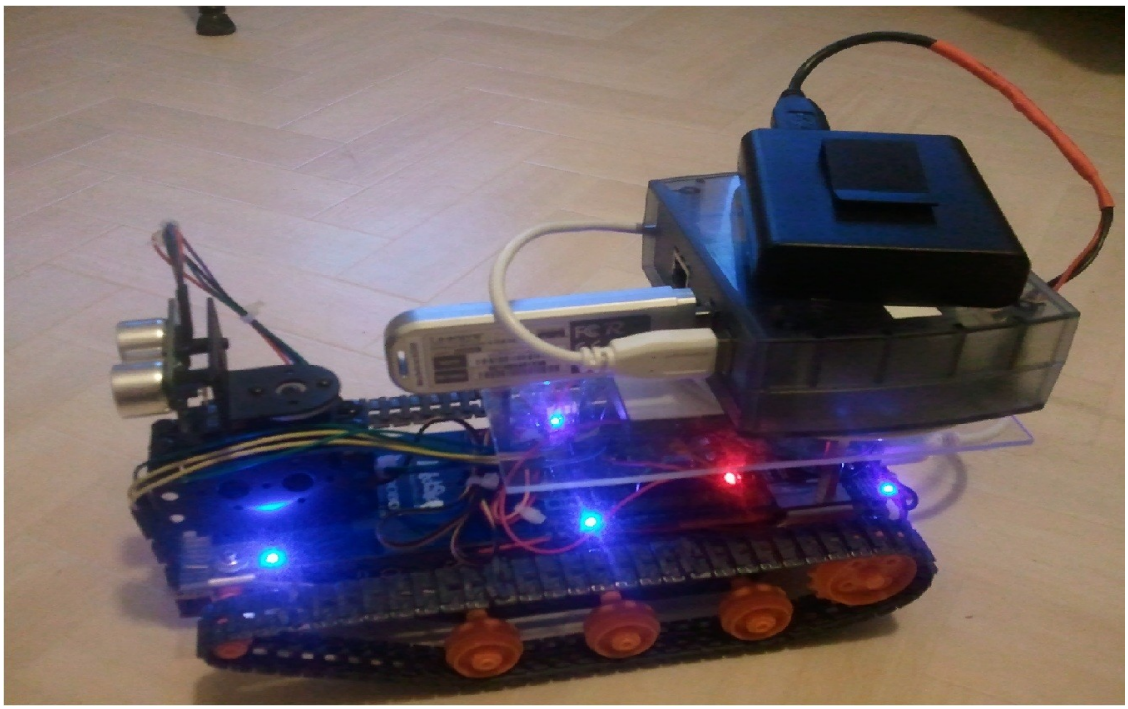
DUJARDIN Maxime IMA4
CONFLAND Benoît

PROJET IMA 4 :
REPRISE DES ROBOTS MOBILES



2011/2012

Introduction

Parmi les nombreux sujets proposés, nous avons réussi à obtenir celui sur les robots mobiles. C'était notre deuxième voeu. C'est en fait une reprise de projet, car il avait déjà été lancé l'année dernière. Nous nous sommes donc confrontés à un genre de problème nouveau, que nous avons essayé de résoudre. Nous avions pour objectifs de modifier les deux robots que nos prédécésseurs avaient créé, cependant nous avons concentré nos efforts sur un seul des deux robots car c'était une reprise de projet sans documentation. En effet, nous avions décidé de ne travailler dans un premier temps que sur un seul des robots. Mais nous avons tellement perdu de temps pour réussir à faire fonctionner le robot (nous verrons cela dans le développement), que finalement nous avons décidé de ne pas nous lancer dans le deuxième robot, car nous aurions perdu autant de temps à comprendre son fonctionnement et nous n'aurions pas pu l'améliorer. Tout au long de notre projet, nous avons essayé de refaire fonctionner le robot, mais nous avons eu pas mal de soucis. Nous verrons donc comment nous avons progressé dans ce projet, non sans difficultés.

Sommaire

I. Matériel et objectifs

I.1 Objectifs

Nous avions pour objectifs d'améliorer deux robots mobiles, nés du projet de l'année dernière du même nom. Nous allons détailler les objectifs demandés par châssis :

Châssis 1 : amélioration du capteur de couleur pour que la luminosité n'influence plus trop les valeurs mesurées.

Châssis 2 : suppression de l'arduino qui commande le capteur de couleurs, et faire communiquer ce dernier directement avec la foxboard.

Dans tout ce rapport, nous ne nous occuperons que du châssis 1, comme expliqué dans l'introduction.

I.2 Matériel

Châssis :

Nous travaillons donc sur le robot Dfrobots Rover V1.5, c'est un châssis originalement équipé d'un arduino ATMega328P ainsi que de deux moteurs Tamiya L293. Nous l'avons trouvé au début de notre projet avec les équipements suivants :

- Servomoteur HS-422
- Sonar SRF05
- capteur de couleur ADJD-S311-CR999
- bus I2C

Foxboard :

Pour la communication, le robot mobile est équipé d'une foxboard qui est reliée à l'arduino du châssis par liaison série. Une cléf wifi est pluggée dans la foaxboard pour permettre la communication avec le réseau de l'école.

Batterie :

Un boitier constitué de 4 piles AA permet l'alimentation par USB de la foxboard. L'arduino étant alimenté par 4 piles AA situées sous le châssis.

II. Fonctionnement du robot

Dans notre reprise de projet, la partie la plus importante avant de vouloir vraiment commencer à coder et améliorer le robot, c'est de comprendre son fonctionnement global. C'est-à-dire aussi bien physiquement (électronique) que virtuellement (code). Nous avons donc étudié l'électronique du châssis en fonction des équipements montés dessus.

Sonar SRF05 :

Le capteur à ultrasons SRF05 Ultrasonic Ranger est l'évolution de son prédécesseur SRF04. Parmi les améliorations notables, on peut noter que sa portée passe de 3 à 4 m et qu'un nouveau mode opératoire permet d'utiliser une seule pin pour le trigger et l'écho. Il est constitué de 5 Pins :

- 5V input
- Echo output
- Trigger Input
- Mode (pas de connexion)
- 0V Ground

Sur notre architecture, ces pins sont connectées sur l'arduino de la manière suivante :

5V ----------> 5V
Echo----------> Pin 2
Trigger----------> Pin 3
0V----------> GND

On peut se référer à l'annexe 1 pour un schéma électrique.

Capteur de couleur ADJD S371 :

Ce capteur de couleurs est l'évolution du ADJD S311 de Sparkfun. Il est compatible avec l'arduino, c'est pour cela que nous l'avons choisi. Il est équipé d'une LED et d'un capteur de lumière (luxmètre) qui vont nous permettre de contrôler la luminosité ambiante pour ne pas trop influencer les valeurs mesurées. Il est constitué de 7 pins dont 5 suffisent à le faire fonctionner :

- LED
- SDA
- SCL
- GND
- 3.3V

Sur notre architecture, ces pins sont connectées sur l'arduino de la manière suivante :

LED----------> Pin 11 (PWM)

SDA----------> Pin 4
SCL----------> Pin 5
GND----------> GND
3.3V----------> 3.3V

Confère annexe 2 pour voir le schéma électrique.

Servomoteur HS-422 :

Ce servomoteur est fixé à l'avant de notre châssis, la partie rotative étant dirigée vers le haut. On y a fixé le sonar SRF05. Il est constitué de 3 Pins :

- 5V input
- Commande (PWM)
- 0V Ground

Sur notre architecture, ces pins sont connectées sur l'arduino de la manière suivante :

5V----------> 5V
Commande--------> Pin 9
0V----------> GND

Il est important de prendre en compte l'architecture mécanique du robot, car le code qu'il y a sur le micro-contrôleur est directement lié à l'électronique. Nous avons fait un tableau récapitulatif des différentes Pins sur lesquelles sont connectés les équipements :

5V	3.3V	GND	Pin 2	Pin 3	Pin 4	Pin 5	Pin 9	Pin 11
5V HS422		0V HS422					Cmd HS422	
5V SRF05		0V SRF05	Echo SRF05	Trigger SRF05				
	3.3V S371	0V S371			SDA S371	SCL S371		LED S371

Principe de fonctionnement pour la communication :

L'objectif de nos prédécesseurs était de commander le robot via une interface Web. Ainsi, le robot pouvait être commandé à distance. Nous expliquerons ici brièvement le principe de communication entre le robot et l'interface Web, car cette partie a été réalisée précédemment et est opérationnelle. Nous nous devons cependant de l'expliquer pour que le projet global soit compréhensible.

