

TRAN-H-201 : Projet Multidisciplinaire II

"Death-Stranding" : Conception d'un robot ramasseur-trieuse d'objets

Etudiants :

BOUBAD Souleimane

NAJDI Sami

NAPOLITANO Matteo

TREMENTOZZI David Judah

TSHIBANGU MPIANA David

VISSOL Arthur

Professeurs : Jérémie Roland

Patrick Simon

Tutrice : Louise MASSAGER

Table des matières

Abstract	1
1 Introduction	2
1.1 But du Projet	2
1.2 Cahier des charges	2
1.3 Stratégie prise par le groupe	3
2 Modélisation	4
2.1 Régulateur	4
2.1.1 Régulateur PID	4
2.2 Modèle Simulink	5
2.2.1 Capteurs et Moteurs	5
2.2.2 Commande PWM	6
2.2.3 Calcul d'erreur et régulateur	7
2.2.4 Résultats	9
3 Programme	10
3.1 Tri	10
3.2 Suiveur de ligne	10
4 Les composants	11
4.1 L'Arduino	11
4.2 Le moteur à courant continu	11
4.3 Les capteurs	12
4.3.1 Capteurs suiveurs de ligne	12

4.3.2	Capteur de distance	14
4.3.3	Capteur de contact	15
4.3.4	Capteur de vitesse	16
4.4	Les servomoteurs	16
4.5	La batterie	17
4.6	Le moteur driver	17
4.7	Roues	17
4.7.1	Roue sphérique	17
4.7.2	Roue motrice	18
4.8	Conclusion	18
5	Design	19
5.1	Design du châssis	19
5.1.1	Premier prototype	19
5.1.2	Deuxième prototype	21
5.2	Design de la pince	21
5.2.1	Équation de statique	23
5.2.2	Méthode adoptée et reconnaissance des objets	24
5.2.3	Mécanisme pour lever la pince	24
6	Réalisation dans Arduino	26
6.1	Pilotage	26
6.1.1	Calibrage	27
6.1.2	L'algorithme PID	27
6.1.3	Détection de Branchement de ligne et Virage 90°	27
6.2	Pince et Reconnaissance Objet	27
6.3	Détection d'Objets et Placement du Robot	28
6.4	Tests et Validation du Modèle	29
7	Gestion de projet	30
7.1	Gestion du projet	30
7.1.1	Risques et Problèmes	30
7.2	SWOT	31
7.3	Budget du projet	32

8 Conclusion	34
8.1 Bilan du projet	34
8.1.1 Premier quadrimestre	34
8.1.2 Second quadrimestre	34
8.2 Leçons apprises	35
8.2.1 Le modèle	35
8.2.2 Le pilotage	35
8.2.3 La pince	36
8.2.4 Le design	36
A Annexe	37
A.1 Datasheet du module de capteur infrarouge	37
A.2 Datasheet de réseau de capteur de réflectance QRE1113RC	37
A.3 Dimensions du réseau de capteur de réflectance QRE1113RC	38
A.4 Datasheet du capteur Ultra-sons HC-SR04	38
A.5 Datasheet de la roue sphérique	39
A.6 Fiche technique de l'Arduino Every	39
A.7 Code pour le suivi de ligne	39
A.8 Circuit pour les virages de 90°	42
A.9 Circuit de détection de branchement	42
A.10 Carte test pour le système de détection	42
A.11 Schéma de pince théorique	43
A.12 Schéma de pince théorique	43
A.13 Modélisation des Moteurs dans Simulink	43
A.14 Calcul d'erreur dans Simulink	44
A.15 Fonctionnement de la Conversion PID-PWM	45
A.16 Image de la pince	45

Abstract

Dans le cadre du projet d'électromécanique de deuxième année du bachelier à l'École Polytechnique, un robot facteur autonome doit être construit. Sur une carte, des colis sont placés dans des maisons, le but est de déplacer les colis aux bons endroits.

Trois différents types d'objet sont traités : un cylindre, un prisme à base rectangulaire et un haltère. La carte contient six emplacements, deux pour chaque type d'objet. Trois à quatre objets seront placés aléatoirement sur la carte et le robot devra être capable de les ramasser et les trier sans les faire tomber. La carte dispose d'un marquage noir sur fond blanc que le robot pourra suivre pour se repérer sur la carte. Pour ce faire l'équipe devra choisir et trouver ses composants afin de construire un prototype se pliant au cahier des charges. Un Arduino Nano Every est utilisé pour le contrôle, deux moteurs à courant continu ainsi que dix capteurs infrarouges pour le suivi de ligne et des servomoteurs pour le ramassage d'objets.

Parmi les défis à relever : avoir un robot capable de se déplacer sur la carte avec une précision suffisante à l'accomplissement de sa tâche, le robot se repérera par odométrie et utilisera une méthode de régulation PID. Le robot doit être capable de trier les objets avec un maximum d'efficacité, l'utilisation d'un bon algorithme de tri sera donc cruciale. Enfin, le robot doit être capable d'attraper et reposer les objets sans les faire tomber, un système de pince articulée devra donc être installé sur le robot. En parallèle de la construction de prototype seront menées des simulations sur SIMULINK afin de prédire au mieux le comportement du robot.

1

Introduction

Dans le cadre de notre deuxième année de bachelier, il nous est demandé de choisir parmi sept projets. Le rapport ici présent portera sur le projet d'électromécanique qui consiste à créer un robot trieur d'objet.

1.1 But du Projet

Le but du projet est de créer un robot facteur qui trie les objets et les ramène dans les bonnes maisons. Les objectifs du projet sont variés et bénéfiques pour de nombreuses personnes. Leur application peut être observée dans des entreprises de grande envergure telles qu'Amazon. En employant des robots entièrement autonomes, il devient envisageable de décharger le personnel, accélérant ainsi le processus de tri des objets et même leur déplacement.

1.2 Cahier des charges

Le cahier des charges pour ce projet exige que le robot puisse trier des objets mis au hasard dans un plan prédéfini. Il faut que le robot soit autonome et capable de déposer les colis sans les faire tomber, le plus rapidement possible et aux bons endroits, comme indiqué sur la carte. 1.1.

Les colis seront au nombre de trois et ont été imprimés en PLA :

- **un haltère** : 8cm de haut et 4cm de diamètre et 2cm de diamètre pour le plus petit cercle.
- **un cylindre** : 8cm de haut et 4cm de diamètre
- **un parallélépipède rectangle à base carrée** : 8cm de haut et 4cm de côté

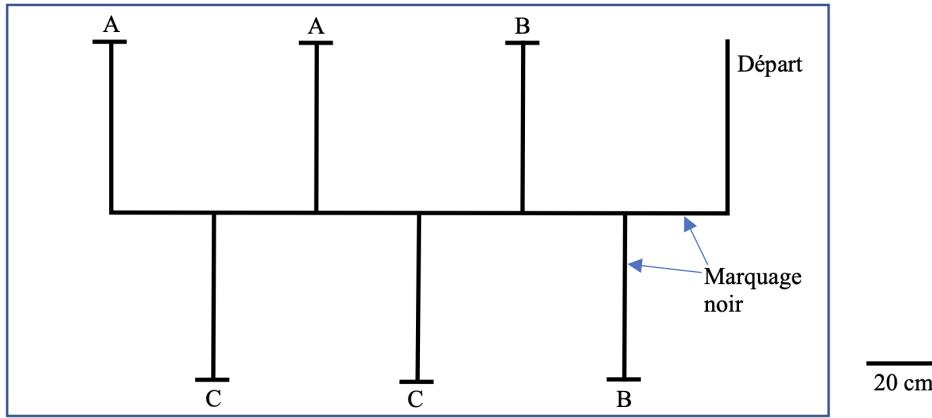


FIGURE 1.1: Carte et emplacement des objets

1.3 Stratégie prise par le groupe

Il a été intéressant de choisir une stratégie pour réaliser le projet. Il a fallu analyser les pistes à suivre et celles à ne pas suivre. Le groupe a décidé de se focaliser sur trois points importants :

- **Une partie modélisation informatique** qui consiste à concevoir un prototype sur Fusion 360 et de tester l'algorithme sur une plateforme de test comme Simulink
- **Une partie construction et choix des composants** qui consiste à trouver les bons composants et le choix des matériaux pour que le robot puisse rouler et prendre les objets sans souci.
- **Tests et Validation du Modèle** qui consiste à tester le prototype avec les données obtenu par le modèle et de vérifier ce dernier et éventuellement changer certains aspects si nécessaire, majoritairement par essai-erreurs.

Modélisation

Pour permettre au robot de bien suivre la ligne, il faut qu'il soit équipé d'algorithmes qui l'aident à corriger les erreurs éventuelles durant son trajet.

2.1 Régulateur

Pour pouvoir gérer le mouvement du robot et s'assurer qu'il suive la ligne de manière efficace, un régulateur a dû être choisi. Il existe plusieurs types de régulateur qui sont vus en profondeur dans le cours d'automatique en BA3. Il a été remarqué après de nombreuses recherches que le régulateur PID serait le plus utile et le plus utilisé pour le cas d'un robot suiveur de ligne. Il s'agit d'une question de simplicité à intégrer dans le code final et permet également d'éviter les surchauffes d'Arduino inutiles.

2.1.1 Régulateur PID

L'information présentée dans cette section sur le fonctionnement des régulateurs PID est basée sur le cours de vulgarisation "What is PID control?" de MathWorks [17]

Un régulateur PID (ou régulateur proportionnel, intégrale et dérivée) est un système de contrôle qui permet d'améliorer un procédé automatique en minimisant l'erreur (l'écart entre l'état désiré et l'état actuel) du système. Le régulateur prend ainsi comme entrée l'erreur et donne une action au système qui a pour but de minimiser cette erreur. Le terme proportionnel traduit le fait que l'action à effectuer sur le système doit être proportionnelle à l'erreur actuelle. Le terme intégral permet de tenir compte des erreurs précédentes permettant de minimiser les erreurs résiduelles. Le terme dérivée permet d'estimer les erreurs futures et d'en prendre une action pour anticiper ceux-ci.

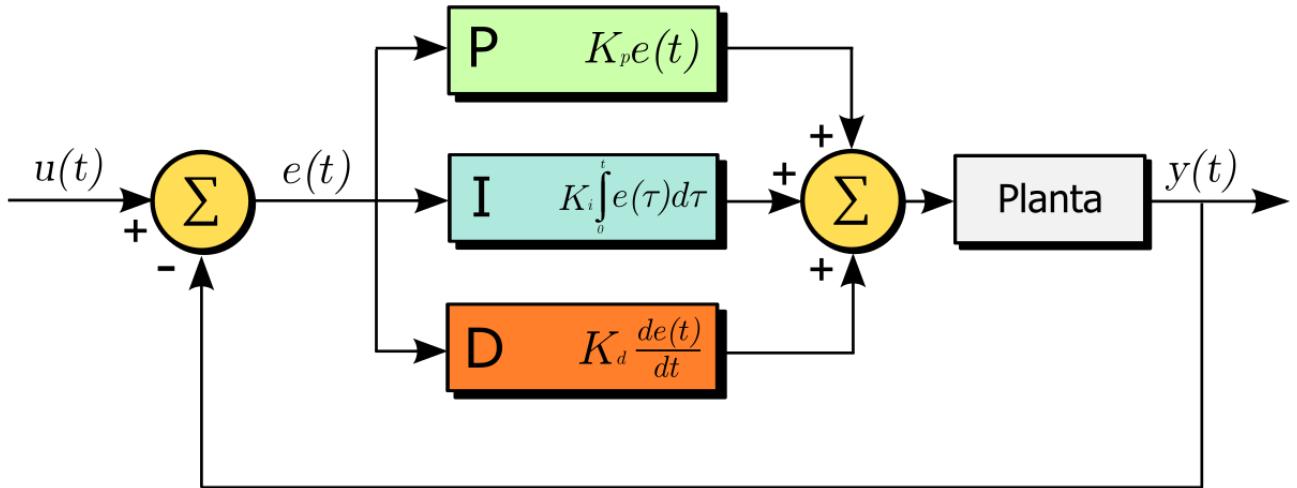


FIGURE 2.1: Exemple de PID général

Chaque terme présent dans le PID est multiplié par un coefficient, il est essentiel de trouver des coefficients qui garantissent un pilotage stable du robot. Pour ce faire, un modèle Simulink a été créé pour pouvoir tester rapidement les différents coefficients.

2.2 Modèle Simulink

Le modèle présenté dans cette section a été réalisé à l'aide des cours de vulgarisation de l'outil Simulink [18] et du cours d'introduction à la bibliothèque Mobile Robotics Training Toolbox [19].

