Synthèse projet d’électronique

Robot télécommandé

Un projet réalisé par Mathys Lopez, Ethan Reuchin, Sami Boujra, Martin Ducrocq et Simon Revanche.

Table des matières

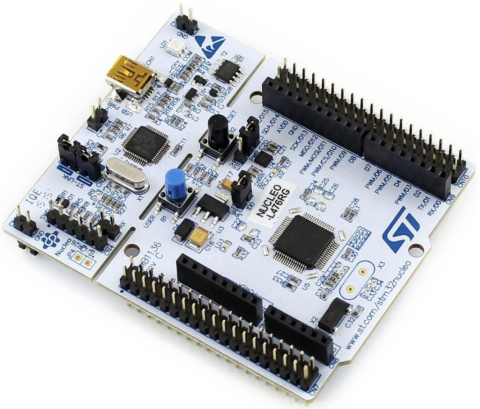
Introduction

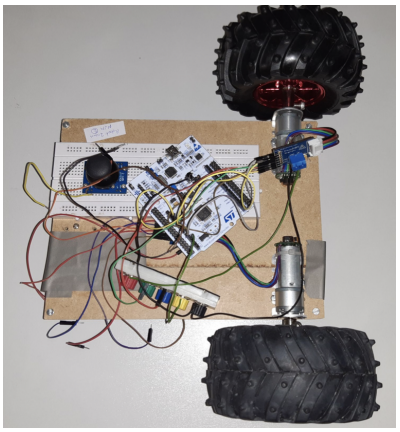
L'objectif de ce projet est de réaliser une machine dotée de deux roues motrices contrôlée à distance par une télécommande. Son usage peut être voué au loisir (jeu) ou à des utilisations professionnelles telles que le transport de charge ou des mesures spécifiques de terrain (dans ce dernier cas le robot sera équipé de capteurs non considérés par la suite).

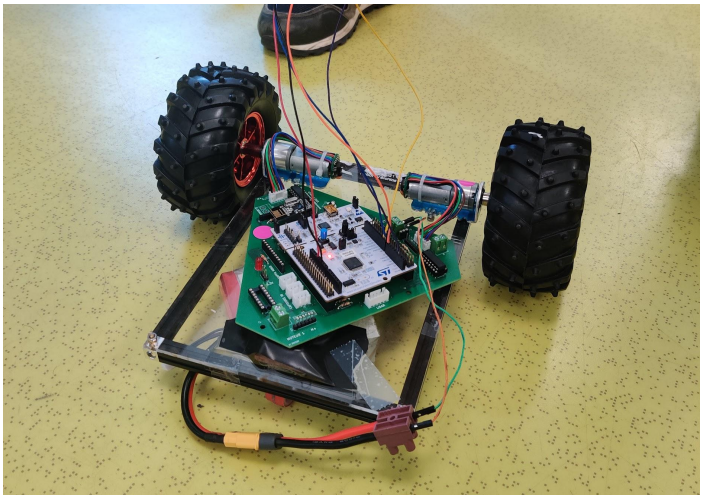
NB : Dans la suite, les termes "robot" et "voiture télécommandée" désignent la même chose.

Matériel utilisé

Plateforme : c'est la base physique de notre robot, celle sur laquelle repose la carte Nucleo, les moteurs et les circuits imprimés. Cette plateforme inclut deux moteurs à courant continu avec codeurs, ponts H, deux roues et un connecteur de batterie.
Carte Nucleo L476RG : elle sera l'intermédiaire entre le code informatique et le déplacement mécanique du robot. Avec cette dernière, on pourra contrôler la vitesse des moteurs et recevoir les commandes de la télécommande sans fil.
Les deux moteurs à courant continu seront reliés à la carte Nucleo et pilotables par la télécommande.
Une batterie : ce sera la source d'alimentation de l'ensemble de notre système.
Un module de communication Bluetooth qui permettra au robot de recevoir les commandes à distance.
Une télécommande : c'est à travers elle que l'utilisateur interagit avec le robot. Elle permettra de contrôler les mouvements du robot à distance.
Outils essentiels d'électronique : résistances, condensateurs, bouton poussoir, fils, fer à souder, circuits imprimés…







