ENSTA Bretagne

UE 4.4
Rapport

Rapport de Projet Industriel

Élèves :
Ludovic DIGUET
Antonin LIZÉ
Maha HALIMI
Badr MOUTALIB

Commanditaire :
ORANGE Labs
Jacques CHODOROWSKI

9 mai 2020

Table des matières

1 Introduction

1.1 Présentation générale

Notre projet a pour objectif de construire un robot patrouilleur pour explorer une ferme de routeurs, ceci afin de prendre des vidéos et photos des serveurs et ainsi permettre à distance un premier diagnostic des problèmes rencontrés dans le centre de serveurs. Les fonctions principales de ce robot ont été définies ci-dessous :

- Être capable de patrouiller dans une zone présentant des obstacles (murs, câbles au sol...).
- Pouvoir prendre des vidéos et des photos exploitables.
- Être capable de retourner de manière autonome à sa station de recharge.
- Communiquer les informations acquises.
- Être intégrable sous ROS.

Pour bien comprendre tous les enjeux de ce projet nous avons eu besoin d'effectuer plusieurs conférences téléphoniques avec Mr CHODOROWSKI qui ont donné lieu à plusieurs rencontres à l'ENSTA afin d'aborder de façon plus détaillée certains points. Ces rencontres nous ont permis de bien définir le projet dans sa globalité et ainsi pouvoir réaliser un cahier des charges fonctionnel que ce soit pour le robot ou pour la station de recharge.

Vous pouvez également retrouver notre projet sur GitHub. Tous les codes Arduino et C++ se trouvent dessus, les liens qui renvoient vers notre code dans la suite du projet pointent tous vers notre GitHub.

A noter que tous les mots colorés sont des liens à suivre qui permettent de se diriger vers un site internet ou vers une autre partie du rapport.

1.2 Abstract

Ce rapport traite du projet en collaboration avec Orange Labs pour la création d'un robot terrestre autonome. Le produit abouti visé est un véhicule capable de naviguer dans des salles de serveurs de manière autonome afin d'y identifier une éventuelle panne. Ce rapport traite du cheminement suivi. Après avoir dressé le cahier des charges du robot en collaboration avec notre encadrant, nous avons fixé deux objectifs majeurs pour ce projet : la construction d'un drone terrestre fonctionnel et la construction de la station de recharge pour ce drone.

Certaines contraintes nous ont poussés à redéfinir nos objectifs : nous sommes arrivés à deux résultats majeurs. Premièrement une description théorique de la manière de construire le robot et la station de recharge. Deuxièmement l'ensemble des composants sont intégrés et prêts à être utilisés sous ROS. Ces résultats sont satisfaisants et permettront d'assurer la continuité du projet.

Ce projet est une opportunité pour nous d'appliquer les connaissances acquises dans de nombreux domaines mais aussi de travailler avec des outils nouveaux pour nous, comme notamment l'utilisation d'un NUC et d'un LIDAR. De plus, la collaboration avec l'entreprise ORANGE LABS fut une occasion de réellement progresser en matière de gestion de projet et de définition d'objectifs.

1.3 Remerciements

La réalisation de ce projet a été possible grâce au concours de plusieurs personnes à qui nous voudrions témoigner toute notre gratitude.

Nous tenions à exprimer toute notre reconnaissance à notre professeur référant, Monsieur Fabrice LEBARS. Nous le remercions de nous avoir encadrés, orientés, aidés et conseillés.

Nous adressons nos sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé nos réflexions et ont accepté de nous rencontrer et de répondre à nos questions durant nos recherches.

Enfin nous voulions remercier tout particulièrement notre encadrant Mr Jacques CHODOROWSKI d'Orange Labs à Lannion pour son aide et sa supervision tout au long du projet. Il nous a permis de toujours garder en ligne de mire des objectifs clairs et il a su adapter les modalités du projet au vu du contexte actuel et des contraintes imposées par celui-ci. Ce fut un réel plaisir pour toute notre équipe de travailler en collaboration avec lui.

À tous ces intervenants, nous présentons nos remerciements, notre respect et notre gratitude.

2 Définition du problème global

2.1 Partie Station de Recharge

Dans cette sous partie, nous allons énumérer les contraintes qui nous permettront d'élaborer et de choisir des solutions pour la réalisation de la Station de recharge.

Voici donc le Cahier des Charges fonctionnel pour la partie Station de Recharge :

Fonction	Désignation	Critères	Niveau	Fléxibilité
FP1	Permettre la recharge du robot de façon autonome			F0
FC1	Permettre une recharge rapide	Temps de charge	2/3h	F1
FC2	Alimentation par le secteur			F0
FC3	Permettre une flexibilité dans l'angle d'approche du robot	Angle entre l'axe du robot et l'axe de la station de recharge	20°	F2
FC4	Renvoyer des informations sur l'état de charge			F0
FC5	Prendre peu de place			F3

FIGURE 1 – Cahier des Charges Fonctionnel - station de recharge

Ces Cahiers des Charges, ont étés élaborés à partir de nos rencontres avec Monsieur CHODOROWSKI et nous ont permis de définir les principaux axes de travail pour la suite. Un document qui nous a également servi pour définir les contraintes et les enjeux se trouve sur le GitHub du projet et se nomme "*OrangeLabs - Robot autonome évolutif*".

Les grands axes sont donnés dans la Figure ci-dessus, et ce sont ces axes qui nous ont dicté nos choix que ce soit pour les choix des composants, ou les choix de l'architecture et de la forme de la Station de Recharge. Nous reviendrons dans la partie dédiée à la conception de la Station de Recharge, sur les différentes contraintes clés.

Il est important de noter que ces cahiers des charges sont ceux de la version finale robot. Les objectifs de notre travail sont plus concentrés sur certaines parties que d'autres et ont évolué au cours du projet et face aux difficultés rencontrées, le détail de l'évolution de nos objectifs est abordé dans la sous-section suivante.

2.2 Partie robot

L'objectif général du projet est de construire un robot terrestre capable de naviguer en intérieur de manière totalement autonome. Il doit également être capable d'évoluer sur un terrain plus ou moins accidenté pour s'adapter à l'organisation de certains data center

où les obstacles sont nombreux. Avec notre encadrant, il a été défini que le robot devrait pouvoir évoluer dans des couloirs d'environ 60 cm de largeur. En plus de sa capacité à patrouiller de manière indépendante, le robot doit également être capable de se recharger sans intervention extérieure. La contrainte de vitesse de déplacement, fixée par l'encadrant du projet, est d'un mètre par seconde.

En plus de sa capacité à naviguer de manière totalement autonome, il faut également qu'un technicien puisse contrôler le robot si nécessaire. Il faudra donc prévoir une commande manuelle et un bouton d'arrêt d'urgence en cas de problème.

Venons en maintenant à la partie échange d'information. La première contrainte importante, exigée par notre encadrant, est l'utilisation du middleware ROS afin de faciliter l'intégration des composants et d'organiser le projet. Le robot doit pouvoir échanger des informations avec l'utilisateur, notamment par un retour vidéo qui permettrait au technicien d'identifier une éventuelle panne. Cette caméra serait dans l'idée, fixée sur un mat d'environ 1.80 m (sur lequel d'autres composants pourraient être ajoutés si nécessaire).

C'est à partir de l'ensemble de ces contraintes nous avons élaboré le cahier des charges ci-dessous.

Fonction	Désignation	Critères	Niveau	Flexibilité
FP1	Patrouiller en intérieur de manière autonome			F0
FC1	Fonctionner sous ROS			F0
FC2	Vitesse de déplacement	Vitesse	1 m/s	F1
FC3	Autonomie	Temps de fonctionnement	1 h 30	F1
FC4	Contrôlable à distance manuellement			F1
FC5	Posséder un mat pour fixer des composants	Taille du mat	1,80 m	F3
FC6	Franchir des obstacles et naviguer sur un terrain accidenté	Taille des obstacles	20 mm	F0
FC7	Posséder un bouton d'arrêt d'urgence			F2
FC8	Naviguer de manière précise	Précision de navigation	< 10 cm	F1
FC9	Posséder un outil de retour vidéo			F1

FIGURE 2 – Cahier des Charges Fonctionnel - Robot

2.3 Définition et évolution des objectifs au cours du projet

La réalisation du robot présenté ci-dessus en partant de zéro est très ambitieuse par rapport au temps alloué au projet : il a donc été convenu que l'objectif principal serait

la construction "hardware" du robot, la partie algorithme pouvant être fournie par notre encadrant en utilisant des projets similaires déjà développés. Le but de notre projet au long de l'année est donc la construction du robot pour arriver à un produit fonctionnel et contrôlable manuellement. Les enjeux se résument de la manière suivante :

- Construire un robot fonctionnel sur lequel pourrait être intégré un algorithme de navigation autonome et respectant les contraintes du cahier des charges.
- Construire une station de recharge pour ce robot.