Ainsi, comme nous l'avons dis précédemment, le robot est équipé d'une foxboard, qui communique avec le contrôleur aduino par liaison série (à l'aide d'un

câble usb). La foxboard est connectée grâce à une clef wifi au réseau de l'école. L'interface Web comporte une page php qui envoie les ordres à la foxboard en passant par le réseau. Voici le lien qui permet de configurer la foxboard à partir du web : http://172.26.167.120/lego_modified/configure.html

Nous avons donc fini de présenter le fonctionnement du robot, nous allons maintenant décrire le déroulement de notre projet, avec ses aléas.

III. Avancement du projet

Le commencement :

Nous avons pris possession des deux robots mobiles que l'on nous a prêté pour ce projet. Nous avons été tout de suite attiré par le châssis 1 (sûrement pour son allure soignée et un peu militaire). Nous avons donc décidé de commencer à travailler sur ce robot, puis dans un second temps sur le châssis 2.

La première chose que nous avons réalisée dans ce projet a été la fabrication de deux câbles d'alimentation USB destinés aux foxboards. Nous avons passé pas mal de temps, mais avons finalement fourni un résultat correct. Nous avons également récupéré une image de la foxboard de l'année passée pour pouvoir commencer à remettre le robot en route (il s'avérera par la suite que l'image du châssis 1 n'était pas la dernière version).

Notre objectif dans un premier temps était de réussir à faire bouger le robot. Mais nous n'avions aucune idée de comment y parvenir. Nous avons regardé la page de projet de l'année dernière, nous avons compris le fonctionnement globale du système mais nous ne savions pas vraiment comment le remettre en route.

Nous avons ensuite eu un problème assez conséquent qui nous a fait perdre beaucoup de temps : une tierce personne a arraché le port USB situé à l'arrière du robot par inadvertance. Nous avons donc été demander de l'aide auprès de M.FLAMAND qui nous a sauvé la vie (une soudure extrêmement délicate). Ainsi, nous avons fait des tests avec les codes d'exemple du robot pour le faire avancer. Nous avons eu des soucis avec la broche 3 du port USB, il y avait un faux contact. Après s'en être rendu compte, et l'avoir légèrement « grattée » le robot était commandable par liaison série à partir de l'ordinateur.

Après avoir discuté avec les anciens responsables du projet, nous nous sommes rendu compte que notre image de la foxboard n'était pas la bonne. Nous avons donc fini par récupérer la version finale et avons commencé à regarder les différents codes présents pour essayer de comprendre comment le système fonctionnait et quels

fichiers nous devions modifier pour améliorer le capteur de couleur. Avec la nouvelle image nous avons de nouveau du reconnecter la foxboard au réseau de l'école.

Difficulté de compréhension du système :

Pendant très longtemps nous avons eu du mal à trouver le fichier principal qu'il fallait modifier. En effet nous avons pris un fichier que nous modifions mais dont certaines parties de codes ne correspondaient pas avec le comportement du robot quand nous le commandions sur l'interface Web. Notre compréhension était donc limitée... En même temps, il y avait 3 fichiers qui portaient le même nom dans des répertoires différents. Il s'est avéré que le bon fichier était un fichier dont le nom comportait une tilde. Nous l'avons donc confondu avec un fichier temporaire, non important. Nous l'avons renommé rfrobot.c et avons supprimé les fichiers inutiles.

Lorsque nous avons rétabli la communication arduino/foxboard/web, nous avons ensuite testé la commande manuelle. Car nous n'avions pas le code de l'arduino, nous l'avons obtenu que plus tard dans l'avancement. Nous avons donc réussi à faire fonctionner le robot en écrivant un code simple pour le mode manuel mais les variables ne correspondaient pas avec les ordres envoyés car nous ne regardions pas le bon fichier sur la foxboard. Pendant un bon moment nous avons été dans le flou à ce niveau.

Capteurs de couleur ADJD-S311 et ADJD-S371 :

Dans un deuxième temps nous avons donc regardé le capteur de couleur ADJD S311 (dont nous avons eu du mal à trouver le nom). Nous avons décidé dans un premier temps que nous le testerions séparément de tout le montage du robot. Ainsi, lors des premiers tests avec une LED 256 couleurs et un code exemple récupéré sur internet, nous nous sommes rendus compte que ça ne marchait pas (rien de très compliqué jusque là). Il fallait donc trouver la source d'erreur, pour cela nous avons faits pleins de tests différents dans l'optique de trouver la défaillance : nous avons changé d'arduino, de câbles, et de capteur ! En utilisant le capteur de couleur monté sur le châssis 2, capteur identique que sur le châssis 1, nous avons pu en déduire que notre capteur était mort. Ainsi, nous avons pu commander un autre capteur de couleur pour ce robot. En attendant, nous utilisions le deuxième pour faire nos tests sur un autre arduino. Nous avons donc réussi à obtenir des couleurs différentes sur la LED 256 couleurs, nous avons ensuite cherché à comprendre les différentes variables que renvoyait le capteur.

Le deuxième problème que nous avons rencontré avec ce capteur, c'est qu'il ne fonctionnait toujours pas sur le châssis 1. Cela nous a vraiment fait douter sur les conclusions que nous avions prises précédemment. Nous avons donc testé à nouveau le capteur « grillé » pour vérifier s'il était bien mort (c'était bien le cas), et nous avons également testé le capteur opérationnel pour vérifier si nous ne l'avions pas grillé à son tour. Fort heureusement, ce n'était pas le cas. Nous avons donc cherché du côté du câblage, et avons vérifié le bus I2C broche par broche. Nous avons ainsi pu

constater qu'une des broches était défaillante, nous pouvions cependant l'utiliser si l'on évitait cette broche. Nous avons par la suite testé le capteur avec le bus I2C bien branché, et le capteur a fonctionné.

Ainsi, nous avions à cet instant un matériel opérationnel, nous pouvions donc commencer à coder pour améliorer la capacité du robot à suivre une ligne. Il a donc fallu comprendre ce que nos prédécesseurs avaient écrit dans leur code arduino. Le code pour le capteur est assez important, et nous avons eu du mal à bien saisir le sens de chaque fonction. Nous avons cependant bien remarqué l'endroit où nous devions modifier le code, il fallait changer les conditions qui permettaient de renvoyer la couleur dans la fonction. Nous avons donc étudier ce que nos prédécesseurs avaient réalisé, voici les conditions que nous devions modifier :

```
if(redValue>10 && greenValue<=15 && blueValue>=0 && blueValue<10) return RED;
if(redValue<=10 && greenValue<=15 && blueValue<10) return BLACK;
if(ccValue>=40 && ccValue<=65 && blueValue>=10 && blueValue<=18) return GREEN;
if(ccValue>=70 && redValue >=30 && greenValue>=30) return WHITE;
```

On voit donc ici que les conditions sont fixées sur les valeurs que retourne le capteur (redValue, greenValue, blueValue, ccValue). C'est donc un code judicieux pour un environnement où la luminosité reste relativement stable, en effet, il y avait un carton attaché autour du capteur pour limiter la luminosité ambiante. Notre objectif était d'enlever ce morceau de carton, et donc de réussir à coder de manière à ce que la luminosité n'influe plus beaucoup sur les mesures.

1. Première approche

Dans un premier temps, nous avons regardé les valeurs que retournait le capteur pour pouvoir avoir une idée de l'ordre de grandeur des conditions. Or nous avons remarqué qu'il y avait une variable dont nous ignorions le rôle : ccValue. Nous avons donc observé ses variations, nous en avons déduit qu'il s'agissait d'une variable qui représentait l'éclairage ambiant.

Nous avons donc dans notre approche du problème essayé de coder dans la même pensée que nos prédécesseurs en fixant des valeurs numériques pour chaque couleur. Nous nous sommes vite rendu compte que c'était impossible, car nous n'arrivions pas à garder des valeurs constantes en fonction de la luminosité : dès que quelqu'un passait sa main au dessus de la lumière, on perdait toutes nos couleurs (sauf le blanc).

Nous n'avons donc pas suffisamment creusé la variable ccValue. Nous avons décidé d'abandonner cette manière de coder les conditions car on ne s'en sortirait pas.

