Compte Rendu
Projet Ingénieur

Réalisation d'un robot pour la coupe de France de robotique

Etudiant Ingénieur : Albin Mouton

Tuteur de Projet : Xavier Redon

Responsable UE Projet : Florian Chevalier

Introduction

La Compétition

La coupe de France de robotique est une compétition nationale qui, cette année, fête ses 30 ans.
Elle réunit une centaine d'équipes chaque année en mai, à la Roche-Sur-Yon.
Le règlement de la compétition est publié tous les ans en septembre. J'ai pu m'appuyer sur celui-ci lors de la rédaction de mon cahier des charges.
Les équipes doivent alors concevoir un robot de toute pièce, afin de remporter un maximum de points.
Les robots participent à des matchs en 1 contre 1. Deux équipes tentent en même temps de réaliser un maximum d'actions.
Le thème de cette année est "Coloniser Mars". Les robots sont dans un scénario de "ferme spatiale" avec plusieurs objectifs :

- Récupérer des plantes, les mettre dans des cache-pots, et venir les trier dans des zones de dépôt ;
- Orienter des panneaux solaires ;
- Faire entrer en contact les PAMI avec les plantes récupérées ;
- Prédire le plus précisément son score.

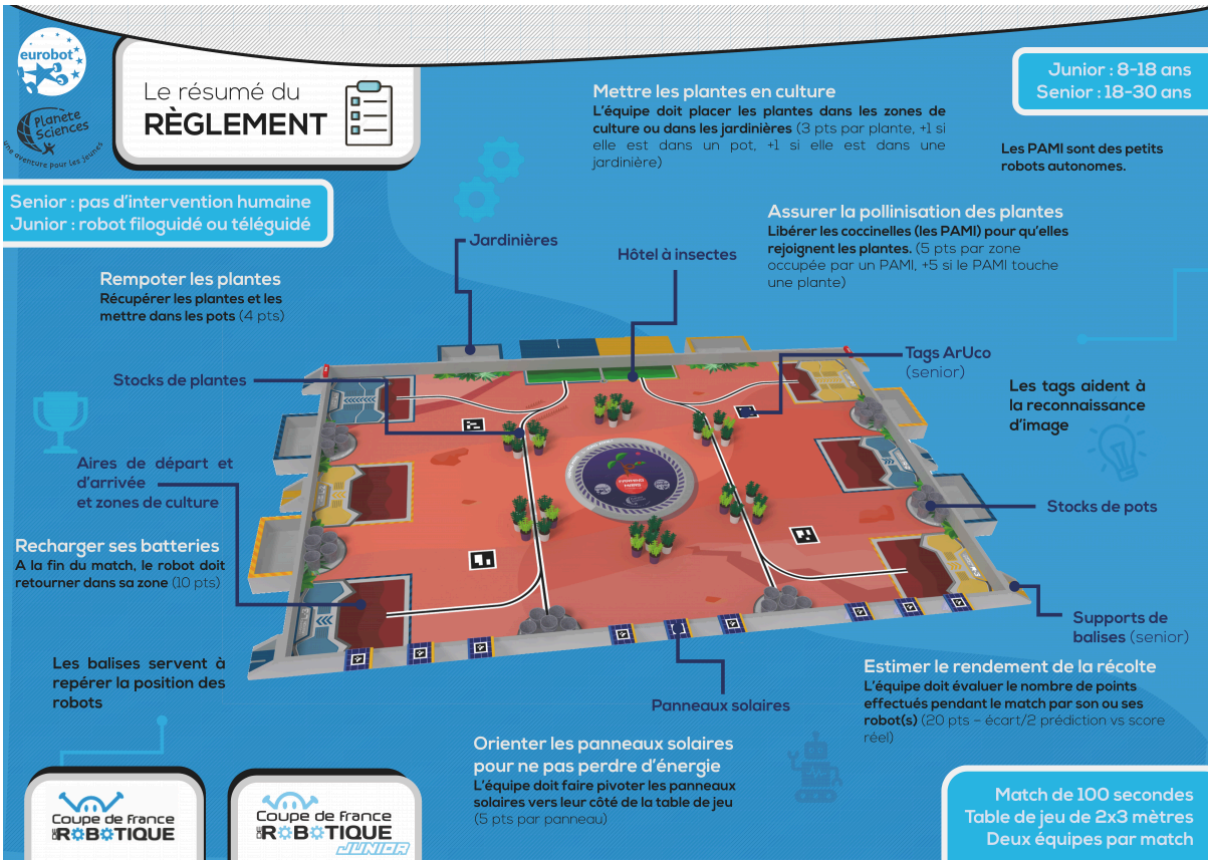

Figure 1. Illustration simplifiée du règlement

Les robots doivent respecter certaines règles :

- Être plus petit qu'un cube de 30x30x30 cm ;
- Etre autonome sur l'ensemble du temps de jeu (100 secondes) ;
- Démarrer en tirant une ficelle ;
- Ne pas entrer en contact avec le robot adverse, ou endommager le matériel de jeu ;
- Etre équipé d'un bouton d'arrêt d'urgence ;
- Les batteries doivent respecter des consignes de sécurité.

Sur place, les robots doivent d'abord passer 2 homologations, effectuer au maximum 5 matchs de placement, avant de potentiellement participer à l'arbre de tournoi de la phase finale.
Les 2 meilleures équipes obtiennent le droit de participer à la coupe d'Europe qui se déroule à la Roche sur Yon le même weekend.

L'Équipe

Cette année, j'ai la chance de participer à la coupe de France de Robotique avec une dizaine d'élèves ingénieurs membres de Robotech. Le club de robotique de l'école, Robotech, a déjà participé à cette compétition l'année passée. La participation du Club après plusieurs années d'absence nous a permis de nous rendre compte du réel défi que représentait la coupe, de nous positionner face à nos adversaires et surtout d'évaluer nos choix de conception grâce à une mise en conditions réelle de notre robot.
C'est ainsi qu'aidé par le bureau du club de robotique de l'école, j'ai pu mettre en œuvre les leçons apprises l'an passé et tirer profit de la conception précédente de notre Robot.

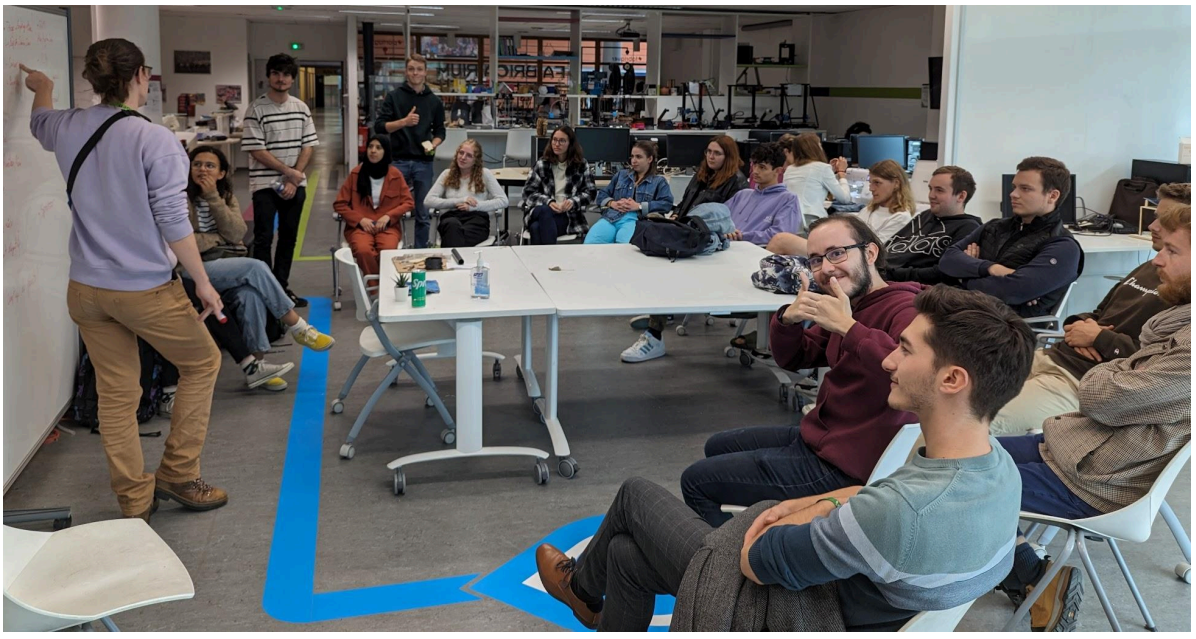

Figure 2. Réunion de présentation du projet aux membres du club

Cahier des charges

Ci-dessous vous trouverez le cahier des charges initial commenté afin évaluer l'état d'avancement du projet.
(✅ : objectif complété, ➖ objectif partiellement réalisé (avec un commentaire), ❌ objectif non-atteint)

- Conception du robot principal ➖
  - Conception/Intégration d'un ou plusieurs lecteurs RFID. (réorienté en IA) ➖
    - Conception du nœud ROS2 de lecture du tag. ✅
  - Conception d'une base roulante (Partiellement assemblée) ➖
    - Base roulante dont la projection a un périmètre est inférieure à 120 cm ✅
    - Conception d'un contrôleur de moteur BLDC ➖
      - Asservi (capteur d'orientation magnétique) en Field Orientation Control (FOC) (Performance Insuffisante) ➖
      - Base STM32 ✅
      - Communication CAN, UART & USB ✅
      - Alimentation batterie Li-Po 6S ✅
      - Support de moteurs jusqu'à ~30A ✅
    - Conception d'un système d'entraînement des roues par BLDC ✅
  - Conception d'actionneurs de saisie ❌
    - Conception mécanique des actionneurs (en cours) ➖
    - Développement de nouveaux noeuds ROS2 adaptés aux nouveaux actionneurs ❌
  - Correction de la carte mère actuelle du robot ✅
    - Développement d'un driver pour permettre d'interfacer les IMU ❌
  - Amélioration du système de Path Planning ❌
  - Amélioration du système de calcul de points ❌
- Conception des PAMI ➖
  - Conception d'une carte mère ✅
    - Taille la plus petite possible ~25cm² ✅
    - Contrôle de 2 moteurs DC ✅
    - Des emplacements pour des vis afin que le PCB soit un châssis ✅
    - Des GPIO permettant une modularité sur les actionneurs ✅
    - Capteurs de couleur pour permettre un suivi de ligne ✅
    - Bouton d'arrêt d'urgence (besoin de modifier la carte d'extension pour soutien mécanique) ➖
    - Communication Radio (bandes 2.4 GHz & 5 GHz proscrites car surchargées, donc probablement du sub-GHz) ✅
  - Conception d'une clé USB pour interfacer les PAMI via cette bande radio choisie ✅
  - Communication avec une tour de supervision selon des objectifs transmis par le robot principal (en cours) ➖

Cahier des spécifications

Ci-dessous se trouve le cahier des spécifications communiqué au tuteur.

Description des éléments

Tour de supervision

La tour de supervision est placée dans une zone dédiée du terrain. Elle peut être suffisamment haute pour permettre à une caméra d'y être installée. Cette caméra sera une caméra 4K avec un FOV de 110° pour voir l'ensemble du terrain.
Sur cette tour se trouve également un micro-ordinateur (NVidia Jetson), venant interpréter les valeurs de la caméra à l'aide de la bibliothèque OpenCV.
ROS2 sera également installé pour permettre de débugger le robot principal et la tour simultanément si tous sont sur un même réseau local.

