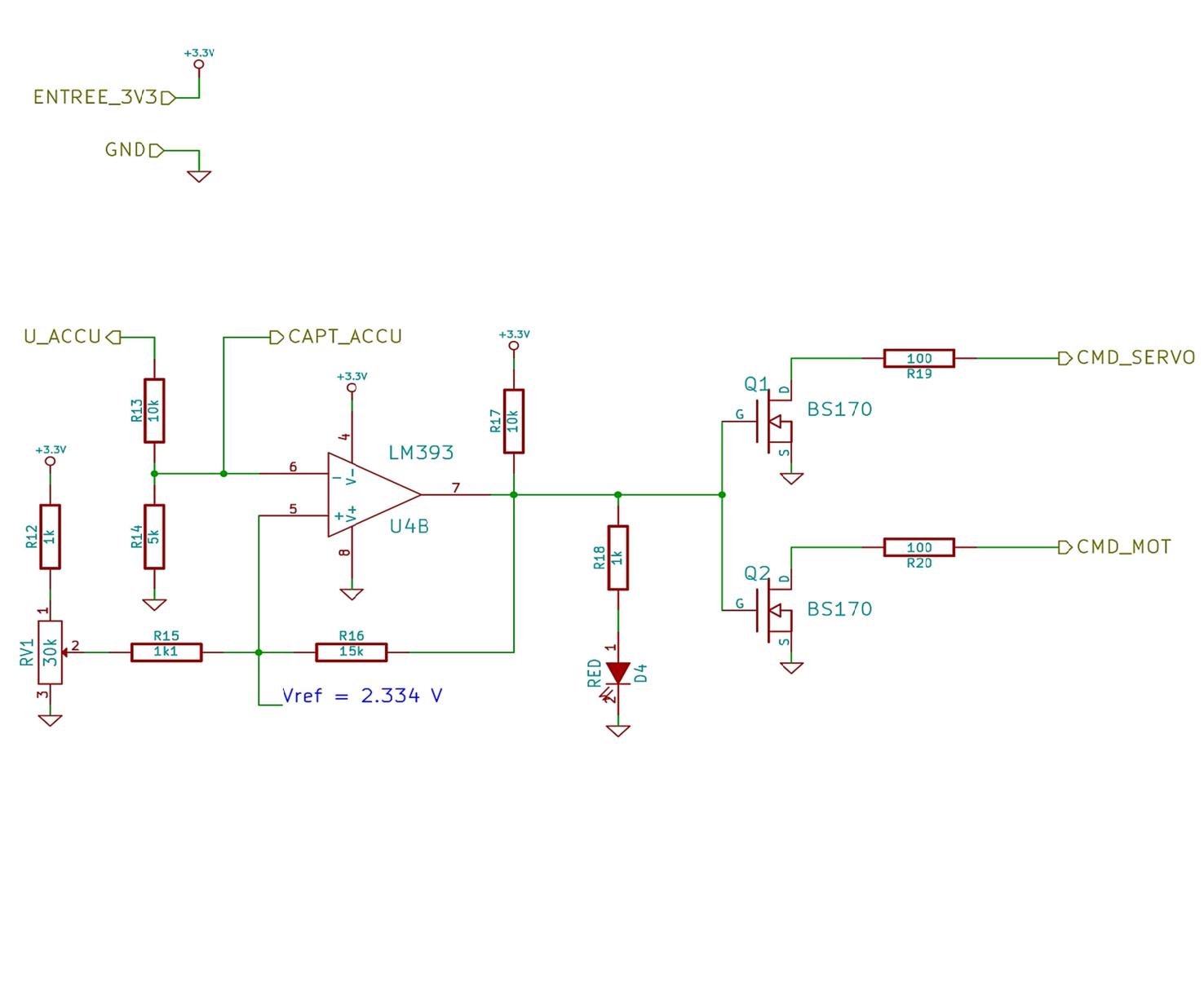
Les trois blocs :



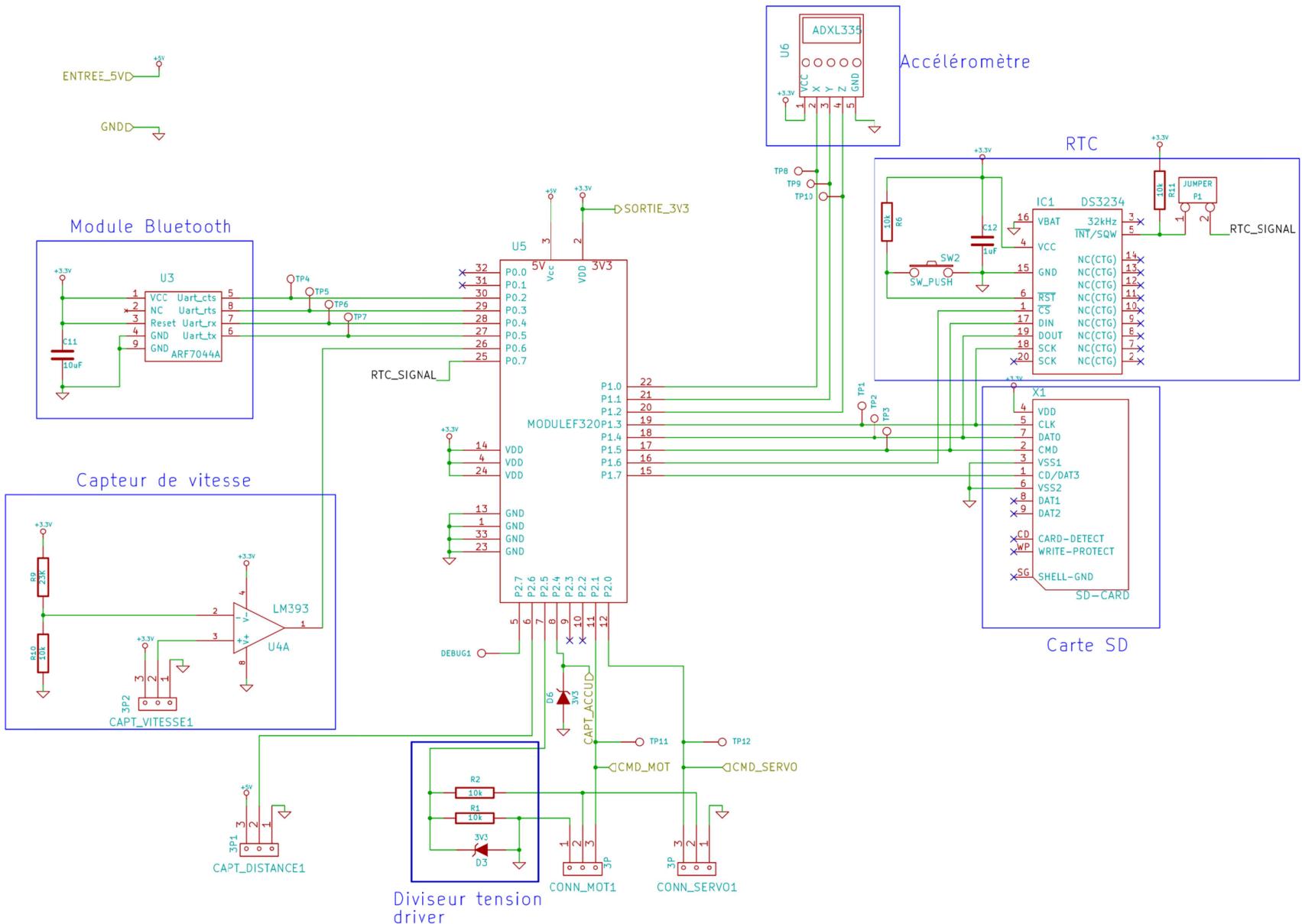
## L'alimentation:



## La protection:



## Le contrôle:



## **7.2. Conception et développement logiciel uC**

### **7.2.1. Conception microcontrôleur**

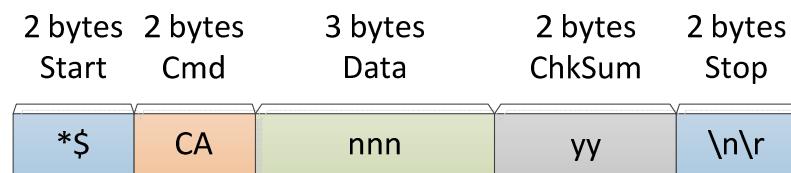
Comme expliqué ultérieurement, le microcontrôleur possède plusieurs périphériques internes que l'on peut configurer pour obtenir le comportement souhaité. Voici un récapitulatif des périphériques qui sont utilisés :

Péphérique interne	Utilisation	fonction
UART	oui	communication bluetooth
SPI	oui	communication carte SD et RTC
TIMER 0	oui	mesure vitesse
TIMER 1	oui	horloge UART (115'200bauds)
TIMER 2	oui	base de temps
TIMER 3	oui	Timeout UART
PCA_CEX0	oui	commande servomoteur
PCA_CEX1	oui	commande moteur principale
ADC0	oui	mesure accélération, distance, tension batterie et tension driver

Ce sont les périphériques utiles au bon fonctionnement du projet.

