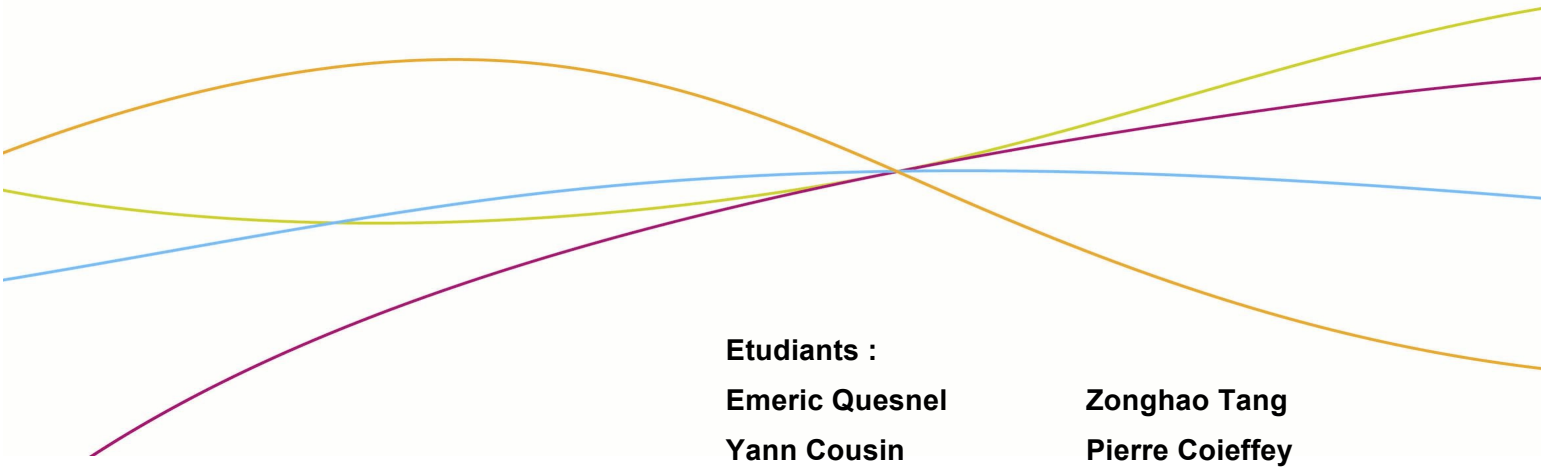
Projet de Physique P6

STPI/P6/2015 – 29

Robot Sumo n°2



Etudiants :

Emeric Quesnel **Zonghao Tang**

Yann Cousin **Pierre Coieffey**

Malek CheikhRouhou **Maxime Lizere**

Enseignant-responsable du projet :

Fabrice Delamare

Date de remise du rapport : ***14/06/2015***

Référence du projet : ***STPI/P6/2015 – 29***

Intitulé du projet : ***Robot Sumo n°2 (pour matchs contre le robot sumo n°1)***

Type de projet : ***Expérimental***

Objectifs du projet :

Conception, codage et réalisation d'un robot sumo dans le but d'affronter celui réalisé par le groupe du projet Robot Sumo n°1.

Mots-clefs du projet *(4 maxi)* : ***Robot Sumo***

Si existant, n° cahier de laboratoire associé : ***/***

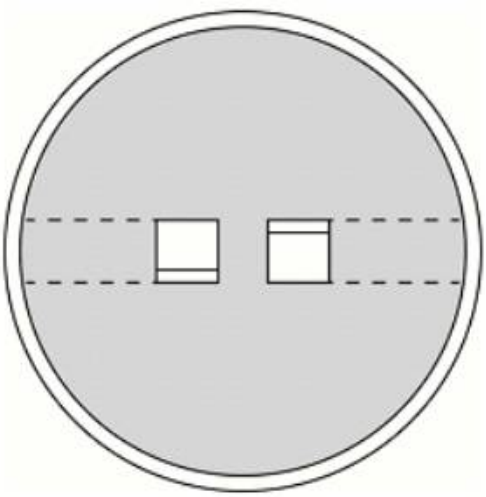
INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE ROUEN
Département Sciences et Techniques Pour l'Ingénieur
BP 8 – avenue de l'Université - 76801 Saint Etienne du Rouvray - tél : +33(0) 2 32 95 66 21 - fax : +33(0) 2 32 95 66 31

TABLE DES MATIERES

1. INTRODUCTION

Durant le quatrième semestre du cycle STPI, dans l'optique de l'EC de P6 (projet de Physique) nous avons été chargés de réaliser un robot sumo. Le but était qu'il soit suffisamment performant pour vaincre le robot de l'équipe adverse.