2. Deuxième approche

Par la suite nous avons posé les choses à plat. Nous voulions compenser la luminosité ambiante, mais comment ? Nous avons remarqué que le capteur de couleur était équipé d'une LED, qui était branché sur 5V. Cette LED relativement puissance devait selon nous éclairer de manière constante le sol pour avoir des valeurs de mesures du même ordre de grandeur. Cependant, ça ne marchait pas ainsi.

Nous nous sommes attardés sur deux points du capteur : sa LED et son ccValue. Y avait il moyen d'éclairer le sol en fonction de l'éclairement mesuré ? Nous avons donc branché la LED sur une pin PWM pour essayer de faire varier l'intensité de la LED. Nous avons effectivement réussi à faire varier l'intensité de la LED. Ensuite nous avons essayé de trouver un lien entre la valeur ccValue et la valeur à mettre dans le analogWrite de la LED. Nous avons tout d'abord écrit la valeur de ccValue sur la pin PWM, c'était une erreur car l'éclairement était trop important. Nous nous sommes vite rendu compte qu'il fallait écrire l'opposé de ccValue ! Ainsi, on compenserait l'ombre par une plus forte luminosité de la LED. Ainsi, il faut initialiser le capteur dans de bonnes conditions d'éclairement, la LED ne pouvant compenser que l'obscurité.

La théorie nous paraissait bien, cependant nous avons eu des problèmes importants lors de sa réalisation : en effet, la LED varie très bien dès que l'on fait des tests à coté sur le même arduino mais sans le code final. Cependant, dès que l'on mettait notre modification dans la fonction : le robot plantait ! Nous avons perdu énormément de temps sur ce problème, nous avons fait beaucoup de tests pour essayer de trouver d'où pouvait venir le problème (tests sur des pins PWM différentes, tests sur différents arduinos, tests séparés et codes complet...). Nous avons fini par comprendre pourquoi ça ne marchait pas ,mais nous n'avons pas réussi à corriger cette erreur. Nous avons même cherché du côté du analogWrite, nous avons demandé de l'aide aux encadrants qui nous ont donné des pistes. Mais nous n'avons pas réussi à résoudre le problème.

A ce stade du projet, nous avions envisagé de réécrire la fonction qui plantait, mais nous avons renoncé car nous avons reçu à ce moment là le nouveau capteur de couleur ADJD S371 CR999, qui est l'évolution de l'ADJD S311. Ainsi, nous avons remplacé le vieux capteur par le nouveau, et sans changer le code (mais le câblage électronique), nous avons réussi à faire fonctionner le robot, il ne plantait plus ! Nous en avons donc conclu que le capteur ADJD S311 n'était pas adéquat pour ce code.

3. Codage des conditions

Ainsi, nous avions notre capteur opérationnel qui nous renvoyait des valeurs correctes en fonction de la luminosité. Cependant la LED variait de manière trop importante : elle s'allumait et s’éteignait quasiment toutes les secondes. Nous avons donc introduit une nouvelle fonction qui prend la valeur moyenne de l'éclairement (ccValue) pour essayer de faire baisser la variation trop importante. Ci dessous les différentes fonctions que nous avons créées dans cette optique :

```
void init(int tab[TAILLE])
```

```
{
   int i;
   for (i = 0; i < TAILLE; i++)
        tab[i] = 0;
}
```

La fonction init comme son nom l'indique permet d'initialiser notre tableau à 0.

```
void gettab(int tab[TAILLE], int *moy)
{
   *moy = 0;

   for (i = 0; i < TAILLE; i++) {
        newcolor = getColorValue(&ccvalue);
        tab[i] = 256 - ccvalue;
        *moy = *moy + tab[i];
        Serial.println(newcolor);
        //Serial.print("Ccvalue");Serial.print(i);Serial.print(" : ");
        //Serial.println(tab[i]);
   }
   *moy = *moy / TAILLE;
}
```

La fonction gettab permet de retourner une moyenne de l'éclairement ccValue. TAILLE étant déclaré en constante au début du code.

```
void analog(int ccvalue)
{
   init(Ccvalue);
   gettab(Ccvalue, &moy);
   analogWrite(ledpin, abs(moy));
}
```

La fonction analog permet d'initialiser, de trouver la moyenne de l'éclairement et de l'écrire sur la pin PWM ledpin (qui est en fait la 11).

Nous avons essayé plusieurs tailles pour la moyenne, car nous nous sommes rendu compte que si l'on mettait un grand nombre pour TAILLE, la moyenne était forcément meilleure, et donc la LED variait moins brutalement. Cependant, le calcul était trop long, et donc gênait le suivi de ligne. Nous avons donc essayé deux stratégies : une première consistait à refaire souvent une mesure de la luminosité mais sur une moyenne courte (TAILLE=4) et l'autre était de faire la mesure moins souvent mais avec une plus grande moyenne. Nous avons finalement opté pour la deuxième option après avoir observé le comportement du robot.

4. Conditions pour les couleurs

Nous avons donc des valeurs cohérentes à ce stade, nous allions donc commencer à chercher les conditions pour les couleurs. Nous avons décidé de changer la méthode utilisée par nos prédécesseurs : les valeurs numériques ne suffisent pas car elles varient trop même malgré la compensation de la luminosité.

Nous avons remarqué suite à des tests que les écarts entre les valeurs rouge/bleu/vert étaient relativement constants. Nous avons donc décidé de travailler avec des rapports : en effet, les valeurs changent en fonction de la luminosité mais les rapports restent proches ! Ainsi nous avons introduit les variables suivantes :

```
float Rrg, Rgb, Rrb;
```

où chaque rapport est défini tel quel : Rrg = Red/Green ; Rgb = Green/Blue ; Rrb = Red/Blue.

```
Rrg = (float) redValue / greenValue;
Rgb = (float) greenValue / blueValue;
Rrb = (float) redValue / blueValue;
```

Suite à de nombreuses mesures, nous avons fini par obtenir les conditions suivantes pour les couleurs :

```
if (Rrg >= 1.4 && (Rgb >= 1.4 && Rgb <= 2.8) && (Rrb >= 2.5))
    return RED;
if ((Rrg >= 0.8 && Rrg <= 1.3) && (Rgb >= 1.3 && Rgb <= 2.2)
    && (Rrb >= 1.4 && Rrb <= 1.9) && redValue <= 140
    && greenValue <= 130 && blueValue <= 80)
    return GREEN;
if ((Rrg >= 0.8 && Rrg <= 1.4) && (Rgb >= 1.0 && Rgb <= 2.2)
    && (Rrb >= 1.2 && Rrb <= 2.5) && redValue >= 190
    && greenValue >= 100 && blueValue >= 80)
    return WHITE;
if ((Rrg >= 1.0 && Rrg <= 1.6) && (Rgb >= 1.3 && Rgb <= 2.6)
    && (Rrb >= 1.9 && Rrb <= 2.9) && redValue <= 150
    && greenValue <= 130 && blueValue <= 100)
    return BLACK;
return -1;
```

Ainsi, nous travaillons avec 4 couleurs. On remarquera que l'on s'est également servi des valeurs numériques pour affiner la détection des couleurs. La prochaine étape est de modifier la fonction du suivi de ligne pour pouvoir travailler avec nos couleurs.

