ENSTA Bretagne
2, rue François Verny
29806 Brest cedex
France
Tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr

Rapport de stage

FISE 2024

17 octobre 2023

Rapport de stage de 2A

STIIMA

Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato

Consiglio Nazionale delle Ricerche

L. RAVAIN
louis.ravain@ensta-bretagne.org

Remerciements

Je tiens à remercier Arianna R., ma collègue, pour le temps et les conseils qu'elle a pu me donner, ainsi qu'Antonio P., pour l'accueil chaleureux qu'il m'a fait, sa bonne humeur, et l'expérience qu'il a partagé avec moi. Je remercie également Annalisa M., pour m'avoir accepté dans le laboratoire et l'encadrement qu'elle a effectué.

Enfin, je remercie tout les membres du laboratoire pour l'accueil qu'ils m'ont fait pendant ce stage et les conseils prodigués.

Résumé

Pour répondre aux besoins de modernisation de l'agriculture italienne, le laboratoire de robotique mobile du STIIMA-CNR a développé, en partenariat avec ROBODYNE, un robot quatre roues motrices à l'orientation contrôlable indépendamment les unes des autres. Ce robot est utilisé pour tester de nouvelles méthodes de surveillance de l'hydratation des champs, de la qualité des fruits ou encore de l'état de santé des vignes. Mais le développement est encore en cours, et c'est sur ce projet que j'ai été pris en stage. Mon rôle a été d'aider l'équipe de recherche à implémenter ces algorithmes. Dans ce cadre, j'ai mis en place le système de localisation par GPS et RTK, ce qui permet aux chercheurs d'obtenir une précision de l'ordre du centimètre. Pour ce faire, j'ai réalisé la configuration initiale des cartes GPS pour que la correction RTK fonctionne dès l'arrivée sur zone d'expérimentation. J'ai également écrit le guide d'utilisation pour qu'à l'avenir, si problème il y a, mes anciens collègues puissent facilement et rapidement le résoudre.

La localisation n'était pas le seul problème rencontré lorsque je suis arrivé. En effet, les algorithmes permettant le contrôle manuel du robot n'étaient pas suffisant. Il m'a donc été demandé d'améliorer ces algorithmes, tâche que j'ai accompli en implémentant un système de contrôle des angles de rotation des roues basé sur la géométrie différentielle d'Ackermann. De la même manière que précédemment, un guide d'utilisation du code a été produit pour une réutilisation et/ou correction aisée.

Enfin, sur mon temps libre, j'ai tenté d'implémenter un algorithme de SLAM, StellaVSLAM, pour améliorer la perception du robot dans l'espace et mieux localiser les points remarquables, ainsi que permettre à l'opérateur du robot de reconstituer l'environnement du robot.

Abstract

To address the modernization needs of Italian agriculture, the mobile robotics laboratory of STIIMA-CNR, in partnership with ROBODYNE, developed a four-wheel-drive robot with independently controllable orientations. This robot is used to test new methods for monitoring field hydration, fruit quality, and vine health. However, the development is still ongoing, and it is on this project that I undertook my internship. My role was to assist the research team in implementing these algorithms. In this context, I set up the GPS and RTK-based localization system, enabling the researchers to achieve centimeter-level accuracy. To accomplish this, I performed the initial configuration of the GPS cards to ensure that the RTK correction worked upon arrival at the experimental site. I also authored the user guide so that in the future, if any issues arise, my former colleagues can easily and quickly resolve them.

Localization was not the only challenge encountered upon my arrival. Indeed, the algorithms for manually controlling the robot were insufficient. Therefore, I was tasked with enhancing these algorithms, a task I accomplished by implementing a wheel rotation angle control system based on Ackermann's differential geometry. Similarly to before, a code usage guide was produced for easy reuse and/or modification.

Finally, in my free time, I attempted to implement a SLAM algorithm, StellaVSLAM, to improve the robot's spatial perception, better locate key points, and enable the robot operator to reconstruct the robot's environment.

Table des matières

1 Présentation du stage, des objectifs et des enjeux

1.1 Présentation du contexte du stage

FIGURE 1 – Le laboratoire de robotique

Le laboratoire de robotique mobile est dépendant de l'Institut des Technologies Industrielles Intelligentes et des Systèmes pour la fabrication avancée, lui-même un organisme du CNR (Consiglio Nazionale delle Ricerche ou Conseil National des Recherches, l'équivalent Italien du CNRS). Ce laboratoire est spécialisé dans la vision par ordinateur et, dans le cadre du projet ATLAS (dont l'objectif est d'améliorer la production agricole et faciliter le travail des agriculteurs à travers une robotisation de la main d'œuvre), de la capture et de l'échange d'informations entre les différents systèmes agricoles.

Il s'agit donc d'un institut de recherche publique, aux financements publiques et notamment régionaux, nationaux et européens. Cette orientation recherche, mais avec un objectif bien concret, à l'échéance relativement courte m'a beaucoup intéressé. En effet, le choix de la filière Robotique Autonome s'est fait en partie grâce à la grande part de concret, de palpable, dans cette formation. Ce projet d'innovation, m'a donc particulièrement motivé à candidater puisqu'il mélange recherche dans un milieu universitaire, mais reste proche de la réalité, du concret. C'est donc dans le cadre de ce projet ATLAS que je suis arrivé dans ce laboratoire, qui est en train de monter en compétences dans le domaine de la robotique et la prise en main du robot 4WD eXplorer. Mon rôle était d'aider les chercheurs à intégrer le robot et les capteurs (GPS, accéléromètres, Caméras, moteurs) au sein d'un écosystème ROS1, pour que les algorithmes avancés de traitement de données puissent contrôler/guider le robot et analyser les informations agricoles fournies par ces mêmes capteurs en autonomie.

1.2 Problématisation du thème, des objectifs et des enjeux

Dans le cadre du projet Antonio et ATLAS, dont je le rappelle, l'enjeu est de développer tout un écosystème interopérable et multi-entreprise, de capture, d'échange et de traitement d'information, il y a notamment la problématique de l'acquisition d'information. Or cette acquisition d'information, si elle est faite manuellement, est extrêmement chronophage. L'intérêt est donc d'automatiser cette tâche. Dans cette partie de l'Italie, les Pouilles, les vigneraies sont nombreuses et la qualité du vin, ainsi que les pertes (qu'elles soient dues à une mauvaise hydratation ou à la maladie) peuvent être contrôlées dans une certaine mesure si l'on suit l'évolution de l'état des vignes au jour le jour. Mais ce contrôle ne peut être quotidien que s'il est automatisé : l'utilité d'un robot qui se déplace et capture l'information de manière autonome pour prévenir le vigneron en cas de problème apparaît donc comme une solution idéale. C'est cette brique technologique qui est donc en train d'être étudiée et développée au sein du laboratoire, en partenariat avec l'entreprise Robodyne qui a conçu le robot utilisé. En tant qu'étudiant roboticien, ma candidature a été vite acceptée puisque ma formation correspond exactement aux besoins du laboratoire. L'enjeu est donc important pour le futur du milieu agricole italien, qui souhaite se moderniser et améliorer ses rendements, tout en facilitant le travail de l'agriculteur.