Le modèle Simulink doit être créé de manière à ce qu'il reflète suffisamment la réalité, en d'autres mots, son fonctionnement doit être suffisamment proche de celui du robot réel. Pour ceci, il faut modéliser les différents composants à utiliser sur le robot. La bibliothèque Mobile Robotics Training Toolbox permet de modéliser des capteurs infrarouges (analogiques ou numériques) ainsi que les moteurs. Ainsi le choix d'utiliser les capteurs de ligne analogiques a été pris (voir figure : 4.5).

2.2.1 Capteurs et Moteurs

Pour les capteurs analogiques (réflectance), les valeurs correspondant à la couleur noire de la ligne ainsi que celle de l'environnement ont été déterminées expérimentalement et valent respectivement 20 et 10 environ. Les capteurs numériques renvoient 1 si la ligne n'est pas présente et 0 sinon.

Les modules "Line Sensor Simulation" définis dans "Mobile Robotics Training Toolbox" ont été

utilisés pour simuler ces capteurs. Les valeurs que doit renvoyer le capteur lorsqu'il voit la ligne et lorsqu'il ne voit pas la ligne doivent être définies pour chaque module, ce qui justifie son utilisation pour les capteurs digitaux ainsi que les capteurs analogiques.

Chaque moteur a dû ensuite être caractérisé par son lien commande PWM et vitesse angulaire. Pour ce faire un test a été effectué pour chaque moteur dans lequel on mesure la vitesse angulaire à l'aide d'incrémentateurs pour chaque commande PWM (de 0 à 256).

2.2.2 Commande PWM

Le "Pulse Width Modulation (PWM)" est une technique utilisée pour générer des signaux pseudo-analogique à partir de signaux digitaux d'impulsions différentes de manière à pouvoir obtenir un ensemble de valeurs de voltage différents à partir de ceux de référence correspondant aux 0 et 1 digitaux.

Les données obtenues sont les suivantes :

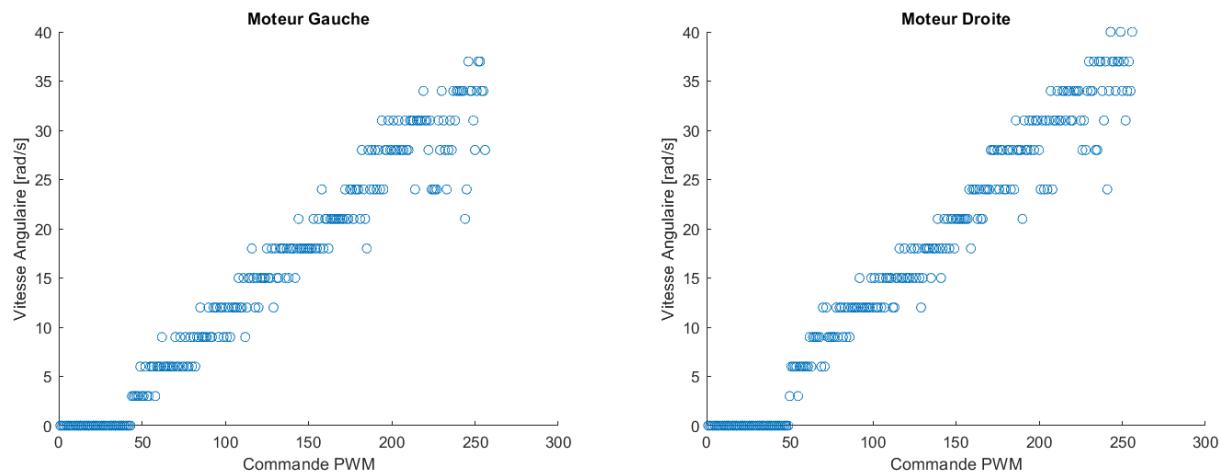


FIGURE 2.2: Vitesse angulaire des moteurs en fonction de la commande PWM

Finalement, pour utiliser ces données dans le modèle Simulink, une approximation linéaire a été effectuée sur la partie au-dessus du PWM minimale (soit PWM_0) pour mettre en rotation la roue. Pour les valeurs inférieures à PWM_0 , la vitesse angulaire vaut 0. On obtient les approximations linéaires suivantes :

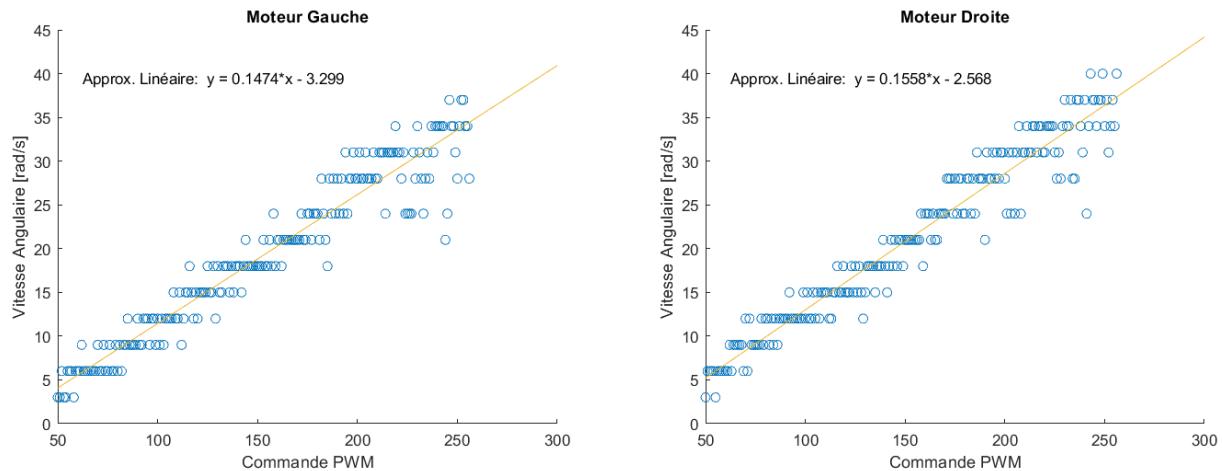


FIGURE 2.3: Approximation linéaire du lien PWM vitesse angulaire

La modélisation des moteurs et des capteurs au sein du modèle Simulink (voir annexe : A.12) est maintenant possible à partir de ces informations et sachant que $PWM_0 = 50$.

2.2.3 Calcul d'erreur et régulateur

Comme énoncé dans la section 3.2, le type de régulateur choisi est un régulateur PID. Celui utilisé dans le modèle de simulation et le prototype prend comme erreur le signal reçu des capteurs IR analogiques et qui renvoie comme action la vitesse angulaire de l'orientation du robot (ω) qu'il doit effectuer pour corriger sa déviation de la ligne. Pour pouvoir utiliser cette action dans le modèle, il faut définir une conversion de la commande ω vers les vitesses angulaires des roues (ϕ_G et ϕ_D pour la roue gauche et droite respectivement).

Capteurs et erreur

Les capteurs analogiques sont des modules prédéfinis dans la bibliothèque

[Mobile Robotics Training Toolbox](#). L'erreur totale est la somme des écarts entre l'état désiré et actuel de chacun des capteurs. Après avoir effectué des tests sur les capteurs analogiques, la valeur correspondant à la couleur noire peut être représentée par 20 et celle de la couleur blanche par 10. Il est à noter que ni le tri des objets ni la gestion des branchements de la ligne ne sont modélisés dans le modèle Simulink. Lorsque le modèle rencontre des branchements dans la ligne il les ignore donc et continue en considérant que la contribution à l'erreur de ce branchement est nulle. Pour visualiser l'implémentation du calcul d'erreurs dans Simulink, voir annexe A.13.

PID

Le régulateur PID utilisé dans le modèle est le suivant :

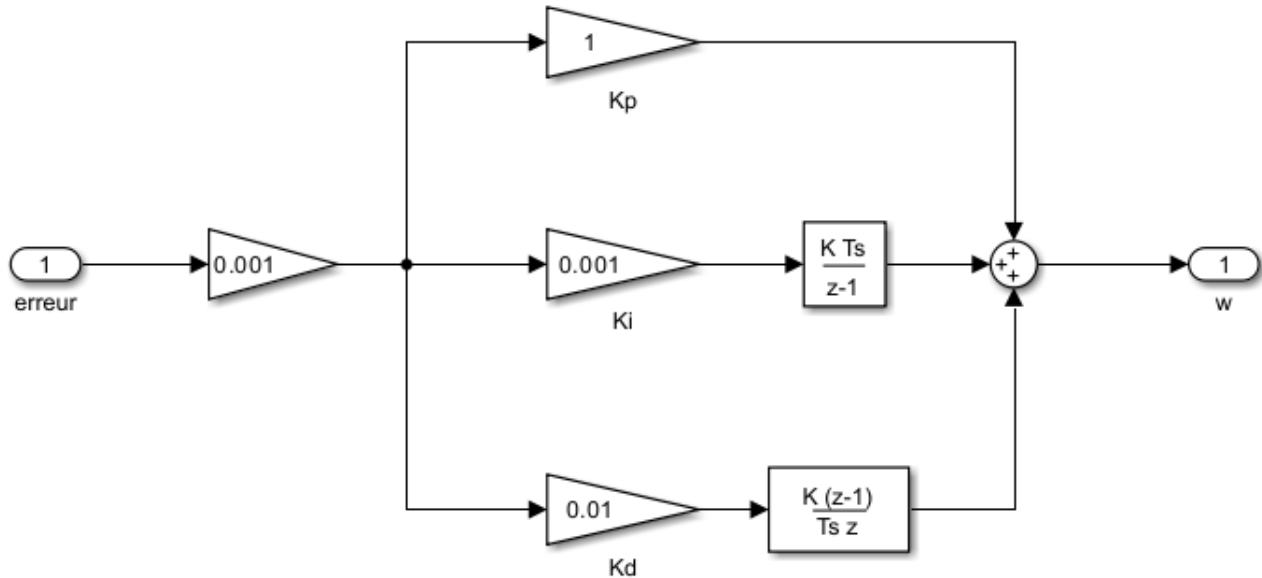


FIGURE 2.4: Subsystem PID utilisé dans le modèle

Ce PID ne présente pas de boucle fermée au sens strict, car dans le cas de la modélisation choisie, la plante est les vitesses angulaires des roues et l'entrée est le signal reçu des capteurs, ce qui dépend directement de la position du robot et donc des vitesses angulaires des roues.

Pour pouvoir utiliser ce PID, il faut convertir la vitesse de rotation du robot en vitesses angulaires des roues. Pour ce faire, un autre Subsystem Simulink a été construit dans ce but.

Pour en apprendre plus sur son fonctionnement voir A.14

Problèmes rencontrés et solutions

Le problème rencontré en testant ce modèle, était qu'il avait du mal à gérer les virages soudain de 90° présents sur la carte dans le cahier de charges. Pour régler ceci, on a eu recours à une condition supplémentaire pour ce cas extrême.

La condition est la suivante : si le capteur IR digital situé à l'avant du robot perd de vue la ligne, le robot doit s'arrêter et tourner dans le sens correspondant au ω précédent pour ensuite reprendre son algorithme PID. Cette condition prend la priorité sur le pilotage PID.

2.2.4 Résultats

Les résultats ci-dessous montre que le robot modélisé arrive à compter les branchements de la ligne et à continuer à suivre son trajectoire sans en être dérangé.

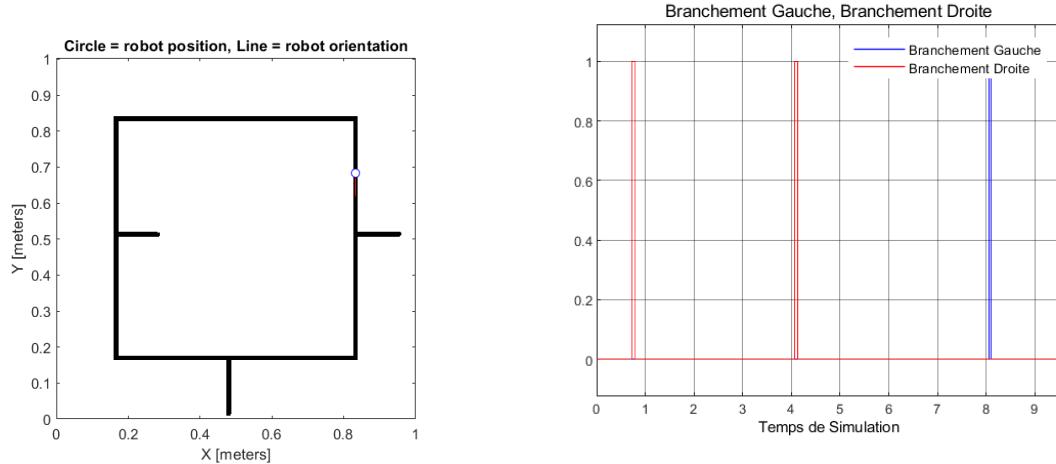


FIGURE 2.5: Carte test pour le système de détection de branchement et résultat

Ceci termine la modélisation de suivi de ligne pour le robot.