5. Modifications du suivi de ligne

Nous avons donc modifié la fonction suivisens1. Nous avons principalement modifier les conditions qui permettent d'actionner les moteurs, les anciennes ne marchant pas. Nous avons simplifié ces conditions, et le robot peut suivre une ligne, cependant, si le virage est trop serré alors le robot sort de la trajectoire. Voici la fonction en question :

```
void suiviSens1()
{
    analog(ccvalue);
    while (incomingByte == 'c') {

    newcolor = getColorValue(&ccvalue);
    serie(&incomingByte);
    distanceToObject = acquisition(2);    /* Verifie la
presence d'un obstacle */
    if (distanceToObject > 30) {
        Serial.println(newcolor);
        Serial.println("HORS SWICH");
            j=j+1;
            //Serial.println("j : ");
            //Serial.println(j);
            if (j==12) {
        analog(ccvalue); j=0; i=1;}
            if (i==1 && j==11) {i=0;
            robot_control(MOVE_BK, SPEED_COM - 10,
SPEED_COM - 10);
            delay(500);
            moteurOff();
            delay(100);
            }
        newcolor = getColorValue(&ccvalue);
        switch (newcolor) {
        case RED:
        while (1) {
            Serial.print("RED : ");
            newcolor = getColorValue(&ccvalue);
            Serial.println(newcolor);
            //analog(ccvalue);
            serie(&incomingByte);
            Serial.print("rouge");
            if (newcolor != RED || incomingByte=='q') {
```

```
            moteurOff();
            break;
            }
            robot_control(MOVE_FW, SPEED_COM-50,
SPEED_COM-50);
        }
        case GREEN:
        while (1) {
            newcolor = getColorValue(&ccvalue);
            Serial.print("GREEN : ");
            Serial.println(newcolor);
            serie(&incomingByte);
            Serial.print("vert");
            if (newcolor != GREEN || incomingByte=='q')
            break;
            turn(MODE_COMMANDE, TURN_LEFT, 92, 93,
TURN_SPEED,
             TURN_SPEED);
            delay(150);
            moteurOff();
            delay(100);
        }
        break;
        case BLACK:
        while (1) {
            Serial.print("BLACK : ");
            newcolor = getColorValue(&ccvalue);
            Serial.println(newcolor);
            serie(&incomingByte);
            delay(10);
            Serial.print("noir");
            if (newcolor != BLACK || incomingByte=='q')
            break;
            turn(MODE_COMMANDE, TURN_RIGHT, 94, 93,
TURN_SPEED, TURN_SPEED);
            delay(150);
            moteurOff();
            delay(100);
        }
        break;
        case WHITE:
        while (1) {
            Serial.print("WHITE : ");
            newcolor = getColorValue(&ccvalue);
            Serial.println(newcolor);
```

```
            serie(&incomingByte);
            distanceToObject = acquisition(2);
            if (newcolor != WHITE || incomingByte=='q')
{
            moteurOff();
            break;
            }
            delay(100);
            robot_control(MOVE_BK, SPEED_COM - 50,
SPEED_COM - 50);
            Serial.print("Track lost");
         }
         default:
         break;
         }
     } else {
         moteurOff();

         Serial.println("STOP!!!");
     }
     }
     delay(100);
}
```

Dans cette fonction, le principe de fonctionnement est assez simple : on commence par faire un échantillonnage de l'éclairement. Puis on regarde sur quelle couleur nous sommes :

- Si nous sommes sur une couleur connue, alors on entre dans la boucle et on fait marcher les moteurs. À partir du moment où la couleur varie, le robot s'arrête et on retourne au switch pour changer de couleur. Si jamais la couleur n'est pas reconnue, alors on sort du switch.

- Si la couleur n'est pas reconnue, alors le robot ne bouge pas et on incrémente le compteur j de 1. Au bout de 12 itérations, on refait une mesure de l'éclairement car on suppose que ça peut être la cause du problème. Si jamais nous sommes toujours bloqué au bout de 11 itérations de j, on déplace volontairement le robot en arrière. Ainsi, on peut débloquer le robot : il ne sera jamais bloqué définitivement.

On modifie de la même manière la fonction suiviSens2. Nous n'allons pas la détailler.

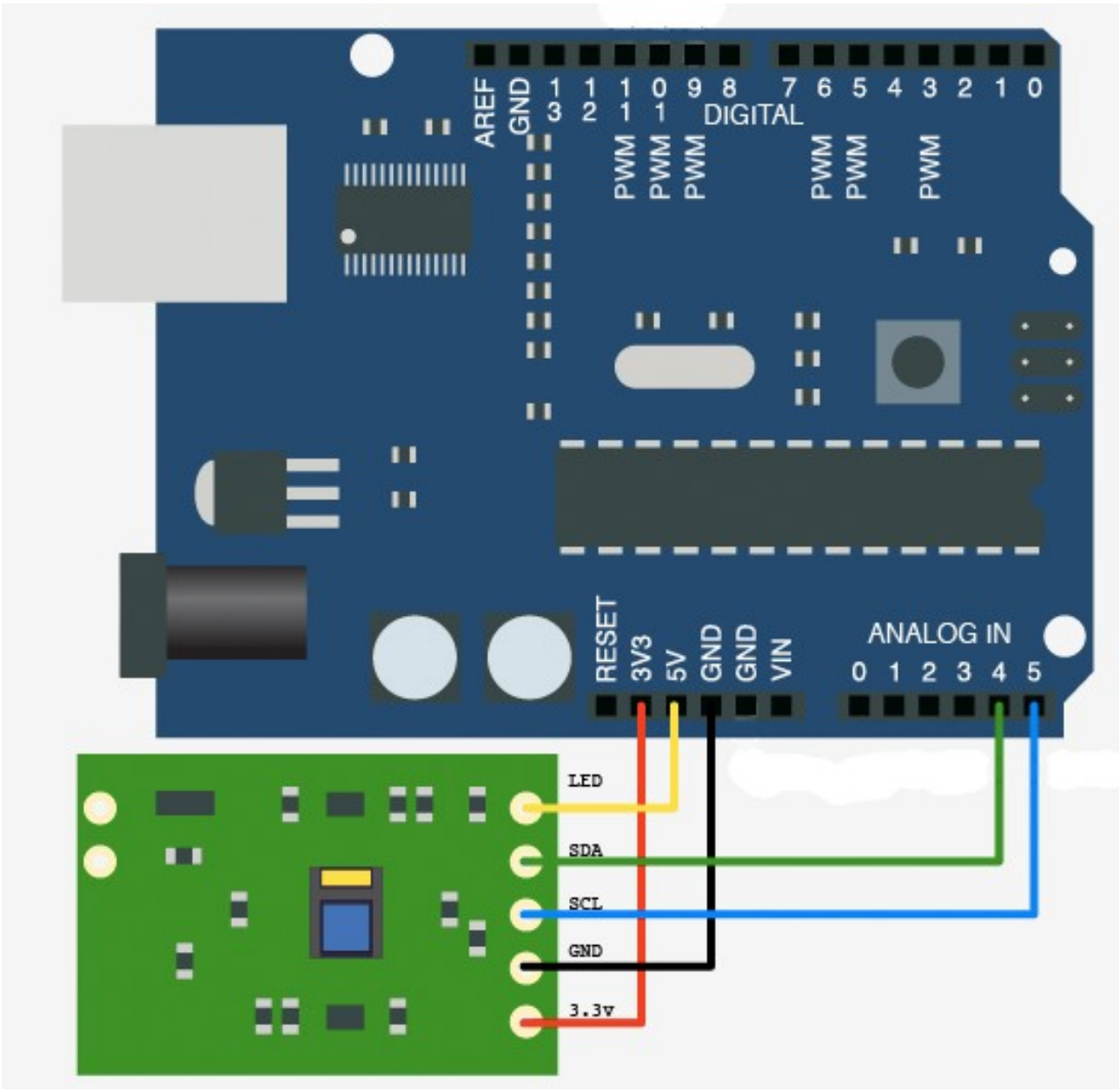
Remarques : Notre fonction suivisens1 n'est pas optimisée, c'est à dire que notre robot arrive à suivre une ligne... mais à quel prix ! Il ne faut pas être pressé. L'avantage est qu'il ne se perd pas et ne se bloque pas. Nous avons déjà amélioré la vitesse de réaction comparé au tout début de notre projet, quand nous avons modifié le code. Cependant, il est évident que nous pouvons certainement encore l'améliorer. Mais nous n'avons plus de temps pour cela.

