

**Département** : Génie Electrique

**Filière** : Licence Professionnelle Electronique des  
Systèmes Embarqués

## Stage de fin d'études

Réalisé au sein de :

**Office Chérifien de Phosphates OCP**



La conduite déportée et autonome des camions de  
transport de phosphates

**Réalisé par :**

➤ Smail LOUKILI

**Encadré par :**

➤ Pr. M. GUERBAOUI (ESTM)

➤ Mr. A. FAHMI (OCP)

**Soutenu le 13/06/2023 devant le jury composé de :**

➤ Pr. M. GUERBAOUI

➤ Pr. S.BRI

**Année universitaire : 2022-2023**

## **Dédicace**

Je dédie ce travail à :

Ma mère Oumkeltoum,

La meilleure mère du monde. Merci pour tes prières, tes sacrifices, ton soutien moral et ta patience inépuisable. Je t'aime tellement et aucune dédicace ne pourrait suffire à exprimer tout ce que tu mérites.

Mon père Lhoucine,

Mon idole qui a fait tant de sacrifices pour me créer les conditions propices à la réussite. Je suis fier de vous. Aucune dédicace ne peut exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous. Merci pour tout.

Mes frères Brahim et Zakaria

Pour leur aide, leurs encouragements, leur assistance et leur soutien.

Mes amis Marouane Benfikri, Zakaria Bouchikhi, Anas Zeroual, Anas Saufi, Achraf Nourredin, Marouane Lamzirai et Ali Elidrissi.

Les compagnons fidèles dans les moments les plus délicats. Merci pour tous les moments de joie partagés et l'amour que vous avez pour moi. Je vous souhaite tout le bonheur et un avenir radieux.

Mes respectables professeurs,

Qui nous ont tant formés pour être à la hauteur de représenter notre honorable école.

Que chacun trouve dans ce travail le témoignage de mon amour sincère, de ma profonde gratitude et de ma reconnaissance.

## **Remerciement**

*C'est avec une profonde gratitude que j'écris ces mots pour exprimer ma reconnaissance à tous ceux qui m'ont soutenu tout au long de mon projet. Leur contribution a été essentielle à sa réalisation.*

*Je tiens à exprimer ma sincère reconnaissance à mon encadrant à l'École supérieure de technologie de Meknès, Monsieur **LGUERBAOUI** pour son encadrement, son soutien et son précieux conseils sur la démarche à suivre pour mener à bien ce travail, et monsieur **Hassan CHADLI** pour son expertise et la qualité de la formation dispensée dans notre filière, méritent bien plus qu'un simple "merci".*

*Je souhaite adresser mes remerciements les plus sincères à Monsieur **Abderahim FAHMI**, mon encadrant au sein du service électronique du groupe OCP. Son accueil chaleureux, le temps que nous avons passé ensemble et le partage de son expertise ont été précieux. Grâce à sa confiance, j'ai pu me réaliser pleinement dans mes missions. Il a été d'une aide précieuse dans les moments les plus délicats. Je suis reconnaissant pour ses efforts, ses directives et ses conseils.*

*Je tiens aussi à exprimer mes plus sincères remerciements et ma reconnaissance à Monsieur **Oussama LAAYATI** le manager d'innovation GTI dans l'UM6P pour son accompagnement tout au long de cette expérience. Ses conseils éclairés ont été d'une grande valeur et ont contribué de manière significative à la réussite de mon projet de fin d'études*

*J'exprime également ma gratitude envers tout le staff technique d'OCP et le staff de département GTI pour leur accueil chaleureux, leurs paroles aimables et leur comportement bienveillant.*

## Résumé

Ce projet de fin d'études se concentre sur la conduite autonome et déportée d'engins de transport de phosphate qui nécessitent une grande précision dans un environnement difficile. Un prototype de camion est utilisé pour réaliser des simulations en temps réel et étudier les différentes tâches qu'il peut effectuer.

Le prototype du camion doit être capable de percevoir les informations extérieures nécessaires à la conduite, de détecter les obstacles, de calculer avec précision sa position et de cartographier son environnement. À partir de ces informations, le camion autonome est en mesure de calculer les commandes à envoyer aux moteurs pour se déplacer dans son environnement et accomplir sa mission.

Dans les cas où le mode autonome ne peut pas être utilisé, un téléopérateur humain doit pouvoir prendre le contrôle de l'engin à distance. De plus, étant donné que les mesures des capteurs embarqués ne sont pas parfaites à 100%, il est nécessaire d'utiliser des techniques de filtrage pour obtenir une estimation précise des paramètres de localisation et garantir des commandes sûres.

## Abstract

## Sommaire

Introduction général .....	1
----------------------------	---

### **Première partie : Groupe OCP & pole mine GANTOUR**

#### **Chapitre 1 : Présentation générale de l'OCP**

1.1. Phosphate marocain .....	4
1.2. Evènements marquants .....	5
1.3. Fiche technique .....	6
1.4. Organigramme.....	7
1.5. Caractère juridique .....	8

#### **Chapitre 2 : présentation de pole mine Gantour (PMG)**

2.1. La direction d'exploitation de Pôle Mine Gantour.....	10
2.2. L'extraction à Ben Guérir.....	10
2.3. Situation géographique .....	11
2.4. Le gisement .....	11
2.5. Les étapes de l'extraction à Ben Guérir.....	13
2.5.1. Extraction .....	13
2.5.2. Traitement .....	16

#### **Chapitre 3 : les camions de transport de phosphates**

3.1. Généralités .....	18
3.2. Description générale de camion .....	18
3.3. Principe de fonctionnement .....	18
3.3.1. Partie de puissance .....	18
3.3.2. Parie de commande .....	19
3.4. conclusion de première partie .....	21

### **Deuxième partie : projet de camion autonome**

#### **Chapitre 4 : généralités sur les véhicules autonomes**

4.1. Description de système .....	23
4.2. Composantes nécessaires pour la conduite autonome .....	24
4.2.1. Les capteurs extéroceptif.. ..	24
4.2.2. Les capteurs proprioceptifs .....	26
4.3. Principe de fonctionnement .....	28
4.4. Contexte de projet .....	28

## **Chapitre 5: Robot operating system (ROS) & windows subsystem for Linux (WSL)**

5.1.	Introduction .....	30
5.2.	C'est quoi une distribution de ROS .....	30
5.3.	Principe de fonctionnement de ROS .....	31
5.4.	Les outils de simulation supportés par ROS .....	33
5.4.1.	GAZEBO .....	33
5.4.2.	Rviz .....	33
5.5.	Windows subsystem for Linux (WSL) .....	34
5.5.1.	Définition .....	34
5.5.2.	Ces avantages .....	34
5.5.3.	Distribution utilisée .....	35

## **Chapitre 6 : simulation d'un modèle Robot**

6.1.	Introduction .....	36
6.2.	Configuration de l'espace de travail (workspace) .....	36
6.3.	Mise en place du robot dans GAZEBO .....	38
6.3.1.	Edit model environnement .....	38
6.3.2.	URDF .....	39
6.3.3.	Ajout des plugins .....	41
6.3.4.	Simulation & contrôle de robot .....	41
6.4.	Gmapping .....	44
6.5.	Localization .....	45
6.6.	AMCL .....	45
6.7.	Navigation .....	46
6.7.1.	Navigation stack .....	46
6.8.	Conclusion de dextieme partie .....	48
	Conclusion général .....	49
	<b>Annexes</b> .....	50
	<b>Webographie</b> .....	62

## Liste des figures

<i>Figure 1 : carte des sites OCP au Maroc</i>	4
<i>Figure 2 : l'organigramme d'OCP</i>	8
<i>Figure 3 : 17km à l'Est de Ben guérir. -77km de Youssoufia. -190km de Casablanca. -70km</i>	11
<i>Figure 4 : les différentes constituions de couches</i>	12
<i>Figure 5 : processus général de l'industrie de phosphate à Ben guérir</i>	13
<i>Figure 6 : machine de sondage</i>	13
<i>Figure 7 : opération de sautage</i>	14
<i>Figure 8 : Dragline Marion 7500</i>	14
<i>Figure 9 : Bulldozers</i>	15
<i>Figure 10 : les camions de transport</i>	15
<i>Figure 11 : l'installation fixe de l'épierrage</i>	16
<i>Figure 12 : Stacker</i>	16
<i>Figure 13 : Roue-pelle</i>	16
<i>Figure 14 : Installation fixe de criblage</i>	17
<i>Figure 15 : Train de transport</i>	17
<i>Figure 16 : principal source de puissance</i>	18
<i>Figure 17 : circuit de puissance</i>	19
<i>Figure 18 : l'unité de commande</i>	20
<i>Figure 19 : l'unité de commande</i>	21
<i>Figure 20 : Configuration du capteur sur un véhicule autonome</i>	23
<i>Figure 21 : Vision globale de la conduite autonome</i>	24
<i>Figure 22 : camera stéréo</i>	25
<i>Figure 23 : LIDAR actif</i>	25
<i>Figure 24 : RADAR millimétriques</i>	26
<i>Figure 25 : Principe de fonctionnement d'un accéléromètre</i>	27
<i>Figure 26 : Gyroscope</i>	28
<i>Figure 27 : LOGO de ROS</i>	30



<i>Figure 28 : les distributions de ROS</i> .....	31
<i>Figure 29 : architecture de service</i> .....	32
<i>Figure 30 : LOGO de GAZEBO</i> .....	33
<i>Figure 31 : LOGO de Rviz</i> .....	34
<i>Figure 32 : LOGO de Ubuntu 22.04.LTS</i> .....	35
<i>Figure 33 : mind-map d'une workspace</i> .....	37
<i>Figure 34 : Les formes géométriques (GAZEBO)</i> .....	38
<i>Figure 35 : joints &amp; links (GAZEBO)</i> .....	39
<i>Figure 36 : structure d'une Link (ex : châssis)</i> .....	40
<i>Figure 37 : exemple d'un plugin de camera stéréo</i> .....	41
<i>Figure 38 : modèle de robot sur GAZEBO équipé par camera stéréo &amp; Lidar</i> .....	42
<i>Figure 39 : définition d'environnement devant le robot</i> .....	43
<i>Figure 40 : interface de commande de twist_keyboead</i> .....	43
<i>Figure 41 : gmapping sur Rviz</i> .....	44
<i>Figure 42 : nuage des points d'estimation de position de robot</i> .....	46
<i>Figure 43 : Les éléments de navigation pour un système ROS</i> .....	47
<i>Figure 44 : navigation de robot (Rviz)</i> .....	48

# Acronymes

**OCP :** office chérifien de phosphate

**PMG :** Pole mine Gantour

**Odom:** odometry

**GPS:** Global Positioning System

**IMU:** Inertial Measurement Unit

**LIDAR:** Light Detection And Ranging

**ROS:** Robotic operating system

**WSL:** windows subsystem for linux

**GAZEBO :** Simulateur 3D des robots

**RVIZ:** Système de visualisation 3D

**XMLRPC:** Extensible Markup Language Remote Procedure Call

**URDF :** Unified Robot Description Format

**SLAM:** Simultaneous Localization and Mapping

**AMCL:** Adaptive Monte Carlo Localization.

**YAML:** Yet Another Markup Language

## Introduction général

Le groupe OCP entreprend une transformation majeure en intégrant une vision digitale, soutenue par l'intelligence artificielle, le big data et l'analyse des données, au sein de ses processus industriels. Cette transformation inclut des avancées telles que la conduite autonome pour le contrôle à distance des machines et la maintenance prédictive. L'introduction de ces nouvelles technologies et outils innovants permettra au groupe de faciliter le travail, d'améliorer l'efficacité opérationnelle, d'optimiser les coûts et de proposer des produits de qualité compétitifs.

Dans ce contexte, l'industrie minière a ouvert la voie au développement des camions autonomes, non seulement pour leur efficacité opérationnelle, mais aussi pour la sécurité des opérateurs. Grâce à ces camions autonomes, les opérateurs peuvent désormais conduire à distance un engin et contrôler chaque paramètre de celui-ci, tout en surveillant leur environnement grâce à des capteurs embarqués.

Ce projet nous offre l'opportunité de combiner des connaissances théoriques et pratiques, en nous plongeant dans le monde de la robotique et de la conduite autonome, afin de les appliquer dans un contexte professionnel.

L'objectif principal de ce projet est de concevoir et de modéliser un robot autonome au sein de la mine Ben Guerir, une première pour le groupe OCP et même en Afrique. Nous cherchons à développer un robot mobile capable d'effectuer des tâches définies de manière autonome, tout en nécessitant un minimum d'aménagements et d'interventions extérieures. Pour répondre à ce besoin, plusieurs étapes ont été identifiées :

- ✚ Définition et mise en place d'un système permettant le déploiement du robot mobile autonome.
- ✚ Développement d'un prototype du robot mobile.
- ✚ Sélection des capteurs nécessaires à la réalisation du système et à l'équipement du robot.
- ✚ Développement de méthodes et d'algorithmes permettant au robot d'évoluer de manière autonome.
- ✚ Réalisation de différentes simulations pour valider la solution développée.

Ce projet marque une avancée significative dans le domaine de la conduite autonome et de la robotique, offrant de nouvelles perspectives pour l'industrie minière et pour le groupe OCP en particulier.



Première partie

Groupe OCP & pole mine GANTOUR

# Chapitre 1

## Présentation général d'OCP

Le Groupe OCP est une entreprise dynamique et efficace, dont la structure est comparable à celle des grandes entreprises privées internationales. La croissance de ses activités et l'importance de ses projets ont conduit à la construction d'une structure de groupe, permettant une complémentarité entre ses différentes entités. La mission du Groupe OCP est de produire et commercialiser du phosphate naturel ou sous forme dérivée, tels que l'acide phosphorique ou les engrais. En tant qu'opérateur international dans le secteur des phosphates et dérivés, le groupe joue un rôle clé dans le développement économique et social du Maroc, et livre ses produits sur les cinq continents. Les exportations du Groupe OCP représentent 25 à 30% du commerce mondial des phosphates et de leurs dérivés.

Il comprend :

- Centres d'exploitation Khouribga, Youssoufia et Benguerir
- Centres de transformation chimique Safi et Jorf Lasfar.
- Ports d'embarquement Casablanca, Safi, Jorf Lasfar et Laayoun

Le phosphate brut extrait du sous-sol marocain est exporté vers les industries chimiques du Groupe à Safi ou Jorf Lasfar pour être transformé en sous-produits commercialisables (acide phosphorique basique, acide phosphorique purifié, engrais solides).

Ainsi constitué, le Groupe « OCP » est constitué ; en plus du Bureau ; d'un ensemble de filiales travaillant pour mettre à la disposition de l'office les moyens nécessaires pour répondre à ses besoins :

- ✓ SOTREG : Société régionale de transport.
- ✓ SMESI : entreprise marocaine d'études spéciales et industrielles.
- ✓ IPSE : Institut de promotion socio éducatif.
- ✓ STAR : Compagnie de transport et d'affrètement ensemble.
- ✓ CERPHOS : Centre d'études et de recherches sur les phosphates minéraux.

✓ MARPHOCEAN : compagnie maritime.

✓ IMSA : société immobilière et hôtelière.

Ce qui donne au pays la position sur le marché mondial est le 1er exportateur de phosphate sous toutes ses formes, le 1er exportateur d'acide phosphorique et le 3ème producteur mondial... !

### 1.1. Phosphate marocain :

- ❖ Le groupe OCP (anciennement Office chérifien des phosphates), a été fondé au Maroc le 7 août 1920 et transformé en 2008 en société anonyme (OCP SA). C'est l'un des principaux exportateurs mondiaux de phosphate brut, d'acide phosphorique et d'engrais phosphatés.
- ❖ Le groupe OCP compte près de 24 000 collaborateurs répartis principalement au Maroc sur 4 sites miniers et 2 complexes chimiques, ainsi que sur d'autres sites internationaux.
- ❖ Le groupe dispose de plusieurs filiales à l'intérieur et à l'extérieur du Maroc.

