

## Travail de Bachelor

# Déplacement autonome d'un robot pour l'agriculture durable



Figure 1 - Bandes du champ d'Agiez

**Non confidentiel**

**Étudiant :**

**Eloïse Martin**

**Travail proposé par :**

Christian Bovigny

Ferme du Moulin BIO

Le Moulin 1

1372 Bavois

**Enseignant responsable :**

**Etienne Messerli**

**Année académique :**

2020-2021

Département TIC

Filière Informatique

Orientation Informatique embarquée

Étudiant Eloïse, Martin

Enseignant responsable Etienne, Messerli

## Déplacement autonome d'un robot pour l'agriculture durable

Entreprise : Ferme bio du Moulin (Bovigny Christian)

### Énoncé :

Nous souhaitons réaliser un robot d'arrosage de plantons dans le but de soutenir une agriculture écologique et durable. Durant les premières semaines, les plantons sont très sensibles à un manque d'eau par le faible volume de leurs racines concentré uniquement dans la motte de culture (4x4 cm). Ce stress peut engendrer des pertes de production assez conséquente. Le projet étant très conséquent, il a été identifié différentes problématiques à résoudre. Celui-ci a été scindé en plusieurs parties, soit: le déplacement autonome du robot, la détection des plantons et la conception mécanique. Nous souhaitons, dans le futur, pouvoir ajouter d'autres fonctionnalités comme le désherbage mécanique. La connaissance précise de l'emplacement des plantons permettrait de contrôler le passage d'une griffe entre ceux-ci sans nécessiter une détection d'image complexe avec la présence des mauvaises herbes. Il serait aussi possible de pourvoir planter les plantons ou de semer des graines.

Un projet précédent projet sur le déplacement autonome a permis la mise en œuvre de l'environnement ROS. Nous disposons ainsi d'un prototype qui se déplace seul en intérieur.

L'objectif de ce projet est de réaliser une commande embarquée permettant le déplacement autonome du robot dans un champ, soit en extérieur. Nous souhaitons pouvoir cartographier la position des plantons avec une grande précision, soit de l'ordre du centimètre. La commande doit permettre au robot de se placer au début d'une ligne de plantation, d'assurer l'arrosage de la ligne et d'aller à un point de remplissage ou de recharge. Il est imaginé que le système de commande sera basé sur une plateforme embarquée, qui dispose de système de géolocalisation, de moyen de communication et de possibilité de stockage de position ou de déplacements. Le projet comprendra le choix, l'adaptation et la mise en œuvre d'un prototype afin de réaliser des essais réels dans un champ.

Pour les aspects de géolocalisation et de positionnement nous allons collaborer avec le professeur Sébastien Guillaume de l'institut INSIT.

Mots clés : robotique, positionnement, vision, système embarqué, prog C, ...

## Cahier des charges :

Le but du projet est de réaliser une commande embarquée afin d'assurer le déplacement autonome d'un robot dans un champ. La solution sera validée à l'aide du prototype de robot développée au REDS. Nous pouvons identifier les étapes suivantes :

- ⑩ Prise en main du prototype de robot agricole (échelle réduite) et compréhension de l'environnement ROS utilisé pour le déplacement autonome en intérieur de celui-ci.
- ⑩ Investigation de diverses solutions possibles afin d'assurer une géolocalisation du robot dans un champ pour l'ensemble des déplacements autonomes. Proposition et choix d'un système de géolocalisation.
- ⑩ Analyse et choix d'une solution pour disposer d'une cartographie du champ avec les différentes zones de plantation des plantons, le point d'eau, la place de recharge et autres éléments nécessaires pour le fonctionnement du robot.
- ⑩ Développement et mise en œuvre sur le prototype de robot disponible au REDS de la solution de géolocalisation choisie précédemment.
- ⑩ Développement d'une cartographie d'un champ à échelle réduit pour le prototype.
- ⑩ Réalisation de test afin de valider les déplacements autonomes en extérieur. Qualification de la précision du système de géolocalisation avec l'utilisation d'un système de référence fourni par INSIT.

Liens: <https://bovigny.ch/ferme-bio-du-moulin/>

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

<https://wiki.ros.org/>

## Préambule

Ce travail de Bachelor (ci-après TB) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en **Ingénierie / Economie d'entreprise**.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'Ecole.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

HEIG-VD

Le Chef du Département

Yverdon-les-Bains, le 21 octobre 2021

# Authentification

La soussignée, Eloïse Martin, atteste par la présente avoir réalisé seule ce travail et n'avoir utilisé aucune autre source que celles expressément mentionnées.

Yverdon-les-Bains, le 21 octobre 2021

Eloïse Martin



**Confidentialité liée au Travail de Bachelor (TB)**

Diplômant : Martin Eloïse  
 Titre du travail de Bachelor : Déplacement autonome d'un robot pour l'agriculture durable  
 Enseignant responsable du TB : Messerli Etienne  
 Entreprise partenaire : Bovigny Christian (Ferme bio du Moulin)

Tous les TB sont déposés à la Bibliothèque de la HEIG-VD qui en gère l'archivage et la consultation. Quel que soit le niveau de confidentialité du TB, le nom du diplômant, le nom de l'enseignant responsable, le titre du TB et le résumé publiable figurent dans tous les documents de présentation des TB ainsi que sur la plateforme de consultation des TB (<http://tb.heig-vd.ch>). L'enseignant responsable veille à ce que le titre du TB et le résumé publiable soient rédigés conformément au niveau de confidentialité voulu.  
 Les TB peuvent être soumis à un logiciel anti-plagiat. Dans ce cas, leur contenu sera traité de manière confidentielle.

**Le TB n'est pas confidentiel**

*Outre les informations mentionnées ci-dessus, les documents de présentation du TB contiennent également le nom de l'entreprise partenaire, le résumé publiable et une affiche. Le TB peut être consulté sur la plateforme des TB.*

**Le TB est confidentiel.**

Les conditions suivantes de diffusion des informations sont appliquées :

*Aucune consultation ou emprunt du TB n'est permis hormis par l'enseignant responsable du TB et le diplômant qui s'engagent à ne pas faire usage des informations mises à leur disposition. Le TB porte la mention « confidentiel ».*

Oui  Non *Nous acceptons que le nom de l'entreprise partenaire figure dans les documents publiés (titre, résumé, affiche, etc.), ainsi que dans la plateforme de consultation des TB.*

Oui  Non *Nous acceptons que l'affiche du TB figure sur la plateforme de consultation des TB (l'affiche est au préalable validée par l'entreprise partenaire).*

Dans tous les cas, un accord de confidentialité doit être signé par le diplômant, l'expert et toutes les personnes participant à l'évaluation du TB.

**Nous déclarons accepter les conditions de diffusion du Travail de Bachelor indiquées.**

**Diplômant**

Date

23.07.2021

Nom et signature

Martin Eloïse

**Enseignant responsable**

Date

23 juillet 2021

Nom et signature

Messerli Etienne

**Ferme bio du Moulin**

Date

31.7.21

Nom et signature

Bovigny Christian

N.B.:

*Ce document fait partie intégrante du cahier des charges du TB.*

*La forme masculine est utilisée comme genre neutre et désigne à la fois les hommes et les femmes.*

## Table des matières

<b>1</b>	<b><i>Introduction</i></b>	<b>8</b>
1.1	Contexte	8
1.2	Objectifs	9
1.3	État actuel du projet	10
1.4	Structure et contenu du rapport	10
<b>2</b>	<b><i>Agribot</i></b>	<b>11</b>
2.1	Matériel	11
2.2	Fonctionnalités existantes	12
<b>3</b>	<b><i>ROS</i></b>	<b>13</b>
3.1	Navigation stack	15
<b>4</b>	<b><i>Géolocalisation</i></b>	<b>16</b>
4.1	Contraintes	16
4.2	Analyse des méthodes de positionnement	16
4.2.1	Photogrammétrie	17
4.2.2	Lasergrammétrie	17
4.2.3	GPS – Système GNSS	19
4.3	Solution existante	22
4.4	Choix du système de géolocalisation	22
<b>5</b>	<b><i>Cartographie</i></b>	<b>25</b>
5.1	Contraintes	25
5.2	Analyse des méthodes de cartographie	25
5.2.1	Cartographie dans ROS	26
5.2.2	Stockage des coordonnées	26
<b>6</b>	<b><i>Conception et Implémentation</i></b>	<b>28</b>
6.1	Navigation avec géolocalisation	28
6.1.1	Conception	28
6.1.2	Implémentation	31
6.1.3	Montage sur le rob	35
6.1.4	Station de base	38
6.1.5	Communication PC – Agribot	38
6.1.6	Packages utilisés	39
<b>7</b>	<b><i>Tests</i></b>	<b>40</b>
7.1	Tests du module Xsens dans MT Manager	40
7.2	Tests du système GNSS	41
7.3	Tests du système GNSS avec correction	41
7.3.1	Conversion WGS84 – MN95	42
7.4	Tests de l'intégration du système GNSS à la navigation	43
7.5	Tests d'envoi d'un but GNSS	43

7.6	Positions absolues de références .....	43
7.7	Tests restants.....	44
<b>8</b>	<b><i>Planification</i></b> .....	<b>45</b>
8.1	Planification initiale .....	45
8.2	Tâches réalisées .....	45
8.3	Problèmes rencontrés et imprévus .....	45
<b>9</b>	<b><i>Conclusion</i></b> .....	<b>47</b>
9.1	Synthèse.....	47
9.2	Améliorations .....	47
9.3	Conclusion personnelle .....	47
<b>10</b>	<b><i>Glossaire</i></b> .....	<b>48</b>
<b>11</b>	<b><i>Bibliographie</i></b> .....	<b>49</b>
<b>12</b>	<b><i>Table des figures</i></b> .....	<b>50</b>
<b>13</b>	<b><i>Annexes</i></b> .....	<b>0</b>

# 1 Introduction

## 1.1 Contexte

La ferme du Moulin BIO à Bavois est une exploitation maraîchère. Elle a de fort besoin en eau lorsque les jeunes plantons viennent d'être mis en terre. Elle s'approvisionne en eau dans le canal collecteur souterrain le plus proche de leurs champs. Ceci implique qu'elle peut facilement se retrouver avec un volume d'eau réduit pour l'arrosage. Dans ce but, elle souhaite obtenir une solution d'arrosage optimisée pour les jeunes plantons. Une solution moins encombrante que les systèmes d'irrigation goutte à goutte existants.

Ci-dessous, le haut du canal collecteur avec la pompe utilisée pour récupérer l'eau :



Figure 2- Pompe à eau du champ d'Agiez

La solution proposée à la ferme du Moulin est d'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 de manière optimale et économique. L'arrosage des jeunes plantons est une tâche importante pour la croissance, sachant que les plantons 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.

Ci-dessous un exemple de bande avec les dimensions ainsi que le pattern des plantons :

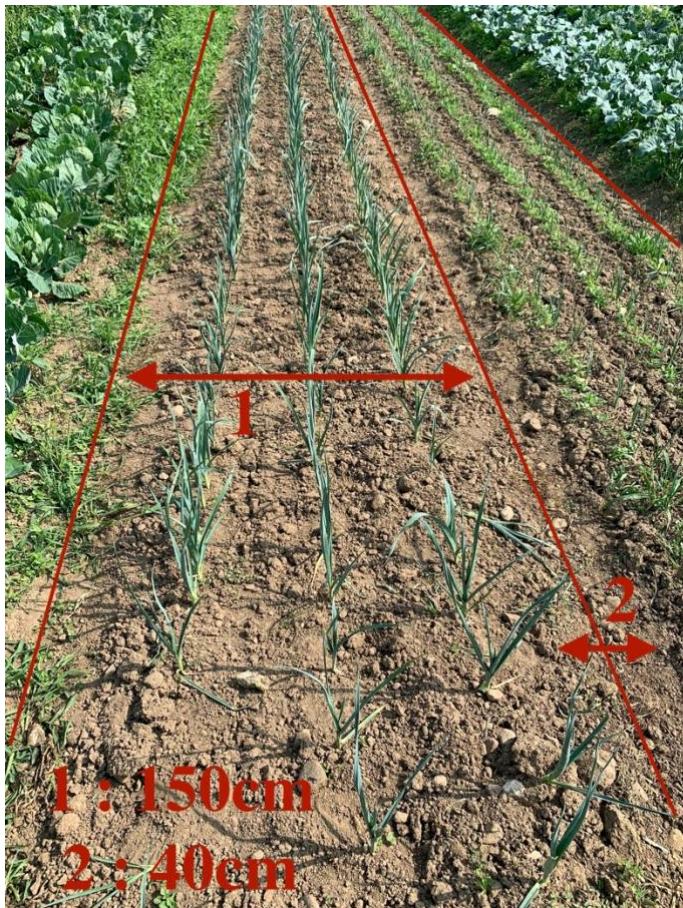


Figure 3 - Exemple de bande d'oignons avec distances



Figure 4 - Exemple de positionnement de plantons dans une bande

Plusieurs travaux ont déjà été réalisé dans ce sens. Chacun de ces travaux a permis de se rapprocher d'un système capable de réaliser cette tâche. Il a fallu permettre la reconnaissance de plantons par imagerie, la réalisation du bras d'arrosage et le déplacement autonome du robot en intérieur. La prochaine étape est le déplacement autonome du robot en extérieur avec la géolocalisation. Le but est de pouvoir fournir des coordonnées de positionnement 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.

## 1.2 Objectifs

L'objectif de ce travail est d'ajouter le déplacement autonome en extérieur au robot Agribot, avec une précision au centimètre. Ceci implique la géolocalisation et la cartographie. Le robot doit pouvoir se géolocaliser dans un champ. Il doit être capable de se déplacer vers un point donné, ce point doit correspondre à une coordonnée de type [longitude, latitude]. Le robot doit également permettre l'enregistrement de points et la lecture de ces points sous forme de carte.

Le travail comprendra une analyse sur la géolocalisation et une analyse sur la cartographie. L'analyse sur la géolocalisation présentera différentes manières d'obtenir une géolocalisation. Un choix final proposera la solution à implémenter. L'analyse sur la cartographie présentera différentes méthodes de cartographie. Un choix final proposera la solution à implémenter.

Une démonstration du déplacement autonome en extérieur terminera ce travail.

### 1.3 État actuel du projet

Le projet initial ayant débuté en 2018, le robot actuel est capable de se déplacer de manière autonome en intérieur, de cartographier son environnement, d'éviter les obstacles qu'il rencontre grâce au lidar. [Un chapitre](#) dédié décrit le robot et ses fonctionnalités avant le début de ce travail de Bachelor.

### 1.4 Structure et contenu du rapport

Ce rapport est structuré en plusieurs chapitres, ils se composent comme suit :

**Chapitre 2 : Agribot** : Ce chapitre décrit l'état actuel du projet, avec un rappel matériel ainsi que les fonctionnalités du robot.

**Chapitre 3 : ROS** : Ce chapitre présente le framework utilisé pour ce projet, ses avantages et désavantages, ainsi que quelques notions importantes à la navigation.

**Chapitre 4 : Géolocalisation** : Ce chapitre décrit les méthodes de positionnement analysées. Il s'agit ici d'étudier les différentes problématiques liées aux demandes de ce projet et de proposer une solution.

**Chapitre 5 : Cartographie** : Ce chapitre décrit les méthodes de cartographie analysées. Il s'agit ici d'étudier les différentes solutions existantes et de proposer celle qui s'intègre le mieux à l'environnement actuel.

**Chapitre 6 : Conception et implémentation** : Ce chapitre décrit la conception et l'implémentation de la navigation avec positionnement ainsi que la conception et l'implémentation de la cartographie.

**Chapitre 7 : Tests & Validation** : Ce chapitre décrit les tests effectués et la validation des résultats obtenus.

**Chapitre 8 : Planification** : Ce chapitre décrit la planification initiale, les tâches réalisées ainsi que les problèmes rencontrés lors la réalisation du projet

**Chapitre 9 : Conclusion** : Ce chapitre fera la synthèse du travail effectué et proposera quelques améliorations.

## 2 Agribot

Ce chapitre décrit l'état actuel du projet de robot autonome pour une agriculture durable. Le premier nom donné au robot est « WateringBot ». Aujourd'hui, « Agribot » est le nom donné au robot du projet d'arrosage automatique optimisé. C'est le nom qui sera utilisé dans le reste du rapport.

