FAUTEUIL ROULANT ELECTRIQUE

REMERCIEMENTS

Le projet présenté dans ce compte rendu, c'est déroulé dans le cadre d'une licence pro, au sein de l'IUT de Belfort Montbéliard.

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

Nous tenons également à remercier 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

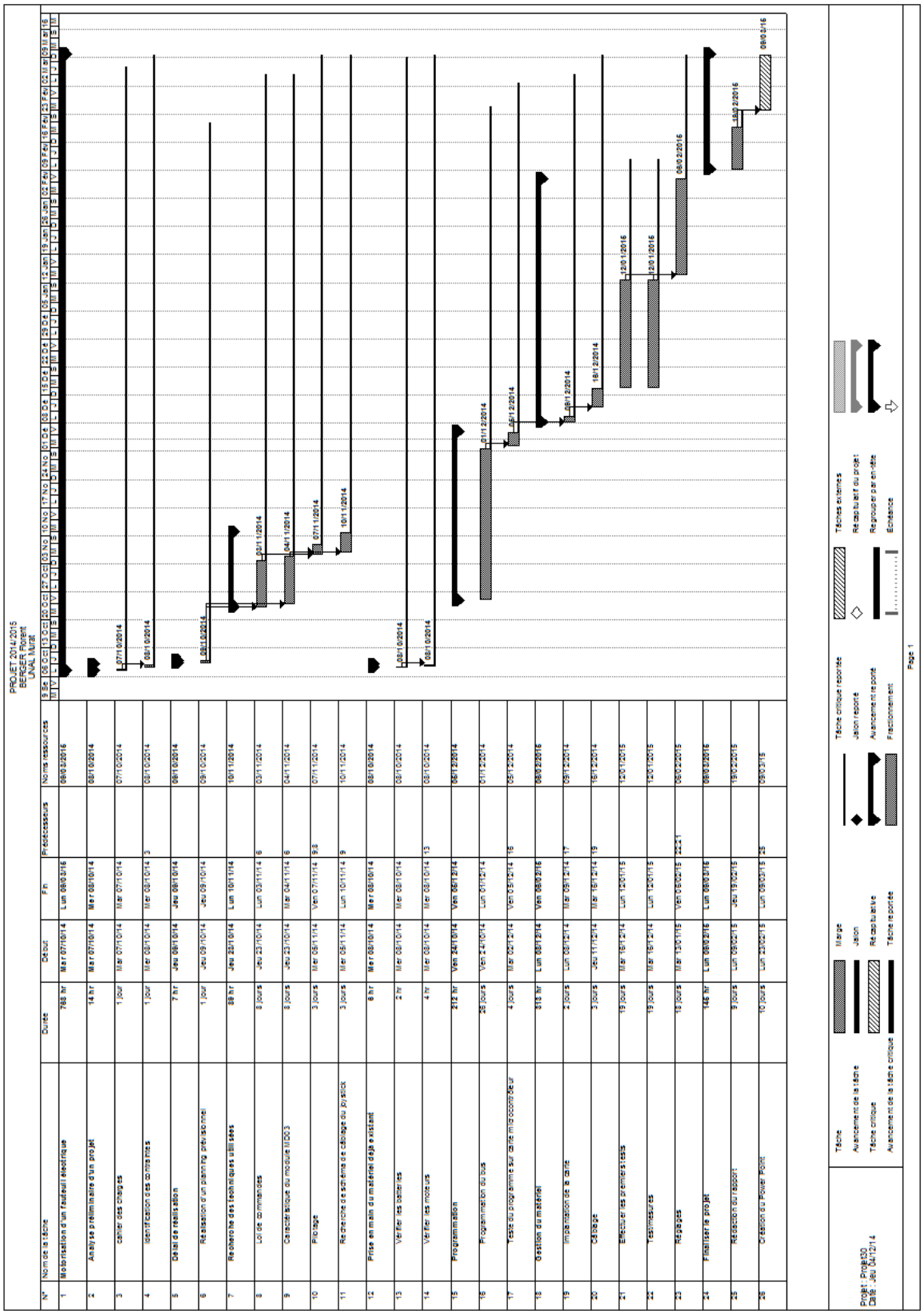
Le projet tutoré a pour but de nous rendre autonomes et de mettre en application les connaissances acquises tout au long de cette année scolaire.

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

Le fauteuil à assistance électrique permet aux personnes à mobilité très réduite de se déplacer plus facilement. Ce fauteuil doit respecter les normes en vigueurs.