Notice d'utilisation / Scénario d'usage

Ce robot peut s'avérer utile dans de nombreuses situations : relevé de terrain, zones difficiles d'accès, jeu…Ainsi, le robot sera équipé de manière appropriée à son utilisation (caméra, capteurs…)

Avant toute manipulation, on s'assure que la batterie du robot est suffisamment chargée. Ensuite, on place le robot sur une surface accessible. On l'allume et on vérifie la connexion entre la télécommande et le robot. À l'aide de la télécommande, on peut mouvoir le robot : il peut avancer, reculer et tourner. À la fin de son utilisation, on éteint la voiture et la télécommande et on range le matériel.

Cahier des charges

Quantification des performances attendues et contraintes prescrites au robot.

1. Fonctionnalités :
    - La voiture doit pouvoir être contrôlée à distance
    - Elle doit pouvoir effectuer les commandes suivantes : avancer, reculer et tourner.
    - Ses mouvements doivent être précis et fluides
    - L'ajout d'éventuels capteurs et caméras embarqués doit être possible

2. Contraintes :
    - La plateforme du robot (fournie) doit être suffisamment compacte pour se déplacer dans des zones difficiles d'accès.

- La batterie (fournie avec la plateforme) doit avoir une autonomie suffisante pour le confort de l'utilisateur.
- La télécommande sans fil doit avoir une portée adéquate.
- La vitesse maximale atteignable par la voiture sera de 1,0 Km/h.
- Angle de pivotement : 360°. Temps de rotation : 2,5 s
- La voiture doit pouvoir être utilisée par un enfant d'au moins 2 ans (facilité d'utilisation).
- Le centre de gravité de la voiture doit se situer en son milieu pour un meilleur amortissement.
- Le câblage doit pouvoir être transporté sur le châssis de la voiture sans perturber sa course.

3. Sécurité :
    - La télécommande doit garantir une connexion sécurisée et un contrôle précis du robot.
    - Les normes de sécurité doivent être respectées (petites pièces à proscrire pour les enfants par exemple)
    - La voiture ne doit présenter aucun risque physique vis-à-vis de l'utilisateur

Objectifs

Nous aimerions que ce projet nous fasse progresser sur les compétences suivantes : progression en programmation, meilleure maîtrise des composants électroniques (moteurs, cartes arduino….), développement de notre esprit critique d'ingénieur (compréhension des documentations techniques, capacité d'adaptation face à un problème, recherche de solution en autonomie face aux problèmes rencontrés dans un premier temps…), amélioration de notre capacité à travailler en équipe (répartition des différentes tâches, centralisation des documents sur un serveur commun, planning prévisionnel…).