Les règles des combats sont relativement simples :



Un exemple de dohyo

- Si un des robots sort du terrain l'adversaire marque un point (*Yuko*)
- Les deux robots sont placés en opposition sur la surface de combat (*dohyo*). 5 secondes après avoir été activés ils peuvent commencer à se déplacer.
- Les combats sont constitués de 3 manches de 3 minutes chacune
- L'équipe qui marque 2 points ou plus gagne
- En cas d'égalité la rencontre est rejouée

2. METHODOLOGIE / ORGANISATION DU TRAVAIL

Ce projet faisant appel à des compétences dans plusieurs domaines, nous nous sommes répartis les tâches dans 3 pôles différents :

- Conception et Réalisation :

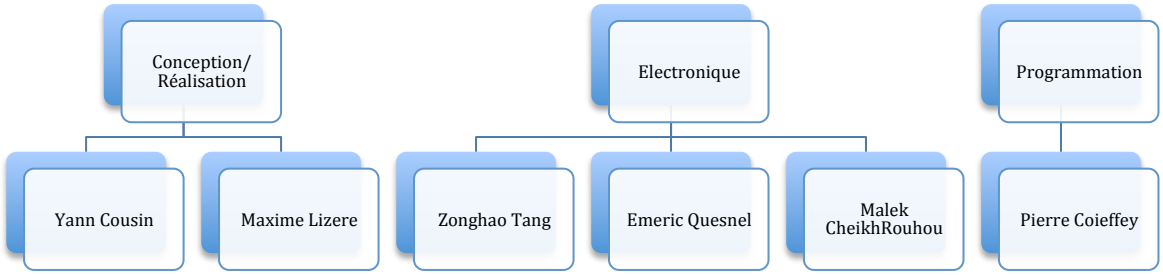
  Pôle chargé de concevoir le robot, de le modéliser sous solidworks, puis de le fabriquer. Des études ont été également effectuées par ce pôle afin d'évaluer en amont les performances du robot et d'ainsi soutenir le développement d'une stratégie adéquate par le pôle programmation.

- Electronique

Pôle chargé de la réalisation des composants électroniques dont le robot est doté (soudures, etc.)

- Programmation

Pôle chargé du développement du programme du robot et de l'élaboration d'une stratégie de combat.



Afin d'être le plus efficace possible, les membres des trois pôles, au cours des différentes séances, se sont réunis afin de faire le point sur les avancées de chacun et ce durant tout le semestre.

Ce projet nous aura demandé en fin de compte beaucoup de travail. En effet que ce soit l'apprentissage de l'utilisation de Solidworks, du langage Arduino ou de notions d'électronique qui nous manquaient, nous avons tous du faire l'apprentissage par nous même de choses nouvelles.

3. TRAVAIL REALISE ET RESULTATS

3.1. Conception et Réalisation

3.1.1. Conception

3.1.1.1. Réflexion

Au commencement du projet, au tout début du semestre cela a été le premier travail réalisé. Nous nous sommes réunis en groupe complet et avons commencé à réfléchir à la forme que nous voulions donner au robot. Pour cela nous avons regardé diverses vidéos, cherché des exemples de robot etc.

Une fois la forme générale décidée, le travail sur le robot physique a été confié à Maxime Lizere et à Yann Cousin. En nous aidant du livre « Je construis mon robot Sumo » [1] nous avons réalisés des calculs sur le couple des moteurs, la masse maximale que nous pouvions admettre pour avoir un appareil fonctionnel.

Après cela, nous avons pris les mesures précises des pièces dont nous disposions déjà, afin de faciliter la modélisation sur le logiciel Soldiworks.

Nous avons ensuite réalisé des croquis approximatifs de la forme du robot, l'emplacement des divers composants etc.

3.1.1.2. Solidworks

Une fois cela fait, nous avons commencé à travailler à la modélisation sur solidworks. Dans un premier temps le processus a été compliqué, nous ne maitrisions pas totalement le logiciel et ne savions pas quelles dimensions avaient certaines pièces que nous n'avions pas sous la main.

Après avoir réalisé des modèles des roues et des moteurs nous avons fait plusieurs essais de forme de châssis avant de nous décider. Nous avons ensuite effectué un assemblage sur le logiciel avec les modèles de chaque pièce. Une fois les ajustements terminés nous nous sommes lancés dans la réalisation concrète du robot.

3.1.2. Réalisation

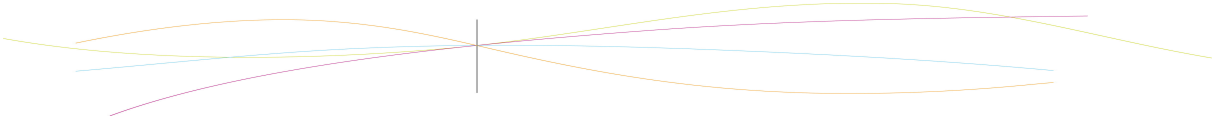
Plusieurs pièces nous étaient déjà fournies, la carte arduino, le shield, les moteurs, les capteurs etc.

Nous avons récupéré des roues de longboard (grand skateboard aux roues très larges) afin d'avoir une adhérence maximale au sol.

Pour le châssis nous avons récupéré une plaque de plexiglas peu épaisse, bon compromis entre solidité et légèreté que nous avons découpé à la scie aux bonnes dimensions. Il a ensuite fallu la percer aux divers endroits afin de pouvoir fixer les autres éléments.

Les moyeux servant a fixer l'axe du moteur a la roue ont été usinés par l'atelier méca pour les deux groupes, aux tailles standards des roulements de roues de skateboard/roller etc. En les essayant, nous nous sommes cependant rendus compte qu'ils étaient légèrement trop grands et avons du les limer a la bonne taille.

Nous avons également sciés deux « équerres » dans une planche d'aggloméré afin de servir au support de la pelle –une plaque d'aluminium.