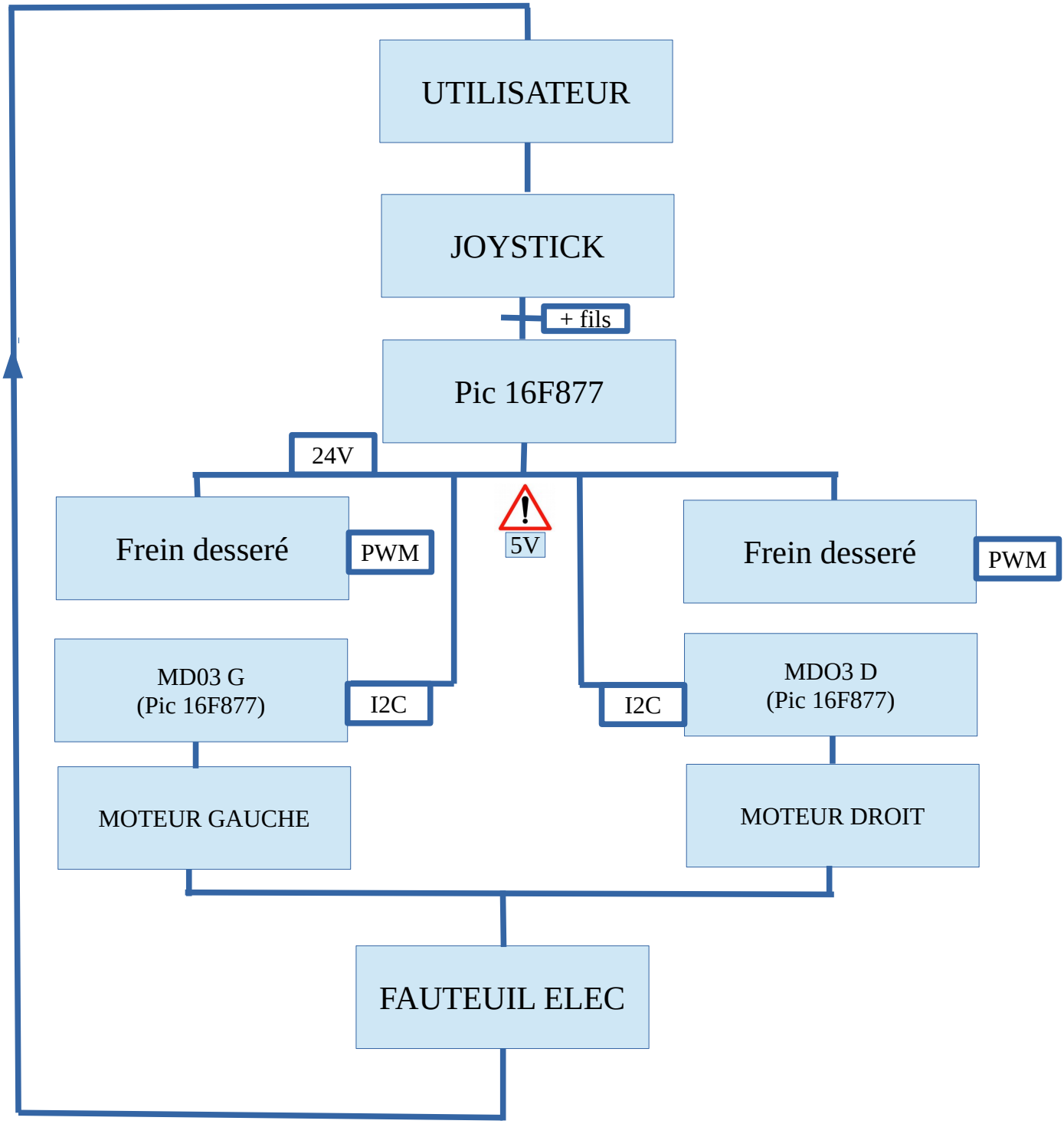
DIAGRAMME DE GANTT

Ci dessous le programme prévisionnel sous la forme de gant.



SYNOPTIQUE

SYNOPTIQUE: Qui offre une vue générale d'un ensemble :



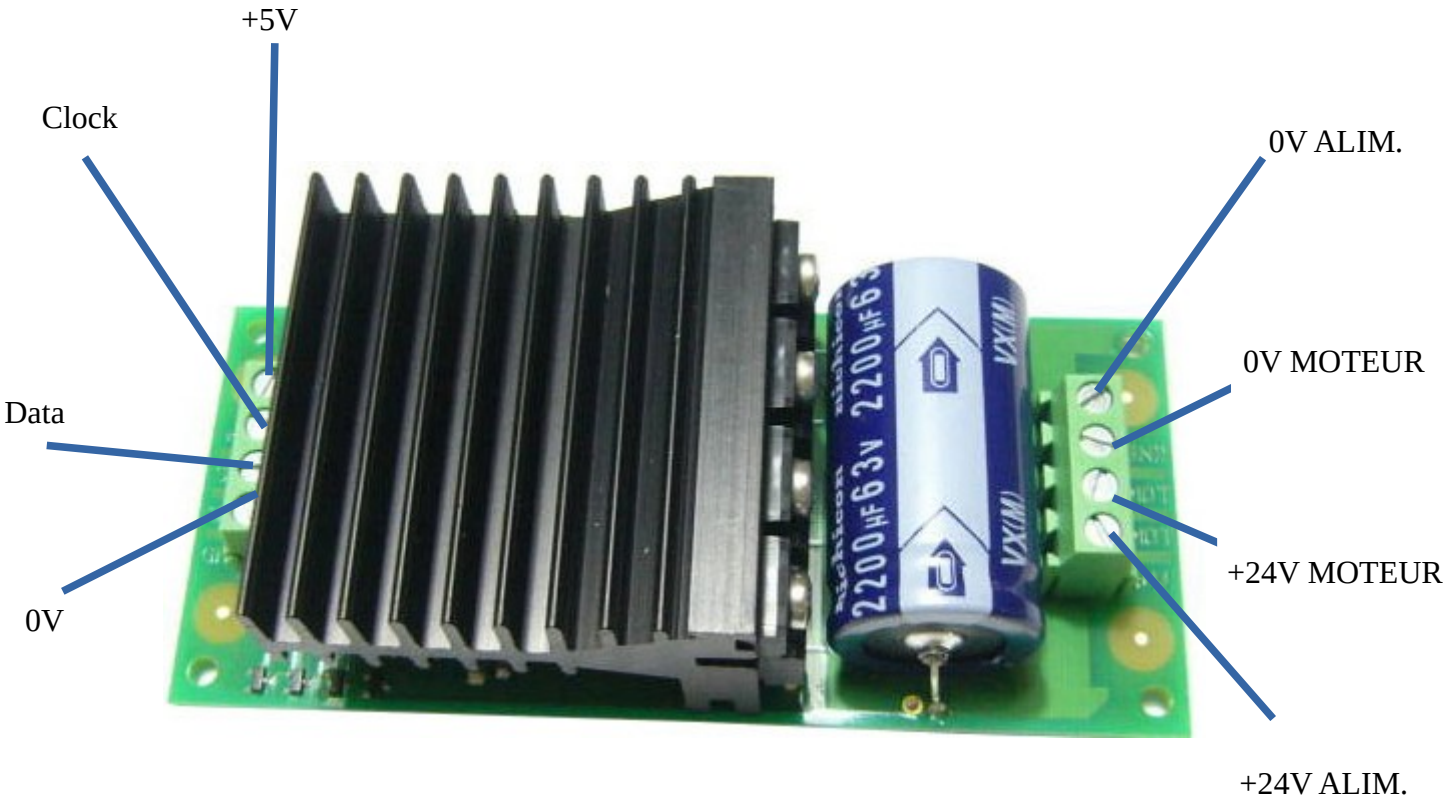
LES DIFFERENTS ORGANES DU FAUTEUIL ELECTRIQUE

Carte de commande moteur (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.