Diagramme fonctionnel du robot télécommandé

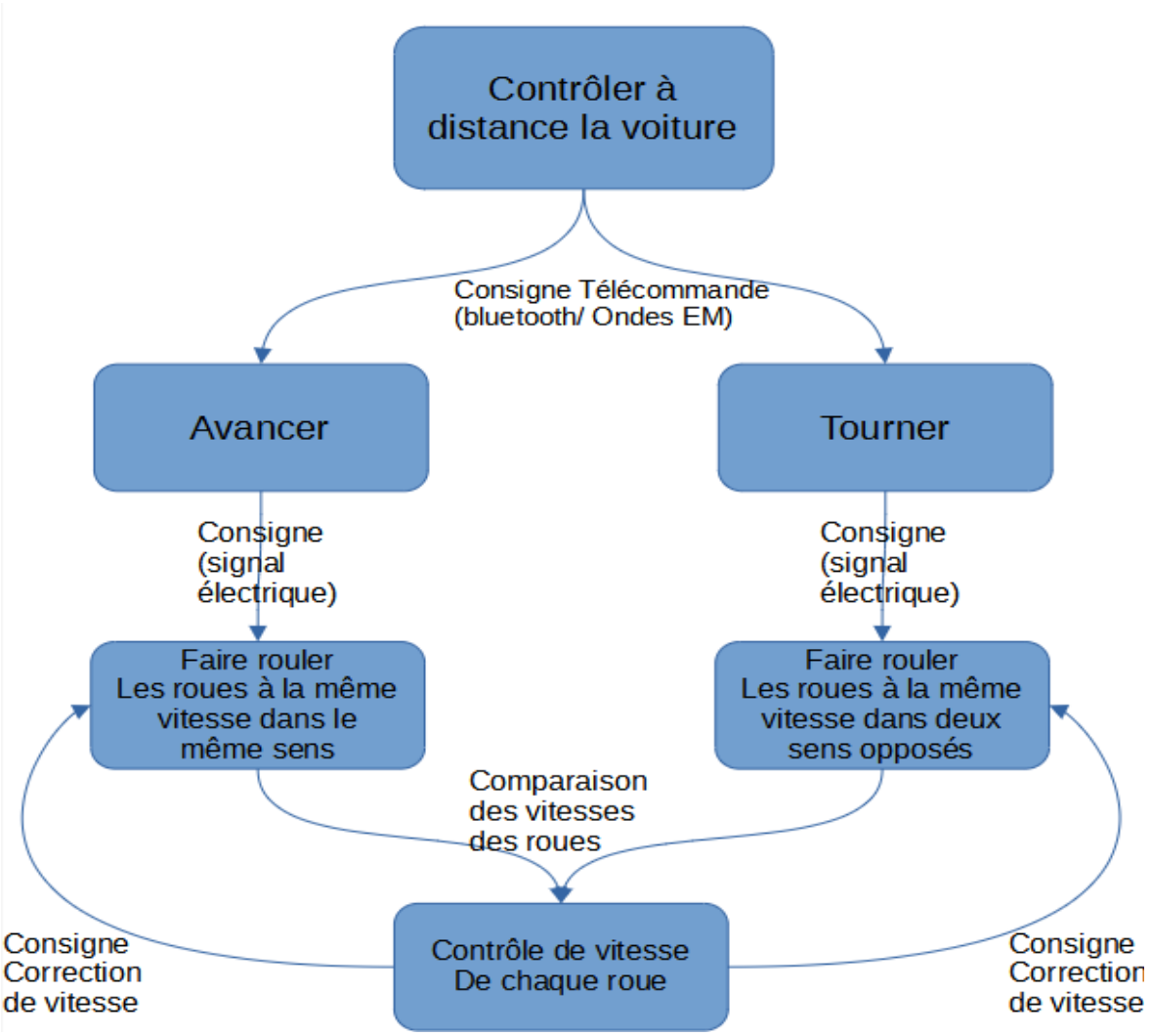
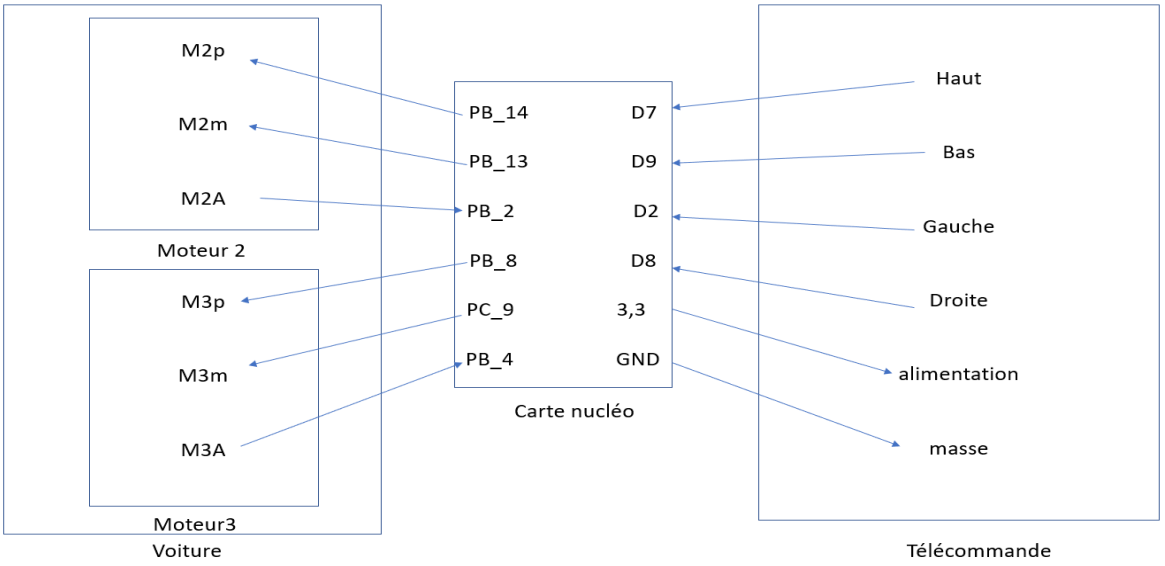


