PSC
COUPE DE FRANCE DE ROBOTIQUE

Rapport final - Groupe INF/MEC20D

30 avril 2018

CASTES Charly, COUGOUREUX Louis, GOLI Kouassi,
GUILLET Quentin, MATHEY Maxence, RENAUT Léo,
SOLLIER Etienne

EXECUTIVE SUMMARY

Notre PSC consiste en la réalisation d'un robot pour participer à la coupe de France de robotique, du 9 au 12 mai 2018. Ce PSC est repris d'année en année, cependant, l'équipe des X2014 n'avait pas réussi à terminer le robot à temps pour participer à la coupe. L'équipe des X2015 a donc dû repartir de zéro, et est parvenue à participer à la coupe, se classant 81ème. Notre objectif était donc d'utiliser le retour d'expérience des années précédentes afin de faire mieux que nos prédécesseurs.

Comme la construction d'un robot nécessite de faire fonctionner ensemble de nombreux systèmes différents, nous nous sommes répartis en plusieurs pôles : un pôle s'est occupé de l'asservissement, un autre des capteurs, un des actionneurs et de la structure du robot, et enfin un de la construction du plateau. Pour un tel projet, la coordination du groupe est importante, car il ne faut pas simplement que les différents systèmes marchent séparément, il faut qu'ils communiquent entre eux pour fonctionner ensemble. La coordination des différents pôles était facilitée par le fait que nous disposions d'une salle au sein du DrahiX, qui contient notamment le plateau sur lequel doit évoluer le robot, où nous nous retrouvions pour échanger et travailler sur le robot.

Au cours de notre projet, nous avons fait face à des difficultés imprévues, comme par exemple la difficulté de faire communiquer les informations des capteurs à la carte principale. Nous avons donc pris du retard sur notre planning initial, et avons dû intensifier nos efforts au cours du dernier mois pour finir à temps. Notre projet a finalement abouti à la réalisation du robot que l'on voit sur la figure 1, avec lequel nous participerons à la coupe de France de robotique.

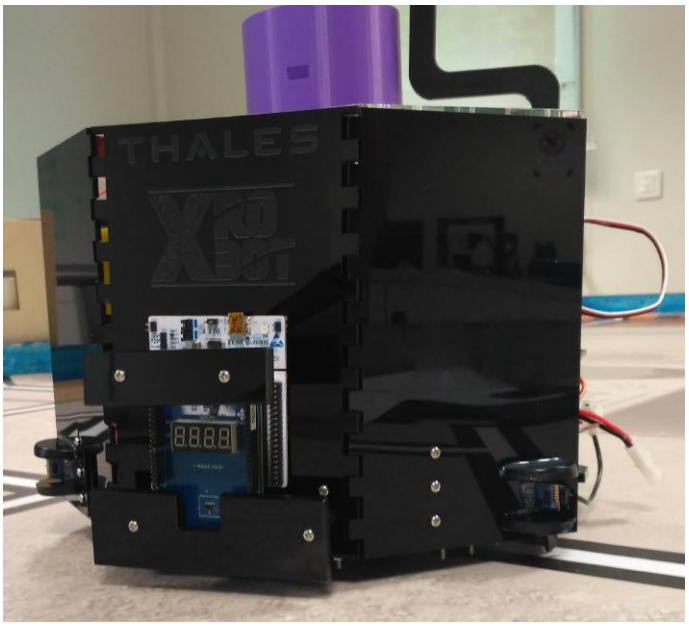

FIGURE 1 – Notre robot final

TABLE DES MATIÈRES

1 INTRODUCTION

Du fait de l'évolution fulgurante du domaine de l'intelligence artificielle et des méthodes de prototypage ces dernières années, la robotique est un sujet passionnant et en plein essor. Déjà très présente dans les chaînes de production depuis plusieurs décennies, la robotique se développe pour automatiser les entrepôts (Amazon) et même notre vie courante : c'est d'ailleurs le sujet d'étude de deux des six start-ups de la nouvelle promotion incubées au Pôle Entrepreneuriat et Innovation (PEI) de l'École polytechnique.

Fasciné par la robotique et ses applications, notre groupe s'est formé autour de cet intérêt commun et d'une forte volonté d'apprentissage. Aucun d'entre nous n'avait eu l'occasion de réaliser de projet dans ce domaine par le passé, notre participation à la Coupe de France de Robotique représente donc un véritable baptême et un réel défi pour l'ensemble de l'équipe.

À l'heure de l'écriture de ces lignes la Coupe approche à grands pas. Nous présentons ici les résultats de notre équipe qui, partant de zéro, a pu réaliser le robot avec lequel nous serons fiers de porter haut les couleurs de l'École.

2 LE PROJET ET SON ORGANISATION

2.1 Notre équipe

Nous avons formé une équipe de 7 personnes dans l'objectif de participer à la Coupe de France de Robotique qui aura lieu du 9 au 12 mai 2018 à la Roche-Sur-Yon.

La Coupe de France est une compétition qui rassemble chaque année plus de 1000 participants répartis dans plus de 200 équipes. On peut y trouver tant des étudiants amateurs de robotique que des professionnels du secteur. Reconnu comme le plus grand rendez-vous annuel dans ce domaine, beaucoup d'ingénieurs et d'entrepreneurs travaillant dans des secteurs connexes (tels que l'informatique ou la mécanique) ont eu l'occasion d'y prendre part : c'est par exemple le cas de plusieurs fondateurs de start-ups incubées à l'X, et d'ingénieurs en poste comme certains d'entre nous l'ont découvert pendant les entretiens de stage.

Notre équipe réunit des élèves issus de filières différentes ayant donc diverses formations, sans connaissances préalables dans le domaine si ce n'est quelque cours connexes dont certains d'entre nous ont pu bénéficier. L'an dernier une équipe de 12 élèves de la X15 a réussi, dans des conditions similaires, à relever le défi de participer à la Coupe (c'est à dire passer avec succès l'étape d'homologation) ce qui n'était

pas le cas pour l'équipe X14. Notre objectif est donc double : Réussir à franchir l'épreuve d'homologation et améliorer le résultat de l'équipe X15 qui a terminé à la 81ème place, réalisant ainsi la meilleure performance d'une équipe de l'École sur ces 5 dernière années.

Ainsi notre premier défi est d'ordre organisationnel : Comment faire aussi bien que l'équipe X15 avec moins de ressources humaines ?

Nous avons structuré l'équipe en 4 pôles, contre 5 l'an dernier, à savoir :

- **Mécanique** : 2 personnes (Maxence et Louis) responsables d'adapter le terrain aux contraintes de l'année, réaliser les parties manquantes et créer les systèmes entièrement mécaniques tel que l'abeille (voir 4.3.3).
- **Asservissement** : 2 personnes (Quentin et Léo) chargées de mettre au point l'asservissement du robot, c'est à dire le système permettant de contrôler sa position et son déplacement au cours du temps.
- **Capteurs** : 1 personne (Kouassi) chargée de gérer l'installation des capteurs de distance permettant d'éviter toute collision (point crucial de l'homologation).
- **Structure** : 2 personnes (Charly et Etienne) responsables de la structure du robot et des différentes parties mécaniques internes permettant de réaliser les actions souhaitées.

Cependant cette répartition ne permet pas de couvrir tout le spectre des activités nécessaires à la réalisation du robot. A titre d'exemple, dans le respect strict de la segmentation proposée ci-dessus personne ne s'occuperait du circuit électrique qui est pourtant essentiel au fonctionnement. Ces pôles constituent donc l'activité principale de chaque élève mais en aucun cas une liste exhaustive des tâches à réaliser.

Les missions de chaque pôle sont fixées pendant les réunions en début de séance, et peuvent s'étaler sur plusieurs semaines (voire mois pour les parties les plus exigeantes, telles que l'asservissement).

Nous étions encadrés par Brieuc de Maugouër, notre tuteur, fondateur de la start-up Auxivia. Ayant une grande expérience de la Coupe de France de robotique (il y a participé 5 fois), il nous a été d'une grande aide, d'autant qu'il avait déjà tutoré les deux groupes d'X 2015 et 2014 ayant participé à la coupe. Comme il travaille à Paris, il nous était difficile de le rencontrer régulièrement en personne. Nous nous sommes déplacés une fois sur son lieu de travail, au début du PSC, pour faire un point sur l'organisation globale. Il est également venu une fois à l'X pour nous aider à solutionner des difficultés que nous rencontrions. Cependant, nous faisions un point avec lui par Skype toutes les semaines ou toutes les deux semaines, pour évaluer régulièrement l'avancement du robot. Il nous a notamment mis en garde en janvier alors nous avions pris beaucoup de retard, et nous a donné de précieux conseils et avis tout au long du projet.

2.2 Nos Sponsors et Moyens

Un enjeu non négligeable de la Coupe est de disposer de ressources financières suffisantes pour ne pas se retrouver entravé dans le développement du robot, ce qui est en soit un défi à part entière.

Dès la première réunion avec notre tuteur Brieuc, celui-ci nous a annoncé que le budget pour la réalisation d'un robot de notre envergure serait entre 3000€et 6000€. Nous nous sommes donc mis à la recherche de partenaires, que ce soit pour récupérer du matériel, comme les cartes de chez ST microelectronics qui avaient été offertes à l'équipe de l'année dernière et que nous avons réutilisées, ou directement des financements.

Malheureusement, nous n'avons pas pu reconduire le partenariat qu'avaient les X2015 avec ST car ces derniers ont décidé d'arrêter de subventionner la Coupe de France de robotique. Nous nous sommes donc tournés vers un ancien partenaire de X-Robot : Thalès. Suite à la non-homologation de l'équipe de l'École en 2014, Thalès avait décidé d'arrêter de nous subventionner. Cependant, grâce au résultat réalisé l'an passé, Thalès nous a accordé un financement à hauteur de 1000€pour cette année. Si nous réussissons à obtenir un bon résultat en mai, nous espérons fidéliser le partenariat avec l'entreprise.

Nous avons par ailleurs bénéficié tout au long de l'année des installations du Fablab du DrahiX Center. En effet, une salle nous est réservée afin que nous nous y réunissions chaque semaine, et nous avons pu y installer le plateau de jeu. De plus, le Fablab nous a mis à disposition ses imprimantes 3D, sa découpeuse laser et de nombreux autres outils qui nous ont été indispensables dans la réalisation du robot.

Enfin, nous avons profité de la vague de subventions PSC proposées par le Pôle Innovation Entrepreneuriat de l'École qui nous a permis de réaliser une grande partie de nos achats de matériel électronique de l'année, puisqu'ils nous ont subventionnés à hauteur de 3000€.

Nous profitons de ce rapport pour réitérer nos remerciements à nos partenaires qui nous ont permis d'arriver à l'état actuel du robot, et nous espérons réaliser une belle performance à la Coupe.

2.3 Gestion de l'équipe

La répartition par pôles à été réalisée dès le début du PSC, peu après la rentrée scolaire fin août. Cette segmentation rapide nous a permis à chacun de se concentrer sur un domaine particulier, d'acquérir les compétences nécessaires et que nous n'avions pas encore. Le règlement n'étant publié que mi-octobre, nous avons exploité cette période pour nous concentrer sur les aspects fondamentaux de la robotique et

qui ne dépendent pas des tâches à accomplir. On peut citer par exemple l'apprentissage des méthodes de prototypage, le fonctionnement général d'un asservissement ou encore une analyse comparative des différents capteurs disponibles sur le marché.

Cette période nous a également été utile pour apprendre des erreurs de nos prédécesseurs. En effet, bien que leur robot ne soit pas réutilisable, même en partie, car il n'a pas été conçu de manière modulaire (créer un robot fonctionnel capable de réaliser un ensemble d'actions définies est déjà suffisamment exigent), il nous a été très utile d'analyser leur réalisation et de discuter avec eux des points noirs de leurs décisions et organisation. À titre d'exemple, ils se sont rendus compte que réaliser une base asymétrique complexifiait énormément l'asservissement, ce qui a directement influencé la conception de nos premiers prototypes.

Après la publication du règlement et la première phase d'apprentissage, nous avons passé une séance entière à discuter collectivement de la stratégie à adopter. Ici nos méthodes divergent de celles des X15 qui avait dédié 2 personnes au pôle stratégie, c'est un choix dû à la fois à notre effectif plus faible et à notre volonté d'impliquer toute l'équipe sur les choix capitaux qui dirigeront nos travaux tout le reste de l'année. Ainsi, chacun a pu décider de ce qui était ou non faisable en fonction de ses compétences nouvellement acquises. Nous n'avons d'ailleurs pas hésité à consacrer du temps à des discussions collectives supplémentaires pour résoudre les problèmes cruciaux ou prendre des décisions d'orientation majeures, telles que l'abandon des cartes de contrôle MD25 pour l'asservissement.

Une fois la stratégie définie, nous avons choisi d'utiliser un outil de gestion de tâches (Trello) pour définir le travail de chaque pôle ou personne via une liste de priorité. Cet outil se montre très souple, les tâches peuvent être ajoutées, supprimées ou décalées sans pour autant perturber les autres. Ce choix méthodologique agile a été très utile pour réagir rapidement aux différents imprévus (besoin de redimensionner une pièce pour tester un actionneur) mais a eu le défaut de ne pas nous alerter assez rapidement des dépassements de timing et des tâches qui auraient dû être avortées plus tôt (utilisation des cartes MD25 encore une fois, ou choix d'exploitation des capteurs).

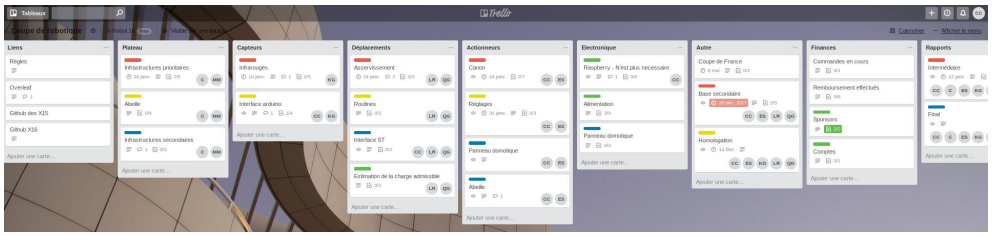

Figure 2 – Logiciel d'organisation de tâches

Comme nous travaillions tous ensemble au DrahiX, les différents pôles pouvaient aisément communiquer entre eux. Ainsi, chaque pôle a pu prioriser certaines tâches en fonction des besoins des autres. Par exemple, pour que le groupe travaillant sur les actionneurs puisse rapidement tester le lanceur de balles, l'équipe s'occupant de la construction du plateau a réalisé en priorité le château d'eau, où les balles sont envoyées.

