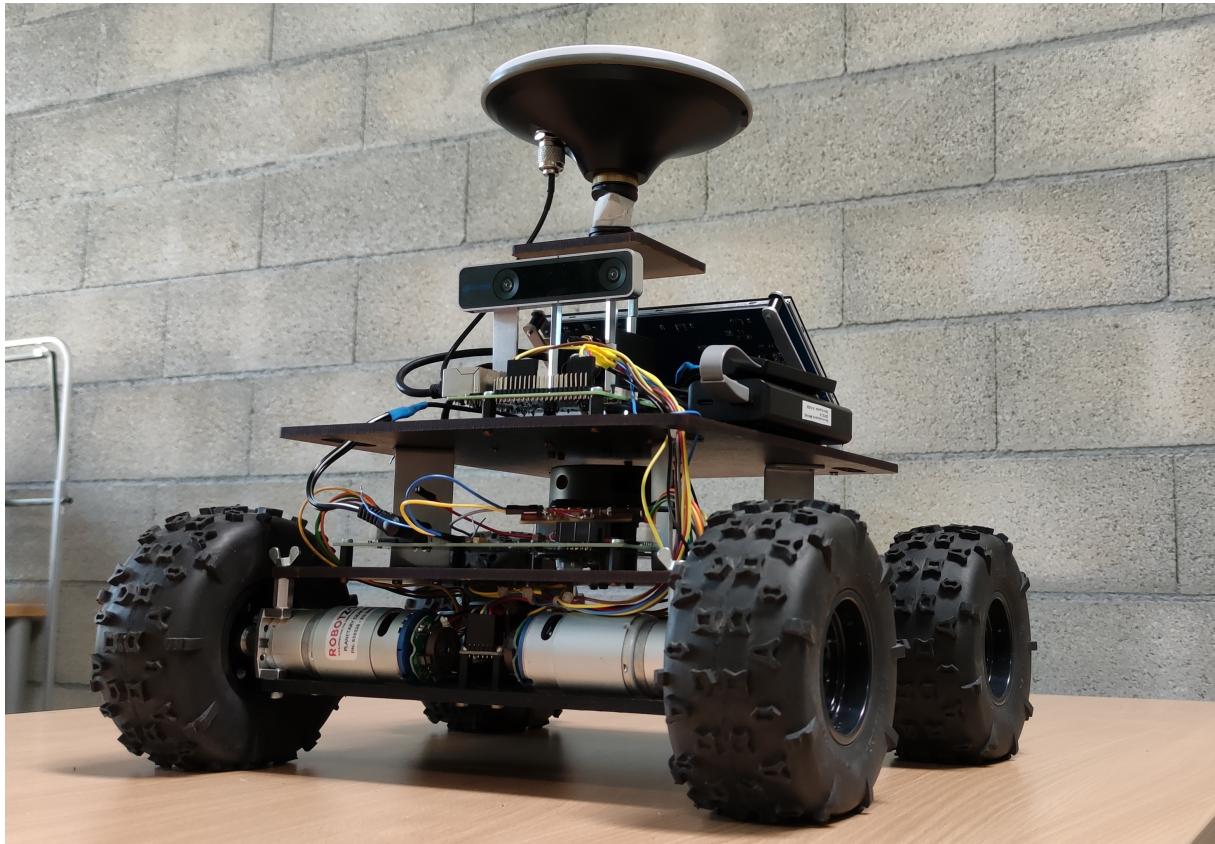


# Agribot - Specs

Robot REDS

A.Gabriel Catel Torres

October 7, 2021



# List of Figures

1	Chassis Agribot . . . . .	5
2	Chassis Agribot - Pièces . . . . .	5
3	Chassis Agribot - Mesures . . . . .	6
4	Moteurs avec encodeurs . . . . .	7
5	Moteurs avec encodeurs - Mesures . . . . .	7
6	Batterie moteurs . . . . .	8
7	Batterie système . . . . .	8
8	Drivers Moteurs Cytron MD10C r3 . . . . .	9
9	Drivers Moteurs - câblage . . . . .	9
10	i2cPWM board - ServoShield . . . . .	10
11	i2cPWM board - Dimensions . . . . .	10
12	REDS sensor board . . . . .	11
13	Raspberry Pi 4 . . . . .	12
14	Raspberry Pi 4 - Mesures . . . . .	12
15	RPLidar A1 . . . . .	13
16	RPLidar A1 - Mesures . . . . .	13
17	Xsens MTI 680G . . . . .	14
18	Xsens MTI 680G - Mesures . . . . .	14
19	RealSense T265 . . . . .	15
20	RealSense T265 - Mesures . . . . .	15
21	Ecran tactile . . . . .	15
22	Agribot - robot . . . . .	16
23	Agribot - Etage 1 . . . . .	17
24	Agribot - Etage 2 . . . . .	18
25	Agribot - Etage 3 . . . . .	19
26	Schéma bloc Agribot . . . . .	20
27	Blocs software . . . . .	23
28	Noeuds de la base mobile . . . . .	24
29	Schéma du fonctionnement des ROS controller . . . . .	25
30	Schéma du fonctionnement des ROS controller . . . . .	25
31	Schéma bloc de la navigation . . . . .	26
32	Modifications pour la navigation en intérieur . . . . .	27
33	Structure software pour la navigation en intérieur . . . . .	28

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Cadre du projet . . . . .	4
1.2	Analyse de la problématique . . . . .	4
<b>2</b>	<b>Structure hardware</b>	<b>5</b>
2.1	Composants mécanique . . . . .	5
2.1.1	Chassis et roues . . . . .	5
2.1.2	Moteurs . . . . .	7
2.2	Alimentation du système . . . . .	8
2.2.1	Batterie moteurs . . . . .	8
2.2.2	Batterie système . . . . .	8
2.3	Composants électroniques . . . . .	9
2.3.1	Drivers moteurs - Cytron . . . . .	9
2.3.2	i2cPWM board - ServoShield . . . . .	10
2.3.3	REDS sensor board . . . . .	11
2.3.4	Raspberry Pi 4 . . . . .	12
2.4	Capteurs . . . . .	13
2.4.1	Scanner laser 360° - RPLidar A1 . . . . .	13
2.4.2	GNSS IMU - Xsens MTI 680G . . . . .	14
2.4.3	Camera - T265 . . . . .	15
2.4.4	Ecran tactile . . . . .	15
<b>3</b>	<b>Structure Agribot</b>	<b>16</b>
3.1	Robot final . . . . .	16
3.2	Etage 1 . . . . .	17
3.3	Etage 2 . . . . .	18
3.4	Etage 3 . . . . .	19
3.5	Schéma bloc Agribot . . . . .	20
3.6	Schéma du câblage . . . . .	21
<b>4</b>	<b>Software</b>	<b>22</b>
4.1	Environnement - ROS . . . . .	22
4.1.1	Qu'est ce que ROS ? . . . . .	22
4.1.2	Pourquoi ROS ? . . . . .	22
4.1.3	Définitions . . . . .	22
4.2	Architecture software d'Agribot . . . . .	23
4.2.1	Robot REDS . . . . .	23
4.2.2	Architecture software globale . . . . .	23
4.2.3	Architecture base mobile . . . . .	24
4.2.4	Architecture de la navigation . . . . .	26

# 1 Introduction

## 1.1 Cadre du projet

La ferme BIO du Moulin à Bavois souhaite utiliser un robot d'aide à l'agriculture durable qui soit capable de se déplacer de manière autonome dans un champ et d'arroser des plantons. L'arrosage de ceux-ci est une tâche importante pour la croissance, sachant qu'ils ont une faible capacité à se fournir en eau étant donné que leurs racines ne sont pas encore bien développées. Le stress d'un manque d'eau sur le planton peut produire de fort ralentissement de croissance et retarder la récolte. L'utilisation du robot permettra de limiter les besoins en eau et faciliter un arrosage responsable avec un arrosage ciblé.

## 1.2 Analyse de la problématique

Ce projet se divise en plusieurs grandes parties distinctes qui permettront, une fois développées, de répondre au besoins précédemment évoqués.

Une étude approfondie a été menée pour définir les composants électroniques et mécaniques nécessaires pour mettre en place un système capable de se déplacer de manière autonome en extérieur.

Il a fallu permettre la reconnaissance de planton par imagerie, la réalisation du bras d'arrosage, le déplacement autonome du robot, dans un premier temps en intérieur et maintenant le déplacement autonome du robot en extérieur avec la localisation GPS. Le but étant de pouvoir fournir des coordonnées GPS ou une carte de champ au robot et qu'il puisse de manière autonome, arroser les plantons avec, si possible, une précision au centimètre. Les méthodes de localisation vont chacune apporter des avantages et des inconvénients, ceux-ci seront mis en avant dans les prochains chapitres.

## 2 Structure hardware

Cette section liste les différents composants présents sur le système. Chaque composant est brièvement décrit et nous expliquons son utilité au sein du projet.

### 2.1 Composants mécanique

#### 2.1.1 Chassis et roues

La structure finale sera différente. Ce chassis est provisoire et permet de mettre en place la structure logicielle nécessaire pour le projet final.



