ProTIS 2021

Robot Piloté à distance

TAKTAK *Rami*
TOUJANI *Adem*
KHITOUS *Hamza*
AZAIZ *Malek*

Sommaire

I. Introduction :

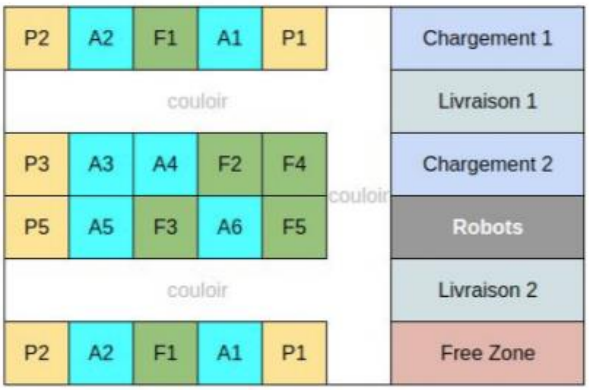
Problématique et description du projet :

Comment transmettre un ordre à un robot pour qu'il se déplace entre différentes régions d'une usine afin de transporter des pièces ?

Cette problématique se sépare lors en plusieurs parties :

- Interface Homme/machine : créer une interface afin d'envoyer les différentes commandes de déplacement
- Transmission : trouver une méthode de transfert de données d'utilisateur de l'ordinateur au moteur et avoir un retour (émission / réception)
- Pilotage des moteurs : utiliser les cartes NUCLEO afin de déplacer moteur en translation et rotation (pour tourner)

Le projet consiste à piloter le moteur à distance à l'aide d'un émetteur radiofréquence (sans fil). Les deux moteurs de chaque roue du robot posséderont un récepteur pour traduire l'information reçue de la part de l'ordinateur telles que le robot avance recule tourne à gauche ou bien à droite en utilisant le protocole de communication RS232.



Exemple de plan d'une usine de production SOLEC

Figure 0 : schémas simplifiant les zones de déplacement de moteur

Le schémas suivant (figure 1) représente le fonctionnement prévu du robot l'utilisateur renvoie des données bien précises pour commander le moteur. Une transmission sans fil qui assure la réception de ces données par le moteur et déplacer le moteur alimenté par une pile de 9V la boucle d'asservissement assure le bon fonctionnement et déplacement du moteur et une rétroaction qui repère la position du moteur est prévu afin d'assurer l'achèvement de la commande.

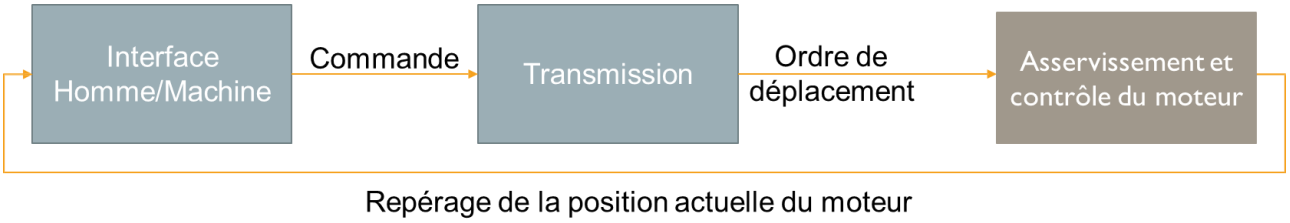


Figure 1 : schémas de fonctionnement de moteur

II. Mise en œuvre :

1. Répartition des tâches

Equipe	Code	Hardware	Gestion d'équipe	Communication
Rami	x	x		x
Malek	x		x	
Hamza	x	x		
Adem		x		x

Tableau 1: Répartition des taches

2. Pilotage des moteurs

Afin de piloter notre robot, nous avons utilisé deux moteurs CC (courant continu), le moteur se compose de : un encodeur, un corps (moteur + actionneur) et 6 pins. Le moteur qu'on a utilisé a un facteur de réduction qui vaut 19. Ce facteur de réduction est un coefficient qui se trouve entre le moteur et la partie qui va mettre en marche la roue, c'est le rapport entre le nombre de tour du moteur et le nombre de tour de l'arbre.

