
# Coupe de France de Robotique 2015
# Robot secondaire

*Coordinateur :* Jean Marc STEYEART
*Tuteur :* David FILLIAT
*CDU Référant :* CNE BUREU

*Membres du PSC :*
Alexis CLARIOND
Chia-Man HUNG
Raymond LI
Yuxiang LI
Étienne SIX
Marc SZAFRANIEC
Bruno TAILLÉ

24 avril 2015

# Table des matières

# 1 Introduction

## 1.1 Coupe de France de Robotique 2015

L'objectif de notre PSC est de concourir à l'édition 2015 de la Coupe de France de Robotique. Il s'agit d'une compétition de robots autonomes, c'est à dire que les robots sont programmés à l'avance et ne reçoivent aucune commande pendant la compétition (ils ne sont pas téléguidés). Les épreuves auront lieu du 13 au 16 mai 2015 à La Ferté Bernard (72).


FIGURE 1 – Coupe de France de robotique 2013

Nous visons bien sûr la meilleure place possible : d'après les résultats de l'année dernière, nous considérons qu'arriver parmi les 32 premières équipes comme eux serait une très bonne performance. D'autant plus qu'avant ce PSC, aucun de nous n'avait fait de robotique.

L'aspect compétitif est un enjeu important de notre PSC puisque nous serons les représentants de l'École dans une compétition scientifique nationale. Il a contribué à notre motivation de donner le meilleur de nous-mêmes au cours de cette année de construction du robot. En effet nous disposons d'une dead-line fixe, avec un objectif clair : avoir un robot opérationnel pour le 13 mai.

### 1.1.1 Présentation

### 1.1.2 Modalité de l'épreuve

Les règles d'attribution des points ont été annoncées le 27 septembre 2014. En voici une description succincte : Chaque équipe est composée d'un ou deux robots indépendants. Pendant les 90 secondes d'un match, ils peuvent réaliser les 5 tâches différentes qui rapportent des points en fonction de leur difficulté. Le but est bien sûr d'avoir le plus de points possibles. Les moyens de gagner des points sont :

1. Rabattre des claps de la couleur de son équipe (5 points par clap)
2. Placer du pop-corn (représenté par des balles de type ping-pong) dans des gobelets et les ramener dans une zone valide, ou bien placer le pop-corn dans des bacs dédiés (1 point par pop-corn)
3. Dans une zone dédiée, monter des spots, composés de plusieurs pieds (des cylindres) de la couleur de l'équipe, et d'une balle de tennis au sommet (2 points par pied valide et 3 points bonus si le pied est dans un spot)
4. Placer deux tapis sur les allées des marches de la couleur de l'équipe (2 points par marche d'une allée recouverte, et 4 points bonus si le tapis recouvre les 4 marches d'une allée)
5. 15 points si l'un des robots se trouve au sommet des marches à la fin de l'épreuve

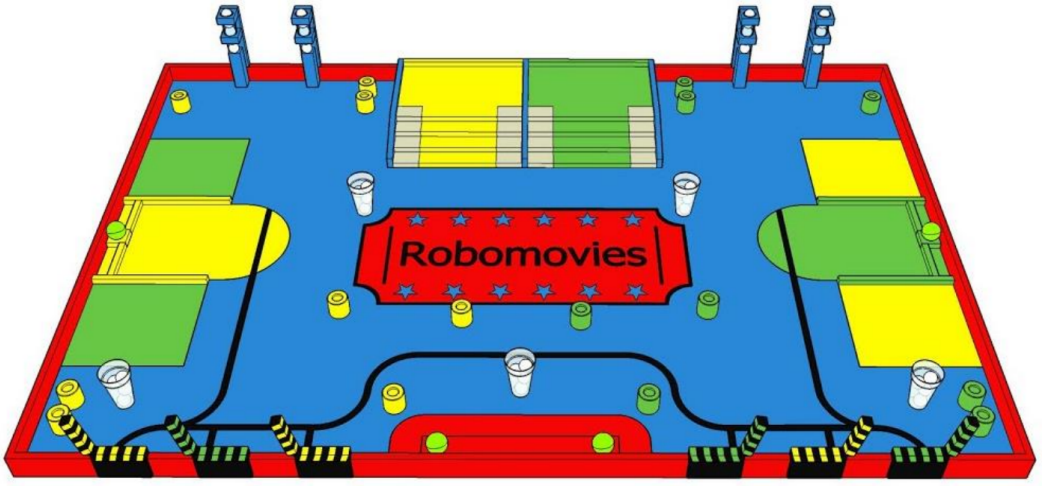



FIGURE 2 – Terrain du jeu

Les robots possèdent des contraintes matérielles assez strictes. Leur périmètre est limité. Le périmètre du robot principal ne doit pas excéder 1200 mm au moment du départ, et 1500 mm au cours du match (les robots peuvent se déployer). Pour le robot secondaire, les contraintes sont de 700 mm au moment du départ et 900 mm déployé. Le périmètre du robot secondaire est, comme on le verra, un de ses principaux challenges.

De plus, chaque robot doit comporter un arrêt d'urgence et une plateforme sur laquelle les équipes adverses peuvent poser une balise pour le repérer.

Pour participer à cette compétition nous avons aussi dû produire un document technique de présentation de notre robot afin de le faire homologuer. Les organisateurs surveillent en particulier le type de batteries et de capteurs laser que nous utilisons, pour des raisons de sécurité.

## 1.2 Notre équipe et objectif final

L'équipe participante à la coupe de France est formée de 14 personnes, réparties sur deux PSC, INF 09 et INF 13. Chaque PSC conçoit un robot. Nous sommes en charge du robot secondaire. Une telle répartition des tâches est possible car les deux robots doivent être indépendants, c'est à dire qu'ils doivent pouvoir fonctionner l'un sans l'autre. En début d'année nous avons décidé de la stratégie d'équipe : notre robot, le plus petit, devrait monter les marches de l'escalier et dérouler les tapis. Cependant en cours d'année nous nous sommes rendu compte que monter l'escalier était assez complexe et que la contrainte de périmètre rendait la tâche des tapis très difficile. Aussi nous avons décidé de nous concentrer sur les escaliers.

Comme nous l'avons évoqué plus haut, l'équipe ne partait pas de zéro, puisque chaque année un PSC participe à la Coupe. Les membres de l'équipe de l'an dernier, et en particulier David-Octavian IACOB nous ont formés lors du premier trimestre. En effet, aucun de nous n'avait de compétences en robotique, et en particulier en électronique. Il nous a présenté le matériel à notre disposition, notamment le robot fraiseuse Charlie-robot, et nous a conseillé en mécanique, électronique et informatique.

Cependant c'est la première année que l'équipe est composée de deux robots. Aussi l'autre PSC a pu reprendre le robot de l'an dernier, en particulier sa base roulante et ses cartes électroniques, pour l'adapter aux nouvelles tâches. Pour nous, le travail a été plus complexe car nous avons dû tout réaliser sans pouvoir reprendre les plans de l'an dernier à cause des contraintes de périmètre d'une part, et de notre tâche particulière d'autre part. En effet la base roulante des X2012 est conçue pour rouler à plat et ne peux gravir un escalier.

Au sein de notre PSC, nous nous sommes réparti les rôles comme suit :

| **Partie** | **Membres** |
|---|---|
| Mécanique | Bruno TAILLÉ et Alexis CLARIOND |
| Électronique | Chia-Man HUNG et Marc SZAFRANIEC |
| Informatique | Etienne SIX, Yuxiang LI et Raymond LI |

Cependant au cours de l'année il est vite apparu que les parties étaient loin d'être indépendantes. Pour réaliser les cartes électroniques il fallait le plan mécanique du robot (nombre de servomoteurs par exemple). Pour tester ces mêmes cartes, il fallait que la partie mécanique du robot soit prête. Enfin , pour tester et ajuster les algorithmes, il fallait que les cartes électroniques soient fonctionnelles. Chacun a donc aidé les membres des autres parties lorsque cela était nécessaire, notamment lors du montage final du robot. Cette entraide était nécessaire pour respecter l'échéancier que nous nous étions fixé. Enfin d'autres tâches ont dû être distribuées, comme la fabrication d'un escalier bien dimensionné pour effectuer des tests ou la rédaction du dossier technique à remettre lors de l'inscription au concours.

# 2 Mécanique

## 2.1 Cahier des charges restraint

En tant que robot secondaire, notre robot est soumis à des conditions très strictes, voici une partie des cahiers des charges.

| | **Fonctions de service** | **Critères** | **Niveau** | **Flexibilité** |
|---|---|---|---|---|
| FP1 | Se déplacer automatiquement | Précision | ±1cm | Aucune |
| | | Vitesse | 30cm/s Max | Négociable |
| | | Capteurs | 4 Min | Négociable |
| | | Encodeurs | 2 | Aucune |
| | | Communication | Non | Aucune |
| FP2 | Monter sur l'escalier | Durée | 45s Max | Aucune |
| | | Centre de gravité | 10cm Max | Négociable |
| FP3 | Déposer le tapis | Précision | Dans la zone | Aucune |
| | | Nombre | 2 | Négociablee |
| FC1 | Assurer l'alimentation | Batteries | 2 | Aucune |
| FC2 | Respecter les dimensions | Hauteur | 35cm Max | Aucune |
| | | Non déployé | 700mm Max | Aucune |
| | | Tout déployé | 900mm Max | Aucune |
| FC2 | Respecter les règles de sécurité | Balise | Equipé | Aucune |
| | | Collision | Aucune | Aucune |
| | | Arrêt d'urgence | Equipé | Aucune |
| | | Sac de batterie | Equipé | Aucune |

Parmi toutes ces demandes, deux critères sont prioritaires : le périmètre et le centre de gravité. On verra dans la suite notre solution pour bien respecter ce cahier des charges.