Figure 1: Chassis Agribot

La liste des composants utilisés pour monter la base sont visibles sur la Figure 2

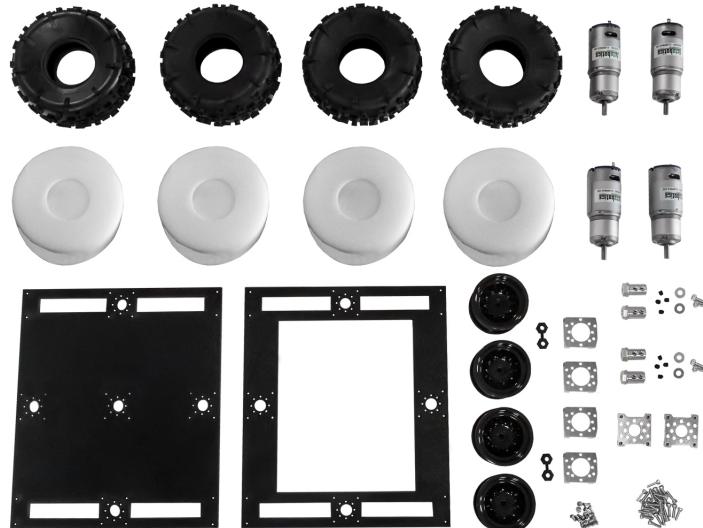


Figure 2: Chassis Agribot - Pièces

Ces informations ne sont pas indispensables puisque la base est fournie en partie montée. Il est toutefois intéressant d'avoir cette liste pour pouvoir se munir de pièces de rechange si nécessaire.

La Figure 3 la schématique du chassis avec les mesures utiles. Ces mesures sont importantes car elles permettent de correctement calibrer le logiciel. Plus la description du matériel est précise, plus le logiciel sera performant (précision et temps de calculs). Une bonne pratique est donc de vérifier ces mesures et d'adapter au besoin les paramètres du logiciel.

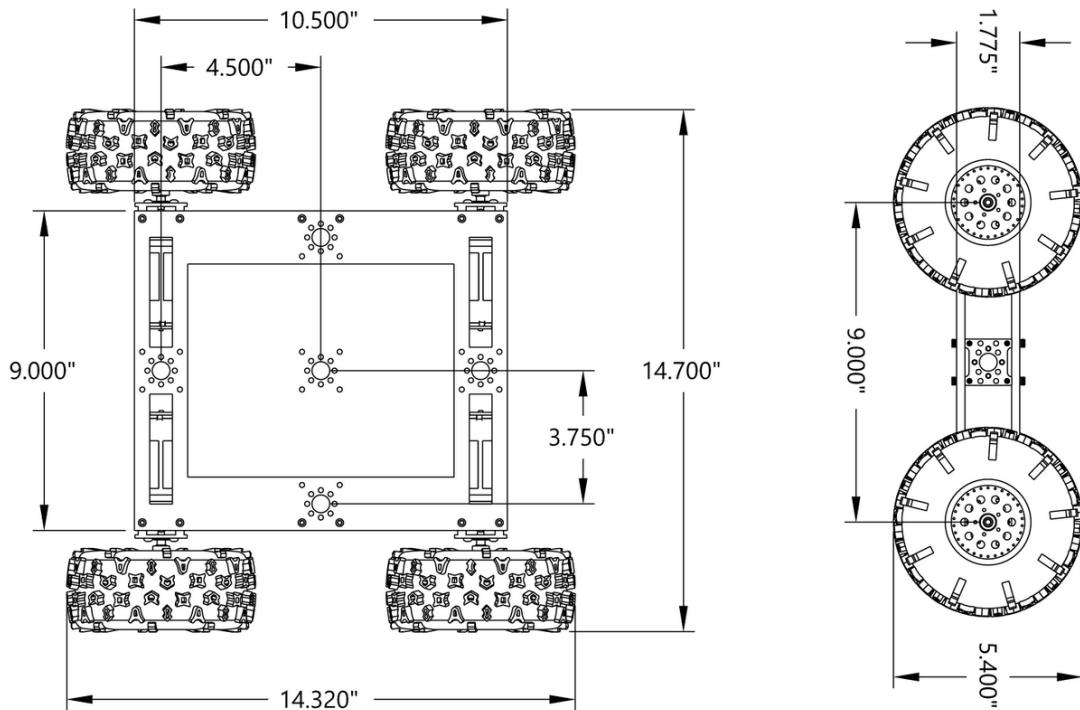


Figure 3: Chassis Agribot - Mesures

Pour plus d'informations sur le chassis, se rendre sur le lien suivant : [Chassis-ServoCity My Images Folder hardware\\_doc](#)

### 2.1.2 Moteurs

Le robot fonctionne avec une gestion des roues en différentiel. Cela signifie que chacune des roues est dirigée par un moteur, donc que le système possède quatre moteurs permettant de contrôler de manière indépendante chaque roue.

Des moteurs de base sont fournis avec le châssis. Cependant, ils ne sont pas adaptés aux besoins du projet. En effet, nous avons besoin d'encodeurs sur les moteurs pour permettre la localisation du robot et ainsi permettre de le rendre autonome.

Pour cela, nous avons utilisés les mêmes moteurs que ceux fournis de base, mais avec des encodeurs permettant de connaître



Figure 4: Moteurs avec encodeurs

La figure 5 donne les mesures pour un moteur.

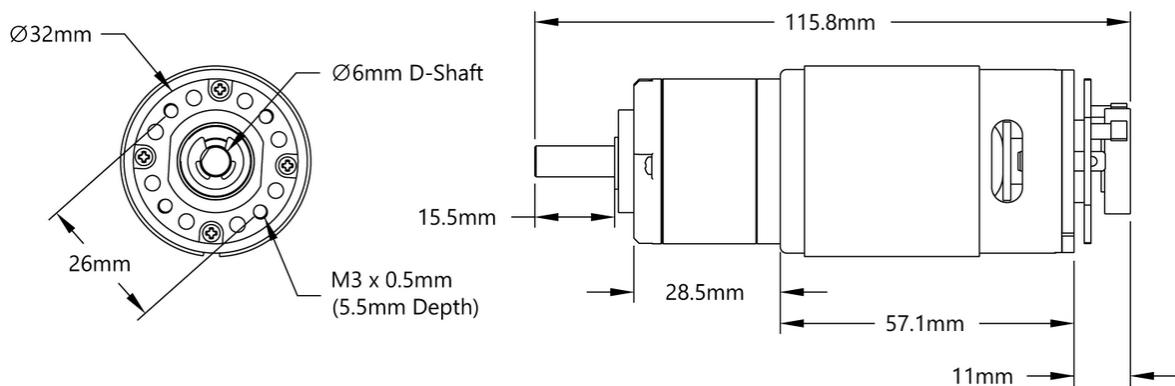


Figure 5: Moteurs avec encodeurs - Mesures

## 2.2 Alimentation du système

### 2.2.1 Batterie moteurs

Les 4 moteurs doivent être alimentés. Une tension entre 6V et 12V est recommandée (12V est la tension nominale). Les moteurs peuvent chacun demander jusqu'à 20 ampères au maximum. Cette batterie a été retenue et est dédiée à l'alimentation des moteurs exclusivement.



Figure 6: Batterie moteurs

### 2.2.2 Batterie système

Cette batterie est prévue pour alimenter une Raspberry Pi 4 et une carte réalisée par le REDS pour les besoins du projet (REDS sensor board). La tension nécessaire est de 5V et le courant minimum est de 3A pour éviter les chutes de tension qui provoquent des redémarrages non souhaités des composants.



Figure 7: Batterie système

## 2.3 Composants électroniques

### 2.3.1 Drivers moteurs - Cytron

Chaque moteur est contrôlé par un driver moteur Cytron MD10C, qui s'élèvent donc au nombre total de quatre pour le système.

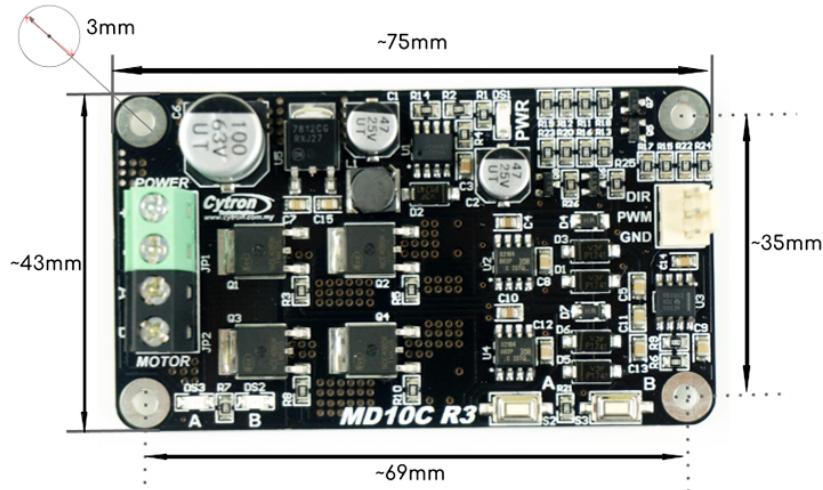


Figure 8: Drivers Moteurs Cytron MD10C r3

Il est possible d'alimenter les drivers moteurs avec une tension comprise entre 5 et 30 Volts. Il va délivrer 13A en continu et peut aller jusqu'à 30A en pique. La figure 9 montre comment câbler les drivers aux autres composants.

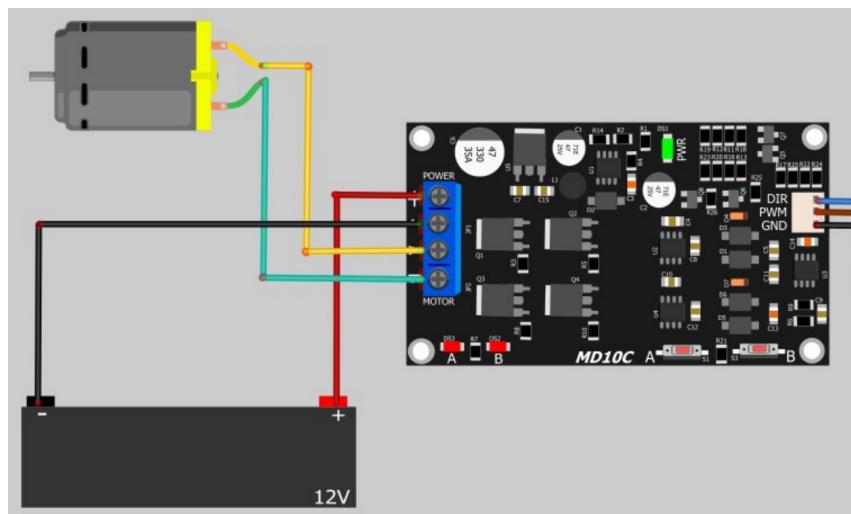


Figure 9: Drivers Moteurs - câblage

### 2.3.2 i2cPWM board - ServoShield

Les drivers moteurs sont contrôlés en par des valeurs de PWM envoyés et une direction (avant/arrière). Le signal DIR est contrôlé directement depuis la Raspberry Pi 4 (en utilisant des GPIO) et les valeurs de PWM sont fournies depuis un ServoShield.