Conclusion :

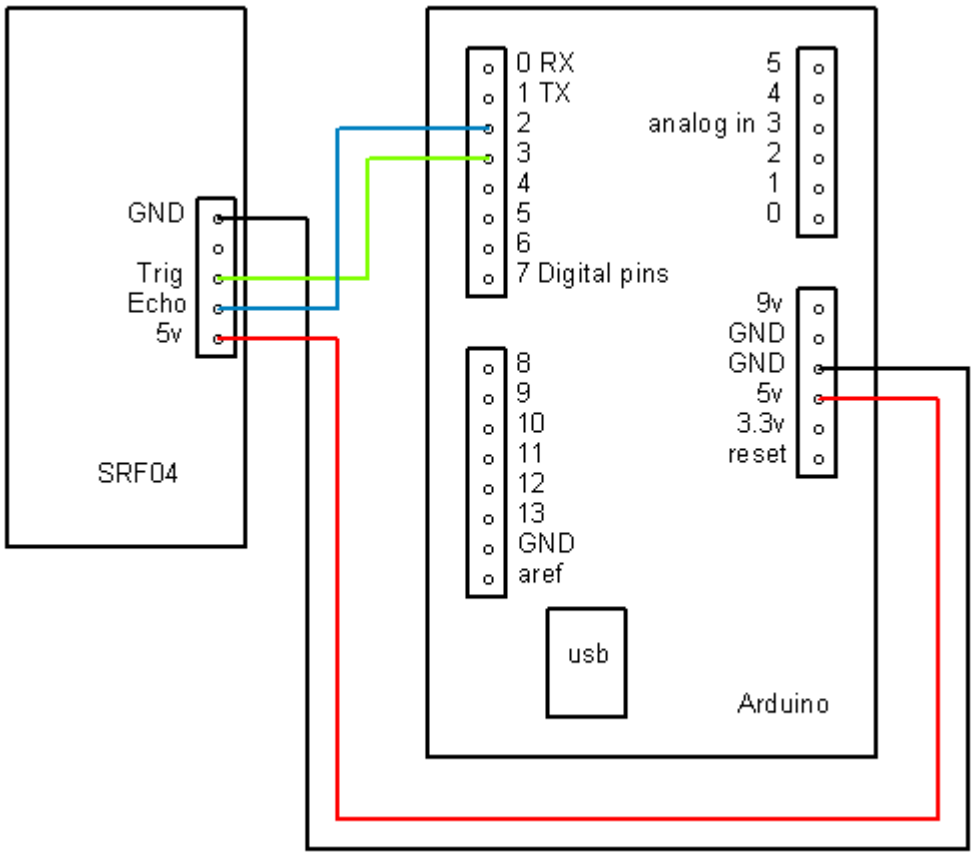
Pour conclure ce rapport, nous nous devons d'aborder certains points qui nous semble importants. Tout d'abord nous devons aborder le fait que ce projet est une reprise. En effet, il a été commencé l'année dernière par nos camarades d'IMA 4. Nous avions donc une base sur laquelle nous pouvions nous appuyer. Le fait que ce précédent binôme n'ai pas laissé de traces ni écrites ni informatiques à propos du fonctionnement, de l'organisation de leur projet nous a énormément pénalisé. En effet, nous avons passé une majeur partie des 40 heure allouées au projet à comprendre les systèmes, leurs codes, leurs manière de fonctionner. Nous les avons contacté au début du projet pour qu'ils puissent nous aider ou nous renseigner de vives voix : ils ne nous ont pas appris grand choses à part nous rappeler que le code était suffisamment commenté. Certes ces commentaires ont été utiles mais ils n'expliquaient en rien le fonctionnement des robots. Nous voulons dire par là, le protocole à suivre pour mettre en marche les robots dans les meilleurs conditions. Ceci a été tellement handicapant que nous nous sommes concentré uniquement sur un seul des deux châssis : le DFRobots Rover (décris tout au long de ce compte rendu). Nous avons laissé l'autre de côté volontairement, par soucis d'efficacité. Nous maîtrisions le robot qu'au bout d'une vingtaine d'heures. Une fois ceci réglé ainsi que les réparations du robot, nous avons pu nous concentrer sur le code arduino à proprement parler. Mais du coup, le temps faisant défaut (même en allant un maximum de fois en salle de projet), nous n'avons pas pu approfondir autant que nous le voulions la gestion du capteur de couleurs du robot.

Ensuite, afin que le binôme suivant qui s'occuperait du robot puisse travailler directement sur le code arduino du châssis DFRobots, nous avons décidé de rendre notre travail le plus claire, le plus commenté et le plus compréhensible possible. Nous avons donc plus commenté le code arduino et nous avons écrit un ReadMe, que nous avons mis dans la foxboard. Sont également présents la plupart des documents relatifs au différents composants du robots (sonar, capteur de couleurs...).

Nous avons tiré de ce projet quelques enseignements : la gestion d'un projet, même s'il n'est pas d'une taille phénoménale reste assez difficile. En effet nous étions complètement autonomes (tout en ayant la possibilité de demander de l'aide bien entendu) et au fur et à mesure que le projet avance, de nouvelles choses apparaissent (que ce soit des problèmes, des changement de directions dans les objectifs fixés...). Également, nous avons compris qu'une reprise de projet pouvait provoquer pas mal de problème si la passation se faisait sans explications ni conseils. Au demeurant, ce projet fût très intéressant car il était autant fait pour la filière SA que SC et car il touchait pas mal de domaines (technique, informatique...). Nous regrettons de ne pas avoir eu plus de temps pour continuer son avancée.



Annexe 2 : branchement capteur de couleur



Annexe 1 : branchement sonar (SRF05)