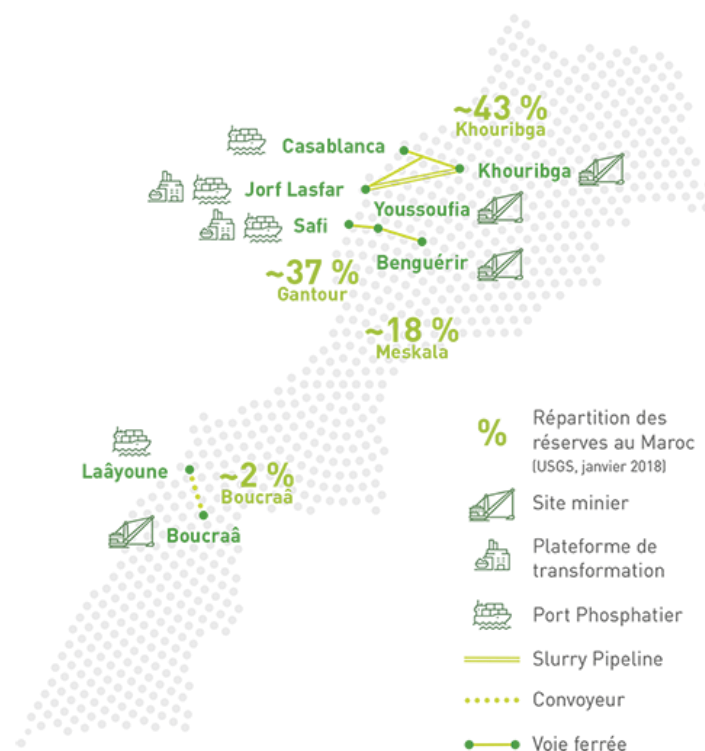


Figure 1 - carte des sites OCP au Maroc

Le Groupe OCP opère dans un modèle intégré qui couvre l'ensemble de la chaîne de valeur, allant de l'extraction des phosphates à la production et la commercialisation de différents produits tels que des engrais et de l'acide phosphorique. Cette entreprise joue un rôle clé sur le plan économique et social au Maroc, représentant près d'un quart des exportations du pays en 2010 et environ 3,5 % du PIB. Depuis 2007, OCP vise à doubler sa capacité de production annuelle de phosphates pour atteindre une capacité de production d'engrais d'environ 12 millions de tonnes d'ici 2017 et tripler sa production d'engrais d'ici 2020. En 2016, Fitch Ratings a confirmé la note « d'Investment Grade » pour le groupe OCP. Après avoir obtenu l'approbation de l'Autorité Marocaine du Marché des Capitaux le 9 décembre 2016, le groupe OCP a réussi à obtenir un emprunt obligataire de 10,2 milliards de dirhams auprès de la communauté nationale, ce qui représente le plus grand emprunt jamais réalisé sur le marché marocain.

## **1.2. Evènements marquants :**

- 1917 : Découverte des premières traces de phosphate dans la région d'Ouled Abdoun à Khouribga.
- 1920 : Création par Dahir de l'Office Chérifien des phosphates.
- 1921 : Démarrage de l'extraction souterraine à Khouribga et de l'activité à l'export via le port de Casablanca.
- 1931 : Démarrage de l'extraction souterraine à Youssoufia.
- 1951 : Lancement de l'extraction à ciel ouvert à Sidi Daoui dans la région de Khouribga.
- 1965 : Création de Maroc-Chimie à Safi entamant ainsi l'ère de la transformation et de la valorisation de phosphore.
- 1976 : Création de 1er centre R& D dédié au phosphate.
- 1980 : Lancement de la mine de Ben Guerir.
- 1984 : Démarrage de la plateforme de Jorf Lasfar pour la production d'énergie.
- 1990 : Ouverture d'un bureau de représentation en Inde A partir de 1996 Expansion à l'international avec la création de plusieurs joint-ventures :

- IMACID
  - ZMPL
  - EMAPHOS
  - PAKPHOS
  - ...
- 2007 : Création de la fondation OCP.
  - 2008 : L'Office Chérifien des Phosphates devient Société Anonyme, OCP SA, et démarre son programme de transformation industrielle.
  - 2010 : Création de JESA, une joint-venture avec Jacobs Engineering Inc., dédiée à l'ingénierie.
  - 2010-2011 : Démarrage de l'installation de plusieurs unités industrielles (laveries et STEP) et renforcement de la présence d'OCP à l'international avec l'ouverture de deux bureaux de représentation Brésil et en Argentine.
  - 2011 : Création des Skills centers, un vaste programme sociétal pour promotion de L'employabilité et l'entrepreneuriat.
  - 2012 : Création d'un bureau de représentation en Turquie et de BSFT (Black Sea Fertilisera Trading).
  - 2013 : Création de Dupont OCP Operations Consulting (DOOC), une joint-venture entre notre Groupe et la division Dupont Sustainable Solution (DSS) du groupe américain Dupont de Nemours.
  - 2014 : Démarrage du Slurry Pipeline sur l'axe Khouribga Jorf-Lasfar sur une longueur de 235 Km. Création de la Fondation Phosboucraà. Emission obligataire inaugurale l'OCP d'un montant de 1,85 milliards de dollars US en deux tranches d'une maturité de 10 ans et 30 ans.
  - 2015 : Deuxième émission obligataire d'OCP d'un montant de 1 milliards de dollars US d'une maturité de 10,5 ans. Ouverture du premier bureau de représentation d'OCP en Afrique subsaharienne à Abidjan (côte d'Ivoire) et d'un bureau à Singapour.



- 2025 : Terme du vaste programme d'investissement engagé par OCP qui devrait permettre d'atteindre une capacité de production de 50 millions de tonnes.

### **1.3. Fiche technique :**

- Raison social : Groupe Office Chérifien des Phosphates (OCP).
- Forme juridique : Etablissement semi-public.
- Date de création de l'OCP : Le 7 août 1920.
- Date de création du Groupe : 1975.
- Location et siège : Angle route d'El Jadida et Bd de la Grande Ceinture CASA.
- Conseil d'administration : Le premier ministre.
- Correspondant à l'étranger : Bureau OCP-Paris.
- Activités principales : Extraction, Traitement, valorisation et commercialisation des Phosphates.
- Sites d'exploitations minières : Khouribga, Youssoufia/ Ben Guérir, Boucraâ/ Laâyoune.
- Sites valorisation chimique : Safi et Jorf Lasfar.
- Port d'embarquement : Casablanca, Safi, Jorf Lasfar et Laâyoune.
- Effectif en personnel (2004) : 19.551 agents dont 850 ingénieurs et assimilés.
- Chiffres d'Affaires (2020) : 54,09 milliards de dirhams, en léger repli de 3 % par rapport à l'année précédente.

### **1.4. Organigramme :**

Les différentes organisations du groupe d'OCP sont citées comme suit :

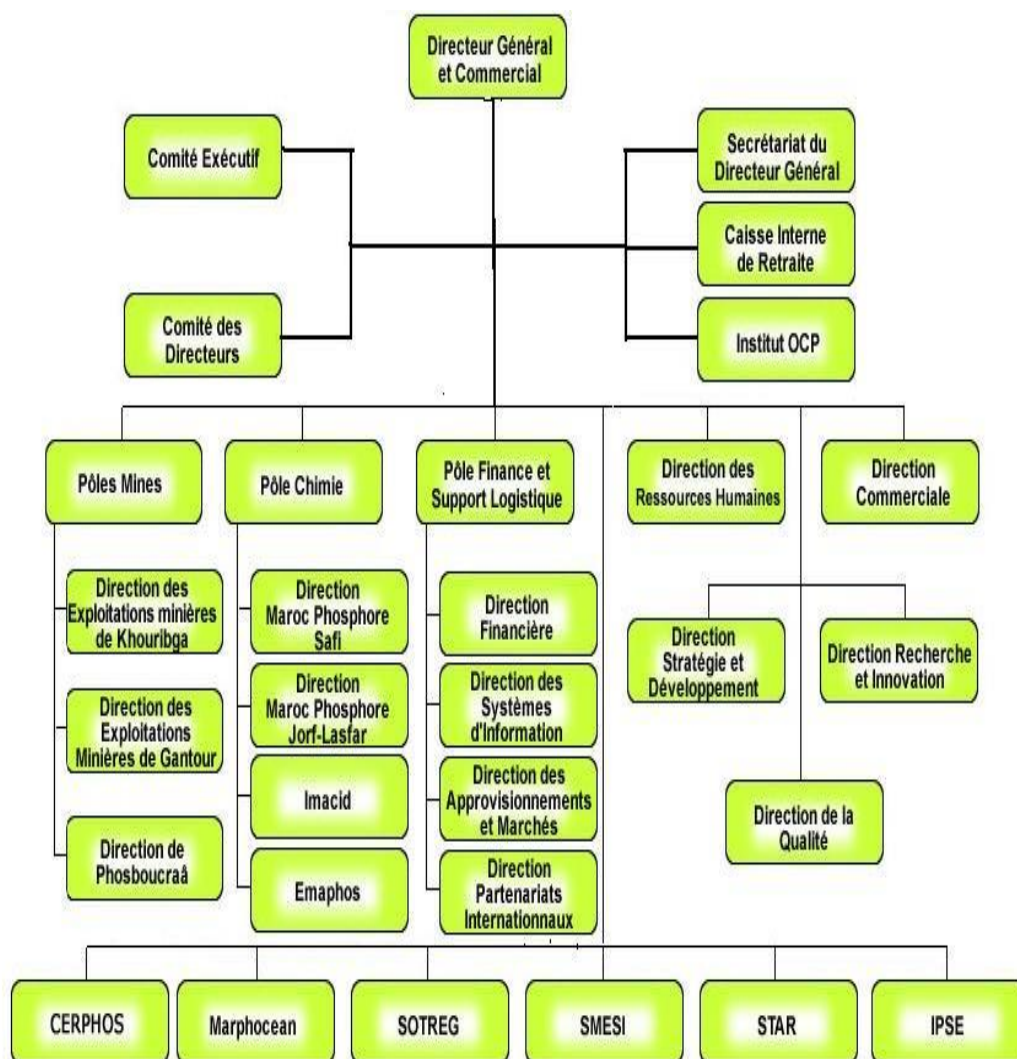


Figure 2 - l'organigramme d'OCP

### 1.5. Caractère juridique :

L'OCP a été constitué sous forme d'un organisme d'état, mais étant donné le caractère de ses activités commerciales et industrielles, le législateur a tenu à la doter d'une organisation, en lui permettant d'agir avec la même dynamique et la même souplesse que les grandes entreprises privées avec lesquelles se trouve en concurrence. L'OCP fonctionne ainsi comme une société dont le seul actionnaire est l'état marocain et est dirigé par un directeur général nommé par Dahir. Le contrôle est exercé par un conseil d'administration qui représente les intérêts de la nation et de l'état.

En ce qui concerne la gestion financière, elle est entièrement séparée de celle de l'état. Et tous les ans l'OCP établit son bilan, son compte d'exploitation de ses prix de revient comme toute entreprise privée.

L'OCP est inscrit au registre de commerce est soumis sur le plan fiscale aux mêmes obligations que n'importe quelle entreprise privée (patente, taxes à l'exploitation, impôts sur les salaires, impôts sur les bénéfices...). Chaque année l'OCP participe au budget de l'état par le versement de ses dividendes.

Son personnel est régi par un « statut du mineur du premier juillet 1964 » ce statut a été élaboré en conformité avec le Dahir 60-007 du 24 décembre 1960, portant statut du personnel des entreprises minières dernièrement refondu.

Les ingénieurs et assimilés (hors cadres) sont régis également par un statut particulier, récemment refondu les structures actuelles ont été définies par l'OS (organisation sociale) n°716 du premier janvier 1971.

## Chapitre 2

### Présentation de pole mine Gantour (PMG)

#### 2.1. La direction d'exploitation de Pôle Mine Gantour :

La mission de PMG est d'extraire, de traiter et de livrer les phosphates du gisement de Gantour. Ce gisement s'étend sur 125 km d'est en ouest, et sur 20 km du nord au sud, il couvre une superficie de 2500 km<sup>2</sup>, ses réserves sont estimées à environ 31 milliards de mètres cubes, soit 35% des réserves reconnues à l'échelle nationale. Deux centres fonctionnent :

- Le centre de Youssoufia (depuis 1931) : extraction sous terrain.
- Le centre de Ben guérir (depuis 1980) : extraction à ciel ouvert.

Le potentiel de production total est actuellement de 6,2 millions de tonnes de minerai par an, dont 3,7 millions de tonnes (60%) tamisées humides et 2,5 millions de tonnes (40%) de phosphates secs et marchands.

#### 2.2. L'extraction à Ben guérir :

La mine de Ben guérir est située dans la partie centrale du gisement Gantour, qui est d nature sédimentaire et consiste en une couche alternée de phosphate et intercouche. La série phosphatée, comme celle des autres zones du bassin du Gantour, prend la forme d'alternances de phosphates et de niveaux stériles. La section synthétique du gisement de Ben guérir permet de distinguer les différentes couches de phosphate ainsi que leurs intercalaires. Ce sont des niveaux identifiés, sur lesquels l'opération est basée pour faire une meilleure sélection. La production de Ben guérir est expédiée :

- A Safi pour un traitement humide avant sa récupération.
- A Youssoufia pour être utilisé en complément du phosphate noir.

La production du phosphate se fait aux étapes suivant :

- ❖ Foration
- ❖ Sautage

- ❖ Décapage
- ❖ Défruitage
- ❖ Transport
- ❖ Epierrage et criblage
- ❖ Chargement

### 2.3. Situation géographique :

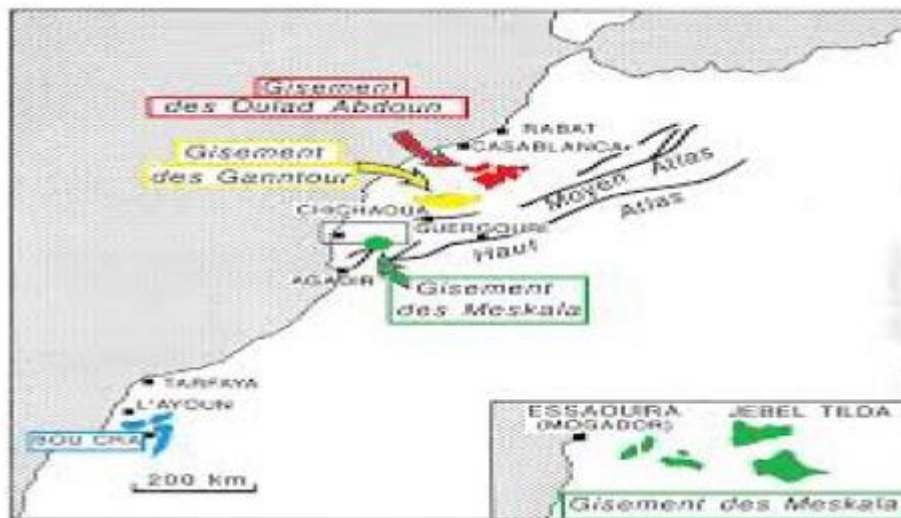


Figure 3 - 17km à l'Est de Ben guérir. -77km de Youssoufia. -190km de Casablanca. -70km

### 2.4. Le gisement (les couches):

L'analyse de la section lithologique du gisement permet d'identifier différentes constitutions de couches ainsi que leurs séparateurs :



*Figure 4 - les différentes constitutions de couches.*

**Couche 1 :** Elle est constituée d'un phosphate sableux oolithique.

Diviseurs C1 / C2 : Il se compose de billes siliceuses et de silex.

**Couche 2 :** Elle est formée de deux niveaux, la couche supérieure 2 et la couche inférieure 2.

Diviseurs C2 / C3 : Il est formé d'un niveau uniforme composé d'argiles jaunes et se termine vers le toit par des marnes. Les argiles constituent un excellent niveau de référence sur toute la couche du gisement.

**Couche 3 :** Elle se compose de sable fin de phosphate sur le toit et devient de plus en plus grossier et riche vers le faible.

Intercalaire C3 / C4 : Il se compose d'un complexe calcaire, de marnes siliceuses à blocs de silex, marne et argilo-calcaire.

**Couche 4 :** Il est formé d'un phosphate sableux grossier et détritique à la base, et il passe à un sable phosphate marneux et riche vers le toit.

Intercalaire C4 / C5 : Il est essentiellement constitué de marnes siliceuses siliceuses dont la totalité du gisement.

**Couche 5 :** formée d'un phosphate sableux avec des débris organiques marneux vers le bas.

**Couche 6 :** Elle est constituée de phosphate de sable grossier gris-beige avec des débris organiques. Il contient blocs de calcaire.



*Figure 5 - processus général de l'industrie de phosphate à Ben guérir*

## **2.5. Les étapes de l'extraction à Ben Guérir :**

### **2.5.1. Extraction :**

Il s'agit d'extraire le phosphate brut, des différentes zones minières où il se trouve soit en découverte (ciel ouvert) ou souterrain. Elle consiste à enlever le phosphate de la terre, et cela s'établit en quatre opérations : forage, sautage, décapage et défruite. Chargé de l'exploitation et de l'extraction de la mine de BEN GUERIR, il effectue les opérations suivantes :

#### **❖ Foration :**

La foration constitue la première opération minière de la chaîne d'extraction. Elle consiste à creuser des trous jusqu'au niveau repère au toit de la couche à exploiter. Cette opération est effectuée par des machines de la foration (sondeuses) : électriques sur chenilles (45R), diesel sur pneus (T4BH) et diesel sur chenilles (SK60).



*Figure 6 - machine de sondage.*



### ❖ Sautage :

C'est l'opération qui consiste à fragmenter les terrains durs à l'aide d'un explosif (ammonix composé de 94% de  $\text{NH}_4\text{NO}_3$  et 6% de fuel). L'opération consiste à remplir les trous forés par l'explosif en tenant compte d'un schéma de tir approprié.



*Figure 7 – opération de sautage.*

### ❖ Decapage :

Cette étape consiste à enlever le terrain mort pour accéder aux couches de phosphate. Cette opération se fait à l'aide de machines draglines de types : P&H et 7500M.



*Figure 8 – Dragline Marion 7500.*



### ❖ **Defruidage :**

C'est la phase de récupération du phosphate. Après avoir constitué des tas de phosphates au chantier, on procède au chargement des chargeuses afin de pouvoir transporter le phosphate vers les installations fixes.