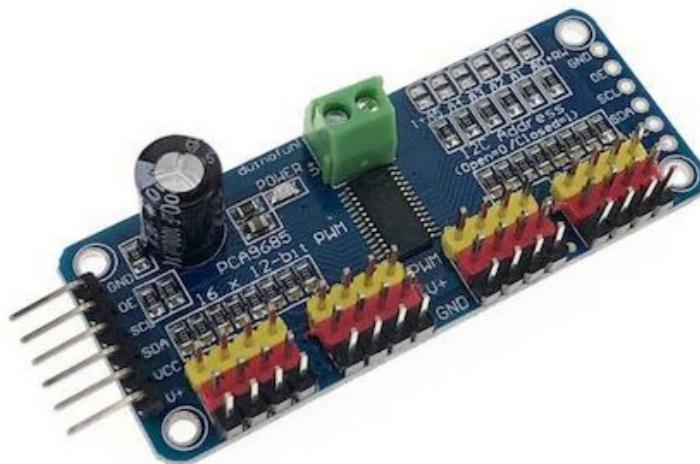


Figure 10: i2cPWM board - ServoShield

Cette carte possède un PCA9685 qui permet la gestion de 16 servos (ou leds). La Rpi4 communique avec cette board en I2C et va fournir des valeurs de PWM à une fréquence maximum de 1.5KHz.

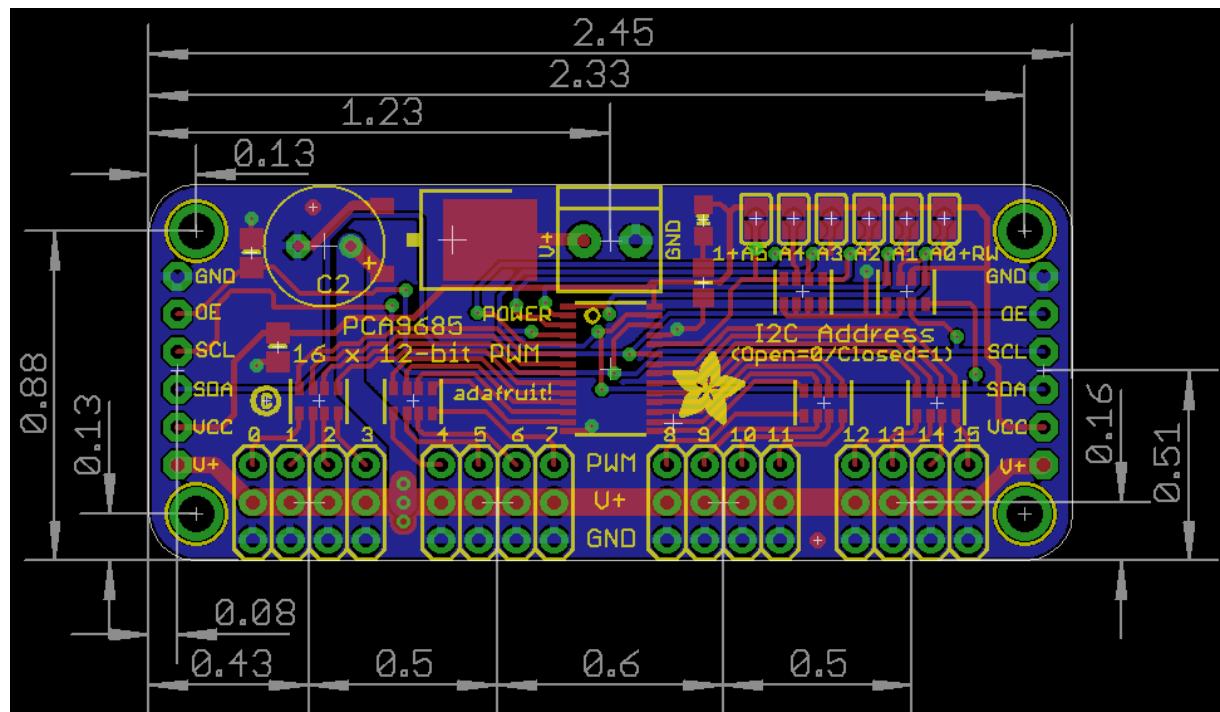


Figure 11: i2cPWM board - Dimensions

La Rpi4 set les registres des PWM avec différentes valeurs sur 12 bits (4096 valeurs). Ces valeurs permettent de contrôler la vitesse des moteurs sur 4096 variations.

### 2.3.3 REDS sensor board

Cette CPLD est une carte développée par le REDS. Celle-ci permet de communiquer avec la RPi4 en i2c. Son utilité première est de récupérer les informations des encodeurs et de fournir le nombre de ‘ticks’, qui représentent la position actuelle des moteurs. Cette information permet de définir la position du robot et sa vitesse à un temps T.

Pour de futurs besoins, il est tout à fait envisageable d’ajouter au module la possibilité de contrôler ou bien de récupérer des informations provenant d’autres capteurs.

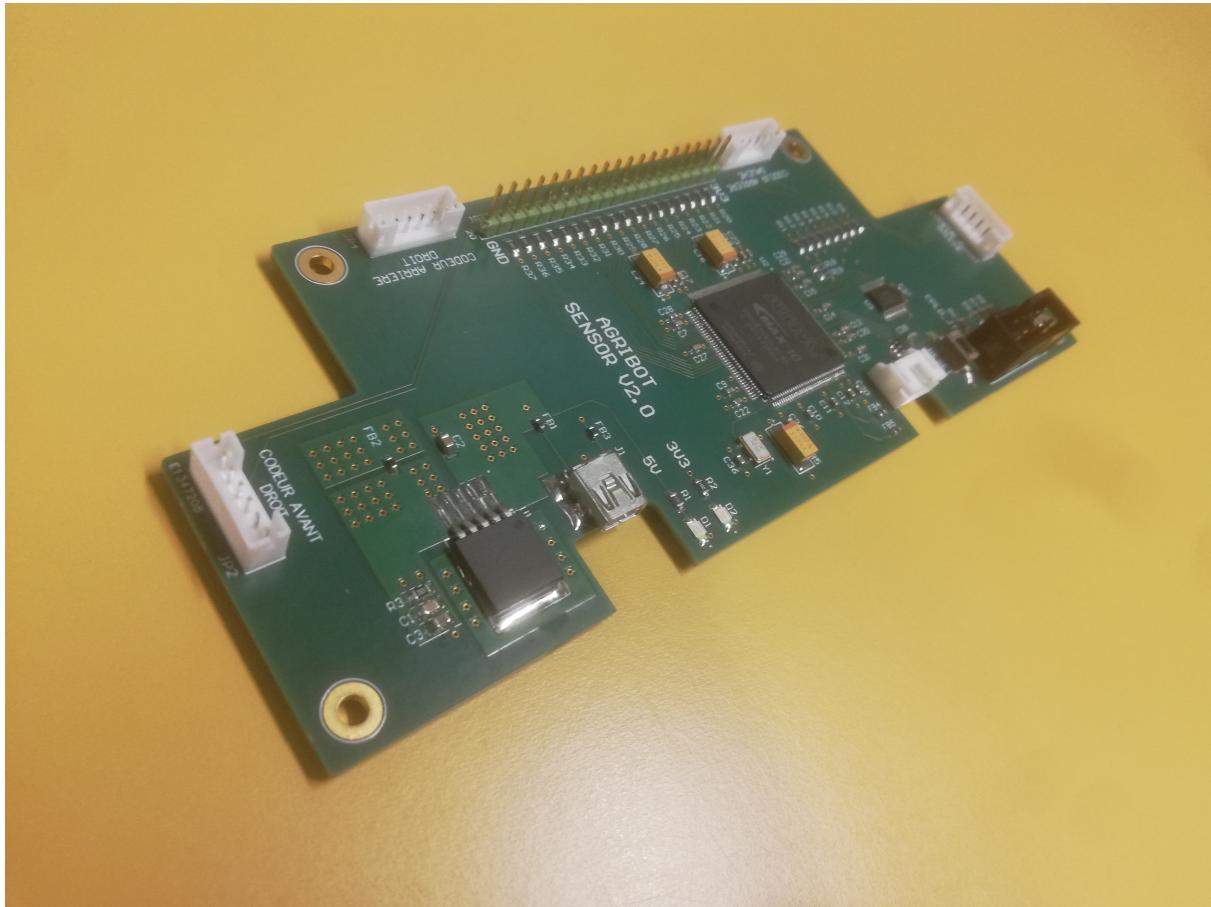


Figure 12: REDS sensor board

Cette board est alimentée en 5V, ce qui permet de l’alimenter avec la même source que la Raspberry Pi 4 sans avoir besoin de voltage converter.

### 2.3.4 Raspberry Pi 4

La Raspberry Pi 4 est la board choisie pour la gestion logicielle du système. En effet, la vaste communauté active et la réactivité des développeurs de la board pour des patchs éventuels permet d'assurer un confort de travail non négligeable.