La partie électronique venant tout juste d'être terminée a la fin de la dernière séance de l'EC nous avons manqué de temps pour assembler capteurs, carte arduino et moteurs sur le corps du robot. Nous avons essayé tout le reste de la semaine d'aller dans la salle de robotique afin de pouvoir terminer l'assemblage. La présence d'un technicien étant indispensable pour entrer dans la salle, nous nous sommes heurtés au problème de son absence. Nous n'avons donc pu, à ce jour, terminer l'assemblage et les tests du robot sumo.

3.2. Electronique

Dans cette partie d'électronique, l'objectif était d'établir le lien entre le programme sur la carte Arduino et les différentes composantes du robot. Ce lien nous permet de recueillir les informations nécessaires travers les capteurs, prendre des décisions selon le programme informatique, et appliquer ces dernières grâce aux moteurs.

3.2.1. Les composantes :

3.2.1.1. Les capteurs

Pour la partie électronique, nous avons préalablement étudié l'impact des capteurs optiques pour, d'une part optimiser le déplacement du robot sur le 'dohyo' afin de ne pas dépasser la ligne noire et d'autre part pour la détection du robot adverse le plus vite possible, dans le but de pouvoir attaquer et appliquer notre stratégie de jeu. Nous avons donc étudié le nombre de capteurs ainsi que leur position.

Après réflexion et discussion avec le professeur, nous avons opté pour deux capteurs TOR (tout ou rien) : un GP2D150 pour localiser l'adversaire sur un rayon de 150 cm, installé au sommet du robot et un détecteur de sol TCRT5000 qui est placé en dessous pour détecter les bords noirs du dohyo par variation d'intensité lumineuse. Ce capteur TCRT5000 est constitué d'un photo-transistor et d'une diode électroluminescente (DEL) destiné à envoyer un signal lumineux à une certaine longueur d'onde qui, réfléchi sur le sol blanc, est reçu par le photo-transistor. Le signal logique transmis correspond donc au fait que le photo-transistor est éclairé, quand il se trouve sur le dohyo « blanc », ou non (quand il s'approche du bord « noir »). Nous avons fait ce choix en raison de ce qu'un seul capteur nous était, semblait-t-il, suffisant pour détecter le robot adverse (un tour étant nécessaire afin de détecter le robot concurrent sur un diamètre appréciable). De plus, au regard de la stratégie, plutôt basique, mise en œuvre, nous n'avons pas jugé indispensable de choisir un télémètre, qui nous aurait renseigné sur la distance à laquelle se trouve l'adversaire, étant donné que nous n'aurions pas exploité cette faculté

De plus, la surface du dohyo étant assez petite et, en conséquence, la distance séparant les robots relativement faible, ces facteurs nous ont confortés dans l'idée qu'un capteur TOR était le plus judicieux compte tenu de notre stratégie offensive, puisque consistant à attaquer dès la détection de l'adversaire. D'autre part, nous avons choisi d'installer un seul capteur de sol au regard de la taille de la bordure noire : assez large pour qu'il soit possible de la repérer et d'agir avant de sortir du dohyo.

3.2.1.2. Les moteurs

On dispose aussi de deux moteurs de type RB-35 qui ne consomment pas beaucoup d'énergie et qui ont les dimensions adéquates pour notre type de robot.

3.2.1.3. Les résistances

Une résistance de 330 Ohm pour la diode et une autre de 10.000 Ohm pour la photorésistance du capteur TCRT5000

3.2.1.4. L'Ardumoto (Motor Driver Shield)

Cette pièce est le lien entre la carte Arduino, sur laquelle il y a notre programme informatique, et le reste des composantes. En effet, le shield se monte sur la carte Arduino grâce a des petits bâtons métalliques et il est connectable aux autres composantes soit en soudant directement ou grâce aux fils conducteurs.

3.2.1.5. Autres composantes

Une batterie, un connecteur pour la batterie, un connecteur pour le GP2D150.

3.2.2. Conception de la carte :

Afin de concevoir la carte nous avons choisi la méthode classique vu que nos connaissances en électronique étaient basiques et que les cours que nous avons eu en P3 en première année n'étaient pas satisfaisants pour faire la conception d'une telle carte. Nous avons donc commencé à faire nos propres recherches sur Internet pour mieux cerner l'ensemble du problème. La conception de la carte électronique s'est déroulée en une seule étape de conception schématique, qui consistait en une représentation symbolique du circuit électronique, schémas des composants etc.