Schéma de principe



Ce schéma montre les différentes entrées et sorties qui relient la carte arduino à la télécommande et aux moteurs, les entrées numériques D… sont associées aux différentes directions, les sorties numériques PB_14, PB_13 ,PB_8 et PC_9 permettent de commander les moteurs et les entrées numériques PB_2 et PB_4 reçoivent des signaux en provenance des moteurs qui renseignent la carte arduino sur la vitesse des moteurs.

Rétro-planning

Numéro de séance	Actions réalisées
Séance 1	- Familiarisation du projet : vue globale du travail à effectuer. - Réalisation du cahier des charges et de la notice. - Codage des fonctions élémentaires pour avancer, reculer et faire tourner la voiture.
Séance 2	- Conception de la carte électronique (correction des faux-contacts et évite de réaliser les mêmes branchements à chaque début de séance). - Idée du joystick : modélisation mathématique.
Séance 3	- Abandon du joystick. Utilisation d'une flèche directionnelle. - Changement de la carte électronique par une autre plus adaptée.
Séance 4	- Création de la commande avec fil : Soudure du branchement de la carte électronique.
Séance 5	- Test et création du premier code sans asservissement. Le code était globalement juste et fonctionnel. - Soudure car apparition de bruits entre les connexions.
Séance 6	- Mise en place de l'asservissement. - Changement du code adapté à l'asservissement.
Séance 7	- Changement de la commande qui prend en compte la suppression des rebonds des commandes. - Finalisation des livrables.
Séance 8	- Présentation du projet.

Description des séances

Dans un premier temps, nous avons cherché à coder des briques de base pour la voiture électronique. Il s'agissait donc de coder les fonctions élémentaires pour avancer, reculer et tourner. Cette étape a été réalisée très vite au cours de la première séance.

Ensuite, nous voulions commander la voiture avec une télécommande dotée d'un joystick. L'idée a rapidement été abandonnée. On souhaitait quantifier la tension communiquée au moteur selon l'inclinaison du joystick par rapport à sa position d'équilibre. Nous avons essayé de mettre en équation ce problème : il s'agissait de considérer l'équation d'un cercle et d'associer une tension d'entrée en fonction du rayon du cercle. Nous nous sommes rendus compte que ce modèle n'était pas applicable à notre cas pratique.

