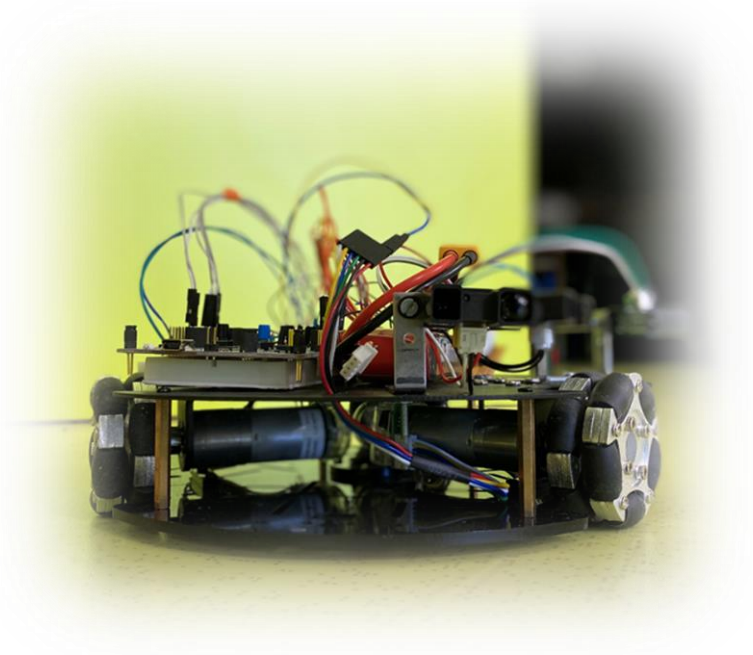
INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech | LEnsE Laboratoire d'Enseignement Expérimental

Robot Omnidirectionnel

Rapport Technique



Projet réalisé dans le cadre de l'enseignement IETI par :

Youssef SAGHIRAN

Hugo LASSIETTE

Audrey ATTIA

Sommaire

I. Introduction

Présentation générale :

Notre projet consiste à réaliser un robot avec des roues omnidirectionnelles qui se déplace et évite les obstacles sans être télécommandé.

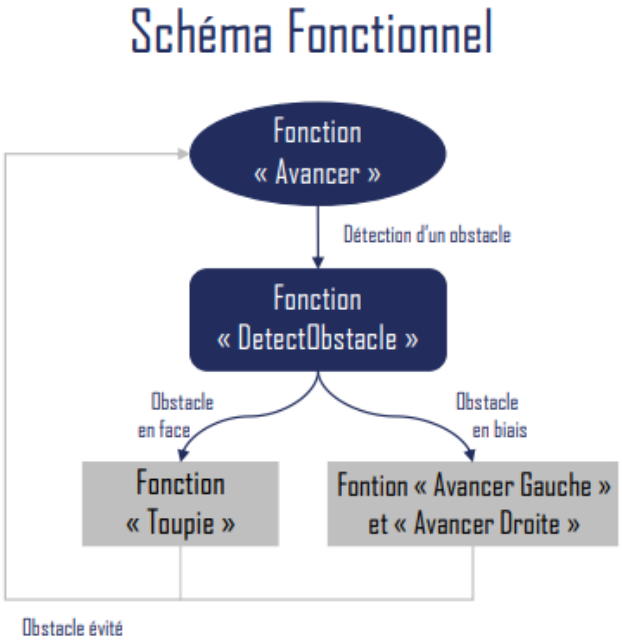
Objectif principal :

Programmer un robot omnidirectionnel non télécommandé qui détecte les obstacles.

Objectifs Secondaires :

- Commander les moteurs par la carte Nucléo
- Acquérir l'information sur la distance de l'obstacle par les capteurs
- Adapter le comportement du robot à l'information acquise par les capteurs
- Alimenter le système par une batterie au lieu des générateurs

II. Découpage fonctionnel



- La fonction ' Avance ' permet de faire tourner deux roues dans sens opposé et de bloquer la dernière (ici R2 associée au capteur C2) pour qu'elle ne tourne pas.

- La fonction ' Detectobstacle ' permet de changer le mouvement du robot en cas de détection. Si un obstacle est détecté, le robot peut agir des deux manières suivantes.

  1er cas : l'obstacle est détecté par C2

- La fonction toupie du robot engendre la rotation du robot sur lui-même en faisant tourner les 3 roues dans le même sens jusqu'à ce que le capteur C2 ne détecte plus d'obstacle.