### **7.2.2. *UART***

Le périphérique UART est donc utilisé pour communiquer entre le microcontrôleur et le module Bluetooth qui est utilisé dans ce projet. Pour s'y faire, il faut en priorité fixer la vitesse. La vitesse de 115'200 bauds a été choisie, afin d'avoir une communication suffisamment rapide. Voici un exemple de trame qui peut être envoyée depuis le smartphone au microcontrôleur :



y : checksum sur 8 bits (0...F) n : commande marche avant (50..100)  
Commande marche arrière (0..50)

Ceci est la commande que l'on veut envoyer pour faire avancer la voiture. Elle a été fixée de façon totalement arbitraire, car les différents programmes que l'on a faits à l'école utilisent des trames plus ou moins égales. Voici à quoi servent les différentes parties de la trame :

2 bytes de start :	pour savoir à quel moment la trame commence
2 bytes de commande :	pour savoir l'intention que l'on veut
3 bytes de données :	pour connaître la valeur que l'on veut envoyer à la voiture
2 bytes de checksum :	contrôle de trames
2 bytes de stop :	pour connaître à quel moment la trame se termine

C'est un protocole simple. Etant donné que le but est de piloter une voiture, il n'y a pas besoin de faire un protocole de communication plus compliqué. Maintenant que nous connaissons la longueur de trame la plus longue : 11 bytes, on peut calculer le temps d'envoi d'une trame complète :

$$\text{Temps d'envoi} = \text{Période baudrate} * \text{nbr de caractères} * \text{nbr de bits pour un caractère}$$

$$\text{Période baudrate} = 1/\text{baudrate}$$

$$\text{Nbr de caractères} = 11 \text{ caractères dans la trame}$$

$$\text{Nbr de bits pour un caractère} = 8 \text{ bits pour un caractère} + 1 \text{ bits start} + 2 \text{ bits stop}$$

$$\text{Temps d'envoi} = 8.68\mu * 11 * 10 = 954.86 \text{ microsecondes}$$

Ceci est le temps d'envoi d'une trame complète. A ceci, vient s'ajouter le temps de communication Bluetooth. Celui-ci n'est pas connu et n'a pas pu être mesuré. Cela dit il est probablement plus cours, car le module fonctionne à 723 kbps. Si on admet que c'est le cas, il faut compter une centaine de microseconde environ, donc pour l'envoi d'une trame complète il faut compter environ 1.1 milliseconde. Ceci n'a pas pu être vérifié du tout notamment au manque de temps. On peut même compter large et arrondir admettons à 2 millisecondes.

Cela reste tout à fait acceptable, car entre les différents temps d'envoi, de traitement, d'action, de réponse, etc. cela prend quelques dizaine de milliseconde de temps de réaction. Sachant que l'humain a un temps de réaction qui s'élève en tout cas à la seconde, nous avons suffisamment de marge. Même si le temps d'envoi une trame, de la traiter, de réagir en conséquence avec la voiture prend 200 millisecondes, nous ne verrions pas vraiment la différence, donc nous sommes suffisamment larges.

### **7.2.3. Les timers**

Dans le microcontrôleur, il y a des éléments très utiles : les timers. Ceux-ci, nous permettent d'avoir un contrôle temporel sur le déroulement de notre code. Nous pouvons aussi nous en servir, afin de générer la baudrate de la communication série, comme c'est le cas avec le Timer 1. Ce sont vraiment les éléments indispensables au déroulement de tout bon programme. Voici comment ils sont configurés :

Le timer 1 fonctionne en mode 8 bits. On s'en sert pour créer l'oscillateur pour l'élément UART. Nous avons une communication qui fonctionne à 115'200 bauds, donc nous devons atteindre une fréquence deux fois supérieurs pour pouvoir lire ce que l'on reçoit. Le cas échéant, il risque d'y avoir des erreurs dans la lecture des valeurs que l'on reçoit.

Le timer 2 est utilisé pour obtenir une base de temps dans notre programme. Il utilise le mode 16 bits, afin de pouvoir atteindre une période de 1 milliseconde, avec un clock système de 12MHz. On s'en sert pour la synchronisation de différents éléments du programme. Notamment, pour la mesure de la vitesse de la voiture et pour la mesure de la distance de celle-ci.

Le timer 3 est utilisé pour connaître la fin d'une trame reçue via le périphérique UART. C'est-à-dire que lorsqu'on reçoit un caractère, on va réarmer le timer. Si le microcontrôleur ne reçoit plus rien, le timer 3 va générer une interruption. C'est grâce à cette dernière que le microcontrôleur sait qu'il a reçu une trame complète.

### **7.2.4. PCAS**

Le « programmable counter array » à un fonctionnement ressemblant à celui d'un timer. La différence avec un timer est que l'on peut l'interconnecter directement avec un port, afin de s'en servir comme générateur de signaux. Dans le cas de ce projet, on va s'en servir pour créer un signal PWM. Ce dernier aura une période fixe, mais une période de conduction qui va varier de 1 à 2 millisecondes. C'est grâce aux PCAs que l'on va pouvoir gérer les moteurs de la voiture.

### **7.2.5. ADC**

L'un des éléments clé à l'asservissement de notre projet. L'ADC est tout simplement un convertisseur AD. Il va convertir une valeur analogique en valeur numérique. Cette dernière pourra ainsi être traitée et le microcontrôleur prendra des décisions en conséquence. La mesure de la distance, de l'accélération, du niveau de la batterie et de la présence driver du moteur principale est rendu possible grâce à l'ADC.

### 7.2.6. Les ports

Les ports du microcontrôleur sont configurés en fonctions de ses besoins et de ses contraintes. Les ports sont disponibles en entrée analogique ou digital et en sortie digital. Voici un graphique qui résume le câblage des ports :

		Configuration				
PORT	bit	DI	DO	AI	Pérophérique interne	Elément externe
0	0					
	1					
	2		x		RTS	bluetooth
	3	x			CTS	bluetooth
	4		x		UART_TX	bluetooth
	5	x			UART_RX	bluetooth
	6	x			TIMER0 avec contrôle externe sur INTO	capteur de vitesse
	7				interruption ext INT1	libre
1	0			x	ADC0	accéléromètre
	1			x	ADC0	accéléromètre
	2			x	ADC0	accéléromètre
	3		x		SPI_CLK	RTC, carte SD
	4	x			SPI_MISO	RTC, carte SD
	5		x		SPI_MOSI	RTC, carte SD
	6		x		SPI_NSS1	RTC
	7		x		SPI_NSS2	carte SD
2	0		x		PCA_CEX0	servomoteur
	1		x		PCA_CEX0	moteur
	2					
	3					
	4			x	ADC0	accumulateur
	5			x	ADC0	driver moteur
	6			x	ADC0	capteur distance
	7		x		PIN DE DEBUG	

### **7.2.7. Le code : les fonctions**

Le microcontrôleur se code en langage de programmation C. Différentes fonctions ont été développées. Des analyses complètes de certaines d'entre-elles seront expliquées, car ça ne vaut pas la peine d'expliquer les fonctions les plus simples. Le code est disponible en annexe pour avoir plus de détails. Voici une liste des fonctions principales développées :

ClockInit() :

*Descriptif: Initialisation du mode de fonctionnement du clock système  
SYSCLK : Oscillateur interne à 12 MHz*

PortInit() :

*Descriptif: Initialisation du mode de fonctionnement des ports*

P0.2	:	entrée (OC)	:	CTS
P0.3	:	sortie (PP)	:	RTS
P0.4	:	sortie (PP)	:	UART_TX
P0.5	:	entrée (OC)	:	UART_RX
P0.6	:	entrée (OC)	:	interruption externe INT0
P2.0	:	sortie (PP)	:	commande servomoteur
P2.1	:	sortie (PP)	:	commande moteur
P2.6	:	entrée (AN)	:	mesure de la distance
P2.7	:	sortie (PP)	:	PIN DE DEBUG

avec : (OC) = Open Collector      (PP) = Push Pull  
(AN) = Analogique

IRQInit() :

*Descriptif: Initialisation du mode de fonctionnement de l'interruption externe 0, sur transition montante et sur P0.6*

UARTInit() :

*Descriptif: Initialisation du mode de fonctionnement de l'UART*

- 115'200 bauds
- mode 8 bits (pas de parité)
- signal de réception seulement si stop bit détecté
- Timer 1 à 230'400 Hz

%%% -> 115'200 8-N-1 <- %%%

### Timer2Init() :

*Descriptif: Initialisation du mode de fonctionnement du timer 2 pour avoir une base de temps de 1 milliseconde*

- Timer2 16 bits auto-reload avec un start soft
- base de temps toutes les 1 ms
- clock = 12 MHz

### Timer3Init() :

*Descriptif: Initialisation du mode de fonctionnement du timer 3 pour mettre en place un Timeout qui se déclenche Lorsqu'une trame complète de L'UART est reçue*

- Timer 3 16 bits auto-reload avec un start soft
- Clock = 12Mhz
- Temps désiré = 3.5 ms (environ)

### PCAIInit() :

*Descriptif: Initialisation du mode de fonctionnement des PCA  
Fréquence du PCA = 183Hz  
Large PWM : de 1 à 2 millisecondes*

### ADInit() :

*Descriptif: Initialisation le mode de fonctionnement du convertisseur AD  
8 bits  
Décalage à gauche  
Tension de référence = 3.3 volts*

