

**BEYOU Nolan**  
**David Marielle**  
**Denormandie Guillermo**  
**Bénard Gabriel**



Semestre A24



UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE

RAPPORT DE TM02 - ROBOT SUIVEUR DE LIGNE

---

## Remerciements

Nous tenons à exprimer nos remerciements aux professeurs de l'UV TM02 pour leur encadrement, et leurs judicieux conseils notamment sur la manipulation des machines et l'utilisation du matériel électronique. Ils nous résolus plusieurs soucis majeurs durant le semestre.

De plus, nous remercions les autres groupes inscrits à l'UV qui nous débloqué dans certaines situations.

## Notations et acronymes

### Acronymes

- **CAO** : Conception Assistée par Ordinateur.
- **PWM** : Modulation de largeur d'impulsion (*Pulse Width Modulation*), utilisée pour contrôler la vitesse des moteurs.

### Notations

- **ENA, ENB** : Broches contrôlant la vitesse des moteurs gauche et droit.
- **IN1, IN2, IN3, IN4** : Broches définissant la direction des moteurs.
- **A1 à A5** : Broches analogiques associées aux cinq capteurs infrarouges.

### Concepts Techniques Importants

- **Arduino** :
  - Carte contenant le microcontrôleur utilisé pour la programmation.
  - Langage de programmation basé sur le C, adapté à la programmation des cartes Arduino.
  - Logiciel permettant d'écrire et de téléverser des programmes sur la carte.
- **Capteurs BFD1000** : Capteurs infrarouges utilisés pour détecter la position de la ligne. Ils comportent cinq détecteurs permettant une lecture en continu.
- **Algorithme de suivi de ligne** :
  - Fonction `setup()` : Initialisation des broches et des variables.
  - Fonction `loop()` : Lecture des valeurs des capteurs et ajustement de la trajectoire.
  - Gestion des cas particuliers : Arrêt lorsque la ligne n'est pas détectée, correction directionnelle en cas de déviation.
- **Structure mécanique** :
  - Châssis à deux étages pour optimiser la répartition des composants et le câblage.
  - Utilisation de roues motrices imprimées en 3D et d'une bille multidirectionnelle pour stabiliser le système.

---

## Table des matières

<b>1</b>	<b>Introduction</b>
<b>2</b>	<b>Organisation du projet</b>
<b>3</b>	<b>Développement du prototype</b>
3.1	Partie mécanique . . . . .
3.1.1	Conception du châssis . . . . .
3.1.2	Conception des roues . . . . .
3.1.3	Création du système de direction . . . . .
3.1.4	Conception du carénage . . . . .
3.2	Parties électronique et informatique . . . . .
3.2.1	Les capteurs et branchements . . . . .
3.2.2	Le logiciel et la Carte Arduino . . . . .
3.2.3	L'algorithme . . . . .
<b>4</b>	<b>Difficultés rencontrées</b>
4.1	Difficultés mécaniques . . . . .
4.2	Difficultés informatiques . . . . .
4.3	Améliorations potentielles pour un projet plus long . . . . .
<b>5</b>	<b>Robots suiveurs de ligne dans la vie quotidienne</b>
5.1	Le TEOR . . . . .
5.2	Les véhicules à guidage automatique (VGA) . . . . .
5.3	Autres exemples . . . . .
<b>6</b>	<b>Conclusion et Perspectives</b>
<b>7</b>	<b>Annexes</b>
7.1	Annexe A - Plans de Conceptions . . . . .
7.2	Annexe B - Code final . . . . .

## 1 Introduction

L'UV TM02, est une unité d'enseignement qui donne l'opportunité de concrétiser un prototype technique durant un semestre. Cet apprentissage permet d'explorer les dimensions de la conception, du prototypage, de l'électronique. Cette matière renforce nos compétences créatives tout en respectant un cahier des charges. De plus L'UV valorise particulièrement le travail en équipe ; ici ce projet collaboratif est réalisé par un groupe de quatre étudiants au FabLab de l'UTC.

Au cours de ce semestre, nous avons travaillé sur un projet de robot suiveur de ligne autonome. Il doit être réalisé à partir de pièces en bois découpées au laser et en impression 3D. Ce rapport technique retrace les points essentiels de notre projet. Nous avons conçu un robot capable de se déplacer de manière autonome autour d'un parcours ovale.



FIGURE 1 – Visuel de la zone de conception du FabLab

## 2 Organisation du projet

Nous avons réparti le travail en deux groupes. Cette organisation se justifie par la division naturelle du projet en deux grandes composantes : d'une part, la conception mécanique, et d'autre part, l'électronique et la programmation. Cette structuration a permis à chaque groupe de travailler de manière relativement indépendante tout en progressant efficacement sur leurs domaines respectifs. Les groupes étaient les suivants :

- **Mécanique** : Gabriel Bénard et Guillermo Denormandie
- **Électronique et Programmation** : Marielle David et Nolan Beyou

### 3 Développement du prototype

Le cahier des charges nous impose les contraintes suivantes : nous devions réaliser un robot autonome capable de suivre une ligne noire dessinée sur le sol puis de se rediriger si besoin pour suivre des courbes. De plus, la longueur du chassis doit être comprise entre 20 et 30 cm. En outre, le mouvement du robot doit être actionné par deux moteurs à courant continu et la direction peut être actionnée par un servomoteur (4 roues) ou à l'aide d'une roue libre (3 roues). Il est de même imposé de prévoir une protection de l'ensemble du robot, celle-ci démontable et donnant accès aux batteries.

La réalisation du projet a donc suivi plusieurs étapes. Tout d'abord, nous avons trouvé notre inspiration dans diverses sources. Le robot proposé par arduino et d'autres voitures suiveuses de ligne nous ont aidés à mieux visualiser nos futurs prototypes.

#### 3.1 Partie mécanique

##### 3.1.1 Conception du châssis

La forme générale du robot s'est définie sur deux étages pour une meilleure répartition du poids, une optimisation de la place et du cable management. Nous avons esquissé le projet sur papier pour définir l'image du premier prototype à réaliser puis la modélisation sur Creo, le logiciel de modélisation 3D, a commencé.