Cet objectif était réalisable dans le temps imparti, cependant entre la perte de deux personnes travaillant sur le projet (pour cause de départ en substitution en janvier) et la difficulté majeure liée à la crise du COVID-19, nous avons dû redéfinir nos objectifs. Au moment du confinement il était impossible d'avoir accès à la totalité des composants du robot, donc, en plus de ce qui était déjà réalisé, nous avons décidé que l'objectif du projet serait, à défaut de pouvoir construire le robot, de réaliser un travail théorique le plus précis possible, détaillant le processus de construction et d'intégration, ainsi que les choix techniques. Ce travail permet de constituer une base de travail intéressante pour la suite du projet et permet d'en assurer la continuité.

3 Choix techniques pour le Robot

3.1 Architecture globale

Le cahier des charges nous permet d'identifier les différentes fonctions que notre robot doit être capable de réaliser, il nous faut intégrer :

- Un support pour le robot.
- Un moyen de propulsion approprié.
- Des capteurs pour l'acquisition des informations nécessaires.
- Des moyens d'alimentation adéquats.
- Des outils de contrôle et d'interface.

L'organisation de ces composants choisis est détallée dans le schéma ci-dessous. Le détail des choix pour chaque composant sera donné dans une partie ultérieure.

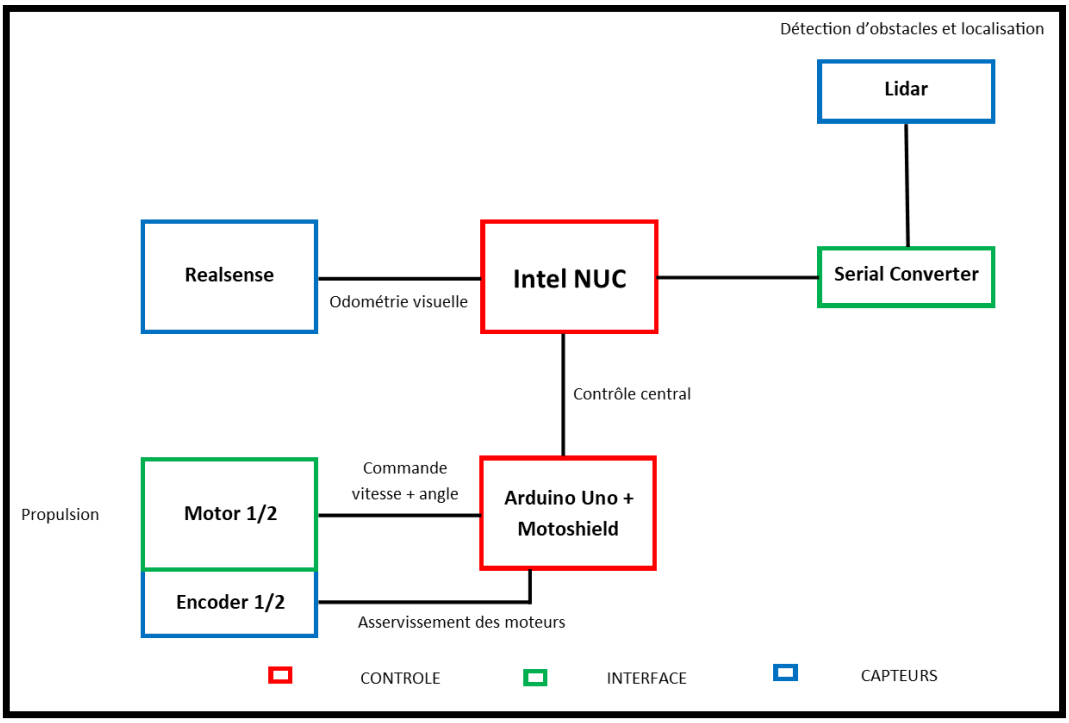


FIGURE 3 – Organisation des composants - Architecture générale

Il est également important de préciser le mode de communication choisi entre ces différents composants, voir ci-dessous.

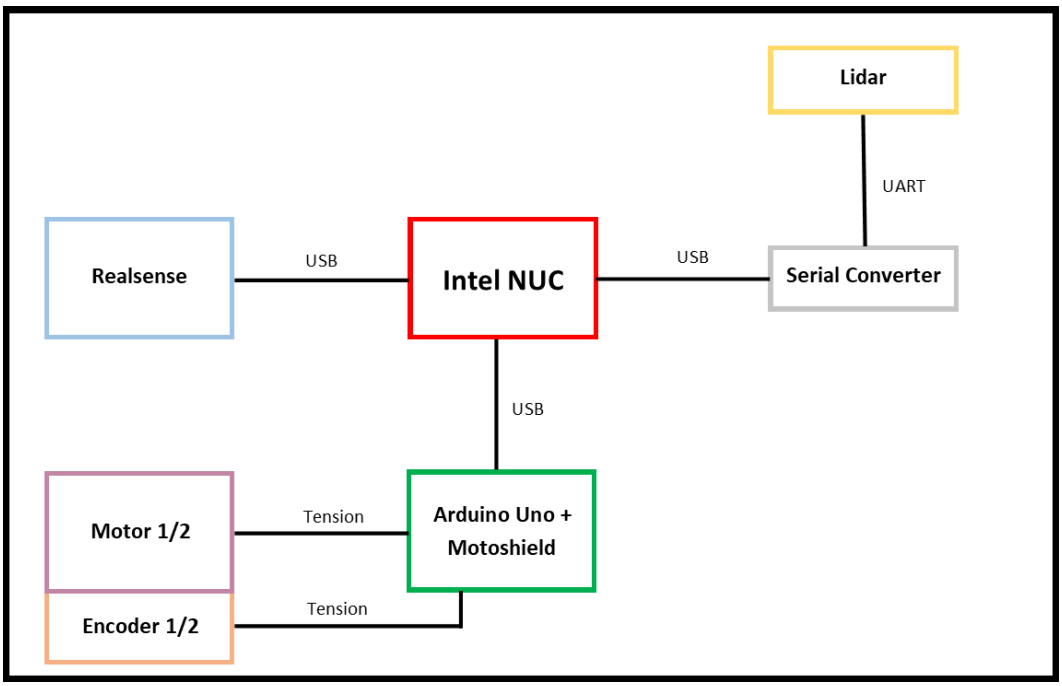


FIGURE 4 – Organisation des composants - Modes de communication

3.2 Partie Mécanique : le châssis

Très vite nous avons compris que le choix du châssis allait être primordial pour la suite du projet et surtout qu'il devait être effectué assez rapidement. En effet en fonction du châssis choisi, les choix tels que ceux des moteurs, des composants électroniques et des capteurs allaient être impactés. Après discussion avec notre encadrant, deux choix apparaissaient :

- Le premier était de construire nous-mêmes un châssis en le modélisant sur *Autodesk Inventor* et ensuite d'acheter les batteries, les roues et les moteurs ;
- Le second était d'utiliser un châssis tout fait avec les roues et les emplacements moteurs.

L'avantage du premier choix était que nous pouvions totalement designer le robot comme nous le souhaitions. Mais l'inconvénient se trouvait au niveau de la robustesse du robot notamment si nous utilisions un système de chenilles.

Pour le second, l'avantage était que nous n'avions pas besoin de tenir compte des problèmes de robustesse cependant les dimensions n'étaient pas modulables à notre convenance.

Avant de choisir nous avions tenté de modéliser un robot imprimable en 3D ou usinable avec les outils disponibles à l'*ENSTA Bretagne*. En voici quelques photos :

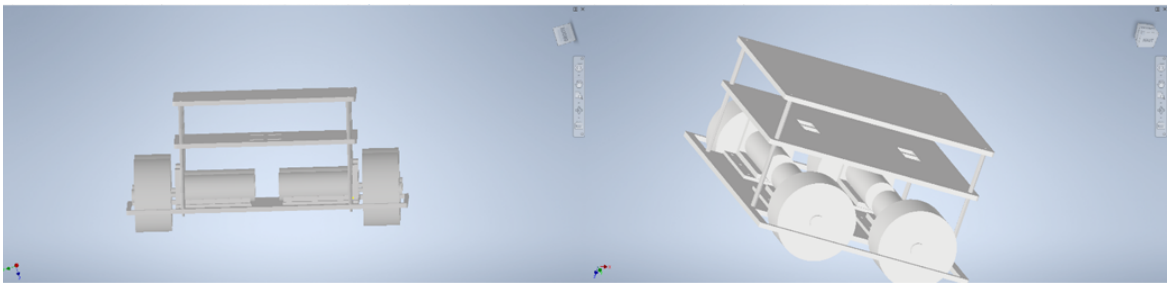


FIGURE 5 – Modélisation 3D du robot

Finalement nous avons opté pour la solution du châssis à acheter sur internet pour des raisons de précision au niveau des roues (pour notre modélisation) ou des chenilles (pour le choix final). Notre encadrant préférait également un robot construit à partir de composants sur étagères ou achetés chez des fournisseurs de son entreprise.

Nous avons donc dû faire la prospection des châssis vendus sur internet en faisant attention à la fois au prix, aux options d'achats (moteur inclus ou non) mais surtout aux dimensions et à la possibilité d'ajouter nos composants facilement.