Partie Commande

Parte Opérative

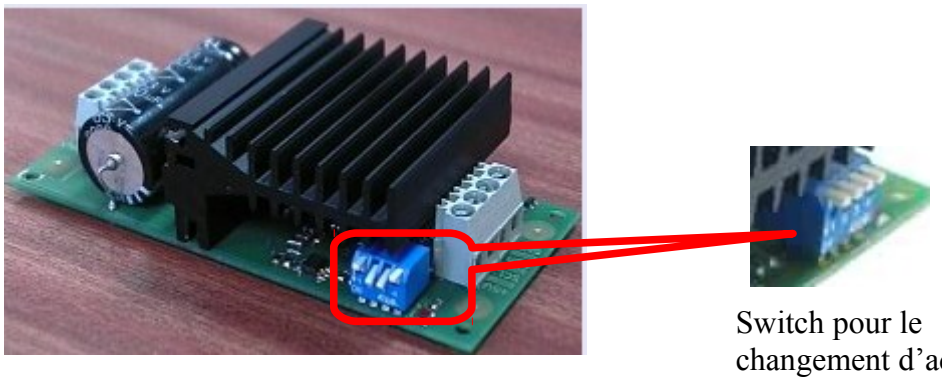


Pour commander nos deux modules MD03 nous utilisons l'I2C.
Ce mode de communication nous permet d'utiliser jusqu'à 8 modules avec seulement 4 fils (Data, clock, alim, masse)

Nous devons définir à chaque module, son adresse. Dans notre cas nous utilisons deux modules de ce types, alors nous avons besoin de deux adresses différentes.

Nous avons choisi les deux premières adresses et configuré les modules en conséquence.
Via le tableau constructeur, nous avons pu configurer les switchs pour l'utilisation des modules.

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



Switch pour le changement d'adresse

Les registres utilisés :

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

Chaque registre correspond à une variable qui va commander un paramètre.

Le registre 0 correspond au sens de rotation : - 00 correspondant à l'arrêt
- 01 correspondant à la marche avant
- 02 correspondant à la marche arrière

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

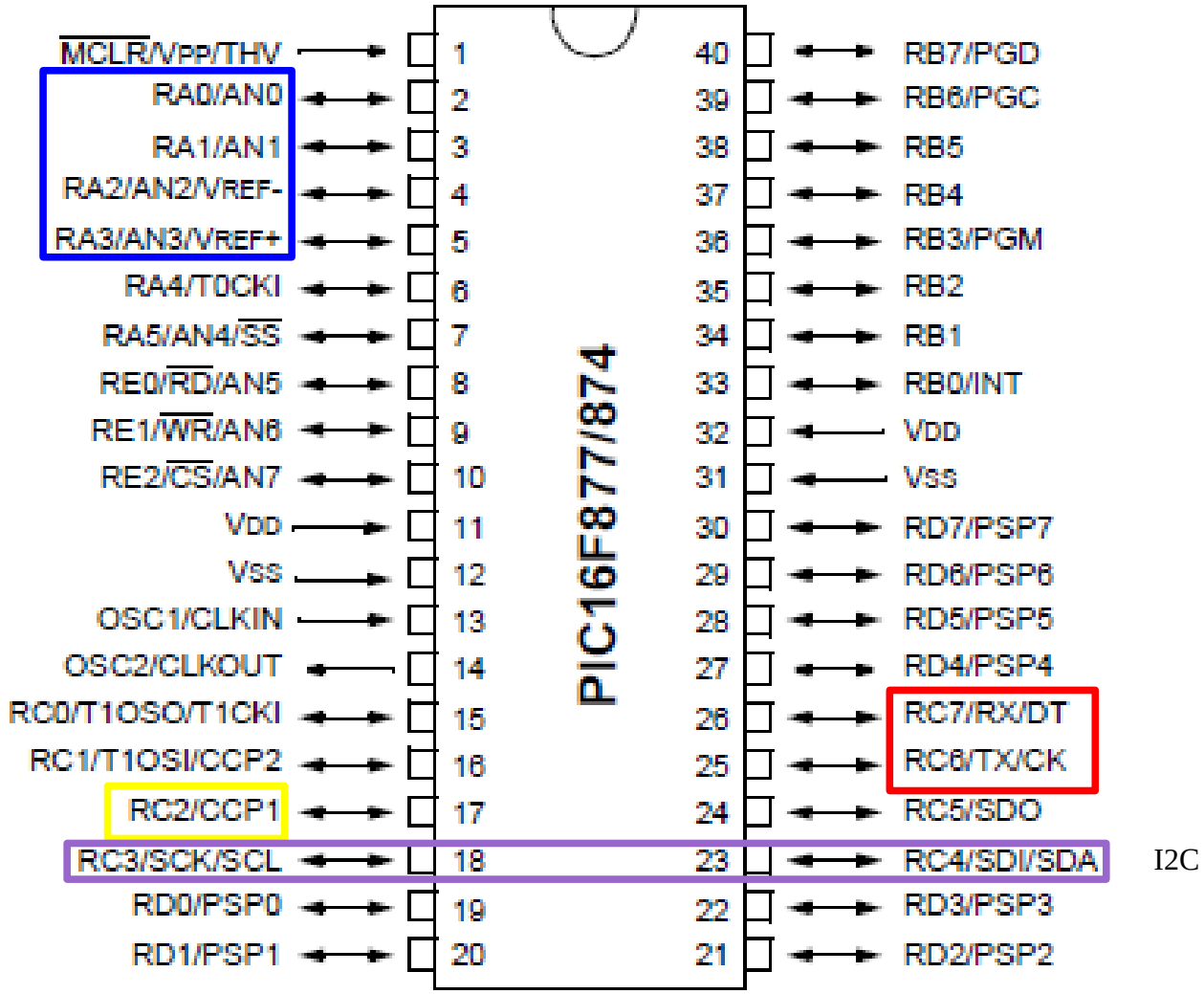
Le registre 3 correspond à l'accélération du moteur (varie entre 0&255)