Carte de contrôle de moteur BLDC

Basé sur un STM32F303 et un DRV8323, le contrôleur peut commuter jusqu'à 36V à ~30A pour venir alimenter un moteur à courant continu sans balai (BLDC).
Le PCB est un PCB 4 couches (70 um - 17.5 um - 17.5 um - 70 um)
Il dispose d'un port USB, de 2 ports série et d'un port CAN pour venir changer la consigne.
Un encodeur (incrémental ou positionnel) peut venir être connecté. Le moteur pourra être asservi en vitesse ou en position.
La carte sera auto-alimentée (générera ses rails 5V et 3.3V) à partir de la tension de la batterie.

Un identifiant peut être programmé dans la mémoire non volatile via le port USB. Cet identifiant définit les adresses à lire sur le bus CAN.

Un nœud ROS2 doit être écrit pour communiquer les consignes de vitesse avec les contrôleurs via le bus CAN. Celui-ci traduit également une commande de vitesse dans un repère orthonormé en les vitesses équivalentes des 3 moteurs placés en Triangle.

Le firmware de la carte mère doit être modifié pour permettre une communication CAN.

Encodeur absolu magnétique

Cette carte est un simple support pour l'encodeur magnétique MT6701, capable de communiquer en I2C ou en incrémental (UVW). Un connecteur JST-SH permettra de l'interfacer.

Le PCB est un PCB 2 couches (35 um - 35 um, sans contrôle d'impédance des lignes)

Clé USB Radio

Pour communiquer avec les PAMI et ne pas modifier en profondeur la carte mère du robot principal, un STM32 vient interfacer un port USB-A et une puce CC1101 pour permettre à un

ordinateur la communication 865 MHz. Un connecteur d’antenne SMA permettra de pouvoir expérimenter avec différentes antennes.

Le PCB est un PCB 4 couches (35 um - 17.5 um - 17.5 um - 35 um)

Le STM32 apparaît comme un port série virtuel, les paquets émis en UART seront émis en radio et vice-versa.
Cette clé servira également sur la tour de supervision

PAMI

La taille du PAMI doit être inférieure à 8x8x8cm.

Le PCB principal est un PCB 4 couches (35 um - 17.5 um - 17.5 um - 35 um)
Les PCB des capteurs de distance et d’extension seront des PCB 2 couches (35 um - 35 um, sans contrôle d’impédance)

Basé sur un STM32F103, les PAMI sont dotés de 2 contrôleurs de moteurs DRV8837, un contrôleur de charge de batterie Li-Po/Li-Ion (MCP73831), un buck boost 2.5V-6V -> 3.3V (TPS63051), et un accéléromètre gyroscope (LSM6DS3). Il peut être alimenté par des piles AA ou une batterie.
3 sorties de puissance sont disponibles (ex : pour alimenter un électro-aimant directement avec la batterie).

Un second PCB sert également à venir intégrer des connecteurs d'extension, ainsi qu’un tag ARUCO, qui pourront être lu par une caméra de supervision pour obtenir la position du PAMI.

Trois PCB supportant des capteurs de distance VL53L3 seront soudés verticalement entre le PCB principal et le PCB secondaire.

Les PAMI sont dépendants des informations de positionnement qui leur seront envoyées via la tour de supervision. Une fois que le robot principal aura démarré, un signal leur sera envoyé. Cela déclenche un timer de 90 secondes, qui, une fois dépassé, démarre le programme de déplacement des PAMI. Les PAMi vont à un objectif aux coordonnées prédéfinies en se basant sur les informations de position renvoyées par la caméra, mais peuvent s’arrêter si un obstacle est détecté devant eux.

Organisation et gestion de projet

Communication avec le tuteur projet

Pour permettre un suivi du projet par mon tuteur, un Wiki a été mis en place. Dans celui-ci j'ai pu documenter toutes mes avancées, ainsi que donner les premières documentations techniques.
Ce wiki est disponible ici (https://projets-se.plil.fr/mediawiki/index.php/SE5_2023/2024_P2)
Ce système de documentation m'a permis de rédiger les premières documentations, poser le contexte, mais aussi d'évaluer la progression hebdomadaire de mon projet.

J'ai également pu rentrer en contact avec M. Redon afin de lui soumettre les problématiques que j'ai pu rencontrer :

- Du matériel a pu être commandé afin de corriger certaines cartes ;
- Nous avons pu discuter d'algorithmes de commande de moteur afin d'optimiser le contrôle de celui-ci ;
- J'ai pu exprimer et formaliser les points méritaient son attention, ce qui m'a permis de synthétiser les problèmes pour qu'ensemble nous trouvions des solutions.

Gestion financière

Le matériel a été acheté par la trésorerie du club. Un rapport complet des dépenses a été effectué. Cependant, des fonds ont dû être levés pour financer de nouvelles générations de PCB. Pour cela, j'ai pu travailler en Freelance pour le club auprès d'un ancien étudiant de Polytech Lille.
J'ai pu concevoir une carte permettant de contrôler un moteur pas à pas via un port USB, ainsi que diverses entrées sorties. Cette carte dispose également d'un Hub USB pour permettre de venir brancher une caméra et d'être reconnue par le PC hôte. La carte est contrôlée en GCode, avec la même bibliothèque que j'ai écrite pour les projets.
Cela a permis de ne pas faire perdre d'argent au club tout en finançant le projet de la coupe de France de Robotique.

Gestion d’équipe

Dans le cadre du projet, j’ai été accompagné par les membres de Robotech, club de robotique de Polytech-Lille. En ma qualité de chef de projet, j’ai pu gérer cette équipe pour atteindre les objectifs listés plus haut dans la partie technique.
Pour se faire, j’ai pu organiser des réunions d’avancement hebdomadaires (tous les mercredis midis), ainsi que des séances de travail (tous les mercredis soir). Dans ces séances, j’étais disponible pour investiguer les obstacles qu’ils rencontraient, former les nouveaux membres, débattre et trancher sur les solutions techniques proposées.

J’ai pu mettre en place différents outils afin d’organiser nos travaux :

- Un serveur discord, permettant la discussion organisée par salons tournés vers des sous parties du projets. Cela avait pour but de laisser une trace simple et informelle des échanges, ainsi que de pouvoir noter les éléments de réponses et astuces qui n’auraient pas eu place dans ce rapport
- Un répertoire Gitlab, pour permettre la mise en commun de code, car la perte de travaux est monnaie courante, comme les membres utilisent leur matériel personnel, et qu’il est pertinent de pouvoir travailler à plusieurs sur un même sous-ensemble du projet.
- Une équipe Fusion-360, pour permettre aux élèves ingénieurs en spécialité mécanique de travailler en commun, d’exporter la nomenclature de tout ou partie du projet, et de faire du versioning sur des pièces ou des assemblages ;
- Un ordinateur de test dans la même configuration que le robot, afin de tester dans des conditions proches, le code qui a pu être développé ;
- Des disques dur bootable & live, sous Linux (Ubuntu 22.04) où ROS2 est installé, pour pouvoir permettre un développement aux personnes sous des machines Windows.

J’ai pu gérer la motivation d’une équipe, encourager l’évolution de chacun, et conseiller leurs décisions afin qu’ils puissent obtenir un résultat tangible.

J’ai également défini une politique d’évaluation de la valeur du temps de chacun passé sur une tâche. En effet, j’ai imposé à chaque membre de se définir un équivalent coût-horaire. Si, pour mettre en place une solution, une option permettait d’économiser N heures de travail coûtait moins que N*(coût horaire auto-défini pour la personne), alors nous investissions pour économiser du temps.
Cette politique est due au fait que le temps disponible de chacun par semaine est réduit (au minimum 2h, au maximum ~8h). De plus, cela a permis à certains membres d’explorer de nouveaux outils.

Gestion du temps

On peut comparer le diagramme de Gantt initial et le diagramme actualisé.

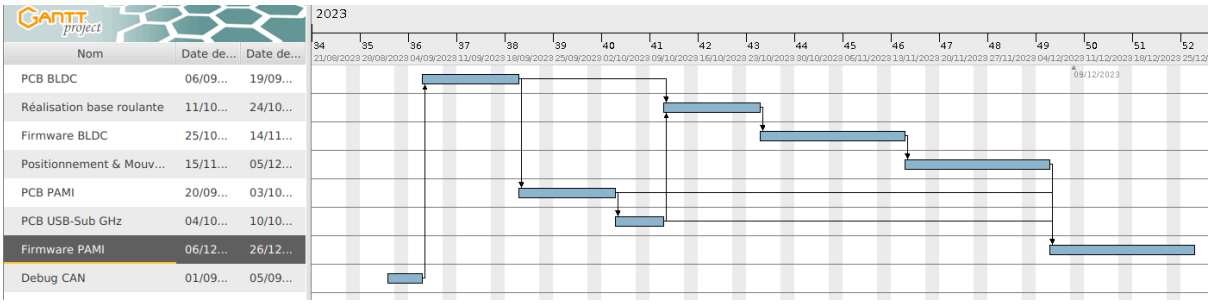

Figure 3. Gantt prévisionnel

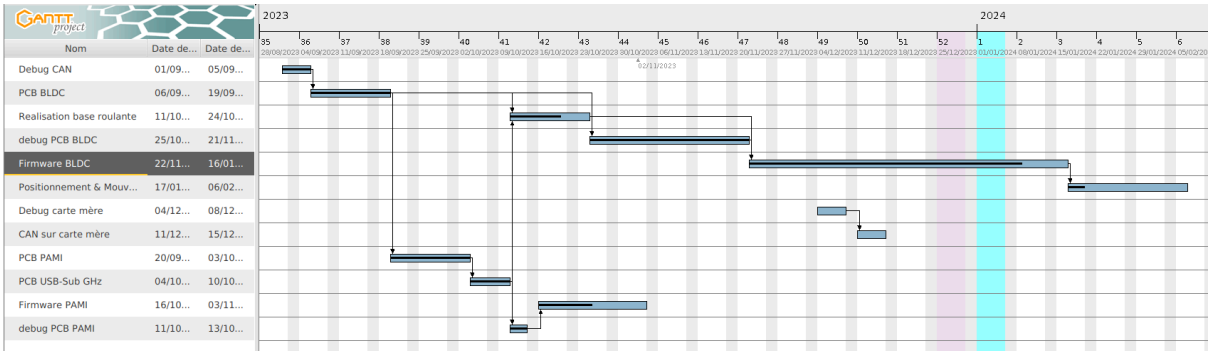

Figure 4. Gantt mis à jour

J’ai pris plus de temps qu’estimé à cause de problèmes techniques rencontrés. En effet, j’ai fait des erreurs sur les cartes qui m’ont pris du temps. Ces périodes de debug (que j’ai rajouté) m’ont fait perdre du temps. Suite à des problèmes de santé, je n’ai pas pu consacrer la semaine 01 de 2024, ce qui m’a également retardé.

