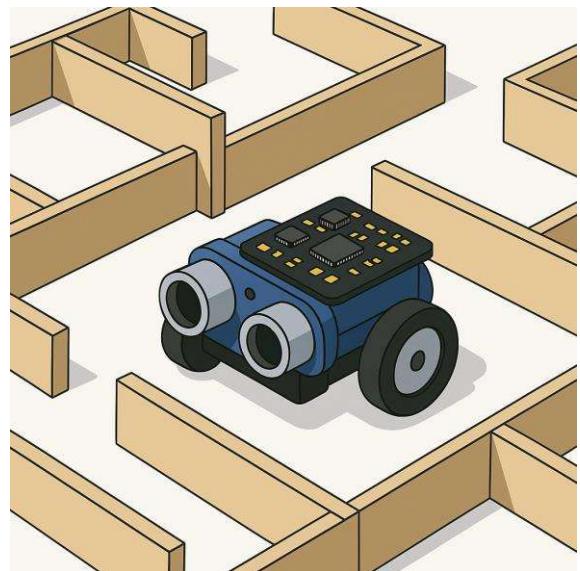


BELLAHBIB Amate-Allah
EL MESTARI Mohamed-Amine
NASSCHAERT Florent
PERRIN Mael
ZEAITER Yehia
Étudiants en 1ère année, ENSEA

Rapport de Projet :

Robot Chercheur et

Cacheur



Pour M. SIMOND Nicolas

Enseignant scolaire

I- Présentation générale du projet.....	4
1) Contexte.....	4
2) Etude du cahier des charges et contraintes	4
3) Problématique : Objectif	4
Analyse du besoin.....	4
II- Gestion du projet.....	6
1) Répartitions des tâches	6
Répartition en deux temps	6
2) Diagramme de GANTT.....	6
3) Méthodologie => diagramme SADT	7
III- Présentation du travail effectué :	8
1) Matériels et composants utilisés	8
Mobilité du robot.....	8
Alimentation.....	9
2) Tests des composants	9
Test des moteurs avec les drivers.....	9
Émetteur infrarouge	10
Récepteur infrarouge	10
Capteur ultrason.....	11
3) Modélisation	13
4) Conception du PCB	13
Schéma électronique	14
Choix des empreintes des composants	19
Vérification du schéma (ERC).....	19
Routage du PCB.....	20
5) Programmation	22
Mode de test.....	22
6) Problèmes rencontrés	23
7) Solutions apportées	24
Première version de la maquette.....	24
Passage aux moteurs à courant continu	25
Utilisation de moteurs avec réducteurs	26

Ajout des capteurs infrarouges	26
Limites de la maquette.....	26
Perspectives d'amélioration.....	27
Conclusion	27
Annexes	28

I- Présentation générale du projet

1) Contexte

Le projet "Cache-cache" s'inscrit dans le cadre de la première année du cycle ingénieur à l'ENSEA. Il s'agit d'un projet collaboratif dont l'objectif est de concevoir deux robots mobiles autonomes capables de jouer à une version robotisée du jeu de cache-cache. Ces robots évoluent sur une plateforme constituée d'un labyrinthe avec des obstacles matérialisés par des planches, ce qui impose une navigation intelligente et stratégique.

L'un des robots, le "chercheur", a pour mission de localiser le second, le "cacheur", qui doit quant à lui éviter d'être détecté. Pour cela, chaque robot est équipé de capteurs et contrôlé par une carte STM32L476, permettant la collecte de données, la prise de décisions et la mobilité autonome. Le projet est encadré par une méthode agile, favorisant une organisation efficace, une répartition claire des rôles et un suivi rigoureux à travers des réunions régulières et une gestion de projet structurée.

Au-delà de l'aspect ludique, ce projet met en œuvre de nombreuses compétences en électronique, programmation embarquée, mécatronique et gestion de projet. Il représente une première immersion concrète dans les défis techniques et humains d'une collaboration en ingénierie.

2) Etude du cahier des charges et contraintes

Le robot doit respecter certains critères :

- Déplacement : avancer, reculer, tourner, balayage, arrêt
- Détection : terrain, autre robot
- Composants : capteurs, moteurs et roues, PCB, boutons/interrupteur, intégration

3) Problématique : Objectif

Analyse du besoin

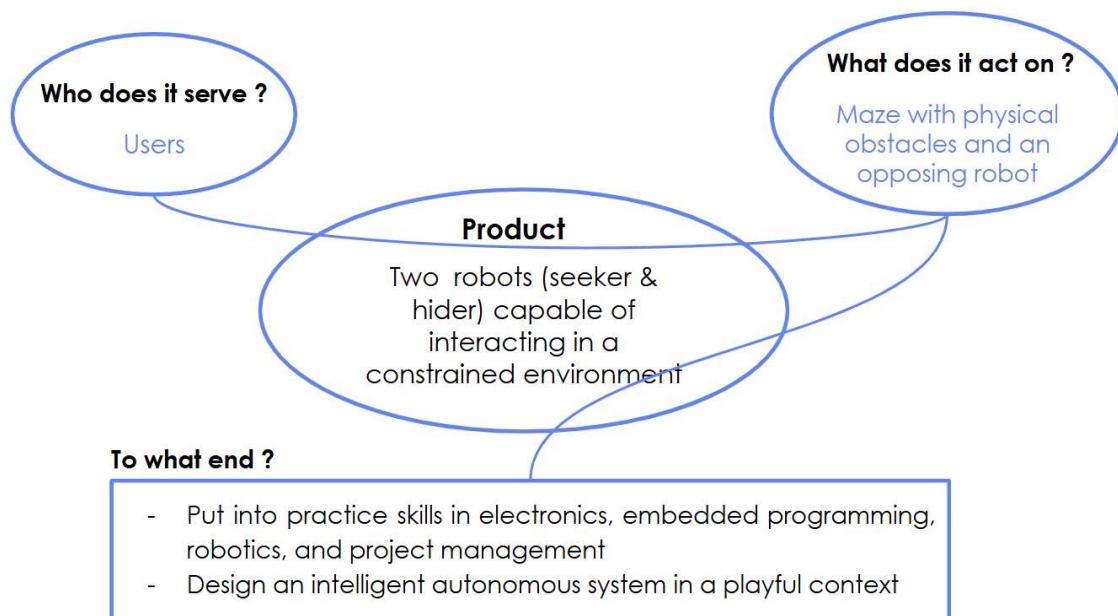
L'analyse du besoin sert à définir clairement ce qu'on doit concevoir, pour qui, dans quel contexte et pourquoi. C'est ce qui permet de bien cadrer le projet dès le départ. Grâce à l'étude du cahier des charges et à la liste qui relèvent de tout ce dont nous avons besoin pour le bon déroulement de notre projet.

Pour cela nous avons donc répondu aux questions suivantes :

- À qui ? => identification du client ;
- Sur quoi ? => identification de l'objet ou matériaux sur lequel notre système va agir ;

- Le produit => identification du produit réel ;
- Le but ? => identification du but de notre intervention sur le système.

Grâce à ces différentes questions, nous avons réalisé un schéma sous forme de bête à corne qui reprend ces différentes informations :



Dans notre cas, le produit final correspond à deux robots autonomes – un chercheur et un cacheur – capables d’interagir dans un environnement contraint, sous forme d’un labyrinthe réel avec des obstacles physiques. Le robot chercheur a pour objectif de localiser le robot cacheur, qui, lui, doit éviter d’être détecté.

Ce système s’adresse avant tout à des utilisateurs tels que nous, les étudiants. Il a été conçu dans un but d’apprentissage, pour nous permettre de mettre en pratique des compétences acquises en électronique, programmation embarquée et gestion de projet.

Le contexte du projet est donc à la fois technique et ludique : il s’agit de concevoir un système intelligent capable de naviguer, détecter, décider et interagir, tout en respectant des contraintes matérielles et de temps. Ce cadre nous a permis de découvrir concrètement les étapes d’un projet robotique, de la réflexion initiale jusqu’à la conception du PCB et à l’implémentation du comportement des robots.