*Figure 9 – Bulldozers.*

### ❖ **Transport :**

C'est une étape qui consiste à acheminer le phosphate issu du chantier d'extraction vers les installations fixes dans l'objectif de subir 3 traitements essentiels à savoir : épierrage, criblage et chargement visant à réduire la quantité de stériles et à assurer la qualité requise par un mélange adéquat des différentes qualités de phosphate.



*Figure 10 – les camions de transport.*

## 5.2. Traitement :

### ❖ Epierrage :

Après le transport des phosphates vers les stations mécaniques appelées installations fixes, il est versé dans deux trémies qui alimentent deux cribles de maille 90\*90 mm destinés à l'élimination des grosses pierres.



*Figure 11 – l'installation fixe de l'épierrage.*

### ❖ Stockage :

Le stockage est assuré par deux machines appelées Stockeuses. Le phosphate stocké par la machine stockeuse est repris par une machine appelée Roue-pelle.



*Figure 12 – Stocker.*



*Figure 13 – Roue-pelle.*

### ❖ Criblage :

Cette étape consiste à faire passer le phosphate sur des cribles de mailles de 10x10mm. Le produit criblé est acheminé par des convoyeurs vers un deuxième parc de stockage pour être homogénéisé afin d'obtenir une qualité bien définie. Le phosphate issu de cette station est donc stocké dans des zones spécifiques selon la qualité du produit.



*Figure 14 – Installation fixe de criblage.*

### ❖ Transport :

Le phosphate criblé sera transporté par les trains vers Safi pour un traitement humide avant sa valorisation ou vers Youssoufia pour être utilisée comme appoint au phosphate noir.



*Figure 15 – Train de transport.*

## Chapitre 3

### Les camions de transport de phosphate

#### 3.1. Généralités :

Les camions TEREX, Komatsu, HP et Unit rig sont d'énormes véhicules capables de transporter des quantités massives de terre, allant de 134 à 240 tonnes. Ils font partie du système de transport intelligent géré par des technologies de pointe telles que "IN VERTEX" pour TEREX et "STATEX III" pour les autres, qui permettent d'effectuer plusieurs opérations rapidement tout en offrant une meilleure protection. Sur le site de Gantour, il y a un total de 15 pistes, et leur rôle principal est de transporter le phosphate depuis la phase de décapage jusqu'à son enlèvement final après l'installation de la pierre.

#### 3.2. Description générale de camion :

Unit Rig est de la famille des camions de transport du phosphate de la zone d'extraction vers les trémies, de capacité de 136 tonnes. Il est basé sur la transformation de l'énergie chimique (gasoil + air) en énergie mécanique par un moteur diesel et à travers un système électrique qui transforme cette énergie mécanique en énergie électrique, qui se retransforme par la suite en énergie mécanique par des moteurs de roues. Le système de traction diesel-électrique a l'avantage d'avoir un rendement supérieur à celui obtenu avec le système conventionnel. Ce camion est géré par un système électronique « STATEXIII » de haute technologie qui assure la commande, le contrôle, l'asservissement et la protection.

#### 3.3. Principe de fonctionnement :

Les camions se composent des éléments comme suit :

##### 3.3.1. La partie de puissance :

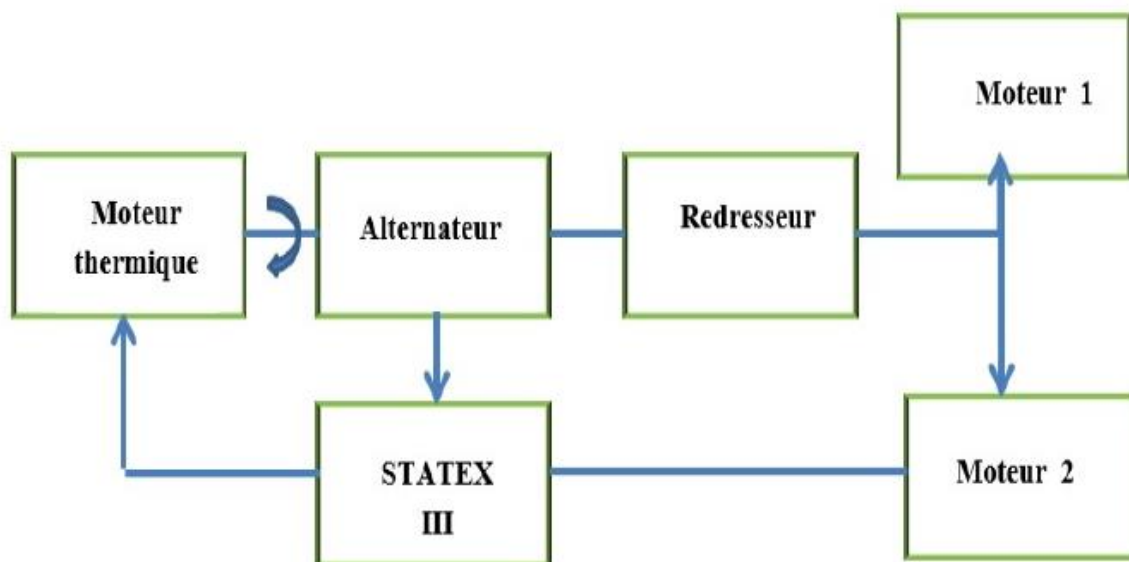


Figure 16 – principal source de puissance.



## Schémas de principe :

Le schéma général de l'installation est le suivant :

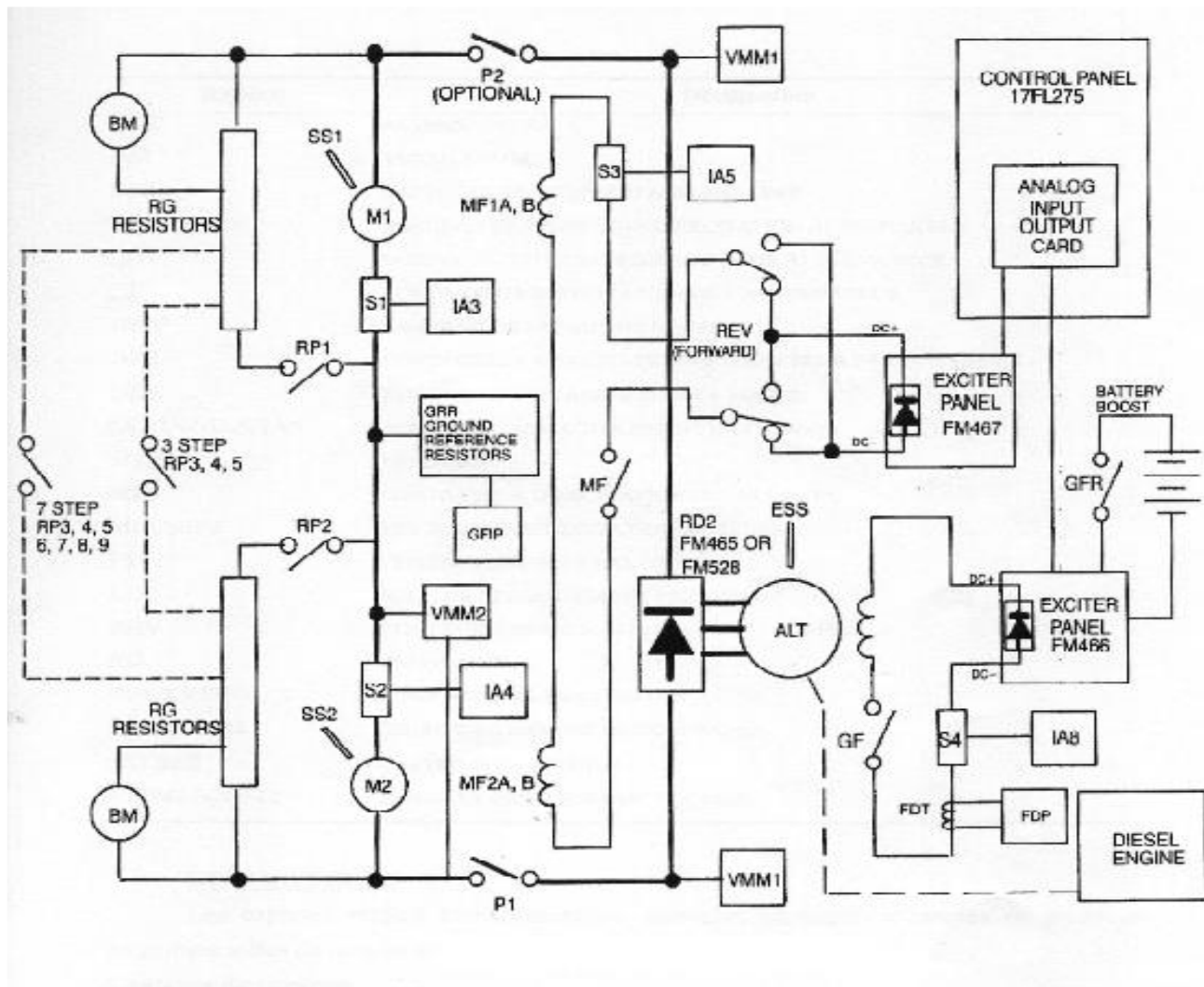


Figure 17 – circuit de puissance.

### 3.3.2. La partie de commande :

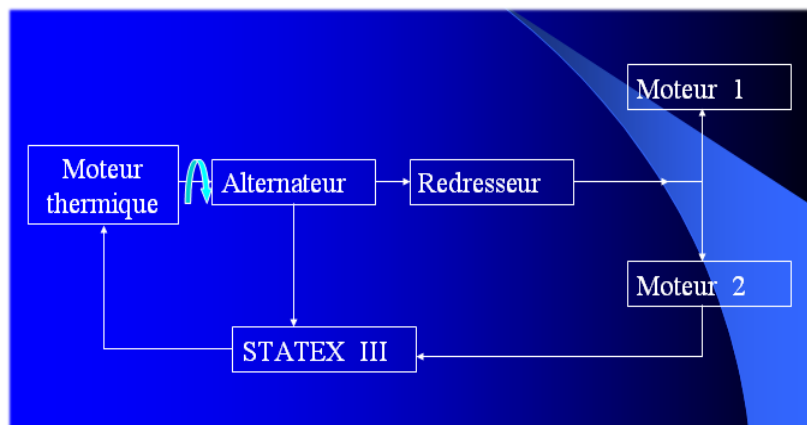


Figure 18 – l'unité de commande.

#### ❖ Le Statex III :

Le panneau des cartes électroniques STATEX III contenant le microprocesseur qui est le cerveau du système de contrôle qui permet, comme un petit ordinateur de bord, de surveiller la contribution de tous les signaux et établit en conséquence les commandes et les actions nécessaires pour :

- Le circuit de propulsion en stimulant les contacteurs pour propulser les roues motorisées.
- Le circuit de freinage en stimulant les contacteurs qui relient les grilles des résistances dans le circuit moteur. La gamme de freinage est réglée automatiquement par action séquentielle sur des contacteurs.
- La fourniture des limites des courants spécifiques à contrôler et à entretenir dans les deux modes de marche (propulsion et freinage).
- Le contrôle de vitesse en freinage par la régulation automatique de vitesse sur longue descente au chantier.
- Le contrôle de survitesse pour limiter les excès de déplacement.
- Les protections de l'alternateur, du moteur thermique et de l'excès de courant des roues motorisées.
- L'affichage des défauts, de l'état du système et des informations d'événements à l'opérateur.
- La fourniture des statistiques et historiques de données. L'historique est accessible en utilisant un PC portable.



*Figure 19 – l'unité de commande.*

### **Système de combustion CENTRY :**

Le CENTRY joue le rôle de régulateur de carburant. Il se compose d'un sous-système électronique qui contrôle l'alimentation en carburant à l'aide d'une électrovanne à commande électronique.

### **Le système « flight system » :**

Ce système électronique permet la détection des défauts et fournit des signaux à Statex III pour protéger l'installation des camions.

### **3.4. Conclusion de première partie :**

Dans le premier chapitre, nous avons examiné une introduction à l'Office Chérifien des Phosphates (OCP), avec un aperçu général sur le phosphate marocain, y compris son histoire, ses dates clés et ses caractéristiques principales des principaux bassins au Maroc, entre autres. Ensuite, dans le deuxième chapitre, nous avons présenté en détail le pôle Gantour, en discutant des différentes phases de production et de transport qui y sont associées. Nous avons également examiné les différents types de camions qui y sont utilisés dans le troisième chapitre, en mettant en évidence le camion de type statex III comme un camion d'esser, et en nous intéressant ensuite à sa description et à son principe de fonctionnement, y compris sa puissance et son système de commande.



## Deuxième partie : Projet de camion autonome



## Chapitre 4

### Généralité sur les véhicules autonomes

Les véhicules autonomes sont des véhicules capables de se déplacer sans intervention humaine directe, grâce à l'utilisation de capteurs, de caméras, de lidar, de radar et d'intelligence artificielle. Leur principal objectif est d'améliorer la sécurité routière en éliminant les erreurs humaines à l'origine de la plupart des accidents de la route. Ils offrent également des avantages tels que la réduction des embouteillages, une meilleure efficacité énergétique et une optimisation de la circulation. Cependant, leur adoption généralisée soulève des questions sur la responsabilité en cas d'accident, la sécurité des données, la réglementation et l'acceptation sociale. Malgré ces défis, de nombreuses entreprises investissent dans la recherche et le développement des véhicules autonomes, et des tests sont déjà en cours sur les routes. Les véhicules autonomes représentent une avancée technologique majeure dans le domaine des transports, avec le potentiel de révolutionner notre façon de nous déplacer.

#### 4.1. Description du système :

Le concept des véhicules autonomes vise à développer et produire un véhicule pouvant réellement circuler sur la voie publique dans le trafic sans intervention humaine en toutes situations.

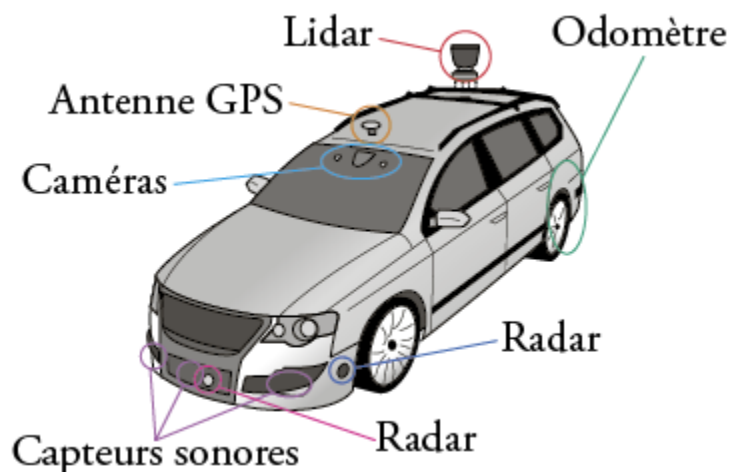


Figure 20 – Configuration du capteur sur un véhicule autonome.

## 4.2. Composants nécessaires pour la conduite autonome :

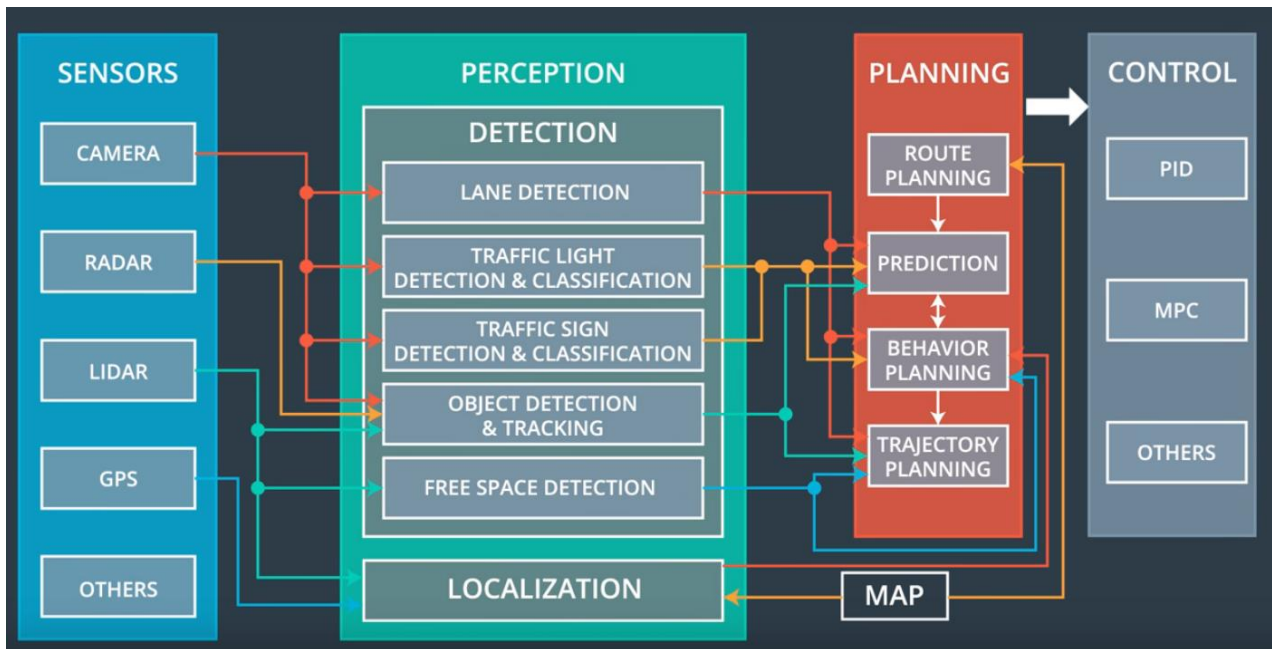


Figure 21 – Vision globale de la conduite autonome.