3
RÈGLEMENT ET STRATÉGIE

3.1 Règlement

La coupe de France de robotique consiste en une suite de matchs qui opposent deux robots sur un plateau de jeu de 2x3m (figure 3), pendant une durée de 100 secondes.

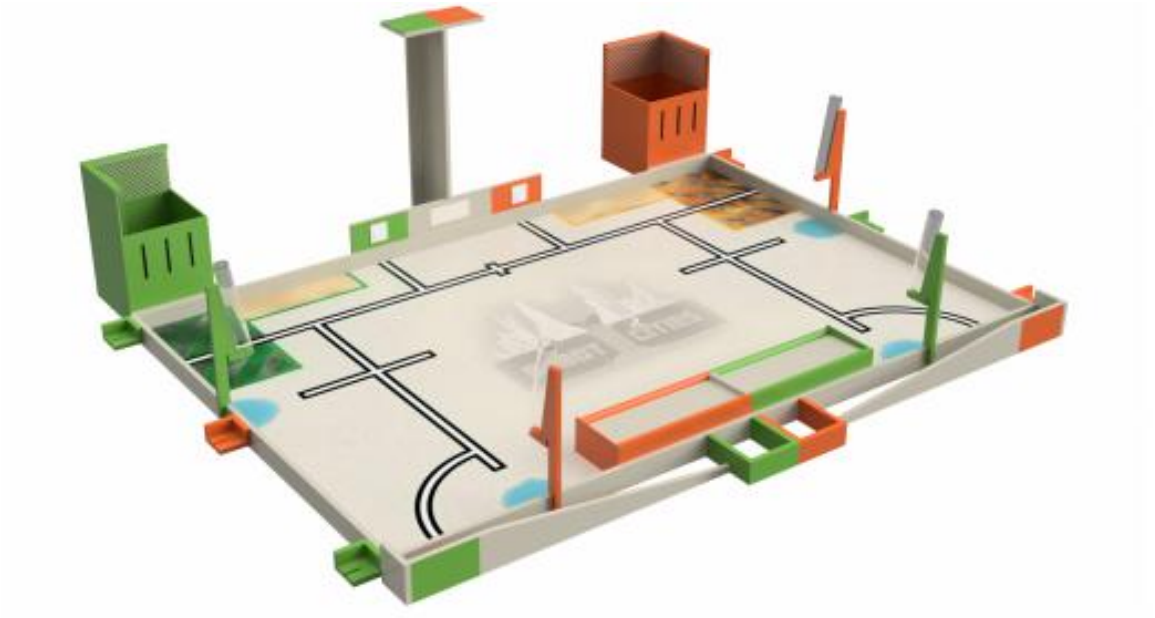

Figure 3 – Plateau de jeu

Au cours de chaque match, les robots doivent effectuer différentes actions afin de marquer un maximum de points. Il est imposé que les robots soient entièrement autonomes et disposent d'un système d'évitement leur permettant de ne pas entrer en collision avec un robot adverse. Ce point est important, car avant de pouvoir participer à la coupe, il faut passer une étape d'homologation qui consiste à vérifier que le robot s'arrête bien lorsqu'il rencontre un obstacle. De nombreuses équipes sont éliminées avant de pouvoir participer à la coupe car elles ne parviennent pas à passer cette homologation. Chaque équipe a le droit d'utiliser 2 robots à la fois, mais l'un des deux doit avoir des dimensions réduites (voir le règlement en annexe pour plus de précisions sur le second robot).

Chaque année, la coupe de Robotique a un thème différent, ce qui implique que les actions à réaliser par les robots sont différentes. Cette année, le thème est "Robot Cities" et les actions à effectuer pour marquer des points ont de fait une coloration écologique. Les actions sont :

— **Alimenter la ville en eau potable.** Chaque équipe dispose de deux récupérateurs d'eaux usées. Ce sont des cylindres contenant 8 balles (figure 4), les balles représentant l'eau.

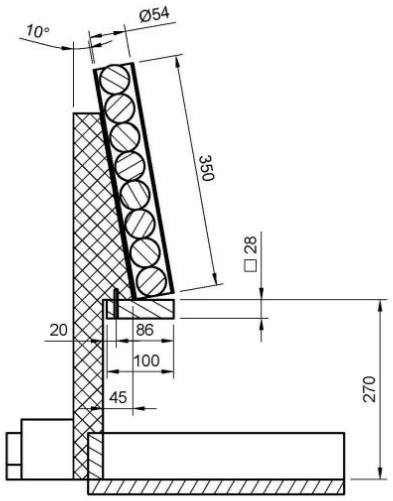

Figure 4 – Schéma du récupérateur d'eaux usées

Il y a des balles de deux couleurs différentes : les balles de la couleur de l'équipe correspondent à de l'eau "propre", tandis que les balles de la couleur de l'équipe adverse correspondent à de l'eau dite "polluée". Le premier récupérateur d'eau contient 8 balles propres et le deuxième contient 4 balles propres alternées avec 4 balles polluées. Les robots doivent récupérer les balles en ouvrant un loquet sous le collecteur d'eau, ce qui rapporte déjà 10 points, puis placer les balles à l'endroit approprié en fonction de leur type :

— Les balles "propres" doivent être placées dans un château d'eau, représenté en figure 5. A noter que l'ouverture dans le château d'eau est située à une hauteur de 40 cm, ce qui est plus haut que la hauteur maximale autorisée du robot, qui est 35cm. Chaque balle propre mise dans le château d'eau rapporte 5 points.

Figure 5 – Château d'eau

— Les balles ”polluées” doivent être mises dans une station d'épuration. La station d'épuration est un rectangle de 60cm par 25cm, entouré de murs de 3cm de haut et situé à 10cm du sol. Elle est représentée en figure 6. Chaque balle ”polluée” mise dans la station d'épuration rapporte 10 points.

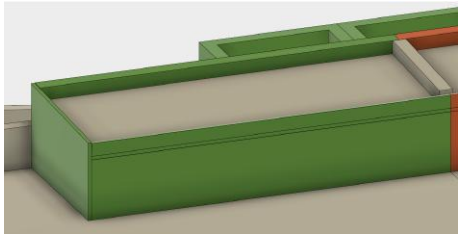

Figure 6 – Station d'épuration

— **Construire des bâtiments Haute Qualité Énergétique.** Les robots doivent également récupérer des cubes, qui sont initialement disposés au sol en forme de croix (figure 7), et les empiler dans une zone de construction. Plus on empile les cubes hauts, plus ils rapportent de points : un cube au sol dans la zone de construction rapporte 1 point, un cube au deuxième étage rapporte 2 points,... jusqu'à 5 points pour un cube au cinquième étage. De plus, il est possible de gagner des points bonus en empilant les cubes dans un ordre particulier. Cet ordre de couleur varie à chaque match, et il doit être lu par le robot sur un des murs du terrain. Si un robot empile des cubes selon le code couleur, il gagne 30 points bonus.

Figure 7 – Cubes de construction. A gauche : Disposition initiale. A droite : Exemples d'empilements

— **Alimenter son panneau domotique.** Chaque équipe doit fabriquer un panneau domotique de largeur comprise entre 40 et 100cm et de hauteur comprise entre 30 et 40 cm, qui doit disposer de fixations velcro de sorte à pouvoir être accroché au château d'eau, sur le bord du terrain, comme sur la figure 8. La fonction du panneau est libre, la seule contrainte étant que l'on puisse clairement voir si le panneau est allumé ou éteint (avec une LED par exemple). Le simple fait de poser un panneau domotique sur le terrain pendant la phase de préparation rapporte 5 points, et il est possible de gagner 25 points supplémentaires si le robot allume le panneau domotique au cours de la partie. Pour cela, il doit appuyer sur un interrupteur de type poussoir, situé à 11cm du sol.

Figure 8 – A gauche : Un panneau domotique fixé au château d'eau. A droite : l'interrupteur.

— **Butiner une fleur.** Chaque équipe doit fabriquer une "abeille", un objet entièrement mécanique qui roule ou glisse sur une rampe, située sur le bord du terrain. Comme on le voit sur le schéma en figure 9, la rampe est plate au début, puis descend. L'abeille est initialement sur la partie plate, et le robot doit la pousser afin qu'elle dévale la pente, au bout de laquelle se situe un gros ballon gonflable, représentant une fleur, que l'abeille doit faire éclater. Le simple fait de poser une abeille sur la rampe pendant la phase de préparation rapport 5 points, et faire éclater la ballon pendant la partie rapporte 50 points.

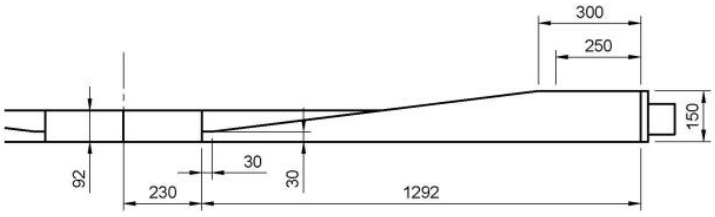

Figure 9 – Schéma de la rampe de l'abeille

— **Évaluer sa performance.** Le robot doit donner une estimation du score qu'il pense faire. L'affichage de ce score peut être statique (écrit sur une feuille de papier par exemple), ou dynamique (avec un écran). Plus l'estimation du score est proche du score réel, plus on gagne de points bonus.

3.2 Stratégie

Nous avons élaboré notre stratégie en nous basant sur le retour d'expérience des groupes d'X ayant participé à la coupe de France de robotique avant nous. Le principal enseignement que nous avons pu tirer de ces retours est qu'il ne faut pas être trop ambitieux et ne pas chercher à faire toutes les actions, ni des actions trop compliquées. En effet, le groupe X2014 avait voulu effectuer beaucoup d'actions différentes, s'est en conséquence dispersé, n'a pas réussi à terminer le robot à temps et n'a de fait pas pu participer à la coupe.

Le groupe X2015 a également été trop ambitieux : il a par exemple voulu faire deux robots différents, mais n'a pas eu le temps de finir le deuxième, et a ainsi perdu du temps à construire un robot qui n'a pas été utilisé. Par ailleurs, notre tuteur nous a également recommandé de ne pas être trop ambitieux mais de nous concentrer sur la fiabilité et l'efficacité de notre robot.

Cette année, le choix de se limiter aux actions simples est d'autant plus justifié qu'il y a des points bonus sur la prédiction du score ! Ainsi réaliser un robot fiable, qui réussit systématiquement à effectuer les actions qu'il est censé faire, nous assure de toujours marquer les points de prédictions.

Nous nous sommes donc focalisés sur les actions les plus simples : l'abeille et le panneau domotique, qui rapportent respectivement 55 et 30 points, alors qu'elles ne nécessitent pas d'actionneurs trop compliqués. A l'inverse, nous avons d'emblée écarté les cubes de nos objectifs. En effet, l'empilement des cubes aurait été très complexe. De plus, en essayant d'empiler des cubes, il aurait été possible qu'en pensant mettre un cube au dessus d'un autre, le robot le mette en fait à côté, ce qui est difficile à détecter à l'aide de capteurs. En empilant les cubes, on prendrait donc le risque de perdre des points de prédiction du score, ce qui nous a semblé être un mauvais calcul. Nous avons tout de même voulu réaliser une action relativement complexe, à savoir récupérer les balles dans les récupérateurs d'eau usées et les mettre dans le château d'eau. A noter que les deux conteneurs d'eau que nous pouvons utiliser ne sont pas situés au même endroit : le premier, qui ne contient que des balles "propres", est situé proche de la zone de départ, tandis que le second, qui contient 4 balles "propres" et 4 balles "polluées" se situe du côté de l'équipe adverse (voir figure 10).

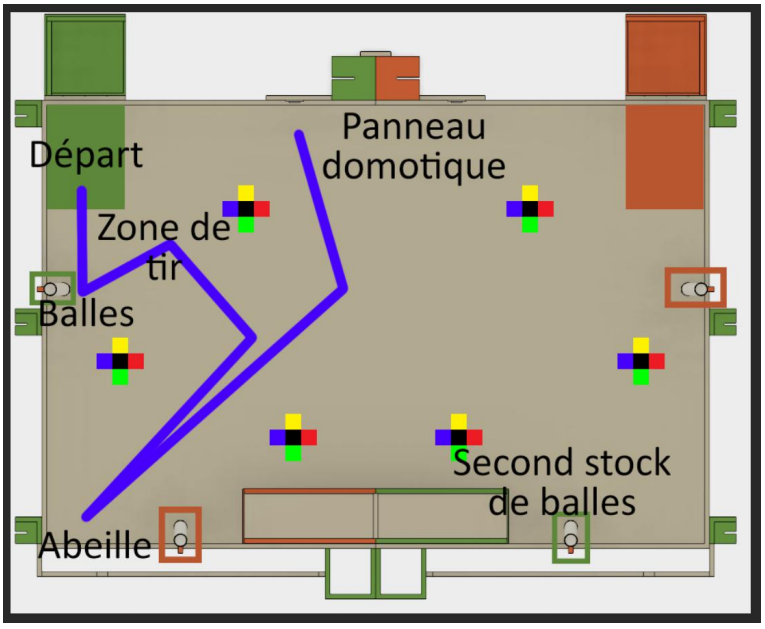

Figure 10 – Le trajet effectué par le robot sur le plateau

Par conséquent, en essayant d'aller récupérer des balles dans le deuxième récupérateur d'eau, notre robot risque de rencontrer le robot adverse. Cela serait

problématique car notre robot est capable de s'arrêter s'il rencontre un obstacle, mais il n'est pas capable de le contourner par manque de temps pour développer les algorithmes. Si nous allons du côté de l'équipe adverse, nous risquons d'aboutir à un blocage mutuel entre notre robot et le robot adverse. Pour cette raison, nous ne récupérons que les balles qui sont dans le premier récupérateur. Cela nous évite également de devoir faire un système de tri des balles.

Notre stratégie nous permet donc d'effectuer toutes nos actions en restant dans notre moitié du terrain, comme on peut le voir sur le parcours du robot en figure 10. Cela signifie que, tant que nous ne faisons pas face aux meilleures équipes, notre robot ne devrait pas croiser le robot adverse. Notre stratégie nous permet de gagner 55 points avec l'abeille, 30 points avec le panneau domotique et 50 points pour les balles mises dans le château d'eau. Cela nous amène à 135 points, et si nous prédisons correctement le score, nous gagnerions 43 points bonus, ce qui nous permettrait d'avoir **un total maximal de 178 points.** A titre de comparaison, les équipes participant aux premiers tours des Coupes de Suisse et de Belgique ont rarement dépassé les 100 points.