Le robot de gauche correspond au prototype réalisé en 2019 et à droite celui réalisé en 2020 :

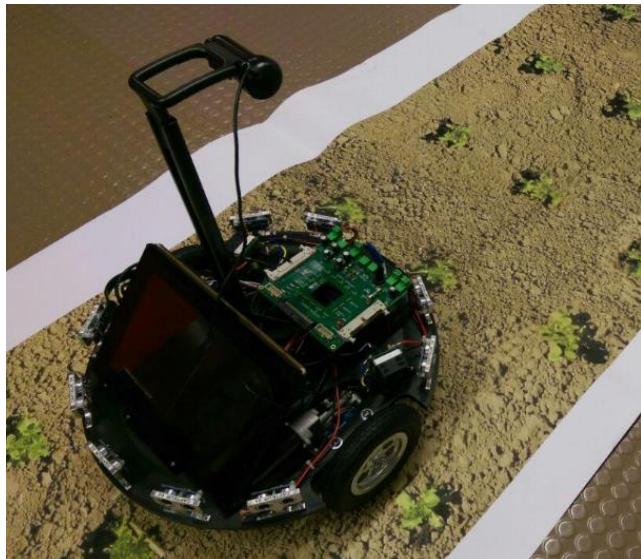


Figure 5 - Prototype du robot - 2019

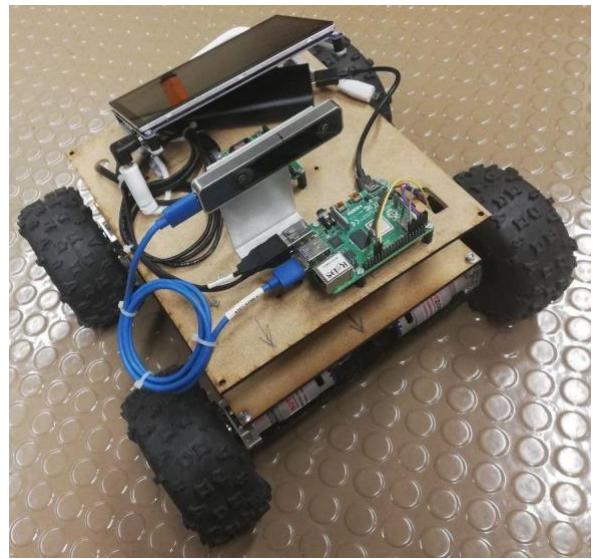


Figure 6 - Prototype du robot - 2020

En 2019, on observe un robot simple, rond sur un étage. C'était le prototype de robot pour la détection de planton par image. En 2020, on observe grande évolution avec l'ajout de 4 roues indépendantes et bien d'autre encore. Le sous chapitre suivant décrit le robot finalisé en 2020.

### 2.1 Matériel

Ce chapitre décrit la composition matérielle du robot en mars 2021. Le robot est composé des éléments suivants :

- 1<sup>er</sup> niveau :
  - o 4 roues de x de diamètre
  - o 4 servo moteurs
  - o 4 drivers moteurs
  - o 1 carte PWM
  - o 1 batterie

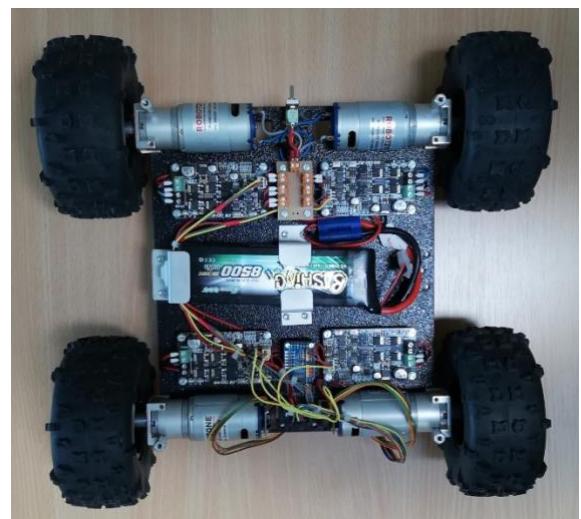


Figure 7 - 1<sup>er</sup> étage du robot

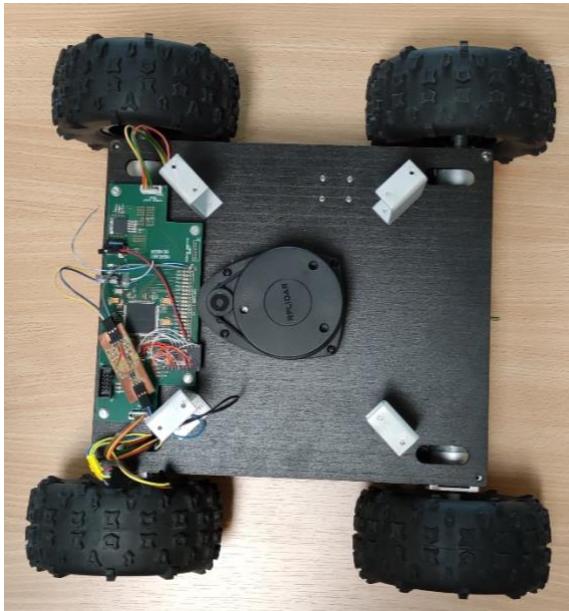


Figure 8 - 2ème étage du robot

- 2<sup>ème</sup> niveau :
  - o 1 sensorboard REDS
  - o 1 RPLidar

Le Lidar est positionné légèrement en dessus de la sensorboard ce qui fait que la détection se fait uniquement pour les éléments environnants se trouvant au moins à 15cm plus haut que le sol. Les éléments de supports pour l'étage du dessus ont été disposé afin d'être le plus discret pour la détection du Lidar.

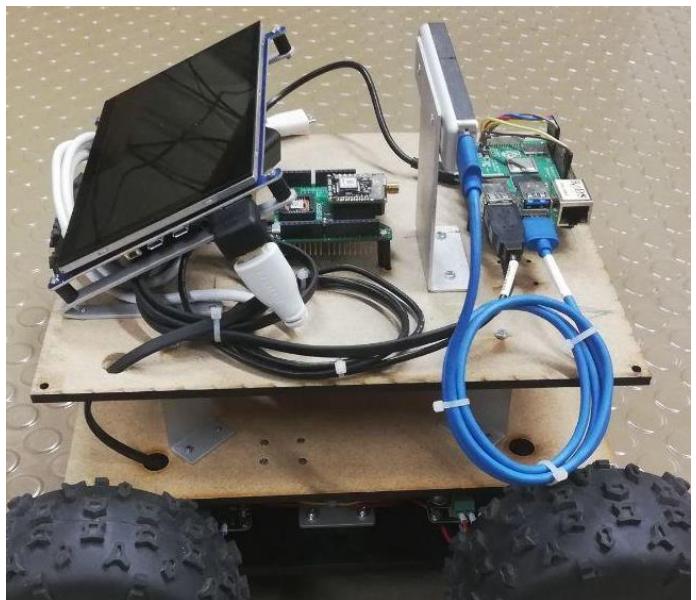


Figure 9 - 3ème étage du robot

## 2.2 Fonctionnalités existantes

Les fonctionnalités diverses du robot ont été séparée en plusieurs parties. Ces parties sont les commandes embarquées et la partie de mécanique. Grâce aux travaux de Bachelor réalisés précédemment, le robot Agribot possède déjà les fonctionnalités et éléments suivants en version prototype :

- Un prototype de reconnaissance de plantons
- Une navigation autonome en intérieur
- Un système d'arrosage avec un bras motorisé
- Un bras motorisé pour l'arrosage
- Un chassi sur mesure

### 3 ROS

Dans ce chapitre, une présentation de l'environnement ROS va être présenté afin de mieux comprendre le chapitre sur l'intégration.

ROS, Robot Operating System, est une plateforme de développement open source. Elle permet de faciliter la réalisation de robot en offrant un cadre applicatif intéressant et simplifié. Ce système permet d'obtenir une abstraction matérielle. Elle offre un contrôle des périphériques de bas niveaux. Elle permet de communiquer facilement entre les processus avec la transmission de messages. Elle jouit d'une grande communauté. Comme la plateforme est open source, elle offre de nombreux modules réalisés par la communauté et bien plus encore. Elle a également l'avantage d'être supporté par plusieurs systèmes d'exploitation, comme Linux ou Mac OS.

Deux langages de programmation sont principalement utilisés pour le développement dans ROS : le C++ (roscpp) et le Python (rospy).

L'environnement met également à disposition des outils de simulation comme RVIZ ou Gazebo. RVIZ est un outil de simulation qui permet de visualiser en temps réel le robot dans l'espace ou sur une carte. Gazebo est un outil de simulation qui permet de visualiser et de tester rapidement les fonctionnalités du robot.

Ci-dessous, un schéma explicatif des notions de base de ROS :

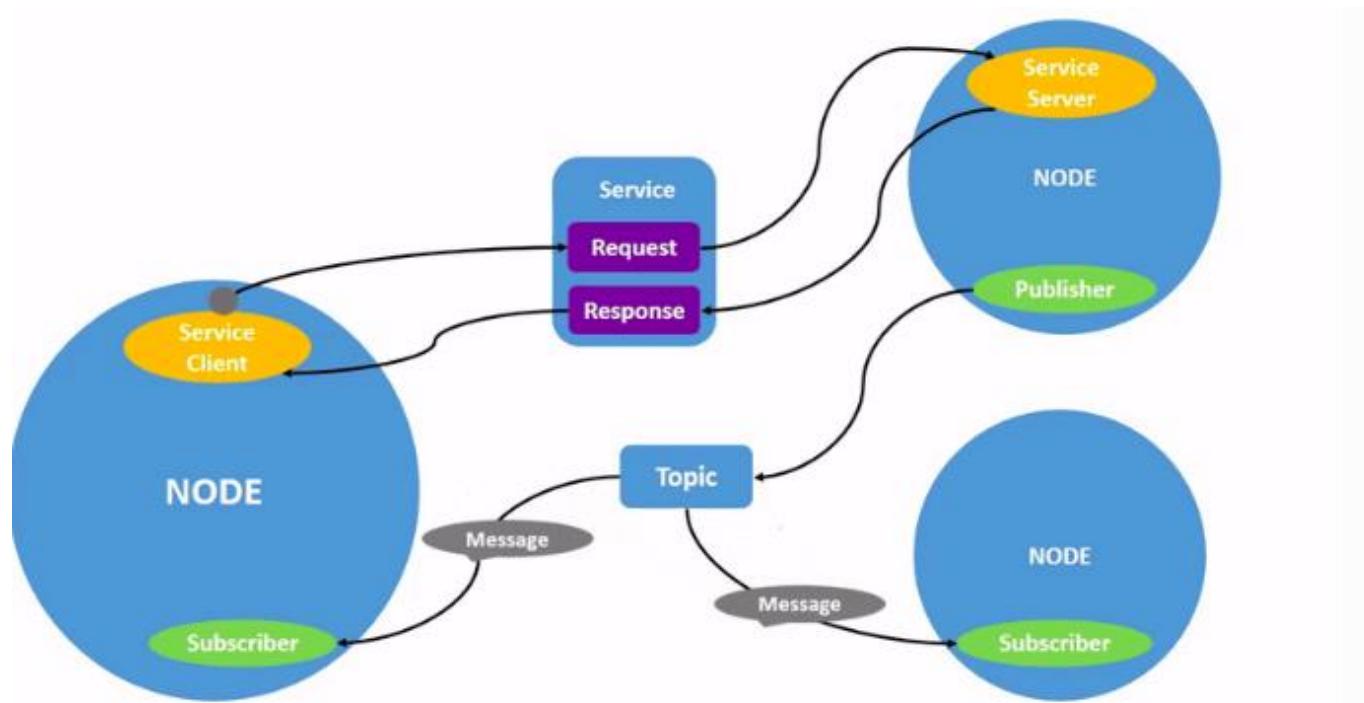


Figure 10 - Schéma explicatif du fonctionnement des nodes, des topics, des messages et des services

Sur l'image ci-dessus, on observe plusieurs objets : des nodes, des topics, des services et des messages. Les nodes peuvent être considérés comme des processus. Un node peut être une fonctionnalité ou une représentation d'objet matériel. Les topics sont utilisés pour la communication en continu, un node peut s'abonner à un topic afin de recevoir le flux de messages publiés par un node. Les services permettent d'obtenir des informations à la demande aux autres nodes avec un système de requête-réponse, de type client- server.

RQT\_Graph est un outil de visualisation de nodes. Il permet de voir quels nodes sont actuellement en route et quels types de messages sont envoyés à qui. L'exemple ci-dessous représente les éléments de navigation du robot Agribot :

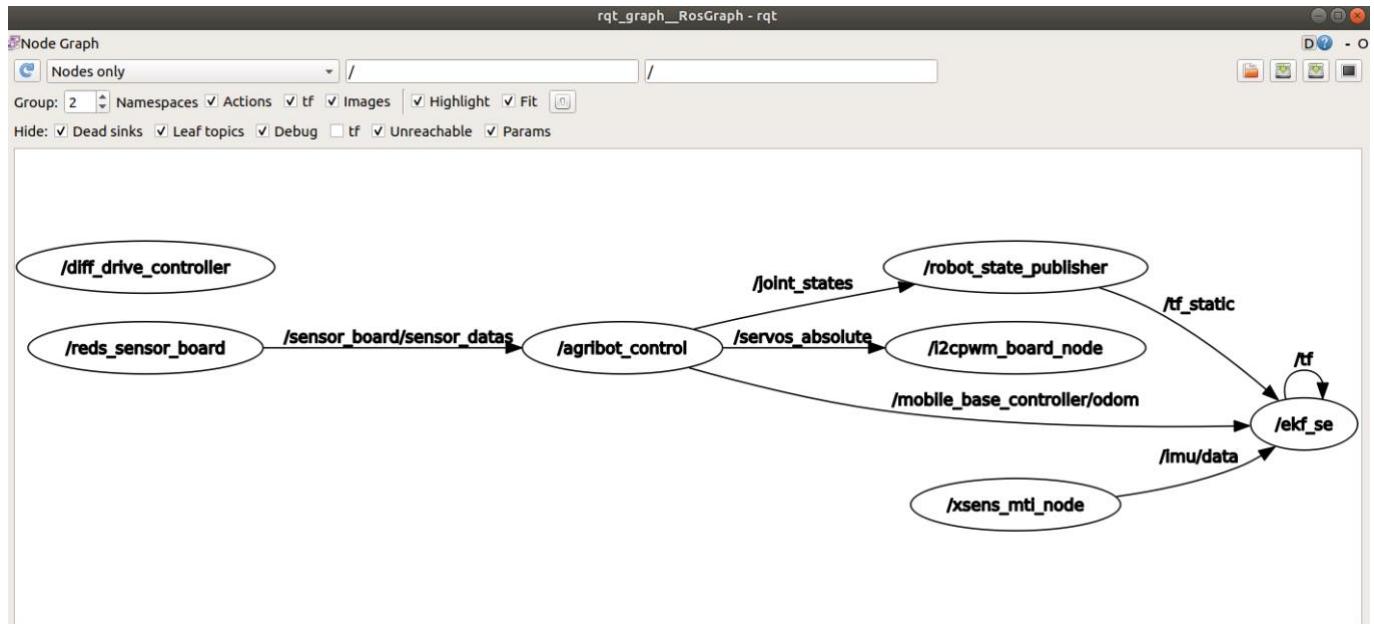


Figure 11 - Graphe généré par rqt\_graph

Le node /xsens\_mti\_node est créé à partir du package xsens\_mti fourni par Xsens. Il permet d'obtenir les données reçues du module Xsens avec les messages au bon format. Ici le node crée le topic /imu/data et la flèche créée indique que le node /ekf\_se est abonné aux publications de ce topic.

Le node ekf\_se est un node créé à partir du package robot\_localization. Il permet d'obtenir une odométrie fusionnée entre les encodeurs de roues et les données de l'imu.

Afin de décrire la structure du robot et les positions des éléments de mesures, il est nécessaire de fournir un fichier URDF (Unified Robot Description Format). Ces fichiers sont au format XML, on peut y décrire les caractéristiques physiques du robot et les positions relatives des capteurs et autres outils de mesures. Ces fichiers sont utilisés par différents nodes afin d'obtenir des résultats corrects pour les mesures faites par le système ROS pour le robot en question.

### 3.1 Navigation stack

Afin de mieux comprendre le système de navigation existant dans ROS qui sera mentionné par la suite, il est nécessaire d'expliquer le fonctionnement global de la stack de Navigation.

La stack de navigation ROS est une collection de nodes et d'algorithmes qui permet au robot de se déplacer de manière autonome. Ces algorithmes permettent notamment de trouver le chemin le plus court en prenant compte de l'environnement et des obstacles potentiels. Cette stack permet le déplacement du robot d'un point A à un point B sans se perdre et sans collision.

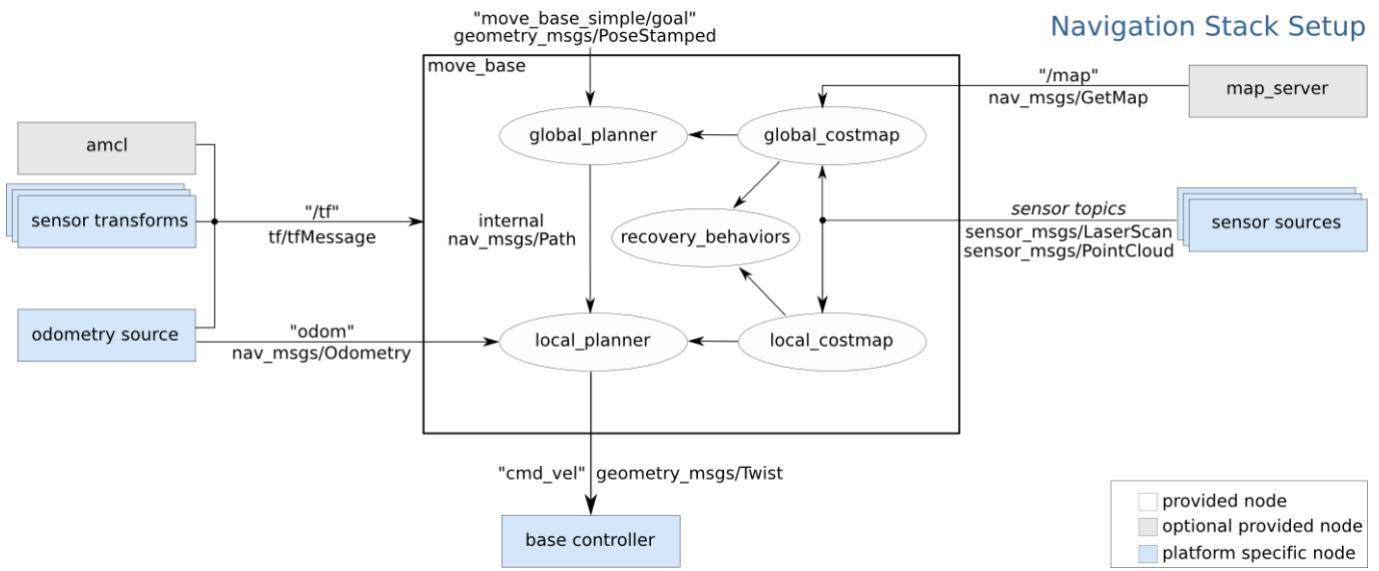


Figure 12 - Schéma de la stack de navigation par ROS

Ci-dessus, un schéma qui illustre les liaisons entre les topics et les nœuds. Les nœuds « \*\_planner » permettent de définir un itinéraire en fonction des informations fournies par les nœuds « \*\_costmap ». Les nœuds « \*\_costmap » permettent de fournir des informations sur l'environnement et les obstacles potentiels. Dans le cas du robot Agribot, les messages `sensor_msgs/LaserScan` sont envoyés par le Lidar.