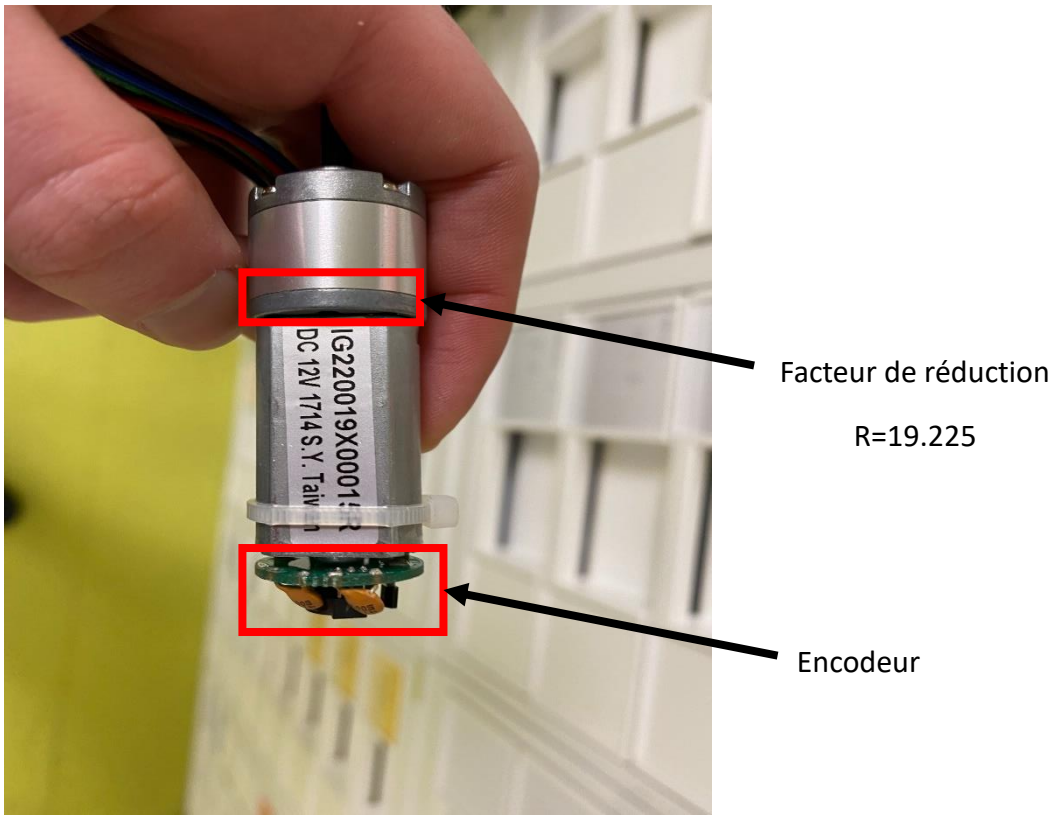


Figure 2 :Moteur utilisé pour piloter le véhicule

- **Fonctionnement des moteurs :**

On utilise le principe de la modulation de largeur d'impulsions c'est-à-dire le mouvement mécanique est directement associé à la valeur moyenne du signal envoyé et par la suite au rapport cyclique ce qui est la variable à changer afin de changer la vitesse du moteur.

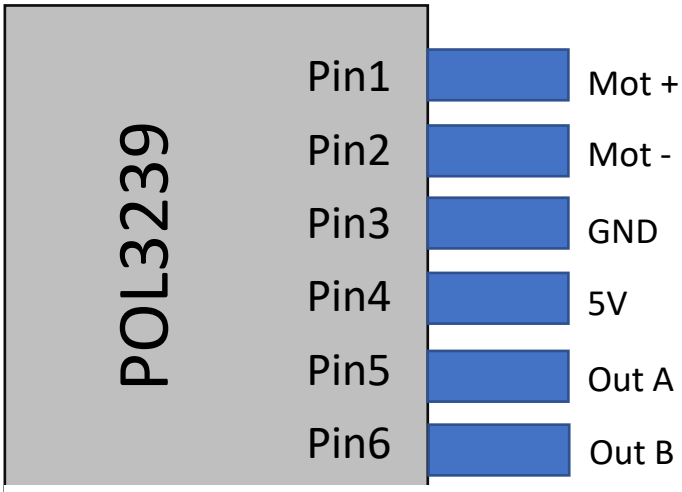


Figure 3 : Modélisation des pins du moteur

Le module POL3239 nous a servi comme pont entre la carte nucléo et le moteur CC.