4
RÉALISATIONS TECHNIQUES

4.1 Structure

La structure constitue l'ensemble des éléments qui permettent aux différentes parties du robot de rester solidaires ; elle doit être pensée afin de ne pas gêner le fonctionnement des différents actionneurs et être relativement facile à démonter pour ainsi pouvoir corriger au fur et à mesure les problèmes que nous pourrions rencontrer tant en amont que pendant la Coupe elle-même.

4.1.1 • Critères

Nos discussions avec notre tuteur et l'équipe X15 nous mettent en garde sur l'impact de la structure sur l'asservissement du robot. En effet, une structure asymétrique (par rapport à l'axe médian parallèle aux roues) favorise un côté et complexifie la mise en place des corrections qui, dans le cas d'un robot symétrique, ne compensent que les erreurs de fabrication. Un deuxième effet pervers serait celui de la masse placée en hauteur, qui tend à déstabiliser le robot et à diminuer ses performances en accélération et décélération.

Un dernier élément à prendre en compte est celui de la facilité de déplacement du robot : Un robot à base carrée peut plus difficilement tourner sans être gêné par un obstacle qu'un robot à base circulaire. En contre-partie, un robot à base circulaire est moins adapté à la forme rectangulaire de la majorité des composants du robot (cartes de commande, de puissance, servomoteurs ou encore batteries) et est donc virtuellement moins spacieux.

4.1.2 • Réalisation

Compte tenu de ces critères, nous avons décider d'utiliser une base de forme octogonale, comme l'équipe X15, car elle constitue un bon compromis entre facilité d'utilisation et faible entrave au mouvement. De plus pour maximiser la modularité, le robot sera constitué d'étages, tous à base octogonale, maintenus par des piliers de hauteur ajustable.

Ici aussi nous avons essayé d'appliquer une méthodologie agile dans la construction de la structure : il est extrêmement difficile, voir utopique, de prévoir l'emplacement de chaque vis dès le début, ainsi nous avons procédé par itérations successives.

La découpe laser est un instrument de prototypage qui permet de découper tout type de plaque (bois, plastique, carton) en un temps très faible, de l'ordre de la minute. C'est grâce à cet outil disponible au Fablab que nous avons pu nous permettre une telle procédure d'essais/erreurs, car il nous est possible de découper un nouvel étage en une trentaine de minutes une fois l'erreur détectée, en prenant en compte

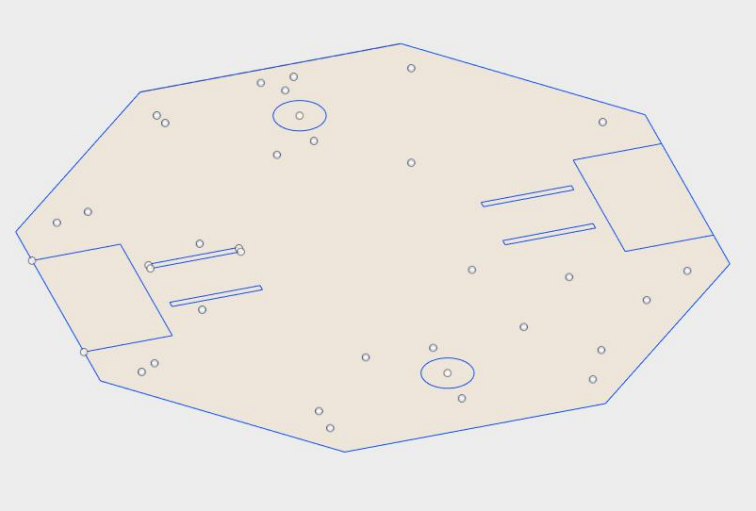

Figure 11 – Plan de la base roulante

le temps de modélisation numérique et de découpe.

Nous avons réalisé en priorité les éléments nécessaires aux autres pôles. En particulier, nous avons commencé par réaliser une première base simpliste permettant de fixer moteurs et batterie dans le but de permettre très rapidement au pôle asservissement de tester les premiers algorithmes sur une base fonctionnelle. Cette base temporaire ne permet pas le réglage de l'asservissement car son inertie n'est pas identique à celle du robot final, mais permet de commencer le développement de la structure de l'algorithme.

Toujours dans un souci d'agilité de développement, nous avons réalisé une seconde base. Ainsi il est possible de travailler à la fois sur l'asservissement avec la première base, et sur la structure avec la seconde. c'est une initiative que nous avons pris tôt dans le projet suite aux besoins des deux pôles : nous avions régulièrement la nécessité d'utiliser la base roulante au même moment pour réaliser des test différents et nous perdions du temps à attendre, monter et démonter la base.

La base utilisée par le pôle asservissement à été mise à jour plusieurs fois au cours du projet, dès qu'une version satisfaisante de la structure était au point. Ainsi ce pôle exploitait une base aussi proche que possible de la base finale (les points particulièrement sensibles sont l'écartement des roues et la répartition de la masse), sans pour autant être gêné dans ses tests car la modification est rapide quand tout les composants sont déjà réalisés et correctement dimensionnés.

Finalement la partie la plus difficile au niveau structurel a été de s'adapter à chaque élément du robot : chaque carte est vissée pour garantir la bonne connexion entre les composants tout en minimisant la place occupée, les différents orifices doivent s'adapter aux actionneurs et capteurs et évoluer en même temps que ces derniers, et enfin il faut tout de même s'assurer d'une solidité suffisante pour résister à un éventuel choc avec un objet ou un robot. Le résultat final est très satisfaisant dans la mesure où la structure est effectivement capable de contenir tous les composants tout en gardant la masse relativement proche du sol : elle est entièrement répartie dans les 20 premiers centimètres sur les 35 autorisés.

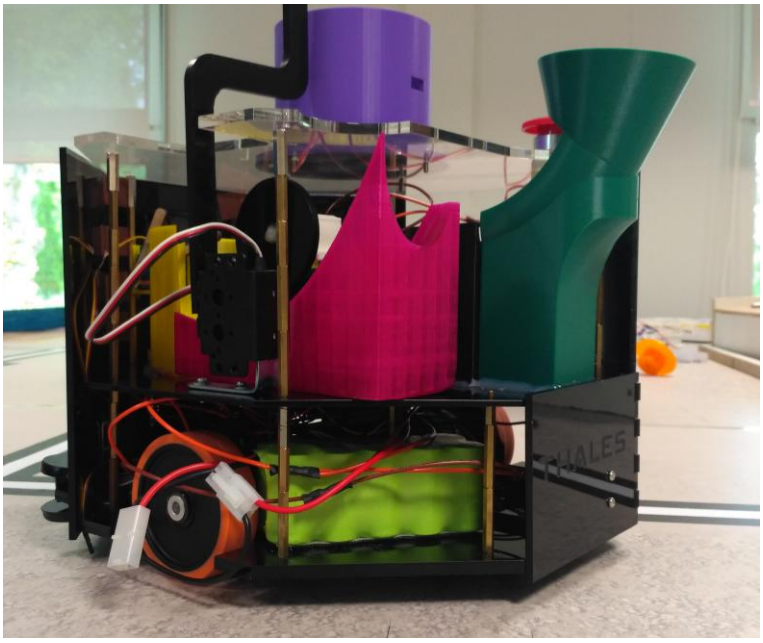

Figure 12 – Vue sur la structure du robot final

4.2 Capteurs

Les capteurs sont essentiels au robot : c'est le seul moyen pour lui de prendre en compte son environnement. Si le robot est capable de réaliser le parcours à l'aveugle grâce à l'odométrie et l'asservissement, c'est par la présence des capteurs qu'il est capable de réagir aux imprévus. Par imprévus, nous entendons bien sûr la présence d'un robot adverse.

Le rôle des capteurs est donc de communiquer la présence d'un éventuel obstacle à la carte Arduino qui gère l'asservissement et la prise de décisions. Il faut alors les choisir en fonction de leur capacité à détecter avec une bonne fiabilité et à ne pas être perturbés par l'environnement.

Deux grandes classes de capteurs de distances sont disponibles :

— Les capteurs ultrasons, faciles d'emploi avec une dispersion non négligeable et une portée largement suffisante dans notre cas, se basant sur l'utilisation d'ondes sonores.

— Les capteurs infrarouges, très directifs, avec une portée plus longue utilisant la lumière infrarouge.

Les deux types de capteurs sont capables de répondre au critère de détection, le choix s'est donc fait sur celui de la résistance aux perturbations. En discutant avec notre tuteur, les X et l'équipe de l'ENSTA nous avons pu avoir une meilleure vision de l'environnement de la coupe : les matchs se déroulent sur un plateau télé. C'est un détail qui peut paraître anodin mais il est en fait important car la présence de projecteurs peut fausser les mesures des capteurs infrarouges. Cependant, d'après les retours d'expérience les capteurs ultrasons sont utilisés par un grand nombre d'équipes, et si deux robots équipés de tels capteurs se font face ils se détectent

comme étant deux fois plus proches qu'ils ne le sont réellement.

Il est possible de se protéger des perturbations des projecteurs en créant de l'ombre sur les capteurs. On sait par nos retours que les projecteurs éclairent la table de façon verticale, il est donc relativement simple de s'en protéger. Cependant, il est dur de détecter un faux positif dans le cas où deux robots équipés de capteurs ultrason se font face, nous avons donc décidé de retenir la détection infrarouge.

Figure 13 – Capteurs de distances infrarouge

4.2.1 • Mise en place

Dans la gamme de capteurs infrarouges disponibles, nous avons fait le choix d'utiliser des capteurs de marque ST pour leur bonne qualité et parce que nous disposions déjà de matériel ST. En effet la marque ST microelectronics était sponsor de l'équipe X15 et nous bénéficions du matériel offert (en quantité) l'an dernier ce qui nous permettait de diminuer la pression budgétaire.

Les capteurs infrarouges sont intrinsèquement plus difficile à utiliser que les capteurs ultrasons, de plus notre modèle (VL53L0X) est relativement récent ce qui rend plus difficile la recherche d'informations.

ST microelectronics propose des bibliothèques de code permettant d'utiliser simplement les capteurs sur des cartes de commande ST, que nous possédons de l'équipe X15. La solution la plus simple à mettre en œuvre était donc d'utiliser une carte ST chargée exclusivement de contrôler les capteurs (qui est une tâche lourde en terme de temps de calcul car il faut nécessairement attendre la réponse de ces derniers, le temps d'attente est ainsi perdu) et de libérer ainsi la carte Arduino Méga chargé de l'asservissement.

Physiquement, il n'est possible d'utiliser que trois capteurs par carte ST. Par choix stratégique le robot ne se déplacera qu'en marche avant, et nous avons décidé

que c'était suffisant pour s'assurer que le passage est libre et s'arrêter en cas de besoin.

La carte ST est donc chargée de récupérer les distances sur ces trois capteurs qui seront répartis sur l'avant du robot, puis doit les communiquer à la carte Arduino qui prendra les décisions adéquates (en fonction du stade de la partie : il est normal de frôler des murs à certains moments par exemple, cela ne doit pas causer l'arrêt du robot).

Nous avons mis beaucoup de temps à réussir à mettre en place une communication série (basée sur le protocole UART prévu à cet effet) entre les deux cartes. En effet nous étions capables de récupérer les informations des capteurs grâce au code fourni par ST microelectronics et de les lire sur PC grâce à une connexion USB, mais pas de les transférer en utilisant des pins (connecteurs de la carte). Il s'est avéré après lecture approfondie de la documentation des cartes ST que les connections avec les pins dédiés la communication série sont coupées par défaut, il est nécessaire de modifier les connections en ajoutant et retirant des soudures pour rendre l'opération possible. Malgré le retard accumulé, les connections ont pu être établies et fonctionnent parfaitement.

4.2.2 • Protocole de communication

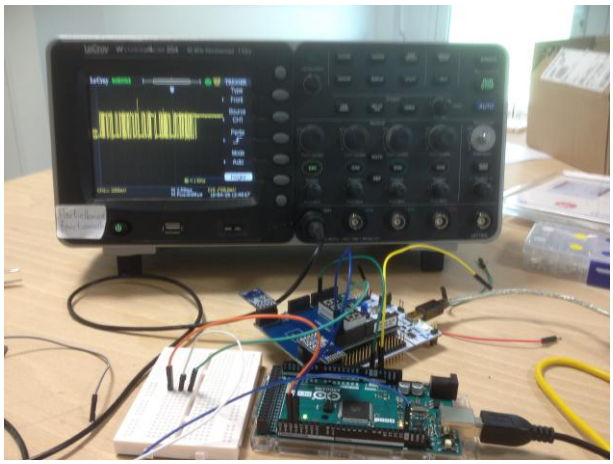

Figure 14 – Détection d'un signal série entre les cartes

Une fois la connexion établie il faut s'assurer que les valeurs envoyées peuvent être comprises et ne sont pas erronées. Le protocole UART permet d'échanger des octets entre les cartes, une unité de 8 bits. Rien ne nous assure que tous les octets seront effectivement reçus, ni qu'ils seront reçus sans être modifiés (à cause d'interférences électromagnétiques par exemple).

Pour s'assurer de recevoir des messages complets, le plus simple est de coder l'information sur un seul octet (huit 0 ou 1) (il faut sinon mettre en place un système de début et de fin de message et s'assurer que l'ensemble est bien reçu). Nous disposons alors de 8 bits pour coder toute l'information, les capteurs retournant une distance en millimètres (comprises entre 0 et 3000 avec les dimensions du plateau) pour un

total de 256 valeurs disponibles en exploitant les 8 bits.

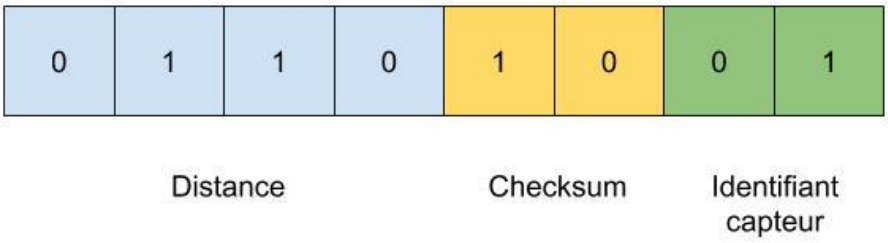

Figure 15 – Organisation d'un octet dans le protocole de communication

Nous avons fait le choix de discrétiser l'espace en 16 domaines de 0 (correspondant à la zone entre le robot et 5 cm) à 15 (plus de 1m 50). Il reste alors 4 bits qui seront séparés en deux groupes de deux bits : le premier permet d'identifier le numéro du capteur (0 pour centre, 1 pour droit et 2 pour gauche) et le dernier groupe de 2 bits est utilisé en tant que *checksum* (une valeur calculée à partir des autres bits de l'octet) pour vérifier que l'octet contient une information valide (la vérification ne porte que sur la distance, le checksum doit être égal à la distance modulo 4, on est ainsi capable d'écarter 3 erreurs sur 4 avec cette méthode).