Annexe 3 : Code arduino

```
#include <Servo.h>
/************* CAPTEUR COULEUR VAR **********/
//Include the I2C Arduino library
#include <Wire.h>
//7 bit I2C address of this sensor
#define I2C_ADDRESS 0x74
#define  REG_CAP_RED            0x06
#define  REG_CAP_GREEN    0x07
#define  REG_CAP_BLUE     0x08
#define  REG_CAP_CLEAR    0x09
#define  REG_INT_RED_LO 0x0A
#define  REG_INT_RED_HI 0x0B
#define  REG_INT_GREEN_LO 0x0C
#define  REG_INT_GREEN_HI 0x0D
#define  REG_INT_BLUE_LO 0x0E
#define  REG_INT_BLUE_HI 0x0F
#define  REG_INT_CLEAR_LO 0x10
#define  REG_INT_CLEAR_HI 0x11
#define  REG_DATA_RED_LO 0x40
#define  REG_DATA_RED_HI 0x41
#define  REG_DATA_GREEN_LO 0x42
#define  REG_DATA_GREEN_HI 0x43
#define  REG_DATA_BLUE_LO 0x44
#define  REG_DATA_BLUE_HI 0x45
#define  REG_DATA_CLEAR_LO 0x46
#define  REG_DATA_CLEAR_HI 0x47
#define  RED      0x00
#define  GREEN    0x01
#define  BLACK    0x02
#define  WHITE    0x03
#define MAX_ACQUI 4
#define TAILLE 10
//Configure gain here
//Higher numbers = less sencitive
// 0x00 through 0x0f
int redGain = 0x0F;
int greenGain = 0x0F;
int blueGain = 0x0F;
int clearGain = 0x0F;
struct colorSense {
    int cc;
    int red;
    int green;
    int blue;
};
/*************** LED ******************/
/* LED leads connected to PWM pins
const int RED_LED_PIN = 5;
const int GREEN_LED_PIN = 6;
const int BLUE_LED_PIN = 10;*/
// Used to store the current intensity level
int redIntensity = 255;
int greenIntensity = 255;
int blueIntensity = 255;
```

```
/************** MOTORS VAR ***************/
// setup pins and variables for SRF05 sonar device
const int echoPin = 2;          // SRF05 echo pin (digital 2)
const int initPin = 3;          // SRF05 trigger pin (digital 3)
const int E1 = 6;          // M1 Speed Control
const int E2 = 5;          // M2 Speed Control
const int M1 = 8;          // M1 Direction Control
const int M2 = 7;          // M2 Direction Control
#define SPEED_COM 175
#define SPEED_SIM 255
#define SPEED_AUT 255
#define TURN_SPEED 255
#define MOVE_FW 'F'
#define MOVE_BK 'B'
#define TURN_LEFT 'L'
#define TURN_RIGHT 'R'
#define TURN_INV 'I'
#define MODE_COMMANDE 'C'
#define MODE_AUTO 'A'
#define STOP 0x03
unsigned long pulseTime = 0;    // stores the pulse in Micro Seconds
unsigned long distance = 0;     // variable for storing the distance
(cm)
/************** SERVO MOTEUR VAR ******************/
Servo myservo;             // Create servo object to control a servo
int servoPin = 9;          // Broche du servomoteur
int angleToObject = 0;
int flag = -1;
#define SET_TO_MIDDLE 94
/************** CAPTEUR ULTRASON VAR *****************/
int distanceToObject = 0;
/************** SERIAL VAR ***************/
int incomingByte = 0;
int choix = 0;
/
**************************************************************************
***************/
/
**************************************************************************
***************/
/
**************************************************************************
***************/
int ledpin = 11;

/********************************** FONCTION PRINCIPALE
**********************************/
void setup(void)
{
    Serial.begin(9600);
/** Initialisation pour le capteur de couleur **/
    Wire.begin();
    analogWrite(ledpin, 90);
    // sensor gain setting (Avago app note 5330)
    // CAPs are 4bit (higher value will result in lower output)
    set_register(REG_CAP_RED, redGain);
    set_register(REG_CAP_GREEN, greenGain);
    set_register(REG_CAP_BLUE, blueGain);
```

```
    set_register(REG_CAP_CLEAR, clearGain);
    int ledGain = getColorGain();
    set_gain(REG_INT_RED_LO, ledGain);
    set_gain(REG_INT_GREEN_LO, ledGain);
    set_gain(REG_INT_BLUE_LO, ledGain);
/**Initialisation pour teste de l'intensitÃ© de la LED**/
    //int ledpin=11;
    pinMode(ledpin, OUTPUT);
/** Initialisation pour le capteur ultrason **/
    pinMode(initPin, OUTPUT);   // set init pin 3 as output
    pinMode(echoPin, INPUT);    // set echo pin 2 as input
/** Initialisation pour le servo moteur **/
    myservo.attach(servoPin);   //attaches the seevo on pin 9 to the
servo object
}

int newcolor;
int i=0;
int ccvalue;
int Ccvalue[TAILLE];
int moy, val2;
float Rrg, Rgb, Rrb;
int j = 0;
void loop()
{
    serie(&incomingByte);
    switch (incomingByte) {     //switch qui fait le lien entre la
foxboard et l'arduino
    case 'r':              //avance droit
     setAngle(SET_TO_MIDDLE);
     if (acquisition(2) > 40)
         robot_control(MOVE_FW, SPEED_SIM - 10, SPEED_SIM);
     else {
         moteurOff();
         Serial.print("STOP!!!");
     }
     break;
    case 'f':              //recule
     robot_control(MOVE_BK, SPEED_SIM, SPEED_SIM);
     break;
    case 'd':              //tourne gauche
     turn(MODE_COMMANDE, TURN_LEFT, 80, getAngle(), TURN_SPEED,
          TURN_SPEED);
     break;
    case 'g':              //tourne droite
     turn(MODE_COMMANDE, TURN_RIGHT, 100, getAngle(), TURN_SPEED,
          TURN_SPEED);
     break;
    case 'a':              //mode automatique
     automatic_obstacle_mode();
     break;
    case 'c':              //suivi de ligne
     suiviSens1();
     break;
    case 'i':
     suiviSens2();
    case 'q':              //c'est necessaire de le dire ?
     moteurOff();
```

```
     break;
    default:
     break;
    }
}


/*************** CAPTEUR COULEUR ***************/
void init(int tab[TAILLE])
{
    int i;
    for (i = 0; i < TAILLE; i++)
     tab[i] = 0;
}

void gettab(int tab[TAILLE], int *moy)
{
    *moy = 0;

    for (i = 0; i < TAILLE; i++) {
     newcolor = getColorValue(&ccvalue);
     tab[i] = 256 - ccvalue;
     *moy = *moy + tab[i];
        serie(&incomingByte);
        if(incomingByte=='q') break;
     Serial.println(newcolor);
     //Serial.print("Ccvalue");Serial.print(i);Serial.print(" : ");
     //Serial.println(tab[i]);
    }
    *moy = *moy / TAILLE;
}

void analog(int ccvalue)
{
    init(Ccvalue);
    gettab(Ccvalue, &moy);
    analogWrite(ledpin, abs(moy));
}

int getColorValue(int *ccvalue)
{
    //Initialize the sensor
    int clearGain = getClearGain();
    //Serial.println("getcleargain");
    set_gain(REG_INT_CLEAR_LO, clearGain);
    int colorGain = getColorGain();
    //Serial.println("getcolorgain");
    set_gain(REG_INT_RED_LO, colorGain);
    set_gain(REG_INT_GREEN_LO, colorGain);
    set_gain(REG_INT_BLUE_LO, colorGain);
    int ledpin = 11;
    //pinMode(ledpin,OUTPUT);
    //Some vars
    int cc = 0;
    int red = 0;
    int green = 0;
    int blue = 0;
    // Take 4 samples, and add them together.
```

```
    for (int i = 0; i < MAX_ACQUI; i++) {
     performMeasurement();
     //Serial.println("performmeasurement");
     cc += get_readout(REG_DATA_CLEAR_LO); // recupere la valeur de la
luminositÃ© ambiante
     red += get_readout(REG_DATA_RED_LO);
     green += get_readout(REG_DATA_GREEN_LO);
     blue += get_readout(REG_DATA_BLUE_LO);
    }
    //now, divide the totals for each by 4 to get their average.
    cc /= MAX_ACQUI;
    red /= MAX_ACQUI;
    green /= MAX_ACQUI;
    blue /= MAX_ACQUI;
    //Associate a map to the values
    int ccValue = map(cc, 0, 1024, 0, 255);
    *ccvalue = ccValue;
    int redValue = map(red, 0, 1024, 0, 255);
    int greenValue = map(green, 0, 1024, 0, 255);
    int blueValue = map(blue, 0, 1024, 0, 255);
    Rrg = (float) redValue / greenValue;
    Rgb = (float) greenValue / blueValue;
    Rrb = (float) redValue / blueValue;
        /*Serial.print("Rrg: ");
        Serial.println(Rrg);
        Serial.print("Rgb : ");
        Serial.println(Rgb);
        Serial.print("Rrb : ");
        Serial.println(Rrb);
        //delay(1000);*/
    if (Rrg >= 1.4 && (Rgb >= 1.4 && Rgb <= 2.8) && (Rrb >= 2.5))
     return RED;
    if ((Rrg >= 0.8 && Rrg <= 1.3) && (Rgb >= 1.3 && Rgb <= 2.2)
     && (Rrb >= 1.4 && Rrb <= 1.9) && redValue <= 140
     && greenValue <= 130 && blueValue <= 80)
     return GREEN;
    if ((Rrg >= 0.8 && Rrg <= 1.4) && (Rgb >= 1.0 && Rgb <= 2.2)
     && (Rrb >= 1.2 && Rrb <= 2.5) && redValue >= 190
     && greenValue >= 100 && blueValue >= 80)
     return WHITE;
    if ((Rrg >= 1.0 && Rrg <= 1.6) && (Rgb >= 1.3 && Rgb <= 2.6)
     && (Rrb >= 1.9 && Rrb <= 2.9) && redValue <= 150
     && greenValue <= 130 && blueValue <= 100)
     return BLACK;
    return -1;
}

int getClearGain()
{
    int gainFound = 0;
    int upperBox = 4096;
    int lowerBox = 0;
    int half;
    while (!gainFound) {
     half = ((upperBox - lowerBox) / 2) + lowerBox;
     if (half == lowerBox) {    //no further halfing possbile
         break;      //no further halfing possbile
     } else {
```

```
          set_gain(REG_INT_CLEAR_LO, half);
          performMeasurement();
          int halfValue = get_readout(REG_DATA_CLEAR_LO);
          if (halfValue > 1000) {
           upperBox = half;
          } else if (halfValue < 1000) {
           lowerBox = half;
          } else {
           break;          //no further halfing possbile
          }
       }
     }
     return half;
}

int getColorGain()
{
     int gainFound = 0;
     int upperBox = 4096;
     int lowerBox = 0;
     int half;
     while (!gainFound) {
      half = ((upperBox - lowerBox) / 2) + lowerBox;
      if (half == lowerBox) {    //no further halfing possbile
          break;      // gain found
      } else {
          set_gain(REG_INT_RED_LO, half);
          set_gain(REG_INT_GREEN_LO, half);
          set_gain(REG_INT_BLUE_LO, half);
          performMeasurement();
          int halfValue = 0;
          halfValue = max(halfValue, get_readout(REG_DATA_RED_LO));
          halfValue = max(halfValue, get_readout(REG_DATA_GREEN_LO));
          halfValue = max(halfValue, get_readout(REG_DATA_BLUE_LO));
          if (halfValue > 1000) {
           upperBox = half;
          } else if (halfValue < 1000) {
           lowerBox = half;
          } else {
           break;          // gain found
          }
       }
     }
     return half;
}

void performMeasurement()
{
     set_register(0x00, 0x01);   // start sensing
     while (read_register(0x00) != 0) {
      // waiting for a result
     }
}

int get_readout(int readRegister)
{
     return read_register(readRegister) +
       (read_register(readRegister + 1) << 8);
```

```
}

void set_gain(int gainRegister, int gain)
{
    if (gain < 4096) {
     uint8_t hi = gain >> 8;
     uint8_t lo = gain;
     set_register(gainRegister, lo);
     set_register(gainRegister + 1, hi);
    }
}

void set_register(unsigned char r, unsigned char v)
{
    Wire.beginTransmission(I2C_ADDRESS);
    Wire.write(r);
    Wire.write(v);
    Wire.endTransmission();
}

unsigned char read_register(unsigned char r)
{
    unsigned char v;
    Wire.beginTransmission(I2C_ADDRESS);
    Wire.write(r);         // register to read
    Wire.endTransmission();
    Wire.requestFrom(I2C_ADDRESS, 1); // read a byte
    while (!Wire.available()) {
     // waiting
    }
    v = Wire.read();
    return v;
}

/************** ROBOT ********************/
/* Mise en marche avant des moteurs*/
void robot_control(const char mvt, int spd_motor1, int spd_motor2)
{
    switch (mvt) {
    case 'F':
     analogWrite(E1, spd_motor1);
     digitalWrite(M1, LOW);
     analogWrite(E2, spd_motor2);
     digitalWrite(7, LOW);
     break;
    case 'B':
     analogWrite(6, spd_motor1);
     digitalWrite(8, HIGH);
     analogWrite(5, spd_motor2);
     digitalWrite(7, HIGH);
     break;
    default:
     break;
    }
}

void turn(const char mode, const char mvt, int degree, int origin,
       int spd_motor1, int spd_motor2)
```

```
{

    switch (mode) {
    case 'A':
     switch (mvt) {
     case 'L':
         for (origin; origin > degree; origin -= 1) {
          analogWrite(E1, spd_motor1);
          digitalWrite(M1, HIGH);
          analogWrite(E2, spd_motor2);
          digitalWrite(M2, LOW);
          delay(10);
         }
         break;
     case 'R':
         for (origin; origin < degree; origin += 1) {
          analogWrite(E1, spd_motor1);
          digitalWrite(M1, LOW);
          analogWrite(E2, spd_motor2);
          digitalWrite(M2, HIGH);
          delay(10);
         }
         break;
     case 'I':
         for (origin; origin < 270; origin += 1) {
          analogWrite(E1, spd_motor1);
          digitalWrite(M1, LOW);
          analogWrite(E2, spd_motor2);
          digitalWrite(M2, HIGH);
          delay(10);
         }
         break;
     default:
         break;
     }
    case 'C':
     switch (mvt) {
     case 'L':
         analogWrite(E1, spd_motor1);
         digitalWrite(M1, HIGH);
         analogWrite(E2, spd_motor2);
         digitalWrite(M2, LOW);
         delay(10);
         break;
     case 'R':
         analogWrite(E1, spd_motor1);
         digitalWrite(M1, LOW);
         analogWrite(E2, spd_motor2);
         digitalWrite(M2, HIGH);
         delay(10);
         break;
     default:
         break;
     }
    }
}

 /* Arret des moteurs */
```

```
int moteurOff()
{
    analogWrite(E1, STOP);
    analogWrite(E2, STOP);
}

/************* CAPTEUR ULTRASON *****************/
int acquisition(int nb)
{
    int distance = 0;
    for (int i = 0; i < nb; i += 1) {
     digitalWrite(initPin, HIGH);    // send 10 microsecond pulse
     delayMicroseconds(10);     // wait 10 microseconds before turning
off
     digitalWrite(initPin, LOW);     // stop sending the pulse
     pulseTime = pulseIn(echoPin, HIGH);   // Look for a return pulse, it
should be
//       high as the pulse goes low-high-low
     distance = distance + (pulseTime / 58);
     delay(100);
    }
    return distance / nb;
}

/************* SERVO MOTEUR *********************/
int setAngle(int angle)
{
    myservo.write(angle);
    delay(100);
    return 0;
}

int getAngle()
{
    return myservo.read();
    return 0;
}

 /* Scanner a gauche */
int scanLeft(int origin)
{
    int angle = 0;
    int distance = 0;
    angle = origin - 50;
    setAngle(angle);
    distance = acquisition(5);
    if (distance > distanceToObject) {
     setAngle(origin);
     turn(MODE_AUTO, TURN_LEFT, angle, origin, SPEED_AUT, SPEED_AUT);
     flag = -1;
     return 1;
    } else {
     flag++;
     return 0;
    }
}

 /* Scanner a droite */
```

```
int scanRight(int origin)
{
    int angle = 0;
    int distance = 0;
    angle = origin + 50;
    setAngle(angle);
    distance = acquisition(5);
    if (distance > distanceToObject) {
     setAngle(origin);
     turn(MODE_AUTO, TURN_RIGHT, angle, origin, SPEED_AUT, SPEED_AUT);
     flag = -1;
     return 1;
    } else {
     flag++;
     return 0;
    }
}

/************ LIASON SERIE **********************/
void serie(int *recept)
{
    if (Serial.available() > 0) {
     /* Read the incoming byte */
     *recept = Serial.read();
    }
}

void automatic_obstacle_mode()
{
    setAngle(SET_TO_MIDDLE);
    while (incomingByte == 'a') {
     distanceToObject = acquisition(2);    /* Verifie la presence d'un
obstacle */
     if (distanceToObject > 30) {
         robot_control(MOVE_FW, SPEED_AUT, SPEED_AUT);
     } else {
         if (flag < 0) {
          angleToObject = getAngle();
         }
         moteurOff();
         if (!scanRight(angleToObject)) {
          if (!scanLeft(angleToObject)) {
              turn(MODE_AUTO, TURN_RIGHT, 0, 360, SPEED_AUT,
                SPEED_AUT);
              delay(1000);
          }
         }
     }
     serie(&incomingByte);
    }
}

void suiviSens1()
{
    analog(ccvalue);
    while (incomingByte == 'c') {

     newcolor = getColorValue(&ccvalue);
```

```
      serie(&incomingByte);
      distanceToObject = acquisition(2);    /* Verifie la presence d'un
obstacle */
      if (distanceToObject > 30) {
          Serial.println(newcolor);
          Serial.println("HORS SWICH");
             j=j+1;
             //Serial.println("j : ");
             //Serial.println(j);
             if (j==12) {
          analog(ccvalue); j=0; i=1;}
             if (i==1 && j==11) {i=0;
             robot_control(MOVE_BK, SPEED_COM - 10, SPEED_COM - 10);
             delay(500);
             moteurOff();
             delay(100);
             }
          newcolor = getColorValue(&ccvalue);
          switch (newcolor) {
          case RED:
           while (1) {
               Serial.print("RED : ");
               newcolor = getColorValue(&ccvalue);
               Serial.println(newcolor);
               //analog(ccvalue);
               serie(&incomingByte);
               Serial.print("rouge");
               if (newcolor != RED || incomingByte=='q') {
                moteurOff();
                break;
               }
               robot_control(MOVE_FW, SPEED_COM-50, SPEED_COM-50);
           }
          case GREEN:
           while (1) {
               newcolor = getColorValue(&ccvalue);
               Serial.print("GREEN : ");
               Serial.println(newcolor);
               serie(&incomingByte);
               Serial.print("vert");
               //turn(MODE_COMMANDE, TURN_LEFT, 80, getAngle(),
TURN_SPEED, TURN_SPEED);
               if (newcolor != GREEN || incomingByte=='q')
                break;
               //if (newcolor == RED || newcolor == BLACK || incomingByte
!= 'a') { Serial.println("raaah"); break;}
               turn(MODE_COMMANDE, TURN_LEFT, 92, 93, TURN_SPEED,
                 TURN_SPEED);
               delay(150);
               moteurOff();
               delay(100);
           }
           break;
          case BLACK:
           while (1) {
               Serial.print("BLACK : ");
               newcolor = getColorValue(&ccvalue);
               Serial.println(newcolor);
```

```
              serie(&incomingByte);
              delay(10);
              Serial.print("noir");
              //if (newcolor == RED || newcolor == GREEN || incomingByte
!= 'a') break;
              if (newcolor != BLACK || incomingByte=='q')
               break;
              //turn(MODE_COMMANDE, TURN_RIGHT, 100, getAngle(),
TURN_SPEED, TURN_SPEED);
              turn(MODE_COMMANDE, TURN_RIGHT, 94, 93, TURN_SPEED,
                TURN_SPEED);
              delay(150);
              moteurOff();
              delay(100);
           }
           break;
         case WHITE:
           while (1) {
              Serial.print("WHITE : ");
              newcolor = getColorValue(&ccvalue);
              Serial.println(newcolor);
              serie(&incomingByte);
              distanceToObject = acquisition(2);
              if (newcolor != WHITE || incomingByte=='q') {
               moteurOff();
               break;
              }
              delay(100);
              robot_control(MOVE_BK, SPEED_COM - 50, SPEED_COM - 50);
              Serial.print("Track lost");
           }
         default:
           break;
         }
      } else {
         moteurOff();

         Serial.println("STOP!!!");
      }
    }
    delay(100);
}

void suiviSens2()
{
    while (incomingByte == 'i') {
      serie(&incomingByte);
      distanceToObject = acquisition(2);    /* Verifie la prÃƒÂ©sence d'un
obstacle */
      if (distanceToObject > 30) {
         newcolor = getColorValue(&ccvalue);
           Serial.println(newcolor);
         Serial.println("HORS SWICH");
           j=j+1;
           //Serial.println("j : ");
           //Serial.println(j);
           if (j==12) {
         analog(ccvalue); j=0; i=1;}
```

```
   if (i==1 && j==11) {i=0;
   robot_control(MOVE_BK, SPEED_COM - 10, SPEED_COM - 10);
   delay(500);
   moteurOff();
   delay(100);
   }
switch (newcolor) {
case RED:
 while (1) {
     Serial.print("RED : ");
     newcolor = getColorValue(&ccvalue);
     Serial.println(newcolor);
     //analog(ccvalue);
     serie(&incomingByte);
     Serial.print("rouge");
     if (newcolor != RED || incomingByte=='q') {
      moteurOff();
      break;
     }
     robot_control(MOVE_FW, SPEED_COM-30, SPEED_COM-30);
 }
case GREEN:
 while (1) {
     newcolor = getColorValue(&ccvalue);
     serie(&incomingByte);
     delay(10);
     Serial.print("vert");
     if (newcolor != GREEN || incomingByte=='q')
      break;
     turn(MODE_COMMANDE, TURN_RIGHT, 94, 93, TURN_SPEED,
       TURN_SPEED);
     delay(150);
     moteurOff();
     delay(100);
 }
 break;
case BLACK:
 while (1) {
     newcolor = getColorValue(&ccvalue);
     serie(&incomingByte);
     delay(10);
     Serial.print("noir");
     if (newcolor != BLACK || incomingByte=='q')
      break;
     turn(MODE_COMMANDE, TURN_LEFT, 92, 93, TURN_SPEED,
       TURN_SPEED);
     delay(150);
     moteurOff();
     delay(100);
 }
 break;
case WHITE:
 while (1) {
     newcolor = getColorValue(&ccvalue);
     serie(&incomingByte);
     distanceToObject = acquisition(2);
     if (newcolor != WHITE || incomingByte=='q') {
      moteurOff();
```

```
                break;
               }
               robot_control(MOVE_BK, SPEED_COM - 50, SPEED_COM - 50);
               Serial.print("Track lost");
               delay(100);
             }
          default:
           break;
          }
      } else {
          moteurOff();
             Serial.print("STOP!!!");
      }
    }
    delay(100);
}
```

