

Projet transverse : Robot mobile qui évite les obstacles

2023-2024

Ariaz, Dounia, Arnold (SIE)

Glen, Alice (GP)

TABLE DES MATIÈRES

1. Introduction du sujet.....	4
1.1. Présentation générale	4
1.2. Choix du sujet d'étude	4
1.3. Problématique	5
1.4. Séparation du problème.....	6
2. Organisation.....	7
2.1. Composition de l'équipe.....	7
2.2. Identification des tâches	7
2.3. Planification des tâches	7
3. Programmation	9
3.1. Prise en main des composants.....	9
3.2. Principe du capteur à ultrasons	10
3.3. Familiarisation avec le moteur shield.....	11
3.4. Programmation des différentes fonctions	12
3.4.1. Fonctions “Avancer/Reculer” et “Arrêter”	12
3.4.2. Intégration du capteur à ultrasons - Fonction “Déetecter”.....	12
3.4.3. Intégration du servomoteur	13
3.4.4. Comportement du robot face aux obstacles	13
3.4.5. Programme final.....	13
4.1. Contrôle de la vitesse des moteurs.....	15
4.1.1. Pulse Width Modulation (PWM).....	15
4.1.2. Importance du contrôle de vitesse	15
4.1.3. Relation PWM/vitesse de rotation des roues	16
4.1.3.1. Courbe théorique	16
4.1.3.2. Courbe pratique	17
4.1.3.2. Analyse	18
4.1.4. Corrections	18
4.2. Contrôle en virage.....	21
4.2.1. Problème	21
4.2.2. Pratique.....	23
5. CONCLUSION.....	24
5.1. Bilan du Projet.....	24
5.2. Leçons Apprises	24
BIBLIOGRAPHIE	26
ANNEXES.....	27

TABLE DES FIGURES

Figure 1: Diagramme bête-à-corne du projet.....	5
Figure 2 : Diagramme de Gantt du projet	8
Figure 3 : Composants du projet.....	9
Figure 4 : Capteur à ultrasons WPSE306.....	10
Figure 5 : Principe de fonctionnement du capteur à ultrasons.....	10
Figure 6 : Diagramme de bloc dans la datasheet du moteur Shield.....	11
Figure 7 : Modélisation du robot sur Proteus.....	11
Figure 8 : Trajectoire du robot pour différentes vitesses de rotation.....	14
Figure 9 : Simulation signal PWM avec rapport cyclique de 0.50505.....	15
Figure 10 : Fonction de transfert caractéristique du moteur	16
Figure 11 : PWM en fonction de la vitesse de rotation (rad/s).....	17
Figure 12 : Comparaison PWM en fonction de la vitesse de rotation (rad/s) dans le cas réel par rapport à la simulation	17
Figure 13 : Fonction de transfert de la vitesse angulaire en boucle ouverte	19
Figure 14 : Vitesse de rotation des roues (rad/s) en fonction du PWM	19
Figure 15 : Fonction de transfert de la vitesse angulaire en boucle fermée	20
Figure 16 : Vitesse de rotation des roues (rad/s) en fonction du PWM	20
Figure 17 : Virage du mobile pour esquive d'obstacle	21
Figure 18 : Rayon de courbure du virage en fonction du rapport k	22
Figure 19 : Durée du virage (s) en fonction de k.....	22

1. Introduction du sujet

1.1. Présentation générale

Dans le domaine de l'enseignement des ingénieurs en systèmes embarqués, l'écart entre les matières de l'automatique et du signal, et celles de la programmation embarquée, constitue un réel défi pédagogique pour les enseignants.

Afin de relever ce défi et de favoriser le développement de compétences transverses, l'initiative “Atelier de l'automatique numérique” a été lancée en 2021.

L'objectif de ce projet est de proposer une série de manipulations dans le domaine de l'automatique numérique, et à concevoir les prototypes de ces manipulations pour renouveler les projets et travaux pratiques de cet enseignement.

Actuellement, les projets et travaux pratiques utilisent Lego EV3 et Arduino pour asservir les systèmes embarqués et pour réaliser les manipulations souhaitées.

Ce rapport présente le cadre et les objectifs du projet transverse “Manipulations pédagogiques en automatique numérique”, ainsi que les différentes étapes de conception et de réalisation des manipulations. Chaque manipulation vise à fournir aux étudiants une expérience pratique complète, allant du design du système physique à l'implémentation du contrôleur en programme embarqué, en passant par la modélisation mathématique et la simulation.

1.2. Choix du sujet d'étude

Une fois le sujet pris en main, l'ensemble du groupe de travail s'est penché sur le choix d'un sujet pour notre projet. C'est ainsi que nous avons réalisé un brainstorming de groupe afin de partager les différentes idées de sujets et de voir quel sujet serait le plus cohérent pour notre projet.

Voici donc la liste des sujets que nous avons évoquée lors de cette première réunion :

- Le pendule inversé (projet transverse réalisé en 2022)
- Le robot mobile attiré par la lumière (projet transverse réalisé en 2023)
- L'asservissement d'un moteur en vitesse
- Le robot mobile suivant une ligne (déjà existant en Lego EV3)
- Le robot mobile qui évite les obstacles (déjà existant en Lego EV3)

Nous avons par la suite décidé de présenter cette liste de sujet à notre tuteur de projet, monsieur Asarin afin de pouvoir discuter avec lui des différentes manipulations qui composent chacun des sujets et de voir lesquelles seraient le plus adaptées.

C'est à la suite de ces deux réunions que l'ensemble du groupe a choisi le sujet du robot mobile qui évite les obstacles. Une fois la manipulation choisie et afin de pouvoir dessiner une problématique de projet, nous avons réalisé le diagramme bête-à-corne de cette manipulation.

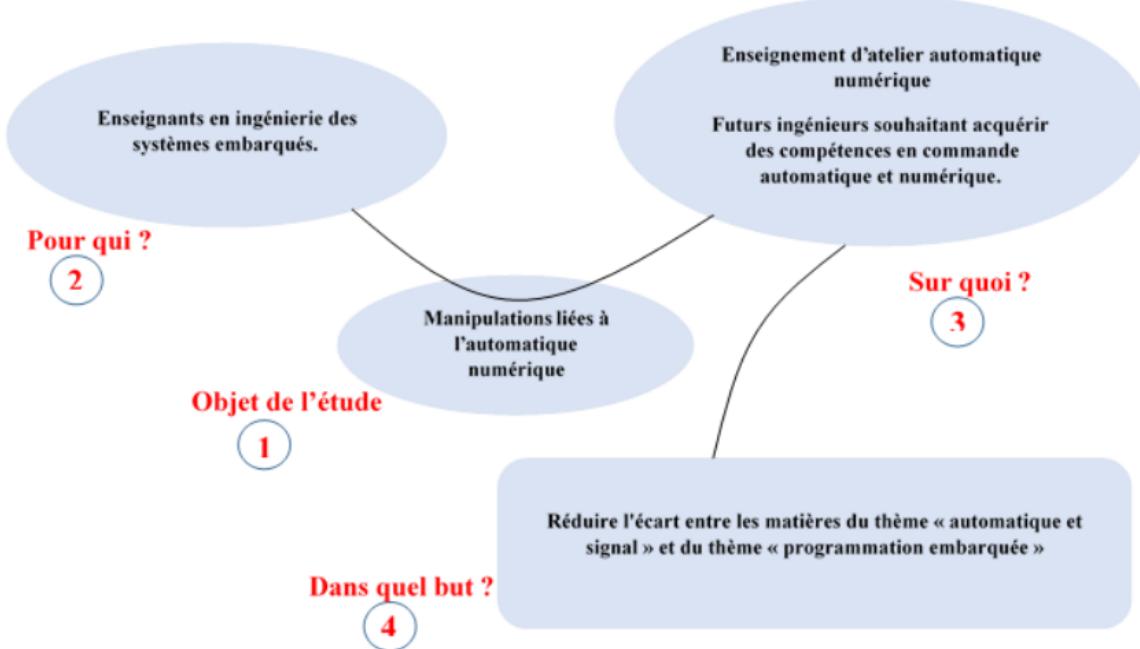


Figure 1: Diagramme bête-à-corne du projet

1.3. Problématique

À partir du diagramme bête-à-corne, nous avons extrait la problématique suivante :

Comment réaliser une étude mathématique approfondie pour concevoir un système de robot mobile évitant les obstacles, en intégrant les fonctionnalités essentielles telles que l'avancement et le recul dans l'espace, la détection précise des obstacles à une distance donnée, ainsi que la capacité à ajuster de manière autonome sa trajectoire pour assurer une navigation sécurisée et efficace ?

