REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université 8 Mai 1945 – Guelma
Faculté des Sciences et de la Technologie
Département de Génie Electrotechnique et Automatique

Réf :. . . . . /2020

Présenté pour l'obtention du diplôme de MASTER Académique
Domaine : Sciences et Technologie
Filière : Automatique
Spécialité : Automatique et Informatique Industrielle
Thème

Navigation et commande d'un robot mobile à 4 roues sous ROS

Par :Imed-Eddine GUEBLI
Soutenu, le 17/10/2020, devant le jury composé de :

Melle. KECHIDA Sihem	Pr	Univ. Guelma	Président/ Encadreur
M. AIDOUD Mohamed	MCB	Univ. Guelma	Examinateur
Mme. LOUCIF Fatiha	MAA	Univ. Guelma	Examinateur

Année Universitaire : 2019/2020

Remerciements

Je profite ces lignes pour exprimer tous mes remerciement à mon encadreur Pr KECHIDA Sihem ; directeur du labo LAIG, d'avoir cru en moi, pour ses conseils importants.

Je tiens aussi à remercier M. Vinecent COCQUEMPOT, pour son accueil et son aide pendant mon séjour en France.

Je présente tous mes respects, et mes remerciements aux personnes qui ont participé de prêt ou du loin dans tous mes succès ; famille, enseignants, collègues...

Résumé

La robotique est l'intersection de la science, de l'ingénierie et de la technologie qui produit des machines, appelées robots, qui se substituent (ou reproduisent) les actions humaines ; elle rassemble des savoir-faire de différentes disciplines et nécessite donc des compétences très différentes, généralement hors de portée d'une seule personne.L'un des plus grands défis de ce modeste travail est de découvrir suffisamment ce domaine et en particulier les systèmes d'exploitation robotiques permettant aux chercheurs de réaliser des travaux expérimentaux plus significatifs. L'objectif de ce mémoire est d'étudier le contrôle et le suivi de trajectoire d'un robot mobile à roues en utilisant la plateforme de développent robotique appelé ROS.

Mot clés : robotique, ROS, robot mobile, contrôle, suivi de trajectoire.

Abstract

Robotics is the intersection of science, engineering, and technology that produces machines, called robots, that substitute for (or replicate) human actions ; it brings together know-how from different disciplines and therefore requires very different skills, generally beyond the reach of a single person. One of the greatest challenges of this modest work is to sufficiently discover this field and in particular the robotic operating systems allowing researchers to carry out more significant experimental work. The objective of this dissertation is to study the control and trajectory tracking of a wheeled mobile robot using ROS.

Key words : robotics, ROS, mobile robot, control, trajectory tracking.

ملخص

تعتبر الروبوتية مجالا تتقاطع فيه جل العلوم والهندسة والتكنولوجيا، و ذلك بإنتاج آلات تسمى الروبوتات قد تحل محل الأفعال البشرية أو تكررها؛ فهو مجال يتطلب معارف فنية متعددة و مهارات من مختلف التخصصات ، ويتمثل أحد أعظم تحديات هذا العمل المتواضع في اكتشاف هذا المجال بشكل كاف وخاصة أنظمة التشغيل الروبوتية التي تسمح للباحثين بتنفيذ أعمال تجريبية مهمة كالتحكم و المحاكاة و تتبع الأخطاء و توقعها بصفة تسهل من البحث و التطوير في هذا المجال المهم جدا. إن هذه المذكرة تهدف إلى التحكم في روبوت متحرك ذي العجلات وتتبع مساره باستخدام منصة التطوير الروبوتية

الكلمات المفتاحية: الروبوتات، الروبوت المتحرك ، التحكم ، تتبع المسار ROS

Table des matières

Table des figures

Introduction générale

Introduction générale

Les robots sont des machines capables d'exécuter les tâches que les humains sont incapables ou préfèrent ne pas accomplir. Certains robots sont fabriqués pour effectuer des tâches dangereuses et difficiles pour les personnes telles que l'exploration de mines, le désamorçage des bombes, etc.

En effet, le mot ''robot" peut évoquer divers niveaux de sophistication technologique, allant d'un simple appareil de manutention à un humanoïde. L'image des robots varie considérablement selon les chercheurs, les ingénieurs et les fabricants de robots... Les robots sont très techniques et difficiles à créer, et la tâche peut être très délicate. Pour cette raison, il n'est pas rare qu'un ingénieur en robotique ne travaille que sur une poignée de projets tout au long de sa carrière. Les professionnels de ce domaine doivent être très patients.

Cependant, l'étude de la robotique est l'un des rares sujets à traiter dans son ensemble en raison de l'extrême diversité des technologies scientifiques qu'il incorpore. Il utilise de nombreux domaines technologiques, par exemple ; génie mécanique, génie électrique, informatique, électronique, instrumentations (capteurs, actionneurs) et l'intelligence artificielle. C'est un domaine multidimensionnel qui profite de toutes les études de l'ingénierie qui existent dans notre vie en plus de l'introduction des mathématiques ''difficiles" qui doivent être appliquées. L'un des plus grands défis de l'écriture de mémoire était de découvrir suffisamment ce domaine et les logiciels qui permettent la simulation et le contrôle pour des tests de commande et de suivi de trajectoire.

C'est dans ce contexte que s'inscrit le travail présenté dans ce mémoire, il s'intéresse à l'étude d'une classe des robots mobile plus particulièrement les robots terrestres dont l'objectif est de commander un robot à roues pour le suivi de trajectoire. L'idée est réaliser des tests en se basant sur une plateforme de développement des robots avancés ''ROS"en utilisant des outils de visualisation et de simulation robotiques.

Le mémoire est organisé comme suit : Après une introduction générale situant le travail à réaliser, le chapitre 1 expose un aperçu général sur la robotique mobile et les principaux composants suivi d'une classification précisant les différents domaines d'utilisation. La modélisation est un point important dans l'étude des robots est abordée en fin de chapitre. Le chapitre 2 est consacré à une présentation détaillée d'un système d'exploitation conçu pour les robots en passant par les concepts, l'architecture, le principe de fonctionnement....ainsi que les outils de simulation et de visualisation (2D et 3D). Le dernier chapitre est dédié à l'application sur le robot Summit-XL de la société Robotnik, en exploitant la simulation, les composants et les algorithmes de suivi prédéfinies sur ROS.

Chapitre 1

État de l'art de la robotique mobile

1.1 Concepts et définition

Jusqu'à nos jours, il est difficile de préciser le concept du robot car il n'hérite pas d'une approche scientifique mais des traditions culturelles et religieuses. Il a été mis en valeur depuis l'apparition du mot lui-même vers 1920 par une littérature de fiction imposante. Après, la transposition de ce concept à des travaux scientifiques conduirait à faire porter les efforts sur la réalisation d'un véritable homme artificiel, caractérisé principalement par la présence d'une intelligence humaine incluant la volonté et la conscience [1].
Au début, les applications de la robotique étaient limitées aux usines, plus précisément à des bras manipulateurs fixes en terme de tâche et en terme d'endroit (soudage, peinture...). En 1970, les premiers robots mobiles ont été réalisés, notamment le robot Shakey développé, par l'institut de recherche de Stanford (voir Figure 1.1). Actuellement, on rencontre de plus en plus des robots mobiles capables d'effectuer une mission spécifique sur un lieu ou dans un endroit donné.

Un robot mobile est donc un système mécanique, électronique et informatique qui agit physiquement dans son environnement en vue d'atteindre un objectif qui lui a été assigné. Cette machine polyvalente est capable de s'adapter à certains variations de ses conditions de fonctionnement. Elle est dotée de fonction de perception, de décision et d'action. Ainsi, le robot devrait être capable d'effectuer, de différentes manières, des tâches diverses et d'accomplir correctement sa propre tâche, même s'il rencontre de nouvelles situations inopinées [2].

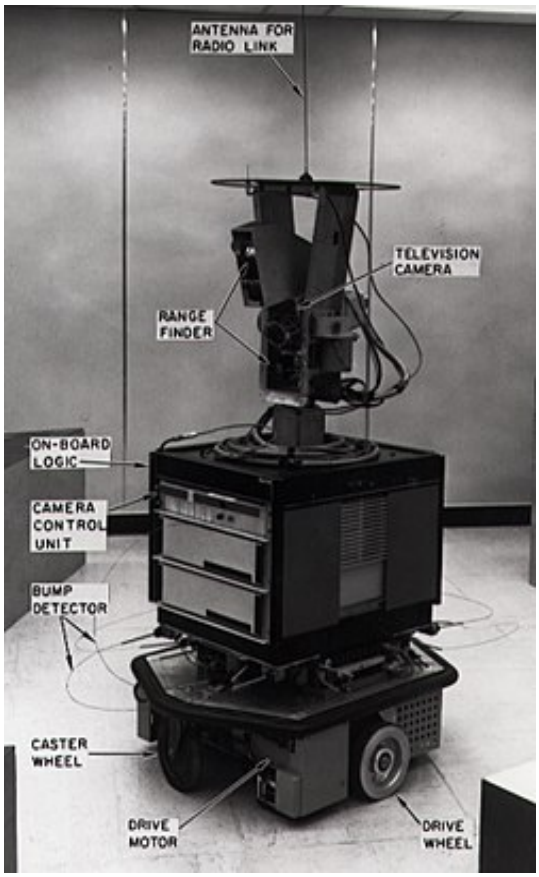


FIGURE 1.1 – Robot Shaackey

1.2 Composants matériels d'un robot mobile

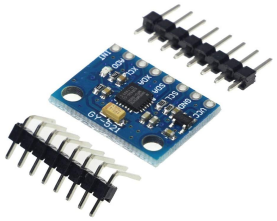
Les robots mobiles partagent la même architecture de base avec tous les systèmes automatisés -acquérir des données, les traiter et réagir par des actions-, les composants principaux sont détaillés ci-dessous :

1.2.1 Les capteurs

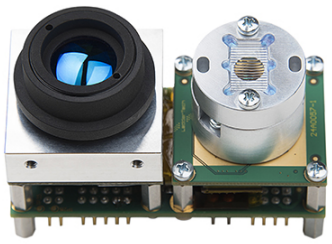
Deux types d'informations sont importants pour la commande des robots mobiles ; des informations proprioceptives (sur l'état interne du robot) et des informations extéroceptives (sur l'environnement). Par la suite, une liste des capteurs les plus utilisés[3] :

Capteurs proprioceptives

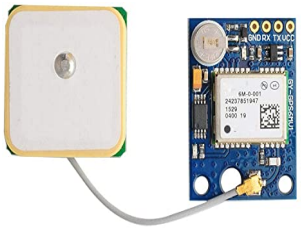
- **L'odomètre** : C'est généralement un encodeur incrémental intégrés avec les moteurs, qui donne l'information sur la distance parcourue en mesurant le nombre des tours d'actionneur.
- **capteurs niveau de batterie** : Un capteur de tension qui donne une information sur l'énergie emmagasinée dans la batterie, il est très important dans les robots aérien.
- **Capteurs du courant** : Ils mesurent en permanence les courants absorbés par les actionneurs, ces courants sont utilisés pour les algorithmes de commande ou pour la sécurité.
- **Accéléromètre et gyroscope** : Pour donner les informations sur la vitesse et l'accélération angulaires et linéaires (Figure 1.2a).



