



Rapport final de PSC

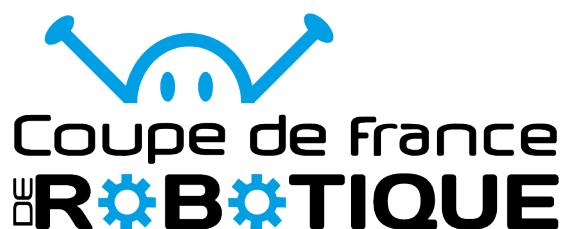
Coupe de France de Robotique

MEC11

Tuteur : Florent Souvestre
Coordinatrice : Laurence Bodelot

Julien Agier, Hamza Chaquiq Elbadre, Raphaël Colson
Maxime Escande, Yahya Iben Brahim

24 Avril 2019



Sommaire

| | |
|---|-----------|
| 1 Executive summary | 3 |
| I Introduction | 4 |
| II Notre projet | 5 |
| 2 Etude du règlement de l'édition 2019 | 5 |
| 3 Stratégie et objectifs | 5 |
| 4 Cadre du projet | 7 |
| 4.1 Organisation du projet | 7 |
| 4.2 Sponsors et Moyens | 8 |
| 4.3 Design Control | 8 |
| III Choix structurels et réalisation | 9 |
| 5 Plateau de jeu | 9 |
| 6 Structure générale du robot | 10 |
| 6.1 Moyen de locomotion | 11 |
| 6.2 Mesurer le déplacement indépendamment | 13 |
| 6.3 Organisation de la base roulante | 15 |
| 7 Alimentation et schéma électrique | 16 |
| 8 Motorisation | 19 |
| IV Automatisation | 23 |
| 9 Asservissement | 24 |
| 9.1 Enjeux | 24 |
| 9.2 État de l'art | 25 |
| 9.3 Solution retenue | 27 |
| 9.4 Difficultés rencontrées | 31 |

SOMMAIRE

Rapport final de PSC

| | |
|-------------------------------------|-----------|
| 10 Capteurs de distance | 33 |
| 10.1 Choix stratégique | 33 |
| 10.2 Mise en place | 35 |
| 11 Lien avec le groupe INF09 | 39 |
| V Conclusion | 40 |
| VI Annexes | 43 |
| 12 Etude du règlement | 43 |
| 12.1 Les atomes | 43 |
| 12.2 Le plateau de jeu | 43 |
| 12.3 Les actions | 43 |
| 13 Design Control | 47 |
| 14 Code Arduino | 49 |

1 Executive summary

Notre PSC consiste en la réalisation d'un robot autonome pour participer à la coupe de France de robotique. Ce PSC est repris d'année en année, et pour tenter d'améliorer le classement de l'Ecole polytechnique dans cette compétition, notre groupe a décidé de travailler sur une base mobile qui soit fiable, modulable et transmissible aux prochaines années. Notre PSC a la particularité d'avoir deux groupes qui travaillent en parallèle : l'un s'occupant de la partie mécanique et l'autre informatique. Notre travail en tant que groupe mécanique a consisté majoritairement à faire des choix techniques et à dimensionner les différentes pièces (moteurs, encodeurs, batteries ...) ainsi qu'à construire le plateau du jeu qui servira aux tests du robot.

Une fois notre stratégie définie et nos objectifs fixés, la première étape consistait à l'organisation du projet. En effet, la construction d'un robot nécessite de faire fonctionner ensemble de nombreux systèmes différents, nous nous sommes donc répartis en plusieurs pôles : Alimentation, Moteurs, Asservissement, Capteurs et Plateau de jeu. Pour un tel projet, la coordination des deux groupes ainsi que celle au sein d'un même groupe est importante, car il ne faut pas simplement que les différents éléments fonctionnent séparément, il faut qu'ils communiquent entre eux pour les faire fonctionner ensemble. La coordination des différents pôles était facilitée par le fait que nous disposions d'une salle de travail au sein du DrahiX, contenant notamment le plateau de jeu sur lequel doit évoluer le robot.

Un apport innovant de notre année est la séparation des roues motrices et des roues codeuses (servant à positionner le robot), ce qui a amené un nombre important de difficultés techniques nouvelles (notamment de l'hyperstatisme dû à 4 points de contact coplanaires). Nous avons donc été amenés à réviser notre planning initial pour rester cohérent avec notre objectif de transmission de notre travail. Nous envisageons la Coupe comme une performance sur la durée. Ainsi, nous avons privilégié le soin apporté à la base roulante, quitte à renoncer à la Coupe pour cette année, afin que les équipes ultérieures puissent s'appuyer sur une base fiable.

Première partie

Introduction

La coupe de France de robotique est une compétition de robotique amateur qui a lieu chaque année et dont la participation fait l'objet d'un PSC régulièrement depuis la X2014. Pour l'anecdote, une équipe de l'Ecole Polytechnique a même gagné en 1998.

La Coupe de France de robotique consiste en une série de matchs opposants deux robots autonomes (sans guidage humain) au cours desquels ils doivent accomplir un certain nombre d'actions, comme envoyer une balle ou empiler des cubes, et plus une action est ardue, plus elle rapporte de points. Si le format de la coupe consiste en des matchs, la philosophie du jeu et le règlement font qu'une stratégie pour gagner ne peut pas consister à gêner son adversaire. En effet, la plupart des éléments de jeu sont en double pour que les deux robots puissent réaliser les mêmes actions en parallèle et par ailleurs, le règlement stipule que pour qu'un robot puisse participer à la coupe, il doit être homologué, test qui vérifie entre autre que le robot s'arrête avant de rencontrer un autre robot. Pour résumer, une bonne stratégie consiste à avoir un robot capable d'obtenir un maximum de points tout en réduisant les chances de trop souvent croiser le robot adverse.

Une particularité de la coupe est que les actions que doit accomplir le robot changent chaque année pour forcer les équipes à réinventer leurs technologies d'une année sur l'autre. Cependant, on observe sur les dernières années que certaines règles, en particulier celles concernant le format des matchs et du robot varient peu. Par exemple, les dimensions de la table de jeu sont systématiquement de 3m par 2m, le périmètre maximal du robot est lui aussi toujours fixé à la même valeur.

En définitive, l'expérience montre que les équipes les plus performantes ne créent en général pas un robot *ex-nihilo* mais plutôt adaptent leur travail précédent. La performance à une édition particulière de la coupe une fois ce travail réalisé passe alors avant tout par le choix d'une stratégie efficace et un temps accru pour la réalisation des actionneurs.

C'est là la stratégie que nous souhaitons adopter dans ce projet en construisant un châssis motorisé adaptable selon les besoins que rencontreront les équipes ultérieures pour qu'elles puissent, petit à petit, en accumulant de l'expérience, monter dans les classements. C'est autour de cet intérêt que s'est formé notre équipe, deux groupes de PSC distincts travaillant sur un même projet. Notre groupe est en charge de la partie mécanique du robot, le groupe INF09 de la partie Software du robot.

Deuxième partie

Notre projet

2 Etude du règlement de l'édition 2019

Comme spécifié plus haut, le règlement de la coupe change considérablement d'une année sur l'autre. La première étape est donc naturellement d'étudier attentivement les spécificités de l'édition 2019 : "Atom Factory" (voir annexe Etude du règlement)

3 Stratégie et objectifs

L'étude du règlement nous permet d'établir comme stratégie le plan suivant :

- Démarrer (!) : 10 pts
- Activer l'accélérateur de particules et faire tomber le Goldenium : 30 pts
- Passer le reste du temps restant à classer des atomes dans les cases : nombre de points plus difficile à estimer, mais contrôlable avec la caméra, qui est l'objet du groupe INF09 avec lequel nous collaborons
- Estimer notre performance à l'avance à 40 points, ou bien plus précisément et de façon automatique par le biais de la caméra

$$TOTAL = 10 + 30 + 30\% \times 40 + \frac{1}{3} \times \text{Points marqués en classant des atomes}$$

Ce qui nous amène à un total de 52 points dans le pire des cas. Bien sûr, cette valeur ne donne que peu d'informations, car on ne peut pas la comparer aux autres années. Mais la stratégie présente les avantages suivants :

- Elle ne nécessite aucun actionneur complexe, et repose sur un positionnement fiable et précis
- Elle réduit au maximum la distance qu'aura à parcourir le robot (et donc le risque de dérive lors du calcul de sa position)
- Elle exploite trois des cinq façons de marquer des points

Comme nous l'avons vu en introduction, pour qu'un robot puisse être performant à la coupe de France de robotique, il est intéressant que son développement se fasse sur plusieurs années. L'exemple de nos prédecesseurs nous a montré qu'il était possible de construire un système capable de marquer des points à la coupe, mais que cependant, de

vouloir à tout prix réaliser des actions rapportant un grand nombre de points se faisait finalement au détriment de l'amélioration d'autres points du robot, comme par exemple son odométrie, c'est-à-dire la capacité du robot à mesurer sa position relative au fur et à mesure de son mouvement, ce qui peut poser des problèmes pour réaliser des actions nécessitant un positionnement précis.

Ces difficultés rencontrées nous ont encouragés à repartir une nouvelle fois de zéro pour la base roulante, avec l'idée de ne pas se concentrer sur des actionneurs complexes, comme en témoigne la stratégie retenue. Nous nous sommes ainsi fixés deux objectifs : premièrement, réaliser une base roulante adaptable, facile d'utilisation et dotée d'une odométrie précise que l'on puisse transmettre pour évolution aux années suivantes et deuxièmement d'utiliser et d'adapter notre base roulante au déplacement d'atomes pour pouvoir adopter la stratégie que nous avons choisie lors de la coupe de France de robotique. Pour être en mesure de remplir ces objectifs, nous avons choisi de jalonner l'année avec des échéances précise pour pouvoir être capables d'anticiper en cas de dérive, nous avons donc réalisé le planning suivant :

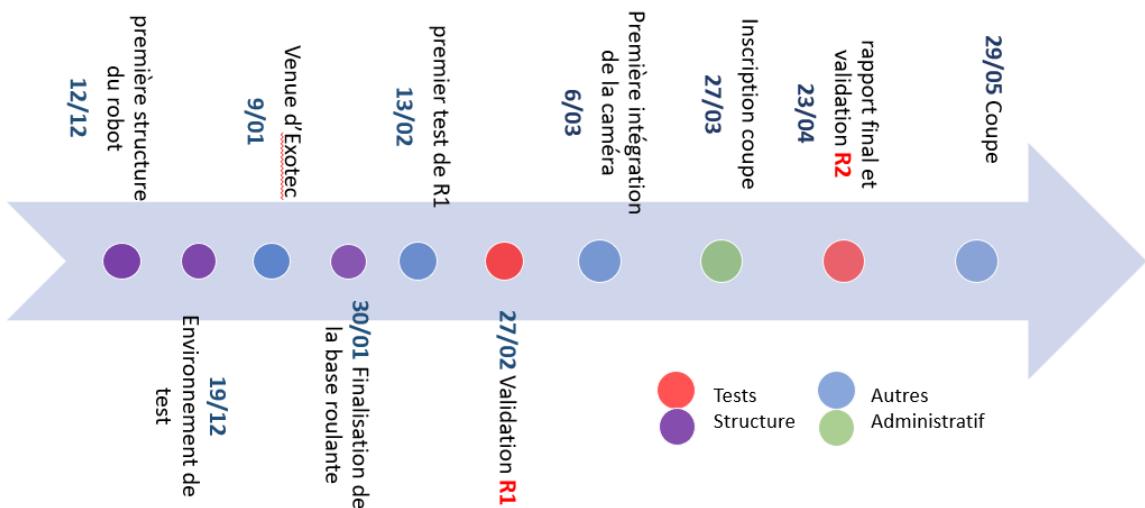


FIGURE 1 – Première timeline, définition de jalons temporels précis en accord avec les plans de Design Control détaillé en partie 3.4

Cependant, au fur et à mesure de l'année nous avons été forcés de constater que certaines tâches étaient plus laborieuses que prévues, nous aurons l'occasion dans préciser quelques unes dans la suite de ce document. Nous nous sommes ainsi retrouvés face au choix suivant : chercher à tout prix à raccrocher au planning pour participer malgré tout à la coupe, ou bien prendre le risque de ne pas avoir de machine satisfaisante à la coupe mais

de penser au mieux le design de la base roulante pour pouvoir transmettre un système satisfaisant et polyvalent. Dans un souci de cohérence avec nos premiers objectifs, nous avons fait le choix de la deuxième option ce qui nous a conduit à repenser notre planning pour parvenir à la version suivante :

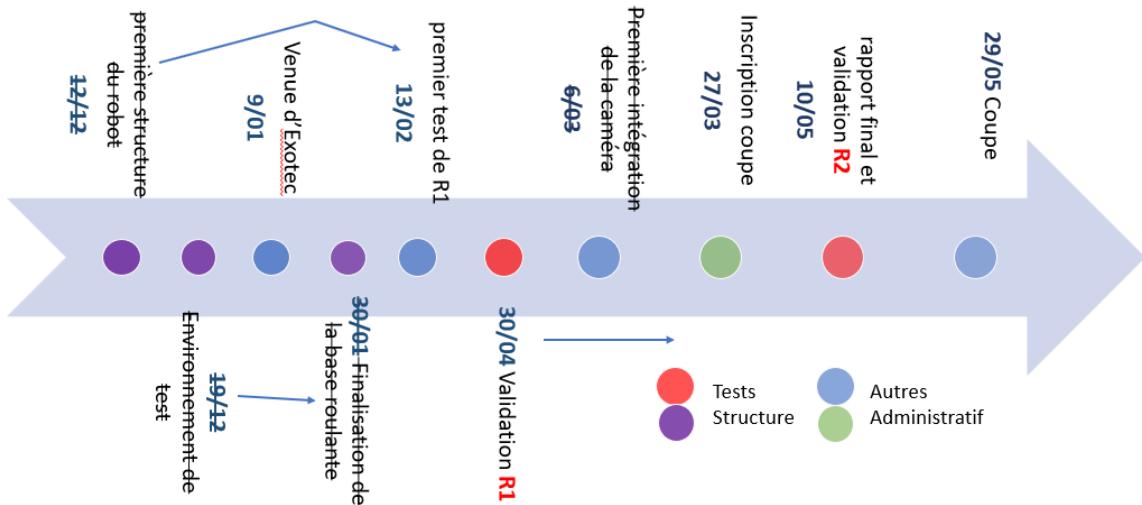


FIGURE 2 – Deuxième version de la Timeline après recentrage des objectifs fin Janvier

4 Cadre du projet

4.1 Organisation du projet

Nous sommes au total 9 personnes à travailler sur le projet. Ce nombre de personnes est relativement conséquent, nous nous sommes répartis clairement les tâches. Pour ce faire, nous avons scindé l'équipe en deux groupes de PSC distincts, le nôtre MEC11 dont la tâche est de réaliser la partie hardware du robot et le groupe INF09 en charge de la partie software. Nous avons aussi divisé les tâches au sein des groupes et choisi des chefs de groupes dont le rôle était de faire régulièrement (environ une fois par semaine) le point sur le travail réalisé dans chaque groupe, dans le but d'une mise en commun régulière permettant d'éviter de se disperser.