1.4. Séparation du problème

Une fois notre problématique définie, nous avons découpé notre projet en trois manipulations distinctes, chacune se concentrant sur un aspect spécifique du problème.

La première manipulation se focalise sur la possibilité d'avancer et de reculer dans l'espace. Celle-ci nécessite l'étude des mécanismes de propulsion du robot, ainsi que la programmation des commandes de mouvement en utilisant des microcontrôleurs comme

Arduino ou Raspberry Pi. Dans le cadre des contraintes imposées, nous devons concevoir cette manipulation de manière qu'elle puisse être réalisée dans une plage horaire de 2 à 10 heures par un groupe de 4 élèves, en utilisant un budget restreint pour l'achat de matériel.

La deuxième manipulation se concentre sur la détection précise des obstacles à une distance donnée. Cela implique l'utilisation de capteurs appropriés, ainsi que le développement de logiciels pour interpréter les données captées et prendre des décisions en conséquence. Cette étape nécessitera également une analyse mathématique du système pour optimiser la précision de la détection tout en respectant les contraintes de temps et de coût.

Enfin, la troisième manipulation abordera la capacité du robot à changer de trajectoire de manière autonome pour éviter les obstacles détectés. Cela exigera la mise en œuvre d'algorithmes de navigation et de contrôle sophistiqués, ainsi que des tests approfondis pour garantir la fiabilité et l'efficacité du système. Tout comme les manipulations précédentes, cette phase sera réalisée dans le cadre des contraintes de temps et de coût énoncées.

2. Organisation

2.1. Composition de l'équipe

Notre équipe de projet se compose de cinq étudiants dont deux étudiants de la filière Génie Physique et trois étudiants de la filière systèmes informatiques embarqués. En suivant les conseils de notre tuteur de projet monsieur Asarin, nous avons décidé de collaborer sur l'ensemble des tâches de notre projet afin d'avancer ensemble dans l'élaboration de ce robot mobile évitant les obstacles.

2.2. Identification des tâches

On a commencé par la prise en main des différentes composantes de notre robot mobile afin de mieux appréhender leurs fonctionnements et la portée de leurs fonctions.

Par la suite, nous avons réalisé les études théoriques de chacune des parties de nos différentes manipulations avant de réaliser concrètement les différentes manipulations.

Nous avons fini par faire des analyses et de tirer des conclusions sur chacune des parties de notre étude.

2.3. Planification des tâches

Ici, nous avons cherché à organiser de la meilleure des manières notre temps afin d'avancer efficacement dans notre projet et de réussir à respecter les délais.

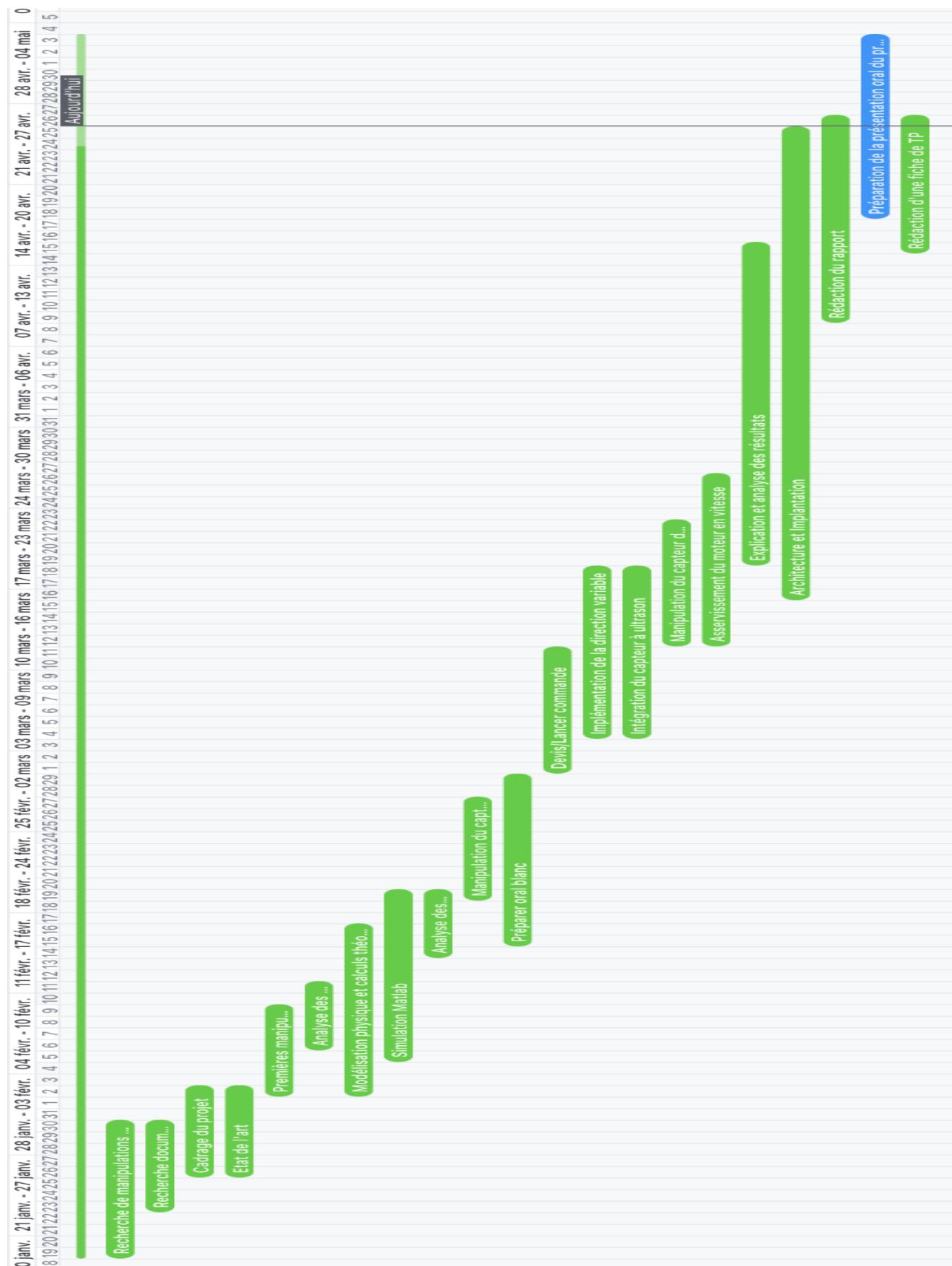


Figure 2 : Diagramme de Gantt du projet

3. Programmation

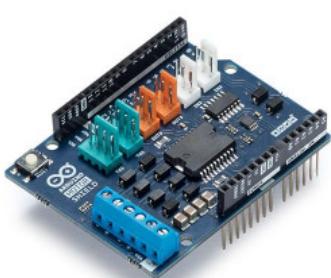
3.1. Prise en main des composants

Dans le cadre de notre projet, nous avons utilisé plusieurs composants clés pour construire notre robot. Voici une description détaillée de ces composants et de leur rôle dans notre système :

- Arduino Uno : Agissant comme le cerveau principal de notre robot, l'Arduino Uno est une carte microcontrôleur. Il est responsable de la gestion de toutes les opérations de notre robot, y compris la lecture des données des capteurs, le contrôle des moteurs, et la prise de décisions basées sur les algorithmes que nous avons programmés.
- Arduino Motor Shield : Cette extension pour Arduino Uno est utilisée pour contrôler les moteurs de notre robot. Le Motor Shield nous permet de contrôler la direction et la vitesse de chaque moteur indépendamment, ce qui est essentiel pour les mouvements précis et les manœuvres de notre robot.
- Moteurs 3 V à 6 V: Ces moteurs fournissent la puissance nécessaire pour le mouvement de notre robot. Ils sont contrôlés par le Motor Shield Arduino, qui ajuste leur vitesse et leur direction en fonction des instructions de l'Arduino Uno.
- Capteur à ultrasons WPSE306: Ce capteur agit comme les yeux de notre robot, lui permettant de détecter et d'éviter les obstacles. Il émet des ondes ultrasoniques et mesure le temps qu'il faut pour que l'écho revienne, ce qui nous permet de calculer la distance entre le robot et l'obstacle.
- Servomoteur : Le servomoteur est un type de moteur qui peut être dirigé pour se déplacer à un angle spécifique. Dans notre projet, nous avons utilisé un servomoteur pour contrôler la direction du capteur à ultrasons.