2 Présentation du robot

2.1 Architecture technique

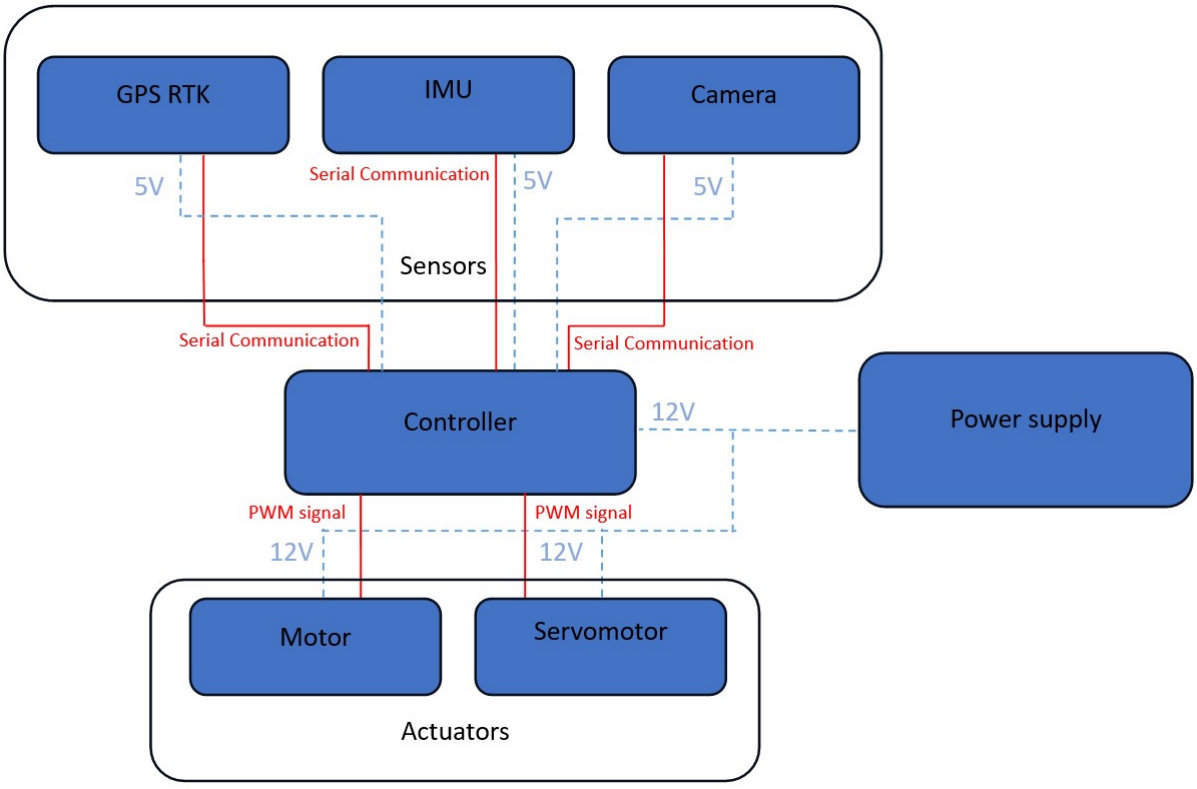

FIGURE 2 – Architecture électronique du robot

FIGURE 3 – Le robot en expérimentation

Dimensions	878x694x550 mm | 34.56x37.95x21.65 in.
Maximum Payload[1]	20 Kg | 44 lbs
All terrains Payload	10 Kg | 22 lbs
Speed	0 - 4+m/s | 0 - 2,23+ mph lbs
Ground clearance	29 cm | 11.41 in
Climb grade[2]	40°
Traversal grade[2]	25°
Operating time	4 Hours typical 48 hours stand-by
Battery pack	24VDC@100Ah
Battery charger	Short-circuit, over-current, over-voltage
User power	+24 VDC@5A regulated output +12 VDC@5A regulated output
Communication	RS232 up to 115200 Baud, Wi-Fi, Bluetooth, LAN
Internal sensing	1024 ppr Encoders, Voltage, current and temperature sensors

FIGURE 4 – Caractéristiques techniques du robot

2.2 Description des éléments

— Le contrôleur de bord : Le contrôleur est un NUC (CPU Intel, 8gb de ram, hdd de 128gb) sous Ubuntu 18.04, connecté en wifi pour le contrôle à distance (SSH), sur lequel tous les algorithmes tournent, ainsi que les nodes ROS Melodic.

FIGURE 5 – L'UC : NUC Intel

— Les capteurs : Trois caméras Intel Realsense D435, un IMU, des capteurs de couple dans les roues, le GPS (Ublox Zed-F9P)

FIGURE 6 – Le système de captation d'image : Intel D435

FIGURE 7 – Ublox ZED-F9P

— Les actionneurs : Chaque roue est équipée d'un moteur et d'un servomoteur
— La batterie : Deux batteries 12VDC de 100Ah

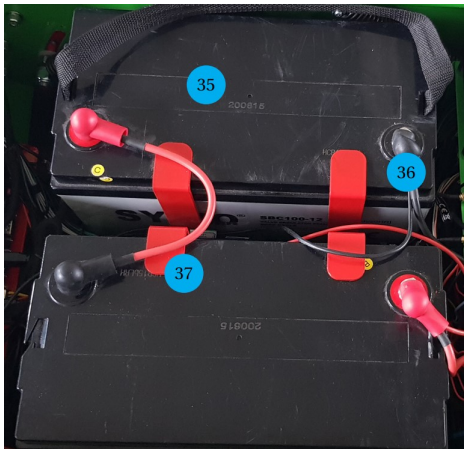

FIGURE 8 – Batteries

3 Mise en place du RTK/GPS

Pour positionner précisément le robot dans l'espace (dans notre cas, dans un champ de vigne), nous nous servons de deux modules GNSS vendus par U-blox, le ZED-F9P.

Tout seul, la précision atteinte est de l'ordre du mètre dans des conditions idéales : ciel dégagé, pas de bâtiments proches, antenne en hauteur et loin de toute surface métallique. Mais

cette précision n'est pas suffisante dans un champ de vigne, où les rangées sont larges d'un à deux mètres tout au plus. Pour atteindre une meilleure précision, de l'ordre du centimètre, nous nous servons donc du deuxième module GNSS pour utiliser la technique du RTK.