En effet, notre plus grande peur sur l'ensemble du projet était que chaque groupe se mette à travailler dans son coin et que tout bloque au moment de la mise en commun. Cette organisation nous a par exemple permis de nous rendre compte rapidement d'une situation problématique : la personne en charge de l'algorithme d'évitement d'obstacle n'avait pas bien noté les performances des capteurs du robot et était en train de fonder

la stratégie d'évitement du robot sur des données que le robot n'était pas en mesure de collecter. Nous avons pu nous rendre compte de cette erreur de communication avant que l'algorithme ne soit trop avancé ce qui à fait gagner des heures de modifications plus tard.

L'articulation de notre projet reprend l'idée d'organisation "podulaire" : nous avons déterminé différentes tâches et selon les compétences et les préférences des membres, nous nous sommes répartis toutes les fonctions. La partie "support" a été nécessaire pour assurer les tâches de gestion sans lesquelles le projet ne pourrait pas avancer.

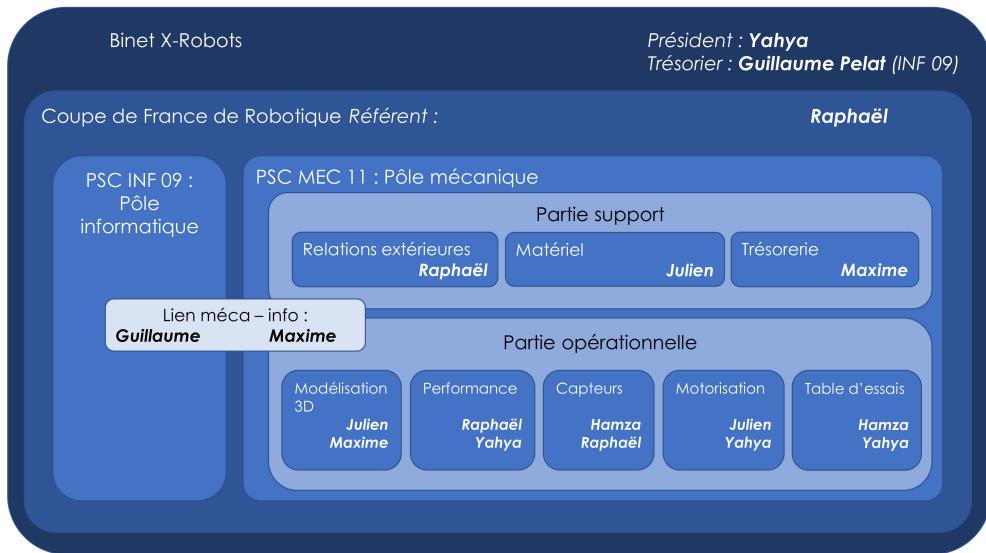


FIGURE 3 – Organisation podulaire du projet

4.2 Sponsors et Moyens

Pour disposer des fonds suffisants à l'achat de matériel performant, nous avons démarché plusieurs sponsors avant d'obtenir une réponse positive de la part d'Exotec Solutions. Il nous a ensuite fallu communiquer et négocier avec eux un contrat acceptable par les deux parties. Cela nous a permis de disposer d'un budget de 5.000€, et les représentants d'Exotec sont venus le 23 janvier à Polytechnique présenter leur entreprise et rencontrer les élèves, ce qui est ce que nous avons convenu de leur offrir en échange de leur aide technique et financière.

4.3 Design Control

Dans le cadre de notre partenariat avec Exotec Solutions nous avons été initiés à la méthode du Design Control. C'est l'application d'une méthodologie formelle pour le



FIGURE 4 – Présentation d’Exotec à Polytechnique le 23/01/2019

développement d’un produit. Le but étant de planifier toutes les étapes de la conception du robot et de les vérifier grâce à des tests précis et non ambigus définis à l’avance (voir annexe Design Control).

Troisième partie

Choix structurels et réalisation

5 Plateau de jeu

La table de jeu a un rôle essentiel et très important pour la préparation à la coupe. Construire une reproduction du plateau de jeu sur lequel va évoluer le robot le jour de la compétition nous permettra en effet d’effectuer les tests préparatoires à la Coupe dans les conditions les plus proches de la réalité.

L’aire de jeu est un plan rectangulaire horizontal de 3000 mm par 2000 mm avec des bordures sur chaque côté, et elle est constituée des plusieurs éléments cités dans la section “règlement de la coupe”.

Nous avons commencé à travailler sur la table de jeu au début du PSC, afin qu’elle soit prête en décembre pour commencer les tests. Il est primordial d’être le plus précis possible et de respecter au mieux les matériaux et dimensions donnés dans les règles annuelles de

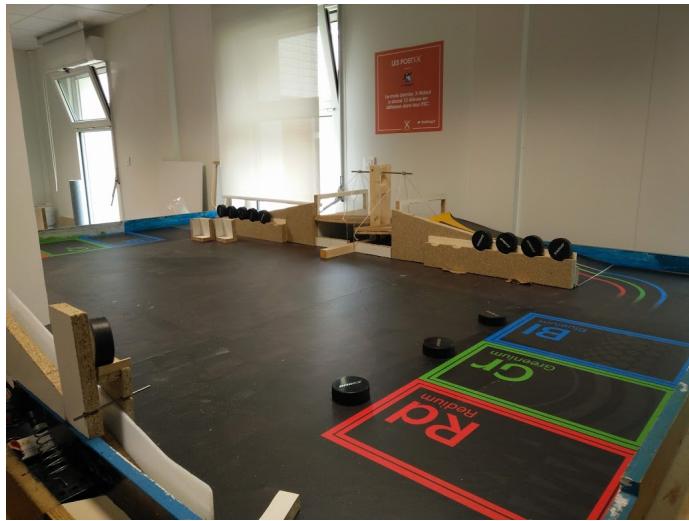


FIGURE 5 – Notre reproduction du plateau de jeu au DrahiX

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 pourrait ne pas marcher le jour de la coupe, car les interactions du robot avec son environnement ne seront pas les mêmes.

Cette nécessité de précision, ainsi que notre manque d’expérience en travail manuel et menuiserie a rendu le travail beaucoup plus dur qu’il ne paraissait, ce qui a retardé le résultat final de plusieurs mois.

Notre redéfinition des objectifs nous a conduit à simplifier certains détails de la table de jeu pour se concentrer beaucoup plus sur le robot.

6 Structure générale du robot

Commençons par définir les besoins. Pour pouvoir effectuer un déplacement quelconque sur l’aire de jeu, c’est-à-dire partir d’un point A avec une certaine orientation A et arriver en un point B et une orientation B, il suffit au robot d’être capable d’avancer, de reculer, de tourner et de mesurer ses mouvements de manière assez précise. Ces déplacements autorisent au système les mêmes capacités de mouvements qu’une voiture de tous les jours, ce qui implique que certains déplacements pourraient nécessiter des manœuvres parfois complexes dans l’éventualité où des obstacles seraient gênants. Les manœuvres, si elles ne sont pas difficiles à programmer sur le papier génèrent pourtant plusieurs problèmes comme une distance et un temps de déplacement supérieurs ou des chances plus grandes de voir sa trajectoire altérée par une rencontre avec le robot adverse ou un choc sur un



FIGURE 6 – Certains éléments du plateau

obstacle. Ainsi, elles sont à proscrire ou en tout cas à limiter au maximum, ce qui peut se faire si l'on ajoute comme capacité au robot d'être capable de tourner sur place, c'est-à-dire que le système soit capable de faire un tour sur lui même sans heurter d'obstacles fixes, et ce quelle que soit la disposition des obstacles.

Pour terminer, la table de jeu de la coupe de France de robotique a des dimensions de 2m par 3m, chaque match ayant une durée de 100s, la vitesse ou l'endurance du robot n'est pas le point limitant, en effet, les actions à effectuer ne nécessitent généralement pas de traverser la table de bout en bout le plus de fois possibles ainsi même une vitesse faible de 10 cm s^{-1} permet de faire la grande traversée en 30s environ ce qui laisse déjà un temps important pour effectuer des actions et marquer des points. Ainsi, l'enjeu n'est pas de mettre l'accent sur la vitesse mais sur l'efficacité de déplacement sous-entendu qu'un déplacement efficace ne heurte pas d'obstacle et amène précisément à la position souhaitée.

6.1 Moyen de locomotion

Il existe de nombreuses manières de faire se déplacer un robot. Certaines sont classiques, comme le recours à des roues simples type roues de voitures ou bien des chenilles comme on peut en trouver sur les chars d'assaut. D'autres sont plus originales comme les roues holonomes, permettant des trajectoires de nature très complexe avec en particulier

la translation dans n'importe quelle direction sans rotation du robot. [1] [2]



FIGURE 7 – Roue holonome

Les innovations portées par la soft robotique font appel à des matériaux innovants et souples dont la maîtrise n'est pas à notre disposition. Pour cette raison, nous avons décidé de rester sur des moyens de locomotions plus classiques. Les chenilles quant à elles ont un réel avantage sur des terrains avec de fortes irrégularités, ce qui n'est pas le cas du plateau de jeu de la coupe. On peut même imaginer qu'un robot capable de monter trop facilement sur des obstacles, qui dans notre édition pourraient être des atomes serait finalement désavantagé en augmentant son risque de basculer ou de rester coincé.

Ainsi, notre raisonnement nous amène à privilégier à ce stade des roues holonomes ou classiques. Les roues holonomes permettent, une fois bien maîtrisées, de réaliser des trajectoires d'une complexité et d'une fluidité notable. Il n'en demeure pas moins que certains inconvénients nous ont pourtant rebutés : elles sont relativement difficiles à faire fonctionner si l'on en croit les essais de différentes équipes de la coupe mettant généralement plus d'un an à les intégrer de manière efficace sur leur système. Le format de la coupe à l'Ecole polytechnique consistant en une expérience transmise sur un an au maximum, le choix de roues holonomes nous a paru ne pas aller dans le sens de nos objectifs. Par ailleurs, lors d'un déplacement avec des roues holonomes, la nature des trajectoires étant assez complexes, il n'est pas évident de parvenir à les analyser, ainsi l'asservissement des trajectoires dont nous parlerons dans la partie suivante se serait lui aussi vu complexifié.

En définitive, notre choix se porte sur des roues simples, il reste cependant à en choisir le nombre et la disposition. Des roues orientables permettent un changement de direction efficace et précis profitant du fait qu'une roue génère une résistance de type frottement d'autant plus grande qu'elle n'est pas dans l'axe. Cependant, comme nous l'avons fait remarquer plus haut, nous cherchons à limiter au maximum les manœuvres. D'autre part, d'un point de vue plus technique, équiper le robot de roues orientables est mécaniquement plus compliqué que des roues à axe fixe : il faut par exemple équiper chaque roue d'un servomoteur pour gérer la direction de l'axe ou encore réaliser un système de direction

géré par un servomoteur qui *tournerait le volant*. Le problème est donc reformulé de la manière suivante : est-ce que le recours unique à des roues à axe fixe peut permettre de réaliser tous les mouvements souhaités ?

Un rapide raisonnement permet de se rendre compte que si l'on souhaite éviter le glissement des roues, il n'est pas utile d'avoir plus de deux roues disposées selon le même axe. En effet, ajouter une roue sur le même axe n'aurait aucun intérêt moteur, et l'ajout d'une roue en dehors de cet axe s'opposerait soit à la rotation du robot soit à son déplacement en ligne droite selon l'orientation. De plus, pour que le robot puisse tourner sur place sans débattement, il faut que l'axe des deux roues motrices soit un *diamètre* de la structure du robot.

6.2 Mesurer le déplacement indépendamment

Pour mesurer son déplacement, le robot doit parvenir à mesurer son déplacement par rapport au sol. Cela peut se faire soit de manière directe en mesurant la rotation du moteur et en multipliant par le rayon de la roue motrice, soit de manière indirecte en disposant des roues codeuses en contact avec le sol dont on quantifie la rotation. Dans les deux cas, les roues motrices et les roues de mesure doivent être sur le même axe pour que tout déplacement puisse se faire sans glissement.

Pour bien comprendre tous les aspects qui rentrent en compte dans ce choix, il est à important de s'attarder à ce stade sur la notion d'hyperstatisme. En effet, si une seule des deux roues motrices est en contact avec le sol, le robot aura une trajectoire incontrôlable et si une roue codeuse n'est pas en contact c'est toute l'odométrie qui se voit erronée.



FIGURE 8 – Photo de la bride en PLA à déformation mécanique

Pour savoir quelle méthode de mesure, directe ou indirecte choisir, il est nécessaire d'avoir une idée d'où vient l'erreur et des précisions nécessaires sur l'encodage. En notant

α l'angle dont tourne une roue codeuse, R son rayon, l la longueur entre le centre du robot et la roue codeuse, θ l'angle de rotation du robot et L la distance parcourue sur un déplacement, on a :

Erreur finale due au déplacement de L :

$$\epsilon(L) = 2dL = 2Ru(\alpha)$$

Erreur finale due à la rotation de θ avant l'avancée de L :

$$\epsilon(\theta) = 2Lu(\theta) \approx \frac{R}{l}Lu(\alpha)\theta$$

L'approximation venant du fait que l'on néglige l'incertitude sur les longueurs, soit :

$$\theta = \frac{R}{l}\alpha, \text{ donc } \frac{u(\theta)}{\theta} = \sqrt{\left(\frac{\alpha}{R}\right)^2 u(R)^2 + \left(\frac{R\alpha}{l^2}\right)^2 u(l)^2 + \left(\frac{R}{l}\right)^2 u(\alpha)^2}$$

Le point important à remarquer est que si l'erreur sur un déplacement en ligne droite est indépendante de la distance parcourue, il n'en va pas de même pour l'erreur due à une rotation qui elle est proportionnelle à la distance parcourue ce qui en fait la génératrice principale d'erreur au cours d'un match.

Pour un déplacement classique de coupe de France choisi arbitrairement à une distance de 4m parcourue avec 5 demi-tours et en souhaitant une erreur de moins de 5 cm à l'arrivée, on obtient (avec $R=5\text{cm}$ et $l = 15\text{cm}$ des valeurs choisies pour être représentatives d'un robot de la coupe) :

$$u(\alpha) \approx \frac{5}{5} \frac{15}{3 \cdot 200 \cdot \pi} = 2\pi \frac{1}{800}$$

Il faut donc que l'encodage soit précis au 1/800ème de tour.

Par ailleurs, un conflit apparaît entre l'encodage et la motricité : pour que le robot puisse se déplacer efficacement, il faut qu'il ait une bonne surface de contact avec le sol au niveau des roues motrices, donc des roues de grand rayon avec des pneus larges. A l'inverse, pour être précis dans l'encodage, il faut tendre vers un contact ponctuel permettant de minimiser l'erreur sur la longueur l , donc des roues très fines et de faible rayon. La précision nécessaire sur l'encodage étant par ailleurs disponible sur une gamme très peu étendue de moteurs, nous avons décidé de faire le choix d'un encodage indirect en disposant deux roues codeuses sur le même axe que les moteurs (voir fig. 9) permettant ainsi d'optimiser indépendamment la motricité et l'encodage.