La première étape vers la conduite autonome d'un véhicule consiste à l'équiper d'un ensemble varié de capteurs. Ces capteurs comprennent des caméras, des radars, des lidars (capteurs laser) et des capteurs ultrasons. Ils jouent le rôle des "yeux" de la voiture en collectant en permanence des informations extérieures essentielles telles que le trafic routier et les panneaux de signalisation. Pour garantir une autonomie fiable, tous ces capteurs sont redondants, et les informations provenant de différents types de capteurs complètent mutuellement.

Il existe deux types de capteurs utilisés : **les capteurs proprioceptifs**, qui mesurent l'état du véhicule lui-même, et **les capteurs extéroceptifs**, qui mesurent l'état de l'environnement.

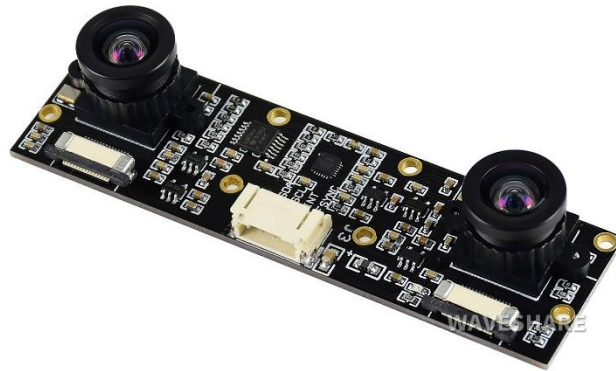
### 4.2.1. Les capteurs extéroceptifs :

Les capteurs extéroceptifs sont conçus pour recueillir des informations sur le monde extérieur à proximité du véhicule. Ils permettent d'obtenir des données spécifiques sur la position du robot dans son environnement en détectant les objets qui l'entourent. Ces informations peuvent prendre diverses formes.

#### ❖ Camera :

Il s'agit d'un dispositif électronique dont la fonction principale est de capturer des images fixes et/ou des séquences vidéo, et qui peut être utilisé pour la conduite autonome. L'idée fondamentale est d'enregistrer une scène à partir de deux angles de vue différents et d'utiliser la disparité entre ces deux vues pour déterminer la position, la relation et la structure des objets présents dans la scène. En comparant les positions des pixels dans les deux images, on peut obtenir une estimation de la profondeur. Lorsque l'objet se trouve dans la zone de

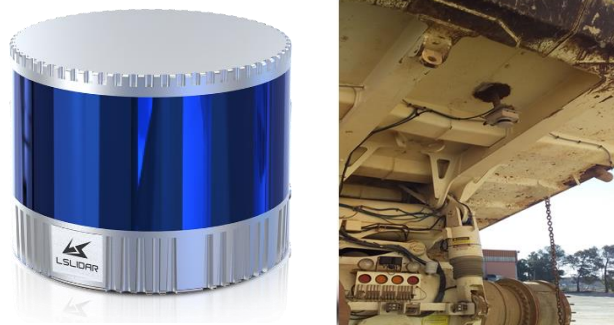
recouvrement des deux caméras, on peut mesurer sa distance. Cette méthode est connue sous le nom de vision stéréo.



*Figure 22 – camera stéréo.*

#### ❖ LIDAR (Light Detection And Ranging) :

Il s'agit d'un système de sonar utilisant des ondes laser pulsées pour cartographier la distance entre un véhicule et les objets environnants. La distance est déterminée en mesurant le délai entre l'émission d'une impulsion laser et la détection du signal réfléchi. Ce type de sonar est largement utilisé par les véhicules autonomes pour naviguer dans divers environnements. Ses principaux avantages résident dans sa capacité à fournir une perception précise de la profondeur, permettant ainsi au véhicule de connaître la distance d'un objet avec une précision allant jusqu'à quelques centimètres, sur une plage allant jusqu'à 60 mètres.



*Figure 23 – LIDAR actif.*

#### ❖ RADAR (Radio Detection And Ranging) :

La technologie radar est utilisée dans les voitures autonomes pour détecter et suivre les objets qui se trouvent à proximité de la voiture. Le système radar émet des ondes radio qui rebondissent sur les objets et reviennent vers le capteur radar. En mesurant le temps de retour des ondes, le système radar peut calculer la distance entre la voiture et l'objet. En outre, le système radar peut déterminer la vitesse et la direction du mouvement de l'objet.



*Figure 24 – RADAR millimétriques.*

#### **4.2.2. Capteurs proprioceptifs :**

Les capteurs proprioceptifs sont utilisés conjointement avec un ou plusieurs capteurs extéroceptifs. Leur rôle est de fournir des informations spécifiques sur les mouvements du robot. Toutefois, ces capteurs ne permettent pas de déterminer la position initiale du robot lorsqu'il débute son fonctionnement. De plus, les mesures qu'ils fournissent sont des mesures de vitesse (dans le cas des encodeurs) ou d'accélération (dans le cas des capteurs inertiels) qui nécessitent une intégration une ou deux fois afin de déterminer la position. Parmi ces capteurs qui sont largement utilisés :

##### **❖ GPS (Global Positioning System) :**

Un système de navigation GPS est un système qui utilise des satellites en orbite pour déterminer avec précision la position d'un appareil sur Terre. Il est largement utilisé dans les véhicules, les téléphones portables, les montres intelligentes et d'autres appareils pour fournir des indications de localisation, des itinéraires et des informations sur les déplacements.

Le système GPS fonctionne en utilisant un réseau de satellites en orbite autour de la Terre. Ces satellites envoient des signaux radio qui sont captés par des récepteurs GPS dans les appareils. En utilisant la triangulation, les récepteurs GPS déterminent la distance entre l'appareil et plusieurs satellites, ce qui permet de calculer la position exacte de l'appareil.

Grâce à ces informations de positionnement précises, un système de navigation GPS peut fournir des indications détaillées pour se rendre à une destination spécifique. Il peut également afficher des cartes, des points d'intérêt, des informations sur le trafic en temps réel et d'autres fonctionnalités utiles pour la navigation.

Les systèmes de navigation GPS sont devenus extrêmement populaires et sont intégrés dans de nombreux appareils électroniques courants. Ils ont simplifié la navigation en fournissant des itinéraires précis et en aidant les utilisateurs à trouver leur chemin dans des endroits inconnus.

### ❖ L'unité de mesure inertielle :

L'IMU (Inertial Measurement Unit) est un capteur qui combine un accéléromètre, un gyroscope et un magnétomètre pour mesurer l'orientation, la vitesse et les forces gravitationnelles. Il est conçu pour être monté de manière fixe dans la carrosserie d'une voiture autonome afin de déterminer sa position ancrée. En étant fixé à cet emplacement, l'IMU peut suivre les mouvements et la position de la voiture. Grâce aux avancées considérables dans les technologies MEMS (Micro Electro Mechanical Systems), les IMU ont été miniaturisés et sont désormais utilisés dans de nombreuses applications.

#### ➤ Accéléromètre :

Lorsqu'il est intégré dans une unité de mesure inertielle, l'accéléromètre est utilisé pour mesurer l'accélération d'un objet immobile dans le référentiel du capteur. Lorsque ce capteur est au repos à la surface de la Terre, il enregistre une accélération de  $9,81 \text{ m/s}^2$ , ce qui correspond à l'accélération due à la gravité. Le principe physique sur lequel repose ce capteur est assez simple. Une masse, généralement appelée "masse sismique" ou "masse de preuve", est soutenue par un ressort  $C$ . Un amortisseur visqueux  $b$  permet un amortissement proportionnel à la vitesse du capteur et à la masse de test. Lorsque le capteur est soumis à une accélération, la masse se déplace, et ce déplacement est mesuré afin de déterminer l'accélération.

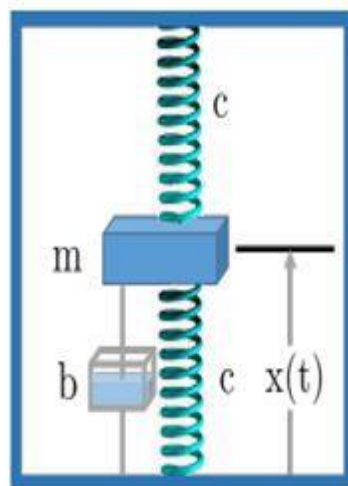
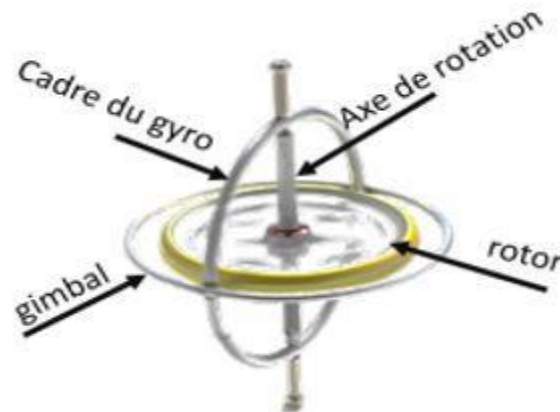


Figure 25 – Principe de fonctionnement d'un accéléromètre

#### ➤ Gyroscope :

On distingue principalement deux catégories de gyroscopes : les gyroscopes mécaniques et les gyroscopes optiques. Les gyroscopes mécaniques sont généralement composés d'une roue ou d'un disque en rotation sur un axe. Lorsque le disque tourne, il génère une résistance

aux changements de rotation. En revanche, les gyroscopes optiques utilisent un procédé différent. Ils se basent sur l'utilisation de deux faisceaux laser qui sont transmis à travers des fibres optiques et effectuent des rotations opposées. Selon l'effet Sagnac, le faisceau se déplaçant dans le sens de rotation aura une fréquence plus élevée. La différence de fréquence entre les deux faisceaux est directement proportionnelle à la vitesse angulaire. Ce type de gyroscope offre une précision extrêmement élevée et présente une insensibilité aux chocs et aux vibrations.



*Figure 26 – Gyroscope.*

#### **4.3. Principe de fonctionnement :**

Le contrôle du véhicule implique de manœuvrer le véhicule en utilisant différents actionneurs tels que le volant, le frein et l'accélérateur afin de suivre une trajectoire de référence. Cette opération est rendue possible grâce aux données captées par des dispositifs tels que le GPS, l'IMU, le LIDAR, etc. En combinant toutes ces informations collectées par les capteurs, un calculateur central est en mesure de corriger les erreurs latérales et longitudinales par rapport à la trajectoire de référence. Il crée ainsi une zone de sécurité et évalue l'environnement avec une précision remarquable.

#### **4.4. Contexte de projet :**

Dans les vastes environnements de travail tels que les mines, les chauffeurs de poids lourds sont exposés à divers risques professionnels lorsqu'ils transportent des marchandises sur la route. Ces risques incluent les accidents de la route, les problèmes liés à la manutention lors du chargement et du déchargement, ainsi que lors du bâchage et du débâchage. Ils peuvent également faire face au danger de chutes en descendant de la cabine, aux problèmes de santé liés aux vibrations et à la position assise prolongée, ainsi qu'aux affections psychosomatiques résultant du stress lié aux contraintes de temps, de sécurité et de risque de vol. De plus, ils peuvent être affectés par l'inhalation d'hydrocarbures.

Cependant, toutes ces raisons, ainsi que d'autres, peuvent être regroupées en quatre principaux arguments favorisant le développement d'un nouveau mode de transport :

1. Faciliter les déplacements pour tous.
2. Réduire les embouteillages et le trafic.
3. Limiter les impacts environnementaux de la circulation.
4. Améliorer la sécurité routière pour les êtres humains et les véhicules.

Il est important de souligner qu'environ 90% des accidents sont causés par des erreurs humaines telles que la fatigue, l'alcool ou des erreurs de conduite. En revanche, les véhicules autonomes ne se fatiguent pas, ne consomment pas d'alcool et semblent être plus performants que les conducteurs humains. Des études scientifiques ont démontré que ce type de transport pourrait réduire le nombre d'accidents.



## Chapitre 5

# Robot operating system (ROS) & Windows Subsystem for Linux (WSL)

### 5.1. Introduction :

Le but de cette partie, consiste à réaliser les algorithmes d'autoguidage de prototype par GPS et IMU avec détection des obstacles. A l'aide de Python et ROS, (Robot Operating System) sous linux (Ubuntu) développons les programmes de commande déportée et autonome du prototype.

ROS (Robot Operating System) est un cadre de développement open-source largement utilisé pour la programmation de robots. Il fournit une plateforme flexible et modulaire qui permet aux développeurs de créer des systèmes robotiques complexes. ROS offre un large éventail d'outils, de bibliothèques et de fonctionnalités pour la perception, la planification, le contrôle et la communication des robots.

ROS est conçu pour être indépendant de la plateforme matérielle et du système d'exploitation sous-jacent. Cela signifie qu'il peut être utilisé sur différents types de robots, qu'ils soient terrestres, aériens, marins ou autres, et qu'il peut être exécuté sur différents systèmes d'exploitation tels que Linux, macOS et Windows.



*Figure 27 – LOGO de ROS.*

### 5.2. C'est quoi une distribution de ROS?

Une distribution de ROS (Robot Operating System) est une version spécifique du système d'exploitation pour robots ROS, il existe plusieurs distributions de ROS, chaque distribution étant associée à une version spécifique de ROS. Chaque distribution inclut un ensemble cohérent de packages logiciels, de bibliothèques et d'outils qui fonctionnent ensemble pour offrir des fonctionnalités spécifiques. Les distributions de ROS sont généralement nommées d'après des animaux et suivent une séquence alphabétique.

Les distributions de ROS les plus courantes incluent :



1. ROS Indigo Igloo
2. ROS Kinetic Kame
3. ROS Lunar Loggerhead
4. ROS Melodic Morenia
5. ROS Noetic Ninjemys

**NB :** Chaque distribution de ROS est accompagnée de sa propre documentation, de guides de démarrage et de tutoriels pour aider les développeurs à utiliser les fonctionnalités spécifiques de cette distribution. Les utilisateurs peuvent choisir la distribution appropriée en fonction de leurs besoins, de la compatibilité matérielle et des exigences de leur projet robotique.















Distro	Release date	Poster	Turtle, turtle in tutorial	EOL date
ROS Noetic Ninjemys (Recommended)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013			May, 2015

Figure 28 – les distributions de ROS.

### 5.3. Principe de fonctionnement du ROS:

ROS est basé sur une architecture distribuée. Il repose sur un système de nœuds (nodes) qui communiquent entre eux via des messages. Chaque nœud effectue une tâche spécifique, comme la perception, la planification ou le contrôle du robot.

Nous décrivons ci-dessous les concepts regroupés dans ROS :

#### ❖ Les nœuds :

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... Chaque nœud qui se lance se déclare au **Master**. On retrouve ici l'architecture microkernel où chaque ressource est un nœud indépendant. Le système de nœuds permet de standardiser des fonctions basiques, et ainsi de développer rapidement des briques technologiques qui pourront être réutilisées, modifiées ou améliorées facilement.

#### ❖ Le master :

Le Master est un service de déclaration et d'enregistrement des nœuds qui permet ainsi à des nœuds de se connaître et d'échanger de l'information. Le Master est implémenté via XMLRPC. Le Master comprend une sous-partie très utilisée qui est le Parameter Server. Celui-ci, également implémenté sous forme de XMLRPC, comme son nom l'indique est une

sorte de base de données centralisée dans laquelle les nœuds peuvent stocker de l'information et ainsi partager des paramètres globaux.

### ❖ Les topics :

L'échange de l'information s'effectue soit de manière asynchrone via un topic ou de manière synchrone via un service. Un topic est un système de transport de l'information basé sur le système de l'abonnement / publication (subscribe / publish). Un ou plusieurs nœuds pourront publier de l'information sur un topic et un ou plusieurs nœuds pourront lire l'information sur ce topic. Le topic est en quelque sorte un bus d'information asynchrone un peu comme un flux RSS. Cette notion de bus many-to-many asynchrone est essentielle dans le cas d'un système distribué.

Le topic est typé, c'est-à-dire que le type d'information qui est publiée (le message) est toujours structuré de la même manière. Les nœuds envoient ou reçoivent des messages sur des topics.

### ❖ Les messages :

Un message est une structure de donnée composite. Un message est composé d'une combinaison de types primitifs (chaines de caractères, booléens, entiers, flottants...) et de message (le message est une structure récursive). Par exemple un nœud représentant un servomoteur du robot, publiera certainement son état sur un topic (selon ce que vous aurez programmé) avec un message contenant par exemple un entier représentant la position du moteur, un flottant représentant sa température, un autre flottant représentant sa vitesse...