Carte de Contrôle moteur

Contexte général

La Coupe de France de Robotique 2023

Cependant, au cours du projet, nous avons réalisé que la motorisation de notre robot était insuffisante. Elle était basée sur des moteurs à courant continu avec balais (Moteurs DC), qui, par le biais de réducteurs, venaient faire tourner des roues Mecanum.
Cette architecture permettait un déplacement latéral du robot, mais a pu engendrer certains problèmes :

- 4 roues Mecanum placées comme sur une voiture sont requises.
    - Pour que ces 4 roues soient en contact simultanément avec le sol, il faut un système de suspension :
        - Trop encombrant
        - Trop complexe (Il faut éviter autant de pièce possible en micromécanique)
        - Trop cher (Plusieurs centaines d'euros)
    - Les 4 roues, avec leur réducteur réduisent grandement la surface au sol disponible pour les actionneurs. En effet, seule la moitié de cette surface permettait l'installation d'un bras manipulateur.
- Les moteurs DC disponibles dans le commerce, de petite taille, tournent souvent très (trop) vite mais développent un couple très faible. Un réducteur important est nécessaire (environ 1:200)
    - Réducteur Planétaire : doit être réalisé sur mesure (très cher)
    - Réducteur à vis : peu volumineux, abordable, et non réversible, mais souvent fragile (plusieurs casses le jour de la compétition)
    - Réducteur à roue dentées : Plus volumineux, mais fragile, difficile à se procurer
    - Réducteur à poulie : sur mesure, mais peut être réalisé au sein de l'école : chronophage
- Complexité de câblage
    - La carte de contrôle de moteur DC faisait 10x10cm
    - Elle supportait 6 moteurs avec leurs encodeurs (soit 6x(2+4) câbles)

De plus, nous avons constaté que notre configuration (moteur DC) n'était pas utilisée par les équipes qui performaient le mieux.

Hardware

Choix de composants

Le coeur de la carte est un STM32F303BT
Celui-ci a été choisi de par :

- Son unité de calcul flottant : capable de faires des opération mathématiques sur des nombres à virgules, important pour pouvoir faire des calculs de PID ;
- Sa connectivité USB & CAN, la première pour pouvoir débugger et la deuxième pour la simplicité de câblage ;
- Les autres cartes du robot utilisent également des STM32. Toutes les fonctions implémentées sur un projet peuvent être portées sur un autre (par exemple le Parser GCode est la même bibliothèque que j'ai écrite pour les cartes de contrôleur moteur DC & BLDC).

Ce microcontrôleur commute les 6 MOSFETs nécessaires au contrôle du moteur par le biais d'un contrôleur DRV8323 de chez Texas Instruments.
Il inclut :

- Des amplificateurs pour la mesure de courant ;
- Un mécanisme de "charge pump" pour commuter les MOSFETs à haute tension ;
- Différentes sécurités (sur-courant, sous-tension, sur-tension entre le drain et la source lors de la commutation, …).

Pour permettre un asservissement en position, un encodeur magnétique 14 bits peut être connecté via un port JST-SH8. Celui-ci s'interface en I2C (débit relativement bas), ainsi qu'en UVW (impulsions successives par cran, plus rapide que la communication I2C)

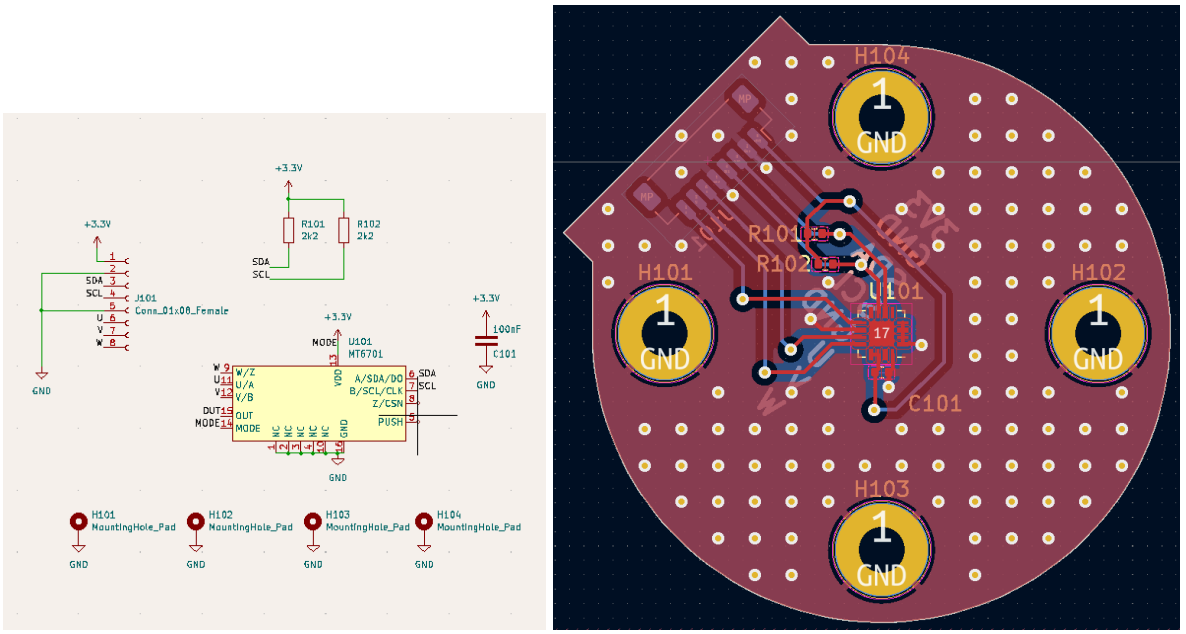

Figure 6. Schéma et PCB de l'encodeur

Principe de fonctionnement

La contrôle peut se faire via port série (De nombreux ports série sont disponibles sur la carte mère) à 115200 bauds, 8b1n, ou port série émulé sur un port USB mini-B.

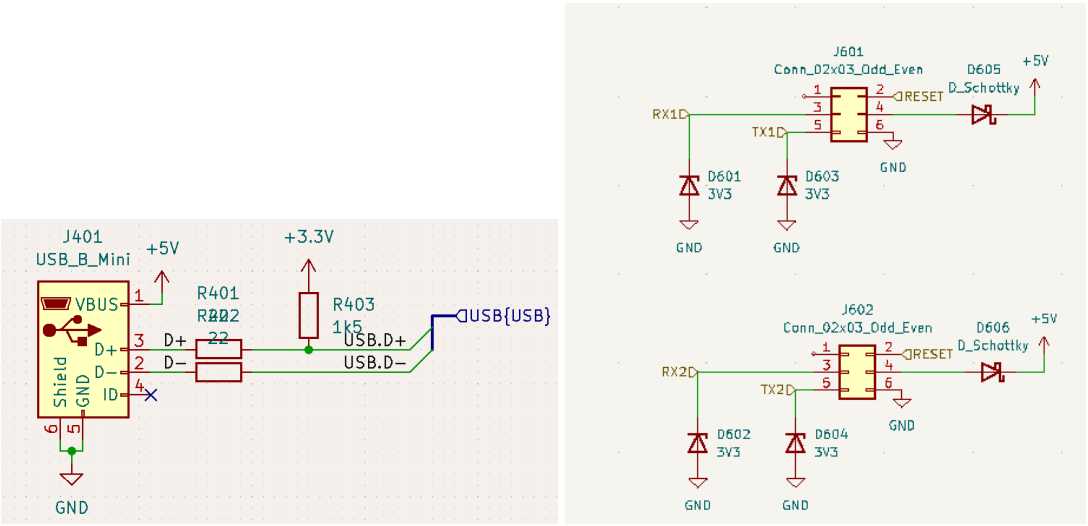

Figure 8. Port USB configuré en USB High Speed, et port série de communication

Il est également possible de communiquer avec la carte par bus CAN 10Mbps, cela permet de simplifier le câblage du robot en “chaînant” les contrôleurs. Le CAN (Signaux différentiels) est converti en CAN-TTL (TX & RX) par un SN65HVD230. Un cavalier peut être installé pour connecter une résistance de terminaison.

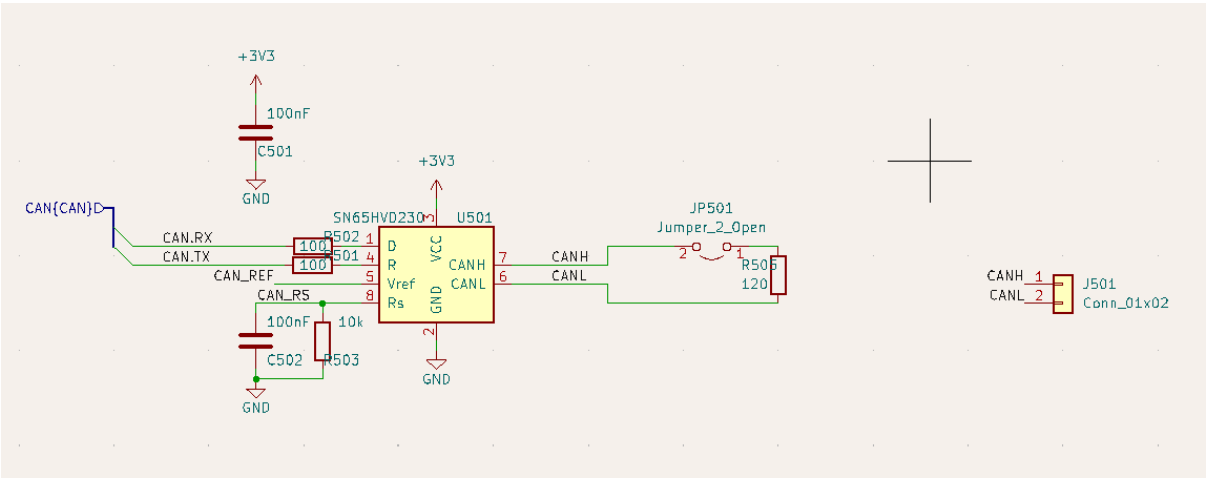

Figure 9. Transceiver CAN ⇔ CAN TTL

Un hacheur Buck convertit la tension de batterie de ~24V en 5V (commun avec le port USB), puis un Low Dropout Regulator (LDO) (AMS-1117) convertit le 5V en 3.3V pour les périphériques.

