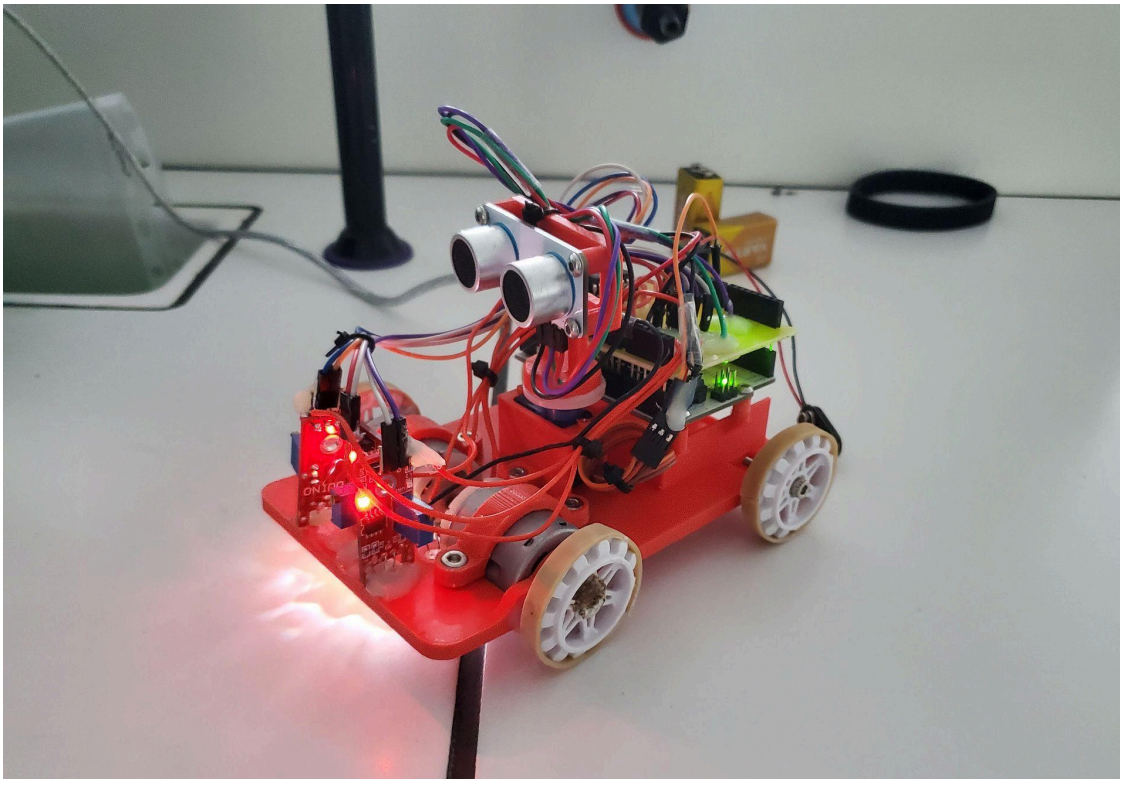
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Projet de Physique P6

STPI/P6/2024 – n°19

ROBOT AUTONOME : DE LA CONCEPTION À L'ÉVITEMENT D'OBSTACLES



Étudiants :

Jade BROCHARD

Stanislas MARIE

Adam BOUFOUS

Noé CAMUS

Quentin MEULEMAN

Romann BONHOMME

Enseignant-responsable du projet :

Hind LAGHMARA

Date de remise du rapport : ***13/06/2024***

Référence du projet : ***STPI/P6/2024 – n°19***

Intitulé du projet : ***Robot autonome : de la conception à l'évitement d'obstacles***

Type de projet : ***Expérimental***

Objectifs du projet :

Concevoir un robot autonome sur la base d'une carte Arduino Uno. Le robot devra être capable de suivre une ligne noire tracée sur le sol mais également de détecter les obstacles se trouvant sur le trajet afin de les éviter et de retrouver la ligne noire située derrière ces obstacles.

Afin d'y parvenir nous devrons choisir les composants adéquats au projet, réaliser un circuit imprimé, modéliser le châssis, programmer l'Arduino Uno et enfin assembler les composants et réaliser les soudures nécessaires.

Mots-clefs du projet : ***Electronique, programmation, conception.***

Remerciements

Pour réaliser ce projet, nous avons eu besoin de l'aide de plusieurs membres de l'équipe pédagogique de l'INSA ainsi que de son personnel.

Tout d'abord, nous tenons à remercier Hind Laghmara, l'enseignante responsable de notre projet, pour nous avoir encadré et aidé à réaliser ce projet. Ses conseils et remarques nous ont permis d'avancer et de nous débloquer dans des moments compliqués.

Nous souhaitons également adresser nos remerciements à Pascal Williams qui nous a conseillé et nous a donné l'accès à la salle de projet, ce qui nous a été fortement utile. De plus, c'est grâce à lui que nous avons pu imprimer le circuit électronique.

Enfin, nous remercions l'INSA pour nous avoir donné le matériel nécessaire au projet mais aussi l'opportunité de réaliser ce dernier.

Table des matières

1. Introduction

Dans un monde de plus en plus orienté vers l'industrialisation, l'automatisation et l'intelligence artificielle, les robots jouent un rôle essentiel dans la réalisation de tâches industrielles et domestiques.

Dans le cadre de nos projets scientifiques encadrés, nous avons entrepris de créer un robot suiveur de ligne. Ce projet avait pour objectif de nous faire acquérir les compétences nécessaires à la réalisation d'un robot, incluant la programmation, l'électronique, la conception de circuits imprimés, et la modélisation.

Nous avons choisi ce projet parce que nous partageons tous une passion pour l'informatique et la mécanique, et nous sommes particulièrement intéressés par les défis qu'il présente. Ainsi, nous avons décidé de concevoir un robot autonome capable de suivre une ligne et de détecter et éviter les obstacles.

Dans un premier temps, nous aborderons la méthodologie suivie pour mener à bien ce projet ainsi que la répartition des tâches au sein de l'équipe. Ensuite, nous expliquerons le processus de réalisation du robot, en détaillant les étapes suivantes : le choix des composants, la programmation, la fabrication du circuit imprimé, la modélisation du châssis, et enfin l'assemblage du robot. Pour conclure, nous décrirons la phase de test du robot et les difficultés que nous avons rencontrées.

Ce projet nous a permis de plonger dans le monde fascinant de la robotique et de l'automatisation, nous préparant ainsi à relever les défis auxquels nous pourrions être confrontés en tant qu'ingénieur.

2. Méthodologique / Organisation du travail

Pour le bon déroulement du projet, nous nous sommes répartis le travail sous forme de tâches à effectuer, le tableau ci-dessous présente la répartition des tâches définie en début de projet. Ainsi nous avons pu travailler en parallèle et se coordonner avec tous les membres du groupe en communiquant sur nos avancées.

Répartition initiale des tâches					
Quentin	Adam	Romann	Jade	Stanislas	Noé
Circuit imprimé (tinkercad et fritzing)	Programmation Arduino (partie motrice)	Programmation Arduino	Programmation (suivi de ligne)	Circuit imprimé (tinkercad) et programmation	Modélisation 3D du chassis

Organigramme des tâches réalisées						
	Stanislas	Romann	Quentin	Noé	Adam	Jade
Choix des composants			X	X		
Programmation	X	X			X	X
Circuit Imprimé	X		X			
Modélisation				X		
Assemblage (soudures)				X		
Tests		X			X	
Ecriture du rapport	X	X	X	X	X	X

3. Réalisation du robot

3.1. Choix des composants

Avant de commencer la réalisation du robot, nous avons dû choisir des composants électroniques. En effet, notre robot doit être capable de suivre une ligne noire et d'éviter des obstacles, il était donc nécessaire de sélectionner des composants nous permettant de réaliser ces tâches. Seul le choix de la carte programmable n'était pas libre, il nous a été imposé d'utiliser un Arduino Uno. Cette carte est simple à programmer et permet la réalisation de projets électroniques divers grâce à ces différentes entrées analogiques et digitales.