3.1 Le fonctionnement et l'intérêt du RTK

Pour se localiser dans l'espace, nous avons décidé d'utiliser un système de positionnement par satellite, que nous connaissons sous le nom de GPS (qui est une antonomase, puisqu'il s'agit du système de positionnement par satellite américain. Il existe également d'autres systèmes qui ont le même principe de fonctionnement, comme Galileo -européen-, GLONASS -russe-, ou encore BEIDOU -choinois-) . Ces systèmes de positionnement par satellite fonctionnent en utilisant des signaux radio émis par une constellation de satellites en orbite autour de la Terre. Ces signaux, contenant leurs dates d'émission, sont reçus par un récepteur GPS sur la Terre, qui mesure le temps qu'il faut pour que le signal atteigne le récepteur. En utilisant la vitesse de la lumière, le récepteur peut calculer la distance entre lui-même et le satellite. La position des satellites visibles par le récepteur est connue puisque le signal émis contient les éphémérides, permettant le calcul de la position des satellites. On peut par la suite déduire la position du récepteur en 3D par trilatération.

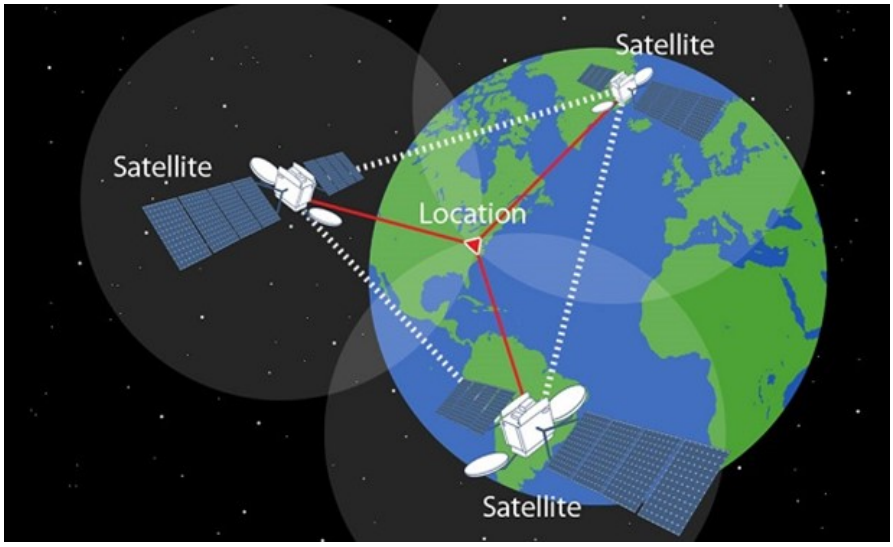

FIGURE 9 – Fonctionnement du GPS

Cependant les signaux GPS peuvent subir des délais et des erreurs de mesure en raison de divers facteurs tels que les réflexions des signaux contre les obstacles, l'ionosphère et la troposphère de l'atmosphère terrestre, ainsi que des erreurs de l'horloge du satellite, ce qui engendre des erreurs de position.

Le système RTK (Real Time Kinematic) est une technique utilisée pour réduire de manière significative l'impact de ces erreurs de mesure. Ainsi, le RTK améliore la précision en utilisant une station de base avec une position connue avec grande précision. La station de base utilise un récepteur GPS pour mesurer les signaux des satellites et calcule les corrections nécessaires pour corriger les erreurs de mesure. Ces corrections, sous la forme de signaux RTCM, sont transmises en temps réel, au récepteur GPS du terrain (ou rover) via un canal de communication. Cela peut être fait par radio, Internet ou d'autres moyens de communication. En utilisant ces corrections, le rover peut compenser les erreurs de mesure et obtenir une position beaucoup plus précise que ce qui serait possible avec le GPS seul.

