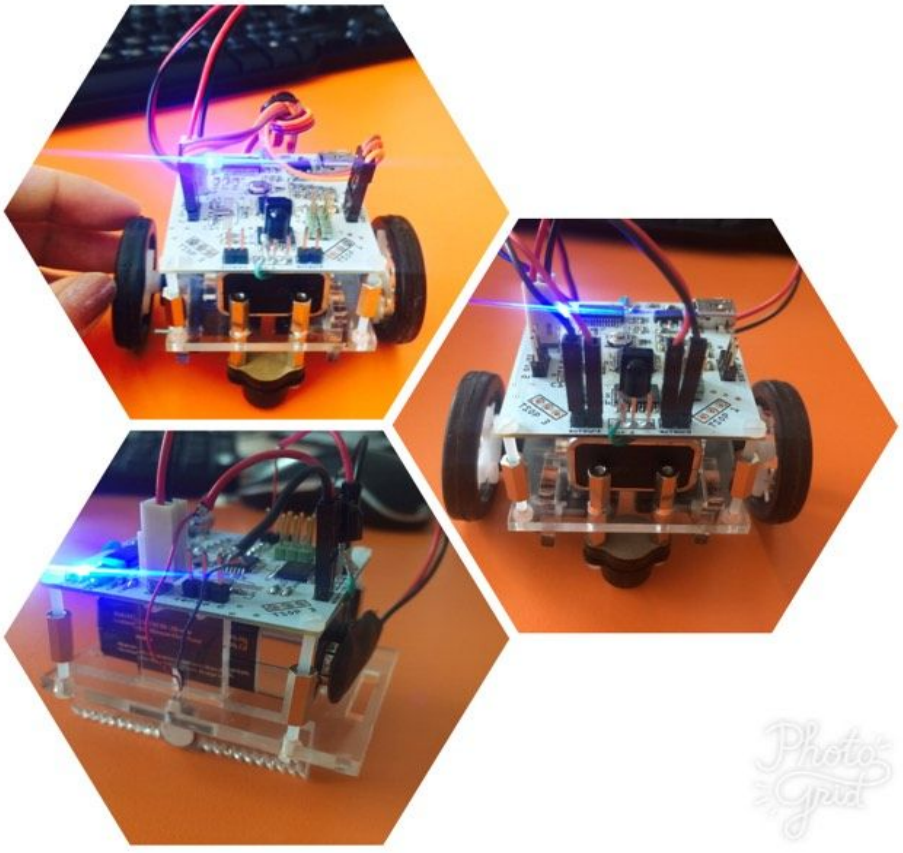
Micro-robots communicants



Réaliser par: Xinyue XU

Encadré par: Xavier REDON

Remerciement

Au terme de ce travail, je saisis cette occasion pour exprimer mes vifs remerciements à toute personne ayant contribué, de près ou de loin, à la réalisation de ce travail.

Je souhaite tout d'abord remercier mon encadrant, monsieur Xavier Redon, qui m'a encadré avec grande patience durant ce projet. Les conseils il m'a donné éteint très utiles.

Ensuite, je souhaite remercier les membres qui travaillent dans le Fabricarium. Il m'a aidé beaucoup comment utiliser les machines pour fabriquer les châssis avec la patience.

Je voudrais remercier également aux membres du jury de soutenance, qui l'ont honorés en acceptant de juger ce projet.

Enfin, je voudrais remercier tous les enseignants de département IMA, merci pour votre patience.

Sommaire

I. Introduction

Ce stage a été effectué à la fin de quatrième année de cycle d'ingénieur à l'Ecole Polytechnique de Lille 1 en Informatique Microélectronique Automatique. J'ai fait ce stage dans le bureau de monsieur Xavier Redon, pour une durée de 6 semaines.

Le but de ce stage est de me faire mieux comprendre les connaissances j'ai appris dans ma formation Informatique Microélectronique Automatique, et de développer des nouvelles compétences autour d'un sujet technologie.

Pour faire le projet de stage, j'ai choisi un sujet 'Micro-robots communicants', qui consiste à créer des robots mobiles relativement petits, simples et peu coûteux.

Dans ce rapport de stage, je vais d'abord faire une présentation du projet. Ensuite, je vais introduire le déroulement du projet, concernant trois parties principales: Création de la carte principale, Création des trois châssis et réaliser les fonctionnements des robots mobiles en program C. Enfin, je vais faire une conclusion de mon stage.

II. Présentation du projet

II.1 Objectif du projet

L'objet de ce projet est de concevoir et fabriquer des robots mobiles relativement petits, simples et peu coûteux. Les robots peuvent communiquer entre eux par infrarouge. La simplicité et le bas côut de fabrication des robots doit permettre d'en fabriquer en nombre suffisant pour simuler des comportements d'essaim d'insectes.

II.2 Description du projet

L'objectif de mon stage est de réaliser une petite carte de contrôle de robot mobile. Les robots pourront avoir trois types de motorisations : vibreurs, servo-moteurs continus et micro-moteurs.
Il faut d'abord concevoir la carte en se basant sur les cartes déjà conçues à l'école. Il m'est demandé de partir d'une carte basée sur un ATMega328p et un contrôleur Ethernet. J'ai retiré ce dernier, le convertisseur de niveaux et d'autres composants inutiles pour mon projet. Il faut ajouter un contrôleur de moteur (TB6612), des détecteurs infrarouges (3 TSOP IR) , une LED infrarouge et des lignes pour les servo-moteurs. La carte doit être la plus petite possible, il a été un temps envisagé de positionner le contrôleur de moteurs sur la face inférieure. Par la suite, il faudra écrire le code ATMega328p pour générer les PWM nécessaires aux servo-moteurs ou au contrôleur de moteurs. Pour la communication nous utiliserons le protocole RC5 qui permet une certaine immunité à la lumière ambiante.

Enfin, nous allons tester les trois motorisations: vibreurs, servo-moteurs continus et micro-moteurs pour vérifier le bon fonctionnement de la carte. Si plusieurs robots fonctionnent certains seront programmés pour repérer les autres et les poursuivre.