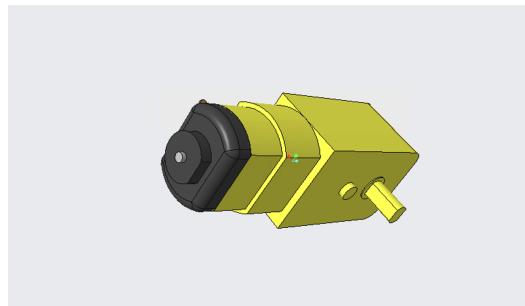


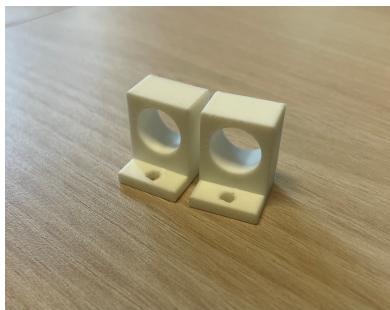
FIGURE 2 – Moteur Arduino

Le châssis a été réalisé en bois, découpé au laser pour une précision optimale, avec des trous et des encoches permettant la fixation et la rotation des roues. Il supporte et relie tous les éléments du robot, cette pièce est donc l'élément principal du prototype. La découpe laser se prête parfaitement au châssis car elle est réitérable et le bois de 3mm est à la fois solide et léger. Les esquisses des châssis du bas et du haut ont été modélisées à l'aide des dimensions exactes des composants. En effet, nous avons récupéré sur un forum, la modélisation des différents composants :

1. Le capteur BFD1000
2. Les moteurs Arduino
3. Carte arduino nano
4. Support des batteries
5. L298N

Ces modélisations obtenues nous ont permis de créer notre châssis avec des références précises.(Voir les plans du châssis en annexe). Celles-ci très utiles pour les trous dédiés aux entretoises, aux fixations du capteur, de la carte arduino, des moteurs...

Après avoir pris les mesures et esquissé les premiers croquis, quatre roues et des tiges de support permettant la répartition en deux étages pour le châssis ont été imprimées en 3D. De même que les supports moteurs qui garantissent une fixation stable des moteurs et une transmission efficace de l'effort. Cette approche est simple, efficace et intègre bien les composants.(Voir Annexe)



(a) Stabiliseurs des axes moteurs



(b) Tige support pour les deux châssis

FIGURE 3 – Quelques pièces modélisées de notre robot.

La conception du châssis a été réitérée plusieurs fois afin d'être parfaite pour notre prototype final. Le fait de monter le robot sur 2 étages a complexifié notre coordination de groupe. Notre principal défi a été de garantir des dimensions cohérentes entre chaque élément pour des fixations optimales. Le tout fait pour respecter la longueur maximale de 30 cm. Enfin, nous avons conçu différents châssis et assemblé les éléments pour tester et déterminer la meilleure configuration. L'application de la théorie nous a demandé de nombreux essais pour aboutir à un robot stable et maniable. Cette étape a donc exigé du temps et de la communication entre les membres du groupe.

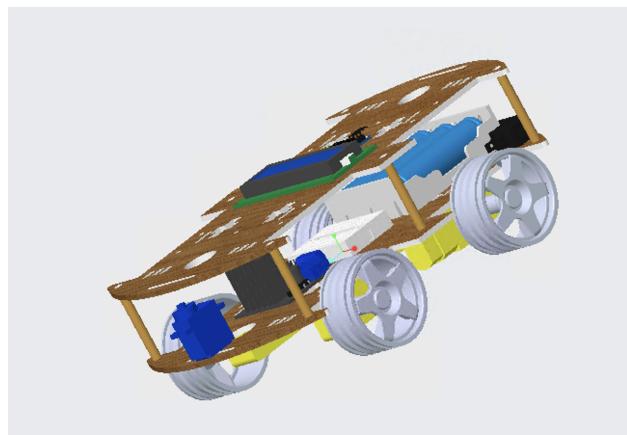


FIGURE 4 – Conception Creo du premier prototype.

### 3.1.2 Conception des roues

Dans un premier temps, nous avons utilisé 4 roues : deux roues motrices à l'arrière et deux roues libres à l'avant pour gérer le système de direction.

Les deux roues motrices, faites à l'imprimante 3D, ont été directement montées sur les arbres moteurs afin de recevoir sans effort leur rotation. Un stabilisateur a été placé entre l'arbre moteur et la roue pour maintenir les roues avant parallèles et réduire l'effort exercé sur l'arbre. L'encastrement doit être optimal pour que le robot ne tremble pas. Le robot ne répondant pas réactivement au code, les roues arrières sont ensuite devenues des jantes modélisées en 3D associées à des pneus récupérés. L'adhérence du robot s'est améliorée de même que ses trajectoires.

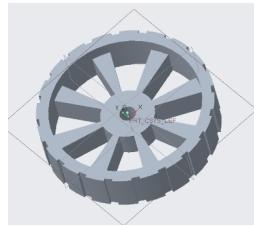


FIGURE 5 – Roues arrières de notre robot conçues sur Creo

Par ailleurs, les deux roues responsables du système de direction étaient reliées parallèlement grâce à une tige en bois. Elles ont d'abord fonctionné en tant que doubles roues libres puis allaient être intégrées au codage avec un servomoteur. Ce système complexe a engendré plusieurs problèmes ce qui nous a amené à enlever ces deux roues et les remplacer par un système plus simple. Ce modèle est pertinent mais aurait demandé trop d'essais et de temps pour être fonctionnel dans le temps imparti. De plus, il nous éloignait de l'objectif principal du projet, qui était de suivre une ligne noire le plus rapidement possible. Ainsi, nous avons opté pour une roue libre multidirectionnelle, d'où les changements de châssis fréquents.