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

PIC 16F877 :

Nous avons à disposition un Pic 16F877 :(voir schéma ci-dessous)

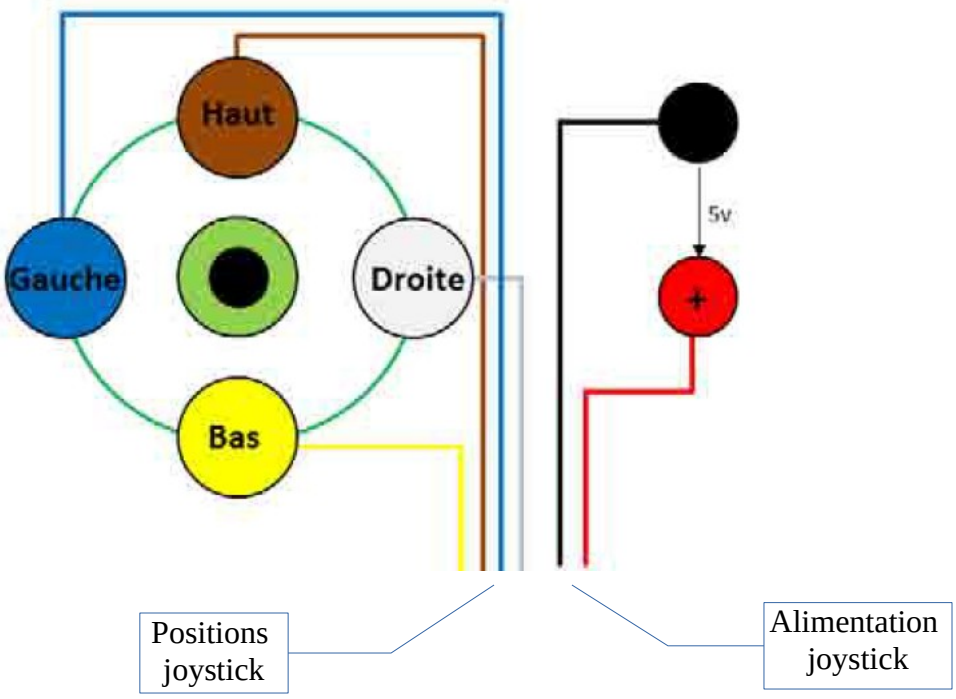
- Les entrées analogiques pour la communication du joystick avec le PIC (bleu).
- Pour moduler la tension d'alimentation des freins nous utilisons la sortie en PWM (jaune).
- La programmation du pic se fait par une liaison RS232 (Rouge).



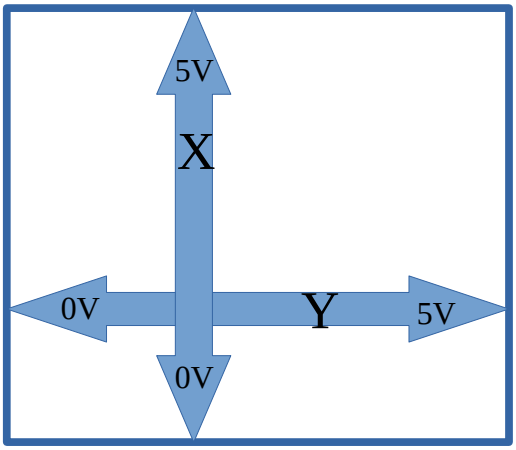
Suite à un problème de PWM nous avons procédés au changement de ce dernier.

Le 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.



La tension du joystick varie entre 0 et 5 V et suivant les axes X et Y.
En position médiane, nous avons 2,5 V.



Nous avons effectué quelques tests pour obtenir la plage de tension du joystick
Nous avons relevé les tensions suivantes :

- Si l'inclinaison est au maximum la valeur relevée est de 4.1V (voir image 1)
- Si l'inclinaison est au minimum la valeur relevée est de 2.4V (voir image 2)