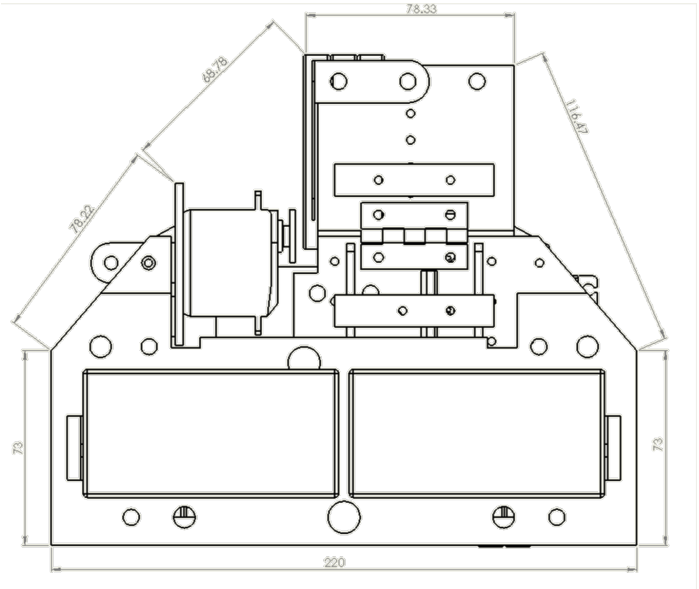



FIGURE 3 – Périmètre du robot déployé 710mm

## 2.2 De SolidWorks à l'usinage

Afin de respecter ce cahier des charges, nous avons décidé de commencer par la modélisation SolidWorks, un logiciel de Conception Assistée par Ordinateur en 3D, grâce à laquelle on peut facilement mesurer et modifier les dimensions des différents composants. Puis nous sommes passés à l'étape d'usinage et d'assemblage tout pièce par pièce. La suite de cette partie sera illustrée avec à la fois le modèle 3D de SolidWorks et les photos de notre robot pour expliquer les détails de cette base roulante.

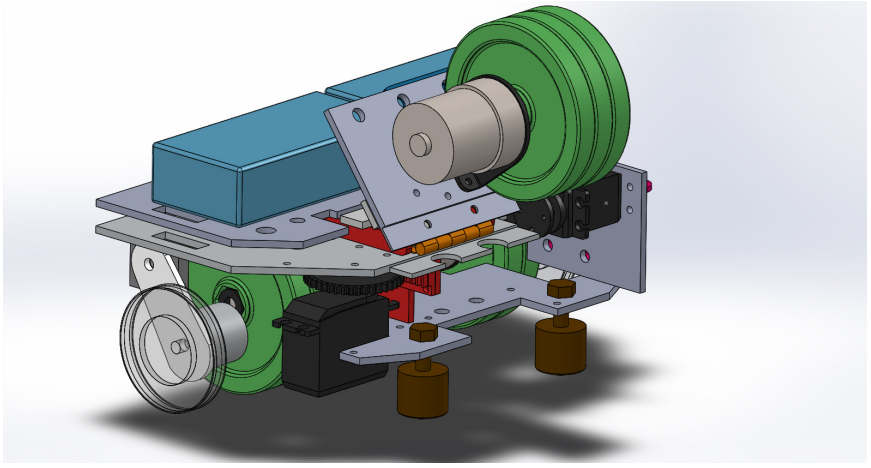


FIGURE 4 – Base roulante version finale

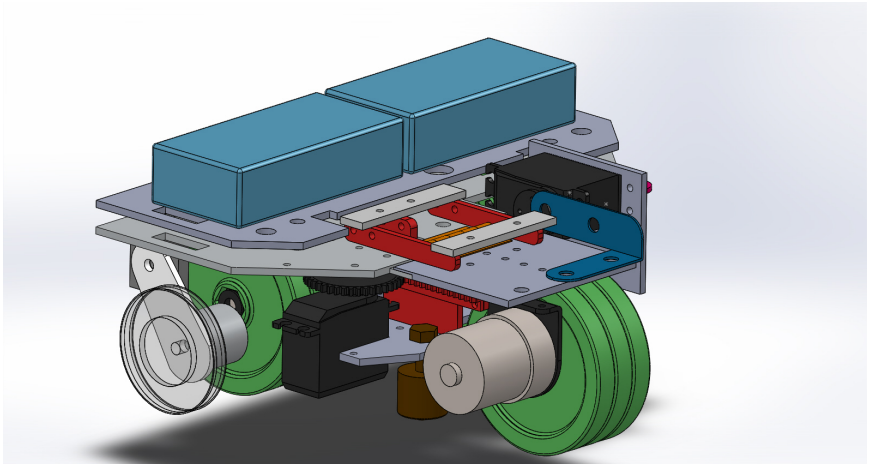


FIGURE 5 – Base roulante déployée version finale

### 2.2.1 Base roulante

Pour monter les escaliers dont les marches ont une hauteur de 2,2 cm, nous avons d'abord pensé à un système à quatre roues motrices de diamètre bien supérieur. Les contraintes de périmètre

imposées nous ont poussés à opter pour une base en tricycle avec des roues de 7,3 cm de diamètre.

L'écart entre les deux axes de 11 cm a été calculé pour qu'à chaque instant toutes les roues soient en contact avec une marche.

Malgré cette disposition réduite et une conception aussi compacte que possible, il n'était pas possible de faire tenir tous les éléments en respectant la contrainte d'un périmètre inférieur à 70 cm pour le robot non déployé. De plus, l'utilisation d'une troisième roue à axe fixe empêchait toute rotation. C'est pourquoi nous avons décidé de monter cette roue sur une charnière qui peut se déployer à l'aide d'un servomoteur puis être bloquée par un deuxième. Cela permettait d'augmenter le périmètre autorisé à 90 cm une fois déployé tout en respectant la contrainte de périmètre non déployé.

Avant la conception de ce mécanisme de déploiement nous avons réalisé un premier prototype de base roulante en un bloc pour nous assurer que le robot pourrait monter les marches une fois déployé et nous avons effectué les premiers tests.

Après une conception sur le logiciel sur Solidworks, nous avons donc usiné le premier prototype de la base roulante simplement composé d'une plaque d'aluminium et doté de ses trois roues motorisées. Nous avons effectué les premiers tests en alimentant directement les moteurs avec un générateur de courant, la base positionnée face à une réplique de l'escalier que nous avions fabriquée.

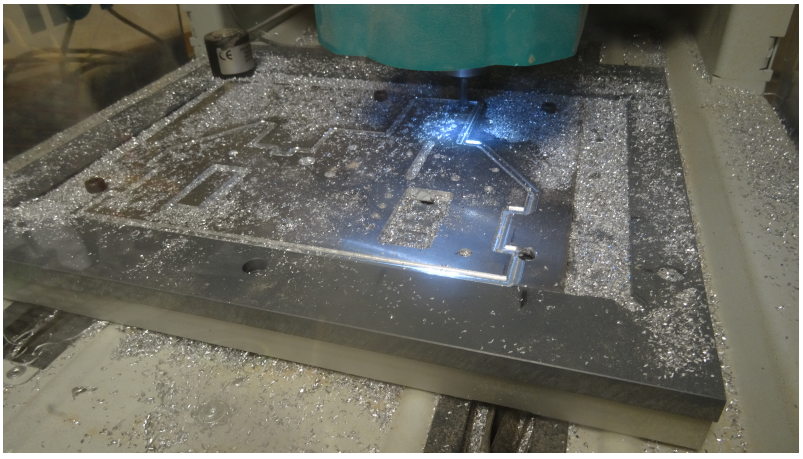


FIGURE 6 – Usinage de la base roulante

### 2.2.2 Premier test

En utilisant le robot comme nous l'avions imaginé, avec la roue seule devant, il parvenait à monter mais au-delà d'une certaine vitesse basculait en arrière, le mouvement semblait assez anarchique. Cela était en partie dû au fait que le chargement était faible et que le centre de gravité était proche de l'axe comportant les deux roues.

Nous avons donc décidé d'inverser le sens de rotation des roues et celui du robot. Le résultat était concluant : non seulement le robot ne basculait plus, mais le fait que les deux roues entraient en contact avec la marche simultanément assurait qu'il montait bien perpendiculairement aux marches. La montée s'est faite de manière régulière et la base est restée stable. Nous avons pu

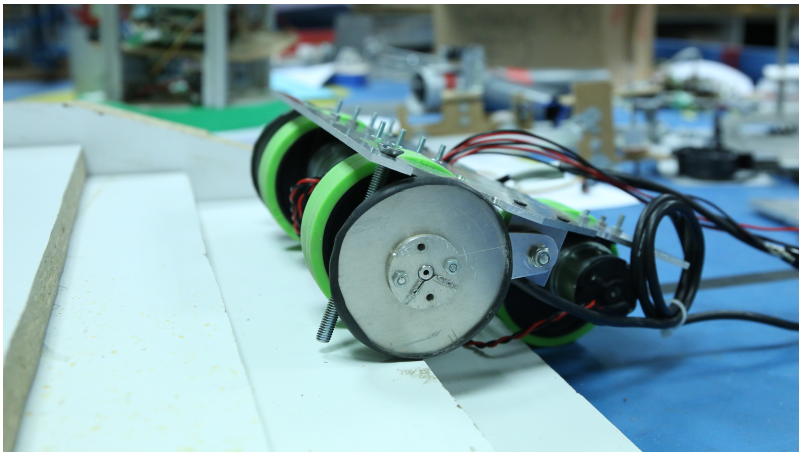


FIGURE 7 – Test sur la première base roulante

mesurer une tension minimale à appliquer de l'ordre de 2,5V pour que la montée s'effectue, ce qui était tout à fait raisonnable. De plus nous avons vérifié que la troisième roue motrice était malheureusement bien nécessaire à la montée.