II- Gestion du projet

1) Répartitions des tâches

Répartition en deux temps

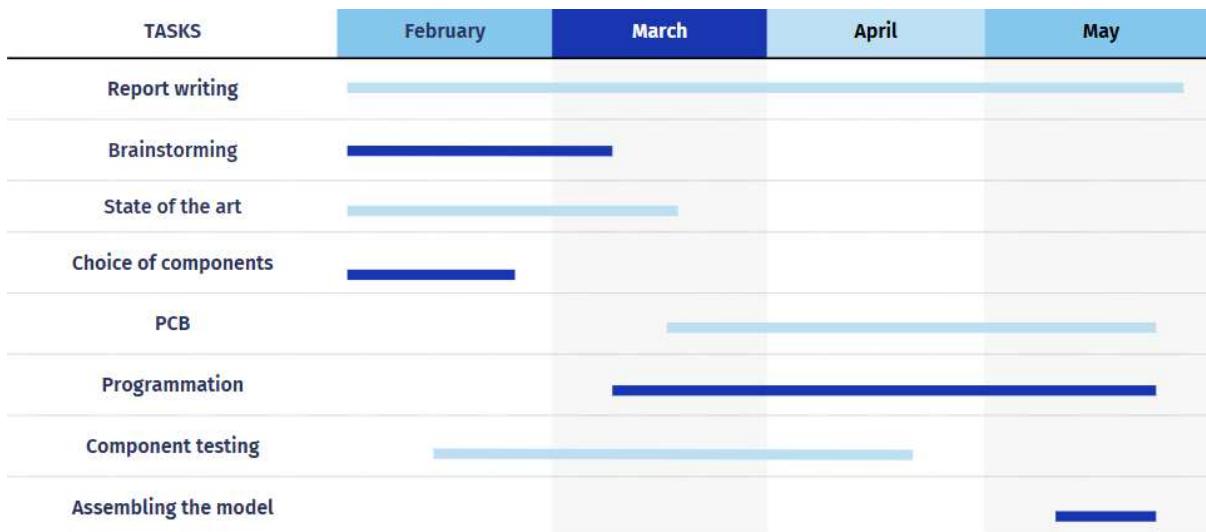
Au début du projet, les tâches ont été réparties par grandes catégories : conception du PCB, fonctionnement des capteurs ultrasonore et infrarouge, fonctionnement des moteurs pas à pas, modélisation 3D, programmation et algorithme. Toutefois, nous avons rapidement constaté que certaines d'entre elles dépendaient directement d'étapes préalables. Par exemple, la conception du PCB et des modèles 3D dépendait du choix des composants : il était donc nécessaire de valider leur bon fonctionnement et leur compatibilité avant d'avancer. De même, le développement de l'algorithme de cache-cache nécessitait une maquette fonctionnelle pour pouvoir être testé et ajusté.

En conséquence, comme le montre le diagramme de GANTT, le projet a été structuré en deux phases principales, tout en conservant certaines tâches réalisées en parallèle. La première était dédiée au choix et à la validation des composants. La seconde phase portait sur la conception du PCB, le développement des différents programmes nécessaires au fonctionnement de chaque composant, leur intégration, ainsi que la réalisation d'une première maquette fonctionnelle.

La répartition des tâches s'est organisée comme suit :

- Capteur ultrasonore : Mohamed-Amine et Yehia
- Capteur et émetteur infrarouges : Florent et Amate-Allah
- Moteurs pas à pas et à courant continu : Maël et Yehia
- Modélisation 3D : Maël
- Conception du PCB : Amate-Allah
- Programmation et algorithme : Florent et Yehia

2) Diagramme de GANTT



Ce diagramme de Gantt permet de visualiser la planification des différentes étapes du projet. Il s'étend sur une période de quatre mois : février, mars, avril et mai.

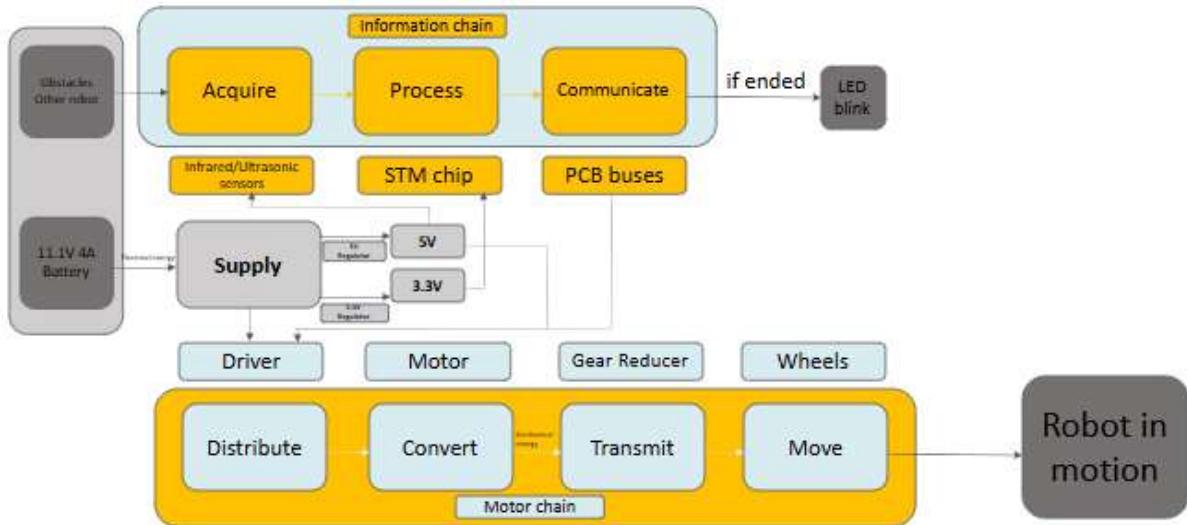
La rédaction du rapport accompagne le projet tout au long de son déroulement. Elle permet de consigner les avancées réalisées à chaque séance, facilitant ainsi une rédaction progressive et structurée du livrable final.

Le brainstorming constitue une étape cruciale dès le lancement du projet. Il vise à définir les besoins, les objectifs et les premières pistes de solution. En parallèle, l'état de l'art consiste à mener une veille technologique et scientifique, afin de s'appuyer sur des travaux existants et de s'inspirer de solutions déjà éprouvées. Ces deux étapes préliminaires sont essentielles pour clarifier la vision globale du projet.

Une fois les idées consolidées, le projet entre dans une phase plus technique, avec la conception du PCB, la programmation et enfin l'assemblage de la maquette du robot. Ces étapes se déroulent principalement entre mars et mai, suivant une logique chronologique : conception électronique, développement logiciel, puis intégration matérielle.

3) Méthodologie => diagramme SADT

Ce diagramme montre une vue globale de la composition du produit final exigé de ce projet.



III- Présentation du travail effectué :

1) Matériels et composants utilisés

Le choix des composants s'est effectué en tenant compte des contraintes du projet, notamment la mobilité, la détection d'obstacles, la communication inter-robots, ainsi que l'alimentation autonome.

Mobilité du robot

Pour l'entraînement des roues, nous avons initialement opté pour des moteurs pas à pas, pilotés à l'aide de drivers adaptés.

Pour la version finale de la maquette, les moteurs pas à pas ont été remplacés par des moteurs à courant continu, plus simples à intégrer.

Détection des obstacles

Concernant la détection d'obstacles, nous avons utilisé un capteur à ultrasons, capable de mesurer les distances avec une précision suffisante pour évoluer dans un environnement de type labyrinthe.

Communication inter-robot

La communication entre les deux robots a été assurée par un système infrarouge, comprenant un émetteur et un récepteur IR, permettant des échanges de signaux simples mais efficaces dans un espace restreint.

Alimentation

Le robot est alimenté par une batterie lithium-ion, choisie pour son bon compromis entre compacité, légèreté et capacité énergétique.

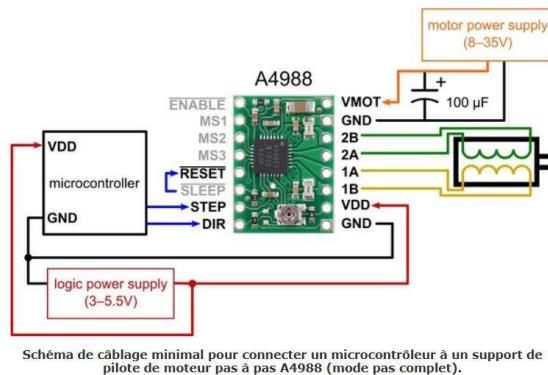
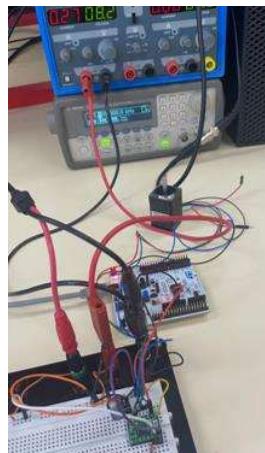
2) Tests des composants

Test des moteurs avec les drivers

Pour les essais de motorisation dans notre projet, nous avons utilisé un moteur pas à pas bipolaire, associé à un driver A4988, commandé par une carte STM32 Nucleo-L476RG.

Ce choix nous a permis de contrôler précisément la rotation du moteur en générant des impulsions. Ces impulsions sont appliquées à l'entrée STEP du driver A4988, tandis que la broche DIR détermine le sens de rotation. Le driver se charge ensuite de gérer le courant dans les enroulements du moteur.

Dans un premier temps, nous avons généré des impulsions à l'aide d'un GBF et ensuite nous avons utilisé la carte STM32 une fois que nous nous étions assurés que le moteur tournait avec le GBF.