Le choix final s'est donc porté sur un châssis robuste et qui rendait l'ajout d'étages possible simplement. Voici l'agent 390 que nous avons décidé de prendre sur *RobotShop* par rapport aux options utiles :

FIGURE 6 – Châssis Robotique Agent 390

Lien du site pour le châssis avec ses caractéristiques.

Ce châssis respectait les dimensions préconisées dans le Cahier des Charges et il permettait de ne pas prendre ces moteurs et donc de pouvoir en choisir avec un couple suffisant. Nous allons d'ailleurs maintenant parler du choix des moteurs.

3.3 Partie Électronique

3.3.1 Choix des moteurs

Le choix des moteurs a été relativement simple. Nous avons cherché à garder la même taille de moteur que ceux qui pouvaient être vendus avec le châssis. Cependant nous avons cherché un moteur avec des encodeurs et possédant un couple plus important et une vitesse de rotation qui soit proche .

Le couple nécessaire a été calculé par Monsieur *Thierry ROPERT* (professeur de mécanique) en prenant en compte le diamètre des roues, la pente maximale à franchir et le poids (frottement pas pris en compte). Nous sommes arrivés à la conclusion qu'un couple de 2 N.m était nécessaire.

Le robot devant avancer à une vitesse de 1m/s, nous avons dû calculer avec le diamètre des roues la vitesse nominale du moteur. Le résultat donne environ *200 RPM*.

Sur la base de nos besoins nous avons donc décidé de choisir le moteur suivant :

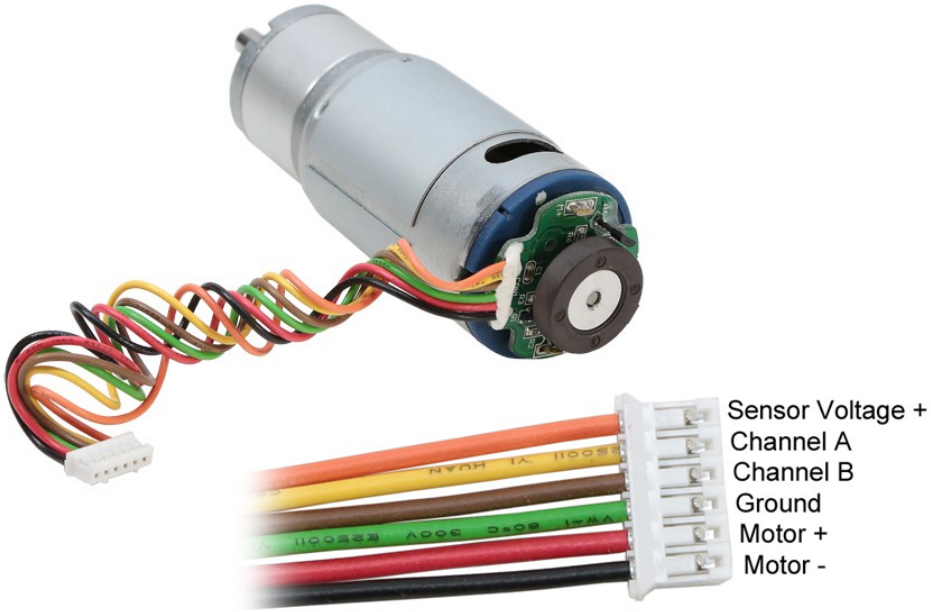


FIGURE 7 – Choix du moteur

Lien site moteur avec caractéristiques.

3.3.2 Choix capteurs

Les choix qui doivent être fait pour les capteurs sont, la caméra, le LIDAR, et les encodeurs.
La caméra a été fournie par notre encadrant, c'est une caméra Realsense T265. Cette caméra est très précise et elle est bien adaptée pour l'optométrie visuelle. Sa facilité d'intégration avec ROS en fait également un excellent choix.
Le choix du LIDAR a été fait grâce aux conseils de Monsieur **Fabrice LEBARS**. Le choix final est un RPLIDAR A2 de chez Slamteck. Il est approprié pour la détection d'obstacle en intérieur à des distances correspondant aux attendus.
Les encodeurs sont choisis par rapport au moteur, ils sont livrés avec.

3.3.3 Choix des batteries

Afin de permettre l'alimentation de tous nos composants électroniques, nous avons dû choisir une batterie. Tout d'abord de notre propre initiative nous avons décidé de séparer l'alimentation ***Moteur + Arduino UNO + Motor Shield*** de l'alimentation ***NUC + Autres Capteurs***. En effet comme nous avions vu en cours, cela permet d'éviter la sous-alimentation de certains capteurs ou contrôleurs dans le cas où les moteurs auraient besoin d'un pic de courant.
Nous avons également dû tenir compte du Cahier des Charges du robot, qui stipule environ 1h30 d'autonomie.

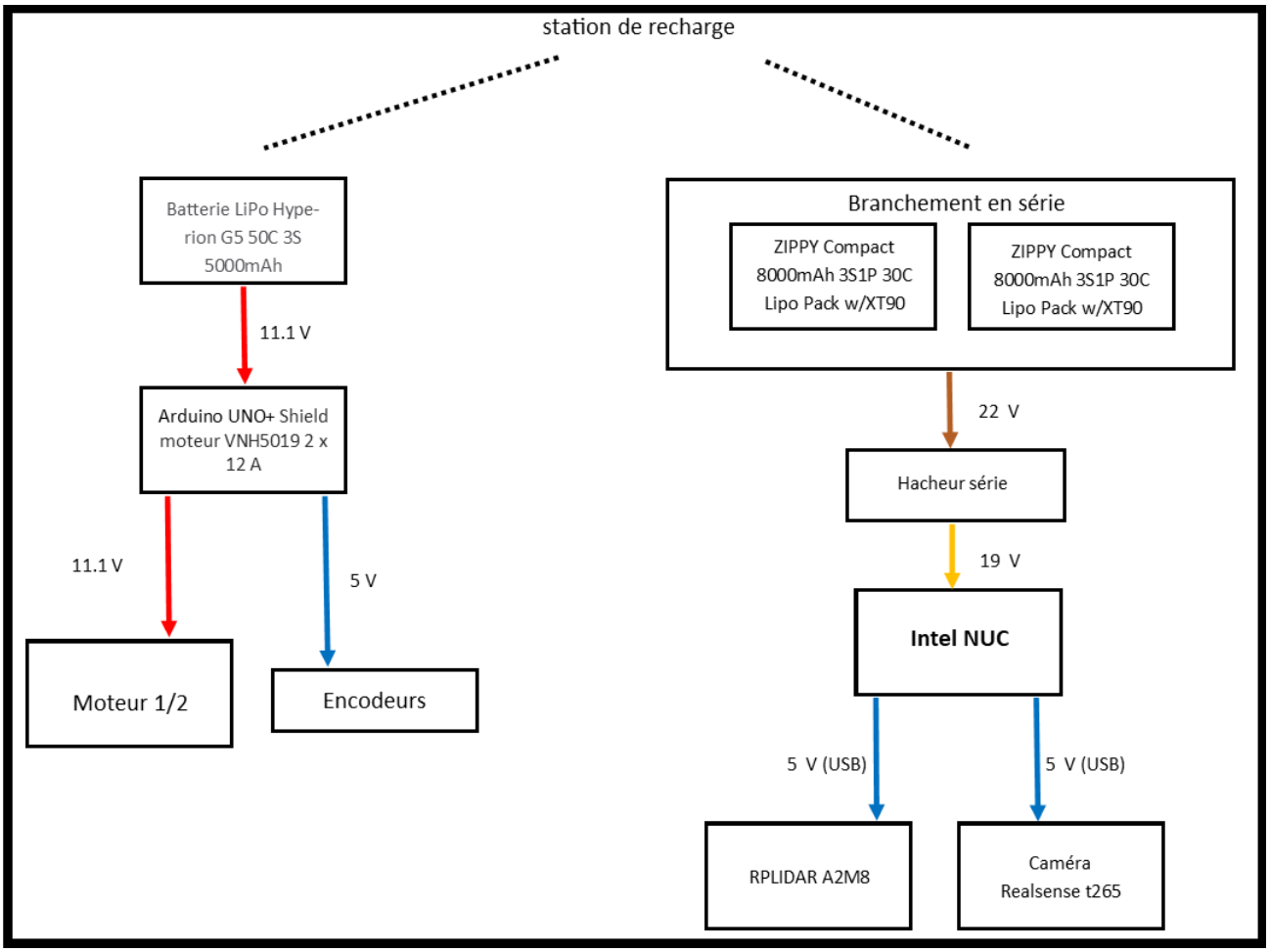


FIGURE 8 – Architecture électronique obtenue

Il est à noter que les batteries que nous possédons actuellement ne sont pas les plus appropriées puisqu'il a fallu en brancher deux en série pour obtenir l'intensité adéquat. Après une étude de la puissance nécessaire au fonctionnement des composants il a été convenu que le choix le plus approprié serait une batterie LiPo 160000 mah (voir ici). Il convient simplement de remplacer dans le schéma ci-dessus les deux batteries branchées en série par la batterie LiPo 16000 mah.