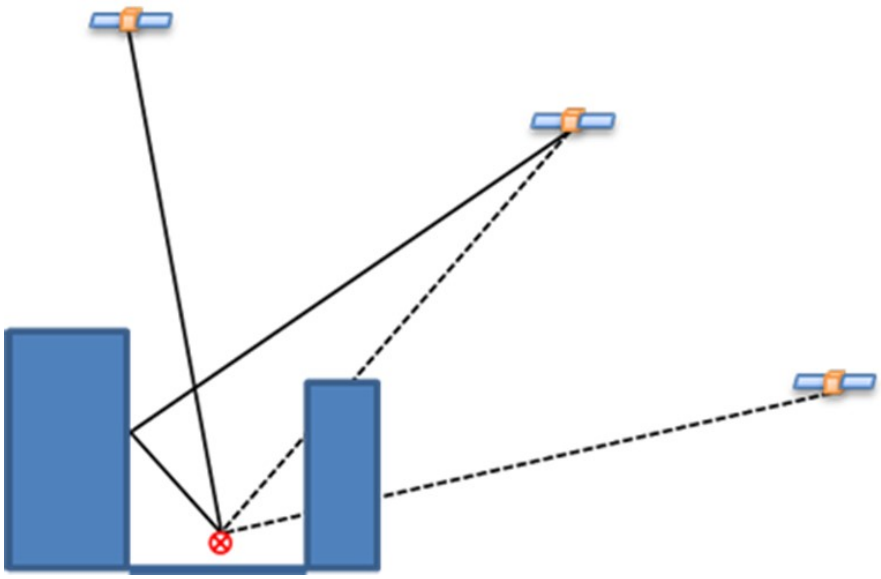

FIGURE 10 – Source d'erreur du GPS

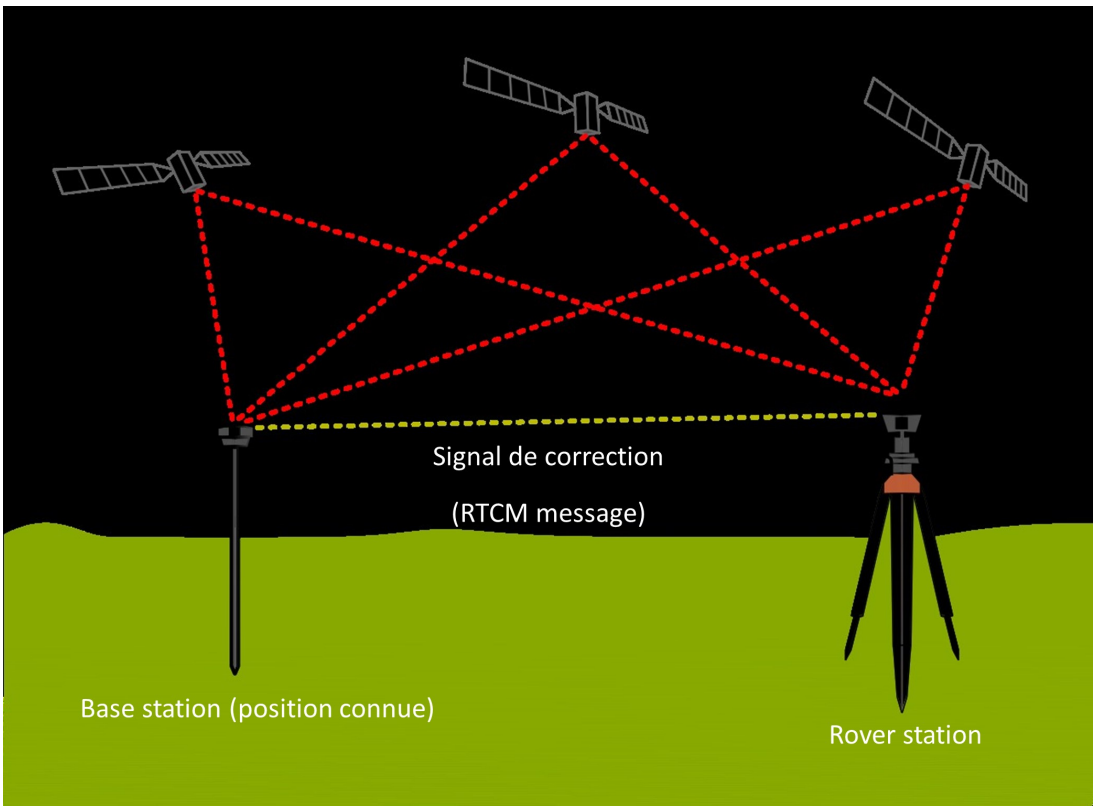

FIGURE 11 – Fonctionnement du RTK

3.2 Choix technique et architecture

Dans le cadre de nos expérimentations, nous avons fait plusieurs choix. Le premier fut d'utiliser deux cartes GPS afin d'avoir une position précise au centimètre. D'autres solutions existent, comme utiliser le SLAM ou l'odométrie visuelle, mais dans un champ, utiliser le RTK est la méthode la plus fiable pour acquérir sa position précisément. Nous avons également décider d'échanger les signaux RTCM par lien radio, car la portée de nos modems radio, de

l'ordre du kilomètre , est bien supérieure à celle du modem WIFI (ou Bluetooth) inclus sur la board C099-F9P (que nous avons utilisé au tout début de nos expérimentations pour tester le fonctionnement du dispositif et le bon échange des signaux de correction RTCM)

FIGURE 12 – Modem radio Satelline

Ainsi, l'un des modules GNSS est installé sur le robot, le deuxième est fixé au bord du champ où l'expérimentation a lieu.

3.3 Utilisation du dispositif

ROS

Le package ROS utilisé est « ublox_gps », qui permet de lancer un certain nombre de topic ROS, dont celui qui nous intéresse, le topic /ublox/fix, qui diffuse (entre autres) la position du robot.

S-center

La configuration se fait à l'aide d'un software dédié, fourni par le constructeur du GPS : S-Center. Ce software nous a permis de configurer les deux boards, d'abord pour pouvoir les connecter en WIFI mais également pour mettre à jour le firmware (si la board est brické par exemple), permettre la connexion ainsi que l'utilisation de U-center et continuer la configuration (voir Figure 22 en annexe)

U-center

Base Board Avant chaque expérimentation, il faut effectuer le « survey-in », c'est-à-dire obtenir la position la plus précise possible de la base station. Pour cela, on place la base station dans un endroit dégagé, en hauteur, et on la fixe. Puis, on lance l'acquisition de données GPS, jusqu'à ce que le software U-center détermine que la précision de la position est meilleure que le seuil donné au début du survey-in. En général, une déviation standard de 1m sur la position de la base station est suffisante. Une fois que le survey-in est terminé, la base station partage les signaux RTCM selon la voie choisie lors de la configuration : dans notre cas, sur la sortie UART2, sur laquelle est connecté notre émetteur radio.

Rover Pour la board en mode « Rover », il suffit de l'allumer et de la brancher au modem radio. Automatiquement, elle va recevoir les signaux de corrections RTCM, les signaux GPS et corriger sa position.

Résultats

Sur le terrain, nous avons réussi à obtenir une précision de l'ordre du centimètre grâce à ce dispositif.

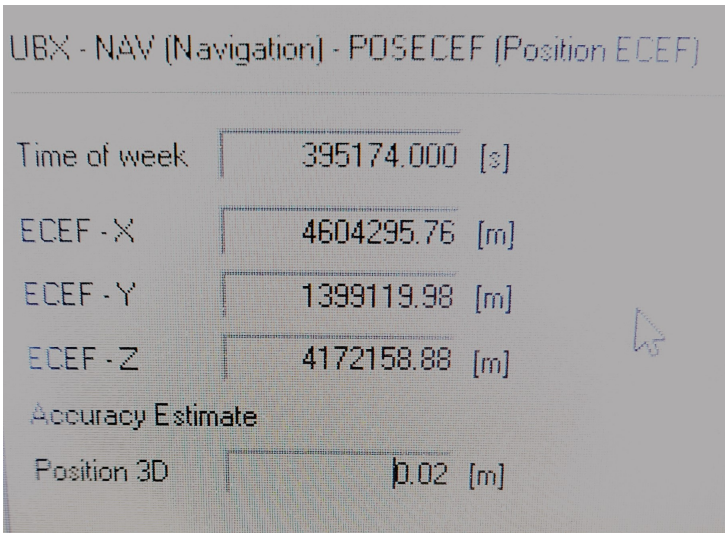

FIGURE 13 – Précision mesurée du dispositif de localisation

4 Amélioration de l'algorithme de contrôle à la manette du robot

Le contrôle manuel du robot se fait à travers une manette (type XBOX) connectée en Bluetooth au contrôleur du robot. Un driver ROS est utilisé ensuite pour transmettre les informations sur le node /joy.

4.1 Le fonctionne initial

Le robot étant doté de quatre roues indépendamment contrôlées, l'algorithme de contrôle est légèrement différent de celui d'une voiture radiocommandée par exemple. Initialement, il était doté de quatre modes distincts :

— Un mode 2wd (mode A) qui simulait le contrôle d'une voiture avec des roues avant liées et des roues arrières fixes
— Un mode 4wd (mode B) où les roues avant et arrières étaient liées : lorsque l'on demande de tourner à droite, les deux roues avant vont tourner d'un angle α et les deux roues arrière vont tourner d'un angle -α
— Un mode de rotation sur lui-même (mode C)
— Un mode de déplacement latéral (mode D)

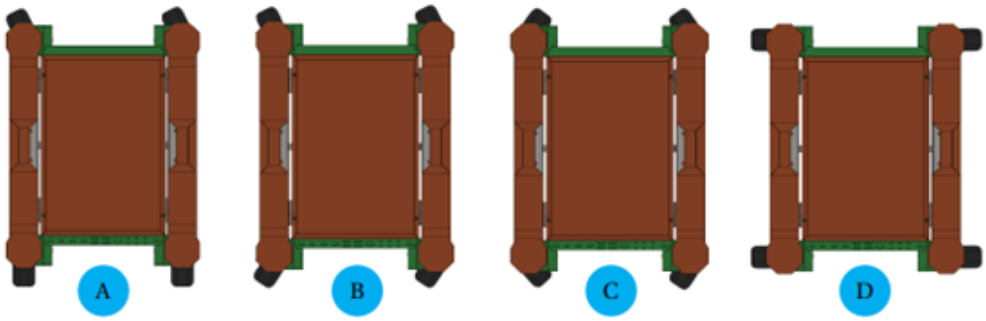

FIGURE 14 – Configurations de directions

Ces quatre modes sont limités et n'exploitent pas toutes les capacités du robot. De plus, deux programmes étaient utilisés : l'un était écrit en python et servait à contrôler la vitesse de rotation des roues, tandis que le second était écrit en C++ et contrôlait l'orientation des roues. Ma tâche a donc été de rendre le robot manœuvrable à 360° et de regrouper le contrôle en un seul programme. J'ai choisi de tout réécrire en python puisque ce langage est mieux maîtrisé par mes collègues, ce qui leur permettra à l'avenir de réutiliser et/ou modifier mon code.