(a) Module MPU 6030



(b) LIDAR pour drone



(c) LIDAR pour drone

FIGURE 1.2 – Quelques capteurs des robots mobiles

Capteurs extéroceptives

— **Les capteurs tactiles** qui sont souvent utilisés pour des arrêts d'urgence lorsqu'ils rencontrent un obstacle imprévu.
— **Les boussoles** servent à mesurer le champ magnétique terrestre, pour générer une référence stable au cours du temps.
— **Le GPS** est un système de balises universel dont les balises sont placées sur des satellites en orbite terrestre. Ce système permet donc d'avoir une mesure de la position dans un repère global couvrant la terre avec une précision variant de quelques dizaines de mètres à quelques centimètres suivant les équipements utilisés (Figure 1.2c).
— **LASER et LIDAR** ceux sont des capteurs très important pour que le robot reçoive des informations précises sur les obstacles dans son environnement (Figure1.2b).

1.2.2 Les actionneurs

C'est la partie qui permet au robot à se déplacer dans des différents endroits ; ils sont généralement des moteurs qui actionnent des roues, des chenilles ou des hélices. Les moteurs synchrones "brushless" sont plus adoptés à cause de leurs puissance.

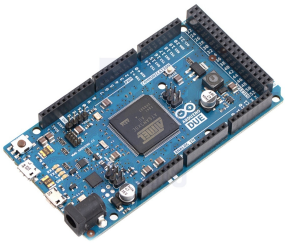
(a) Moteurs et hélices pour drone

(b) Roues avec moteurs

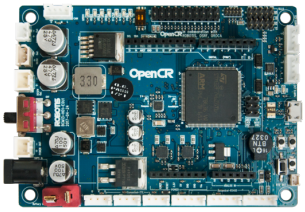
FIGURE 1.3 – Exemples des actionneurs

1.2.3 Les CPUs

Les unités centrales de calculs comme dans tous les systèmes ; exécutent les algorithmes de commande prédéfinies, traitent les données des capteurs, fournissent des ordres aux actionneurs, en bref, leur rôle est de maintenir le bon fonctionnement du robot. En effet, lors de la réalisation d'un robot plusieurs choix sont possibles, selon la complexité d'application, on peut prendre des micro-contrôleurs simples comme l'un des Arduino (Figure1.4a), ou un microprocesseur dédié comme OpenCR (Figure 1.4b), si on procède à une autonomie très élevée, ou encore des carte FPGA.



(a) Arduino Due



(b) OpenCR

FIGURE 1.4 – Exemples des CPU

1.2.4 Les sources d'énergie

Vue la mobilité élevée des robots mobiles et la consommation de leurs actionneurs, il important de trouver des solutions pour la durée de vie des batterie, d'autres solution énergétique sont en développement, les super-condensateurs sont en fait une technologie très importante dans le domaine.

(a) Batterie lipo 6000mha

(b) Super-condensateur

FIGURE 1.5 – Exemples des sources d'énergie

1.3 Autonomie d'un robot

Le robot mobile autonome est un être physique qui percepts, décide, actionne et réalise des tâches dans son environnement ; ces tâches et interactions sont faites rationnellement. En fait, l'autonomie d'une machine peut être résumée à chercher une commande pour l'envoyer aux actionneurs à chaque fraction de temps tout en considérant le but à atteindre et les données des différents capteurs. Le robot doit accomplir les tâches suivantes[3] :

— **Percevoir** : Le robot doit acquérir des informations sur son environnement issue de ses capteurs. Ces informations permettent de mettre à jour un modèle de l'environnement (architectures hiérarchiques ou délibératives) ou peuvent être directement utilisées comme entrées de comportement de bas niveau (architectures purement réactives).
— **Décider** : Le robot doit définir des séquences d'actions résultant d'un raisonnement appliqué sur un modèle de l'environnement ou répondant de manière réflexe à des stimuli étroitement liés aux capteurs.

— **Agir** : Il doit exécuter les séquences d'actions élaborées en envoyant des consignes aux actionneurs par l'intermédiaire des boucles d'asservissements.

1.4 Classification des Robots

Le domaine de la robotique mobile est en développement continu et très rapide, c'est pourquoi il difficile de normaliser la classification des robots mobiles. En effet, il existe plusieurs critères pour la classification ; l'endroit de mobilité, le domaines d'application, le types de commande et l'architecture mécanique ...
Par la suite, on cite les trois catégories selon l'endroit où se mobilise le robot ; la terre, l'air ou l'eau. Pour chacune de ces classes, il existe plusieurs modèles et plein d'applications. Dans ce mémoire, on se concentre sur les robots terrestres.

1.4.1 UAV Unmanned Aerial Vehicle

Les améliorations apportées aux technologies de détection et de contrôle ont permis aux drones d'acquérir une importance sans précédent dans un large éventail d'applications non militaires. Cela est particulièrement vrai pour les unités à rotors multiples en raison de leur faible coût et de leur grande flexibilité (due à leur petite taille). Leur popularité est encore favorisée par les capacités de contrôle à distance qui peuvent réduire l'exposition de l'opérateur de l'aéronef à des environnements dangereux pour diverses formes de documentation et d'enquête. Selon une enquête réalisée par l'Association "Unmanned Vehicle Systems International" (AUVSI) sur 3136 exemptions aux États-Unis. Dans de nombreuses juridictions, l'utilisation est notamment limitée par les réglementations des autorités de l'aviation, par opposition à la viabilité commerciale. La figure 1.6a est la dernière version des drone Mavic.
Les drones peuvent être classés de plusieurs façons, selon l'utilisation civile ou militaire, la masse maximale au décollage (MTOW) et le type de portance (à voilure fixe ou multirotor).

(a) Mavic Air 2 (b) Drone militaire

FIGURE 1.6 – Quelques types des drones

Les drones à voilure fixe (Figure1.6b) sont plus stables et plus grands et ont des capacités de vol plus élevées, une plus grande capacité de charge utile et une meilleure endurance, mais au détriment d'un certain niveau d'agilité et de coût [4].

En question d'automatique, c'est le type le plus difficile à modéliser et à commander vue le nombre élevé des degrés de liberté illustré par la figure 1.7.

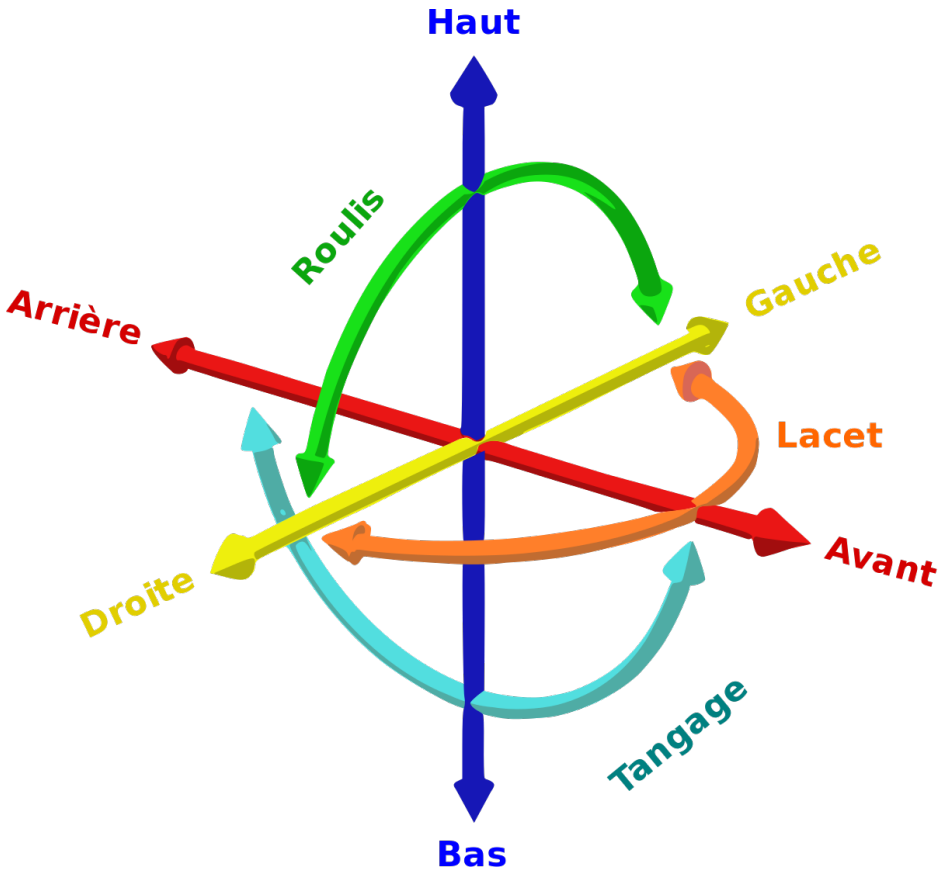


FIGURE 1.7 – Les degrés de liberté

1.4.2 AUVAutonumous Underwater Vehicle

Un véhicule sous-marin autonome (AUV) est un robot mobile sans équipage qui se déplace sous l'eau. Dans les applications militaires, les AUV sont également connus sous le nom de véhicules sous-marins sans équipage (UUV) (Figure1.8). Les AUV jouent un rôle important à l'exploration et l'exploitation de ressources situées en milieu océanique profond. Semblablement aux UAV les AUV ont les six degrés de liberté, en plus les conditions physique d'un milieu aquatique (la viscosité, la pousser darchimede) sont pas facilement négligeables, ce qui rend leurs modélisations et commande plus difficiles.

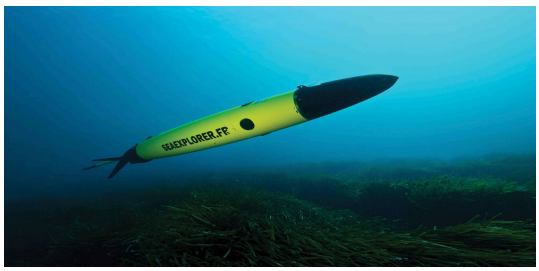


FIGURE 1.8 – Robot sous marin sea-explorer

1.4.3 UGV Unmanned Ground Vehicle

C'est la catégorie dont notre travail est autour, en ajout au robot à roues tel que les exemples de la figure 1-9 il existe d'autres modèles qui peuvent être considérés des robots terrestres ; on peut citer les robots à chenilles (Figure 1.9b), les robot à pattes (Figure1.9a), et les robots humanoïdes. Pour les robots à roues on peut trouver plusieurs types selon le nombre des roues, le système de braquage, la taille...etc

(a) Robot à pattes de Boston dynamics

(b) Robot à chenille

FIGURE 1.9 – Exemples de UGV

1.5 Domaines d'utilisation

Il est très difficile de limiter les applications des robots mobiles en quelques tirets, ci-dessus sont les plus connues :

— **Commerce** : L'assistance des clients, la livraison et la gestion des stocks sont des activités où on trouve des robot mobiles. L'exemple le plus connu est les stocks de Amazon qui sont totalement gérés par des robots mobiles (Voir Figure 1.10a).

— **Militaire** : Le domaine militaires souvent donne naissance à des nouvelles technologies et c'est le cas de plusieurs robots mobiles. Bien que les drones sont très utilisés dans ce domaine, on trouve aussi d'autres robots à roues ou hybrides pour le transport des armes, l'espionnage et l'intervention (1.10b).

— **Recherche** : Les robots mobiles sont largement utilisés dans plusieurs domaines de recherche ; l'intelligence artificielle, le diagnostic ou encore l'exploitation des profondeurs des océans ou de l'espace comme le robot de la figure 1.10c.

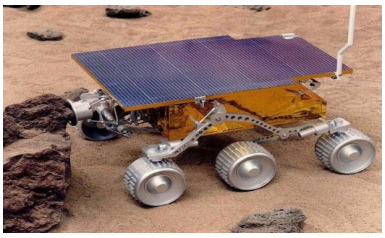
— **Agriculture** : Dans l'agriculture moderne, on trouve de plus en plus de nouvelles technologies et des innovations tel que le robot dans la figure 1.10d.