II.3 Choix techniques

Pour réaliser ce projet, j'ai besoin des matériels, des logiciels et des outils. J'ai établi la liste de matériels principaux comme le tableau suivant:

Matériel	Logiciels	Outils
ATMEGA328P	Fritzing (Pour concevoir la carte principale)	La machine découpeuse (couper les matériels pour les châssis)
leds infrarouges		
recepteurs infrarouges		
servo moteurs		
vibration moteurs		
TB6612FNG	Inkscape (Pour concevoir les châssis)	
roue de balance		
roues libres		
batteries		
moteurs	Compilateur GNU (Pour écrire et uploading les programs)	
USB connecteurs		
USB chips		
Les composants CMS		

III. Avancement du projet

III.1 Création de la carte principale

Pour la carte principale, il faut d'abord dessiner le schématique dans le logiciel Fritzing.

J'ai conçu la carte en se basant sur les cartes déjà conçues à l'école. Il faut je concevoir une carte basée sur un ATMega328p et un contrôleur Ethernet. A la référence des cartes déjà conçues à l'école, j'ai retiré ce dernier, le convertisseur de niveaux et d'autres composants inutiles pour mon projet.

Dans la schématique de ma carte, il y a 4 partie: la partie d'alimentation, la partie moteur, la partie de USB et la partie de microcontrôleur.

La partie d'alimentation est comme la figure suivante:

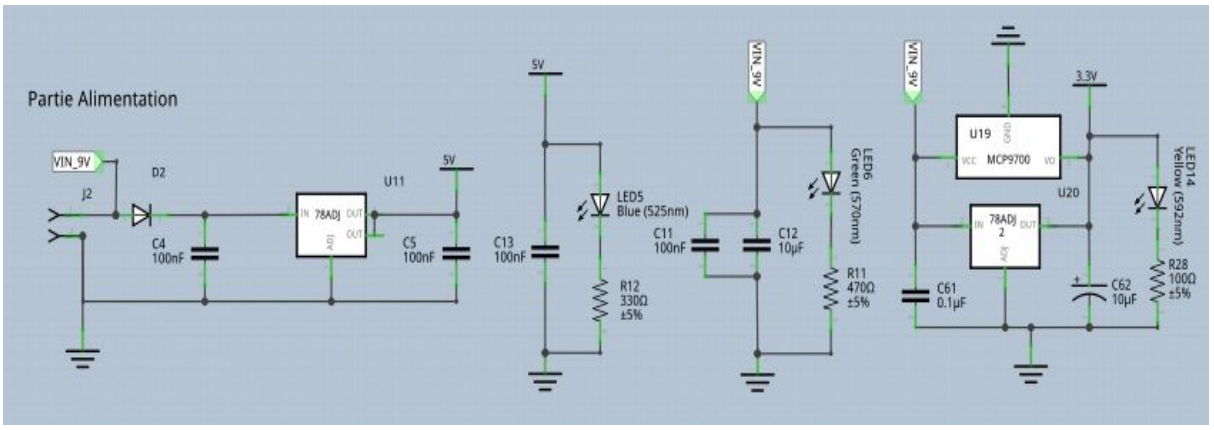


Figure1: Partie Alimentation

La partie d'alimentation permet de fournir la tension aux autres parties. J'ai choisi une alimentation externe de la tension 9V (via une batterie de 9V) et j'ai fait des circuits pour transformer la tension à 5V et 3.3V pour alimenter les autres parties comme la partie microcontrôleur, ex.

La partie moteur est comme la figure suivante:

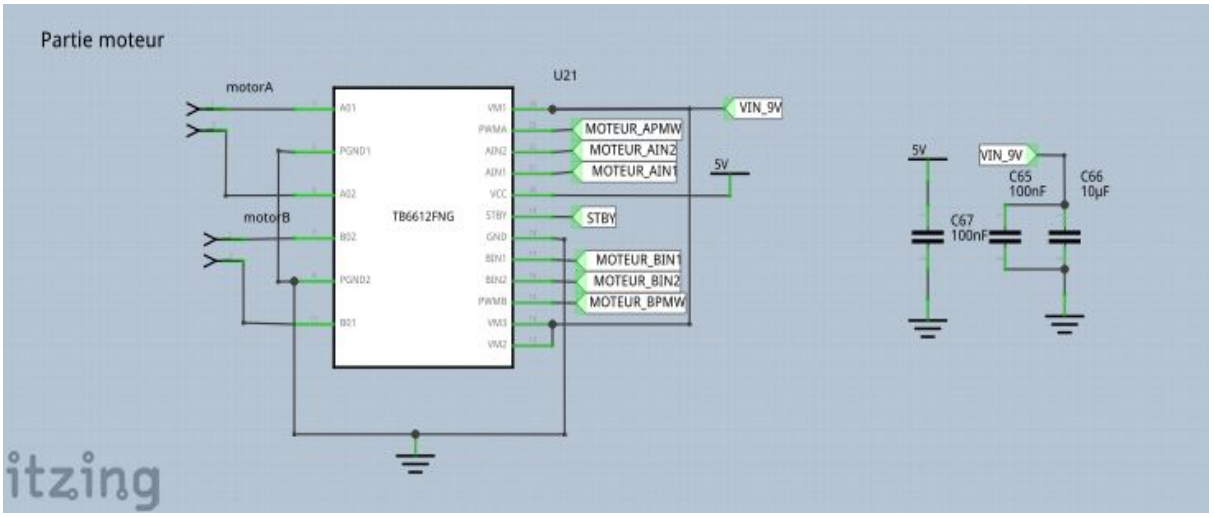


Figure2: Partie moteur

J'ai ajouté une partie moteur autour d'une contrôleur de moteur (TB6612FNG), celui permet de contrôler les 2 vibreurs de viber aux vitesses différentes avec la commande PWM ou permet de contrôler les 2 moteurs à courant continu d'avoir rotations en 2 sens et aux vitesses différentes.