Carte Arduino Uno



Arduino Motor Shield



Moteur 3V-6V Arduino

Capteur à ultrasons
WPSE306

Servomoteur

Figure 3 : Composants du projet

Chacun de ces composants est essentiel au bon fonctionnement de notre robot. La compréhension et la maîtrise de chacun ont été des éléments clés pour le succès de notre projet. Nous avons appris à les utiliser de manière efficace, à résoudre les éventuels problèmes qu'ils pourraient poser, et les intégrer dans le système voulu.

3.2. Principe du capteur à ultrasons

On utilise le capteur à ultrasons WPSE306 afin d'évaluer la distance à un obstacle.

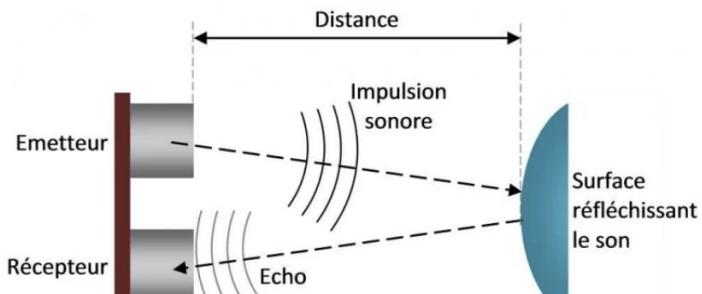


Figure 4 : Capteur à ultrasons WPSE306

Le fonctionnement du capteur à ultrason est la suivante :

- Émission d'un signal sonore très court (~ 8 oscillations à la fréquence 40 kHz)
- La sortie "trigger" passe à l'état HAUT à l'émission du signal
- La sortie "écho" passe à l'état HAUT à la réception du signal
- Envoi par l'Arduino d'une courte impulsion à l'entrée du capteur

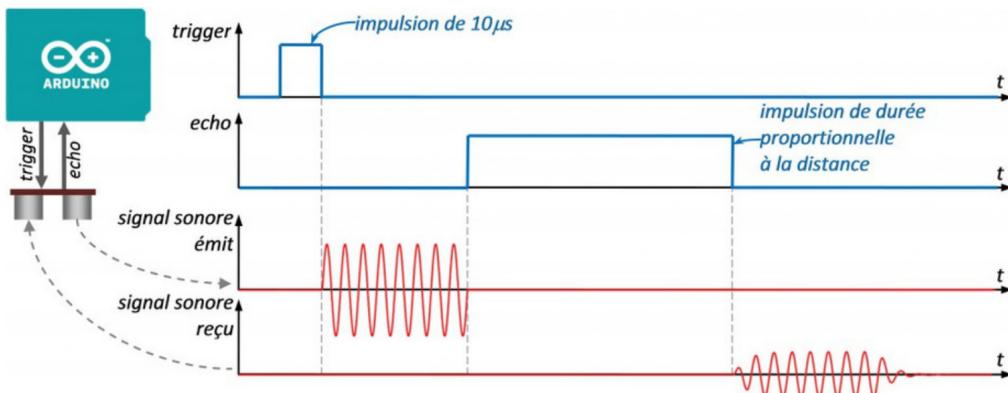


Figure 5 : Principe de fonctionnement du capteur à ultrasons

3.3. Familiarisation avec le moteur Shield

Dans le but de nous familiariser avec le moteur Shield, nous avons modélisé les quatre moteurs sur le logiciel Proteus à partir du diagramme de bloc présent dans la datasheet du moteur Shield :

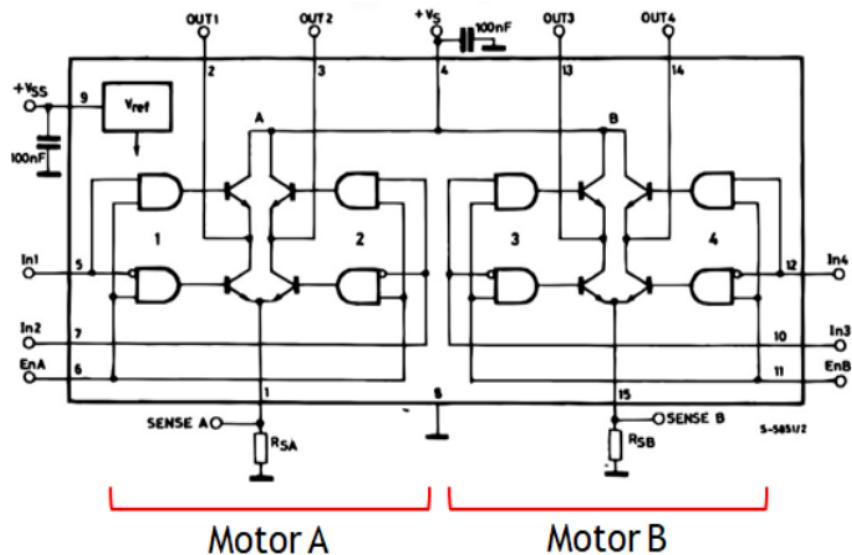


Figure 6 : Diagramme de bloc dans la datasheet du moteur Shield

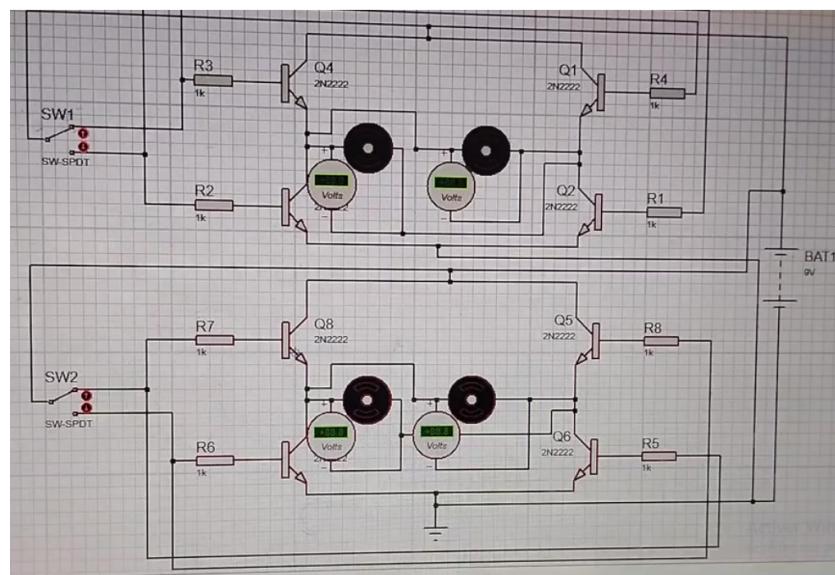


Figure 7 : Modélisation du robot sur Proteus

3.4. Programmation des différentes fonctions

3.4.1. Fonctions “Avancer/Reculer” et “Arrêter”

Au commencement de notre projet, nous avons choisi d'adopter une approche progressive pour le développement de notre robot. Nous avons commencé par implémenter et tester des fonctions de base, afin de nous assurer que les composants essentiels de notre robot fonctionnaient correctement.

On a ainsi d'abord distingué deux fonctions à mettre en pratique :

- Fonction “Avancer et Reculer” : Nous avons débuté avec un code simple qui permet au robot d'avancer et de reculer (annexe). Cette fonction utilise les moteurs 3 V à 6 V contrôlés par l'Arduino Motor Shield. En ajustant la vitesse des moteurs, nous avons pu faire avancer et reculer notre robot.
- Fonction Arrêter : Nous avons également implémenté une fonction “arrêter” (annexe) qui utilise le frein de la carte Arduino Motor Shield. Cette fonction permet d'arrêter le robot en utilisant les freins des moteurs.

[Voici une vidéo du test de ce premier code simple intégrant ces deux fonctions](#)

Ces premiers tests ont été cruciaux pour vérifier le bon fonctionnement des composants de base de notre robot et nous familiariser avec notre système, de comprendre son fonctionnement, et de préparer le terrain pour l'ajout de fonctionnalités plus complexes. Ils ont également constitué la première étape de notre progression vers un robot capable d'éviter les obstacles.

On a pu ensuite écrire un programme qui permet au robot d'avancer en boucle pour intégrer au robot la fonction de détection d'obstacle, permise par le capteur à ultrasons.