— **Industrie** : Plusieurs tâches sont faites à l'aide des robots mobiles ; collecte dans des entrepôts, inspection dans des usines, dans des zones à risque catastrophique (sites contaminés ou radioactifs).
— **Loisirs** : Plusieurs jouets sont en fait des robots mobiles, de plus il existe plusieurs compétitions en robotique qui attirent des milliers de suiveurs chaque année.

— **Cinématographie** : La production des films -plus précisément les documentaires de nature- est très évoluée grâce à l'utilisation des robots.

(a) Les robots Amazon

(b) Robot militaire



(c) Le robot mobile Sejourner, pathfinder de la NASA[2]

(d) le Robot MARS de FENDIT

FIGURE 1.10 – Exemples des applications des robots mobiles

1.6 Modélisation des robots mobile à roues

Comme il est mentionné dans la section 1.4.3, notre étude se concentre sur la denière catégorie, et par conséquent, on doit passer par la modélisation pour procéder aux différentes commandes. Pour simplifier les modèles on ne considère pas quelques conditions réels tels que les frottement, les reliefs et la saturation des actionneurs. De plus les robots équipés par un système de braquage indépendant ne sont pas pris en considération.
En effet, le robot à deux roues actionnées séparément et l'architecture la plus simple à modéliser, elle est la base de modélisation d'autres architecture compliquées.
En contrôlant les vitesses de rotation des deux roues et notamment en jouant sur la différence de vitesse, plusieurs manières de déplacement sont possible : vers l'avant, vers l'arrière, à droite, et à gauche. Il est même possible de tourner sur place. Un schéma illustratif des mouvements du robot et des vitesses asso-

ciées est donné à la figure 1.11[5].

Il est important à noter que ce robot à roues a un nombre limité de degré de liberté (Figure 1.7) ; contrairement au drone qui a six degrés de liberté, le robot dit mono-cycle n'est libre qu'au lacet et l'axe x (avant et arrière).

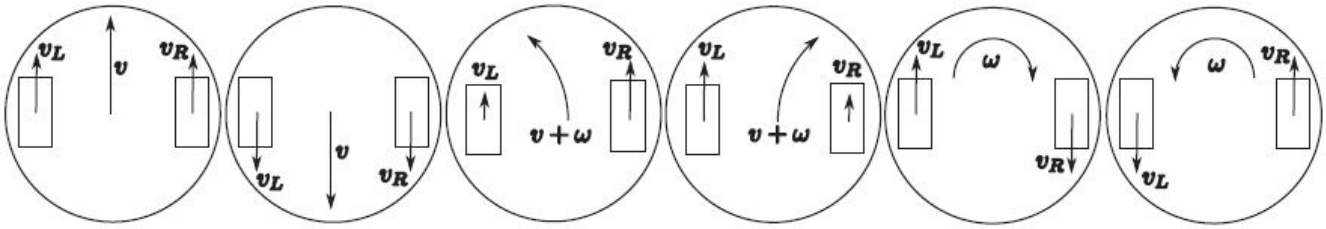


FIGURE 1.11 – Les déplacements possibles du robot

1.6.1 Modèle cinématique mathématique

A l'instant d'échantillonnage k , on considère comme variables d'état la position et l'orientation par rapport au repère global fixe :

$$X_k = [x \ y \ \theta]_k^T \tag{1.1}$$

La figure1.12 schématise le robot dans le référentiel :

Ainsi, le modèle du robot doit connecter les entrées, qui sont V_r et V_l aux états, d'une manière ou d'une autre. Il faut donc trouver un moyen de faire cette transition. C'est le modèle de robot à entraînement différentiel qui assure cette transition et il est donné par :

$$\begin{cases} \dot{x} = \frac{R}{2}(V_r + V_l)\cos(\theta) \\ \dot{y} = \frac{R}{2}(V_r + V_l)\sin(\theta) \\ \dot{\theta} = \frac{R}{L}(V_r - V_l) \end{cases} \tag{1.2}$$

Où est le R rayon de la roue et L est la distance entre les deux roues.
Le problème dans ce modèle est que c'est contre naturel de penser en termes de vitesses des différentes roues. Cependant, il est évident d'avoir autres entrées de commande qui sont plus faciles à comprendre et manipuler et qui restent dépendantes de V_l et V_r . C'est pourquoi, au lieu d'utiliser directement le modèle ci-dessus, nous allons passer à ce qu'on appelle le modèle monocycle. Ainsi, les nouvelles entrées du système seront les vitesses linéaires et angulaires du robot (v et$w$ respectivement). On aboutit donc à :

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = w \end{cases} \tag{1.3}$$

1.6.2 Modèle cinématique odométrique

L'autre modèle d'état du robot qui est plus utilisé se base sur l'odométrie. C'est une technique de localisation relative qui repose sur les mesures fournies par les encodeurs incrémentaux des roues du robot afin de déterminer son déplacement et sa rotation élémentaires.

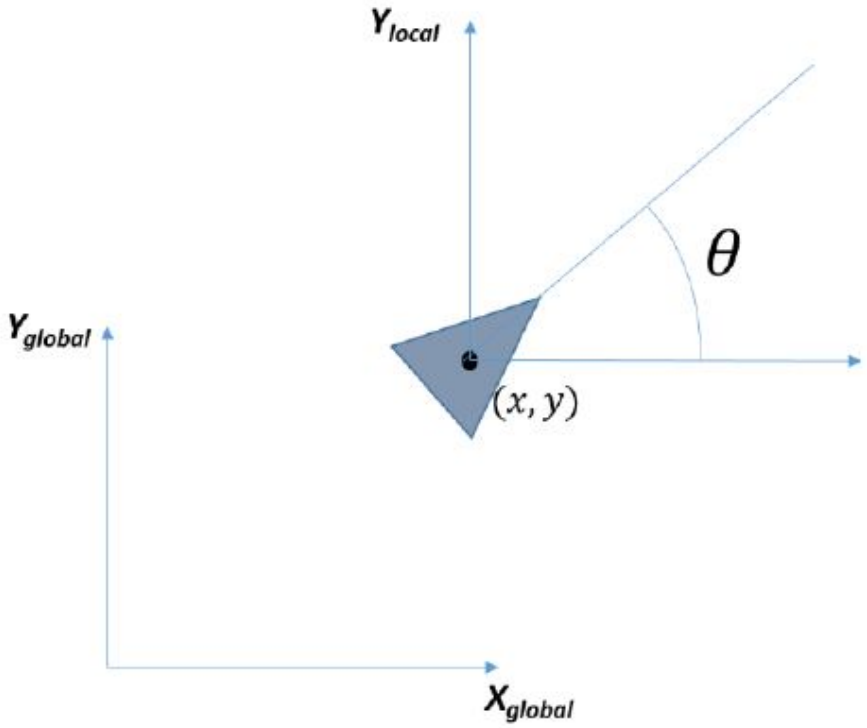


FIGURE 1.12 – Robot différentiel [6]

$$\begin{cases} x_{k+1} = x_k + D_c \cos(\theta) \\ y_{k+1} = y_k + D_c \sin(\theta) \\ \theta_{k+1} = \theta_k + \frac{D_r - D_l}{L} \end{cases} \tag{1.4}$$

$$D_c = \frac{D_r + D_l}{2} \tag{1.5}$$

Où D_r et D_l sont les distances parcourues par la roue droite et la roues gauche respectivement ; ces distances doivent être calculer en utilisant les signaux fournis par les encodeurs incrémentaux. Puisque cette méthode se base sur des calculs incrémentaux elle a un point faible au cours du temps qui est l'accumulation des erreurs.

1.7 Recherches et travaux actuels

Les grands axes de recherche dans la robotique mobile. Actuellement, des milliards sont réservés pour la recherche et l'innovation dans la robotique mobile.

- Les robot moues
- "bio-inspired robots"
- Les voitures autonomes
- AI
- La réalité augmenté et virtuelle

— Le consensus

1.8 Conclusion

Au cours de ce chapitre, nous avons exposé un aperçu général sur la robotique mobile. Nous avons présenté les composants matériels à savoir les capteurs, les actionneurs, les CPU ... suivi d'une classification des robots et de leur domaine d'utilisation. Enfin deux modèles cinématiques, selon que l'on veut déterminer la position relative d'un robot par rapport à son point de départ ou non, ont été abordés. Le chapitre suivant sera consacré à la présentation d'une plateforme de développement logicielle pour la robotique connu sous le nom de ROS et qui sera adopté dans notre étude.

Chapitre 2

ROS (Robotic Operating System)

2.1 Introduction

En lisant le premier chapitre, il est clair que la robotique englobe plusieurs domaines ; la mécanique, l'électronique, l'informatique, la biologie etc. Les spécialistes multidisciplinaires avaient besoins d'un outil pour simplifier leur tâche, pour que l'informaticien ne doive plus connaitre les détails de l'électronique d'un robot et faciliter la programmation pour les mécaniciens. ROS est actuellement l'outil le plus utilisé au niveau mondiale en robotique, pas seulement dans la recherche scientifique réalisés sur des systèmes tels que les robots mobiles et humanoïde ou les drones, mais aussi dans la gestion des produit commerciaux, personnelles et même militaires. Ce chapitre explique les bases, le mode de fonctionnement et l'importance du ROS.

2.2 Définitions

Avant de parler sur ROS, il est important de définir les notions suivants :

— **Système d'exploitation**
En informatique, un système d'exploitation ou encore OS "Operating System", est un ensemble de programmes qui relie les différentes ressources matérielles, les applications et l'utilisateur, qui constitue en général les composants de tout écosystème. La figure 2.1 résume les trois catégories des OS les plus communs avec leurs écosystèmes.

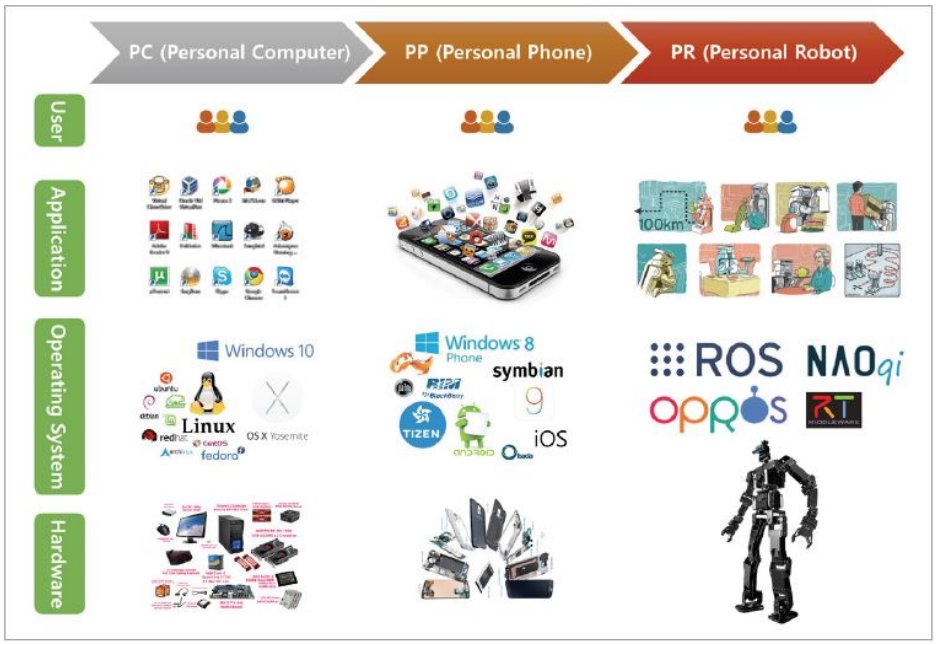


FIGURE 2.1 – Les composants d'un système d'exploitation

Dans un ordinateur, le système d'exploitation est responsable de gérer le processeur et la mémoire vive, optimiser l'exécution des applications en leur attribuant les ressources nécessaires et fournir les informations sur le bon fonctionnement de l'ordinateur. Il assure aussi l'utilisation des périphériques dans les meilleures conditions et protège l'accès aux ressources.