Pour pouvoir piloter un moteur à courant continu dans les deux directions, il faut inverser le sens du courant (et la tension aux bornes du moteur) entre *Mot+* et *Mot-.*

Le rapport cyclique (ou RC) décrit la durée pendant laquelle le signal est à l'état haut (actif) en pourcentage de la durée d'un cycle complet. La fréquence détermine la vitesse à laquelle la MLI effectue un cycle (par exemple, 1000 Hz serait 1000 cycles par seconde) et par conséquent à quelle vitesse il passe de l'état haut à l'état bas et vice versa.

Le pin 4 (*5V*) sert à alimenter l'électronique du moteur (composants électroniques) mais pas le moteur lui-même. Pour cela, nous avons utilisé des batteries.

Dès la première séance nous avons réussi à faire marche les deux moteurs simultanément dans le même avec la même consigne d'entrée. Nous avons remarqué que le robot n'allait pas en une ligne droite lorsqu'on l'a mis au sol.

A partir de la deuxième séance, nous avons pu changer le châssis du véhicule afin de bien répartir le poids des composants équitablement sur la surface du robot. Cette étape est cruciale afin de faciliter la mise en œuvre de l'asservissement.

- **Fonctionnement des encodeurs :**

Nous utilisons un encodeur magnétique qui possède 12 ticks/tour c'est-à-dire à chaque tour l'encodeur va renvoyer 12 périodes d'un signal rectangulaire.

Un encodeur est un dispositif électromécanique qui génère un signal électrique en fonction de la position ou du déplacement de l'élément mesuré. Ce dispositif va servir principalement dans partie asservissement (mesure de l'erreur).

L'encodeur est en quadrature de phase (décalage de 90° entre les deux signaux). L'intérêt d'une telle forme de signal est de pouvoir connaitre le sens de rotation du moteur. En effet, en regardant quel signal est en avance par rapport à l'autre on peut en déduire si le moteur avance ou recule. Sur l'image ci-dessus, c'est la courbe jaune (Out A) qui est en avance sur la courbe bleue (Out B), le moteur fonctionne en marche avant.

Sur la figure à droite, on visualise deux signaux en quadrature de phase. L'intérêt d'une telle forme de signal est de pouvoir connaitre le sens de rotation du moteur. En effet, en regardant quel signal est en avance par rapport à l'autre on peut en déduire si le moteur avance ou recule.

Ici c'est la courbe jaune qui est en avance de phase par rapport à la courbe en bleu, le moteur fonctionne en marche avant.

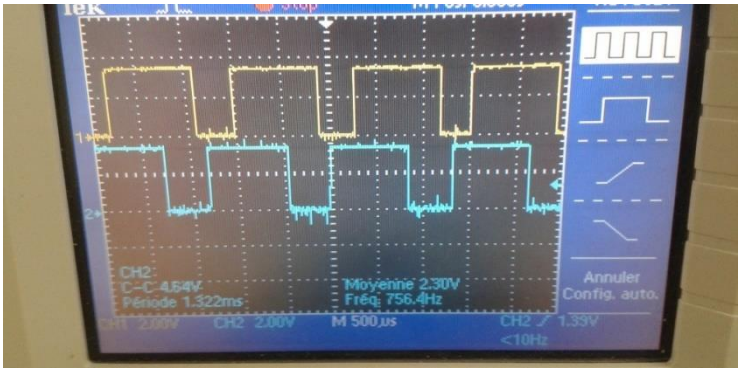


Figure 4 : Signaux des capteurs de l'encodeur

courbe jaune= voix A et la courbe bleue = voix B

3. Transmission

Dans cette partie, on s'intéresse à la manière de transmettre les ordres de commande du robot communiqués via l'interface Matlab sous la forme des données de la trajectoire (distance, vitesse) par l'intermédiaire d'un module RF (KAPPA-M868).