4.2 Mise à jour des algorithmes

Ainsi, le contrôle du robot se fait désormais, à la demande de mes collègues, dans quatre modes distincts :
— Un mode 2wd similaire mais avec un ajustement de l'angle de rotation de chaque roue, de sorte que les roues pivotent autour d'un même point et éviter un dérapement de la roue extérieure lors de rotations fortes
— Un mode Side-steering qui permet au robot de se déplacer dans toutes les directions mais sans changement d'orientation
— Un mode 4wd modifié comme le mode 2wd, pour les mêmes raisons
— Un mode Circle-steering, qui permet au robot de tourner sur lui-même, en tournant chaque roue d'un angle de 45°, de sorte que les axes de rotation des roues se croisent au milieu du robot

Pour améliorer les capacités de rotation du robot, j'ai implémenté artificiellement le système de géométrie directionnelle d'Ackermann. Ainsi, peu importe l'angle de rotation désiré, le robot tourne autour d'un point (défini de sorte que l'angle de rotation maximal d'une roue soit de 90°).

Cela permet à chacun en fonction de ses compétences en pilotage et des besoins de choisir le mode de conduite le plus adapté, tout en limitant le risque d'erreurs.

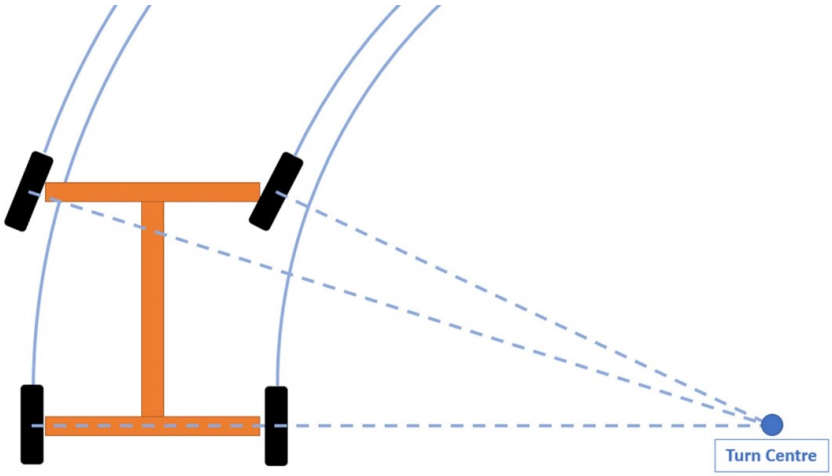

FIGURE 15 – Rotation différentielle Ackermann

$$\delta_{f,in} = \tan^{-1}\left(\frac{L}{R - \frac{T}{2}}\right) \quad \delta_{f,out} = \tan^{-1}\left(\frac{L}{R + \frac{T}{2}}\right)$$

FIGURE 16 – Angle de rotation des roues selon la géometrie d'Ackermann

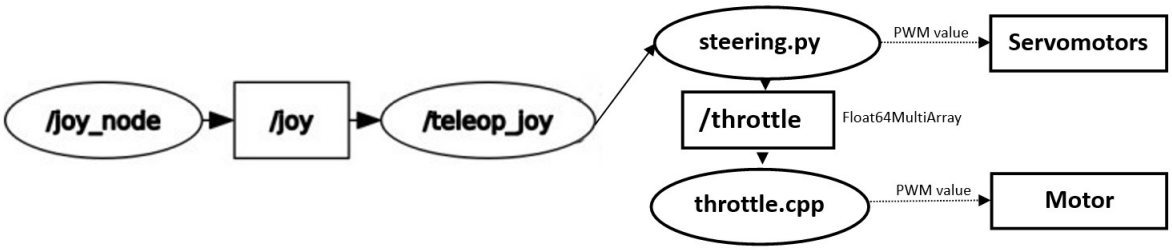

FIGURE 17 – Architecture logicielle de l'algorithme de contrôle du robot

Il est néanmoins à noter que les 2 programmes sont utilisés : si la gestion des inputs de la manette est centralisée dans le code python, j'ai dû utiliser le programme en C++ pour l'envoi du signal PWM au moteur. En effet, le drivers des moteurs n'étaient ni commentés ni explicites, avec un grand nombre de fonctions aux noms similaires. J'ai donc préféré réutiliser les fonctions qui existaient et qui fonctionnaient plutôt que de perdre trop de temps à analyser un code complexe.

5 Tentative d'implémentation d'un algorithme de SLAM (variante du ORB-SLAMV3 -> Stella-Slam)

Pour améliorer les performances de navigation autonome du robot, ainsi que ses capacités à localiser les informations capturées et les rendre plus aisément lisibles et compréhensibles,

j'ai tenté d'implémenter un algorithme de SLAM, l' openVSLAM (qui est très similaire à ORB-SLAMv3)

5.1 Principe du SLAM

SLAM, qui signifie Simultaneous Localization and Mapping, est un problème fondamental dans le domaine de la robotique et de la vision par ordinateur. Il fait référence au processus par lequel un robot ou un système autonome navigue et construit une carte de son environnement en temps réel tout en déterminant simultanément sa propre position dans cet environnement.