— **Logiciel**
Formé à partir des mots logique et matériel, le mot logiciel a été inventé en 1969 pour être la traduction officielle du terme anglais "software". Il désigne l'ensemble des programmes et des procédures nécessaires au fonctionnement d'un système informatique ; c'est lui qui indique à l'ordinateur comment effectuer les tâches. Chaque logiciel est conçu pour fonctionner dans un environnement matériel donné.

— **midlleware**
Un terme anglais dont la traduction adopté est l'intergiciel, c'est un logiciel intermédiaire dédié à la relation entre plusieurs applications informatiques de haut niveau[7].

— **Open source**
Un logiciel Open Source est un programme informatique dont le code source est distribué sous une licence permettant à quiconque de lire, modifier ou redistribuer ce logiciel. Cette notion n'est plus limitée à des logiciels, elle existe aussi pour des produits matériels tel que Arduino.

— **GUI**
La GUI "Graphical User Interface" désigne en français une interface graphique. Il s'agit de la manière selon laquelle un logiciel est présenté à un utilisateur sur un écran. Pour rendre simple, l'interface graphique, ou GUI, se résume à l'affichage des commandes permettant d'effectuer des actions dans un logiciel, comme des menus, des boutons, des fonctionnalités, etc., sans avoir besoin à saisir des lignes de commandes.

— **API**
Un API est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications. API est un acronyme anglais qui signifie « Application Programming Interface » que l'on traduit par interface de programmation d'application.

— **IDE**
Dans le domaine du développement informatique, l'IDE "Integrated Development Environment" regroupe un ensemble d'outils spécifiques. Ceux-ci sont dédiés aux programmeurs afin qu'ils puissent optimiser leur temps de travail et améliorer leur productivité. Autrement dit, l'IDE facilite la mise en œuvre de projets tels que le développement de logiciels ou d'applications, on peut citer visual studio, codeblocks et eclipse.

— **ROS**
ROS est définie par le site officiel comme suite :
*"**ROS** est un système de méta-opération à code source ouvert pour votre robot. Il fournit les services que vous attendez d'un système d'exploitation, y compris l'abstraction de matériel, le contrôle des dispositifs de bas niveau, la mise en œuvre de fonctionnalités d'usage courant, le passage de messages entre les processus et la gestion des paquets. Il fournit également des outils et des bibliothèques permettant d'obtenir, de construire, d'écrire et d'exécuter du code sur plusieurs ordinateurs[8]".*

2.3 Historique du ROS

Avant 2007, les premiers pas du projet ROS ont commencé à l'université de Stanford aux États Unis. Eric Berger et Keenan Wyrobek deux doctorants qui travaillent sur un projet de robotique, ils ont remar-

qué que leurs collègues multi-disciplinaires trouvent des difficultés pour s'adapter avec d'autres domaines importants pour la robotique, d'où vient l'idée pour trouver une solution innovante pour résoudre ce problème. Pour décrire leur idée, Eric Berger a dit : " Quelque chose qui ne craignait pas, dans toutes ces différentes dimensions". Quelques mois avant l'annonce officielle du ROS, la start-up a été incubé par "Willow Garage" ; une grande société des robots personnels et des services. Grâce au concept "open source", ROS Willow Garage a tenté de pénétrer le marché des robots de service commercial en 2013, mais a fini par se scinder en plusieurs start-ups, et a finalement été remis à l'Open Source Robotics Foundation (OSRF)[8]. La figure 2.2 est met en ordre chronologique les versions du ROS.

FIGURE 2.2 – Les versions du ROS

2.4 Pourquoi utiliser ROS ?

Une question fréquente dont la réponse courte est *" Pour réduire le temps du développement "* . Mais on peut répondre par les 5 principales caractéristiques du ROS qui sont :

1. **Réutilisable** : Un utilisateur peut développer la fonction qu'il souhaite et télécharger le "package" correspondant pour les autres fonctions. Après, il peut partager le programme qu'il a développé afin que d'autres puissent le réutiliser. L'exemple à mentionner est le Robonaut29 de NASA dont les développeurs ont utilisé ROS [8].

2. **Basé sur la communication** : Le ROS est un programme basé sur la communication. Souvent, afin de fournir un service, des programmes tels que les pilotes de matériel pour les capteurs et les actionneurs et des caractéristiques telles que la détection, la reconnaissance et le fonctionnement

sont développés dans un cadre unique. Toutefois, pour permettre la réutilisation des logiciels des robots, chaque programme et chaque fonction est divisé en plusieurs parties plus petites en fonction de son rôle. C'est ce qu'on appelle la "componentization" ou "modularisation" selon la plate-forme. Les données doivent être échangées entre les nœuds (un processus qui effectue des calculs dans le ROS) qui sont divisés en unités de fonctions minimales, et les plateformes disposent de toutes les informations nécessaires pour l'échange de données entre les nœuds. La programmation en réseau, qui est très utile pour le contrôle à distance, devient possible lorsque la communication entre les nœuds est basée sur le réseau, de sorte que les nœuds ne sont pas limités par le matériel. Le concept de fonctions minimales connectées au réseau est également appliqué à l'Internet des objets (IoT), de sorte que les ROS peuvent remplacer les plateformes IoT. Il est remarquablement utile pour trouver des erreurs car les programmes qui sont divisés en fonctions minimales peuvent être débogués séparément [8].

3. **La disponibilité des outils de développement** : En effet, ROS fournit des outils de débogage, qu'on va les expliquer plus tard, un outil de visualisation 2D (rqt, rqt est un cadre logiciel de ROS qui implémente les différents outils d'interface graphique sous forme de plugins) et un outil de visualisation 3D (RViz) qui peut être utilisé sans développer les outils nécessaires au développement de robots. Par exemple, il y a de nombreuses occasions où un modèle de robot doit être visualisé pendant le développement d'un robot. Le simple fait de correspondre au format de message prédéfini permet aux utilisateurs non seulement de vérifier directement le modèle du robot, mais aussi d'effectuer une simulation à l'aide du simulateur 3D fourni (Gazebo). L'outil peut également recevoir des informations de distance en 3D d'Intel RealSense ou de Microsoft Kinect récemment mis en lumière et les convertir facilement en forme de nuage de points, et les afficher sur l'outil de visualisation. En outre, il peut également enregistrer les données acquises au cours des expériences et les rejouer chaque fois qu'il est nécessaire de recréer l'environnement exact de l'expérience. Comme indiqué ci-dessus, l'une des caractéristiques les plus importantes du ROS est qu'il fournit les outils logiciels nécessaires au développement de robots, ce qui maximise la commodité du développement[8].

4. **La communauté active** : Le monde universitaire et l'industrie du robot, qui étaient jusqu'à présent relativement fermés, sont en train de changer de direction pour mettre l'accent sur la collaboration grâce à ces fonctions mentionnées précédemment. Indépendamment de la différence d'objectifs individuels, la collaboration par le biais de ces plates-formes logicielles se produit réellement. Au centre de ce changement, il y a une communauté pour les logiciels "Open source". Dans le cas de ROS, il existe plus de 5 000 paquets "package" qui ont été volontairement développés et partagés à partir de 2017, et les pages Wiki qui expliquent leurs paquets dépassent les 18 000 pages par la contribution des utilisateurs individuels. En outre, une autre partie essentielle de la communauté, à savoir les questions-réponses, a dépassé les 36 000 messages, créant ainsi une communauté collaborative en pleine croissance. La communauté ne se contente pas de discuter des instructions, mais s'efforce de trouver les composants nécessaires des logiciels de robotique et d'en établir la réglementation. En outre, elle progresse jusqu'à un stade où les utilisateurs se réunissent et réfléchissent à ce que les logiciels de robotique devraient impliquer pour l'avancement de la robotique et collaborent à fin de remplir les pièces manquantes du puzzle[8]. Les courbes dans la figure 2.3 montre comment augmente le nombre des utilisateurs au niveau mondial.

5. **La constitution de l'écosystème** : La révolution de la plateforme des smart-phones s'est produite parce qu'il existait un écosystème créé par des plateformes logicielles telles qu'Android ou iOS. Ce type de progression est également en cours pour le domaine de la robotique. Au début,

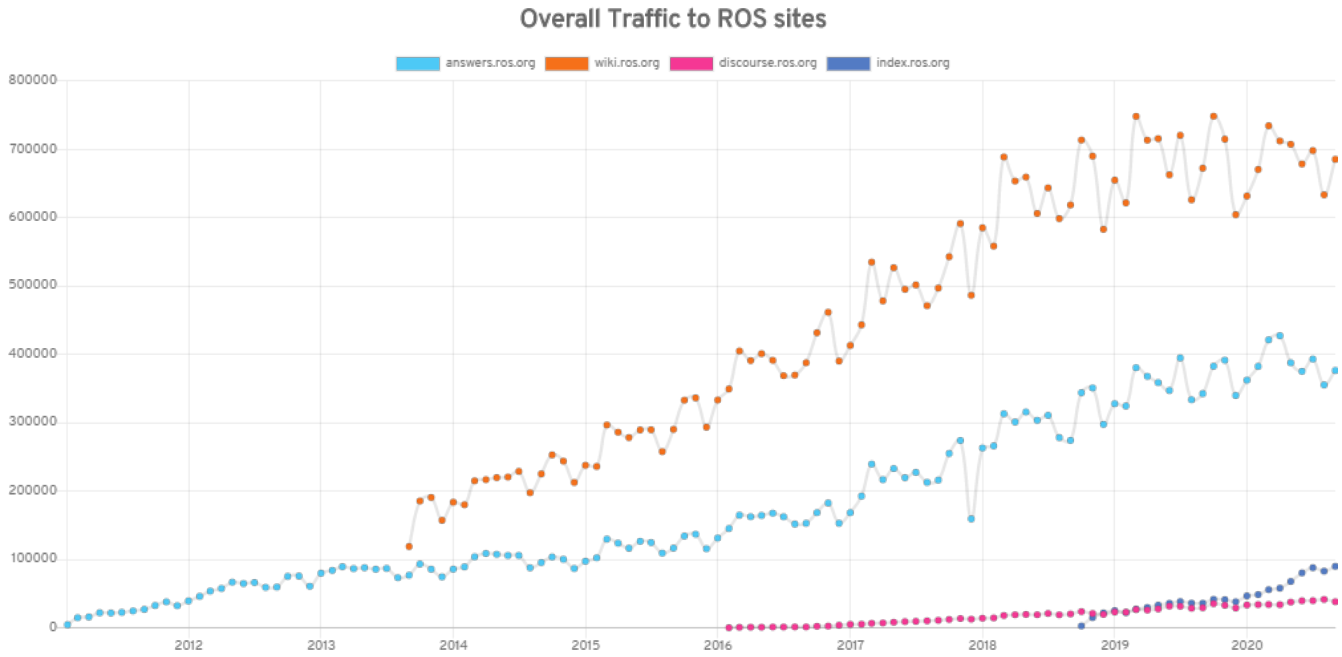


FIGURE 2.3 – Courbe de nombre des visiteurs

toutes les technologies matérielles débordaient, mais il n'y avait pas de système d'exploitation pour les intégrer. Diverses plateformes logicielles se sont développées et la plus estimée d'entre elles, ROS, est en train de façonner son écosystème. Elle crée un écosystème dont tout le monde -les développeurs de matériel du domaine de la robotique tels que les entreprises de robots et de capteurs, l'équipe opérationnelle de développement de ROS, les développeurs de logiciels d'application et les utilisateurs - peut être satisfait. Bien que les débutants soient encore marginaux, si l'on considère le nombre croissant d'utilisateurs et d'entreprises liées aux robots et la multiplication des outils et des bibliothèques connexes, on peut s'attendre à un écosystème vivant dans un avenir proche[8].