  2ème cas : l'obstacle est détecté par un des capteurs périphériques

- La fonction stopper du robot s'active quand le robot détecte un obstacle avec le capteur C2. Elle permet de stopper l'avancée du robot.

- Les fonctions ' Avancer Gauche ' et ' Avancer Droite ' permettent de faire un arc de cercle. On les lance simultanément, les deux roues ne tournent pas à la même vitesse.

III. Schéma électrique

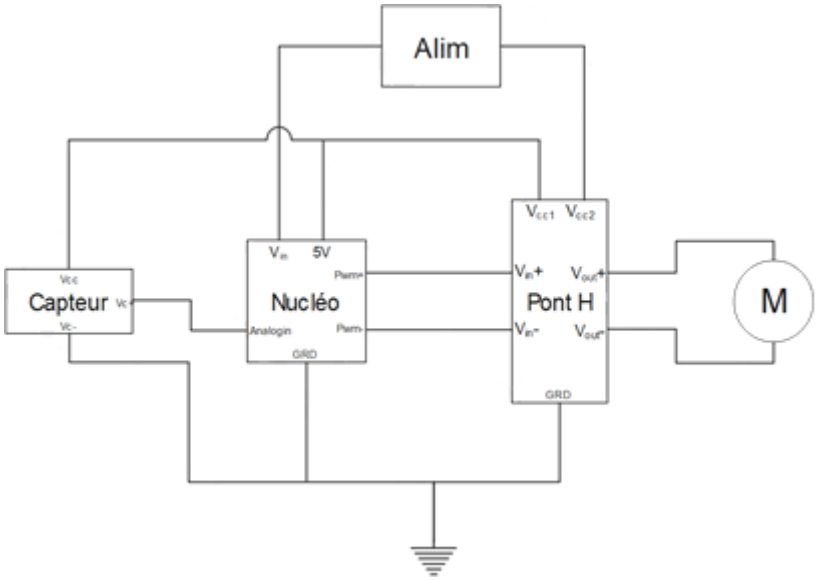


Schéma électrique pour un moteur et un capteur

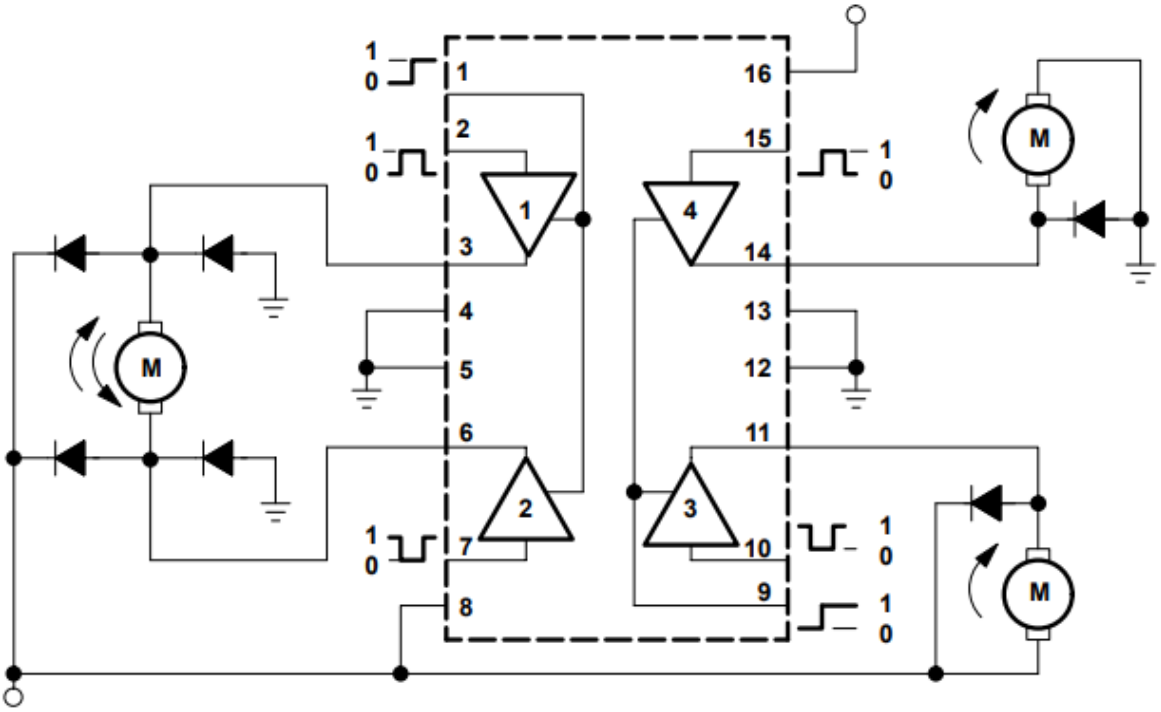
Alimentation : Fournit le système en électricité. Dans notre cas, on l'utilise pour alimenter la carte nucléo et les moteurs (via V_{cc2}). En pratique, on a utilisé une batterie LiPo 11V.