La puissance de calcul n'est cependant pas suffisante pour supporter les demandes du système complet (traitement image inclus).

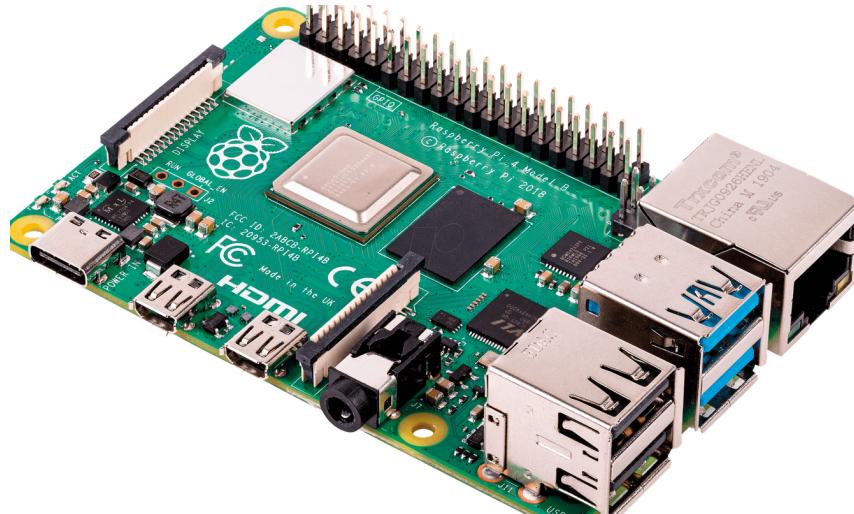


Figure 13: Raspberry Pi 4

Les dimensions de la board et les différentes options qu'elle propose la rendent aussi très flexible pour tester différentes approches et ainsi définir la solution la plus optimale d'après les besoins et contraintes.

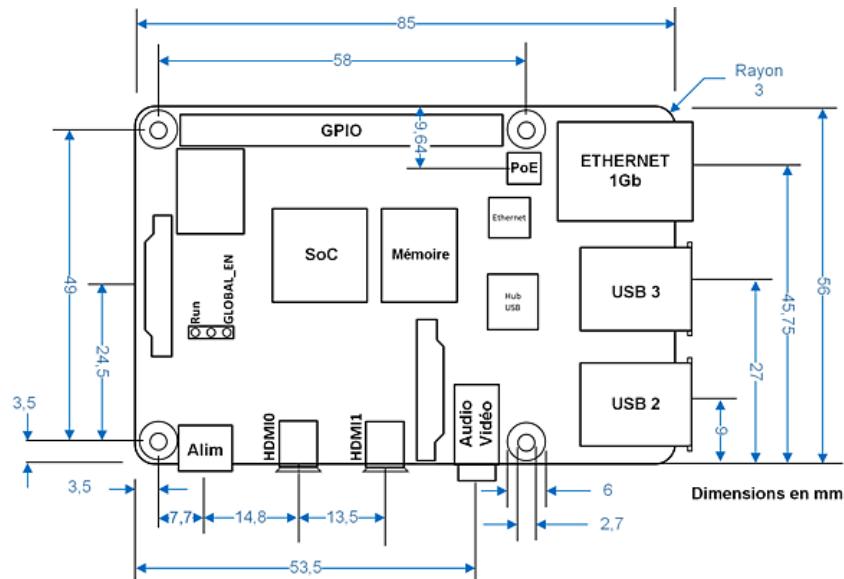


Figure 14: Raspberry Pi 4 – Mesures

## 2.4 Capteurs

### 2.4.1 Scanner laser 360° - RPLidar A1

Scanner rotatif low cost qui permet de scanner un environnement à 360°. Le système fonctionne à 5.5Hz et permet de scanner dans un rayon de 12 mètres. Il est possible de modifier la fréquence entre 2Hz et 10Hz en alternant le signal PWM du moteur.



Figure 15: RPLidar A1

Ce scanner est spécialement conçu et pensé pour la localisation des robots, ce qui en fait un composant de choix pour le système que nous développons. Ceci dit, pour améliorer la précision, il serait envisageable d'utiliser un modèle plus performant, une version plus récente.

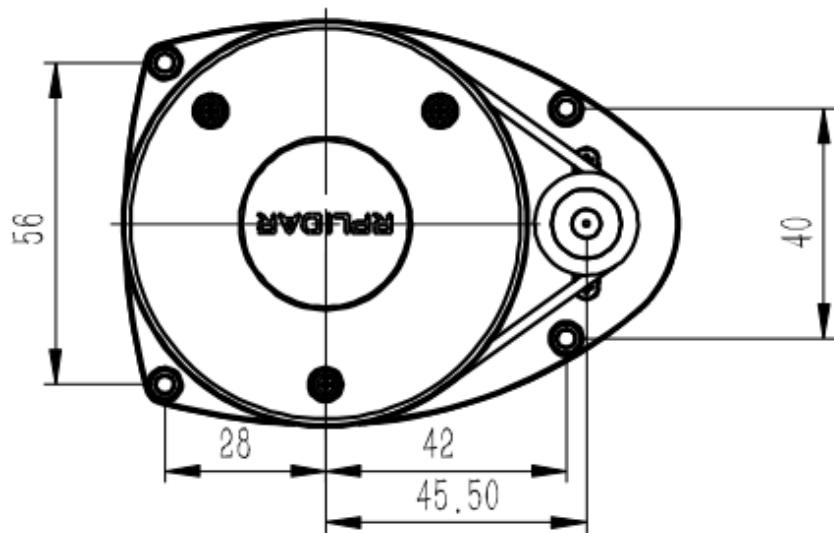


Figure 16: RPLidar A1 - Mesures

### 2.4.2 GNSS IMU - Xsens MTI 680G

Ce bijou de technologie provenant de Xsens est le point central qui permettra à Agribot de se localiser dans un environnement en extérieur. Equipé d'une centrale intertelle et d'un GNSS avec un récepteur RTK interne, la localisation en extérieur avec une précision au centimètre envisageable.



Figure 17: Xsens MTI 680G

Xsens propose ce genre de matériel spécialement pour ce genre d'application et propose une interface utilisateur pour rapidement configurer les données souhaitées en output. La taille et le poids en font un excellent choix pour nos besoins.

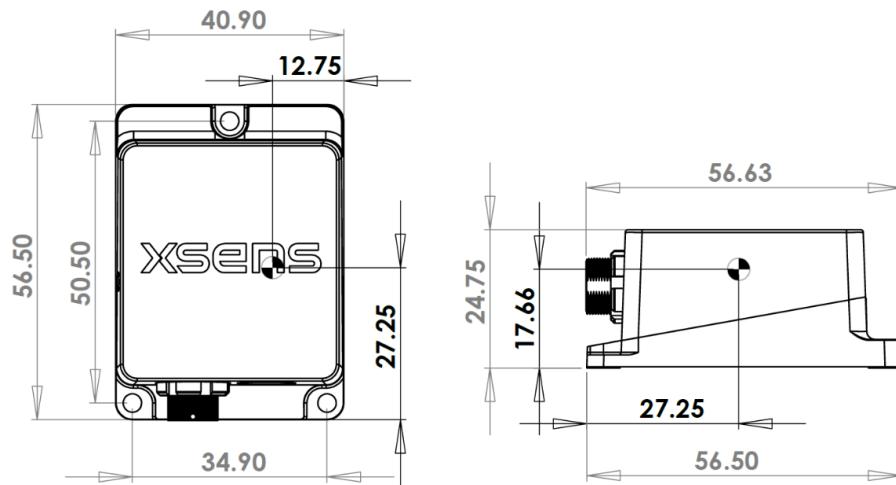


Figure 18: Xsens MTI 680G - Mesures

### 2.4.3 Camera - T265

Une caméra de RealSense qui permet d'obtenir des images RAW, mais aussi de générer une odométrie visuelle grâce à son IMU intégrée. Il est possible d'améliorer la précision de localisation, mais au prix d'un overhead en temps de calcul élevé.



Figure 19: RealSense T265

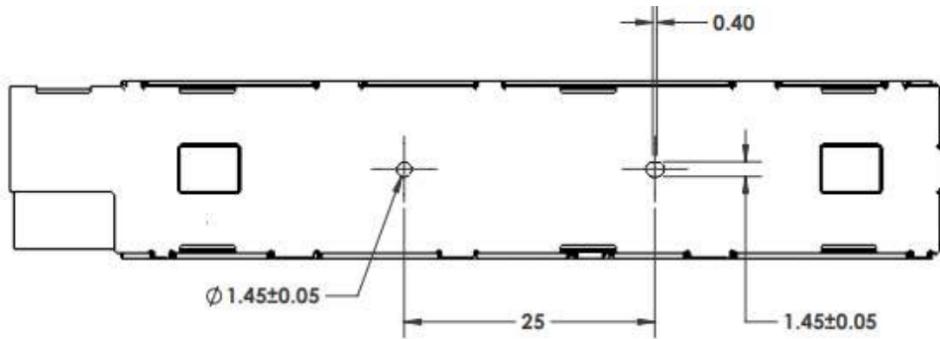


Figure 20: RealSense T265 – Mesures

### 2.4.4 Ecran tactile

Un écran tactile est disponible pour le debug et pour l'affichage de certaines informations. Il permet également d'initialiser les démos sans passer forcément par un device supplémentaire. Sa consommation étant élevée, son utilité n'est pas totalement justifiée dans le cadre d'un prototype à échelle réduite tel qu'il est actuellement.



Figure 21: Ecran tactile

## 3 Structure Agribot

Cette partie présente le robot tel qu'il a été monté et justifie les différents choix effectués pour obtenir le prototype.