- **Schéma récapitulatif du circuit de Transmission :**

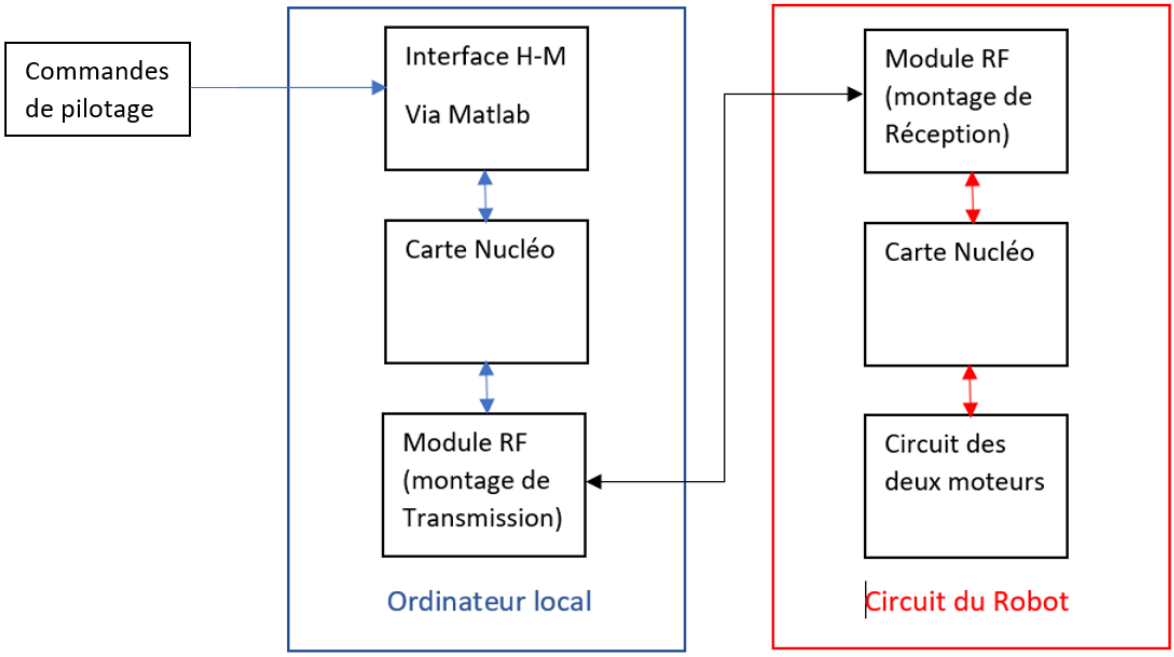


Figure 5 : Le circuit de transmission

On donne ici le détail du circuit de liaison entre la carte Nucléo et le module RF valide pour les deux configurations de la transmission ainsi que la réception :