Carte Nucléo : Permet d'envoyer des signaux binaires PWM prédéterminés aux ponts H en fonction des informations transmises par les capteurs. On utilise le port 5V de la carte pour alimenter les capteurs et pour fournir V_{cc1} ainsi qu'un échelon (non indiqué sur le schéma) nécessaire au bon fonctionnement du pont H.

Ponts H : Il y en a 3 (un par moteur). On les utilise pour contrôler l'alimentation des moteurs à partir des informations transmises par la carte nucléo. On l'utilise de façon à pouvoir faire tourner le moteur dans les deux sens.

Capteurs : On utilise 3 capteurs qui renvoient chacun une tension à la carte nucléo en fonction de la présence ou non d'un obstacle.

Branchements sur le pont H :



Ports utilisés sur le pont H :

16 : V_{cc1}

8 : V_{cc2}

1 : échelon nécessaire au fonctionnement mais pas représenté sur le schéma

2 : $V_{in}+$

3 : $V_{out}+$

4 : Masse

6 : $V_{out}-$

7 : $V_{in}-$

Selon si on envoie un signal en $V_{in}+$ ou en $V_{in}-$, on a en sortie $V_{out}+ - V_{out}- = \pm V_{cc2}$

IV. Algorithmes

Il s'agit maintenant de programmer le robot de sorte à répondre aux attentes. On a donc réalisé le programme suivant que l'on a annoté directement sur Mbed.

```
#include "mbed.h"
// Sous MBED 2 !!

#define     FINAL_VALUE     200
#define     PERIODE_STEP    0.05

#define     DISTANCE_L_MAX  2.0
#define     DISTANCE_S_MAX  2.0



Serial      pc(USBTX, USBRX);
Serial      bt(PC_10, PC_11);
Ticker      mainStep;

// Sorties PWM vers les 3 ponts H
//Les "a" correspondent à une rotation dans le sens positif
//Les "b" correspondent à une rotation dans le sens négatif
PwmOut      moteurA_a(D11);
PwmOut      moteurA_b(D12);
PwmOut      moteurB_a(D8);
PwmOut      moteurB_b(D9);
PwmOut      moteurC_a(D14);
PwmOut      moteurC_b(D15);

// Entrées analog pour les capteurs distance IR
AnalogIn    captL(A0);
AnalogIn    captS_D(A1);
AnalogIn    captS_G(A4);

// Variables pour stocker les valeurs acquises par les capteurs
float distanceL = 0;
float distanceS_D = 0;
float distanceS_G = 0;


// Fonction d'initialisation des valeurs de pwm
void initRobot(void){
    pc.printf("ROBOT - Init...\r\n");
    bt.printf("ROBOT - Init...\r\n");
    moteurA_a.period_ms(2);
    moteurA_a.write(0);
    moteurA_b.period_ms(2);
    moteurA_b.write(0);
    moteurB_a.period_ms(2);
    moteurB_a.write(0);
    moteurB_b.period_ms(2);
    moteurB_b.write(0);
    moteurC_a.period_ms(2);
    moteurC_a.write(0);
    moteurC_b.period_ms(2);
    moteurC_b.write(0);

    pc.printf("ROBOT - OK\r\n");
    bt.printf("ROBOT - OK\r\n");
}