### 3.1.3 Crédit du système de direction

Nous avons décidé pour le système de direction d'utiliser une tige et un servo-moteur. Celle-ci, étant fixée à deux supports d'essieux, permet la rotation des roues vers la droite et la gauche. Ce système de direction est composé des quatre éléments suivant :

1. Une tige directionnelle liant les deux essieux
2. Essieux mobiles en pivot sur le châssis
3. Vis pour fixer les tiges aux essieux
4. Vis passant au travers des roues et des essieux



FIGURE 6 – Visuel du système de direction avec tige et servomoteur

La longueur de la tige doit être parfaite pour le parallélisme des roues. Or avec les changements de châssis et de leurs dimensions, plusieurs tiges ont dû être découpées pour avoir un résultat concluant. Ce prototype de direction en 2D, a été installé dans un premier temps sur un châssis provisoire afin de déterminer les contraintes de jeu entre les différentes pièces, les efforts engendrés et la position du servomoteur. Ce système de direction, basé sur la rotation d'un clou et un jeu important entre les pièces, présentait des faiblesses, compromettant ainsi la stabilité et la fluidité des mouvements de notre robot.

Après analyse, nous avons pris la décision de simplifier notre modèle par crainte d'un système non fonctionnel. Certaines pièces n'étant pas optimisées pour les exigences du projet, nous avons choisi de concevoir un prototype simplifié, intégrant une roue libre unique à l'avant. Cette modification a eu un impact sur la conception de plusieurs éléments, notamment sur celle du châssis.



FIGURE 7 – Bille multidirectionnelle

Nous avons encastré la bille multidirectionnelle dans un support hexagonal vissé sur le châssis du bas. Nous avons réalisé un macaron de centrage sur la partie supérieure du châssis du bas afin de mettre en position et régler la hauteur de la bille par rapport au robot et ses capteurs. Il fallait un robot parallèle au sol qui ne pointe pas vers l'avant. La hauteur des roues a aussi été adaptée et le châssis est finalement parallèle au sol. En outre, nous avons obtenu une hauteur de robot de moins de 15mm du sol, ce qui permet au capteur BFD1000 de fonctionner de façon optimale.

---

### 3.1.4 Conception du carénage

Le carénage a été pensé pour être assez simple et pratique d'utilisation. Il offre une protection optimale tout en facilitant l'accès aux composants internes. À noter que nous avons dû imprimer le carénage en 3D, même si les pièces étaient simples, nous avons eu des problèmes avec la découpe laser. Notre carénage est entièrement démontable grâce à des encoches intégrées qui s'encastrent les unes dans les autres. Il est composé de cloisons de différentes tailles adaptées aux éléments à protéger. De plus, il a été important de penser aux trous permettant le passage de fils d'alimentation de la batterie. La coque est compact et facilite la prise en main du robot. Elle évite que les batteries et le driver L298N situés sur la châssis du haut ne prennent la poussière. Cette coque simple permet de personnaliser le robot en fonction de nos goûts sans changer le modèle final. Nous avons ajouté des fusées dans une idée de robot d'armement.

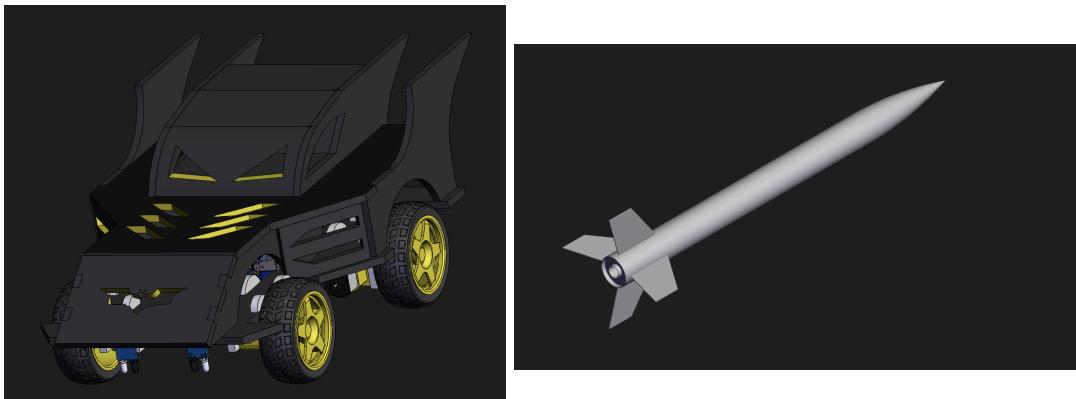


FIGURE 9 – Visuel de la fusée modélisée sur Creo

FIGURE 8 – Première idée de carénage

FIGURE 10 – Deux idées de design du robot

La première idée de carénage était orientée vers une structure plus complexe, mais après avoir constaté que trop de temps était perdu à cause du système de servomoteur, il a été décidé de prioriser le bon fonctionnement du robot. L'objectif était de simplifier la conception pour garantir une efficacité optimale.

## 3.2 Parties électronique et informatique

### 3.2.1 Les capteurs et branchements

La partie électronique représentait une nouveauté pour l'ensemble du groupe. Que ce soit au niveau programmation ou branchements. Nous nous sommes donc intéressés au fonctionnement du capteur et à sa sensibilité afin d'optimiser le déplacement du robot le long de la ligne. Le capteur est complexe et comporte cinq détecteurs à programmer. Leur position et les trous dans le châssis ont fait partie des enjeux majeurs du projet côté programmation et mécanique, car ils conditionnaient le bon fonctionnement du robot.

Le capteur BFD-1000 à l'origine utilisé en digital ne fonctionnait pas pour la majorité des groupes. En effet, il ne proposait pas une sensibilité suffisante pour différencier le blanc, le noir et le sol. De ce fait, nous l'avons modifié afin qu'il soit utilisé en analogique. Cela a permis de mieux étudier les différents paliers de couleurs et surtout de les distinguer, notamment grâce à l'utilisation d'un seuil que nous avons dû déterminer

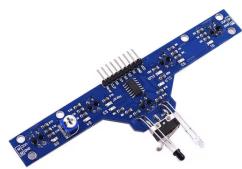


FIGURE 11 – Capteur BFD-1000

grâce à plusieurs tests de code.

### 3.2.2 Le logiciel et la Carte Arduino

Le logiciel Arduino présente une interface épurée et permet de développer facilement des programmes. Le langage utilisé par le logiciel est basé sur le C, auquel s'ajoutent des instructions spécifiques au langage Arduino. Il dispose d'une banque d'exemple très riche qui permet d'utiliser des morceaux de codes pré-écrits. Nous nous sommes formés à l'aide des cours d'initiation et de forums Arduino, où différents utilisateurs échangent sur la programmation. Aussi, chat Gpt nous a aidé à trouver les erreurs dans nos codes. Cela nous a permis d'aboutir à un code fonctionnel et concis.