Ce chapitre porte sur les algorithmes utilisés par le robot lui permettant de satisfaire de le cahier des charges. Ceux-ci sont écrits dans le langage Arduino C++, un dérivé du C++ basique, dans l'application Arduino IDE.

3.1 Tri

La fonction première du robot est de pouvoir trier les objets, de sorte à ce que chaque maison aie le colis demandé. Pour cela, il faut un algorithme de tri. L'algorithme de tri a pour information donnée le nombre de maisons, leur position, la position des colis et l'endroit où ils sont demandés. Le code travaille uniquement avec les maisons ne contenant aucun colis, et ceux qui ont un colis qui n'est pas le bon. Il ne prend pas en compte les maisons qui ont déjà les bons colis.

Le tri s'effectue en 2 étapes. Première étape, si un colis est au mauvais endroit, et qu'une maison vide veut ce colis, le colis est donné à la maison vide. Deuxième étape, si les colis présents dans les mauvaises maisons ne conviennent pas aux maisons vides, les colis des mauvaises maisons vont dans les maisons vides. Les 2 étapes tournent en boucle jusqu'à ce que tous les colis soient au bon endroit. Le code imprime les déplacements que le robot doit faire pour tout trier.

3.2 Suiveur de ligne

Maintenant que le robot sait trier, il lui faut apprendre à suivre la ligne afin d'apporter les colis aux maisons. Pour l'algorithme suiveur de ligne, dix capteurs infrarouges sont utilisés. Deux binaires, voir figure 4.2 et un analogique comportant 8 capteurs comme montré à la figure. 4.5. Le capteur analogique permet au robot de savoir où se trouve la ligne, et la suit grâce à un algorithme de régulateur PID (voir chapitre "**Modélisation**"). Les 2 capteurs digitaux (ou binaires) sont placés sur les côtés du robot et servent à reconnaître les branchements.

Les composants

Ce chapitre concerne les composants qui ont permis la construction et l'aboutissement du projet. Grâce à ceux-ci, le robot peut suivre la ligne, porter les objets et les détecter.

4.1 L'Arduino

En raison de la complexité de notre mission et de ses besoins en alimentation électrique, le choix logique s'est porté sur l'Arduino inclus dans notre kit de démarrage, à savoir l'Arduino Nano Every. Ce modèle d'Arduino est reconnu pour sa convivialité et sa popularité dans la conception de petits robots. Cependant, il présente une limitation importante qu'il est essentiel de noter : sa capacité de stockage limitée. En cas de code non optimisé, l'Arduino est exposé à un risque significatif de dommages.

L'Arduino Nano Every possède également des caractéristiques pratiques pour notre projet, telles qu'une tension d'alimentation identique à celle fournie par le chargeur inclus dans le kit de démarrage, des dimensions compactes et un poids léger. Malheureusement, il présente également des inconvénients, notamment son Vmax, qui, s'il est dépassé, peut entraîner la fonte de l'appareil.

La fiche technique de l'Arduino Nano Every : voir annexe A.7

En termes de comparaison, il est possible de choisir d'autres microcontrollers, par exemple : Raspberry Pi, mais celle-ci est trop chère pour le budget donné dans le cahier des charges.

4.2 Le moteur à courant continu

Pour faire rouler notre robot et le faire avancer sur le plan, il a fallu chercher des moteurs électriques. Notre choix s'est porté sur les moteurs à courant continu (voir figure : 4.1). Ils sont très sou-

vent utilisés dans les petits robots du fait qu'ils sont stables pour faire avancer un robot contrairement aux moteurs à courant alternatif qui eux sont bien moins stables. Une contrainte importante du moteur choisi est le voltage optimal pour que celui-ci fonctionne, soit compris entre 5V-7V. Ces valeurs correspondent à la tension maximale que peut donner en sortie le motor driver DRI0044. Nous reviendrons sur son utilisation plus tard.



FIGURE 4.1: Kit de moteur à courant continue et de roues standards [13]

4.3 Les capteurs

Le choix des capteurs est, quant à lui, bien plus délicat. En effet, il est possible qu'au cours du projet, le choix de ceux-ci soit modifié selon les différentes améliorations qui seront apportées.

4.3.1 Capteurs suiveurs de ligne

Module de capteur infrarouge

Le premier capteur est le module de capteur infrarouge qui était inclus dans le kit de départ (voir figure 4.2). Ce capteur aura pour fonction de détecter lorsque le robot quittera la trajectoire de la ligne noire. Ce module peut aussi être utilisé comme un capteur de distance, mais d'autre choix seront abordés dans la section suivante.

Comme expliqué par [7], ce module capte la lumière émise et renvoie une intensité de courant proportionnelle à l'intensité lumineuse captée. Dans notre cas, il fera la différence seulement entre le blanc et le noir de la carte. Les spécificités du capteur se trouvent au tableau dans l'annexe A.1.

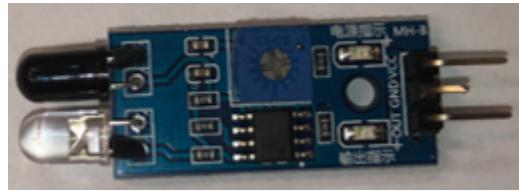
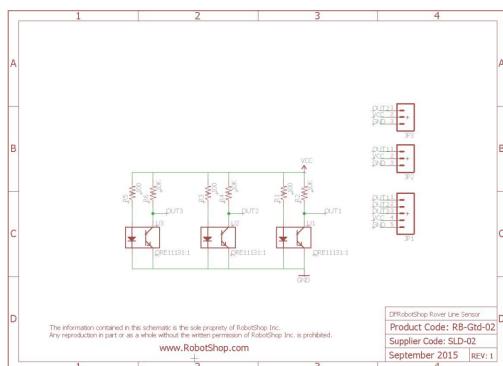


FIGURE 4.2: Capteur infrarouge

Réseau de capteurs infrarouge

Un autre capteur permettant de suivre la ligne noire est le réseau de capteurs de réflectance ou plus simplement capteur analogique. C'est une fine plaque sur laquelle trois capteurs infrarouges ou plus sont placés à intervalles réguliers et renvoient chacun les données reçues, comme montré sur le schéma, voir figure 4.3. Cela permet de déterminer si le robot suit la ligne, car alors les trois capteurs renvoient la valeur associée à la couleur noire. Par contre, le capteur doit être très proche du sol pour pouvoir fonctionner correctement, de l'ordre du 3 à 6 mm du sol, comme expliqué dans le tableau en annexe A.2.

Durant le premier quadrimestre, c'est le capteur QRE1113RC, voir figure 4.4 qui fut utilisé. Celui ayant été fourni avec le kit de projet dont les dimension ce trouve en annexes A.3 et A.4. Mais il a dû être remplacé. En effet, les trois capteurs présents sur la plaque avaient tendance à se comporter comme des capteurs digitaux. Vu que le sol est soit blanc, soit noir, les capteurs analogiques détectent soit le sol, soit la ligne, mais aucune nuance par rapport au centre ce qui rendait impossible de savoir si le robot se trouvait à gauche ou à droite de la ligne et donc d'appliquer le PID. Il sera alors remplacé par le capteur QTR-8RC, voir figure 4.5 muni de huit capteurs différents qui permettent une meilleure précision et détection de la ligne noir.

FIGURE 4.3: Schéma du circuit électrique du capteur QRE1113RC
[25]

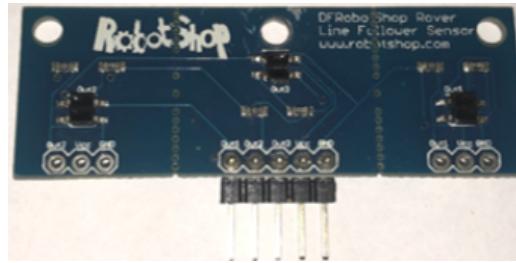


FIGURE 4.4: Capteur de réflectance QRE1113RC

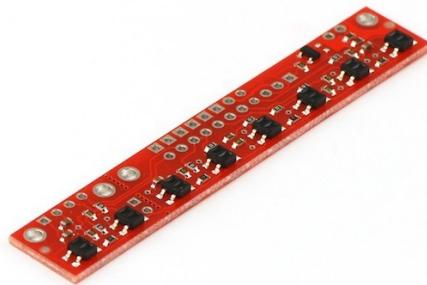


FIGURE 4.5: Capteur analogique infrarouge QTR-RC8
[1]

4.3.2 Capteur de distance

Capteur Infrarouge de distance

Les capteurs infrarouges sont utilisés dans la détection d'objets. Dans le cas du capteur GP2Y0A41SKOF, voir figure 4.6, celui-ci détecte des objets pour des distances entre 4cm et 30cm. Ce capteur ne pourra donc pas permettre au robot de détecter s'il y a un objet sur une branche en étant au bout de celle-ci, mais pourrait s'avérer utile pour que le robot se place à la distance optimale pour attraper l'objet.

Capteur Ultra-sons

La deuxième option pour les capteurs de distance est le capteur à ultrasons. Ce capteur est capable de déterminer s'il y a des objets sur de longues distances.

Dans le cadre du projet, un capteur HC-SR04 nous a été fourni, voir figure 4.7. Celui a une distance minimale de détection de 2cm, maximale de 400cm et une erreur de 0.3cm, ce qui devrait permettre au robot de déterminer si oui ou non, il y a un objet sur une branche, comme expliqué dans [22].

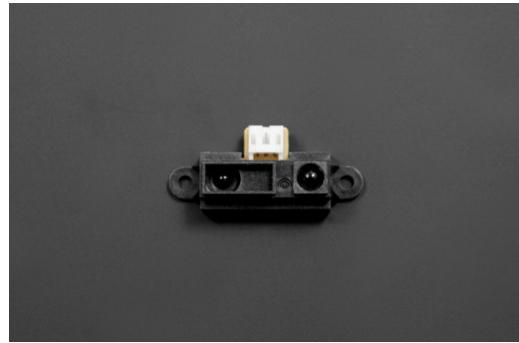


FIGURE 4.6: Capteur infrarouge GP2Y0A41SKOF
[11]

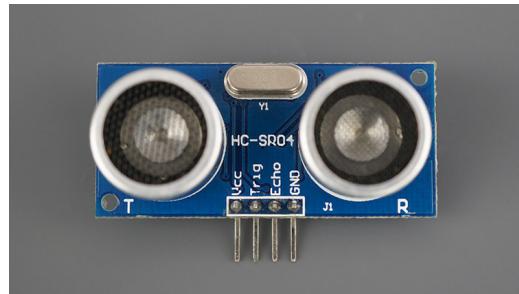


FIGURE 4.7: Capteur ultra-sons HC-SR04
[22]

4.3.3 Capteur de contact

Le capteur de contact, aussi appelé miniature snap-action sensor ou micro switch est un capteur muni d'une ventouse relié à un interrupteur telle que l'on voit sur l'image, voir figure 4.8. Quand une force est appliquée sur la ventouse, l'interrupteur se ferme ou s'ouvre, modifiant le signal, comme expliqué par la fiche du constructeur [8].

Ce capteur de contact sera dédié à la pince, nous permettant de savoir quand est-ce que celle-ci aura attrapé un objet et de déterminer celui-ci.

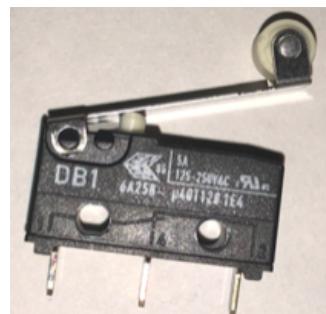


FIGURE 4.8: Capteur de toucher

4.3.4 Capteur de vitesse

Le capteur de vitesse utilisé est un encodeur optique. D'après [24], ce capteur mesure la vitesse angulaire d'une roue auquel il est branché et permet d'en déduire sa vitesse et donc, en connaissant le diamètre des roues, la distance parcourue par celles-ci.

L'encodeur optique fonctionne à l'aide d'une source lumineuse et d'un capteur lumineux. Un disque fin et troué sera placé entre la source et le capteur lumineux, déformant ainsi le signal lumineux, un signal normal et une zone d'ombre où le signal est bloqué. Ainsi, quand la roue est mise en mouvement et connaissant le nombre de trous dans le disque, on peut déterminer la vitesse angulaire de la roue et donc la distance qu'elle a parcourue.