```

```
58  // Fonction stopper
59  void stopper(void){
60      moteurA_a.write(0);
61      moteurA_b.write(0);
62      moteurB_a.write(0);
63      moteurB_b.write(0);
64      moteurC_a.write(0);
65      moteurC_b.write(0);
66  }
67
68  // Fonction avancer
69  void avancer(float vitesse){
70      stopper();
71      //On fait tourner deux roues dans des sens opposés pour faire avancer le robot
72      moteurA_a.write(vitesse);
73      moteurB_b.write(vitesse);
74  }
75  //Les deux fonctions suivantes servent à ne faire tourner qu'une seule roue
76
77  // Fonction avancer gauche
78  void avancer_gauche(float vitesse){
79      moteurA_a.write(vitesse);
80  }
81  // Fonction avancer droite
82  void avancer_droite(float vitesse){
83      moteurB_b.write(vitesse);
84  }
85
86  // Fonction toupie
87  // Fait tourner les 3 roues dans le même sens
88  //Prend le sens (+1 ou -1) et la vitesse en argument
89  void toupie(float vitesse, int sens){
90      stopper();
91      if(sens == 1){
92          moteurA_a.write(vitesse);
93          moteurB_a.write(vitesse);
94          moteurC_a.write(vitesse);
95      }
96      else{
97          if(sens == -1){
98              moteurA_b.write(vitesse);
99              moteurB_b.write(vitesse);
100             moteurC_b.write(vitesse);
101         }
102         else{
103             stopper();
104         }
105     }
106 }
107
108 // Fonction de récupération des données
109 // Calcule la tension renvoyée par les capteurs
110 void collectData(void){
111     distanceL = captL.read() * 3.3;
112     distanceS_D = captS_D.read() * 3.3;
113     distanceS_G = captS_G.read() * 3.3;
114     // Les distances sont en Volts
115     pc.printf("DL = %.2lf / DD = %.2lf / DG = %.2lf\r\n", distanceL, distanceS_D, distanceS_G);
116 }
```

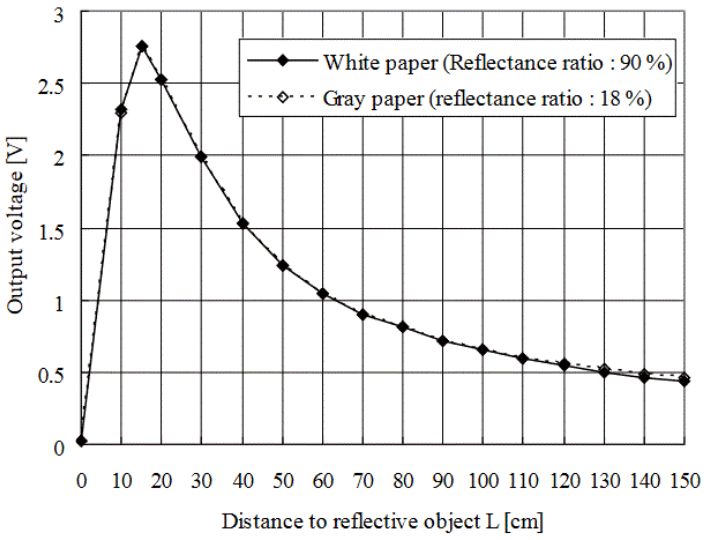
```
118 // Fonction de détection
119 // Fais le lien entre les signaux reçu par les capteurs et les actions du robot
120 void detectObstacle(void){
121     // Si le capteur central détecte un objet le robot tourne sur lui même
122     if(distanceL > DISTANCE_L_MAX){ // Objet Avant
123         stopper();
124         pc.printf("Objet_L\r\n");
125         if(distanceS_D > distanceS_G){
126             pc.printf("Turn_1\r\n");
127             while(distanceL > DISTANCE_L_MAX){
128                 collectData();
129                 toupie(0.3, 1);
130             }
131         }
132         else{
133             pc.printf("Turn_2\r\n");
134             while(distanceL > DISTANCE_L_MAX){
135                 collectData();
136                 toupie(0.3, -1);
137             }
138         }
139     }
140     else{
141         // Si seul le capteur de droite détecte un objet, virage à gauche
142         if(distanceS_D > DISTANCE_S_MAX){   // Objet coté droit
143             pc.printf("Droite_\r\n");
144             while(distanceS_D > DISTANCE_S_MAX){
145                 collectData();
146                 stopper();
147                 avancer_gauche(0.4);
148                 avancer_droite(0.2);
149             }
150         }
151         else{
152            // Si seul le capteur de gauche détecte un objet, virage à droite
153            if(distanceS_G > DISTANCE_S_MAX){   // Objet coté gauche
154                 pc.printf("Gauche_\r\n");
155                 while(distanceS_G > DISTANCE_S_MAX){
156                     collectData();
157                     stopper();
158                     avancer_droite(0.4);
159                     avancer_gauche(0.2);
160                 }
161             }
162             else{
163                 // Si les cateurs ne détectent rien, le robot avance tout droit
164                 pc.printf("Go\r\n");
165                 avancer(0.4);
166             }
167         }
168     }
169 }
170 


179 // Fonction principale
180 int main() {
181     initRobot();
182     while(1) {
183         // acquisition des données
184         collectData();
185         // Réaction du robot en conséquence
186         detectObstacle();
187         wait(1);
188 
189     }
190 }
```

V. Tests et Validation

Test des capteurs :

Vérification expérimentale en rapprochant progressivement un obstacle du capteur de la courbe spécifique au capteur utilisé de la tension en fonction de la distance au capteur. Les capteurs à notre disposition étaient les Sharp GP2Y0A02YK0F.