La description des messages est stockée dans `nom_package/msg/monMessageType.msg`. Ce fichier décrit la structure des messages

### ❖ Les services :

Le topic est un mode de communication asynchrone permettant une communication many-to-many. Le service en revanche répond à une autre nécessité, celle d'une communication synchrone entre deux nœuds. Cette notion se rapproche de la notion d'appel de procédure distante (remote procedure call). La description des services est stockée dans `nom_package/srv/monServiceType.srv`. Ce fichier décrit les structures de données des requêtes et des réponses.

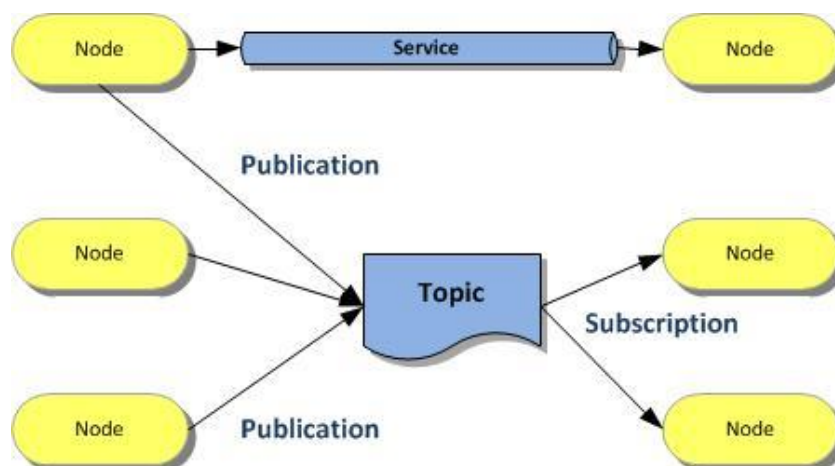


Figure 29 – architecture de service.

### ❖ Les bags :

Les « bags » sont des formats pour stocker et rejouer les messages échangés. Ce système permet de collecter par exemple les données mesurées par les capteurs et les rejouer ensuite autant de fois qu'on le souhaite afin de faire de la simulation sur des données réelles. Ce système est également très utile pour déboguer un système a posteriori.

## 5.4. Les outils de simulations supportés par ROS:

### 5.4.1. GAZEBO :

GAZEBO est un outil permettant de concevoir des mondes en 3D comprenant des robots, une topographie et d'autres objets. Il dispose également d'un moteur physique qui permet de modéliser l'éclairage, la gravité et d'autres forces.

Dans notre projet, nous allons utiliser GAZEBO pour évaluer et tester notre robot dans différents scénarios. Cela est souvent plus rapide que d'utiliser des robots réels et des environnements physiques. GAZEBO facilitera également les tâches de test liées à d'autres aspects de notre robot, tels que la gestion des erreurs, la localisation et les algorithmes de navigation. GAZEBO fournit deux programmes exécutables pour exécuter les simulations :

- Un serveur **gzserver** qui simule la physique, le rendu et les capteurs.
- Un client **gzclient** qui offre une interface graphique permettant de visualiser et d'interagir avec la simulation.

Le client et le serveur communiquent en utilisant la bibliothèque de communication de GAZEBO. Cette bibliothèque utilise actuellement une solution open source pour la sérialisation des messages et le mécanisme de transport. Elle prend en charge le paradigme de communication publication/abonnement. Par exemple, un monde simulé publie des mises à jour de la position des objets, et la génération de capteurs ainsi que l'interface graphique consomment ces messages pour produire une sortie.

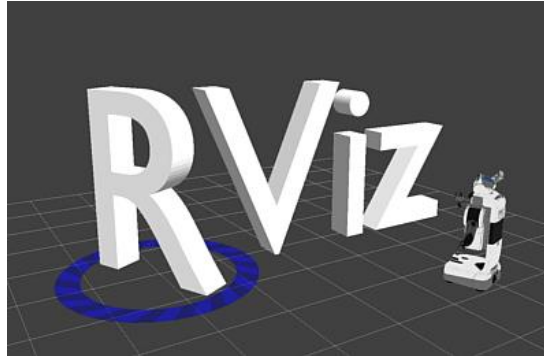


*Figure 30 – LOGO de GAZEBO.*

### 5.4.2. Rviz :

Rviz est un logiciel permettant la visualisation en 3D des applications ROS. Il offre la possibilité d'observer votre modèle de robot, d'acquérir des informations provenant des capteurs

du robot, et de diffuser les données collectées. Il est capable d'afficher des données provenant de caméras, de scanners laser, d'appareils 2D et 3D, incluant des images et des nuages de points.

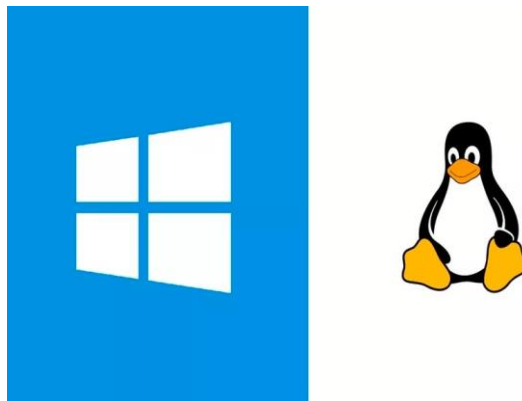


*Figure 31 – LOGO de Rviz.*

## **5.5. Windows subsystem for Linux (WSL):**

### **5.5.1. Definition:**

Le sous-système Windows pour Linux (WSL) est une fonctionnalité développée par Microsoft qui permet aux utilisateurs d'exécuter un environnement Linux complet directement sur leurs systèmes Windows. Il s'agit d'une solution qui vise à combler l'écart entre les deux systèmes d'exploitation en offrant une compatibilité et une expérience utilisateur transparente.



### **5.5.2. Ces avantages :**

Traditionnellement, l'exécution d'applications Linux sur Windows nécessitait des techniques de virtualisation telles que l'installation de machines virtuelles (VM VirtualBox, VM ware) ou l'utilisation de solutions de double démarrage. Cependant, ces approches peuvent être complexes à mettre en place et nécessitent des ressources supplémentaires. Avec WSL, Microsoft a introduit une solution intégrée qui permet d'exécuter un environnement Linux natif sans avoir besoin de machines virtuelles ou de redémarrer le système.



WSL utilise une couche de compatibilité entre le noyau Windows et le noyau Linux, ce qui permet aux utilisateurs d'exécuter des binaires Linux directement sur leur système Windows, sans nécessiter de modifications supplémentaires. Cela signifie que les utilisateurs peuvent profiter des avantages de l'écosystème Linux, tels que les outils de ligne de commande, les utilitaires de développement et les applications open-source, tout en conservant l'environnement Windows familier.

WSL offre deux versions principales : WSL 1 et WSL 2. WSL 1 utilise une traduction dynamique pour convertir les appels système Linux en appels Windows. Bien que cela permette d'exécuter des applications Linux, il peut y avoir une légère perte de performances. En revanche, WSL 2 utilise une véritable virtualisation légère, offrant des performances améliorées et une meilleure compatibilité avec le noyau Linux. WSL 2 utilise également une distribution Linux en tant que noyau, offrant ainsi une expérience plus complète.

### **5.5.3. Distribution utilisé :**

Pour utiliser WSL, les utilisateurs doivent installer une distribution Linux prise en charge, telles que Ubuntu, Debian ou Fedora, à partir de la Microsoft Store. Une fois la distribution Linux installée, les utilisateurs peuvent accéder à un terminal Linux directement à partir de Windows et exécuter des commandes, installer des packages et utiliser des outils de développement Linux.

Dans le cadre de notre projet on a travaillé par Ubuntu 22.04 LTS



*Figure 32 – LOGO de Ubuntu 22.04.LTS*

# Chapitre 6

## Simulation d'un model robot

### 6.1. Introduction :

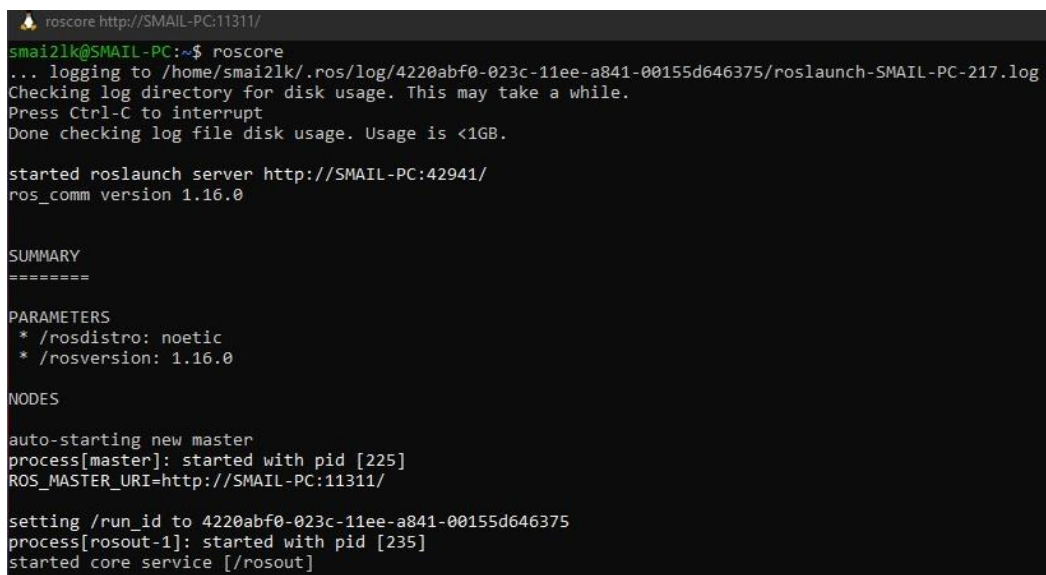
Ce chapitre présente les étapes pour créer un espace de travail ("workspace") et installer les packages ROS nécessaires pour notre projet. Ensuite, nous aborderons la création d'un modèle de robot et de son environnement à l'aide de Gazebo, ainsi que l'ajout de plugins pour différents capteurs utilisés. Nous explorerons ensuite la simulation et le contrôle du robot par l'utilisateur (conducteur). Enfin, on va avoir les améliorations nécessaires pour transformer ce modèle en un véhicule autonome, fonctionnant sans intervention du conducteur.

### 6.2. Configuration de l'espace de travail ROS « workspace » :

Afin de pouvoir utiliser ROS il est nécessaire de configurer son environnement Shell.

➤ Start ROS :

En utilisant la commande : `$ roscore`



```
roscore http://SMAIL-PC:11311/
smai21k@SMAIL-PC:~$ roscore
... logging to /home/smai21k/.ros/log/4220abf0-023c-11ee-a841-00155d646375/roslaunch-SMAIL-PC-217.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://SMAIL-PC:42941/
ros_comm version 1.16.0

SUMMARY
=====

PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [225]
ROS_MASTER_URI=http://SMAIL-PC:11311/

setting /run_id to 4220abf0-023c-11ee-a841-00155d646375
process[rosout-1]: started with pid [235]
started core service [/rosout]
```

Le « roscore » est un ensemble de nœuds et de programmes essentiels pour un système basé sur ROS. La présence d'un roscore en cours d'exécution est requise pour permettre la communication entre les nœuds ROS.

NB : roscore démarre :

- ROS Master.
- ROS Parameter Server.

- rosout logging node.

Définissons maintenant un nouveau dossier comme notre espace de travail, sous le nom de « Robotics\_ws ». Ou dedans un autre dossier « src ». (mkdir = make directory).

```
$ mkdir -p Robotics_ws/src
```

```
$ cd Robotics_ws
```

```
$ ls
```

```
$ catkin_make
```

```
$ source devel/setup.bash
```

- La commande 'cd' est utilisée pour se déplacer d'un répertoire à un autre.
- La commande 'ls' est utilisée pour afficher les fichiers et répertoires présents dans le répertoire actuel.
- La commande 'catkin\_make' permet de compiler les packages ROS dans un workspace catkin sous Ubuntu. Lorsqu'elle est exécutée, elle crée par défaut deux répertoires importants : 'build' et 'devel'.
- La commande 'source devel/setup.bash' charge les variables d'environnement et configure les chemins nécessaires pour utiliser les packages ROS d'un workspace de développement. Cela garantit que le système peut trouver les ressources appropriées lors de l'exécution des packages ROS.

```
smai21k@SMAIL-PC:~$ cd Robotics_ws
smai21k@SMAIL-PC:~/Robotics_ws$ source devel/setup.bash
smai21k@SMAIL-PC:~/Robotics_ws$ ls
LICENSE  README.md  build  devel  docs  map.pgm  map.yaml  src
smai21k@SMAIL-PC:~/Robotics_ws$
```

Généralement voici un schéma qui illustre la notion d'une workspace :



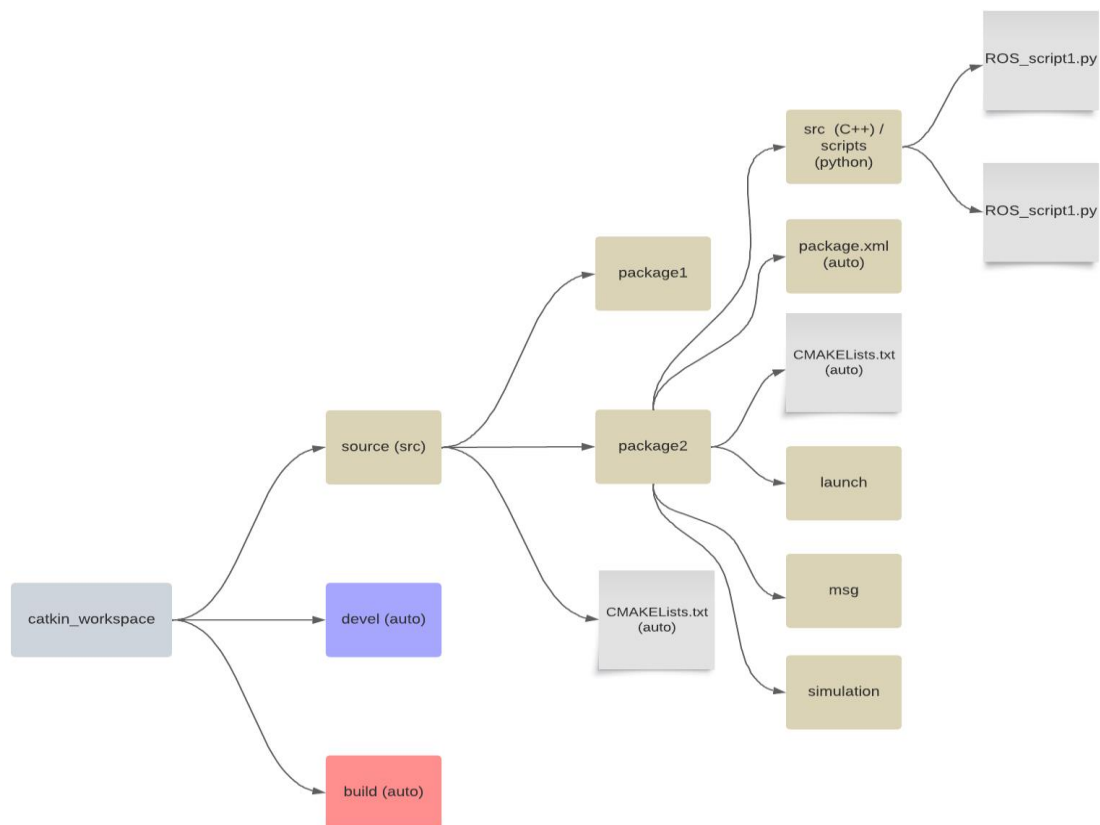


Figure 33 – mind-map d'une workspace.

## 6.3. Mise en place du Robot dans GAZEBO :

### 6.3.1. Edit model environnement :

Dans l'environnement "model editor", GAZEBO comprend principalement trois formes géométriques globales : le cube, le cylindre et la sphère. Ces trois formes nous permettent de créer notre modèle. Dans notre cas, nous avons utilisé uniquement le cylindre et le cube.

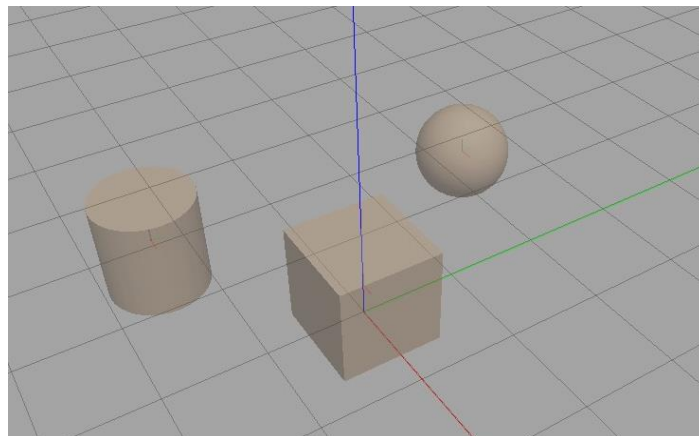


Figure 34 – les formes géométriques (GAZEBO).