### 3.1 Robot final

Voici le prototype utilisé pour développer le déplacement autonome en intérieur et en extérieur.



Figure 22: Agribot - robot

Le robot est entièrement monté et comporte tous les éléments précédemment présentés.

Le prototype est utilisé pour présenter le produit et différentes démonstrations sont possibles. L'objectif est de montrer ce dont le système est capable dans l'état actuel du projet.

### 3.2 Etage 1

Le premier étage comporte :

- 4 moteurs
- 4 drivers moteurs
- Batterie moteurs
- Carte i2c PWM

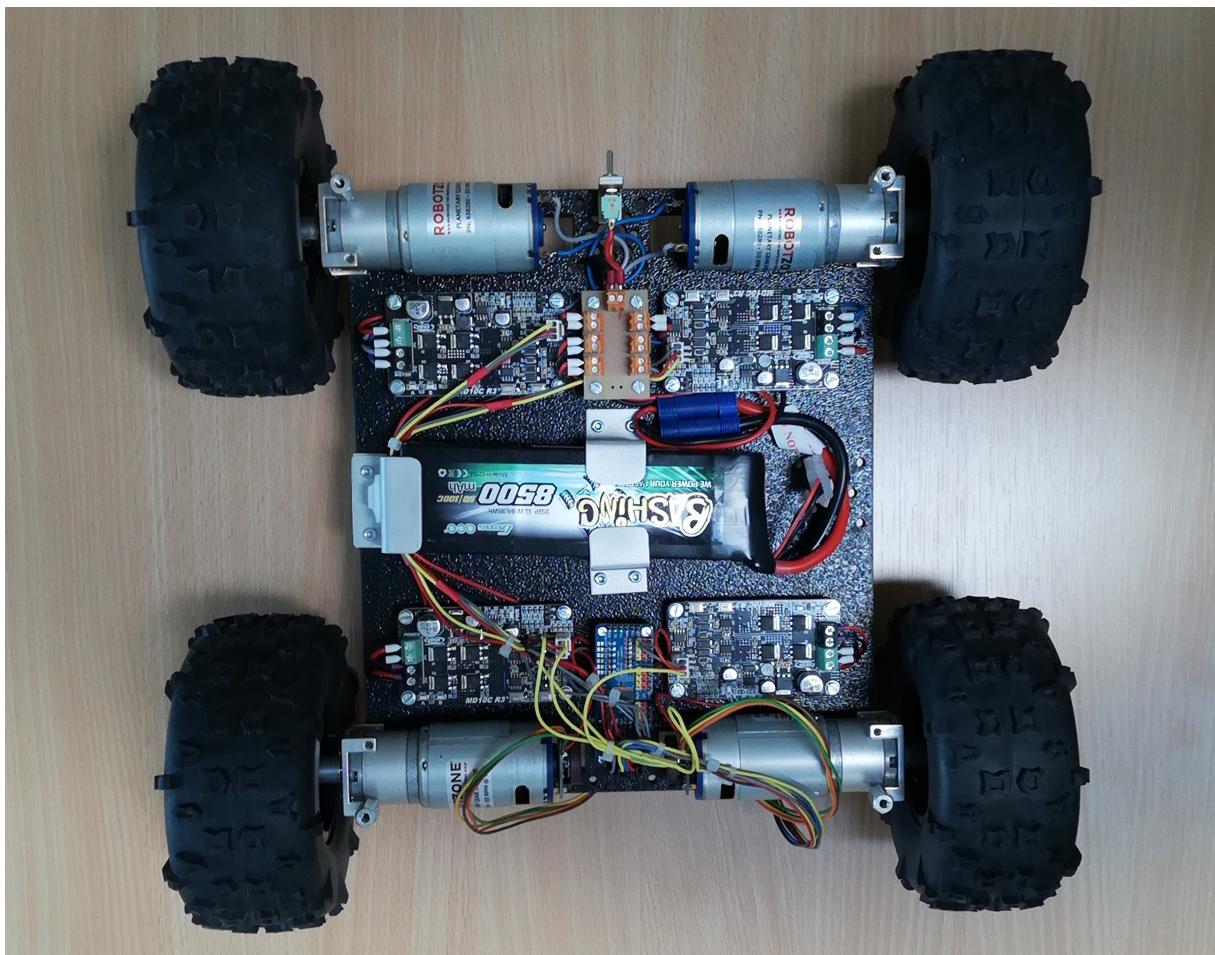


Figure 23: Agribot - Etage 1

Les drivers moteurs sont contrôlés par le module PCA9685 qui est communiqué avec la RPi4 en i2c. Ce module est placé à cet étage pour éviter un câblage long trop important entre le 1er et le 3ème étage.

Le câblage est simple et minimaliste. De plus la batterie est facilement rétractable, même lorsque le robot est entièrement monté.

### 3.3 Etage 2

Le deuxième étage comporte :

- Scanner rotatif - RpLidar A1
- REDS sensor board

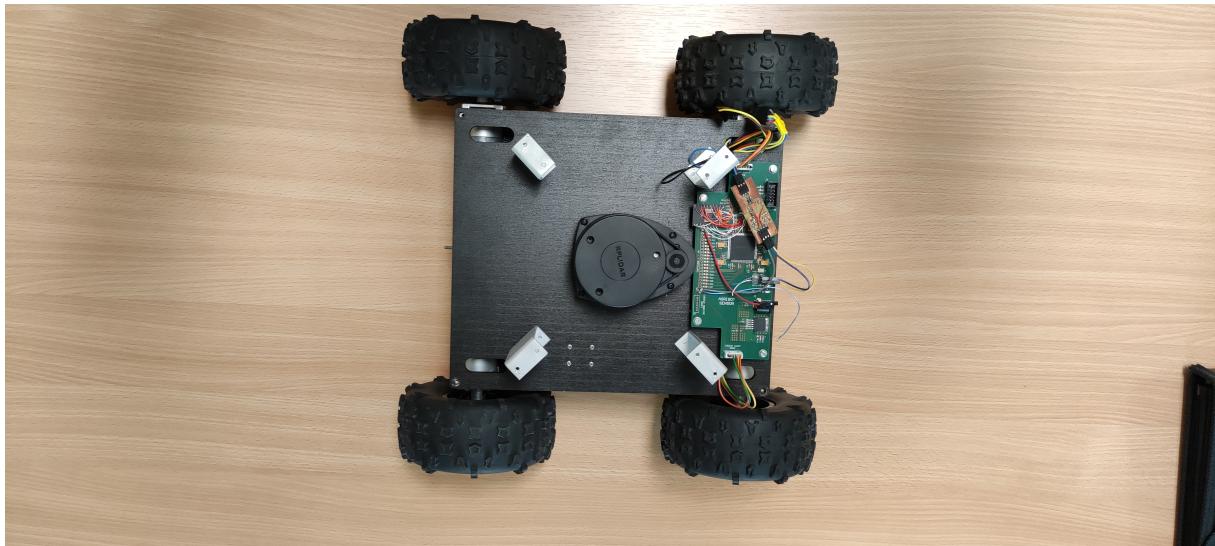


Figure 24: Agribot - Etage 2

La hauteur de la REDS sensor board permet de la placer sur cet étage, qui est particulièrement vide pour donner un rayon de visibilité maximale au RpLidar.

Il est important de noter que – toujours dans l’optique d’optimiser le champs de vision – les structures pour soutenir l’étage supérieur sont les plus fines possibles et le cablage qui transite par cet étage est adapté à l’épaisseur de ces éléments pour

Le scanner rotatif se situe à une hauteur estimée comme étant le meilleur compromis pour obtenir un maximum d’informations sur l’environnement avec une seule source. Il manque à ce jour des capteurs de distance à des hauteurs différentes pour de scanner l’environnement de manière plus précise et plus fiable et ainsi garantir le déplacement autonome. Il serait intéressant d’ajouter des capteurs pour assurer les obstacles minimums que le robot est capable de franchir ainsi que de garantir le scan d’obstacles à une certaines hauteur pour assurer qu’Agribot n’a pas une hauteur ne permettant pas de passer par certains endroits.

### 3.4 Etage 3

Le troisième étage comporte :

- Raspberry Pi 4
- Batterie système
- Caméra T265
- Xsens Mti 680G
- Ecran tactile

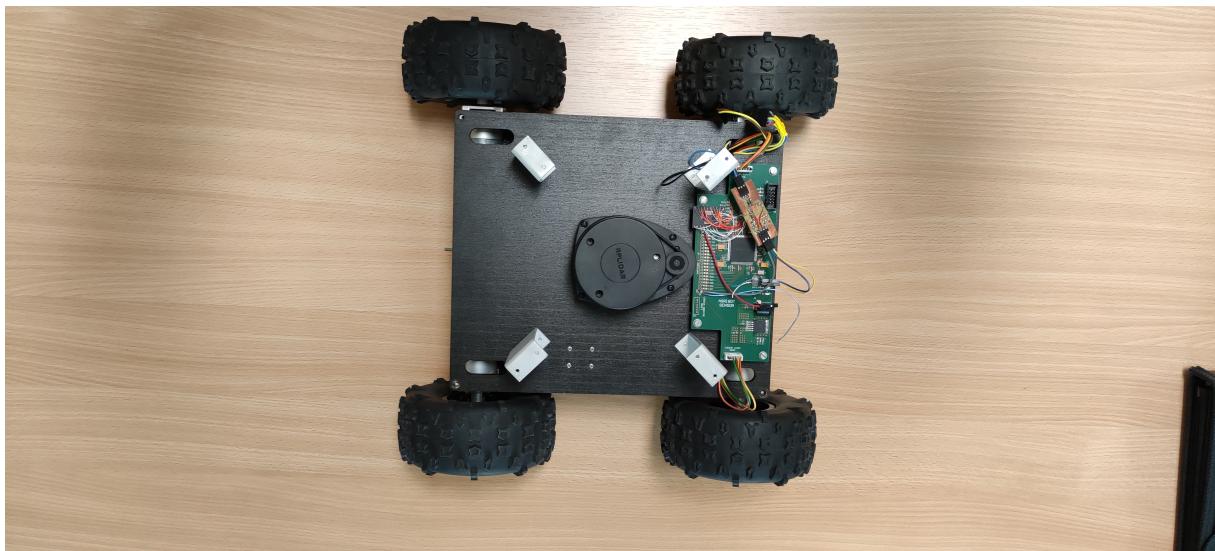


Figure 25: Agribot - Etage 3

Les drivers moteurs sont contrôlés par le module PCA9685 qui est communiqué avec la RPi4 en i2c. Ce module est placé à cet étage pour éviter un câblage long trop important entre le 1er et le 3ème étage.