Cependant, puisque nous avions initialement pensé à une montée dans l'autre sens, la fixation ultérieure des roues codeuses récupérées sur le robot de l'équipe X2013 a posé problème lors de la montée, il fallait inverser leur sens de montage et nous avons décidé de réduire leur taille pour éviter qu'elles ne bloquent la montée.

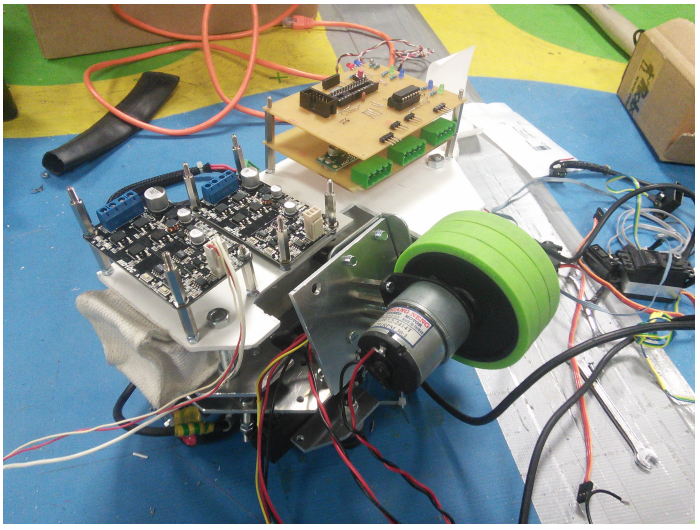


FIGURE 8 – Base roulante

Nous avons donc usiné une nouvelle base en implémentant directement les modules supplémentaires à partir de début février, travaillant désormais sur une version finale du robot. Nous verrons dans la suite que cette nouvelle base est compatible avec les mécanismes dont nous avons besoin et elle a pour particularité d'être à la fois compacte et facile à démonter et assembler.

*Composants utilisés et détails techniques de la base roulante :*

| **Composant** | **Quantité** | **Paramètres** |
|---|---|---|
| Roue | 7 | Marque : BaneBots<br>Diamètre extérieur : 7,3 cm<br>Epaisseur : 1,0 cm |
| Moyeu de roue | 3 | Marque : BaneBots<br>Caractéristiques : 1,27 cm Hexagonal |
| Moteur | 3 | Marque : Pololu<br>Caractéristiques : 12V DC, 350 rpm 29 :1 |
| Support de moteur | 3 | Marque : Lynxmotion |

### 2.2.3 Système de pliage

Après la base roulante, abordons les autres mécanismes du robot. La roue arrière est fixée sur une plaque qui pivote sur la base roulante grâce à une charnière. Deux servomoteurs sont nécessaires : le premier, directement relié à la plaque, la fait pivoter et le second fait coulisser un loquet qui assure que les deux parties sont rigidement liées.

Avec le poids du moteur et des trois roues, il faut nécessairement mettre une charnière pour soulager le bras du servomoteur, qui doit seulement servir à faire pivoter la plaque.

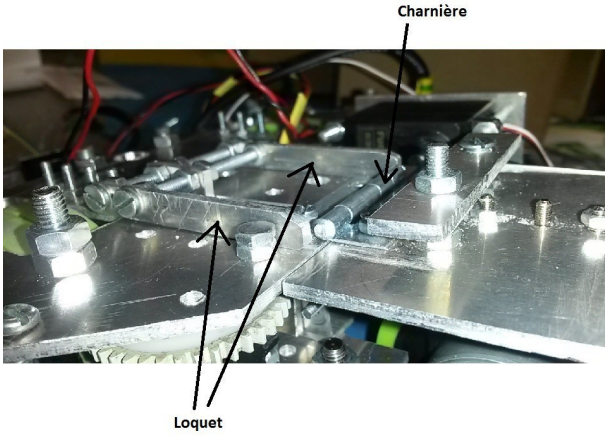



FIGURE 9 – Roue dépliée

La difficulté est d'aligner l'axe du servomoteur avec celui de la charnière. Nous avons fait le choix de d'abord fixer la charnière, pour ensuite ajuster le placement du servomoteur. Lors de premiers tests, le fait que les deux axes n'étaient pas parfaitement alignés provoquait une légère torsion du bras du servomoteur, augmentant l'effort à fournir pour replier la plaque et le servomoteur ne supportait pas. Ayant rectifié l'axe du servomoteur, il peut désormais soulever le poids de la roue arrière mais avec une certaine difficulté.

Rappelons que nous baisserons la roue pour monter l'escalier. Ne voulant pas prendre le risque d'user, voire de casser le servomoteur mentionné ci-dessus, nous avons mis en place un système de loquet : une fois le loquet en place, on pourra éteindre le premier servomoteur.

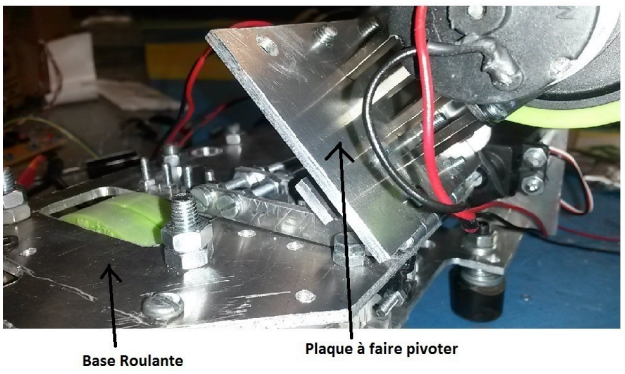



FIGURE 10 – Roue repliée

### 2.2.4 Système de blocage

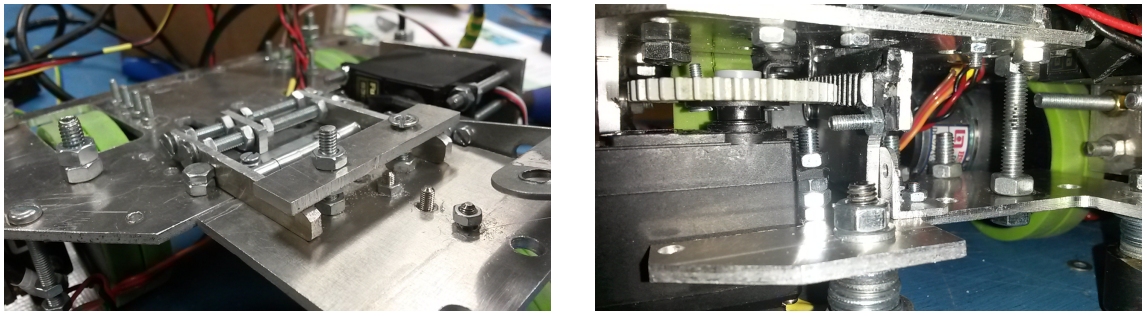


FIGURE 11 – Système de blocage

Nous avons utilisé un engrenage et une crémaillère pour contrôler le loquet avec un servomoteur dont l'axe de rotation est vertical. Le loquet est formé de trois étroites plaques d'aluminium reliées par trois vis, et un morceau qui se prolonge en dessous de la base roulante, sur lequel on a fixé une crémaillère. Ce loquet va venir se coincer entre la plaque qui supporte la roue et un morceau d'aluminium que nous avons fixé à cette plaque. Nous avons aussi pensé à simplement bloquer la plaque par le bas avec un morceau d'aluminium fixé à la base roulante, et par le haut avec le loquet ; mais avec le moteur, les différentes pièces de guidage du loquet et la charnière, l'espace n'était pas suffisante.

L'espace dans lequel vient s'insérer le loquet doit laisser assez peu de jeu pour que la roue ne branle pas trop, mais assez pour que le loquet puisse rentrer. Aussi, nous avons donné au bout du loquet une forme pointue. De cette manière, même avec un angle imparfait entre les deux plaques, le loquet peut rentrer dans l'interstice. Concernant le rayon de l'engrenage, il a fallu le choisir assez grand pour que les 180°de rotation du servomoteur permettent d'effectuer le déplacement voulu, mais assez petit pour limiter l'effort à fournir par le servomoteur.

Le loquet est guidé de manière à ce qu'il ne puisse que coulisser dans l'axe voulu. Pour cela, nous

avons fixé un morceau d'aluminium sur la plaque inférieure, et deux vis sur la base roulante.