Branchements pour le moteur avec le driver

Nous avons alimenté le moteur via une alimentation externe 12 V connectée à la borne VMOT du driver, avec un condensateur de 100 μ F ajouté en parallèle afin d'absorber les éventuels pics de tension et protéger le driver. Le signal logique (STEP, DIR) est quant à lui alimenté en 3,3 V par la carte STM32.

Le moteur recevait un signal périodique, dont la durée et la forme déterminaient à la fois le sens de rotation et la vitesse angulaire. En modulant la période de l'impulsion, nous pouvions ajuster dynamiquement ces deux paramètres.

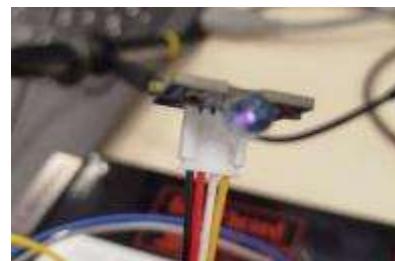
Enfin, une étape importante a été le réglage du courant limite du driver à l'aide du potentiomètre intégré. En se basant sur la fiche technique du moteur (courant nominal : 0,67 A), nous avons ajusté la tension de référence ($V_{ref} \approx 0,54$ V) afin de garantir un fonctionnement sûr et stable, sans surchauffe du moteur ni du driver.

Émetteur infrarouge

L'émetteur infrarouge est composé d'une LED IR connectée à l'alimentation (+5V et GND) et pilotée par une broche de commande reliée au microcontrôleur. Cette broche permet d'activer ou désactiver l'émission du signal infrarouge par simple changement d'état logique.



Émetteur infrarouge

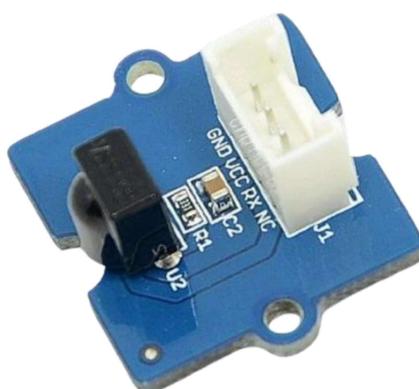


Émetteur infrarouge en fonctionnement

Récepteur infrarouge

Le récepteur infrarouge est alimenté en +5V et GND, et comporte une broche de sortie permettant de lire son état en temps réel. Cette broche est connectée à une entrée numérique du microcontrôleur.

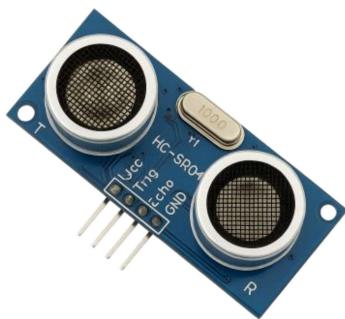
Après plusieurs essais expérimentaux, nous avons observé que lorsque l'émetteur infrarouge passe d'un état bas à un front montant (activation), le récepteur génère un front descendant en sortie. Ce comportement nous a permis de synchroniser la détection de signal : le robot chercheur peut ainsi identifier précisément le moment où le robot dissimulé active son émission, en surveillant ce changement d'état.



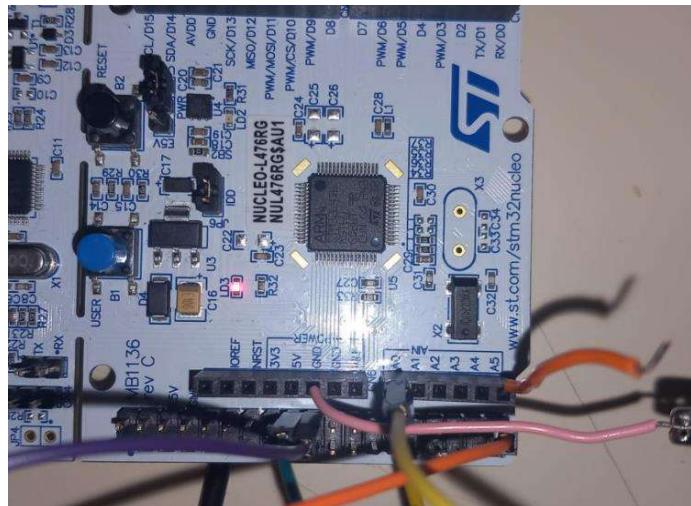
Récepteur infrarouge

Capteur ultrason

Afin de tester notre capteur, nous avons utilisé une carte STM32L476RG, créé un fichier IOC et utilisé le timer 2, nous avons mis le port PA0 en output (port du trigger) et le port PC0 en input (port de l'echo). Nous avons choisi de configurer les valeurs du PSC à 79 et de l'ARR à 9 de façon à avoir une période de 500ms. On alimente le capteur avec une tension de 5V, puis on envoie une impulsion qu'on observe à l'aide d'un oscilloscope. Chaque émission envoyée doit durer 10µs et deux émissions doivent être espacées de 500ms. Nous avons ensuite connecté la borne écho du capteur afin de vérifier qu'il y a bien un envoi et une réception. Lorsque notre capteur détecte un obstacle, on obtient une réception qui est une impulsion plus ou moins longue liée à la distance de l'obstacle par rapport au capteur.



Capteur ultrason HC-SR04



Branchements de la carte

Fil bleu => Power (5V)

Fil violet => Ground (GND)

Fil jaune => Trigger (PA0)

Fil orange => Echo (PA5)

Fil rose => Masse de l'oscilloscope

Fil orange => Echo de l'oscilloscope



Période entre deux émissions : 500ms

Durée d'une émission : 10μs



Envoi et réception au cours du temps

```

/* USER CODE BEGIN PD */
static int trigger_counter = 0; // reprend sa valeur quand on sort de là il est utilisé

/* USER CODE END PD */

/* Private macro */
/* USER CODE BEGIN PM */
#define Trigger_Port GPIOA
#define Trigger_PIN GPIO_PIN_0
#define Trigger_Period 20000 //200ms

void TIM2_IRQHandler(void)
{
    /* USER CODE BEGIN TIM2_IRQHandler */
    if (trigger_counter == Trigger_Period - 1){ //500ms - 10 micro-secondes
        HAL_GPIO_WritePin(Trigger_Port, Trigger_PIN, SET);

    }
    else if (trigger_counter == Trigger_Period){
        HAL_GPIO_WritePin(Trigger_Port, Trigger_PIN, RESET);
        trigger_counter = 0;
    }

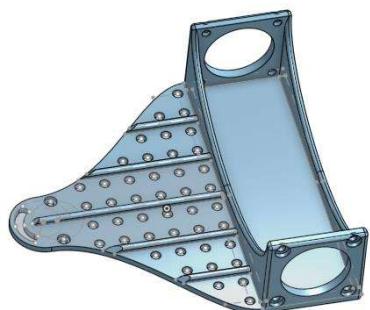
    trigger_counter += 1;
}

```

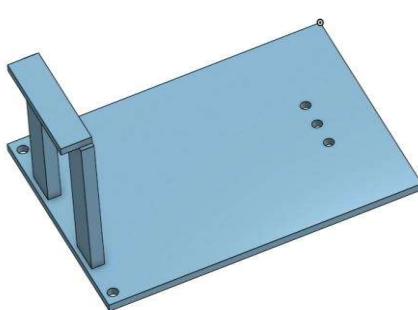
Code de test

3) Modélisation

L'ensemble de la modélisation 3D a été réalisé sur Onshape. Nous avions déjà à disposition la maquette principale. Nous avons décidé de l'agrandir et donc de la superposer avec celle en figure 2, en y intégrant un support pour notre capteur ultrasonore.