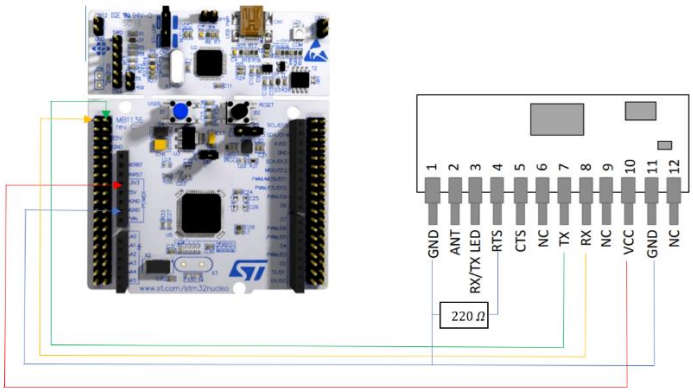


Figure 1: Câblage pour un circuit de Transmission/Réception

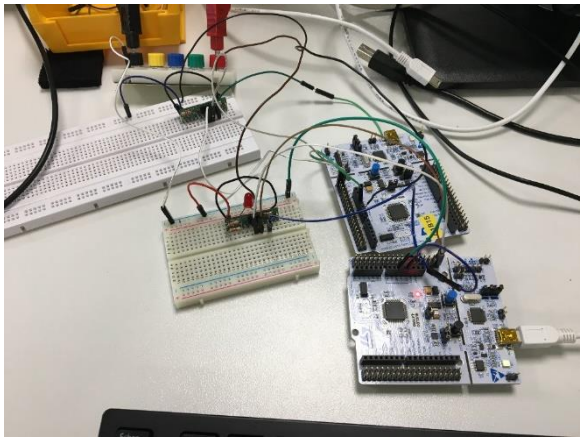


Figure 7 : Le circuit réel de la transmission

On teste notre circuit en envoyant une simple lettre 'A' (sans passer par l'interface Matlab) via un petit code de transmission sous MBED puis on la reçoit de la même façon via un petit code de réception sous MBED : ces deux codes sont à la base de toute communication possible développée ultérieurement entre la carte Nucléo et le module RF *(cf. fig8)*.

Ensuite cette lettre est traduite en une commande via un code de pilotage des deux moteurs sous MBED qui permet de faire avancer le robot suivant une ligne droite (les deux roues tournent à la même vitesse).

```
#include "mbed.h"
#include <string>

Serial carte2(PC_10, PC_11);
Serial pc(USBTX, USBRX);
int j;

char commande1[]="A\n"; // A désigne une commande pour avancer vers l'Avant

int main() {

    carte2.baud(19200);
    while(1) {
        for(j=0;j<sizeof(commande1);j++){
        carte2.putc(commande1[j]);
        pc.printf(commande1);
        wait(0.1);
        } } }
```

```
/// Reception :
#include "mbed.h"

Serial carte1(PC_10, PC_11);
Serial pc(USBTX, USBRX);

int main() {
    char data ;
    carte1.baud(19200);

    while(1) {
        if (carte1.readable())
        {
            data=carte1.getc();
            pc.putc(data);
            pc.prrintf(reception); // On affiche la commande reçue via une liaison TeraTERM
            }
        }
```

Figure 8: La transmission d'une commande pour avancer vers l'Avant

4. Asservissement des moteurs

Objectif : Dans cette partie on se propose d'asservir les deux moteurs **en vitesse** pour que le robot marche en ligne droite.

Cahier de charge : une erreur relative de 2 %

On modélise l'asservissement du moteur par la boucle suivante :

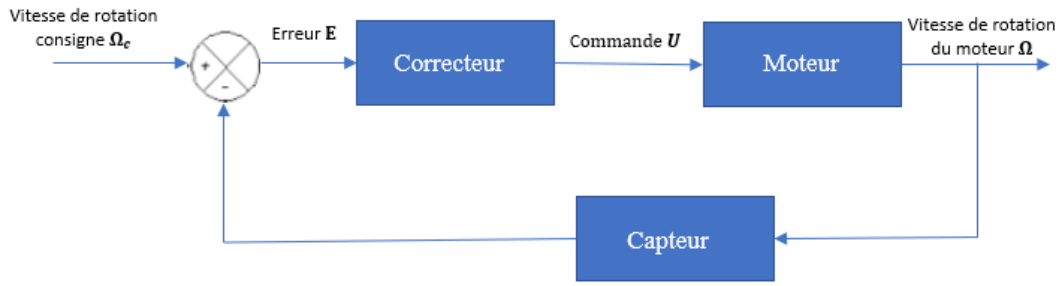


Figure 9: boucle d'asservissement du moteur

On choisit un correcteur PI (proportionnel intégral) qui annule l'erreur statique et rejette toute perturbation. La commande u à l'instant t est de la forme :

$$K_p\, e(t) + K_i\, \frac{1}{T_i} \int_0^t e(t)dt$$

L'action intégrale du correcteur est calculée en multipliant à chaque fois la somme de toutes les erreurs par $\frac{K_p}{T_i}$.

Mais pour calculer l'erreur, il faut comparer la valeur consigne à la vitesse réelle du moteur. Pour cela, on a créé la fonction $\boldsymbol{v\_ang}$:

> On détermine d'abord la résolution angulaire du moteur, c'est-à-dire la variation angulaire minimale entrainée par chaque incrémentation : A l'aide de la bibliothèque QEI et de la fonction $\boldsymbol{getPulses}$ on mesure **12 positions par tour complet**.
> Ensuite, la vitesse angulaire est déterminée en multipliant cette résolution par la variation du nombre de $Pulses$ pendant une courte période et en divisant par cette durée.