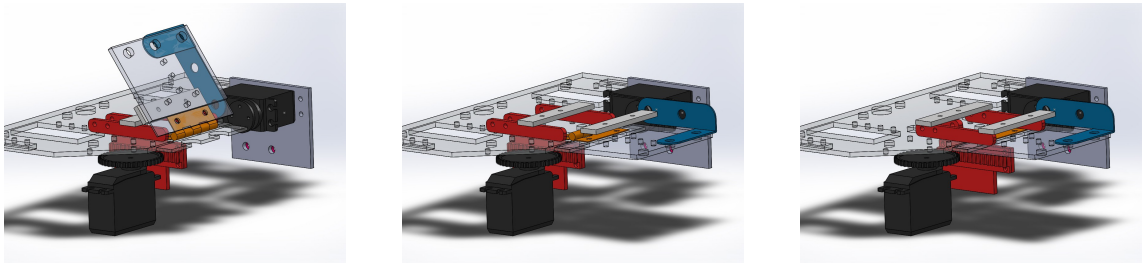


FIGURE 12 – Système de blocage en SolidWorks

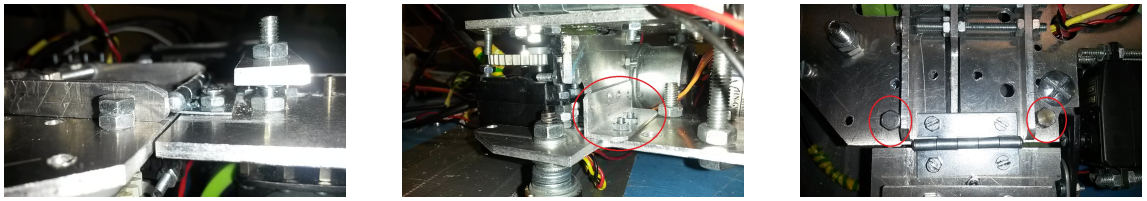


FIGURE 13 – Système de blocage : forme arrondie et les guidages

La difficulté est de guider le loquet, sans pour autant qu'il n'y ait trop de frottements, et ensuite de placer adéquatement le servomoteur, pour que la crémaillère et l'engrenage soient en contact. En particulier, les différentes vis de la base roulante ont posé des problèmes lorsqu'elles bloquaient le mouvement du loquet.

Nous huilerons certainement les zones de frottement si jamais elles posent problème et empêchent le déplacement du loquet.

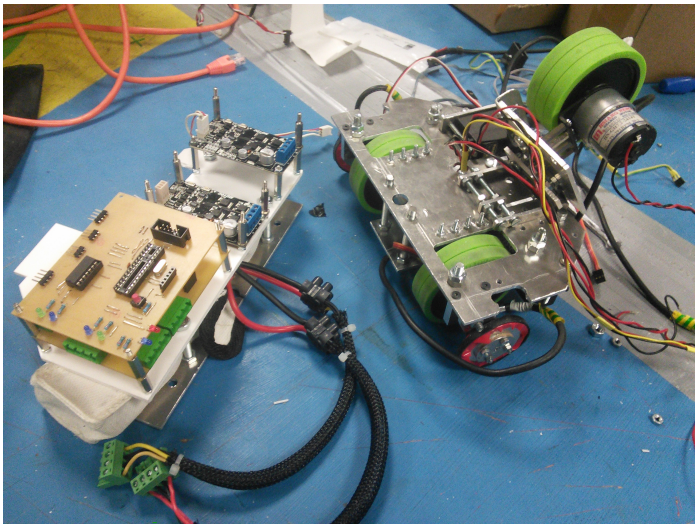


FIGURE 14 – Les étages supérieurs (gauche) et la base roulante

## 2.3 Etages supérieurs

Une fois finie la base roulante, il ne reste plus qu'a mettre toutes les cartes électroniques et faire passer les fils électriques dans le robot. Avec l'expérience des anciens, nous avons prévus plusieurs endroits sur la base pour faire passer les cables et nous avons utilisé toute une plaque comme nouveau support des cartes, ce qui fait que nous pouvons séparer la partie mécanique et la partie électronique en enlevant simplement trois écrous de fixation.

# 3 Electronique

## 3.1 Description générale

La partie électronique consiste à réaliser les cartes électroniques et programmer en Arduino sur les cartes programmables, ce qui permet de relier la partie mécanique, les différents composants mécanique du robot (moteurs, servo-moteurs et autres manipulateurs) ainsi que les capteurs qui aide au positionnement et à l'évitement des obstacles (capteurs de distance, roue codeuse), et la partie informatique, le cerveau du robot. La conception des cartes a été rapide, s'appuyant sur celles utilisées par l'équipe de l'année précédente mais leur réalisation nous a fait perdre un temps précieux car certaines ne fonctionnaient pas après leur fabrication. Il a fallu diagnostiquer les problèmes et les résoudre ce qui nous a mis en retard sur notre programme et nous a poussé à abandonner la tâche de déploiement du tapis.

FIGURE 15 – Fabrication manuelles des cartes électroniques

Nous commencerons par une explication générale de l'ensemble, à savoir comment les cartes sont reliées entre elles. Nous expliquerons ensuite ce que fait chaque carte et comment les messages leur permettent de communiquer. Dans chaque sous-partie, nous expliquerons les principes de fonctionnement et les idées de codes en Arduino intégrés dans chaque carte. La dernières sous-partie, à savoir les microcontrôleurs, est mise à part puisqu'ils apparaissent sur toutes les cartes électroniques programmables.

### 3.1.1 Flux de commande - Bas niveau

Le robot utilise un système de cartes électroniques qui utilisent des microcontrôleurs ATMEL ATMega 328p programmées dans l'environnement de programmation Arduino. Le langage d'Arduino est construit autour des librairies C `avr-gcc` et permet un niveau relativement bon de flexibilité. Pour les opérations qui n'ont pas encore été implémentées dans le langage Arduino, le code permet toujours d'utiliser les fonctions et les variables des librairies avr-gcc ainsi que des opérations implémentées directement sur les registres du microcontrôleur.

Au niveau de l'architecture générale des cartes, le robot est contrôlé par le pcDuino. Le messages

provenant du pcDuino sont dispatchés par l'UART aux cartes électroniques programmables, à savoir la carte d'acquisition, la carte de mécanisme et la carte de contrôleur de moteur. Chaque carte programmable traite un message à la fois et dès qu'elle le reçoit. La carte d'acquisition transmet des messages aux quatres capteurs de distance, toujours un message à la fois et à un des quatres capteurs. Ce capteur lui répond par un message et la carte le transmet au pcDuino. La carte de mécanisme sert à transmettre des messages au servo-moteur et au moteur de la roue arrière. La carte de contrôleur de moteurs transmet des messages à la roue codeuse et au moteur des roues avant. La roue codeuse répond à la carte et le message est monté jusqu'au pcDuino. Les LEDs sont ajoutées sur les cartes pour vérifier leur bon fonctionnement. Nous allons décrire le fonctionnement de ces cartes dans la suite.

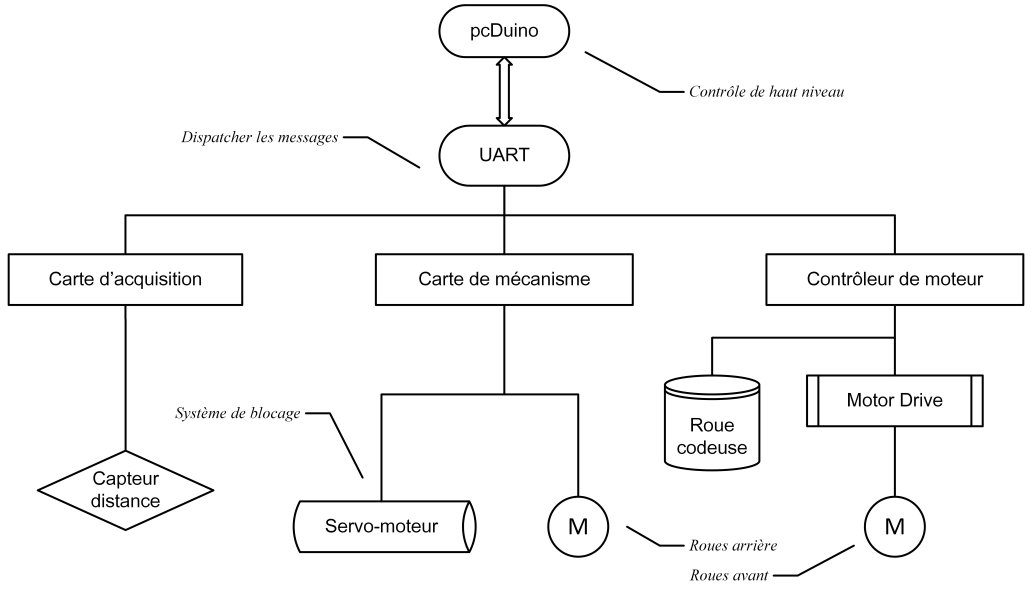



FIGURE 16 – Circuit électrique

### 3.1.2 pcDuino et UART

La carte pcDuino fait office de mini-ordinateur équipé d'Ubuntu 12.04, et ainsi peut se programmer et s'utiliser de la même façon. Elle utilise un processeur ARM Cortex-A8 de 1GHz et 1Gb de RAM.

Sa taille réduite la rend particulièrement utile lorsqu'il s'agit de respecter les contraintes de dimensions du robot. De plus, sa consommation faible (ne dépassant jamais 5-6W) ainsi que son alimentation à 5V nous facilitent la tâche.

Afin d'améliorer encore son efficacité nous y avons installé les applications Linux de transfert de fichiers à distance `ssh`, qui permet le contrôle de la carte via un ordinateur relié au même réseau Wi-Fi, et `sshsf`, permettant l'accès sécurisé aux fichiers présents sur la carte. Ceci est complété par un script permettant la compilation rapide du logiciel sans recours à un IDE.

L'ensemble permet une meilleure productivité dans les tests et le développement du logiciel de contrôle du robot. Enfin, la communication entre la pcDuino et les cartes électroniques se fait

par USB, ce qui la rend facilement remplaçable par une autre carte voire par un ordinateur. Le processus de communication par UART sera détaillé en 3.2.