Modélisation 3D du support du robot



Extension du support du robot

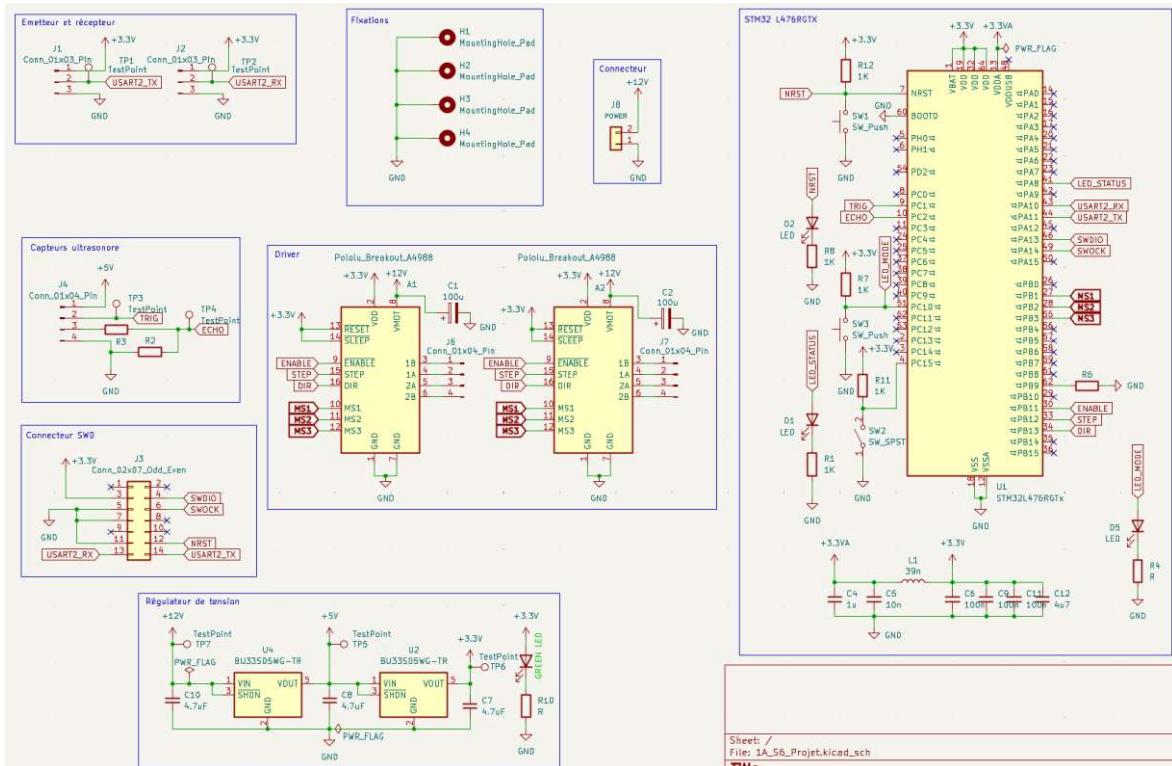


Support pour le capteur ultrasonore

4) Conception du PCB

Dans le cadre du projet, nous avons conçu un circuit imprimé (PCB) à l'aide du logiciel KiCad afin d'intégrer l'ensemble des composants nécessaires au fonctionnement du robot sur une carte unique et d'assurer la bonne communication entre les différents éléments électroniques du robot.

Schéma électronique



Vue d'ensemble du schéma électrique

Nous avons d'abord réalisé le schéma électrique (schematic editor) en organisant les composants par blocs fonctionnels :

Microcontrôleur STM32L476 : qui assure le contrôle global du système et communique avec les différents périphériques (capteurs, moteurs, boutons, LED...). Il est responsable du traitement de toutes les informations, du contrôle des moteurs, de la communication série et de l'analyse des capteurs. Il reçoit les signaux des capteurs (ultrasons, infrarouges). Il génère les signaux logiques de commande pour les drivers moteurs (STEP, DIR, ENABLE). Il communique avec l'extérieur via l'interface série (USART2). Il peut être programmé via l'interface SWD.

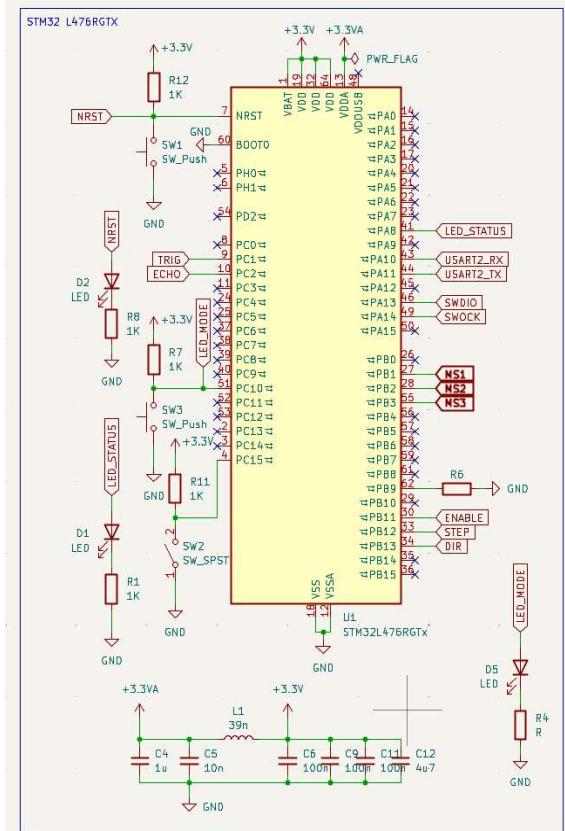
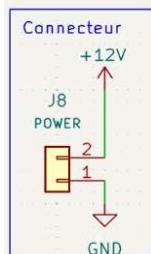


Schéma électrique du Microcontrôleur

Nous l'avons alimenté en 3,3 V et avons placé une inductance et des condensateurs de découplage positionnés à proximité des régulateurs et du STM32 pour stabiliser les tensions et filtrer les perturbations entre l'analogique et le numérique.



Alimentation : Nous avons prévu un connecteur de 2 broches pour raccorder l'alimentation de la carte via la batterie délivrant une tension nominale de 12 V. Cette tension est ensuite régulée sur la carte grâce aux deux régulateurs de tension, afin d'alimenter les composants logiques en 5 V et en 3,3 V. L'arrivée d'alimentation a été pensée pour desservir à la fois les circuits de puissance (drivers moteurs) et les circuits de commande (microcontrôleur, capteurs, etc.).

Régulateurs de tension BU33SD5WG-TR : Nous avons aussi intégré deux régulateurs de tension BU33SD5WG-TR, pour permettre de convertir l'alimentation de 12 V fournie par la batterie en 5 V (tensions nécessaires pour alimenter les composants logiques du système) et 3,3 V (tension de fonctionnement du STM32 et des signaux logiques).

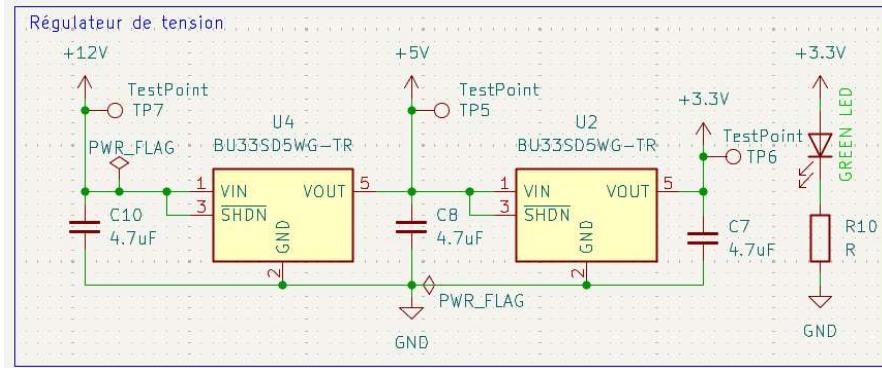


Schéma électronique des régulateurs de tension BU33SD5WG-TR

Nous avons associé pour chaque régulateur des condensateurs de découplage en entrée et en sortie, afin de stabiliser les tensions, limiter les pics de courant et garantir une alimentation propre pour les composants sensibles.

LED verte d'alimentation : Nous avons placé une LED verte, associée à une résistance entre le rail d'alimentation 3,3 V et la masse. Elle sert d'indicateur visuel permettant de vérifier que la carte est bien alimentée. Dès que la tension 3,3 V est présente, la LED s'allume.

Drivers moteurs A4988 : Nous avons intégré deux drivers moteur A4988 et connecteurs chargés de piloter les moteurs pas-à-pas grâce aux signaux fournis par le microcontrôleur. Nous avons alimenté les drivers en 12 V sur les bornes VMOT pour fournir les puissances nécessaires aux moteurs, et en 3,3 V sur les bornes VDD pour la logique.

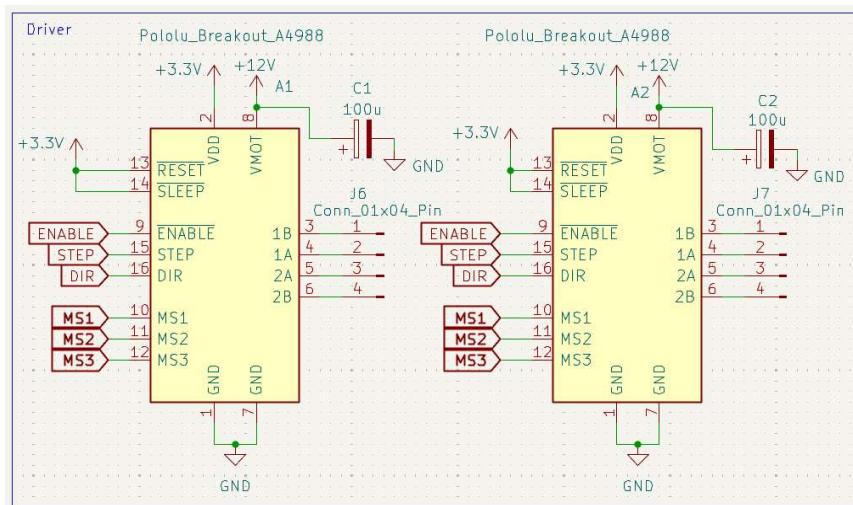


Schéma électronique des drivers A4988

Chaque driver reçoit des signaux logiques du microcontrôleur STM32 :

- ❖ STEP (pour faire avancer le moteur d'un micro-pas à chaque impulsion)
- ❖ DIR (pour indiquer le sens de rotation)
- ❖ ENABLE (pour activer ou désactiver le moteur)

Les quatre sorties du driver (1A, 1B, 2A, 2B) correspondent aux deux bobines du moteur pas-à-pas. Nous les avons reliées à des connecteurs à 4 broches sur le PCB, prévu pour brancher le moteur.