Cependant, un tel choix conduirait notre base roulante à avoir au minimum 4 points de contacts nécessaires en permanence sur le sol et donc à être à priori hyperstatique. Pour pouvoir remédier à ce problème, il nous a fallu trouver une solution pour maintenir les quatre points de contact. Nous avons donc opté pour un système d'amortisseur à équiper sur les roues codeuses. nous avons tout d'abord essayé de réaliser un amortisseur très simple, une bride imprimée en PLA grâce à de l'impression 3D et censée se déformer mécaniquement, cependant la rigidité du PLA était trop importante et ne permettait pas de lever l'hyperstatisme de manière satisfaisante. Ainsi, dans un deuxième temps nous avons décidé de recourir à un système de liaison pivot avec un bras tenant le codeur et la roue rappelé vers le sol par un élastique.

Pour obtenir des roues codeuses correspondant au mieux à nos exigences, nous les avons réalisées nous-mêmes par impression 3D et pris un joint torique pour la bande de roulement. Après avoir étudié plusieurs matériaux de joints toriques, il est apparu que le matériau grand commerce le plus adapté à nos besoins était le NBR70. Plus déformable que les NBR80 et 90, il est le plus rigide permettant un bon contact de roulement en maintenant une déformation faible, soit un contact proche du ponctuel.

6.3 Organisation de la base roulante

Comme expliqué plus haut, nous souhaitons que le robot soit capable de tourner sur place sans heurter d'obstacles dans le mouvement. La seule forme qui peut garantir cela est un disque puisque c'est la seule forme dont l'espace occupé au sol ne varie pas au cours de la rotation. Cependant, un disque présente plusieurs défauts. Tout d'abord, le règlement de la coupe impose une contrainte sur le périmètre du robot. Or, la majorité des cartes, moteurs et autres composants à intégrer dans le robot étant proches de parallélépipède, essayer de les agencer sur une surface circulaire aurait pour effet de perdre de l'espace utile.

De plus, l'environnement dans lequel évolue le robot est une aire de jeu rectangulaire, donc pour pouvoir se recaler sur un mur ou interagir avec les objets du plateau il peut-être utile d'avoir des côtés droits sur la structure. Un compromis nous a donc paru être un octogone, proche du cercle qui favorise tout de même l'intégration de composants et présente en particulier l'avantage d'avoir des côtés droits parallèles sur certains diamètres.

Avec tous ses points de contacts sur un seul axe, le robot ne serait pas stable, il est nécessaire de lui ajouter des points d'appuis supplémentaires n'entravant pas son mouvement et n'aggravant pas l'hyperstatisme. La solution la plus simple que nous avons trouvée pour cela est justement inspiré de l'exemple de la chaise à 4 pieds. Imaginons

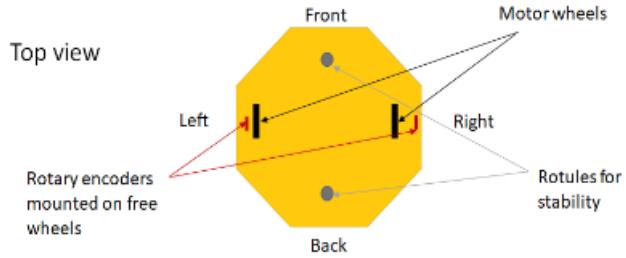


FIGURE 9 – Schéma de la base roulante

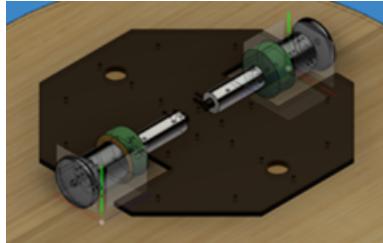


FIGURE 10 – Design 3D de la base sans les brides

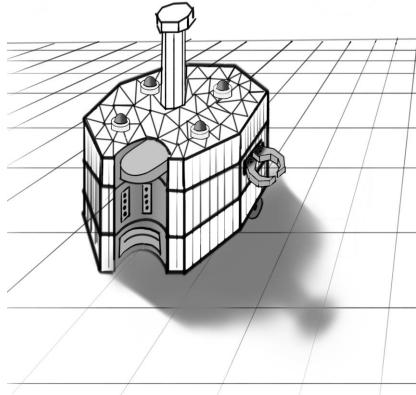


FIGURE 11 – Architecture étagée du robot

que deux pieds en face l'un de l'autre soient légèrement plus courts que les deux autres. Le contact serait alors sur 3 points au plus les trois points comprenant toujours les deux pieds les plus longs. Il suffisait donc de disposer sur deux rotules, une à l'avant et une à l'arrière du robot et légèrement plus hautes que les roues motrices. Ainsi lorsque le robot avance il s'appuiera sur ses roues motrices et la rotule avant et réciproquement avec la rotule arrière lorsqu'il recule.

Pour que l'architecture soit modulaire en accord avec nos objectifs exprimés plus haut, nous avons opté pour une architecture par étage. L'idée est que tout ce qui est nécessaire à la base roulante soit regroupé dans l'étage du bas apportant une certaine masse permettant de descendre le sens de gravité du robot. Les actionneurs spécifiques permettant d'accomplir des tâches particulières seraient alors placés dans les étages supérieurs.

7 Alimentation et schéma électrique

Enjeu

Dans la réalisation d'un robot, le choix et le dimensionnement de la batterie ne doit pas être pris à la légère pour avoir une base mobile efficace et fiable. De plus, il faut s'assurer que le montage électrique peut répondre aux besoins des circuits de commande et des actionneurs, tout en protégeant les éléments sensibles des pics de courants. L'objectif du dimensionnement est de trouver le bon compromis entre énergie et poids, sans négliger la résistance à l'usure. Il faut aussi tenir compte des questions de sécurité relatives aux risques d'explosion et de surchauffe, par exemple les règles de la coupe spécifient que les batteries au lithiums doivent être dans des sacs ignifugés ce qui augmente leur encombrement.

Cahier de charge

Une fois que nous avions une idée relativement précise des composants que la batterie devrait alimenter, nous avons établi un cahier des charges en accord avec le règlement :

- Le robot doit être capable de fonctionner au moins 30 min en continu sans recharger la batterie.
- La tension de sortie de la batterie ne doit pas être inférieure à 12V afin de pouvoir alimenter les moteurs
- La batterie doit avoir un courant de décharge suffisant pour fournir la puissance nécessaire aux moteurs dans les phases de démarrage et de freinage sans qu'elle ne soit endommagée.
- La batterie ne doit pas surchauffer à cause des courants de fonctionnement des moteurs.

État de l'art

Quatre types de batteries sont disponibles sur le marché :

- Li-Ion : le meilleure ratio capacité/poids, mais une mauvaise résistance et un courant de décharge limité.
- Li-Po : bon ratio capacité/poids et un excellent courant de décharge.
- NiMh : un ratio raisonnable, une bonne résistance et un prix raisonnable.
- Pb : un mauvais ratio, mais très grande résistance à l'usure, facile à utiliser [3].

| Qualités et limites des batteries | | | |
|-----------------------------------|---------|--------------------|-----------------|
| Type de batterie | Wh/kg | Wh/dm ³ | Puissance |
| Li-ION | 100-200 | 200-400 | Modérée à bonne |
| LI-PO | 80-120 | 200-300 | Excellente |
| NiMH | 50-80 | 150-200 | Modérée à bonne |
| Pb | 30-40 | 70-100 | Moyenne |

Choix de la batterie

Le robot a trois sources principales de consommation :

- Deux moteurs Maxon, 14 W chacun : $P_m = 14W \cdot 2 = 28W$
- La carte Arduino : $P_a = U \cdot I = 5V \cdot 1A = 5W$
- La carte Raspberry : $P_r = U \cdot I = 5W$

Les autres actionneurs et capteurs sont directement alimentés par la carte Arduino, ou ne fonctionnent pas en même temps que les moteurs (nous pensons ici aux futurs actions que le robot pourrait être conduit à faire dans les prochaines années). On a donc besoin d'une puissance totale

$$P_t = P_m + P_a + P_r = 38W$$

On peut en déduire l'énergie nécessaire pour un fonctionnement d'environ 30min sans recharge :

$$E = P_t \cdot \Delta(T) = 38W \cdot 0.5h = 19Wh$$

En choisissant une batterie de 12V (Les composants ont au maximum une tension nominale de 12V), on aura besoin au minimum de la capacité suivante :

$$\text{Capacité}(Ah) = \frac{\text{Energie}(Wh)}{\text{Tension}(V)} = \frac{19Wh}{12V} \approx 1.6Ah = 1600mAh$$

Nous avons donc d'abord opté pour une batterie Li-Po. Mais les contraintes logistiques (risque d'explosion et utilisation obligatoire d'un sac de sécurité pour la coupe) qui les accompagne nous a conduit à privilégier une batterie NiMh.

Design du schéma électrique

Une autre question s'est posée : veut-on avoir deux batteries distinctes, une pour la motorisation et une pour l'électronique ? Cela nous a paru une bonne idée pour deux raisons.

D'abord, les appareils électroniques (comme la carte Arduino) ont très peu de variations dans leur demande de courant contrairement aux moteurs : une batterie Li-ion serait alors adaptée car elle délivre un courant stable. De plus, les courants de retour des moteurs peuvent causer le redémarrage du circuit de commande en plein milieu du fonctionnement du robot en cas de demande excessive, les données des cartes électroniques sont alors perdues.

Cependant, cette solution n'est pas parfaite non plus, notre tuteur nous a par exemple expliqué avoir connu bon nombre d'équipes ayant perdu beaucoup de temps à réparer un robot dont le seul problème était qu'une des deux batteries était déchargée. Ainsi, un compromis qui a en plus l'avantage d'être moins gourmand en volume de robot est d'utiliser des convertisseurs de tension qui stabilisent les courants du circuit de commande (voir fig. 12)

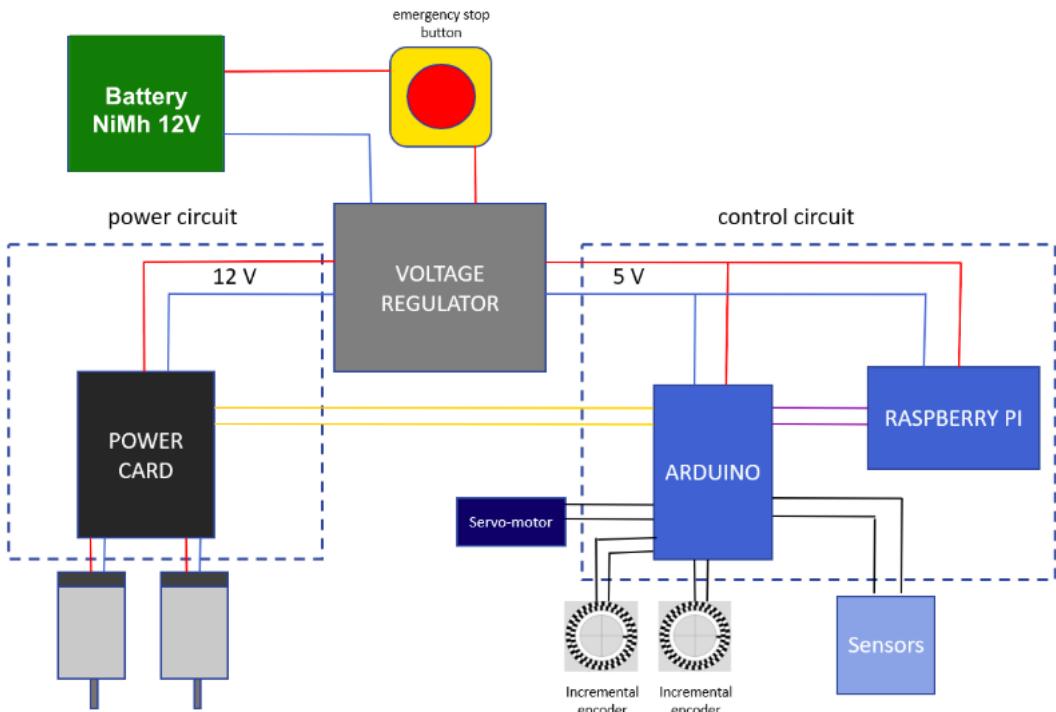


FIGURE 12 – Schéma électrique

8 Motorisation

Le bon choix des moteurs est un point important dans la réalisation d'un robot. En effet si les moteurs n'ont pas assez de couple, il sera très difficile pour le robot de se mouvoir à faible vitesse, si la vitesse de rotation est trop faible, le robot sera lent, si le

moteur est sous-dimensionné il sera utilisé en sur-régime, ce qui peut être bénéfique pour une utilisation de courte durée comme critique selon les cas.

Un dernier point d'importance qui a influencé le cadre de notre recherche de moteurs est que par l'intermédiaire de notre contrat de partenariat, nous avions accès à des fonds relativement élevés dont l'intérêt était justement de pouvoir investir dans des moteurs haut de gamme offrant de meilleures performances et des rendements plus élevés pour un encombrement moindre. Après quelques recherches, nous nous sommes rendus contre que la référence dans le domaine était la marque suisse *Maxon Motors* qui offre par ailleurs la possibilité de composer son assemblage de moto-réducteur parmi un catalogue très fourni.

Pour être sûrs que le robot soit capable d'appliquer des stratégies variables dépendant des règles des années suivantes, il est important que l'équipe souhaitant l'utiliser ne soit pas limitée par la vitesse du robot. Nous avons donc décidé que le robot devait être capable d'atteindre une vitesse de pointe de 0.5m s^{-1} lui permettant de traverser le plateau de jeu en comptant une phase d'accélération et de décélération en moins de 15s à 20s selon la trajectoire de vitesse choisie. Pour être plus sûrs de ne pas faire de bêtises, nous avons choisi de prendre une marge de 2, c'est-à-dire que les moteurs doivent être capables de faire avancer le robot à une vitesse d'au moins 1m s^{-1} . Soit avec des roues de 6cm de rayon :

$$\omega_{\text{arbre moteur}} \geq 136\text{rpm}$$

L'autre paramètre à évaluer est le couple nécessaire pour remplir les *requirements* que nous avons choisis dans les SRS. Plus précisément deux phases peuvent fixer le couple minimal nécessaire, une accélération au démarrage et atteignant la vitesse de 0.5m s^{-1} en moins de 1s ou bien au moment d'un freinage d'urgence à pleine vitesse sur moins de 20cm.