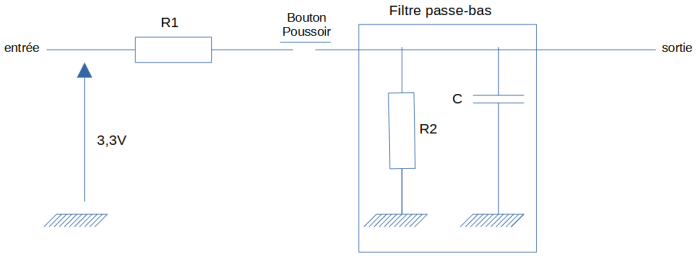
Nous avons préféré sécuriser nos arrières en modélisant, dans un premier temps, des flèches directionnelles (pour avancer, reculer, gauche, droite). La modélisation était plus adaptée aux contraintes de temps auxquelles nous étions soumis pour obtenir des résultats efficaces. Nous avons reçu l'aide des professeurs pour apprendre à souder les différents composants de la télécommande. Cette étape était cruciale pour éviter de reconstruire la télécommande à chaque séance et éviter les faux contacts.

À partir de là, les bugs et les dysfonctionnements n'ont pas cessé, d'une semaine à l'autre la voiture ne fonctionnait plus et l'on passait une bonne partie de nos séances à résoudre les problèmes de faux contacts, télécommande mal soudée, erreurs de code et latence des moteurs. Cela a représenté une perte de temps considérable qui a limité notre marge de manœuvre pour améliorer la voiture.

Nous avons également corrigé l'asservissement de la vitesse des moteurs. Même si les moteurs sont en théorie identiques, ils réagissent différemment à des consignes analogues. Par exemple, si l'on demande à la voiture d'aller en ligne droite (les deux moteurs doivent tourner dans le même sens à la même vitesse), on remarque rapidement que la voiture diverge (avance "en biais" sur la droite ou la gauche), si aucun asservissement est fait. Cette amélioration représente beaucoup pour notre projet car elle nous a permis de comprendre plus en profondeur comment marche un moteur et en tirer le meilleur.

Certains rendez-vous avec le professeur encadrant étaient fixés le jeudi après-midi afin de corriger les erreurs suivantes:

- Amélioration des flèches directionnelles avec intégration d'un filtre passe-bas



(système anti-rebonds)

- Adaptation de l'asservissement : meilleur calcul de l'erreur de vitesse entre les deux moteurs (on prend le moteur M2 en référence).
- Problèmes de soudure.

Détail de chaque fonction

Vocabulaire

M2 (resp.M3) : nom associé au moteur 1 (resp.2)
M2-A et M3-A : réponses des moteurs indiquant sur leur vitesse
M2p et M2m (resp. M3p et M3m) : indique le sens d'alimentation pour faire tourner la roue associée au moteur 2 (resp. 3) dans un sens donné.
p signifie "plus" pour faire avancer la voiture.
m signifie "moins" pour faire reculer la voiture.

Les différentes fonctions

Fonction tourner à droite, cette fonction sert à faire tourner le véhicule à droite lorsqu'on appuie sur la touche droite, on envoie un signal sur les broches "plus" des moteurs 2 et 3, et 0 sur les broches "moins". Exemple:

```
if (R==1)//on tourne à droite
{
M2p.pulsewidth_us(4000);
M2m.pulsewidth_us(0);
M3p.pulsewidth_us(4000);
M3m.pulsewidth_us(0);
//printf("r, R=%i\n",R);
}
```

Fonction tourner à gauche, cette fonction sert à faire tourner le véhicule à gauche lorsqu'on appuie sur la touche gauche, on envoie un signal sur les broches "moins" des moteurs 2 et 3, et 0 sur les broches "plus". Exemple:

```
else if (L==1)//on tourne à gauche
{
M2p.pulsewidth_us(0);
M2m.pulsewidth_us(4000);
M3p.pulsewidth_us(0);
M3m.pulsewidth_us(4000);
//printf("l, L=%i\n",L);
}
```