Fig. 2 Example of distance measuring characteristics (output)



Test des modes de marche du moteur :

Afin de tester la réponse des trois moteurs aux fonctions ''Avance'', ''Toupie'' et ''Stopper'', on a alimenté le robot avec une alimentation externe et on a stimulé

chacun des capteurs en plaçant un obstacle devant. Puis on a testé ultérieurement ces différentes fonctions en plaçant dans le couloir le robot (qui ne possédait à ce moment qu'un seul capteur) alimenté par une batterie externe. On a remarqué qu'avec une certaine inclinaison le robot ne détectait pas les obstacles. Donc on a rajouté deux capteurs latéraux. De plus, on a remarqué que la vitesse du robot était telle qu'il n'avait pas le temps de réagir avant de rencontrer l'obstacle ce qui nous a amené au test qui suit.

Tests de la vitesse du robot :

On a effectué différents tests de vitesse du robot de sorte à voir quelle vitesse serait adaptée pour que le robot est le temps d'adapter son comportement lorsqu'il détecte un obstacle

VI. Planning

Séance	Tâches
Séance 1 du 25/01/21	• Assemblage du robot omnidirectionnel • Assimilation du projet avec les attendus
Séance 2 du 01/02/21	• Élaboration d'une solution pour le fonctionnement du robot : Comment le programmer de sorte qu'il évite les obstacles ? Quel circuit est adapté ? • Test du capteur central et assemblage de ce capteur sur le robot

Séance 3 du 10/03/21	• Réalisation d'un circuit électrique • Écriture des fonctions "avance", "stoppe", "toupie" • Test de chaque fonction
Séance 4 du 24/03/21	• Ajouter deux capteurs latéraux pour éviter que le robot rentre dans les murs s'il arrive avec une inclinaison sur un obstacle • Créer un programme complet qui intègre toutes les fonctions • Test du robot avec une alimentation externe
Séance 5 du 12/04/21	• Optimisation du circuit électrique • Test du robot à l'aide d'une batterie externe
Séance 6 du 10/05/21	• Résolution des derniers soucis techniques
Séance 7 du 19/05/21	• Tester que le robot fonctionne toujours • Optimisation du programme avec ajout des fonctions 'avance gauche' et 'avance droite' • PowerPoint pour la présentation du 26/05/21
Séance 8 du 26/05/21	• Présentation orale accompagnée d'une démonstration

VII. Analyse du travail :

Difficultés rencontrées :

- Au début, on a utilisé un seul capteur (Sharp 150 cm) centré sur l'avant du robot. Nous nous sommes aperçus que le champ de vision du capteur n'était pas suffisant pour éviter toutes les collisions. Nous avons donc choisi de rajouter deux capteurs supplémentaires (Sharp 80 cm) sur les côtés pour palier à ce problème.

- Par la suite, nous avions toujours des problèmes de robot qui percute les murs. Nous avons donc remplacé les capteurs périphériques par des capteurs de plus longue portée (150 cm)

- Un autre problème était la vitesse du robot. Il se déplaçait trop vite pour avoir le temps de détecter les murs. Le fait qu'on utilise des signaux PWM devait nous permettre de modifier cette vitesse mais cela n'a pas fonctionné à cause d'erreurs dans le programme. Au lieu d'initialiser les variables PWM une fois au début, il y avait une initialisation dans chaque sous-fonction ce qui a conduit à un dysfonctionnement.

- Quand nous avons commencé à alimenter le robot par une source externe, nous avons brûlé quelques cartes Nucléo car nos branchements n'étaient pas optimisés. Nous avons essayé de tout alimenter à partir de la carte Nucléo ce qui conduisait à dépasser la limite de courant de la carte. Nous avons alimenté les moteurs directement à partir de la batterie à la place, ce qui a résolu le problème.

Répartition des tâches :

Hugo :

- Test des ponts H
- Câblage
- Code
- Schémas électriques

Youssef :

- Assemblage du robot
- Test des capteurs
- Code
- Poster

Audrey :

- Assemblage du robot
- Cahier de charges
- Code
- Support présentation PowerPoint

Et au fil de chaque séance, nous avons travaillé ensemble à débugger le câblage et le code.