A. Partie motrice

Dans un premier temps nous avons dû nous pencher sur les composants liés à la propulsion du robot. Nous avons choisi des moteurs à courant continu alimentés en 9V. L'avantage de choisir 2 moteurs de ce types est que l'on peut gérer leur vitesse séparément par le biais d'un pont en H..

Il s'agit d'un composant électronique qui reçoit des signaux de notre Arduino et qui permet de délivrer une tension différente et variable à chacun des deux moteurs. Étant branché à une pin 5V de l'arduino, le pont en H est alimenté séparément de la pile 9V réservée aux moteurs.

Dans un deuxième temps, il a fallu nous pencher sur le choix du module de roulement. Nous avons fini par opter sur un système de chenille pour avoir une transmission entre les roues avants et arrières et faciliter la direction.

B. Partie détection

Afin de suivre la ligne noire placée au sol, nous avons choisi de prendre trois capteurs de lumières disposés à l'avant du robot. Ces capteurs envoient des signaux analogiques à notre carte en fonction de la présence ou non de lumière. De cette façon, nous avons pu le calibrer de manière à ce qu'il soit capable de détecter le noir et le blanc.

Pour l'évitement d'obstacles, nous nous sommes dotés d'un capteur ultrason permettant de relever la distance entre le capteur et l'objet à éviter face à lui. Pour finir, dans l'optique de faire tourner notre capteur ultrason sur lui-même afin de détecter les obstacles autour de notre robot, nous avons choisi de prendre un servomoteur.

Vous trouverez ci-dessous, un tableau résumant l'ensemble des composants nécessaires à la réalisation de notre projet ainsi que leur prix et leur quantité.

Composant	Lien	Prix	Quantité	Prix total
Moteur courant continu	https://www.gotronic.fr/art-moteur-mot3-788.htm	4,5	2	9
Servo-moteur	https://www.gotronic.fr/art-servomoteur-ft90r-36020.htm	6,2	1	6,2
Capteur ultrason	https://www.gotronic.fr/art-capteur-a-ultrasons-wpse306-26096.htm	6,9	1	6,9
Capteur lumineux	https://www.gotronic.fr/art-capteur-d-intensite-lumineuse-gt541-36072.htm	2,8	3	8,4
Pile 9V	https://www.gotronic.fr/art-pile-alcaline-9v-varta-19146.htm	3,2	2	6,4
Connecteur pile 9V	https://www.gotronic.fr/art-connecteur-eco-9v-en-i-5696.htm	0,3	2	0,6
Jeu de roues et chenilles	https://www.gotronic.fr/art-jeu-de-chenilles-30-dents-roues-3035-27651.htm	16,9	1	16,9
Led	https://www.gotronic.fr/art-led-5-mm-l5fwc-2213.htm	0,55	2	1,1
Pont en H (L293D)	https://www.gotronic.fr/art-l293d-14072.htm	6,9	1	6,9
Arduino Uno	https://www.gotronic.fr/art-arduino-uno-a000066-12420.htm	23,9	1	23,9

Figure 1 : Tableau listant tous les composants nécessaires à la réalisation du projet

Pour conclure cette première sous-partie, nous aimerions souligner le fait que la plupart des composants qui nous ont été fournis n'étaient pas exactement ceux que nous avions demandés. Ainsi, même si nous avons conscience que faire une nouvelle commande pour chaque projet chaque année est impossible et constituerait un énorme budget, composer avec le matériel des projets précédents peut générer des problèmes et des difficultés. Nous y reviendront dans la partie dédiée.

3.2. Programmation du robot

Dans ce projet, la partie liée à la programmation représente une part extrêmement importante de la réalisation du robot. En effet, ce sont les codes que nous allons détailler dans cette partie qui feront le lien entre tous les composants, et qui garantiront l'autonomie à notre robot.

A. Bibliothèques et variables

La bibliothèque 'Servo.h' est utilisée pour contrôler le servo-moteur. La fonction 'setup()' a pour but d'initialiser les broches comme sorties ou entrées. Le servomoteur est attaché à la broche 12 et elle est configurée en sortie.

Les broches utilisées sont les suivantes :

Composant	Fonction	Broche
Moteur A	ENA (PWM)	6
	IN1	5
	IN2	8
Moteur B	ENB (PWM)	11
	IN3	10
	IN4	4
	TRIG_PIN	2
	ECHO_PIN	3
Servo Moteur	SERVO_PIN	12
Capteurs lumineux	capteurGauche	A2
	capteurCentre	A1
	capteurDroitœ	A3

B. Broches initialisées en sortie

- Broches de contrôle des moteurs (IN1, IN2, IN3, IN4):

Ces broches sont utilisées pour contrôler la direction des moteurs. Elles envoient des signaux logiques (HIGH ou LOW) permettant de contrôler le sens de rotation des moteurs ou de les arrêter.

- Broches PWM pour les moteurs (ENA, ENB):

Les broches ENA et ENB sont utilisées pour contrôler la vitesse des moteurs via des signaux PWM (Pulse Width Modulation). Grâce aux signaux envoyés, on arrive à contrôler la vitesse des moteurs.

- Broche de déclenchement (TRIG_PIN):

La broche de déclenchement du capteur à ultrasons (TRIG_PIN) est initialisée en sortie pour envoyer des impulsions ultrasoniques.

C. Broches initialisées en entrée

- Capteur de lumière:

Nous avons choisi d'utiliser des pins analogiques pour les capteurs lumineux plutôt que des numériques en raison de la disponibilité limitée des pins numériques sur la carte Arduino. La polyvalence des pins analogiques pouvant lire des valeurs numériques nous a donc orientée vers ce choix. Cela nous a ainsi permis de simplifier le câblage et de libérer des pins numériques pour d'autres composants.

- Capteur à ultrasons:

La broche d'écho du capteur à ultrasons (ECHO_PIN) est initialisée en entrée pour recevoir les impulsions réfléchies.

D. Boucle principale 'loop()':

Le principe de la boucle principale est de vérifier si le capteur central détecte la ligne noire et s'il n'y a pas d'obstacle. Si c'est le cas, le robot va continuer d'avancer en suivant la ligne noire. Sinon, (dans le cas où un obstacle est détecté), la boucle permet de l'éviter et de retrouver la ligne noire. La boucle principale fait appel à différentes fonctions qui permettent le bon fonctionnement du code.

E. Fonctions

- Fonctions de mouvements:

Le robot avance grâce à deux moteurs que l'on contrôle avec plusieurs fonctions (moteurAvancer, moteurAgauche, moteurAdroite, moteurArret) qui reposent toutes sur le même principe. Nous agissons sur les broches IN1, IN2, IN3 et IN4 pour définir la direction des moteurs ou les arrêter, et sur les broches PWM pour adapter la vitesse des moteurs (par exemple, on fera tourner le moteur B plus rapidement que le moteur A pour tourner à gauche).

- Fonctions de capteur:

Nous avons besoin de détecter d'une part les obstacles et de l'autre la ligne noire. La fonction 'detecterObjet()' se repose sur le capteur ultrason pour mesurer la distance d'un obstacle. Elle retourne '1' si un obstacle est détecté à moins de 10 cm, sinon '0'. Pour cette partie, on lit seulement les valeurs que les capteurs de lumière renvoient ('1' si la ligne noire est détectée sinon '0'). On utilise 3 capteurs différents ce qui nous permet de couvrir une zone plus large.