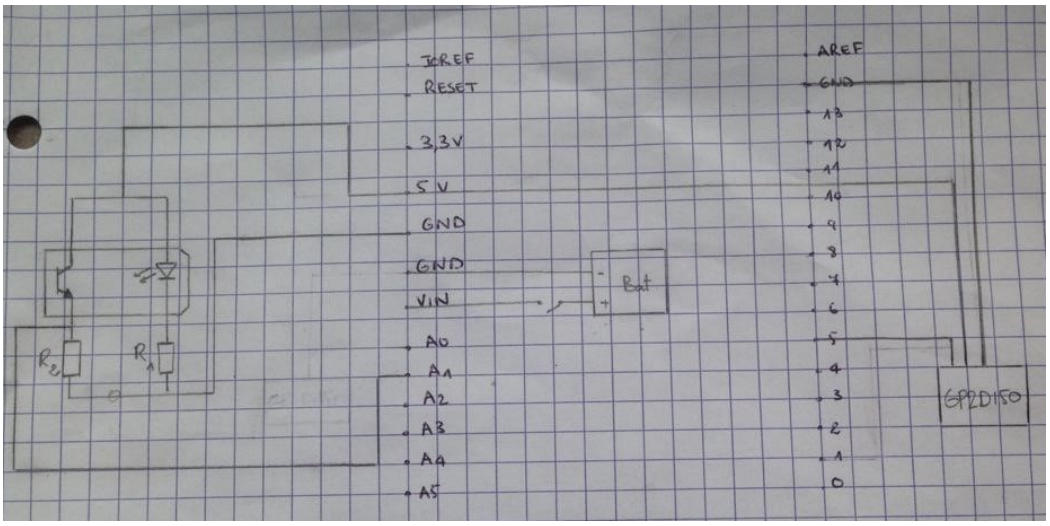


figure 1 : Représentation symbolique du circuit électronique

3.2.3. Réalisation technique :

Apres vérification auprès de Fabrice Delamare et correction de notre schéma sur papier, nous avons monté chaque composant sur le Shield. En effet, sur ce dernier, nous avons soudé deux résistances (en rapport avec le TCRT5000), deux connecteurs. Nous avons aussi relié les deux moteurs aves les fils connecteurs.

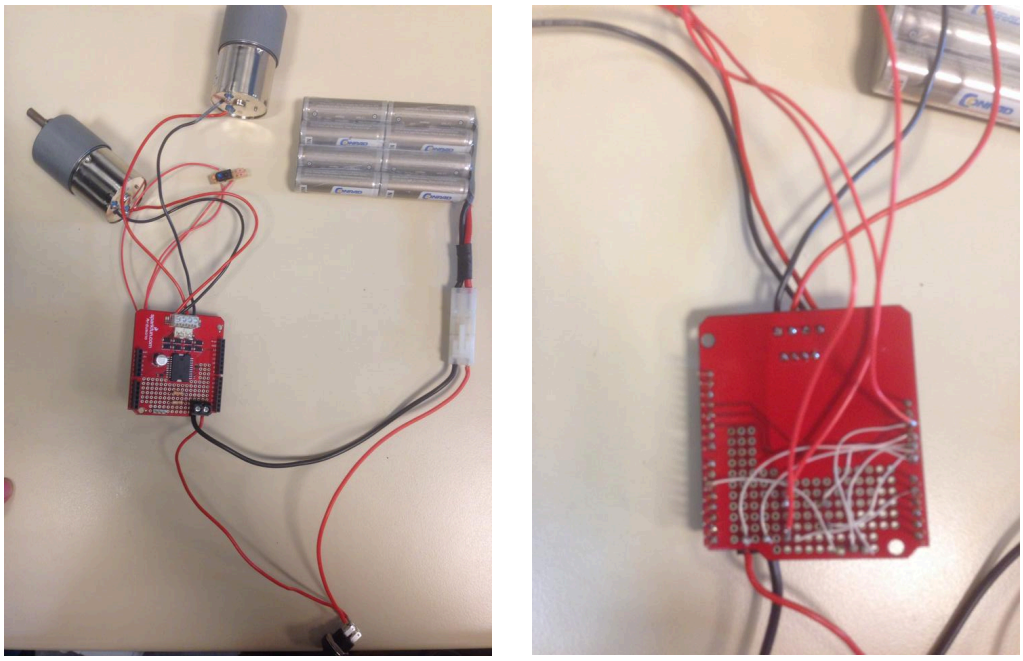


figure 2 : Shield, vue de dessus et de dessous

3.3. Codage du robot

Le but de cette partie était de mettre en œuvre un programme informatique permettant à tous les composants d'être utilisés ensemble dans le but de faire faire à notre robot sumo ce que nous attendions de lui. Ce programme, une fois terminé sera transmis à la carte arduino qui exécutera ce dernier.

3.3.1. Les principales fonctions du langage arduino

La première partie du travail consistait à s'approprier le langage arduino, qui est en fait très proche du C et du C++. Après quelques recherches nous avons donc compris qu'un programme arduino tournait autour de deux fonctions principales : la fonction « setup » et la fonction « loop ».

3.3.1.1. La fonction « setup »

Cette fonction comme son nom l'indique va nous permettre d'initialiser tout un tas de choses comme les entrées et sorties de la cartes arduino ainsi que la communication entre la carte arduino et l'ordinateur avec l'utilisation de la fonction « serial.begin ». Cette fonction ne sera exécutée qu'une seule fois au début du programme contrairement à la fonction « loop » qui va tourner tout le temps.

```
void setup()
{
// On précise si les broches sont utilisées en entrée (INPUT) ou en sortie (OUTPUT).

pinMode(sens_rota_moteur_a, OUTPUT);
pinMode(sens_rota_moteur_b, OUTPUT);
pinMode(vitesse_moteur_a, OUTPUT);
pinMode(vitesse_moteur_b, OUTPUT);
pinMode(capteur_adversaire, INPUT);
pinMode(capteur_ligne, INPUT);

// On initialise la communication entre la carte arduino et l'ordinateur à une vitesse de 9600 bit/s.

Serial.begin(9600);
}
```

3.3.1.2. La fonction « serial »

Cette fonction permet à la carte arduino de renvoyer à l'ordinateur, par l'intermédiaire du moniteur série, du texte ou certaines valeurs importantes comme celles renvoyées par les capteurs. Cette fonction va nous permettre de suivre l'évolution de notre robot au cours des tests pour voir éventuellement dans quelle partie du code se situe un problème. De plus, cette fonction va nous permettre de tester les valeurs renvoyées par certains capteurs pour pouvoir ajuster notre programme à celles-ci.