Le topic `tf` permet de fournir des frames dans le temps. Les frames sont définis par chaque jointures et composants du robot, les frames sont des coordonnées de points à un instant T. Le `tf` permet donc de suivre l'état des transformations de ces frames dans le temps.

Plus d'information est disponible [ici](#).

## 4 Géolocalisation

Plusieurs problématiques se soulèvent afin de réaliser une géolocalisation au centimètre. Dans un premier temps, les recherches sont faites sans se limiter au matériel déjà présent sur le prototype actuel. Les sous sections qui suivent mettent en avant les points importants à prendre en compte pour la réalisation de ce projet.

### 4.1 Contraintes

La plus grande contrainte de ce projet est la précision du système de géolocalisation. La précision des méthodes de géolocalisations varie fortement, elles peuvent aller de +/- 300 cm à +/- 2,5 cm. La précision du positionnement du robot en extérieur est importante car la précision des actions du robot avec son environnement. Par exemple le bras mécanique du robot sera en mesure d'effectuer des actions très précise, de manière relative à la position du robot. Donc si la précision du robot dans son environnement est mauvaise, la précision du bras sera également mauvaise.

De plus, il est plus difficile de déterminer la position d'un robot en mouvement. Le robot devra connaître en tout temps sa position et déterminer sa trajectoire. Si on lui fournit une liste de coordonnées, il doit être en mesure d'aller de point en point et de rester dans un terrain délimité.

Le but est de positionner le robot au début d'une bande, potentiellement en une position connue, et lui fournir une liste de coordonnées à suivre pour l'arrosage des plantons.

Nous avons également des contraintes terrestres. Les champs ne sont pas toujours plats, il peut y avoir des dérapages odométriques si un trou survient ou si un caillou se trouve sur le passage. Il y a potentiellement des petits dénivélés à prendre en compte. Ceci implique l'obtention de l'altitude actuelle du robot.

Le système complet ne doit pas être trop cher, comparativement aux outils agricoles pouvant aller jusqu'à quelques dizaines voire centaines de milliers de francs. Il ne doit pas être trop envahissant dans le champ, les bandes changent chaque saison et donc les positions vont changer dans le temps.

### 4.2 Analyse des méthodes de positionnement

Dans un premier temps, une analyse des moyens de positionnements en extérieur est proposée. Si une solution de correction/amélioration de position existe pour la méthode de positionnement elle sera également analysée.

Une rencontre avec un ingénieur de l'INCIT permettra de valider les informations sur les méthodes de positionnement ainsi que le choix établi pour l'implémentation.

#### 4.2.1 Photogrammétrie

La photogrammétrie fait partie des méthodes de positionnement possible pour l'extérieur. Elle peut être utilisée comme méthode de positionnement en deux dimensions ou en trois dimensions. Elle permet de déterminer une position relative en utilisant un point fixe ou un objet de repère fixe. Cette technique utilise la géométrie des images obtenues (parallaxe) pour mesurer la distance entre l'objectif et le repère. Ainsi on peut obtenir la position de la caméra relative au repère. Le repère doit avoir une position connue.

Ci-dessous, un exemple de système de positionnement avec photogrammétrie :



Figure 13 - Exemple de solution avec photogrammétrie

Dans cet exemple de champ, un positionnement par photogrammétrie pourrait être réalisé à l'aide d'un élément tendu (2), sur élevé qui fait office de repère. Le robot utilise alors l'appareil photo (1) pour mesurer la distance entre l'élément et la caméra ce qui permet d'obtenir la distance relative entre le robot et l'élément. En couplant ceci avec l'odométrie des roues, on peut obtenir la position du robot dans le champ. Cette solution est loin d'être optimale et n'offre pas une précision inférieure à 3 cm.

Afin d'obtenir une position plus précise avec l'exemple ci-dessus, il est possible d'utiliser l'élément de référence comme source d'information. Par exemple l'élément de référence peut avoir sur sa surface des informations de positionnement, par exemple sous forme de code barre ou de QR-code. Avec de telles informations, on peut obtenir une précision inférieure à 2 cm.

Au niveau de l'infrastructure, l'exemple ci-dessous fonctionnerai pour une bande, il faut donc imaginer la répétition d'éléments de références sur tout un champ.

Une possibilité d'implémentation :

- Méthode : Appareil photo
- Précision : < 2 cm (dépend fortement de la qualité de l'appareil photo (marque, modèle), de la distance de l'objet et de la vitesse de traitement d'image)
- Avantages : Méthode relativement simple au niveau de la technologique et indépendante
- Inconvénients : installation de l'infrastructure lourde pour un champ, matériel non fiable suites aux intempéries (fils qui se détend avec la chaleur), difficile à maintenir, infrastructure couteuse, il ne faut pas d'obstacle entre la caméra et l'élément fixe de repère

#### 4.2.2 Lasergrammétrie

La lasergrammétrie fait partie des méthodes de positionnement possible pour l'extérieur. Elle peut être utilisée comme méthode de positionnement relative lorsqu'un objet de base ou une position de base est connue. Cette technologie utilise les faisceaux lumineux renvoyés par l'environnement afin de mesurer la distance relative aux objets.

Dans certain cas, elle permet également d'obtenir une visualisation en trois dimensions d'objets, il faut alors un système de plusieurs faisceaux lumineux avec des angles différents. On peut imaginer, dans l'exemple ci-après, que les éléments de références sont des lasers et que le robot est l'élément à reconstituer.

Dans un champ, un positionnement par lasergrammétrie pourrait être réalisé à l'aide d'objets disposés à des positions fixes, connues. Ces points doivent être légèrement en dehors des champs pour ne pas obstruer le passage du robot.

Le schéma ci-dessous illustre un exemple d'utilisation :

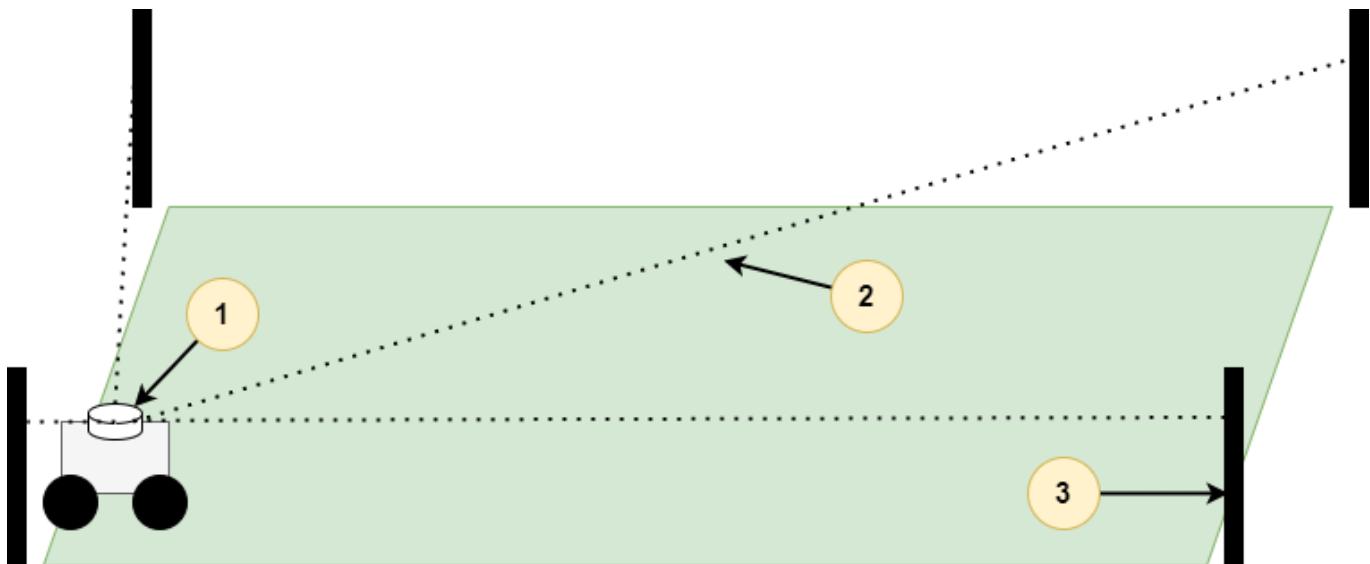


Figure 14 - Exemple avec lasergrammétrie

En numéro (1), on observe que le Lidar se trouve au dernier niveau du robot pour un champ de visibilité optimal (2). Dans cet exemple, il est nécessaire 4 éléments fixes sont positionnés autour du champ. Il est nécessaire, pour cette méthode, de positionner au minimum 3 éléments stables autour du champ afin d'avoir les repères nécessaires pour la réalisation des mesures de positionnement (relative aux objets). Il est également nécessaire d'avoir un champ de vision (2) clair entre le lidar et la cible, si le nombre d'éléments est supérieur à 3, on peut imaginer que le système se base alors sur ceux qui restent visibles.

Une possibilité d'implémentation :

- Méthode : Lidar
- Précision : +/- 5cm, dépend fortement du matériel à disposition (marque, modèle) et de la distance des objets de références
- Avantages : solution indépendante, relativement simple
- Inconvénients : nécessite une installation matérielle relativement lourde pour l'infrastructure car dépend de la distance supportée par le lidar (marque, modèle), nécessite un champ de vision libre entre le robot et les éléments de références

#### 4.2.3 GPS – Système GNSS

La géolocalisation par système GNSS, par satellite artificiel ou par GPS (abus de langage) peut être utilisée comme méthode de positionnement en utilisant des constellations de satellites artificiels.

Le schéma ci-dessous illustre un système de GNSS classique, c'est-à-dire en utilisant uniquement les signaux reçus (1) par les satellites visibles. Ces signaux (1) permettent d'obtenir une géolocalisation avec une précision s'approchant du mètre. Elle utilise les positions des satellites envoyées pour estimer une position.

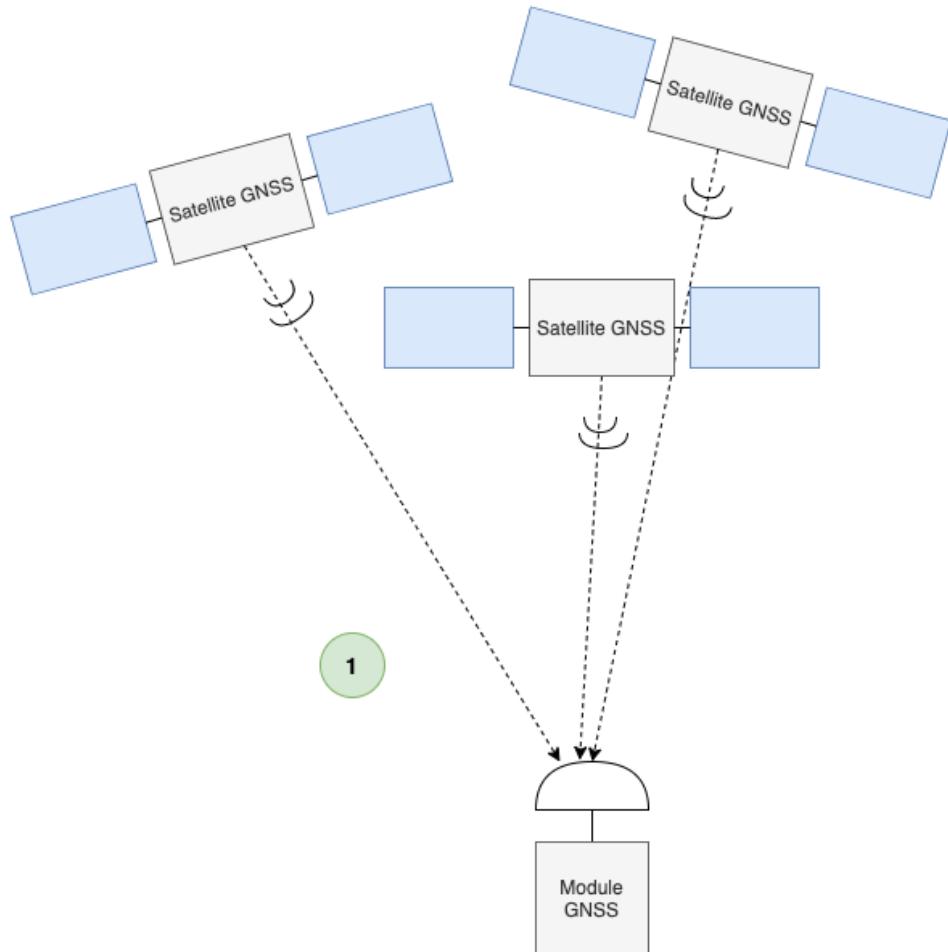


Figure 15 - Schéma de solution GNSS

Il existe plusieurs constellations de satellites pour le positionnement et la navigation :

- GPS : pour les Etats-Unis, à 20'200km d'altitude, avec 31 satellites opérationnels
- GLONASS : pour la Russie, à 19'100km d'altitude, avec 24 satellites opérationnels
- Galileo : pour l'Europe, à 23'222km d'altitude, avec 22 satellites opérationnels
- Beidou : pour la Chine, à 21'528km d'altitude, avec 20 satellites opérationnels

Une possibilité d'implémentation :

- Méthode : système GNSS du module Xsens
- Précision : < 1 m CEP
- Avantages : pas d'infrastructure nécessaire
- Inconvénients : mauvaise précision

Les sources d'erreurs qui engendrent un manque de précision des système GNSS simples sont principalement :

- La réfraction des signaux dans l'ionosphère, lorsque le signal se dégrade en traversant l'ionosphère à cause du milieu chargé en ions
- La réfraction des signaux dans la troposphère, lorsque le signal se dégrade en traversant la troposphère à cause de la teneur en eau de ce milieu
- La précision des orbites des satellites, lorsque le satellite ne mesure pas correctement sa position en orbite
- Les erreurs de synchronisation, lorsque d'une mauvaise corrélation temporelle entre le satellite et le récepteur
- Les multi-trajets, lorsqu'une partie du signal est renvoyé par l'environnement

Avec le choix du GPS, il est possible d'améliorer la précision de la position reçue par les satellites. Ci-après, une solution de correction de position.

#### 4.2.3.1 DGPS – GPS Différentiel

Le GPS différentiel est une méthode de calcul qui permet l'amélioration de la précision de géolocalisation. Cette amélioration de positionnement est possible en utilisant une station de base comme référence. Cette station de base peut être connue-publique ou privée-créée. Comme service publique, il existe [EGNOS](#), un service européen de correction de positionnement. Pour une station de base privée, il est possible de réaliser une version simple avec par exemple un module GNSS u-blox. Ces modules sont configurables pour être soit station de base soit rover, appareil mobile.

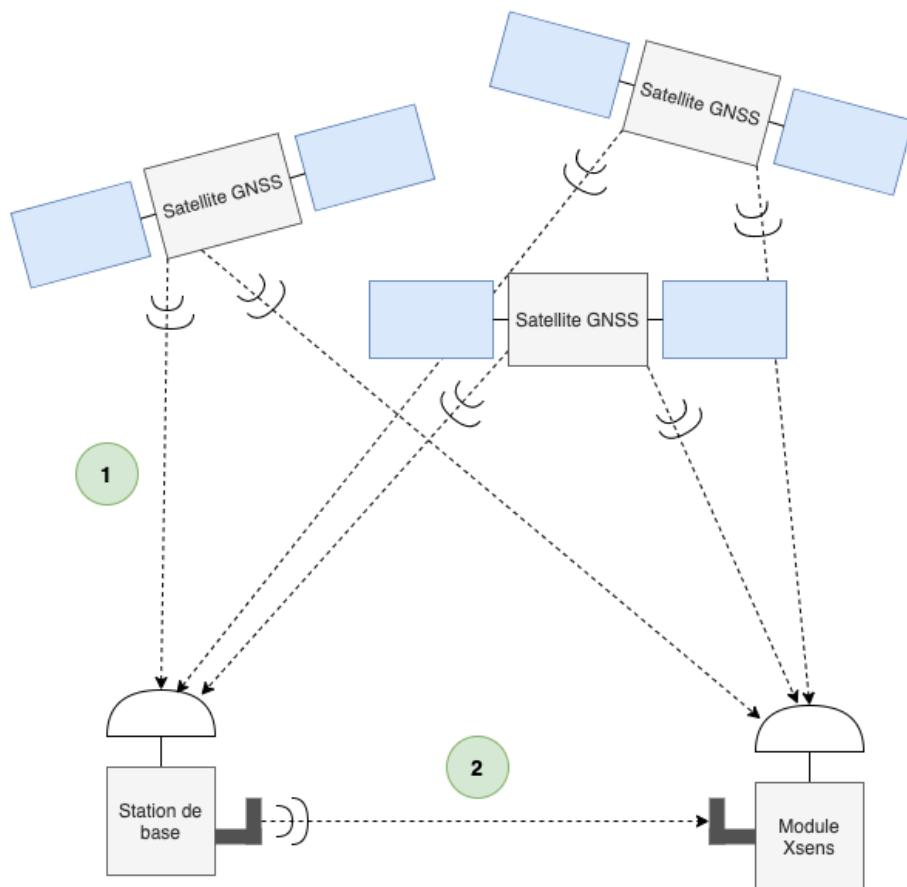


Figure 16 - Schéma d'une solution GNSS avec correction

Sur le schéma ci-dessus, on observe des satellites qui envoient des données de géolocalisation au module GNSS ainsi qu'à une station de base (1). Afin de réaliser les calculs, il est nécessaire n'avoir au minimum 4 satellites visibles. La station de base va calculer un delta (pseudo-distances) entre la position connue (fixe) et la position reçue par les satellites. Elle va ensuite envoyer ce delta sous forme de messages RTCM au robot afin qu'il puisse corriger la position reçue par les satellites.