4 Choix techniques pour la Station de Recharge

4.1 Partie Mécanique

Afin de réaliser la station de recharge, nous avons souhaité la modéliser sur un logiciel de conception *Autodesk Inventor*. Ce qu'il faut rappeler tout d'abord, ce sont les caractéristiques principales que doit respecter l'ensemble *robot + station de recharge* :

- Permettre le contact des deux parties suivant différents angles d'approche du robot ;
- Permettre un contact quasi parfait afin que la station de recharge délivre sa puissance maximale au robot.

Pour le premier point, nous avons opté pour une forme arrondie qui permettrait donc au robot de pouvoir arriver de travers tout en conservant un contact suffisant avec la station de recharge.

Pour le second, nous avons décidé que la station de recharge puisse avoir un degré de liberté. Pour réaliser cela, nous avons inséré une partie à l'avant de la station de recharge qui sera soutenue par un support en dessous mais également par des ressorts de compression dans les quatre coins de la pièce.

Ces deux choix peuvent s'observer dans les figures ci-dessous :

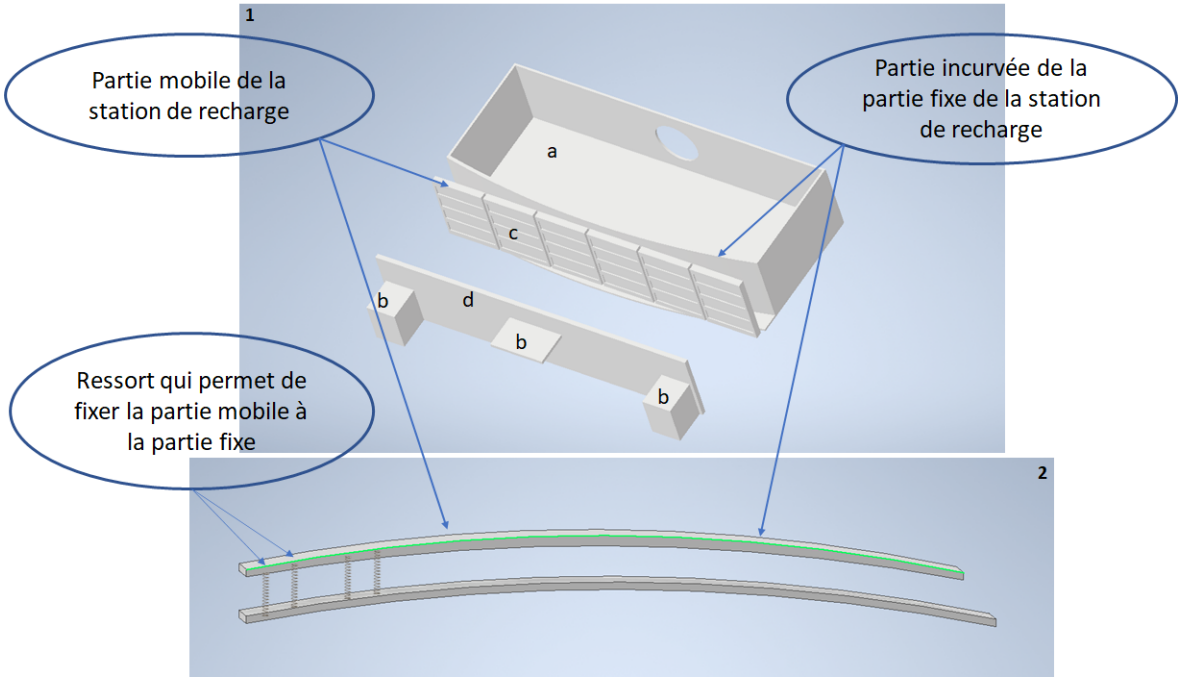


FIGURE 9 – Fonctionnement de la station de recharge en tenant compte des différentes caractéristiques

Élément de compréhension de la *Figure 1* : Sur l'image `1` il faut imaginer que lorsque le robot n'est pas en contact avec la station de recharge les parties mobiles de celle-ci reviennent à leur place initiale et forment, comme l'image `2` le montre, un arc de cercle de même rayon de courbure que la partie fixe.

Une fois la conception 3D effectuée, nous avons dû choisir les procédés de fabrication pour chaque pièce. Après consultation des professeurs du *pôle mécanique de l'ENSTA Bretagne*, nous avons convenu que la partie fixe de la station de recharge, qui contient une face incurvée (élément a), ainsi que les éléments rectangulaires permettant l'accroche de la partie contacteur sur le robot (éléments b) seraient imprimés en 3D. Tandis que les parties mobiles de la station de recharge (éléments c) et la partie contacteur du robot (éléments d) seraient, elles, usinées dans du PVC afin de gagner du temps, d'obtenir des pièces plus rigides et également de réduire les coûts de fabrication.

Pour réaliser les contacteurs, nous avons choisi un plat d'aluminium brut afin de faire circuler le courant à travers. Ce plat de largeur 1cm et d'épaisseur 2mm s'insèrera dans les encoches prévues pour les contacteurs du robot et de la station de recharge. Les tests de conductivité n'ont pas pu être effectués à cause du contexte général sanitaire cependant il paraît important de pouvoir vérifier que la puissance émise par la station de recharge puisse être acheminée jusqu'à la batterie sur le robot.

Afin de mieux visualiser la station de recharge, des figures sont à retrouver en Annexe 2.

4.2 Partie Électronique

Le deuxième partie importante se trouve être la partie électronique et électrique de la station de recharge. En effet après avoir réussi l'aspect mécanique du contact entre le robot et sa borne de recharge, nous avons dû réfléchir sur la façon de recharger la batterie *LiPo*, ainsi que de transmettre une puissance nécessaire afin de recharger rapidement le robot et ainsi satisfaire les exigences.

Le fait d'utiliser une batterie *LiPo*, complexifie légèrement l'approche de la recharge. En effet dans les chargeurs classiques *LiPo*, il y a un contrôleur qui vérifie et permet la recharge des différentes cellules de la batterie de façon à ne pas les abîmer mais également de façon à ce que chaque cellule soit chargée correctement. Ce type de chargeur "*intelligent*" nécessite une action humaine qui devait être substituée dans notre cas.
Après recherche de l'existant, nous avons rencontré un composant électronique qui permettait justement de contrôler la charge. Son nom : BMS pour Battery Management System. Ce composant surveille l'état de différents éléments de la batterie, tels que :

- la tension ;
- la température ;
- l'état de charge ;
- ou encore le courant dans la batterie.

Ce BMS s'insère entre l'alimentation qui servira à charger la batterie, et la batterie elle-même. Cependant il ne peut pas être inséré comme cela. En effet nous avons choisi un transformateur de type *ordinateur* afin de délivrer une puissance permettant une charge rapide. Cependant ce transformateur délivre une tension de $19{,}5V$ et le BMS nécessite une alimentation de $12{,}6V$. Il a donc fallu rajouter un *hacheur série* permettant de convertir la tension alimentant le BMS. Afin d'illustrer de façon plus parlante cette architecture électronique, voici une figure illustrant cela :