Fonction avancer, cette fonction sert à faire avancer le véhicule lorsqu'on appuie sur la touche haut, on envoie un signal sur la broche "plus" du moteur 2 et la broche "moins" du moteur 3, et 0 sur la broche "moins" du moteur 2 et la broche "plus" du moteur 3. Exemple:

```
else if (U==1)//on avance
{
M2p.pulsewidth_us(4000);
M2m.pulsewidth_us(0);
M3p.pulsewidth_us(0);
M3m.pulsewidth_us(4000+dx);
//printf("u, U=%i\n",U);
}
```

Fonction reculer, cette fonction sert à faire reculer le véhicule lorsqu'on appuie sur la touche bas, on envoie un signal sur la broche "moins" du moteur 2 et la broche "plus" du moteur 3, et 0 sur la broche "plus" du moteur 2 et la broche "moins" du moteur 3. Exemple:

```
else if (D==1)//on recule
{
M2p.pulsewidth_us(0);
M2m.pulsewidth_us(4000);
M3p.pulsewidth_us(4000+dx);
M3m.pulsewidth_us(0);
//printf("d, D=%i\n",D);
}
```

Fonctions compte2 et compte3 on compte le nombre de période des signaux (impulsions) qui sont envoyés en réponse de chaque moteur pour avoir une idée de leur différences de vitesse. Exemple:

```
void compte2()//on définie la fonction compte2
{
N2=N2+1;//on itère N2
}
void compte3()
{
N3=N3+1;//on itère N3
}
```

Fonction compare on compare le nombre d'impulsions en réponse des deux moteurs et on en déduit une correction du moteur M3 par rapport au à M2 (la référence) .Cette correction (dx) sera prise en compte par les fonctions avancer et reculer. On réinitialise le nombre d'impulsions comptées par les fonctions compte2 et compte3 . Exemple:

```
void compare()//on définie la fonction compare
{
 dx=floor(4000*(N2-N3)/N3);//on compare N2 et N3 et on corrige l'écart avec la variable dx
 printf("N3=%i,N2=%i,dx=%i",N3,N2,dx);
 printf("\n");
 N2=0;//on remet N2 à 0
 N3=0;//on remet N3 à 0
}
```

Méthodologie pour souder à l'étain

Voici la méthodologie que nous avons suivie pour souder les composants.

Dans un premier temps, on allume le fer à souder et on attend quelques instants le temps qu'il chauffe. On n'hésite pas à nettoyer le bout du fer pour éliminer les résidus des utilisations précédentes. On peut éventuellement raccourcir les fils des composants électroniques (résistances, condensateurs…) en fonction de leur emplacement sur la carte. Ensuite on insère les composants dans les trous de la carte. On tient le fer à souder d'une main et l'étain de l'autre. On chauffe le bout du fer à souder sur la connexion entre le composant et la carte. On applique ensuite l'étain au même endroit : celui-ci se répartit uniformément. Il est important de ne pas trop mettre d'étain au risque de faire des courts circuits ou de mal souder les connexions. Chaque soudure doit être bien formée et brillante.

Critique des performances / possibilité d'amélioration

Le robot télécommandé est capable de réaliser les fonctions qu'on attend de lui : il avance, recule et tourne de manière précise. La télécommande est évidemment le point faible de notre projet. En effet, la télécommande doit être systématiquement accordée au robot pour qu'il fonctionne. Nous avons passé trop de temps à corriger la télécommande avec fil et ainsi pas pu créer la télécommande sans fil. Voici quand même une idée non aboutie pour faire fonctionner une télécommande sans fil.

On utilise un module de communication sans fil Bluetooth HC-05 qu'on connecte à la carte Nucléo et qui va permettre d'établir une communication sans fil avec le microcontrôleur.