3.3.1.3. 3/ La fonction « loop »

La fonction « loop » quant à elle, est une fonction qui va tourner tout le temps. C'est le cœur du programme, elle comprend toutes les instructions données à notre robot.

3.3.2. La gestion des composants

3.3.2.1. Les moteurs

Bien entendu, les instructions données à notre robot sont toutes liées à ses déplacements. C'est pourquoi, la plus grande partie du programme et de notre travail a été d'apprendre à gérer les moteurs.

A. Le sens de rotation

Tout d'abord, pour savoir gérer les moteurs dans un programme il faut savoir comment ils fonctionnent dans la réalité. Il est donc impératif de savoir que pour pouvoir avancer il faut que les moteurs tournent dans le sens contraire et pour pouvoir tourner il faut que ces derniers tournent dans le même sens. Dans la suite de ce paragraphe, nous avons choisi de définir les sens de rotation (horaire et trigonométrique) des moteurs en regardant le robot de l'extérieur.

Mais comment savoir dans quel sens il faut que le moteur 'a' tourne pour que le robot avance ?

Ici seuls les tests peuvent nous le dire et nous avons donc déterminé que le moteur 'a' devait tourner dans le sens horaire et le moteur 'b' dans le sens trigonométrique pour que le robot puisse avancer, et dans les deux sens contraires pour que le robot puisse reculer. De la même façon, il faut que les moteurs tournent dans le sens horaire pour que le robot se dirige vers la gauche et dans le sens trigonométrique pour que le robot se dirige vers la droite.

Mais comment représenter les sens horaire et trigonométrique dans le langage arduino ?

Nous avons choisi une entrée numérique pour le sens de rotation des moteurs donc il ne nous restait plus qu'à choisir aléatoirement un état (HIGH ou LOW) pour chacun des deux sens. Dans notre cas, nous avons choisi de représenter le sens horaire avec HIGH et le sens trigonométrique avec LOW.

B. La vitesse de rotation

Une fois le cas du sens de rotation résolu, il ne nous reste plus qu'un dernier point à voir pour en finir avec l'utilisation de ces moteurs. Et ce dernier point n'est autre que la vitesse de rotation. En effet, c'est bien de pouvoir programmer dans quelle direction va aller notre robot mais ce serait encore mieux si nous pouvions programmer à quelle vitesse il va y aller. Et l'on est parvenu à faire cela avec la PWM (Pulse Width Modulation ou Modulation de la largeur d'impulsion) qui nous a permis de convertir une donnée numérique (HIGH ou LOW) en une valeur analogique (tension).

En effet, le fait de faire varier pendant un laps de temps plus ou moins long les valeurs HIGH et LOW revient à obtenir une tension régulière comprise entre 0 et 5V, et cette tension s'obtient en multipliant le rapport cyclique avec la tension maximale soit 5V. Le rapport cyclique étant le rapport entre la durée où le niveau HIGH est présent et la durée où le niveau LOW l'est à son tour. La PWM permet donc de gérer l'intensité d'éclairage d'une LED ou la vitesse de rotation d'un moteur par exemple. Cependant, plutôt que d'indiquer le rapport cyclique souhaité, il nous a fallu dans notre programme, indiquer un nombre compris entre 0 et 255 (0 représentant un rapport cyclique de 0 % et 255 représentant un rapport de 100%). En effet, ceci permet aux valeurs transmises d'être stockées sur un octet.

C. Exemple

Nous sommes donc à présent capable de faire fonctionner nos moteurs. Voici un exemple d'une fonction de direction pour le robot, en l'occurrence ici la fonction « tourner_vite_droite », les autres fonctions étant d'une même genre.

```
void tourner_vite_droite()
{
  digitalWrite(sens_rota_moteur_a,LOW);
  digitalWrite(sens_rota_moteur_b,LOW);
  analogWrite(vitesse_moteur_a,0);
  analogWrite(vitesse_moteur_b,255);
}
```

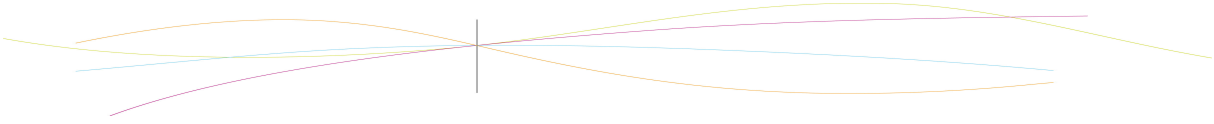
3.3.2.2. Les capteurs

Maintenant, il ne nous reste plus qu'à gérer les capteurs pour adapter les mouvements de notre robot aux valeurs transmises par ces derniers. La gestion du capteur d'adversaire