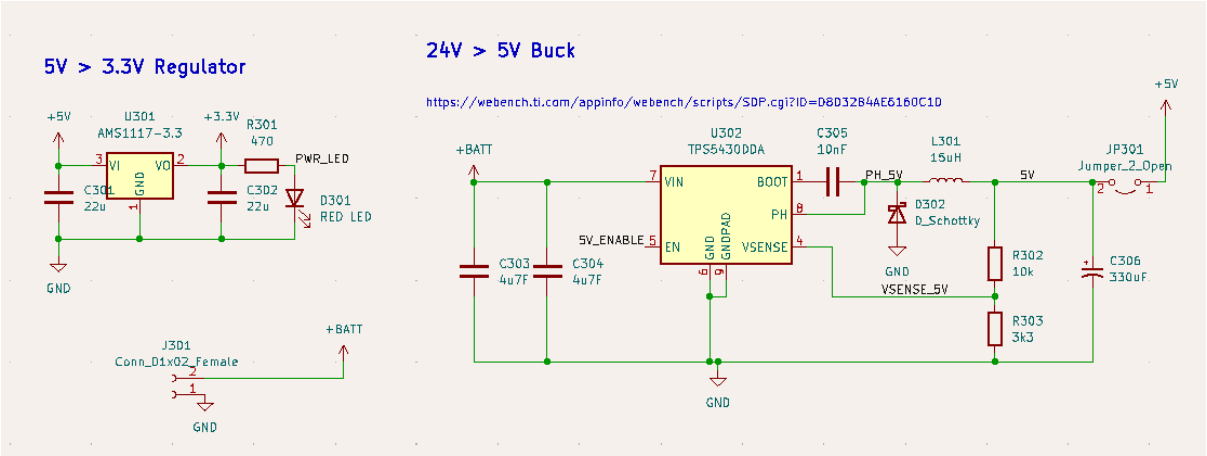

Figure 10. Schéma du Hacheur abaisseur 24V vers 5V, et dur régulateur 5V vers 3.3V

Le moteur est alimenté par 3 demi-pont en H de 6 MOSFETs, contrôlés le DRV8323, lui-même contrôlé par 6 PWM provenant du STM32. Cette configuration par MOSFETs externes pourrait permettre de contrôler des moteurs d'environ 1kW.
(Valeur théorique, n'a pu être testé : les bobinages du moteur atteignent les 150°C avant que les MOSFETs n'atteignent 30°C)

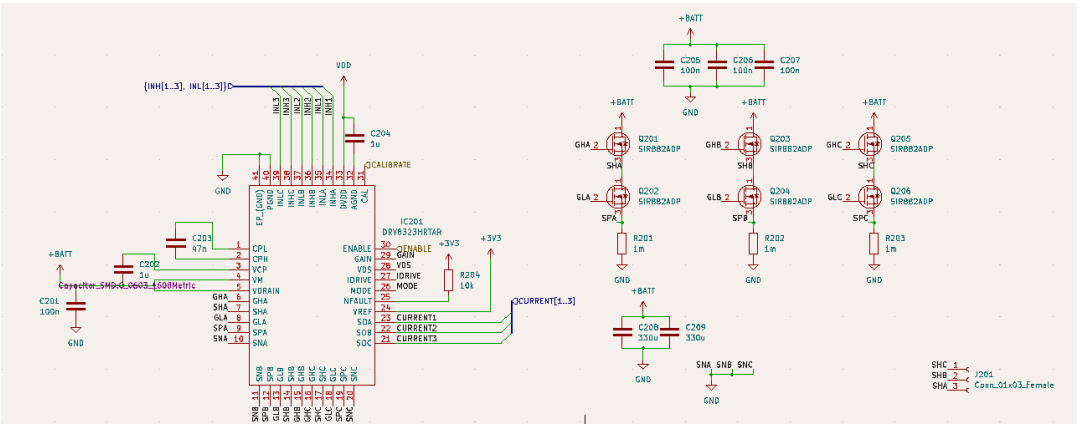

Figure 11. Schéma du driver de MOSFETs

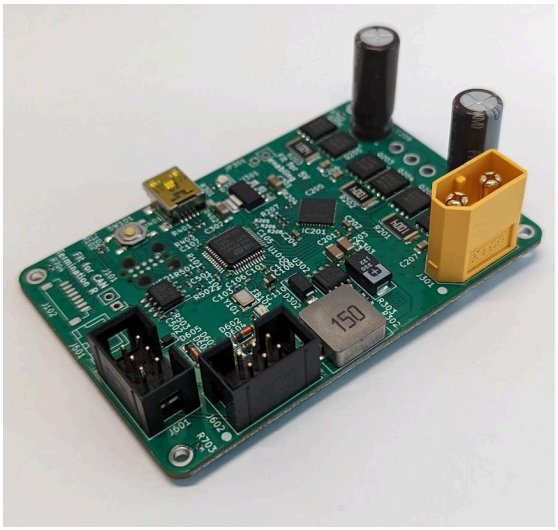

Figure 12. Carte de contrôle moteur

Firmware

Description générale

Dans le logiciel, un objet moteur est défini. Celui-ci contient tous les paramètres du contrôleur moteur. Au commencement, le moteur se place à la coordonnée 0°. Il prend une mesure par le biais de l'encodeur. C'est sa valeur de référence, son zéro-angulaire.
Par la suite, une boucle s'exécute à 10 kHz (fréquence constante assurée par FreeRTOS) :

Les paramètres du moteur peuvent être modifiés via les ports série ou l'interface CAN selon les commandes suivantes :

Adresse CAN	GCODE	Description
0x000 {vitesse}	G0 X{vitesse};	Définit la vitesse angulaire du moteur (rad/s).
0x010 {vitesse}	G10 X;	Lit la vitesse actuelle du moteur (en rad/s)
0x001 {puissance}	G1 X{puissance};	Puissance pour le moteur en condition normale (en %).
0x002 {duree}	G2 T{duree};	Définit la durée du blocage en ms.
0x003 {puissance}	G3 X{puissance};	Définit la puissance lors d'un blocage (en %)
0x004 {multiplicateur}	G4 X{multiplicateur}	Multiplicateur de la vitesse angulaire
0x005 {poles}	G5 X{poles}	Définit le nombre de paires de pôles du moteur
	G6 {ID}	Définit l'ID du contrôleur (paramètre stocké en Flash du moteur)
0x00A {kp} 0x00B {ki} 0x00C {kd}	G7 P{kp} I{kp} D{kd}	Définit les constantes du PID

Table 1. Protocoles de commande de la carte contrôleur de moteur

L'adresse évoquée ci dessus doit être ajoutée à 256 * ID du contrôleur moteur.
Ex : 0x010 = vitesse du moteur 0 ; 0x310 = vitesse du moteur 3 ;

Principe de fonctionnement

Le moteur BLDC est alimenté par ses 3 phases (A, B & C) reliées en étoile comme suit. Chacune de ces phases doit être alimentée par une sinusoïde, mais toutes déphasées de 120°. Cet enchaînement de phase crée un champ magnétique tournant au stator du moteur. Le rotor, doté de 7 paires de pôles magnétiques (grâce à des aimants permanents) suivra ce chant tournant, entraînant la rotation.
Il est à noter que 7 tours du champ tournant sont nécessaires pour faire un tour mécanique de rotor.
Ainsi, pour faire varier la vitesse du moteur, on peut faire varier la période des sinusoïdes ABC. De même, pour faire varier le couple, on peut faire varier l'amplitude de ces sinusoïdes.

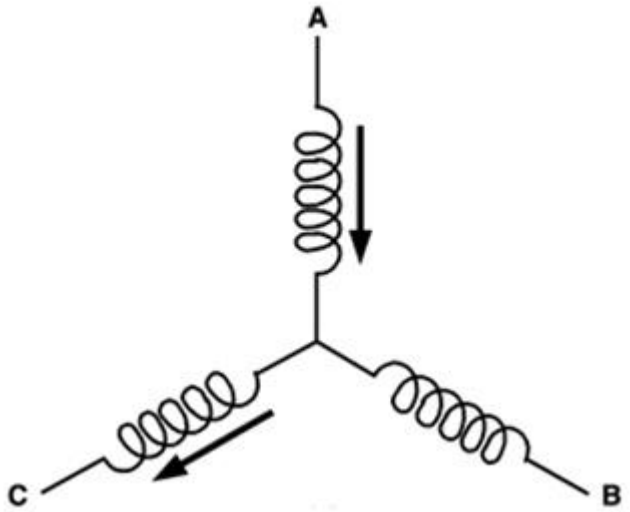

Figure 13. Modélisation du moteur BLDC

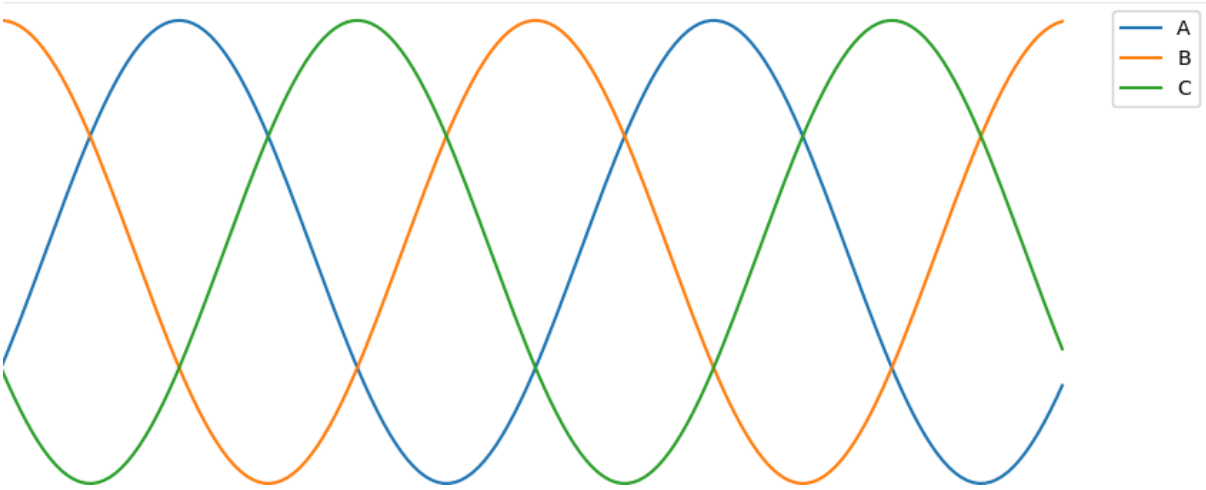

Figure 14. Sinusoïdes d'alimentation du moteur

L’intérêt de cette configuration est de pouvoir ajouter un encodeur absolu (souvent magnétique). Selon l’erreur angulaire entre la consigne et la valeur lue, nous pouvons moduler l’amplitude. Cela permet des rendements importants.

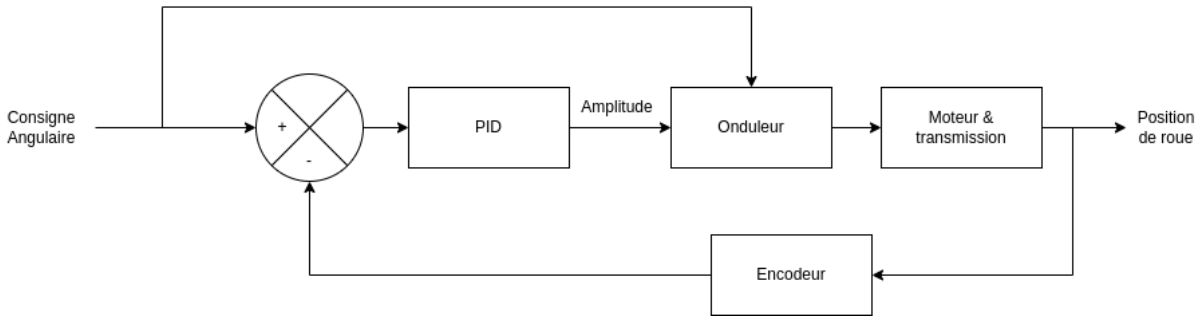

Figure 15. Boucle de régulation du moteur

Code

Une structure C ‘moteur’ contient tous les paramètres pouvant être définis plus hauts.
Une fonction update est appelée périodiquement (10kHz) par FreeRTOS. Celle-ci vient lire la position de l’encodeur, incrémenter la consigne d’angle selon la vitesse, et effectuer les opérations de calcul décrites ci-dessus.
En résulte 3 valeurs flottantes (intensités à un instant t pour chacune des phases).
Ces valeurs sont converties en 6 PWM synchronisées (2 MOSFETs pour chaque phase, un amenant la ligne à la masse, un à la tension de batterie).

Une phase d’initialisation a lieu à l'alimentation du contrôleur : un champ à 0° est appliqué au moteur pendant 500ms pour lire la valeur de l’encodeur, et en faire le zéro. En effet, l’encodeur peut être déphasé de par la position de l’aimant/capteur sur le moteur.