2.5 Architecture et principe de fonctionnement

2.5.1 Notion de "meta operating system"

ROS est l'abréviation de "Robot Operating System", on peut donc dire sans risque qu'il s'agit d'un système d'exploitation. En particulier, ceux qui sont nouveaux à ROS pourraient penser qu'il s'agit d'un système d'exploitation similaire aux systèmes d'exploitation classiques. Cependant, une description plus précise serait que le ROS est un système de méta-exploitation. Bien que le terme anglais "Meta-Operating System" ne soit pas défini dans le dictionnaire, il décrit un système qui effectue des processus tels que la planification, le chargement, la surveillance et le traitement des erreurs en utilisant la couche de virtualisation entre les applications et les ressources informatiques distribuées. Par conséquent, le ROS fonctionne dans un système d'exploitation existant tel que Ubuntu, qui est une des distributions de Linux et le plus recommandé par les experts. Après avoir terminé l'installation du ROS sur Ubuntu, les fonctionnalités fournies par le système d'exploitation tel que le système de gestion des processus, le système de fichiers, l'interface utilisateur et le programme l'utilité (compilateur, modèle de fil) peut être utilisée. En plus des

fonctionnalités de base fournies par Ubuntu, Le ROS fournit les fonctions essentielles requises pour les programmes d'application des robots, comme les bibliothèques, la transmission/réception de données entre des matériels hétérogènes, l'ordonnancement et le traitement des erreurs. Ce type de logiciel est également appelé intergiciel ou cadre logiciel [8].
Cette notion est illustrée par la figure 2.4.

FIGURE 2.4 – Notion de Meta OS

2.5.2 Terminologie et composants du ROS

Afin de bien comprendre ROS, plusieurs composants et notions doivent être clarifiés, qui sont détaillés dans le chapitre 4 de [8] .

Maître "Master"

Le maître agit comme un serveur de noms pour les connexions de nœud à nœud et la communication de messages. La commande "roscore" est utilisée pour exécuter le maître, et si vous exécutez le maître, vous pouvez enregistrer le nom de chaque nœud et obtenir des informations si nécessaire. La connexion entre les nœuds et la communication de messages tels que les "Topics" et les "services" -qui seront détaillés par la suite- sont impossibles sans le maître. Le maître communique avec les esclaves en utilisant XMLRPC (XML-Remote Procedure Call), qui est un protocole basé sur HTTP qui ne maintient pas la connectivité. En d'autres termes, les nœuds esclaves ne peuvent accéder que lorsqu'ils ont besoin d'enregistrer leurs propres informations ou de demander des informations à d'autres nœuds. L'état de connexion des uns et des autres n'est pas vérifié régulièrement. Grâce à cette caractéristique, le ROS peut être utilisé dans des environnements très vastes et complexes. XMLRPC est très léger et supporte une variété de langages de programmation, ce qui le rend bien adapté au ROS, qui supporte une variété de matériel et de langages de programmation. Lorsque vous exécutez un ROS, le maître sera configuré avec l'adresse URI et le port configuré dans le ROS_MASTER_URI. Par défaut, l'adresse URI utilise l'adresse IP de l'adresse locale PC, et le numéro de port 11311, sauf modification contraire.

Nœud "node"

Un nœud fait référence à la plus petite unité de processeur fonctionnant en ROS. Il s'agit d'un programme exécutable. ROS recommande de créer un seul nœud pour chaque objectif, et il est recommandé pour une réutilisation facile. Par exemple, dans le cas des robots mobiles, le programme permettant de faire fonctionner le robot est décomposé en fonctions spécialisées. Un nœud spécialisé est utilisé pour chaque fonction, telle que l'entraînement des capteurs, la conversion des données des capteurs, la reconnaissance des obstacles, l'entraînement des moteurs, l'entrée des encodeurs et la navigation. Au démarrage, un nœud enregistre des informations telles que le nom, le type de message, l'adresse URI et le numéro de port du nœud. Le nœud enregistré peut agir en tant qu'éditeur, abonné, serveur de service ou client de service sur la base des informations enregistrées, et les nœuds peuvent échanger des messages en utilisant des sujets "topics" et des services "services" (ces notions vont etre détaillées dans la section2.5.3) Le nœud utilise XMLRPC pour communiquer avec le maître et utilise XMLRPC ou TCPROS des protocoles TCP/IP pour communiquer entre les nœuds. La demande de connexion et la réponse entre les nœuds utilisent XMLRPC, et la communication des messages utilise TCPROS car il s'agit d'une communication directe entre les nœuds, indépendante du maître. Quant à l'adresse URI et au port numéro, une variable appelée ROS_HOSTNAME, qui est stockée sur l'ordinateur où se trouve le nœud est utilisée comme adresse URI, et le port est fixé à une valeur unique arbitraire.

Paquet "package"

Un paquet une unité de base du ROS. L'application ROS est développée sur la base de paquets, et le paquet contient soit un fichier de configuration pour lancer d'autres paquets ou nœuds. Le paquet contient également tous les fichiers nécessaires au fonctionnement du paquet, y compris les bibliothèques de dépendances ROS pour l'exécution de divers processus, les ensembles de données et le fichier de configuration. Le nombre de paquets officiels est d'environ 2 500 pour ROS Indigo en juillet 2017 (http ://repositories.ros.org/status_page/ ros_indigo_default.html) et d'environ 1 600 paquets pour ROS Kinetic http ://repositories.ros.org/status_page/ros_kinetic_default.html). En outre, bien qu'il puisse y avoir quelques redondances, il existe environ 4 600 paquets développés et diffusés par les utilisateurs (http ://rosindex.github.io/stats/).

Méta-paquet "metapackage"

Un méta-paquet est un ensemble de paquets qui ont un objectif commun. Par exemple, le métapaquet Navigation est composé de 10 paquets comprenant AMCL, DWA, EKF et map_server.

Message

Un nœud envoie ou reçoit des données entre les nœuds par l'intermédiaire d'un message. Les messages sont des variables telles que les nombres entiers, les nombres à virgule flottante et les booléens. Une structure de message imbriquée qui contient un autre message ou un ensemble de messages peut être utilisée dans le message. Le protocole de communication TCPROS et UDPROS est utilisé pour la livraison des messages.

Catkin

Le catkin fait référence au système de construction des ROS. Le système de compilation utilise essentiellement CMake (Cross Platform Make) et l'environnement de compilation est décrit dans le fichier "CMakeLists.txt" dans le dossier des paquets. CMake a été modifié dans ROS pour créer un système de

compilation spécifique à ROS. Catkin a lancé le test alpha à partir de ROS Fuerte et les paquets de base ont commencé à passer à Catkin dans la version ROS Groovy. Catkin a été appliqué à la plupart des paquets dans la version ROS Hydro. Le système de compilation Catkin facilite l'utilisation des compilations liées à ROS, la gestion des paquets et les dépendances entre les paquets. Si vous comptez utiliser ROS à ce stade, vous devriez utiliser Catkin au lieu de la compilation ROS (rosbuild).

Bag

Les données des messages ROS peuvent être enregistrées. Le format de fichier utilisé est appelé bag, et ".bag" est utilisé comme extension de fichier. Dans le ROS, bag peut être utilisé pour enregistrer les messages et les lire si nécessaire pour reproduire l'environnement dans lequel les messages sont enregistrés. Par exemple, lors d'une expérience de robot utilisant un capteur, les valeurs du capteur sont stockées dans le formulaire de message à l'aide du bag. Ce message enregistré peut être chargé à plusieurs reprises sans effectuer le même test en lisant le fichier enregistré dans le sac. Les fonctions d'enregistrement et de lecture de rosbag sont particulièrement utiles lors du développement d'un algorithme avec des modifications fréquentes du programme.

Wiki ROS

Le Wiki ROS est une description de base des ROS basée sur le Wiki (http ://wiki.ros.org/) qui explique chaque paquet et les fonctionnalités fournies par les ROS. Cette page Wiki décrit l'utilisation de base des ROS, une brève description de chaque paquet, les paramètres utilisés, l'auteur, la licence, la page d'accueil, le repository et le tutoriel. Le Wiki ROS contient actuellement plus de 18 800 pages de contenu.

Repository

Un paquet ouvert spécifie le dépôt dans la page Wiki. Le dépôt est une adresse URL sur le web où le paquet est enregistré. Le dépôt gère les problèmes, le développement, les téléchargements et d'autres fonctionnalités en utilisant des systèmes de contrôle de version tels que svn, hg et git. De nombreux paquets ROS actuellement disponibles utilisent GitHub comme dépôt de code source. Afin de visualiser le contenu du code source de chaque paquet, consultez le dépôt correspondant.

Graph

La relation entre les nœuds, les "topics", les "publisher" et les "subscribers" présentée ci-dessus peut être visualisée sous forme de graphique. La représentation graphique de la communication des messages n'inclut pas le service car il ne se produit qu'une seule fois. Le graphique peut être affiché en exécutant le nœud "rqt_graph" dans le paquet "rqt_graph". Il existe deux commandes d'exécution, "rqt_graph" et "rosrun rqt_graph rqt_graph".

Client Library

ROS fournit des environnements de développement pour différents langages en utilisant la bibliothèque client afin de réduire la dépendance vis-à-vis du langage utilisé. Les principales bibliothèques clients sont C++, Python, Lisp et d'autres langages tels que Java, Lua, .NET, EusLisp et R sont également pris en charge. À ce sujet, des bibliothèques clients telles que roscpp, rospy, roslisp, rosjava, roslua, roscs, roseus, PhaROS et rosR ont été développées.

CMakeLists.txt

Catkin, qui est le système de construction de ROS, utilise CMake par défaut. L'environnement de compilation est spécifié dans le fichier "CMakeLists.txt" dans chaque dossier de paquets.

package.xml

Un fichier XML contient des informations sur les paquets qui décrivent le nom du paquet, l'auteur, la licence et les paquets dépendants.

2.5.3 Modes de communication

Comme il est mentionné ROS est basé sur la communitcation entre les différentes nœuds, cette communication peut être établit en trois mode selon le besoin du développeur.

Topic

Ce mode est asynchrone, il est basée sur la notion du "publisher" et "subscriber". Comme les noms en Anglais l'indique, ce mode fonctionne comme suivant : Si les topics sont les routes permettant le transfert, la récupération, ou encore l'envoi de messages aux noeuds, alors les publishers et subscribers sont les bus transportant ces messages. C'est par leur biais que communiqueront les topics et noeuds. Ainsi, pour envoyer un message, un noeud utilisera un publisher pour transmettre son information au topic associé, s'il est possible de recevoir une information du composant relatif au noeud. Dans la même idée, un subscriber sera utilisé pour recevoir des informations venant d'un topic [9]. De plus, plusieurs nœuds peuvent être des subscribers sur le même topic, voir la figure 2.5.