- Fonctions de comportement:

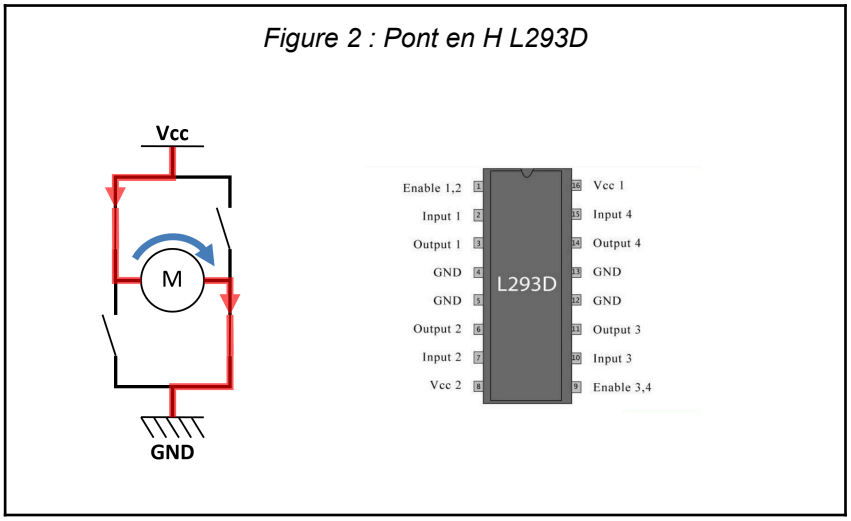
La fonction 'suiviedeligne()' utilise les informations renvoyées par les capteurs de lumières pour faire tourner le robot à gauche, à droite ou continuer tout droit afin de suivre la ligne noire. La fonction 'eviterObstacle()' permet de balayer la zone avec l'aide du servo-moteur pour trouver une direction sans obstacle et diriger le robot vers cette direction.

Enfin la fonction ‘rechercherLigneNoire()’ permet au robot de retrouver la ligne noire après avoir évité l’obstacle.

- Gestion des moteurs

Pour contrôler au mieux nos moteurs, nous nous sommes servis d’un pont en H. Ce composant électronique permet de contrôler la direction et la vitesse des moteurs à courant continu (DC) en changeant la polarité du courant et en utilisant des signaux PWM pour moduler la vitesse. Il utilise quatre interrupteurs pour diriger le courant, permettant au moteur de tourner dans les deux sens. Dans notre projet, le pont en H gère les moteurs en ajustant la direction via des broches de commande et la vitesse via des signaux PWM, tout en protégeant la partie propulsion des courts-circuits.

Figure 2 : Pont en H L293D



- Programme d'évitement d’obstacle

Pour cette partie on utilise la fonction detecterObjet qui, à intervalle régulier, indique si un obstacle est présent à l’aide du capteur d’ultrason.

Lorsqu'un obstacle est détecté à moins de 10 cm la fonction detecterObjet renvoie la valeur 1 (Vrai) si le capteur détecte un obstacle et 0 (Faux) sinon.

Lorsque que le robot détecte un obstacle, le servomoteur sur lequel se trouve le capteur ultrason tourne de 90° dans le sens anti-horaire en faisant des angles de 10° jusqu'à ne plus capter d’obstacle.

Si il continue de capter un obstacle alors il revient à sa position initiale pour scanner dans le sens-horaire.

Au moment où le capteur ne capte plus d’obstacle, le robot tourne dans la direction où pointe le capteur, le servomoteur tourne pour remettre le capteur face au robot puis il se remet à avancer.

Dans le cas où il est dans une impasse, le robot fait demi-tour et repart.

- Programme de suivi de ligne

Lorsque le mode suivi de ligne est activé, les capteurs lumineux détectent s’il y a du noir ou du blanc sur la surface du sol. Le robot va alors avancer tout droit ou tourner à gauche ou à droite en suivant la table de vérité suivante :

Capteur centre	Capteur gauche	Capteur droit	État
1	1	1	A
1	1	0	TG
1	0	1	TD
1	0	0	A
0	1	1	A
0	1	0	TG
0	0	1	TD
0	0	0	A

Figure 3 : Table de vérité du programme de suivi de ligne
Légende : A : avancer tout droit , TG : tourner à gauche , TD : tourner à droite

3.3. Réalisation d'un circuit imprimé

Nous allons ici détailler l'intérêt du circuit imprimé ainsi que la démarche que nous avons mis en place pour sa conception et son impression.

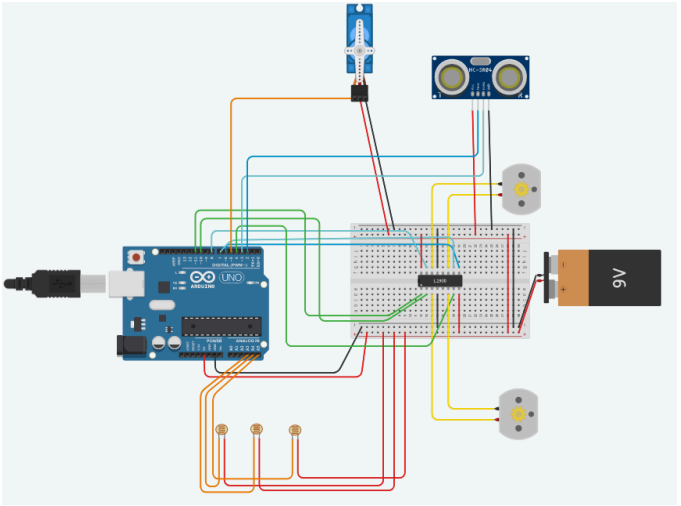
D'une part, même si le nombre de composants est limité et que les branchements ne sont pas d'une grande complexité, nous tenions à réaliser un circuit imprimé. En effet, avoir des pistes en cuivre de 1mm d'épaisseur gravées sur la carte plutôt que des dizaines de fils se mêlant et bougeant librement sur le robot est, d'abord, beaucoup plus pratique mais également plus esthétique. L'apparence de l'ensemble est bien plus propre et épurée. Néanmoins, il faut garder à l'esprit qu'une fois les pistes imprimées et les soudures faites, un changement est difficilement envisageable, c'est le seul point négatif de cette démarche.

D'autre part, réaliser ce circuit imprimé nous a permis d'acquérir de nouvelles compétences fondamentales en électronique, en soudage et en conception sur Fritzing. En réalité, il était surtout prétexte à nous faire découvrir ces aspects. En effet, pour beaucoup d'entre nous, c'était la première fois que nous soudions ou que nous utilisions des logiciels de conception de carte électronique tels que Fritzing.

A. Conception

Premièrement, nous avons réalisé une version en simulation sur tinkercad afin d'avoir une première ébauche des branchements à réaliser, des composants à brancher, et des ports auxquels les connecter.

Figure 3: Première version des branchements sur Tinkercad.



En effet, suivant les capteurs, les moteurs, ou encore les alimentations, les ports à utiliser sur la carte arduino sont différents :

- Le capteur d'ultrasons doit être branché sur un port numérique.
- Le servomoteur a besoin d'un port numérique PWM.
- Le pont en H assurant la gestion des moteurs , est partagé entre numérique, PWM et input en 9V et 5V.
- Nos capteurs de lumières nécessitent quant à eux des ports analogiques.

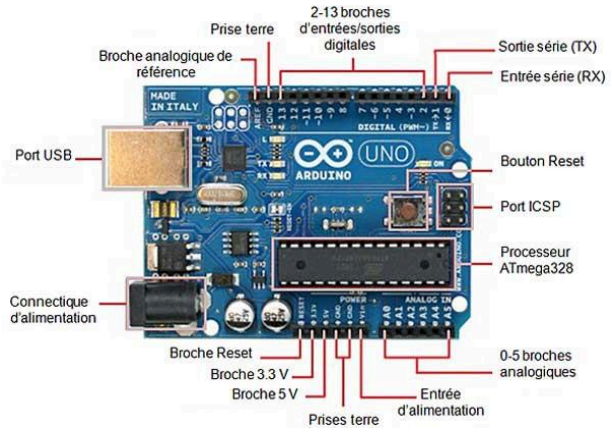


Figure 4 : Liste des ports sur arduino UNO

Deuxièmement, nous avons porté cette version sur fritzing afin de modéliser le circuit électronique. Cependant, pour des raisons techniques, les ports de la version tinkercad ne sont pas les mêmes que le fritzing. En effet, les pistes de cuivre ne peuvent se chevaucher, cette contrainte nous a donc forcé à faire des choix de ports dans le but d'optimiser la surface de la carte. Évidemment, ces choix ont été pris en collaboration avec les membres du groupe principalement chargés du code et de l'automatisation.