GP2D150 est très simple étant donné que ce capteur est de type tout ou rien, nous l'avons donc branché sur une broche numérique afin qu'il nous renvoie des valeurs du type HIGH ou LOW : HIGH signifiant qu'il a détecté un adversaire et LOW signifiant qu'il n'a rien détecté. La gestion du capteur de ligne TCRT5000 quant à elle, s'est avérée être plus difficile étant donné que ce dernier nous renvoie des valeurs comprises entre 0 et 1023. En effet, celui-ci est branché sur une broche analogique. Par conséquent, nous avons dû faire un petit programme test afin de voir comment varier les valeurs transmises par le capteur quand on approchait celui-ci de la ligne noire du dohyo.

```
// On associe le capteur de ligne à la broche analogique A1.

const int capteur_ligne = A1;

// On déclare une variable.

int ligne;

void setup()
{
  // On définit la broche utilisée comme étant une entrée (INPUT).

  pinMode(capteur_ligne, INPUT);

  // On initialise la communication entre la carte arduino et l'ordinateur à une vitesse de 9600 bit/s.

  Serial.begin(9600);
}

void loop()
{

  // On stocke la valeur transmise par le capteur de ligne dans une variable.

  ligne= analogRead(capteur_ligne);

  // On transmet cette valeur au moniteur série de l'ordinateur.

  Serial.print("ligne :");
  Serial.println(ligne);

  // On attend une seconde pour avoir le temps de rapprocher le capteur
  // de la ligne noire du dohyo et obtenir ainsi une nouvelle valeur.

  delay(1000);
}
```

On s'est alors aperçu grâce à ce petit programme que les valeurs augmentaient au fur et à mesure que l'on se rapprochait de la ligne. On a donc pu fixé une valeur (qui correspond en réalité à une distance) à laquelle on souhaitait que notre robot s’arrête pour qu'il ne sorte pas du dohyo.



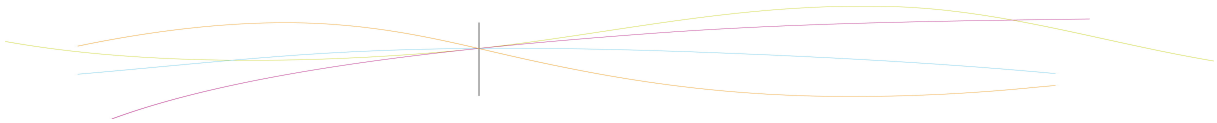
3.3.3. La stratégie adoptée

Une fois la gestion des moteurs et des capteurs acquise il ne nous reste plus qu'à choisir une stratégie de combat qui sera contenue dans la fonction « loop ».

Premièrement, à chaque tour de boucle, il faut lire les données transmises par les capteurs car ce sont ces dernières qui nous permettent de diriger le robot là où nous le souhaitons. Ensuite, vient le moment du début du combat mais avant cela il est prévu dans les règles de combat de robots sumos d'attendre 5 secondes avant que les robots de démarrent. Cette attente va être réalisée à l'aide de la fonction « delay » qui permet de faire une pause au programme durant un temps en millisecondes, indiqué dans les paramètres d'entrée de la fonction. Une fois ces 5 secondes écoulées, le combat peut commencer. Nous avons choisi pour débuter la partie de partir directement sur la droite sans se soucier de notre adversaire. Le but étant d'éviter un face et face directement après le départ. Une fois cette étape qui ne se produira qu'au départ, nous avons structuré notre programme sous la forme suivante 'if...else if...else…'. La condition la plus importante à respecter étant bien sûr placée dans le premier 'if' alors que la moins importante est située dans le dernier 'else'. Il nous est apparu évident que la condition principale à respecter était de ne pas sortir du dohyo c'est à dire que dès que notre robot approche trop prêt de la ligne noire nous le faisons reculer. La deuxième condition que nous avons situé dans le 'else...if' est la détection de l'adversaire, c'est à dire que quand notre robot va détecter l'ennemi, il va avancer rapidement vers lui, cependant il faut faire très attention au temps d'attaque de notre robot. En effet, si notre robot avance trop rapidement et trop longtemps sans se soucier des capteurs, il ne va pouvoir voir si l'ennemi s'est déplacé entre temps et il ne va pas non plus être capable de détecter la ligne noire et il va donc sortir tout seul du dohyo. C'est pourquoi, la durée du « delay » est très faible. Finalement, il nous reste la troisième et dernière condition : que faire si on ne détecte ni la ligne noire ni notre adversaire ? Eh bien il faut que notre robot tourne lentement sur lui-même pour essayer de détecter au mieux l'ennemi. Cependant, pendant qu'il tourne sur lui même notre robot est vulnérable, donc nous avons choisi de le faire avancer sur une courte distance après qu'il ait réalisé une légère rotation sur lui-même afin d'éviter qu'il ne reste trop longtemps à la même place. Pour finir, pour éviter de brusquer les moteurs, nous avons choisi avant chaque changement de direction de notre robot de passer par la case arrêt. En effet, il serait mal venu de passer d'une marche avant rapide à une marche arrière tout aussi violente. Ceci est d'ailleurs totalement impossible pour une voiture.

```
void loop()
{
  // On stocke les valeurs transmises par nos capteurs dans nos variables.

  adversaire= digitalRead(capteur_adversaire);
  ligne= analogRead(capteur_ligne);

  if (debut==true)
  {
    delay(5000);
    tourner_vite_droite();
    delay(500);
```

```
  arreter();
  avancer();
  delay(1000);

  // On assigne false à la variable debut pour éviter que cette séquence
  // prévue uniquement au départ ne se reproduise plus tard dans le combat.

  debut=false;
 }

 if (ligne>600)
 {
  arreter();
  reculer();
  delay(1000);
  Serial.print("ligne :");
  Serial.println(ligne);
 }

 else if (adversaire==HIGH)
 {
  arreter();
  avancer();
  delay(10);
  Serial.println("adversaire trouvé");
 }

 else
 {
  arreter();
  tourner_lentement_gauche();
  delay(50);
  arreter();
  avancer();
  delay(10);
  Serial.println("recherche adversaire");
 }
}
```