Une communication octet par octet est possible uniquement en exploitant des fonctionnalités de bas niveau (proches de l'architecture de la machine, et dont l'effet peut varier d'une carte à l'autre).

Les cartes ST et Arduino se programment en C++ et offrent donc des outils appropriés à condition de prendre les précautions nécessaires. Les cartes ST utilisent par exemple un microprocesseur 32-bits là où les cartes Arduino Mega 2560 utilisent un microprocesseur 8-bits, il est donc par exemple nécessaire d'utiliser le type uint8_t fournit par C++ qui code les entiers (positifs) sur exactement 8 bits plutôt que le type int (plus général) qui s'adapte au processeur utilisé.

Les fonctions usuelles telles que *printf* qui permet d'envoyer un message sur un canal en C++ ne sont plus adaptées, en effet dans nos premiers tests nous recevions les messages des capteurs par blocs toutes les 7 secondes environs, ce qui est bien-sûr inacceptable pour l'évitement d'obstacles. En creusant un peu plus dans le fonctionnement bas-niveau, on apprend que la fonction printf fait appel à un buffer (espace mémoire de stockage) d'une taille variable et qui se trouve être de 1000 octets dans le cas de l'implémentation des cartes ST : un message n'était envoyé que quand cette limite des 1000 octets était atteinte, ce qui correspond à 7 secondes d'acquisition.

Pour pallier à ce problème il est nécessaire de passer par des fonctions de plus bas niveau, telle que puts qui envoie le message dès qu'un symbole terminal (codé par le caractère '\0') est lu. Ainsi sur la carte ST on crée un buffer de taille 4 dans lequel elle va stocker successivement les trois octets correspondant aux mesures des trois capteurs puis le symbole terminal qui occupe lui aussi un octet (c'est en fait la valeur binaire 000000000) et qui va donc transmettre les trois premiers octets

stockés dans le buffer. Ainsi la carte Arduino peut effectivement lire les valeurs des capteurs dès que les mesures sont effectuées du côté de la carte ST.

Après contrôle des messages émis et reçus grâce à un ordinateur intermédiaire, l'ensemble du protocole a pu être validé. Les messages sont correctement transmis et encodés à une fréquence convenable (de l'ordre de trois messages toute les 20 microsecondes) par la carte ST, correctement lus et décodés par la carte Arduino et correctement éliminés si le message est modifié (par vérification du *checksum*). Les objectifs fixés au pôle capteurs ont donc été atteints.

4.3 Actionneurs

4.3.1 • Lanceur de balles

Le système de lancement des balles est le premier actionneur que nous avons réalisé. En effet, il allait être l'actionneur le plus volumineux, et allait donc déterminer la structure globale du robot. Il était donc important de le faire au plus vite. Avant de voir comment récupérer les balles, nous avons donc commencé par étudier comment les lancer dans le château d'eau. En effet, comme l'ouverture du château d'eau est plus haute que la hauteur maximale du robot, il est nécessaire de lancer les balles pour les mettre à l'intérieur. Nous avons envisagé plusieurs solutions techniques pour lancer les balles :

— Une roue qui tourne rapidement et entraîne les balles au contact.
— Un slingshot de flipper (c'est la partie triangulaire qui renvoie la balle à l'aide d'un élastique et d'un système mécanique)
— Une soufflerie

Malheureusement, nous nous sommes vite rendus compte que le slingshot ne marcherait pas. En effet, les slingshots propulsent les balles avec un coup sec, mais les balles utilisées pour la coupe de robotique sont en mousse et amortissent donc beaucoup l'impact. La balle ne partait alors pas assez loin.

La soufflerie nous semblait plus complexe à mettre en œuvre que le système avec une roue, et nous avons donc commencé par tester la propulsion à l'aide d'une roue. Nous avons donc imprimé en 3D un prototype de canon pour réaliser les premiers tests (figure 16). En faisant tourner une roue au dessus de ce canon, nous avons constaté que ce système permettait d'envoyer les balles suffisamment haut, et nous avons donc conservé cette idée.

Figure 16 – Prototype de canon pour lancer les balles

Cependant, pour le premier prototype, nous avions un problème de précision. En effet, la roue n'était pas équilibrée et elle vibrait beaucoup. Cela était dû au fait que nous ne disposions pas de moyeu (intermédiaire entre l'axe moteur et la roue) adapté au moteur que nous utilisions pour le prototype. Les vibrations sont indésirables car

elles dissipent beaucoup d'énergie, et de manière fluctuante. Ainsi, certaines balles allaient beaucoup plus loin que d'autres.

Par conséquent, nous avons dû imprimer en 3D une roue adaptée à l'axe du moteur, mais cela ne marchait toujours pas bien. Nous avons donc voulu acheter un moteur et une roue dimensionnés ensemble. Cependant, cela s'est révélé plus difficile que prévu puisque tous les ensembles roue + moteur que l'on trouve dans le commerce sont conçus pour faire avancer un robot. Ils ont donc tous un réducteur, leur permettant d'avoir un couple important et une faible vitesse de rotation. Or, pour notre lanceur de balles, nous avions besoin que la roue tourne très vite (environ 5000 tours par minute). Nous n'avons donc pas trouvé de moteurs et roue qui nous conviennent. Nous avons tout de même essayé de commander des moteurs avec réducteur ayant les vitesses de rotations les plus élevées que l'on a trouvé (environ 500 tours par minute), mais ils se sont avérés trop lents.

Nous avons donc gardé notre moteur initial, en essayant de fabriquer une roue qui puisse tout de même fonctionner sur ce moteur. Nous avons utilisé la découpeuse laser pour découper la roue, et nous avons découpé l'axe central de la roue de telle sorte qu'il ait les mêmes dents que l'axe du moteur. Après plusieurs essais, nous avons obtenu une roue qui ne vibrait quasiment plus, et qui permettait de lancer successivement les 8 balles dans le château d'eau avec une précision suffisante.

4.3.2 • Rampe pour récupérer et stocker les balles

Dans un second temps nous avons conçu la rampe, permettant à la fois de récupérer les balles et de les stocker en attendant de les tirer. Pour pouvoir stocker les 8 balles, nous avons réalisé une rampe qui fait le tour complet du robot. Initialement, nous avons essayé de faire une rampe en fils de fer qui passent par des guides (figure 17). Cependant, cette méthode n'a pas fonctionné car la rampe n'était pas suffisamment lisse, notamment au niveau des guides, et les balles restaient bloquées au milieu de la rampe. Nous avons donc dû imprimer toute la rampe en 3D pour qu'elle soit suffisamment lisse pour que les balles ne se coincent pas.

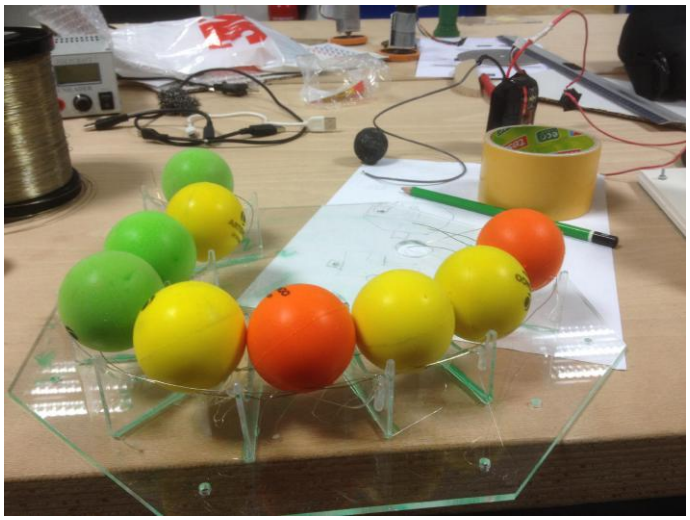

Figure 17 – Prototype de rail pour amener les balles au canon

Comme les dimensions de la rampe excèdent la taille du plateau des imprimantes 3D dont nous disposons (22cm par 22cm), nous avons séparé la rampe en 4 parties : le canon, l'entonnoir pour récupérer les balles, et 2 parties pour la boucle entre les 2 (figure 18). La séparation de la rampe en plusieurs parties a aussi permis une modularité : il est possible de changer le canon ou l'entonnoir sans tout réimprimer. Cela a été particulièrement utile pour l'entonnoir, car il nous a fallu faire plusieurs versions successives avant d'aboutir à un entonnoir qui permette de récupérer toutes les balles sans qu'aucune ne se coince ni ne sorte de la rampe.

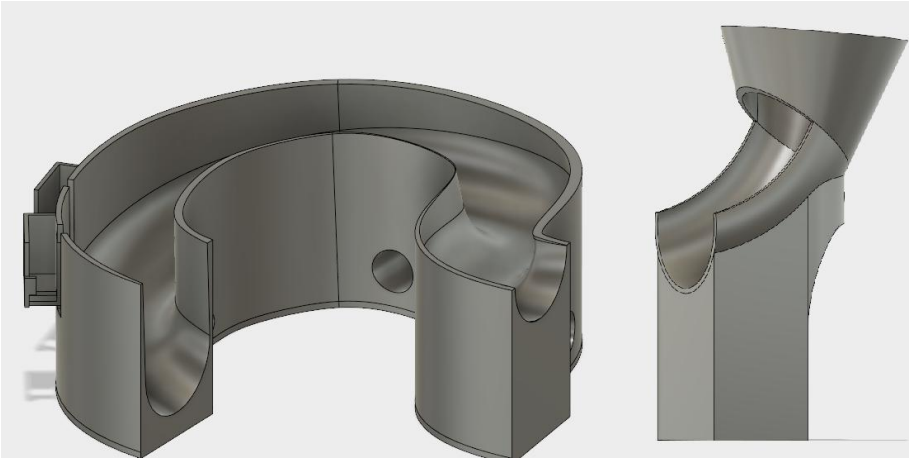

Figure 18 – Modèles 3D de la rampe (à gauche) et de l'entonnoir (à droite)

La figure 19 montre les prototypes successifs d'entonnoir, avec le plus ancien à gauche et le plus récent à droite. Avec le premier, lorsque toutes les balles tombaient en même temps dans l'entonnoir, certaines rebondissaient en dehors de la rampe. Nous avons donc ajouté un anneau pour être sûr que les balles restent à l'intérieur, ce qui a aboutit à une structure ressemblant plus à un entonnoir. Cependant, nous avions encore un problème : les balles restaient parfois coincées. Cela se produisait lorsque la balle du dessus était plus en avant que la balle en dessous. Nous avons corrigé ce problème en donnant une forme courbe à l'entonnoir. La version finale permet de récupérer toutes les balles sans qu'aucune ne reste bloquée.

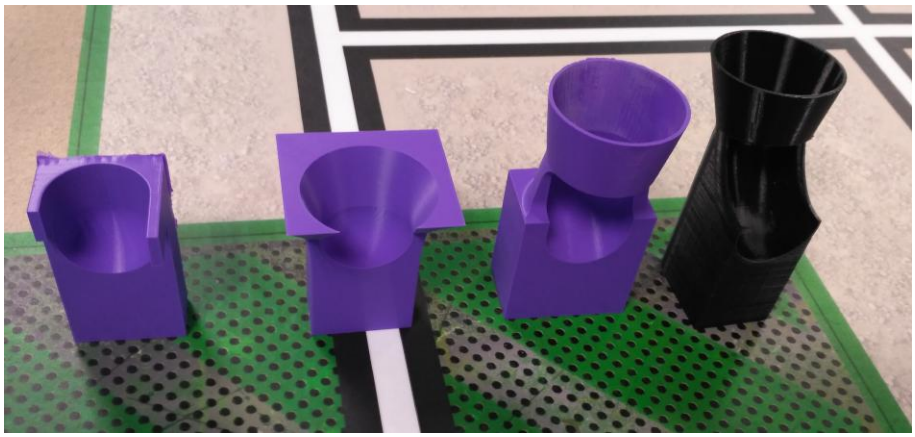

Figure 19 – Les prototypes successifs d'entonnoir

Pour ouvrir le loquet du récupérateur d'eau, nous n'utilisons pas d'actionneur particulier : le haut de l'entonnoir est à la hauteur du loquet, de sorte que le robot ouvre le loquet simplement en le poussant avec l'entonnoir. Ainsi, nous sommes sûrs que, lorsque le loquet est ouvert, l'entonnoir est bien en dessous, et qu'on récupère bien les balles. Pour tirer les balles, nous pensions initialement que, si on envoie toutes les balles en même temps, le passage d'une balle risque de ralentir la roue, et les autres balles seraient donc envoyées à des endroits différents. C'est pourquoi nous avons fait une porte avec deux servomoteurs, permettant de faire passer une balle à la fois. Cependant, nos tests ont montré qu'il est en fait possible de lancer toutes les balles en même temps tout en gardant une précision suffisante. Nous avons donc finalement utilisé un seul servomoteur, qui est initialement fermé pour stocker les balles, et que l'on ouvre pour les lancer. La rampe finale est illustrée en figure 20.

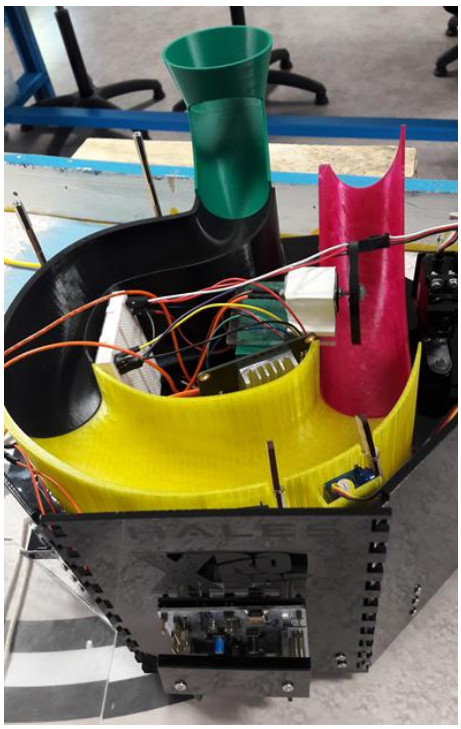

Figure 20 – Rampe finale

4.3.3 • Abeille

L'abeille est un dispositif entièrement mécanique qui doit être actionnée par le robot pour descendre la pente à l'avant du plateau et éclater le ballon en butté. Elle est soumise à un cahier des charges présent en annexe (figure 41) qui impose entre autre de restrictions sur les dimensions.

