GEII
DUT Génie Électrique
& Informatique Industrielle
IUT Belfort-Montbéliard

Motorisation et pilotage d'un fauteuil pour personne handicapée

RAPPORT DE PROJET TUTORÉ

2016/2017

Carlos Eduardo JUÁREZ CASTILLO
&
Erick VILLAGRANA RAMOS

Institut Universitaire de Technologie de Belfort-Montbéliard

Licence Pro VEGA

Patrick HIEBEL– Professeur référent

REMERCIEMENTS

Le projet présenté dans ce rapport, c'était développé dans le cadre de la Licence Pro VEGA, au sein de l'IUT de Belfort Montbéliard.

Tout d'abord nous devons remercier à Mr Patrick HIEBEL, tuteur de notre projet, qui a nous aidé tout au long de notre projet.

Nous tenons, également, remercier à tout l'équipe pédagogique pour l'aide quelle nous a apportée tous le long de l'année, à travers les cours et les travaux pratiques.

SOMMAIRE

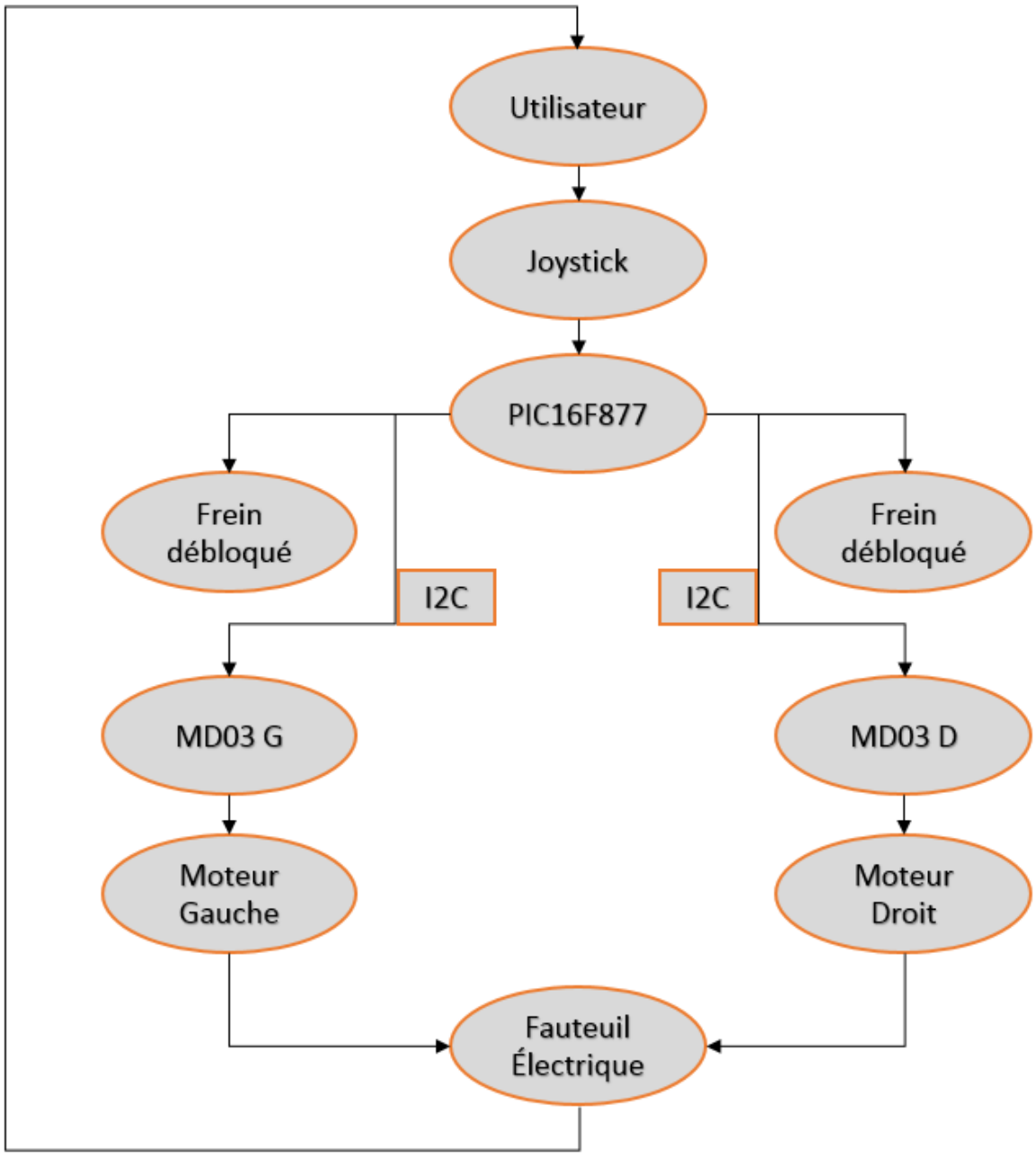
INTRODUCTION

Le projet tutoré a pour but de nous rendre autonomes et de mettre en marche les connaissances appris tout au long de la Licence Pro.

Le but de ce projet est de remettre le fauteuil électrique en fonctionnement.

Le fauteuil électrique permet aux personnes handicapées se déplacer plus facilement. Ce fauteuil doit respecter les normes en vigueurs.