### 3.1.3 Carte d'acquisition

Les cartes d'acquisition ont pour tâche de traiter les signaux d'entrée des capteurs en les échantillonnant qu'ils soient de nature analogique (sur 10 bits, 6 canaux) ou numérique (1 bit, 9 canaux). Les signaux analogiques sont en plus filtrés à raison de 8 échantillons consécutifs en moyenne, qui sont communiqués à l'ordinateur central sur sa demande.

Cette carte n'interprète pas elle-même les signaux analogiques pour ne pas avoir à réaliser des calculs mathématiques complexes comme des interpolations sur ces cartes. Ce sont des fonctions implémentées sur la carte pcDuino qui s'en chargent.

### 3.1.4 Carte de contrôle de moteur et de servo-moteur

La carte d'asservissement de moteurs et la carte mécanique sont deux pièces maîtresses de l'électronique du robot : elles réagissent aux signaux fournis par les roues codeuses, permettant de compter et d'enregistrer leur nombre de rotations, et par suite calculer constamment la position du robot sur la table de jeu. Cette position est stockée dans des coordonnées de type $(x, y, \alpha)$ afin de connaître à la fois la position du robot et sa direction, et transmise à l'ordinateur central. Réciproquement, les cartes reçoivent des commandes de mouvement, limitées au déplacement en ligne droite et à la rotation sur place.

Idéalement, la carte mécanique se charge des servo-moteurs et la carte d'asservissment des moteurs. Nous avons prévus trois Motor Drive pour contrôler les trois roues. Cependant, notre carte d'asservissement n'a que deux ports pour les moteurs et la roue arrière sera commandée via la carte mécanique.

De plus, pour assurer la précision du mouvement et de la position, la carte d'asservissement utilise deux algorithmes PD (Proportionnel Dérivée) sur la vitesse, puis sur la position pour la fin du mouvement et le stationnement, d'où le nom de l'asservissement.

La carte mécanique, contrôleur des composants mécaniques, peut gérer dix composants dans trois modes de fonctionnement différents :

— **Mode PWM** : Signal de contrôle PWM de fréquence 1kHz et de précision 8-bit qui permet par exemple de contrôler la puissance d'un moteur.
— **Mode Servo** : PWM de période 20 ms et un intervalle où le signal est au niveau haut qui varie entre 0.5 ms et 2 ms en fonction du type de servomécanisme et de l'angle désiré. Ce mode sert évidemment à contrôler des servomécanismes.
— **Mode 1/0** : Il représente un signal qui reste soit haut (+5V) soit bas (0V) pour contrôler des composantes simples dont la puissance ne doit pas varier, notamment l'éclairage.

Chacun des dix canaux peut être alimenté à 5 V (servomécanismes) ou à 14-16V par quatre cellules Li-Po. Le signal de commande qui sort de la carte de commande est de faible intensité (10-20 mA) et doit être amplifié par des transistors MOSFET avant d'alimenter des composants de puissance (moteurs, électroaimants). Les servomécanismes ont déjà tous les circuits électroniques dont ils ont besoin pour fonctionner correctement.

Dans notre cas, la roue arrière sera commandée via un port en mode PWM et un autre en mode 1/0 pour le sens de rotation.

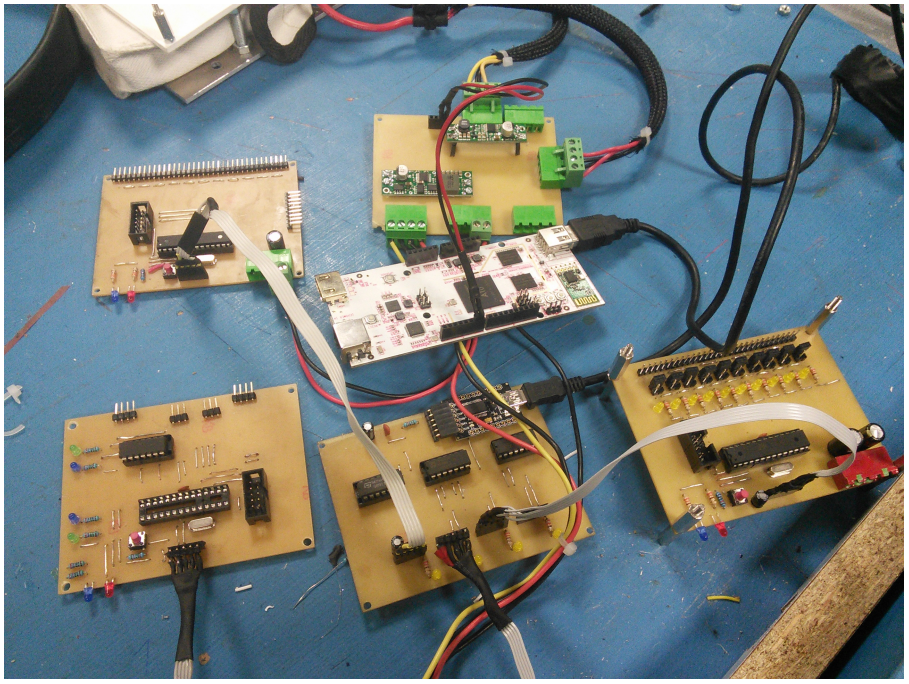


FIGURE 17 – Interconnexion des cartes électroniques

### 3.1.5 Microcontrôleurs

Les microcontrôleurs sont programmés dans l'environnement de programmation Arduino construit autour des librairies `avr-gcc`. Nous avons utilisé ATmega 328p nu avec un programmeur USBtinyISP au lieu de carte Arduino pour une simple raison de coût et cette méthode nous paraissait plus instructive que d'utiliser une carte déjà faite. L'IDE d'Arduino n'est pas conçu pour travailler sur une grande échelle, néanmoins, à l'aide du plugin Stino pour l'éditeur de texte Sublime Text, qui est une application gratuite, nous sommes parvenus à les programmer sans problème.

En général, une Arduino a plusieurs types d'entrées : les entrées numériques, qui détectent tout signal électrique (de type « tout ou rien ») renvoyé par un capteur et les entrées analogiques qui sont capables de lire la valeur d'une tension renvoyée par un capteur, comme un potentiomètre. Les entrées numériques de l'Arduino sont au nombre de quatorze. Leur nom va de D0 à D13, mais les deux premières (D0 et D1, aussi appelées Tx et Rx) peuvent aussi être utilisés lors de la communication série avec un ordinateur. Sur une Arduino méga, il y a 54 ports numériques. Les entrées analogiques sont moins nombreuses, elles sont au nombre de six allant de A0 à A5. Nous n'avons pas besoin de tous ces ports, donc il était possible utiliser le microcontrôleur tout nu et ajouter les périphériques à notre gré.

Pour la carte d'acquisition, la carte de mécanisme et la carte de contrôleur de moteur, le code a une structure commune comme dans toutes les cartes d'Arduino. Il contient deux fonctions essentielles : `setup()` et `loop()` . `setup()` permet d'initialiser toutes les fonctions et les ariables,

à savoirs les interruptions hardware et les pins d'entrée et de sortie. `loop()` contient la boucle principale d'exécution du logiciel. Nous avons créé des fonctions spécifiques à chaque carte comme `doSerial()` pour recevoir le signal et remettre l'état de chaque PIN. Un simple switch a été utilisé pour traiter des commandes diverses et variées venant du pcDuino.

## 3.2 Protocole de communication

Le protocole de communication a été réalisé par l'équipe de l'année passée, et nous n'en avons pas changé, étant donné sa portabilité. La transmission des messages est assurée par la carte UART et les PIN TX, RX sur chaque microcontrôleur. On utilise la librairie d'Arduino pour transmettre les messages.

La communication entre le pcDuino et les cartes, dont les canaux de communication sont branchés en parallèle, s'effectue de façon sérielle. Le protocole de communication a donc pour but de vérifier que chaque carte, et elle seule, reçoive l'instruction qui la concerne, et que le logiciel du pcDuino puisse reconnaître de quelle carte proviennent les données qu'il reçoit.

Pour cela, si les paquets de données du protocole sont de longueur variable, ils commencent toujours par 4 bytes qui ont pour but de définir la carte destinataire, la commande transmise et son nombre de paramètres :

| **Byte** | **Fonction** | **Valeur** |
|---|---|---|
| Byte 1 | Type de carte destinataire | à partir de 128 |
| Byte 2 | Index de carte destinataire* | à partir de 128 |
| Byte 3 | Numéro de commande | à partir de 128 |
| Byte 4 | Nombre de bytes de paramètres de la commande | 128 + le numéro de bytes |

* Dans le cas de plusieurs cartes identiques (comme les cartes d'acquisition)

Le nombre de bytes de paramètres est supérieur ou égal au nombre de paramètres. Supérieur notamment dans le cas où les variables sont codées sur 16 bits. Il faut alors envoyer deux bytes pour chaque variable.

A partir du moment où tous les cartes ont reçu ces quatre bytes, elles essayent de valider la commande. Si la commande est bien adressée à une des cartes, elle va attendre que les bytes de paramètres attendus arrivent dans son buffer d'entrée pour ensuite lire les variables et exécuter la commande. Dans le cas contraire, elle ignore les bytes envoyés ensuite et vide son buffer d'entrée.

Le transfert de données dans l'autre sens, c'est à dire des cartes vers le pcDuino, utilise le protocole, les deux premiers bytes définissant cette fois la carte émettrice. Cependant, l'envoi de données à partir d'une carte est limité au cas d'une demande de la part du pcDuino, pour éviter les pertes de données.