Une fois la connexion est faite, le Topic transmit et reçoit les message en continue, c'est pour cela, ce

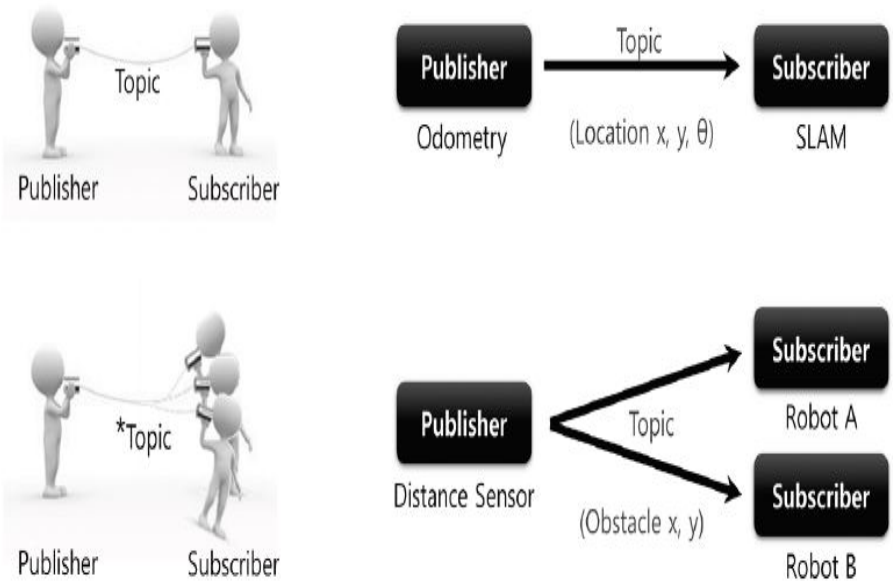


FIGURE 2.5 – Exemple de communication en mode Topic [8]

mode de communication est idéal pour les applications de mesure continue comme les radars.

Service

Dans ce mode illustré par la figure 2.6, la communication est synchrone entre le client "client" qui demande un service concernant une tâche particulière et le serveur "server" qui est chargé de répondre aux demandes. Le serveur ne répond qu'après la demande arrivant d'un client, le client attends la réponse, c'est la raison pour laquelle ce mode est dit synchrone. Une fois le client reçoit le message demandé la connexion sera interrompue, grâce à cette caractéristique le mode service est très important pour réduire l'encombrement du réseau.

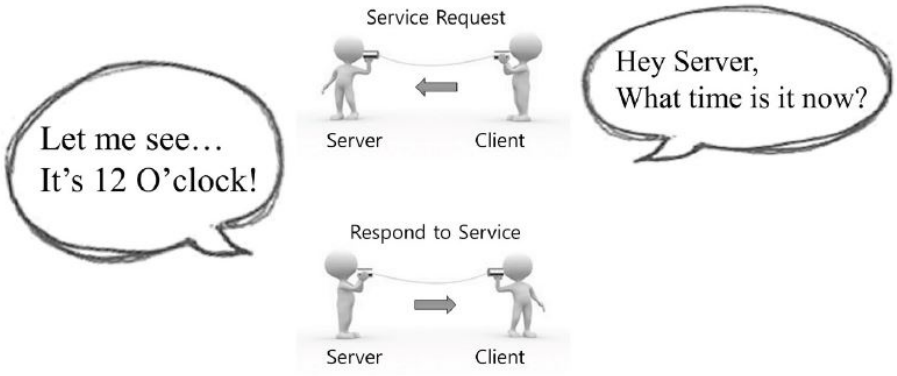


FIGURE 2.6 – Communication en mode Service [8]

Action

C'est une autre méthode de communication de messages utilisée pour une communication bidirectionnelle asynchrone. L'action est utilisée lorsqu'il faut plus de temps pour répondre après avoir reçu une demande et que des réponses intermédiaires sont nécessaires jusqu'à ce que le résultat soit renvoyé. La structure du fichier "Action" est également similaire à celle du service. Cependant, la section des données de retour d'information pour la réponse intermédiaire est ajoutée ainsi que la section des données d'objectif et de résultat qui sont présentées respectivement comme demande et réponse en service. Il existe des clients d'action qui fixent l'objectif de l'action et un serveur d'action qui exécute l'action spécifiée par l'objectif et renvoie un retour et un résultat au client d'action. un exemple illustratif est dans la figure 2.7.

Paramètres

La communication des messages est largement divisée en sujets, services et actions. Les paramètres sont les suivants variables globales utilisées dans les nœuds et dans le contexte plus large, elles peuvent également être considérées comme un message communication. La configuration est définie avec des valeurs par défaut et peut être lue ou écrite en externe si nécessaire. En particulier, les valeurs configurées peuvent être modifiées en temps réel de l'extérieur en utilisant la fonction d'écriture. Elle est très utile pour faire face avec souplesse à un environnement changeant.

Bien que les paramètres ne soient pas strictement une méthode de communication de messages, ils peuvent faire partie du champ d'application de la communication de messages. Par exemple, vous pouvez modifier les paramètres pour définir le port USB auquel se connecte, obtenir la valeur de correction des

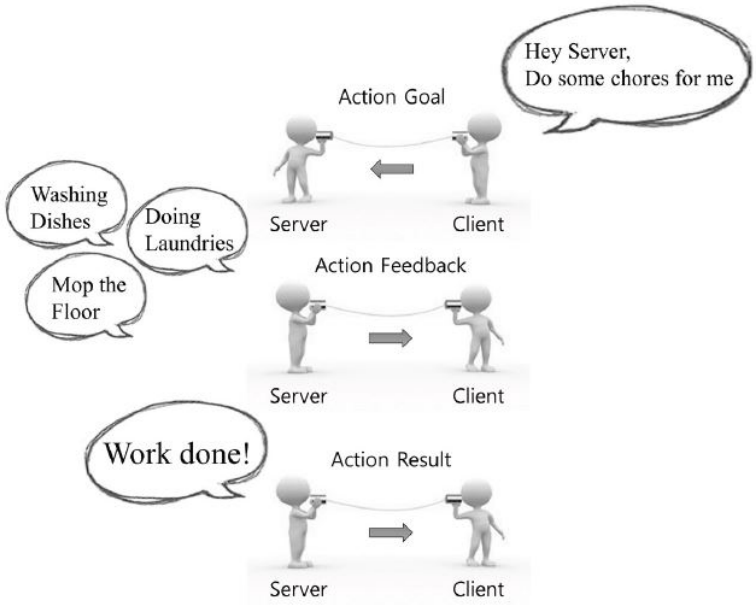


FIGURE 2.7 – Communication en mode Action [8]

couleurs de l'appareil photo, et configurer les valeurs maximales et minimales de la vitesse et des commandes.

La figure 2.8 illustre tous ces modes de communication.

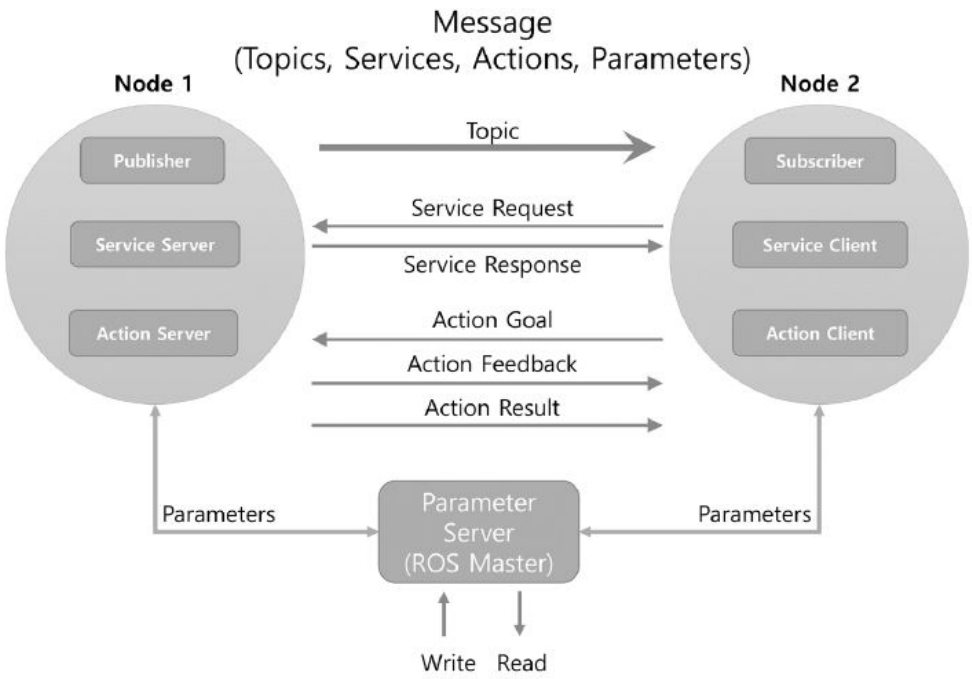


FIGURE 2.8 – Modes de communication [8]

2.6 ROS control

ROS Control est un aspect important du ROS ; c'est une couche qui rassemble tous les contrôleurs d'un robot. C'est notamment à ce niveau que les feedback sont établis pour assurer le contrôle du robot. A cette couche plusieurs contrôleurs sont définis, qu'ils sont possible d'utiliser. Par exemple, les velocity_controllers permettrons l'envoie direct d'une consigne de vitesse de roue aux moteurs dans le cas d'un robot à roues. Il existe plusieurs types de contrôleurs, nous pouvons les retrouver dans la documentation ROS comme dans la figure 2.9.

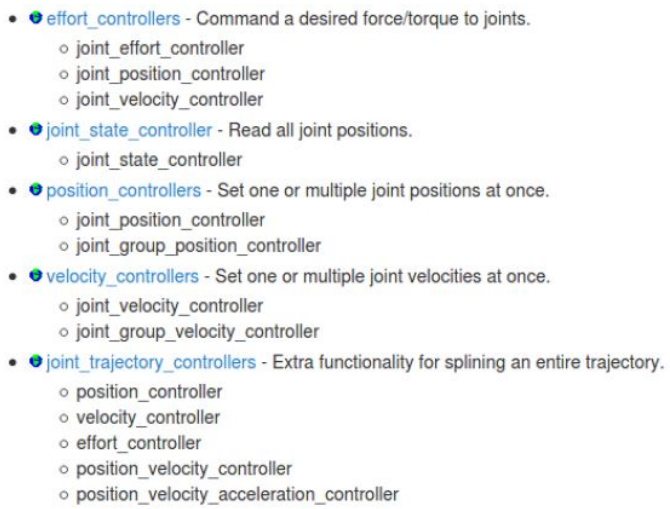


FIGURE 2.9 – Les contrôleurs du ROS control

L'architecture du ROS control et le flux des données entre ces composants sont illustrés par la figure 2.10.

En résumé, ROS Control nous permet de lancer divers contrôleurs ainsi que de les utiliser, permettant ainsi la communication avec les différents actionneurs du robot.

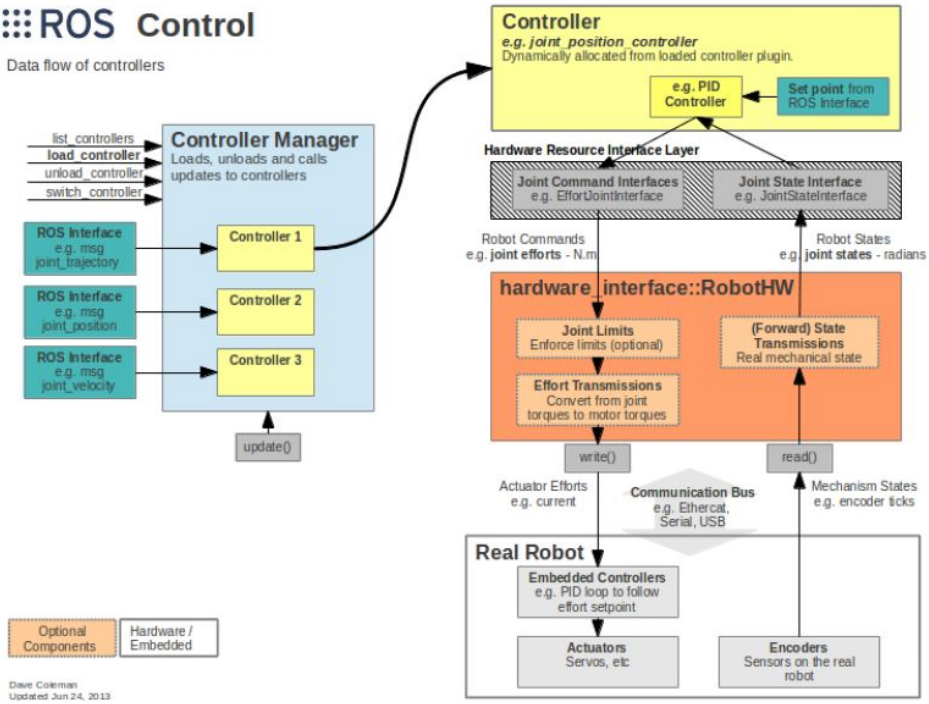


FIGURE 2.10 – Architecture du ROS control

2.7 Outils de développement

Comme nous l'avons déjà mentionné auparavant, ROS est une collection d'outils et d'algorithmes. Ces outils sont simple à utiliser et en mode "Open source". Certains sont très utilisés lors de la programmation, de la simulation ou de l'essai réel des robots. Citons quelques outils ou algorithmes que le programmeur de ROS retrouvera souvent, après on démontre l'utilisation de ces outils avec des applications dans le chapitre 3[10].