Pour concevoir ce fritzing, il a fallu prendre en compte d'une part le fait que le circuit soit retourné sur la carte arduino et donc que les positions des ports soient inversées et d'autre part que le pont en H soit placé sur le dessus du circuit. En effet, le circuit imprimé est retourné sur la carte arduino uno, qui est quant à elle, est placée à l'endroit sur le robot.

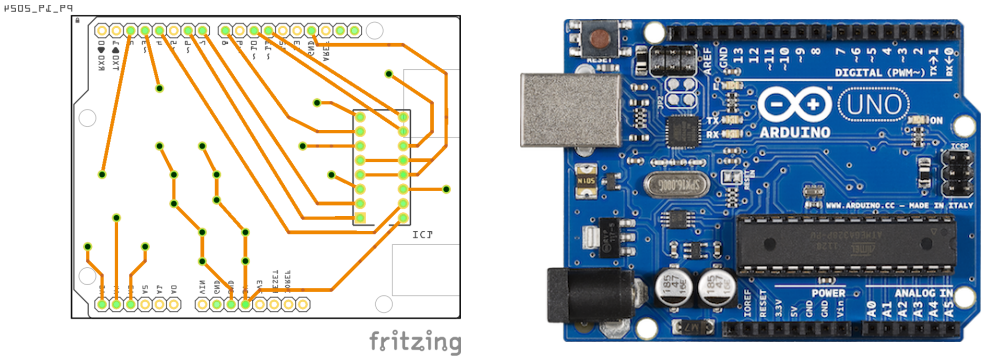
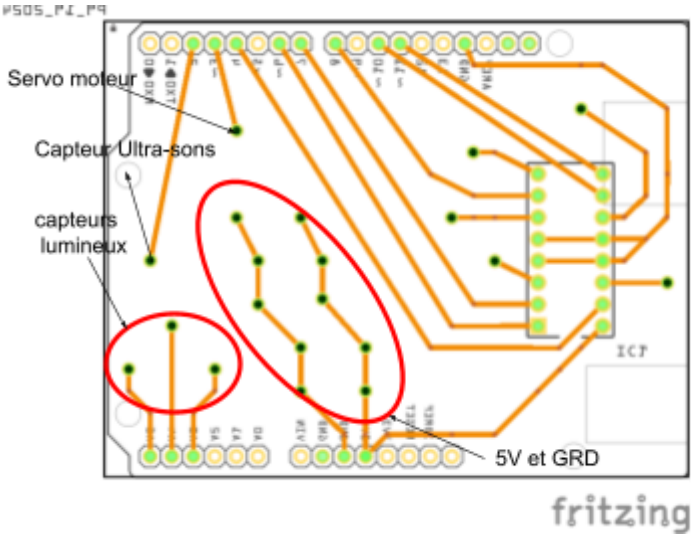


Figure 5 : A gauche : vue du bas du circuit imprimé. A droite : vue du haut de la carte arduino UNO. En bas : vue agrandie et détaillée du circuit



Sur cette carte, on distingue quatre grandes parties :

- <u>Branchements du pont en H :</u>

  Les 4 pistes en cuivre débouchant sur une jonction (trou) sont réservées aux input/output moteurs ainsi qu'aux alimentation 5V (pour alimenter le composant) et 9V (alimentation des moteurs). On viendra souder les câbles des moteurs et des alimentations sur ces jonctions.

  Les 4 pins centrales reliées entre elles par deux pistes en cuivre sont directement reliées au GND.

  Le reste des pistes sont orientées vers des pins numériques ou PWM de la carte arduino.

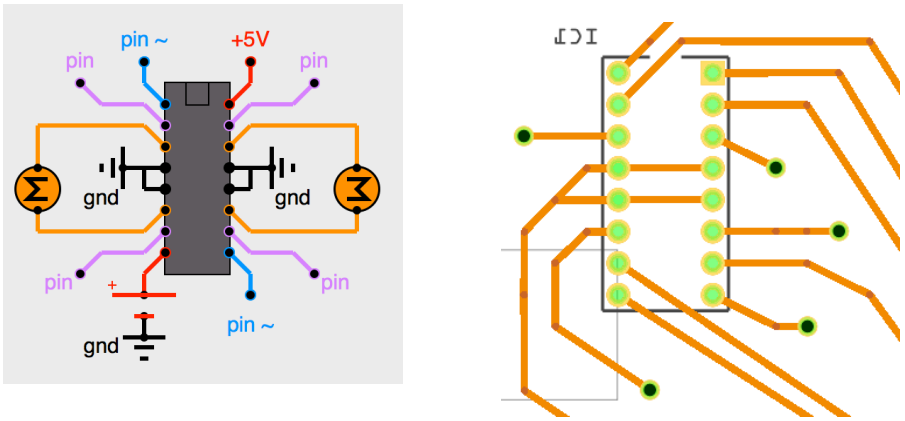


Figure 6 : inventaire des ports du pont en H

- Branchements GND et 5V :

  Cette partie concerne les 5 pistes partant du 5V et les 5 pistes partant du GND en bas du circuit. C'est eux qui vont assurer une alimentation et une prise terre aux 3 capteurs lumineux, au capteur son et au servo moteur. Comme pour le pont en H, les câbles seront soudés sur les jonctions (en noir) où s'arrêtent les pistes en cuivre.

- Branchement des capteurs lumineux :

  À la gauche des 5 pistes précédentes, se trouvent les 3 pistes reliées aux pins analogiques pour les 3 capteurs lumineux, c'est grâce à celles-ci que la carte UNO reçoit les informations.

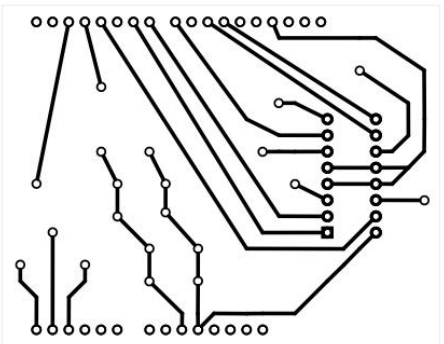
- Branchements du servo-moteur et du capteur ultra-sons :

  Enfin, les 2 pistes en haut à gauche sont celles qui relieront le servo moteur et le capteur ultra-sons à leurs pins respectives sur la carte arduino UNO.

B. Impression

Premièrement, nous avons dû exporter le fichier fritzing en un PDF que l'imprimante pouvait déchiffrer. En effet, les machines dont disposent l'INSA ne peuvent lire directement le fichier fritzing.

Figure 7 : PDF vue de bas
Circuit imprimé



Deuxièmement, nous avons envoyé ce PDF à Pascal William qui nous l'a gentiment imprimé, puis, après avoir été le chercher, nous avons réalisé l'ensemble des soudures des câbles des composants sur les points de jonctions (trous).

3.4. Modélisation du châssis

Dans cette partie nous nous attarderons à la conception du châssis de notre robot. Le châssis est la pièce servant de support à tous les composants du robot. Cette pièce doit donc respecter les dimensions de chacun des composants afin de les accueillir, être suffisamment résistante pour supporter le mouvement du robot mais également permettre au robot de réaliser les fonctions attendues. Afin d'avoir un résultat le plus fonctionnel possible, nous avons décidé de réaliser cette pièce via impression 3D. Nous allons donc voir dans cette partie toutes les étapes réalisées durant la conception du châssis de notre robot.

A. Relevé des mesures des composants

Afin d'imprimer le châssis via une imprimante 3D, il est d'abord nécessaire de modéliser cette pièce sur un logiciel de modélisation 3D, dans notre cas nous avons utilisé le logiciel SolidWorks. Toutefois, la première étape d'une réalisation 3D est la prise de mesure. Afin de relever toutes les dimensions de chacun des composants constituant le robot nous avons utilisé un pied à coulisse, cet outil permet de mesurer une distance avec une précision très élevée (dizaine de microns).