La partie USB est comme la figure suivante:

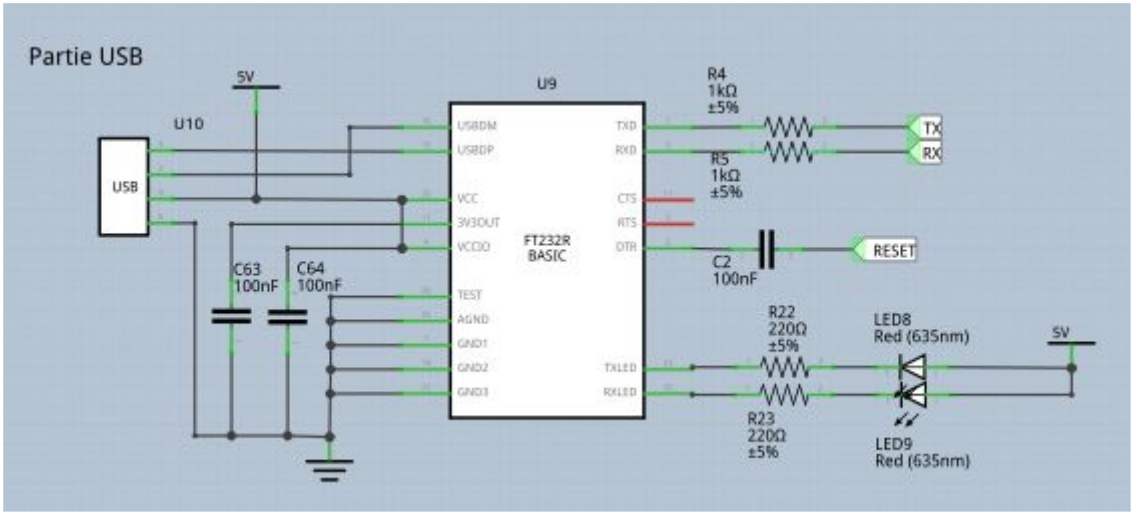


Figure3: Partie USB

J’ai fait ce partie USB pour télécharger les programs d’ordinateur à la carte pour programmer la carte. Pour le USB connecteur, j’ai choisi le micro USB connecteur.

La partie microcontrôleur est comme la figure suivante:

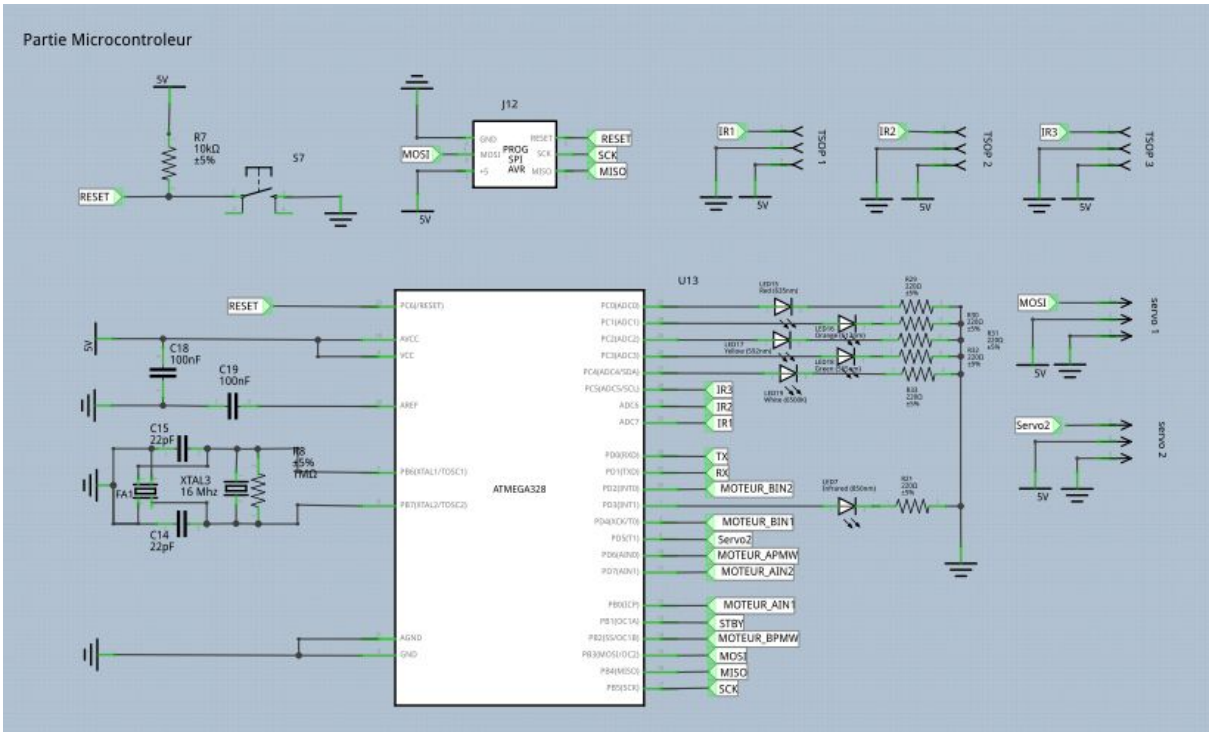


Figure4: Partie microcontrôleur

Pour la partie microcontrôleur, j’ai ajouté des détecteurs infrarouges (3 TSOP IR) , une LED infrarouge et des lignes pour les servo-moteurs. J’ai aussi ajouté un quartz de la taille petite à la place du quartz de la taille grande, comme ça on peut souder soit un quartz petit soit un quartz grand. Les utilisations de la LED infrarouge et des détecteurs infrarouges est de permettre les robots mobiles faire les mouvements un suive l’autre. Ou on peut commander les mouvements des robots mobiles avec les télécommandes qui en protocole RC5.

Après concevoir la schématique de la carte, j'ai fait le routage. Quand je faisais le routage, j'ai essayé de faire une carte la plus petite possible. Le PCB j'ai fais est comme la figure suivante:

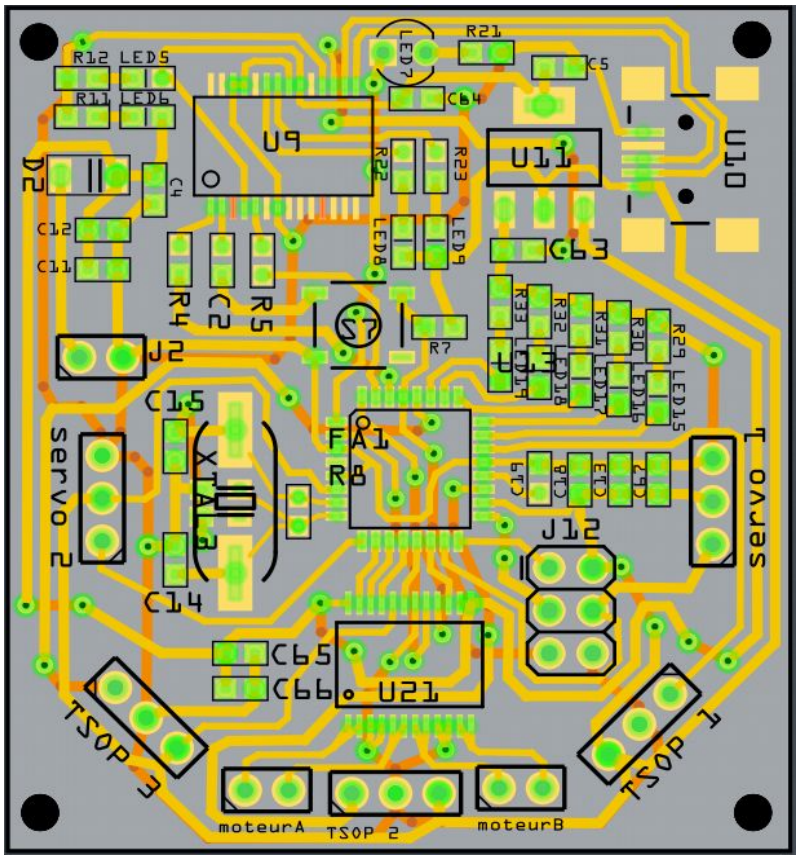


Figure5: PCB de la carte

Après j'ai soudé la carte, j'ai finalement fais ma carte comme la figure suivante:

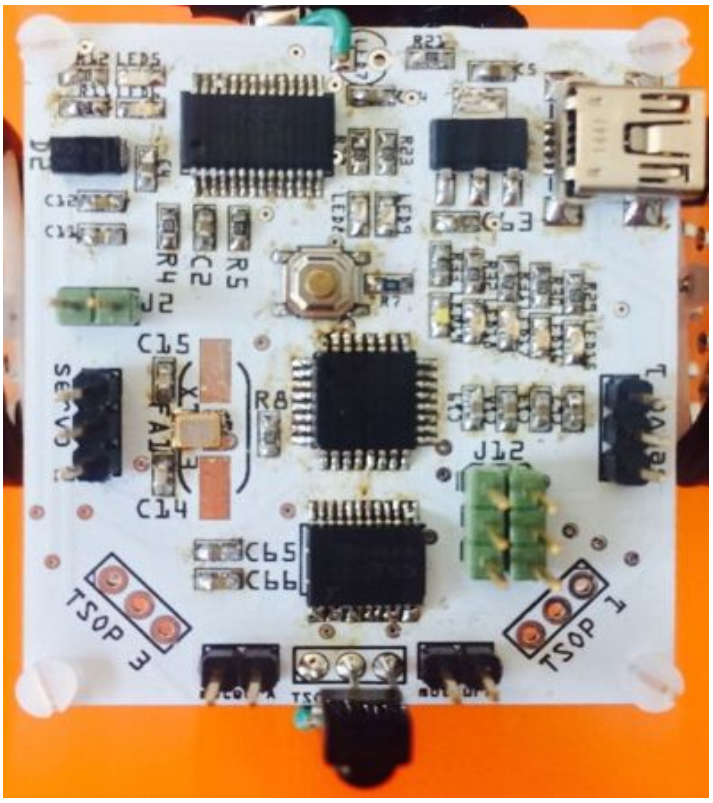


Figure6: Photo de la carte

La carte est très petite et il marche très bien. Mais il y a quelques problèmes à modifier:

- ➢ Le sens de TSOP1 est inversé dans le PCB, il faut qu'on prendre attention quand on souder.
- ➢ les TSOPs doivent être lié à les broches d'interruption mais j'ai les lié à les broches de ADC, il faut modifier. Mais pour l'instant j'ai juste lié le broche de TSOP2 (un de détecteurs infrarouges) à broche de LED infrarouge avec un fil externe, et j'ai pas soudé LED infrarouge pour cette première carte.

Et il y a encore quelques choses à faire pour améliorer la carte:

- ➢ Pour la partie d'alimentation, il faut que je ajouter un interrupteur pour commander le pile connecté ou déconnecté.

➢ On peut lier le VCC de chaque TSOP directement sur les pins de ATmega pour qu'on peut choisir lequel TSOP on veut alimenter.

Les deux problèmes à corriger et les deux problèmes à améliorer doivent être prise en compte à la prochaine version de la carte.

III.2 Création des 3 châssis

Pour les châssis, j'ai créé trois types: un châssis de robot qui marche avec servo-moteur, un châssis de robot qui marche avec moteur continu et un châssis qui marche avec vibreurs.

J'ai utilisé le logiciel 'inkscape' pour concevoir les châssis.

❖ Pour le châssis de servo-moteur, j'ai essayé 2 fois et j'ai fait 2 versions.

La première fois, j'ai mis la batterie comme la figure suivants, et j'ai utilisé les 2 roues grandes:

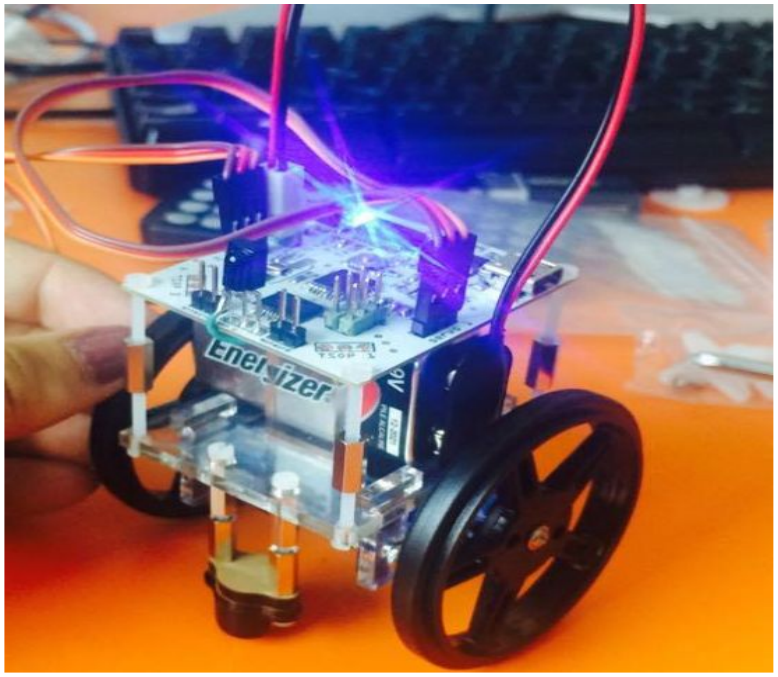


Figure7 châssis de servomoteur version1

Mais le problème est que c'est pas très stable. Quand j'ai lui commander de courir rapidement, il va tomber car le batterie est lourd, le centre de gravité est très haut. Donc j'ai corrigé le châssis à changer la direction de batterie et j'ai utilisé les 2 roues plus petites. La version finale est comme la figure suivante:

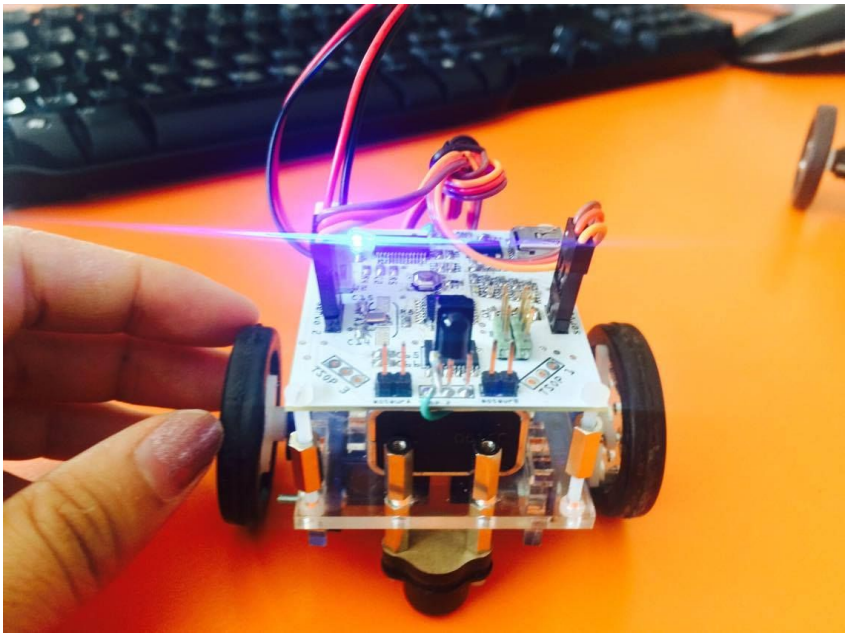


Figure8 châssis de servomoteur version finale

La figure svg de la version finale ce que j'ai conçu est comme suivant:

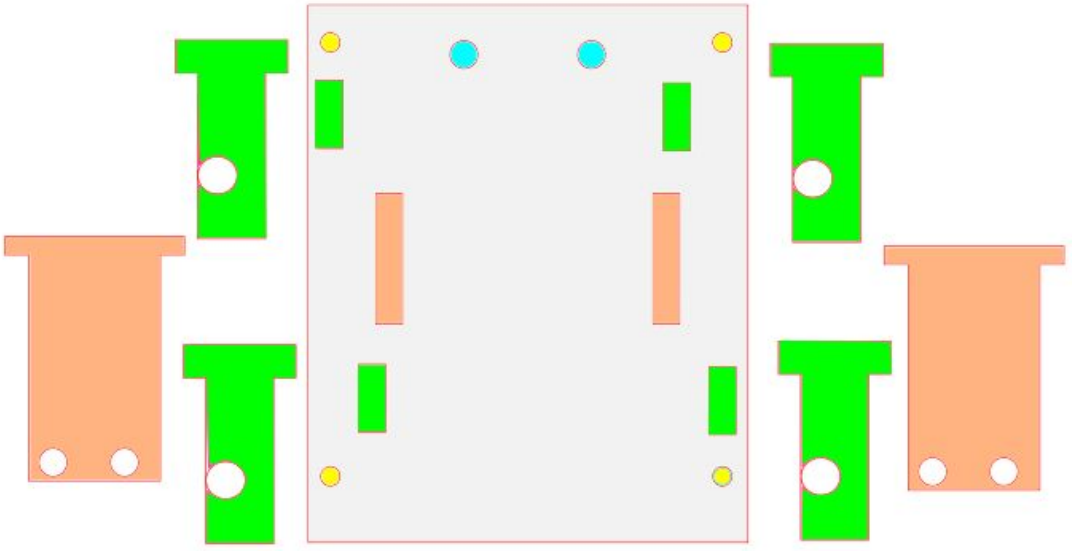


Figure7: châssis de servo-moteur version finale en svg

Pour mieux expliquer ce châssis de servo-moteur, j'ai coloré ce image à 5 couleurs. La plaque gris est la corp principale du robot. Les trous jaunes sont pour fixer la carte. Les 2 trous bleus sont pour fixer la roue de balance. Les corps verts sont pour fixer les 2 servomoteurs et les corps oranges permettent de fixer la batterie de 9V.

La version finale est plus stable que la version première, mais il est encore pas suffit. Donc je propose que, pour la prochaine version, on peut ajouter 2 trous pour ajouter une roue de balance, comme les 2 trous rouges dans la figure suivante:

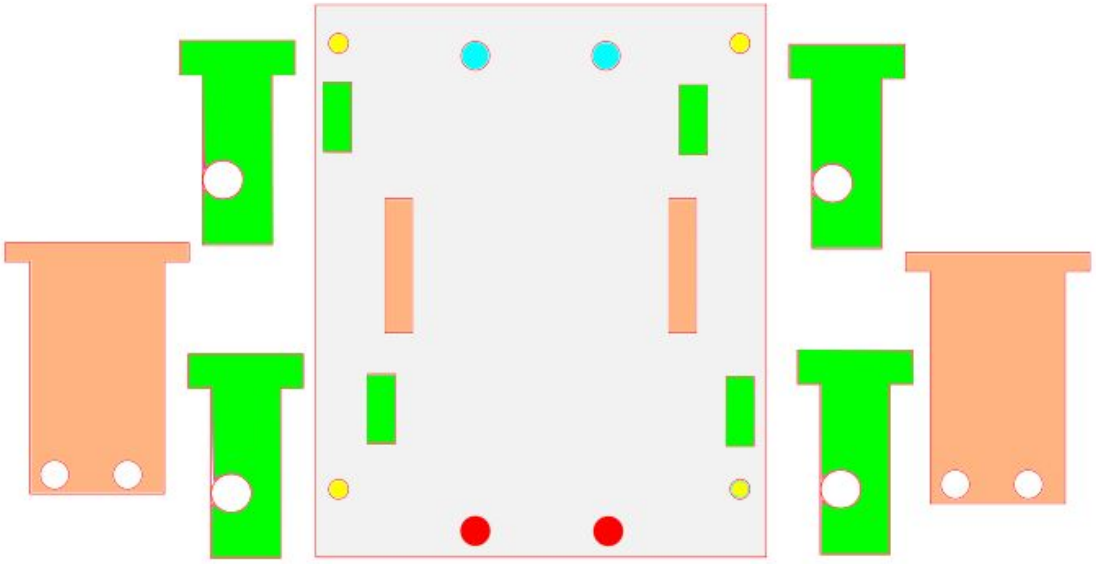


Figure7: châssis de servo-moteur la version je propose pour améliorer

Je crois si on ajoute les 2 trous rouges pour fixer une roue de balance extra comme la figure au dessus, le robot de servomoteur sera assez stable. Cette solution doit être prise en compte si on a le temps de continuer ce projet.

- Pour le châssis de moteur à courant continu, c'est à peu près la même chose que le châssis de servomoteur, mais la taille de moteur à courant continu est un peu plus grande que servomoteur. Donc j'ai juste modifié

un peu la distance de corps verts (dans figure7). La châssis de moteur à courant continu est comme la figure suivante:

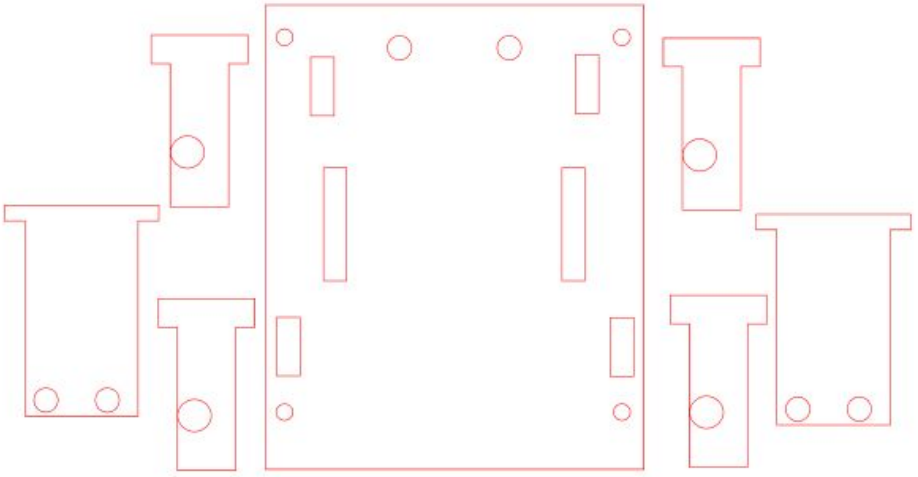


Figure7: châssis de moteur à courant continu en svg

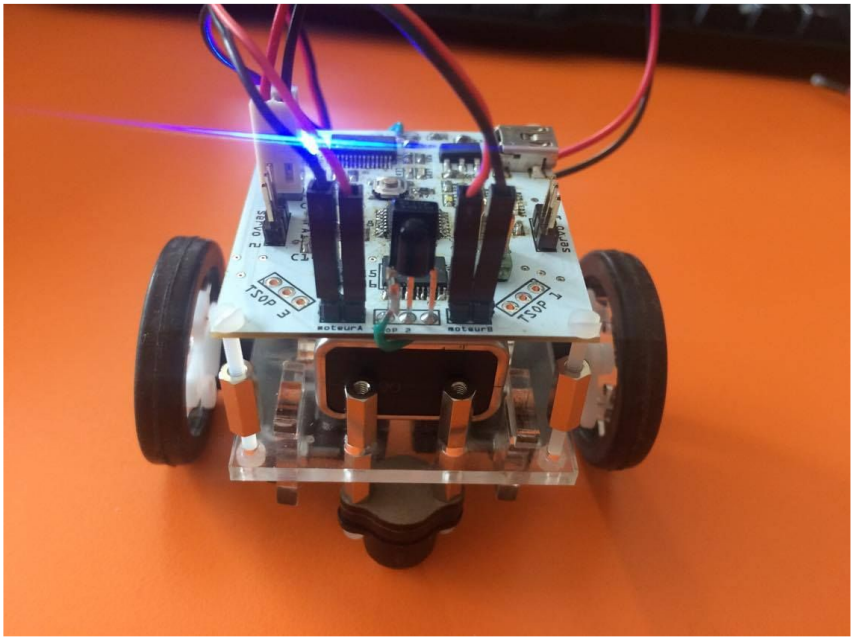


Figure8: châssis de moteur à courant continu

Comme j'ai dis pour améliorer le châssis de robot de servo moteur, on peut utiliser la même solution pour améliorer le châssis de robot de moteur à courant continu.

❖ Pour le châssis de vibreur, j'ai effectué comme la figure suivante:

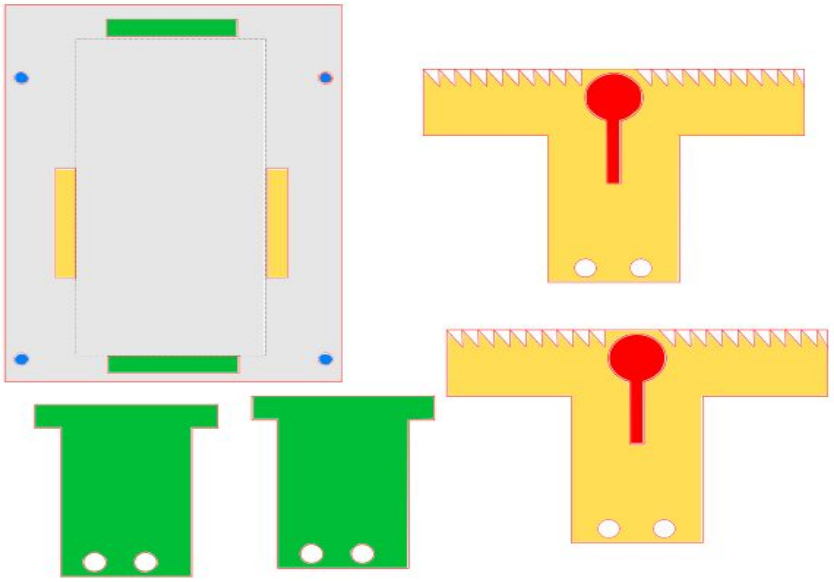


Figure9: châssis de robot de vibreur en svg

Comme la figure au dessus, la partie grise est le corp principal du robot. Les 4 trous bleus sont pour fixer la carte. Les corps verts permettent de fixer la batterie. Les corps jaunes sont pour fixer la batterie et ils peuvent aussi fixer les 2 vibreurs avec les 2 trous rouges. Les dents triangles dans les corps jaune permettent le robot de marcher dans le sens propre.