Avant de chercher à trouver lequel de ces deux cas est le plus contraignant, il est intéressant de chercher à voir si l'hypothèse de roulement sans glissement reste toujours valable dans ces deux phases. On considère donc le modèle suivant : En se plaçant dans le référentiel non galiléen du robot et en appliquant le PFD et le TMC en P1 et P2, on obtient que :

$$N_2 = \frac{mgl_1}{D} - \frac{mah}{D}$$

Le premier terme correspondant à un terme statique et le second à un transfert de charge. En écrivant ensuite la condition de non-glissement au niveau du contact roue2-sol et en

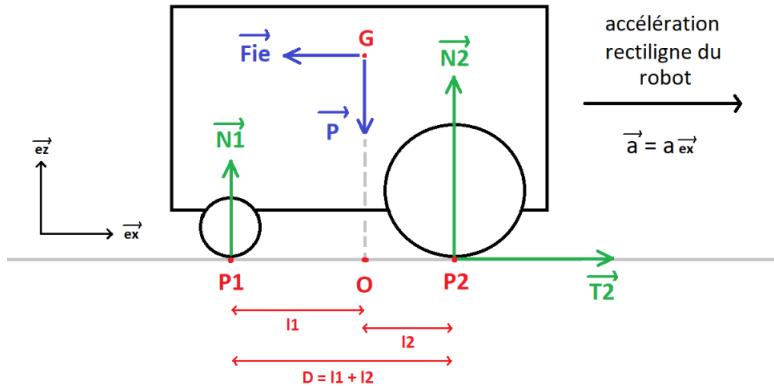


FIGURE 13 – Schéma du modèle d'étude de non glissement, R1 est une roue folle R2 est motrice

notant f le coefficient de frottement, il vient :

$$a_{\max} \leq \frac{fgl_1}{D + fh}$$

En prenant un coefficient de friction $f = 0.7$ ce qui correspond à un pneu sur une route et permet d'avoir un ordre d'idées, on a alors dans le cas où le centre de gravité est le plus excentré soit $l_1 = l_2 = 15\text{cm}$ $a_{\max} = 2.8\text{m s}^{-2}$ hors nos deux valeurs précédentes valaient respectivement $a_{\text{freinage}} = 1.25\text{m s}^{-2}$ et $a_{\text{accélération}} = 0.5\text{m s}^{-2}$, on observe donc que nous sommes *a priori* bien dans une phase d'adhérence sur ces différentes phases.

On peut maintenant évaluer le couple minimal que doit avoir le moteur en sortie de réducteur (à noter que le facteur $1/2$ vient du fait qu'il y a deux roues et que l'application numérique est faite avec une masse de 8kg et une roue de 6cm de rayon) :

$$C_{\min} = a_{\max} m \frac{R_{\text{roue}}}{2} = 0.35 \text{Nm}$$

Il ne reste donc maintenant qu'à choisir le moteur pour répondre à ces contraintes. Il existe deux technologies de moteurs électriques, les moteurs *brushed* à l'intérieur desquels le changement de polarité se fait dans l'enceinte du moteur par l'intermédiaire de petits patins qui viennent *brosser* les parois, et les moteurs *brushless* dans lesquels l'inversion de polarité est gérée électroniquement. Les moteurs *brushless* n'ont presque que des avantages par rapport à des moteurs *brushed* étant d'un meilleur rapport encombrement couple fourni et d'une plus grande résistance à l'usure. Cependant, le fait que le changement de polarité doive être effectué électroniquement fait que le moteur *brushless* est plus difficile

d'emploi[4].

Dans notre cas, les moteurs ne subiront pas une utilisation intensive et l'avantage des moteurs Maxon est qu'ils sont réputés pour leur robustesse, l'argument de la durée de vie n'est pas à prendre en compte.

La question qui demeure dans le choix de la technologie des moteurs est donc la nature des cartes de commandes que nous souhaitons utiliser. Comme nous avons déjà pu le voir dans la partie schéma électrique et comme nous aurons l'occasion de l'expliquer dans la partie suivante traitant de l'automatisation, notre robot, pour une simplicité d'utilisation accrue, est piloté par une carte Arduino et une carte Raspberry, deux micro-contrôleurs grand public et open source conçus pour pouvoir être rapidement mis en œuvre. Cependant, les possibilités qu'ils intègrent ne sont pas infinies et il nous a en particulier semblé plus difficile de gérer des driver pour moteur *brushless*. En effet, le driver idéal était une carte éditée par Maxon : la carte Epos 4[5]. Elle offre la possibilité de piloter un moteur *brushed* ou *brushless* en intégrant même directement la partie asservissement. Cependant, cette carte ne pouvait piloter qu'un moteur à la fois, ainsi, son utilisation aurait soit complexifié l'organisation décisionnelle du robot, soit consisté en une sous-utilisation d'outils professionnels. Pour cette raison, nous avons décidé d'utiliser la même carte de puissance que celle utilisée l'année passée qui offre la possibilité de piloter deux moteurs simultanément et communique simplement avec l'Arduino et les moteurs en PWM (pulse width modulation).

La technique de PWM consiste à moduler un signal avec un créneau irrégulier à haute fréquence. Selon la forme du créneau envoyé, le récepteur analogique *voit* un signal moyen sous la forme d'un pourcentage de la tension d'entrée.

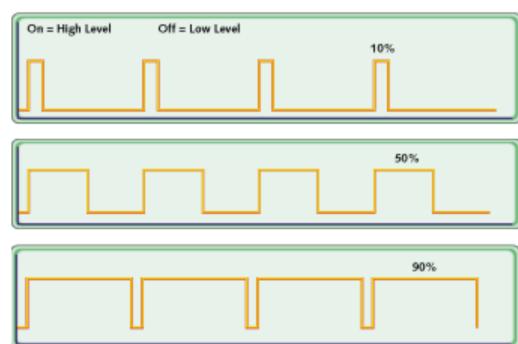


FIGURE 14 – Forme d'un signal PWM

Ainsi, les moteurs reçoivent en consignes un pourcentage de la tension de la batterie selon la consigne PWM. A noter toutefois que si cette méthode possède l'attractif de la simplicité, il n'en demeure pas moins qu'elle ne fonctionne pas pour des moteurs *brushless*[6].

Maintenant que le choix de la technologie des moteurs *brushed* est fait, il reste à trouver le duo moteur-réducteur approprié. On parle en effet de duo moteur réducteur car les moteurs électriques ont généralement une vitesse de rotation très élevée doublée

d'un couple très faible, il est donc le plus souvent utile de recourir à un réducteur, une série d'engrenages, qui va augmenter le couple d'autant plus qu'il réduit la vitesse de rotation. Comme nous l'avons vu dans la partie précédente, la batterie avec laquelle nous souhaitons équiper notre robot est une 12V, il est donc logique de chercher un moteur dont la tension nominale ne dépasse pas 12V.

Avec ces informations en tête et les valeurs calculées précédemment, nous sommes arrivés au choix d'un moteur de puissance nominale de 14W associé à un réducteur planétaire de rapport de réduction 44 :1 et d'efficacité énergétique de 71%. Ce choix s'est opéré en prenant le moteur 12V le plus puissant, et en l'équipant du réducteur le plus fort permettant de maintenir la vitesse de pointe souhaitée des 136 rpm de l'arbre moteur soit $44 \cdot 136 = 6000$ rpm pour l'axe du moteur. Les différentes valeurs calculées que nous serions amenés à demander au moteur sont récapitulés dans le tableau suivant et comparées aux valeurs nominales qui correspondent aux valeurs de fonctionnement normal.

| | | |
|-------------------------------|----------|-----|
| Vitesse de rotation du moteur | 6002,415 | rpm |
| Nominale | 10700 | rpm |
| Couple en amont du réducteur | 1,12E-02 | N.m |
| Nominal | 1,46E-02 | N.m |
| Puissance | 6,72E+01 | W |

FIGURE 15 – Confrontation des valeurs estimées aux valeurs nominales

Pour terminer l'étude du choix des moteurs, il est intéressant de vérifier que les points de fonctionnement auxquels nous comptons utiliser les moteurs sont adéquats, c'est-à-dire qu'ils n'entraîneront pas de surchauffe ou de dommage du matériel. Le graphique de fonctionnement du moteur est disponible sur le site de Maxon.

On observe que tous les points de fonctionnements sont dans des zones où le moteur opère sans risque de dommages. Seul le freinage d'urgence empiète dans un léger sur-régime mais ces phases étant normalement de courtes durées, le moteur ne devrait pas avoir tendance à surchauffer. On peut toutefois relever que les phases de freinage d'urgence, si elle ne demandent pas d'exploits de la part du moteur peuvent pour autant demander un courant important à la batterie, tolérés par la batterie d'après la partie précédente.

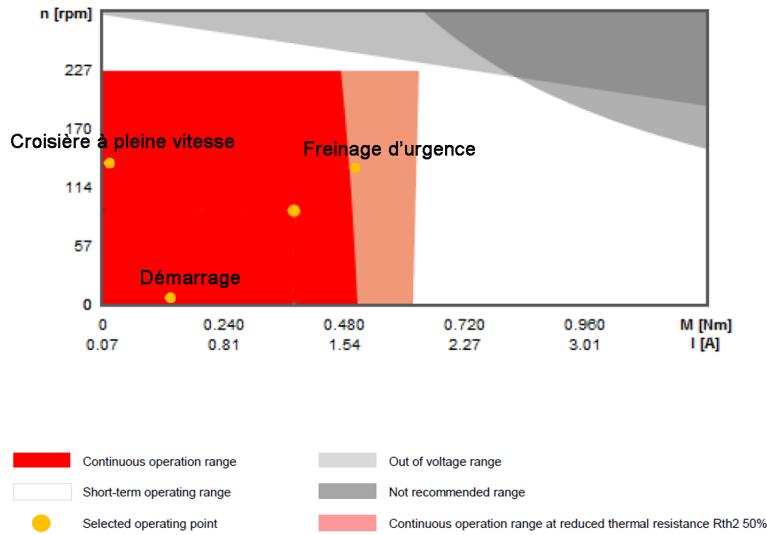


FIGURE 16 – Drive disposition du motoréducteur

Quatrième partie

Automatisation

9 Asservissement

9.1 Enjeux

Notre robot évoluant de façon autonome, il doit donc être capable de connaître sa position en permanence et avec une précision suffisante pour effectuer les actions (de l'ordre de quelques cm pour la plupart).

Dans l'asservissement, on distingue deux processus : la mesure de la position du robot, et la consigne en vitesse donnée aux moteurs. Le robot possède une sorte de "carte mentale" du plateau de jeu et de la position/orientation qu'il doit atteindre pour effectuer la prochaine action. Il calcule et corrige en permanence la consigne en vitesse qu'il doit fournir aux moteurs à partir de sa position actuelle, de celle qu'il veut atteindre, et éventuellement d'informations supplémentaires (obstacles sur la trajectoire) fournies par la caméra (voir fig. 17).

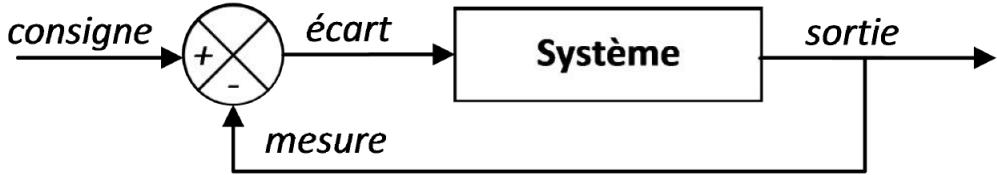


FIGURE 17 – Principe général de l'asservissement

9.2 État de l'art

Plusieurs technologies nous auraient permis de calculer la position du robot, nous les avons donc étudiées et comparées.

Tout d'abord, les lidar (contraction de "light detection and ranging") nous ont paru intéressants.

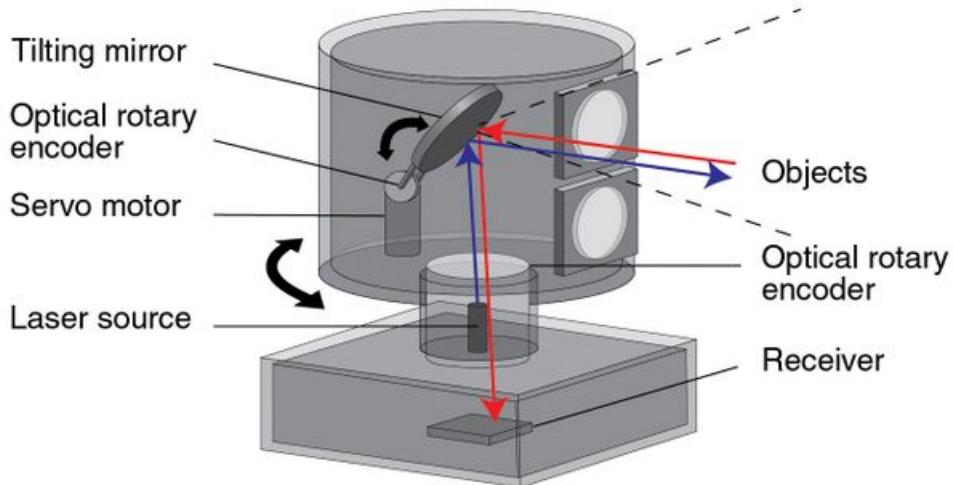


FIGURE 18 – Exemple de lidar

Un lidar est un télémètre laser qui tourne sur lui-même à 360° (voir fig. 18) et qui fournit au robot une sorte de carte mentale de tous les points qui sont le plus proches de lui dans une "tranche" d'espace d'altitude donnée. En se basant sur un certain nombre de points de repère de positions connues, le robot peut donc se situer (position + orientation) par triangulation. Le grand avantage de cette technique est qu'elle permet un positionnement absolu, et n'est donc pas sujette à des dérives avec le temps. Toutefois, nous y avons renoncé car elle nous paraissait trop complexe à mettre en oeuvre et surtout trop sujette

à des perturbations (notamment : les arbitres sont autorisés à se pencher par-dessus la table de jeu, et surtout les robots doivent être munis d'un mât qui est leur point le plus haut, et donc notre lidar risquait d'être perturbé par le mât du robot adverse).

Nous avons opté pour une technique plus simple et plus fiable : l'odométrie avec utilisation de roues codeuses. En plus de ses deux roues motrices, notre robot est muni de deux roues folles (sur le même axe) qui sont chargées de mesurer leur propre rotation. Elles délivrent une tension qui vaut soit 0V (0 binaire) soit 5V (1 binaire), et cette tension alterne 600 fois par tour. En comptant le nombre de fois où la tension varie, on peut donc facilement mesurer la rotation de chaque roue, et puis remonter au déplacement du robot. Les encodeurs que nous avons choisis sont dits à quadrature de phase, c'est-à-dire qu'ils ont deux pistes (la piste A et la piste B, cf. fig. 19) constituées chacune de 600 créneaux par tour, et décalées de $\frac{\pi}{2}$. Ceci permet, en comparant les deux pistes, de connaître le sens de rotation de la roue (et pas seulement sa rotation en valeur absolue). Par exemple, sur le schéma, si la piste A passe de 0 à 1 alors que la piste B est à l'état 0, alors la roue tourne dans le sens horaire. Enfin, nous avons calculé plus haut qu'une précision de $\frac{1}{800}$ de tour était nécessaire, mais un encodeur avec 600 créneaux par tour permet largement d'atteindre cette précision : en comptant les montées et les descentes des deux pistes en quadrature, on a en fait 2400 signaux par tour (résolution multipliée par 4).