3.4.2. Intégration du capteur à ultrasons - Fonction “Déetecter”

Pendant que le robot roule, le capteur mesure constamment la distance à l'obstacle devant le robot. Si le capteur détecte un obstacle à une distance choisie de moins de 20 cm, le robot s'arrête automatiquement.

[Voici une vidéo pour ce test](#)

Cette étape nous a donc permis de nous assurer que l'on peut maintenant détecter les obstacles et faire réagir le robot en conséquence.

3.4.3. Intégration du servomoteur

Nous avons ensuite introduit le servomoteur à notre système. Le servomoteur permettra au capteur à ultrasons de scanner l'environnement à 180 degrés. Cela permet de fournir plus d'informations sur d'autres obstacles potentiels et laisser le robot prendre une meilleure décision pour la direction à suivre pour éviter les obstacles.

Nous avons donc mis en place un programme qui permet au capteur à ultrasons de pivoter de gauche à droite pour collecter des informations sur l'environnement.

[Voici une vidéo pour ce test](#)

3.4.4. Comportement du robot face aux obstacles

L'étape suivante a été de réaliser des tests pour évaluer la capacité du robot à éviter les obstacles. Dans les vidéos suivantes, vous pourrez voir le robot prendre une décision lorsqu'il se retrouve dans différentes situations.

[Éviter un obstacle vers la droite](#)

[Éviter un obstacle vers la gauche](#)

Ces vidéos illustrent clairement la manière dont notre système permet au robot de prendre des décisions efficaces pour éviter les obstacles et continuer son parcours.

3.4.5. Programme final

Dans notre programme final, le robot avance tout en surveillant constamment la présence d'obstacles devant lui grâce au capteur à ultrasons.

Lorsque le capteur détecte un obstacle à moins de 30 cm devant le robot, il s'arrête et utilise le servomoteur pour balayer l'environnement à gauche et à droite. Les informations collectées permettent au robot de prendre une décision sur la direction à suivre. Si un obstacle est détecté à gauche, le robot évite vers la droite, et vice versa. S'il y a des obstacles des deux côtés, le robot effectue un demi-tour. Et s'il ne détecte aucun obstacle à droite et à gauche, nous lui imposons arbitrairement d'éviter l'obstacle vers la droite.

Les ajustements de la vitesse des moteurs se font grâce à la courbe de simulation des virages. En effet, cette courbe nous indique quelle valeur donner au robot afin qu'il effectue un virage correct pour éviter un obstacle.

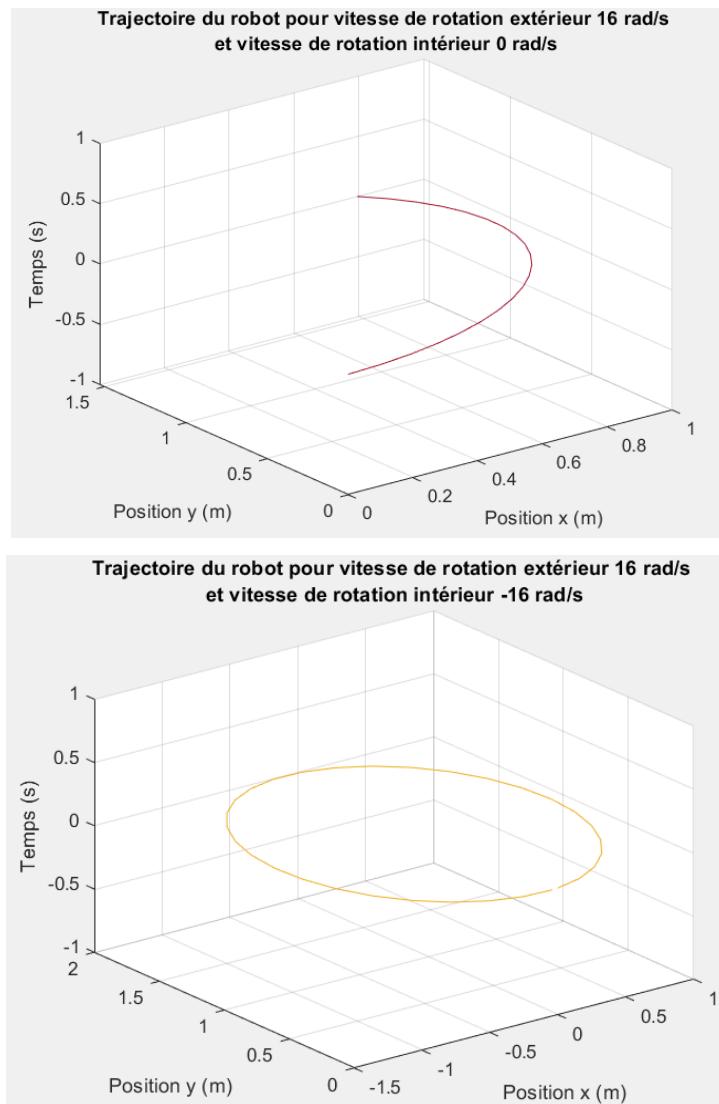


Figure 8 : Trajectoire du robot pour différentes vitesses de rotation

4. Considérations théoriques

4.1. Contrôle de la vitesse des moteurs

4.1.1. Pulse Width Modulation (PWM)

Le contrôle de vitesse d'un véhicule robot avec un Arduino implique l'utilisation de la modulation de largeur d'impulsion (PWM) pour ajuster la vitesse des moteurs du véhicule. Cette méthode permet de varier la puissance fournie aux moteurs en contrôlant la durée pendant laquelle le signal de commande est activé sur chaque cycle, ce qui régule la vitesse de rotation des moteurs et donc la vitesse de déplacement du véhicule.

L'Arduino génère les signaux PWM nécessaires pour contrôler les moteurs du véhicule. En ajustant la valeur du rapport cyclique du signal PWM envoyé à chaque moteur, il est possible de réguler la vitesse de rotation de manière proportionnelle.

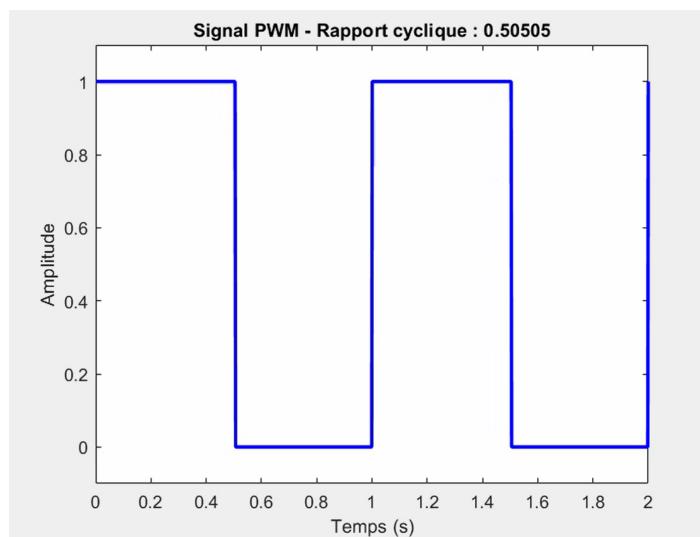


Figure 9 : Simulation signal PWM avec rapport cyclique de 0.50505

4.1.2. Importance du contrôle de vitesse

Le but principal du contrôle de vitesse dans notre utilisation est de permettre une manipulation précise et adaptative du véhicule. Cette fonctionnalité est cruciale pour plusieurs aspects :

- Contrôle de la vitesse : Le réglage précis de la vitesse des moteurs permet au véhicule de s'adapter aux exigences de différents scénarios d'application, tels que la navigation dans des environnements variés, le suivi de trajectoires spécifiques ou la réalisation de mouvements précis.

- Contrôle de la direction du mobile : En ajustant les vitesses des moteurs de chaque côté du véhicule de manière indépendante, le contrôle de direction permet au véhicule de tourner avec précision, de s'orienter autour d'obstacles et de suivre des trajectoires spécifiques.
- Efficacité énergétique : En optimisant la puissance fournie aux moteurs en fonction des besoins réels du véhicule, le contrôle de vitesse contribue à une utilisation plus efficace de l'énergie, prolongeant ainsi l'autonomie de la batterie et réduisant la consommation d'énergie.

En résumé, le contrôle de vitesse d'un véhicule robot avec un Arduino offre une méthode flexible et précise pour ajuster la vitesse de déplacement, ce qui permet au véhicule de s'adapter à son environnement.