NB : Il faut tenir compte du facteur de réduction entrainé par l'encodeur attaché au moteur.

```
37 float resolution=2*3.1415/12 ;
38 QEI encoder0(D10, D11, NC, 48, QEI::X4_ENCODING);
39
40 float v_ang1(){                                   // fonction qui calcule la vitesse angulaire du premier moteur
41     int p0, p1 ;
42     float v ;
43     p0=encoder0.getPulses();                      // position de l'incrémant à t
44     wait_ms(10) ;
45     p1 = encoder0.getPulses();                    // position de l'incrémant à t+dt avec  dt = 10 ms
46     v=(p1-p0)*resolution/(0.01*19.225) ;          // calcul de la vitesse angulaire
47     return(v);
48     }
49
```

Figure 10: fonction qui calcule la vitesse angulaire avec D10 et D11 connectés directement aux voix A et B du moteur.

On utilise tera-term afin d'afficher les valeurs les plus significatives : la vitesse du moteur, l'erreur, la consigne, le rapport cycle ….

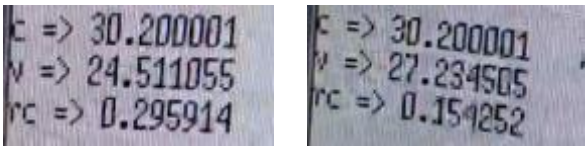


Figure 21 : Valeurs affichées par tera-term pour une vitesse angulaire consigne de 30 rad/s

L'asservissement est fonctionnel et l'erreur relative converge vers 0%.

Tera-term nous permet aussi de valider le bon fonctionnement de la fonction $\boldsymbol{v\_ang}$. Par exemple, lorsqu'on fait tourner le moteur à une vitesse constante, le freinage progressif de la roue à la main se manifeste par une diminution de la vitesse affichée par tera-term.

De plus, si on envoie la même commande au deuxième moteur, on constante que chaque moment du freinage du premier moteur est accompagné par une accélération du deuxième moteur : **l'asservissement est mis en place.**

5. *Interfaçage*

Pour que l'utilisateur puisse contrôle le robot à distance, on a essayé de créer une interface graphique sous *AppDesigner* : application de design d'interface sous.
Avec cette interface l'utilisateur pourra commander le robot en envoyant la valeur de la vitesse de contrôle et la distance à parcourir pour les déplacements linéaires : on a deux boutons pour orienter le robot dans la bonne direction et tourner comme ça il y a plus d'interaction entre l'utilisateur et le robot en cas d'obstacle ou un long trajet il suffit juste de connaitre les distances des couloirs et la largeur des rangées.

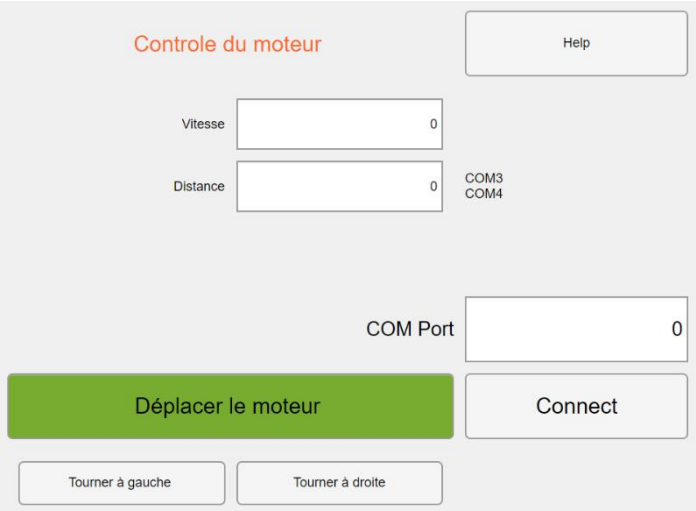


Figure 12 : interface homme machine

La valeur numérique de la vitesse et la vitesse sont ensuite envoyées en format chaine de caractère à la carte NUCLEO afin d'envoyer les informations nécessaire de déplacement au robot.