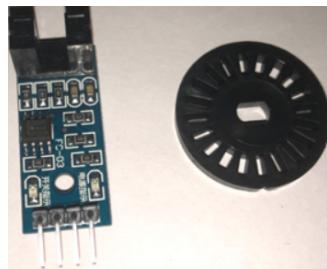


FIGURE 4.9: Encodeur optique

4.4 Les servomoteurs

Le servomoteur est un moteur électrique ne nécessitant pas l'apport d'un driver moteur pour fonctionner. Dans notre cas, le servomoteur DF9GMS est comparable à un petit moteur à courant continu mettant en rotation à 360° un engrenage comme montré dans l'image ci-dessous voir figure : 4.10. Ceci est parfait pour faire tourner les engrenages de la pince du robot.



FIGURE 4.10: Servomoteur DF9GMS 360 degré
[9]

4.5 La batterie

La tension requise pour alimenter l'Arduino se situe dans une fourchette de 7 à 12 V. Une pile 9V a donc été choisie comme moyen d'alimenter le robot en premier lieu. Cette solution s'est avérée ne pas être la plus optimale, en effet les piles 9V ont une capacité d'environ 600 mAh, elles ont donc une autonomie très faible et le courant qu'elles délivrent est bien trop faible pour alimenter la multitude d'éléments présents dans le robot (moteurs, servomoteurs, capteurs,...).

La solution finalement adoptée est de mettre 6 piles AA ayant une tension de 1,5V chacune en série. On obtient donc 7,5V de tension à l'alimentation et une capacité de 2700 mAh, le courant délivré est bien supérieur et permet d'alimenter les éléments nécessaires aux fonctionnement complet du robot.

4.6 Le moteur driver

Le driver moteur est un composant important dans la construction de robots. En effet, une puce de circuit y est intégrée permettant de lier le microcontrôleur avec le moteur. De plus, des drivers moteurs sont nécessaires pour fournir du courant étant donné que les moteurs ont tendance à fonctionner à hauts courants contrairement aux microprocesseurs qui fonctionnent à des courants plus bas.[6]

Le DRI0044 est un moteur driver déjà installé sur l'Arduino Nano Every qui a été donné dans le kit. Ce moteur driver permet de gérer le voltage donné au moteur à courant continu. C'est ce qui permet au robot d'avancer, de tourner ou encore de reculer.

4.7 Roues

4.7.1 Roue sphérique

Comme expliqué dans [24], les roues sphériques sont constituées d'une sphère, voir figure 4.11, qui agira comme une roue. N'ayant pas d'axe de rotation propre comme les roues normales, celle-ci sont donc omnidirectionnels et permettent donc d'avoir une grande liberté de mouvement.



FIGURE 4.11: Roue Sphérique FIT0007
[10]

4.7.2 Roue motrice

Le kit des moteurs à courant continu était fourni avec des roues standards qui seront donc les roues motrices du robot, voir figure 4.1.

Ces roues ont un diamètre extérieur de 65mm et une largeur de 26mm.

4.8 Conclusion

Pour conclure ce chapitre, l'équipe a décidé d'utiliser des composants nécessaires aussi bien pour le suivi de ligne mais également pour tester et améliorer le prototype. Il est important de vérifier que le voltage demandé par les composants n'excède pas ceux envoyés par l'Arduino. Si cela arrive, le robot ne pourra pas fonctionner et il est possible que l'Arduino brûle sous l'effet d'une trop grande arrivée de tension.

Pour pouvoir placer tout les capteurs, moteurs et la pince, le robot a été construit sur deux étages. Cette disposition permet une plus grande marge de manœuvre concernant l'emplacement des différents composants du robot, voir le chapitre Les composants.

5.1 Design du châssis

Cette section est dédiée au design du robot, les choix des dimensions et le placement des composants.

5.1.1 Premier prototype

Le châssis du robot a fortement évolué au cours des recherches et améliorations apportées par le groupe. En effet, plus de fonctionnalités ont été rajoutées au fur et à mesure, mais la taille du robot n'étant pas infinie, il a fallu fixer une taille définitive. C'est pour cela que le groupe a choisi de faire un design sur deux étages, afin de pouvoir rajouter ou inter-changer des composants sans changer la taille finale, comme montré sur la photo du premier prototype, voir figure 5.1.

Les dimensions du robot ont été déterminées via une planche de plexiglas de 4mm de hauteur

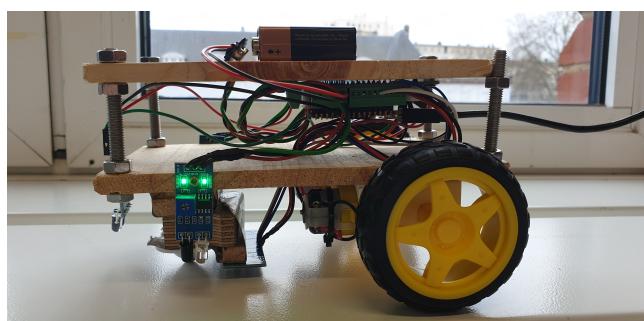


FIGURE 5.1: Photo du premier prototype vue de profil

que le groupe a pu récupérer. Ainsi, un étage fait 17cm de long sur 10cm de large et 4mm de hauteur.

Finalement, le premier prototype a été construit avec des plaques de bois car c'est un matériau facile à travailler et permet d'avoir une plus grande marge de manœuvre. Mais, le bois est plus épais, ce qui a fait passer la hauteur d'une plaque d'un étage de 4mm à 8mm.

L'équipe s'est penchée sur un modèle de robot à trois roues avec deux motrices. Celles-ci se trouvent à l'arrière du robot tandis qu'une roue sphérique, voir figure 4.11 et le tableau A.6, a été placé à l'avant pour offrir une bonne stabilité. Des supports ont été imprimés pour les moteurs des roues motrices afin de pouvoir les attacher sur le robot. 5.2.

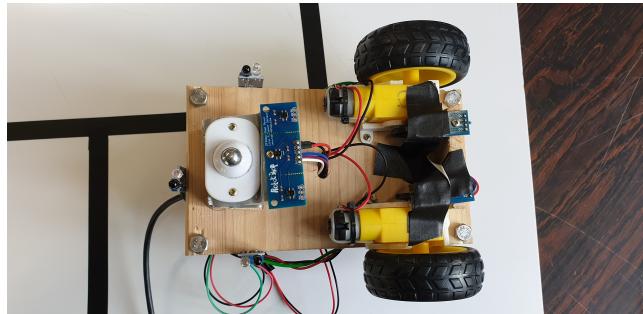


FIGURE 5.2: Photo du premier prototype vue du dessous

Après avoir fait les tests et simulations, voir le chapitre Modélisation, il fallait placer trois modules de capteur infrarouge, voir 4.2 et un réseau de capteurs de réflectance 4.4. Le premier module de capteur infrarouge devait être placé le plus à l'avant et le plus centré possible afin de pouvoir détecter tout changement dans la direction des marquages au sol. Les trois autres capteurs devaient former une ligne droite parallèle à l'axe de la largeur du robot. Les deux modules de capteur infrarouge ont été placés sur les côtés en vue de profil du robot et le réseau de capteur de réflectance au centre, juste après la roue sphérique. Idéalement, la ligne que forment les deux modules de capteur infrarouge et le réseau de capteur de réflectance aurait dû se trouver sur l'axe des deux roues motrices pour faciliter les calculs.

La pince a été placée tout à l'avant du robot étant donné que le prototype ne stocke pas d'objet.

En ce qui concerne l'Arduino, celui-ci prend beaucoup de place et étant léger, le groupe a pris la décision de le mettre au deuxième étage. Afin que l'on puisse connecter tous les composants qui doivent y être reliés, la carte a été attachée à l'envers, l'exposant donc au premier étage.

La place libre sur le deuxième étage a été réservée pour placer la batterie.

5.1.2 Deuxième prototype

Le second prototype est resté presque identique au premier, mais quelques améliorations lui furent apportées comme montré sur les figures 5.3 et 5.4. Tout d'abord, le châssis était trop haut. Les roues furent alors placées sur la face supérieure du premier étage. Par après, le capteur de réflectance QRE1113RC, voir figure 4.4 a été remplacé par le capteur de réflectance QTR-RC8, voir figure 4.5. Grâce à ce changement, le robot n'avait plus besoin de son capteur infrarouge avant, ne laissant que les deux autres sur les côtés droit et gauche pour détecter les branchements et lignes d'arrêts. Ensuite, la pince fut installée sur le châssis.

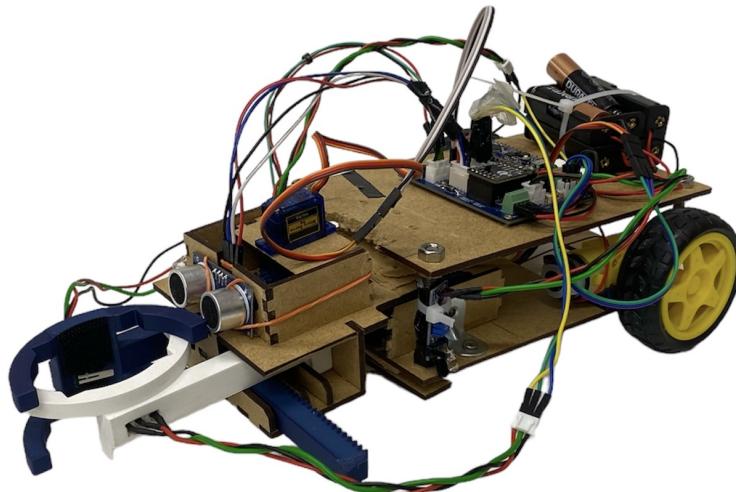


FIGURE 5.3: Vue en perspective du second prototype développé

Finalement, la batterie 9V a été remplacée par six piles de 1,5V.

5.2 Design de la pince

Afin de soulever et de déplacer les objets, l'utilisation d'une pince appropriée est indispensable. Il existe plusieurs options de pinces envisageables pour soulever d'objets, et dans un premier temps, une pince à deux doigts a été envisagée, mais des complications ont rapidement fait surface. Il n'était pas possible de placer des capteurs directement sur la pince pour reconnaître les objets et elle avait des difficultés à les agripper. [4]

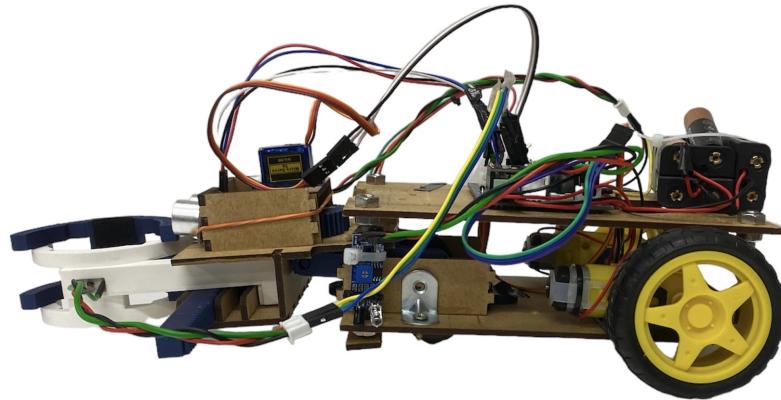


FIGURE 5.4: Vue de côté du second prototype développé

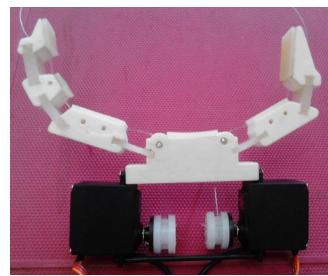


FIGURE 5.5: Une pince avec 3 articulations [14]

D'autres idées de pinces furent explorées au cours du deuxième quadrimestre. Ci-dessus, un exemple de pince à trois articulations similaire à une pince sans articulation, voir figure 5.5 et 5.6.

L'option la plus simple était de créer un modèle de pince et de l'imprimer afin qu'il convienne le mieux possible à nos besoins.

Le principe de la nouvelle pince consiste en un système de glissement complexe entre 2 bras. Avec l'aide d'engrenage, si le capteur présent sur la pince détecte un objet en face de lui, les bras se resserreront jusqu'à ce que l'objet soit parfaitement calé puis ils le soulèveront pour pouvoir le déplacer. L'avantage de cette nouvelle pince par rapport à l'ancienne est qu'il est possible de reconnaître directement l'objet agrippé lors du contact avec celui-ci grâce au design de la pince. La partie attrapant l'objet est modélisé de sorte qu'elle épouse les formes de diverse façon pour pouvoir directement les différencier avec le positionnement des bras.