Le code configure la communication série entre la carte Nucléo et le module Bluetooth HC-05. Il lit les commandes envoyées par le module Bluetooth puis les traite pour contrôler les mouvements de la voiture. On utilise ensuite un appareil compatible Bluetooth (un smartphone par exemple) pour envoyer les commandes sans fil à la voiture télécommandée. Pour ce faire, on peut utiliser une app qui communique avec le module Bluetooth. Lorsqu'on code cette application, on décide d'un format spécifique pour les commandes à envoyer à la voiture : ‘'A' pour avancer, 'R' pour reculer, 'G' pour tourner à gauche, 'D' pour tourner à droite… On envoie ainsi ces commandes au module Bluetooth.

Voici un exemple de code pour la communication entre la carte Nucléo et le module Bluetooth.

```
#include "mbed.h"
#include "Serial.h" //librairie spécifique de mbed pour la communication série.

Serial pc(USBTX, USBRX); // Communication série avec le PC
Serial bluetooth(PA_9, PA_10); // Communication série avec le module Bluetooth

int main() {
   pc.baud(9600); // vitesse de communication série carte <--> PC
   bluetooth.baud(9600); // vitesse de communication série carte <--> module Bluetooth

   while(1) {
      if(bluetooth.readable()) { //est-ce qu'une commande a été reçue ?
         char command = bluetooth.getc(); //lecture et stockage de la commande
         //traitement des commandes reçues ici

      }
   }
}
```

Bilan

Nous avons retenu plusieurs choses de l'expérience de ce projet. Tout d'abord, nous avons pris conscience de l'écart entre ce qui est censé fonctionner et ce qui fonctionne en pratique : nous avons passé trop de temps à corriger des erreurs et des bugs au lieu de progresser sur de nouveaux aspects du projet. C'est assez frustrant car on a l'impression de stagner d'une semaine à l'autre. Nous sommes aussi frustrés de voir que nous n'avons pas pu amener le projet aussi loin qu'on le souhaitait en termes de fonctionnalités (télécommande sans fil, joystick…).

Compétences acquises :

De façon générale, notre savoir-faire pratique en électronique a progressé.

Tout d'abord, nous avons appris à souder des fils d'étain pour notre carte directionnelle. La soudure permet à la fois de miniaturiser les composants, d'empêcher les faux-contacts et d'éviter de recommencer le montage à chaque nouvelle séance. On utilise des fibres en cuivre si on souhaite enlever l'étain (capillarité) et recommencer notre soudure.

Nous avons également appris à asservir les moteurs de la voiture. En pratique, les roues ne sont pas rigoureusement identiques et des écarts de vitesse entre les deux moteurs peuvent apparaître. Il est donc très intéressant de corriger cet écart en asservissant les deux moteurs, cependant par manque de temps nous avons seulement pu asservir un moteur en boucle ouverte en utilisant le deuxième comme référence.

Nous avons également progressé en termes de démarche de l'ingénieur et d'esprit critique. Si nous ne comprenons pas une erreur, ou un problème auquel nous faisons face, nous savons où chercher l'information. Nous devions choisir les solutions les plus adaptées à nos compétences. De plus, nous sommes conscients des défauts de notre projet et nous cherchons toujours à les dépasser.
Par exemple, nous avons appris à analyser les fiches techniques des différents composants électroniques. La recherche constante d'amélioration de notre technologie a aussi été un point sur lequel nous avons progressé. Par exemple, notre première carte directionnelle que nous avons soudé fonctionnait, mais nous avons préféré la recommencer pour enlever l'effet de rebond des boutons.