## 3.3 Capteurs

Nous avons utilisé quatres capteurs de distance, qui se situent sur l'avant, l'arrière, l'avant à gauche et l'avant à droite respectivement, et deux roues codeuses, l'une à gauche et l'autre à droite.

### 3.3.1 Capteur de distance - Sharp

Un capteur de distance sert à l'évitement des obstacles. Il utilise un principe optique pour mesurer la distance : un rayon lumineux infra-rouge collimaté est émis, va se réfléchir sur un objet présent dans le champ de détection et vient frapper en retour une bande de récepteurs à l'intérieur du capteur Sharp permettant d'évaluer la distance. Plus précisément, il s'agit du principe de la triangulation. A l'aide d'un détecteur linéaire sensible à la position, le capteur de distance mesure la distance à l'objet indépendamment de la quantité de lumière réfléchie. La lumière renvoyée par l'objet $P_1$ est représentée sur la ligne grâce au point $P_1$. C'est ainsi que le capteur détermine le signal de distance. De la même façon, pour la distance à l'objet $P_2$, la lumière atteint le détecteur à un autre endroit $P_2$.

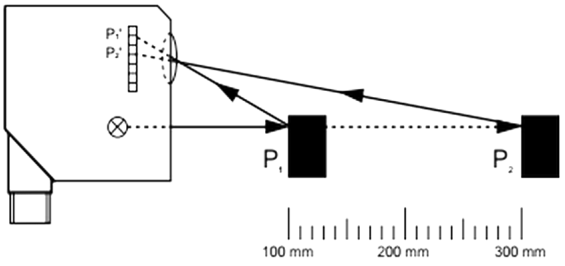
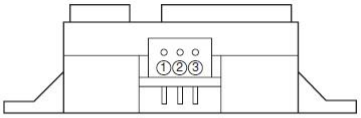



FIGURE 18 – Principe de fonctionnement du capteur de distance

Au niveau des caractéristiques spécifiques de nos capteurs de distance, ils opèrent sur une plage de distance de 10cm à 80cm. Un capteur est lié à une entrée analogique (tension de sortie entre 0V et 2,4V) et fonctionne sous 5V.

Le brochage est standard avec trois broches : $V_{cc}$ = +5V, GND = 0V, tension de sortie $V_0$.




| PIN | SIGNAL NAME |
|---|---|
| ① | $V_O$ |
| ② | GND |
| ③ | $V_{CC}$ |

FIGURE 19 – PIN du capteur de distance SHARP

La sortie $V_0$ est une fonction non-linéaire de la distance de l'obstacle. Cette information est ensuite envoyée au microcontrôleur correspondant et est tranmise jusqu'au pcDuino pour être traduit en distance métrique.

Afin d'approximer cette fonction, nous avons effectué un test d'un capteur Sharp avec la carte mbed NXP LPC1768 et un lecteur sérial implémenté en Python. Nous avons pu avoir le résultat suivant (les abscisses sont les distances en centimètre et les ordonnées sont la tension de sortie en Volts) :

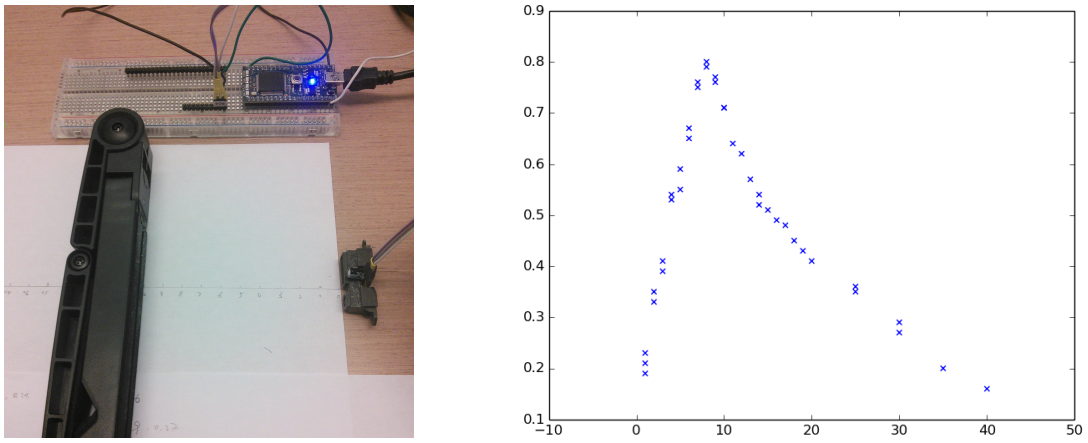



FIGURE 20 – Test sur le capteur Sharp

### 3.3.2 Roue codeuse

Une roue codeuse sert à déterminer la vitesse et le sens de rotation du robot. Elle possède quatre pattes : deux pour l'alimentation (il y a une LED à l'intérieur qui nécessite du courant pour s'allumer) et deux pour la sortie des données. Ces dernières sont mises à disposition sur des pattes nommées A et B. Le principe de base est le suivant : la LED intégrée éclaire des capteurs sensibles à la lumière au travers de trous pratiqués dans un disque physiquement solidaire de l'axe manoeuvrable par l'utilisateur.

S'il n'y avait qu'un seul capteur sensible à la lumière, on disposerait déjà d'un système capable de produire des impulsions électriques à un rythme qui dépendrait de la vitesse de rotation de l'axe. Par exemple, si le disque intercalé entre LED émettrice et récepteur photosensible possèdait 200 trous régulièrement espacés sur son pourtour, on obtiendrait 200 impulsions pour une rotation complète de l'axe. Ce serait intéressant, mais ne permettrait en aucun cas de savoir dans quel sens l'axe évolue. Pour déterminer le sens de rotation, un deuxième capteur photosensible est installé à une certaine distance du premier, de telle sorte que les informations délivrées par les deux capteurs - qui reçoivent tous deux de la lumière mais à des moments différents - soient décalées dans le temps. Ainsi, il suffit de prendre une des deux sorties comme référence et de regarder ce qui se passe sur l'autre.

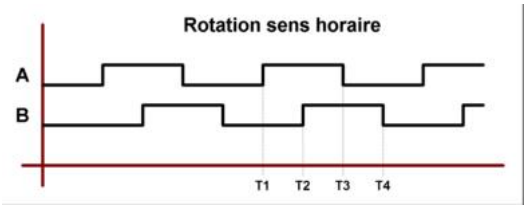



FIGURE 21 – Roue codeuse dans le sens horaire

Dans le premier exemple, pour quatre moments différents :

— $T_1$ : A = 1 et B = 0
— $T_2$ : A = 1 et B = 1
— $T_3$ : A = 0 et B = 1
— $T_4$ : A = 0 et B = 0

On remarque que A est en avance par rapport à B. On en déduit que la rotaiton est dans le sens des aiguilles d'une montre.

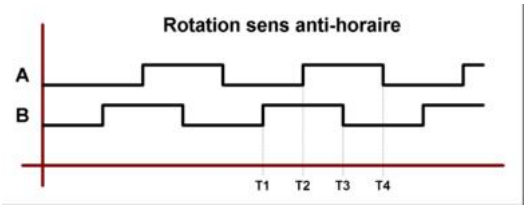



FIGURE 22 – Roue codeuse dans le sens trigonométrique

Dans le deuxième cas, B est en avance par rapport à A, donc la rotation est dans le sens inverse.

# 4 Informatique

La partie informatique permet au robot d'exécuter une série de tâches automatiquement. Elle fait la jonction entre la partie électronique et les programmes, un algorithme développé pour un autre robot est facilement transporté si le code est bien rédigé. Dans cette partie, on expliquera le fonctionnement du robot et une bonne structure permettant de réutiliser le code.

## 4.1 Fonctionnement global du robot

### 4.1.1 Flux de commande - Haut niveau

Pour que le robot puisse prendre connaissance de ses composants, nous avons utilisé la structure ci-dessous pour commander le robot.

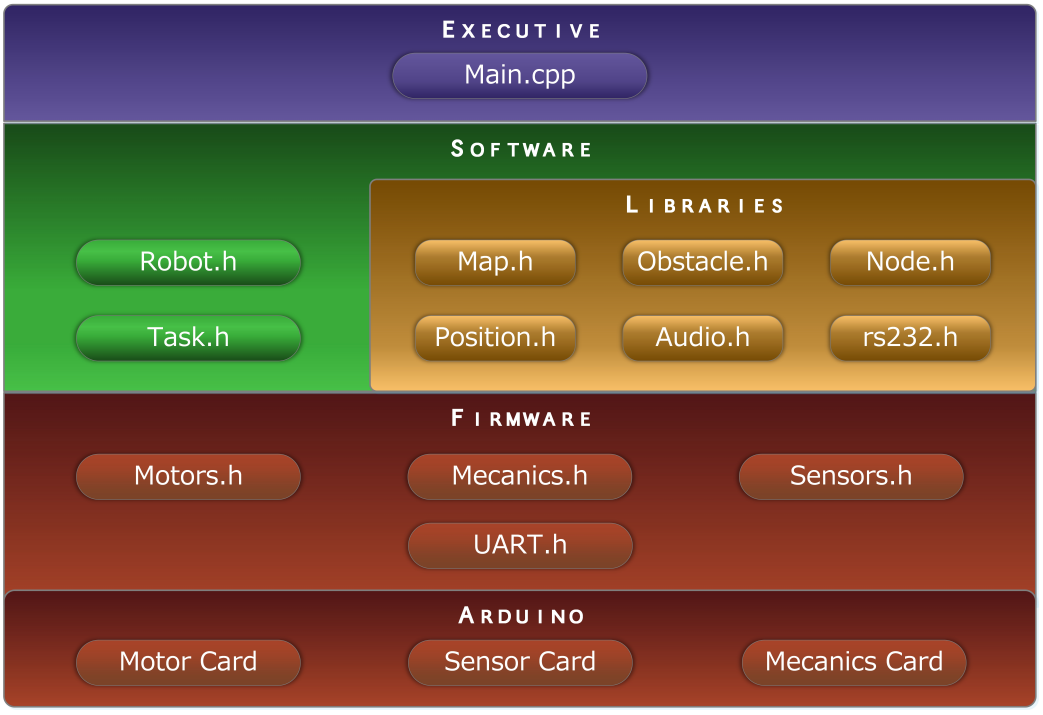



FIGURE 23 – Structure du code du robot

Les codes se divisent en trois principales parties : l'exécution, le logiciel et la communication avec les cartes électroniques.