### RTK

La correction RTK, est une correction différentielle de géolocalisation basée sur la phase du signal. Elle est basée sur le déphasage des ondes porteuses reçues par les systèmes de satellites. Une station de base fournit des messages de corrections de positionnement en temps réel. Ces messages peuvent être envoyé par une station de base connue via 4G ou LTE ou en réalisant une petite station de base avec du matériel prévu à cet effet.

Une possibilité d'implémentation :

- Méthode : système GNSS avec RTK du module Xsens
- Précision : 1 cm [CEP](#)
- Avantages : pas d'infrastructure lourde supplémentaire
- Inconvénients : nécessite soit un abonnement pour la réception des messages RTCM ou une station de base

#### 4.3 Solution existante

Il existe sur le marché des solutions clés en mains qui permettent d'obtenir une géolocalisation avec une précision aux alentours de 3cm. Parmi celles-ci, et dans le même domaine, il existe les solutions du constructeur de tracteur John Deere.



Figure 17 - Starfire 6000 avec modem RTK par John Deere

John Deere propose plusieurs solutions dans le domaine de l'agriculture de précision. Soit des tracteurs avec module ou simplement l'ajout du module à un tracteur existant. L'entreprise met à disposition un système d'abonnement aux divers services fournis. L'une des meilleures solutions proposées, meilleure en précision, est celle du nom de Starfire 6000 avec le modem RTK mobile 4G LTE, image ci-dessus. Cette combinaison permet d'obtenir une précision +/- 2,5 cm d'un passage à l'autre. Cette solution permet aussi, en combinant d'autres appareils et d'abonnement, de cartographier les champs, d'utiliser le guidage automatique ou encore la vision en directe sur la console à l'aide de caméras. Le but étant d'augmenter le rendement des agriculteurs en proposant des outils de qui utilise l'agriculture de précision toute prête à l'emploi.

#### 4.4 Choix du système de géolocalisation

Le choix du système de géolocalisation s'est fait principalement par rapport à la complexité de l'infrastructure à mettre en place. Sachant que le but est d'être le moins invasif possible dans le champ et d'offrir une solution flexible qui permet au paysan de changer ses bandes à chaque saison, le tout avec une précision proche du centimètre et un coût relativement bas. L'infrastructure à mettre en place joue un grand rôle dans le choix. Il faut que le choix permette la modularité des bandes.

L'implémentation de la solution sera la version du module GNSS sur le robot avec une station de base en plus. La station de base permettra d'envoyer des messages de corrections au module du robot via des ondes radio. Le modèle GNSS-IMU ci-dessous utilise la technologie RTK afin de calculer la position avec les messages RTCM.

Un nouveau module GNSS a été commandé pour le robot, c'est celui-ci qui va être utilisé. Ci-dessous, la liste de matériel choisi pour la réalisation du système de positionnement :

Ci-contre, le module [MTi-680G de Xsens](#). Il est capable de recevoir des messages RTCM et utilise la correction RTK.

Ci-dessous, [l'antenne GNSS calibrée et multi bande](#) recommandée par la [NGS](#) :



Figure 19 - Antenne GNSS multi bande, par ArduSimple



Figure 18 - Module MTi-680G de Xsens, par Xsens

Afin de recevoir les corrections RTCM, un [module radio XBee](#) ainsi qu'un [adaptateur XBee – RS232](#) :



Figure 20 - Module Radio XBee avec antenne par arduSimple

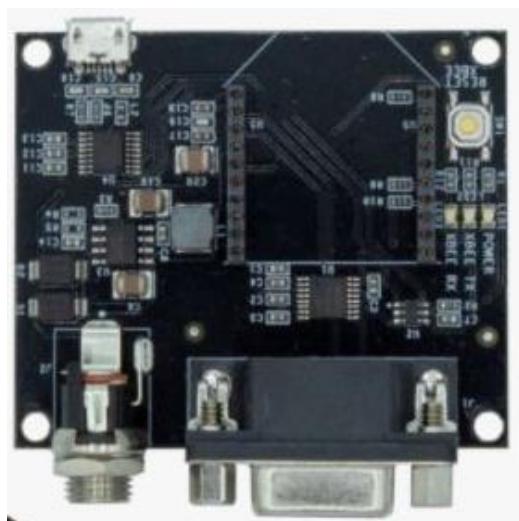


Figure 21 - Adaptateur XBee - RS232 par ardusimple

La station de base nécessite la même antenne GNSS que le module Xsens. Elle a également besoin du même module radio.

Ci-contre, la [carte simpleRTK2B](#) de la station de base :

On peut voir sur la carte le module GNSS de u-blox, il correspond aux modèles [ZED-F9P](#). C'est le même modèle que celui utilisé par le module MTi-680G de Xsens.

Le [chapitre 6](#) contient plus de détails sur le matériel utilisé.

Ces choix ont été fait en fonction des recommandations du fabricant Xsens. Une implémentation de cette solution ainsi que des instructions d'intégration sont disponibles [ici](#).

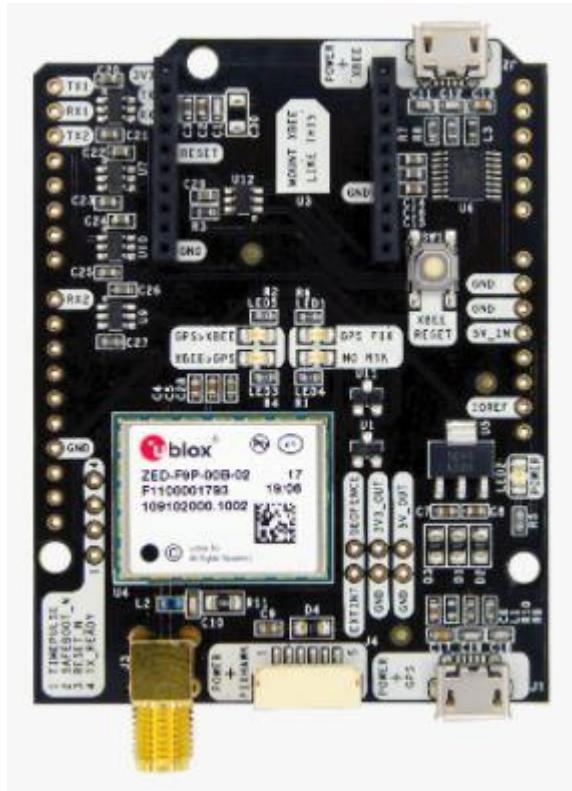


Figure 22 - Carte simpleRTK2B par ardusimple

La solution proposée par le référent de l'INCIT, Monsieur Sébastien Guillaume, est un système constitué de quatre antennes, quatre modules GPS et une station de base. Ceci permettrait d'obtenir 4 mesures indépendantes et de comparer les multi trajets observés entre les quatre mesures et ainsi d'améliorer davantage la précision. Ce travail pourrait être réalisé comme futur travail de Bachelor pour permettre d'obtenir une position avec une précision en dessous du centimètre.

Les validations de positionnement pourront être effectuées à l'aide d'un outil extrêmement précis à l'aide de l'équipe de Sébastien Guillaume, membre de l'INCIT, avec un théodolite. Un théodolite est un outil de mesure de distance qui utilise le laser pour mesurer la distance avec un objet, un prisme est utilisé sur l'objet ce qui permet d'obtenir une précision de l'ordre du millimètre. Cette précision permettra de valider les mesures GPS reçues du robot et la station de base. Pour réaliser ces validations un prisme 360° doit être monté sur le robot au même endroit que l'antenne afin de déterminer la précision du système GPS.

## 5 Cartographie

La cartographie comme décrite dans la plupart des littératures consiste à la réalisation et l'étude des cartes géographiques et géologiques.

### 5.1 Contraintes

Les contraintes du système de cartographie ou plutôt les besoins sont les suivants :

- Afficher les délimitations d'un champ
- Afficher les délimitations d'une bande
- Afficher l'emplacement des plantons
- Afficher l'emplacement d'un point d'eau
- Afficher l'emplacement d'une station de charge

La solution doit permettre de stocker et de gérer les différentes cartes.

### 5.2 Analyse des méthodes de cartographie

Le prototype actuel permet de cartographier un espace en intérieur avec des murs. En effet dans les exemples et les tutoriels ROS, on observe une majorité de projet pour des robots en intérieur. Le but de cette cartographie et du système existant est de déterminer ce qui se trouve dans l'environnement avoisinant afin de se déplacer en évitant les obstacles.

Pour ce projet, nous avons besoin de pouvoir cartographier un champ, une bande ou un planton. Pour ce faire nous devons avoir des limites. Qu'est ce qui fait partie du champ et qu'est ce qui est en dehors. Nous ne possédons pas de murs comme pour la cartographie en intérieur. Afin de simplifier l'utilisation du système GNSS, il est important de considérer une méthode de cartographie en extérieur. Cette méthode devra remplir les contraintes précédemment listées.

Le stockage des positions des plantons pour la cartographie permet de connaître la position exacte des bandes, elle pourra également être enregistrée au moment du premier passage. Ci-dessous un exemple de cartographie :

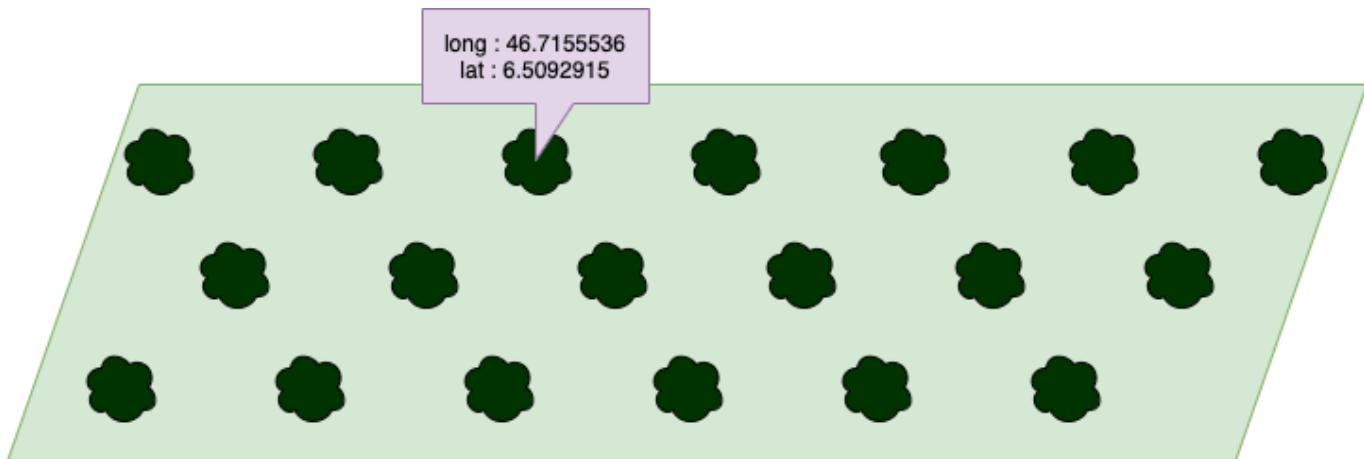


Figure 23 - Schéma d'une bande d'un champ avec la position des plantons

L'ajout de la cartographie permet de stocker les informations sur la position des plantons de manière propre et lisible à l'utilisateur.

Ci-dessous, un exemple de cartographie d'une bande où chaque planton a ses coordonnées de géolocalisation :

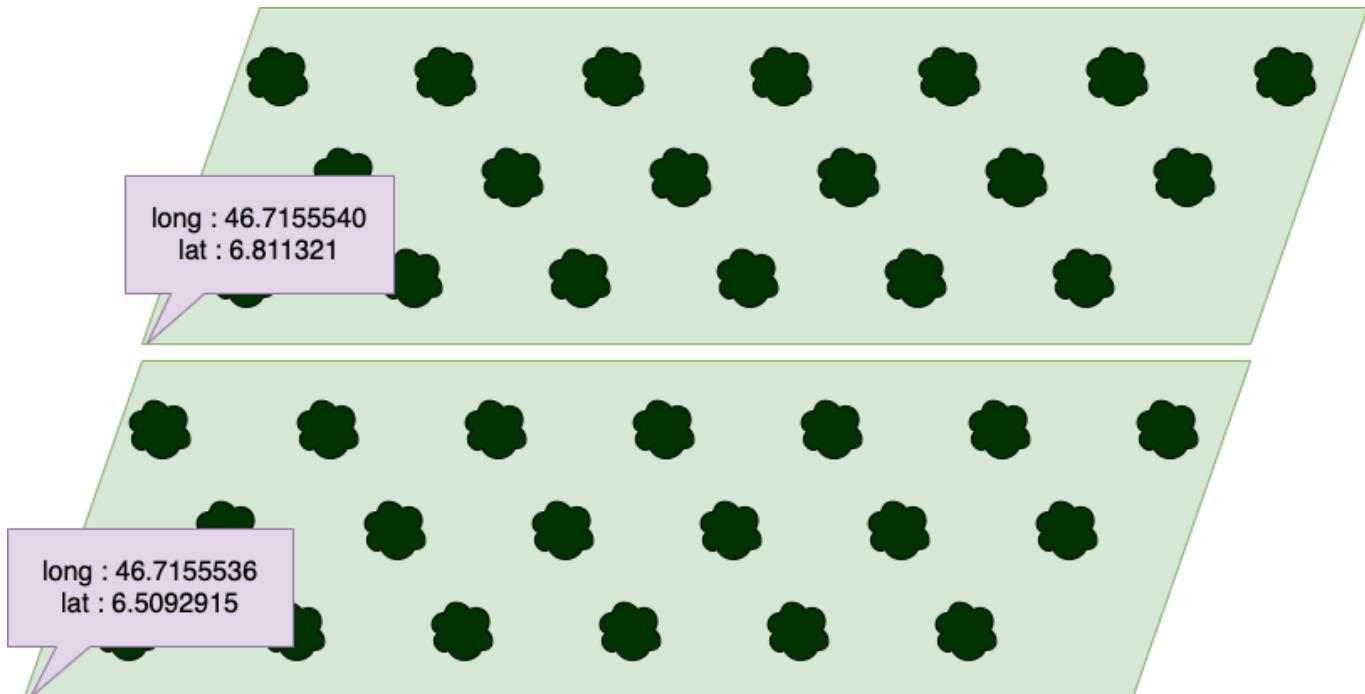


Figure 24 - Schéma d'un champ avec les coordonnées des différentes bandes

L'idée serait de pouvoir choisir une bande et d'envoyer une commande pour que le robot se déplace jusqu'au point choisi sur la carte.

### 5.2.1 Cartographie dans ROS

La gestion de la cartographie dans l'environnement ROS est réalisé par le package map\_server. Comme expliqué précédemment, cette solution est très pratique pour le déplacement en intérieur et la détection d'obstacle mais ne convient pas tel quel comme solution de cartographie.

Des projets ont été entrepris pour intégrer une carte de la terre et l'intégration des coordonnées au format [UTM](#). Ici se trouve l'un des projets qui va dans ce sens.

Comme on peut le voir sur la figure 12 du [chapitre 3.1](#), la stack de navigation permet de configurer un comportement lorsqu'un obstacle est détecté ou lorsqu'il. Une solution serait d'avoir deux modes, un mode classique qui dirige le robot à une position au début du champ et un mode de croisière lorsque le robot suit les points au-dessus de la bande. Ceci permettra d'éviter l'écrasement des plantons lorsque le robot rencontre un obstacle.

### 5.2.2 Stockage des coordonnées

[KML](#) est un langage destiné à la gestion de coordonnées du système WGS84. Ce langage est compatible avec la plupart des API (Google, Bing, Open Geospatial Consortium, ...).