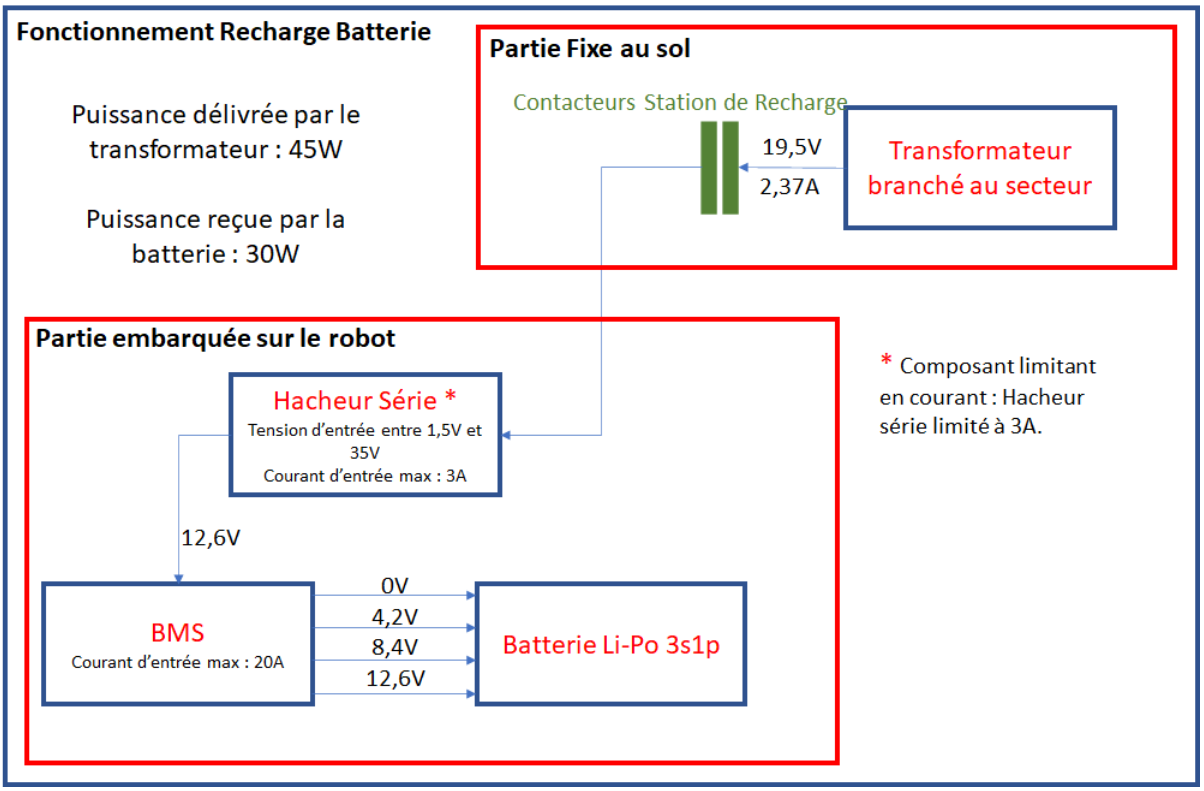


FIGURE 10 – Architecture Électronique

Ces composants permettent une charge de la batterie à $30W$ ce qui est suffisant pour un prototype. Afin d'augmenter la puissance de charge il est nécessaire d'identifier le composant limitant qui est ici le *hacheur série* (voir figure ci-dessus). En effet cet élément n'accepte pas un courant supérieur à 3A. Si l'on souhaite donc augmenter la vitesse de recharge il faudra changer ce composant qui a été choisi à la fois pour sa disponibilité et pour son coût relativement faible.

Enfin une remarque importante se situe autour de la nécessité d'alimenter le BMS en $12,6V$. En effet nous n'avons trouvé aucune documentation en ligne sur le BMS dont nous avions besoin afin de charger une LiPo 3S. La seule indication qui nous a poussés à utiliser un *hacheur série* se trouve dans une vidéo YouTube montrant la recharge d'une batterie avec le BMS en question (lien ici). Dans une continuité de projet, il serait intéressant de tester avec et sans le *hacheur série*, au vue du faible coût de ce BMS. Un tel test serait intéressant afin de choisir un transformateur délivrant un courant supérieur, afin d'augmenter la puissance reçue par la batterie et ainsi de diminuer le temps de charge.

4.3 Partie Informatique

Nous allons maintenant traiter la partie informatique de la Station de Recharge. En effet bien que la charge soit autonome et ne nécessite pas de programmation particulière, le robot doit recevoir les informations suivantes :

- si le robot est bien en contact avec la station de recharge ;
- l'état de charge de la batterie.

Pour réaliser cela, voici un schéma mettant en évidence les composants nécessaires :

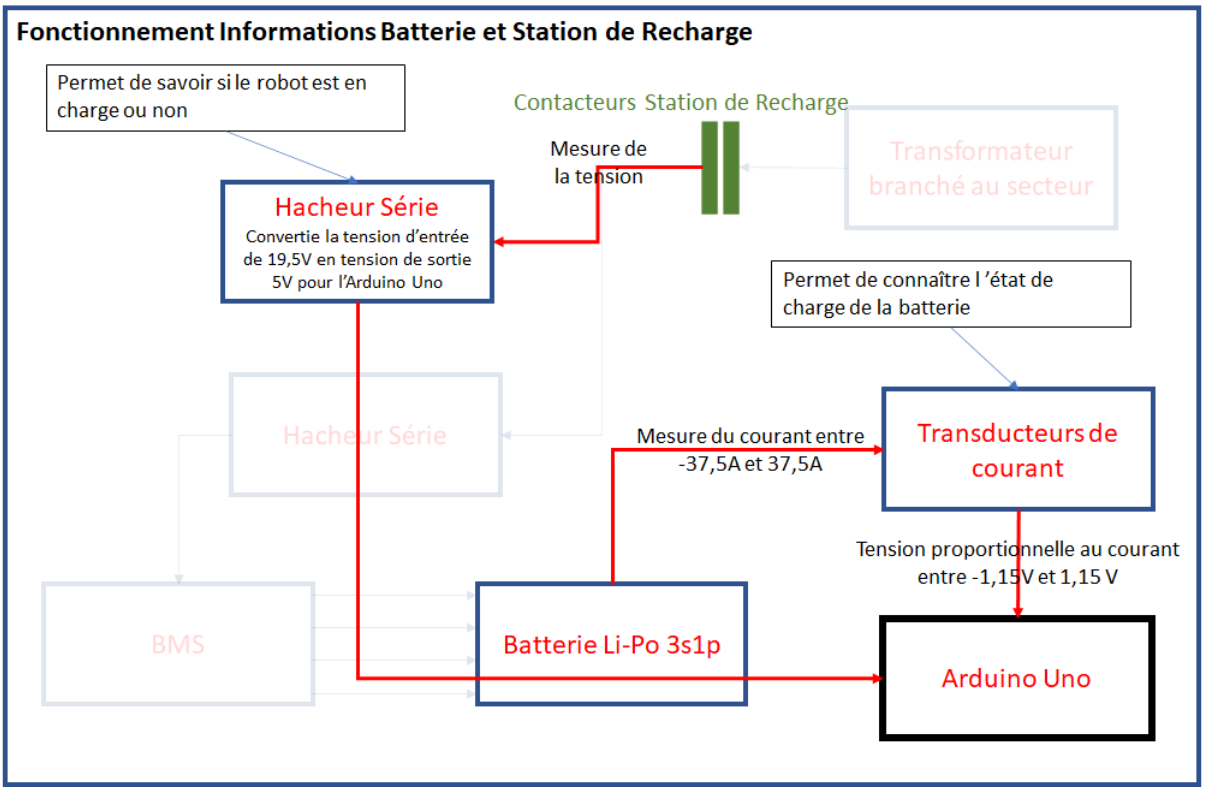


FIGURE 11 – Manières d'obtenir les informations importantes pour le robot

Ce schéma illustre les deux composants nécessaires à la prise d'information :

- D'une part un hacheur série connecté à un pin de l'Arduino Uno qui permet de savoir si de la tension passe le long des contacteurs de la partie robot. Cela permet de savoir si le contact avec la station de recharge est fait ou non.
- D'autre part un transducteur de courant qui mesure la courant circulant au sein de la batterie et qui renvoie comme information, une tension proportionnelle à l'intensité mesurée, sur un pin de l'Arduino Uno .

Il ne restera plus qu'à fournir un code Arduino qui renverra par topics ROS ces informations.

4.4 État d'avancement de la station de recharge

Nous allons maintenant faire un point sur l'avancement de la Station de Recharge qui était une partie attendue par Monsieur CHODOROWSKI. Vous pouvez retrouver la liste des composants en Annexe 3.

D'un point de vue mécanique, la station de recharge est opérationnelle sur le papier. Il reste :

- à tester la conductivité des contacteurs ;
- à voir comment accrocher de façon simple, les ressorts qui permettent la liaison entre la partie mobile et la partie fixe de la Station de Recharge ;

Un premier test avait été effectué en laissant délibérément des petites encoches carrées dans les petites pièces rectangulaires mobile, et cela tenait. Nous n'avons pas pu tester l'accroche sur la partie fixe incurvée.

Pour la partie électronique, du fait de la séparation des alimentations moteur et contrôleur, le système et les composants mis en place devront être dupliqués afin que chaque batterie ait son propre BMS associé. Cela ne modifie en rien les contacteurs qui resteront les mêmes, il faudra cependant envisager de choisir un *transformateur* délivrant un courant plus important, et donc envisager de changer de hacheur série.

Enfin, du côté informatique, il ne restera plus qu'à fournir un code Arduino qui renverra par topics ROS ces informations. Cela ne pouvait se faire qu'en ayant les composants en main propre.

5 Intégration des composants

Dans cette partie nous aborderons le thème de l'intégration des différents composants avec le middleware ROS.

5.1 Encodeur

5.1.1 Procédure générale

Tout d'abord nous allons aborder l'aspect intégration sous ROS des *moteurs et encodeurs* et la commande des moteurs via topic ROS.

Tout d'abord, cette partie a été effectuée entièrement à distance, et l'ensemble de la commandabilité et contrôlabilité du robot n'a pu être faite. Les composants que nous disposions durant le confinement sont :

- L'ensemble carte Arduino UNO + Motor Shield,
- Un moteur et son encodeur,
- Une batterie LiPo.

Dans ce cadre limité, nous avons redéfini des objectifs atteignables qui sont : de réussir à s'abonner à un topic ROS émit par le NUC qui envoie la commande moteur, et de publier via l'Arduino un topic contenant le nombre de *ticks* de l'encodeur.

Une fois cette mise au point faite, nous avons commencé par faire fonctionner le moteur et lire les valeurs de l'encodeur uniquement en utilisant un code Arduino. Cela nous a offert une meilleure compréhension du fonctionnement de ces composants. Afin de commander le moteur nous avons utilisé une librairie qui se nomme : "DualVNH5019MotorShield". Son installation sur Arduino est expliquée dans le README de ce lien.