Dans la partie Execution, on initialise le robot et le déclenche, toute la partie d'initialisation se fait en cette période, on y a mis les constructeurs de presque toutes les classes.

La partie Logiciel est très diverse, nous avons utilisé une classe `Robot` pour représenter notre robot avec ses bras, ses cartes et ses moteurs. La classe `Task` permet d'unifier la présentation des actions et est très simple, car les actions de notre robot sont relativement simples. Les autres

classes fournissent certains services, `Map` pour la navigation, `rs232` pour la communication et `Audio` pour jouer de la musique. Mais comment effectuer réellement une action dans `Task` ?

Pour exécuter une tâche, il faut savoir communiquer avec les moteurs et les capteurs, d'où l'utilité de la partie firmware. Nous avons utilisé la classe `UART` pour communiquer avec la carte électronique et 3 autres classes pour spécifier le type de commande. Le seul lien avec la réalité se trouve dans `UART`.

Les codes sont entièrement modulables avec cette structure qui fait descendre une commande couche par couche jusqu'à toucher la partie physique.

### 4.1.2 Robot multithread

Le robot est implémenté comme un programme multithread pour qu'il puisse s'arrêter au bout de 90 secondes. Un thread principal s'exécute lors du démarrage, tandis que le thread chargeant des tâches se déclenchera manuellement pour signaler le début du match. Le thread action tourne en boucle pour essayer d'exécuter le plus de tâches possible et il sera interrompu par le thread principal une fois les 90 secondes découlées.

## 4.2 Motion Planning

La seule difficulté de la partie informatique est d'implémenter un algorithme efficace pour trouver un chemin qui évite tous les obstacles y compris le robot adverse.

### 4.2.1 Construction du terrain de jeu

Le terrain est modélisé comme un tableau de dimension 2 et de taille $2000 \times 3000$dont chaque case représente un carré de 1cm. Les positions des obstacles sont préchargées dans le code du robot et le robot adversaire sera ajouté grâce au capteur. De ce fait, le robot doit pouvoir recalculer la trajectoire en temps très court. Un maillage plus complexe sera difficile à calculer et n'améliorera pas la précision à cause de l'incertitude sur la précision des moteurs des roues codeuses.

### 4.2.2 Recherche de chemin

Une fois la carte établie, il faut maintenant trouver un bon chemin. Le plus courts chemin est optimal, mais pas nécessaire, car nous avons assez de temps pour faire des détours. D'un point de vue pratique, le robot a besoin du temps pour accélérer et pour décélérer, il a aussi besoin du temps pour tourner. Nous devons donc privilégier un chemin avec le moins de changements de direction possible. Pour ce faire, nous avons utilisé l'algorithme A* pour trouver d'abord un plus court chemin puis on parcourt à nouveau le résultat pour enlever certains changements de direction.

Sans entrer dans les détails, l'algorithme A* permet d'éviter des chemins inutiles (ceci n'est complètement vrai que si la fonction heuristique correspond à la distance réelle) grâce à sa fonction heuristique, comme si on pouvait prévoir plusieurs étapes dans une partie d'échecs. Mais on constate que sur une carte compliquée où les chemins sont étroits, la recherche A* n'est pas plus rapide que l'algorithme de Dijkstra.

### 4.2.3 Simulation en Python

Afin de visualiser le résultat de l'algorithme A* présenté ci-dessus, nous avons développé un simulateur qui cherche un chemin (pas nécessairement optimal selon la fonction d'estimation choisie) avec pour donnée une carte avec les obstacles et la taille du robot. On dessine en suite le résultat en Python : les obstacles sont dessinés en noir et la trajectoire en rouge, le cercle jaune représente la position actuelle du robot.

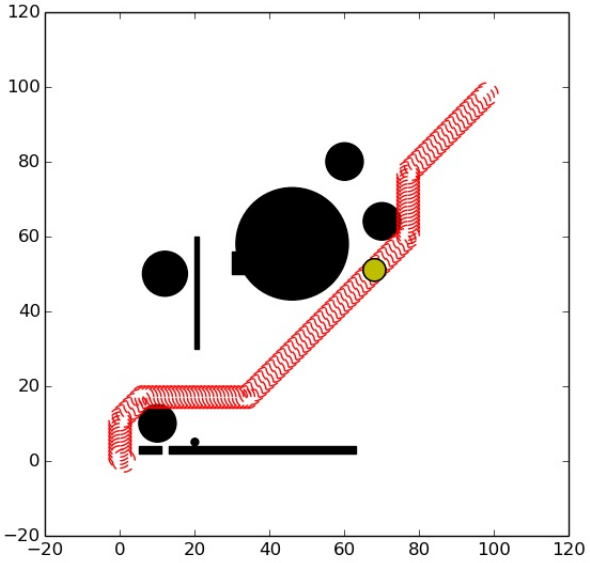



FIGURE 24 – Simulation de l'algorithme A*

On constate lors de cette simulation que le calcul d'un chemin optimal peut prendre un temps non négligeable, ceci devient encore plus important si le processeur du robot est moins puissant. Dans la pratique, on voit qu'un maillage de l'ordre de $1000 \times 1000$ est plus raisonnable.

## 4.3 Stratégie du jeu

En parallèle avec le développement de l'algorithme, nous avons aussi étudiée aussi la stratégie du jeu pour que le robot profite bien de toutes les capacités que nous lui avons fournies.

### 4.3.1 Notre stratégie

La stratégie de notre robot est très simple :

1. Se déplacer le plus vite possible jusqu'au pied de l'escalier
2. Descendre la roue arrière
3. Bloquer le support de la roue arrière

4. Monter l'escalier
5. Eventuellement jeter le(s) tapis(s) du haut de l'escalier

Nous avons choisi cette stratégie, parce que le terrain du jeu est très compliqué et deviendra encore plus compliqué une fois le jeu commencée : les balles de ping-pong remplirons le terrain et il sera très difficile de bien se positionner à ce moment-là. Notre montée rapide permettra aussi au robot principal d'avoir plus de place pour se déplacer.

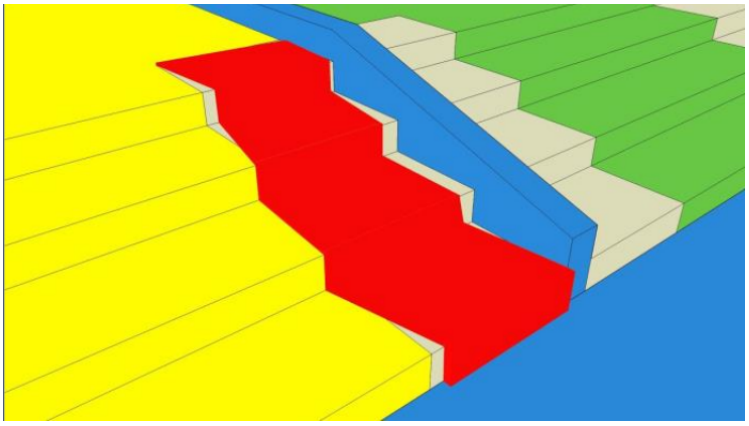


FIGURE 25 – L'escalier et le tapis valide

Nous avons choisi d'oublier le tapis dans un premier temps, car la montée des marches est prioritaire et nous ne sommes pas sûr d'avoir la bonne place pour mettre le mécanisme du tapis.

### 4.3.2 Exécution des tâches

#### Diviser pour régner

Tout peut se passer sur le terrain du jeu, le but de notre stratégie est de finir le plus de tâches possibles. Comme certaines tâches peuvent être interrompues par l'adversaire, on divise les tâches en sous-tâches et les exécute séparément tant qu'elles sont entièrement indépendantes.

Prenons l'exemple de la montée de marches. Le but principal est de monter sur les marches à la fin du jeu. Elle se divise en 4 petits morceaux : se déplacer vers le pied des marches ; descendre la $3^{\text{ème}}$ roue et la bloquer ; monter sur les marches ; arrêter les moteurs. Malheuresement, le nombre de tâches de notre robot est si limité qu'on ne voit pas l'avantage de cette stratégie. Cet avantage est d'autant plus fort qu'on a plus de tâches indépendantes.

#### Ordonnanceur

L'ordre des tâches est prédéfini à la main en donnant une priorité à chaque tâche. Lors de l'exécution des tâches, cette priorité est susceptible d'être modifiée. Par exemple, une fois qu'un déplacement est interrompu, soit on génère un nouveau chemin, soit on réinsère la tâche dans la queue de priorité en diminuant sa priorité et choisissant une nouvelle tâche à exécuter. Cela permet au robot d'exécuter un plus grand nombre de tâches tout en étant bloqué par l'adversaire.