Puis nous avons reliées les broches MS1, MS2 et MS3 du driver au microcontrôleur. Ces trois signaux permettent de configurer le mode de microstepping du moteur (1/1, 1/2, 1/4, 1/8 ou 1/16 de pas), ce qui influence la précision du déplacement. Nous avons ajouté des condensateurs de $100 \mu\text{F}$ en parallèle sur l'alimentation VMOT pour protéger les drivers contre les chutes de tension et les pointes de courant.

Capteur à ultrasons HC-SR04 : Nous avons ajouté un connecteur de 4 broches pour permettre de raccorder le capteur à ultrasons, utilisé pour la détection d'obstacles. Ce capteur fonctionne avec deux broches : TRIG (pour l'émission) et ECHO (pour la réception) qu'on a relié au microcontrôleur via ces signaux.

La broche TRIG, commandée par le microcontrôleur, envoie une impulsion pour déclencher l'émission d'une onde ultrasonore. La broche ECHO renvoie un signal dont la durée correspond au temps mis par l'onde pour revenir après réflexion sur un obstacle. Ce signal est mesuré par le microcontrôleur pour en déduire la distance.

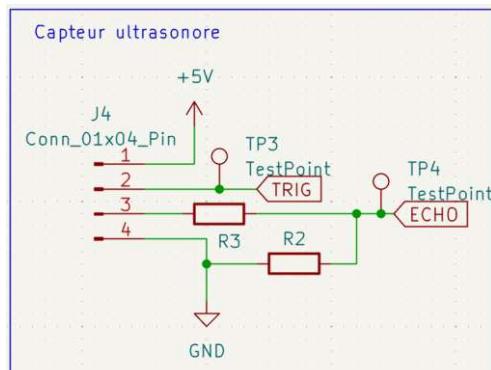


Schéma électrique du capteur à ultrasons HC-SR04

Pour protéger les entrées du STM32, nous avons placés des résistances en série et en pont diviseur pour adapter le signal ECHO en 5 V, au niveau logique 3,3 V attendu par le microcontrôleur.

Émetteur et récepteur infrarouges : Nous avons ajouté deux connecteurs de 3 broches pour relier un émetteur et un récepteur infrarouges.

- ❖ Nous avons relié le connecteur de l'émetteur (TX) à la broche USART2_TX du microcontrôleur STM32, permettant d'envoyer un signal modulé vers une LED infrarouge.
- ❖ Nous avons relié le connecteur du récepteur (RX) à la broche USART2_RX permettant de recevoir le signal capté par le récepteur infrarouge.

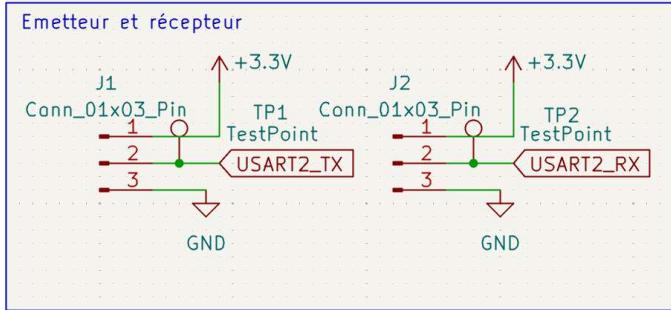


Schéma électrique de l'émetteur et du récepteur

Ce montage permet au système de détecter la présence d'un autre robot ou d'un obstacle dans le champ du capteur infrarouge, et d'éventuellement interagir avec lui (déttection, fuite, ou poursuite). Les signaux sont ensuite traités par le microcontrôleur pour adapter le comportement du robot (par exemple : s'arrêter, tourner...).

Connecteur SWD : Nous avons intégré un connecteur SWD afin de pouvoir programmer ou déboguer le microcontrôleur STM32L476 via l'interface SWD (Serial Wire Debug).

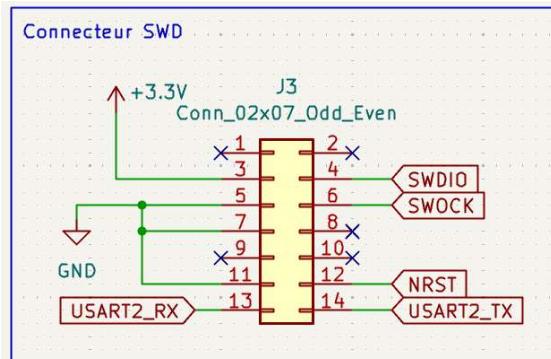
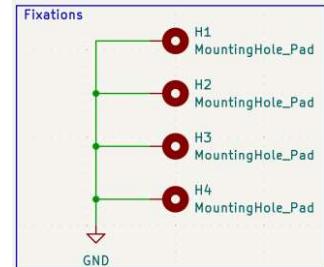


Schéma électrique du module SWD

Ce connecteur donne accès aux broches SWDIO (données), SWCLK (horloge) et NRST (réinitialisation) permettant de transférer le code compilé dans la mémoire du microcontrôleur, mais aussi de déboguer le programme depuis un environnement de développement comme STM32CubeIDE.

Fixations mécaniques : Nous avons ajouté et reliées quatre pastilles de fixation à la masse afin de visser le PCB au châssis du robot tout en améliorant la dissipation électromagnétique grâce à leur connexion au plan de masse.

LED, boutons, points tests et labels : Nous avons ajouté plusieurs éléments pour faciliter les tests et l'interaction avec la carte.



Des LED de statut et de changement de mode du robot pour les tests et pour visualiser l'état de certaines sorties du microcontrôleur que nous avons associés à des résistances et reliées à des broches du STM32.

Trois boutons poussoirs (SW1, SW2, SW3) pour tester certaines fonctionnalités (reset, changement de mode du robot, allumé ou éteindre la STM32).

Plusieurs points de test (TestPoint) sur les lignes d'alimentation (+12 V, +5 V, +3,3 V) ainsi que sur certaines lignes logiques (pour les capteurs). Ils permettent, lors des phases de mise au point ou de test, de venir placer une sonde d'oscilloscope ou de multimètre pour vérifier les tensions ou observer les signaux.

Nous avons utilisé des labels pour nommer clairement les signaux pour permettre de rendre le schéma plus lisible et de faciliter le routage. Nous avons également veillé à bien gérer les alimentations en séparant les parties logiques et puissances, et en ajoutant des condensateurs de découplage.

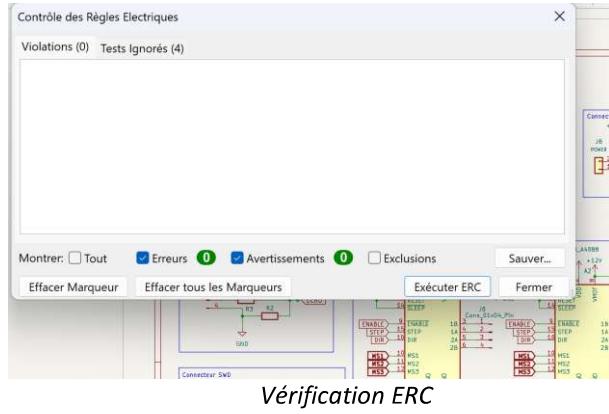
Choix des empreintes des composants

Après avoir terminé le schéma électronique, nous avons associé les empreintes (footprints) adaptées à chaque composant. Nous avons utilisé principalement des composants traversants (connecteurs, borniers, boutons, LED, résistances) pour faciliter la soudure manuelle. Les empreintes ont été choisies à partir des bibliothèques standard de KiCad ou créées/modifiées manuellement lorsque nécessaire, en s'assurant qu'elles soient compatibles avec les composants réels disponibles à l'école. ([Voir annexe](#))

On a fait attention à ce que chaque symbole du schéma soit bien relié à la bonne empreinte pour éviter les erreurs ensuite sur le PCB. Cette étape était indispensable avant de pouvoir placer les composants sur la carte et faire le routage.

Vérification du schéma (ERC)

Une fois le schéma électronique terminé et l'attribution des empreintes, nous avons procédé à une vérification automatique à l'aide de la fonction ERC (Electrical Rules Check) de KiCad. Cette étape permet de détecter les éventuelles erreurs dans le schéma, telles que des broches non connectées, des alimentations manquantes ou des incohérences de type (par exemple, une sortie reliée à une autre sortie). L'outil nous a permis de corriger plusieurs oubliés avant de passer à l'étape de création du PCB. Cette validation permet d'avoir un fonctionnement cohérent du circuit une fois routé.