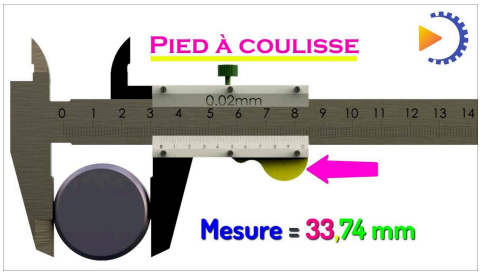


Figure 8 : Illustration d'utilisation d'un pieds à coulisse

B. Première modélisations et impressions 3D

Après avoir relevé toutes les cotes de nos composants, nous pouvons passer à une première étape de modélisation 3D sur SolidWorks. Cette partie consiste à modéliser les pièces sur-mesure servant de support à chaque composant. Il est donc impératif de s'être assuré au préalable que les mesures relevées sont les plus précises possibles. Afin de fixer chaque composant sur son support imprimé en 3D nous avons utilisé des vis à tête hexagonale creuse M4x10mm ainsi que d'écrous M4. Il est donc nécessaire de prévoir des emplacements pour ces vis et écrous dans nos modélisations. Vous trouverez ci-dessous quelques exemples de support individuel pour nos composants électroniques.

Après avoir modélisé les premiers supports, vient l'étape des premières impressions 3D. Cette étape est nécessaire car elle permet de vérifier que les supports coïncident avec les composants mais également qu'ils sont assez résistants. Voici quelques photos des premiers supports imprimés.

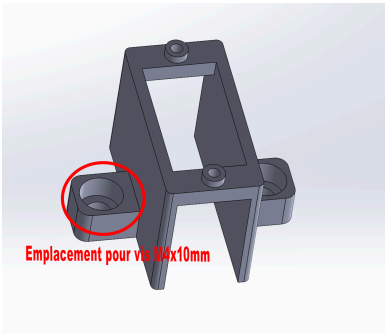


Figure 9 : Modèle 3D du support pour le servomoteur

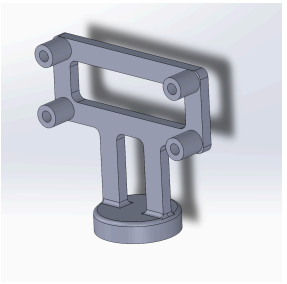


Figure 10: Modèle 3D du support pour le capteur ultrason

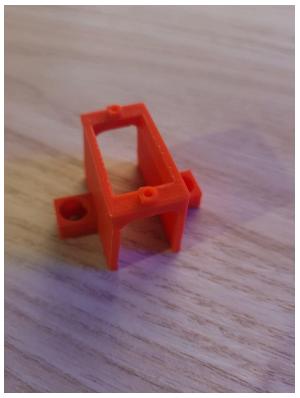


Figure 11 : Support pour le servomoteur imprimé en 3D

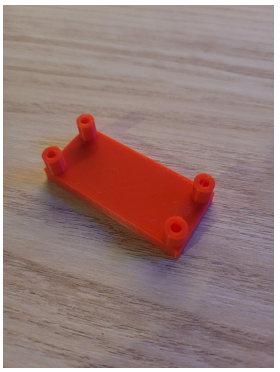


Figure 12 : Support pour le capteur ultrason imprimé en 3D

C. Modélisation du châssis et impressions 3D

Après avoir vérifié que tous les supports individuels étaient aux bonnes dimensions, nous avons ensuite modélisé le châssis sur SolidWorks. Pour ce faire, nous avons respecter plusieurs conditions :

- placement des capteurs de lumière et LED à l'avant du robot
- placement du servomoteur au centre du robot
- placement du capteur ultrason sur le servomoteur

- placement des moteurs à l'avant et des piles 9V à l'arrière afin d'équilibrer le robot
- respect de l'entraxe des roues déterminé par la longueur des chenilles
- prise en compte des contraintes mécanique afin de modéliser un châssis résistant

De la même manière que pour les supports individuels, cette première version du châssis a été imprimée en 3D puis testée. Nous avons décelé plusieurs axes de progression sur ce modèle. Premièrement certaines parties n'étaient pas assez résistantes et ont cédées sous la contrainte mécanique comme vous pouvez le constater ci-dessous.

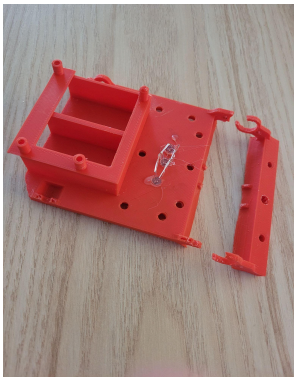
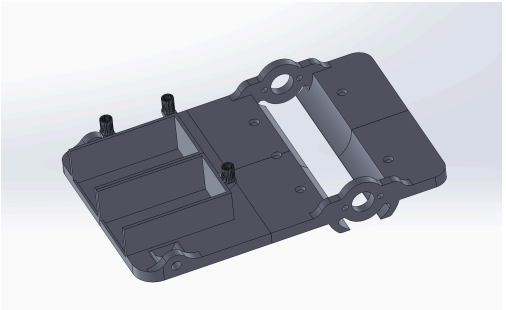


Figure 13 : Première version du châssis du robot imprimée en 3D

Nous avons donc effectué plusieurs modifications telles qu'un redimensionnement des pièces ayant cassé, une suppression de matière inutile, un ajout de congé sur les quatres coins du châssis permettant aux chenilles de coulisser sans se coincer, un allongement de l'avant du châssis nécessaire au positionnement des capteurs de lumière et des LED et enfin une augmentation de la densité de la pièce qui est passé de 20% de remplissage de matière à 40% pour s'assurer d'avoir un châssis suffisamment résistant. Vous trouverez ci-dessous une photo de la modélisation finale du châssis.

Figure 14 : Modèle 3D de la version finale du châssis du robot



3.5. Assemblage du robot

La dernière étape de la réalisation du robot est l'assemblage final. Cette étape s'est déroulée en deux parties, premièrement nous avons monté les composants sur le châssis puis nous avons réalisé toutes les soudures et branchements électroniques.

Le montage des composants sur le châssis n'a pas été la partie la plus compliquée étant donné que le châssis a été conçu dans l'optique de correspondre parfaitement aux dimensions des composants. En effet, le seul usage d'une clé Allen a été suffisant pour monter l'intégralité des composants électroniques sur le châssis. Le montage des moteurs et des roues n'était cependant pas aussi simple. L'arbre des moteurs étant de section circulaire nous avons dû le poncer afin de permettre la transmission d'un couple aux roues. De plus, les roues motrices ne comportaient pas de moyen de fixation, c'est pour cette raison que nous avons fait fondre le centre des roues motrices afin de les enfoncer sur l'arbre poncé de notre moteur.

La deuxième partie, quant à elle, a été bien plus délicate. Tout d'abord nous avons dû souder les broches sur notre circuit imprimé. Pour ce faire, nous avons utilisé un fer à souder ainsi que de l'étain après quoi nous avons vérifié que toutes nos soudures étaient bonnes grâce à un multimètre. Par la suite, nous avons soudé l'intégralité de nos composants électroniques à notre circuit imprimé prévu à cet effet. Cette étape nous a demandé une grande précision étant donné la taille des fiches du circuit imprimé. Vous trouverez ci-dessous une photo de notre robot après avoir été entièrement assemblé.

4. Test du robot

4.1. Test Programme avant l'assemblage

Dans un premier temps, nous avons utilisé le programme de modélisation 3D en ligne tinkercard pour écrire et tester notre code. Cela nous a permis d'apporter des modifications sur la logique de notre programme.

Ensuite, nous avons utilisé la platine d'essai pour connecter et tester tous les composants avec arduino. Au fur et à mesure des tests nous avons dû ajuster notre programme, mais également le circuit électrique en ajoutant de nouvelles pins. En effet, les composants à notre disposition n'étaient pas exactement les mêmes que ceux sur tinkercard comme le capteur ultrason et les capteurs lumineux qui avaient des sorties différentes.

Après avoir fait les modifications et tests nécessaires, nous avons pu observer que le capteur ultrason, les moteurs et les capteurs lumières fonctionnaient correctement. Cependant, nous avons dû faire quelques réglages supplémentaires sur le servo moteur, car il tournait trop vite ce qui allait perturber la prise d'information par le capteur ultrason. De plus, en s'échangeant les composants pour faire nos tests, nous avons remarqué qu'il était nécessaire de calibrer les capteurs lumières à chaque fois que l'on changeait d'endroit pour qu'ils captent au mieux les différences de contrastes.