Image 1

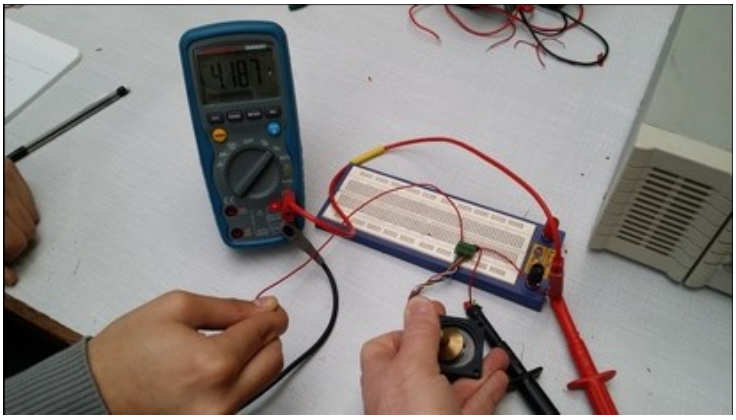
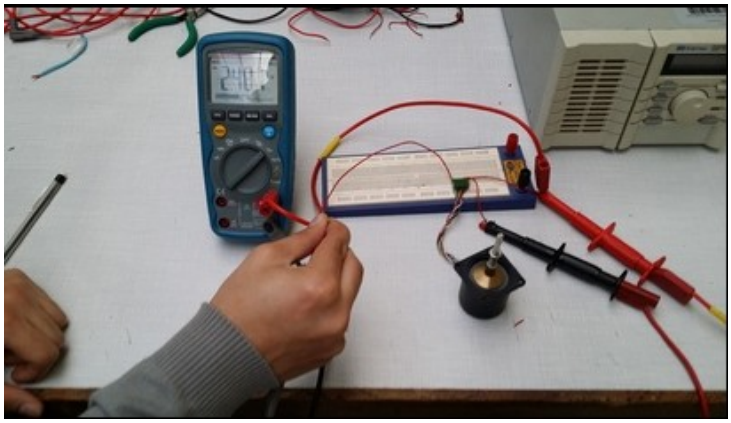
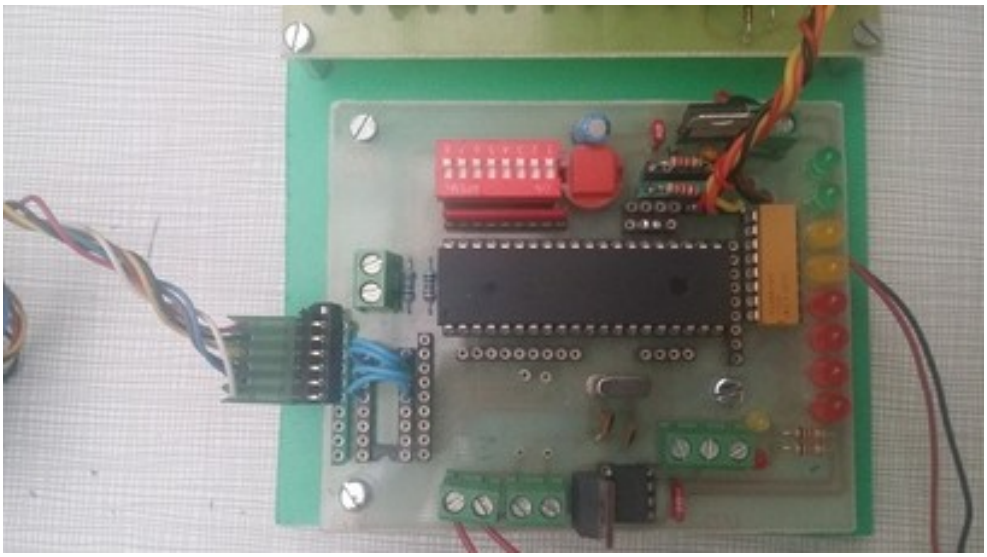
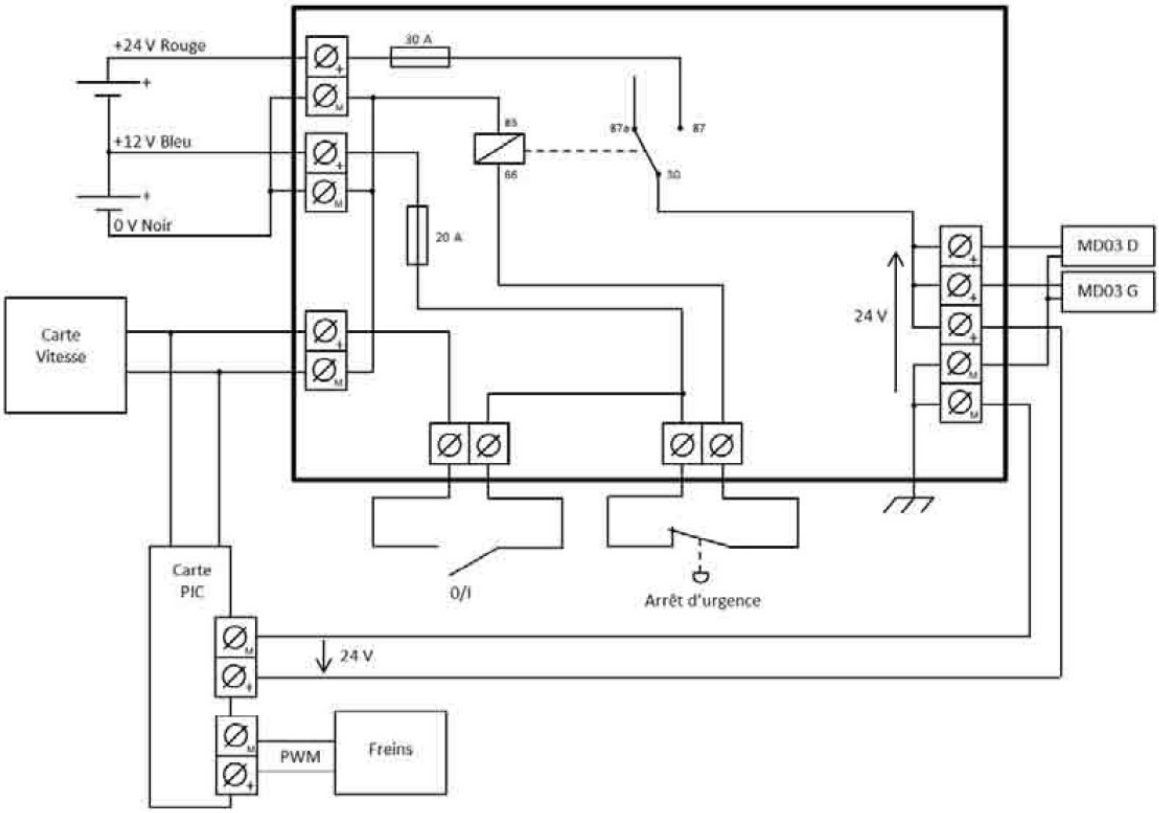


Image 2



La carte de commande : Celle-ci est alimentée en 12V depuis les batteries
L7805 : Réduit la tension de 12V a 5V
Pic 16F877A : C'est le cerveau de notre système qui gère toutes les communications et toutes les commandes entre les différents organes (frein, joystick..)