4.1.3. Relation PWM/vitesse de rotation des roues

4.1.3.1. Courbe théorique

À partir de ce point, nous avons donc voulu caractériser la vitesse de rotation du mobile en fonction du PWM. Pour cela, nous sommes partis des caractéristiques du moteur à courant continu. Nous avons ainsi obtenu la fonction de transfert suivante :

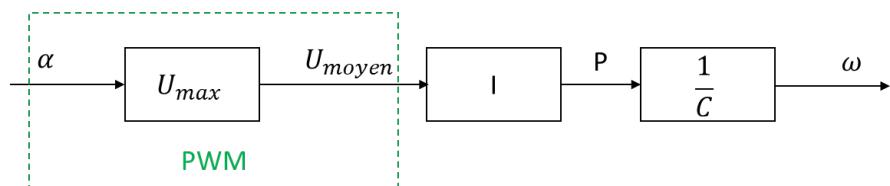


Figure 10 : Fonction de transfert caractéristique du moteur

Pour bien comprendre le fonctionnement du PWM et de la rotation des roues, nous avons donc fait une simulation avec Matlab de notre système et on obtient la courbe ci-dessous :

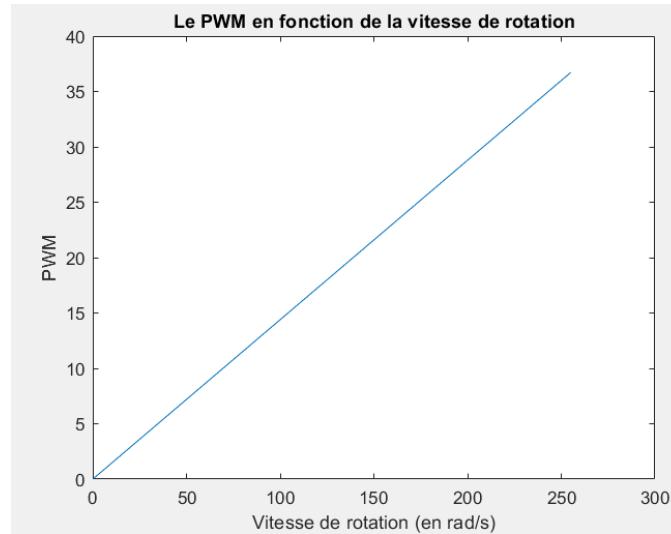


Figure 11 : PWM en fonction de la vitesse de rotation (rad/s)

Après la simulation effectuée par le groupe des GP, nous avons donc voulu vérifier que le comportement réel du module se rapproche du modèle théorique.

4.1.3.2. Courbe pratique

Nous avons fait varier le rapport cyclique du signal PWM progressivement, en mesurant à chaque étape la distance parcourue par le robot. Avec cette distance on peut donc calculer la vitesse correspondante du module. Ces données ont ensuite été utilisées pour créer la courbe réelle.

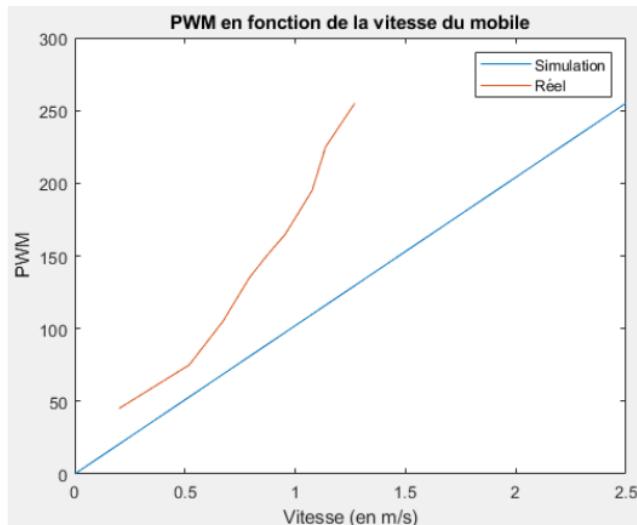


Figure 12 : Comparaison PWM en fonction de la vitesse de rotation (rad/s) dans le cas réel par rapport à la simulation

4.1.3.2. Analyse

En comparant attentivement les courbes théoriques et réelles, nous avons observé que la courbe réelle n'est pas linéaire comme on le souhaitait. Nous pensons que ces écarts sont dus à l'usure des moteurs et au poids ajouté sur le robot.

4.1.4. Corrections

Afin de réduire les écarts, nous avons opté pour l'utilisation d'un capteur de vitesse de rotation. Le capteur de vitesse de rotation est un composant crucial dans le contrôle d'un véhicule robotisé. Intégré au système de contrôle via une carte Arduino, ce capteur utilise un principe optique pour détecter les rotations de la roue du véhicule. Voici comment il fonctionne :

- Le capteur optique à fourche intercepte un signal lumineux émis d'un côté et reçu de l'autre côté de la fourche. Ce signal est régulièrement "coupé" par une roue perforée en rotation, créant ainsi des interruptions dans le flux lumineux.
- En mesurant le nombre d'interruptions par seconde, on peut déterminer la vitesse de rotation de la roue. Cette vitesse est directement proportionnelle au nombre d'interruptions détectées.
- Il est également important de tenir compte du nombre de trous présents dans la roue codeuse. Chaque trou représente une interruption dans le signal détecté par le capteur. Ainsi, plus il y a de trous dans la roue, plus le nombre d'interruptions par seconde sera élevé.

Le but de tout cela est de calculer le déplacement linéaire du véhicule en utilisant la circonference C des roues. Il est par ailleurs possible de calculer la distance parcourue et donc la position angulaire du véhicule par rapport à son point de départ.

Pour voir l'intérêt et l'approximation qu'il existe avec la réalité, nous avons alors voulu mesurer la vitesse de rotation des roues gauches et droites du véhicule grâce au code Arduino en annexe.

Dans un premier temps, nous avons fait des expériences en boucle ouverte. Dans une boucle ouverte, le système prend des décisions basées uniquement sur l'entrée donnée et sur une relation préétablie entre l'entrée et la sortie. En boucle ouverte, la fonction de transfert de notre système est donnée par :

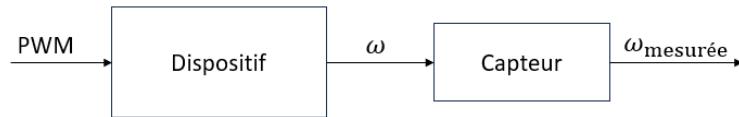


Figure 13 : Fonction de transfert de la vitesse angulaire en boucle ouverte

On obtient ainsi :

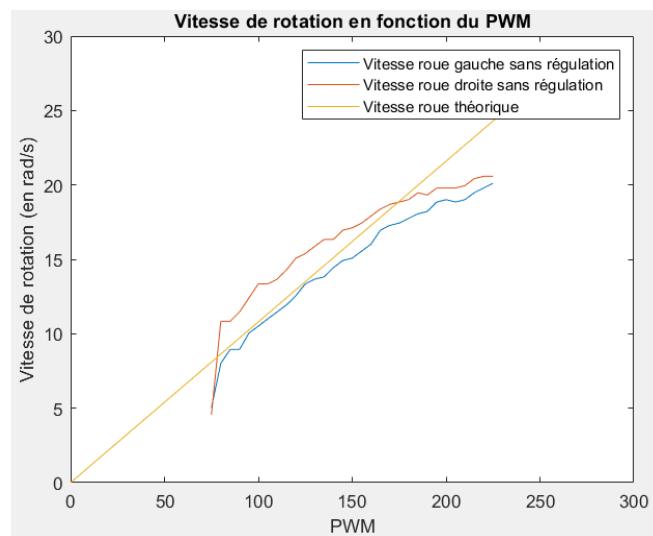


Figure 14 : Vitesse de rotation des roues (rad/s) en fonction du PWM

On a ici un graphique montrant la vitesse de rotation en fonction du PWM. Le résultat qu'on a ici se rapproche plus de la théorie que ce qu'on a pu remarquer précédemment. Néanmoins, nous ne retrouvons pas une courbe linéaire en sortie à cause des perturbations externes qu'on ne prend pas en compte. Grâce au capteur, on se rapproche bien de la courbe théorique, mais ce qu'on désire, c'est de s'y rapprocher le plus possible. Pour cela, on fait donc l'expérience en boucle fermée. Dans une boucle fermée, le système compare en permanence la sortie réelle du système avec la sortie désirée et utilise cette différence, également appelée erreur, pour ajuster le système afin de minimiser cette erreur. Dans notre cas, les valeurs théoriques seront nos valeurs de référence, ainsi en boucle fermée, on a :