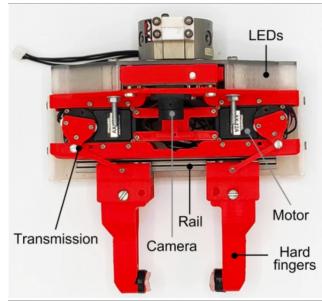


FIGURE 5.6: Autre pince possible à imprimer avec une caméra [20]

5.2.1 Équation de statique

D'après [4], les équations suivantes seront utiles pour calculer le moment de force nécessaire afin d'empêcher tout mouvement de l'objet dans la pince, tant pendant l'immobilisation que durant le déplacement du robot. Ce moment variera pour chaque objet, permettant ainsi de déterminer l'objet correct en fonction du moment qui lui est appliqué.

Le poids de l'objet est défini par l'équation :

$$W = m_0 g$$

Par la suite, une autre relation que l'on peut exprimer est la loi de Coulomb, établissant un lien entre la force de frottement et la force normale à l'aide d'un coefficient de frottement :

$$F_f = \mu N$$

- F_f est la force de frottement entre l'objet et la pince
- N est la force normale de l'objet sur la pince ou de la pince sur l'objet (ici, nous prendrons le premier cas)
- μ est le coefficient de frottement et dépend fortement du type du matériau

Il existe un lien entre le poids de l'objet et la force de frottement exercée sur cet objet qui est :

$$F_f = \frac{W}{2}$$

Grâce à cette équation de proportionnalité, il devient possible de calculer rapidement tant la force de frottement que la force normale qui s'exerce sur nos objets.

Il est nécessaire d'effectuer une résolution sur l'axe normale à surface rugueuse de la pince afin de finaliser notre système d'équations pour la statique de l'objet. L'équation fournie par [4] est la

suivante :

$$-F_1 \cos(\alpha + \beta) - F_2 \cos\phi + N = 0$$

- α , β et ϕ dépendant de la position des joints.
- F_1 et F_2 sont la décomposition de la force F sur l'axe X et Y.

Grâce à ces équations, il devient envisageable de déterminer le temps requis pour soulever des objets d'un diamètre de 4 cm. En se référant au tableau ci-dessous 5.7, on constate que le travail nécessaire pour soulever ces objets dépasse 3,02 Nm.

Object dimension in mm	Torque (T_G) in N m
03	2.20
20	2.88
30	2.95
40	3.02
50	3.11
60	3.18
70	3.12
80	3.05
90	2.67
100	2.39

FIGURE 5.7: Variation du moment appliqué par la pince en fonction des dimensions d'objet rectangulaire [4]

5.2.2 Méthode adoptée et reconnaissance des objets

Afin de satisfaire le cahier des charges et assurer l'exécution de la tâche par le robot, il est impératif qu'il puisse reconnaître les objets. Il a alors été décidé d'utiliser des capteurs de pression (micro switch) ainsi que des capteurs infrarouges et ultrasons.

Une pince dans laquelle tous ces capteurs peuvent être placés a ainsi été conçue. Le capteur à ultrasons permet de détecter à distance s'il y a un objet ou non sur la branche, il doit donc se trouver vers l'avant au centre. La pince se chargera de différencier les objets.

5.2.3 Mécanisme pour lever la pince

Différents mécanismes ont été pensés pour soulever la pince telle qu'utiliser une charnière et un fil pour mettre en rotation le bout de la pince. Finalement, il a été décidé d'utiliser un modèle tel que celui montré à la figure 5.9.

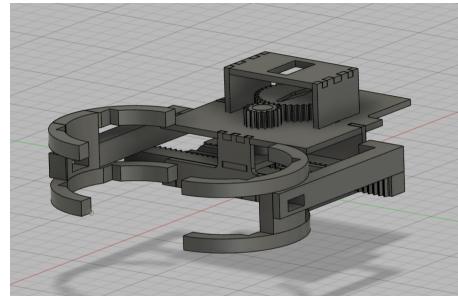


FIGURE 5.8: Vue en perspective de la pince utilisée

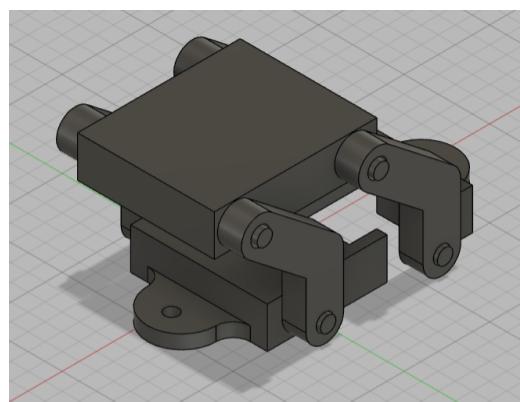


FIGURE 5.9: Mécanisme pour lever la pince

6

Réalisation dans Arduino

Après avoir réalisé le modèle, les données obtenues ont été utilisées pour construire le code du suivi de ligne pour le robot réel. Le robot utilise une carte Arduino Nano. Les actions du robot ont donc dû être traduites en code C++.

6.1 Pilotage

Une fonction de suivi de ligne est appelée en boucle tant que les capteurs ne renvoient pas de signal d'arrêt ni de branchement.

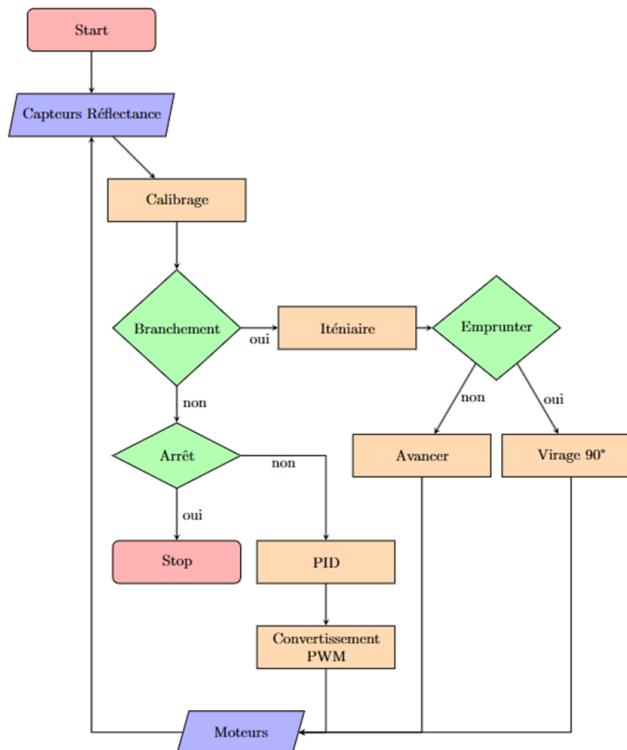


FIGURE 6.1: Organigramme du code de pilotage

Dans la figure 6.1, On voit apparaître plusieurs décisions et procédures qui doivent être appli-

quées jusqu'à ce que la condition d'arrêt soit satisfaite. Le robot doit d'abord calibrer les capteurs de manière que les valeurs utilisées dans la fonction erreur pour les couleurs noir et blanc soient les plus proches que possible de la réalité.

6.1.1 Calibrage

En testant le capteur de suivi de ligne analogique, voir figure 4.5, il a été constaté que la couleur noire correspond à des valeurs de renvoi plus élevées des capteurs de réflectance, ainsi que la couleur blanche correspond à des valeurs plus basses.

Ainsi la fonction calibrage se résume à trouver la valeur la plus haute renvoyée par les capteurs lui attribuant la valeur associée à la couleur noire et inversement pour la couleur blanche.

6.1.2 L'algorithme PID

Le code du PID a été basé sur l'exemple donné dans l'article "PID controller implementation using Arduino" trouvé sur le site "microcontrollerslab.com" [16]. La première étape de l'algorithme PID est la fonction d'erreur. Celle-ci a été modifiée par rapport à celle utilisée dans le modèle, voir figure A.13.

Il a été remarqué par des tests que le capteur suiveur de ligne à série de trois, voir figure 4.4, n'était pas apte à appliquer le régulateur PID. Comme avec ce capteur on ne peut pas déduire le degré de déviation de la ligne (soit le capteur voit la ligne noir soit il ne la voit pas), il était nécessaire d'utiliser le capteur série de huit 4.5 et donc de modifier la fonction d'erreur.

La fonction de conversion ω vers PWM est identique à celle du modèle. Pour visualiser sa réalisation en code C++, voir annexe A.14.

6.1.3 Détection de Branchement de ligne et Virage 90°

Le code diffère du modèle dans sa manière de gérer les virages à 90°(voir figure : A.8). En effet il n'était pas nécessaire de brancher un capteur digital à l'avant. Le code prend en compte le fait qu'il y ait un virage avant le premier branchement. Comme la carte n'est pas sujette à changer, nous avons décidé que cette solution est suffisante.

6.2 Pince et Reconnaissance Objet

Comme énoncé au point 5.2.2, la pince se charge de la reconnaissance objet et comportera 3 capteurs : un micro-switch, un capteur ultrason et un capteur infrarouge.

En effet, pour le parallélépipède rectangle, la finition du bras a été fait de sorte à ce que l'objet s'emboîte sans activer de capteur de toucher (voir A.15) mais il allumera un capteur infrarouge situé où la pince s'arrête lorsqu'elle tient cet objet. Pour le cylindre, la pince est munie d'un microswitch situé entre les deux crochets pour détecter uniquement la forme circulaire du cylindre. Finalement, pour reconnaître l'altère, la pince sera nettement plus refermée que pour les 2 autres objets et aucun capteur ne sera activé.

6.3 Détection d'Objets et Placement du Robot

Pour pouvoir reconnaître et trier les objets, une fonction scan a dû être implémentée. Pour cela, le capteur ultrason 4.7 a été utilisé. L'algorithme de scan est le suivant.

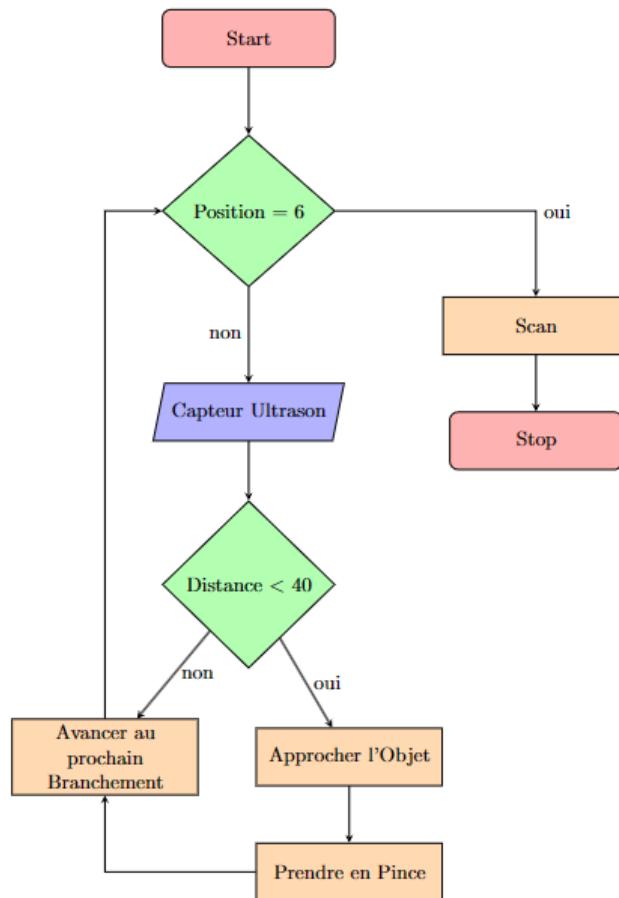


FIGURE 6.2: Organigramme de la fonction Scan

Le robot va ainsi approcher l'objet s'il en détecte un et ensuite le prendre en pince pour déterminer le type d'objet pour chaque maison. Le procédé scan qui sera effectué au dernier branchement consiste à détecter la distance avec le capteur ultrason et ensuite s'il y a un objet, l'approcher et prendre en pince.

Pour approcher les objets, il suffit de faire avancer le robot jusqu'à ce que la distance renvoyée par le capteur ultrason soit inférieure à une distance seuil.

Une fois le Scan de la carte terminé, l'algorithme de tri donnera l'ordre dans lequel le robot doit visiter les maisons, ainsi triant les objets.