### InterruptionUART() :

*Descriptif: Interruption qui se déclenche Lorsqu'on envoi ou reçoit une trame  
Si réception : enregistrement des données reçues dans un tableau  
Si transmission : envoi des données Les unes après les autres*

controleTrame() :

*Descriptif:* Vérifie si on a reçu une trame de connexion ou de déconnexion, sinon, vérifie la validité des champs <START>, <CHECKSUM> et <STOP>

*Entrée* : *ptrTxtIn*, l'adresse de la trame reçue  
*Sortie* : *nbPosition*, le nombre de caractères dans la trame (0..3)

*Sortie* : information de trame valide (1 si ok et 0 sinon)

determineCommande() :

*Descriptif:* Détermine le numéro de la commande (255 si inconnue)

*Entrée* : *ptrTxtIn*, l'adresse de la trame reçue  
*Sortie* : le numéro de la commande (0..255)

Traitement() :

*Descriptif:* Fonction permettant de faire le traitement de la trame reçue si commande moteur, commande tourner, commande anti-patinage ou commande connexion, on aura des réactions différentes

*Entrée* : *noCMD*, valeur de la commande  
          *ptrIn*, l'adresse de la chaîne reçue  
          *ptrOut*, l'adresse de la chaîne réponse  
*Sortie* : --

traitementVitesse() :

*Descriptif:* Transforme le nombre de pulse en vitesse

*Entrée* : compteur, nombre de pulses sur l'arbre de transmission (0..~560)  
*Sortie* : buffer, valeur de la vitesse en mètres / secondes

traitementDistance() :

*Descriptif:* convertie la valeur du convertisseur AD en distance

*Entrée* : valeur de la distance brute (0..255)  
*Sortie* : valeur de la distance en centimètre (0..38)

Basetemps() interruption timer 2 :

*Descriptif: Fonction d'interruption permettant la mise en place d'une base de temps de 1 milliseconde*

CompteVitesse() interruption externe :

*Descriptif: Fonction d'interruption permettant de compter le nombre de tour que l'arbre de rotation effectue*

lectureDistance() interruption AD :

*Descriptif: mémorise le résultat de la conversion de la valeur du capteur de température sur 8 bits et active un flag*

Timeout() interruption timer 3 :

*Descriptif: Fonction d'interruption du timer 3, gestion de la fin de trame sur timeout après 3.5 millisecondes d'absence de caractères*

commandeMoteur() :

*Descriptif: Pilote le moteur principale en marche avant ou arrière*

*Entrée : valCmd, valeur de commande pour le moteur principale (0..100)  
Sortie : --*

commandeServomoteur() :