SCHEMA DU FONCTIONNEMENT



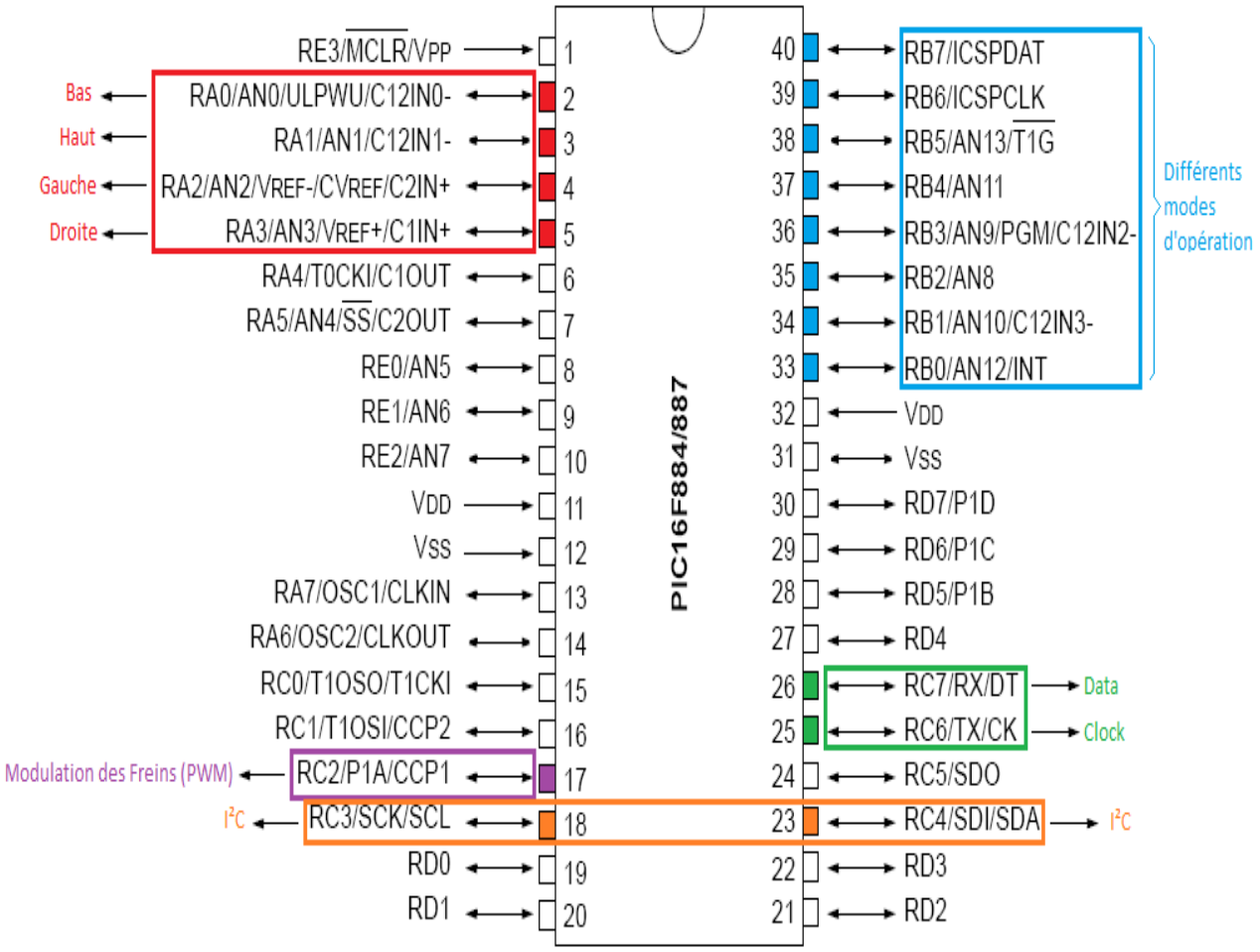
Tout d'abord, l'utilisateur agit le joystick et le PIC 16F877 reçoit les donnes. Grâce à eux le microcontrôleur détecte la position vers où va se déplacer le fauteuil. Le PIC ordonne aux modules se mettre en fonctionnement. Les modules font fonctionner les moteurs (gauche et droit) en dépendant de l'information que le microcontrôleur reçoit.

UFC UNIVERSITÉ DE FRANCHE-COMTÉ
IUT Belfort-Montbéliard
INSTITUT UNIVERSITAIRE DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD
Tél. +33 (0)3 84 58 77 00 • iut-bm.univ-fcomte.fr