4. CONCLUSIONS ET PERSPECTIVES

Malek :

Ce projet m'a permis de découvrir une discipline différente de ce que j'avais l'habitude d'étudier, à savoir la robotique. La réalisation et la conception d'un robot sumo du début à la fin m'a permis de faire le lien entre l'aspect informatique, électronique et mécanique du robot qui était quelque peu ambigu à mon gout. De plus, grâce à ce projet, j'ai pu m'initier d'une part à l'électronique, à travers la conception de la carte et le soudage des différentes composantes sur cette dernière ; et d'autre part à un nouveau langage de programmation (C). En outre, j'ai appris à être plus autonome et rigoureux dans mon travail.

Emeric :

Peu désireux de m'occuper de la partie électronique au départ, en particulier en raison de ce que le nom m'inspirait une certaine appréhension (le domaine m'étant le moins familier de tous), j'ai été surpris d'y prendre assez vite goût. En effet, j'ai éprouvé une certaine satisfaction à réaliser une tâche requérant à la fois précision et concentration de manière assez exigeante, notamment au niveau des soudures sur la carte Ardumoto. Et, malgré notre manque d'organisation du départ en grande partie dû à une absence de directives de groupe clairement définies (et qui aura très certainement été l'obstacle majeur à l'aboutissement de ce projet de manière définitive), la réalisation d'un tel travail en commun, alliant collaboration et prise d'initiatives, m'a apporté un grand contentement à la fois, grâce à la vue de l'avancement progressif du robot malgré nos soucis, ainsi qu'à la synergie de groupe qui a finalement finie par s'instaurer.

Maxime :

Au travers de l'EC de P6 j'ai pu découvrir la conception et de la construction concrète de quelque chose. Ce projet m'aura permis de mettre en œuvre des compétences différentes de celles utilisées habituellement en école d'ingénieur.

Même si le projet n'a pas été couronné de succès, il était intéressant de travailler en groupe sur quelque chose d'aussi concret.

Yann :

Tout d'abord, j'étais très enthousiaste à l'idée de réaliser ce projet, non seulement car la robotique est un domaine qui m'impressionne et qui m'attire mais aussi parce que ce projet est assez proche de l'enseignement de spécialité mécanique, qui est le département que j'envisage de suivre. Au cours de la réalisation du projet, j'ai beaucoup aimé travailler en groupe et construire le robot. Je pense que ce projet m'a permis de gagner en autonomie et de conforter mon projet professionnel.

Pierre :

Ce projet m'a tout d'abord permis de travailler sur une partie qui me plaît, en l'occurrence ici l'informatique. J'ai donc approfondi mes connaissances en programmation avec notamment l'apprentissage d'un nouveau langage sur une interface inconnue (arduino). Concernant, l'aspect collectif du projet, celui-ci m'a donné un aperçu encore plus concret de ce qui nous attend dans notre futur métier d'ingénieur. Je veux parler ici des délais à respecter, de la répartition des tâches, de la communication dans le groupe. Ça a parfois était compliqué car ce n'était que notre premier projet avec un groupe aussi conséquent cependant je pense que ce projet ne peut être que bénéfique pour la suite de mes études.

Zonghao :

Au cours de ce projet, j'ai acquis beaucoup de connaissances. Comme j'étais principalement en charge de la partie électronique, cela m'a fourni une bonne opportunité de travailler dans un atelier, chose nouvelle pour moi. Le soudage, le découpage et l'assemblage furent une nouvelle expérience pour moi. J'ai appris à respecter les règles de l'atelier, à savoir : économiser les matériaux, prendre des précautions au niveau de la sécurité et ranger les outils après utilisation. Au fur et à mesure de l'avancement du robot et de l'assemblage des composants, j'ai été heureux de voir notre collaboration et nos efforts porter peu à peu leurs fruits. Ce projet m'a également fait acquérir des capacités de travail manuel et de travail en groupe. En revanche, nous avons été confrontés à quelques problèmes notamment au niveau de l'efficacité au début et d'un certain manque de communication entre les différents groupes, un point qu'il conviendra d'améliorer lors de projets ultérieurs.

5. BIBLIOGRAPHIE

[1] Frédéric Giamarchi, “Je construis mon robot sumo”, *ETSF,Dunod*, 2011.