4.2. Test après l'assemblage et problèmes rencontrés

Nous avons poursuivi les tests une fois le robot complètement assemblé. Tout d'abord, les capteurs et le servomoteur fonctionnaient correctement. Néanmoins, dès les premières expérimentations, nous avons pu distinguer plusieurs problèmes. Le plus important étant que l'un des moteurs n'arrivait pas à tourner tandis que l'autre ne générait pas assez de puissance pour faire bouger le robot.

4.3. Solutions apportées

Pour rechercher d'où provenait le problème lié au moteur, nous avons dans un premier temps regardé si le problème ne venait pas du moteur lui-même, en l'alimentant directement à l'aide d'un générateur. Après quoi, nous avons supposé que le problème provenait soit du pont en H, soit de la carte arduino. Quand nous avons essayé de changer le pont en H, nous nous sommes rendus compte que la tension délivrée par la carte arduino au pin 5 volts était trop faible. Nous avons donc également changé de carte et cela a fonctionné, les moteurs tournent normalement et permettent de faire avancer le robot.

Les tests suivants nous ont permis d'observer que les moteurs n'étaient pas assez puissants pour faire tourner les chenilles. Par manque de temps, pour remédier à ce problème, il a fallu les retirer, ce qui empêche le robot de tourner. Ceci nous a malheureusement empêché de tester le suivi de lignes en conditions réelles.

Dès lors, nous nous sommes focalisés sur l'évitement d'obstacle et avons modifié plusieurs fois la distance de détection d'obstacle pour arriver à une valeur de 1 mètre. Nous avons estimé cette distance suffisante pour que le robot ait le temps de déclencher la fonction marche arrière qui lui permet de freiner plus vite. Toutefois, n'étant pas capable de tourner notre robot se contente de ne pas heurter d'obstacle.

5. Conclusions et perspectives

Le domaine de la robotique étant très étendu, il est difficile de l'appréhender dans sa globalité. C'est pourquoi ce projet scientifique encadré nous a été bénéfique par la découverte d'une partie de ce monde vaste à travers la création du robot. En effet, en travaillant sur l'ensemble des étapes de production, nous avons pu nous rendre compte de la complexité d'un tel projet.

Tout d'abord, avant de choisir les composants, il faut avoir une idée précise de comment le robot va fonctionner et quelles fonctionnalités nous voulons lui donner, une fois cette étape de conception terminée, le choix des composants se fait aisément.

Ensuite, les étapes de programmation sur un simulateur, de réalisation du circuit imprimé et de modélisation du châssis se font simultanément et présentent chacune leurs difficultés.

L'assemblage du robot a été une partie très intéressante et instructive car elle nous a permis de nous initier au soudage. En effet, tout le groupe a pu participer au soudage du circuit imprimé.

Enfin, tester le robot est la phase qui a été la plus éprouvante car beaucoup de problèmes se sont montrés alors qu'il ne restait plus beaucoup de temps pour finir le projet.

Nous avons alors fait de notre mieux pour résoudre ces problèmes un à un, cependant, nous n'avons malheureusement pas réussi à résoudre tous les problèmes et à faire fonctionner correctement le robot car un grand nombre d'interférences se créent entre les composants à l'assemblage du robot et empêchent ce dernier de déployer toutes ces capacités. Cela peut être dû à la qualité d'impression du circuit imprimé.

Si nous avions eu plus de temps, nous aurions aimé compléter ce projet, notamment en faisant fonctionner notre robot automatiquement. De plus, il aurait pu être intéressant de lui ajouter des fonctionnalités comme une intelligence artificielle pour reconnaître des objets (une caméra aurait été installée à l'avant du robot et ce dernier aurait dû retrouver un objet par la reconnaissance à travers la caméra).

6. BIBLIOGRAPHIE

Pour en apprendre davantage sur l'électronique pratique, le fonctionnement de la carte arduino uno ou encore pour avoir accès à la documentation des composants utilisés, nous nous sommes référés aux sources citées ci-dessous :

[1]https://www.tutoriel-arduino.com/piloter-deux-moteurs-dc-avec-un-l293d/

Stéphane ROBERT,2019,Comment piloter 2 moteur Dc avec un L293D et un arduino

[2]https://www.tutoriel-arduino.com/calculer-une-distance-avec-un-capteur-ultrason-hc-sr04/,

Stéphane ROBERT,2018,Comment calculer une distance à l'aide d'un capteur à ultrason HC-SR04 et d'un arduino

[3]https://fritzing.org/media/uploads/learning/translations/FRITZING1_Construire_une_maquette.pdf

University of Applied Sciences Potsda, PRISE EN MAIN DE FRITZING

[4]https://www.robotique.tech/tutoriel/construction-dun-robot-commande-par-arduino-qui-detecte-et-evite-les-obstacles/

Par Robotique, Construction d'un robot commandé par Arduino qui détecte et évite les obstacles.

[5]https://docs.arduino.cc/

Par Arduino, Documentation de la carte et du langage.

[6]https://forum.arduino.cc/t/robot-100-pour-100-autonome/1079614

Par forum Arduino, discussion sur la conception d'un robot autonome.