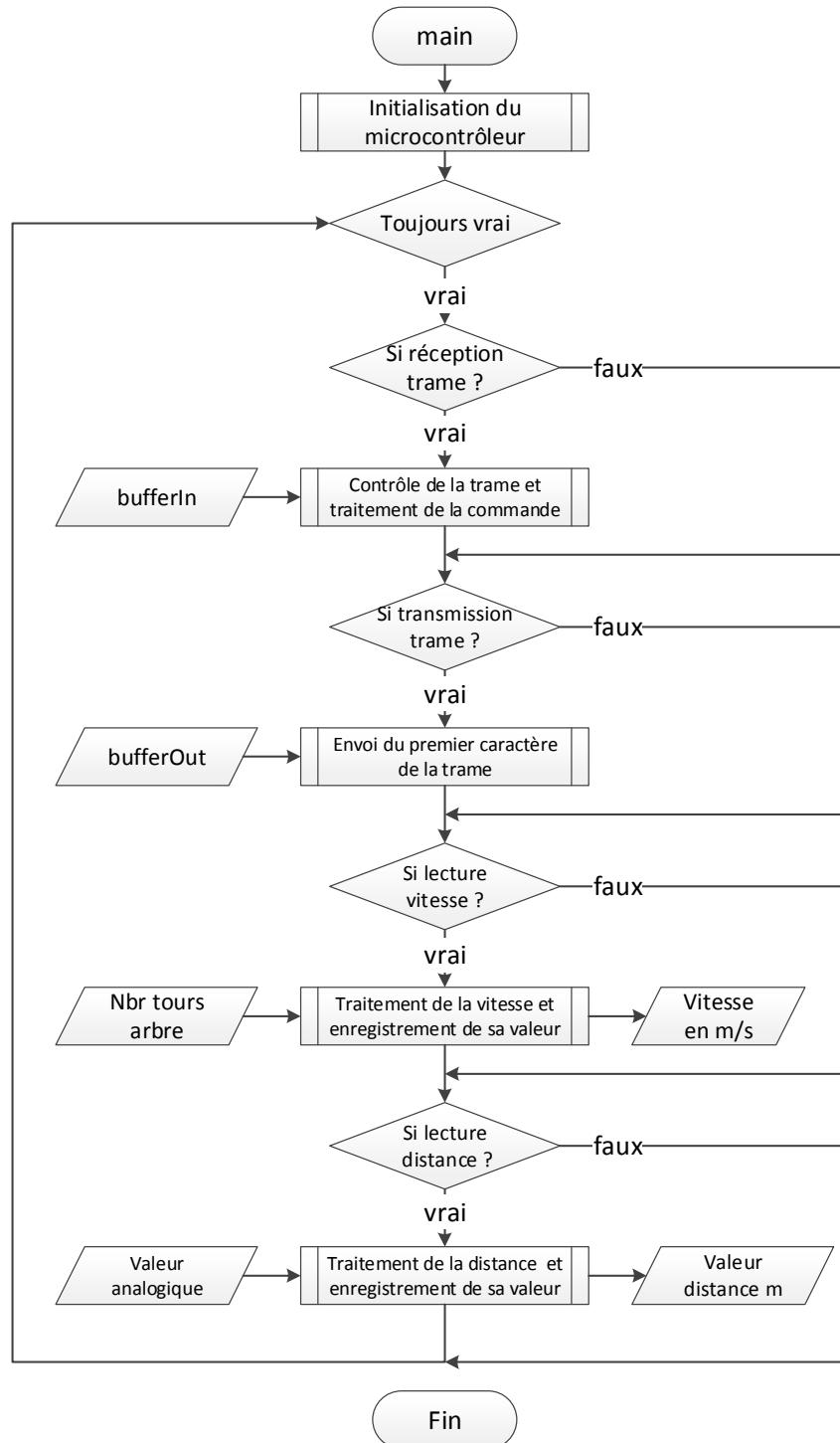
*Descriptif: Pilote le servomoteur*

*Entrée : valCmd, valeur de commande pour le servomoteur (0..100)  
Sortie : --*

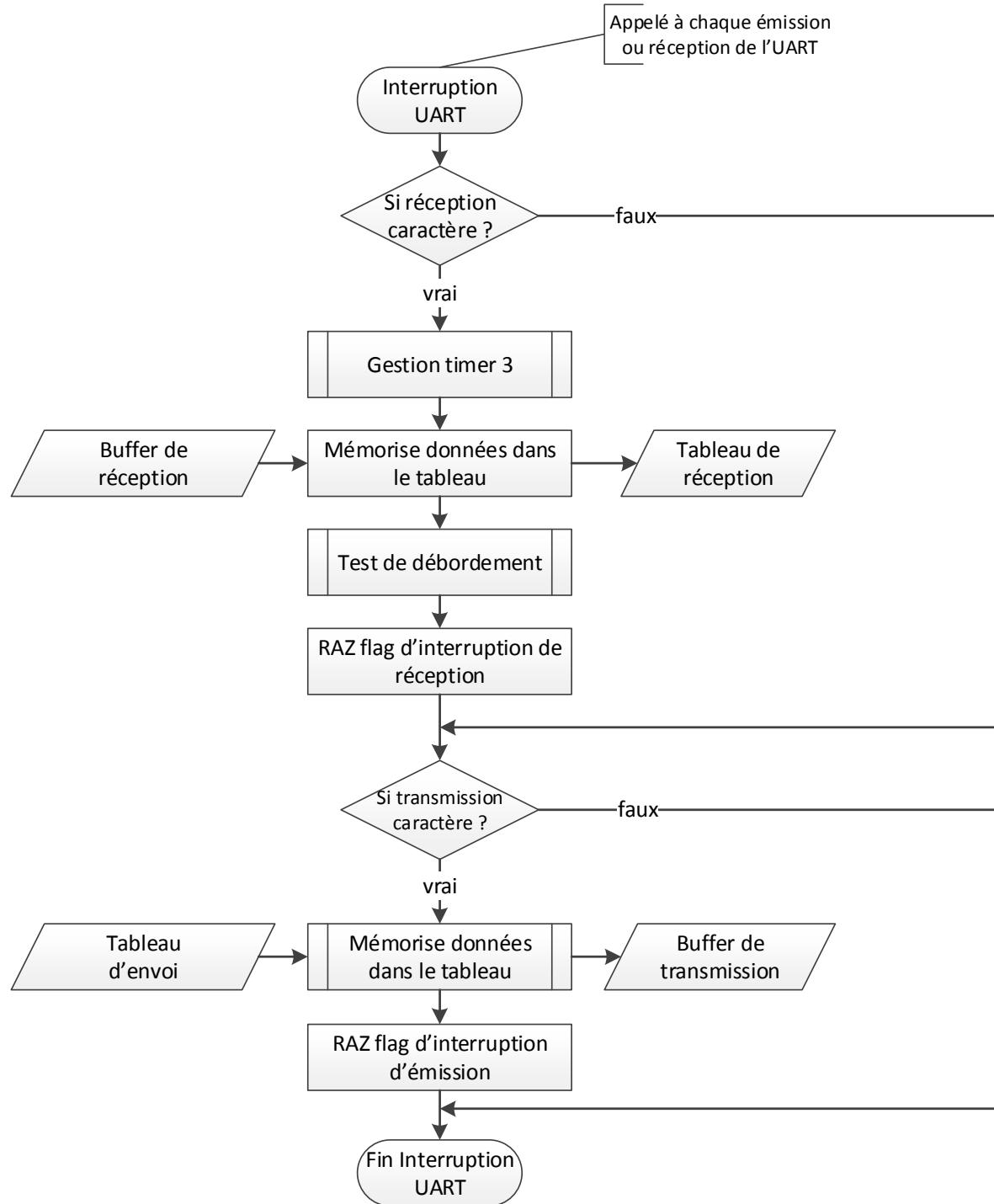
### 7.2.8. Le code : fonctionnement

Comme on peut le constater, plusieurs fonctions sont utilisées pour faire fonctionner le système. Ces différentes fonctions vont se partager certaines variables qui serviront à différents calculs et à exécuter différentes commandes.

Le fonctionnement du programme principal est relativement simple. Il attend constamment des interruptions. A chaque fois qu'un flag est activé, le programme principale le détecte et agis en conséquence. Voici un petit diagramme qui montre comment fonctionne le programme principal :

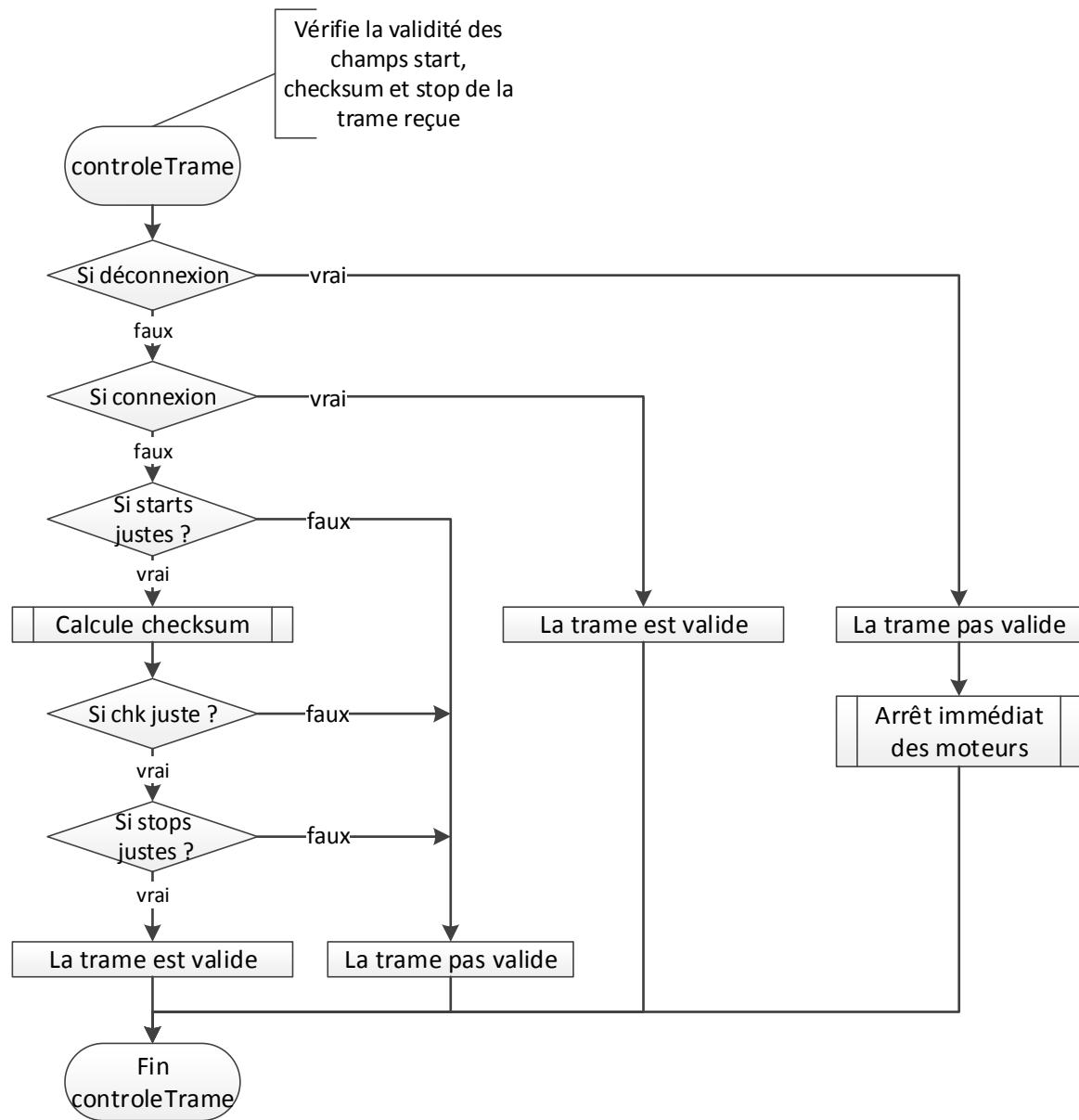


Comme on peut le voir, le programme ne se retrouve jamais bloqué. Il peut donc effectuer toutes les opérations sans avoir un comportement incertain. Il y a d'autres fonctions qui sont aussi importantes. L'une des plus importantes est l'interruption de l'UART. Voici comment elle fonctionne :

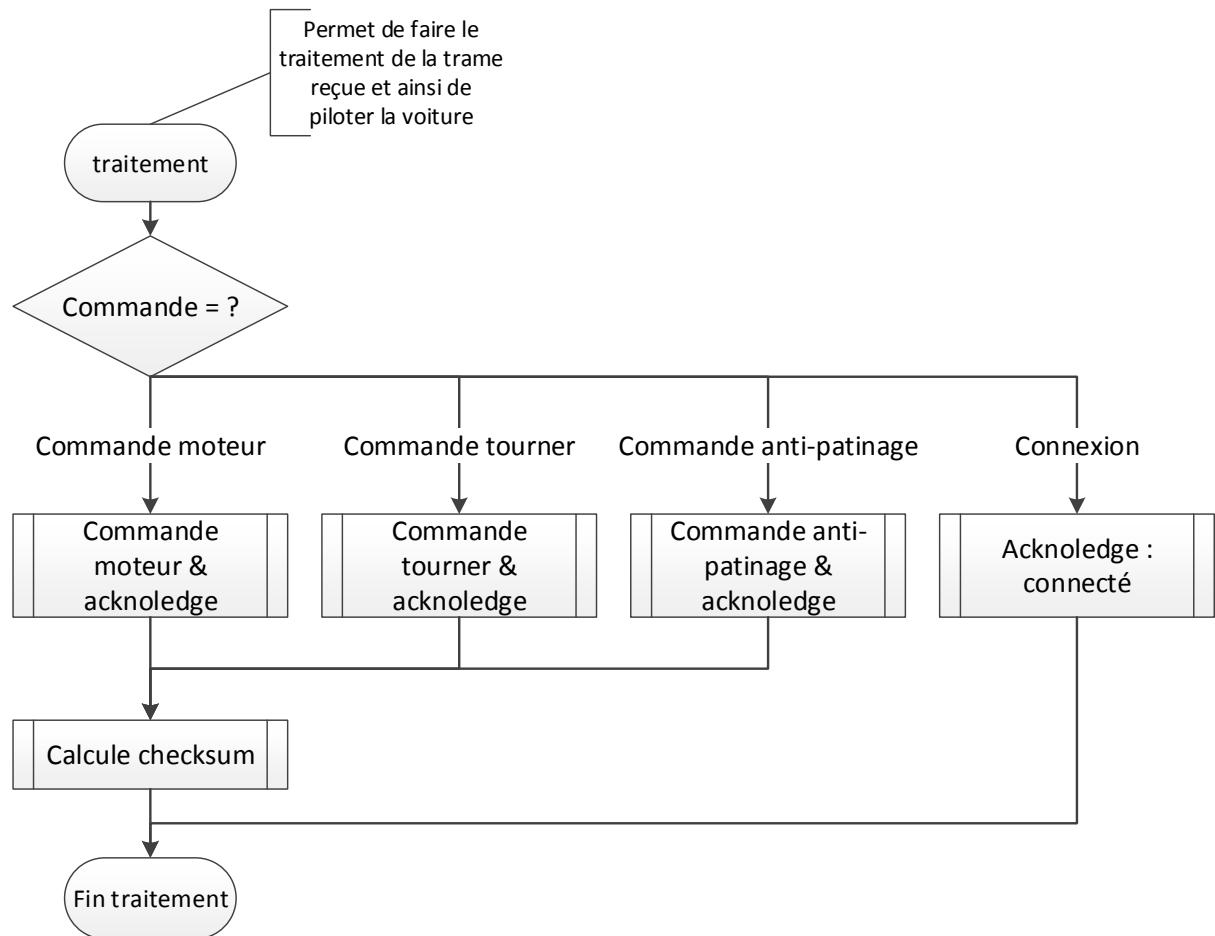


Le but de cette fonction est la gestion de la réception et la transmission de trames. Elle sera appelée à chaque fois qu'on reçoit une trame et à chaque fois que l'on veut envoyer une trame.