6.4 Tests et Validation du Modèle

Après avoir effectué des tests sur le prototype, il s'est avéré que les valeurs des coefficients PID obtenues par le modèle n'étaient pas bonnes pour le pilotage du prototype. En effet, le modèle ne prenait pas en compte la possibilité de glissement des roues, les imprécisions des capteurs, la nécessité de la fonction calibrage, etc. Nous avons donc procédé par tests essai-erreurs pour trouver des valeurs qui permettent un pilotage stable. Il a été également remarqué, comme cité à la section 6.1.2, que trois capteurs infrarouges analogiques ne suffisent pas pour la suivi de ligne alors que le modèle en utilise autant. Nous pensons que ceci est dû justement aux imprécisions dans les capteurs ainsi que l'absence d'une transition continue à l'interface de la ligne et du sol blanc. Cela ne gênait pas le pilotage dans le modèle car les capteurs modélisés renvoient des valeurs différentes en fonction de sa position au dessus de la ligne noire alors que les capteurs réels ont un fonctionnement plus proche des capteurs digitaux (une intervalle de valeurs associées à la ligne noire et une autre pour le sol blanc).

Gestion de projet

7.1 Gestion du projet

Plusieurs outils ont été utilisés concernant la gestion du projet, mais également du groupe.

- Le diagramme de Gantt qui permet d'organiser les tâches et avoir aussi une vue d'ensemble.
- GitLab est par ailleurs pratique pour partager des codes et des fichiers.
- Les Status Report complets avec les explications des tâches réalisées par les membres du groupe.

7.1.1 Risques et Problèmes

Tout au long du projet, nous avons dû faire face à un certain nombre de risques à mitiger et de problèmes à régler. Nous avons constaté un certain nombre de risques souvent portant sur le manque ou la disponibilité du matériel ainsi que la défaillance de plusieurs composants et la durée du projet. Nous avons pu mitiger certains de ces risques soit par commande de composants supplémentaires, soit en s'assurant de réserver le matériel requis afin d'avancer sur le projet. Nous avons également constaté un nombre de problèmes souvent associé au design et au mauvais fonctionnement du code. La résolution de ces problèmes se résume souvent à prévoir plus d'heures de travail pour les sections problématiques.

7.2 SWOT

Le SWOT est un outil permettant de mettre en évidence les forces et faiblesses de l'équipe afin d'améliorer la gestion du temps et de l'équipe. Ci-dessous le SWOT du Q1.

Forces :	Faiblesses :
1) Belle ambiance d'équipe 2) Remise en question 3) Bonne communication avec la supervisrice 4) Bonne communication dans le groupe 5) Considération de plusieurs manières de réaliser les choses 6) Enthousiasme par rapport au projet	1) Mauvais départ pour l'organisation 2) Mauvaise gestion de Gitlab 3) Non-respect de l'attribution des tâches. 4) Mauvaise identification des problèmes
Opportunités :	Menaces :
1)Personnes compétentes à disposition 2) Accès aux espaces projets 3) Accès au Fablab et à Cible+ 4) Mise à disposition de composants	1) Retards de livraison 2) Mauvaise identification du scope 3) Gestion des autres cours 4) Défaillance de composants.

Voici les résultats de l'analyse SWOT menée à l'entame du second quadrimestre.

Forces :	Faiblesses :
1) Expérience du premier quadrimestre 2) Communication efficace au sein du groupe 3) Motivation du groupe 4) Vision claire des tâches à réaliser	1) Mauvaise estimation du temps nécessaire pour réaliser les tâches 2) Communication extérieure (présentation orale, rapport) 3) Manque d'alarme pour les problèmes
Opportunités :	Menaces :
1)Plus de temps disponible durant la semaine 2) Matériel disponible 3) Ressources disponibles 4) Conseils extérieurs	1) Délais très courts 2) Forte demande du matériel 3) Absences 4) Imprévus de dernière minute

7.3 Budget du projet

Ci dessous, à la figure 7.2, la liste des composants nécessaire à la fabrication du robot. Il est à noter que certains composants étaient déjà fournis dans un kit de démarrage ou donnés/échangés avec d'autres groupes et n'ont donc pas dû être achetés. Certains furent aussi vendus en trop grande quantité par rapport à ce qui était réellement nécessaire. Le prix total du projet dépensé

Composants	Prix	nombre requis
Moteur DC (4 pièces)	11,99€ 2	4
tige filté (1m)	1,79€	1
écrou (30 pièces)	3,99€	8
Pile Duracel AA 1,5V (8 pièces)	14,59€	6
équerres (4 pièces)	1.39€	2
Roue sphérique (16 pièces)	10,99€	1
micro servo 9g (4 pièces)	11,99€	2
vis 3,5x30mm (75 pièces)	7,99€	4
vis 3,5x12mm (100 pièces)	6,49€	4
support piles (10 pièces)	10,99€	3
jumper wire (120 pièces)	13,99€	18
Total	96,19€	53

FIGURE 7.1: Tableau récapitulatif des prix des composants acheté par le groupe

par le groupe est inférieur à 100€ comme imposé par le budget alloué au projet 7.1.

Composants	Prix	nombre requis
Moteur DC (4 pièces)	11,99€ 2	4
Capteur de réflectance QTR-RC8	11,47€	1
Capteur ultrasons (9 pièces)	3,16€	1
microswitch DP1	2,65€	1
tige filtré (1m)	1,79€	1
écrou (30 pièces)	3,99€	8
pille Duracel AA 1,5V (8 pièces)	14,59€	6
équerres (4 pièces)	1,39€	2
Plaque en bois contre-plaquée 3mm	7€	1
Roue sphérique (16 pièces)	10,99€	1
micro servo 9g (4 pièces)	11,99€	2
vis 3,5x30mm (75 pièces)	7,99€	4
vis 3,5x12mm (100 pièces)	6,49€	4
3x Capteur infrarouge digitaux	4,77€ (1.59€/pièce)	3
Arduino nano every	12,50€	1
support piles (10 pièces)	10,99€	3
Motor driver DRI0044	4,19€	1
jumper wire (120 pièces)	13,99€	18
impression 3D	6,38	NA
Total	148,31€	60

FIGURE 7.2: Tableau récapitulatif des prix des composants pour la construction du robot

8

Conclusion

Pour conclure notre rapport, il est important de faire un bilan du projet mais également un bilan de la gestion du groupe.

8.1 Bilan du projet

8.1.1 Premier quadrimestre

Le projet a démarré assez lentement, un retard s'est créé et amplifié d'entrée de jeu en raison d'une répartition inefficace des tâches et de mauvaises estimations de la priorité et du temps nécessaire à leur réalisation. Des progrès rapides ont été réalisés au niveau de la modélisation informatique de la régulation PID, ce qui a permis une meilleure compréhension des concepts à manipuler et une accélération de la mise en pratique de ceux-ci. En effet, la modélisation et les simulations menées en amont ont permis de construire un premier prototype équipé de capteurs nécessaires à la réalisation de notre premier objectif, à savoir suivre une ligne noire. En parallèle, la stratégie de reconnaissance des objets a été étudiée et réfléchie afin d'entamer la réalisation de ces objectifs dès le commencement du second quadrimestre. Bien que le quadrimestre ait commencé assez lentement, le groupe a pris conscience de ses lacunes et a pu corriger le tir en ayant un prototype sur lequel expérimenter avant la session d'examens de janvier marquant une trêve dans le projet.

8.1.2 Second quadrimestre

Une base solide de connaissances et d'expérience ayant été bâtie au premier quadrimestre, le projet a pu reprendre avec une vision claire des objectifs à atteindre lors des 6 semaines restantes. Un peaufinage du contrôle et de la régulation a été effectué suite à la construction d'un nouveau prototype "remis au propre" basé sur l'apprentissage apporté par le premier jet. La stratégie de tri

a été codée en Arduino afin que le robot puisse l'appliquer pour accomplir sa tâche. Cela en parallèle de la modélisation et la confection de mécanismes servant à attraper et soulever les objets. Le premier dispositif est une mâchoire dont la forme a été pensée pour épouser celle de n'importe quel objet prévu dans le cahier des charges, chaque objet déclenchant ou non certains capteurs de contact lorsqu'ils sont saisis par la pince. Ce dispositif permet donc à la fois d'attraper et reconnaître les objets. Les défis les plus conséquents de cette fin de projet furent sans le moindre doute la gestion du temps, mais aussi de faire fonctionner en symbiose tous les éléments du projet développés séparément.

8.2 Leçons apprises

Les leçons apprises ont été nombreuses durant le projet. Celle-ci ont permis au groupe d'avancer et de prendre des décisions importantes pour finir le projet dans les temps. Celles-ci peuvent aussi être résumées en quatre grands points de notre projets qui sont :

- Le modèle
- Le pilotage
- La pince
- Le design

8.2.1 Le modèle

Pour modéliser le robot, le groupe à utilisé Simulink du logiciel Matlab et ces différentes bibliothèques. Ceci a permit d'aider à la compréhension du fonctionnement d'un PID. Toutefois, le modèle n'est pas capable de représenter la réalité, ce qui a fortement retardé le groupe car trop de temps y a été consacré pour finalement obtenir des résultats pas entièrement concluant.

8.2.2 Le pilotage

Diriger le robot correctement sur la carte n'a pas été une tâche évidente. A force d'essai-erreurs, il a été très rapidement compris comment répartir la masse des différents composants sur le robot ainsi que de faire attention à la consommation de puissance de chacun des composants. Cela a aussi permis de mieux comprendre où placer les capteurs sur le robot, c'est-à-dire le plus à l'avant possible pour permettre à celui-ci de corriger les erreurs avant qu'il ne soit "trop tard". Mais tout les capteurs ne peuvent être utilisé avec la méthode de pilotage utilisée. En effet, les capteurs de

réflectance triple telle que le QRE1113RC, voir figure 4.4, ne peuvent s'utiliser seulement si il n'y a pas de gradient noir-blanc, ce qui a fortement retarder le groupe. Enfin, les tests essaie-erreurs ont été commencés très tard, ce qui n'a fait que s'accumuler à celui déjà existant.

8.2.3 La pince

Le groupe a appris très rapidement à réserver des plages horaires pour les équipements disponible au Fablab telle que les imprimantes 3D ou les découpeuses laser. Malgré son design fonctionnel, l'usage d'élastique ou d'autres composants passifs tels que des ressorts auraient pu être mieux explorés.

8.2.4 Le design

Conceptualiser le robot a permis au groupe de se familiariser avec le logiciel Fusion 360 et ces compléments. En effet, certains composants à géométrie 2D pouvaient être découpés à la découpeuse laser puis assemblés au lieu d'être imprimés en 3D, ce qui permettait de gagner du temps tout en étant plus écologique. Par contre, le design du châssis n'a globalement pas changer depuis sa première conceptualisation, rendant certaines tâches beaucoup plus compliquées.



Annexe

Ici, nous avons les différentes datasheet des composants et quelques images.