2.7.1 Rviz

RViz (Ros vizualisation) est l'outil de visualisation 3D de ROS. Le but principal est de montrer les messages ROS en 3D, ce qui nous permet de vérifier visuellement les données. Par exemple, il peut visualiser la distance entre le capteur (LDS) et un obstacle, les données de nuage des points (PCD) du capteur de distance 3D tel que RealSense, Kinect ou Xtion, la valeur de l'image obtenue à partir d'une caméra, et bien d'autres choses encore, sans avoir besoin de développer le logiciel séparément[8].

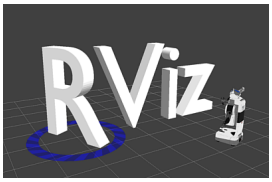


FIGURE 2.11 – Logo Rviz

Il prend également en charge diverses visualisations à l'aide de polygones spécifiés par l'utilisateur,

et Les marqueurs interactifs permettent aux utilisateurs d'effectuer des mouvements interactifs avec les commandes et les données reçues du nœud d'utilisateur. En outre, le ROS décrit les robots en format URDF (Unified Robot Description Format) qui est exprimé sous la forme d'un modèle 3D pour lequel chaque modèle peut être déplacé ou Les systèmes d'information sont conçus pour être utilisés en fonction de leur degré de liberté correspondant afin de pouvoir être utilisés à des fins de simulation ou de contrôle. Le modèle de robot mobile peut être affiché et les données de distance reçues du capteur de distance laser (LDS) peuvent être utilisées pour la navigation, comme le montre la figure 2.12. RViz peut également afficher l'image de la caméra montée sur le robot, comme indiqué dans le coin inférieur gauche de la figure 2.12. En outre, il peut recevoir des données de divers capteurs tels que Kinect, LDS, RealSense et les visualiser en 3D comme indiqué dans les images 2.13, 2.14 et 2.15.

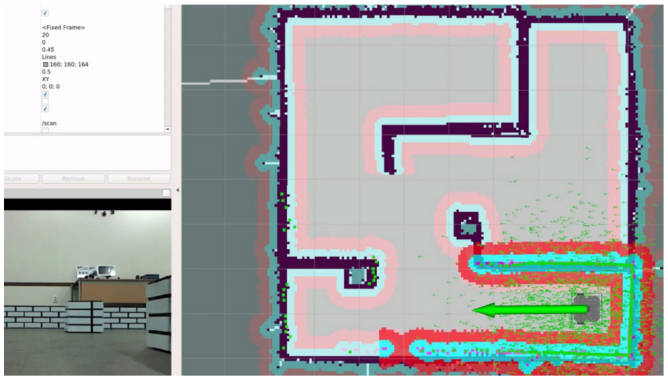


FIGURE 2.12 – Navigation deTurtleBot3 et LDS

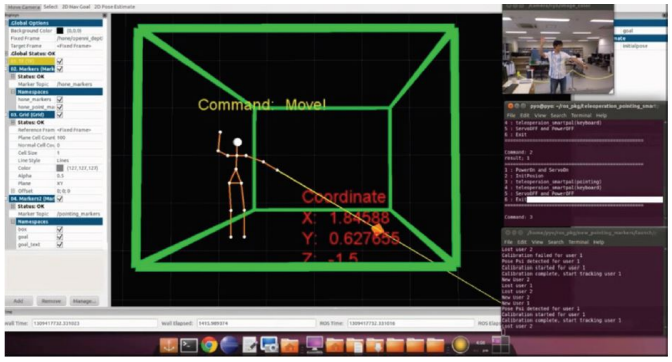


FIGURE 2.13 – Obtenir le squelette d'une personne en utilisant Kinect et Command avec une motion

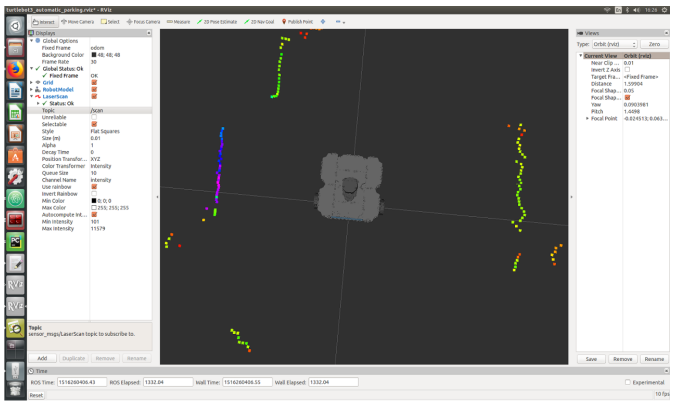


FIGURE 2.14 – Mesure de la distance à l'aide du LDS

FIGURE 2.15 – Distance, infrarouge, valeur de l'image couleur obtenue à partir de Intel RealSense

2.7.2 Gazebo

Gazebo offre la possibilité de simuler avec précision et efficacité des populations de robots dans des environnements intérieurs et extérieurs complexes. Au bout de vos doigts, vous disposez d'un moteur physique robuste, de graphiques de haute qualité et d'interfaces graphiques et programmatiques pratiques. Mieux encore, Gazebo est gratuit avec une communauté dynamique (source site officiel gazebosim.com). Ces caractéristiques sont :

— **Simulation de la dynamique**
Accédez à de multiples moteurs de physique haute performance, notamment ODE, Bullet, Simbody et DART.

— **Graphiques 3D avancés**
En utilisant OGRE, Gazebo fournit un rendu réaliste des environnements, y compris un éclairage, des ombres et des textures de haute qualité.

— **Capteurs et bruit**
Générer des données de capteurs, éventuellement avec du bruit, à partir de télémètres laser, de caméras 2D/3D, de capteurs de style Kinect, de capteurs de contact, de couple de force, etc.

FIGURE 2.16 – Gazebo logo

— **Plugins**
Développer des plugins personnalisés pour le contrôle des robots, des capteurs et de l'environnement. Les plugins fournissent un accès direct à l'API de Gazebo.

— **Modèles de robots**
De nombreux robots sont fournis, notamment PR2, Pioneer2 DX, iRobot Create et TurtleBot. Vous pouvez également construire votre propre robot en utilisant SDF.

— **Transport TCP/IP**
Exécuter la simulation sur des serveurs distants, et s'interfacer avec Gazebo par le biais d'un système de transmission de messages basé sur des sockets en utilisant Google Protobufs.

— **Simulation de nuages**
Utilisez CloudSim pour faire fonctionner Gazebo sur Amazon AWS et GzWeb pour interagir avec la simulation via un navigateur.

— **Outils en ligne de commande**
Des outils de ligne de commande étendus facilitent l'introspection et le contrôle de la simulation.

2.7.3 rqt

Outre l'outil de visualisation 3D RViz, ROS fournit divers outils d'interface graphique pour le développement des robots. Par exemple, il existe un outil graphique qui montre la hiérarchie de chaque nœud sous forme de diagramme, indiquant ainsi l'état du nœud et du "topic" en cours, et un outil de traçage qui schématise un message sous forme de graphique en 2D. À partir de la version ROS Fuerte, plus de 30 outils de développement d'interface graphique ont été intégrés en tant qu'outil appelé rqt qui peut être utilisé comme un outil GUI complet.

De plus, RViz a également été intégré comme un plugin de rqt, faisant de rqt un outil GUI essentiel pour les ROS. De plus, comme son nom l'indique, rqt a été développé sur la base de Qt, qui est un cadre multiplateforme largement utilisé pour la programmation d'interfaces graphiques, ce qui permet aux utilisateurs de développer et d'ajouter librement des plugins. Dans cette partie, nous allons découvrir les plugins "rqt" "rqt_image_view", "rqt_graph", "rqt_plot" et "rqt_bag".

Les plugins rqt sont :

- **Action**
  - action Navigator : Il s'agit d'un plugin permettant de vérifier la structure des données d'un type d'action.
- **Configuration**
  - dynamic Reconfiguration : Il s'agit d'un plugin permettant de modifier la valeur des paramètres d'un nœud.
  - Launch : Il s'agit d'un plugin d'interface graphique de roslaunch, qui est utile lorsque nous ne pouvons pas nous souvenir du nom ou la composition du roslaunch.
- **Introspection**
  - Node Graph : c'est un plugin pour la vue graphique qui nous permet de vérifier la relation diagramme des nœuds ou des flux de messages en cours.
  - Package Graph : c'est un plugin pour la vue graphique qui affiche les dépendances des paquets.
  - Process Monitor : Nous pouvons vérifier le PID (Processor ID), l'utilisation du processeur, l'utilisation de la mémoire et nombre de fils des nœuds actuellement en cours d'exécution.
- **Logging**
  - bag : Ceci est un plugin concernant l'enregistrement des données ROS.
  - Console : Il s'agit d'un plugin permettant de vérifier les messages d'avertissement et d'erreur qui se produisent dans les nœuds d'un même écran.
  - Logger Level : Il permet de sélectionner un enregistreur, qui est responsable de la publication des journaux, et de définir le niveau de l'enregistreur pour publier un journal spécifique tel que "Debug", "Info", "Warn", "Error" et "Fatal". Il est très pratique de sélectionner "Debug" pendant le processus de débogage.
- **Miscellaneous Tools**
  - Python Console : Il s'agit d'un plugin pour l'écran de la console Python.
  - Shell : Ce plugin lance un shell.
  - Web : Ce plugin lance un navigateur web.
- **Robot tools**
  - Contrôleur de gestion : C'est un plugin pour vérifier le statut, tapez, interface ardware les informations du contrôleur du robot.
  - Diagnostic Viewer : Il s'agit d'un plugin permettant de vérifier l'état et les erreurs du robot.
  - Moveit ! Moniteur : Il s'agit d'un plugin permettant de vérifier les données MoveIt ! utilisées pour la planification des mouvements.
  - Robot Steering : Il s'agit d'un outil GUI pour le contrôle manuel du robot, et cet outil GUI est utile pour contrôler un robot à distance.
  - Runtime Monitor : Il s'agit d'un plugin permettant de vérifier les avertissements ou les erreurs des nœuds en temps réel.
- **Services**
  - Service Caller :c'est un plugin GUI qui se connecte à un serveur de service en cours d'exécution et demande un service. Ceci est utile pour tester le service.
  - Service Type Browser : Pour la vérification de structure des données d'un type de service.

- **Topics**
  - Easy Message Publisher : Pour publier un Topic dans un environnement GUI.
  - Topic Publisher : Il s'agit d'un plugin qui permet de publier un sujet dans un environnement GUI. Il est utile pour tester les sujets.
  - Topic Type Browser : Il s'agit d'un plugin qui peut vérifier la structure des données d'un sujet.