Liaison RS232 : C’est une liaison qui permet de programmer le microcontrôleur.
Cette liaison se fait grâce au MAX232.
RX : Pour la réception des données.
TX : Pour la transmission des données.
La connexion au PC se fait avec un connecteur DB9.

Liaison RS232

Frein à perte de tension :

Les freins sont intégrés au moto-réducteur, lorsqu'une tension est appliquée au frein, un champ électromagnétique est crée, ce qui entraîne l’ouverture du frein, dans le cas contraire s'il n’y a pas d’alimentation électrique, on a un blocage des freins.

Les freins sont alimentés par une tension entre 0 et 24V, pour obtenir cette tension nous pilotons un transistor par l’intermédiaire d’une PWM.
Suite à des essais nous avons déterminé que la tension nécessaire au déblocage des freins se situe aux environs de 17V. Une fois 17 V atteint les freins peuvent rester toujours desserrés mais avec une tension réduite pour augmenter l’autonomie des batteries.La valeur minimum est de 3,3 V.

Le protocole I2c :

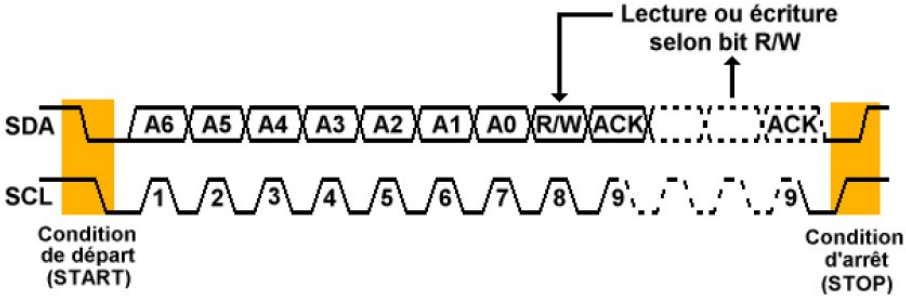
Pour effectuer le mode de communication de notre projet nous avons utilisés I²c.

Le but du mode I2c est de communiquer avec plusieurs modules MD03 (max.8)
avec seulement 4 fils.
Ces fils sont caractérisés de la façons suivante :

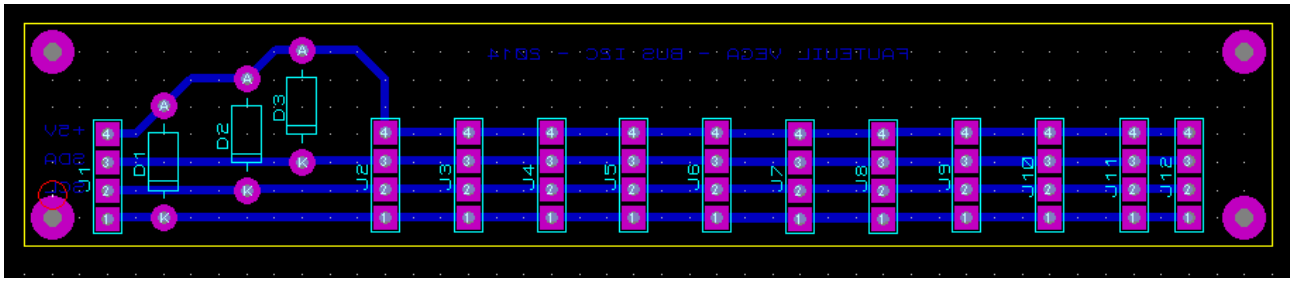
- Le signal de données « DATA »
- Le signal horloge « CLOCK »
- L'alimentation
- Masse

Le premier fil, DATA est utilisé pour transmettre les données. Le seconds fil CLOCK est utilisé pour transmettre un signal d'horloge synchrone (signale qui indique le rythme d'évolution de la ligne SDA). La vitesse de transfert du bus I²c n'est pas un élément très important de notre système, ce mode convient parfaitement.

Fonctionnement du bus I²c



Le bus I2C des deux modules MD03 :



LES MOTOREDUCTEURS

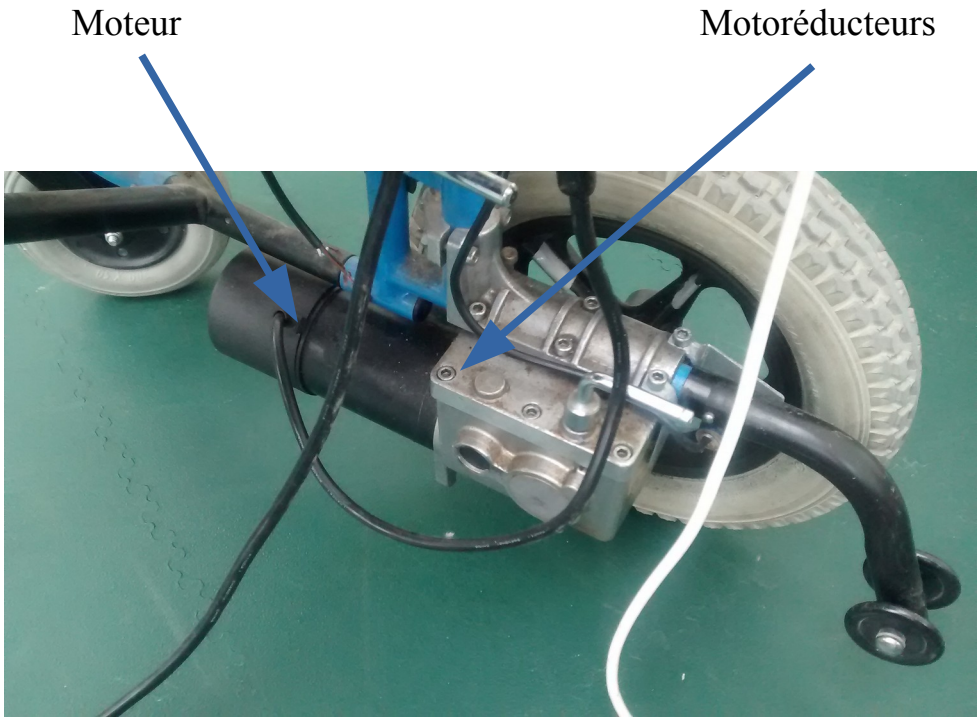
Le fauteuil roulant électrique est équipé de deux moteurs à courant continu à excitations en série avec un moto-réducteur qui entraînera les roues. Les moteurs sont accouplés à des freins électromagnétiques où en cas d'absence de tension on à un blocage des roues.