Annexe

Code final

```
#include "mbed.h"
#include "math.h"

Ticker log_timer;//importation du timer

//initialisation des différentes entrées
DigitalIn right (D8); //direction droite
DigitalIn left (D2); //direction gauche
DigitalIn up (D7); //direction haute
DigitalIn down (D9); //direction basse

//initialisation des différentes sorties
PwmOut M2p (PB_14);//broche + moteur 2
PwmOut M2m (PB_13);//broche - moteur 2
PwmOut M3p (PB_8);//broche + moteur 3
PwmOut M3m (PC_9);//broche - moteur 3

//initialisation des différentes entrées d'interuption
InterruptIn  M2_A (PB_2);//signal réponse du moteur 2
InterruptIn  M3_A (PB_4);//signal réponse du moteur 3

//initialisation des différentes variables
int N2=0;//nombre de période (créneau) envoyé par signal réponse du moteur 2 pendant 0.1ms
int N3=0;//nombre de période (créneau) envoyé par signal réponse du moteur 3 pendant 0.1ms
int dx=0;//variable d'asservissement (erreur)

//initialisation des fonctions
void compte2(void);//fonction qui compte le nombre de période (créneau) envoyé par signal réponse du moteur 2
void compte3(void);//fonction qui compte le nombre de période (créneau) envoyé par signal réponse du moteur 3
void compare(void);//fonction d'asservissement qui corrige les écarts de vitesse entre le moteur 1 et 2

int main()

{
    //initialisation des périodes des signaux de sorties à 10000us
    M2p.period_us(10000);
    M2m.period_us(10000);
    M3p.period_us(10000);
    M3m.period_us(10000);

    log_timer.attach(&compare,0.1);//fonction d'interruption qui appelle la fonction compare toute les 100ms
```

```
log_timer.attach(&compare,0.1);//fonction d'interruption qui appelle la fonction compare toute les 100ms


M2_A.rise(&compte2);//fonction d'interuption qui appelle la fonction compte2 lorsqu'un front montant apparaît sur le signal réponse du moteur 2
M3_A.rise(&compte3);//fonction d'interuption qui appelle la fonction compte2 lorsqu'un front montant apparaît sur le signal réponse du moteur 3


while(1)//boucle infini
{
    //lecture des touches appuyés
    int R = right.read();
    int L = left.read();
    int U = up.read();
    int D = down.read();

    //affiliation des différents rapports cycliques en fonction des touches appuyées
    if (R==1)//on tourne à droite
    {
    M2p.pulsewidth_us(4000);
    M2m.pulsewidth_us(0);
    M3p.pulsewidth_us(4000);
    M3m.pulsewidth_us(0);
    //printf("r, R=%i\n",R);
    }

    else if (L==1)//on tourne à gauche
    {
    M2p.pulsewidth_us(0);
    M2m.pulsewidth_us(4000);
    M3p.pulsewidth_us(0);
    M3m.pulsewidth_us(4000);
    //printf("l, L=%i\n",L);
    }

    else if (U==1)//on avance
    {
    M2p.pulsewidth_us(4000);
    M2m.pulsewidth_us(0);
    M3p.pulsewidth_us(0);
    M3m.pulsewidth_us(4000+dx);
    //printf("u, U=%i\n",U);
    }
```

```
        else if (U==1)//on avance
        {
        M2p.pulsewidth_us(4000);
        M2m.pulsewidth_us(0);
        M3p.pulsewidth_us(0);
        M3m.pulsewidth_us(4000+dx);
        //printf("u, U=%i\n",U);
        }
        else if (D==1)//on recule
        {
        M2p.pulsewidth_us(0);
        M2m.pulsewidth_us(4000);
        M3p.pulsewidth_us(4000+dx);
        M3m.pulsewidth_us(0);
        //printf("d, D=%i\n",D);
        }
        else// la voiture est immobile
        {
        M2p.pulsewidth_us(0);
        M2m.pulsewidth_us(0);
        M3p.pulsewidth_us(0);
        M3m.pulsewidth_us(0);
        }
    }
}
void compare()//on définie la fonction compare
{
 dx=floor(4000*(N2-N3)/N3);//on compare N2 et N3 et on corrige l'écart avec la variable dx
 printf("N3=%i,N2=%i,dx=%i",N3,N2,dx);
 printf("\n");
 N2=0;//on remet N2 à 0
 N3=0;//on remet N3 à 0
}
void compte2()//on définie la fonction compte2
{
N2=N2+1;//on itère N2
}
void compte3()
{
N3=N3+1;//on itère N3
}
```