Enfin, j'ai assemblé le châssis de vibreur comme la figure suivante:

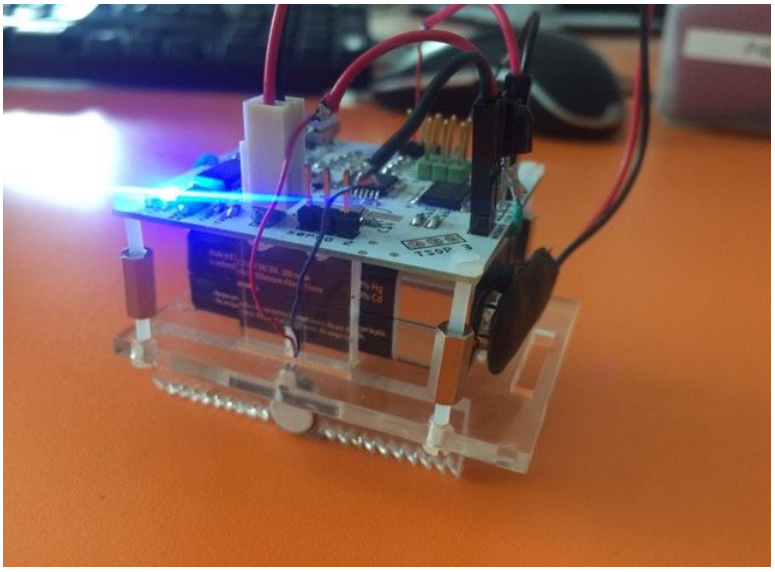


Figure10: châssis de robot de vibreur

III.3 Réaliser les fonctionnements en program C

Pour contrôler les robots mobiles, j'ai décidé d'écrire les programs en C et le compiler avec compilateur GNU.

A cause de temps, j'ai juste réalisé à écrire un program pour contrôler le mouvement de robot de servomoteur avec un télécommande PHILIPS qui est en protocole RC5.

Pour réaliser de contrôler le mouvement de robot de servomoteur avec un télécommande PHILIPS, il y a 2 partie principales: la partie de contrôler servomoteur et la partie de recevoir les commandes envoyé par le télécommande avec le récepteur d'infrarouge.

D'abord, j'ai étudié de réaliser la partie d'infrarouge.

J'ai trouvé les exemples sur Github (https://github.com/guyc/RC5). J'ai download le libraire RC5.h, et basé sur le code RC5.c trouvé dans le site Github, j'ai modifié pour l'appliquer sur ma carte. Dans le fichier RC5.c, pour initialiser rc5, j'ai fait comme suivant:

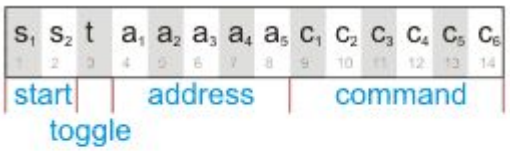
```
void RC5_Init()
{
    /* Set INT1 to trigger on any edge */
    EICRA |= _BV(ISC10);
    /* Set PD3 to input */
    DDRD &= ~_BV(PD3);

    /* Reset Timer1 Counter */
    TCCR1A = 0;
    /* Enable Timer1 in normal mode with /8 clock prescaling */
    /* One tick is 500ns with 16MHz clock */
    TCCR1B = _BV(CS11);

    RC5_Reset();
}
```

Parce que sur ma carte, le récepteur d'infrarouge est lié sur le pin PD3 (Timer1) de la carte atmega328p, donc j'ai initialisé Port D et Timer1. (Les autres detailes voir le fichier RC5.c).

Pour tester le fonctionnement du récepteur d'infrarouge, j'ai écrit un fichier serial.c pour établir une fonction de transmission les données de commande via une porte série et l'afficher sur l'écran. Et dans le fichier de main.c, j'ai écrit comme suivant:

```
int main()
{
    init_printf();
    RC5_Init();

    /* Set PB5 to output */
    DDRB |= _BV(PB5);

    /* Enable interrupts */
    sei();

    for(;;)
    {
        uint16_t command;

        /* Poll for new RC5 command */
        if(RC5_NewCommandReceived(&command))
        {
            /* Reset RC5 lib so the next command
             * can be decoded. This is a must! */
            RC5_Reset();

            /* Do something with the command
             * Perhaps validate the start bits and output
             * it via UART... */
            if(RC5_GetStartBits(command) != 3)
            {
                /* ERROR */
            }

            uint8_t toggle = RC5_GetToggleBit(command);
            uint8_t address = RC5_GetAddressBits(command);
            uint8_t cmdnum = RC5_GetCommandBits(command);
            printf("t=%d\n",toggle);
            printf("a=%d\n",address);
            printf("cmd=%d\n",cmdnum);

        }
    }

    return 0;
}
```

Basé sur le type de données de protocole RC5 j'ai trouvé sur internet:



A 14-bit RC5 sequence starts with two sync bits (always 1), followed by a toggle bit (which alternates value on each key press), a 5 bit device address, and a 6-bit command number. RC6 is a minor variation on the RC5 code.

J'ai vu qu'il y a trois trucs principaux: toggle, address et command. Donc j'ai utilisé les fonctions 'RC5_GetToggleBit()', 'RC5_GetAddressBits()', et 'RC5_GetCommandBits()' dans la librairie RC5.h, pour récupérer les trois types de données envoyé par le télécommande. Et après, j'ai les affiché sur l'écran.