[2] lien internet : http://doc.ubuntu-fr.org/arduino (valide à la date du 14/06/2015).

[3] lien internet : http://fr.flossmanuals.net/arduino/historique-du-projet-arduino/(valide à la date du 14/06/2015).

[4] lien internet : http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.HomePage (valide à la date du 14/06/2015).

[5] lien internet : http://eskimon.fr/category/arduino (valide à la date du 14/06/2015).

[6] lien internet : http://www.arduino.cc/ (valide à la date du 14/06/2015).

[7] lien internet : http://playground.arduino.cc/Main/InterfacingWithHardware (valide à la date du 14/06/2015).

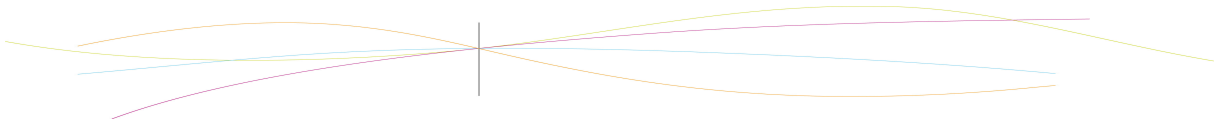
[8] lien internet : http://www.robotshop.com/blog/en/arduino-5-minute-tutorials-lesson-4-ir-distance-sensor-push-button-2-3637 (valide à la date du 14/06/2015).

6. ANNEXES (NON OBLIGATOIRE – EXEMPLES CI-DESSOUS)

6.1. Listings des programmes réalisés

```
// On fixe les entrées et sorties à un numéro de broche sur la carte arduino.

const int capteur_adversaire = 5;
const int capteur_ligne = A1;
const int vitesse_moteur_a =10;
const int vitesse_moteur_b =11;
const int sens_rota_moteur_a= 12;
const int sens_rota_moteur_b= 13;

// On déclare et initialise les différentes variables utilisées dans la suite du programme.

int adversaire;
int ligne;
boolean debut=true;

void setup()
{
// On précise si les broches sont utilisées en entrée (INPUT) ou en sortie (OUTPUT).

  pinMode(sens_rota_moteur_a, OUTPUT);
  pinMode(sens_rota_moteur_b, OUTPUT);
  pinMode(vitesse_moteur_a, OUTPUT);
  pinMode(vitesse_moteur_b, OUTPUT);
  pinMode(capteur_adversaire, INPUT);
  pinMode(capteur_ligne, INPUT);

// On initialise la communication entre la carte arduino et l'ordinateur à une vitesse de 9600 bit/s

  Serial.begin(9600);
}

// On assigne dans les fonctions suivantes un sens ainsi qu'une vitesse de rotation à nos moteurs.
// (LOW : sens trigonométrique/HIGH : sens horaire)
// (0 : vitesse nulle/255 : vitesse maximale)

void tourner_vite_droite()
{
  digitalWrite(sens_rota_moteur_a,LOW);
  digitalWrite(sens_rota_moteur_b,LOW);
  analogWrite(vitesse_moteur_a,0);
  analogWrite(vitesse_moteur_b,255);
}

void avancer()
```

```
{
  digitalWrite(sens_rota_moteur_a,HIGH);
  digitalWrite(sens_rota_moteur_b,LOW);
  analogWrite(vitesse_moteur_a,255);
  analogWrite(vitesse_moteur_b,255);
}

void reculer()
{
  digitalWrite(sens_rota_moteur_a,LOW);
  digitalWrite(sens_rota_moteur_b,HIGH);
  analogWrite(vitesse_moteur_a,255);
  analogWrite(vitesse_moteur_b,255);
}

void tourner_lentement_gauche()
{
  digitalWrite(sens_rota_moteur_a,HIGH);
  digitalWrite(sens_rota_moteur_b,HIGH);
  analogWrite(vitesse_moteur_a,127);
  analogWrite(vitesse_moteur_b,0);
}

void arreter()
{
  digitalWrite(sens_rota_moteur_a,LOW);
  digitalWrite(sens_rota_moteur_b,LOW);
  analogWrite(vitesse_moteur_a,0);
  analogWrite(vitesse_moteur_b,0);
}

void loop()
{
  // On stocke les valeurs transmises par nos capteurs dans nos variables.

  adversaire= digitalRead(capteur_adversaire);
  ligne= analogRead(capteur_ligne);

  if (debut==true)
  {
    delay(5000);
    tourner_vite_droite();
    delay(500);
    arreter();
    avancer();
    delay(1000);

    // On assigne false à la variable debut pour éviter que cette séquence de code
    // prévue uniquement au départ ne se reproduise plus tard dans le combat.

    debut=false;
  }

  if (ligne>600)
```



```
 {
   arreter();
   reculer();
   delay(1000);
   Serial.print("ligne :");
   Serial.println(ligne);
 }

 else if (adversaire==HIGH)
 {
   arreter();
   avancer();
   delay(10);
   Serial.println("adversaire trouvé");
 }

 else
 {
   arreter();
   tourner_lentement_gauche();
   delay(50);
   arreter();
   avancer();
   delay(10);
   Serial.println("recherche adversaire");
 }
}
```