Exemple simple d'un fichier au format KML :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <kml xmlns="http://www.opengis.net/kml/2.2">
3  <Document>
4  <Placemark>
5  <name>Pilonne 6170</name>
6  <description>Pilonne 6170, Yverdon</description>
7  <Point>
8      <!-- Longitude, Latitude, Altitude (optionnelle) -->
9      <coordinates>46.7802294, 6.6611095, 0</coordinates>
10 </Point>
11 </Placemark>
12 </Document>
13 </kml>
```

Figure 25 - Exemple de stockage de coordonnées au format KML

[GeoJSON](#), est un format ouvert qui permet d'encoder un ensemble de données géo-spatiales. Il utilise la norme JSON ce qui le rend très facile à manipuler et à sérialiser. Il existe d'ailleurs plusieurs librairies python qui permettent de le faire, notamment [gejson](#).

Exemple simple d'un fichier au format GeoJSON :

```
1  {
2      "type": "FeatureCollection",
3      "features": [
4          {
5              "type": "Feature",
6              "geometry": {
7                  "type": "Point",
8                  "coordinates": [46.7802294, 6.6611095]
9              },
10             "properties": {
11                 "name": "pilonne 6170"
12             }
13         },
14         {
15             "type": "Feature",
16             "geometry": {
17                 "type": "Polygon",
18                 "coordinates": [
19                     [
20                         {
21                             "type": "LineString",
22                             "coordinates": [
23                                 [46.7153642, 6.5094504], [46.7163901, 6.5101327],
24                                 [46.7165448, 6.5088060], [46.7158118, 6.5081559]
25                             ]
26                         }
27                     ]
28                 ]
29             },
30             "properties": {
31                 "name": "Champ d'Agiez",
32                 "nb_bandes": 12.0
33             }
34         }
35     ]
36 }
```

Figure 26 - Exemple de stockage de coordonnées au format GeoJSON

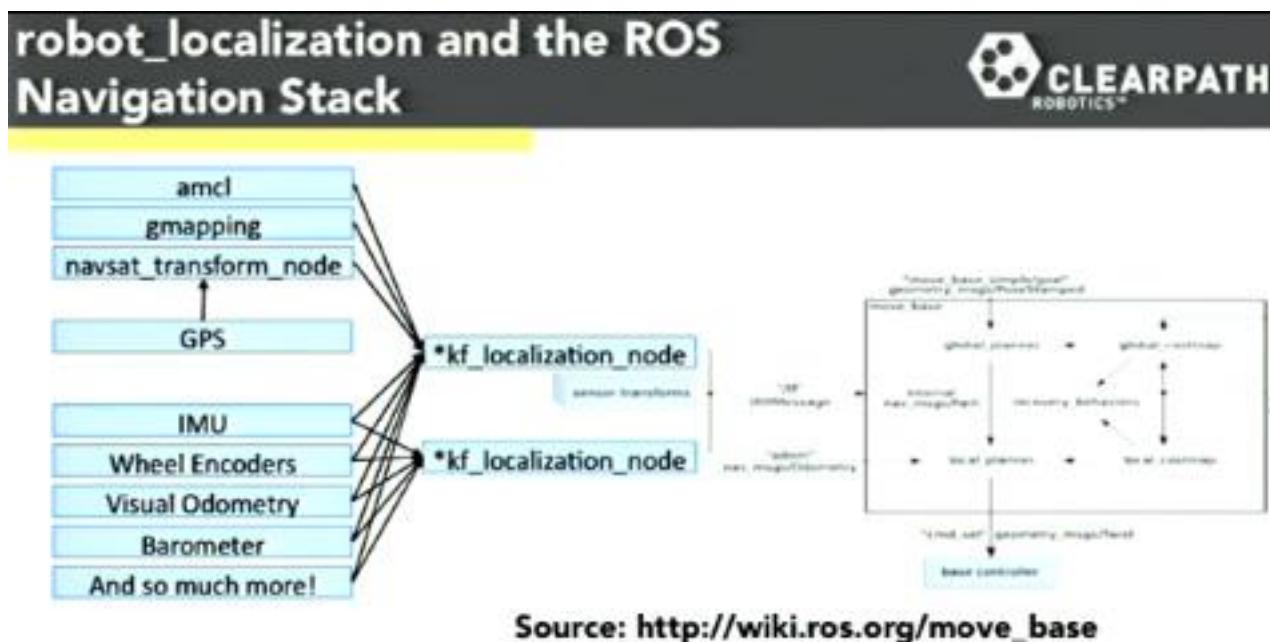
## 6 Conception et Implémentation

### 6.1 Navigation avec géolocalisation

Afin d'obtenir une commande embarquée de navigation qui utilise des données de géolocalisation, il est nécessaire de procéder étape par étape. Le paramétrage et la configuration du matériel en fait également partie.

#### 6.1.1 Conception

Dans un premier temps, il faut intégrer les données de l'IMU à la localisation du robot. Pour intégrer et fusionner les données GNSS, il faut avoir un bon système d'odométrie. Ceci permettra d'améliorer la précision de la localisation du robot relative à la base. Les données de l'IMU combinées aux données odométriques des roues seront utilisées par le package de localisation du robot « [robot\\_localization](#) ». Ce package est mis à disposition de la communauté ROS par l'entreprise [Clearpath Robotics](#), une entreprise de consulting spécialisée en déplacement autonome qui propose des solutions en robotique. Ce module permet de fusionner plusieurs mesures d'état de différents capteurs telles qu'ilustré ci-dessous :



Source: [http://wiki.ros.org/move\\_base](http://wiki.ros.org/move_base)

Figure 27 - Fusion des données de capteurs pour estimer la position du robot

Afin de fusionner les données odométriques des roues avec les données de l'IMU, il faut fournir des paramètres très spécifiques au node `ekf` de `robot_localization`. Ci-dessous, une liste des états possibles pour la fusion entre différentes mesures d'état :

- $x, y, z$  : pour l'état linéaire
- roll, pitch, yaw : pour l'état angulaire
- $vx, vy, vz$  : pour l'état de vitesse linéaire
- $vroll, vpitch, vyaw$  : pour l'état de vitesse angulaire
- $ax, ay, az$  : pour l'état d'accélération linéaire

Ces paramètres permettront de créer un node d'estimation de position basé sur les paramètres fournis. Deux filtres différents peuvent être appliqués aux données : Le filtre de Kalman étendu ou le filtre de Kalman moins étendu. Ces paramètres seront expliqués plus en détails dans le chapitre suivant.

Dans un deuxième temps, il faut intégrer les données GNSS au système de localisation du robot afin qu'il puisse se localiser en utilisant les données GNSS fournies par le module Xsens. Le type de données attendu est

nav\_msgs/Odometry. Pour faire la transformation, il faut créer un node navsat\_transform. Les données GNSS du module sont transformées en message de type sensors\_msgs/NavSatFix.

Ci-dessous, le header des messages de type de nav\_msgs/Odometry :

## Compact Message Definition

```
std_msgs/Header header
string child_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist
```

Figure 28 - Définition du message Odometry

Les messages de ce type sont utilisés pour définir une estimation de position dans l'espace à un temps donné. Ces messages peuvent être reçus par le node de robot\_localization. Ci-dessous, le header des messages de type de sensors\_msgs/NavSatFix fournit par le module GNSS:

## Compact Message Definition

```
uint8 COVARIANCE_TYPE_UNKNOWN=0
uint8 COVARIANCE_TYPE_APPROXIMATED=1
uint8 COVARIANCE_TYPE_DIAGONAL_KNOWN=2
uint8 COVARIANCE_TYPE_KNOWN=3
std_msgs/Header header
sensor_msgs/NavSatStatus status
float64 latitude
float64 longitude
float64 altitude
float64[9] position_covariance
uint8 position_covariance_type
```

Figure 30 - Définition du message NavSatFix

Pour faire la transformation, il faut créer un node du type `navsat_transform` qui prend en entrées les données brutes du module GNSS de type `sensors_msgs/NavSatFix` et retourne un message avec des coordonnées UTM.

Ci-dessous un schéma qui illustre la transmission d'information avec tous les nodes nécessaires pour fournir une localisation :

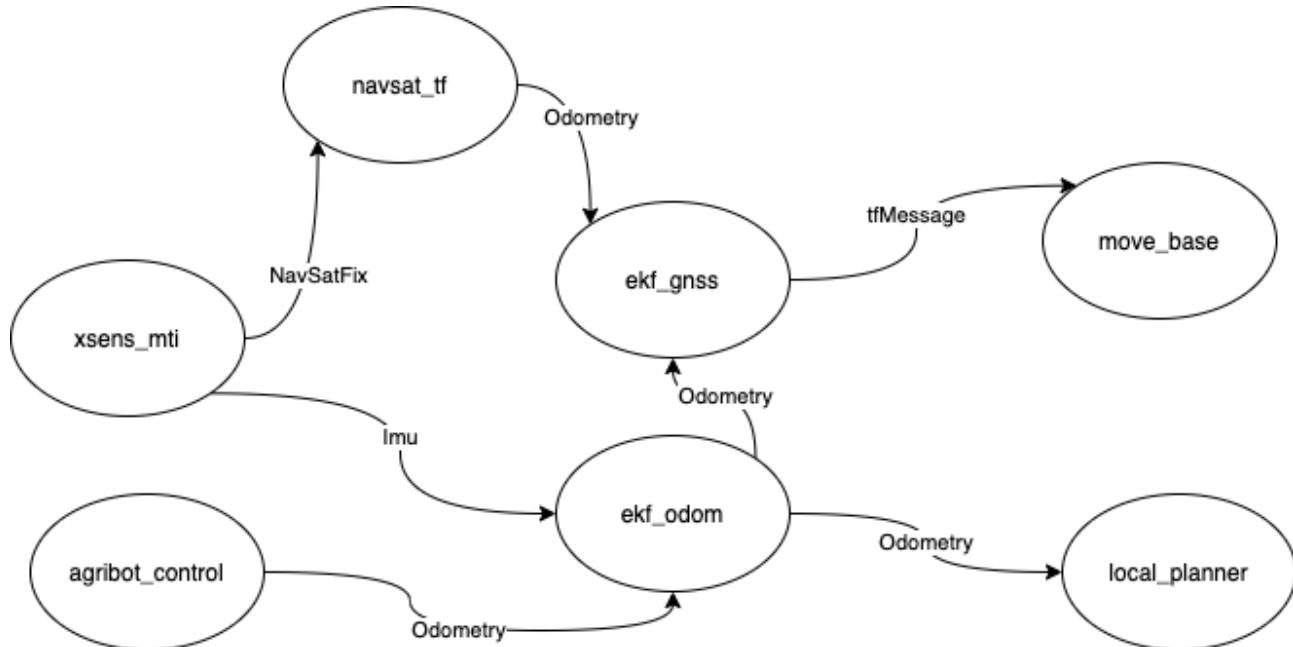


Figure 31 - Schéma de node avec messages pour l'intégration des données GNSS

Les ellipses représentent les nodes. Les noms sur les liens correspondent aux messages. Ce schéma correspond à ce qui doit être mis en place afin d'obtenir une localisation avec le système GNSS et RTK.

Ci-dessous, le schéma logique de l'intégration du système GNSS avec la station de base, le système existant est simplifié avec un élément Agribot :

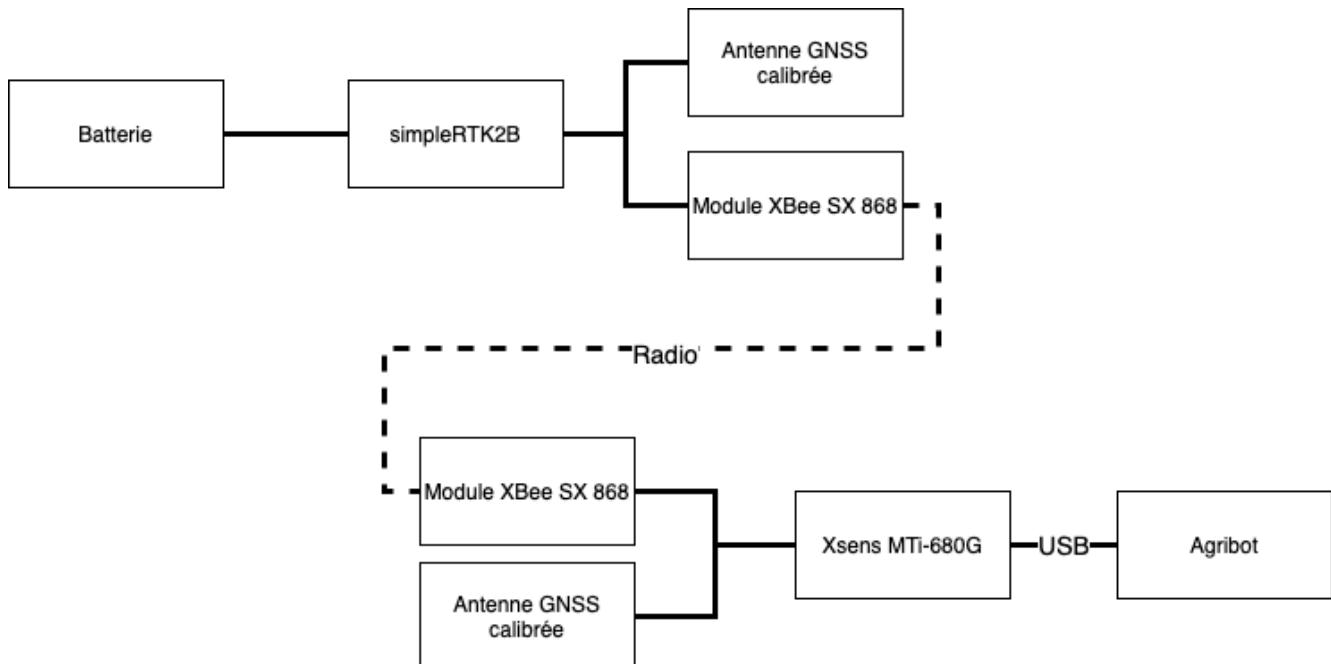


Figure 32 - Schéma bloc du matériel à mettre place pour l'intégration du système GNSS avec RTK

## 6.1.2 Implémentation

Pour pouvoir configurer le module Xsens, il est nécessaire d'installer le logiciel MT Manager fourni par Xsens. Celui-ci est prévu pour Windows mais peut également être installer sur Linux. En annexe, un [tutoriel](#) d'installation explique les démarches à réaliser.

Une fois le logiciel installer, il est possible de configurer les données de sorties du module. Pour l'utilisation de ce projet, la configuration actuelle est disponible en [annexe](#). Le document d'utilisation fournit par Xsens sur MT Manager permet de configurer facilement le matériel.

Ci-dessous, un aperçu de l'interface avec les données de l'IMU :

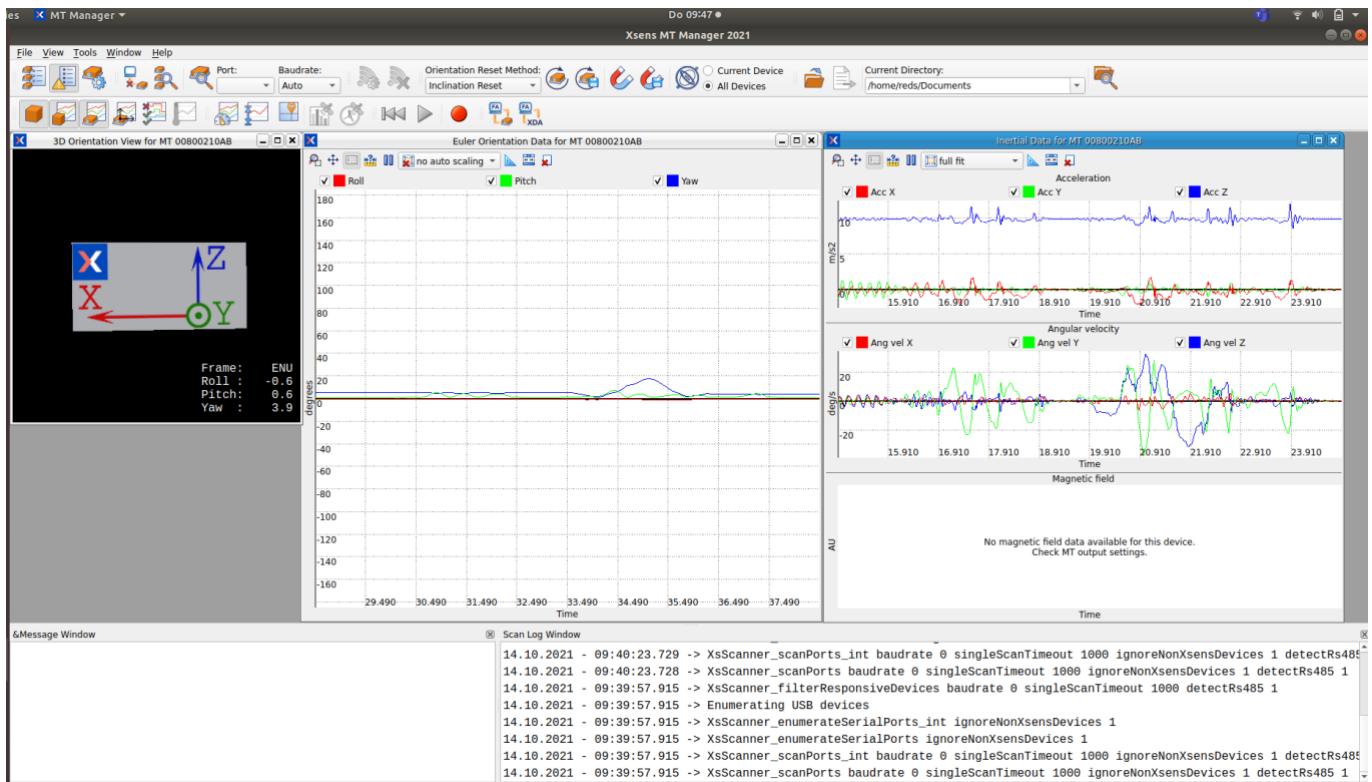


Figure 33 - Aperçu de l'interface MT Manager avec les données de l'IMU affichées

Les paramètres que l'on souhaite obtenir pour l'intégration de la navigation GNSS sont les suivants :

- Données d'orientation : quaternion, à 50Hz
- Données inertielles : delta q et delta v, à 50Hz
- Status : status world (pour le debug)
- Position and velocity : latitude et longitude, velocity, à 50Hz
- GNSS data : pvt data et sat info, à 4Hz (pour le debug)

Une fois ces données correctement paramétrées, il faut ajouter le driver du module Xsens dans ROS. Ce driver doit être compiler soit sur la Raspberry soit cross-compilée depuis le pc de développement. Il faut ensuite vérifier que les messages reçus par le node correspondent bien aux messages attendus comme spécifié dans la configuration. Le driver est monté à chaque redémarrage de la Raspberry pi, le fichier init\_system a été modifié en conséquence.

Ensuite il faut installer le package robot\_localization et le compiler. La compilation sur la Raspberry pi surcharge la demande de courant et la Raspberry redémarre. Pour les compilations de gros module préférer l'utilisation de 2 cœurs.

Une fois le package installé, Il faut commencer par créer le node qui fera la fusion des données de l'IMU (reçue par le module Xsens) avec les données reçues par de l'odométrie des roues. L'extrait de configuration suivant montre quels états de mesures on veut fusionner :