Voici une explication des composantes clés de SLAM :

- **Localisation :** La localisation consiste à estimer la position précise et l'orientation du robot dans son environnement. Cela est réalisé en utilisant divers capteurs tels que l'odométrie (qui mesure les données d'encodeur de mouvement ou de roue), le GPS, l'IMU (Unité de Mesure Inertielle), le LIDAR (Détection et Télémétrie par Laser), ou des caméras.
- **Cartographie :** La cartographie implique la création d'une représentation de l'environnement que le robot explore. La représentation peut prendre diverses formes, telles qu'une grille 2D ou 3D, un nuage de points, ou une carte basée sur des caractéristiques. Cette carte est construite de manière incrémentale à mesure que le robot se déplace dans son environnement.
- **Simultanéité :** "Simultanéité" dans SLAM fait référence au fait que le processus de localisation et de cartographie se déroule simultanément. En d'autres termes, le robot met à jour sa position et construit la carte en même temps.
- **Temps réel :** SLAM est conçu pour fonctionner en temps réel, ce qui signifie qu'il doit traiter les données des capteurs et mettre à jour la carte et les estimations de localisation suffisamment rapidement pour que le robot prenne des décisions en fonction de sa compréhension actuelle de l'environnement.
- **Fermeture de Boucle :** L'un des aspects difficiles de SLAM est de tenir compte du fait qu'un robot peut revisiter ou faire une boucle vers des zones qu'il a déjà cartographiées. Détecter et gérer ces fermetures de boucle est crucial pour maintenir une carte cohérente et précise.
- **Association de Données :** L'association de données implique de faire correspondre les mesures des capteurs à des caractéristiques ou repères déjà cartographiés. Cela est important pour mettre à jour la carte et affiner l'estimation de localisation.
- **Incertitude et Optimisation :** Les algorithmes SLAM utilisent généralement des techniques probabilistes pour modéliser l'incertitude à la fois sur la position du robot et sur la carte. Cela implique des techniques telles que les filtres de Kalman, les filtres à particules, ou des méthodes d'optimisation plus avancées comme le SLAM basé sur les graphes.

Les algorithmes de SLAM ont une large gamme d'applications, pour les véhicules autonomes, les drones, l'exploration robotique, et même la réalité augmentée. Il joue un rôle critique en permettant aux robots de naviguer et d'interagir avec leur environnement de manière autonome.

5.2 StellaVSLAM

L'algorithme StellaVSLAM (qui est dérivé de l'algorithme openVSLAM, destiné uniquement à des fins de recherche), est un algorithme SLAM indirect avec des caractéristiques éparses.

Cela signifie qu'il utilise des points caractéristiques (ou points d'intérêt ou points clés) pour reconstruire le monde et minimiser les erreurs. En revanche, une approche directe utilise la luminosité globale des images. Direct vs indirect, épars vs dense.

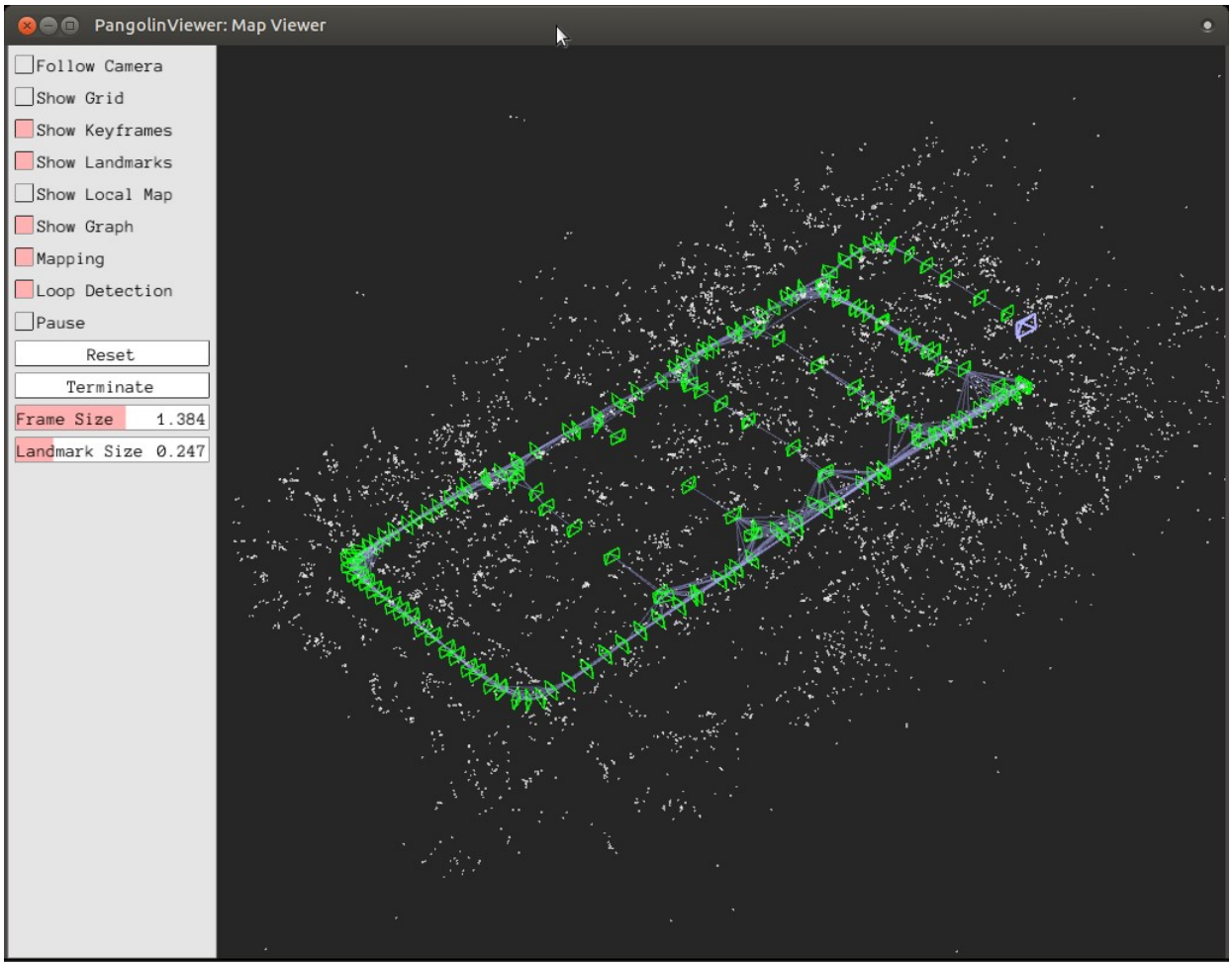

FIGURE 18 – Exemple de résultat obtenu par OpenVSLAM

5.3 Résultats

Selon la documentation, StellaVSLAM est censé fonctionner avec ROS, mais je n'avais pas accès à un fichier ROS bag ni au robot, j'ai donc dû travailler avec ce que j'avais, un fichier Realsense bag. Cependant, le fichier Realsense bag est dans un format différent et contient des informations de profondeur. Par conséquent, j'ai dû convertir le fichier Realsense bag en deux séquences d'images, l'une contenant les informations de couleur, et l'autre contenant toutes les informations de profondeur. Cependant, pour garder la taille des dossiers raisonnable, l'outil créé par l'équipe Realsense utilise un algorithme pour convertir les informations de profondeur en une image RVB, ce qui signifie que la profondeur est codée par une couleur.