MATERIEL ET COMPOSANTS

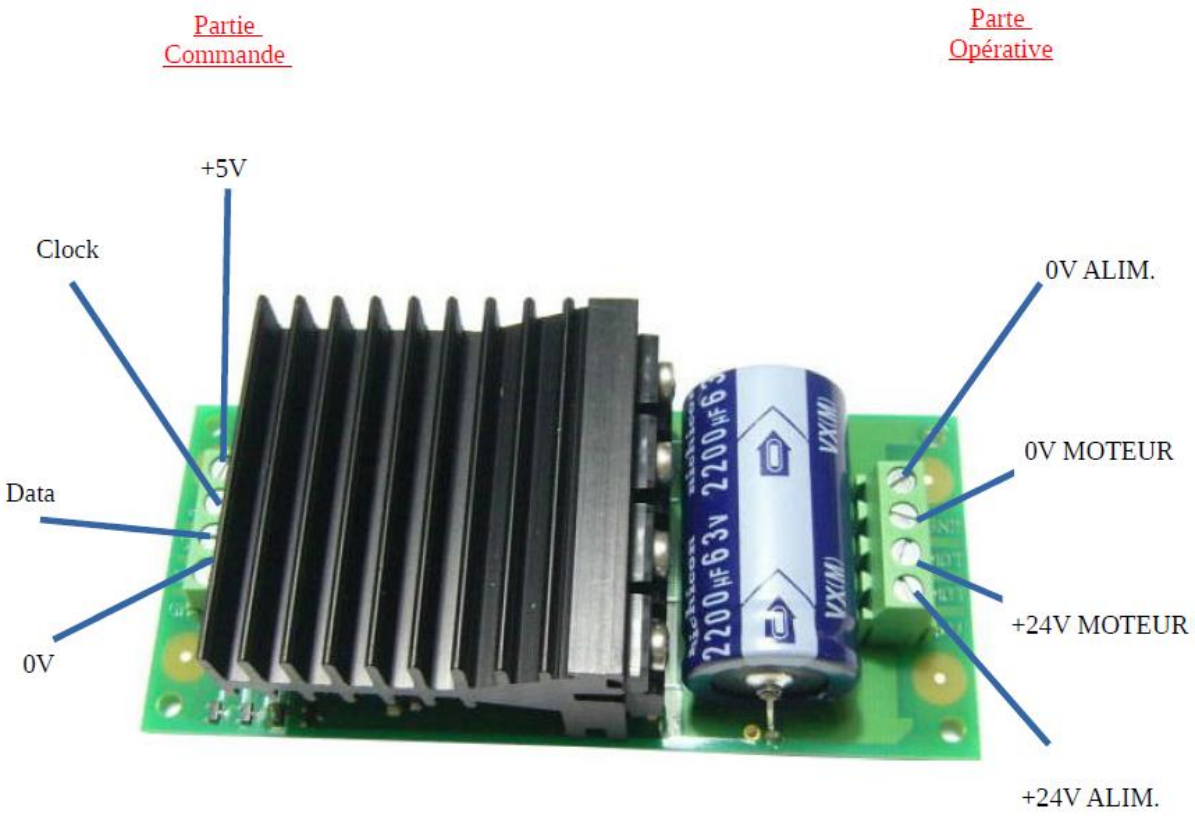
PIC 16F877

Nous avons disposé d'un microcontrôleur PIC 16F877, ci-dessous se trouve le schéma et l'utilisation que nous avons fait des entrées et des sorties.



Carte de commande pour les moteurs (MD03)

Le module MD03 est une commande de puissance pour moteurs à courant continu. La puissance du moteur est contrôlée par une régulation PWM du pont en H à une fréquence de 15 kHz.



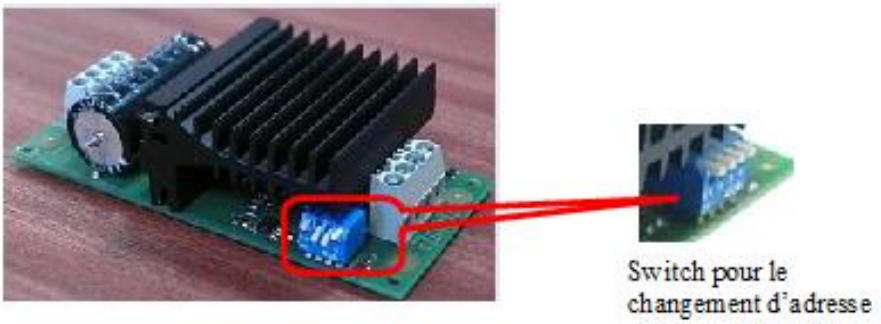
Pour commander les deux modules MD03 nous avons utilisé le protocole I2C.

Ce protocole de communication nous permet d'utiliser jusqu'à 8 modules avec seulement 4 fils (Data, clock, alimentation et masse). À chaque module nous devons définir son adresse. Dans notre cas nous utilisons deux modules de ces types, alors nous avons besoin de deux adresses différentes.

Dans ce cas nous avons utilisé les deux premières adresses (0xB0 et 0xB2).

Mode	Switch 1	Switch 2	Switch 3	Switch 4
I2C Bus - address 0xB0	On	On	On	On
I2C Bus - address 0xB2	Off	On	On	On
I2C Bus - address 0xB4	On	Off	On	On
I2C Bus - address 0xB6	Off	Off	On	On
I2C Bus - address 0xB8	On	On	Off	On
I2C Bus - address 0xBA	Off	On	Off	On
I2C Bus - address 0xBC	On	Off	Off	On
I2C Bus - address 0xBE	Off	Off	Off	On
0v - 2.5v - 5v Analog	On	On	On	Off
0v - 5v Analog + Direction	Off	On	On	Off
Radio Control	On	Off	On	Off

UFC UNIVERSITÉ DE FRANCHE-COMTÉ
IUT Belfort-Montbéliard
INSTITUT UNIVERSITAIRE DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD
Tél. +33 (0)3 84 58 77 00 • iut-bm.univ-fcomte.fr



Pour l'utilisation du protocole I2C sur les modules MD03, nous avons eu recours aux registres 0, 2 et 3.

Chaque registre correspond à une variable qui commande un paramètre.

Le registre 0 correspond au sens de rotation :

- 00 correspond à l'arrêt.
- 01 correspond à la marche avant.
- 02 correspond à la marche arrière.