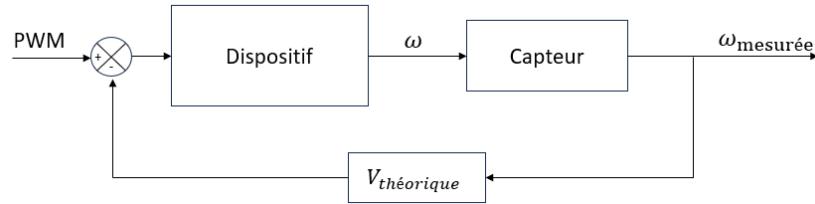


Figure 15 : Fonction de transfert de la vitesse angulaire en boucle fermée

On obtient alors :

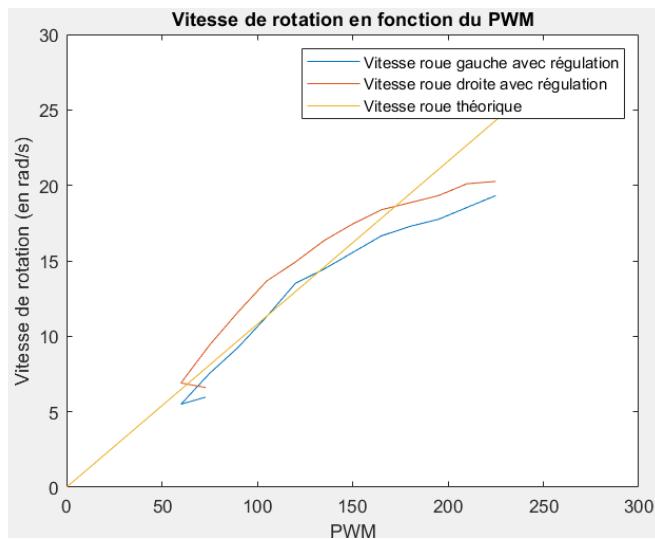


Figure 16 : Vitesse de rotation des roues (rad/s) en fonction du PWM

Après régulation, on ne constate pas de vraie différence ou plutôt d'amélioration. Cela nous montre que la régulation n'a pas eu d'impact positif sur notre système. L'utilisation du capteur de vitesse n'est pas indispensable, car pour l'instant, elle nous permet juste de mesurer la vitesse de rotation des roues et rien d'autre.

4.2. Contrôle en virage

4.2.1. Problème

Dans l'approche des virages, on a besoin de maîtriser la trajectoire de la voiture afin d'éviter l'obstacle de la façon voulue.

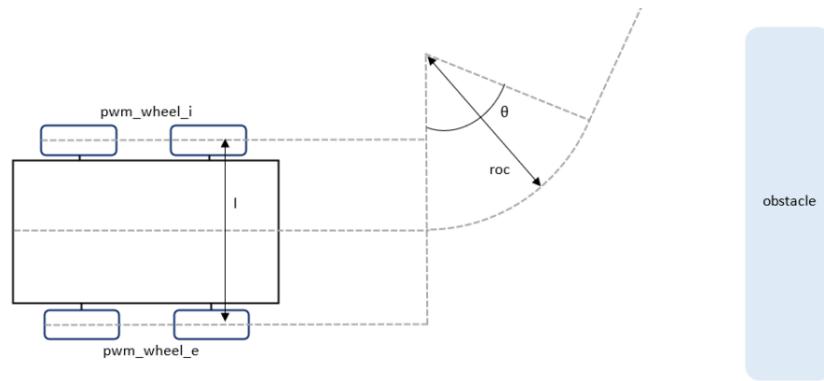


Figure 17 : Virage du mobile pour esquive d'obstacle

En virage, les paramètres à prendre en compte sont :

- Le rapport k des vitesses de rotation des roues
- Le rayon de courbure roc du virage
- L'angle de rotation θ
- L'empattement l imposé par le châssis

De plus, on a le paramétrage : $k = \frac{w_{wheel_i}}{w_{wheel_e}}$, $roc = \frac{l}{2} * \frac{1+k}{1-k}$.

Les équations principales qui régissent le virage sont :

$$k = \frac{w_{wheel_i}}{w_{wheel_e}}$$

$$roc = \frac{l}{2} * \frac{1+k}{1-k}$$

$$\dot{\theta} = \frac{\pi * D_{wheel} * (w_{wheel_i} + w_{wheel_e})}{l}$$

$$\dot{\theta}_i = \dot{\theta} * \frac{(2 * roc - l)}{D_{wheel}}$$

$$\dot{\theta}_e = \dot{\theta} * \frac{(2 * roc + l)}{D_{wheel}}$$

On cherche d'abord à étudier l'influence du rapport k sur le rayon de courbure du virage. En effet, on cherchera à avoir un rayon de courbure pas trop grand pour atteindre rapidement l'angle voulu, mais assez important pour éviter l'obstacle.

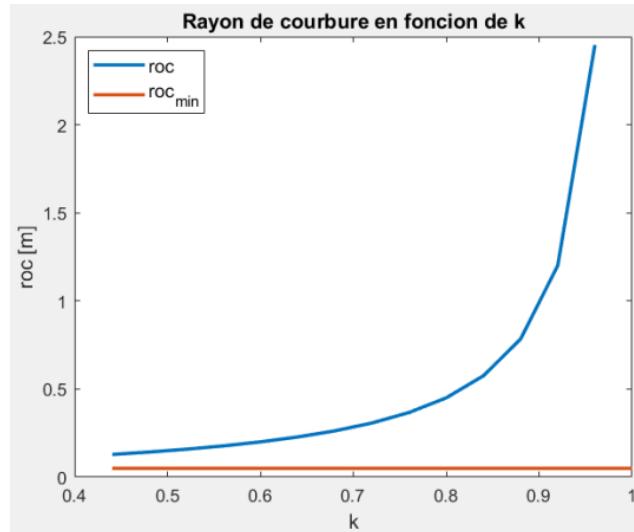


Figure 18 : Rayon de courbure du virage en fonction du rapport k

On voit qu'on a un intérêt important à prendre un faible k, c'est-à-dire une grande différence de PWM. Toutefois, en pratique, ce rapport k est limité par le PWM minimal de fonctionnement du moteur qui est de 45 environ.

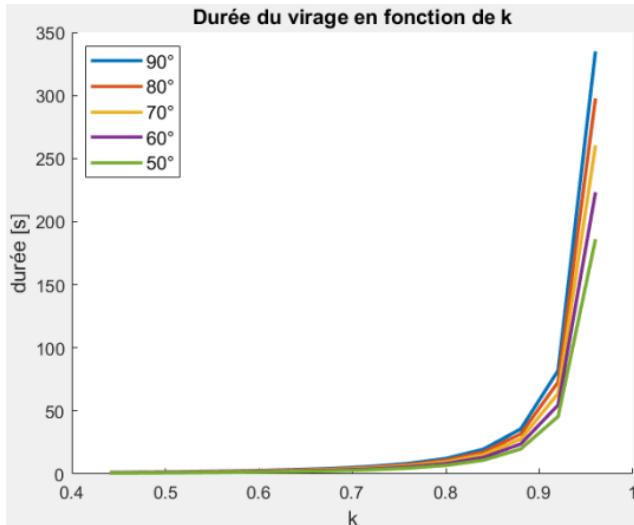


Figure 19 : Durée du virage (s) en fonction de k

Ici, on voit bien qu'on a comme critère une durée raisonnable pour atteindre l'angle voulu, un angle suffisant pour éviter l'obstacle, mais un angle pas trop grand non plus pour avoir une rotation.

4.2.2. Pratique

À partir de ces simulations et combinées aux simulations faites en partie 3.4.5, cela donne la durée pendant laquelle on donne un certain PWM aux roues pour faire le virage voulu.

5. CONCLUSION

5.1. Bilan du Projet

Ce projet transverse sur le robot mobile qui évite les obstacles nous a permis de développer nos connaissances et nos compétences dans les domaines de l'automatique, du signal et de la programmation embarquée. Durant la réalisation de ce projet, nous avons tous travaillé ensemble afin que chaque membre du groupe participe au projet et contribue à son bon déroulement.