$$d_{\text{normal}} = \frac{d - d_{\min}}{d_{\max} - d_{\min}} 1529$$

$$p_r = \begin{cases} 255\,(0 \leq d_{\text{normal}} \leq 255 \cup 1275 < d_{\text{normal}} \leq 1529) \\ 255 - d_{\text{normal}}\ (255 < d_{\text{normal}} \leq 510) \\ 0\,(510 < d_{\text{normal}} \leq 1020) \\ d_{\text{normal}} - 1020 \\ (1020 < d_{\text{normal}} \leq 1275) \end{cases}$$

$$p_g = \begin{cases} d_{\text{normal}}\ (0 < d_{\text{normal}} \leq 255) \\ 255\,(255 < d_{\text{normal}} \leq 510) \\ 765 - d_{\text{normal}}\ (510 < d_{\text{normal}} \leq 765) \\ 0\,(765 < d_{\text{normal}} \leq 1529) \end{cases}$$

$$p_b = \begin{cases} d_{\text{normal}}\ (0 < d_{\text{normal}} \leq 765) \\ d_{\text{normal}} - 765\,(765 < d_{\text{normal}} \leq 1020) \\ 255\,(1020 < d_{\text{normal}} \leq 1275) \\ 1529 - d_{\text{normal}}\ (1275 < d_{\text{normal}} \leq 1529) \end{cases}$$

FIGURE 19 – Principe de fonctionnement de l'algorithme de compression des informations de profondeur d'Intel

$$d_{rnormal} = \begin{cases} p_{rg} - p_{rb} (p_{rr} \geq p_{rg} \cap p_{rr} \geq p_{rb} \cap p_{rg} \geq p_{rb}) \\ p_{rg} - p_{rb} + 1529 (p_{rr} \geq p_{rg} \cap p_{rr} \geq p_{rb} \cap p_{rg} < p_{rb}) \\ p_{rb} - p_{rr} + 510 (p_{rg} \geq p_{rr} \cap p_{rg} \geq p_{rb}) \\ p_{rr} - p_{rg} + 1020 (p_{rb} \geq p_{rg} \cap p_{rb} \geq p_{rr}) \end{cases}$$

$$d_{recovery} = d_{min} + \frac{(d_{max} - d_{min}) d_{rnormal}}{1529}$$

FIGURE 20 – Principe de fonctionnement de l'algorithme de récupération des données de profondeur à partir d'une image RGB

Mais StellaVSLAM nécessite des informations de profondeur brutes : chaque pixel est une nuance de gris, avec sa valeur directement proportionnelle aux informations de profondeur. J'ai essayé de reconvertir dans le bon format, mais je n'y suis jamais parvenu, malgré l'aide de ma tutrice. J'ai donc utilisé uniquement les images, sans les informations de profondeur. De cette façon, j'avais moins d'informations, et donc l'algorithme était moins performant : cela fonctionnait bien en ligne droite, mais dès que le robot tournait, il perdait toutes les informations de suivi, et peu importe comment je modifiais les paramètres, cela ne fonctionnait pas.

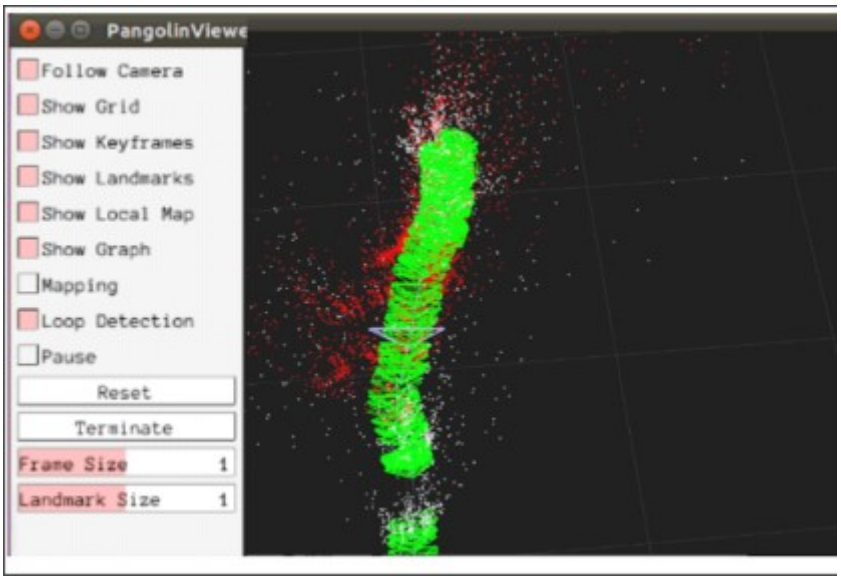

FIGURE 21 – Résultat de l'algorithme StellaVSLAM sur le Realsense bag

J'aurais aimé implémenter l'algorithme directement sur le robot avec ROS et voir ce que cela donnait avec les bonnes informations de profondeur, car en l'état, mon implémentation du SLAM n'est pas fonctionnelle, bien qu'encourageante.

6 Conclusion

Conclusion technique

Ayant accompli tout ce qui m'était demandé, en mettant en place tout le système de localisation et en améliorant les algorithmes de contrôle du robot, et ayant essayé d'aller plus loin en implémentant un algorithme SLAM pour améliorer les capacités de localisation du robot, je regrette de ne pas avoir pu aller plus loin, en rendant le robot capable de naviguer en autonomie dans les vigneraies. Cependant, j'ai beaucoup appris sur la manière de rechercher un sujet, de travailler dans le monde de la recherche, sur la manière de suivre les progrès et comment rédiger un rapport de stage.

Acquis personnels

Ce stage m'a permis de me perfectionner en robotique, notamment sur l'utilisation de ROS dans le cadre de projet de recherche universitaire. Je suis désormais beaucoup plus confiant dans son utilisation. J'ai également participé à des campagnes d'essais. Mais le milieu étant universitaire, ce stage s'inscrivait parfaitement dans la continuité des cours suivis à l'ENSTA, je n'ai donc pas eu de surprise sur ce point. C'est toutefois plutôt positif puisque l'ambiance y était très similaire, l'entraide, la bonne humeur et la recherche de l'excellence étaient de mise. Le contact avec des personnes d'une autre langue, d'une autre culture fut très intéressant et je me suis bien intégré, malgré la différence d'âge (j'étais le benjamin à 23 ans, le deuxième plus jeune avait 26 ans). Je considère donc que mes compétences interpersonnelles se sont améliorées grâce à ce stage, qui m'a rendu plus ouvert d'esprit et plus à même de m'intégrer dans une équipe internationale, multi-culturelle.

Apport au projet professionnel

En tant qu'IETA, je suis très heureux d'avoir pu découvrir la recherche en robotique puisque les postes à la DGA qui permettent d'être au plus proche de la robotique et de l'innovation sont rares. Par la suite et dès que possible si je parviens pas à trouver de poste en ce genre, je prévois de me réorienter dans le privé pour pouvoir travailler en robotique dans l'industrie, et si possible, pouvoir participer à un projet innovant. C'est pourquoi, afin de valider ce projet initial, je compte réaliser un PFE dans une entreprise privée pour m'assurer que le travail dans ce cadre est tel que je l'imagine.