Des interruptions sur les ports série viennent alimenter différents buffer pour être parsé lors de “temps libre” du contrôleur moteur. Une fois parsés, les paramètres sont injectés dans la structure ‘moteur’.

Un mécanisme d'anti blocage est nécessaire. Il semblerait que les paires de pôles ne soient pas homogénéiquement réparties sur le rotor du moteur. Cela pourrait être dû à des différences dans les performances des aimants permanents. Pour se faire, après un certain temps où le moteur ne bouge pas, la consigne angulaire est augmentée afin de “sauter” cette zone où les champs sont plus faibles.

Pour que chaque adresse sur le bus CAN soit unique malgré la présence de plusieurs contrôleurs moteurs, ces derniers ont tous un identifiant supposé unique, qui peut leur être programmé via interface série. Cet identifiant est stocké dans la flash de programme par le code lui-même, et chargé au boot.

Software

Contexte

La commande de vitesse se fait par le biais de la carte mère. Sur celle-ci se trouve ROS2, un Framework tourné vers la robotique. Nous pouvons reprendre toute la partie navigation de l'année précédente et venir modifier uniquement la partie qui était propre aux moteurs du précédent robot.

Chaîne de transmission

Le Raspberry Pi Compute module 4 sur lequel tourne ROS2 n'a pas la possibilité de communiquer sur un bus CAN. C'est pourquoi un STM32F103 est présent sur la carte mère. Celui-ci dispose de plusieurs ports série, ainsi que d'un périphérique CAN-TTL.
Les deux communiquent via un port série et un port USB (ce dernier ne servant que pour la mise à jour du STM32).
On peut donc faire communiquer le Raspberry PI CM4 et les contrôleurs moteurs comme suit :

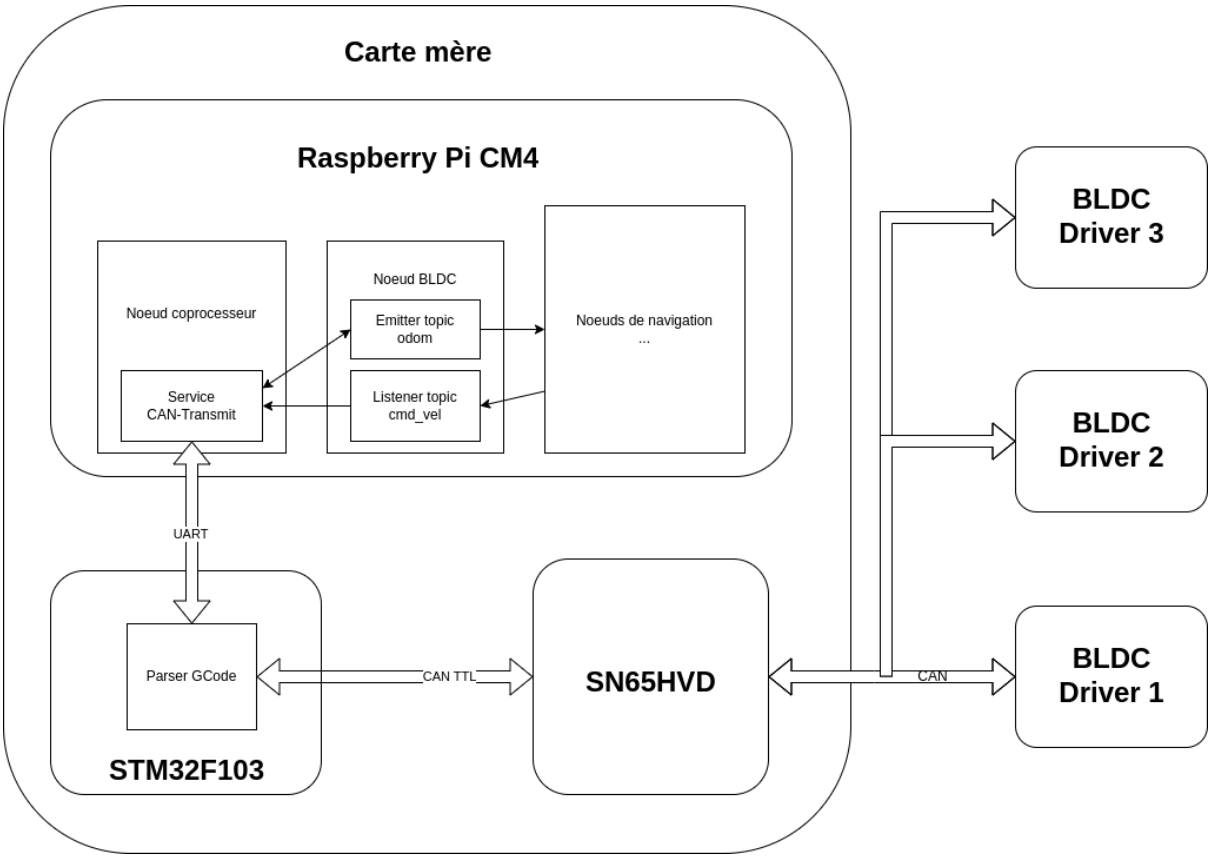

Figure 16. Schéma de la communication interne du robot, du micro-ordinateur vers les contrôleurs de moteur

Noeuds ROS2

Un nœud permettant la communication entre le STM32 et le Raspberry Pi avait déjà été réalisé l'an passé. Il permettait, entre autres, de contrôler des entrées sorties, de lire la tension de la batterie et de contrôler la vitesse du ventilateur du raspberry. La communication se fait par envoi de trames GCode. La solution à cela est d'implémenter une nouvelle commande GCode sur le STM32 qui envoie des messages sur le bus CAN.

GCODE	Description
G80 A{adresse} W{valeur};	Écrit à l'adresse {adresse} la valeur {valeur} sur le bus CAN.
G81 A{adresse} I{id};	Lit à l'adresse {adresse} sur le bus CAN, et renvoie un message : G81 V{valeur} I{id}; (l'id permet des lectures asynchrones).

Table 2. Commandes ajoutées au Coprocesseur STM32 de la carte mère

Ce nœud se chargeant de la liaison série offre déjà plusieurs services (fonctions pouvant être appelées par d'autres nœuds) pour les fonctions évoquées plus haut. De la même manière, 2 services ont été créés pour la lecture et l'écriture sur le bus CAN. (services appelés can_write, prenant un paramètre int_8[], les octets à écrire, et can_read, retournant un int_8[]).

Pour venir appeler ces services un nouveau noeud est créé :
Il vient écouter les messages sur le topic standard cmd_vel. Ces messages représentent une commande de vitesse dans un repère cartésien relatif au repère du robot.
Le noeud vient traduire cette consigne en vitesses pour les 3 moteurs

$$\begin{bmatrix} \omega_A \\ \omega_B \\ \omega_C \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{R} & -\frac{L}{R} \\ \frac{\sqrt{3}}{2R} & -\frac{1}{2R} & -\frac{L}{R} \\ -\frac{\sqrt{3}}{2R} & -\frac{1}{2R} & -\frac{L}{R} \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ \omega \end{bmatrix}$$

Figure 17. Relation entre consigne de vitesse cartésienne et vitesse des roues.

Où :
ω_N est la vitesse de la roue N
L la distance entre la roue et le centre du robot
R le diamètre de la roue
Une fois les 3 vitesses des roues calculées, le service de transmission CAN est appelé 3 fois pour communiquer les vitesses au moteur.

Périodiquement, ce nœud vient lire les vitesses réelles des moteurs et évaluer sa propre odométrie, pour l'envoyer sur le topic standard 'odom'.

Par la suite, des nœuds standard de ROS2 intégrant les algorithmes de pathfinding peuvent écouter et écrire sur ces topics pour commander le robot pour l'emmener à des coordonnées fixées.

Mécanique

Chaîne de transmission de puissance

Un réducteur par courroie a été conçu par Clovis Charrier (Méca 4).
Deux arbres en acier viennent maintenir une poulie chacun dans deux paliers découpés dans des plaques de PMMA.
Pour ajouter un encodeur, l'arbre du moteur a été modifié pour inclure un aimant néodyme diamétrique.
Le PCB de l'encodeur est fixé à l'arrière du support de moteur.

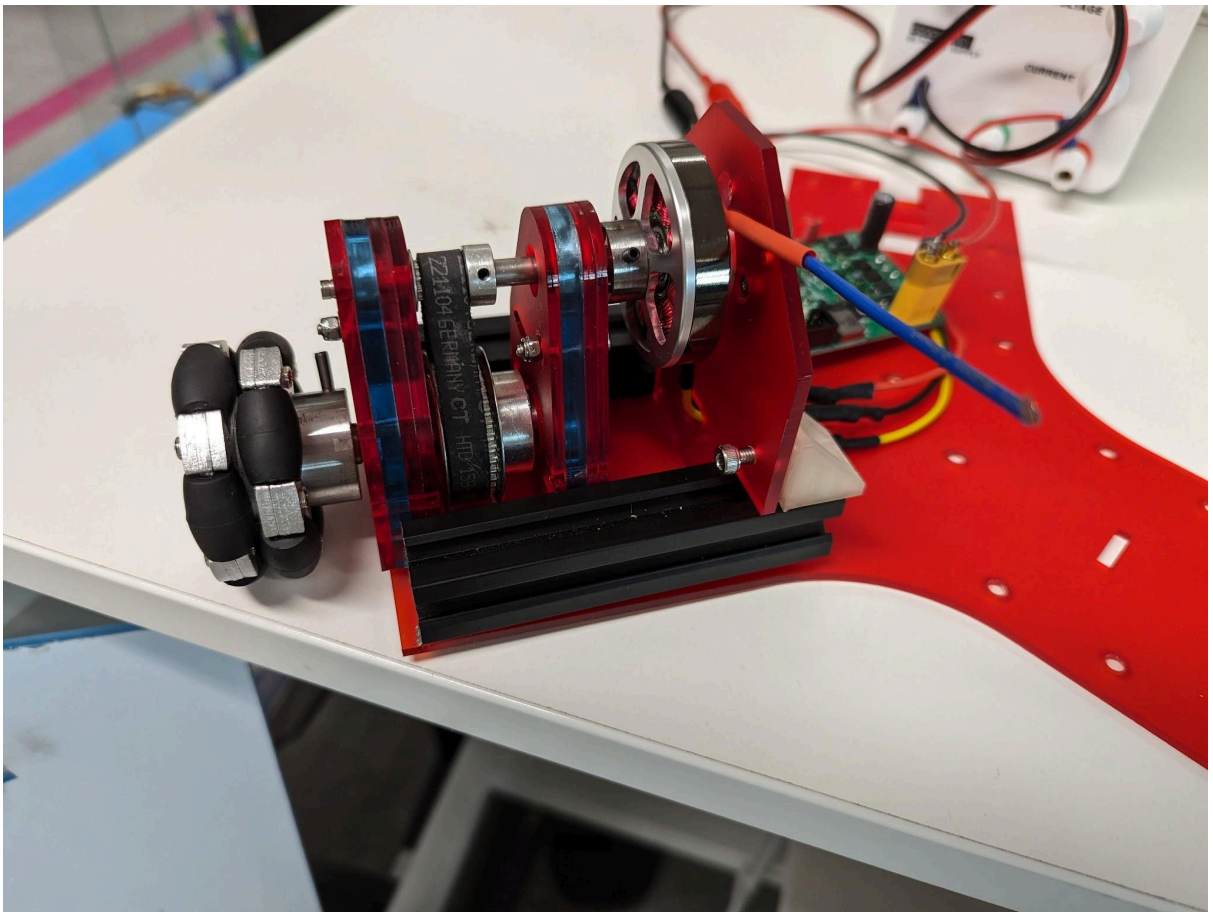

Figure 19. Chaîne de transmission de puissance mécanique.