```
odom0: /mobile_base_controller/odom

# Each sensor reading updates some or all of the filter's state. These options give you greater control over which
# values from each measurement are fed to the filter. For example, if you have an odometry message as input, but only
# want to use its Z position value, then set the entire vector to false, except for the third entry. The order of the
# values is x, y, z, roll, pitch, yaw, vx, vy, vz, vroll, vpitch, vyaw, ax, ay, az. Note that not some message types
# do not provide some of the state variables estimated by the filter. For example, a TwistWithCovarianceStamped message
# has no pose information, so the first six values would be meaningless in that case. Each vector defaults to all false
# if unspecified, effectively making this parameter required for each sensor.
odom0_config: [false, false, false,
               false, false, false,
               true, true, false,
               false, false, true,
               false, false, false]
odomN_queue_size: 1000

# [ADVANCED] When measuring one pose variable with two sensors, a situation can arise in which both sensors under-
# report their covariances. This can lead to the filter rapidly jumping back and forth between each measurement as they
# arrive. In these cases, it often makes sense to (a) correct the measurement covariances, or (b) if velocity is also
# measured by one of the sensors, let one sensor measure pose, and the other velocity. However, doing (a) or (b) isn't
# always feasible, and so we expose the differential parameter. When differential mode is enabled, all absolute pose
# data is converted to velocity data by differentiating the absolute pose measurements. These velocities are then
# integrated as usual. NOTE: this only applies to sensors that provide pose measurements; setting differential to true
# for twist measurements has no effect.
odom0_differential: false
odom0_twist_rejection_threshold: 3.0

imu0: /imu/data
imu0_config: [false, false, false,
              false, false, true,
              false, false, false,
              false, false, true,
              true, false, false]
imu0_differential: false
```

Figure 34 - Extrait de la configuration du node de fusion IMU

Dans cette configuration, les données que l'on souhaite fusionner pour l'odométrie des roues sont celles des vélocités linéaires pour x, y et z, la vélocité angulaire z. Pour l'IMU, on souhaite fusionner l'angle z, la vélocité angulaire z et l'accélération x.

Les fichiers de configuration contiennent des explications sur l'utilité des paramètres ajoutés. Certains paramètres ont des valeurs par défaut d'où l'importance de les mettre à false dans la configuration.

Une fois ce node créé et fonctionnel, il faut réaliser le node navsat\_transform qui va s'occuper de fournir les messages au bon format. Ci-dessous la configuration du node navsat :

```
<launch>
  <!-- Start robot_localization with gps, gps data is transformed into odometry data -->
  <node name="navsat_transform" pkg="robot_localization" type="navsat_transform_node" clear_params="true" output="screen" >
    <param name="frequency" value="30" />
    <param name="delay" value="3.0" />
    <param name="magnetic_declination_radians" value="0.00" />
    <param name="yaw_offset" value="0.0" />
    <param name="zero_altitude" value="true" />

    <param name="broadcast_cartesian_transform" value="true" />
    <param name="publish_filtered_gps" value="true" />

    <param name="use_odometry_yaw" value="false" />
    <param name="wait_for_datum" value="false" />

    <remap from="/imu/data" to="/imu/data" />
    <remap from="/gps/fix" to="/gnss" />
    <remap from="/odometry/filtered" to="/odometry/gps" />
    <remap from="/odometry/gps" to="/odometry/navsat" /> <!-- Usefull ? -->
  </node>
</launch>
```

Figure 35 - Configuration du node navsat\_transforme

Ensuite on peut créer le node ekf qui va fusionner les données GNSS avec celles de IMU et de l'odométrie des roues. Ci-dessous, un extrait de la configuration :

```
# Wheel encoders data
odom0: /mobile_base_controller/odom

# Each sensor reading updates some or all of the filter's state. These options give you greater control over which
# values from each measurement are fed to the filter. For example, if you have an odometry message as input, but only
# want to use its Z position value, then set the entire vector to false, except for the third entry. The order of the
# values is x, y, z, roll, pitch, yaw, vx, vy, vz, vroll, vpitch, vyaw, ax, ay, az. Note that not some message types
# do not provide some of the state variables estimated by the filter. For example, a TwistWithCovarianceStamped message
# has no pose information, so the first six values would be meaningless in that case. Each vector defaults to all false
# if unspecified, effectively making this parameter required for each sensor.
odom0_config: [false, false, false,
                false, false, false,
                true, true, false,
                false, false, false,
                false, false, false]
odom0_queue_size: 10
odom0_nodelay: true
odom0_differential: false
odom0_relative: false

# GPS odometry (GNSS data)
odom1: /odometry/gps
odom1_config: [true, true, false,
                false, false, false,
                false, false, false,
                false, false, false,
                false, false, false]
odom1_queue_size: 10
odom1_nodelay: true
odom1_differential: false
odom1_relative: false

# IMU data
imu0: /imu/data
imu0_config: [false, false, false,
                true, true, false,
                false, false, false,
                true, true, true,
                true, true, true]
imu0_queue_size: 10
imu0_nodelay: true
imu0_differential: false
imu0_relative: false
imu0_remove_gravitational_acceleration: true
```

Figure 36 - Extrait de la configuration du node ekf GNSS

Ici on fusionne deux odométries différentes, celle en provenance des roues et celles calculée à partir des données GNSS grâce au node navsat\_transform.

### 6.1.3 Montage sur le rob

Ci-dessous, le robot avec le système GNSS monté :

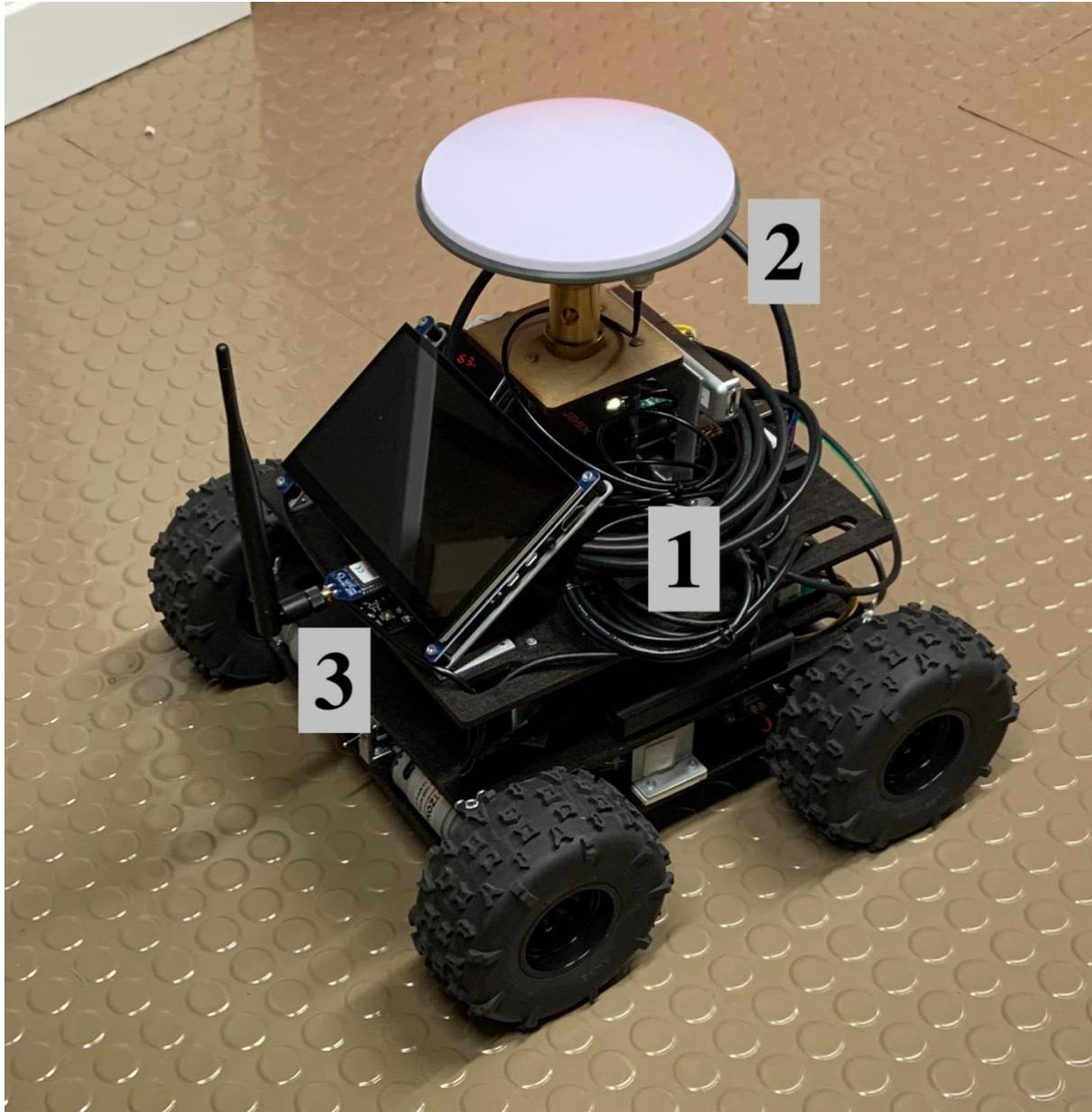


Figure 37 - Robot assemblé avec le système GNSS

Le numéro (1) correspond au module Xsens MTi-680G, caché derrière les câbles. Le numéro (2) correspond à l'antenne GNSS calibrée. Le numéro (3) correspond au module radio XBee connecté au module Xsens. Un nouveau petit plateau a été ajouté afin de pouvoir positionner les nouveaux éléments les uns sur les autres. Ceci permet de pouvoir garder les fichiers de configurations URDF intacte. Le positionnement du module doit être indiqué avec précision afin d'obtenir des résultats justes pour le robot.

Ci-dessous, la station de base assemblée sur un pilonne pour des tests :



Figure 38 - Station de base assemblée sur le terrain

L'image ci-dessus illustre la nouvelle station de base. Elle n'a pas encore de protection contre les intempéries.

#### 6.1.3.1 MTI-680G, Xsens

L'image ci-dessous correspond au module Xsens MTi-680G. Elle illustre le sens des axes définis par défaut. Les numéros en rouges pointent sur les interfaces input-output du module.

Le numéro (1) correspond à l'interface qui permet de brancher l'antenne. Le numéro (2) correspond à l'interface hôte qui fait input-output, elle permet la communication entre le module Xsens et la cible (Raspberry Pi). Cette interface permet également d'alimenter le module. Le numéro (3) correspond à l'input RTCM, elle permet de connecter un module de réception de message de correction de type RTCM.

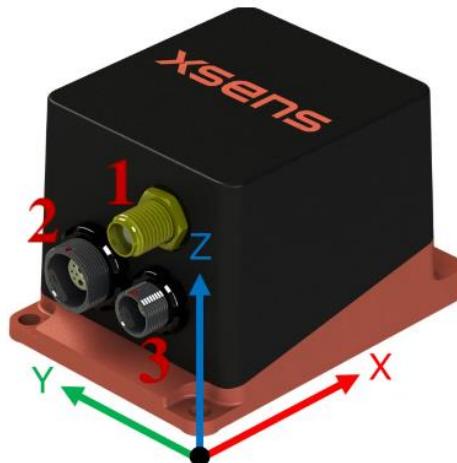


Figure 8: Default sensor coordinate system for the MTi-680G

Figure 39 - Module Xsens

Les axes présents sur l'image permettent d'indiquer le sens des axes pour une intégration correcte des mesures.

Ce module GNSS possède un gyroscope, un accéléromètre, un magnétomètre, un baromètre ainsi qu'un module de réception GNSS u-blox ZED-F9P. Il permet de recevoir les données suivantes :

- Roll, Pitch avec une précision de 0.2°
- Yaw, Heading avec une précision de 0.8°
- Position avec 1cm de précision
- Velocity avec une précision de 0.05m/s

Le module permet de faire des lissages de mesure et réalise des algorithmes de corrections afin de pouvoir fournir les précisions affichées.

Consommation d'énergie : 720mW en 5V.

#### 6.1.3.2 Antenne GNSS calibrée

Cette antenne GNSS active, calibrée par la NGS possède le label IP67 et elle est donc résistante à l'eau. Elle consomme environ le double que l'antenne précédente soit 45mA.

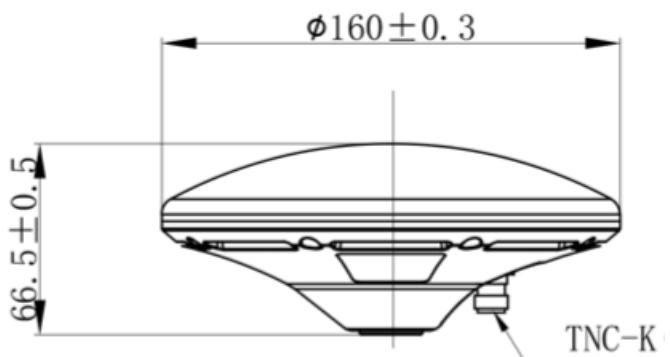


Figure 40 - Schéma de l'antenne GNSS calibrée

### 6.1.3.3 Module radio XBee

Le module XBee SX 868 permet d'envoyer des données à 868MHz sur une distance allant de 10 à 14Km.

### 6.1.4 Station de base

La station de base va permettre d'envoyer des messages RTCM au module GNSS du robot via des ondes radios. La réalisation de la station de base peut se faire en suivant les explications du projet [xsens imu gps rtk](#) avec quelques détails près.

En s'a aidant du document présent en [annexe](#), il est possible configurer la station de base avec une position fixe. Ci-dessous, l'interface U-Center de configuration des messages :

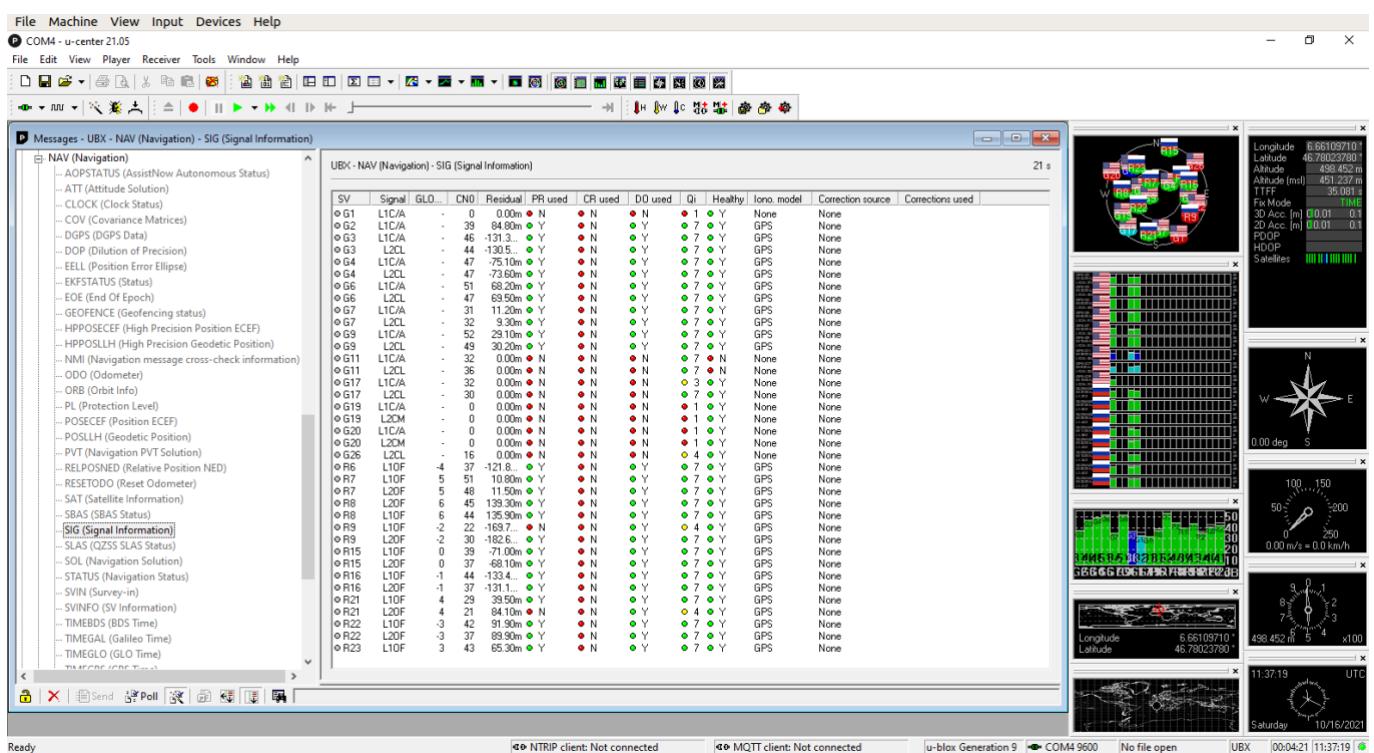


Figure 41 - U-Center, interface de configuration et de statistiques

Les fichiers de configurations se trouvent dans les annexes. Il y a une configuration pour un mode Survey-In et une configuration pour une position fixe. La position fixe actuelle est celle du pilonne 6170.

Les outils libres de RTKLib sont intéressants pour observer les données qui sorte du module radio ou faire du debug.

### 6.1.5 Communication PC – Agribot

La configuration initiale permet à un ordinateur de communiquer sur le réseau wifi de l'école (HEIG-Devices) avec le robot. Pour pouvoir effectuer des tests en extérieurs, il est donc nécessaire de réaliser un réseau wifi portable. Dans ce but, un réseau wifi portable a été réalisé.

Ci-dessous, le matériel nécessaire à la réalisation d'un wifi :

- Une Raspberry pi 4 B (minimum 3b)
- Une batterie portable

Un tutoriel est disponible afin de transformer/configurer la Raspberry pi en point d'accès wifi sur le site de [Raspberry](#).

La consommation maximum d'un modèle 4 B est de 1280mA, soit 6.4 W. Avec une batterie de 10'000mAh on peut tenir au minimum 7 h 45 minutes.

## 6.1.6 Packages utilisés

Ici sont listé les nouveaux packages installés pour l'intégration de la localisation avec système GNSS.

### 6.1.6.1 *Xsens\_mti - ROS*

Xsens\_mti permet de transformer facilement les données du module en message compatible ROS.

### 6.1.6.2 *Robot\_localization - ROS*

Robot\_localization permet de fusionner des données d'état de capteur en utilisant des algorithmes d'estimation, EKF et UKF.

### 6.1.6.3 *Geographiclib - Python*

[Geographiclib](#) est un package Python qui permet de mesurer une distance entre deux coordonnées GNSS du système WGS84 en degré décimal.

## 7 Tests

### 7.1 Tests du module Xsens dans MT Manager

Ci-dessous, une capture d'écran de l'outil de gestion des produits Xsens avec, en rouge, un suivi du déplacement du module :

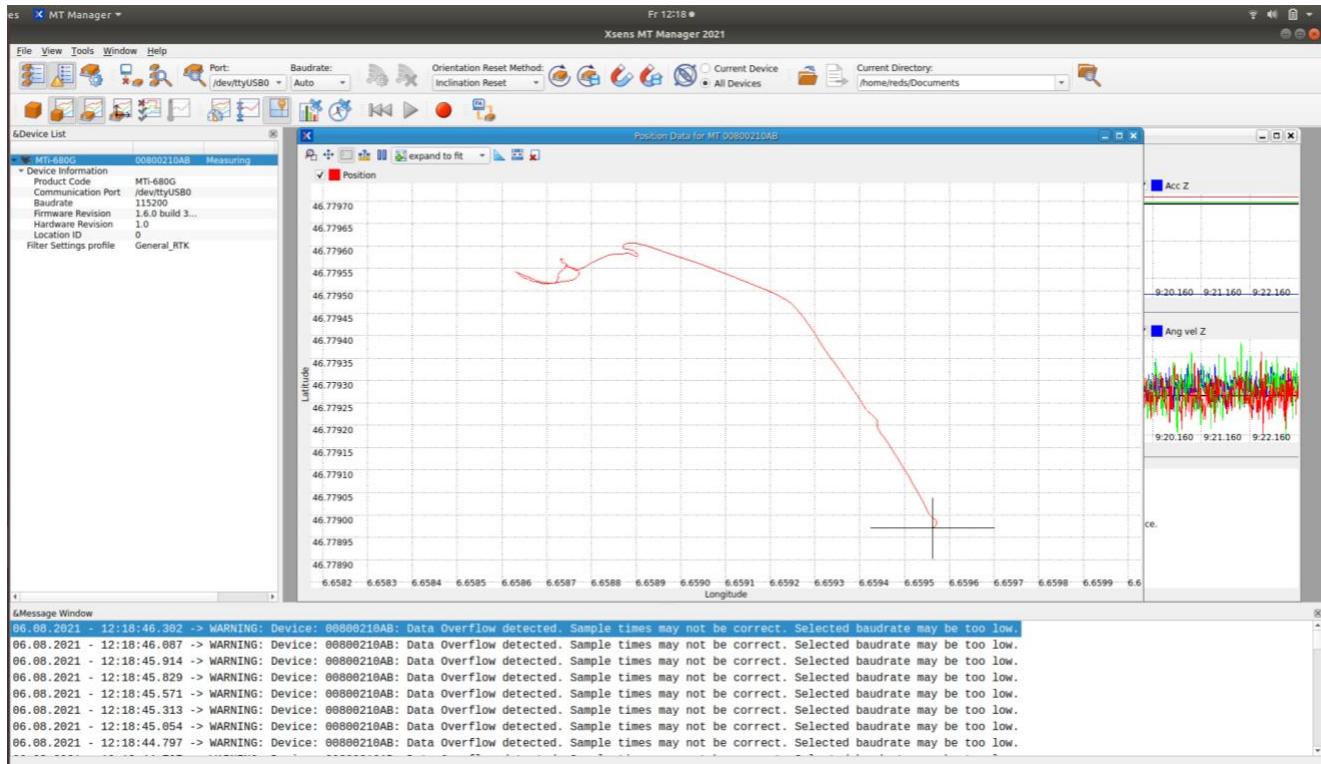


Figure 42 - MT Manager, interface

On peut voir dans la fenêtre de message des warnings. Ils sont survenus la configuration était trop chargée par rapport à la fréquence d'envoi des données. Afin de ne pas surcharger la Raspberry pi, le débit de transmission reste à 115200 bauds. La fréquence de transmission d'information choisie est donc limitée à 50Hz.

## 7.2 Tests du système GNSS

Afin de mettre en avant la précision sans système de correction, des tests de variations ont été effectués. Le détail des résultats se trouve [ici](#). Ces mesures ont permis de définir la précision du module GNSS sans correction. On observe des variations absolues maximales pour l'est de 2,311 m et pour le nord de 2,992 m.

On remarque que si on attend entre 5 et 10 minutes, les variations absolues maximales baissent. Elles deviennent pour l'est de 1,286 m et pour le nord de 1,905 m.

Ces mesures ont été réalisées à la main, en positionnant le robot toujours au même endroit : sur le pilonne 6170 situé derrière l'école. Les chiffres indiqués correspondent à une moyenne faite sur une minute de mesures. La précision du positionnement de l'antenne sur le pilonne est vérifiée avec une précision au millimètre grâce au niveau présent sur le pied :

Ci-dessous une image de l'endroit où ont les mesures ont été prises :



Figure 44 - Installation de tests

Sur cette image, on voit le module Xsens connecté à une antenne GNSS positionnée avec un maximum de précision sur l'un des pilonnes de l'école. La position de ce point, 6170, nous a été fourni par le professeur de l'INCIT.

## 7.3 Tests du système GNSS avec correction

Dans un deuxième temps, des mesures avec corrections ont été faites. Le détail des résultats se trouve [ici](#). Ces mesures ont permis de définir la précision du module GNSS avec correction.



Figure 43 - Précision de positionnement de l'antenne

Des tests en mode Survey-In ont été réalisés. Avec ces résultats, on observe des variations absolues maximales pour l'est de 1,533 m et pour le nord de 1,561 m.

Un test en mode position fixe a été réalisé. Avec ces résultats, on observe toujours la même variation, pour l'est une variation de 0,014 m et pour le nord une variation de 0,535 m.

Mais ce qui est intéressant pour ces mesures ce sont les variations relatives. On observe des variations relatives maximales de 0,076 m pour l'est et de 0,111 m pour le nord avec le mode Survey-In. Avec le mode fixe il n'y a pas de variations. Vu qu'on ne bouge pas on a bien une position fixe.

Les résultats obtenus sont très intéressants mais il manque deux ou trois tests afin de pouvoir dire avec certitude que la précision du système GNSS avec correction implémenté avec une précision < à 2cm.

Ces mesures ont été réalisées à la main, en positionnant l'antenne du robot à une position connue, sur le pilonne 6170. La station de base est en mode Survey-In et en mode fixe, elle se situe à côté du pilonne. À une distance d'environ 50cm.

Ci-dessous, la station de base :



Figure 45 - Station de base pour les tests

La configuration de la station de base est disponible dans cette [annexe](#).

### 7.3.1 Conversion WGS84 – MN95

Afin d'obtenir des résultats comparables entre les données de tests les positions de références, et de pouvoir parler de variations en mètre, il faut convertir les données en degré décimal du système WGS84 en données métriques du système MN95.

Pour réaliser ces transformations facilement sur les résultats des tests effectués, la réalisation d'un script de conversion rend la tâche plus agréable.

La confédération met à disposition une API REST qui permet de faire des requêtes de conversion. Un document explicatif est disponible [ici](#) pour l'utilisation de l'API.

Le script réalisé peut s'utiliser principalement de la manière suivante :

- long\_lat\_csv\_to\_wgs84\_csv.py src.csv dst.csv

Les entêtes de fichier doivent être : Longitude mesurée, Latitude mesurée.

## 7.4 Tests de l'intégration du système GNSS à la navigation

Lorsque l'intégration des coordonnées GPS sont comprises par le node, on observe les messages suivants :