Une fois que nous avons reçu une trame complète, la chose la plus importante à faire est la déchiffrer. Pour cela, on va d'abord contrôler si elle est bonne. C'est pour cela qu'il y a une fonction qui contrôle les trames reçues, afin de trier les bonnes et les mauvaises. Dans le cas où la trame est bonne, le microcontrôleur va la traiter. Dans le cas où la trame est mauvaise, l'uC va juste répondre et prévenir le smartphone qu'il y a eu une erreur.



Une fonction de traitement de la trame reçue sert à contrôler la voiture. Lorsqu'une trame est reçue. Si la trame est juste, on va interpréter ce qu'elle veut et exécuter ce qu'elle demande. Il n'y a pour le moment que 4 commandes/requêtes qui sont traitées. Voici comment les commandes sont interprétées :



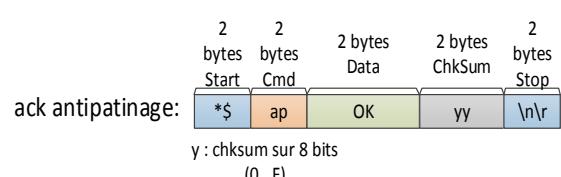
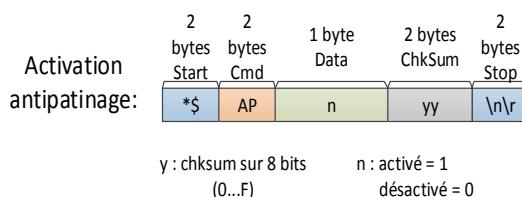
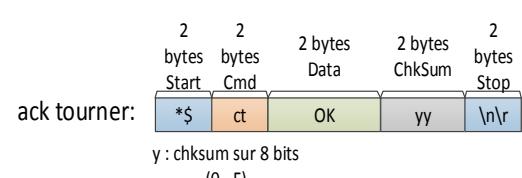
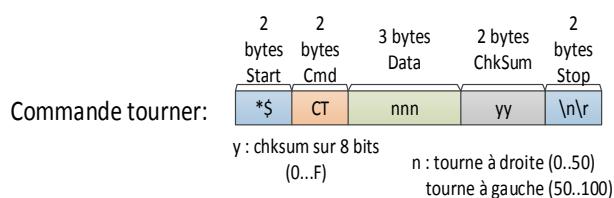
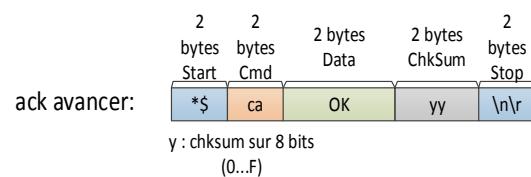
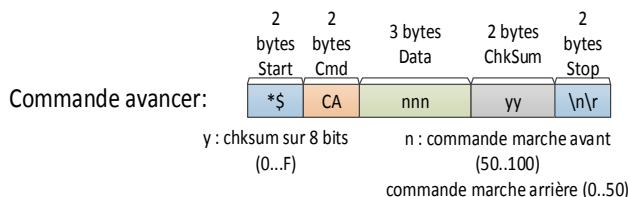
### 7.2.9. Protocole de communication

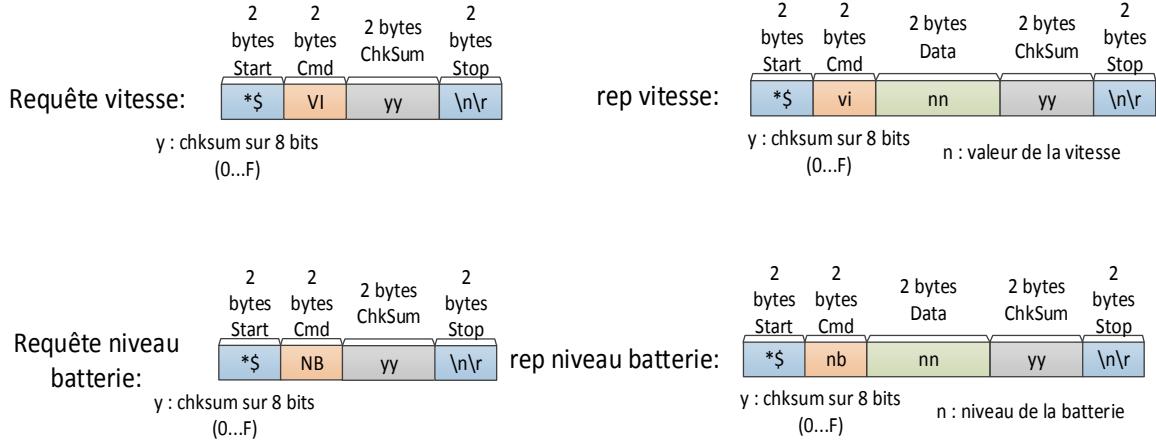
Une fois que le code est fait dans ses grandes lignes, on peut commencer à réfléchir à la stratégie utilisée pour mettre en place le protocole de communication. Il n'y a pas énormément de commande à mettre en place. Le principe retenu pour la communication est le master-slave. Le smartphone étant le maître, c'est lui qui va gérer les commandes et les requêtes. Du côté de la voiture, le microcontrôleur va simplement attendre que le smartphone le questionne ou lui envoie une commande. Bien sûr, il a une certaine autonomie, car si un obstacle se trouve devant la voiture ou si la connexion est perdue, il va automatiquement arrêter le véhicule.

Il y a 5 commandes/requêtes qu'il faut mettre en place :

- commande avancé
- commande tourné
- commande anti-patinage
- requête vitesse
- requête niveau batterie

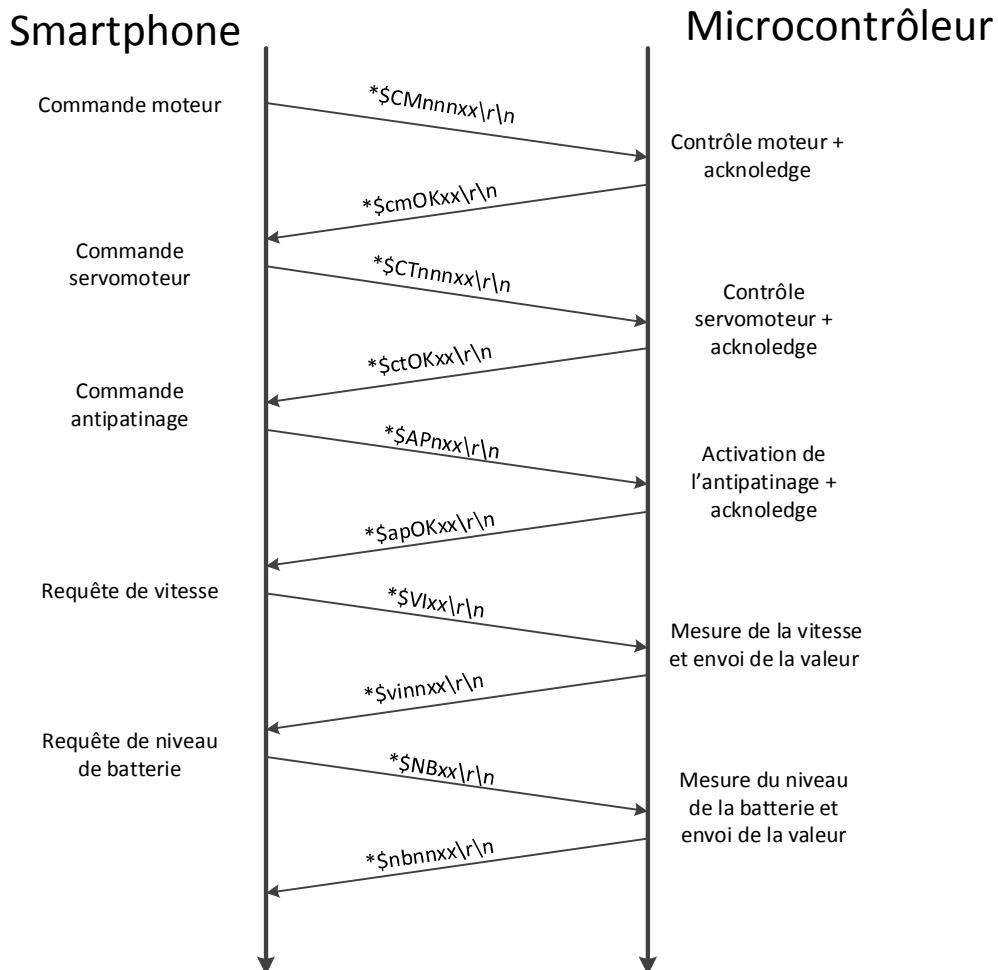
Il faut les mettre en place correctement. Durant la formation de technicien, on voit comment mettre en place des protocoles de communication simple. Il n'y a pas besoin de faire trop compliqué, car une sécurité accrue n'est pas nécessaire dans un tels produit. Voici comment les commandes et les requêtes ont été mis en place :