Pour que l'abeille ne tombe pas de la rampe, une première solution était de mettre des petites roues de part et d'autre de la rampe, le long de guides rigides. Cependant, les premiers essais ont été infructueux. En effet soit les roues serraient trop la rampe et alors l'abeille n'avançait pas, soit nous laissions un peu de jeu mais l'abeille tombait (il faut aussi prendre en compte le fait que nous avons réalisé la

rampe nous même, la face supérieure n'était pas parfaitement droite ce qui pouvait causer des déraillements). Finalement, nous avons changé d'idée et avons décidé d'imprimer en 3D des roues ayant, sur les côtés, des disques de rayon plus important que la roue (figure 21), pour apporter de la stabilité afin que la roue reste sur la rampe, un peu à la manière d'une roue de chemin de fer. Cela induit certes des frottements, mais les tests que nous avons réalisés ont montré que de telles roues permettaient de descendre la rampe. De plus, le jour de la Coupe, la rampe sera très lisse et l'ensemble roulera assurément mieux que lors des tests.

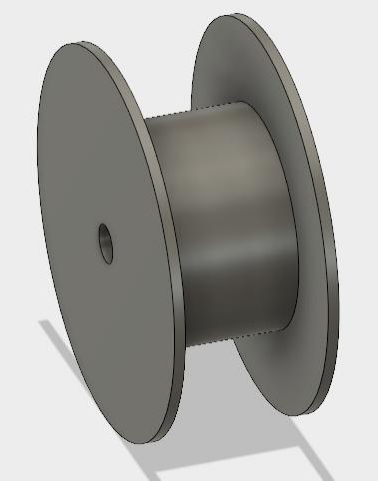

Figure 21 – Modèle 3D des roues de l'abeille

Pour la structure de l'abeille elle-même, nous avons utilisé des outils de construction "KNEX" (jeu). En effet, nous en avions dans le local du binet X-Robot, et nous nous sommes rendus compte que cela permettrait de fabriquer l'abeille rapidement. Afin d'ajouter une touche de personnalité à notre abeille (les seules contraintes concernent la taille), nous avons découpé un polytechnicien pour tenir l'aiguille permettant de percer le ballon et l'avons découpé au laser(figure 22). La longueur de l'aiguille est déterminée par la longueur totale de l'abeille, qui ne doit pas excéder 20 cm.

Figure 22 – Abeille

Pour être capable de pousser l'abeille, nous avons muni notre robot d'un bras, qui est initialement vertical et qui se déploie à l'horizontale à l'aide d'un servomoteur (figure 23) afin de de donner l'impulsion initiale en touchant l'abeille au niveau de l'un des deux montants verticaux qui encadrent le polytechnicien.

L'objectif est de placer le robot à proximité de l'abeille, puis de la pousser lors d'une rotation du robot principal. Avec son inertie, elle dévale la pente et éclate le ballon.

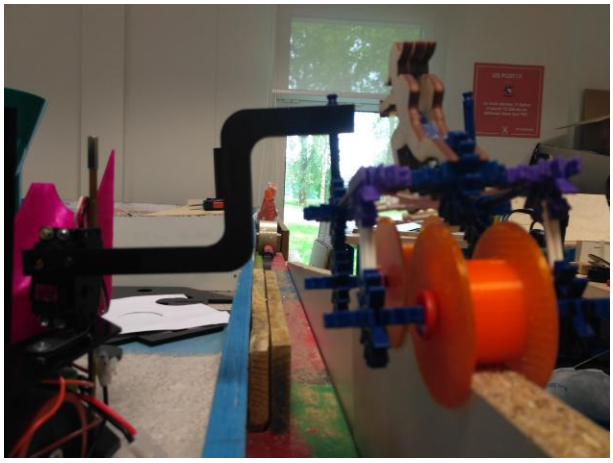

Figure 23 – Bras permettant de pousser l'abeille

4.3.4 • Panneau domotique

Le panneau domotique (voir cahier des charges en annexe figure 40) est un élément qui doit avoir deux états facilement discernables à la vue : allumé ou éteint. Nous avons réalisé un bicorne en LED, représenté en figure 24, qui s'allume lorsque l'interrupteur est fermé.

Composé de 65 LEDs, pour être en mesure d'alimenter le panneau il est nécessaire de réaliser un montage bien spécifique : il faut une tension de l'ordre de 2V aux bornes de chaque LED. Vu que nous utilisons une batterie 12V, cela implique de rassembler les LED par blocs de 6 et d'alimenter ces blocs en parallèle.

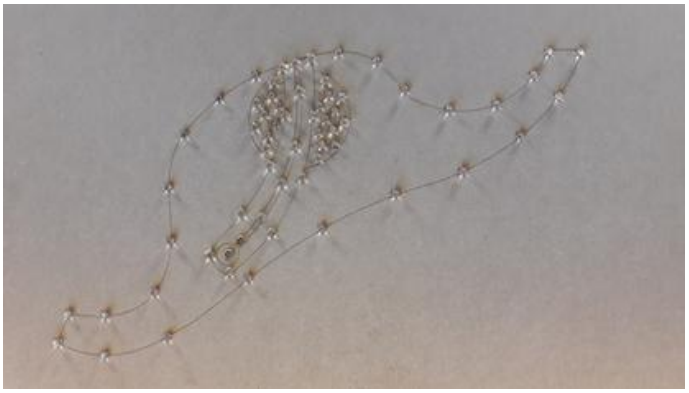

Figure 24 – Panneau domotique

Pour appuyer sur l'interrupteur, nous avons fait un système simple : un pare-choc, représenté en figure 25, est placé à l'avant du robot, à la hauteur du bas de l'interrupteur, car l'interrupteur se ferme lorsqu'on appuie sur sa partie basse. Le robot doit donc simplement avancer face à l'interrupteur pour le fermer.

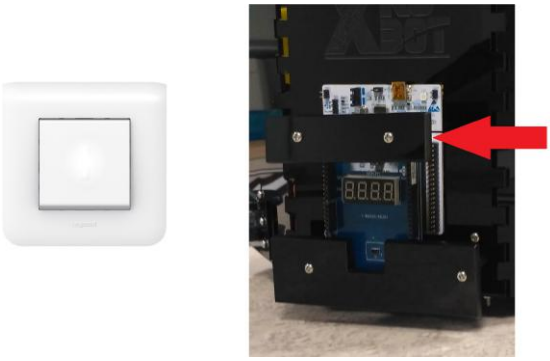

Figure 25 – A gauche : l'interrupteur. A droite : rectangle permettant d'appuyer sur l'interrupteur

4.4 Asservissement

4.4.1 • Enjeux et principe de l'asservissement

• Intérêt d'un asservissement

Lorsque l'on souhaite faire avancer un robot, ou plus généralement si l'on souhaite transmettre une consigne à un quelconque actionneur en robotique, on ne peut transmettre à celui-ci qu'une consigne, mais la réalité, ce que l'on appelle la sortie, sera sûrement différente de la consigne. Dans le cas de nos moteurs par exemple, en raison de la non-linéarité entre la tension qui sera transmise et la vitesse de rotation du moteur, et de la différence qu'il peut y avoir entre deux moteurs du même modèle, si l'on transmet la même tension d'entrée aux deux moteurs faisant tourner les roues de notre robot, ces deux moteurs ne tourneront pas en sortie à la même vitesse : le robot n'avancera alors pas en ligne droite.

Afin d'éviter ce phénomène, il y a besoin d'avoir un retour sur la consigne qui est transmise aux moteurs, afin par exemple d'augmenter la consigne si la sortie effective n'est pas suffisante, ou à l'inverse de diminuer celle-ci si la sortie est trop importante. Le schéma suivant illustre donc le principe général de tout asservissement. Ce schéma nous montre également la nécessité d'avoir des capteurs afin de mesurer la sortie.

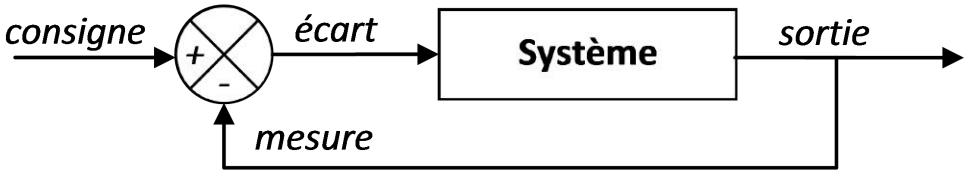

Figure 26 – Principe d'un asservissement

Dans notre cas, il faut que la position du robot soit précisément celle commandée tout le long de la partie, que le robot roule droit et atteigne sa destination. Pour atteindre cet objectif, le principe est toujours de comparer la sortie mesurée, de la comparer avec l'entrée théorique et d'en déduire une nouvelle consigne qui prend en compte les erreurs pour les corriger.

• Enjeux et exigences de la coupe

Pour la coupe, notre robot doit pouvoir se déplacer de manière complètement autonome : il n'est pas télécommandé, et doit donc être capable de se déplacer sur le plateau de manière préprogrammée pendant les 100 secondes que dure un match. Ce que nous avons choisi de faire afin de connaître la position de notre robot à chaque instant durant la partie, c'est de nous fier à notre asservissement : de fait, si nous avons ordonné à notre robot de tourner de 45 degrés et d'avancer de 1m à partir du point de départ, nous connaissons sa position, en supposant que notre asservissement soit précis, ce qui est le point clé.

Étant donné que pour effectuer toutes nos actions (récupérer les balles, les envoyer, pousser l'abeille, allumer l'interrupteur) nous avons besoin d'être positionnés de manière très précise sur le terrain, il est nécessaire que notre asservissement soit très précis, avec comme objectif d'être positionné au quart de centimètre près sur une ligne droite d'un mètre, et d'être à deux degrés près en angle sur une rotation de 360 degrés.

Nous n'avons pas d'autre système (GPS, utilisation des lignes au sol du terrain, balises sur le terrain...) pour nous positionner sur la carte. En effet, l'utilisation d'un GPS n'aurait pas été suffisamment précise pour nous positionner au centimètre près. Par ailleurs, l'utilisation d'un système de balises (que le règlement nous autorise à placer autour du terrain afin de se positionner) s'avère être compliquée à mettre en place, et trop sensible aux interférences si l'équipe adverse utilise le même système. Nous avançons donc « à l'aveugle » mais avons confiance en notre « oreille interne » qu'est notre asservissement. A certains endroits du plateau de jeu, les coins par exemple, nous pourrons malgré tout nous replacer grâce à nos capteurs de distance que nous utilisons pour l'anti-collision. Ces repositionnements périodiques nous permettent de ne pas accumuler et propager les éventuelles erreurs de l'asservissement. Enfin, nous avons optimisé notre parcours afin que les actions qui demandent un positionnement exact (récupération des balles dans le réservoir d'eau par exemple) soient exécutées en premier. Ainsi une erreur de quelques centimètres n'impactera pas nos dernières actions.

• Types d'asservissement : le PID

Il existe plusieurs types d'asservissement : en position, en vitesse, en angle, etc. qui correspondent à la grandeur qui est mesurée pour être comparée avec l'entrée (cf schéma précédent). Ils ont des intérêts différents : un asservissement en position permet d'être précis à l'arrivée, mais ne permet pas de bien contrôler le mouvement entre la position initiale et celle finale. Au contraire, un asservissement en vitesse permet d'éviter certains pics de vitesse et de rendre le mouvement plus fluide mais ne permettra pas d'assurer un positionnement précis à l'arrivée. Les systèmes complexes d'aujourd'hui possèdent des doubles boucles d'asservissement pour régler ces problèmes.

L'asservissement le plus classique et le plus répandu pour le déplacement est le PID (« proportionnel, intégrale, dérivée »). Le principe est de comparer à la fois le signal et la commande mais aussi l'évolution de l'écart (cf schéma précédent) que l'on appelle plus souvent l'erreur. L'intérêt de cet asservissement est d'assurer la stabilité de l'asservissement (si jamais le système s'écarte de la position, il ne risque pas de diverger : l'asservissement est donc possible dans toutes les configurations) mais aussi la rapidité de l'asservissement et de limiter les oscillations, qui sont souvent néfastes pour l'asservissement. Nous reviendrons par la suite sur la façon de régler les 3 constantes qui permettent de donner une bonne proportion entre les différentes corrections.

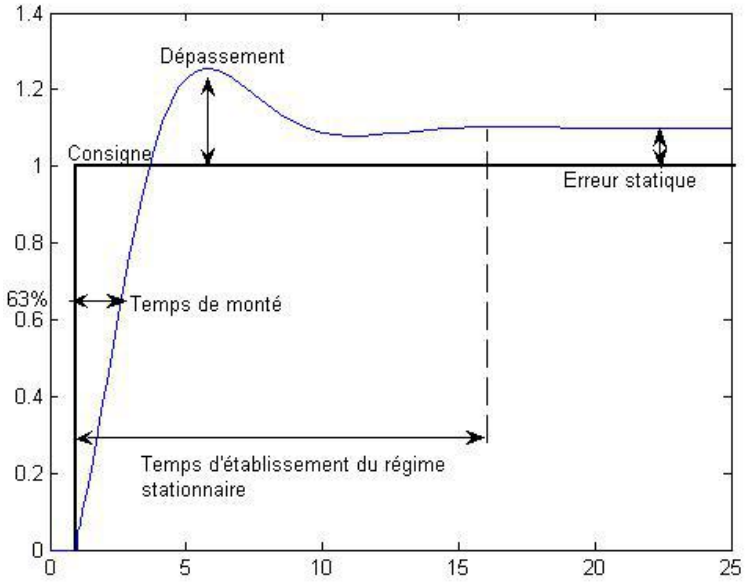

Figure 27 – Différents biais d'un asservissement

L'influence des effets inertiels sur l'asservissement

L'étude du glissement en cinématique et en dynamique est une discipline très complexe car elle implique des phénomènes microscopiques (le frottement est intimement lié à la structure et la géométrie microscopique des matériaux). Nous allons néanmoins présenter l'influence que peuvent avoir ces effets sur le déplacement et ici sur l'asservissement.