```
/home/ubuntu/robot_reds/dev/software/robot_ws/src/robot/agribot/launch/agribot_manual_test_gnss.launch http://192.168.4.14:11311/○ ×
File Edit View Search Terminal Tabs Help
reds@reds-T... × reds@reds-T... × reds@reds-T... × ubuntu@agri... × ubuntu@agri... × /home/ubuntu... × reds@reds-T... × + ▾
[INFO] [1634225069.359601]: Started controllers: mobile_base_controller, joint_state_controller
[INFO] [1634225071.193437065]: Initial odometry pose is Origin: (0 0 0)
Rotation (RPY): (0, -0, 0)

[ INFO] [1634225071.193933325]: Datum (latitude, longitude, altitude) is (46.780173, 6.659103, 506.041000)
[ INFO] [1634225071.194125732]: Datum UTM coordinate is (321306.368786, 5183396.876985) zone 32
[INFO] [1634225071.226735251]: Corrected for magnetic declination of 0.000000, user-specified offset of 0.000000 and
meridian convergence of -0.029781. Transform heading factor is now -3.167625
[INFO] [1634225071.227064269]: Transform world frame pose is: Origin: (0 0 0)
Rotation (RPY): (0, -0, 0)

[ INFO] [1634225071.227440603]: World frame->cartesian transform is Origin: (186279.12783282547025 5190003.9517953395
844 -505.85603196016035099)
Rotation (RPY): (0, -0, -3.1155607636369766489)
```

Figure 46 - Message informatif d'intégration des données GNSS

Un message informatif nous donne les coordonnées brutes reçues par le module puis on voit les données transformées en projection UTM.

## 7.5 Tests d'envoi d'un but GNSS

Deux tests d'envoi de gps\_goal, but gps, ont été réalisés avec un script open source est disponible sur [Github](#). Il est adaptable en fonction des besoins. Les tests n'ont jamais passés les erreurs d'attente de node. Les coordonnées inscrites en dur dans le code correspondent à un emplacement pris sur le [cadastre](#).

## 7.6 Positions absolues de références

N° de pastille	E_MN95	N_MN95	H_MN95	X_WGS84	Y_WGS84	Z_WGS84
4909	2540431.566	1181219.905	438.4601	4346401.963	507402.604	4625366.860
4911	2540459.792	1181274.677	435.2355	4346356.711	507425.190	4625402.211
6071	2540621.750	1181303.931	447.4133	4346323.902	507584.127	4625432.220
6170	2540621.753	1181302.435	448.6834	4346325.847	507584.373	4625432.121

Les données sont représentées en mètre.

## 7.7 Tests restants

Des tests manquent pour valider entièrement la navigation avec position GNSS. Notamment le déplacement du robot vers un but fourni par des coordonnées GNSS. Ci-dessous une liste des tests qui permettra de valider complètement l'intégration :

1. Tests sans robot avec position fix de la station de base connue et position du module Xsens connu afin d'observer la précision obtenue
2. Le même test qu'en 1 mais avec la notion de temporalité, pour observer la variation dans une durée de temps de 3h par exemple
3. Tests avec le robot d'envoi de but gps

## 8 Planification

Le chapitre actuel illustre la planification prévue de ce projet ainsi que les éléments réalisés et les problèmes rencontrés.

### 8.1 Planification initiale

[En annexe](#) se trouve la planification initiale du projet. Il y a une plage de 4 semaines qui n'ont pas été prise en compte pour la planification car un accord avec le professeur a été conclu afin de terminer les cours sans travail de Bachelor en sus.

### 8.2 Tâches réalisées

[En annexe](#) se trouve les tâches réalisées en fonction de la planification initiale. En rouge, ce sont les temps sous-estimés, c'est-à-dire ceux qui dépasse le temps prévu. Comme mentionné plus haut, il y a également une plage de 4 semaines où aucunes tâches n'a été réalisées, en accord avec le professeur qui encadre le projet.

Le journal de travail, [en annexe](#), permet de voir plus en détails les tâches effectuées pour la réalisation de ce projet.

Les tâches qui concernent la cartographie n'ont pratiquement pas été réalisée par manque de temps.

### 8.3 Problèmes rencontrés et imprévus

Le premier problème rencontré est survenu lors de l'ajout du module Xsens, de sa nouvelle antenne et de sa nouvelle configuration. Dans un premier temps le problème est survenu lors d'une tentative de lancement d'un nœud d'exploration du robot, la Raspberry Pi 4 redémarre sans raison apparente. Puis lors d'une tentative de compilation d'un package, la Raspberry Pi 4 redémarre à 21% de compilation. Après quelques jours de recherche et en pensant d'abord à un problème de surcharge CPU, l'ingénieur du REDS qui est également sur le projet global propose le problème de tension.

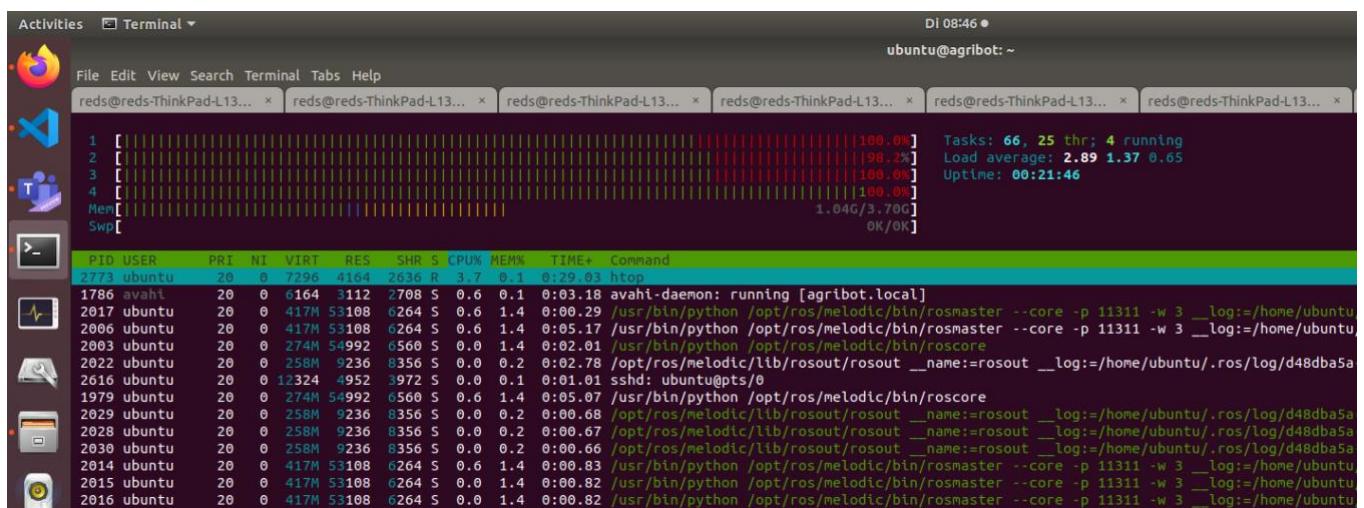


Figure 47 - Utilisation CPU lors de la compilation du paquet robot\_localization

Ci-dessus, un aperçu de htop lors de la compilation du paquet robot\_localization nécessaire à l'approximation de position.

En effet, l'ajout de la nouvelle antenne, ainsi que l'ajout des nouveaux paramètres du module affectent la consommation de la Raspberry Pi 4. La nouvelle antenne consomme effectivement un plus d'énergie que la précédente. Des essais avec le câble d'alimentation continu ont été réalisé et on n'observe plus de redémarrage intempestif ni de messages de sous tension. Pour pallier ce problème, deux nouvelles batteries ont été commandées afin de remplacer les anciennes, ces nouvelles batteries permettent de distribuer jusqu'à 3A, ce qui correspond à la capacité du câble d'alimentation.

Ci-contre, on observe les messages de sous tension qui apparaissent régulièrement si l'alimentation de la Raspberry pi n'est pas unique sur la source de tension. Même avec la nouvelle batterie, qui permet de délivrer plus de courant.

En utilisant une batterie uniquement pour la Raspberry pi on élimine le problème de sous tension et potentiellement des redémarrages intempestifs de la carte. Il faudra commander deux batteries supplémentaires car la batterie de secours est utilisée pour palier à ce problème.