A.1 Datasheet du module de capteur infrarouge

Tableau de spécification du module de capteur infrarouge
Operating voltage : 5.0 V
Supply current : 20 mA
IR LEDs light emitting angle : 20-60 degree
Range : Up to 20cm
Adjustable Sensing range
Built-in Ambient Light Sensor
Dimensions : 31 mm x 15 mm

FIGURE A.1: Tableau de spécification du module de capteur infrarouge
[7]

A.2 Datasheet de réseau de capteur de réflectance QRE1113RC

Tableau de spécification du réseau de capteur de réflectance QRE1113RC
Operating voltage : 5.0 V (Nominal)
Interface type : 0.5V Analog x3
Supply current : 75 mA for the entire module
Pin definition : Out 1 (Right sensor), Out 2 (Left Sensor), Out 3 (Middle Sensor)
Sensing range : 3mm (0.125") to 6mm (0.25")
Optimal line thickness : black electric tape is ideal ($\approx 17mm$)

FIGURE A.2: Tableau de spécification du réseau de capteur de réflectance QRE1113RC
[25]

A.3 Dimensions du réseau de capteur de réflectance QRE1113RC

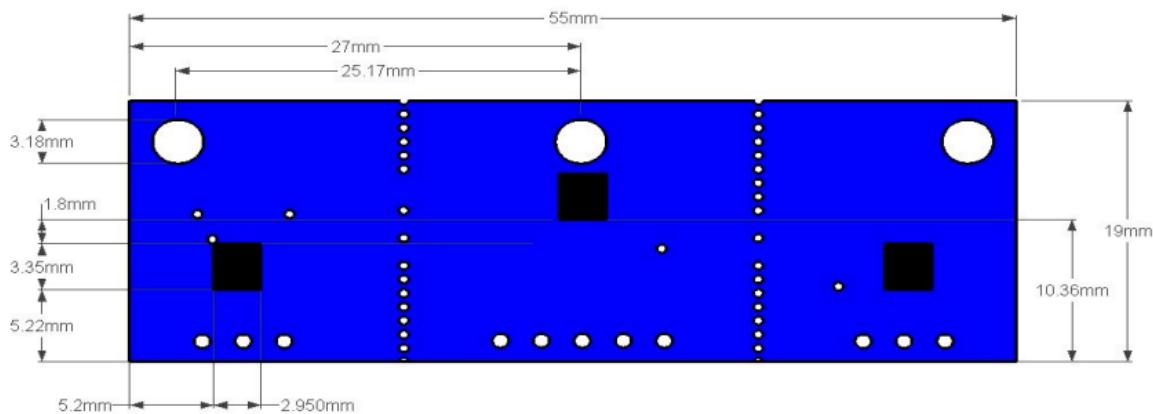


FIGURE A.3: dimensions du capteur QRE1113RC vue de haut
[25]

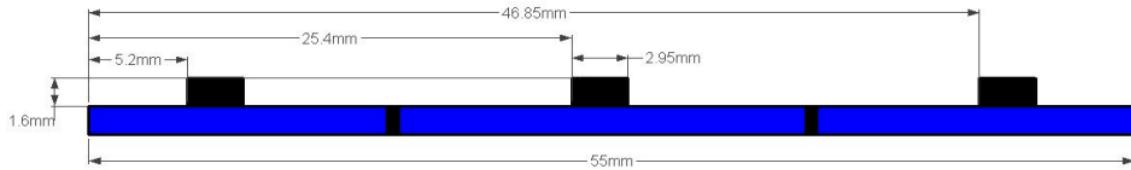


FIGURE A.4: dimensions du capteur QRE1113RC vue de profil
[25]

A.4 Datasheet du capteur Ultra-sons HC-SR04

Capteur Ultra-sons HC-SR04
Power Supply :+5V DC
Quiescent Current :<2mA
Working Current : 15mA
Effectual Angle : <15°
Ranging Distance : 2cm – 400 cm
Resolution : 0.3 cm
Measuring Angle : 30 degree
Trigger Input Pulse width : 10uS TTL pulse
Echo Output Signal : TTL pulse proportional to the distance range
Dimension : 45mm x 20mm x 15mm

FIGURE A.5: Tableau des données du capteur ultra-sons HC-SR04
[22]

A.5 Datasheet de la roue sphérique

Metal Ball Caster
Fixed hole : 4mm
Center Distance to Fixed holes : 40mm
Ball body height : 20mm
Ball protruding height : 4mm
Bearing ball number : 40
Maximum load : 15KG
Ball diameter : 15mm
Weight : 37g

FIGURE A.6: Tableau de la fiche technique de la roue Sphérique FIT0007
[10]

A.6 Fiche technique de l'Arduino Every

Arduino nano every
Microcontroller and ATMega4809
Operating Voltage and 5V
VIN min-MAX and 7-21V
DC Current per I/O Pin and 20 mA
DC Current for 3.3V Pin and 50 mA
Clock Speed and 20MHz
CPU Flash Memory and 48KB (ATMega4809)
SRAM and 6KB (ATMega4809)
EEPROM 256byte (ATMega4809) PWM Pins and 5 (D3, D5, D6, D9, D10)
UART and 1
SPI and 1
I2C and 1
Analog Input Pins and 8 (ADC 10 bit)
Analog Output Pins and Only through PWM (no DAC)
External Interrupts and all digital pins
LED_B UILTIN and 13
USB and Uses the ATSAMD11D14A (datasheet)
Length and 45 mm
Width and 18 mm
Weight and 5 gr (with headers)

FIGURE A.7: Tableau de la fiche technique de l'Arduino

A.7 Code pour le suivi de ligne

```
1 void recolteDonnees() {
```

```

1   digitalWrite(DIR_G, LOW); // sens de rotation des moteurs
2   digitalWrite(DIR_D, HIGH);
3   while (pwm < 256) {
4       nTicks = 0;
5       isrnticks = 0;
6       pwm++;
7       analogWrite(MoteurG, pwm);
8       analogWrite(MoteurD, pwm);
9       delay(PERIOD);
10      noInterrupts();
11      nTicks = isrnticks; // isrnticks est une variable globale qui
12      comptabilise le nombre de ticks vus par les encodeurs
13      interrupts();
14      vitesseAngulaire = (-1 * 2 * 3.14 * nTicks/ticksParRotation) /
15      PERIOD;
16      Serial.print(vitesseAngulaire);
17      Serial.print(',');
18      }
19      analogWrite(MoteurG, 0);
20      analogWrite(MoteurD, 0);
}

```

Listing A.1: Code test des moteurs

```

1 int e(){
2     int erreur = 0;
3     for(int i = - SENSORCOUNT/2 ; i < SENSORCOUNT/2; i++){
4         if(i < -1){
5             erreur += i * i *(BLANC - analogRead(CAPTEURS[i + SENSORCOUNT/2]));
6         }
7         else if(i == -1){
8             erreur += i * (NOIR - analogRead(CAPTEURS[i + SENSORCOUNT/2]));
9         }
10        else {
11            erreur -= (i + 1) * (i + 1) *(BLANC - analogRead(CAPTEURS[i +
12            SENSORCOUNT/2]));
13        }
14    }
15    return erreur;
}

```

Listing A.2: Function d'erreur

Comme indiqué dans le code chaque erreur est pondéré par sa distance du centre au carré, ceci s'est révélé être plus efficace par tests essaie-erreurs.

```

1 // Code PID Inspire de https://microcontrollerslab.com/pid-controller-
  implementation-using-arduino/
2 void PID(){
3     unsigned long tActuel = millis();
4     long delta_T = tActuel - tPrecedent;
5     if (delta_T >= T){
6         long erreur = e();
7         eTotal += erreur; // Terme Integrale
8         long delta_erreur = erreur - ePrecedent; // Terme Differentielle
9         action = toPWM(K_P*erreur + (K_I*T)*eTotal + (K_D/T)*delta_erreur);
10        ePrecedent = erreur;
11        tPrecedent = tActuel;
12    }
13 }
```

Listing A.3: Fonction PID

```

1 int* toPWM(long pid){
2     pwm[0] = abs(pid * 0.01 - pwm_min);
3     pwm[1] = pid * 0.01 + pwm_min;
4     for(int i = 0; i < 2; i++){
5         if(pwm[i] > pwm_max){
6             pwm[i] = pwm_max;
7         }
8     }
9     return pwm;
10 }
```

Listing A.4: Fonction de conversion ω vers PWM

A.8 Circuit pour les virages de 90°

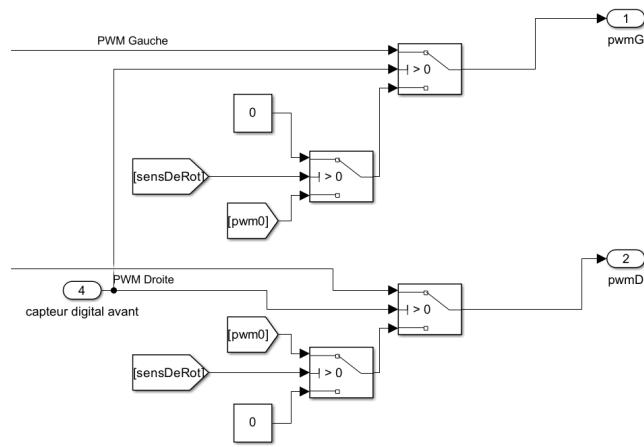


FIGURE A.8: Condition de virage 90° dans Simulink

A.9 Circuit de détection de branchement

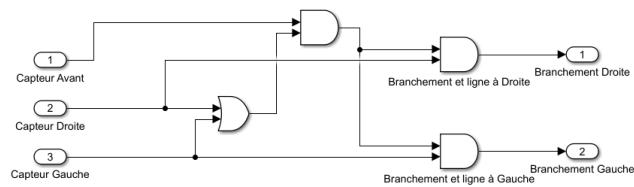


FIGURE A.9: Système de détection de branchement

A.10 Carte test pour le système de détection

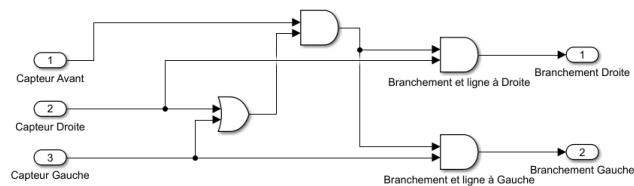


FIGURE A.10: Système de détection de branchement

A.11 Schéma de pince théorique

A.12 Schéma de pince théorique

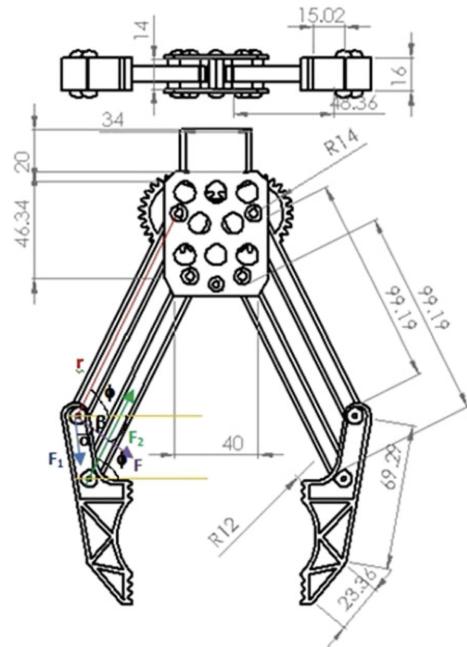


FIGURE A.11: Schéma de la pince [4]

A.13 Modélisation des Moteurs dans Simulink

Conversion pwm en vitesse angulaire des moteurs

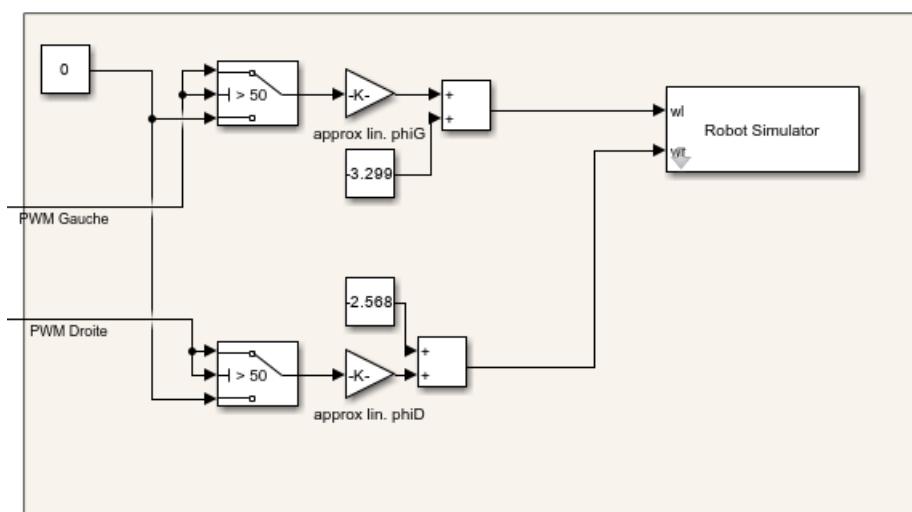


FIGURE A.12: Modélisation des moteurs dans Simulink

A.14 Calcul d'erreur dans Simulink

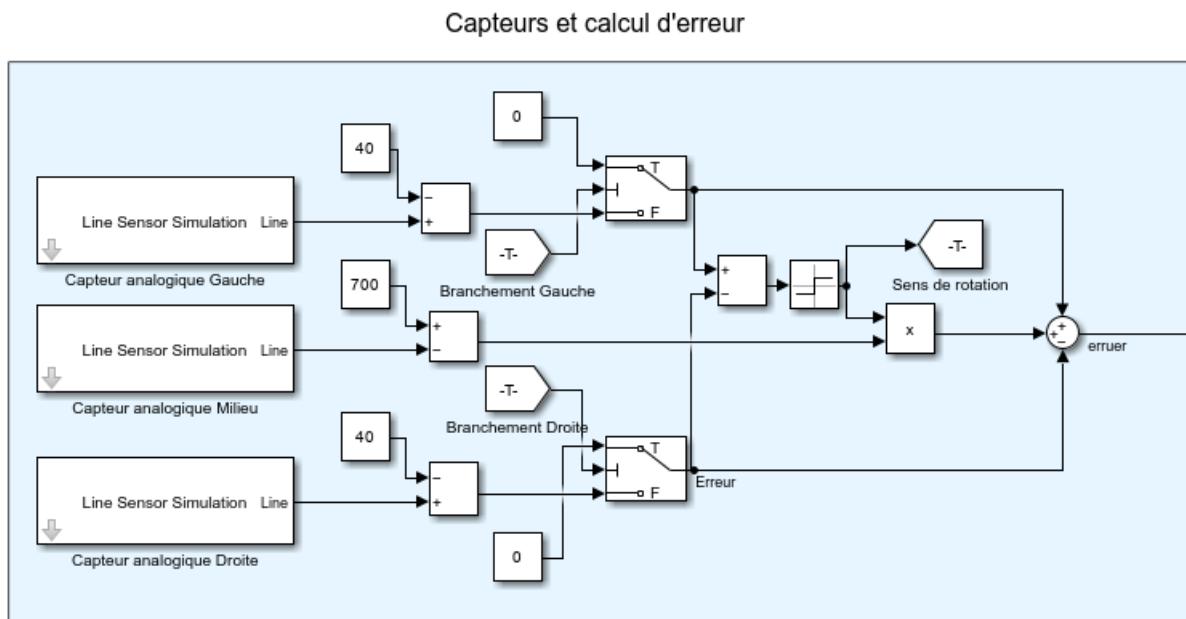


FIGURE A.13: Capteurs analogiques et calcul d'erreur dans Simulink

Le capteur au milieu voit son erreur multipliée par le signe de la différence entre l'erreur à gauche et celle à droite, car l'erreur mesurée par ce capteur ne peut pas tenir compte de quel côté de la ligne provient cette erreur.