Annexe 4 : Readme

```
**************************************************************************
**************************************README*****************************
**************************************************************************
Tout d'abord :

   login foxboard : root
   mot de passe : glopglop

   Il semble utile de savoir (meme si cela semble evident) que vous
pouvez tester vos programme ARDUINO directement en connectant ce dernier
sur le port USB : en choisissant
     * le bon port dans Tools->Serial Port
     * le bon arduino dans Bord->Arduino Duemilanove ATMega328P

   Et ensuite vous pouvez acceder au terminal de ARDUINO (Serial
Monitor).
Mais nous ne attarderons pas plus sur la maniere d'utiliser le logiciel
ARDUINO que vous etes censes deja  savoir utiliser !

   Egalement, le fichier .c de la foxboard se situe dans FoxLego, dans ce
coin la. Pas d'erreur possible, jai supprime tout ceux qui etaient
inutiles.

**************************************************************************
*****************************Utilisation du robot*************************
**************************************************************************

   Entrons dans le vif du sujet : Pour que le robot fonctionne, il vous
faut
mettre le programme arduino (extension .pde ou .arduino maintenant il me
semble) dans ... le arduino ! il est dans ce meme dossier sous le nom de
arduino.txt pour pouvoir le lire depuis la foxboard, il vous suffit de
vous
mettre dans le repertoire cible et de faire
       cat arduino.txt
comme vous l'avez surement fait pour lire ce readme.

* vous copiez son contenu dans ARDUINO, vous compilez, transferez tout en
respectant ce qui a ete marque le paragraphe d'avant !
* Voila, pour le arduino, c'est fini. N'oubliez pas d'allumer
l'interrupteur sur le robot, ca peut etre utile (c'est l'experience qui
parle la ...)

Ensuite, vous avez sans doute compris que le but est de piloter le robot
via la page presente sur la foxboard dans le dossier
/etc/www/lego_modified/ .

Pour pouvoir y acceder :
* brancher la clef wifi sur la foxboard.
* relier la foxboard (grace au port usb restant) au robot (mini port
USB).
--------------------------------------------------------------------------
ATTENTION : LE PORT USB DE L'ARDUINO EST TRES FRAGILE ! Il a ete arrache
et a ete repare mais bon, c'est pas du titanium le truc quoi...
```