Le registre 2 correspond à la vitesse (varie entre 0-255).

Le registre 3 correspond à l'accélération du moteur (varie entre 0-255).

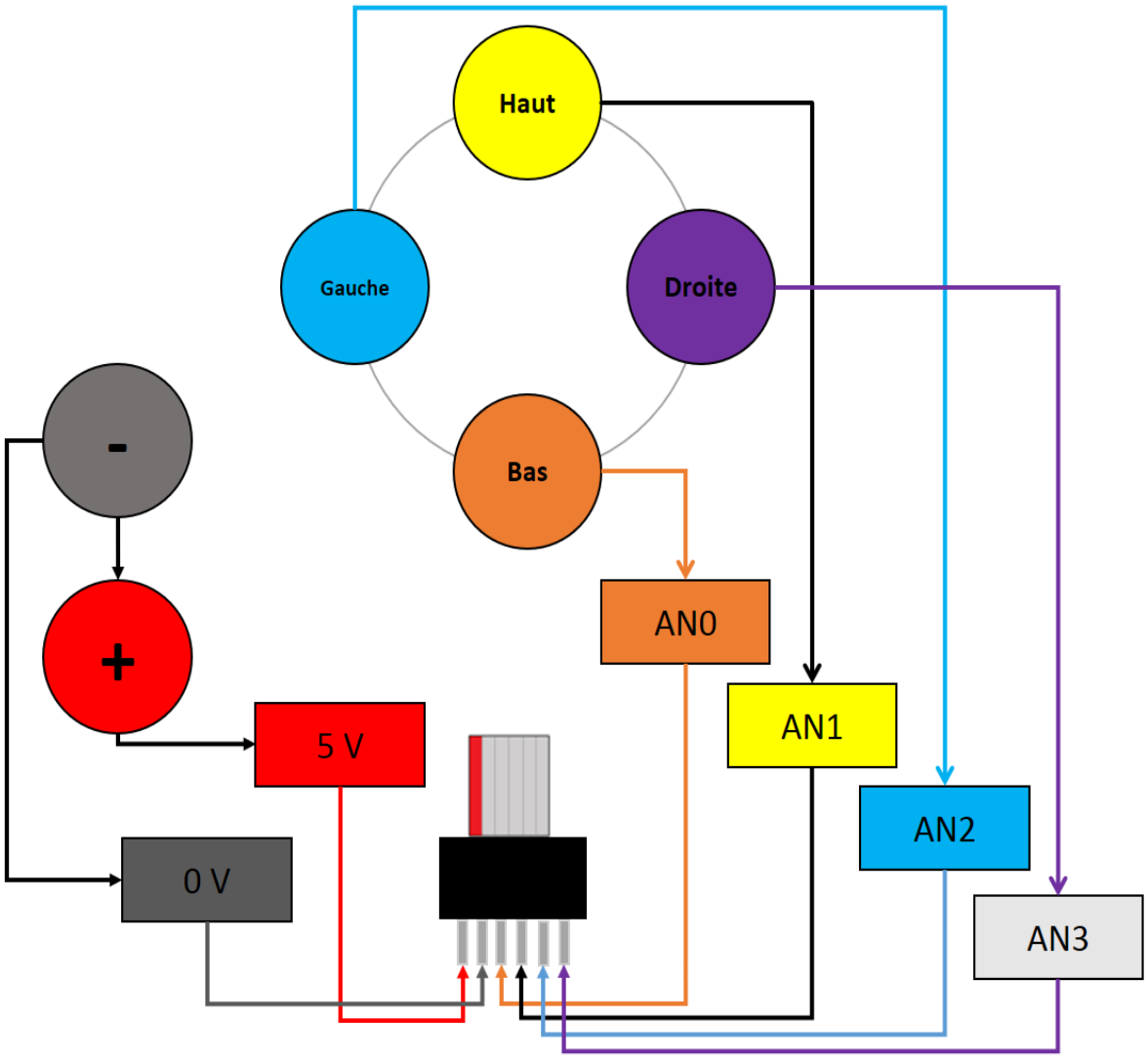
Register Address	Name	Read/Write	Description
0	Command	R/W	Write 01 Forwards - 02 Reverse (00 for instant stop - Rev9 firmware only)
1	Status	Read only	Acceleration, Temperature and Current Status
2	Speed	R/W	Motor Speed 0-255 (0x00 - 0xFF)
3	Acceleration	R/W	Motor Acceleration 0-255 (0x00 - 0xFF)
4	Temperature	Read only	Module temperature
5	Motor Current	Read only	Motor Current
6	Unused	Read only	Read as zero
7	Software Revision	Read only	Software Revision Number- Currently 9

Joystick

Le joystick transforme une position mécanique en un signal électrique. Avec l'acquisition d'un signal analogique, le programme fait une conversion d'un signal analogique vers un signal numérique.

Lorsque l'on incline le joystick la tension change en fonction de l'inclinaison. Variation de la tension : 0,4V à 4,7V.

Une fois que nous avons fait la conversion analogique-numérique nous avons obtenu une valeur entre 22 et 239.



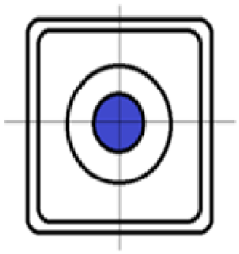
Lois de commande entre le joystick et les moteurs

Pour effectuer une rotation avec le fauteuil vers la droite ou vers la gauche nous avons mis une loi de commande.