Faute de temps, on a trouvé le bon code qui nous permet d'envoyer les informations enregistrés par Matlab qui doivent êtres transférer à mbed pour être envoyer comme données d'entrée au moteur pour générer son déplacement.

III. Résultats et interprétations :

A la fin des 6 séances, notre équipe a réussi à concevoir un robot **automne** qui avance à la suite d'une commande envoyée sans fil. Nous avons pu faire **tourner et asservir les deux roues simultanément**. La partie interface est sur laquelle, on n'a pas eu beaucoup de temps afin de la travailler.

Pistes d'améliorations :

- Concevoir un châssis mieux adapté à notre application.
- Appliquer un asservissement en position sur le moteur.
- Ajouter des indications lumineuses (par exemple pour la marche arrière).
- Réussir à faire une étude énergétique sur la consommation des moteurs (comme les autres groupes on fait).
- Ajouter des détecteurs afin d'éviter d'obstacles.
- Améliorer l'algorithme afin d'optimiser le traitement des données lors de l'asservissement.
- Réussir à afficher la position et la vitesse en direct sur l'interface homme-machine.

IV. Retour d'expérience :

Nous avons trouvé cette expérience très riche en matière de nouvelles compétences, techniques de travail et outils du domaine de l'électronique et de la programmation.

Au début, nous avons eu quelques problèmes d'organisation et de gestion du temps ce qui a impacté la suite du projet puisqu'il y avait une cumulation de tâches qui n'étaient pas encore faites.

Grâce à ce projet, nous avons pu interagir avec les autres étudiants et les encadrants ce qui n'était pas évident puisque nous n'avons pas eu la chance d'avoir autant de cours en présentiel.

Ces échanges nous ont permis de mieux développer notre projet à travers les problèmes rencontrés et les solutions envisagées par les autres groupes, ce fut une expérience enrichissante sur le plan humain mais aussi sur le plan technique et académique.

Travailler sur un tel projet nous a aussi permis de mettre en évidence nos compétences en électronique, en programmation et aussi en travail équipe.

Dans l'idéal, nous aurions dû améliorer les performances de notre système mais faute temps nous étions dans l'incapacité de terminer le projet.

V. Annexes :

Initialisation :

```
/*
 * piloter deux moteurs à courant continu à distance
 * Auteurs : TAKTAK Rami, AZAIZ Malek, KHITOUS Hamza & TOUJANI Adem
 */

#include "mbed.h"
#include <stdint.h>

//moteur à gauche = ROUE 1
PwmOut moteur1(D8);                 // controle de la roue 1
DigitalOut moteur1_direction(D6);   // sens de rotation de la roue 1

//moteur à droite = ROUE 2
PwmOut moteur2(D9);                 // controle de la roue 2
DigitalOut moteur2_direction(D3);   // sens de rotation de la roue 2

Serial pc(USBTX, USBRX);            // affichage sur ordinateur
Serial carte1(PC_10, PC_11);        // affichage sur ordinateur
```

Pilotage des moteurs :

```
void marche_arriere(float rc){
    moteur1_direction = 0 ;
    moteur2_direction = 1 ;

    moteur1.write(rc);
    moteur2.write(rc);

    wait(1) ;

    moteur1.write(0);
    moteur2.write(0);

    }

void marche_avant(float rc){

        moteur1_direction = 1 ;
        moteur2_direction = 0 ;

        moteur1.write(rc);
        moteur2.write(rc);

        wait(1) ;

        moteur1.write(0);
        moteur2.write(0);

    }
```

```
void turn_R(float rc) //( tourner de 90° à droite pour une consigne de 30 rad/s)
{       moteur1_direction = 1;
        moteur1.write(rc);
        moteur2.write(0);

        moteur2_direction = 1;

        wait(0.14);  // disance de l'arc  = pi/2 * R(=25cm , rayon de la roue)
        moteur1.write(0);
        }

void turn_L(float rc) //( tourner de 90° à gauche pour une consigne de 30 rad/s)
{       moteur1_direction = 1;
        moteur2_direction = 0;

        moteur1.write(0);
        moteur2.write(rc);

        wait(0.14);  // disance de l'arc  = pi/2 *R(=25cm , rayon de la roue)
        moteur2.write(0);
        }
```