---
* on allume la foxboard (vous avez la possibilite de vous y connecter en utilisant minicom -os /*voire votre cours de reseau*/ si vous voulez voir ce qu'il s'y passe : rien d'incroyable, ne soyez pas decu...)
* la foxboard a ete configurer pour se connecter directement sur le reseau de
l'ecole donc pas de manip pour la connecter.
* il est possible que le wifi ne soit pas tres puissant la ou vous etes (ce sera surement le cas d'ailleur). Vous avez donc la possibilitÃ© d'utiliser un cable ethernet relie aux bornes de la salle.
manip via minicom:

```
ifdown wlan0
ifup eth0
```

et le tour est joue ! (inversement pour revenir au wifi of course).
* a ce niveau la , le robot doit etre operationnel. Z'avez plus qu'a le poser a terre pres d'un point d'acces wifi de preference.
* depuis votre ordinateur vous n'avez qu'a lancer votre navigateur favoris et vous connecter a

http://172.26.167.120/lego_modified/configure.html

172.26.167.120 etant l'adresse IP de la foxboard (vous pouvez trouver cette derniere en tapant ifconfig dans la fox (iwconfig, oui oui, w, permet
de montrer sur quel reseau vous etes connectes, ou pas.
* si tout ce passe bien (il est possible que le systeme ne soit pas tres reactif auquel cas il vous faut patienter et actualiser ou reboot...)
la page sur laquelle vous etes vous demande de lancer le demon, allez y !
un numero doit s'afficher en VERT pas en rouge, sinon recommencez !
* ceci etant fait, allez sur la page conduire (je crois^^) dans les onglets
de la page en haut.
* et voila ! vous pouvez, a partir de la faire les operations que vous desirez !
* pour ateindre la foxboard, il suffit de faire arreter systeme sur la page precedente.

****
********** Plan arduino**********
****

Mon binome et moi avons galere a maitriser le robot. En effet il n'y avait
pas de ReadMe comme nous l'avons si bien fait (par compassion en fait). Les
cables du robot peuvent paraitre assez mystiques au premiers abords mais en
fait non.
Dans notre immense bonte

```
5V ------------------> 5V
Echo-----------------> Pin 2 digital
Trigger--------------> Pin 3 digital
0V-------------------> GND
```

Capteur Couleur ADJD S371

```
LED------------------> Pin 11 (PWM)
SDA------------------> Pin 4 analog
SCL------------------> Pin 5 analog
GND------------------> GND
```

```
      3.3V-----------------> 3.3V
***Servomoteur HS422***
      5V-------------------> 5V
      Commande (PWM)--------> Pin 9 digital
      0V-------------------> GND
```

****
****
****