### 3.2.3 L'algorithme

Un programme Arduino doit impérativement être divisé en 2 parties : une fonction setup et une fonction loop. Ces deux fonctions sont de type void, c'est à dire qu'elles ne peuvent pas prendre de valeurs. La fonction setup est la fonction qui se lance au début du programme. Elle initialise les variables et définit les broches de la carte Arduino qui seront utilisées. La fonction loop se lance après et lance le programme en boucle.

Le code commence par définir les broches pour les moteurs et les capteurs IR. Les broches ENA et ENB contrôlent la vitesse des moteurs gauche et droit via PWM, tandis que IN1, IN2, IN3, et IN4 déterminent la direction. Les broches A1 à A5 sont utilisées pour lire les états des cinq capteurs IR.

Dans la fonction setup(), les broches des moteurs sont définies en tant que sorties avec pinMode(), et les broches des capteurs sont définies comme entrées. Le branchement est à la fois en analogique et en digital. En effet, modifier le capteur a été nécessaire pour le projet donc les capteurs sont en analogique et les moteurs en digital.

La fonction loop() est utilisée pour lire les valeurs des capteurs avec analogRead(). Ces valeurs déterminent la réaction du robot :

Si le capteur central (capteur3) détecte la ligne, les moteurs avancent à une vitesse modérée avec la fonction avancer(). Il est prioritaire sur la détection des capteurs extrêmes, soient les capteurs 1 et 5. Si un capteur latéral détecte la ligne, le robot ajuste sa direction en tournant plus ou moins fort dans la direction opposée grâce aux fonctions gauche(), droite(), fortgauche() et fortdroite(). Si aucun capteur ne détecte la ligne, le robot cherche la ligne grâce à la fonction retrouverligne() qui enregistre la dernière action effectuée par le robot et lui applique le mouvement inverse à celle-ci.

La détection de la ligne s'effectue grâce à un seuil qui a été défini à l'aide de la fonction serialprint(). Cette fonction affiche les valeurs prises par les capteurs selon les couleurs. De ce fait, le seuil optimal que

---

nous avons défini est 270. Aussi, un seuil maximal est utilisé lorsque le robot est pris en main et ne détecte rien pour arrêter les moteurs.



FIGURE 12 – Fonctionnement simplifié du robot suiveur de ligne

Enfin, un délai de 10 millisecondes est introduit à la fin de chaque itération de la boucle `loop()` pour éviter des lectures trop rapides et permettre une gestion plus fluide des mouvements du robot. Il minimise les oscillations du robot.

Le plus gros challenge de la programmation du micro-controleur est de tester l'efficacité de son code. En effet, même si notre code source semble juste et compile, il est impossible de savoir s'il sera efficace. Le poids, la vitesse, la luminosité et l'adhérence influent sur les changements de direction du robot donc de nombreux tests sont utilisés pour changer de peu quelques valeurs cibles. Les essais permettent de cerner le problème afin de le résoudre au mieux. Lors de la première expérimentation, le robot ne semblait rien détecter et trouver la source du problème n'était pas évident (digital/analogique). Dans la deuxième partie du semestre, à l'inverse, le capteur détectait tout à toutes les hauteurs possibles, ce qui nous a posé des difficultés pour poser un seuil.

Le code final obtenu est correct, le robot suit la ligne mais se perd. Néanmoins, il la retrouve toujours et revient sur le circuit, certes parfois dans le sens inverse.

## 4 Difficultés rencontrées

Au cours du projet, plusieurs défis techniques et organisationnels ont été identifiés. Parfois, un manque de cohésion et de communication au sein de l'équipe a pu compliquer la coordination des tâches. L'absence de réunions claires pour définir un prototype commun a conduit à un travail autonome parfois désordonné, chaque membre se concentrant davantage sur ce qui lui plaisait plutôt que sur un objectif collectif. Cela a créé quelques obstacles.

### 4.1 Difficultés mécaniques

En ce qui concerne la synchronisation des moteurs, leur fonctionnement n'était pas parfaitement coordonné, ce qui compromettait la précision des déplacements du robot. Par ailleurs, il a été constaté que les premiers supports moteurs modélisés n'étaient pas adaptés aux modèles Arduino. Un jeu important s'est créé ce qui provoquait un glissement au sein du support. Cette instabilité affectait le robot en empêchant les roues d'être correctement entraînées.

Pour le changement de direction, différentes approches, avec et sans servomoteur, ont été explorées afin de garantir un changement de direction simultané et efficace. Pour y accéder, le système à bille a été ajouté à l'avant du châssis, ce qui a nécessité plusieurs modifications. Parmi ces adaptations, un trou a été réalisé pour permettre le passage de la bille, d'autres ont été ajoutés pour le passage des fils du capteur. Enfin, certains éléments de l'ancien système de direction, désormais abandonné, ont été retirés.

Par ailleurs, des problèmes liés à la fabrication des pièces ont émergé. Par exemple, imprimer des dimensions correctes pour les alésages est compliqué à cause de la dilatation du plastique lors de l'impression 3D. Cela a exigé un ajustement manuel, comme l'agrandissement des trous avec un foret pour plusieurs pièces.

Une dernière difficulté survenue vers la fin du semestre a été un problème sur le plan de conception des pièces du carénage. En effet, après les avoir modélisés sur créo et transférés sur le logiciel de découpe laser, les contours de nos pièces apparaissaient en double trait. Le problème était sans doute issu d'une mauvaise mise en plan initiale sur créo. Malgré la connaissance de la défaillance, nous ne sommes pas parvenus à la régler, ce qui nous a mené à imprimer les éléments du carénage par impression 3D. Bien que cette solution ne soit pas la plus adaptée, elle a permis de minimiser le poids du robot qui était déjà important grâce à la légèreté des pièces plastiques.