```
ubuntu@agribot:~$ dmesg | grep -i volt
[    14.623528] Under-voltage detected! (0x000050005)
[   24.703513] Voltage normalised (0x00000000)
[   26.719491] Under-voltage detected! (0x000050005)
[   30.751450] Voltage normalised (0x00000000)
[   40.831461] Under-voltage detected! (0x000050005)
[   44.863473] Voltage normalised (0x00000000)
[   341.215465] Under-voltage detected! (0x000050005)
[  345.247475] Voltage normalised (0x00000000)
[  597.247454] Under-voltage detected! (0x000050005)
[  601.279455] Voltage normalised (0x00000000)
[  611.359479] Under-voltage detected! (0x000050005)
[  615.391457] Voltage normalised (0x00000000)
[  772.639472] Under-voltage detected! (0x000050005)
[  776.671459] Voltage normalised (0x00000000)
[  800.863447] Under-voltage detected! (0x000050005)
[  804.895462] Voltage normalised (0x00000000)
[  998.431460] Under-voltage detected! (0x000050005)
[ 1002.463462] Voltage normalised (0x00000000)
ubuntu@agribot:~$
```

Figure 48 - Message de sous tension

Lors des premières tentatives de tests en extérieur avec le robot, une tâche non prévue dans la planification est apparue. La communication avec le robot se fait soit par câble, en UART, soit sur un réseau WiFi. La communication UART est pratique si le robot est immobile lors des tests. Il a donc fallu réaliser un réseau WiFi mobile afin de garder une communication entre le pc de développement et le robot lors de tests en extérieur avec déplacement du robot. Un guide d'installation du Wifi est disponible en annexe.

Un problème matériel est survenu lors des tests en extérieur pour le système GNSS avec station de base. Les soudures du câble RS232 – RTCM lâchent. Dans un premier temps, n'étant pas à 100% sûr de la connectique à faire entre les fils du câble et la fiche RS232 mâle, les soudures n'étaient pas protégées. À la suite de cet incident, les soudures sont maintenant protégées par un boîtier en plastique.

## 9 Conclusion

### 9.1 Synthèse

Les objectifs de ce travail étaient la prise en main de l'environnement ROS, l'analyse des méthodes de positionnement en extérieur, le choix d'une méthode de positionnement, l'analyse d'une méthode de cartographie et le choix d'une méthode de cartographie. La validation de la précision devait être faite par des tests en extérieur.

État actuel du projet :

Le projet, dans l'état actuel, implémente une solution de positionnement. Des tests supplémentaires permettraient d'apporter davantage de mesures qui valide la précision actuelle. La validation n'a pas pu être effectuée au terme du projet.

La partie cartographie a été laissée de côté par manque de temps. Il serait éventuellement pertinent d'en faire un travail de Bachelor à part entière si elle se tourne effectivement vers une application complète.

### 9.2 Améliorations

Ce projet a de très bon potentiel d'améliorations ou d'ajout de fonctionnalités. Tout d'abord, au niveau matériel si on continue d'ajouter des fonctionnalités au robot je pense nécessaire de changer la Raspberry pi par une carte plus puissante ou de séparer les tâches. En réalisant certains tests on remarque que dès que la navigation est lancée les CPU sont vite sollicités.

Une application pourrait permettre de gérer la partie de l'affichage de la cartographie.

### 9.3 Conclusion personnelle

La réalisation de ce projet a été un vrai challenge. N'ayant aucune connaissance au préalable sur l'environnement ROS, j'ai passé beaucoup de temps à regards des tutoriels explicatifs sur le fonctionnement de ROS et de la navigation. Je suis contente d'avoir pu bénéficier de l'aide de l'ingénieur qui travaille également sur ce projet en parallèle.

La planification et la réalisation de ce projet ont été assez difficile surtout au niveau de l'estimation de temps nécessaire. J'ai finalement passé beaucoup de temps à tester des configurations et des variations de paramétrage alors que la partie design conceptuelle était plus légère que prévue.

J'aurai aimé pouvoir donner mon 100% à ce projet car c'est pour une cause qui me tient à cœur mais l'état actuel des choses fait que je n'ai pas pu faire mieux.

### 9.4 Remerciements

Je tiens tout d'abord à remercier Monsieur Etienne Messerli et Monsieur Romuald Mosqueron pour leur soutien durant cette dernière épreuve. Sans leur empathie je n'aurai probablement pas eu le courage de terminer ma formation.

Je souhaite également remercier tous les acteurs qui ont joué un rôle important au bon déroulement de ce projet : Monsieur Gabriel Arzur Catel Torres, Monsieur Sébastien Guillaume, Monsieur Marc-André Bovet, Monsieur Steve Maillard et Madame Mathilde Chammartin.

Et finalement, un grand merci à tous les élèves qui ont fait de ces 4 ans une formation pleine de rebondissements.

## 10 Glossaire

Abréviation - Expression	Description
<a href="#">CEP</a>	Circular Error Probability, erreur de probabilité circulaire, abréviation utilisée pour parler de marge d'erreur
<a href="#">DGPS</a>	<a href="#">GPS différentiel</a> , est une amélioration de la géolocalisation GNSS
<a href="#">EGNOS</a>	European Geostationnary Navigation Overlay Service, est un service de navigation géostationnaire additionnel qui permet d'obtenir des corrections sur la position GPS. Le service dispose de station de références un peu partout en Europe
<a href="#">GeoJSON</a>	<a href="#">Geographic JSON</a> est un format ouvert qui permet d'encoder des données géospatiales
<a href="#">GNSS</a>	Global Navigation Satellite System, Système global de navigation par satellite
<a href="#">KML</a>	<a href="#">Keyhole Markup Language</a> , est un langage destiné à la gestion de coordonnées GNSS
<a href="#">MN95</a>	Système géodésique de référence suisse
<a href="#">Mode Survey-In</a>	Le mode Survey-In permet à la station de base de ne pas forcément connaître le point sur lequel elle se trouve et de fournir des données de corrections en fonction de la position estimée sur un temps donné avec une précision donnée. Plus le temps est long et la précision grande plus vite la station de base sera en mesure de fournir les RTCM
<a href="#">NGS</a>	National Geodetic Survey, est une agence du gouvernement des États-Unis qui définit et gère le système de coordonnées nationales
<a href="#">RTCM</a>	Real Time Correction Message, Message de correction en temps réel. Cette abréviation correspond à un protocole de communication utilisé pour transmettre les messages de corrections entre une station de base et un récepteur mobile (robot)
<a href="#">RTK</a>	<a href="#">Real Time Kinematic</a> , est une méthode de géolocalisation, une méthode de DGPS
<a href="#">UTM</a>	Universal Transverse Mercator, est une projection cartographique de la surface de la terre
<a href="#">WGS84</a>	Système géodésique de référence mondial

## 11 Bibliographie

Travail de Bachelor de 2018 : [https://gitlab.com/REDS-tdb/2021-martin/-/blob/master/doc/anciens\\_tb/TB\\_richard\\_2018\\_rapport.pdf](https://gitlab.com/REDS-tdb/2021-martin/-/blob/master/doc/anciens_tb/TB_richard_2018_rapport.pdf)

Travail de Bachelor de 2019 : [https://gitlab.com/REDS-tdb/2021-martin/-/blob/master/doc/anciens\\_tb/TB\\_lemdjo\\_2019\\_rapport.pdf](https://gitlab.com/REDS-tdb/2021-martin/-/blob/master/doc/anciens_tb/TB_lemdjo_2019_rapport.pdf)

Travail de Bachelor de 2020 : [https://gitlab.com/REDS-tdb/2021-martin/-/blob/master/doc/anciens\\_tb/TB\\_CatelTorres\\_2020\\_rapport.pdf](https://gitlab.com/REDS-tdb/2021-martin/-/blob/master/doc/anciens_tb/TB_CatelTorres_2020_rapport.pdf)

Introduction aux systèmes GNSS : [https://gitlab.com/REDS-tdb/2021-martin/-/blob/master/doc/Intro\\_GNSS.pdf](https://gitlab.com/REDS-tdb/2021-martin/-/blob/master/doc/Intro_GNSS.pdf)

Introduction à ROS : <http://wiki.ros.org/ROS/Introduction>

Introduction à ROS : <https://homepages.laas.fr/ostasse/Teaching/ROS/rosintro.pdf>

Documentation sur la stack de navigation : <http://wiki.ros.org/navigation/Tutorials/RobotSetup>

Guichet cartographique cantonal : <https://www.geo.vd.ch>

Sources d'erreurs GNSS : <http://reperageterrestre.free.fr/erreurs.html>

EGNOS : [https://fr.wikipedia.org/wiki/European\\_Geostationary\\_Navigation\\_Overlay\\_Service](https://fr.wikipedia.org/wiki/European_Geostationary_Navigation_Overlay_Service)

Services et produits John Deere : <https://www.deere.com/en/technology-products/precision-ag-technology/>

Documentation du package robot\_localization :

[http://docs.ros.org/en/melodic/api/robot\\_localization/html/index.html](http://docs.ros.org/en/melodic/api/robot_localization/html/index.html)

Page wikipédia sur le langage KML : [https://fr.wikipedia.org/wiki/Keyhole\\_Markup\\_Language](https://fr.wikipedia.org/wiki/Keyhole_Markup_Language)

Plop

## 12 Table des figures

Figure 1 - Bandes du champ d'Agiez.....	0
Figure 2- Pompe à eau du champ d'Agiez .....	8
Figure 3 - Exemple de bande d'oignons avec distances .....	9
Figure 4 - Exemple de positionnement de plantons dans une bande.....	9
Figure 5- Prototype du robot - 2019 .....	11
Figure 6 - Prototype du robot - 2020.....	11
Figure 7 - 1er étage du robot .....	11
Figure 8 - 2ème étage du robot.....	12
Figure 9 - 3ème étage du robot.....	12
Figure 10 - Schéma explicatif du fonctionnement des nodes, des topics, des messages et des services .....	13
Figure 11 - Graphe généré par rqt_graph .....	14
Figure 12 - Schéma de la stack de navigation par ROS .....	15
Figure 13 - Exemple de solution avec photogrammétrie.....	17
Figure 14 - Exemple avec lasergrammétrie .....	18
Figure 15 - Schéma de solution GNSS .....	19
Figure 16 - Schéma d'une solution GNSS avec correction .....	21
Figure 17 - Starfire 6000 avec modem RTK par John Deere .....	22
Figure 18 - Module MTi-680G de Xsens, par Xsens.....	23
Figure 19 - Antenne GNSS multi bande, par ArduSimple .....	23
Figure 20 - Module Radio XBee avec antenne par arduSimple .....	23
Figure 21 - Adaptateur XBee - RS232 par arduSimple.....	23
Figure 22 - Carte simpleRTK2B par arduSimple .....	24
Figure 23 - Schéma d'une bande d'un champ avec la position des plantons .....	25
Figure 24 - Schéma d'un champ avec les coordonnées des différentes bandes .....	26
Figure 25 - Exemple de stockage de coordonnées au format KML.....	27
Figure 26 - Exemple de stockage de coordonnées au format GeoJSON .....	27
Figure 27 - Fusion des données de capteurs pour estimer la position du robot .....	28
Figure 28 - Définition du message Odometry .....	29
Figure 29 - Message de type nav_msgs/Odometry .....	29
Figure 30 - Définition du message NavSatFix .....	29
Figure 31 - Schéma de node avec messages pour l'intégration des données GNSS .....	30
Figure 32 - Schéma bloc du matériel à mettre place pour l'intégration du système GNSS avec RTK .....	30
Figure 33 - Aperçu de l'interface MT Manager avec les données de l'IMU affichées.....	31
Figure 34 - Extrait de la configuration du node de fusion IMU .....	32
Figure 35 - Configuration du node navsat_transforme.....	33
Figure 36 - Extrait de la configuration du node ekf GNSS .....	34
Figure 37 - Robot assemblé avec le système GNSS .....	35
Figure 38 - Station de base assemblée sur le terrain .....	36
Figure 39 - Module Xsens .....	37
Figure 40 - Schéma de l'antenne GNSS calibrée .....	37
Figure 41 - U-Center, interface de configuration et de statistiques .....	38
Figure 42 - MT Manager, interface.....	40
Figure 43 - Précision de positionnement de l'antenne .....	41
Figure 44 - Installation de tests .....	41
Figure 45 - Station de base pour les tests .....	42
Figure 46 - Message informatif d'intégration des données GNSS .....	43
Figure 47 - Utilisation CPU lors de la compilation du module robot_loc .....	45
Figure 48 - Message de sous tension .....	46

## 13 Annexes

### *Planification initiale - Image*

			Dépendance	Temps estimé [h]	8	10	11	14	15	16	17	18	19	20	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	Temps réalisé [h]	
<b>1.0</b>	<b>Analyse</b>			106																										106		
1.1	ROS			56																										56		
1.1.1	Prise en main			24				5	5	5	4	5																		24		
1.1.2	Package Navigation			16																										16		
1.1.3	Package GPS			16																										16		
1.2	Méthode de positionnement			18							8	5	5																	18		
1.3	Méthode de correction et communication			18								8	5	5																18		
1.4	Méthode de cartographie			14								6	4	4																14		
<b>2.0</b>	<b>Développement</b>			184																										184		
2.1	Navigation avec des coordonnées GPS			116																											116	
2.1.1	Design			54																16	16	10	6	6						54		
2.1.2	Intégration	1.1   1.2   1.3		30																10	8	8	4							30		
2.1.3	Tests & Correctifs			32																4	10	8	10							32		
2.2	Cartographie			68																											68	
2.2.1	Design			26																				8	8	10				26		
2.2.2	Intégration	1.4		18																				8	8	2				18		
2.2.3	Tests & Correctifs			24																				10	14					24		
<b>3.0</b>	<b>Documentation</b>			120																											120	
3.1	Rapport			96												4	4	2	8	4	4	4	4	4	2				8	26	26	96
3.2	Guides d'utilisation, installation			24																						4	10	10		24		
<b>4.0</b>	<b>Gestion de projet</b>			38																											38	
4.1	Meetings			30	1	2	1		1		1		1		1	1	1	2	1	2	2	2	2	2	2	2	2			30		
4.2	Plannification			8												2	2	4													8	
	<b>TOTAL</b>			448	1	2	1	5	6	5	5	8	6	23	16	14	14	17	16	28	32	28	28	28	30	30	36	36	448			

Les semaines non affichées sur l'image correspondent aux semaines sans travail. Le fichier original se trouve [ici](#).

Tâches réalisées - Image

			Dépendence	Temps estimé [h]	8	10	11	14	15	16	17	18	20	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	Temps réalisé [h]
<b>1.0</b>	<b>Analyse</b>			106																										101,5		
1.1	ROS			56																										67,5		
1.1.1	Prise en main			24	1,5	13,5		1,5	2								1												21,5			
1.1.2	Package Navigation			16														5	2	10	2			2	5	6			1	33		
1.1.3	Package GPS			16														4	2	4					3	5	1		13			
1.2	Méthode de positionnement			18										4	4	3	1	1			4	1			3	1			22			
1.3	Méthode de correction et communication			18									4							3				2	1				10			
1.4	Méthode de cartographie			14																									2			
<b>2.0</b>	<b>Développement</b>			184																										119		
2.1	Navigation avec des coordonnées GPS			116																										119		
2.1.1	Design			54														3	1	2	1	3	4	3	2	1			20			
2.1.2	Intégration	1.1   1.2   1.3		30														3	6	5	4	6	3	6	7	2	5		47			
2.1.3	Tests & Correctifs			32														2		2	8	4	10	7	5	14		52				
2.2	Cartographie			68																									0			
2.2.1	Design			26																									0			
2.2.2	Intégration	1.4		18																									0			
2.2.3	Tests & Correctifs			24																									0			
<b>3.0</b>	<b>Documentation</b>			120																									93			
3.1	Rapport			96										1		4	12	4	5	2	2	12	1	2	7	1	6	8	18	85		
3.2	Guides d'utilisation, installation			24														2							1	1	2	2	8			
<b>4.0</b>	<b>Gestion de projet</b>			38																									37,5			
4.1	Meetings			30	1	2	1	2,5	1		2	1				1	1	2	1	1	3	3		1	4	4		31,5				
4.2	Plannification			8										2			3	1										6				
<b>TOTAL</b>				<b>448</b>	1	3,5	1	13,5	2,5	0	2,5	2	3	7	8	7	20	7	21	16	12	12	25	13	28	25	28	19	19	33	22	<b>351</b>

Les semaines non affichées sur l'image correspondent aux semaines sans travail. Le fichier original se trouve [ici](#).

[Journal de travail](#)

[Guide d'installation du logiciel MT Manager sur linux](#)

[Fichier de configuration de Xsens](#)

[Guide d'installation du logiciel U-Center et configuration de la station de base](#)

[Mesures GNSS simples](#)

[Mesures GNSS avec corrections](#)

[Fichier de configuration de la station de base en mode Fixe](#)

[Script de conversion WGS84 – MN95](#)

[Script gps\\_goal](#)

[Autre](#)