[7]http://stssnsb.free.fr/telecharger/blusson/cours/cours_pontenH.pdf

Par BTS SN-CIEL Sembat, documentation du pont en H

7. ANNEXES

7.1. LISTINGS DES PROGRAMMES RÉALISÉS

A) PROGRAMME PRINCIPAL

Code principale de base:

```
#include <Servo.h>

// Définition des broches pour le contrôle du moteur
const int ENA = 6;  // Broche PWM pour le moteur A
const int IN1 = 5;  // Broche de commande 1 du moteur A
const int IN2 = 8;  // Broche de commande 2 du moteur A
const int ENB = 11; // Broche PWM pour le moteur B
const int IN3 = 10; // Broche de commande 1 du moteur B
const int IN4 = 4;  // Broche de commande 2 du moteur B
// Capteur lumineux
const int capteurGauche = A5;
const int capteurCentre = A4;
const int capteurDroit = A3;
// Récepteur électromagnétique
const int TRIG_PIN = 2;
const int ECHO_PIN = 13;

Servo monServo;  // Crée un objet Servo pour contrôler le servo-moteur

bool etatCapteurGauche;
bool etatCapteurCentre;
bool etatCapteurDroit;

int angles;

void setup() {
 // Initialisation des broches du moteur en tant que sortie
 Serial.begin(2400);
 pinMode(ENA, OUTPUT);
 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(ENB, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);
 // Capteur lumière
 pinMode(capteurGauche, INPUT);
 pinMode(capteurCentre, INPUT);
 pinMode(capteurDroit, INPUT);
 // Servomoteur
 pinMode(TRIG_PIN, OUTPUT);
 pinMode(ECHO_PIN, INPUT);
 monServo.attach(3);
}
void loop() {
 // Lecture des capteurs de ligne
 etatCapteurGauche = digitalRead(capteurGauche);
 etatCapteurCentre = digitalRead(capteurCentre);
 etatCapteurDroit = digitalRead(capteurDroit);

 if (etatCapteurCentre && !detecterObjet()) { // Si le capteur du centre détecte du noir
  suiviedeLigne();
 } else { // Si le capteur du centre détecte du blanc ou un obstacle est détecté
  if (detecterObjet()) { // Si un obstacle est détecté
   // Éviter l'obstacle en tournant le servo-moteur
   eviterObstacle(angles);
  } else { // Si aucun obstacle n'est détecté mais que la ligne noire n'est pas détectée
   rechercherLigneNoire(angles);
  }
 }
}

int detecterObjet() {
 long duree, distance;

 digitalWrite(TRIG_PIN, LOW);
 delayMicroseconds(2);
 digitalWrite(TRIG_PIN, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIG_PIN, LOW);

 duree = pulseIn(ECHO_PIN, HIGH);
 distance = (duree / 2) / 29.1; // Conversion en centimètres

 if (distance <= 100) {
  return 1; // Objet à moins de 100 cm
 } else {
  return 0; // Pas d'objet à moins de 100 cm
 }
}

void moteurAgauche() {
 // Moteur A
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
 analogWrite(ENA, 200); // Vitesse PWM

 // Moteur B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, LOW);
 analogWrite(ENB, 100); // Vitesse PWM
}

void moteurAdroite() {
 // Moteur A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, LOW);
 analogWrite(ENA, 100); // Vitesse PWM

 // Moteur B
```

```
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, HIGH);
 analogWrite(ENB, 200); // Vitesse PWM
}

void moteurAvancer() {
 // Moteur A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, LOW);
 analogWrite(ENA, 200); // Vitesse PWM

 // Moteur B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, LOW);
 analogWrite(ENB, 200); // Vitesse PWM
}

void moteurArret() {
 // Moteur A
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, LOW);
 analogWrite(ENA, 0); // Vitesse PWM

 // Moteur B
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, LOW);
 analogWrite(ENB, 0); // Vitesse PWM
}
void moteurReculer() {
 // Moteur A
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
 analogWrite(ENA, 200); // Vitesse PWM
 // Moteur B
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, HIGH);
 analogWrite(ENB, 200); // Vitesse PWM
}

void suiviedeLigne() {
 if (etatCapteurGauche && !etatCapteurDroit) { // S'il y a du noir à gauche et du blanc à droite, tourner à gauche
   moteurAgauche();
 } else if (!etatCapteurGauche && etatCapteurDroit) { // S'il y a du blanc à gauche et du noir à droite, tourner à droite
   moteurAdroite();
 } else { // Si les conditions plus haut ne s'appliquent pas, continuer tout droit
   moteurAvancer();
 }
}

void eviterObstacle(int &angles) {
 int angle;
 // Balayer l'environnement à l'aide du servo-moteur pour rechercher un espace libre
 for (angle = 0; angle <= 180; angle += 10) { // Balayer de 0 à 180 degrés par incréments de 10 degrés
   angles = angle;
   monServo.write(angle); // Positionner le servo-moteur à l'angle actuel
   delay(500); // Attendre un bref moment pour que le servo-moteur se stabilise
   // Vérifier la distance dans la direction actuelle
   if (!detecterObjet()) { // Si aucun obstacle n'est détecté dans cette direction
    // Tourner le robot dans la direction du servo-moteur
    tournerVersDirection(angle);
    return; // Sortir de la fonction
   }
 }
 angles = angle;
 monServo.write(90);
}

void tournerVersDirection(int angle) {
 // Convertir l'angle du servo-moteur en mouvement de direction pour le robot
 if (angle < 90) {
  // Tourner vers la gauche
  moteurAgauche();
 } else {
  // Tourner vers la droite
  moteurAdroite();
 }
 delay(1000);
}

void rechercherLigneNoire(int angles) {
 if (angles < 90) {
  while (!etatCapteurCentre) {
   moteurAdroite();
   etatCapteurCentre = digitalRead(capteurCentre);
  }
  suiviedeLigne();
 } else {
  while (!etatCapteurCentre) {
   moteurAgauche();
   etatCapteurCentre = digitalRead(capteurCentre);
  }
  suiviedeLigne(); }
}
```

B)PROGRAMME DE TEST

Programme test des moteurs:

```
// Définition des broches pour le contrôle du moteur
const int ENA = 6;  // Broche PWM pour le moteur A
const int IN1 = 7;  // Broche de commande 1 du moteur A
const int IN2 = 8;  // Broche de commande 2 du moteur A
const int ENB = 11; // Broche PWM pour le moteur B
const int IN3 = 10; // Broche de commande 1 du moteur B
const int IN4 = 4;  // Broche de commande 2 du moteur B

void setup() {
  // Initialisation des broches du moteur en tant que sortie
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  Serial.begin(9600); // Initialiser la communication série pour le débogage

  moteurAvancer(); // Faire avancer les moteurs au démarrage
}

void loop() {
  // Le code principal reste vide car les moteurs sont contrôlés dans la fonction setup
}

void moteurAvancer() {
  // Moteur A
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 200); // Vitesse PWM

  // Moteur B
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, 200); // Vitesse PWM
}
```

Programme avec Ultrason:

```
// Définition des broches pour le contrôle du moteur
const int ENA = 6;  // Broche PWM pour le moteur A
const int IN1 = 5;  // Broche de commande 1 du moteur A
const int IN2 = 8;  // Broche de commande 2 du moteur A
const int ENB = 11; // Broche PWM pour le moteur B
const int IN3 = 10; // Broche de commande 1 du moteur B
const int IN4 = 4;  // Broche de commande 2 du moteur B

// Récepteur électromagnétique
const int TRIG_PIN = 2;
const int ECHO_PIN = 3;

void setup() {
  // Initialisation des broches du moteur en tant que sortie
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  // Initialisation des broches du capteur ultrason
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  Serial.begin(9600); // Initialiser la communication série pour le débogage
}

void loop() {
  if (!detecterObjet()) {
          moteurAvancer();
  } else {
          moteurArret();
  }
  delay(100); // Petite pause pour éviter les rebonds de capteur
}

int detecterObjet() {
  long duration, distance;

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH);
  distance = (duration / 2) / 29.1; // Conversion en centimètres

  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance <= 20) { // Objet à moins de 20 cm
          return 1;
  } else {
          return 0;
  }
}

void moteurAvancer() {
  // Moteur A
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 200); // Vitesse PWM

  // Moteur B
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, 200); // Vitesse PWM
}

void moteurArret() {
  // Moteur A
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 0); // Vitesse PWM

  // Moteur B
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, 0); // Vitesse PWM
}
```

Programme Ultrason et capteurs Infrarouge:

1

```
// Définition des broches pour le contrôle du moteur
const int ENA = 6;  // Broche PWM pour le moteur A
const int IN1 = 5;  // Broche de commande 1 du moteur A
const int IN2 = 8;  // Broche de commande 2 du moteur A
const int ENB = 11; // Broche PWM pour le moteur B
const int IN3 = 10; // Broche de commande 1 du moteur B
const int IN4 = 4;  // Broche de commande 2 du moteur B

// Capteurs lumineux
const int capteurGauche = A5;
const int capteurCentre = A4;
const int capteurDroit = A3;

// Récepteur électromagnétique
const int TRIG_PIN = 2;
const int ECHO_PIN = 13;

void setup() {
  // Initialisation des broches du moteur en tant que sortie
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  // Initialisation des broches du capteur ultrason
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Initialisation des broches des capteurs lumineux
  pinMode(capteurGauche, INPUT);
  pinMode(capteurCentre, INPUT);
  pinMode(capteurDroit, INPUT);

  Serial.begin(9600); // Initialiser la communication série pour le débogage
}

void loop() {
  if (!detecterObjet()) {
    suivreLigne();
  } else {
    moteurArret();
  }
  delay(100); // Petite pause pour éviter les rebonds de capteur
```

2

```
}

int detecterObjet() {
  long duration, distance;

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH);
  distance = (duration / 2) / 29.1; // Conversion en centimètres

  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance <= 20) { // Objet à moins de 20 cm
    return 1;
  } else {
    return 0;
  }
}

void suivreLigne() {
  int etatCapteurGauche = digitalRead(capteurGauche);
  int etatCapteurCentre = digitalRead(capteurCentre);
  int etatCapteurDroit = digitalRead(capteurDroit);

  // Si la ligne est détectée sur le côté gauche (noir)
  if ((!etatCapteurGauche && etatCapteurCentre && etatCapteurDroit) ||
(!etatCapteurGauche && !etatCapteurCentre && etatCapteurDroit)) {
    // Réduire la vitesse du moteur gauche pour tourner à gauche
    moteurAvancer(200, 150); // Vitesse du moteur gauche à 200, vitesse du moteur droit à
150
  }
  // Si la ligne est détectée au centre (noir)
  else if ((etatCapteurGauche && etatCapteurCentre && !etatCapteurDroit) ||
(etatCapteurGauche && !etatCapteurCentre && !etatCapteurDroit)) {
    // Garder les moteurs à la même vitesse pour avancer tout droit
    moteurAvancer(150, 200); // Les deux moteurs à la vitesse 200
  }
  // Si la ligne est détectée sur le côté droit (noir)
  else if (etatCapteurGauche && etatCapteurCentre && etatCapteurDroit) {
    // Réduire la vitesse du moteur droit pour tourner à droite
    moteurArret(); // Vitesse du moteur gauche à 150, vitesse du moteur droit à 200
  }
  // Si aucune ligne n'est détectée, arrêter
```