Le code Arduino qui permet uniquement de lire la valeur des encodeurs se trouve ici.

Une fois avoir maîtrisé les commandes Arduino pour le moteur et l'encodeur, nous sommes passés sur la partie intégration sous ROS.

Pour cela nous avons dû installer ROS sur Arduino. Nous avons donc suivi cette procédure en remplaçant la version de ROS indigo par la version du ROS utilisé. Pour nous ce fut le ROS Melodic. En suivant toutes ces instructions et en relançant Arduino, dans les exemples nous pouvons retrouver des exemples de ROS sous Arduino.

Pour lancer le programme il faut se référer au README du dossier PubSub. Une fois les instructions suivies le programme fonctionne et le moteur se lance avec la commande donner dans le code C++ "Talker.cpp".

Les explications de code et les problèmes rencontrés au moment de la programmation, sont à retrouver en Annexe 4 page 25.

5.1.2 État d'avancement de l'intégration des Encodeurs

L'intégration des Encodeurs sous ROS est presque terminée. Il reste à modifier les types de message dans lesquels nous publions les informations. Cela sera relativement rapide si Arduino accepte le type de message en question.

Également nous n'avons pas eu, ni le temps ni les moyens d'effectuer le code permettant le contrôle du robot. Ainsi il reste à développer une régulation du robot en fonction de sa vitesse, de la trajectoire souhaitée, des obstacles détectés par le Lidar et enfin en fonction des erreurs corrigées par le PID.

5.2 Caméra Realsense

Pour la caméra realsense T265 que nous utilisons, il existe déjà un module ROS permettant la publication de tout les topics que nous utilisons. Ici, le lien pour l'installation du SDK.

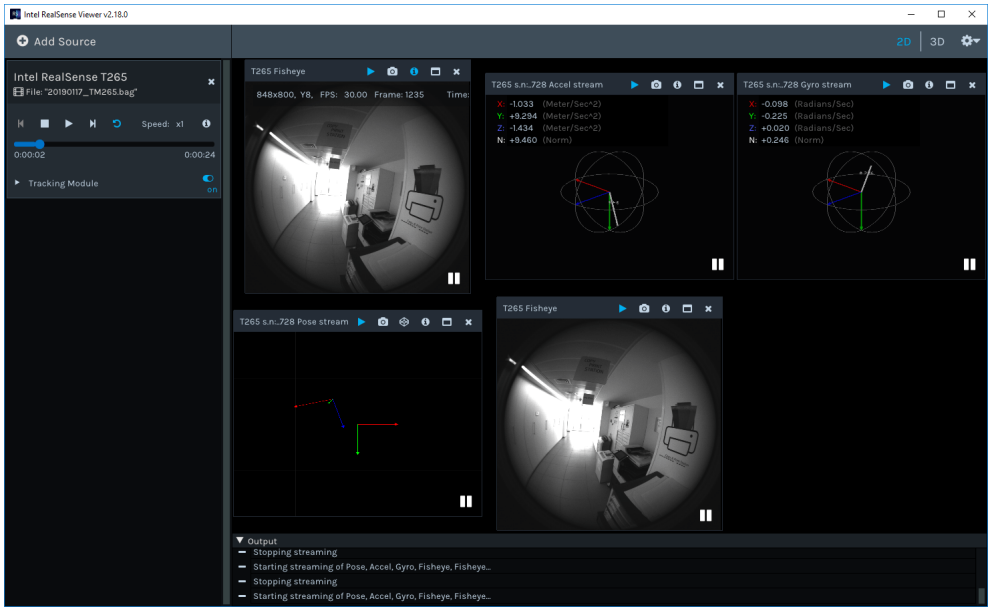


FIGURE 12 – Outil de visualisation pour la caméra Realsense

Une fois ce module installé, il suffit de lancer les nodes avec la commande :
roslaunch realsense2-camera rs-camera.launch

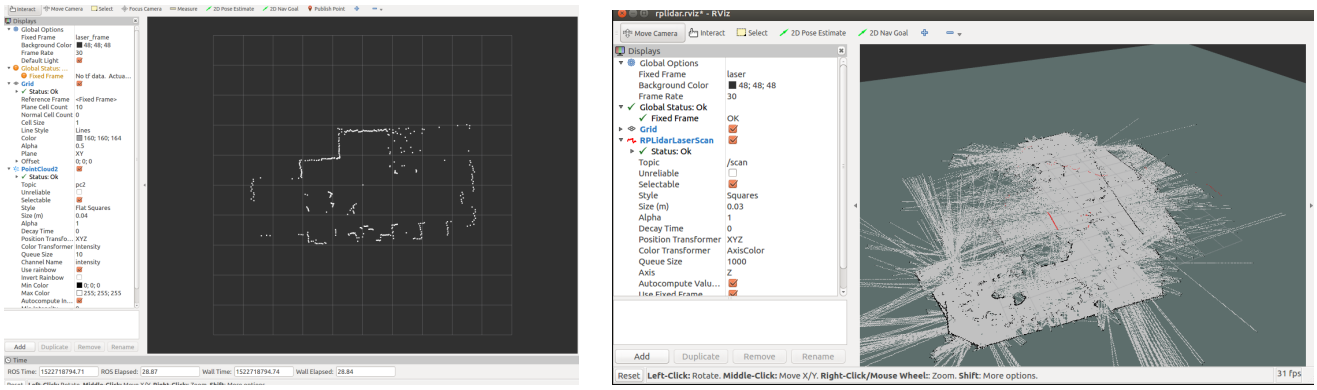
Les topics publiés qui sont intéressants pour notre projet sont :

- Les topics image brute,
- Le topic IMU,
- Le topic odométrie.

5.3 RP Lidar

Le LIDAR que nous utilisons est un RPLIDAR A2 de chez Slamtech, l'ensemble des SDK nécessaires à l'utilisation de ce LIDAR sont fournis par le constructeur(lien ici)

Pour l'installation, il faut tout d'abord installer le module permettant de faire fonctionner le LIDAR puis le module permettant le lancement avec ROS (et donc la visualisation avec RVIZ). Une fois ces packages installés, on peut utiliser le LIDAR et avoir accès au topic classique **scan**.



a) Détection des points b) Application du SLAM

FIGURE 13 – Visualisation avec RVIZ

Touts les fichiers contenant les drivers sont à retrouver sur le Git du projet. Le node-graph est, quant à lui, à retrouver en page 27.

6 Résultats et continuité

Dans cette ultime partie, nous allons traiter des résultats et de l'avancement général du projet. Nous parlerons également des possibilités de suite et de la continuité des différentes tâches. Pour cela, nous allons diviser la suite en deux paragraphes : le premier pour parler du robot et le second pour la station de recharge.

Concernant le robot, l'installation sur le robot des capteurs tels que le Lidar ou la caméra Realsens n'ont pas pu être fait puisque nous n'avions plus le robot sous la main depuis début Mars. Concernant les intégrations sous ROS du NUC, du Lidar et de la caméra, elles sont opérationnelles et ne nécessitent qu'un code qui gère la navigation. L'alimentation a été traité antérieurement, elle devra être tout de même testée en situation réelle. Le mât a été délibérément abandonné dans le contexte actuel mais avec une éventuelle reprise de projet, ceci redeviendra un critère important. Concernant le châssis, celui-ci permet d'accueillir tous les capteurs et installations souhaités. Les intégrations des moteurs et encodeurs a été effectuées. Il reste cependant à coder un programme permettant de contrôler la vitesse et la trajectoire du robot.

Concernant la station de recharge, nous n'allons pas nous attarder dessus. Vous pouvez retrouver page 15 l'état d'avancement de la station, les tâches à réaliser et les possibles

améliorations envisagées. Pour résumer, toute la partie théorique a été effectuée. Il reste l'aspect pratique et le test des solutions envisagées.

7 Conclusion

Afin de conclure ce rapport mais également ce projet, nous tenions à remercier de nouveau Monsieur CHODOROWSKI qui nous a donné l'opportunité et les moyens de travailler sur ce beau défi.

Les premiers objectifs fixés ont, certes, été revus à la baisse, néanmoins les leçons apprises tout au long de ce travail sont très nombreuses. Nous n'allons pas dresser la liste de chacune, cependant au delà de l'aspect théorique et pratique en robotique, nous avons acquis tout au long de ce projet des compétences de gestion et de suivi de projet. Les rencontres et les discussions hebdomadaires avec notre encadrant, les libertés de décisions qu'il nous a offertes, la répartition du travail entre nous, tout ceci nous a permis de mieux découvrir le travail d'un ingénieur en relation avec une entreprise commanditaire d'un robot.

Ce projet professionnalisant nous servira à coup sûr dans nos expériences futures, que ce soit en stage ou dans notre futur métier.

Pour conclure, notre travail de présentation, d'explication et de synthèse a pour vocation la continuité de ce projet. Nous espérons qu'avec toutes nos recherches et nos travaux, ce projet ambitieux pourra arriver à son terme. Nous sommes heureux d'avoir pu y ajouter notre contribution.