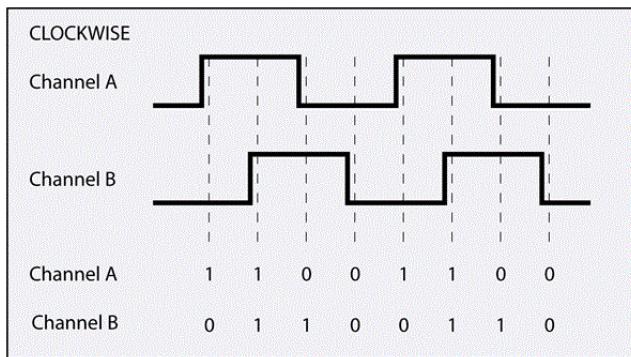


FIGURE 19 – Pistes A et B d'un encodeur rotatif à quadrature de phase

Le gros défaut de l'odométrie est qu'elle ne permet pas une mesure absolue de la position à l'instant t , mais qu'elle repose sur l'intégration de mesures tout au long du match, et est donc sujette à des dérives. Malgré tout, nous l'avons retenue en raison de sa précision, de sa simplicité de mise en oeuvre et surtout de sa réactivité : comme tous les calculs sont faits en interne, le robot peut connaître sa position en temps réel. Pour minimiser le risque de dérive, nous allons régulièrement nous recaler contre les bords du plateau (ce qui permet de ramener l'erreur sur l'angle du robot à 0, et nous avons vu que c'est cette erreur là qui est la plus néfaste), et nous pourrons aussi effectuer des

recalages grâce à la caméra, recalages moins précis (la précision de la caméra est de l'ordre de quelques cm) et qui prendront un petit peu de temps à cause des délais de communication entre le robot et la caméra, mais qui ont l'avantage de fournir une mesure absolue non sujette à la dérive.

9.3 Solution retenue

Nous avons donc opté pour deux encodeurs rotatifs 600 P/R de la marque MUJE, montés sur deux roues folles, le plus loin possible du centre du robot pour avoir une précision maximale en rotation.



FIGURE 20 – Encodeur rotatif utilisé

Calcul d'un déplacement élémentaire : Centre instantané de rotation

Pour pouvoir calculer le déplacement du robot à partir des informations fournies par les roues codeuses, nous avons utilisé le centre instantané de rotation (ou CIR). A chaque instant, le mouvement d'un solide peut être interprété comme une rotation autour d'un point virtuel, le CIR (voir fig. 21). Celui-ci peut se déplacer au cours du temps, tout comme la distance qui le sépare du solide (dans le cas limite où le solide se déplace en ligne droite, le CIR est à l'infini). Dans notre cas, comme les deux roues motrices du robot sont situées sur le même axe, le CIR se trouve nécessairement également sur cet axe, ce qui simplifie légèrement les calculs.

Pour réaliser les calculs, nous avons discrétisé le temps en intervalles de longueur $dt = 10\text{ms}$. A chaque intervalle de temps, le robot compte le nombre de "clicks" parcourus par chaque roue (un "click" correspondant à $\frac{1}{600}$ tour), et on suppose que le mouvement du robot pendant dt était très proche d'une rotation autour de son CIR.

On se place dans un référentiel lié au robot, la direction $\vec{e_x}$ pointant vers la droite du robot, et $\vec{e_y}$ vers l'avant.

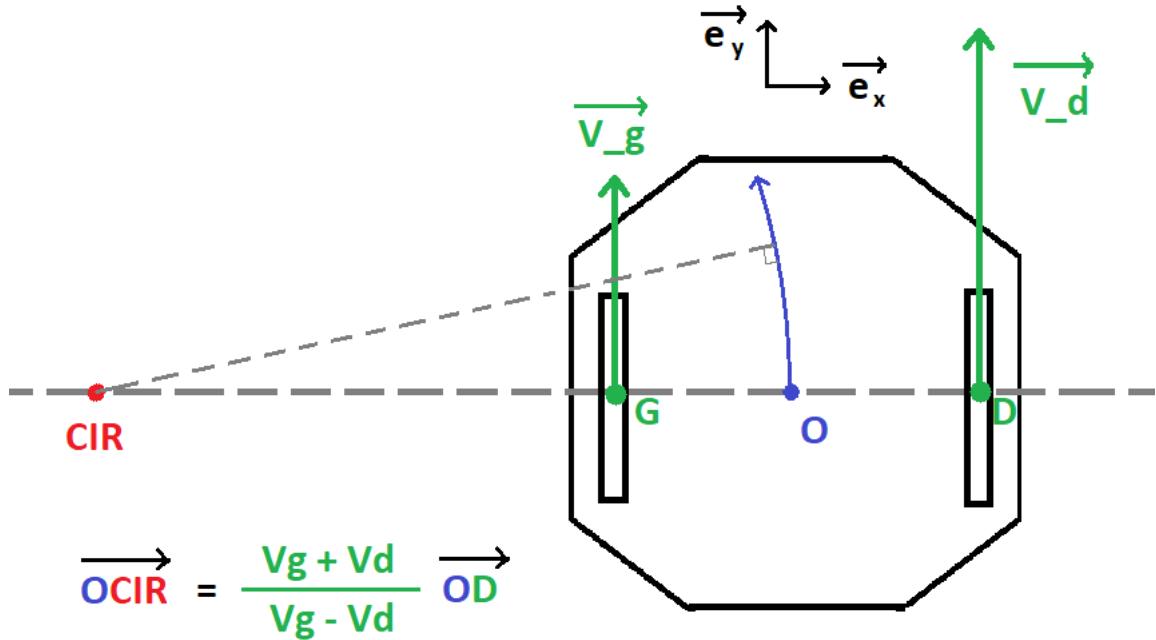


FIGURE 21 – Centre instantané de rotation du robot

Pendant l'intervalle de temps dt , la roue gauche (resp. droite) a parcouru une distance d_g (resp. d_d), ce qui nous permet de calculer sa vitesse $v_g = \frac{d_g}{dt}$ (resp. $v_d = \frac{d_d}{dt}$).

On calcule d'abord la position du CIR :

$$\overrightarrow{OCIR} = \frac{v_g + v_d}{v_g - v_d} \times \overrightarrow{OD}$$

Puis on calcule la vitesse du robot autour du CIR :

$$v_0 = \frac{v_g + v_d}{2}$$

On en déduit sa vitesse angulaire :

$$\omega = -\frac{v_0 \times dt}{||\overrightarrow{OCIR}||} = \frac{v_d \times dt - v_g \times dt}{2 \times ||\overrightarrow{OD}||}$$

Puis son déplacement et sa rotation élémentaire durant dt :

$$dx = ||\overrightarrow{OCIR}|| \times (1 - \cos(\omega \times dt))$$

$$dy = -||\overrightarrow{OCIR}|| \times \sin(\omega \times dt)$$

$$d\phi = \omega \times dt$$

Enfin on se ramène à la position du robot dans repère lié à la table $(\vec{X}, \vec{Y}, \Theta)$ qui est celui qui nous intéresse (Θ représente l'angle orienté entre l'axe \vec{X} et l'axe $\vec{e_y}$) :

$$dX = \cos(\Theta) \times dy + \sin(\Theta) \times dx$$

$$dY = -\cos(\Theta) \times dx + \sin(\Theta) \times dy$$

$$d\Theta = d\phi$$

Boucle de position

La première boucle d'asservissement de notre programme est une boucle de position : elle sert à calculer la consigne en vitesse à envoyer aux moteurs.

Dans un premier temps, le robot tourne sur lui-même pour ajuster le paramètre Θ , puis une fois qu'il est bien orienté il avance en ligne droite jusqu'à son objectif. Si jamais il n'y parvient pas du premier coup (notamment si l'erreur sur l'angle était trop importante au début et donc que la ligne droite l'a amené trop loin de son but), il recommence. L'intérêt de cette méthode est que la consigne en vitesse à envoyer aux moteurs est toujours égale en valeur absolue (si les roues tournent en sens inverse, le robot tourne sur lui-même sans débattement, si elles tournent dans le même sens, il avance en ligne droite).

La consigne en vitesse est calculée selon le principe suivant :

$$\text{Consigne} = K \times \sqrt{\text{Distance à l'objectif}}$$

Et si la consigne dépasse une certaine limite en valeur absolue (qui correspond à la limite définie dans les SRS), on la ramène à cette limite. La formule en racine carrée permet d'avoir une décélération constante à l'approche de l'objectif, et de lisser la trajectoire.

Boucle de vitesse : PID

A partir de la consigne en vitesse fournie par la boucle de position, nous avons choisi d'asservir les moteurs selon le principe éprouvé du Proportionnel - Intégrale - Dérivée (PID).

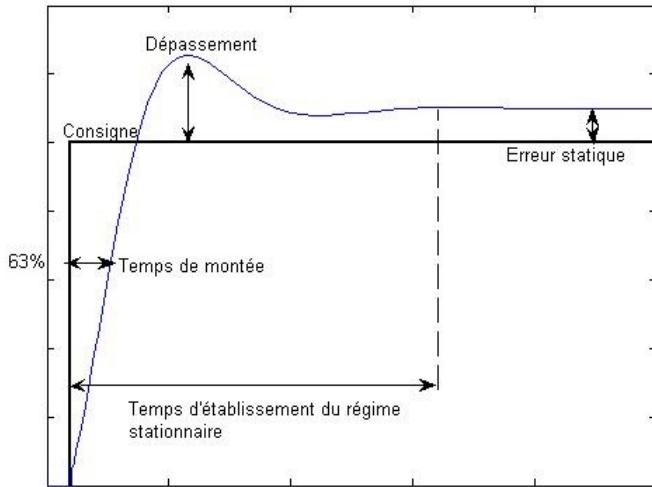


FIGURE 22 – Les différents biais d'un asservissement, que l'on cherche à contrôler

Le PID permet de trouver un compromis entre les différents biais d'un asservissement (voir fig. 22) en réglant ses trois paramètres. Si $\epsilon(t)$ représente l'erreur, alors la commande $u(t)$ est donnée par :

$$u(t) = K_p \times \epsilon(t) + K_i \times \int_0^t \epsilon(t) \times dt + K_d \times \frac{d\epsilon}{dt}(t)$$

- Le paramètre Proportionnel K_p permet de se rapprocher de la consigne avec une vitesse proportionnelle à l'erreur actuelle. Il permet donc de s'en rapprocher rapidement si l'erreur est très grande, mais il provoque souvent des dépassemens de consignes et des oscillations autour de celle-ci.
- Le paramètre Dérivée K_d permet de tempérer l'instabilité causée par le paramètre Proportionnel. Lorsque le système se rapproche de la consigne, le paramètre Proportionnel devient moins important, et si la vitesse (la Dérivée) de rapprochement est trop importante, c'est le paramètre Dérivée qui devient prépondérant et permet de "freiner" avant d'atteindre la consigne, et donc de limiter les oscillations.
- Enfin, le paramètre Intégrale K_i permet de corriger une erreur statique, car il devient prépondérant avec le temps si le système se stabilise à une valeur différente de la consigne.

Pour régler notre PID, nous nous sommes basés sur les règles de Ziegler-Nichols [7], puis nous avons ajusté les constantes K_p

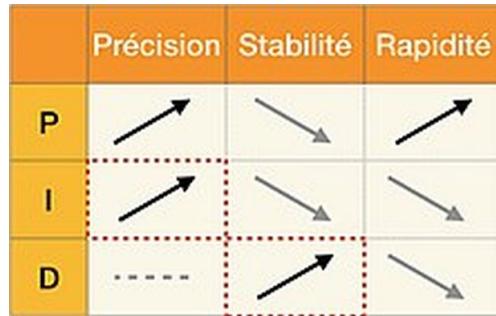


FIGURE 23 – Influence des trois paramètres P, I, D sur l'asservissement

9.4 Difficultés rencontrées

Résistance pull-up

Un encodeur rotatif possède deux canaux de sortie, qui oscillent entre la valeur 0 et la valeur 5V (interprétables comme des 0 ou 1 numériques par la carte Arduino). Toutefois, quand nous avons testé notre premier encodeur, nous n'avions pas du tout la sortie espérée : la tension en sortie était quasi constante et autour de 2V.

Après de nombreuses tentatives infructueuses, c'est finalement notre sponsor et notre tuteur qui nous ont mis sur la piste : en réalité, l'encodeur rotatif donne en sortie une tension de 0V pour coder 0, et une tension non définie (un fil « flottant » relié à rien) pour coder 1. Il faut donc rajouter entre la sortie et le +5V de l'alimentation une résistance dite « pull-up » (voir fig. 24).

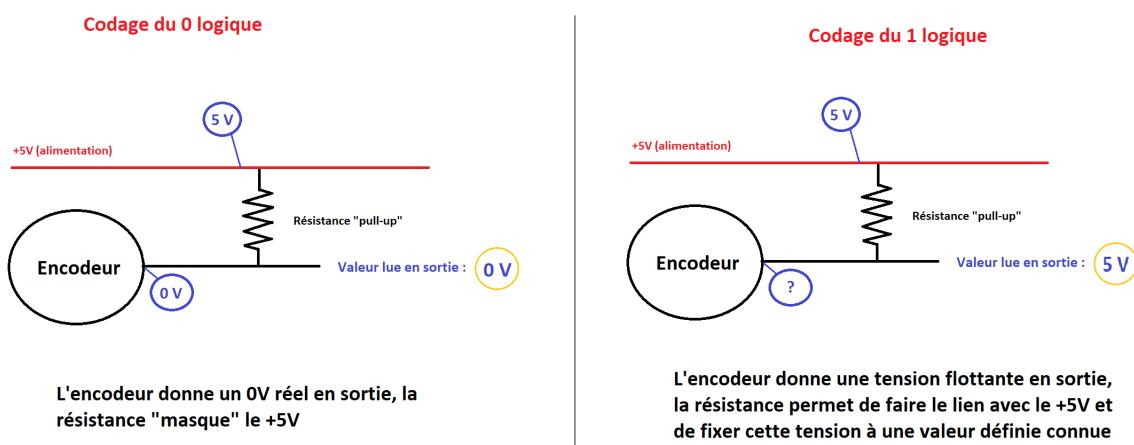


FIGURE 24 – Ajout d'une résistance pull-up à la sortie de l'encodeur

Après quelques tâtonnements, nous avons trouvé la valeur idéale de cette résistance (environ $150\ \Omega$), et l'encodeur nous donne maintenant des 0 et des 1 logiques parfaitement reconnaissables (fig. 25)).

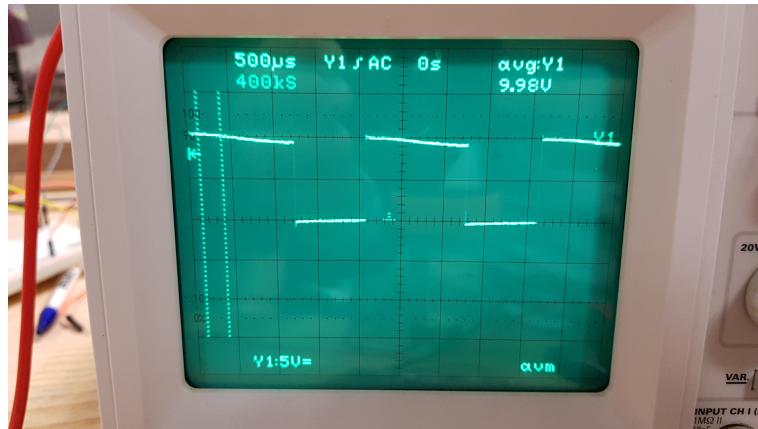


FIGURE 25 – Tension corrigée en sortie de l'encodeur

Surdimensionnement des encodeurs

Le cerveau de notre odométrie est une carte Arduino Mega. La difficulté liée à la lecture de l'encodeur est que le programme doit en permanence être "attentif" aux encodeurs : on ne sait jamais quand les roues vont tourner. Le programme doit donc guetter le changement de valeur d'un des deux canaux même lorsqu'il est en train d'effectuer un autre calcul. Heureusement, Arduino implémente une solution à ce problème : elles possède des pins capables de générer des "interruptions" : en permanence, le programme va guetter un changement de valeur de la tension mesurée par ces pins, et lorsqu'un changement se produit, le programme s'interrompt pour traiter une petite procédure (ici, incrémenter le compteur de tours de la roue droite ou gauche de +1 ou -1). Le code Arduino permettant de gérer les interruptions est disponible dans l'annexe Code Arduino.