Commençons par introduire deux facteurs clés pour comprendre les effets inertiels : le coefficient d'adhérence et celui de glissement. Le coefficient d'adhérence se définit comme le rapport entre deux forces : la force de freinage et le poids, ou encore le rapport entre deux accélérations (en simplifiant par la masse). Ce coefficient est absolument central car le poids ne s'applique pas uniquement sur le centre de gravité mais de manière uniforme. Ainsi, à coefficient d'adhérence constant, si la masse n'est pas répartie de manière uniforme, le freinage ne le sera donc pas plus, ce qui va amener le robot à dériver d'un côté.

Le deuxième coefficient clef est celui de glissement défini par le rapport de la vitesse circonférentielle d'une roue et de la vitesse de translation, c'est à dire entre la distance que devrait parcourir le robot s'il adhérait parfaitement et la distance réellement parcourue. Ce coefficient est nécessairement compris entre 0 et 1 et est une mesure du patinage des roues. Le problème lié au glissement est celui du repérage dans l'espace. En effet, comme expliqué précédemment, nous n'utilisons pas de capteurs pour nous repérer, nous nous repérons par odométrie (reconstruction de la trajectoire réelle grâce aux capteurs de déplacement). Or la grandeur mesurée est la rotation de l'axe du moteur ; s'il y a du glissement, nous ne sommes pas en mesure de savoir la vraie distance parcourue par le robot.

Voilà les deux principaux phénomènes inertiels que nous avons rencontré lors de l'élaboration du robot. Nous développerons dans la partie 4.4.3 notre stratégie pour assurer un asservissement précis qui tient compte de la gravité.

Trajectoires et asservissement

Amener le robot depuis une certaine position et un certain angle vers une autre position est un problème de contrôlabilité classique, qui peut être résolu de nombreuses manières. Le contrôle est ici appliqué sur la vitesse, et le retour de nos capteurs permet d'assurer la contrôlabilité, c'est à dire qu'il est théoriquement toujours possible d'amener le robot à n'importe quelle position (théorème de Kalman). Le vrai problème est donc de choisir le bon critère pour trouver la trajectoire optimale. Une idée pourrait de limiter le temps de parcours en fixant une vitesse maximum, ce qui impliquerait des résolutions d'équations différentielles. Cela fait beaucoup de calculs pour une carte Arduino et nous avons préféré choisir un critère de simplicité. Nos trajectoires seront décomposées en rotations et lignes droites, ce qui prend plus de temps mais qui nous permet d'assurer une meilleure précision et qui facilite grandement l'odométrie.

4.4.2 • Présentation de notre matériel et de ses limites

• Les cartes Arduino

Afin de communiquer avec les différents actionneurs et de recevoir les informations des capteurs, nous utilisons une carte de la marque Arduino, qui sont les cartes les plus répandues chez les programmeurs amateurs. Ces cartes disposent de nombreux canaux de communication entrée-sortie (on parle de PINs) et d'un microcontrôleur facilement programmable en langage C++ avec un logiciel fourni par Arduino.

La carte que nous utilisons, l'Arduino MEGA, est pratique car elle dispose de très nombreux pins d'entrée-sortie (figure 28). Ceci nous est nécessaire car la carte Arduino MEGA sur laquelle sera programmée l'asservissement est la carte maîtresse de notre robot : C'est aussi elle qui va donner l'ordre d'exécuter toutes les autres actions du robot (ouvrir les servomoteurs pour laisser passer les balles, faire tourner la roue pour lancer les balles, déployer le bras afin de pousser l'abeille...). Elle doit donc être reliée à tous les moteurs pour les commander (via une carte de puissance, voir section 5.1 où l'on parle de l'architecture électrique du robot), et à tous les capteurs (codeurs incrémentaux, capteurs de distance) pour recevoir leurs informations.

La carte MD25

Cette carte électronique permet de contrôler deux moteurs à la fois avec un asservissement PID, et sert aussi de pont en H (permet de faire fonctionner le moteur dans les deux sens). Un certain nombre de fonctions sont déjà codées sur cette carte afin de pouvoir lui donner des instructions simples telles qu'un profil d'accélération

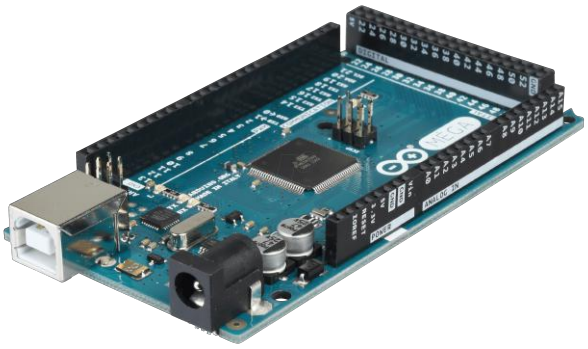

Figure 28 – Carte Arduino MEGA

par exemple (idéalement il suffit d'accélérer avec l'accélération maximale pendant la première moitié du trajet puis de décélérer pendant la seconde). Tous les calculs sont ensuite directement réalisés par la carte d'asservissement. Afin de contrôler la MD25, il faut utiliser une carte Arduino, et le protocole I2C pour faire communiquer les deux cartes. C'est un protocole maître-esclave assez basique et souple (il peut y avoir plusieurs maîtres par exemple). Il existe une bibliothèque "wire" sur Arduino afin d'utiliser simplement ce protocole.

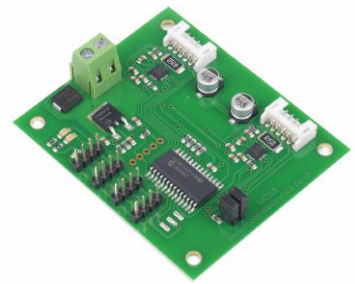

Figure 29 – Carte d'asservissement MD25

• Moteurs et codeurs incrémentaux

Afin de déplacer notre robot, nous avons utilisé deux moteurs classiques de la marque pololu, de rapport de réduction 19 :1 (le moteur fait 19 tours pour que l'arbre moteur qui entraîne la roue fasse un tour), alimentés par un courant continu 12V. Ce moteur est suffisamment puissant pour faire avancer notre robot correctement, a le mérite d'être fiable et simple d'utilisation.

Étant donné que nous ne recherchons pas une vitesse importante (le robot avance pour l'instant à un tiers de ce que pourrait être sa vitesse maximale), nous aurions pu choisir des moteurs plus puissants avec plus de couple (donc un rapport de réduction plus élevé). Ceci nous aurait permis de pouvoir nous déplacer à vitesse très faible, alors que nous sommes pour l'instant obligés de donner à notre robot une vitesse

minimale pour qu'il ne se bloque pas : Nos moteurs, même s'ils sont satisfaisants, pourraient être plus puissants.

Pourquoi ne pas avoir changé de moteurs ? Tout d'abord parce que ces moteurs ont le mérite d'être très compacts, et nous manquons de place sur notre robot. Nous avions des moteurs plus puissants mais les deux moteurs alignés dépassaient la taille autorisée pour le robot. De plus, ces moteurs sont fournis avec des capteurs (codeurs incrémentaux) très précis et fiables, qui nous permettent de mesurer les tours du moteur et donc les tours effectués par les roues avec précision.

Un autre défaut des moteurs, mais qui n'est pas spécifique à ce modèle, est la relation non-linéaire qui existe entre la tension d'entrée qu'on lui transmet et la vitesse de rotation que l'on obtient en sortie. Ainsi, la vitesse que l'on va obtenir en sortie n'est pas prévisible en fonction de la consigne que l'on va donner, et il faut donc regarder cette relation expérimentalement. De plus, la non-linéarité de cette relation et son caractère assez empirique font que pour une même tension, les deux moteurs ne tourneront pas forcément à la même vitesse. C'est d'ailleurs pour cette raison qu'il est nécessaire d'avoir un asservissement.

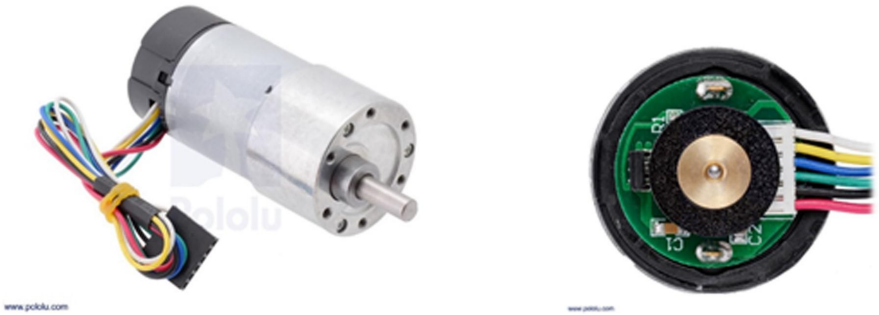

Figure 30 – Moteur utilisé et codeur incrémental rattaché

Chaque moteur est monté avec un codeur incrémental (derrière le cache noir sur la photo) dont le principe est le suivant : Un disque rotatif est lié à l'arbre moteur et va tourner avec lui. Ce disque comporte deux pistes (A et B) comme sur le schéma qui suit. De part et d'autre de chaque piste, il y a une LED fixe et un récepteur de l'autre côté. De fait, dès que la LED est en face d'une fenêtre transparente, le capteur enregistre un signal (1 logique), et redescend à 0 dès que la fenêtre est passée.

Le décompte du nombre de signaux enregistrés par les deux capteurs permet de déterminer l'angle avec lequel a tourné le disque depuis le début du décompte. Il suffit pour cela de connaître le nombre de fenêtres (on parle d'incréments) qu'il y a sur le disque. Notre codeur en compte 16 par piste. Or les deux pistes étant déphasées, il y a 4 configurations possibles (A vaut 0 ou 1 et B aussi) entre chaque fenêtre de la piste A par exemple. Il y a donc 64 états possibles à chaque tour,

c'est-à-dire qu'il est possible de mesurer l'angle du disque au 64ème de tour près. C'est pourquoi en multipliant par le rapport de réduction de 19, il est possible de connaître l'angle de l'axe de sortie, donc de la roue, au 1216ème de tour près. Mais comment connaître le sens de rotation ? Les deux pistes étant légèrement déphasées (en quadrature), il suffit de regarder quel est l'état d'une des pistes lorsque l'autre état varie pour connaître le sens de rotation.

En pratique, nous ne mesurons que le passage de la piste A de 1 à 0 (variation repérée grâce à la fonction *attachInterrupt* d'Arduino) et regardons alors quel est l'état de la piste B (pour avoir le sens de rotation). Nous n'avons donc qu'un quart des configurations possibles soit 304 incréments par tour, ce qui est largement suffisant à notre niveau de précision.

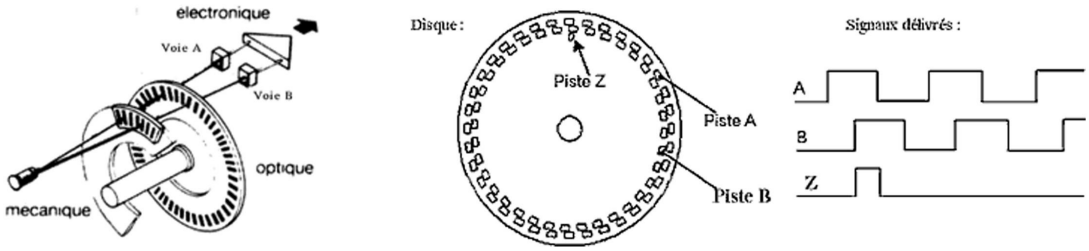

Figure 31 – Schématisation du fonctionnement d'un codeur incrémental

4.4.3 • Différentes solutions implémentées au fur et à mesure de l'année

• Solution envisagée initialement : asservissement automatique(MD25)

Sur les conseils de l'équipe de l'école qui a participé l'année dernière, nous avons commencé à implémenter un asservissement en se servant de la carte, utilisée par de nombreuses écoles pour leur robot.

Nous avons premièrement eu des difficultés avec le protocole de communication I2C présenté ci-dessus, mais nous avons finalement réussi à faire communiquer les cartes. Cependant, nous n'utilisions pas les moteurs recommandés par le distributeur de la carte MD25, ce qui est probablement la cause du grillage de plusieurs de nos cartes d'asservissement. De plus, l'opacité du fonctionnement de l'asservissement de cette carte ne nous satisfaisait pas, et nous avons opté pour un asservissement plus transparent que nous avons réalisé nous-mêmes.

• Solution retenue : implémentation du PID

Nous avons opté pour implémenter nous-mêmes l'asservissement PID de notre robot afin de pouvoir en maîtriser tous les paramètres et de pouvoir le régler exactement comme nous le souhaitions. Notre asservissement est un asservissement en position : Nous ne cherchons pas à maîtriser exactement la vitesse de déplacement du robot mais uniquement sa position afin qu'il parcoure exactement la distance souhaitée, et en ligne droite bien entendu.

Pour cela, les variables que nous mesurons en sortie (voir schéma section 4.4.1) sont les informations fournies par les codeurs incrémentaux de chaque roue (que nous appellerons $leftClicks$ pour la roue gauche et $rightClicks$ pour la roue droite), et nous nous intéressons plus spécifiquement à deux données :

La distance parcourue (en nombre d'incréments) $d = \frac{leftClicks+rightClicks}{2}$

L'avance de la roue droite sur la roue gauche : $erreurAngle = rightClicks - leftClicks$

Notre asservissement fonctionne alors de la manière suivante : Nous transmettons à chaque roue une consigne en vitesse :

$consigneDroite = v - correction$
$consigneGauche = v + correction$

Il y a donc deux termes à ajuster en permanence : La vitesse v transmise, afin de parcourir exactement la distance voulue, et la valeur de $correction$, variable qui assure que les deux roues aient en permanence parcouru la même distance, donc que le robot avance bien en ligne droite.