### 4.2 Difficultés informatiques

Nous avons rencontré de nombreuses difficultés lors de la création de notre code Arduino, notamment puisque il s'agissait du premier contact que nous avions avec cette interface. Étant novices en codage C, nous avons pris du temps pour comprendre le fonctionnement de ce langage et surtout les branchements pratiques associés. En effet, les branchements en PMW au début n'étaient pas bien compris et donc placé ce qui a induit une réponse aléatoire du robot. Le voltage de la batterie, au départ trop faible, ne nous permettait pas une vitesse moyenne du robot qui était alors trop rapide. Nous avons rapidement réussi à faire fonctionner le système en activant les moteurs dès que le capteur détectait un élément sombre. Ensuite, la gestion des capteurs IR s'est avérée complexe, notamment pour obtenir des lectures constantes des capteurs et éviter les comportements erratiques du robot.

Le suivi de ligne n'était pas optimal au début. Au lieu d'une réponse fluide et contrôlée aux variations de la ligne, le robot avait tendance à effectuer des mouvements brusques ou à perdre ses repères dès que plusieurs capteurs étaient activés simultanément. Nous avons dû ajuster les vitesses des moteurs pour rendre les changements de direction plus progressifs, en modifiant les paramètres de la fonction de contrôle des moteurs. Enfin, il a fallut gérer les pertes de repères du robot dans le but d'éviter qu'il s'arrête de manière inappropriée ou qu'il tourne indéfiniment. Tout devait être fait en prenant en compte le poids, la taille et le système de direction du robot. Ces paramètres influant fortement sur la réactivité du robot à l'injonction des moteurs et sur sa vitesse. Beaucoup d'heures ont été nécessaires (pendant et en dehors des heures de TM02). Nous avons dû réaliser de nombreux tests, qui aboutissaient rarement à un résultat satisfaisant. Les modifications ont été réalisées principalement sur les seuils, vitesse des moteurs, le système de correction et la différence de sensibilité des capteurs.

Ces diverses éléments ont fait que énormément de codes ont été écrit pour essayer de résoudre ces soucis. Des codes basiques avec 3 capteurs uniquement, d'autres avec cinq, et aussi des codes de correction proportionnelle ont été tentés sans succès. Nous avons finalement abouti à un code correct mais qui nécessite des améliorations pour éviter que le robot s'éloigne de la ligne.

### 4.3 Améliorations potentielles pour un projet plus long

Les objectifs de notre projet se concentrent sur plusieurs axes d'amélioration. Le premier concerne le câble management : il s'agit de trouver une disposition des câbles moins encombrante, en profitant de la structure à deux étages du système.

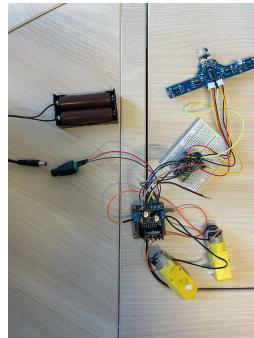


FIGURE 13 – Visuel du branchement sans cable management

Par ailleurs, le montage des roues nécessite des ajustements pour améliorer la stabilité et assurer un parallélisme correct. Cela passe par un serrage adéquat des écrous afin d'éliminer tout jeu excessif, qui pourrait compromettre la trajectoire en ligne droite.

Ensuite, l'amélioration du programme constitue une étape essentielle. Actuellement, le robot avance sans précision optimale dans le suivi de la ligne noire. Il est donc nécessaire de développer un programme permettant un suivi plus rigoureux et réactif. Un autre enjeu clé réside dans l'optimisation de la rapidité. Il est indispensable d'ajuster le programme pour répondre aux contraintes du parcours, tout en prenant en compte l'interaction entre le code et la conception mécanique. En effet, le poids du robot a un impact direct sur sa vitesse, et cette dernière peut être régulée via le programme. Notre prototype étant relativement lourd, le

défi a consisté à trouver un équilibre entre vitesse et précision. Enfin, nous devons gérer l'entrée perturbatrice créée par les changements de direction : l'objectif est de corriger la trajectoire chaque fois que le robot s'écarte de la ligne sans toutefois la perdre. Ces améliorations visent finalement à optimiser la performance et la fiabilité de notre robot.

Pour finir, il serait aussi possible d'améliorer le carénage de notre robot qui protège uniquement la partie supérieure du châssis mais laisse la partie inférieure découverte

## 5 Robots suiveurs de ligne dans la vie quotidienne

Les robots suiveurs de ligne ne servent pas uniquement aux compétitions, ils sont aussi utilisés dans l'industrie ou dans les transports en commun.

### 5.1 Le TEOR

Le TEOR (Transport est-ouest rouennais) de Rouen est un réseau de bus à haut niveau de service. Ces bus utilisent un système de guidage optique lors de leur entrée en station. Ce ne sont pas des bus sans chauffeurs comme dans certaines agglomérations, mais à l'approche d'un arrêt, le chauffeur laisse la main au système, qui permet au bus un accostage précis et régulier de la station, ce qui facilite l'accessibilité des personnes à mobilité réduite. Le système fonctionne de la manière suivante : – une caméra placée derrière ou au dessus du pare-brise lit un marquage au sol composé de deux lignes pointillées – un ordinateur de bord analyse la position du bus par rapport à la voie et ajuste la position de celui-ci si nécessaire. Ce système apporte une grande fiabilité à l'arrivée en station. En effet, le risque de panne du guidage optique est très faible (99,98 %).



FIGURE 14 – Visuel du Transport est-ouest rouennais

### 5.2 Les véhicules à guidage automatique (VGA)

Le véhicule à guidage automatique est une version plus développée du robot suiveur de ligne mais repose sur le même principe. Une de leur application en industrie est de pouvoir transporter de lourdes marchandises d'un point A à un point B et ce de manière rapide et sûre. Pour ce faire le véhicule a besoin de se repérer dans son environnement afin de se déplacer correctement. Il existe plusieurs technologies permettant au véhicule de se repérer :

1. **Le filoguidage** : Les AGV se déplacent par détection d'une piste, tracée dans ou sur le sol. Il peut s'agir de fil émetteur d'ondes enfoui, de rail métallique au sol ou de fil électrique noyé au sol. Le signal transmis dans la piste est détecté par le VGA qui le suivra comme s'il s'agissait d'une voie ferrée.
2. **L'optoguidage** : Le système optoguidage permet au VGA de suivre une ligne peinte au sol grâce à des caméras embarquées. Il s'agit du même principe que le filoguidage mais avec une complexité et un coût d'installation moindres, car il n'est pas nécessaire de réaliser des travaux de gros œuvre.