Comme expliqué plus haut, les calculs effectués lors de la conception montraient que nous avions besoin de roues codeuses précises au moins au 1/600ème de tour. Notre budget le permettant, nous avons commencé par acheter des encodeurs précis au 1/5000ème car leur encombrement est exactement le même, et nous pensions ainsi améliorer encore la précision du placement du robot. Toutefois, un problème auquel nous n'avions pas pensé s'est alors posé : les roues du robot tournant à environ 10 tours/seconde, les deux encodeurs produisaient, à eux deux, 100.000 "clicks" par seconde, et cette fréquence est trop élevée pour la carte Arduino Mega, qui ratait des clicks et n'arrivait pas à suivre la cadence.

Nous avons d'abord essayé de faire malgré tout fonctionner ces encodeurs en remplaçant la carte Arduino par une carte Raspberry Pi, plus puissante, mais les résultats étaient encore pires (en fait la puissance de calcul de la Raspberry Pi est pensée pour faire fonctionner un OS ou d'autres procédures complexes, mais elle n'est pas optimisée

pour l'utilisation des interruptions).

Ce problème, que nous avons en définitive surmonté en changeant d'encodeurs, nous a appris les dangers du surdimensionnement, et l'importance de rester "humble" en quelque sorte : même si le budget nous permet d'acquérir un composant très performant, c'est en définitive la pièce la plus limitante qui impose sa performance au système.

10 Capteurs de distance

10.1 Choix stratégique

Les capteurs de position sont l'unique et essentiel outil qui permet une localisation directe et rapide du robot dans son entourage. En effet, l'utilisation de l'odométrie avec des encodeurs aussi précis que les nôtres permet de retracer le chemin parcouru par le robot, cela lui permet donc de se situer sur la table avec une certaine précision. Néanmoins, le robot doit être prêt à réagir devant n'importe quel imprévu. Celui-ci peut être les palets pour notre cas, et plus généralement le robot adverse ou les obstacles sur la table de jeu, d'où la nécessité des capteurs. Il est important de signaler que l'utilisation des ces capteurs reste en complète adéquation avec la stratégie de transmettre notre base mobile aux équipes suivantes.

L'information reçue par les capteurs est ensuite transmise à la carte Arduino qui gère le déplacement du robot.

Du fait de notre problématique, des capteurs simples monodirectionnels nous ont parus suffisants. En effet, avoir recours à un système type lidar permettant de cartographier en deux dimensions nous est apparu comme difficile à mettre en œuvre pour un gain d'une simple redondance d'information avec le système de caméra. Nous devions donc choisir entre deux grandes familles répandues de capteurs : Infrarouges et Ultrasons. Une étude était donc nécessaire pour pouvoir faire un choix adéquat en prenant en compte tous les paramètres entrants en jeu tels que les perturbations qui peuvent provenir de l'adversaire ou des installations liées à la coupe.

Le tableau ci-dessous résume cette étude :

| | Infrarouge | Ultrason |
|-----------------------------------|---|--|
| <i>Précision</i> | Précision meilleure | Précision plus faible |
| <i>Directivité du signal émis</i> | Signal plus ciblé : moins de perturbations à cause du sol ou d'autres obstacles | Signal moins directif : peut détecter des obstacles pas nécessairement devant le robot |
| <i>Interférences</i> | Peut être perturbé par un signal infrarouge de caméra ou du robot adverse (peu probable : signaux sont de direction précise). | Facilement perturbable par un autre émetteur ultra-son si les fréquences coïncident. |

L'équipe de la X2016 avait utilisé des capteurs infrarouges de ST Microelectronics, mais nous adopterons une autre technologie pour les raisons suivantes :

- La communication entre ST Microelectronics et l'électronique Arduino est compliquée.
- Une carte ST ne peut contrôler que 3 capteurs à la fois (système peu flexible).
- Les protocoles de communication à mettre en oeuvre fond que les données des capteurs ne peuvent être écrites que sur 4 bits (16 valeurs) ce qui est une perte de précision non négligeable.

En conséquence, nous avons choisi d'utiliser le capteur infrarouge SHARP GP2Y0A41SK0F ci-dessous, qui est capable de mesurer un obstacle de 4 à 30 cm. Ce capteur délivre une tension qui dépend de la distance mesurée selon le graphe caractéristique présenté fig. [8] :

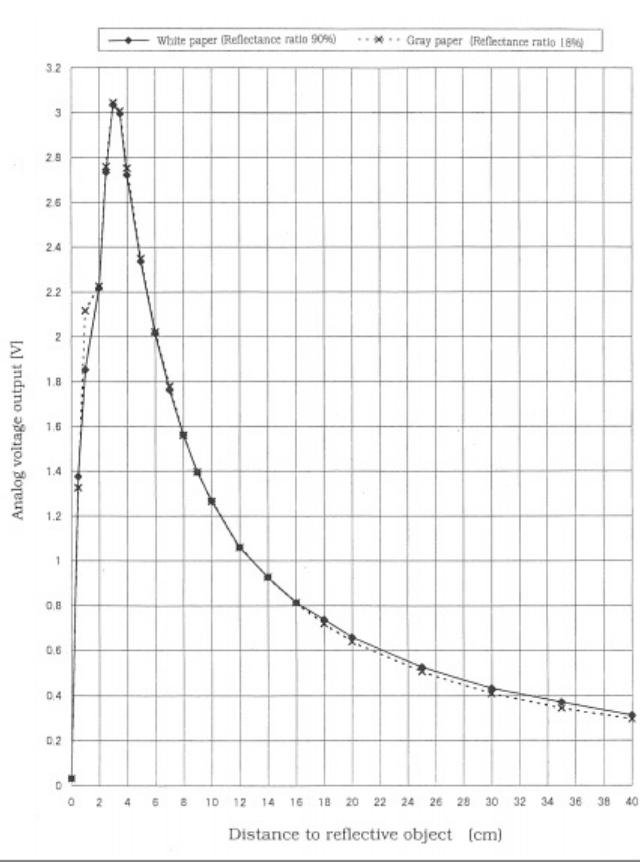


FIGURE 26 – Tension délivrée en fonction de la distance

Toutefois, nous remarquons que cette courbe n'est pas bijective : une tension de 1,8V peut correspondre à deux distances différentes : 1 cm ou 7 cm. Cela peut induire des erreurs de déplacement. Toutefois, la courbe devient bijective à partir d'une distance de 3 cm à peu près. Ainsi pour passer outre ce problème, nous avons décidé de placer le capteur 3cm à l'intérieur du robot, ce qui nous permet de retrouver une bijection entre tension délivrée et distance lue.

10.2 Mise en place

Comme décrit précédemment, les capteurs choisis peuvent détecter des objets jusqu'à 30cm, et nous avons décidé de les utiliser dans notre cas pour capter le robot adverse mais aussi pour détecter les palets.

En effet, la localisation absolue du robot est assurée par la caméra, ce qui permet de communiquer les positions des points à atteindre. Nous ne pouvons toujours pas nous prononcer sur le délai de la transmission de l'information, mais nous estimons que les

capteurs sont assez rapides et adéquats quand le robot est très proche d'un obstacle.

La caméra transmet au robot la position des palets, mais celle-ci n'est pas exacte, d'où le rôle des capteurs : fournir une position exacte de l'obstacle quand le robot s'approche de celui-ci. Pour ce faire, il a fallu calculer le nombre de capteurs à mettre et leurs dispositions sur le robot pour une précision donnée.

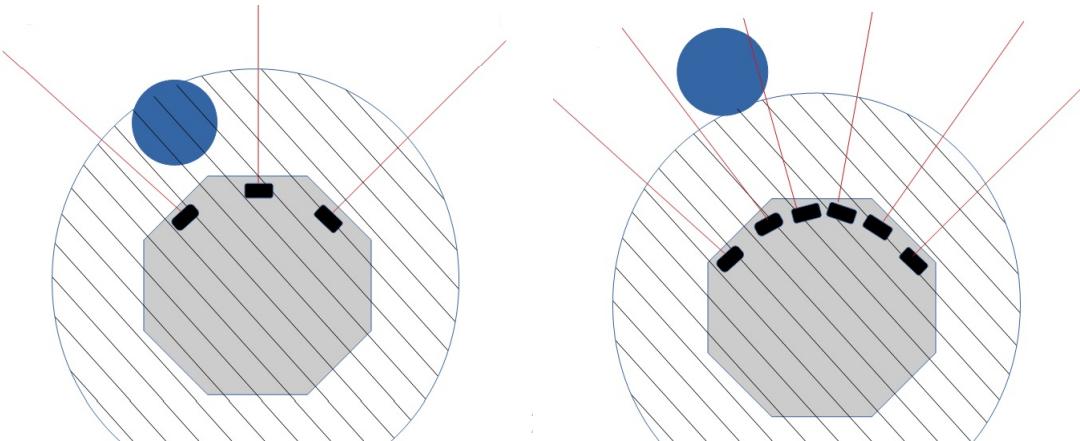


FIGURE 27 – Situation indésirable vs Situation désirable

La zone hachurée est déterminée de façon à ce qu'il n'y ait pas d'objet qui ne soit pas scanné par les capteurs.

Calculs du nombre de capteurs nécessaires On suppose que l'on veut détecter un objet une fois qu'il est dans un cercle de rayon R du centre du robot, on l'appelle dorénavant *précision* et on note aussi R' le rayon de l'obstacle (robot, palet, etc...). Notre problématique est donc de calculer l'espacement qu'il faut mettre en place entre les capteurs pour qu'ils puissent satisfaire l'exigence qu'on a posée : détecter les obstacles à moins de distance R du centre de robot. On note α l'angle entre deux capteurs successifs, et on s'aide du schéma ci-dessous pour résoudre l'équation déterminant α .

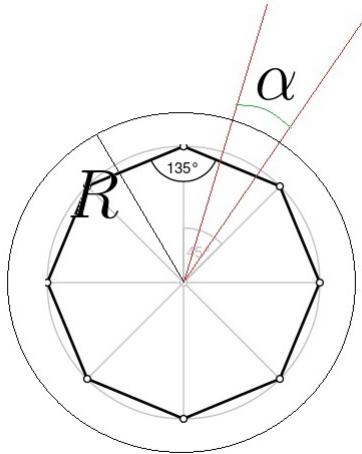


FIGURE 28 – Angle entre deux capteurs et cercle de précision R

Un obstacle de distance caractéristique R' est détecté avant d'entrer dans le cercle de rayon R si et seulement si :

$$\alpha R = R'$$

Soit :

$$\alpha = \frac{R'}{R}$$

Il reste à déterminer la précision souhaitée ainsi que l'extension angulaire dans laquelle on veut placer les capteurs pour déterminer le nombre nécessaire.

En se concertant avec le groupe INF09 qui travaille sur la partie Software du robot, il nous a paru qu'un calcul plus rigoureux doit être fait afin de déterminer le centre des palets que le robot détecte par ses capteurs, afin que l'algorithme de déplacement soit d'une meilleure précision. On a décidé de faire le chemin inverse : fixer un nombre de capteurs et déterminer la précision, et finalement voir si elle convient aux attentes. On a donc choisi d'installer 6 capteurs (en mettre plus induirait un problème d'encombrement), répartis sur les 135° de la face avant du robot (3 côtés de l'octogone).

Puisqu'il y a 5 espacements entre 6 capteurs, ils sont répartis donc avec un angle :

$$\alpha = \frac{135}{5} = 27^\circ$$

Le calcul du centre de l'atome se fait donc grâce aux mesures des deux capteurs qui pointent vers lui, selon la figure suivante :

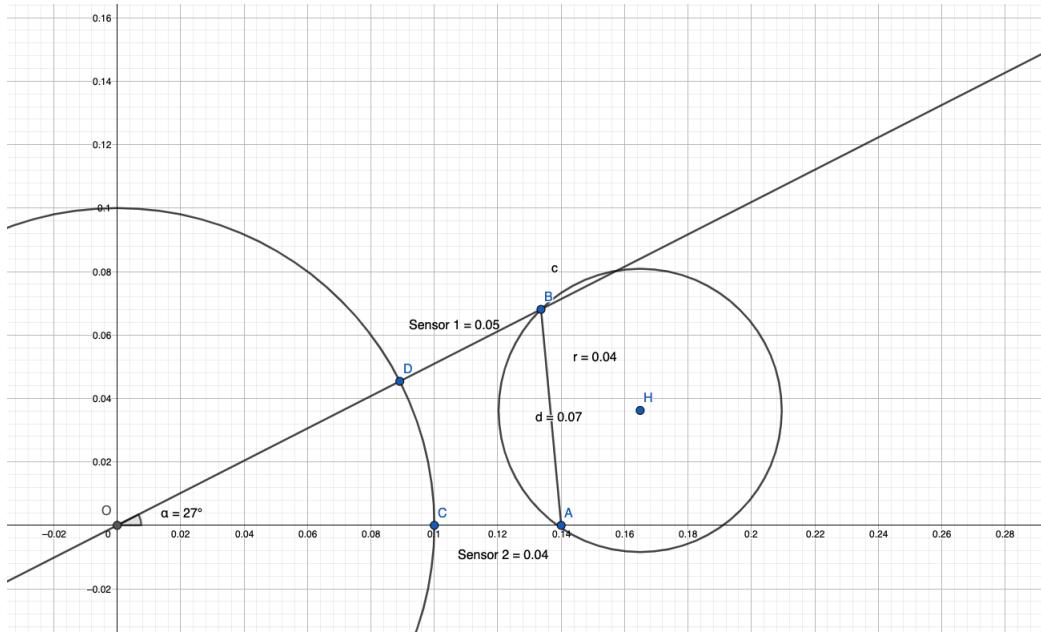


FIGURE 29 – Problème plus précis

On obtient une solution numérique donnée par la zone grise qui correspond à la zone captée par les deux capteurs en fonction des deux distances de l'obstacle de ceux-ci.