La vitesse v est déterminée de la manière suivante :

- Elle augmente avec une accélération constante tant qu'on a parcouru moins que $0,3 \times distanceTotale$, où $distanceTotale$ est la distance à parcourir.
- Elle est ensuite constante.
- Elle décroit ensuite de la manière suivante afin que la distance parcourue soit exactement la distance que l'on souhaite atteindre. Si la distance parcourue est supérieure à $0.85 \times distanceTotale$, alors

$$v = \frac{distanceàparcourir - distanceparcourue}{0.15 \times distanceTotale} \cdot (vmax - vmin) + vmin$$

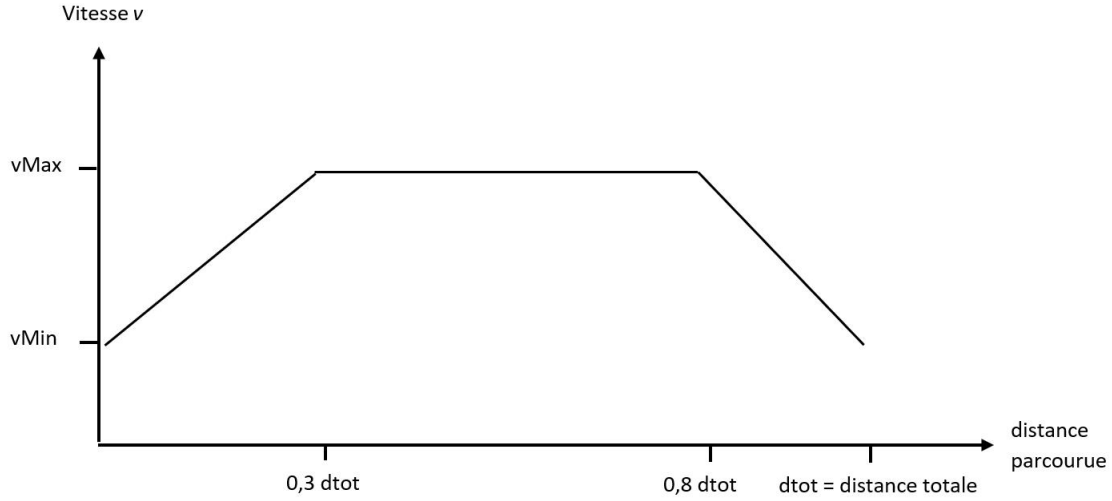

Figure 32 – Consigne en vitesse v

L'asservissement comme on l'entend et comme nous l'avons présenté ne porte pas sur la distance parcourue. Au contraire, le PID est utilisé afin d'assurer que le robot ait une trajectoire droite : Pour les raisons déjà détaillées (non-linéarité du moteur...), les deux roues ne vont pas avancer à la même vitesse. Le correcteur PID sera donc à chaque instant utilisé pour déterminer la correction qu'il faudra appliquer afin de revenir à une valeur de $erreurAngle$ nulle :

$$correction = K_p \cdot erreurAngle + K_d \cdot \frac{\mathrm{d}(erreurAngle)}{\mathrm{d}t} + K_i \cdot \int_0^t erreurAngle(t)\,\mathrm{d}t$$

• Réglage des constantes

Pour que la correction soit optimale, il faut régler les différentes constantes (K_p, K_d, K_i) du PID. Quels sont les rôles de ces constantes ?

La composante proportionnelle K_p est essentielle, car c'est elle qui va nous informer sur l'état du système : Si l'erreur en angle est importante, il faut appliquer une correction importante afin de revenir vers la valeur voulue, en l'occurrence zéro. En pratique, nous avons commencé par régler cette constante-ci en fixant les autres constantes à zéro. Si K_p est trop petit, le système ne va pas converger assez vite, il va falloir trop de temps au robot pour se remettre droit. Si la correction proportionnelle est trop importante, le système va converger trop rapidement vers la consigne et osciller autour de la position voulue : le robot va zigzaguer.

Ensuite, nous avons réglé la composante dérivée K_d. Cette composante permet d'améliorer la stabilité du système en diminuant les oscillations. En effet, si le système converge trop vite vers la valeur voulue, cette composante va avoir comme effet de diminuer la correction afin d'éviter un dépassement. A l'inverse, si le système s'écarte de la consigne, la correction va être très importante afin de ne pas

s'écarter davantage. Néanmoins, si K_d est trop importante, le système va perdre en précision car une sortie stable sera préférée à une sortie précise.

Enfin, la composante intégrale K_i permet de gagner en précision en empêchant la formation d'une erreur statique (cf schéma de la section Type d'asservissement : le PID). En effet, si l'on est légèrement écarté de la consigne pendant longtemps (erreur statique), la composante intégrale va devenir très importante et le système va être corrigé afin de converger vers la valeur voulue. Cette composante se règle en dernier.

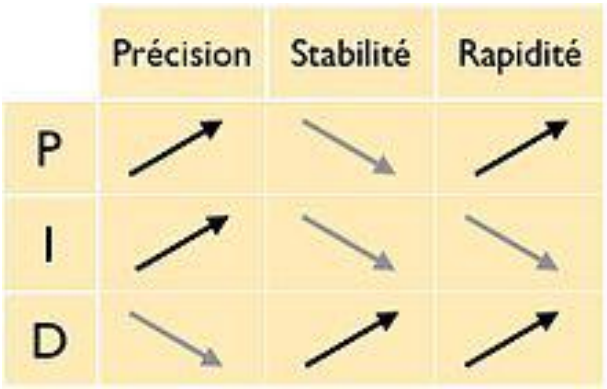

Figure 33 – Influence des constantes du PID

Prise en compte de la non linéarité et des effets inertiels

Comme nous avons essayé de l'expliquer plus tôt, l'asservissement d'un robot n'est pas parfaitement fidèle aux modèles que nous avons pu étudié par le passé, et nous avons donc du nous familiariser et nous adapter à ces phénomènes non pris en compte initialement.

La non linéarité de la réponse face à notre commande nous est apparu assez claire dès le début : du fait du frottement sec, le sol exerce un couple résistant sur le moteur (qui est d'ailleurs plus élevé à l'arrêt qu'en mouvement) qui impose une puissance minimale à fournir au moteur. Celui-ci est contrôlé par un hachage de la tension (on fait varier le rapport cyclique, qui est défini, sur une période temporelle du signal, comme le temps à la valeur maximale sur celui à celle minimale. Sur la figure, cela revient à faire varier τ).

Ainsi, plus la vitesse demandée au moteur est faible, plus la puissance fournie l'est. Il n'est donc pas surprenant que lorsque nous demandions de très faibles vitesses, le robot ne puisse avancer et que pour de faibles vitesses, les effets du couple résistant soient non négligeables, alors qu'ils le sont à grande vitesse (par rapport au couple moteur). Nous avons donc introduit une vitesse minimale dans notre code, afin que la réponse reste le plus linéaire possible par rapport à l'entrée .

En revanche, nous avons mis plus de temps à comprendre la portée des effets inertiels, en particulier de l'adhérence. Bien que nous connaissions les phénomènes

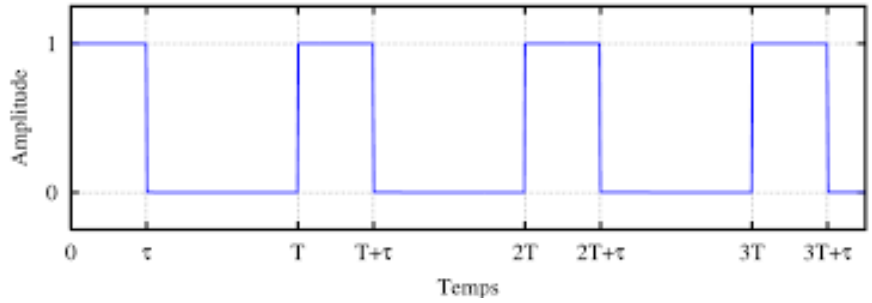

Figure 34 – Exemple de signal de commande du moteur

décrits en section 4.4.1 sur les effets inertiels, nous pensions qu'ils seraient infimes et et qu'il n'était pas nécessaire de les prendre en compte. Nous avons ainsi longtemps pensé que la source de nos erreurs était lié à notre code, et non à la répartition du poids de notre robot. Cependant, nous avons réalisé plusieurs expériences avec une masse de 500g, et nous avons directement mis en relation la position du centre de gravité du robot et la trajectoire du robot en ligne droite. En effet, nous avons obtenu un écart de dérive (déplacement latérale sur la ligne droite) de plus de dix centimètres sur une ligne droite d'un mètre.

Afin de résoudre ce problème, nous nous sommes procurés de petites masses d'environ 100g, et nous avons équilibré le robot (de manière expérimentale) une fois celui-ci terminé, et nous avons considérablement réduit la dérive de notre robot.

Le dernier phénomène que nous avons dû prendre en compte était bien sûr le glissement. En vu du manque de temps, nous n'avons pas décidé de modéliser le phénomène de glissement, et avons décidé de directement nous mettre dans les conditions de la Coupe pour pouvoir inclure le glissement dans le choix de nos constantes de réglage. Ainsi nous avons réalisé tous nos tests sur le revêtement qui sera utilisé lors de la Coupe, et dès le début nous avons lesté notre premier modèle de base roulante pour simuler le poids réel du robot final. Nous avons enfin introduit une constante de proportionnalité nous permettant d'adapter les distances (nous savons que le robot parcourt moins que ce qui lui est demandé à cause du glissement, mais ce dernier sera toujours identique si les conditions extérieures ne changent pas). Il nous a suffit de déterminer expérimentalement cette constante et de l'utiliser pour être précis en distance lors de nos lignes droites et de nos rotations. Ce n'est pas la même constante car le mouvement est différent, et la matrice d'inertie n'est pas isotrope. Il était donc nécessaire de déterminer une constante pour chaque mouvement.

4.5 Plateau et environnement

En binôme, nous avons entrepris depuis le mois de septembre de construire une reproduction du plateau de jeu sur lequel va évoluer le robot le jour de la compétition afin d'effectuer les tests préparatoires à la Coupe dans les conditions les plus proches de la réalité. Après avoir rassemblé les matériaux et outils nécessaires, nous nous sommes d'abord attaqués au château d'eau ainsi qu'à la station d'épuration, puis les distributeurs d'eau, et enfin la rampe de l'abeille et l'abeille elle-même.

Il est primordial de respecter au mieux les matériaux et dimensions donnés dans les règles annuelles de la Coupe de France de Robotique. En effet, si nous faisions pendant des mois les tests des capteurs et de l'asservissement sur un matériau différent, cela ne pourra pas marcher le jour J, car les interactions du robot avec son environnement ne seront pas les mêmes. C'est notamment important pour la rampe de l'abeille, qui est bien plus lisse sur les vidéos de la compétition que sur notre modèle au DrahiX.

Dès lors, une grande partie du travail a été manuelle, avec l'achat de matériaux (bois d'épaisseur 22mm, PVC...) les plus proches des références données dans les règles, qui ont ensuite été travaillés à l'aide d'une scie sauteuse, puis poncés. Le travail, qui se doit d'être méticuleux, s'est avéré plus long que prévu, notamment à cause des emprunts de matériel successifs.

Figure 35 – Le plateau en construction

La principale difficulté rencontrée a été le changement inopiné fin octobre dans

la disposition, la taille et la forme de certains des éléments du plateau de jeu (station d'épuration et distributeur notamment), car auparavant les règles avaient effectivement été annoncées comme provisoires, mais nous nous attendions à des changements relatifs au robot mais pas au terrain. Nous nous sommes adaptés au mieux, et avons finalement produit une station d'épuration extrêmement proche du résultat visé (figure 36).

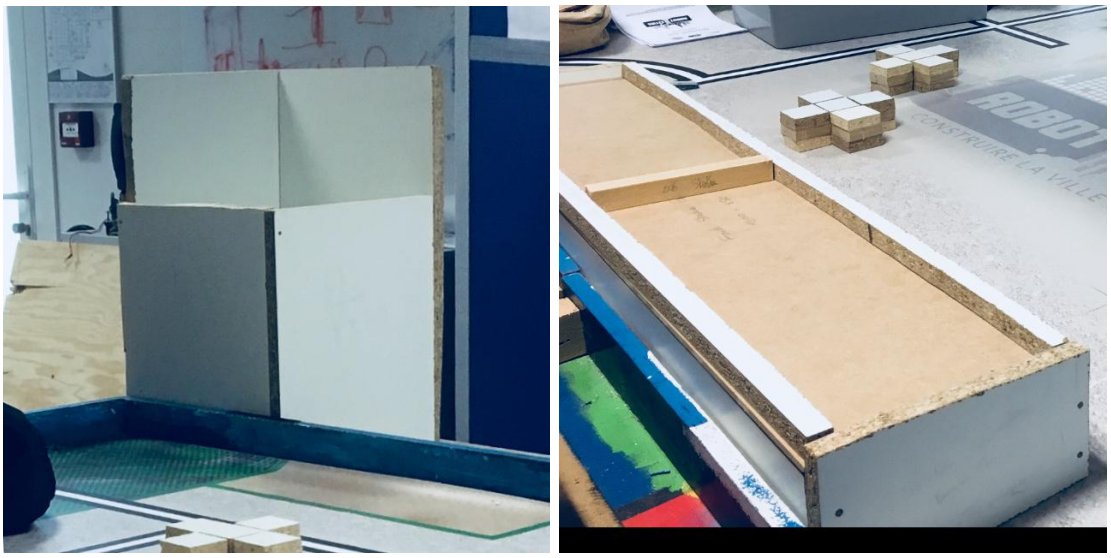

Figure 36 – Le château d'eau et la station d'épuration

5 MISE EN COMMUN

5.1 Architecture électrique

La gestion de la structure électrique du robot ne fait pas l'objet d'un des pôles mais est capitale, en particulier elle doit respecter certains critères du cahier des charges au niveau de l'alimentation et avoir un arrêt d'urgence (voir figure 39 en annexe).

De plus il faut s'assurer que le montage soit capable de répondre aux besoins des différents composants : cartes de commande, de puissance et actionneurs. Ceci nécessite la mise en place de circuits de commande et de puissance isolés pour protéger les éléments sensibles des pics de courants et les alimenter avec les tensions prévues.