GAZEBO offre aussi la possibilité d'ajouter d'autres formes géométriques ainsi que des objets prédéfinis en utilisant des logiciels de CAO tels que SolidWorks, Autodesk, Inventor, Catia, etc.



Le principe de création de robot à partir de GAZEBO est assez simple, le châssis (box) doit être relié avec les roues (cylindres) par des 'joints'. Le châssis est considéré comme 'parent' et les roues comme des 'childs' la figure ci-dessous illustre le processus de jonction.

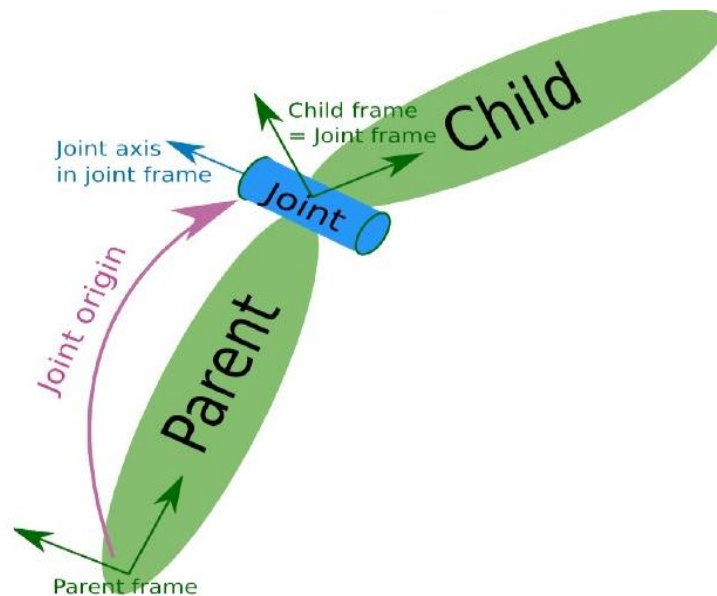


Figure 35 – joints & links (GAZEBO).

### 6.3.2. URDF :

Après qu'on termine avec le modèle de robot (châssis & quatre roues & capteurs), on obtient un code URDF (Unified Robot Description Format), un format XML standardisé permettant de décrire notre robot complet.

L'URDF est utilisé par des simulateurs tels que Gazebo pour représenter le robot. Dans ce processus, nous spécifions les paramètres du robot tels que les couleurs, le châssis, les roues

et les dimensions de chaque objet. Il est possible de définir le robot comme étant statique ou dynamique, et d'ajouter des propriétés physiques et de collision.

```
18      <!-- Make Chassis of Bot -->
19      <link name="chassis">
20          <pose>0 0 0.1 0 0 0</pose>
21
22          <inertial>
23              <mass value="${robot_chassis_mass}"/>
24              <origin xyz="0.0 0 0" rpy=" 0 0 0"/>
25
26              <inertia
27                  ixx="0.147116667" ixy="0" ixz="0"
28                  iyy="0.334951167" iyz="0"
29                  izz="0.3978345"
30              />
31          </inertial>
32
33          <collision name="collision">
34              <origin xyz="0 0 0.05" rpy=" 0 0 0"/>
35              <geometry>
36                  <box size="0.5 0.5 0.2"/>
37              </geometry>
38          </collision>
39
40          <visual name="chassis_visual">
41              <origin xyz="0 0 0.05" rpy=" 0 0 0"/>
42              <geometry>
43                  <box size="0.5 0.5 0.2"/>
44              </geometry>
45          </visual>
46      </link>
```

Figure 36 – structure d'une Link (ex : châssis).

Dans ce Link on a défini le châssis comme un box (cube) dont la taille est définie dans la ligne 36.

Ce Link contient trois parties :

- **Partie Visual** : ce qu'on peut voir dans GAZEBO.
- **Partie de collision** : les conditions de collision avec d'autre objet.
- **Partie inertial** : les moments d'inertie.

### 6.3.3. Ajout des Plugins :

Un plugin d'un capteur est un logiciel ou un module complémentaire qui permet de connecter et d'intégrer un capteur spécifique à une plateforme, à un système ou à une application existante.

Lorsqu'on parle d'un plugin pour un capteur, cela signifie généralement qu'il existe une interface logicielle ou une bibliothèque développée spécifiquement pour ce capteur, afin de faciliter son utilisation et son intégration dans différents contextes. Le plugin peut fournir des fonctionnalités de communication, de configuration, de traitement des données ou d'interaction avec d'autres composants du système.

```
37  <!-- camera -->
38  <gazebo reference="camera">
39    <sensor type="camera" name="camera1">
40      <update_rate>30.0</update_rate>
41      <camera name="head">
42        <horizontal_fov>1.3962634</horizontal_fov>
43        <image>
44          <width>800</width>
45          <height>800</height>
46          <format>R8G8B8</format>
47        </image>
48        <clip>
49          <near>0.02</near>
50          <far>300</far>
51        </clip>
52      </camera>
53      <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
54        <robotNamespace>/atom</robotNamespace>
55        <alwaysOn>true</alwaysOn>
56        <updateRate>0.0</updateRate>
57        <cameraName>camera</cameraName>
58        <imageTopicName>rgb/image_raw</imageTopicName>
59        <cameraInfoTopicName>rgb/camera_info</cameraInfoTopicName>
60        <frameName>camera</frameName>
61        <hackBaseline>0.07</hackBaseline>
62        <distortionK1>0.0</distortionK1>
63        <distortionK2>0.0</distortionK2>
64        <distortionK3>0.0</distortionK3>
65        <distortionT1>0.0</distortionT1>
66        <distortionT2>0.0</distortionT2>
67      </plugin>
68    </sensor>
69  </gazebo>
```

Figure 37 – exemple d'un plugin de camera stéréo.

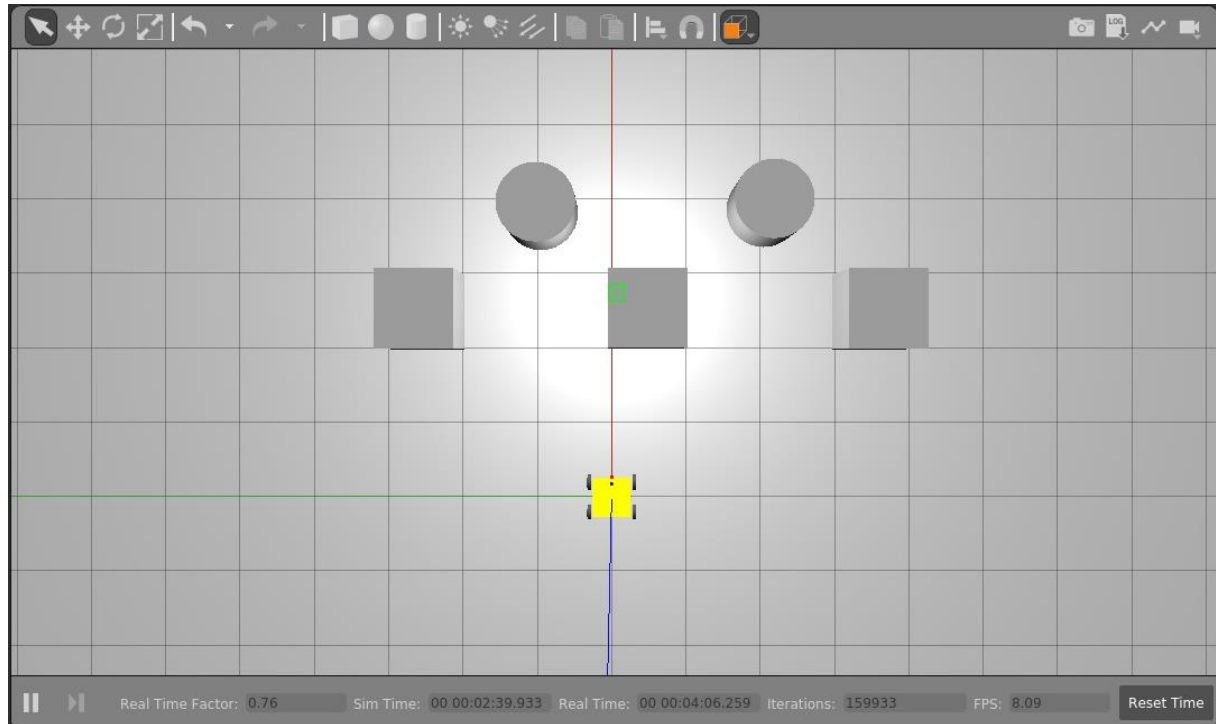
### 6.3.4. Simulation & contrôle de robot (conduite déportée):

- Pour lancer la simulation il faut utiliser la commande :

`$ roslaunch atom world.launch`

Cette commande est utilisée pour démarrer le nœud ROS "atom" en utilisant la configuration spécifiée dans le fichier "world.launch". Cela permet de préparer le système ROS pour exécuter les tâches et les fonctionnalités du robot "atom".

**NB :** chaque fichier de simulation contient un fichier 'launch' pour lancer GAZEBO&Rviz simultanément.



*Figure 38 – modèle de robot sur GAZEBO équipé par camera stéréo & Lidar*

En utilisant Rviz, il est possible d'observer clairement le fonctionnement de ces deux capteurs. À gauche, une représentation en 3D de l'environnement devant le robot est visible, tandis qu'au centre, on peut facilement repérer l'apparition des obstacles positionnés en face du robot.

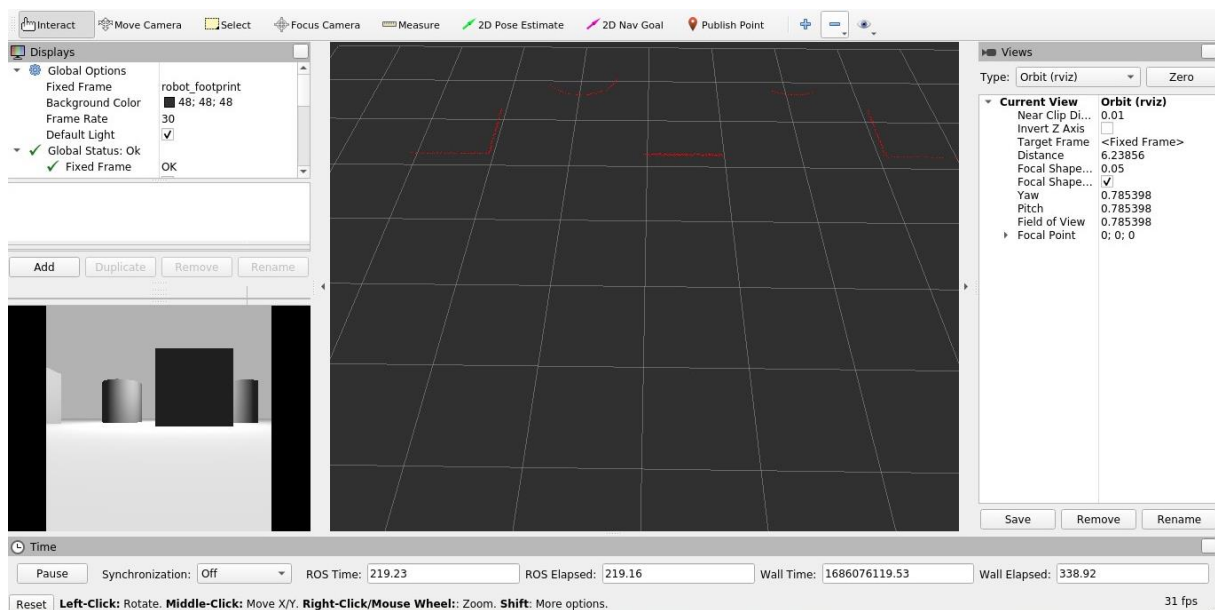


Figure 39 – définition d'environnement devant le robot.

- Ce qui concerne la commande de notre objet on utilise twist keyboard qui est un package ROS qui permet de contrôler les mouvements de notre robot à l'aide du clavier. Il fournit une interface simple pour envoyer des commandes de mouvement linéaire (avancer, reculer) et de rotation (tourner à gauche, tourner à droite) à un robot à partir du clavier de l'ordinateur.

On peut l'utiliser à l'aide de cette commande :

```
$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/atom/cmd_vel
```

```
smai21k@SMAIL-PC:~$ cd Robotics_ws/
smai21k@SMAIL-PC:~/Robotics_ws$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/atom/cmd_vel

Reading from the keyboard and Publishing to Twist!
-----
Moving around:
u   i   o
j   k   l
m   ,   .

For Holonomic mode (strafing), hold down the shift key:
-----
U   I   O
J   K   L
M   <   >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5      turn 1.0
```

Figure 40 – interface de commande de twist\_keyboard.



Résultat de simulation : **voir la vidéo.**

## 6.4. Gmapping :

Dans ROS (Robot Operating System), "gmapping" est un package qui permet la construction de cartes et la localisation simultanée (SLAM - Simultaneous Localization and Mapping) pour les robots mobiles.

Le package "gmapping" utilise les données provenant de capteurs tels que le Lidar & Radar montés sur le robot pour estimer la position du robot dans l'environnement et construire une carte en temps réel. Il utilise un algorithme appelé Grid-based FastSLAM (FastSLAM basé sur une grille) pour effectuer la SLAM.

Pour faire une carte à partir d'un robot avec un laser publiant des scans sur notre topic, on utilise la commande :

```
$ roslaunch atom gmapping_demo.launch
```

```
smai21k@SMAIL-PC:~$ cd Robotics_ws/  
smai21k@SMAIL-PC:~/Robotics_ws$ source devel/setup.bash  
smai21k@SMAIL-PC:~/Robotics_ws$ roslaunch atom gmapping_demo.launch
```

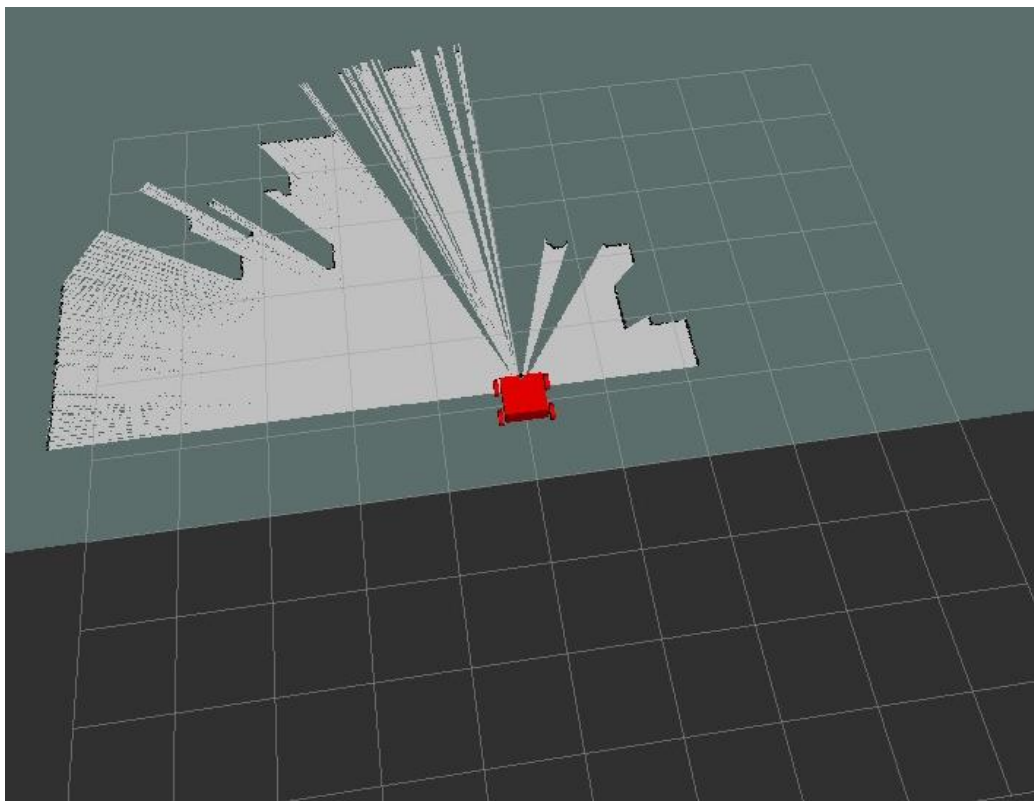


Figure 41 – gmapping sur Rviz.

Et Maintenant, pour enregistrer la carte, utilisez la commande suivante :

```
$ cd Robotics_ws
```

```
$ rosrun map_server map_saver -f map
```

Il convient de noter que ROS offre également d'autres packages et outils pour la cartographie et la localisation, tels que "cartographer" et "amcl" (Adaptive Monte Carlo Localization), qui peuvent être utilisés dans des simulations de robots autonomes basées sur ROS. Cependant, gmapping est l'un des packages populaires et bien établis pour la cartographie et la localisation dans l'écosystème ROS.

## 6.5. Localization :

Dans le domaine de la robotique autonome, la localisation est une tâche fondamentale qui permet à un robot de déterminer sa position et son orientation dans son environnement. Pour réaliser cette localisation précise, les robots utilisent des algorithmes et des techniques spécifiques. L'un de ces algorithmes couramment utilisés est AMCL, ou Adaptive Monte Carlo Localization.