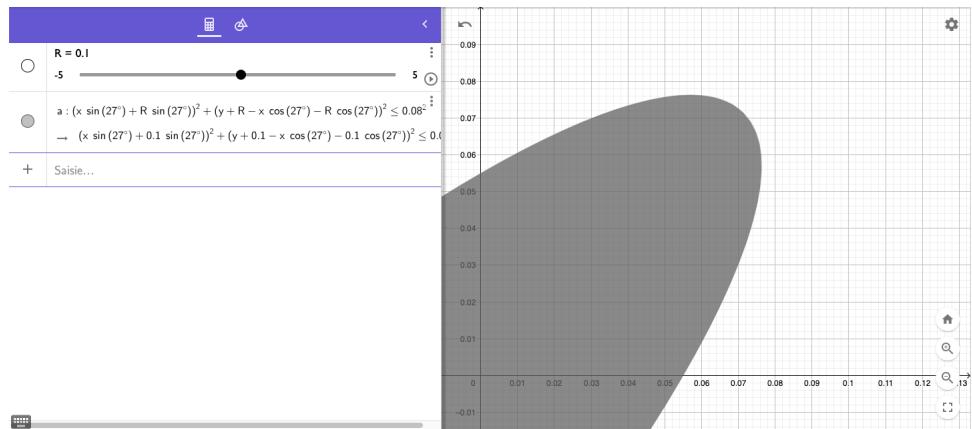


FIGURE 30 – En abscisses CA et en ordonnées DB

On obtient donc une précision maximale de 7 cm

11 Lien avec le groupe INF09

Comme mentionné plus haut, nous travaillions tout au long de l'année sur ce projet en collaboration avec le groupe INF09 qui s'occupait de la partie *Software* du robot. On veillait à ce qu'on travaille ensemble dans le Fablab pour assurer une certaine communication et cohérence entre nos travaux respectifs afin de mener à bien ce projet.

En effet, le groupe d'informatique comportait 4 personnes dont chacune s'occupait d'un aspect différent. Leur stratégie, toujours en cohérence avec la nôtre, se concentre sur la base mobile du robot en implémentant un algorithme de déplacement qui permet d'atteindre les palets suivant un chemin optimal et qui met la table à jour quand la position des éléments est modifiée. Cet algorithme capte les informations grâce aux capteurs, mais aussi la caméra qui est déposée sur le mat de la table et communique la position des objets au robot. Une autre mission du groupe consistait à concilier entre les différentes cartes Arduino qui capte les informations et Raspberry qui fait tourner l'algorithme. Finalement, l'asservissement était un point commun entre nos deux groupes car comporte une partie code mais aussi des consignes vitesse à communiquer au moteur.

Cinquième partie

Conclusion

Acquis

Le PSC nous a permis, en addition avec les connaissances scientifiques en lien avec la robotique développées, de mener un projet en équipe. La gestion de ce projet en groupe nous aura appris à gérer un budget de plusieurs milliers d'euro, de la recherche de sponsor à la commande du matériel adapté. Le Design Control, suggéré par notre partenaire, nous aura éclairé sur une méthode industrielle originale. Il ne faut toutefois pas oublier que ce projet nous aura rendu polyvalent en raison de l'étendue des domaines balayés par la robotique et de tous les problèmes soulevés à résoudre en s'adaptant.

Objectifs initiaux

Même si nous à l'heure actuelle, le robot n'est pas encore apte à passer les tests d'homologation. Le but notre mandat en premier lieu était de fournir une base roulante stable et évolutive pour les prochaines années. C'est une stratégie à long terme.

Le premier étage du robot (base roulante) est prêt à être assemblé dans une coque et l'objectif "rouler en ligne droite" est atteint. Nous avons vérifié la correction des roues codeuses sur l'asservissement, en lien avec le pôle informatique. Même en appliquant une perturbation (en déséquilibrant le robot pour amener une des roues motrice à déraper), le caractère intégrateur du correcteur corrige l'erreur et le robot a une trajectoire globale droite.

Transmission des connaissances

Comme nous l'avons mentionné, nous avons fait le choix d'établir une base solide et modulable pour les éditions futures de la coupe de France de robotique. Cela implique de transmettre notre projet à la nouvelle promotion X2018. Nous même avons été un groupe formé en autonomie avant de contacter le groupe des X2016 pour reprendre leur PSC. Ils nous informé des différentes problématiques et présenté leur robot.

Nous pensons qu'il est désormais nécessaire de mettre l'accent sur la passation du projet. Nous comptons présenter le PSC lors de la présentation des PSC scientifiques (binets AstronautiX et X-Robot). Les équipes qui sont gagnantes sont celles qui ont l'expérience des années. Malheureusement, à l'X le cursus laisse moins d'un an sur le projet et la période de passation est trop courte¹. Il nous semble donc pertinent, si les

1. On peut constater la même problématique pour les autres compétitions étudiantes telles que le

effectifs le permettent suite à notre présentation, de faire une sélection des profils pour monter la prochaine équipe de PSC (entretien / lettre de motivation).

De plus, la partie la plus importante de la transmission sera de :

- Faire une présentation du robot actuel,
- Former les nouveaux membres aux bases des connaissances mises en oeuvres (selon leur nouveaux postes respectifs avec un système de parrainage),
- Leur confier leur premières missions avant de les laisser en autonomie.

Il faut éliminer la période flottement du début de PSC pour donner un cadre organisationnel solide au futur groupe, le but étant qu'ils se concentrent sur les prochains objectifs.

Dans les évolutions futures, nous estimons qui faudra traiter différentes thématiques. Pour gagner en vitesse, nos successeur pourront notamment implémenter les nouvelles commandes moteurs que nous avons commandé qui optimisent le contrôle des moteur. La préhension des objets avec un bras articulé (technologie universelle) par exemple permettra de réaliser des tâches plus complexes pendant la compétition car chaque année la question se pose (édition 2019 : Goldenium, édition 2018 : empiler des cubes). Notre architecture prévoit ces améliorations futures avec l'ajout des bloc modules par étage.

Nous souhaitons faire remonter l'X dans le classement de la coupe de France de robotique. Pour rivaliser avec les meilleures équipes et tirer parti de nos profils de jeune polytechniciens, il faut d'abord résoudre ce problème de transmission propre à notre cursus ingénieur, en mettant l'accent sur la passation. C'est dans cette optique qu'a été créé la plate-forme des binets scientifiques.

Références

- [1] Yasuo Matsunaka. Caterpillar robot. *Autodesk*, Dec 2018.
- [2] Gerboni Giada. The incredible potential of flexible soft robots, Apr 2018. Ted Talks.
- [3] Jean-François Fauvarque. Batteries... tout l'art de stocker l'énergie. CNAM Paris.
- [4] Yedamale Padmaraj. *Brushless DC (BLDC) Motor Fundamentals*. Padmaraja Yedamale Microchip Technology Inc., 2003.
- [5] Maxon Motor. *Epos 4 hardware reference*, Nov 2017.
- [6] Michael Barr. Introduction to pulse width modulation. *Embedded*, Aug 2001.
- [7] Microstar Laboratories. Ziegler nichols tuning rules for pid, 1999.
- [8] SHARP. *GP2Y0A41SK0F Sensor reference*.

Sixième partie

Annexes

12 Etude du règlement

12.1 Les atomes

Les éléments de jeu principaux sont les quatre types d’“atomes” représentés par des palets de hockey de masses différentes. Il y en a 38 au total sur le plateau. Pour marquer des points, le robot doit les ramasser, les identifier, les déplacer et les ranger. Plus les atomes sont lourds, plus ils sont rares et plus ils rapportent de points.

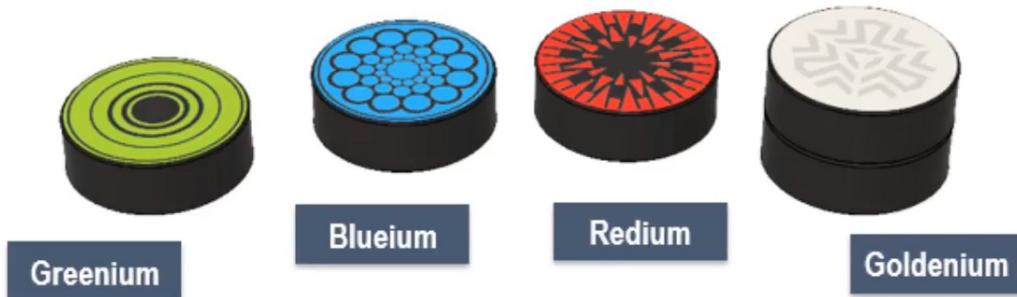


FIGURE 31 – Les éléments de jeu : les ”atomes”

12.2 Le plateau de jeu

Il constitue la zone où évoluent les robots lors du match. Comme précisé plus haut, il est quasi symétrique, à la couleur près (pour faciliter la distinction des équipes par le public, l'une d'entre elles est marquée en jaune, et l'autre en violet).

Le départ s'effectue dans les cases rouges (marquées "Rd"). En plus de ce qui est représenté ci-dessus, chaque équipe dispose de trois supports de balises au bord du terrain (des points d'attache) sur lesquels elles peuvent fixer des dispositifs de repérage. Elles disposent également de la tour centrale (en haut de la fig. 32, tronquée sur l'image) haute de 1 mètre, sur laquelle elles peuvent installer des appareils.

12.3 Les actions



FIGURE 32 – Le plateau de jeu

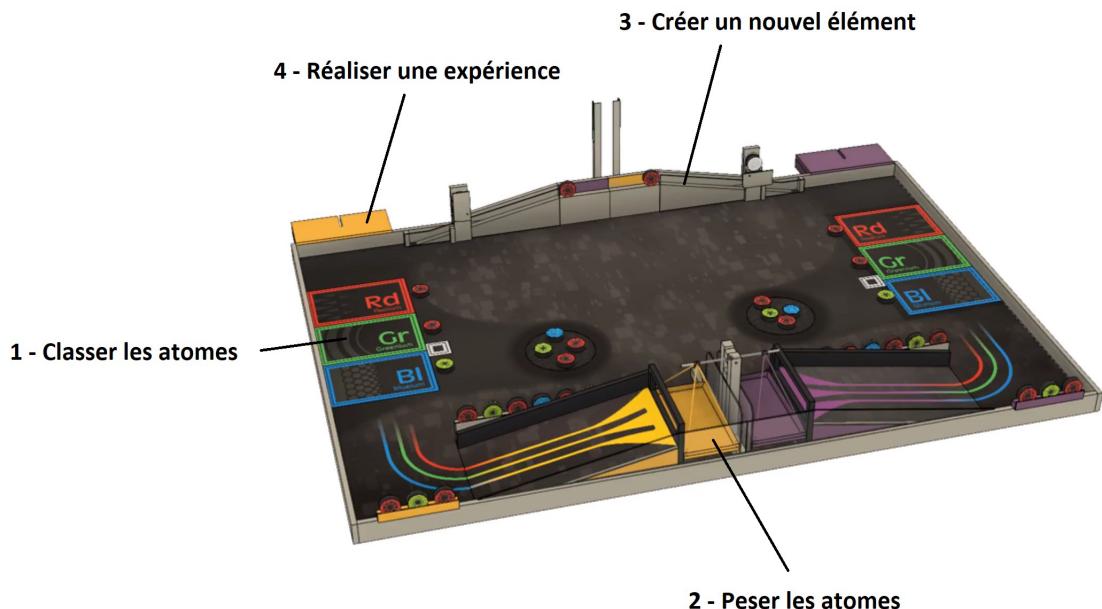


FIGURE 33 – Vue d'ensemble des différentes actions permettant de marquer des points

Classer les atomes

- 1 point par atome dans une case d'une autre couleur
- 5 points par atome dans une case de sa couleur
- 20 points pour le Goldenium (atome le plus lourd, présent en un seul exemplaire par équipe) dans sa case (petit carré blanc)

Cette action est intéressante pour nous car elle ne nécessite aucun actionneur compliqué : un nombre important d'atomes est disposé sur le sol dès le début du match, il



FIGURE 34 – Action : classer les atomes

nous suffira de les pousser dans les cases pour marquer des points.

Peser les atomes

Le but est de déposer des atomes dans la balance.

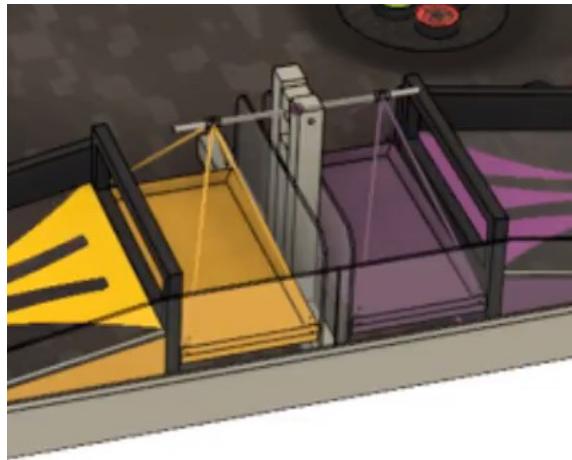


FIGURE 35 – Action : peser les atomes

Plus les atomes sont lourds, plus ils rapportent de points, et ceci est d'autant plus important qu'on ne peut placer que 6 atomes maximum dans la balance. Toutefois, nous avons renoncé à cette action, car l'accès à la balance peut se faire soit en empruntant le plan incliné, soit en soulevant les atomes pour les déposer depuis le milieu du plateau. Ces deux possibilités nous ont paru trop contraignantes : pour soulever les atomes, il nous aurait fallu munir le robot d'un actionneur complexe, ce qui allait à l'encontre de notre philosophie ; et emprunter le plan incliné posait des problèmes techniques qui nous pa-

raissaient difficilement surmontables, comme la déformation du polygone de sustentation du robot (qui n'est plus plan au moment de la transition plat-plan incliné).

Créer un nouvel élément

Au début du match, un atome est positionné en haut d'une rampe inclinée appelée "accélérateur de particules" (voir fig. 36). Le robot doit pousser cet atome pour le faire dévaler la rampe, ce qui libère le Goldenium, atome le plus lourd, présent en un seul exemplaire par équipe.

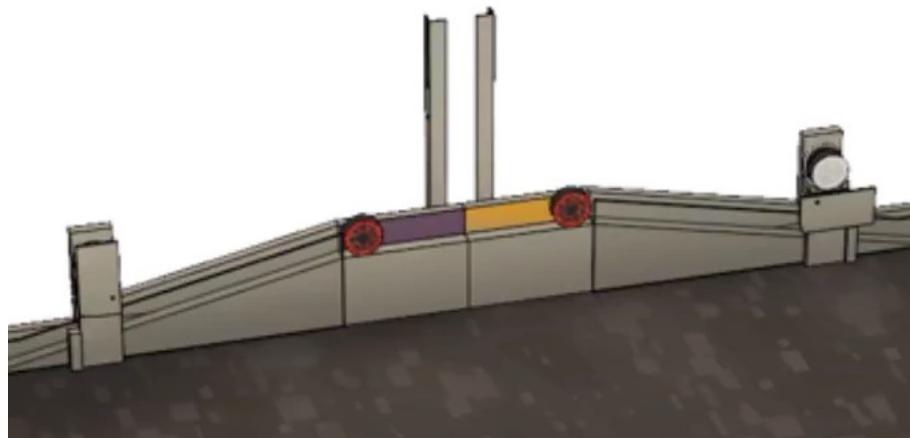


FIGURE 36 – Action : Créer un nouvel élément

- 10 pts pour avoir poussé l'atome prépositionné dans la rampe
- 10 pts supplémentaire pour chaque atome de plus inséré dans la rampe
- 20 pts pour avoir récupéré le Goldenium (l'avoir juste fait tomber de son support suffit)

Cette action est très intéressante car elle rapporte des points même si elle n'est exécutée que partiellement. La difficulté principale est qu'elle nécessite de s'aventurer du côté "adverse" du plateau. Bien que ce soit parfaitement autorisé, cela augmente les risques de croiser la route du robot adverse, ce qui implique de réagir de façon adéquate (l'éviter, recalculer notre trajectoire, etc.). Nous avons décidé de pousser l'atome déjà prépositionné (une simple tige non motorisée fixée à la bonne hauteur sur le robot suffit à effectuer cette action) et de faire tomber le Goldenium une fois celui-ci libéré, selon le même principe d'actionneur passif. En revanche nous avons renoncé à insérer des atomes supplémentaires dans la rampe, car nous ne voulons pas doter notre robot de la capacité de soulever un atome, ce qui nous paraît trop compliqué.