Le câblage est simple et minimaliste. De plus la batterie est facilement rétractable, même lorsque le robot est entièrement monté.

### 3.5 Schéma bloc Agribot

La figure 26 montre schématiquement le câblage d'Agribot sous forme de schémas blocs reliés.

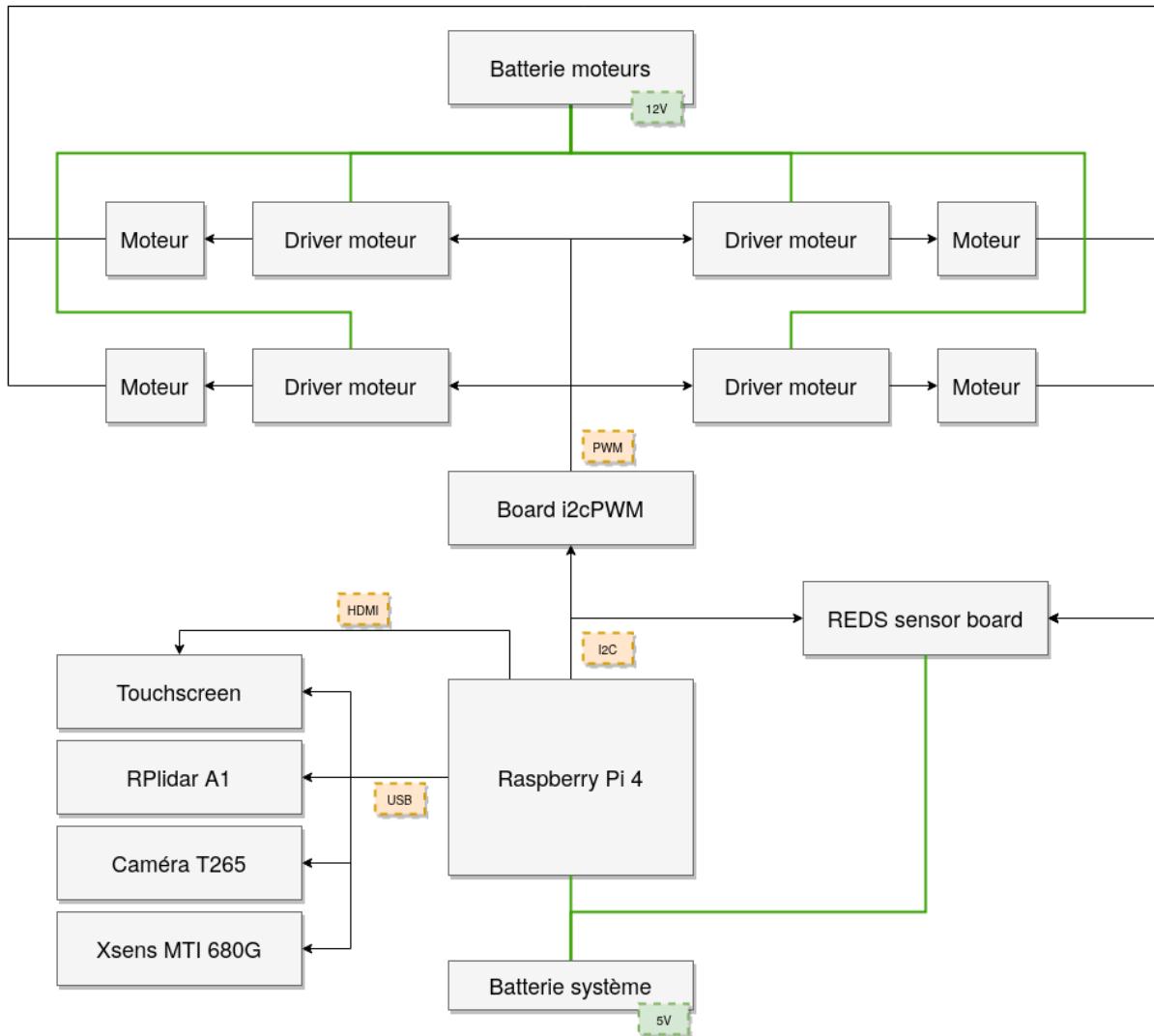


Figure 26: Schéma bloc Agribot

Nous pouvons voir sur ce schéma les différents éléments qui communiquent entre eux. L'objectif de cette image est d'avoir une vue globale sur le système et de mieux comprendre l'interaction des différents composants.

### 3.6 Schéma du câblage

## 4 Software

### 4.1 Environnement - ROS

#### 4.1.1 Qu'est ce que ROS ?

Comme son nom l'indique, ROS (Robot Operating System) est un système d'exploitation pour robots. De même que les systèmes d'exploitation pour PC, serveurs ou appareils autonomes, ROS est un système d'exploitation complet pour la robotique de service.

ROS est un méta système d'exploitation, quelque chose entre le système d'exploitation et le middleware.

Il fournit des services proches d'un système d'exploitation (abstraction du matériel, gestion de la concurrence, des processus...) mais aussi des fonctionnalités de haut niveau (appels asynchrones, appels synchrones, base de données centralisée de données, système de paramétrage du robot...).

#### 4.1.2 Pourquoi ROS ?

ROS possède de nombreux avantages, mais le plus important est la communauté grandissante et active qui ajoute et met à jour régulièrement du contenu. La communauté met à disposition de nombreux packages qui permettent d'ajouter en quelques manipulations de nombreuses fonctionnalités. Les entreprises qui développent certains modules destinés au monde de la robotique proposent de plus en plus les packages pour une utilisation dans ROS.

La possibilité de modifier le code source et/ou de créer soi-même un package 'from scratch' sans se préoccuper de toutes les problématiques de concurrence permet de développer des solutions plus rapidement pour du prototypage.

Enfin le système est assez commun. Toutes les interactions entre threads s'articulent autour d'un système de 'publisher - subscriber' que l'on retrouve dans de nombreux frameworks.

#### 4.1.3 Définitions

Package : C'est l'unité principale d'organisation logicielle de ROS. Un package est un répertoire qui contient les nœuds, les bibliothèques externes, des données, des fichiers de configuration et un fichier de configuration xml nommé manifest.xml.

Node : dans ROS, un nœud est une instance d'un exécutable. Un nœud peut correspondre à un capteur, un moteur, un algorithme de traitement, de surveillance.

Topic : Un topic est un système de transport de l'information basé sur le système de l'abonnement / publication. Des 'messages' (msg) sont échangés sur les topics pour transporter l'information.

Il y a bien d'autres concepts, mais ceux-ci sont les plus basiques et permettent de comprendre comment fonctionne globalement le framework et donc la structure logicielle d'Agribot.

## 4.2 Architecture software d'Agribot

### 4.2.1 Robot REDS

Robot REDS est un projet développé depuis Décembre 2020 qui propose une plateforme multi-robot. L'objectif est d'intégrer Agribot dans cet environnement.

Ainsi, il serait possible d'utiliser les différentes parties software avec les différentes bases mobiles développées ou récupérées, du moment que celles-ci possèdent les capteurs minimum requis pour que l'application soit en mesure de fonctionner.

### 4.2.2 Architecture software globale

La plateforme multi-robots robot REDS est organisé de sorte à abstraire la gestion de la base hardware du robot. L'organisation est la suivante :

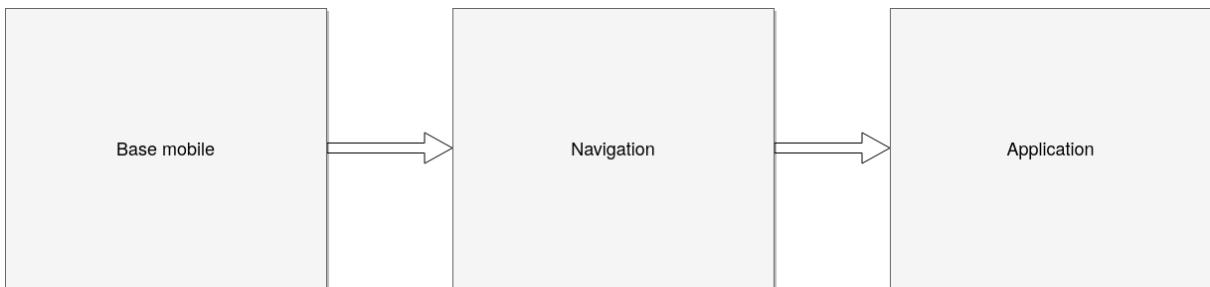


Figure 27: Blocs software

Voici une explication des différents blocs :

- Base mobile : c'est l'abstraction logicielle qui permet de recevoir en entrée une commande de vitesse de type ‘Twist’ (vitesse linéaire ( $x, y, z$ ) et vitesse angulaire (roll, pitch, yaw)). Cette commande est ensuite traitée puis les bonnes valeurs de courant sont envoyées aux moteurs. Cette abstraction permet de complètement séparer le logiciel de gestion du déplacement de l'application qui effectue les tâches.
- Navigation : en principe, la navigation est très liée à l'application. Cependant, ces deux parties sont tout de même séparées pour des questions pratiques de modifications. En effet c'est en théorie la seule partie qui aura besoin de modifications suivant les plateformes, les objectifs et les différents environnements.
- Application : ce bloc concerne les différentes tâches qui seront assurées par le système. Typiquement pour Agribot, cela concernera le placement du robot sur les lignes de plantation, la gestion de l'alimentation, etc...