### 6.5.1. AMCL :

AMCL est un algorithme de localisation basé sur la méthode Monte Carlo Localization (MCL). Il utilise une approche probabiliste pour estimer la position du robot en utilisant des particules, également appelées échantillons.

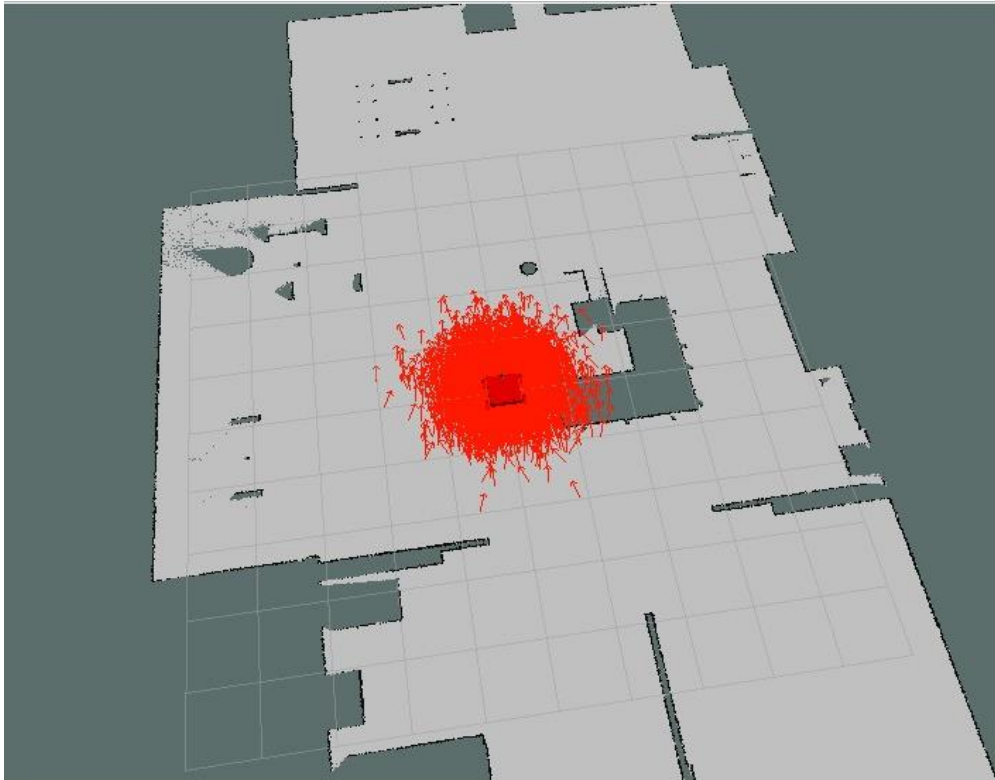
- Le principe de base d'AMCL est d'estimer la pose du robot en utilisant des mesures de capteurs et des modèles de mouvement. Le robot est équipé de capteurs tels que des capteurs laser ou des caméras, qui lui permettent de percevoir l'environnement et d'obtenir des informations sur les obstacles, les repères ou les caractéristiques de l'environnement.
- AMCL utilise ces informations des capteurs pour mettre à jour les particules représentant les différentes hypothèses de position du robot. Les particules qui correspondent le mieux aux mesures des capteurs reçoivent une probabilité plus élevée, tandis que celles qui ne correspondent pas reçoivent une probabilité plus faible.
- En utilisant des modèles de mouvement, AMCL prédit la nouvelle position probable du robot en tenant compte de ses commandes de mouvement. Cette prédiction est basée sur la propagation des particules et les informations de mouvement du robot.
- L'algorithme AMCL effectue également un rééchantillonnage des particules, où les particules les plus probables sont conservées, tandis que les moins probables sont remplacées par de nouvelles particules. Ce processus permet de converger vers une estimation de la pose précise du robot.

Pour effectuer cette estimation on utilise la commande suivante :

```
$ roslaunch atom localization.launch
```

```
smai21k@SMAIL-PC:~$ cd Robotics_ws/  
smai21k@SMAIL-PC:~/Robotics_ws$ source devel/setup.bash  
smai21k@SMAIL-PC:~/Robotics_ws$ roslaunch atom localization.launch
```

Le package AMCL sur Rviz permet d'afficher des informations de localisation, telles qu'un nuage de points ou la position estimée du robot, comme illustré dans l'image ci-dessous :



*Figure 42 – nuage des points d'estimation de position de robot.*

## 6.6. Navigation :

Une fois que le processus de cartographie avec gmapping et de localisation avec AMCL est terminé, l'objectif principal est d'atteindre la navigation autonome pour notre robot. Le fonctionnement de la navigation autonome repose sur les étapes suivantes : le robot utilise ses capteurs pour collecter des données sur son environnement, qu'il utilise ensuite pour construire une carte en utilisant l'algorithme gmapping. Une fois que la carte est prête, l'AMCL est utilisé pour estimer la position et l'orientation actuelles du robot en se basant sur les informations sensorielles et les données cartographiques. Ce processus de localisation est répété régulièrement pour mettre à jour en temps réel la position estimée du robot pendant qu'il se déplace.

Pour atteindre cet objectif on va utiliser un package ROS de navigation qui s'appelle : **navigation stack**.

### 6.6.1. Navigation stack :

La navigation stack (ou pile de navigation) est un ensemble de packages logiciels dans le Framework Robot Operating System (ROS) qui permettent à un robot de naviguer de manière autonome dans un environnement. Elle fournit une architecture modulaire et flexible pour la planification de trajectoires, la perception de l'environnement, la localisation et le contrôle du robot.



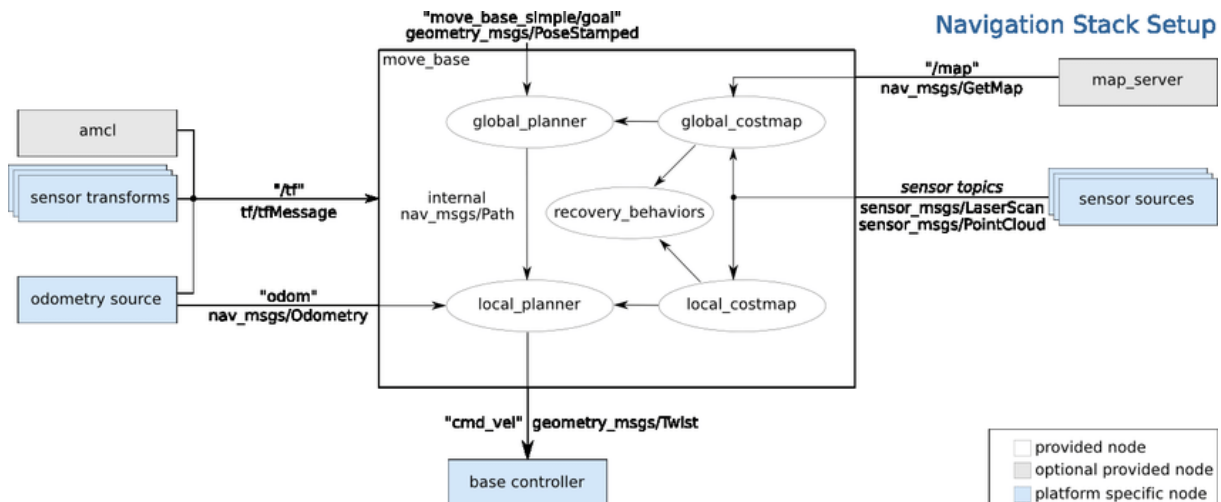


Figure 43 – Les éléments de navigation pour un système ROS.

La navigation stack de ROS se compose de plusieurs composants clés :

- **Map Server** : Il fournit la carte de l'environnement au robot. La carte peut être créée à l'aide d'algorithmes de SLAM (Simultaneous Localization and Mapping) ou être pré-construite.
- **AMCL (Adaptive Monte Carlo Localization)** : Il est responsable de la localisation du robot dans la carte. En utilisant des méthodes probabilistes, AMCL estime la position et l'orientation actuelles du robot en fonction des données du capteur et de la carte.
- **Move Base** : C'est le package principal de planification de trajectoires. Il utilise la carte fournie et les informations de localisation pour générer une trajectoire de déplacement pour le robot, en tenant compte des obstacles, des contraintes de vitesse et d'autres paramètres.
- **Costmap** : Il génère une représentation en grille de l'environnement, où chaque cellule représente l'occupation ou l'accessibilité de l'espace. La costmap est utilisée par Move Base pour la planification de trajectoires en évitant les obstacles.
- **Global Planner** : C'est le composant responsable de la planification globale des trajectoires. Il génère une trajectoire de grande envergure pour atteindre la destination souhaitée en évitant les obstacles majeurs.
- **Local Planner** : Il gère la planification locale des trajectoires, en réagissant aux obstacles à court terme et en effectuant des ajustements de trajectoire en temps réel pour éviter les collisions.

Pour lancer notre navigation il faut utiliser la commande suivante :

`$ roslaunch atom navigation.launch`



Figure 44 – navigation de robot (Rviz).

Résultat de simulation : **voir la vidéo.**

## 6.7. Conclusion de deuxième partie :

Dans cette dernière partie, nous avons exploré les débuts du véhicule autonome, son histoire et son évolution, ainsi que la description du système et les capteurs utilisés. Ensuite, nous avons examiné le principe de fonctionnement du véhicule autonome, suivi d'une découverte du ROS (Robot Operating System), y compris ses différentes distributions et son principe de fonctionnement, ainsi que ses outils de simulation. Nous avons également abordé la nouvelle fonctionnalité de Windows, WSL (Windows Subsystem for Linux), et son intérêt d'utilisation, ainsi que la distribution Linux Ubuntu que nous avons utilisée.

Le dernier chapitre a été réservé pour explorer la création d'un espace de travail (workspace), la création d'un robot à l'aide de GAZEBO, sa simulation et son contrôle avec Twist Keyboard. Enfin, nous avons amélioré notre projet pour permettre une navigation autonome en utilisant les différents packages de ROS tels que gmapping, AMCL et la navigation stack.

## CONCLUSION GENERAL

En conclusion, le projet de conduite autonome et déportée des camions de transport de phosphate basé sur ROS a ouvert de nouvelles perspectives dans le domaine de la logistique et du transport. En utilisant le Windows Subsystem for Linux (WSL) et plus précisément Ubuntu 22.04, nous avons pu développer un modèle robotique simple mais fonctionnel pour la conduite autonome.

La conduite autonome requiert l'utilisation de capteurs avancés tels que le LiDAR, la caméra stéréo, l'IMU, etc., qui permettent de collecter des données précises sur l'environnement et d'effectuer une perception de qualité. En intégrant ces capteurs avec ROS, nous avons pu développer un système de perception et de navigation performant, capable de prendre des décisions intelligentes en temps réel.

Le Raspberry Pi a joué un rôle essentiel dans ce projet en tant que plateforme de calcul embarquée. En utilisant un Raspberry Pi, nous avons pu créer un système compact et économique pour le contrôle et la communication entre les différents composants du système de conduite autonome. Le Raspberry Pi offre une flexibilité et une extensibilité considérables, ce qui en fait un choix idéal pour réaliser des projets de conduite autonome à petite échelle.

Ce projet a démontré l'efficacité de l'intégration de ROS, du Raspberry Pi et des capteurs avancés dans le développement d'un système de conduite autonome pour les camions de transport de phosphate. Les résultats obtenus ouvrent la voie à de futures améliorations et à des applications plus larges dans le domaine de la conduite autonome et de la logistique. L'utilisation de ROS comme plateforme de développement a facilité la mise en œuvre du système et a permis d'exploiter pleinement les capacités des capteurs et du Raspberry Pi.

# Annexes

## Annexe 1 : Code URDF de Robot

```
1  <?xml version='1.0'?>
2
3  <robot name="atom" xmlns:xacro="http://www.ros.org/wiki/xacro">
4      <xacro:property name="robot_name" value="atom" />
5      <xacro:property name="robot_chassis_mass" value="15"/>
6      <xacro:property name="robot_chassis_length" value="0.2"/>
7      <xacro:property name="robot_chassis_radius" value="0.25"/>
8      <xacro:property name="robot_caster_wheel_radius" value="0.05"/>
9      <xacro:property name="robot_caster_wheel_radius_collision" value="0.0499"/>
10
11      <xacro:property name="robot_wheel_mass" value="5"/>
12      <xacro:property name="robot_wheel_length" value="0.05"/>
13      <xacro:property name="robot_wheel_radius" value="0.1"/>
14
15      <xacro:property name="camera_mass" value="0.1"/>
16      <xacro:property name="hokoyu_mass" value="1e-5"/>
17
18      <!-- Make Chassis of Bot -->
19      <link name="chassis">
20          <pose>0 0 0.1 0 0 0</pose>
21
22          <inertial>
23              <mass value="${robot_chassis_mass}"/>
24              <origin xyz="0.0 0 0" rpy="0 0 0"/>
25
26              <inertia
27                  ixx="0.147116667" ixy="0" ixz="0"
28                  iyy="0.334951167" iyz="0"
29                  izz="0.3978345"
30              />
31          </inertial>
32
33          <collision name="collision">
34              <origin xyz="0 0 0.05" rpy="0 0 0"/>
35              <geometry>
36                  <box size="0.5 0.5 0.2"/>
37              </geometry>
38          </collision>
39
40          <visual name="chassis_visual">
41              <origin xyz="0 0 0.05" rpy="0 0 0"/>
42              <geometry>
43                  <box size="0.5 0.5 0.2"/>
44              </geometry>
45          </visual>
46      </link>
47
48      <!-- Right Wheel Back -->
49      <link name="right_wheel_back">
50          <inertial>
51              <mass value="${robot_wheel_mass}"/>
52              <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
53              <inertia
54                  ixx="0.1" ixy="0.0" ixz="0.0"
55                  iyy="0.1" iyz="0.0"
56              </inertia>
57          </inertial>
58      </link>
```

Suite de code :

```
60     />
61   </inertial>
62
63   <visual>
64     <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
65     <geometry>
66       <cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
67     </geometry>
68   </visual>
69
70   <collision>
71     <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
72     <geometry>
73       <cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
74     </geometry>
75   </collision>
76
77 </link>
78
79 <!-- Right Wheel Front-->
80 <link name="right_wheel_front">
81   <inertial>
82     <mass value="${robot_wheel_mass}"/>
83     <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
84     <inertia
85       ixx="0.1" ixy="0.0" ixz="0.0"
86       iyy="0.1" iyz="0.0"
87       izz="0.1"
88     />
89   </inertial>
90
91   <visual>
92     <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
93     <geometry>
94       <cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
95     </geometry>
96   </visual>
97
98   <collision>
99     <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
100    <geometry>
101      <cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
102    </geometry>
103  </collision>
104
105 </link>
106
107 <!-- Left wheel Back-->
108 <link name="left_wheel_back">
109   <inertial>
110     <mass value="${robot_wheel_mass}"/>
111     <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
112     <inertia
113       ixx="0.1" ixy="0.0" ixz="0.0"
114       iyy="0.1" iyz="0.0"
115       izz="0.1"
116     />
```



Suite de code :

```
117     </inertial>
118
119     <visual>
120       <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
121       <geometry>
122         <cylinder radius="{robot_wheel_radius}" length="{robot_wheel_length}"/>
123       </geometry>
124     </visual>
125
126     <collision>
127       <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
128       <geometry>
129         <cylinder radius="{robot_wheel_radius}" length="{robot_wheel_length}"/>
130       </geometry>
131     </collision>
132
133   </link>
134
135   <!-- Left wheel Front -->
136   <link name="left_wheel_front">
137     <inertial>
138       <mass value="{robot_wheel_mass}"/>
139       <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
140       <inertia
141         ixx="0.1" ixy="0.0" ixz="0.0"
142         iyy="0.1" iyz="0.0"
143         izz="0.1"
144       />
145     </inertial>
146
147     <visual>
148       <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
149       <geometry>
150         <cylinder radius="{robot_wheel_radius}" length="{robot_wheel_length}"/>
151       </geometry>
152     </visual>
153
154     <collision>
155       <origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
156       <geometry>
157         <cylinder radius="{robot_wheel_radius}" length="{robot_wheel_length}"/>
158       </geometry>
159     </collision>
160
161   </link>
162
163   <!-- Camera -->
164   <link name="camera">
165     <inertial>
166       <mass value="{camera_mass}"/>
167       <origin xyz="0 0 0" rpy="0 0 0"/>
168       <inertia
169         ixx="1e-6" ixy="0.0" ixz="0.0"
170         iyy="1e-6" iyz="0.0"
171         izz="1e-6"
172       />
173     </inertial>
```