Le joystick permet de diriger le fauteuil dans le sens où l'utilisateur l'incline. Lorsque on incline le joystick on reçoit la tension en fonction de l'inclination.

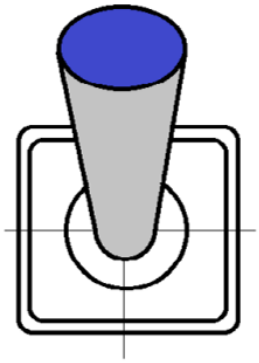
MG = Moteur Gauche.

MD = Moteur Droite.



MG MD

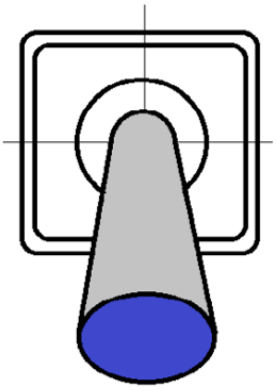
Point neutre



MG = MD

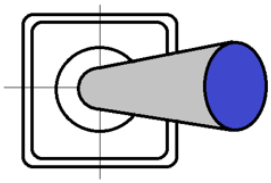
Les deux vitesses

Marche avant



MG = MD

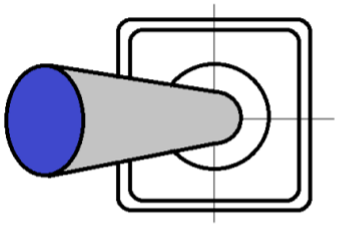
Marche Arrière



Rotation Droite

MG	MD	
↑	↓	Bas Vitesse
↑	●	Haut Vitesse

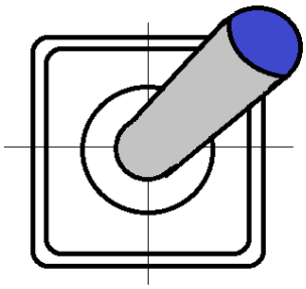
Quand la vitesse du MG soit haut diminuer la Vitesse du MD



Rotation Gauche

MG	MD	
↓	↑	Bas Vitesse
●	↑	Haut Vitesse

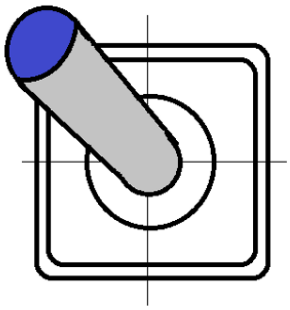
Quand la vitesse du MD soit haut diminuer la Vitesse du MG



Marche Avant-Droite

MG > MD

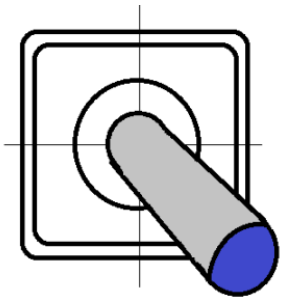
Soit plus grande la vitesse du Moteur Gauche



Marche Avant-Gauche

MG < MD

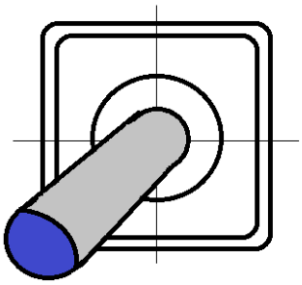
Soit plus grande la vitesse du Moteur Droite