Nous pouvons moduler la tension 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



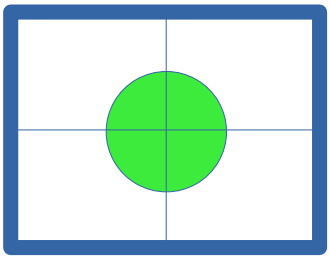
LOI DE COMMANDE

Pour effectuer une rotation avec le fauteuil vers la droite ou vers la gauche, nous avons mis en place, une loi de commande.
Cette loi de commande fonctionne de la manière suivante :

- Tourner à droite : la roue de gauche tourne dans le sens trigo tandis que la roue droite tourne dans le sens anti-trigo.
- Tourner à gauche : la roue de droite tourne dans le sens trigo tandis que la roue de gauche tourne dans le sens anti-trigo.

Pour effectuer une marche avant en ligne droite les roues droite et gauche tournent du même sens et à la même vitesse.
Pour effectuer une marche arrière en ligne droite les roues droite et gauche tournent du même sens mais à une vitesse deux fois moins élevée.

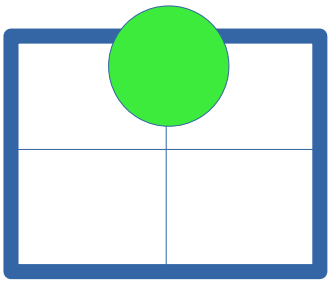
Point mort



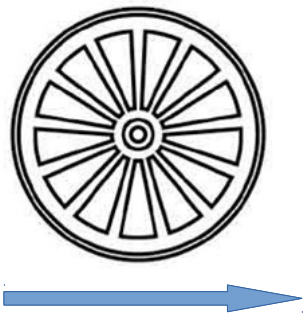
Roue droite

Roue gauche

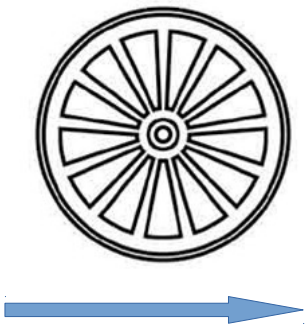
Marche avant



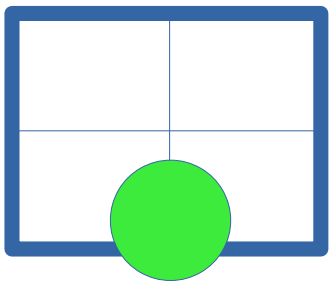
Roue droite



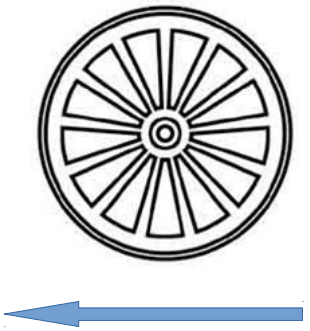
Roue gauche



Marche arrière

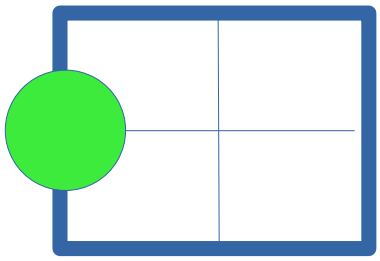


Roue droite

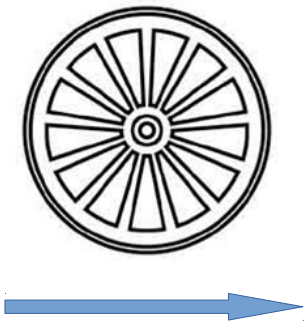


Roue gauche

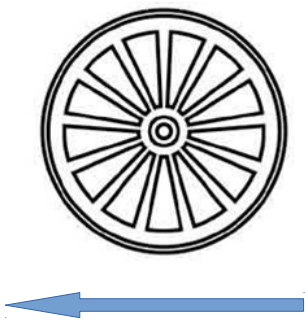
Tourner a gauche



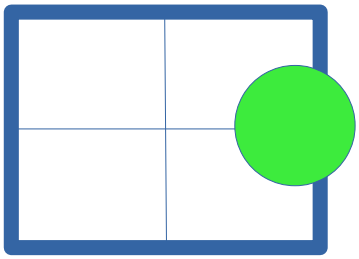
Roue droite



Roue gauche

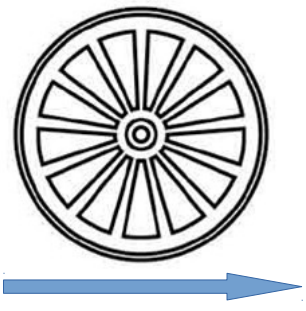


Tourner a droite



Roue droite

Roue gauche



LES TESTS ET ESSAIS

Pour avancer dans notre projet rapidement nous avons organisé une méthode de travail afin de ne pas nous perdre dans nos activités.