Vérification ERC

Routage du PCB

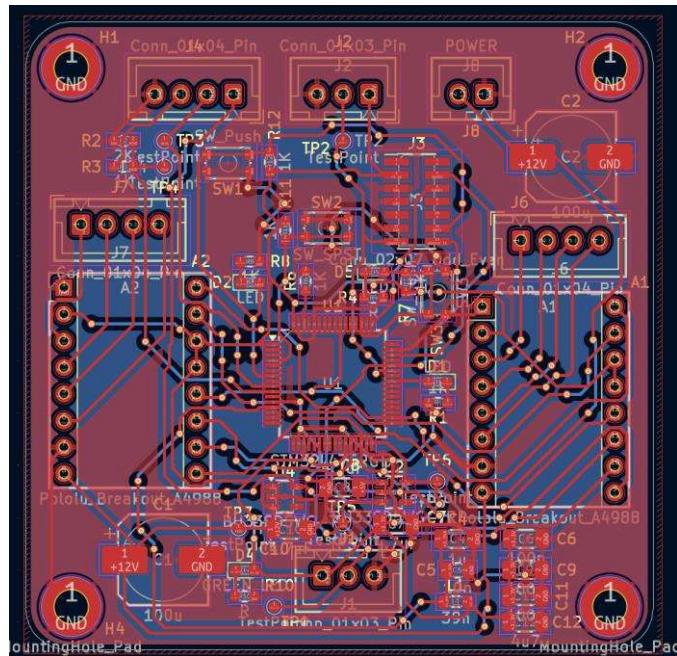
Une fois le schéma électronique terminé et vérifié (ERC), nous sommes passés au routage du PCB avec le logiciel KiCad (éditeur PCB). Le routage a été réalisé sur deux couches de cuivres (la couche supérieur (F.Cu) et la couche inférieure (B.Cu)), ce qui nous a permis de traverser les plans par des vias lorsque nécessaire.

Nous avons placé les composants avant de commencer le routage, en regroupant les blocs fonctionnels entre eux (alimentation, capteurs, drivers, microcontrôleur...), afin de limiter la longueur des pistes et d'éviter les croisements. La largeur minimale des pistes a été fixée à 0,3 mm, valeur imposée par les capacités de fabrication de l'école. Chaque via a été défini avec un diamètre extérieur de 0,8 mm pour un perçage de 0,4 mm. Nous avons placé les Vias de manière stratégique pour relier les couches tout en évitant des parcours inutiles.

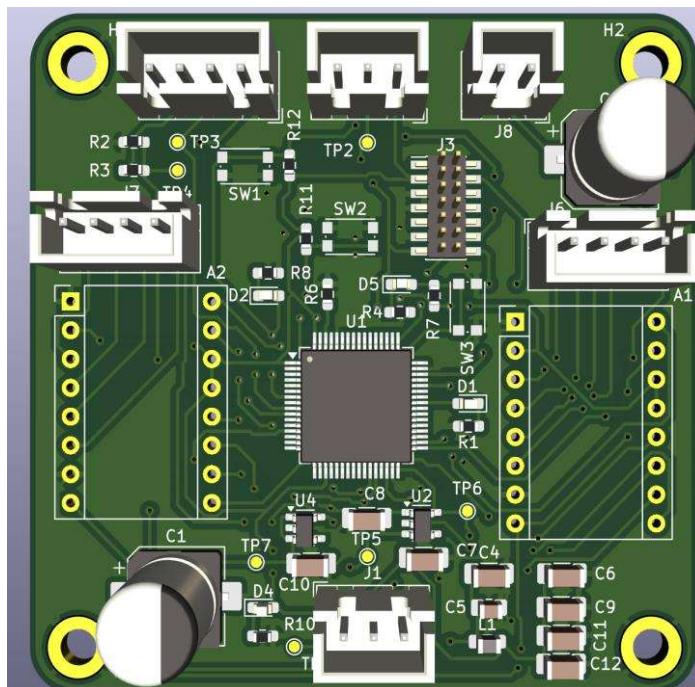
Nous avons appliqué les règles de conception pour que le routage soit conforme, soit :

- Le respect des espacements entre pistes et pads
- La minimisation de la longueur des pistes
- La séparation entre pistes de puissance et pistes logiques
- L'utilisation des Vias pour connecter les pistes des différentes couches entre elles

Le routage a été réalisé avec soin : nous avons respecté les règles de conception (Design Rules), évité les angles droit (90°), placé les condensateurs de découplage au plus proche du microcontrôleur et des régulateurs, et optimisé l'orientation des composants.

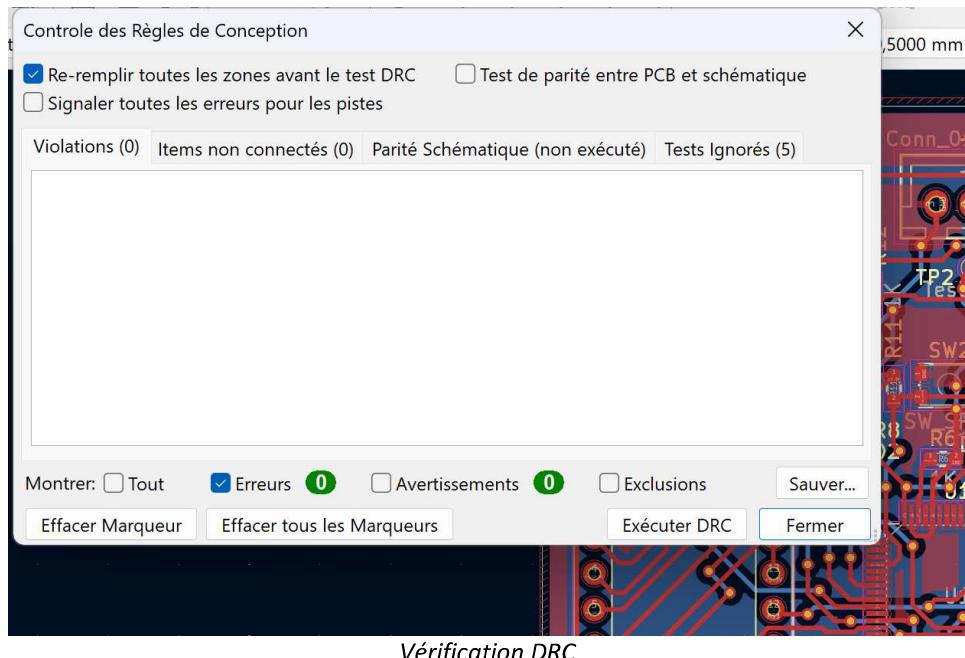


Routage du PCB (avec l'éditeur PCB de KiCad)



Vue de haut du PCB en 3D (avec l'éditeur PCB de KiCad)

Après le routage, nous avons lancé la vérification DRC (Design Rule Check), afin de nous assurer qu'aucune piste ne se chevauchait, qu'il n'y avait pas de court-circuit ou d'espacement insuffisant entre les éléments et que tous les éléments étaient connectés. Cette étape a permis de valider l'intégrité électrique et physique de notre conception avant export des fichiers de fabrication.



Vérification DRC

5) Programmation

La logique embarquée du robot repose sur l'implémentation de trois modes de fonctionnement distincts, sélectionnables via un bouton poussoir. Cette approche permet de rendre le robot polyvalent, en lui attribuant tour à tour un rôle dédié aux phases de test et de mise au point, un rôle cacheur ou un rôle chercheur.

Mode de test

Ce mode est destiné à la vérification fonctionnelle des différentes fonctions du robot. Lors de l'activation, le robot exécute en premier lieu un déplacement en forme de carré, ce qui permet de valider le bon fonctionnement des moteurs et de l'algorithme de contrôle du mouvement.

Par la suite, des LEDs témoins sont utilisées pour tester individuellement les capteurs (ultrasons et infrarouge). Ces indicateurs visuels permettent de s'assurer rapidement de la réception des signaux et de la réactivité des composants, facilitant ainsi les phases de débogage et de calibration.

Mode se cacher

Dans ce mode, le robot se déplace de manière autonome dans le labyrinthe pendant une durée de 30 secondes, en évitant les obstacles à l'aide de son capteur à ultrasons. À l'issue de cette période, il s'immobilise et active son émetteur infrarouge, ce qui permettra au robot chercheur de le localiser s'il se trouve à proximité.

Mode chercher

Lors de son activation, le robot commence par attendre 30 secondes, afin de laisser à son homologue le temps nécessaire pour se déplacer et se dissimuler.

Ensuite, il entame une phase d'exploration autonome du labyrinthe, en se déplaçant librement tout en évitant les obstacles grâce au capteur à ultrasons. Parallèlement, il interroge régulièrement son récepteur infrarouge afin de détecter la présence du robot dissimulé. Une détection positive indique que le robot est à portée de signal, ce qui marque la réussite de la phase de recherche. Et pour signifier cette fin, le robot fait clignoter une LED.