Table des figures

8 Annexe

8.1 Annexe 1 : Liste matériel pour le robot

Nom Composant	Nom Fournisseur	Quantité	Prix unitaire	Commentaire
Intel NUC NUC8BEK	Fourni par OrangeLabs	1	Non spécifié	Ordinateur embarqué permettant de contrôler le robot
RPLIDAR A2M8	Slamtec	1	301,32 €	Permet la localisation et la détection d'obstacle alentours
Caméra realsense t265	Fourni par OrangeLabs	1	Non spécifié	Odométrie visuelle + IMU
313 RPM HD Premium Planetary Gear Motor w/Encoder	Robotshop	2	54,43 €	
Shield moteur VNH5019 2 x 12 A	Gotronic	1	51,90 €	Pour la commande des moteurs
Carte Arduino UNO	Gotronic	1	21,50 €	Pour l'acquisition des encodeurs et la communication moteur/NUC
Chassis Agent 390	Servocity	1	297,73 €	Support
Batterie LiPo Hyperion G5 50C 3S 5000mAh	Robotshop	1	50,58 €	Alimentation
EPS - Batterie Lipo 22,2V (6S) 16000 mAh 25C prise AS150/XT150 - DJI	cdiscount	1	200,00 €	Alimentation

FIGURE 14 – Liste matériel pour le robot

8.2 Annexe 2 : Complément Modélisation Station de Recharge

Les modélisations suivantes permettent une meilleur compréhension de la partie mécanique de la station de recharge.