FIGURE 15 – Visuel du VGA adapté au monde industriel

### 5.3 Autres exemples

D'une manière plus primitive, on peut assimiler certains jouets à des robots auto-guidés. Prenons l'exemple des trains de modélisation ou bien des voitures de courses miniatures. Ces répliques se déplacent sur un circuit mais qui est lui « matériel » dans le sens où le véhicule est guidé grâce aux force de frottement qui sont appliqués quand le véhicule est en mouvement. Cet exemple peut sembler anodins mais des applications de plus grand envergure voient le jour. On peut par exemple citer la ligne 14 du métro parisien qui est entièrement automatisée selon ce même principe.



FIGURE 16 – Visuel de la ligne 14 du métro parisien

## 6 Conclusion et Perspectives

La mise en œuvre de ce projet dans le cadre de l'UV TM02 a été une expérience très enrichissante tant sur le plan technique, que personnel notamment en ce qui concerne le travail en équipe. Nous avons eu la chance de manipuler différentes machines et outils, telles que des imprimantes 3D BambuLab ou des machines de découpe Laser. Grâce à cette expérience, nous avons élaboré et mis en œuvre un projet complet, ce qui nous a ouvert les portes des divers aspects de la gestion de projet. Étant donné la longueur de nos impressions 3D, il fallait planifier des horaires pendant le semestre pour imprimer en dehors des heures de cours. Il était donc indispensable de s'organiser et de planifier les tâches à accomplir pour la prochaine séance. Pour un projet réalisé en 4 mois, nous avons pris conscience de la grande quantité d'améliorations potentielles et de la difficulté de travailler en équipe dans des projets de cette ampleur. Il est évident que l'adaptabilité et l'ingéniosité sont essentielles dans un projet pour trouver des solutions satisfaisantes aux problèmes rencontrés. Ces aptitudes constituent les compétences majeures pour occuper le poste d'ingénieur. Enfin, ce projet expose les défis auxquels nous , futurs ingénieurs, seront confrontés au quotidien.