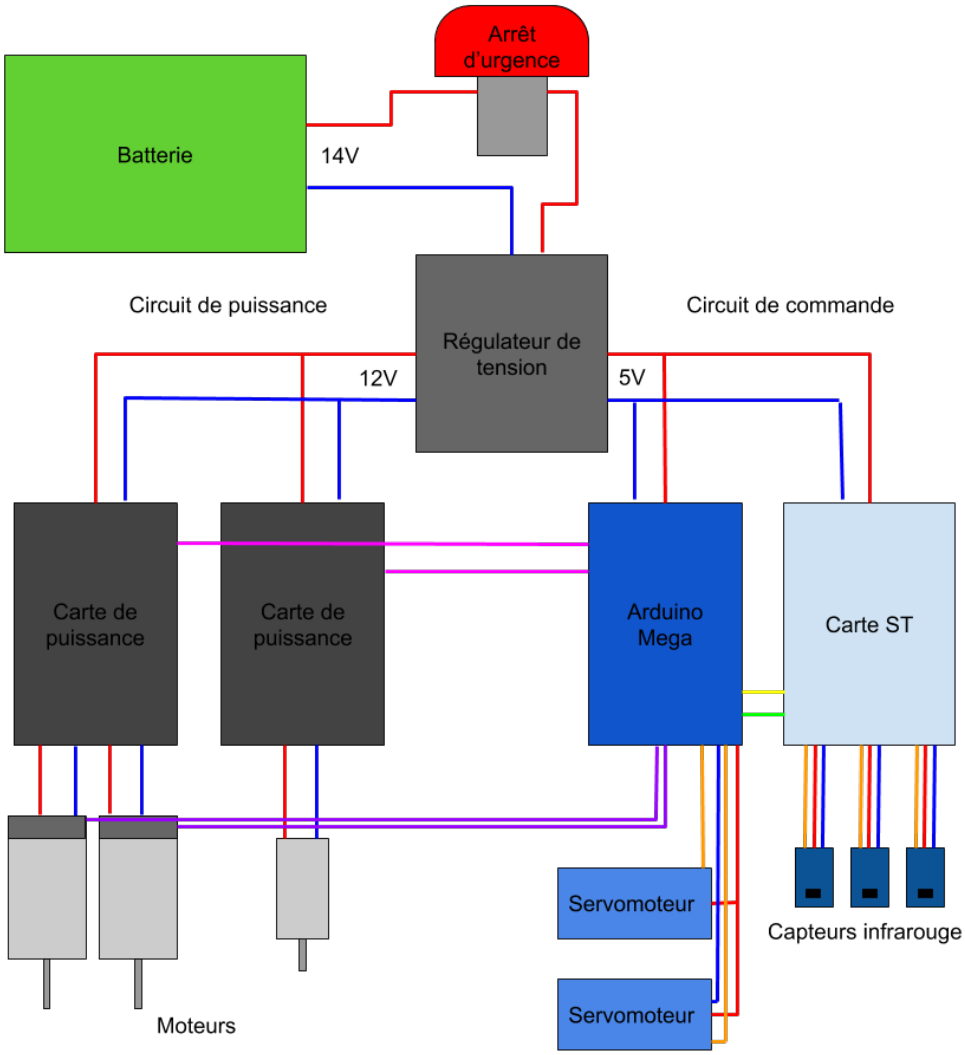

Figure 37 – Schéma électrique

Pour cela on propose l'architecture schématisée figure 37 qui utilise un régulateur de tension à deux sorties (12 et 5V) pour créer deux circuits isolés alimentés par une seule batterie. Il y a communication entre ces deux parties du circuits par les retours des codeurs incrémentaux (à la base des moteurs asservis) et par le signal de commande émis par la carte Arduino qui dicte la tension d'alimentation de sortie délivrée aux moteurs par les cartes de puissance.

Le bouton d'arrêt d'urgence doit provoquer l'immobilisation du robot, ce qui peut être fait de façon informatique ou électrique. Nous avons fait le choix d'un arrêt électrique en coupant toute source d'alimentation à l'activation, ainsi le robot ne sera pas capable de repartir en cas d'utilisation du bouton (qui de toute façon est utilisé en situation non contrôlée) mais nous avons la garantie qu'il s'arrêtera effectivement dans les meilleurs délais.

5.2 Architecture du code de la carte maîtresse

La carte Arduino est la carte maîtresse, c'est elle qui commande tous les actionneurs et récupère les informations de tous les capteurs. Elle décide donc de toutes les actions à effectuer, de leur ordre, et de la priorisation des informations qu'elle reçoit, suivant la stratégie que nous avons fixée.
L'architecture globale du code est résumée dans le schéma suivant :

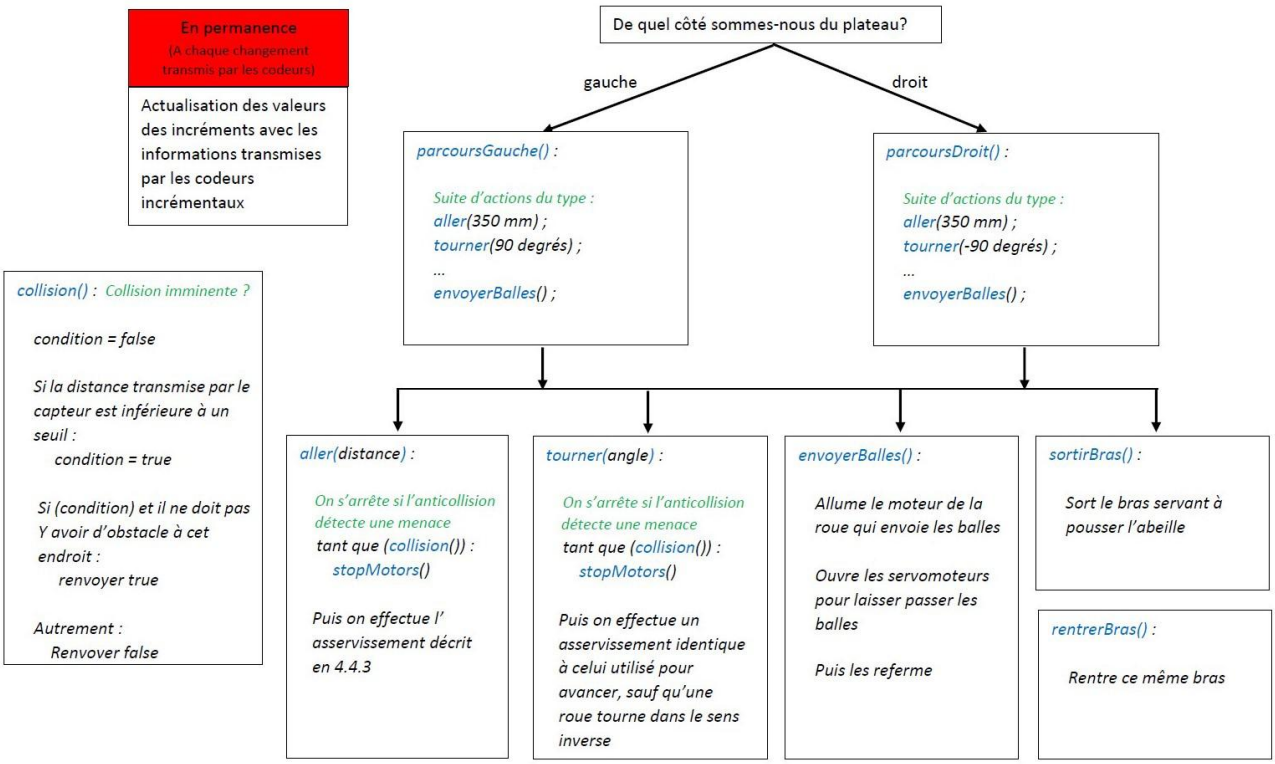

Figure 38 – Architecture du code Arduino

5.3 A faire avant la coupe

À la date de rendu du rapport final (le 30 avril) il reste encore un peu de travail à effectuer avant de se rendre à la coupe, le 9 Mai. Le robot est, à l'heure de l'écriture de ces lignes, intégralement monté (que ça soit au niveau physique et électronique) et capable d'effectuer toute les actions séparément : avancer, tourner, lancer les balles et déployer les servomoteurs sur commande.

Il reste cependant à assembler toute ces actions pour réaliser le parcours prévu par la stratégie (voir figure 10). Ceci implique en particulier d'affiner l'asservissement pour s'assurer que l'écart de trajectoire à l'arrivé est suffisamment faible pour être en mesure d'appuyer sur l'interrupteur en fin de parcours, éventuellement en rajoutant une phase de calibration au milieu du parcours, par exemple en s'appuyant sur un mur pour fixer l'angle du robot de façon fiable.

Nous sommes également prêts à réaliser des ajustements structurels si besoin, par exemple modifier la forme ou la taille du bras en fonction pour s'assurer de pousser l'abeille avec une fiabilité aussi grande que possible. Il est également envisageable de lester le robot pour augmenter sa masse si l'asservissement l'exige.

Enfin le panneau domotique n'est pas complètement opérationnel, il reste à câbler les connections entre les différents groupes de LEDs et l'alimentation. Sa conception est un peu tardive car jugée comme n'étant pas prioritaire vis-à-vis des autre composants à développer qui eux sont absolument nécessaire au fonctionnement du robot.

Pour terminer, nous avions prévu de longue date, si le robot était fini à temps, de mettre en place un match amical face à nos voisins de l'ENSTA. Nous allons donc tenter d'organiser cette rencontre d'ici le 8 mai, afin d'éprouver notre robot à l'aide de ce test grandeur nature.

6
CONCLUSION

La Coupe de France approche à grands pas (moins de deux semaines) et les échéances du groupe semblent avoir été tenues. Effectivement le robot n'est à l'heure actuelle pas tout à fait opérationnel, mais nous sommes confiants quant à notre capacité à finir ces derniers réglages, et nous sommes déjà très satisfaits de tout le travail accompli jusqu'ici.

Nous avons été confronté à de la gestion de projet. D'abord, au-delà du travail technique détaillé dans ce rapport, nous avons dû contacter des entreprises, chercher des sponsors, effectuer des demandes de subventions, commander du matériel, se placer en relation avec l'équipe de l'ENSTA... Autant de composantes qui se retrouvent dans n'importe quel projet professionnel. Ensuite, il est important de souligner que tout ne s'est pas passé comme prévu initialement, et qu'il a fallu changer ponctuellement de direction tout en gardant notre cap global : changements dans les règles entre la version beta et la version finale ; abandon des cartes MD25 ; mise en place difficile des capteurs ; modifications structurelles apportées au robot... Au niveau du management de l'équipe, la répartition des différentes tâches de la manière la plus efficace possible en fonction des affinités de chacun semble avoir fonctionné. D'autre part, nous avons réussi à rester impliqués durant les 9 mois en se re-motivant les uns les autres lorsque le travail avançait moins vite que prévu.

Tout au long du projet, nous avons été confrontés à des difficultés dues à notre inexpérience du domaine de la robotique. Il est important de noter que, dans la plupart des autres écoles, les élèves participent deux années de suite à la coupe : les premières années découvrent la robotique, tandis que les deuxièmes années dirigent le projet en utilisant l'expérience acquise lors de la première année. A l'X, l'organisation de la première et de la troisième année fait qu'il n'est possible de participer qu'une seule fois à la coupe. Par conséquent, la transmission de l'expérience d'une année à l'autre n'est pas optimale. Certes, les X2015 nous ont donné quelques conseils, mais ils n'ont pas toujours pu nous donner les informations dont nous avions besoin. Par exemple, nous avons perdu beaucoup de temps à comprendre comment faire fonctionner les capteurs, alors que les X2015 avaient utilisé les mêmes l'année dernière. Une idée pour améliorer la transmission de connaissances d'une promotion à la suivante serait de créer un Wiki, commun à tous les PSC de robotique, qui expliquerait comment utiliser les composants dont nous disposons. Le binet X-Robot est actuellement en train d'organiser ce wiki.

A la lumière de l'état actuel des travaux, nous seront effectivement en mesure d'homologuer le robot le 9 mai et de participer à la Coupe. Il ne reste plus qu'à mettre toutes les chances de notre côté pour réaliser une performance à la hauteur de notre investissement, et peut-être améliorer la 81ème place des X2015.

7
BIBLIOGRAPHIE

1. Règlement de l'édition 2018 de la coupe - http ://www.coupederobotique.fr/wpcontent/uploads/C2018_Rules_final_FR.pdf
2. Documentation des capteurs infrarouge ST microelectronics - http ://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html
3. Documentation des cartes de contrôle ST microelectronics - http ://www.st.com/en/ecosystems/stm32-nucleo.html ?querycriteria=productId=SC2003
4. Caractéristiques Arduino Mega 2560 - http ://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php ?n=Main.MaterielMega2560
5. Documentation C++ - http ://www.cplusplus.com/
6. Documentation protocole UART - http ://tpil.projet.free.fr/TP_Arduino/01_UART.html
7. Guide d'utilisation de Fusion 360 (logiciel de conception assistée par ordinateur) - https ://www.autodesk.com/products/fusion-360/get-started
8. Forum stackoverflow - https ://stackoverflow.com/
9. Le grand livre d'Arduino d'Erik Bartmann
10. Mécanique du solide d'Yves Granjon
11. Atelier de robotique de Pierre Gaucher, d'Arnaud Puret, de Nicolas Monmarché et de Mohamed Slimane
12. Moteurs électriques pour la robotique de Pierre Mayé
13. Électronique de puissance : structures, fonctions de base, principales applications de Guy Séguier, Francis Labrique et Philippe Delarue

8 ANNEXES

Robot		
Fonction	Critère	Niveau
Dimensionnement	Déployé	contour max de 1500mm
	Non déployé	contour max de 1200mm
	Hauteur max	350 mm
Alimentation	Pouvoir jouer 3 matchs de suite	
	Sécurité	Sac ignifuge
	Tension maximale	48 V
Visibilité	Espace libre sur face latérale	100 x 70 mm
Dispositif de démarrage	Cordon à tirer	500mm minimum
Bouton d'arrêt d'urgence	Doit arrêter le robot	20 mm de diamètre Rouge Facilement accessible
Arrêt automatique	S'arrête à la fin du match	100 s de match
Système d'évitement	Empêcher les collisions entre robots	Nécessaire pour l'homologation
Support de balise	Peut acceuillir une balise de l'équipe adverse	Cylindrique 80 à 100mm de diamètre Hauteur de 430mm Velcro face crochets

Figure 39 – Cahier des charges du robot

Panneau Domotique		
Fonction	Critère	Niveau
Alimentation	Interrupteur commute l'alimentation : un des fils relié à l'interrupteur	Câbles 1,5 m de type « banane » de 4 mm mâle
	Batterie interne	
Fixation	Velcro côté velour à l'arrière	
	En appui sur la bordure	
Dimensionnement	Ne pas dépsser le périmètre de l'aire de jeu (sauf câbles)	Largeur : 400 – 1000mm Hauteur : 297 – 400mm Épaisseur max : 80 mm
Visibilité	Visible depuis le public	Éclairage

Figure 40 – Cahier des charges du panneau domotique

Abeille		
Fonction	Critère	Niveau
Dimensionnement	Dimensions masximales	Hauteur : 200 mm Largeur : 150 mm Longueur : 200 mm
Action mécanique	Aucun composant électrique Activée par le robot	
Sécurité	Protection lors du transport	

Figure 41 – Cahier des charges de l'abeille