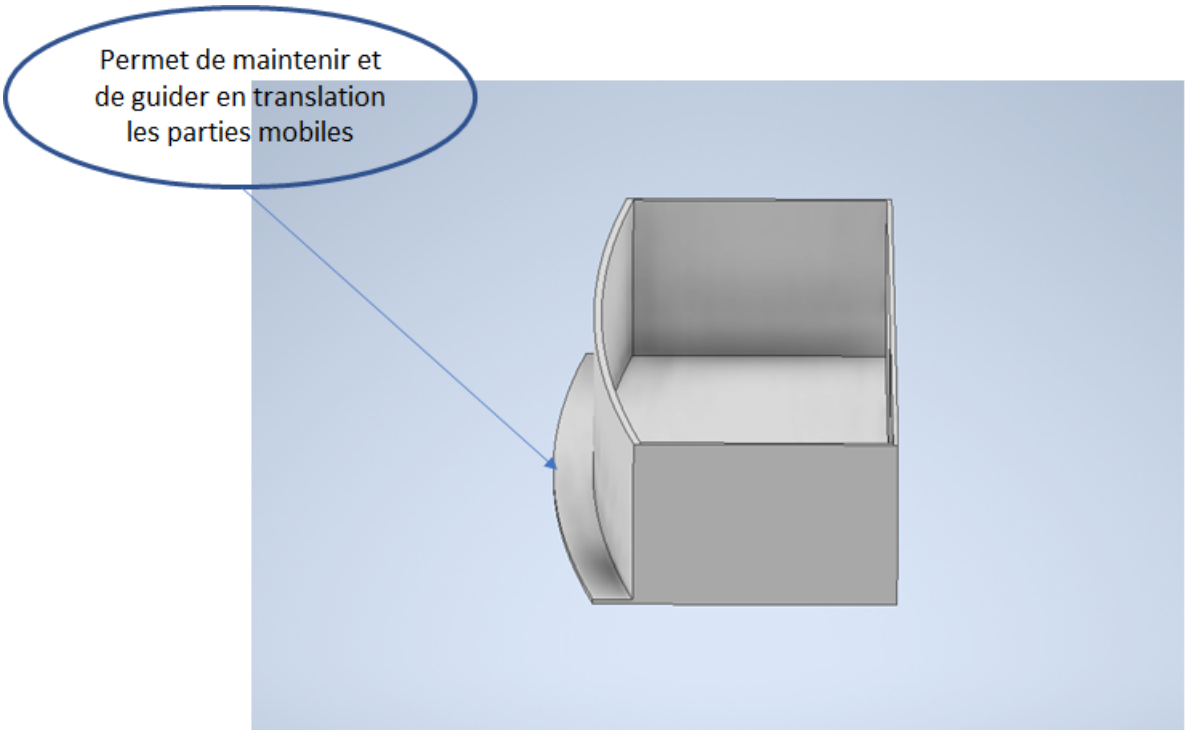


FIGURE 15 – Partie Fixe Station de Recharge

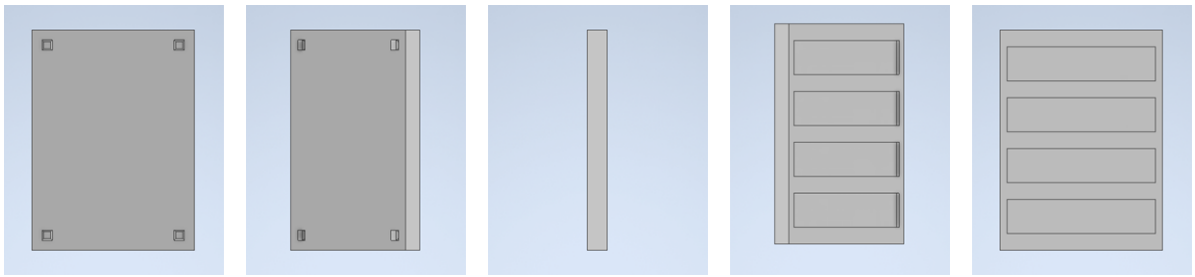


FIGURE 16 – Partie Mobile Station de Recharge

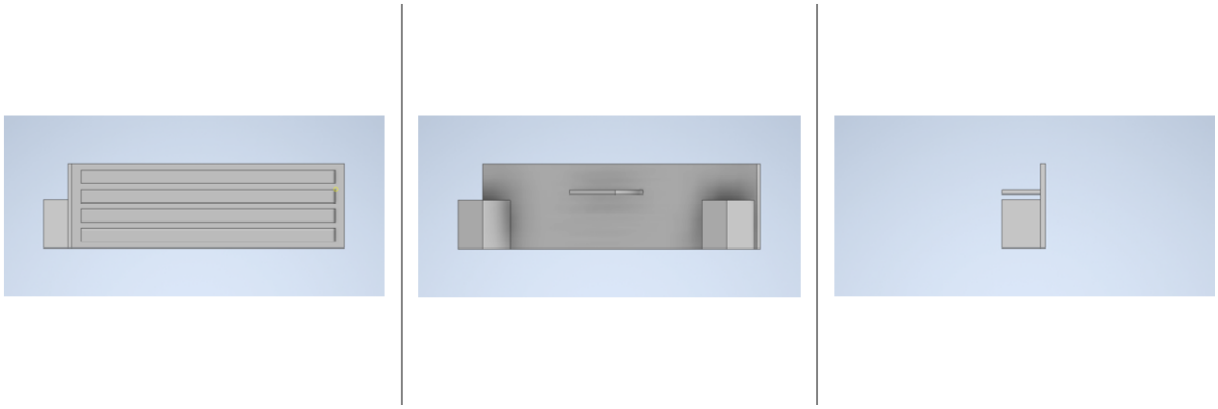


FIGURE 17 – Partie Fixe Robot

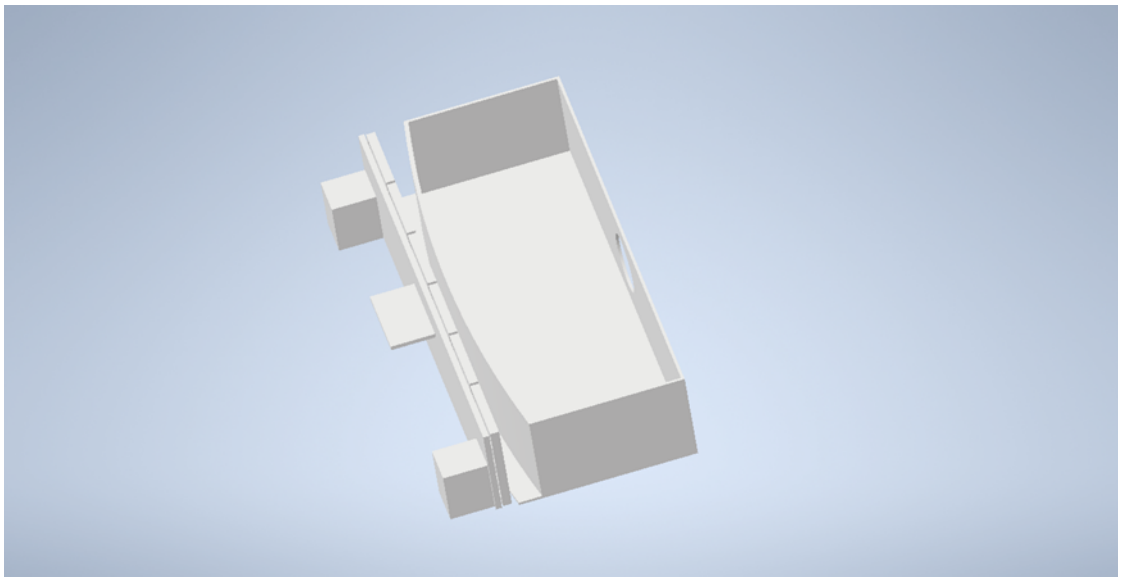


FIGURE 18 – Contact Robot + Station de Recharge

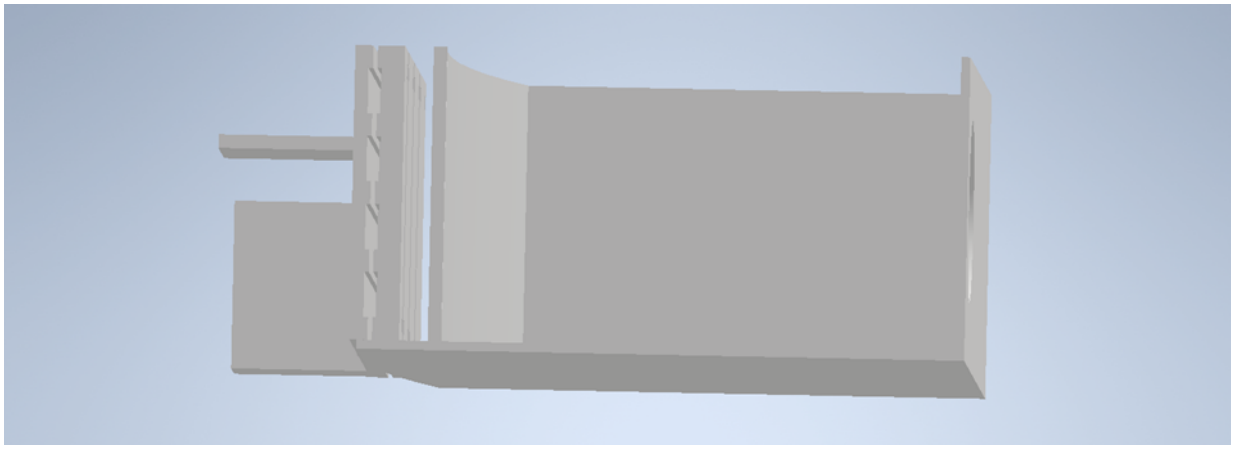


FIGURE 19 – Contact Robot + Station de Recharge de profil

8.3 Annexe 3 : Liste matériel pour la station de recharge

Nom Composant	Fournisseur	Quantité	Prix unité	Commentaires
Battery Management System	Amazon	2 pour pouvoir effectuer le test d'alimentation (cf commentaire hacheur série)	5,39€	Permet de gérer la charge de la batterie 3S. Courant Maximal 20A (pas limitant)
Hacheur série 3V-40V, 3A	Amazon	2	5,83€	Permet d'adapter la tension de sortie
Transformateur 19.5V / 2,37A / 45W	Amazon	1	39,49 €	Permet l'alimentation de la station de recharge
Ressort de compression 15,7mm	RS	3 (paquets de 10)	4,75€	Permet la liaison amovible des contacteurs de la station de recharge
Transducteurs de courant	RS	1	8,95£	Permet de mesurer le courant de la batterie et donc de savoir le % de batterie restant au cours de la mission
Plaque ALU pour contacteur	LeRoyMerlin	2	6€	Permet de faire les contacteurs

FIGURE 20 – Liste matériel pour la station de recharge

8.4 Annexe 4 : Tutoriel codes Encodeurs et Moteurs

8.4.1 Explication des codes

Le code Arduino qui permet de commander le moteur en envoyant une commande entre -400 et 400 via les fonctions *setM1Speed* et *setM2Speed* (dans le cas où il y a deux moteurs) se trouve ici. Le code a été écrit de telle sorte à pouvoir piloter les moteurs via le clavier numérique :

- "8" pour avancer,
- "5" pour s'arrêter,
- "2" pour reculer.

C'est à partir de l'exemple *pubsub* que nous avons construit le code permettant de lire la commande sur un topic et de publier sur un autre les valeurs encodeurs. Pour le code suivre ce lien. Le code en lui même est relativement simple, cependant le ROS sous Arduino est très légèrement différent que sous C++ et quelques problèmes sont apparus comme par exemple le fait de devoir utiliser un *char ** afin de récupérer les valeurs de commandes moteurs.

Pour le moment le code ne reçoit et n'envoie que des topics de type String, dans les besoins du projet il faudra utiliser des messages déjà implémentés comme par exemple un message de type Twist.

Enfin afin de publier les topics de commande, nous avons implémenté un code que l'on peut retrouver dans un workspace ROS sur notre GitHub. Les codes sont dans le dossier /src qui contient deux codes en C++ :

- Talker.cpp qui permet de publier sur un topic ROS nommé *cmd motor* la commande moteur,
- PID.cpp qui est un exemple de PID utilisé sur un autre projet et qui pourra être réadapté pour ce projet.

8.4.2 Problèmes rencontrés

L'un des principaux problème fut au moment de publier la valeur de l'encodeur. En effet avec les premières versions nous pouvions lire sur le topic "value encoder" des valeurs qui ne correspondaient pas aux valeurs attendues. En effet nous avions environ 50 ticks par tour alors que la documentation moteur indiquait 322 ticks par tour et que nous avions déjà testé l'encodeur uniquement avec Arduino et les valeurs étaient cohérentes.

Après recherche nous avons réalisé que la fréquence à laquelle nous parcourions la boucle loop() du programme Arduino n'était pas suffisante par rapport à la fréquence de rotation du moteur. Autrement dit, nous ne lisions qu'une valeur encodeur sur six. Pour comprendre d'où venait le problème, nous avons dû regarder combien de cycles d'horloge durait chaque commande dans le loop(). Les résultats de ces tests sont en commentaires du programme Arduino "pubsub.ino". Nous avons constaté que ce qui prenait le plus de temps était la conversion en *char ** ainsi que la publication du topic *"value encoder"* et de la commande "*spinOnce()*.

Afin d'y remédier, nous avons choisi de ne publier la valeur du topic *"value encoder"* que toutes les 100 passages de boucle loop(). Ceci nous a permis de pouvoir lire la valeur encodeur à chaque loop() de façon plus rapide et ainsi éviter de sauter des valeurs encodeurs, tout en publiant assez régulièrement afin d'afficher tous les changements de ticks.

8.5 Annexe 5 : Node graph

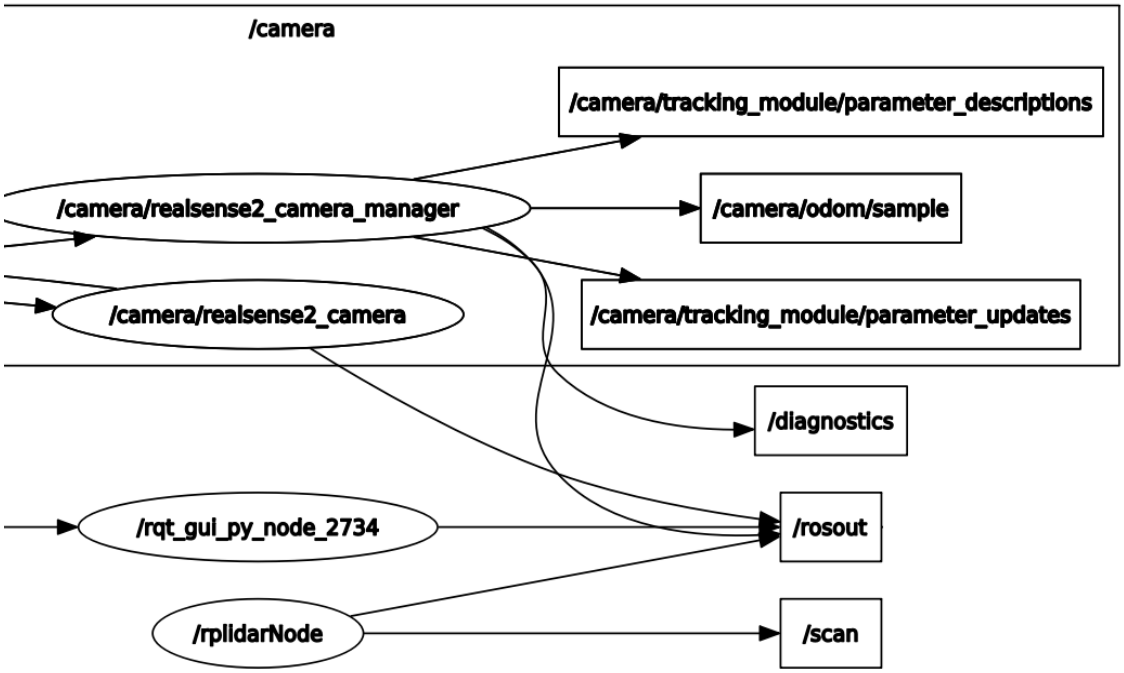


FIGURE 21 – Node Graph : LIDAR et Realsense