### 4.2.3 Architecture base mobile

Le concept est qu'Agribot est à la base une application qu'il est possible d'exécuter sur une base mobile différente. Il est cependant nécessaire que la base assure le hardware minimum requis pour l'exécution de l'application.

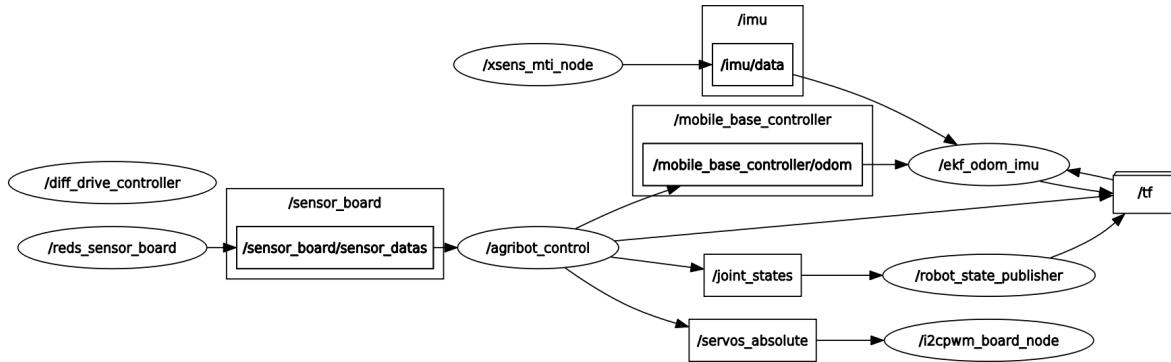


Figure 28: Noeuds de la base mobile

Cette figure présente ce que l'on retrouve dans le bloc 'base mobile' de la figure 27

Toutes les interactions de cette boîte noires sont les noeuds et les communications requises pour le bon fonctionnement de la base mobile. Concrètement, c'est cette partie qui transmets et formate comme il se doit les valeurs de vitesses angulaires/linéaires pour que les déplacements du robot correspondent aux commandes de déplacements

Ce bloc a besoin en comme informations d'entrée une commande de vitesse encapsulée dans un messages standard de ROS (Twist message) qui est formaté de la manière suivante :

#### Twist Message

- Vector3 linear
  - float64 x
  - float64 y
  - float64 z
- Vector3 angular
  - float64 x
  - float64 y
  - float64 z

Ros propose un système de controller et de hardware interface générique permettant d'encapsuler la partie des calculs suivant la façon de diriger le robot (différentielle 4 roues pour Agribot).

La figure 29 est un schéma bloc du fonctionnement des ‘ROS controller’.

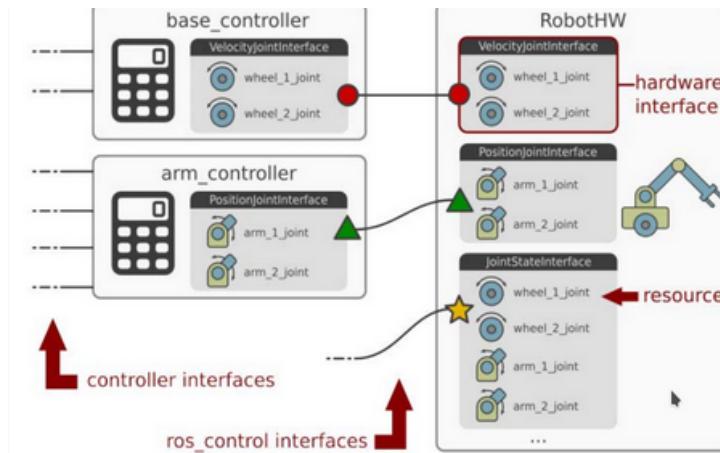


Figure 29: Schéma du fonctionnement des ROS controller

Les controllers et les Hardware interface communiquent mais fonctionnent de manière complètement indépendante pour que le système soit le plus générique possible. Il est ainsi possible d'avoir un controller commun pour des bases mobiles différentes. L'adaptation se fait donc notamment dans la partie Hardware Interface.

La figure 30 décrit le fonctionnement et l'intéraction entre les controllers et les hardware interface.

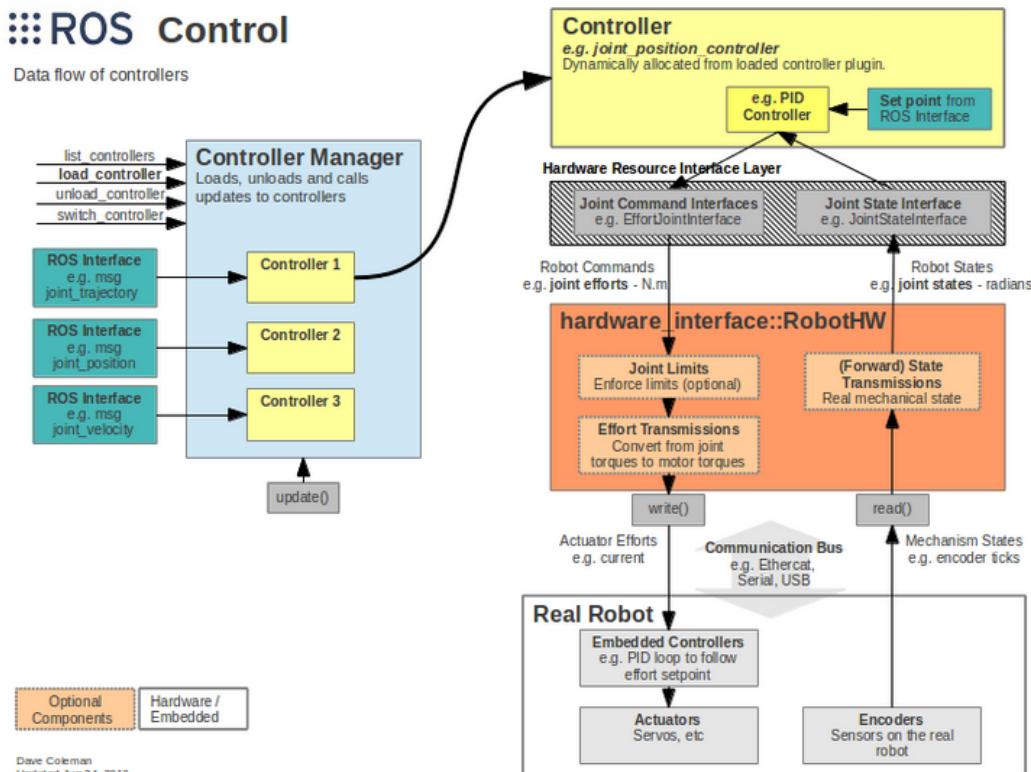


Figure 30: Schéma du fonctionnement des ROS controller

#### 4.2.4 Architecture de la navigation

Agribot fonctionne avec la navigation proposée par ROS. Il est possible de modifier chacun des packages pour adapter le système de navigation aux besoins du système et aux contraintes du projet.

La figure 31 montre les différents noeuds qui interagissent.

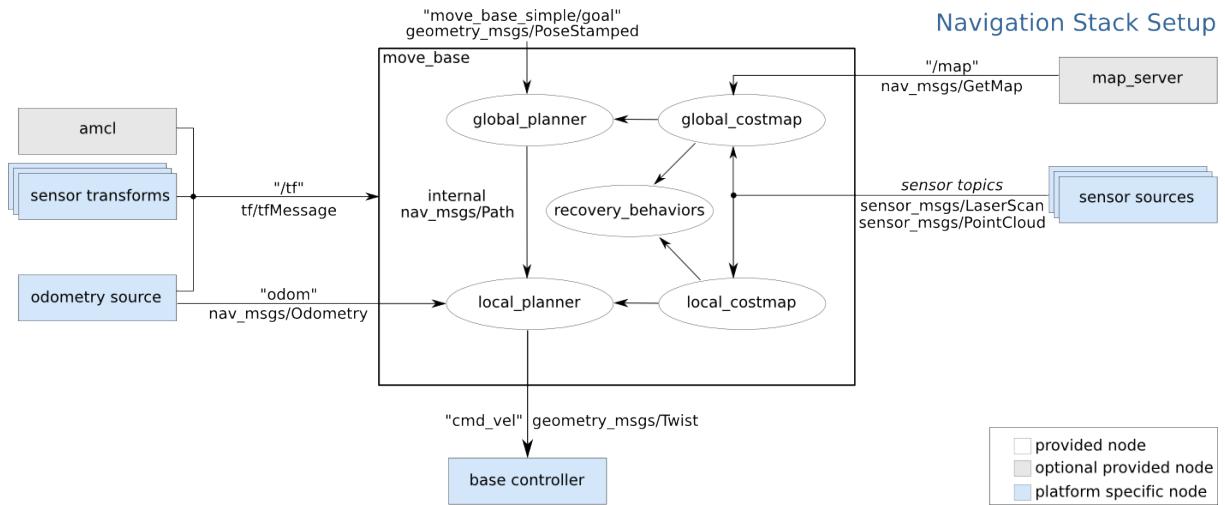


Figure 31: Schéma bloc de la navigation

- Global planner : lorsqu'un objectif est fourni (position dans la map), ce bloc utilise un des algorithmes disponibles définis dans les paramètres pour créer un chemin que le robot va suivre pour arriver à la position. Ce noeud prend en compte les obstacles connus et présents dans la map.
- Local planner : on peut considérer le local planner comme étant une correction au premier chemin calculé par le global planner. L'analyse des obstacles et/ou modification de l'environnement est prise en compte et permet de recalculer un chemin tandis que le robot se déplace vers son objectif.
- Costmaps : les global et local costmaps sont utilisées par les planners respectifs pour le calcul des chemins possibles. De nombreux paramètres permettent de gérer la taille, la précision etc... des costmaps, ce qui permet de trouver un équilibre entre temps de calcul et performances.
- Informations à fournir : il est indispensable d'avoir une odométrie en entrée. La précision de celle-ci facilitera les calculs et la localisation de tout le système. Les informations du lidar permettent de scanner l'environnement et de fournir en temps réel à une certaine fréquence les obstacles que le robot devra prendre en compte.