Évaluer sa performance

Un bonus est accordé aux équipes qui estiment avec précision le nombre de points qu'elles marquent. Celui-ci peut être calculé en cours de match par le robot, ou bien fixé par l'équipe avant le début du match.

$$Bonus = 30\% \times \text{Score réalisé} - \text{Écart à la prédition}$$

Si l'écart est tellement grand que ce bonus devient négatif, il est ramené à zéro (pas de malus). Introduit l'année dernière, ce bonus sert à favoriser les robots plus modestes (qui ne vont pas tenter de réaliser énormément d'actions) mais plus fiables, ce qui correspond exactement à notre stratégie.

13 Design Control

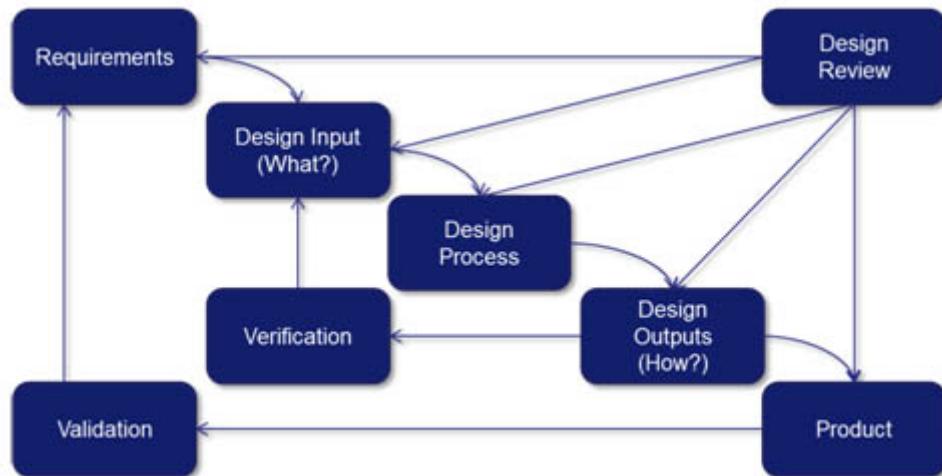


FIGURE 37 – La démarche du Design Control

Après un après-midi de formation avec Exotec Solutions, nous avons donc commencé la planification, qui consistait en la rédaction de plusieurs documents :

SRS (System Requirement Specification)

Un tableau qui décrit exactement ce que le système doit être capable de faire selon les demandes de l'utilisateur et des autres acteurs. Il faut prendre en compte les contraintes environnementales et réglementaires.

TDR (technical detail review)

Une présentation qui résume le brainstorming et les différentes réflexions qui ont conduites

6 - Robot abilities

| | |
|---------|--|
| REQU 24 | The robot shall be able to play 3 matches (3 x 100s) in a row when its batteries are fully charged |
| REQU 25 | The robot shall be able to move forward on a straight line with a precision of 5cm after 1m |
| REQU 26 | The robot shall be able to spin around a vertical axis with a precision of 1° (rotation <= 90°) or 2° (rotation > 90°) |
| REQU 27 | The top speed of the robot shall be >= 0,5m/s |
| REQU 28 | The robot shall reach the speed of 0,5m/s in less than 1s |
| REQU 29 | When rolling at full speed (0,5 m/s), the robot shall be able to brake and stop in less than 200mm |

FIGURE 38 – Exemple de l'un de nos SRS

au choix d'un élément du système. Elle doit résoudre un problème et justifier les choix du matériel du robot. Le document commence d'abord par l'exposition du problème à traiter : "Input", ensuite on passe à la phase "Analysis" où on expose les solutions existantes, leurs avantages et inconvénients. Enfin, la présentation se clôt par une partie "Choice" où on explique la solution retenue pour notre problème initial.

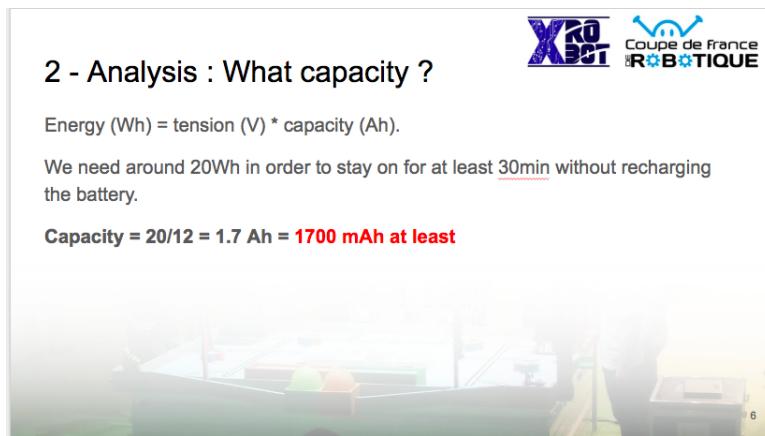


FIGURE 39 – Une page extraite de la présentation TDR ”Alimentation”

TP (Test Plan)

Un tableau qui décrit la façon dont on va tester le système pour vérifier qu'il remplit les exigences des SRS. Il faut prendre des marges sur les besoins du SRS pour s'assurer de respecter le cahier des charges. Le document TP doit absolument être non ambigu : le robot passe ou échoue le test. Il faut aussi penser à tout tester, même les points les

plus évidents. Les SRS ont normalement été établis de manière propre et complète, en les respectant scrupuleusement, on peut-être sur que le système remplira au moins la fonction voulue.

Un dernier point qui nous a été conseillé est de répéter dix fois d'affilée le test au lieu d'une seule, pour s'assurer de la fiabilité du robot.

| Step | Actions | Means of measurement | Expected outcome | Tested requirement(s) | PASS/FAIL |
|-------------------------------|--|---|---|---|-----------|
| Programming of the trajectory | Store a chain of 6 adequate waypoints, randomly placed on the match table in a text file. It is also possible to store a set of 10 chains of 6 waypoints (to avoid reprogramming the robot 10 times), then this step shall be done only once. The points shall be chosen randomly for every time. | PL | PL | REQU 44 | PL |
| Batteries | Equip the robot with a fully charged battery. This battery can be replaced by a charged battery only once during all the test. | PL | The capacity of the two batteries is sufficient for the robot to pass all the test. | REQU 26 | ? |
| Installation of the robot | Place all necessary elements (robot, wedge, etc.) next to the match table. Measure the duration of the installation until the robot is correctly placed in its starting zone and all annex elements are removed from the table. | Chronometer (smartphone) | Duration <= 3min | REQU 2 REQU 24 REQU 25 REQU 27 | ? |
| Starting the robot | The robot shall be placed at rest (not deployed). Pull the string. Measure the time taken by the robot to start moving. | Chronometer (smartphone) | The robot starts in <= 1s | REQU 7 | ? |
| Starting position | Measure the initial position (x, y) of the robot. This position is calibrated with the two lasers. | Tape measure Eventually, erasable pen to notch the initial position on the table | PL | PL | PL |

FIGURE 40 – Exemple d'une partie d'un test plan

TR (Test Result)

C'est une mise à jour du document Test Plan qui fait figurer les résultats des tests et les valident ou non. Ce document permet de garder une trace des résultats obtenus à chaque test, en effet, on ne peut pas parier sur le fait que le système passe tous les tests du premier coup. La possibilité de pouvoir comparer les résultats des différentes versions permet de revenir en arrière sur des modifications entraînant des pertes de performances.

Budget

Dans tout projet, on a en général des fonds en quantité limitée. Il est donc nécessaire de garder à jour un budget précis. Ici, il consiste donc en la liste de tout le matériel nécessaire à la construction du robot, avec les prix et le nombre de pièces tout en prévoyant des pièces de remplacement ainsi que des dépenses secondaires comme le déplacement à la coupe de France.

14 Code Arduino

```

1 #if defined(ARDUINO) && ARDUINO >= 100
2 #include "Arduino.h"
3 #else

```

```

4 #include "WProgram.h"
5 #endif
6 // #include <Servo.h>
7
8 #include <digitalWriteFast.h> // library for high performance reads and
      writes by jrraines
9                                     // see http://www.arduino.cc/cgi-bin/yabb2/
   YaBB.pl?num=1267553811/0
10                                    // and http://code.google.com/p/
     digitalWriteFast/
11
12 // It turns out that the regular digitalRead() calls are too slow and bring
      the arduino down when
13 // I use them in the interrupt routines while the motor runs at full speed
      creating more than
14 // 40000 encoder ticks per second per motor.
15
16 #define EN_L 8 // direction left
17 #define PWML 7
18
19 #define EN_R 6 // direction right
20 #define PWMR 5
21
22 #define A_L 19 // encodeur gauche
23 #define B_L 25
24
25 #define A_R 18 // encodeur droit
26 #define B_R 24
27
28 #define diametreR 57          // diam tre de la roue codeuse droite en mm
29 #define diametreG 56.7        // diam tre de la roue codeuse gauche en mm
30 #define ecartRoues 327        // cart entre les roues codeuses en mm
31 #define K_L 0.297            // 1 tick de l'encodeur correspond un
      placement de la roue de 0.297mm
32 #define K_R 0.2985           // pareil avec la roue droite
33
34 // Left encoder
35 #define LeftEncoderIsReversed
36 volatile bool _LeftEncoderBSet;
37 volatile long _LeftEncoderTicks = 0;
38 volatile long _LastLET = 0;
39
40

```

```
41 // Right encoder
42 volatile bool _RightEncoderBSet;
43 volatile long _RightEncoderTicks = 0;
44 volatile long _LastRET = 0;
45
46
47 // Données de positionnement
48 volatile float X;
49 volatile float Y;
50 volatile float Theta;
51
52 volatile long dt = 50; // IMPORTANT : INTERVALLE DE TEMPS DANS LA BOUCLE D'ASSERVISSEMENT
53
54 volatile float vgdt;
55 volatile float vddt;
56
57 volatile float dx;
58 volatile float dy;
59 volatile float dTheta;
60 volatile float OD = ecartRoues / 2;
61 volatile float OCIR;
62 volatile float vodt;
63 volatile float omegadt;
64
65
66
67 void setup() {
68     // put your setup code here, to run once:
69
70     // Initialisation de la position dans le repere de la table
71     X = 0;
72     Y = 0;
73     Theta = 0;
74
75     Serial.begin(9600);
76
77     // Quadrature encoders
78     // Left encoder
79     pinMode(A_L, INPUT);      // sets pin A as input
80     //digitalWrite(A_L, LOW); // turn on pullup resistors
81     pinMode(B_L, INPUT);      // sets pin B as input
82     //digitalWrite(B_L, LOW); // turn on pullup resistors
```

```

83     attachInterrupt(digitalPinToInterruption(A_L), HandleLeftMotorInterruptA,
84     RISING);
85
86     // Right encoder
87     pinMode(A_R, INPUT);           // sets pin A as input
88     //digitalWrite(A_R, LOW);    // turn on pullup resistors
89     pinMode(B_R, INPUT);           // sets pin B as input
90     //digitalWrite(B_R, LOW);    // turn on pullup resistors
91     attachInterrupt(digitalPinToInterruption(A_R), HandleRightMotorInterruptA,
92     RISING);
93
94     pinMode(PWML, OUTPUT);        // PWM left output
95     pinMode(PWMR, OUTPUT);        // PWM right output
96     pinMode(EN_L, OUTPUT);        // direction left output
97     pinMode(EN_R, OUTPUT);        // direction right output
98
99
100    int speed = 0;
101
102    analogWrite(PWML, speed);
103    analogWrite(PWMR, speed);
104 }
105
106 void loop() {
107     // put your main code here, to run repeatedly:
108
109     for(int i = 0 ; i < 1000 ; i++){
110         delay(dt);
111
112         // On calcule d'abord le placement du robot pendant la dure dt
113         // dans le rep re local (x, y, theta)
114
115         vgdt = (_LeftEncoderTicks - _LastLET)*K_L;
116         vddt = (_RightEncoderTicks - _LastRET)*K_R;
117
118         // Imm diatement apr s, on update les compteurs de clicks :
119         _LastLET = _LeftEncoderTicks;
120         _LastRET = _RightEncoderTicks;
121
122         // Puis on calcule tout

```

```

123     if (vgdt - vddt != 0){
124         OCIR = (vgdt + vddt)/(vgdt - vddt)*OD;
125         omegadt = (vddt - vgdt)/ecartRoues ;
126
127         dx = OCIR * (1 - cos(omegadt));
128         dy = -OCIR* sin(omegadt);
129     }
130     else{ // vgdt == vddt c'est- -dire que le robot est all tout droit (
131     ou qu'il n'a pas boug )
132         dy = vddt;
133         dx = 0;
134         dTheta = 0;
135     }
136
137
138 // Puis on update le positionnement absolu par rapport au rep re de la
139 // table (X, Y, Theta)
140 X = X + cos(Theta) * dy + sin(Theta) * dx;
141 Y = Y - cos(Theta) * dx + sin(Theta) * dy;
142 Theta = Theta + omegadt;
143
144
145
146
147 //Serial.print("Ticks      droite : ");
148 //Serial.println(_RightEncoderTicks);
149 //Serial.print("Ticks      gauche : ");
150 //Serial.println(_LeftEncoderTicks);
151 //Serial.println(" ");
152
153 Serial.print("X      = ");
154 Serial.println(X);
155
156 Serial.print("Y      = ");
157 Serial.println(Y);
158
159 Serial.print("Theta = ");
160 Serial.println(Theta * 180 / 3.14159);
161 Serial.println(" ");
162 }
163

```

```
164     analogWrite(PWML, 0);
165     analogWrite(PWMR, 0);
166 }
168
169 void HandleLeftMotorInterruptA()
170 {
171     // Test transition; since the interrupt will only fire on 'rising' we don
172     // 't need to read pin A
173     _LeftEncoderBSet = digitalReadFast(B_L);    // read the input pin
174
175     // and adjust counter + if A leads B
176     #ifdef LeftEncoderIsReversed
177         _LeftEncoderTicks -= _LeftEncoderBSet ? -1 : +1;
178     #else
179         _LeftEncoderTicks += _LeftEncoderBSet ? -1 : +1;
180     #endif
181 }
182
183 // Interrupt service routines for the right motor's quadrature encoder
184 void HandleRightMotorInterruptA()
185 {
186     // Test transition; since the interrupt will only fire on 'rising' we don
187     // 't need to read pin A
188     _RightEncoderBSet = digitalReadFast(B_R);    // read the input pin
189
190     // and adjust counter + if A leads B
191     #ifdef RightEncoderIsReversed
192         _RightEncoderTicks -= _RightEncoderBSet ? -1 : +1;
193     #else
194         _RightEncoderTicks += _RightEncoderBSet ? -1 : +1;
195     #endif
}
```