MG > MD

Soit plus grande la vitesse du Moteur Gauche

Marche Arrière-Droite



MG < MD

Soit plus grande la vitesse du Moteur Droite

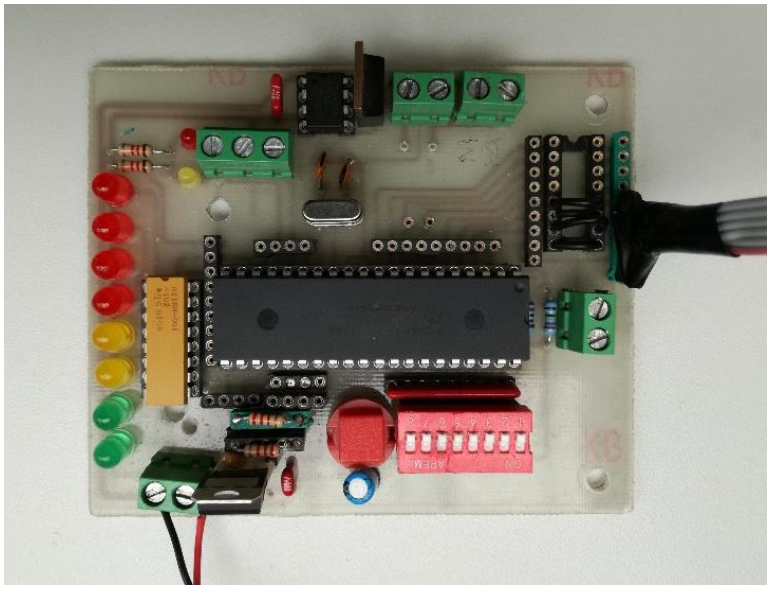
Marche Arrière-Gauche

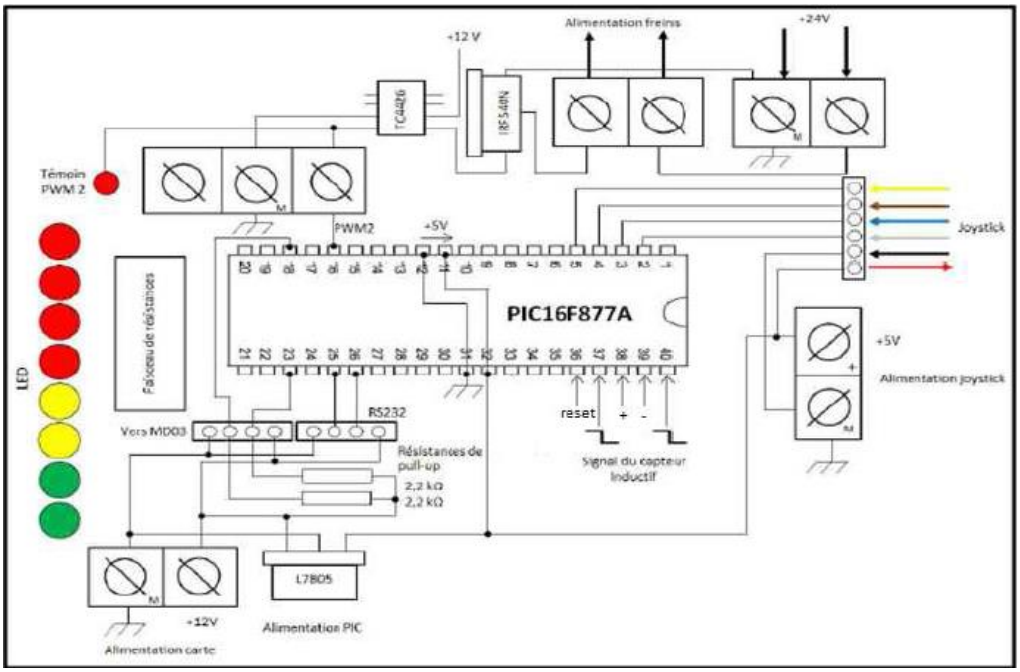
UFC UNIVERSITÉ DE FRANCHE-COMTÉ
IUT Belfort-Montbéliard
INSTITUT UNIVERSITAIRE DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD
Tél. +33 (0)3 84 58 77 00 • iut-bm.univ-fcomte.fr

La carte de commande

Celle-ci est alimentée en 12 V depuis les batteries.

- L7805 : C'est le régulateur de tension qui réduit la tension de 12 v à 5 v.
- PIC 16F877 : C'est le contrôleur du système qui gère toutes les communications et toutes les commandes des entrées et sorties entre les différents organes (joystick, freins, etc.).





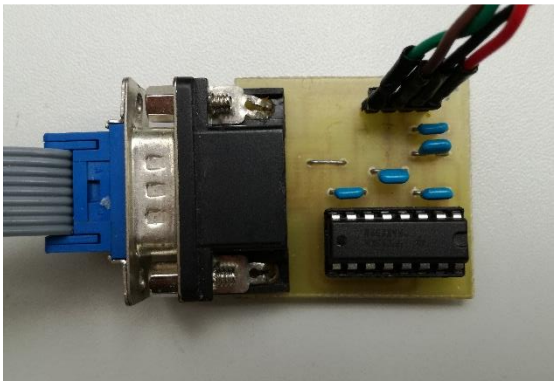
Liaison RS232

C'est une liaison qui permet de programmer le microcontrôleur.

La liaison se fait grâce à l'aide du MAX232.

RX : Pour la réception de données.

TX : Pour la transmission de données.



Les freins à perte de tension

Les freins sont intégrés au motoréducteur, lorsqu'une tension est appliquée au frein, un champ électromagnétique est créé, ce qui entraîne l'ouverture du frein, dans le cas contraire s'il n'y a pas d'alimentation électrique nous aurions un blocage des freins.

La tension qui alimente les freins est de entre 0 et 24V, pour obtenir cette tension nous avons utilisé un transistor par l'aide d'un signal PWM.

Après d'avoir fait des essais nous avons déterminé que la tension nécessaire au déblocage des freins se situe aux environs de 17V. La valeur minimum avec laquelle les freins peuvent être débloqués se situe aux environs de 3,3V.

Les motoréducteurs

Le fauteuil roulant électrique a deux moteurs à courant continu à excitations en série avec un motoréducteur.

Nous pouvons moduler la tension d'alimentation des freins par commande PWM et faire de l'économie d'énergie. Le constructeur préconise une tension, d'alimentation entre 0 et 24 V.

- Freins serrés : 0V (Pas d'alimentation)
- Freins desserrés : 24V

Par la suite nous avons effectué quelques tests et nous avons constaté que les tensions des freins sont les suivants:

- Freins serrés : 5V
- Freins desserrés : 17V

Le protocole I²C

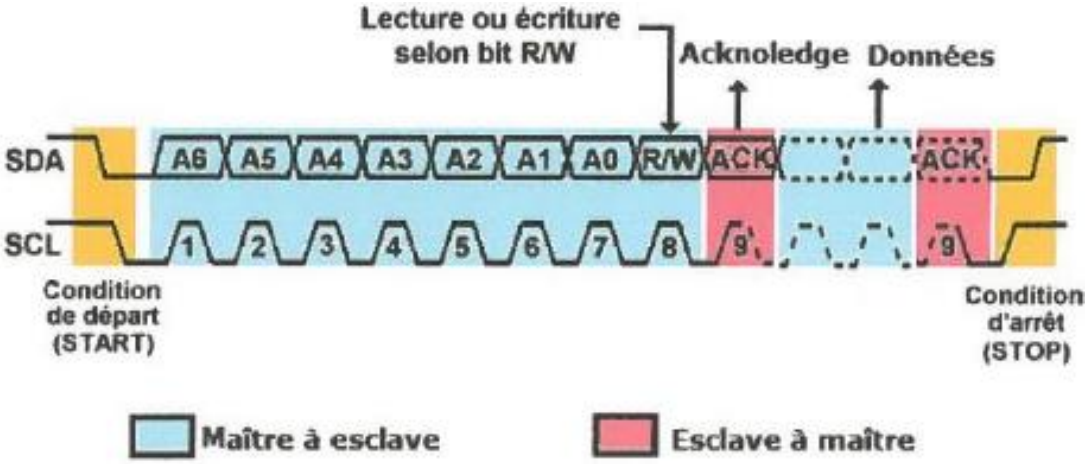
Pour effectuer le mode de communication du projet nous avons utilisé I²C.

Le but du protocole I²C est de communiquer avec plusieurs modules MD03 avec seulement 4 fils. Ces fils sont caractérisés des façons suivantes :

- Le signal de données « DATA »
- Le signal horloge « CLOCK »
- L'alimentation
- Masse

Le premier fil (DATA) est utilisé pour transmettre les données. Le deuxième fil (CLOCK) est utilisé pour transmettre un signal d'horloge synchrone (signal qui indique le rythme d'évolution de la ligne SDA).