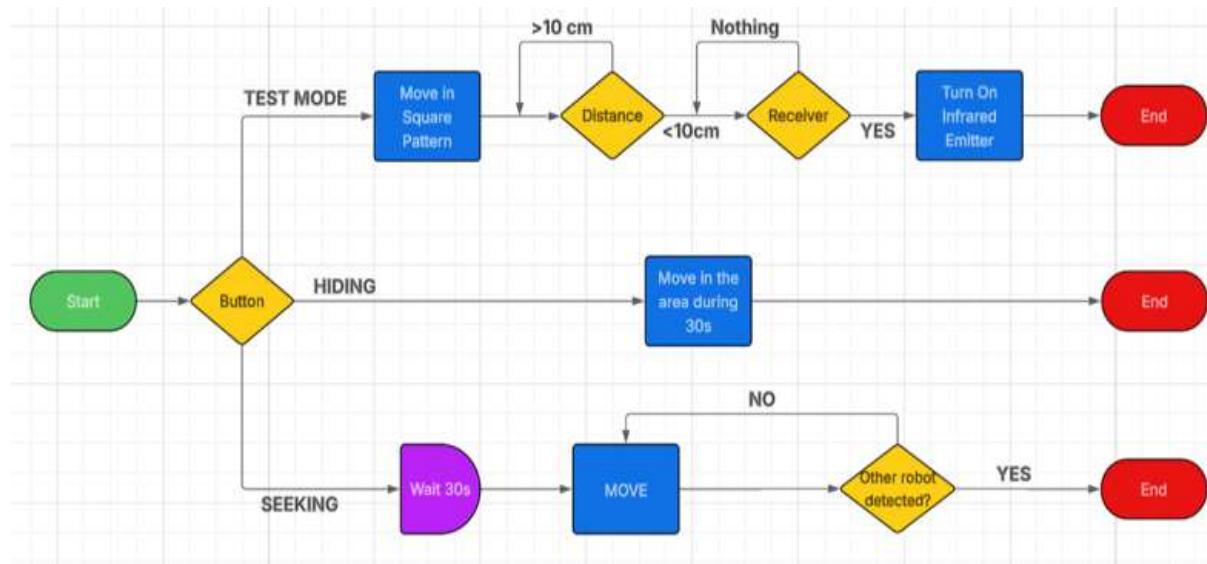


Diagramme de logique du robot

6) Problèmes rencontrés

Retour sur la conception du PCB et les difficultés rencontrées :

La conception du PCB s'est déroulée en plusieurs étapes. Nous avons d'abord réalisé le schéma électronique en organisant les composants par blocs fonctionnels. Après cela, nous avons attribué les empreintes à chaque composant, étape indispensable avant de passer au PCB. Une première vérification ERC (Electrical Rules Check) nous a permis de corriger les erreurs de connexion dans le schéma.

Nous avons ensuite effectué le routage du circuit dans l'éditeur de KiCad. À la fin du routage, une vérification DRC (Design Rule Check) a été lancée pour s'assurer qu'aucune erreur ne subsistait.

Cependant, plusieurs problèmes ont été rencontrés au cours du processus. Dans une première version, certains composants n'étaient pas disponibles à l'école et les tailles des pistes et des trous de via n'étaient pas correcte, ce qui nous a obligés à refaire une partie du travail avec des composants compatibles et d'ajuster les largeurs de pistes et via pour qu'elles respectent les normes de fabrication.

Lors d'une deuxième itération du PCB, le nombre de vias était trop important, ce qui a finalement rendu l'impression impossible avec les moyens disponibles à l'école.

Malgré nos efforts et les corrections apportées, nous n'avons pas pu faire imprimer la carte dans les délais impartis, faute de disponibilité matérielle à l'école et du manque de temps. Cette expérience nous a cependant permis de progresser significativement sur l'outil KiCad.

Retour sur le test du capteur ultrason :

Utilisation d'un tutoriel pour une carte STM différente de la nôtre et donc câblage différent ce qui ne permettait pas le fonctionnement de notre capteur.

Le signal émis pouvait être trop rapide pour s'afficher correctement sur l'oscilloscope, il fallait modifier le rapport cyclique.

Alimentation de la carte avec 3.3V qui est la mauvaise valeur de tension car il faut 5V.

7) Solutions apportées

Ne disposant pas encore de PCB au début du projet, nous avons choisi de construire une première maquette sur breadboard afin de tester les composants et le fonctionnement général du robot.

Première version de la maquette

Cette première version intégrait :

- Une carte STM32L476RG,
- Deux moteurs pas à pas avec leurs roues et drivers,
- Un capteur ultrason.

Nous avons cependant rencontré des difficultés pour contrôler correctement les moteurs. Ceux-ci réagissaient de manière imprévisible, probablement à cause de problèmes de

connexion ou d'incompatibilités avec les drivers. Malgré plusieurs tentatives, nous n'avons pas réussi à identifier clairement l'origine du problème ni à le corriger.

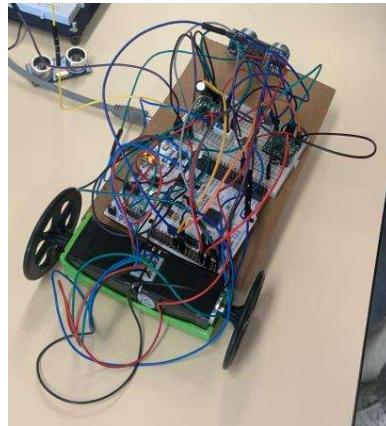


Image de la première maquette

Passage aux moteurs à courant continu

Pour simplifier le câblage et faciliter les tests sur breadboard, nous avons remplacé les moteurs pas à pas par des moteurs à courant continu, plus simples à mettre en œuvre et nécessitant moins de connexions. Ces moteurs étaient pilotés par la carte STM32 via un **driver L298N**.

Cette modification nous a permis de :

- **Contrôler la vitesse des moteurs** grâce à un signal PWM généré par un timer de la carte STM32, avec un rapport cyclique ajustable ;
- **Contrôler le sens de rotation** en modifiant l'ordre des signaux digitaux envoyés aux bornes du moteur.
- **Arrêter automatiquement** les moteurs lorsque le capteur ultrason détectait un obstacle à une distance définie (10 cm).

Cependant, une fois la maquette posée au sol, les moteurs ne parvenaient pas à déplacer le robot. Le couple fourni était insuffisant pour supporter le poids de la structure.



Image du moteur à courant continu

Utilisation de moteurs avec réducteurs

Pour résoudre ce problème, nous avons remplacé les moteurs par des modèles à courant continu équipés de réducteurs mécaniques. En réduisant la vitesse, ces réducteurs augmentent le couple moteur, ce qui a permis à la maquette de se déplacer correctement.



Image du moteur à courant continu avec réducteur mécanique

Ajout des capteurs infrarouges

Enfin, nous avons ajouté un émetteur et un récepteur infrarouge. Une LED intégrée à la carte STM32 a été programmée pour clignoter dès que le capteur reçoit un signal.

Les détails techniques relatifs au câblage, à la génération des signaux PWM et à la configuration des composants sont disponibles dans les guides « How to » mis à disposition sur notre dépôt GitHub.

Limites de la maquette

Cette maquette, construite dans les derniers instants, présentait néanmoins certaines limites. Elle ne disposait pas de régulateur de tension pour alimenter correctement la carte STM32, ce qui nous obligeait à la relier en permanence à une source d'alimentation externe pour pouvoir faire fonctionner le robot.

Par ailleurs, la structure manquait de rigidité : plusieurs composants n'étaient pas fixés solidement au support principal (une simple planche de bois), ce qui compliquait les déplacements sur de longues distances.

Malgré ces contraintes, la maquette a rempli son rôle principal : démontrer que les différents composants pouvaient fonctionner ensemble. Elle a ainsi servi de base pour envisager une version finale du robot, plus robuste et plus aboutie.

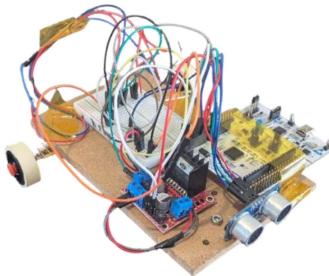


Image de la maquette

Perspectives d'amélioration

Tout d'abord, l'optimisation du routage du PCB pourrait permettre de réduire le nombre de Vias, de raccourcir certaines pistes critiques, ou encore d'améliorer la compacité globale de la carte. Une meilleure anticipation des composants disponibles à l'école aurait également évité certaines refontes de la carte.

Sur le plan organisationnel, plusieurs pistes d'amélioration ont été identifiées pour renforcer l'efficacité collective. Tout d'abord, réussir à être plus rapide dans la réalisation des tâches permettrait de respecter davantage les délais et de dégager du temps pour les tests. Ensuite, une meilleure distribution des rôles dès le début du projet aurait facilité l'avancement, en clarifiant les responsabilités de chacun. Enfin, la mise en place de réunions plus fréquentes, entre membres du groupe ou avec le groupe partenaire, aurait permis de mieux clarifier les objectifs, d'améliorer la coordination et d'éviter certains malentendus.