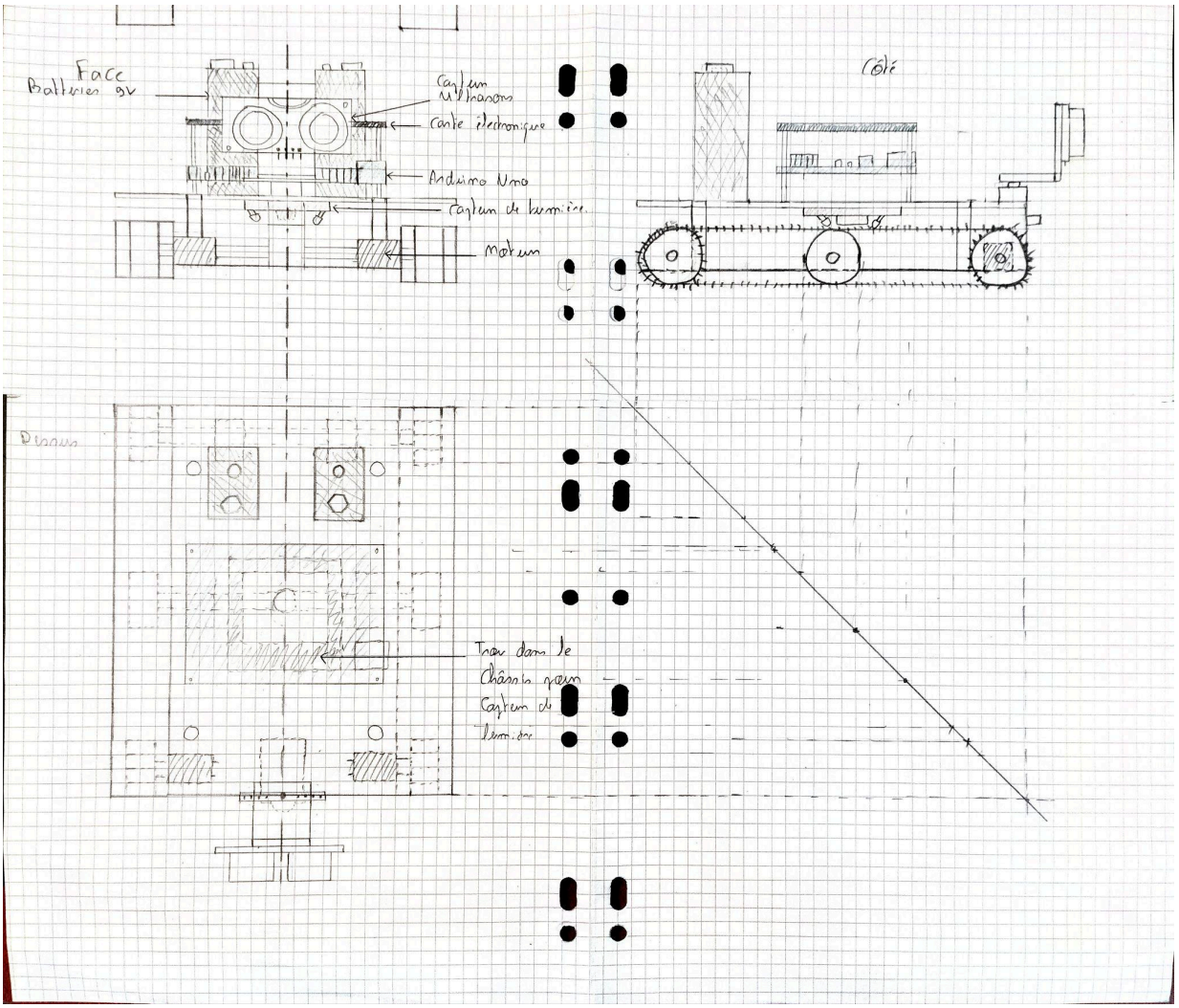
3

```
  else {
    moteurAvancer(200, 200);
  }
}

void moteurAvancer(int vitesseGauche, int vitesseDroit) {
  // Moteur A
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, vitesseGauche); // Vitesse PWM pour le moteur gauche

  // Moteur B
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, vitesseDroit); // Vitesse PWM pour le moteur droit
}

void moteurArret() {
  // Moteur A
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 0); // Arrêt du moteur gauche

  // Moteur B
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, 0); // Arrêt du moteur droit
}
```

Programme lecture capteurs infrarouge:

```
// Définition des broches des capteurs de lumière
const int sensorPin1 = A0;
const int sensorPin2 = A1;
const int sensorPin3 = A2;

void setup() {
  // Initialisation de la communication série à 9600 bps
  Serial.begin(9600);
}

void loop() {
  // Lecture des valeurs des capteurs de lumière
  int sensorValue1 = analogRead(sensorPin1);
  int sensorValue2 = analogRead(sensorPin2);
  int sensorValue3 = analogRead(sensorPin3);

  // Affichage des valeurs lues sur le moniteur série
  Serial.print("Sensor A0: ");
  Serial.print(sensorValue1);
  Serial.print("\tSensor A1: ");
  Serial.print(sensorValue2);
  Serial.print("\tSensor A2: ");
  Serial.println(sensorValue3);

  // Attente de 500 ms avant la prochaine lecture
  delay(500);
}
```

7.2. Schémas de montages, plans de conception...



Au tout début de notre projet, et avant d'entamer une conception 3D sur Solidwork, nous avions réfléchi à quoi notre robot pouvait ressembler. Il s'agissait notamment de voir où disposer les composants et de trouver le moyen de propulsion et de transmission du robot.

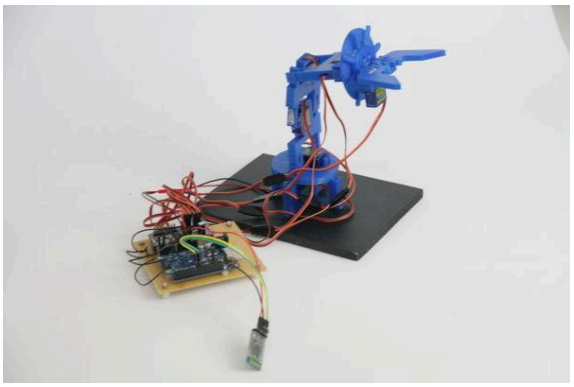
Finalement, nous avons abandonné les chenilles en cours de projet, mais l'apparence globale entre ce premier dessin industriel et le rendu final n'a pas énormément évolué.

7.3. Propositions de sujets de projets

Pour finir, un projet qui pourrait être réalisé serait la création d'un bras industriel à trois axes et commandé par un joystick ou autonome.

Un tel projet pourrait être intéressant car il y aurait différents aspects à prendre en compte comme la modélisation 3D du bras, le branchement électronique du système et la programmation. De plus ce projet sera enrichissant car ce genre de bras à une utilité concrète dans le monde de l'industrie.

Nous pourrions également lui ajouter une capacité de détection pour trier des cubes de différentes couleurs dans des boîtes par exemple.



De plus, comme montré ci-contre, de nombreux projets du même type ont déjà été menés, offrant alors une grande documentation pour s'aiguiller en cas de difficulté.

De la même façon que pour notre projet, celui-ci pourrait largement être mené avec une carte arduino uno des composants recyclés d'autres projets.

L'articulation du bras pourrait quant à lui, soit être directement acheté et utilisé pour les projets suivants soit être conçu en CAO puis imprimé.