Fonctionnement du bus I²c



ANNEXES

```
//#include <18F4580.h>
#include <16F877A.h>                          //On utilise le PIC16F877A
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)                    //Frequence du microcontroleur à 20Mhz.
#use rs232(baud=9600, xmit=PIN_C6,rcv=PIN_C7) //Liaison RS232 pour faire la programmation dés l'ordinateur
#use i2c(MASTER,sda=PIN_C4,scl=PIN_C3)        //Utilisation bus I2C pour les modules MD03
#include <math.h>                             //Librarie pour utiliser des fonctions mathematiques

#include <MD03_2014.c>                        //Librarie des modules MD03
//Ecriture_speed_MD03(int number, sens, speed)
//number 0 ou 2 % 0xB0 (0xB0 ou 0xB2) seleccion de la salida a MD03
//sens 0: Stop    -- arrete
//sens 1: Forward -- marche avant
//sens 2: Reverse -- marche arrière
//spped: 0..255   -- vitesse pour les rous


int bas, haut, gauche, droite,mode;           //Variables des potentiometres de 0 a 254.
int bas1, haut1, gauche1, droite1;            //Variables des potentiometres de 0 a 254.

int ordreD,ordreG,mode_max,mode_min;          //Variables pour determiner la sortie de vitesse aux moteurs
int sensMG,sensMD,tolerance;
int pwmg,pwmd;                                //Variables pour PWM
float x,a,coef_vit_h;                         //Variables pour determiner le sens des moteurs

void actualiza_joystick()
{
   set_adc_channel(0);          //Choix "voie bas Axe Y"
   delay_us(10);                //Delay stabilisation avant acquisition
   bas=read_adc();              //valeur de 128 a 216
   set_adc_channel(1);          //Choix "voie haut Axe Y"
   delay_us(10);                //Delaystabilisation avant acquisition
   haut=read_adc();             //valeur de 128 a 216
   set_adc_channel(2);          //Choix "voie gauche Axe X"
   delay_us(10);                //Delaystabilisation avant acquisition
   gauche=read_adc();           //Valeur de 128 a 216
   set_adc_channel(3);          //Choix "voie droite Axe X"
   delay_us(10);                //Delay stabilisation avant acquisition
   droite=read_adc();           //Valeur de 128 a 216

}

void neutro()
{
    output_d(0xFF);
    ordreD=0;               //Vitesse droit=0
    ordreG=0;               //Vitesse gauche=0
    sensMG=0;               //Sens gauche stop
```

```
    sensMD=0;                //Sens droit stop
    output_low(PIN_C1);      //Freins bloqués.
    output_low(PIN_C2);      //Freins bloqués.
    pwmg=0;
    pwmd=0;
}

void joystick_back()
{
   ordreG= 50;
   ordreD= 50;
   sensMG= 2;               //Sens de la roue gauche avant
   sensMD= 2;               //Sens de la roue droite avant
}

void joystick_front()
{
   bas1=(bas-128)*coef_vit_h+mode_min;       //Bas1 entre une vitesse max choisie et une vitesse min
   droite1=(droite-128)*coef_vit_h+mode_min;
   gauche1=(gauche-128)*coef_vit_h+mode_min;
   haut1=(haut-128)*coef_vit_h+mode_min;

   if(droite>128+tolerance)                      //Avant et gauche
   {
      output_d(0x02);
      x=sqrt(pow(droite,2)+pow(bas,2));
      ordreD=(x-181)*a;
      ordreG=bas1;
      sensMG=1;
      sensMD=1;
   }

   else                                          //avant et droite
   {
   if(droite<=128-tolerance)
      {
         output_d(0x80);
         x=sqrt(pow(gauche,2)+pow(bas,2));
         ordreG=(x-181)*a;
         ordreD=bas1;
         sensMG=1;                        //sens gauche avant
         sensMD=1;                        //sens droit avant
      }

      else                                //Marche avant
      {
         delay_ms(500);
         x=sqrt(pow(bas,2)+pow(droite,2));
```

```
            ordreG= bas1;
            ordreD= bas1;
            sensMG= 1;                          //Sens de la roue gauche avant
            sensMD= 1;                          //Sens de la roue droit avant
        }
    }
}

void joystick_front0()
{
    bas1=(bas-128)*coef_vit_h;
    droite1=(droite-128)*coef_vit_h;
    gauche1=(gauche-128)*coef_vit_h;
    haut1=(haut-128)*coef_vit_h;

    if(droite>=128+tolerance)               //Avant et gauche
    {
        output_d(0x02);
        x=sqrt(pow(droite,2)+pow(bas,2));
        ordreD=(x-181)*a;
        ordreG=ordreD;
        sensMG=2;
        sensMD=1;
    }

    else                                    //Avant et droite
    {
        if(droite<128-tolerance)
        {
            output_d(0x80);
            x=sqrt(pow(gauche,2)+pow(bas,2));
            ordreG=(x-181)*a;
            ordreD=ordreG;
            sensMG=1;                       //Sens gauche avant
            sensMD=2;                       //Sens droit avant
        }

        else                                //Marche avant
        {
            x=sqrt(pow(bas,2)+pow(droite,2));
            ordreG= bas1;
            ordreD= bas1;
            sensMG= 1;                      //Sens de la roue gauche avant
            sensMD= 1;                      //Sens de la roue droit avant
        }
    }
}
```

```
144
145 void initialisation()
146
147 {
148    set_adc_channel(ALL_ANALOG);           //Selection d'entrees analogiques
149    setup_adc(ADC_CLOCK_INTERNAL);
150    tolerance=25;                          //Marge d'erreur joystick
151    setup_ccp1(CCP_PWM);                   //Inizialitation PWM1
152    setup_ccp2(CCP_PWM);                   //Inizialitation PWM2
153    setup_timer_2(T2_DIV_BY_4,128,1);
154    output_high(pin_C1);                   //Blocage freins si C1=1
155    output_high(pin_C2);                   //Blocage freins si C2=1
156 }
157
158 void mode_switch()
159 {
160    mode=input_b();
161    switch (mode)                      //Choix de mode de fonctionement et de la vitesse
162          {
163          case 0:     //tourner en toupie
164          if(bas>=127)
165          {
166             joystick_front0();
167          }
168          output_d(0x00);
169          break;
170
171          case 1:   // vitesse tres basse
172          mode_max=100;
173          mode_min=0;
174          a=mode_max/(255-181);
175          coef_vit_h=(mode_max-mode_min)/(216-128);
176
177          if(bas>=127)
178          {
179             joystick_front();
180          }
181          break;
182
183          case 2: //Vitesse moyenne
184          mode_max=200;
185          mode_min=150;
186          a=mode_max/(255-181);
187          coef_vit_h=(mode_max-mode_min)/(216-128);
188
```

```
189         if(bas>=127)
190         {
191            joystick_front();
192         }
193         break;
194
195         case 3: //Vitesse elevée
196         mode_max=255;
197         mode_min=225;
198         a=mode_max/(255-181);
199         coef_vit_h=(mode_max-mode_min)/(216-128);
200
201         if(bas>=127)
202         {
203            joystick_front();
204         }
205         break;
206
207         case 4: //Marche arriere
208         if (bas<127)
209         {
210            joystick_back();
211         }
212         break;
213         }
214 }
215
216 void main()
217 {
218    initialisation();
219
220    while(TRUE)
221    {
222       actualiza_joystick();
223       mode=input_b();
224
225       if ((((droite>128-tolerance)&&(droite<128+tolerance))&&(bas>128-tolerance))&&(bas<128+tolerance))  //Point mort
226       {
227          neutro();
228       }
229
230       else
231       {
232          output_high(PIN_C1);              //Freins debloqués
233          output_high(pin_C2);              //Blocage freins si C2=1
234
235          set_pwm1_duty(pwmd);
236          set_pwm2_duty(pwmg);
```

```
236             set_pwm2_duty(pwmg);
237
238             mode_switch();
239         }
240
241         Ecriture_speed_MD03(0,sensMG,ordreG);  //roue droite (module, sens, vitesse)i2c
242         Ecriture_speed_MD03(2,sensMD,ordreD);  //roue gauche (module, sens, vitesse)i2c
243         //printf("Y= %2u\r",bas);              //Pour la reception de données de l'axe Y
244         //printf("X= %2u\r",gauche);           //Pour la reception de données de l'axe X
245     }
246 }
```

Conclusion

Grâce à la réalisation du projet tutoré nous avons pu mettre en pratique nos connaissances théoriques et pratiques acquises durant la formation dans la Licence Pro VEGA.

Nous avons eu quelques difficultés au niveau de la programmation et avec le fonctionnement du joystick.

Enfin, le projet nous a permis de devenir autonomes. Cette expérience pour nous va nous aider beaucoup au futur, tant personale que professionnellement.

UFC UNIVERSITÉ DE FRANCHE-COMTÉ
IUT Belfort-Montbéliard
INSTITUT UNIVERSITAIRE DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD
Tél. +33 (0)3 84 58 77 00 • iut-bm.univ-fcomte.fr