Conclusion

En conclusion, ce projet nous a permis de développer à la fois des compétences techniques et des aptitudes en gestion de projet. Grâce à une organisation structurée (GANTT, réunions, répartition des tâches), nous avons appris à travailler efficacement en équipe et à respecter les délais.

Sur le plan technique, nous avons acquis de solides connaissances en électronique embarquée (configuration des Timers STM, gestion des capteurs infrarouges et ultrasoniques), en modélisation 3D avec Onshape, et en diagnostic via oscilloscope. La conception du PCB nous a également permis de mettre en pratique l'ensemble du processus de réalisation d'une carte électronique, de la saisie du schéma à la vérification finale. Ce projet a ainsi été une expérience complète, mêlant théorie, pratique et travail collaboratif, dans un contexte ludique.

Annexes

Symbole: Attribution Empreintes	
1	A1 - Pololu_Breakout_A4988 : Module:Pololu_Breakout-16_15.2x20.3mm
2	A2 - Pololu_Breakout_A4988 : Module:Pololu_Breakout-16_15.2x20.3mm
3	C1 - 100u : Capacitor_SMD:CP_Elec_8x10.5
4	C2 - 100u : Capacitor_SMD:CP_Elec_8x10.5
5	C4 - 1u : Capacitor_SMD:C_1206_3216Metric_Pad1.33x1.80mm_HandSolder
6	C5 - 10n : Capacitor_SMD:C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
7	C6 - 100n : Capacitor_SMD:C_1206_3216Metric_Pad1.33x1.80mm_HandSolder
8	C7 - 4.7uF : Capacitor_SMD:C_1206_3216Metric_Pad1.33x1.80mm_HandSolder
9	C8 - 4.7uF : Capacitor_SMD:C_1206_3216Metric_Pad1.33x1.80mm_HandSolder
10	C9 - 100n : Capacitor_SMD:C_1206_3216Metric_Pad1.33x1.80mm_HandSolder
11	C10 - 4.7uF : Capacitor_SMD:C_1206_3216Metric_Pad1.33x1.80mm_HandSolder
12	C11 - 100n : Capacitor_SMD:C_1206_3216Metric_Pad1.33x1.80mm_HandSolder
13	C12 - 4u7 : Capacitor_SMD:C_1206_3216Metric_Pad1.33x1.80mm_HandSolder
14	D1 - LED : LED_SMD:LED_0603_1608Metric_Pad1.05x0.95mm_HandSolder
15	D2 - LED : LED_SMD:LED_0603_1608Metric_Pad1.05x0.95mm_HandSolder
16	D4 - GREEN LED : LED_SMD:LED_0603_1608Metric_Pad1.05x0.95mm_HandSolder
17	D5 - LED : LED_SMD:LED_0603_1608Metric_Pad1.05x0.95mm_HandSolder
18	H1 - MountingHole_Pad : MountingHole:MountingHole_3.2mm_M3_DIN965_Pad
19	H2 - MountingHole_Pad : MountingHole:MountingHole_3.2mm_M3_DIN965_Pad
20	H3 - MountingHole_Pad : MountingHole:MountingHole_3.2mm_M3_DIN965_Pad
21	H4 - MountingHole_Pad : MountingHole:MountingHole_3.2mm_M3_DIN965_Pad
22	J1 - Conn_01x03_Pin : Connector_JST:JST_XH_B3B-XH-A_1x03_P2.50mm_Vertical
23	J2 - Conn_01x03_Pin : Connector_JST:JST_XH_B3B-XH-A_1x03_P2.50mm_Vertical
24	J3 - Conn_02x07_Odd_Even : Connector_PinHeader_1.27mm:PinHeader_2x07_P1.27mm_Vertical_SMD
25	J4 - Conn_01x04_Pin : Connector_JST:JST_XH_B4B-XH-A_1x04_P2.50mm_Vertical
26	J6 - Conn_01x04_Pin : Connector_JST:JST_XH_B4B-XH-A_1x04_P2.50mm_Vertical
27	J7 - Conn_01x04_Pin : Connector_JST:JST_XH_B4B-XH-A_1x04_P2.50mm_Vertical
28	J8 - POWER : Connector_JST:JST_XH_B2B-XH-A_1x02_P2.50mm_Vertical
29	L1 - 39n : Inductor_SMD:L_0805_2012Metric_Pad1.05x1.20mm_HandSolder
30	R1 - 1K : Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm_HandSolder
31	R2 - 2K : Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm_HandSolder
32	R3 - 1K : Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm_HandSolder
33	R4 - R : Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm_HandSolder
34	R6 - 1K : Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm_HandSolder
35	R7 - 1K : Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm_HandSolder
36	R8 - 1K : Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm_HandSolder
37	R10 - R : Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm_HandSolder
38	R11 - 1K : Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm_HandSolder
39	R12 - 1K : Resistor_SMD:R_0603_1608Metric_Pad0.98x0.95mm_HandSolder
40	SW_Push : Button_Switch_SMD:SW_Push_1P1T_NO_CK_KMR2
41	SW2 - SW_SPST : Button_Switch_SMD:SW_Push_1P1T_NO_CK_KMR2
42	SW3 - SW_Push : Button_Switch_SMD:SW_Push_1P1T_NO_CK_KMR2
43	TP1 - TestPoint : TestPoint:TestPoint_Pad_D1.0mm
44	TP2 - TestPoint : TestPoint:TestPoint_Pad_D1.0mm
45	TP3 - TestPoint : TestPoint:TestPoint_Pad_D1.0mm
46	TP4 - TestPoint : TestPoint:TestPoint_Pad_D1.0mm
47	TP5 - TestPoint : TestPoint:TestPoint_Pad_D1.0mm
48	TP6 - TestPoint : TestPoint:TestPoint_Pad_D1.0mm
49	TP7 - TestPoint : TestPoint:TestPoint_Pad_D1.0mm
50	U1 - STM32L476RGTx : Package_QFP:LQFP-64_10x10mm_P0.5mm
51	U2 - BU33SD5WG-TR : Package_TO_SOT_SMD:SOT-23-5
52	U4 - BU33SD5WG-TR : Package_TO_SOT_SMD:SOT-23-5

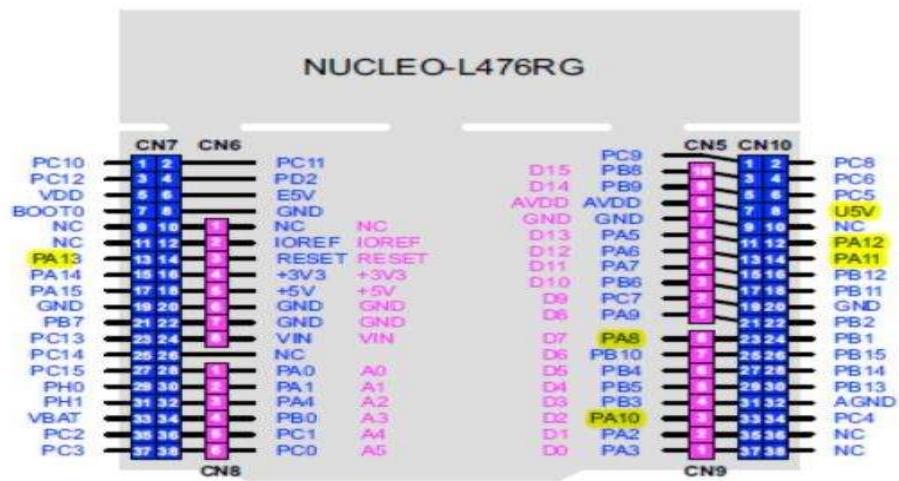
13 Battery:BatteryHol
 14 Battery:BatteryHol
 15 Battery:BatteryHol
 16 Battery:BatteryHol
 17 Battery:BatteryHol
 18 Battery:BatteryHol
 19 Battery:BatteryHol
 20 Battery:BatteryHol
 21 Battery:BatteryHol
 22 Battery:BatteryHol
 23 Battery:BatteryHol
 24 Battery:BatteryHol
 25 Battery:BatteryHol
 26 Battery:BatteryHol
 27 Battery:BatteryHol

Appliquer, Sauver Schéma & Continuer

OK

Annuler

Choix et attribution des empreintes



HC-SR04 specifications and dimensions

- Working Voltage: DC 5V
- Working Current: 15mA
- Working Frequency: 40Hz
- Max Range: 4m
- Min Range: 2cm
- Measuring Angle: 15 degree
- Trigger Input Signal: 10µS TTL pulse
- Echo Output Signal Input TTL lever signal and the range in proportion
- Dimensions: 45 * 20 * 15mm

Brochage de la STM32 et datasheet du capteur ultrason HC-SR04