### 4.3.3 Simulation avec Unity 3D

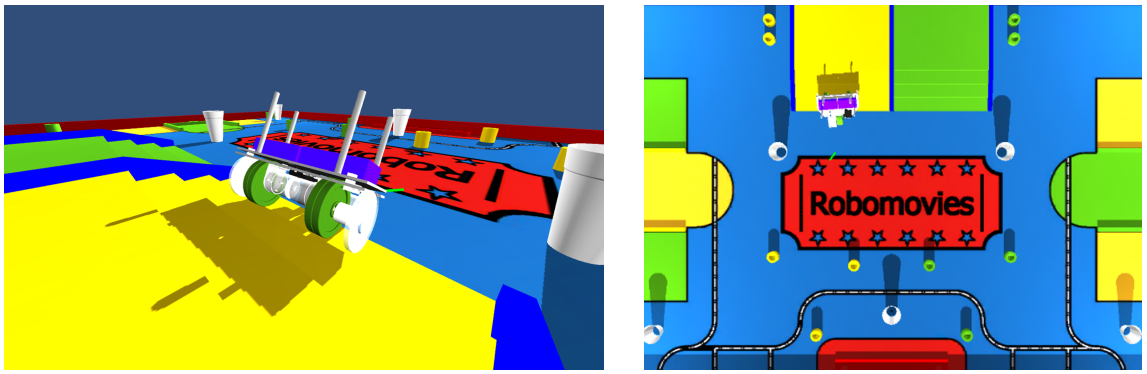



FIGURE 26 – Simulation avec Unity 3D

**Idée principale**

Pour illustrer cette stratégie, nous avons développé une simulation avec **Unity 3D**. Le but de cette simulation n'est pas de simuler les conditions réelles, car le moteur physique de ce logiciel n'est pas assez puissant, mais à travers cette simulation, on comprend mieux les différents états du robot et on voit que la division des tâches permet de suspendre certaines activités et de les réexécuter par la suite. Etant une simple simulation, le robot est sous commande manuelle et ne peut pas détecter les obstacles, le déplacement n'est donc pas automatique. Cette deuxième simulation est donc entièrement complémentaire à la simulation précédente.

Afin d'enchaîner les actions, on utilise le principe d'automate pour décrire le robot. Cet automate possède huit états :

0. Freiner
1. En attente de tâche
2. Lever la roue arrière
3. Baisser la roue arrière
4. Trouner vers une direction
5. Monter sur l'escalier
6. Déplacer
7. Se préparer pour descendre l'escalier
8. Descendre l'escalier

Le fonctionnement du simulateur se résume donc par un simple GRAFCET sur la page suivante :

**Différence critique avec le réel**

Bien que cette simulation se base sur un vrai terrain, elle est loin de la réalité en terme de la commande et de l'acquisition d'information. Le robot virtuel ne possède aucun capteur et sa position et son orientation sont connues uniquement via les API du logiciel. Cette différence a rendu l'adaptation de la stratégie sur notre robot réel très compliquée, mais grâce aux capteurs utilisés dans la partie électronique, on a réussi à construire tout un système de communication jouant le rôle des API de position.

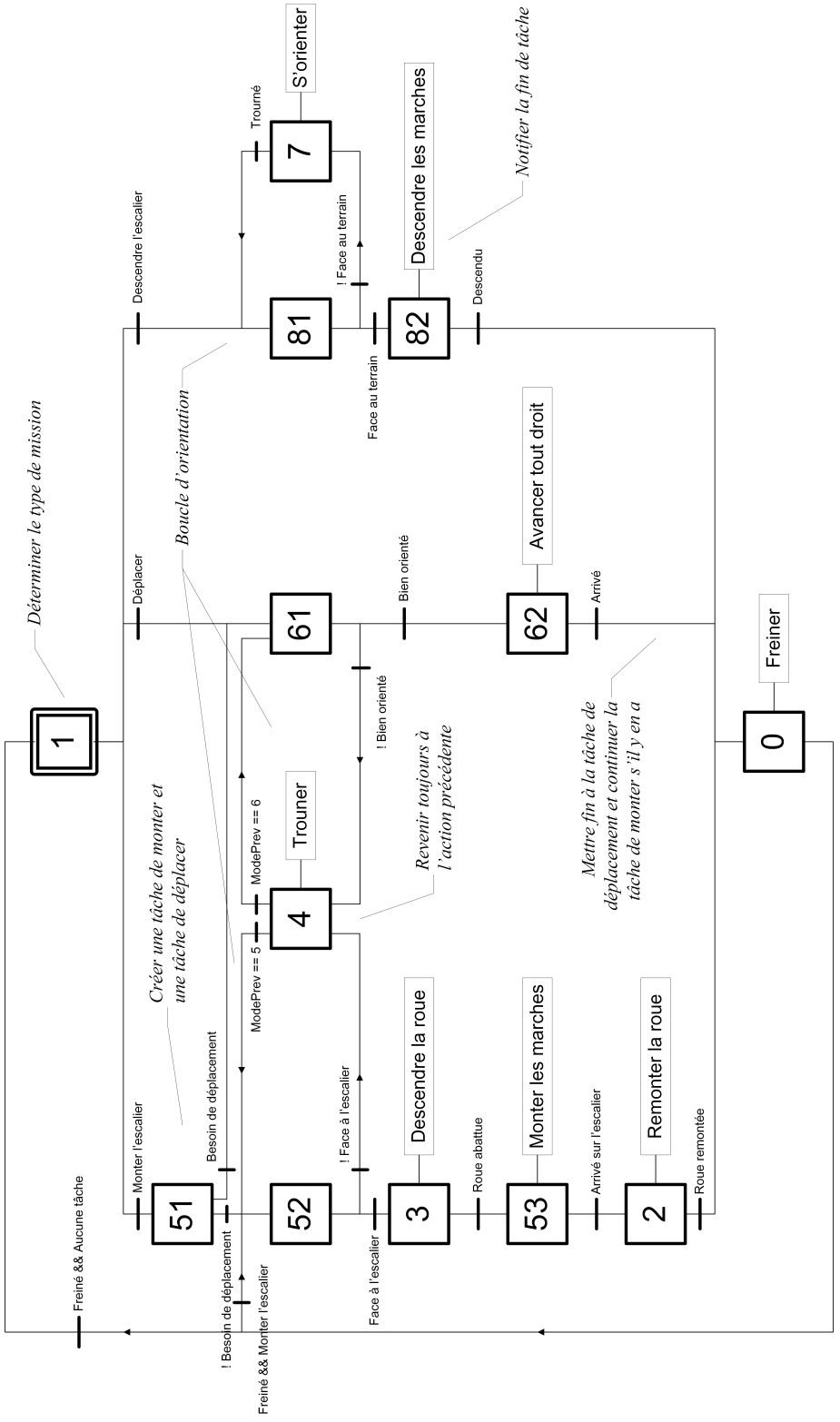



FIGURE 27 – GRAFCET représentant l'ordre d'exécution des tâches

**Avantage de la simulation**

Cette simulation nous a permis de mieux connaître le comportement éventuel du robot sans avoir besoin de faire le test, sachant que les autres parties du robot avaient besoin de temps pour se développer. Nous avons pu constater dans la simulation que la montée des marches présente une périodicité, une alternance entre les roues avant et la roue arrière. Cette considération nous a conduit à contrôler la roue arrière par une suite de petits accoups qui permet de pousser tout le robot et de monter les marches une par une tranquillement.

# 5 Conclusion

Au cours de ce projet, nous nous sommes répartis les tâches selon trois parties : la mécanique du robot, les cartes électroniques et le codage. En discutant les uns avec les autres, nous avons pu apprendre beaucoup de choses. Nous connaissons maintenant comment fonctionne le robot et surtout nous avons appris à travailler ensemble.

L'année dernière, ce projet était un grand succès, malheureusement, nous manquions d'expérience initiale dans ce domaine, devions construire un robot de zéro et respecter des contraintes plus strictes. C'est pourquoi notre robot est loin d'être comparable avec celui de l'années passée en terme de tâches accomplies.

Nous sommes tout de même aujourd'hui parvenu à construire notre robot sans réutiliser de matériel existant et sans connaissance préalable sur la robotique. Et bien que notre robot ne fonctionne que partiellement comparé avec nos objectifs initiaux, nous sommes fiers d'avoir participé à sa réalisation. Certaines cellules du projet n'ont pas abouti au résultat prévu jusqu'à ce jour, mais nous continuerons nos efforts jusqu'au dernier moment afin de se placer à la meiileure place lors de la Coupe de France.

# 6 Remerciements

Nous avons été beaucoup aidés par l'ancien président du binet robot David Octavian tout au long de notre projet et nous tenions à l'en remercier vivement.

# Références

ZHENG Xuesong CAI Linqin and WANG Niu. Detection of target based on color space of hsv stereo vision of orthongnal. *New type Industrialization*, 2013.

Giuseppina Gini and Alberto Marchi. Indoor robot navigation with single camera vision. In *PRIS*, 2002.

LAVALLE Steven M. *Planning algorithms*. Cambridge university press, 2006.

Rubaiat I. Linda Md. A. Hossain. Nafis A. Chowdhury and Shamiuzzaman Akhtar. Design and manufecturing of a stair climbing vehicle. In *International Conference on Industrial Engineering and Operations Management*, 2010.

Christopher A. Ryther and Ole B. Madsen. Onstacle detection and avoidance for mobile robots. Bachelor's thesis, Technical University of Denmark, 2009.

Parveen Sharma. Wheel modification of a wheel type stair climber. *International Journal of Engineering and Advanced Technology*, 3, 2013.

Evans Brian W. *Arduino Programming Notebook*. 2007.