Ensuite, j'ai étudié les caractéristiques de servomoteur. Pour programmer le servomoteur j'ai déjà connait comment faire avec les codes exemples dans IDE, mais la difficulté est de programmer le servomoteur par le langage C.
En langage C, il faut que je faire un signal PWM pour contrôler le servomoteur. Pour ma carte, il y a deux broches de données pour envoyer PWM aux 2 servomoteurs. Le broche de donnée de servomoteur A est lié sur le PIN PB3 du microcontrôleur, tant que le servomoteur B est lié sur le PIN PD5 du microcontrôleur. Donc il faut activer les registres TCCR2A, TCCR2B pour servomoteur A et les registres TCCR0A, TCCR0B pour servomoteur B.
Afin de comprendre ces configurations, il est nécessaire de consulter les tableaux de la datasheet correspondants:

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

Timer/Counter Control Register 0 A

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00

Timer/Counter Control Register 0 B

MODE	WGM02	WGM01	WGM00	DESCRIPTION	TOP
0	0	0	0	Normal	0xFF
1	0	0	1	PWM, Phase Corrected	0xFF
2	0	1	0	CTC	OCR0A
3	0	1	1	Fast PWM	0xFF
4	1	0	0	Reserved	-
5	1	0	1	Fast PWM, Phase Corrected	OCR0A
6	1	1	0	Reserved	-
7	1	1	1	Fast PWM	OCR0A

Waveform Generator Mode bits

CS02	CS01	CS00	DESCRIPTION
0	0	0	Timer/Counter0 Disabled
0	0	1	No Prescaling
0	1	0	Clock / 8
0	1	1	Clock / 64
1	0	0	Clock / 256
1	0	1	Clock / 1024
1	1	0	External clock source on T0 pin, Clock on Falling edge
1	1	1	External clock source on T0 pin, Clock on rising edge

CS bits

Après avoir vu les tableaux, j'ai initialisé les registres comme suivant:

```
void init(){
        DDRD |= (1<<DDD5);  //PD5 output (motB est OC0B)
        DDRB |= (1<<DDB3);   //PB3 output (motA est OC2A)
        //pour motA
        TCCR2A = 0x83;   // Fast PWM, TOP=0xFF
        TCCR2B = 0x05;   // clk(entree/sortie)/1024
        //pour mot B
        TCCR0A = 0x23; // fast PWM sur 0c0B
        TCCR0B = 0x05;
}
```

Et il y a deux registres OCR2A et OCR0B pour changer la vitesse de servomoteur A et servomoteur B. Après beaucoup de fois d'essayer, j'ai trouvé les données pour chaque type de mouvement du robot:

Pour avancer avec la vitesse normale ou vite:

```
void avancer(){
    OCR2A=198;
    OCR0B=22;
}

void avancer_vite(){
  OCR2A=242;
  OCR0B=13;
}
```

Pour reculer avec la vitesse normale ou vite:

```
void reculer(){
  OCR2A=182;
  OCR0B=24;
}

void reculer_vite(){
    OCR2A=170;    // reculer motA 13 motB 25
    OCR0B=25;
}
```

Pour tourner à droite ou à gauche:

```
void gauche(){
  OCR2A=255;
  OCR0B=22;
}

void droite(){
  OCR2A=197;
  OCR0B=255;
}
```

Pour s'arreter:

```
void arreter(){
    OCR2A=255;
    OCR0B=255;    // pour arreter
}
```

Enfin, j'ai mis les deux parties ensembles pour réaliser: si on tape un des boutons de télécommande, le récepteur d'infrarouge va reçu les données de la

commande, et la carte va contrôler les servomoteurs. J'ai écrit les codes comme suivant:

```
if(RC5_NewCommandReceived(&command))
{
    /* Reset RC5 lib so the next command * can be decoded. This is a must! */
    RC5_Reset();

    /* Do something with the command
     * Perhaps validate the start bits and output
     * it via UART... */
    if(RC5_GetStartBits(command) != 3)
    {
        /* ERROR */
    }
    uint8_t cmdnum = RC5_GetCommandBits(command);
    if(cmdnum==2){
      avancer();
      _delay_ms(200);
    }
    if(cmdnum==3){
      avancer_vite();
      _delay_ms(200);
    }
    if(cmdnum==8){
      reculer();
      _delay_ms(200);
    }
    if(cmdnum==9){
      reculer_vite();
      _delay_ms(200);
    }

    if(cmdnum==5){
      arreter();
      _delay_ms(200);
    }
    if(cmdnum==4){
      gauche();
      _delay_ms(200);
    }
    if(cmdnum==6){
      droite();
      _delay_ms(200);
    }
}
```

Une fois on tape le télécommande, la carte va reçu la commande. Le bouton 2 est pour avancer normalement, bouton 3 est pour avancer vite, etc.

J'ai testé le robot de servomoteur avec le télécommande PHILIPS et ça marche très bien. Malheureusement il reste plus le temps de programmer pour le robot de vibreur et le robot de moteur à courant continu.

IV. Conclusion

Ce projet s'inscrit dans le cadre de la fin de ma quatrième année d'IMA (Informatique, Microélectronique et Automatique) à Polytech Lille.

Ce stage m'a apporté une expérience très intéressante et enrichissante. L'objectif de mon stage était de faire un projet pour réaliser une petite carte de contrôle de robot mobile et concevoir les robots pourront avoir trois types de motorisations : vibreurs, servo-moteurs continus et micro-moteurs.

Ce projet m'a permis la mise en pratique de mes connaissances. J'ai également appris beaucoup durant la réalisation de ce projet, par exemple, j'ai appris comment utiliser le logiciel Fritzing (car avant j'ai juste utilisé Altium designer pour faire les cartes). J'ai aussi appris l'utilisation de logiciel inkscape pour concevoir les châssis. Pour fabriquer les châssis, j'ai appris comment utiliser la machine découpeuse dans la salle Fabricarium.

De plus, j'ai amélioré ma compétence de programmer. J'ai réussi de programmer le robot de servomoteur qui peut être contrôlé par un télécommande. Mais c'est un peu dommage que j'ai pas assez de temps pour écrire les programmes pour les autres 2 robots.

Ce stage est un excellent stage, j'ai appris beaucoup durant ce stage.