L'ensemble de ce projet nous a permis de comprendre l'importance de notre projet afin de casser les barrières qui existent entre les matières de l'automatique et celle des systèmes embarqués. Pour ce faire, nous avons étudié et conçu nos manipulations de la manière la plus ludique possible afin de pouvoir finalement proposer un sujet de TP aux étudiants. Au cours de la réalisation de ce projet, nous avons fait face à de nombreux problèmes tant liés au matériel qu'aux différences entre résultats théorique et pratique, et cela constitue un point important de notre projet afin de comprendre et d'analyser les raisons de ces problèmes.

L'ensemble de ce projet nous a également permis d'acquérir de nombreuses compétences dans la gestion de projet, en effet, nous avons appris à gérer notre temps afin de pouvoir répondre aux attentes de nos professeurs et livré en temps voulu un résultat satisfaisant et fonctionnel.

Ce projet nous a également permis d'utiliser des outils tels que le diagramme de Gantt ou encore le diagramme Bête à corne qui sont des outils indispensables pour un futur ingénieur.

5.2. Leçons Apprises

Le projet a mis en lumière plusieurs leçons clés :

- L'importance de la planification : Une organisation rigoureuse et une répartition claire des tâches ont été cruciales pour le succès du projet.
- Flexibilité et résolution de problèmes : La capacité de prendre du recul lors de la résolution de problème technique afin de pouvoir trouver les meilleures solutions possibles.
- Intégration de la théorie et de la pratique : L'importance d'une mise en parallèle des données pratiques et théoriques dans le but d'analyser nos résultats de la bonne manière.

5.3. Perspectives et Améliorations Futures

Pour continuer ce projet, plusieurs axes d'amélioration sont envisagés :

- Amélioration des composants : utiliser des capteurs de meilleure qualité ou de technologies plus avancées pour une détection plus précise et fiable.
- Optimisation des algorithmes : Poursuivre le développement des algorithmes de navigation pour améliorer la fluidité et la sécurité du mouvement du robot dans des environnements plus complexes.
- Élargissement des fonctionnalités : Intégrer des fonctionnalités supplémentaires telles que le suivi de trajectoire préprogrammée et la reconnaissance d'objets pour rendre le robot plus polyvalent.

En conclusion, ce projet nous a permis d'apprendre à travailler en équipe sur un sujet d'étude complexe. Il nous aura fallu faire preuve d'une bonne communication et d'un sens du travail d'équipe afin de surmonter chacun des problèmes qui se sont dressés sur notre route. Ce type de projet sont des moyens très concrets de toucher du bout du doigt le métier d'un ingénieur et les tâches auxquelles celui-ci fait face.

BIBLIOGRAPHIE

Forlot, J. (2021, Mai 05). *ACTIVITÉ ARDUINO/PYTHON : Mesurer une vitesse à l'aide d'un module capteur de vitesse de rotation LM293*. Récupéré sur Labo Physique Pothier:

<http://jonas.forlot.free.fr/index.php/2021/05/14/activite-arduino-python-mesurer-une-vitesse-a-laide-dun-module-capteur-de-vitesse-de-rotation-lm293-type-fc-03-ou-vma347-trace-de-graphe-en-temps-reel/>

Gotronic. (s.d.). *Notice de montage du châssis ROBOT03*. Récupéré sur Gotronic:

<https://www.gotronic.fr/pj2-35326-robot03-montage-1601.pdf>

Lhomme-Desages, D. (2008, Avril). *Commande d'un robot mobile rapide à roues non directionnelles sur sol naturel*. Récupéré sur HAL Theses: <https://theses.hal.science/tel-00812508/>

MEDANI, S. M. (2017, Septembre 20). *Mémoire de Fin d'Etude*. Récupéré sur DS Space:
<https://dspace.ummt.dz/server/api/core/bitstreams/6a4aadb6-0bdf-4e53-83b3-1b7575f6160b/content>

Robert, S. (2023, Mars 29). *Tutoriel Arduino et Raspberry Pi*. Récupéré sur Comment contrôler un servomoteur avec un Arduino ?: <https://www.tutoriel-arduino.com/controler-un-servomoteur-avec-arduino/>

Söderby, K. (2023, Février 02). *Controlling a DC Motor with Motor Shield Rev3*. Récupéré sur Arduino: <https://docs.arduino.cc/tutorials/motor-shield-rev3/msr3-controlling-dc-motor/>

ST. (2023, Octobre). *Dual full-bridge driver*. Récupéré sur ST:
<https://www.st.com/resource/en/datasheet/l298.pdf>

Velleman. (2017, Février 02). *HC-SR05 ULTRASONIC SENSOR*. Récupéré sur Velleman: https://cdn.velleman.eu/downloads/29/vma306_a4v01.pdf

Whadda. (s.d.). *IR speed sensor module*. Récupéré sur Whadda:
<https://www.velleman.eu/downloads/25/wpse347a4v01.pdf>

ANNEXES

ANNEXE 1 : Devis pour le TP



GO TRONIC
35 ter route Nationale
08110 BLAGNY

Tel : +33.(0)3.24.27.93.42
Fax : +33.(0)3.24.27.93.50
E-mail : contact@gotronic.fr
N° TVA : FR80.438.306.680
N° SIRET : 438.306.680.00028

Date du devis : 26/04/2024

DEVIS

Panier N° 1506114

www.gotronic.fr est la propriété de la société Go Tronic - SARL au capital de 8000 euros
CODE APE 4759B - N° Siret 43830668000028 - N° de TVA FR80.438.306.680
Réserve de propriété : nous nous réservons la propriété des marchandises jusqu'au paiement intégral de nos factures (loi N°80335 du 12 mai 1980).
Loi n°92.1992 du 31/12/1992, un intérêt de retard égal à 1,5 fois le taux d'intérêt légal sera appliquée au montant de la facture lors d'un paiement tardif.

FRAIS DE PORT : (France métropolitaine)

Envoi postal économique (délai 1 semaine) : 5,90 € TTC
Envoi colissimo suivi (±48h) : 7,90 € TTC
Envoi par DPD (24 à 48h – pros uniquement) : 7,90 € TT
Port gratuit à partir de 180 € TTC

ANNEXE 2 : Code Arduino pour nos manipulations

```

1 #include <NewPing.h>
2 #include <Servo.h>
3 #include <time.h>
4
5 // Pins pour les moteurs (moteurs du même côté couplés ensemble)
6 const int moteurGauche_dir = 12;
7 const int moteurGauche_pwm = 3;
8 const int moteurGauche_brake = 9;
9 const int moteurDroit_dir = 13;
10 const int moteurDroit_pwm = 11;
11 const int moteurDroit_brake = 8;
12
13 //sonar
14 const int trigPin = 4;
15 const int echoPin = 5;
16
17 //pour calcul de la distance du sonar
18 long duration;
19 int distance;
20 int obstacle = 0;
21 |
22 double A = 0.146;
23 double B = 3.72;
24 double C = 0;
25 double r = 0.065/2;
26
27 double L = 10;
28
29 double fin=0;
30 double debut=0;
31 double fin_boucle;
32 double duree_virage;
33 double dt = 0;
34

35
36 double theta = 0;
37
38 int pwm =150;
39 int pwm_g = 150;
40 int pwm_d = 150;
41
42 bool cas = false; // Rester a l'arret pour analyser les c&ot;es
43
44 int obstacle_droit = 0;
45 int obstacle_gauche = 0;
46
47 Servo servo; // cr&eacute;ation de l'objet "servo"
48
49 void setup() {
50     //Configurations des pins pour le sonar
51     pinMode(trigPin, OUTPUT);
52     pinMode(echoPin, INPUT);
53     Serial.begin(115200);
54
55     // Configuration des pins pour les moteurs
56     pinMode(moteurGauche_dir, OUTPUT);
57     pinMode(moteurGauche_pwm, OUTPUT);
58     pinMode(moteurGauche_brake, OUTPUT);
59     pinMode(moteurDroit_dir, OUTPUT);
60     pinMode(moteurDroit_pwm, OUTPUT);
61     pinMode(moteurDroit_brake, OUTPUT);
62
63     servo.attach(6); // attache le servo au pin sp&ecirc;cifi&eacute;
64
65 }