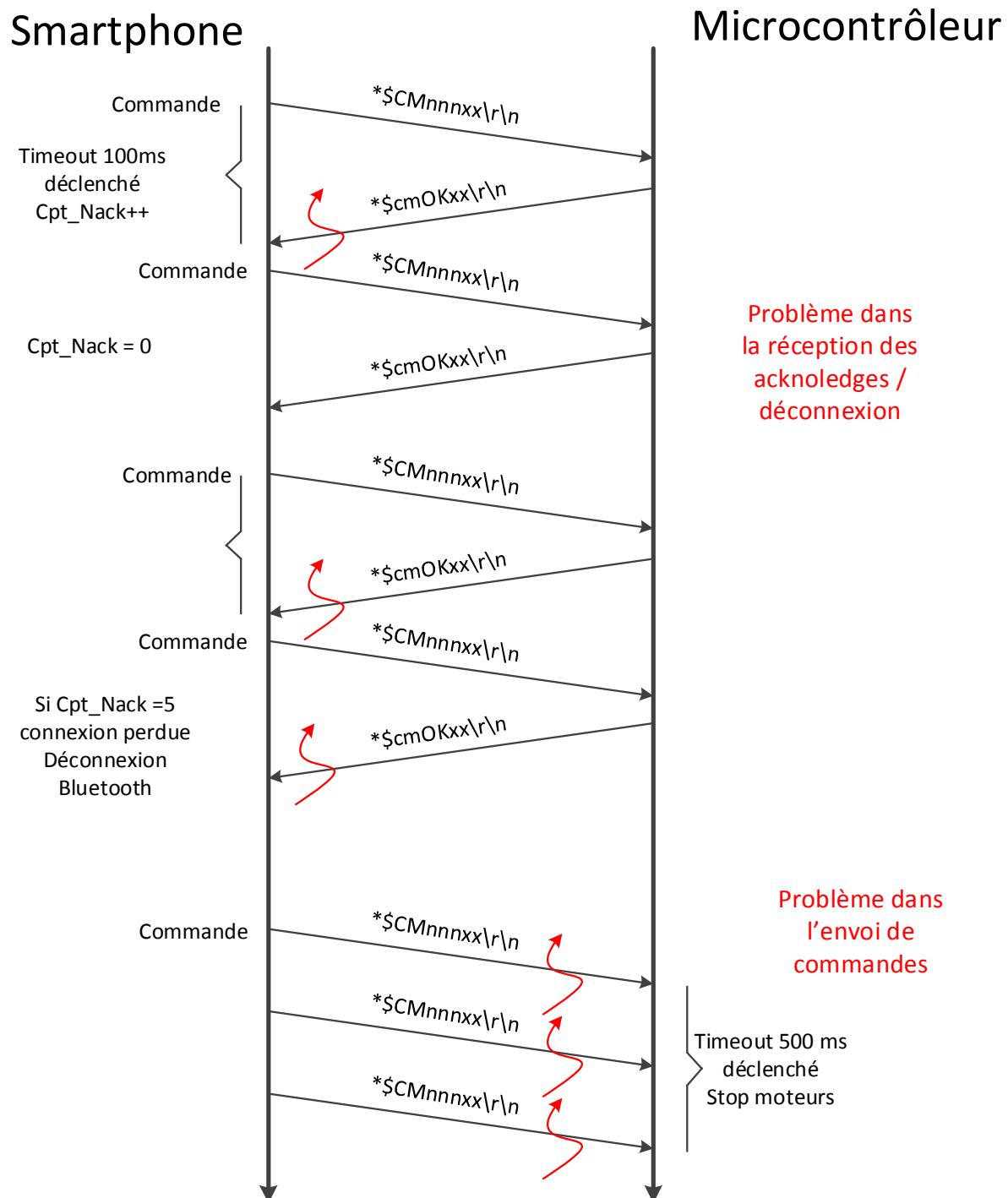


Une fois que les commandes et les requêtes sont connues, il faut trouver de quelle façon la communication va se passer. Plusieurs méthodes sont envisageables. De nouveau, le produit est destiné au grand public. Ça ne vaut pas la peine de faire trop compliqué. Il vaut mieux rester dans des choses plus tôt basique, mais fonctionnel et efficace.

C'est pour cela que pour communiquer, il faut envoyer une trame et attendre une réponse. De ce fait, nous aurons la confirmation que la commande/requête est bien parvenu à la voiture et qu'ainsi, on peut envoyer la trame suivante. Voici un diagramme qui montre comment cela fonctionne :



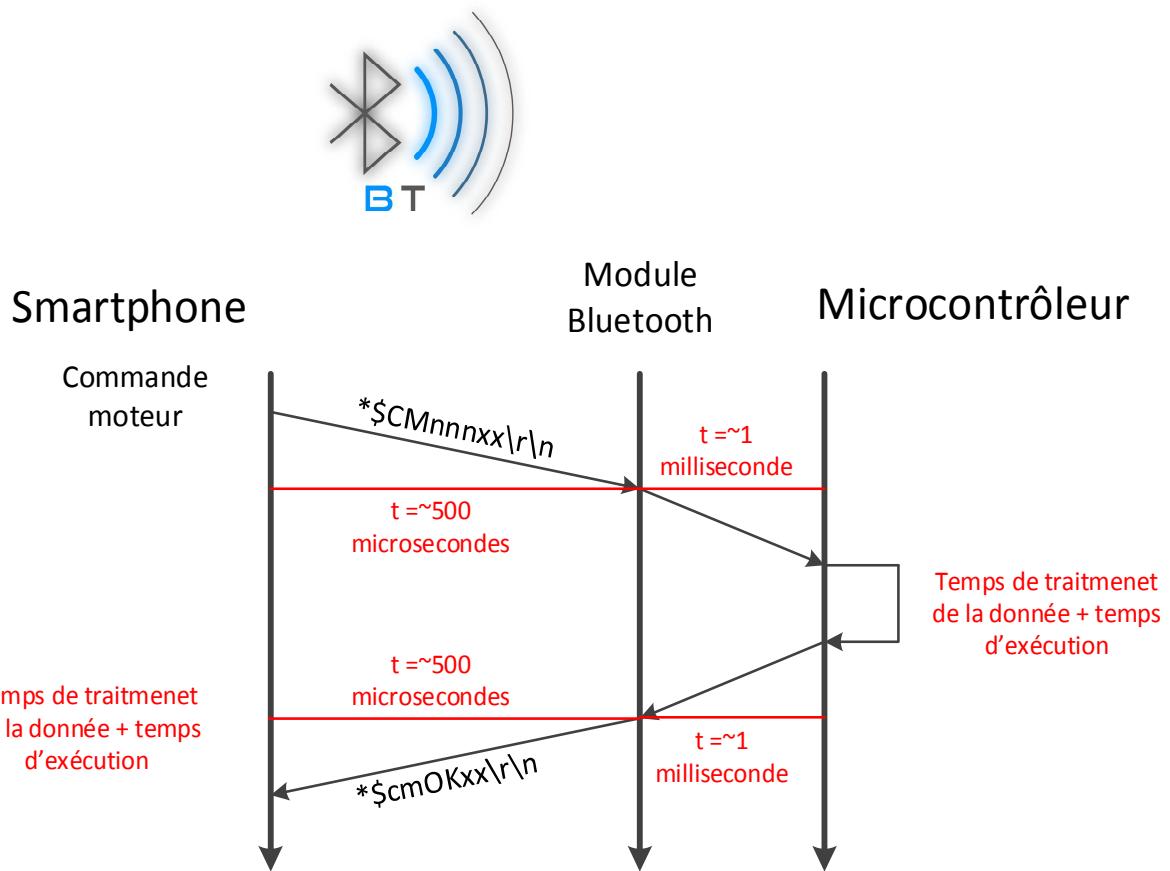
A chaque fois que l'on envoie une trame, on attend une réponse. Si la réponse correcte nous parvient, on peut envoyer la trame suivante depuis le smartphone. C'est ainsi que la communication a été pensée. C'est un modèle classique. Il est suffisant pour être utilisé dans ce projet. Ça ne vaut pas la peine de mettre en place un modèle plus compliqué pour un produit de loisirs grand public. Cela dit, la communication peut avoir certains problèmes ou des erreurs. Voici les différentes perturbations qui peuvent se produire :



Ce sont les erreurs qui peuvent arriver pendant la communication. C'est juste que l'un des cotes communique mal ou pas du tout. On peut imaginer encore d'autres cas de figures et imaginer d'autres solutions. Pour l'instant, ce sont les cas les plus probables et ceux qui doivent être résolus en premier lieu. Ce sont des cas qui se produisent principalement lors de la perte de la communication entre le smartphone et la voiture. C'est pour cela qu'il est important de les détecter.

On peut imaginer un compteur de mauvaises commandes du côté du microcontrôleur aussi comme c'est le cas pour le smartphone.