Suite de code :

```
175     <visual>
176       <origin xyz="0 0 0" rpy="0 0 0"/>
177       <geometry>
178         <box size="0.05 0.05 0.05"/>
179       </geometry>
180     </visual>
181
182     <collision>
183       <origin xyz="0 0 0" rpy="0 0 0"/>
184       <geometry>
185         <box size="0.05 0.05 0.05"/>
186       </geometry>
187     </collision>
188   </link>
189
190   <!-- Hokuyo Lidar -->
191   <link name="hokuyo">
192     <inertial>
193       <mass value="${hokoyu_mass}"/>
194       <origin xyz="0 0 0" rpy="0 0 0"/>
195
196       <inertia
197         ixx="1e-6" ixy="0.0" ixz="0.0"
198         iyy="1e-6" iyz="0.0"
199         izz="1e-6"
200       />
201     </inertial>
202
203     <visual>
204       <origin xyz="0 0 0" rpy="0 0 0"/>
205       <geometry>
206         <mesh filename="package://aton/meshes/hokuyo.dae"/>
207       </geometry>
208     </visual>
209
210     <collision>
211       <origin xyz="0 0 0" rpy="0 0 0"/>
212       <geometry>
213         <box size="0.1 0.1 0.1"/>
214       </geometry>
215     </collision>
216   </link>
217
218   <!-- Project center to the ground -->
219   <link name="robot_footprint"></link>
220
221
222   <!-- Define Joints -->
223
224   <!-- Right Wheel Joint Back-->
225   <joint type="continuous" name="right_wheel_hinge_back">
226     <origin xyz="-0.2 -0.30 0" rpy="0 0 0" />
227     <parent link="chassis"/>
228     <child link="right_wheel_back" />
229     <axis xyz="0 1 0" rpy="0 0 0" />
230     <limit effort="10000" velocity="1000" />
231
```

Suite de code :

```
232     <physics damping="1.0" friction="1.0" />
233 </joint>
234
235 <!-- Right Wheel Joint Front-->
236 <joint type="continuous" name="right_wheel_hinge_front">
237   <origin xyz="0.2 -0.30 0" rpy="0 0 0" />
238   <parent link="chassis"/>
239   <child link="right_wheel_front" />
240   <axis xyz="0 1 0" rpy="0 0 0" />
241   <limit effort="10000" velocity="1000" />
242   <physics damping="1.0" friction="1.0" />
243 </joint>
244
245
246 <!-- Left Wheel Joint Back-->
247 <joint type="continuous" name="left_wheel_hinge_back">
248   <origin xyz="-0.2 0.30 0" rpy="0 0 0" />
249   <parent link="chassis"/>
250   <child link="left_wheel_back" />
251   <axis xyz="0 1 0" rpy="0 0 0" />
252   <limit effort="10000" velocity="1000" />
253   <physics damping="1.0" friction="1.0" />
254 </joint>
255
256 <!-- Left Wheel Joint Front-->
257 <joint type="continuous" name="left_wheel_hinge_front">
258   <origin xyz="0.2 0.30 0" rpy="0 0 0" />
259   <parent link="chassis"/>
260   <child link="left_wheel_front" />
261   <axis xyz="0 1 0" rpy="0 0 0" />
262   <limit effort="10000" velocity="1000" />
263   <physics damping="1.0" friction="1.0" />
264 </joint>
265
266 <!-- Camera Joint -->
267 <joint name="camera_joint" type="fixed">
268   <origin xyz="0.26 0 0" rpy="0 0 0" />
269   <parent link="chassis"/>
270   <child link="camera" />
271   <axis xyz="0 1 0"/>
272 </joint>
273
274 <!-- Hokuyo Joint -->
275 <joint name="hokuyo_joint" type="fixed">
276   <origin xyz="0.2 0 0.2" rpy="0 0 0" />
277   <parent link="chassis"/>
278   <child link="hokuyo" />
279   <axis xyz="0 1 0"/>
280 </joint>
281
282 <joint name="robot_footprint_joint" type="fixed">
283   <origin xyz="0 0 0" rpy="0 0 0" />
284   <parent link="robot_footprint"/>
285   <child link="chassis" />
286 </joint>
287
```



Suite de code :

```
289 <!-- Color of bot -->
290 <gazebo reference="left_wheel_front">
291   <material>Gazebo/DarkGrey</material>
292   <kp>1000000.0</kp> <!-- kp and kd for rubber -->
293   <kd>100.0</kd>
294   <mul>1.0</mul>
295   <mu2>1.0</mu2>
296   <maxVel>1.0</maxVel>
297   <minDepth>0.00</minDepth>
298 </gazebo>
299
300 <gazebo reference="left_wheel_back">
301   <material>Gazebo/DarkGrey</material>
302   <kp>1000000.0</kp> <!-- kp and kd for rubber -->
303   <kd>100.0</kd>
304   <mul>1.0</mul>
305   <mu2>1.0</mu2>
306   <maxVel>1.0</maxVel>
307   <minDepth>0.00</minDepth>
308 </gazebo>
309
310 <gazebo reference="right_wheel_front">
311   <material>Gazebo/DarkGrey</material>
312   <kp>1000000.0</kp> <!-- kp and kd for rubber -->
313   <kd>100.0</kd>
314   <mul>1.0</mul>
315   <mu2>1.0</mu2>
316   <maxVel>1.0</maxVel>
317   <minDepth>0.00</minDepth>
318 </gazebo>
319 <gazebo reference="right_wheel_back">
320   <material>Gazebo/DarkGrey</material>
321   <kp>1000000.0</kp> <!-- kp and kd for rubber -->
322   <kd>100.0</kd>
323   <mul>1.0</mul>
324   <mu2>1.0</mu2>
325   <maxVel>1.0</maxVel>
326   <minDepth>0.00</minDepth>
327 </gazebo>
328 <!--<gazebo reference="right_wheel">
329   <material>Gazebo/Green</material>
330 </gazebo-->
331
332 <gazebo reference="camera">
333   <material>Gazebo/Red</material>
334 </gazebo>
335
336 <gazebo reference="chassis">
337   <material>Gazebo/Yellow</material>
338 </gazebo>
339
340 <!-- Motor, Camera and Lidar Simulation -->
341 <xacro:include filename="$(find atom)/urdf/atom.gazebo" />
342
343 </robot>
```

## Annexe 2: gmapping

- Auto\_mapping.launch :

```
1 <launch>
2
3
4   <arg name="cmd_vel" default="/atom/cmd_vel"/>
5   <arg name="odom" default="/atom/odom"/>
6
7   <param name="cmd_vel_topic_name" value="$(arg cmd_vel)"/>
8   <param name="odom_topic_name" value="$(arg odom)"/>
9
10  <node name="bot_drive" pkg="navigation_stack" type="bot_drive" required="true" output="screen"/>
11
12 </launch>
```

- Amcl.launch :

```
1 <launch>
2
3   <arg name="base_link_name" default="base_footprint"/>
4
5   <!-- Map Server Arguments -->
6   <arg name="map_file" default="$(find atom)/../maps/aws_house/map.yaml"/>
7
8   <!-- <arg name="map_file" default="$(find turtlebot_tutorials)/maps/map1.yaml"/> -->
9   <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_file)" />
10
11   <!-- Arguments -->
12   <arg name="scan_topic" default="scan"/>
13   <arg name="initial_pose_x" default="0.0"/>
14   <arg name="initial_pose_y" default="0.0"/>
15   <arg name="initial_pose_a" default="0.0"/>
16
17   <!-- AMCL -->
18   <node pkg="amcl" type="amcl" name="amcl">
19
20     <!-- <param name="min_particles" value="500"/>
21     <param name="max_particles" value="3000"/>
22     <param name="kld_err" value="0.02"/>
23     <param name="update_min_d" value="0.20"/>
24     <param name="update_min_a" value="0.20"/>
25     <param name="resample_interval" value="1"/>
26     <param name="transform_tolerance" value="0.5"/>
27     <param name="recovery_alpha_slow" value="0.00"/>
28     <param name="recovery_alpha_fast" value="0.00"/>
29     <param name="initial_pose_x" value="$(arg initial_pose_x)"/>
30     <param name="initial_pose_y" value="$(arg initial_pose_y)"/>
31     <param name="initial_pose_a" value="$(arg initial_pose_a)"/>
32     <param name="gui_publish_rate" value="50.0"/>
33     <remap from="scan" to="$(arg scan_topic)"/>
34     <param name="laser_max_range" value="3.5"/>
35     <param name="laser_max_beams" value="180"/>
36     <param name="laser_z_hit" value="0.5"/>
37     <param name="laser_z_short" value="0.05"/>
38     <param name="laser_z_max" value="0.05"/>
39     <param name="laser_z_rand" value="0.5"/>
40     <param name="laser_sigma_hit" value="0.2"/>
41     <param name="laser_lambda_short" value="0.1"/>
42     <param name="laser_likelihood_max_dist" value="2.0"/>
43     <param name="laser_model_type" value="likelihood_field"/>
44     <param name="odom_model_type" value="diff"/>
45     <param name="odom_alpha1" value="0.1"/>
46     <param name="odom_alpha2" value="0.1"/>
47     <param name="odom_alpha3" value="0.1"/>
48     <param name="odom_alpha4" value="0.1"/>
49     <param name="odom_frame_id" value="odom"/>
50     <param name="base_frame_id" value="base_footprint"/> -->
51     <param name="base_frame_id" value="$(arg base_link_name)"/>
52
53   </node>
54 </launch>
```

- Map.yaml :

```
1  image: map.pgm
2  resolution: 0.010000
3  origin: [-11.240000, -12.200000, 0.000000]
4  negate: 0
5  occupied_thresh: 0.65
6  free_thresh: 0.196
```

## Annexe 3 : localisation

- localization.launch :

```
1 <launch>
2
3   <arg name="custom_amcl_launch_file" default="$(find navigation_stack)/launch/amcl.launch"/>
4
5
6   <include file="$(find atom)/launch/house.launch"/>
7
8   <include file="$(arg custom_amcl_launch_file)">
9     <arg name="base_link_name" value="chassis"/>
10    <arg name="map_file" value="$(find atom)/../maps/aws_house/map.yaml"/>
11  </include>
12
13  <node name="rviz" pkg="rviz" type="rviz" respawn="false"
14    args="-d $(find navigation_stack)/rviz/amcl.rviz"/>
15 </launch>
```

## Annexe : navigation

- Package.xml de navigation stack

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>navigation_stack</name>
4   <version>0.0.0</version>
5   <description>The navigation_stack package</description>
6
7   <!-- One maintainer tag required, multiple allowed, one person per tag -->
8   <!-- Example: -->
9   <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
10  <maintainer email="harshmittal2210@gmail.com">harshmittal</maintainer>
11
12
13  <!-- One license tag required, multiple allowed, one license per tag -->
14  <!-- Commonly used license strings: -->
15  <!-- BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
16  <license>TODO</license>
17
18
19  <!-- Url tags are optional, but multiple are allowed, one per tag -->
20  <!-- Optional attribute type can be: website, bugtracker, or repository -->
21  <!-- Example: -->
22  <!-- <url type="website">http://wiki.ros.org/navigation_stack</url> -->
23
24
25  <!-- Author tags are optional, multiple are allowed, one per tag -->
26  <!-- Authors do not have to be maintainers, but could be -->
27  <!-- Example: -->
28  <!-- <author email="jane.doe@example.com">Jane Doe</author> -->
29
30
31  <!-- The *depend tags are used to specify dependencies -->
32  <!-- Dependencies can be catkin packages or system dependencies -->
33  <!-- Examples: -->
34  <!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
35  <!--   <depend>roscpp</depend> -->
36  <!-- Note that this is equivalent to the following: -->
37  <!--   <build_depend>roscpp</build_depend> -->
38  <!--   <exec_depend>roscpp</exec_depend> -->
39  <!-- Use build_depend for packages you need at compile time: -->
40  <!--   <build_depend>message_generation</build_depend> -->
41  <!-- Use build_export_depend for packages you need in order to build against this package: -->
42  <!--   <build_export_depend>message_generation</build_export_depend> -->
43  <!-- Use buildtool_depend for build tool packages: -->
44  <!--   <buildtool_depend>catkin</buildtool_depend> -->
45  <!-- Use exec_depend for packages you need at runtime: -->
46  <!--   <exec_depend>message_runtime</exec_depend> -->
47  <!-- Use test_depend for packages you need only for testing: -->
48  <!--   <test_depend>gtest</test_depend> -->
49  <!-- Use doc_depend for packages you need only for building documentation: -->
50  <!--   <doc_depend>doxygen</doc_depend> -->
51  <buildtool_depend>catkin</buildtool_depend>
52  <build_depend>amcl</build_depend>
53  <build_depend>dwa_local_planner</build_depend>
54  <build_depend>gmapping</build_depend>
55  <build_depend>map_server</build_depend>
56  <build_depend>move_base</build_depend>
57  <build_depend>roscpp</build_depend>
58  <build_depend>rospy</build_depend>
```



```

36 <!-- Note that this is equivalent to the following: -->
37 <!-- <build_depend>roscpp</build_depend> -->
38 <!-- <exec_depend>roscpp</exec_depend> -->
39 <!-- Use build_depend for packages you need at compile time: -->
40 <!-- <build_depend>message_generation</build_depend> -->
41 <!-- Use build_export_depend for packages you need in order to build against this package: -->
42 <!-- <build_export_depend>message_generation</build_export_depend> -->
43 <!-- Use buildtool_depend for build tool packages: -->
44 <!-- <buildtool_depend>catkin</buildtool_depend> -->
45 <!-- Use exec_depend for packages you need at runtime: -->
46 <!-- <exec_depend>message_runtime</exec_depend> -->
47 <!-- Use test_depend for packages you need only for testing: -->
48 <!-- <test_depend>gtest</test_depend> -->
49 <!-- Use doc_depend for packages you need only for building documentation: -->
50 <!-- <doc_depend>doxygen</doc_depend> -->
51 <buildtool_depend>catkin</buildtool_depend>
52 <build_depend>amcl</build_depend>
53 <build_depend>dwa_local_planner</build_depend>
54 <build_depend>gmapping</build_depend>
55 <build_depend>map_server</build_depend>
56 <build_depend>move_base</build_depend>
57 <build_depend>roscpp</build_depend>
58 <build_depend>rospy</build_depend>
59 <build_depend>sensor_msgs</build_depend>
60 <build_depend>tf</build_depend>
61 <build_export_depend>amcl</build_export_depend>
62 <build_export_depend>dwa_local_planner</build_export_depend>
63 <build_export_depend>gmapping</build_export_depend>
64 <build_export_depend>map_server</build_export_depend>
65 <build_export_depend>move_base</build_export_depend>
66 <build_export_depend>roscpp</build_export_depend>
67 <build_export_depend>rospy</build_export_depend>
68 <build_export_depend>sensor_msgs</build_export_depend>
69 <build_export_depend>tf</build_export_depend>
70 <exec_depend>amcl</exec_depend>
71 <exec_depend>dwa_local_planner</exec_depend>
72 <exec_depend>gmapping</exec_depend>
73 <exec_depend>map_server</exec_depend>
74 <exec_depend>move_base</exec_depend>
75 <exec_depend>roscpp</exec_depend>
76 <exec_depend>rospy</exec_depend>
77 <exec_depend>sensor_msgs</exec_depend>
78 <exec_depend>tf</exec_depend>
79
80 <depend>geometry_msgs</depend>
81 <depend>nav_msgs</depend>
82 <depend>gazebo_ros</depend>
83 <exec_depend>gazebo</exec_depend>
84
85
86 <!-- The export tag contains other, unspecified, tags -->
87 <export>
88   <!-- Other tools can request additional information be placed here -->
89
90 </export>
91 </package>

```

- navigation.launch :

```
1 <launch>
2
3   <arg name="custom_amcl_launch_file" default="$(find navigation_stack)/launch/amcl.launch"/>
4
5
6   <include file="$(find atom)/launch/house.launch"/>
7
8   <include file="$(arg custom_amcl_launch_file)">
9     <arg name="base_link_name" value="chassis"/>
10    <arg name="map_file" value="$(find atom)/../maps/aws_house/map.yaml"/>
11  </include>
12
13  <include file="$(find navigation_stack)/launch/includes/move_base.launch.xml">
14    <arg name="base_frame_id" value="chassis"/>
15    <arg name="odom_topic" value="atom/odom" />
16    <arg name="odom_frame_id" value="odom"/>
17    <arg name="cmd_vel_topic" value="atom/cmd_vel" />
18    <arg name="custom_param_file" value="$(find navigation_stack)/param/r200_costmap_params.yaml"/>
19  </include>
20
21  <node name="rviz" pkg="rviz" type="rviz" respawn="false"
22    args="-d $(find navigation_stack)/rviz/navigation.rviz"/>
23
24 </launch>
```

## Webographie

1. <https://fr.wikipedia.org/wiki/Lidar>
2. [https://en.wikipedia.org/wiki/Stereo\\_camera](https://en.wikipedia.org/wiki/Stereo_camera)
3. [https://en.wikipedia.org/wiki/Inertial\\_measurement\\_unit](https://en.wikipedia.org/wiki/Inertial_measurement_unit)
4. <http://wiki.ros.org/gmapping>
5. <http://wiki.ros.org/amcl>
6. [http://wiki.ros.org/robot\\_localization](http://wiki.ros.org/robot_localization)
7. <http://wiki.ros.org/navigation>
8. <http://wiki.ros.org/navigation/Tutorials/RobotSetup>