PAMI

Contexte général

Les PAMI (Petits automates motorisés indépendants) sont des petits robots autonomes qui sont actifs dans le match à partir de la 90ème seconde et sur une durée de 10 secondes.
Leur objectif est de rentrer en contact avec des objets rangés dans les zones de dépôt par le robot principal pour rapporter des points bonus.
Les règles de sécurité propres à tous les robots s'appliquent également aux PAMI :
Ils ne doivent pas entrer en contact avec un robot adverse, doivent disposer d'un bouton d'arrêt d'urgence, et doivent être déclenchés à la suite d'un tirage d'un câble (90 secondes après le déclenchement).
Puisqu'une équipe peut proposer jusqu'à 15 PAMIs, il est nécessaire que le prix de ceux-ci soit faible si l'on veut en proposer un maximum au cours de la compétition.

Hardware

Principe de fonctionnement

Le STM32F103 est le microcontrôleur au coeur du PAMI car celui-ci :

- est basse consommation, permettant plusieurs tests
- Dispose de nombreuses entrées sorties et périphériques :
    - 1 port USB pour le débugger/mettre à jour facilement
    - 2 ports I2C
    - 1 port SPI
    - De nombreuses interruptions
- Les programmes ayant été écrits sur d'autres plateformes du projet peuvent y être portés facilement.

Le contrôleur de moteur est un DRV8837 pour moteurs DC.

- Ils intègrent un pont en H pouvant commuter une charge jusqu'à 1.8A à de 0 à 11V ;
- Ils peuvent être mis en veille pour une consommation de ~30nA ;
- Ils sont peu onéreux (10cts pour 50 unités) ;
- Ils sont simples à interfacer (2 PWM) ;
- Ils utilisent des moteurs DC : existent en petits formats, en composant "off the shelf", et avec des réducteurs, pour très peu cher (2€).

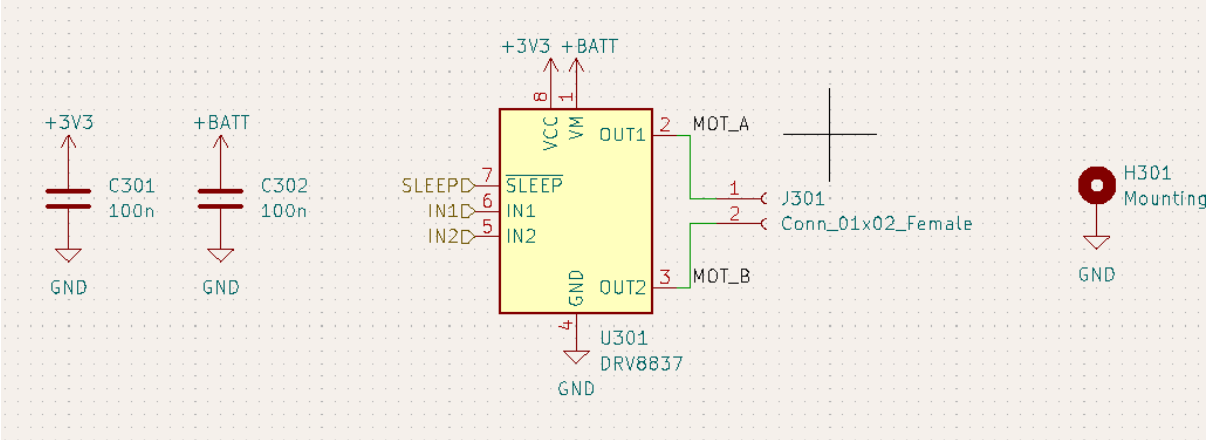

Figure 18. Schéma du pont en H commutant le moteur DC

Les capteurs de distance sont des VL53L1. Les avantages de ces capteurs sont qu'ils sont suffisamment immunes au bruit (très présent le jour de la compétition) et simples à interfacer en I2C. Jusqu'à 3 capteurs peuvent être placés sur un PAMI, mais pour des contraintes de coût, un seul est présent. Les GPIO d'évènements peuvent être reliées au STM32 par le biais du jumper JP101.
Ces cartes servent également à relier la carte principale à la carte d'extension.

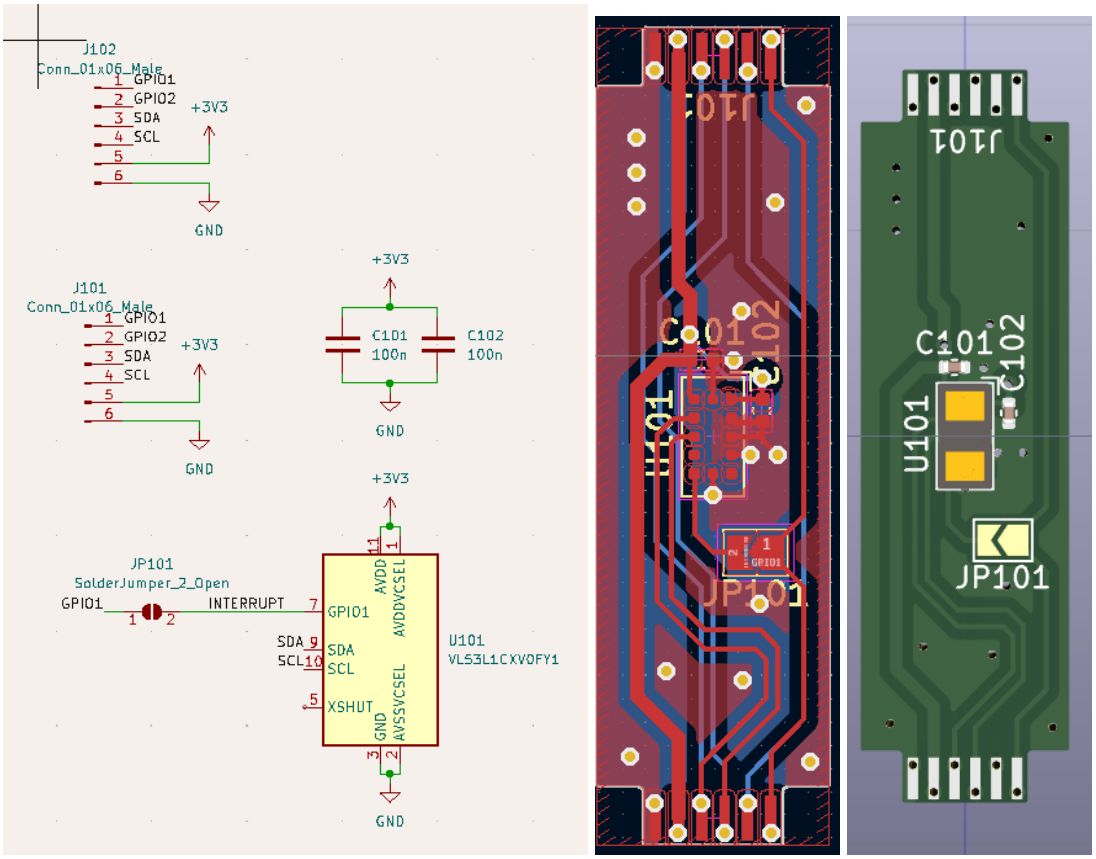

Figure 19-29-21. Carte de capteur de distance

Pour permettre une diversité d'alimentation (une batterie pendant le développement, et des piles rapides à changer pour la compétition), le PAMI embarque un convertisseur Buck/Boost TPS6305 pour générer son rail 3.3V depuis une alimentation 2.5V jusqu'à 6V. Il embarque également un chargeur de batterie li-ion MCP73831 désactivable si alimenté par des piles.

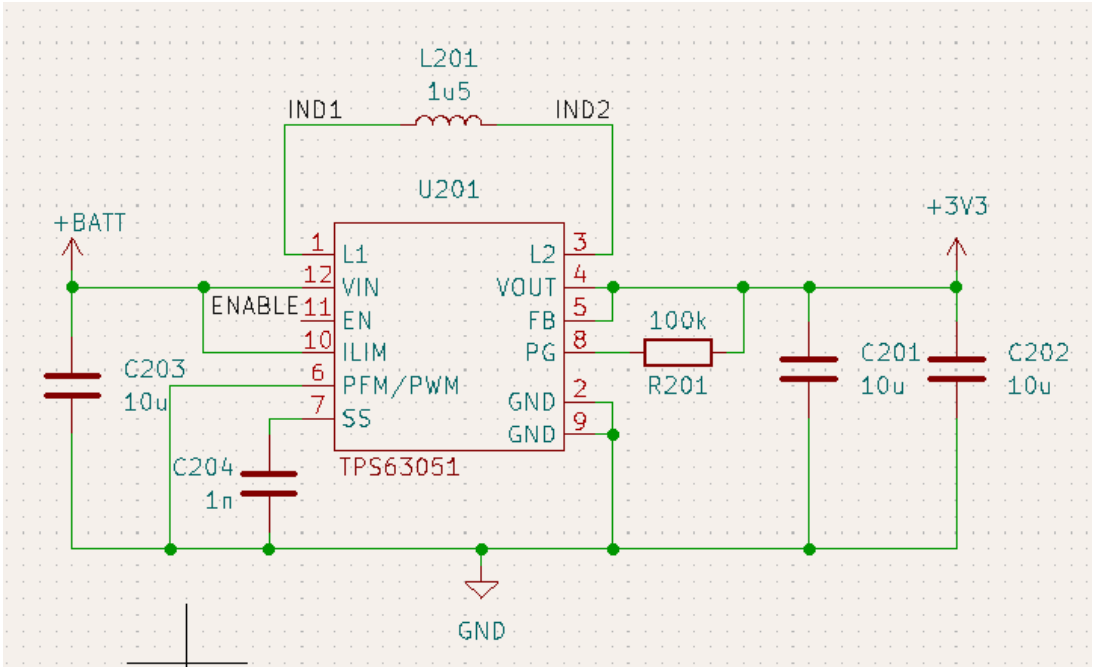

Figure 22. Hacheur abaisseur-élévateur de tension

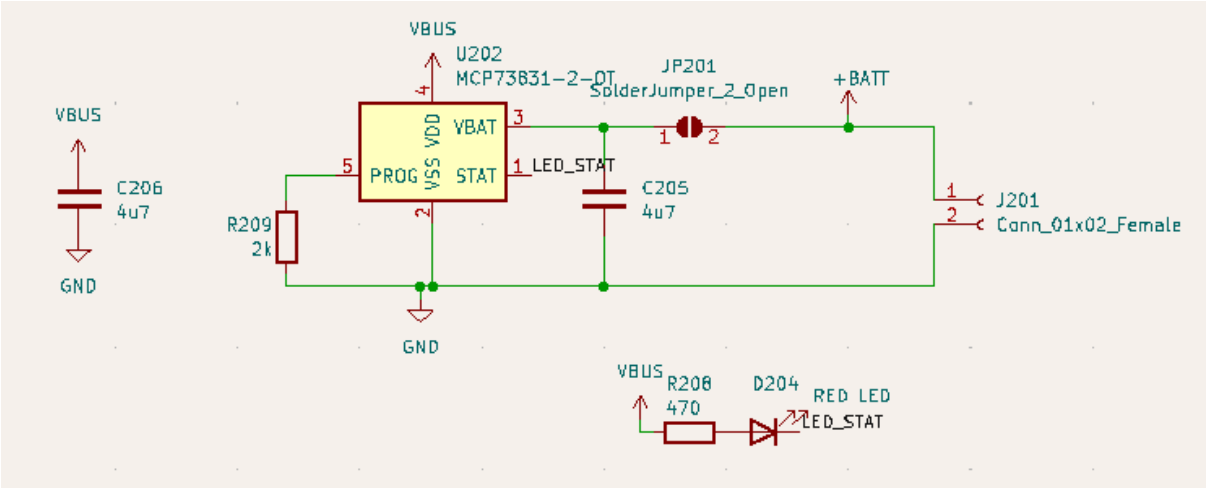

Figure 23. Circuit de charge de la batterie

En désactivant le TPS (pin ENABLE à la masse), la consommation à vide est de l'ordre du nanoampère. Pour activer le TPS et donc la carte, 3 méthodes existent :

- Appuyer sur un bouton poussoir
- Brancher le port USB
- Que le microcontrôleur maintient une GPIO à l'état haut.
    - De ce fait, à l'allumage par un bouton, le microcontrôleur attend un certain temps puis de lui-même maintient le hacheur actif. Autrement, il s'éteindrait dès que le bouton serait relâché.