Annexe

```
References :
https://www.u-blox.com/en/product/c099-f9p-application-board user guide

Prerequisites :
1/teraterm
2/u-center
3/s-center
4/the ZEDF9P should be ver1.32


1 - Set-up of the odin boards (do this on both boards):
a/ Download stm32flash here :   https://sourceforge.net/projects/stm32flash/
b/ In the same folder as the stm32flash exe, download
   https://raw.githubusercontent.com/u-blox/ublox-C099_F9P-mbed-3/master/c099mbed3_v2.0.0.bin
(The following (c-d-e-f) has to be done on both boards)
c/ in cmd go to the folder with stm32flash
d/ jumperwire on the odin safeboot pins (pin9)
e/ run ".\stm32flash.exe -b 115200 -w c099mbed3_v2.0.0.bin -S 0x8000000 COM<port number>"
f/ remove jumper-wire and restart using the left white button
g/ open S-center on the com port of the odin board with baudrate 115200 and hardware control flow enabled
h/ click on EVK-ODIN2-W2 via ST-link
i/ Navigate to “User Defines” AT command tab
j/ Execute the following command set sequentially:
AT+UMRS=460800,2,8,1,1,0
AT&W
AT+CPWROFF

4 - setup of the base in AP mode
b/ restart the base board
c/ open a teraterm with baudrate=460800 and the COM port corresponding to       the base
d/ run the following commands "/mem_store/run wifi_ap" and then         "/mem_store/run base"
e/ restart the board

4 - setup of the rover in STA mode
b/ restart the rover board
c/ open a teraterm with baudrate=460800 and the COM port corresponding to       the rover
d/ run the following commands "/mem_store/run wifi_sta" and then        "/mem_store/run rover"
e/ restart the board

5/both boards should connect automatically whenever they are started

6 - Setup of the ZED-F9P (both the rover and the base)
a/ Open u-center with either the uart port corresponding to the ZEDF9P(baudrate should be 460800)
   or the usb serial(baudrate doesnt matter)
b/ click on the "create a config view"
c/ go to MSG
d/ for RTCM3.3 1005, 1074, 1084, 1094, 1124, 1230, set every parameters to On and don't forget to
   press the send button in the bottom left to apply the changes
e/ then go to PRT (ports) and for I2C in and out is RTCM3, for UART1 it's RTCM3-UBX+NMEA+RTCM3
   baudrate=460800, for UART2 it's 5-5 baudrate=38400
   (uart2 is not used for wifi connection, the data is sent via I2C), for USB and SPI it's the same
   as UART1
f/ go to CFG and click send (saves the modification in the memory of the board so you dont have
   to do that at every start up)

7 - Survey-in on the base to get RTCM correction messages
a/ Open u-center with either the uart port corresponding to the ZEDF9P(baudrate should be 460800)
   or the usb serial(baudrate doesnt matter)
b/ Create a message view
c/ Go to UBX then CFG then TMODE3
d/ Mode 1-Survey-in with parameters 60s observation time and 5m required pos accuracy,
   then click send, and wait
e/ To monitor the survey-in process, go to UBX-NAV-SVIN (right click on SVIN to enable messages)
   and wait until it says "Successfully finished"
f/ Click on the "Create a packet console to verify that RTCM correction message are sent.
   Additionally, the fix mode should change to TIME/DGNSS

8 - Rover board
a/ Open u-center with either the uart port corresponding to the ZEDF9P(baudrate should be 460800)
   or the usb serial(baudrate doesnt matter)
b/ create a config view
c/ Go to UBX then RXM then RTCM and enable messages by right clicking : you should see a window
   showing the message type and the number of messages corresponding and other infos
```

Figure 22

```
Modification of the structure of the project :
- There were 2 files to control the robot : the robo_explorer.cpp (throttle) and the steering.py
  (steering)
- Now everything is done from steering.py, the throttle instructions are sent to the robo_explorer.cpp
  over ROS on the topic throttle, and it utilizes the Float64MultiArray msg type, and the steering
  instructions are done directly from the .py

'State' of the robot :
The robot has 6 state : "4wd", "2wd", "lateral", "louis" (test mode), "circle" and "stop"
Depending on the state of the robot, the xbox controls the robot differently (see after)
To modify the state of the robot, press A,B,X,Y or Back. If you press A, you go in 4wd mode.
B is for stopping, X is for lateral mode, Y is for circle mode and Back is for louis mode.
If you are already in the mode you press the button of, you go in 2wd mode.

Throttle control theory:
For all of these modes, the throttle can go from 0 (stop) to 1000 (full speed) but is limited to 350
to not damage anything
- in stop mode : throttle set to 0
- in 4wd mode : throttle controlled by the left joystick vertical axis(up==forward, down==backward),
  and modified by the no_slippage function to ensure that the wheels dont go too fast or too slow
  (on a turn, the outside wheels have more distance to travel and therefore must go faster)
- in 2wd mode : same as 4wd concerning the throttle
- in circle mode : throttle controlled by the right joystick, horizontal axis
- lateral mode : throttle controlled by the left joystick, vertical axix
- in louis mode : idk

Steering control theory :
For all of these modes, we first calculate the wanted angles, and then convert these in values understood
by the motors drivers using the function 'angle2pid' wich convert the angle to a value between 'valore_sx'
which is the int value of the wheel turned 90deg to the right (ik, inversed), and 'valore_dx'
(int value of the wheel at 90deg to the left), and valore_centrale is the value of the wheel pointing
straight forward, at 0deg

- stop mode : steering set to 0
- 4wd mode : steering controlled by left joy, horizontal axis, but the steering angle sent to the
  outside wheels is modified by the no_slippage fct so that the 4 wheels have the same center of rotation
  (all lines perpendicular to the wheels cross at the same point)
- 2wd_mode : same as 4wd but only the 2 front wheels can rotate (back wheels angle set to 0)
- circle mode : angles of the wheels set to 45 (or -45 depending on the wheel), no control of the steering
  possible herer
- lateral mode : the angle of every wheel is the same and controlled by the right joystick.
  The angle of the wheels is determined by the angle the joystick has with the vertical axis
- louis mode : idk


Functions help :
- helping functions
        angle2pid2 : former function to convert from wanted angle to int value
        angle2pid : new one more understandable (to me at least)
        clamp
        remap
        read_enc
        stop_robot
        sign
        nonzero
        angle

- 'big' functions
        joy_listener
        f
        no_slippage :
                the xbox controller decides the steering angle of the inside front wheel and we then modify
                the speed and steering angle of the other wheels, by calculating the intersection point of
                the lines perpendicular to the inside wheels, and then calculating the angles of the other wheels
                so that their perpendicular cross at this same point. We can then also calculate the distance
                between each wheel and this center/intersection point,
                and calculate a speed modifyier for each wheel by dividing this distance by the biggest distance
        what_angle
        set_steering_angle
        what_speed
        set_speed
        set_state
        tj_callback_final
```

FIGURE 23

Références