A.15 Fonctionnement de la Conversion PID-PWM

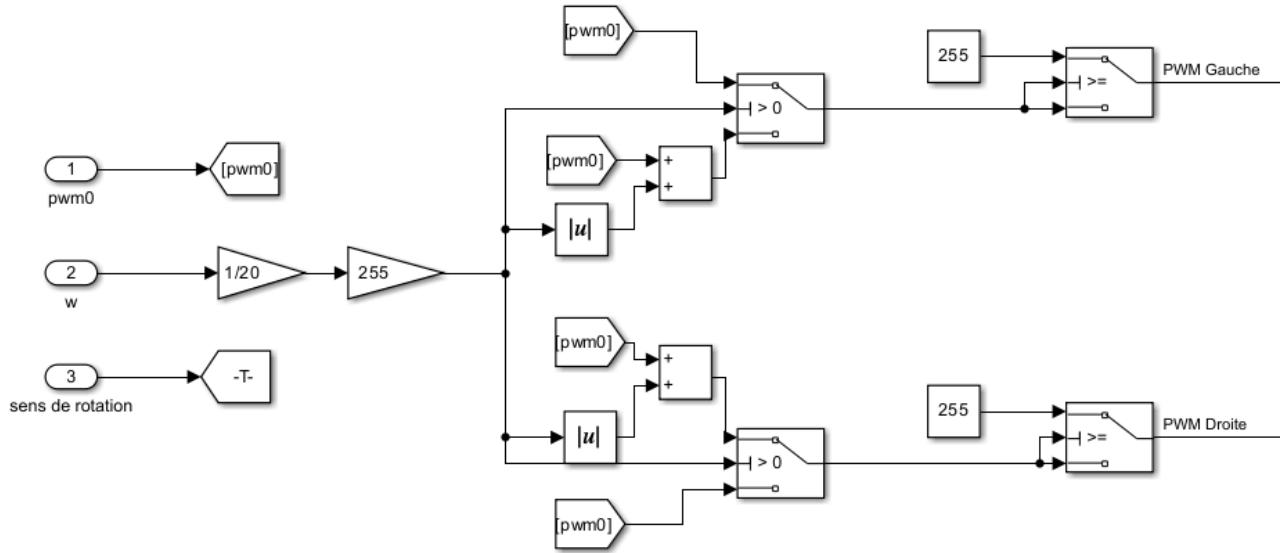


FIGURE A.14: Conversion ω vers PWM

La première étape dans ce Subsystem est de convertir ω en commande PWM, pour se faire ω est divisé par le omega maximal possible, c'est-à-dire la vitesse de rotation maximale que le robot pourrait effectuer. Ensuite, on le multiplie par la commande PWM maximale. Finalement, on additionne la valeur de PWM minimale pour faire tourner les roues. La vitesse de rotation maximale du robot a été déterminé à l'aide de la vitesse angulaire maximale observée lors des tests et vaut environ $50 \frac{rad}{s}$.

On a :

$$\omega_{max} = \frac{\phi_{max} \cdot R_{Roue}}{L} \implies \omega_{max} \approx 20 \frac{rad}{s} \quad (\text{A.1})$$

(avec ϕ_{max} étant la vitesse angulaire maximale des roues, $R_{Roue} = 3.25 \text{ cm et } L = 10 \text{ cm}$)

Ensuite, si $\omega > 0$ alors le robot doit accélérer la roue droite pour entraîner une rotation de sens anti-horlogique. Pour faciliter ceci et obtenir une différence de rotation plus grande entre les deux roues, le robot demande à la roue gauche de tourner à la vitesse minimale, et vice-versa pour $\omega < 0$.

Finalement, si $\text{PWM} > 255$ le robot envoie la commande de 255 simplement de cette manière on évite de demander un PWM qui n'appartient pas au domaine des valeurs PWM possibles.

A.16 Image de la pince

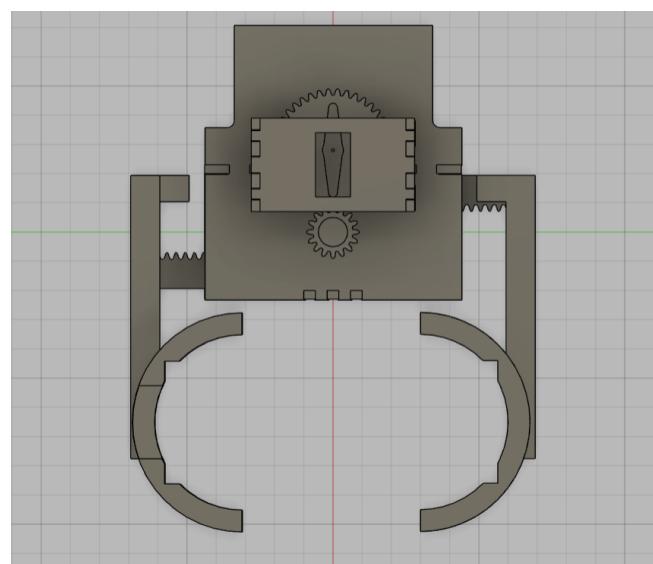


FIGURE A.15: Vue haute de la pince utilisée

Bibliographie

- [1] anglais. Vente de produits robotiques et électroniques. URL : <https://eu.robotshop.com/fr/products/pololu-qtr-8rc-infrared-sensor-array>.
- [2] Adnan AL TAHTAWI, Yoyo SOMANTRI et Erik HARITMAN. « Design and Implementation of PID Control-based FSM Algorithm on Line Following Robot ». In : *Jurnal Teknologi Rekayasa* 1 (jan. 2017), p. 23. DOI : 10.31544/jtera.v1.i1.2016.23-30.
- [3] Mazin ALWAN et al. « Design and Implementation of Line Follower Arduino Mobile Robot Using Matlab Simulink Toolbox ». en. In : *Iraqi Journal for Electrical and Electronic Engineering* 17.2 (déc. 2021), p. 11-16. ISSN : 2078-6069, 1814-5892. DOI : 10.37917/ijeee.17.2.2.
- [4] Nisha BHATT et Nathi Ram CHAUHAN. *Design of a two fingered friction gripper for a wheel mobile robot*. Jan. 2016. DOI : 10.1007/978-981-10-1023-1\{_}20. URL : https://doi.org/10.1007/978-981-10-1023-1_20.
- [5] Marco CECCARELLI. *Fundamentals of Mechanics of Robotic Manipulation*. Jan. 2022. DOI : 10.1007/978-3-030-90848-5. URL : <https://doi.org/10.1007/978-3-030-90848-5>.
- [6] RS COMPONENTS. *Drivers de moteur | RS*. URL : <https://fr.rs-online.com/web/c/semi-conducteurs/gestion-de-l-alimentation/drivers-de-moteur/>.
- [7] COMPONENTS101. *IR Sensor Module*. Août 2020. URL : <https://components101.com/sensors/ir-sensor-module>.
- [8] DFROBOT.COM. *Crash Sensor SKU SEN0138*. URL : https://wiki.dfrobot.com/CRASH_Sensor_SKU_SKU0138_.
- [9] DFROBOT.COM. *DFRobot DF9GMS 360 Degree Micro Servo (1.6Kg)*. Oct. 2023. URL : <https://www.dfrobot.com/product-1579.html>.
- [10] DFROBOT.COM. *Metal ball casters*. URL : <https://www.dfrobot.com/product-225.html>.

- [11] DFROBOT.COM. *SHARP GP2Y0A41SKOF Infrared Distance Sensor (4-30cm)*. URL : <https://www.dfrobot.com/product-1083.html>.
- [12] Auteur EXTERNE. *Le tri par insertion*. fr. text. Juill. 2014. URL : <https://zestedesavoir.com/tutoriels/262/le-tri-par-insertion/>.
- [13] GEBILDET. *Gebildet 4pcs DC3V-12VDC moteur adapté pour voiture jouet à quatre roues motrices/Corps robotique/Jouets d'avion+4pcs poue en plastique : Amazon.com.be : Jouets*. URL : https://www.amazon.com.be/Gebildet-DC3V-12V-motrices-robotique-Plastique/dp/B08D39MFN1/ref=sr_1_1_sspa?crid=17FKAN8TT8PIB&keywords=roue%2Brobot&qid=1701944684&sprefix=roue%2Brobot%2Caps%2C88&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&th=1.
- [14] Long KANG et al. « Design and implementation of a Multi-Function Gripper for grasping general objects ». In : *Applied sciences* 9.24 (déc. 2019), p. 5266. DOI : 10.3390/app9245266. URL : <https://doi.org/10.3390/app9245266>.
- [15] Lae Yin Mon KHIN KHIN SAW. *Design and Construction of Line Following Robot using Arduino*. Juin 2019. URL : <https://www.ijtsrd.com/papers/ijtsrd2320977.pdf>.
- [16] Microcontrollers LAB. *PID Controller implementation using Arduino*. Jan. 2020. URL : <https://microcontrollerlab.com/pid-controller-implementation-using-arduino/>.
- [17] MATHWORKS. *What is PID control?* URL : <https://nl.mathworks.com/discovery/pid-control.html>.
- [18] MATHWORKS. *Simulink Onramp*. URL : <https://nl.mathworks.com/products/simulink/getting-started.html>.
- [19] Student Competitions Team MATHWORKS. *Mobile Robotics training*. URL : <https://nl.mathworks.com/videos/series/student-competition-mobile-robotics-training.html>.
- [20] Pedro MATOS et Pedro NETO. « A 3D-printable modular robotic gripper ». In : *The International Journal of Advanced Manufacturing Technology* 126.1-2 (mars 2023), p. 845-855. DOI : 10.1007/s00170-023-11114-9. URL : <https://doi.org/10.1007/s00170-023-11114-9>.
- [21] Hidek OGAWA. *Moving Robot with Arm mechanism*. Jan. 2010. URL : <https://image-pubs.uspto.gov/dirsearch-public/print/downloadPdf/7645110>.

- [22] Rui SANTOS et Rui SANTOS. *Complete guide for ultrasonic sensor HC-SR04 with Arduino | Random nerd tutorials.* Oct. 2021. URL : <https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>.
- [23] *Server Busy.* URL : https://www.amazon.com/Dealikee-engrenages-magn%C3%A9tiques-voiture-plastique/dp/B09BDXBP6N/ref=pd_sbs_d_sccl_3_4/261-5530382-4373058?pd_rd_w=KYjBR&content_id=amzn1.sym.121ab853-f436-4222-b4e9-1df38e4e9cf9&pf_rd_r=ZXWGQTY6VK2W7VT81C93&pd_rd_wg=MsPT2&pd_rd_r=751addc6-e353-4c96-8606-3f9793feca58&pd_rd_i=B09BDXBP6N&psc=1.
- [24] Davide SCARAMUZZA SIEGWART ROLAND Illah R. NOURBAKHSH. *Introduction to autonomous mobile robots.* 2^e éd. The MIT Press, 2011. 456 p. URL : <https://doi.org/10.1016/j.artint.2005.10.007>.
- [25] RobotShop USA. *DFRobotShop Rover Line Follower Sensor.* URL : <https://www.robotshop.com/products/dfrobotshop-rover-line-follower-sensor>.