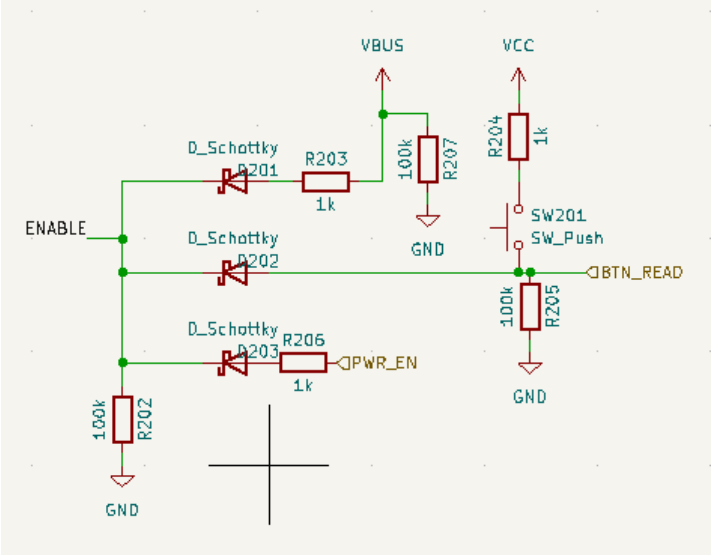

Figure 24. Schéma du ou logique à diode permettant d'activer le hacheur abaisseur-élévateur

Enfin, pour assurer la communication avec le robot principal, il embarque un CC1101 réglé pour communiquer à 815 MHz.
En effet, le Raspberry Pi de la carte principale peut communiquer en WiFi, cependant, la bande de fréquence 2.4GHz est surchargée lors de l'évènement. (ainsi que la bande 5.8GHz & 433 MHz).
Un réseau d'adaptation d'impédance est présent à la sortie de la puce, menant vers un connecteur SMA.

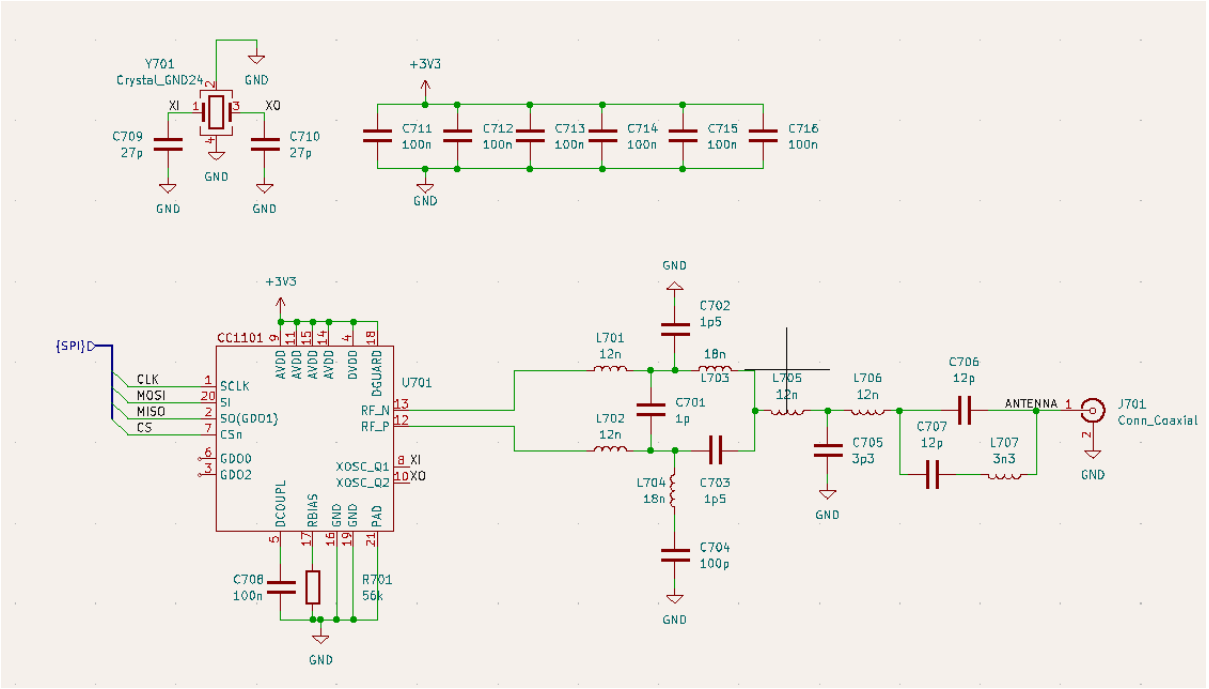

Figure 25. Circuit de communication Sub-GHz

Mécanique

Pour réduire les coûts, le PCB est également le châssis du robot. Les moteurs viennent directement se visser sur le PCB. De même, une roue folle est installée à l'avant du robot dans le PCB. Les PCB des capteurs de distance servent à maintenir un PCB secondaire, qui ne sert qu'à maintenir le boîtier à piles et pour y mettre des connecteurs d'extension.
Il permet également d'accrocher un tag Aruco (QR code simplifié sans code correction d'erreur) qui sert à localiser le PAMI.

Clé Radio

Une clé USB a été conçue pour pouvoir interfacer les PAMI avec n'importe quel ordinateur, y compris la carte mère du PAMI. Celle-ci dispose du même microcontrôleur et puce radio que les PAMI pour simplifier le développement.

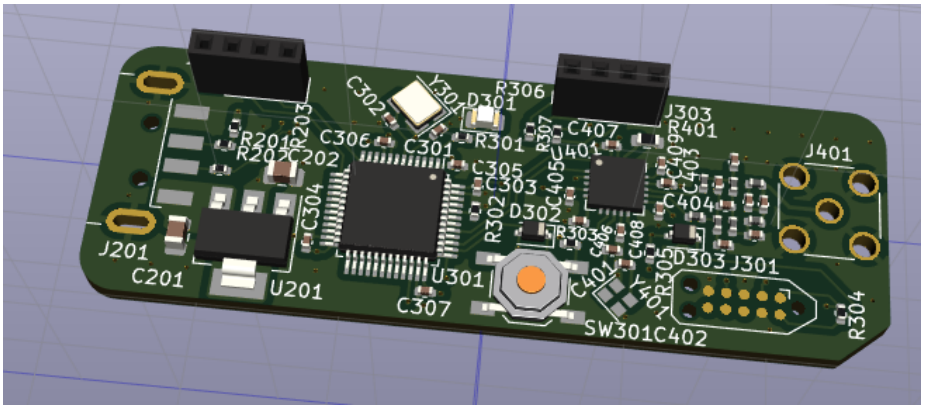

Figure 26. PCB de clé radio

Firmware

Le STM32 est lui aussi programmé via le STM32CubeIDE. Il vient lire une consigne qui lui est fournie par le CC1101. Cette consigne lui est transmise par un paquet consistant de 3 octets : un premier donnant l'ID du PAMI à laquelle s'adresse le contrôleur, un octet pour la vitesse du moteur gauche, un octet pour la vitesse du moteur droit.
Cette vitesse est prohibée par la présence d'un obstacle détecté par le capteur de distance, ou par le bouton d'arrêt d'urgence.

De plus, à l'allumage, le STM32 fait clignoter une LED pendant 2 secondes pour indiquer son état, puis active la broche ENABLE de l'alimentation, pour se maintenir allumé. Cela permet au PAMI de s'éteindre lui-même en passant cette broche à l'état bas. Grâce à ça, un message peut être envoyé pour éteindre tous les PAMI à la fin de la manche, pour garantir que les robots ne se déplacent pas.

Logiciel

Le logiciel a principalement été rédigé par les SE3 de Robotech : Victorien Détrez, Valentin Piroux & Ibrahim Tepeli

Sur le bord de l'aire de jeu se trouve une zone où une caméra peut être installée. Celle-ci vient reconnaître les tags Aruco propres à chacun des PAMI via un script python et la librairie openCV. Une transformation pour corriger l'effet de Fisheye de la caméra et une transformation affine vient corriger l'image. Une fois le code du tag Aruco reconnu, le script guide les Pami en les alignant vers leurs cibles et en les faisant avancer. Ces informations sont envoyées via la clé USB radio, qui se présente comme un port série

A l'avenir, il serait pertinent que le robot principal communique les objectifs à la tour d'observation pour que les PAMIs n'aillent qu'à des endroits où le robot principal a déposé des objectifs.

Cette reconnaissance des tags Aruco sert également pour détecter l'orientation des panneaux solaires (éléments à venir orienter)

Logiciel Divers

Reconnaissance d’images

Le logiciel de cette partie a principalement été rédigé par Benjamin Cart, SE3 de Robotech.

Un des éléments initiaux du cahier des charges était la reconnaissance des pots de plantes-objectifs. En effet, les plantes à récupérer peuvent être de deux types : plante faible (pot blanc) ou plante forte (pot violet). Initialement, nous envisagions d’utiliser un lecteur NFC afin de différencier ces deux types. Cependant, nous avons pivoté vers une méthode basée sur l’IA pour reconnaître les différentes plantes, et axer le préhenseur vers celles-ci.

Nous utilisons le modèle YoloV8 qui a déjà, parmi les éléments qu’il est capable de prédire, les plantes que nous devons récupérer. Ce modèle nous donne les “bounding boxes” des plantes dans l’image (sorte de cadre autour de l’objet). De ces coordonnées nous pouvons aller échantillonner la partie basse pour chercher une couleur (soit violet soit blanc), et traduire ces coordinées dans le repère de l’image en orientation pour notre robot.

Figure 27. Démonstration du modèle de reconnaissance des plantes

Détection de bordure

Le logiciel de cette partie a principalement été rédigé par Rémi Giner, SE3 de Robotech.

Trois capteurs 3D sipeed A010 seront installés sur le robot. Ils renvoient une image de 100 par 100 pixels, indiquant non pas la couleur d'un élément mais sa distance.