## 7 Annexes

### 7.1 Annexe A - Plans de Conceptions

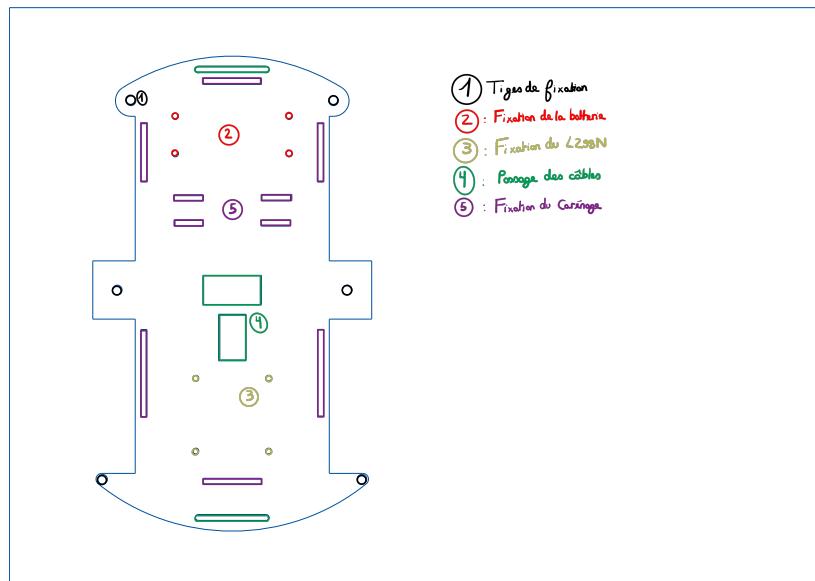


FIGURE 17 – Visuel du châssis du haut et de la distribution des pièces

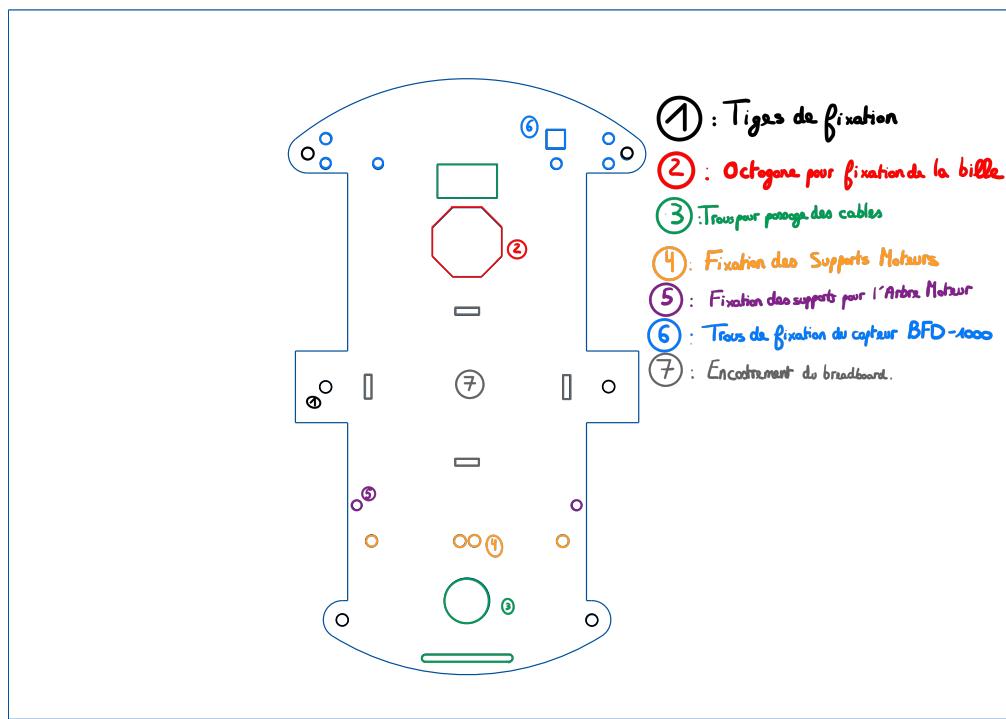


FIGURE 18 – Visuel du châssis du bas et de la distribution des pièces

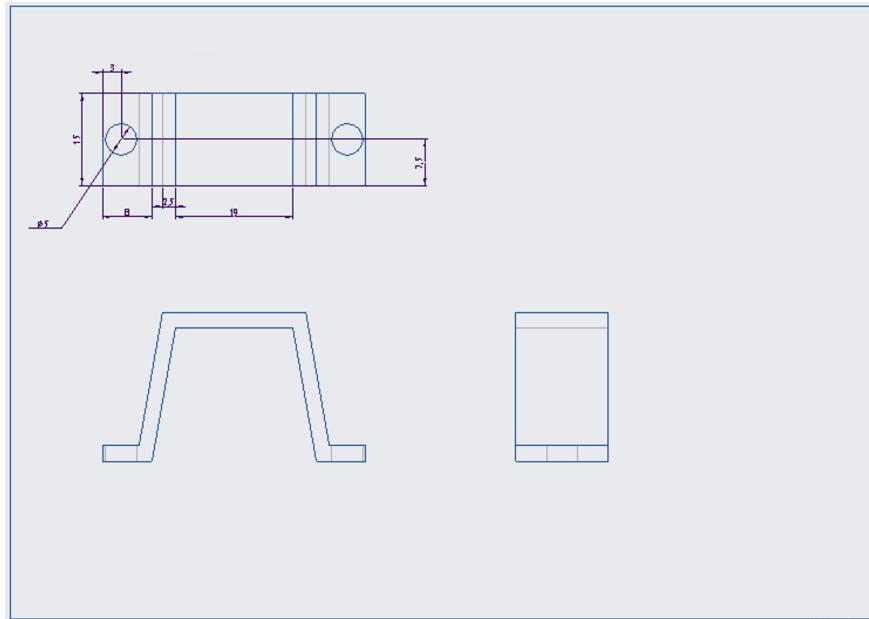


FIGURE 19 – Plans A3 sur Creo des supports moteurs

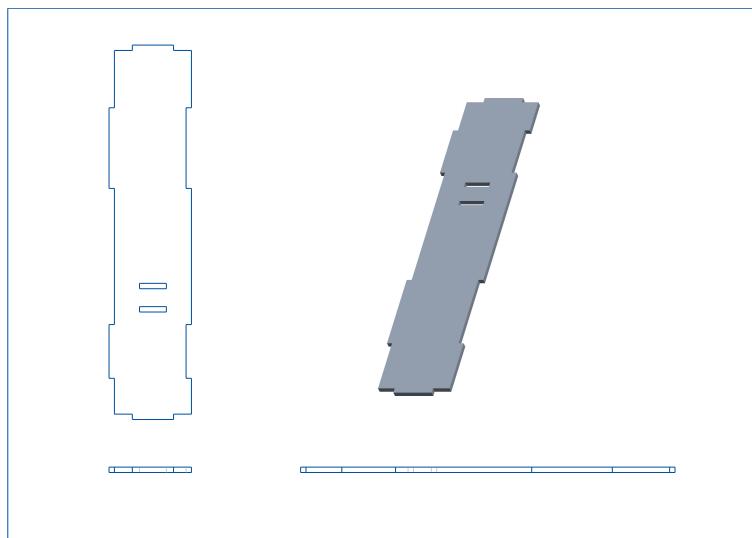


FIGURE 20 – Plans A3 sur Creo du mur transversal du carénage

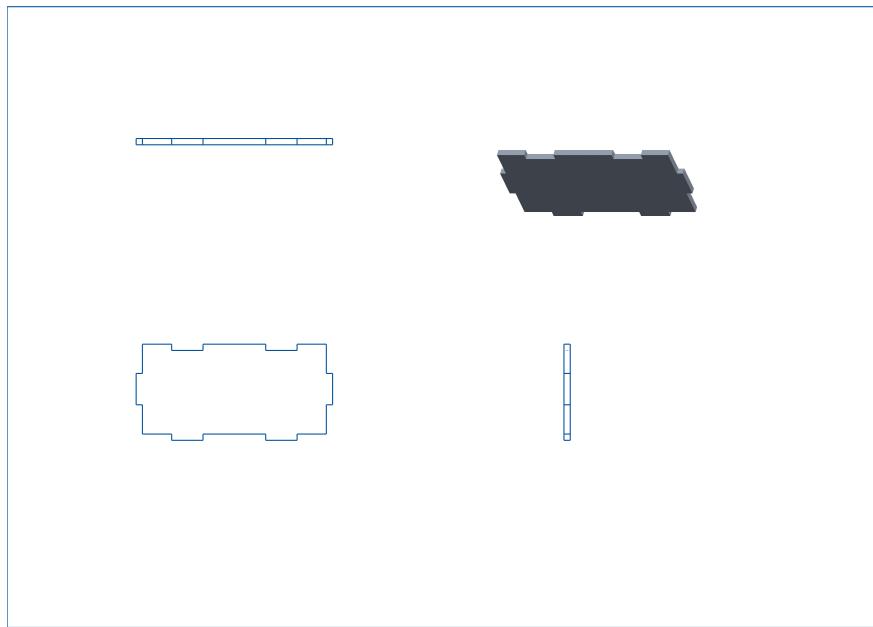


FIGURE 21 – Plans A3 sur Creo du couvercle supérieur du carénage

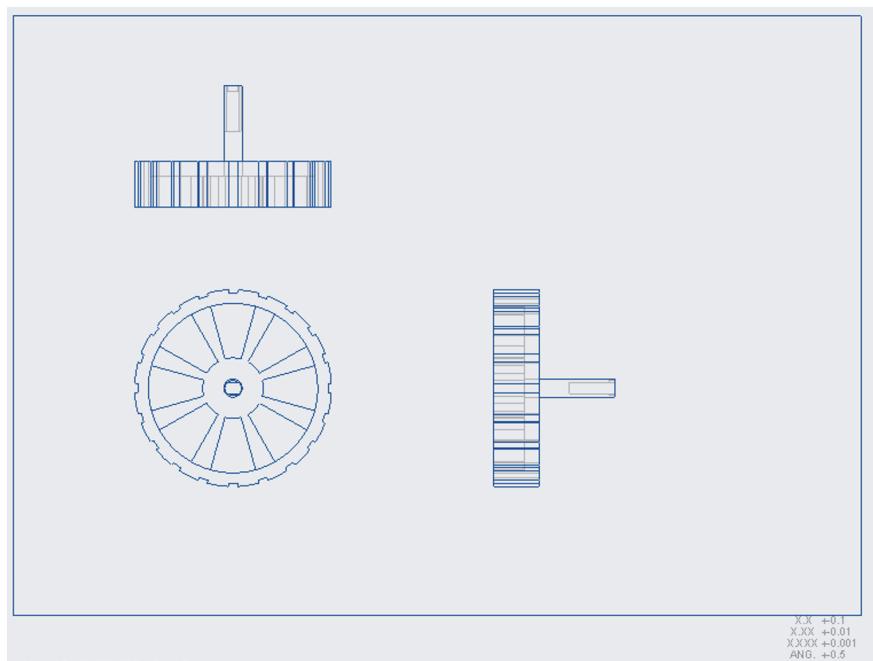


FIGURE 22 – Plans A3 sur Creo des roues arrière

---

## 7.2 Annexe B - Code final

```
int IN1 = 3;
int IN2 = 4;
int ENA = 10 ; // broches relatives au moteur droit
int IN3 = 5;
int IN4 = 6;
int ENB = 11; //broches relatives au moteur gauche

// initialisation des capteurs
int ir1 = A1; // capteur extrême gauche
int ir2 = A2; // capteur gauche
int ir3 = A3; // capteur central
int ir4 = A4; // capteur droit
int ir5 = A5; // capteur extrême droit

int x; // variable uqui enregistre les actions effectuées
const int seuil = 200; //seuil à partir duquel les capteurs détectent le noir
const int seuilmax = 600; //seuil à partir duquel les capteurs détectent le vide
int capteur1, capteur2, capteur3, capteur4, capteur5;
//capteur1 = extrême gauche, capteur2 = gauche, capteur3 =centre, capteur4 = droite,
capteur5 = extrême droite

// initialisation des variables, sens des broches et des moteurs
void setup() {
    pinMode(ENA, OUTPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);

    pinMode(ir1, INPUT);
    pinMode(ir2, INPUT);
    pinMode(ir3, INPUT);
    pinMode(ir4, INPUT);
    pinMode(ir5, INPUT);

    digitalWrite(ENA, LOW);
    digitalWrite(ENB, LOW);
```

---

```
Serial.begin(9600); // permet de lire les seuils des capteurs}

void loop() {
    // lecture des valeurs recueillies par les capteurs
    capteur1=analogRead(ir1);
    capteur2=analogRead(ir2);
    capteur3=analogRead(ir3);
    capteur4=analogRead(ir4);
    capteur5=analogRead(ir5);

    // affichage des valeurs sur l'ordinateur
    Serial.print("capteur1 :");
    Serial.print(capteur1);
    Serial.print("capteur2 :");
    Serial.print(capteur2);
    Serial.print("capteur3 :");
    Serial.print(capteur3);
    Serial.print("capteur4 :");
    Serial.print(capteur4);
    Serial.print("capteur5 :");
    Serial.print(capteur5);

    // boucle principale
    if (capteur1 >= seuilmax && capteur2 >= seuilmax && capteur3 >= seuilmax &&
        capteur4 >= seuilmax && capteur5 >= seuilmax) {
        // si le capteur est tenu dans les mains et ne détecte que le vide, arrêt des moteurs
        avancer(0, 0);
        x = 0;