```

```
68 void loop() {
69     delay(4000);
70     redemarrer(); // Retirer les freins des moteurs
71
72     avancer(); // Avancer le robot
73
74     detection_obstacle(); // Si un obstacle est détecté le robot s'arrete
75
76     regarder_environnement(); // Le robot regarde autour de lui pour connaitre l'environnement autour de lui
77
78     decision_direction(); // Le robot choisit la direction à prendre
79 }
80
81 void avancer() {
82     // Moteur gauche
83     digitalWrite(moteurGauche_dir, LOW); // Avancer
84     analogWrite(moteurGauche_pwm, pwm_g);
85
86
87     // Moteur droit
88     digitalWrite(moteurDroit_dir,HIGH); //Avancer
89     analogWrite(moteurDroit_pwm, pwm_d);
90 }
91
92
93 void reculer() {
94     // Moteur gauche
95     digitalWrite(moteurGauche_dir, HIGH); // Reculer
96     analogWrite(moteurGauche_pwm, pwm);
97
98     // Moteur droit
99     digitalWrite(moteurDroit_dir, LOW); // Reculer
100    analogWrite(moteurDroit_pwm, pwm);
101 }
102
103
104 void arreter() {
105     // Arrêter les moteurs gauche et droit
106     digitalWrite(moteurGauche_brake, HIGH);
107     digitalWrite(moteurDroit_brake, HIGH);
108
109     // Attendre un court moment pour assurer l'arrêt complet
110     //delay(4000);
111 }
112
113 void redemarrer(){
114     // Arrêter les moteurs gauche et droit
115     digitalWrite(moteurGauche_brake, LOW);
116     digitalWrite(moteurDroit_brake, LOW);
117 }
```

```
119 void eval_obstacle_d devant() {
120 //CALCUL DE LA DISTANCE
121 // Clears the trigPin
122 digitalWrite(trigPin, LOW);
123 delayMicroseconds(2);
124 // Sets the trigPin on HIGH state for 10 micro seconds
125 digitalWrite(trigPin, HIGH);
126 delayMicroseconds(10);
127 digitalWrite(trigPin, LOW);
128 // Reads the echoPin, returns the sound wave travel time in microseconds
129 duration = pulseIn(echoPin, HIGH);
130 // Calculating the distance
131 distance = duration * 0.034 / 2;
132
133
134 //EVALUATION SI OBSTACLE OU NON
135 if(distance <= 30){
136 | obstacle = 1;
137 }
138
139 }
140
141 void detection_obstacle(){
142 while(obstacle !=1){
143 | eval_obstacle_d devant();
144 }
145
146 arreter();
147 }
148
149 void regarder_a_droite(){
150 servo.write(180); // demande au servo de se déplacer à cette position
151 delay(1000);
152 eval_obstacle_d devant();
153 servo.write(90);
154 delay(500);
155 }
156
157 void regarder_a_gauche(){
158 servo.write(0); // demande au servo de se déplacer à cette position
159 delay(1000);
160 eval_obstacle_d devant();
161 servo.write(90);
162 delay(500);
163 }
164
165 void regarder_environnement(){
166 delay(500);
167 obstacle = 0;
168 regarder_a_gauche();
169 delay(1000);
170 if(obstacle == 1){
171 | obstacle_gauche = 1;
172 }
173 obstacle = 0;
174 regarder_a_droite();
175 delay(1000);
176 if(obstacle == 1){
177 | obstacle_droit = 1;
178 }
179 obstacle = 0;
180 }
```

```

183 void eviter_vers_droite(){
184     // Virage droit
185     pwm_d=50;
186     digitalWrite(moteurDroit_dir, HIGH); // Avancer
187     analogWrite(moteurDroit_pwm, pwm_d);
188
189     digitalWrite(moteurGauche_dir, LOW); // Avancer
190     analogWrite(moteurGauche_pwm, pwm_g);
191     delay(700);
192
193     // Virage gauche
194     pwm_d=150;
195     digitalWrite(moteurDroit_dir, HIGH); // Avancer
196     analogWrite(moteurDroit_pwm, pwm_d);
197     pwm_g=50;
198     digitalWrite(moteurGauche_dir, LOW); // Avancer
199     analogWrite(moteurGauche_pwm, pwm_g);
200     delay(1200);
201
202     // Ligne droite
203     pwm_d=150;
204     digitalWrite(moteurDroit_dir, HIGH); // Avancer
205     analogWrite(moteurDroit_pwm, pwm_d);
206     pwm_g=150;
207     digitalWrite(moteurGauche_dir, LOW); // Avancer
208     analogWrite(moteurGauche_pwm, pwm_g);
209     //delay(700);
210

```

```

211     // Virage gauche
212     pwm_d=150;
213     digitalWrite(moteurDroit_dir, HIGH); // Avancer
214     analogWrite(moteurDroit_pwm, pwm_d);
215     pwm_g=50;
216     digitalWrite(moteurGauche_dir, LOW); // Avancer
217     analogWrite(moteurGauche_pwm, pwm_g);
218     delay(900);
219
220     // Virage droit
221     pwm_d=50;
222     digitalWrite(moteurDroit_dir, HIGH); // Avancer
223     analogWrite(moteurDroit_pwm, pwm_d);
224     pwm_g=150;
225     digitalWrite(moteurGauche_dir, LOW); // Avancer
226     analogWrite(moteurGauche_pwm, pwm_g);
227     delay(1100);
228
229     arreter();
230     // Attendre un court moment pour assurer l'arrêt complet
231     delay(4000);
232     redemarrer();
233 }
234

```

```

235 void eviter_vers_gauche(){
236     // Virage gauche
237     pwm_g=50;
238     digitalWrite(moteurDroit_dir, HIGH); // Avancer
239     analogWrite(moteurDroit_pwm, pwm_d);
240     digitalWrite(moteurGauche_dir, LOW); // Avancer
241     analogWrite(moteurGauche_pwm, pwm_g);
242     delay(600);
243
244     // Virage droit
245     pwm_g=150;
246     digitalWrite(moteurGauche_dir, LOW); // Avancer
247     analogWrite(moteurGauche_pwm, pwm_g);
248     pwm_d=50;
249     digitalWrite(moteurDroit_dir, HIGH); // Avancer
250     analogWrite(moteurDroit_pwm, pwm_d);
251     delay(1500);
252
253     // Ligne droite
254     pwm_d=150;
255     digitalWrite(moteurDroit_dir, HIGH); // Avancer
256     analogWrite(moteurDroit_pwm, pwm_d);
257     pwm_g=150;
258     digitalWrite(moteurGauche_dir, LOW); // Avancer
259     analogWrite(moteurGauche_pwm, pwm_g);
260     //delay(1200);
261

```

```

261 // Virage gauche
262 pwm_g=150;
263 digitalWrite(moteurGauche_dir, LOW); // Avancer
264 analogWrite(moteurGauche_pwm, pwm_g);
265 pwm_d=50;
266 digitalWrite(moteurDroit_dir, HIGH); // Avancer
267 analogWrite(moteurDroit_pwm, pwm_d);
268 delay(800);

270 // Virage droit
271 pwm_g=50;
272 digitalWrite(moteurGauche_dir, LOW); // Avancer
273 analogWrite(moteurGauche_pwm, pwm_g);
274 pwm_d=150;
275 digitalWrite(moteurDroit_dir, HIGH); // Avancer
276 analogWrite(moteurDroit_pwm, pwm_d);
277 delay(1000);

279 arreter();
280 // Attendre un court moment pour assurer l'arrêt complet
281 delay(4000);
282 redemarrer();

284 }

285

286 void demi_tour(){
287 pwm_d=150;
288 digitalWrite(moteurDroit_dir, LOW); // Avancer
289 analogWrite(moteurDroit_pwm, pwm_d);
290 pwm_g=150;
291 digitalWrite(moteurGauche_dir, LOW); // Avancer
292 analogWrite(moteurGauche_pwm, pwm_g);
293 delay(1200);
294 arreter();
295 // Attendre un court moment pour assurer l'arrêt complet
296 delay(4000);
297 redemarrer();

298 }

299

300 void decision_direction(){
301 if(obstacle_droit == 1 && obstacle_gauche == 0){
302 redemarrer();
303 eviter_vers_gauche();
304 }
305 if(obstacle_gauche == 1 && obstacle_droit == 0){
306 redemarrer();
307 eviter_vers_droite();
308 }
309 if(obstacle_gauche == 1 && obstacle_droit == 1){
310 redemarrer();
311 demi_tour();
312 }
313 if(obstacle_gauche == 0 && obstacle_droit == 0){
314 redemarrer();
315 demi_tour();
316 }
317 }

318

319 obstacle = 0;
320 obstacle_droit = 0;
321 obstacle_gauche = 0;
322 pwm_d = 150;
323 pwm_g = 150;
324 }

325 }

326

```

ANNEXE 3 : Fiche de TP pour étudiant

Fiche de TP pour le projet de Robot mobile évitant les obstacles