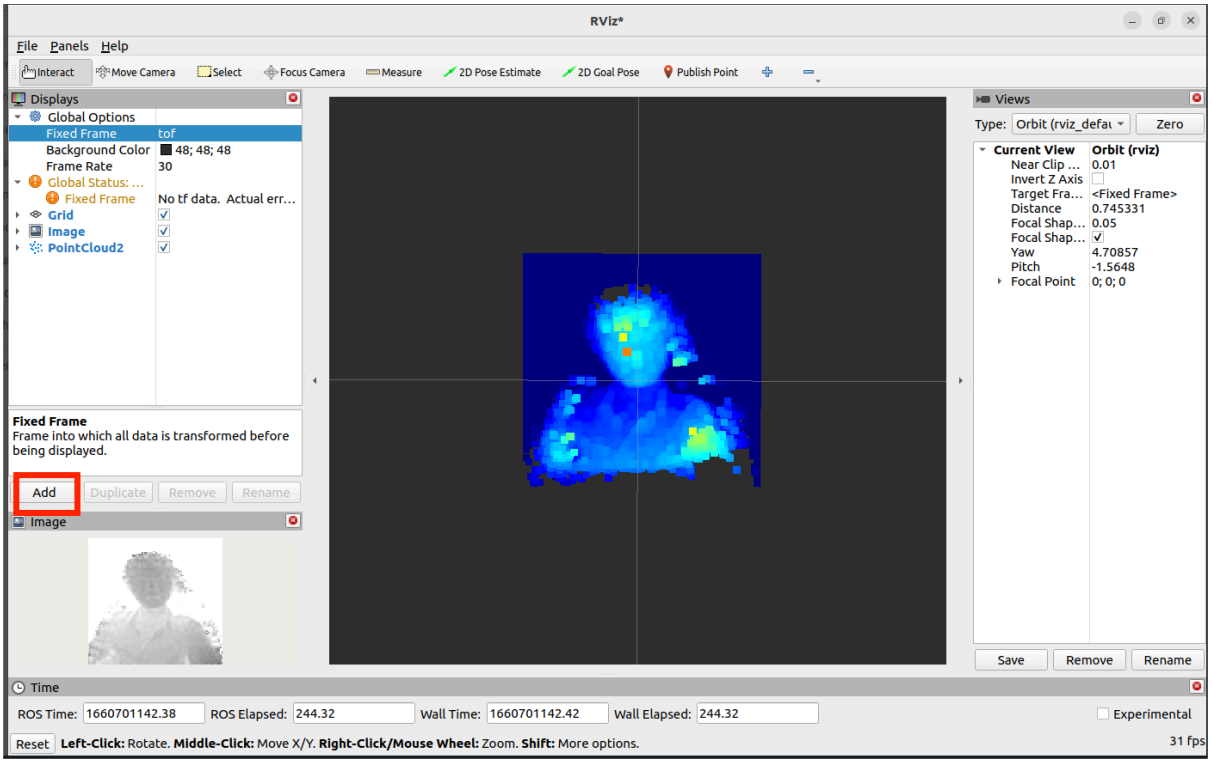

Figure 28. Données retournées par le capteur 3D

Nous utilisons cependant des techniques conventionnelles d'analyse d'image (Détection de Canny et algorithme de Hough) pour détecter les bordures du terrain. A partir des angles & des distances des bordures relatifs au robot, nous pouvons estimer une position de celui-ci.

Analyse et résolution de problèmes

Driver de MOSFETs en erreur

Une source de mon retard a été le diagnostic de la carte de contrôle des moteurs BLDC.
J'ai longtemps eu du mal à commuter les phases du moteur, car le contrôleur de MOSFETs DRV8323 rapportait une erreur.
Cela se traduit par une impulsion sur la broche de diagnostic lors de la commutation.
J'ai tout d'abord changé les résistances de configuration, afin d'augmenter la tolérance du contrôleur moteur sur ses sécurités, jusqu'à les désactiver.
Cela n'ayant pas suffi j'ai pu observer que seuls les MOSFETs reliant la phase à la masse déclenchent cette erreur, et que l'erreur apparaissait 4µs après la consigne émanant du STM32.
Suite à des échanges avec des ingénieurs de Texas Instruments, et par analyse du diagramme du circuit de protection, j'ai pu constater que ma résistance de mesure de courant était mal dimensionnée, et empêchait la mise à la masse du drain du MOSFET.
En effet, suite à une erreur dans la nomenclature, la résistance de Shunt R_{SENSE} n'était pas de 1m Ohm mais 1M Ohm.

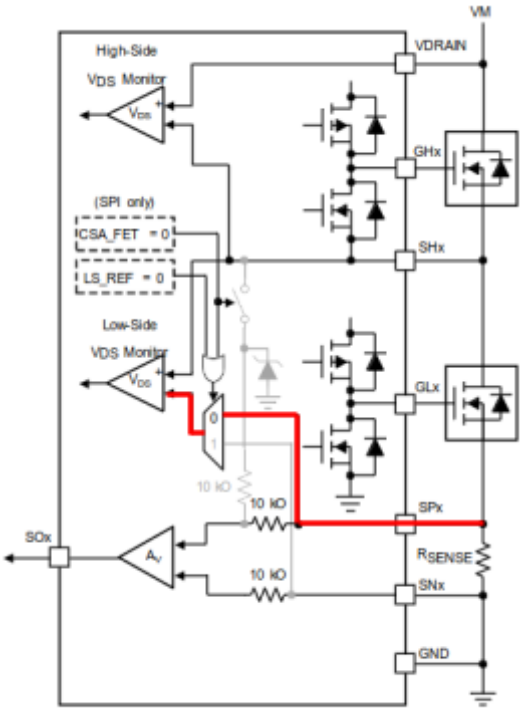

Figure 41. Resistor Sense Configuration

Figure 29. Schéma du fonctionnement de la fonction de détection de circuit ouvert à VDS

Blocage moteur

Lors du développement du moteur, j'ai rencontré des blocages. Lorsque le moteur est contrôlé en boucle ouverte (consigne angulaire qui s'incrémente de façon constante, sans lecture du capteur), le moteur tourne correctement, même à faible vitesse.
Lorsqu'il est placé en boucle fermée (lire la position du moteur via l'encodeur, et incrémenter cet angle par une constante), le moteur venait se bloquer.
Cela peut s'expliquer par le fait qu'avec un incrément trop faible, ou pointant vers l'écart entre deux aimants empêche le rotor de tourner.
C'est pour ça que j'ai dû mettre en place un anti-blocage dans le code.

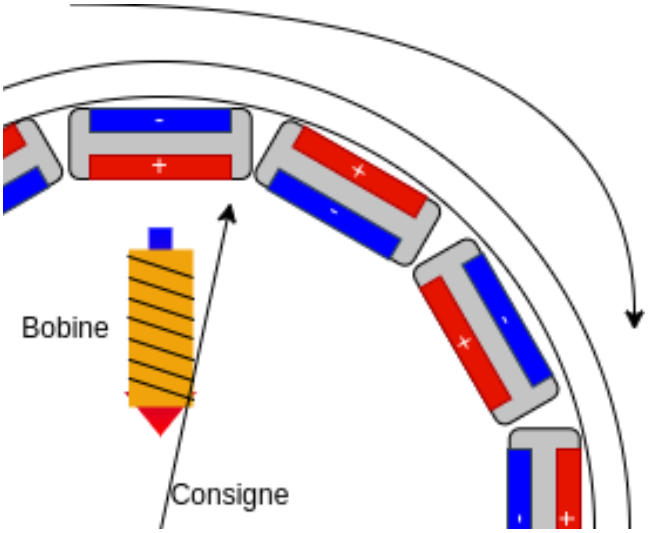

Figure 30. Schéma du blocage moteur : la consigne entraîne la bobine vers une zone entre 2 aimants permanents.

Avant de découvrir cette erreur, j'ai pu constater que les différentes PWM que j'utilisais pour contrôler le driver de MOSFET n'étaient pas synchronisées, car elles utilisaient des Timer différents. Cela causait des moments où le moteur était mal-alimenté.
J'ai pu trouver ce problème en modifiant le moteur pour avoir accès au point central du moteur. J'ai donc pu brancher un oscilloscope sur les 3 phases, et la référence sur ce point central, et constater le problème.

Bilan du projet

Discussion et Suite du projet

Les performances du contrôleur moteur ne sont pas encore au niveau attendu.
La technique de déblocage vient perturber la régulation de vitesse, causant des accoups, et une vitesse mesurée loin de la consigne initiale.
C'est pourquoi j'ai pu commander de nouveaux moteurs pensés pour le FOC pour tester si mes algorithmes sont pertinents et fonctionnels dans de meilleures conditions. Ces nouveaux moteurs sont conçus pour être utilisés dans des stabilisateurs pour de la vidéo, où un aucun accoup n'est tolérable.
Un contrôleur de moteur BLDC FOC de la marque ODrive dans le format qui nous intéresse vient d'être commercialisé ce mois-ci. J'ai pu en commander afin de tester si cette solution est pertinente dans notre cas.
Si cette solution ne marche pas, je pourrais adapter mes travaux de contrôleur de moteur pas à pas qui ont été éprouvés, pour en faire une carte de contrôle basée sur la même architecture que celle de moteur BLDC.

Je reprendrais également les travaux de l'équipe afin de finaliser le système de navigation. L'odométrie du robot de l'an dernier était mauvaise de par la nature des roues choisies. Ce système à 3 roues semble prometteur, tout comme le système de visualisation des bordures du terrain.

Une fois les financements reçus, je commanderai le terrain afin d'effectuer les prochains tests en conditions réelles.

Nous avons prévu de motoriser les actionneurs avec des moteurs pas à pas. Je porterai donc ma carte de contrôle de moteurs DC de l'an dernier pour les supporter, avec des drivers Trinamic TMC2208 (design déjà validé et éprouvé).

En conclusion et comme énoncé plus haut le projet n'est pas fini. Nous n'avons pas encore de base roulante éprouvée en février.
Cependant, de part l'aspect modulaire de la base de code de l'an dernier, nous pourrons très vite programmer le robot dès que la base roulante sera terminée.

Leçons et Apprentissages

Ce projet m’a permis de mettre en pratique de nombreuses compétences :
Techniques :

- Apprentissage à propos de la théorie des moteurs synchrones, chaîne de transmission, impédance et traitement du signal
- Conception de PCB à haut courant (aspect thermique et compatibilité électromagnétique)

Transversales :

- Travailler avec une nouvelle équipe
- Gestion du temps
- Résolution de problèmes
- Gestion de budget
- Définition d’un cahier des charges

Comportementales :

- Capacité à faire face à des imprévus/problèmes, réactivité
- Relation Client (dans le cadre de la mission « Free-lance »)
- Gestion d’équipe
- Capacité à former d’autres personnes à des outils (KiCAD, système Linux, ROS2)

Après cette expérience, je décide d'opter pour des technologies que je maîtrise parfaitement, afin de progresser de manière plus assurée dans mes projets de conception de contrôleurs de moteur.

Pour ce projet d'ingénierie, l'orientation vers le recherche plutôt que le développement à entraîné des retards pour l'équipe. Néanmoins, cette décision a constitué une opportunité précieuse d'apprentissage et de progrès technique.

J’aurais aimé pouvoir me concentrer davantage sur la gestion d’équipe, et mettre en place des rituels Agile comme vu en gestion de projet, et mettre en place un système de ticketing, tout en donnant des tâches plus petites à chaque individu pour les former point par point.