Il faut faire attention à une chose : la communication n'est pas directe. Il ne faut pas oublier qu'on passe par un module qui interface le smartphone et l'uC. Le diagramme de communication devrait plus tôt être représenté comme cela :



On constate qu'il y a un certain temps de propagation pour envoyer une trame complète. C'est pour cela qu'il faut faire très attention à la communication, car entre le temps d'envoi, le temps de traitement, le temps de réponse, etc. il n'y a pas loin de 10 millisecondes qui s'écoulent. Ça peut paraître lent en fonction de la situation, mais à l'échelle humaine c'est tout à fait acceptable. Une fois que la communication est définie et que la stratégie est mise en place, on peut commencer à faire l'application pour le smartphone.

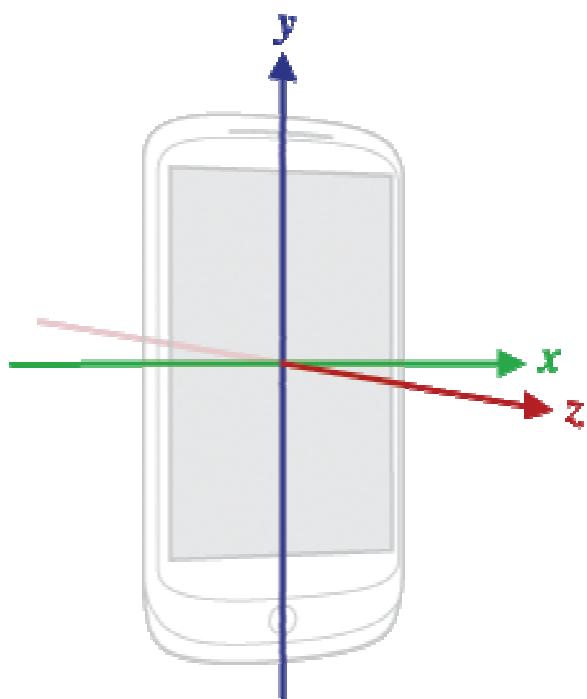
### **7.3. Conception et développement logiciel Android**

L'une des parties des plus compliquées du projet est certainement la mise en place du code pour l'application Android. Les techniciens étant principalement formés en électronique, il faut se mettre à jour, afin de pouvoir faire des programmes sous Android. Pour faire des applications, on va utiliser le langage de programmation Java qui est assez ressemblant au langage C# qui est enseigné aux techniciens.

Le téléphone utilisé pour mettre en place l'application est un Samsung Galaxy S5, avec la version Lollipop d'Android. Le but de l'application est de mettre en place une communication Bluetooth, pour contrôler la voiture. En plus de cela, pour contrôler la voiture, il faut utiliser l'inclinaison du téléphone. C'est deux problématiques bien distinctes.

#### **7.3.1. L'inclinaison du smartphone**

De nouveaux jours, la majorité des téléphones possèdent des capteurs internes pouvant indiquer plusieurs informations. On peut connaître l'altitude, l'accélération, la vitesse, etc. Dans le cas de ce projet, ce qui nous intéresse est l'inclinaison du téléphone. Pour cela, nous avons besoin de deux informations : l'accélération et l'électromagnétisme. Ces deux capteurs du téléphone, vont nous servir pour calculer des vecteurs qui par la suite peuvent être transformés en angle. Ce n'est pas très clair, on trouve sur internet quelques informations là-dessus, mais c'est compliqué à comprendre et à trouver. Heureusement, après plusieurs essais, un code a pu être mis en place, afin de pouvoir connaître l'angle d'inclinaison du téléphone. Le code se trouve en annexe.



### **7.3.2. La communication Bluetooth**

La communication Bluetooth est certainement la plus grosse difficulté du projet. Sur internet, on trouve énormément d'informations plus ou moins fiables. C'est assez dur de comprendre le fonctionnement. Cela dit, Google fournit quelques informations sur le fonctionnement de la communication. On apprend ainsi comment créer une connexion Bluetooth simple et aussi comment échanger les données entre deux périphériques. Il faut savoir que pour échanger les données entre deux périphériques, il faut obtenir un socket. Une fois que l'on a le socket, tout fonctionne. C'est ce qu'il faut comprendre et quelqu'un qui n'a pas passé l'étape de la compréhension, ne pourra pas mettre en place facilement une communication Bluetooth. C'est pour cela que la programmation Android a pris du temps pour la mise en place.

### **7.3.3. Les différentes class**

Voici les différentes class développées dans l'application et leurs méthodes :

<b>ConnectedThread</b>	<b>ConnectThread</b>
-read -write -cancel	-run -cancel -returnSocket
<b>GestionTrames</b>	<b>MainActivity</b>
-trameToSend -convertValues -verificationTrame	-onCreate -onResume -onPause -gestionInterruptionCapteur -gestionCommunication -onCreateOptionsMenu -onOptionsItemSelected -recherchePeripheriques

#### **7.3.4. *MainActivity***

La class « *MainActivity* » est la class principale. C'est elle qui permet les interactions sur l'interface utilisateur. Les différentes méthodes sont utilisées pour des fonctions bien précises :

- *onCreate* : C'est la méthode qui est appelé lorsque l'activité se crée. C'est principalement là que l'on va instancier tous les objets et les éléments dont nous avons besoin.
- *onResume* : Cette méthode est appelée à chaque fois que l'on revient dans l'application. Dès qu'on quitte l'application et que l'on revient dessus, cette méthode est appelée. Ici, le but est surtout d'instancier les différents éléments, afin que l'application reprenne son cours normale.
- *onPause* : Lorsqu'on quitte l'application, cette méthode est appelée. C'est à ce moment-là qu'il faut mettre en pause toutes les communications et empêcher les événements de se produire. Sinon, il y aura des problèmes avec l'environnement Android.
- *gestionInterruptionCapteur* : Il faut instancier l'interruption des capteurs et c'est à cet endroit que l'on fait. Le but de cette méthode est simplement l'instanciation. Une fois que c'est fait, à chaque interruption des capteurs, c'est le code qui se trouve dans cette méthode qui sera exécuté.
- *gestionCommunication* : C'est cette méthode que l'on va utiliser pour gérer la communication. C'est à elle qu'on passe en paramètre ce que l'on veut envoyer.
- *onCreateOptionsMenu* : C'est la méthode de base qui va créer le menu d'options.
- *onOptionsItemSelected* : A chaque fois que l'on clique sur un élément se trouvant dans la liste du menu, on va exécuter du code se trouvant cette méthode.
- *recherchePeripheriques* : Cette méthode va simplement gérer la recherche des périphériques et l'enregistrement de ceux-ci dans une liste.

#### **7.3.5. *ConnectThread***

Cette classe va permettre de mettre en place la connexion entre le smartphone et un autre périphérique. Elle implémente la class *Thread*, afin que la tâche de connexion ne fasse pas planter le téléphone et ne bloque pas celui-ci. L'explication des méthodes :

- *run* : Permet de mettre en place la connexion à l'aide du socket.
- *cancel* : Ferme la connexion Bluetooth.
- *returnSocket* : Retourne le socket de la connexion, afin de pouvoir être réutilisé pour la communication. On en a besoin lors de l'instanciation de la class *ConnectedThread*.

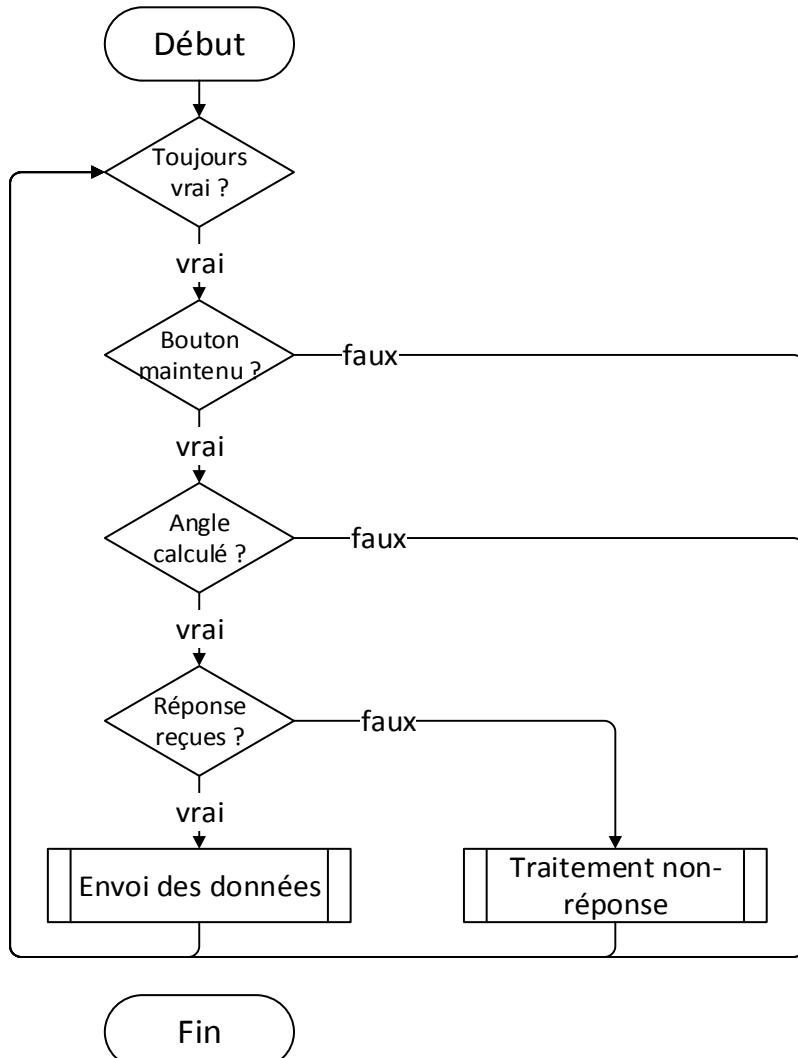
### 7.3.6. *ConnectedThread*

Une fois que la connexion avec un autre périphérique a lieu, il faut pouvoir communiquer avec celui-ci. Pour cela, on utilise la class « *ConnectedThread* » qui permet l'envoi et la réception des données. Cela est permis grâce aux méthodes suivantes :

- *read* : Pour récupérer la donnée reçue, on utilise cette méthode qui nous la retourne sous forme de chaîne de caractères.
- *write* : Il faut pouvoir envoyer des données. Pour cela, on se sert de cette méthode qui permet d'envoyer des données sous format de bytes.
- *cancel* : Permet de fermer le socket.