Voici l'ordre des différentes activités que nous avons réalisé :

- Tester le bon fonctionnement des moteurs, frein et la batterie

- Prise de connaissance des différentes cartes (carte puissance moteur, carte de commande)

- Prise de connaissance du protocole I2C

-Analyse du joystick (entrées analogiques, les plages de tensions)

-Prise de connaissance sur les lois de commandes

-Élaboration du programme I2C

-Essai de la procédure de communication entre la carte de commande et les modules MD03 via le bus I2C avec un petit moteur de test.

-Essai du programme de la PWM pour le serrage et desserrage des freins

-Élaboration d'une carte ISIS ARES

-Câblage

LE PROGRAMME

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
//#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#use i2c(MASTER, sda=PIN_C4, scl=PIN_C3)        // utilisation bus I2C

#include <MD03_2014.c>
// Ecriture_speed_MD03(int number,sens,speed)
   // number 0 ou 2 % OxB0 (OxB0 ou OxB2)
   // sens 0: Stop 1: Forward 2: Reverse
   // spped: 0..255

   int i,motG,motD,speedG,speedD,sensG,sensD,,AlphaD,Alpha,AccG,AccD;
   int Delta,Jaune,Bleu,Blanc;
   unsigned int enc1[4],enc2[4];
   unsigned int32 enc_M1,enc_M2;
   int valeurV,valeurD,ordreD,ordreG;
   int consigneV,consigneD;
   int sensRD,sensRG;
```

Initialisation générale PWM, Analogique, DS1307

```
void init_generale()
{
                                                  //Initialisation PWM
     setup_ccp1(ccp_pwm);                         //PWM basse
     setup_ccp2(ccp_pwm);                         //PWM haute
     setup_timer_2(t2_div_by_1,250,1);           // alpha compris entre 0 et 250
     set_pwm1_duty(0);
     set_pwm2_duty(0);
                                  //Initialisation Port A² pour acquisition Analogique
     setup_port_a( ALL_ANALOG );
     setup_adc( ADC_CLOCK_INTERNAL );

  motG=0;
  motD=1;
  accG=0;
```

```
 accD=0;
 delta=5;

 Ecriture_acc_MD03(0,MotG);
 Ecriture_acc_MD03(0,MotD);
}
```

Programme principal

```
void main ()
{
do
 {
 set_adc_channel( 1 );          //Choix "voie jaune axe y"
 delay_us(10);                  //Délai stabilisation avant acquisition
 consigneV=read_adc();          //Acquisition capteur
 set_adc_channel( 0 );          //Choix "voie BLEU axe x"
 delay_us(10);                  //Délai stabilisation avant acquisition
 consigneD=read_adc();          //Acquisition capteur


 valeurV=(consigneV-256)*2;        // Marche avant + traitement de la consigne
 valeurD=(consigneD-256)*2;


 -----------------ligne droite------------------
{
if(consigneD>108 && consigneD<148)

      if (consigneV>108 && consigneV<148)        //détection point mort
       {
        ordreD=0;
        ordreG=0;
        set_pwm1_duty(0);
        set_pwm2_duty(0);
       }
      else
       {                                         // si non marche avant
        ordreD=valeurV;
        ordreG=valeurV;
        set_pwm1_duty(255);
```

```
        set_pwm2_duty(255);
       }

    --- Detection marche arrière---

 if(consigneV<=108)
     {
       ordreD=((valeurV-256)*-1)/2;//en marche arriere on fais *-1 pour faire varier de
0 a max
       ordreG=((valeurV-256)*-1)/2;
        set_pwm1_duty(255);
        set_pwm2_duty(255);
       sensRG=2;
       sensRD=2;
     }
     else
     {
       sensRG=1;
       sensRD=1;
     }
     output_d(Bleu);
}

------------tourner sur place------------------

else if(consigneV<=148 && consigneV>=108)
   {

     if(consigneD>148)        // tourner a gauche

     {

     sensRG=2;
     sensRD=1;
     ordreD=valeurD/2;       // division pour limiter la vitesse
     ordreG=valeurD/2;      // division pour limiter la vitesse
     set_pwm1_duty(255);
     set_pwm2_duty(255);
     output_d(0X0F);

      }
```

```
    else if(consigneD<108)     //tourner a droite
    {

    sensRG=1;
    sensRD=2;


    ordreD=((valeurD-256)*-1)/2;     // division pour limiter la vitesse
    ordreG=((valeurD-256)*-1)/2;     // division pour limiter la vitesse
    set_pwm1_duty(255);
    set_pwm2_duty(255);
    output_d(0xF0);
    }


   else   // arret
    {
    output_d(0x55);                    //visualisation point mort
     sensRG=0;
     sensRD=0;
     set_pwm1_duty(0);
     set_pwm2_duty(0);
    }

 }

else

 ---------arret---------

{
output_d(0xAA);  //visualisation point mort
sensRG=0;
sensRD=0;
ordreD=0;
ordreG=0;
set_pwm1_duty(0);
set_pwm2_duty(0);
}
```

```
  Ecriture_speed_MD03(0,sensRG,ordreD);          //roue droite (module,sens,
vitesse)i2c
  Ecriture_speed_MD03(2,sensRD,ordreG);          //roue gauche (module,sens,
vitesse)


  }while(true);
}
```

*

CONCLUSION

Lors de notre projet nous avons pu mettre en pratique nos connaissances théoriques acquises durant notre formation.

Nous avons eu quelques difficultés au niveau de la programmation notamment avec l'I2C et la loi de commande.

Enfin, le projet nous a permis d'être autonome et méthodologique à travers les différents travaux effectués. Cela peut être une expérience dans notre futur vie professionnelle.