    } else if (((capteur3 >= seuil) || ((capteur3 >= seuil) && (capteur5 >= seuil))) ||
               ((capteur3 >= seuil) && (capteur5 >= seuil))) {
        // si le capteur central détecte la ligne, il est prioritaire sur les capteurs extrêmes
        avancer(90, 90);
        x = 1;
    } else if (capteur2 >= seuil) {
        //le capteur de gauche détecte la ligne : le robot tourne à gauche
        gauche();
        x = 2;
    } else if ((capteur1 >= seuil) && (capteur3 <= seuil) && (capteur4 <= seuil)) {
        //le capteur extrême gauche détecte la ligne (et les capteurs centraux non): le robot
        tourne fort à gauche
        fortgauche();
        x = 22;
    }
}
```

---

```
    } else if (capteur4 >= seuil) {
        //le capteur de droite détecte la ligne : le robot tourne à droite
        droite();
        x = 3;
    } else if ((capteur5 >= seuil) && (capteur3 <= seuil) && (capteur2 <= seuil)) {
        //le capteur extrême droite détecte la ligne (et les capteurs centraux non) : le robot
        tourne fort à droite
        fortdroite();
        x = 33;
    } else {
        // si le robot est perdu et les capteurs ne détectent rien alors il tourne pour retrouver la
        ligne
        retrouverLigne(x);
    }
    delay(10);
}

//fonction qui permet d'avancer ou de s'arrêter en fonction des vitesses entrées en variable
void avancer(int vitessemoteurG, int vitessemoteurD) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA, vitessemoteurD);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, vitessemoteurG);
}

//tourner doucement à droite
void droite() {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA, 90);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, 70);
}

//tourner doucement à gauche
void gauche() {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
```

---

```
analogWrite(ENA, 70);
digitalWrite(IN3, HIGH);
digitalWrite(IN4, LOW);
analogWrite(ENB, 90);
}

//tourner fortement à droite
void fortdroite() {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA, 90);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, 30);
}

//tourner fortement à gauche
void fortgauche() {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA, 30);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, 90);
}

//lorsque que le robot est perdu, la dernière action effectuée par le robot est enregistrée
//et le mouvement inverse est ensuite appliquée au robot
void retrouverLigne(int x) {
    if (x == 2) {
        fortgauche();
    } else if (x == 22) {
        fortgauche();
    } else if (x == 3) {
        fortdroite();
    } else if (x == 33) {
        fortdroite();
    } else { // s'il a avancé ou rien détecté, il tourne en rond à droite
        fortdroite();
    }
}
```

---