Asservissement des moteurs :

```
float resolution=2*3.1415/12 ;
QEI encoder1(D10, D11, NC, 48, QEI::X4_ENCODING);
QEI encoder2(D0 , D1 , NC, 48, QEI::X4_ENCODING);

float v_ang1(){                                     // fonction qui calcule la vitesse angulaire du premier moteur
    int p0, p1 ;
    float v ;
    p0=encoder1.getPulses();                        // position de l'incrémant à t
    wait_ms(10) ;
    p1 = encoder1.getPulses();                      // position de l'incrémant à t+dt avec  dt = 10 ms
    v=(p1-p0)*resolution/(0.01*19.225) ;            // calcul de la vitesse angulaire
    return(v);
    }

float v_ang2(){                                    // fonction qui calcule la vitesse angulaire du deuxieme moteur
    int p0, p1 ;
    float v ;
    p0=encoder2.getPulses();                       // position de l'incrémant à t
    wait_ms(10) ;
    p1 = encoder2.getPulses();                     // position de l'incrémant à t+dt avec  dt = 10 ms
    v=(p1-p0)*resolution/(0.01*19.225) ;           // calcul de la vitesse angulaire
    return(v);
    }
```

```
// La boucle d'asservissement en vitesse

void boucle_regul(void){
    float v1;
    float v2;
    float erreur=0.0;                                   // erreur proportionnelle du moteur 1 initiale
    float Ierreur=0.0;                                  // erreur intégrale du moteur 1 initiale  ( erreur accumulée)
    float erreur2=0.0;                                  // erreur proportionnelle du moteur 1 initiale
    float Ierreur2=0.0;                                 // erreur intégrale du moteur 2 initiale  ( erreur accumulée)

    float Kp = 1.0/19.225;
    float KiTe = 0.0001;

    Ierreur = Ierreur + erreur;                         // calcul de l'erreur accumulée du moteur 1
    Ierreur2= Ierreur2+ erreur2;                        // calcul de l'erreur accumulée du moteur 2

    v1=v_ang1();                                        // vitesse angulaire réelle du moteur 1
    v2=v_ang2();                                        // vitesse angulaire réelle du moteur 2

    erreur = consigne - v1;                             // erreur du moteur 1
    erreur2= consigne - v2;                             // erreur du moteur 2

    moteur_1(Kp* erreur + KiTe *Ierreur );              // la nouvelle commande u(t) appliquée au moteur 1
    moteur_2(Kp* erreur2 + KiTe *Ierreur2 );            // la nouvelle commande u(t) appliquée au moteur 2

    //pc.printf("c1 => %lf \r\n",consigne);            // affichage de la consigne
    //pc.printf("v1 => %lf \r\n",v1);                  // affichage de vitesse angulaire actuelle du moteur 1
    //pc.printf("rc1 => %lf \r\n\n",Kp * erreur + KiTe *Ierreur);  // affichage de la commande appliquée au moteur 1

    }
```

Transmission sans fils :

```
// fonction qui traduit la lettre reçuie en une commande à applioquer au robot
void marche( char direction, float rc ){
    if (direction=='F'){                            // F = forward
        marche_avant(rc);}

    if (direction=='B'){                            // B = backward
        marche_avant(rc);}

    if (direction=='R'){                            // R = right
        turn_R(rc);}

    if (direction=='L'){                            // L = left
        turn_L(rc);}
    }
```

```
29
30 int main() {
31     double rc;
32     char data ;
33
34     rc =0.35;
35     carte1.baud(19200);
36
37     moteur1.period_ms(10);
38     moteur2.period_ms(10);
39
40
41     while(1) {
42         if (carte1.readable())
43         {
44             data=carte1.getc();     // récupération de la lettre de commande
45             marche(data, rc ) ;     // commande associée à la lettre reçue
46             pc.putc(data);          // affiche sur tera term la lettre reçue
47
48             }
49         }
50
51
52     }
53
```