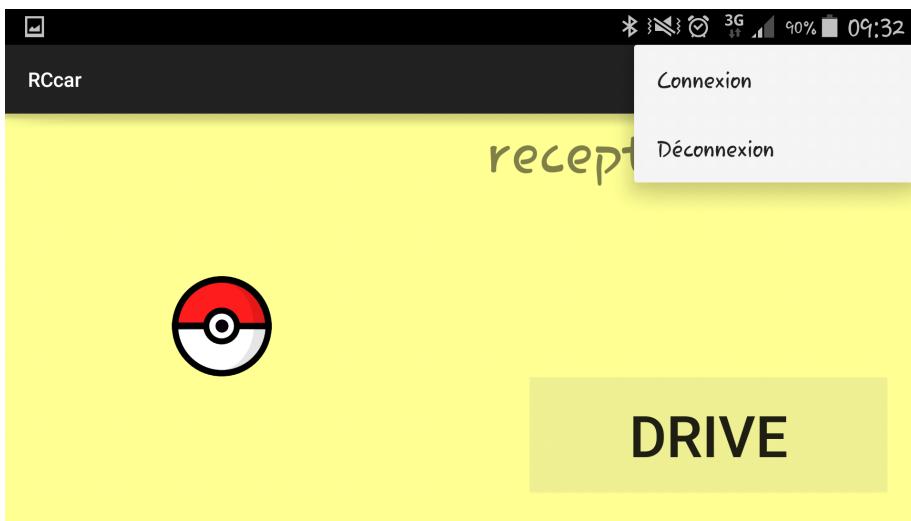
### 7.3.7. *Principe de fonctionnement*

Le programme est plus tôt complexe. C'est pour cela que le code source est fourni en annexe. Voici comment il fonctionne dans les grandes lignes :



### 7.3.8. L'application

Voici à quoi ressemble l'application développée :



Le bouton doit être maintenu pour que le véhicule puisse se déplacer. S'il est relâché, la voiture s'arrête. La « pokéball » est un témoin. Le but est de mettre un graphique autour, afin d'avoir un retour sur la puissance que l'on applique sur les roues de la voiture. En haut, à droite, se trouve le menu avec lequel on peut se connecter ou déconnecter avec un périphérique. Lorsqu'on clique sur la connexion, une fenêtre apparaît avec les différents périphériques à proximité et les périphériques déjà connu par le smartphone :



#### **7.4. Test Fonctionnel**

Au final, le prototype fonctionne correctement. Tout ne s'est pas fait d'un coup et les différents tests sont réalisés tout au long du déroulement du projet. Un test final avait été fait et le driver du moteur principal a grillé. Ce phénomène n'est pas vraiment expliqué. Il est fort probable que ce soit dû au bruit qu'il y a sur l'accumulateur lors de l'enclenchement du moteur principal. Ça a peut-être provoqué des « reset » sur les différents composants. Ce qui a ensuite engendré une fonctionnalité non-attendue. Quoi qu'il en soit, ce qui est certain, c'est qu'il y a eu un court-circuit qui a provoqué la destruction du driver du moteur.

A cause de ça, une carte a été refaite et la décision d'utiliser un deuxième accumulateur a été prise. Comme cela, nous évitons les dysfonctionnements du véhicule.

Au niveau de la consommation, le circuit se situe à 30 milliampères en moyenne. C'est bien moins que les 126 maximums prévu. C'est une bonne nouvelle. Comme cela, la carte électronique permet une autonomie suffisante pour utiliser le véhicule pendant un moment.



## **8. Conclusion**

### **8.1. Aspects techniques sur le projet**

Une bonne partie du projet a été réalisé. Il reste encore pas mal de travail, mais la difficulté du projet est grande. Principalement à cause de l'accéléromètre et en partie à cause d'Android. Le pilotage de la voiture fonctionne, on peut déjà s'en servir. Le capteur de distance quant à lui fonctionne tout à fait et empêche la voiture d'avancer si un obstacle se trouve devant celle-ci. La carte SD et le Real Time Clock n'ont pas été mis en place, car le temps était très limité. C'est principalement pour cela que le projet n'a pas été mené à terme.

Fonctions techniques	Pourcentage réalisé par catégorie			Apports personnels	Remarques / commentaires
	HW	SW	Doc		
Alimenter	100	-	100		Alimentation
Déplacer	100	100	100		Moteurs + driver
Protéger des chocs	100	100	100		Capteur distance
Anti-patinage	100	30	30		Capteur vitesse + accéléromètre
Enregistrer les données	100	0	0		Carte SD + RTC
Communiquer	100	80	70		Module Bluetooth
Traiter les informations	100	70	60		Microcontrôleur
Contrôler à distance	-	60	60		Application Android

### **8.2. Améliorations possibles**

Divers améliorations sont possibles. Pour commencer, il faut finir l'étude des différents éléments qui n'ont pas eu le temps d'être étudiés et les mettre en place. L'application Android doit être finalisée aussi. Une fois que cela est fait, on peut imaginer diverses petites améliorations sur la voiture. Une chose qui pourrait être fort sympathique, serait la mise en place d'une interface Android avec des boutons au lieu d'utiliser l'inclinaison du téléphone. Une chose qui pourrait être très intéressante aussi, serait d'utiliser une manette de console de jeux au lieu du smartphone.

### **8.3. Aspects gestions du projet**

Durant tout le projet, la gestion a été bonne. Il n'y a pas eu de problème, mise à part la gestion du temps. C'est le facteur qui a été très problématique. Il est très facile de se retrouver confronté à un problème et d'avoir du mal à le résoudre. Par ce fait, le temps passe et la solution tarde souvent à venir. C'est la seule problématique dans la plus part des projets. Mise à part ça, le projet a été bien géré.

### **8.4. Aspects personnels**

Ce projet aura été satisfaisant pour moi. Ça aura été une réelle opportunité de travailler sur un projet aussi captivant. Durant le projet, il y a eu plusieurs difficultés rencontrées. Souvent c'était des problèmes mineurs qui étaient résolu assez rapidement. D'autres fois, c'était des problèmes beaucoup plus importants comme par exemple le driver du moteur qui a brûlé. Il y a eu aussi la programmation Android. J'avais fait auparavant un cours au CFPT-Informatique sur la programmation Android et j'en avais fait beaucoup à la maison. Cela ne m'a pas empêché de m'être retrouvé face à des difficultés. Heureusement, débrouillard et autodidacte, j'ai su trouver des solutions plus ou moins fiable, mais fonctionnel.

Au bout du compte, si je devais refaire ce projet, je l'aurai fait mais en travaillant plus de temps et dès le début. C'est vrai qu'au tout début du projet, ce n'était pas facile à se mettre au travail, car la fin semblait si lointaine. De plus, la rédaction du rapport est souvent laissé de coter jusqu'au dernier moment, malheureusement. C'est à la fin qu'on se rend compte qu'il faut absolument faire le rapport et que le temps devient de plus en plus réduit. C'est pour cela que j'ai une forte impression que le rapport est bâclé. Je sais que j'aurai pu faire beaucoup mieux. Cependant, les informations principales sont présentes. L'essentiel est là.

En tout cas, ça aura été pour moi un réel plaisir de travailler sur un sujet si captivant. Je n'ai jamais eu un projet aussi enthousiaste et aussi intéressant. Je ressens cependant une certaine frustration dû au fait que le projet n'est pas fini à 100%. Ce n'est pas bien grave, car je verrai s'il y'a possibilité de garder la voiture, afin de la finaliser chez moi. Au cas échéant, au moins venir l'être quelques jours à l'école pour mettre en place au moins le système anti-patinage.

J'ai acquis une grande autonomie. J'ai appris à mieux gérer mon stresse aussi. Notamment, à la fin lorsqu'il a fallu commencer à faire le rapport de façon sérieuse. C'est malheureusement, toujours la partie la plus laborieuse et la moins intéressante. Il faut tout de même passer par là, pour que notre travail soit évalué et validé. J'espère que mon travail est satisfaisant et qu'il me permettra d'accéder à des études supérieures.