La mise en place de la navigation ne fait pas partie de l'application elle-même, mais n'est pas non plus spécifique à une base hardware. En effet, un même robot peut naviguer dans différents environnements, et cela implique une navigation différente et adaptée. Actuellement, Agribot possède un mode de navigation en intérieur et un mode de navigation en extérieur.

## Navigation en intérieur

La figure 32 permet d'identifier les adaptations effectuées pour la navigation en intérieur.

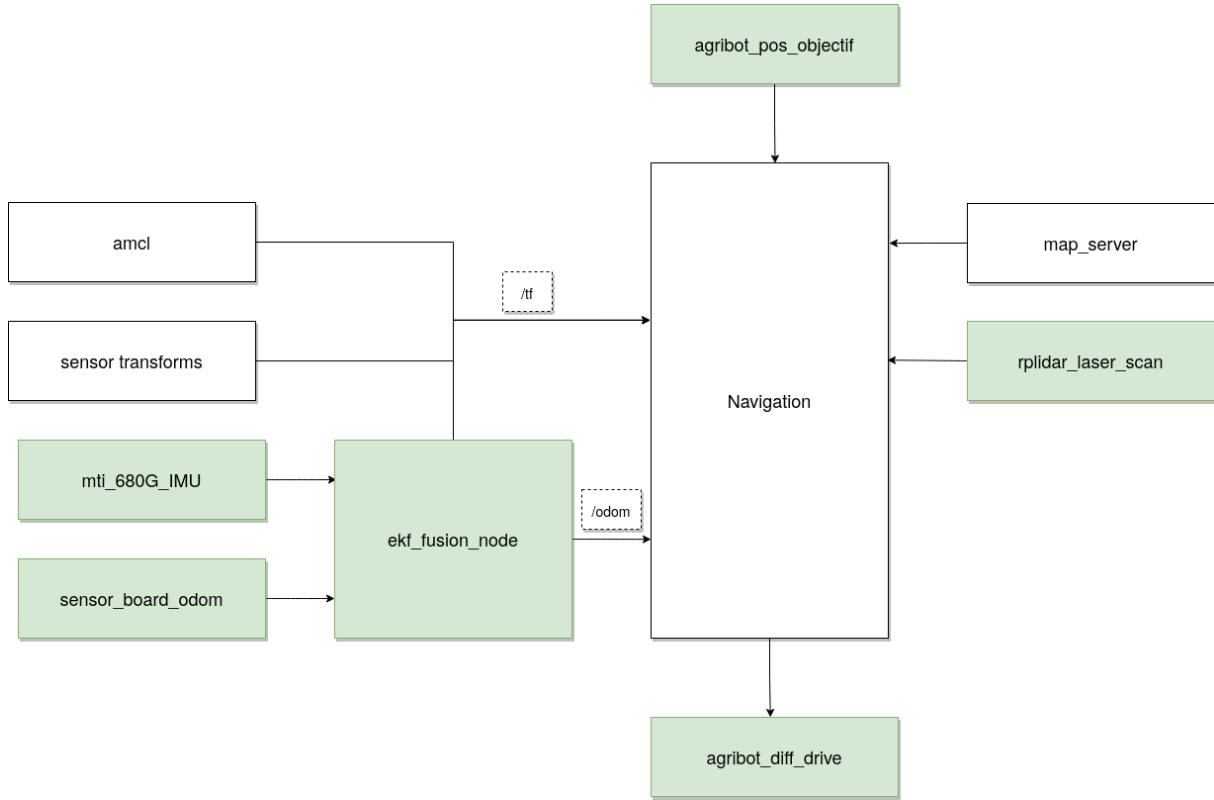


Figure 32: Modifications pour la navigation en intérieur

- Concernant la localisation, un noeud utilisant l'algorithme EKF (Extended Kalman Filter) permet de fusionner l'odométrie fournie par les encodeurs et les informations d'IMU provenant du MTI 680G pour fournir une odométrie plus précise (que celle uniquement obtenue avec les encodeurs) et les tf nécessaires pour le bon fonctionnement de la navigation.
- Le point d'entrée est un objectif traduit sous forme de position sur la carte provenant de l'application déterminant les tâches et donc les déplacements d'Agribot. La sortie est une commande de vitesse (Twist message) qui fournit une vitesse linéaire et une vitesse angulaire à la base mobile.
- Les données de scan proviennent du RPlidar et ces informations permettent à la fois de générer une carte (pour la réutiliser plus tard par exemple) et de mettre à jour une carte utilisée pour conserver les obstacles détectés lors d'un déplacement. Ce scanner est notamment utilisé pour la détection des obstacles dynamiques qui peuvent survenir lors d'un déplacement.

La figure 33 représente sous forme de schéma bloc la structure software et la communication entre les éléments.

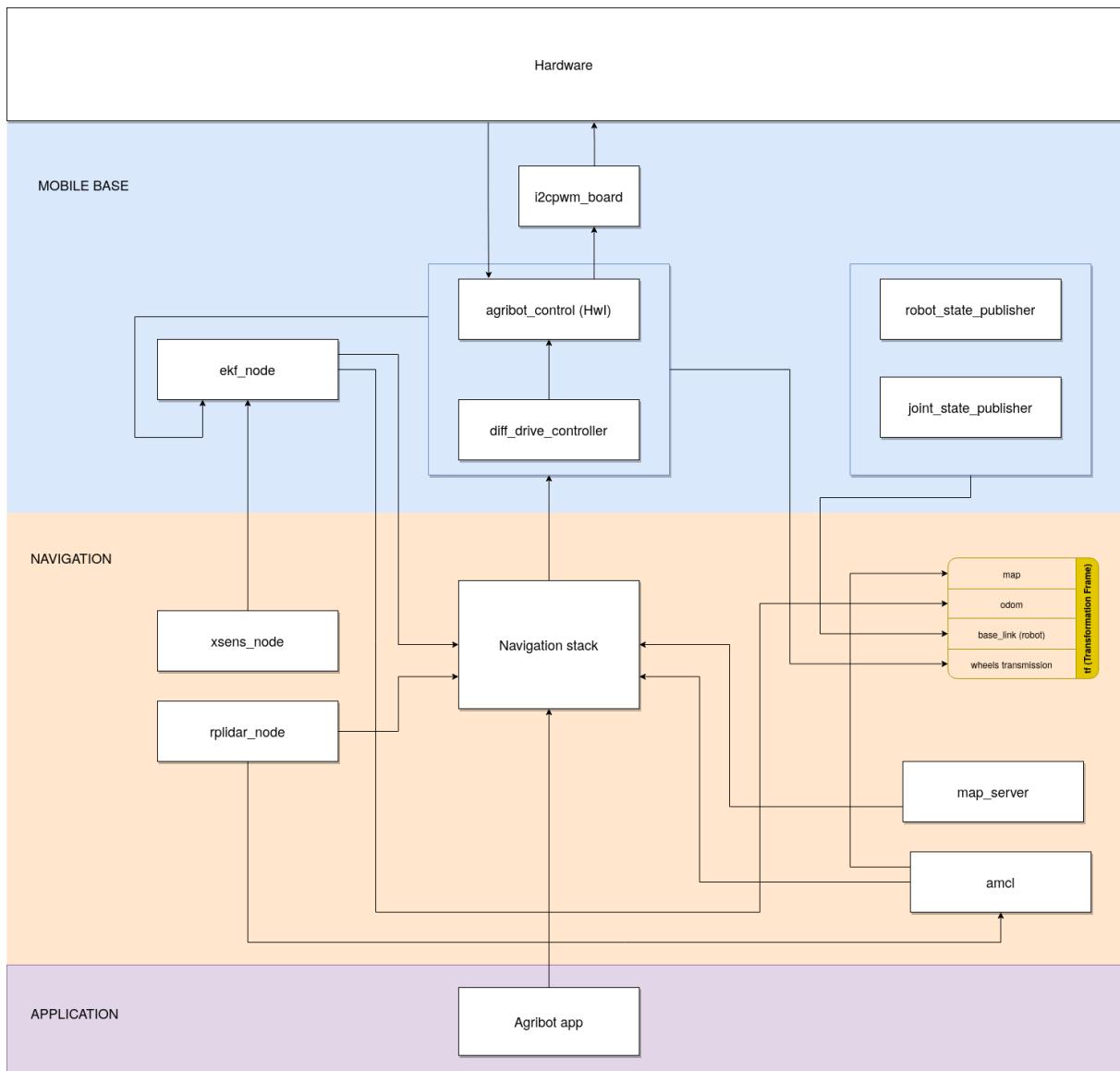


Figure 33: Structure software pour la navigation en intérieur

**Navigation en extérieur**[link](#)

```
$ sudo nmap -sP 10.192.91.0/24 | grep agribot
```

<https://www.generationrobots.com/blog/fr/ros-robot-operating-system-3/>