Il s'agit de utile pour vérifier le type de sujet.
— Topic Monitor : Il s'agit d'un plugin qui liste les sujets actuellement utilisés et vérifie le des informations sur le thème choisi dans la liste.

— **Visualization**
  — Image View : Il s'agit d'un plugin qui permet de vérifier les données d'images provenant d'une caméra. Il est utile pour un simple test des données de l'appareil photo.
  — Navigation Viewer : Il s'agit d'un plugin permettant de vérifier la position ou le point de but du robot dans la avigation.
  — Plot : Il s'agit d'un plugin GUI permettant de tracer des données en 2D. Il est utile pour schématiser des données en 2D.
  — Pose View :Il s'agit d'un plugin permettant d'afficher la pose (position+orientation) d'un modèle de robot ou d'un TF.
  — RViz : C'est le plugin RViz qui est un outil de visualisation 3D.
  — TF Tree : Il s'agit d'un plugin de type vue graphique qui montre la relation de chaque coordonnée acquise à partir du TF dans l'arborescence.

2.7.4 Rosserial

Le rosserial est un ensemble qui convertit les messages, les sujets et les services ROS pour les utiliser dans une communication en série. En général, les micro-contrôleurs utilisent la communication série comme UART plutôt que TCP/IP qui est utilisé comme communication par défaut dans les ROS. Par conséquent, pour convertir la communication de messages entre un micro-contrôleur et un ordinateur utilisant des ROS, rosserial doit interpréter les messages pour chaque appareil.

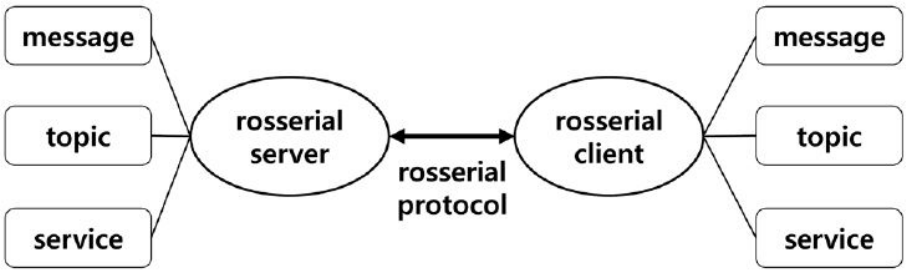


FIGURE 2.17 – Serveur rosserial (pour PC) et client (pour système embarqué)

Dans la figure 2.17, le PC qui exécute le ROS est un serveur rosserial et le micro-contrôleur connecté au PC devient le client rosserial. Le serveur et le client envoient et reçoivent tous les deux des données en utilisant le protocole rosserial, tout matériel capable d'envoyer et de recevoir des données peut être utilisé. Cela permet d'utiliser les messages ROS avec l'UART qui est souvent utilisé dans les micro-contrôleurs.

2.8 Limites du ROS

— ROS est conçu pour travailler sur un seul robot ; il n'existe pas des bibliothèque pour les systèmes multi-robots ou le consensus, donc, il faut développer vos propre outils lors des applications multi-robots.

— L'architecture du ROS nécessite un réseaux fiable et une large bande passante.
— L'exécution sur ROS n'est pas en temps réel.
— L'architecture Maître-esclave augmente le risque des pannes.

2.9 ROS 2.0

Pour résoudre ces points une deuxième génération du ROS a été publiée, c'est ROS 2.0. Les développeurs du ROS ont annoncé ROS2 pour s'adapter avec les nouveautés dans la robotique depuis 2007 et voila les Nouveaux cas d'utilisation :

— **Équipes de robots multiples** : Bien qu'il soit possible de construire des systèmes multi-robots en utilisant ROS aujourd'hui, il n'existe pas d'approche standard et ils sont tous un peu comme un hack sur la structure à maître unique de ROS.
— **Petites plateformes embarquées** : Le but est que les petits ordinateurs, y compris les microcontrôleurs "bare-metal", soient des participants de premier ordre dans l'environnement ROS, au lieu d'être séparés des ROS par un pilote de périphérique.
— **Systèmes en temps réel** : Prendre en charge le contrôle en temps réel directement dans les ROS, y compris la communication inter-processus et inter-machines (en supposant un système d'exploitation et/ou un support matériel appropriés).
— **Réseaux non idéaux** : ROS doit se comporter aussi bien que possible lorsque la connectivité du réseau se dégrade en raison d'une perte et/ou d'un retard, du WiFi de mauvaise qualité aux liens de communication sol-espace.
— **Environnements de production** : S'il est essentiel que les ROS continuent d'être la plateforme de choix dans les laboratoires de recherche, l'équipe ROS veut garantir que les prototypes de laboratoire basés sur les ROS puissent évoluer vers des produits basés sur les ROS adaptés aux applications du monde réel.
— **Modèles prescrits pour la construction et la structuration des systèmes** : tout en maintenant la flexibilité sous-jacente qui est la marque de fabrique des ROS, l'équipe va fournir des modèles clairs et des outils de soutien pour des caractéristiques telles que la gestion du cycle de vie et les configurations statiques pour les déploiements.

2.10 Conclusion

ROS est une plateforme de développment logicielle trés répandus dans les applications de la robotique, il est utilisé dans de nombreux robots : drones, voitures autonomes, robots humanoïdes, bras robotisés, et bien d'autres. ROS est un ensemble d'outils dont l'architecture de base permet de réaliser des applications avec une plus grande abstraction. Grace à des outils intégrés à ROS, il est possible de faire de la planification de mouvement, reconnaissance d'objets, navigation 2D, cartographie 3D. Le chapitre 3 présente des exemples montrant une façon de contrôler le robot, avec l'aide de ROS.

Chapitre 3

Applications sur Ros

3.1 Simulation du robot Summit_XL

Robotnik est un producteur référence en robotique mobile, Fondée en 2002 comme une entreprise de conception et production des robots mobiles, actuellement Robotnik a plus de 4700 robots autour de 50 payés, dont la plupart ont ROS intégré. Parmi ces robots, on a travaillé sur le Summit_xl dans les figures 3.1.

FIGURE 3.1 – Le robot SUMMIT_XL de Robotnik

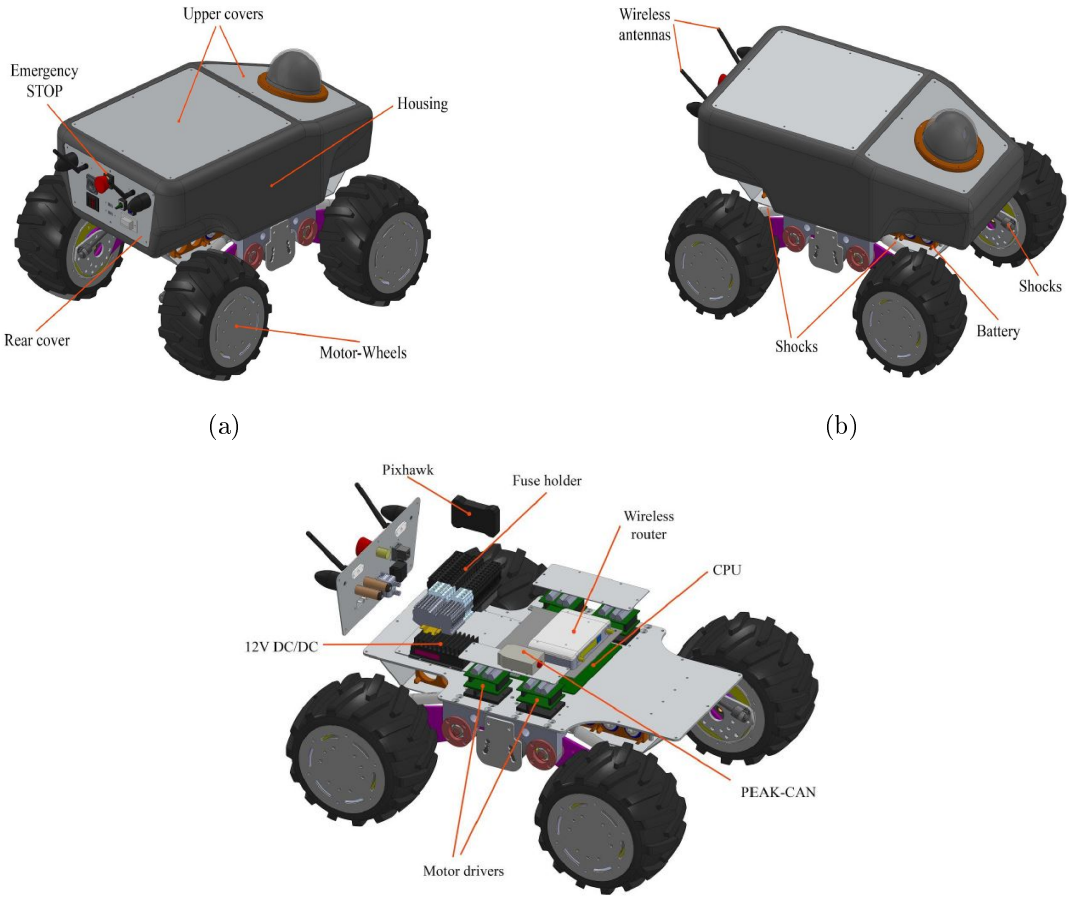
3.1.1 Caractéristiques matérielles

Paries principales

Les parties principales qui forment le robot sont :

— **Boîtier** Il est fait de fibre de verre et contient les couvercles supérieur et arrière. Les composants électriques sont placés à l'intérieur de l'armoire. Seule la batterie est à l'extérieur.
— **Couvercles supérieurs** Peuvent être retirés pour accéder à l'intérieur du robot où sont placés certains des composants de contrôle comme l'ordinateur de contrôle.
— **Couverture arrière** Contient le panneau de contrôle, les boutons et les antennes sans fil.
— **Roues du moteur** : Quatre roues motrices Moteur sans balais de 250W avec capteur à effet Hall et une boîte de réduction à l'intérieur de la roue en aluminium dans la figure 3.3a.
— **Batterie** : batterie LiFePO4 24V (8x3.2V 15Ah).
— **Amortisseurs** : Quatre puissants amortisseurs.
— (en option) Caméras PTZ protégées par des dômes et des capteurs remplaçables.
— **Arrêt d'urgence**, pare-chocs avant et arrière désactivent les conducteurs qui arrêtent le robot si activé.

Les différentes parties du robot se communique selon le diagramme dans la figure 3.4

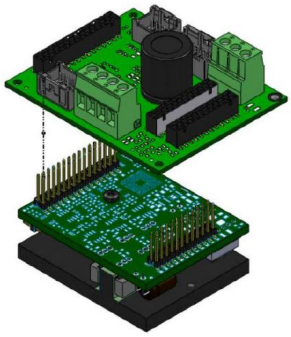


(a) (b)

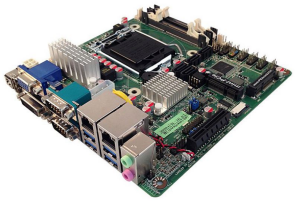
(c) Vue interne

FIGURE 3.2 – Les différentes vues du robot

(a) La roue motrice



(b) Le contrôleur du moteur



(c) Le PC embarqué

3.1.2 Caractéristiques logicielles

L'architecture ROS du summit_xl est le résultat du fonctionnement coopératif de plusieurs nœuds. le PC embarqué du robot est muni de ROS Kinetic Desktop Full. Pour qu'on puisse commander un robot il nous faut les deux meta-paquets summit_xl_robot et summit_xl_common si on veut procéder à la simulation un autres meta-paquet serais important le summit_xl_sim, le contenu de ces trois est comme

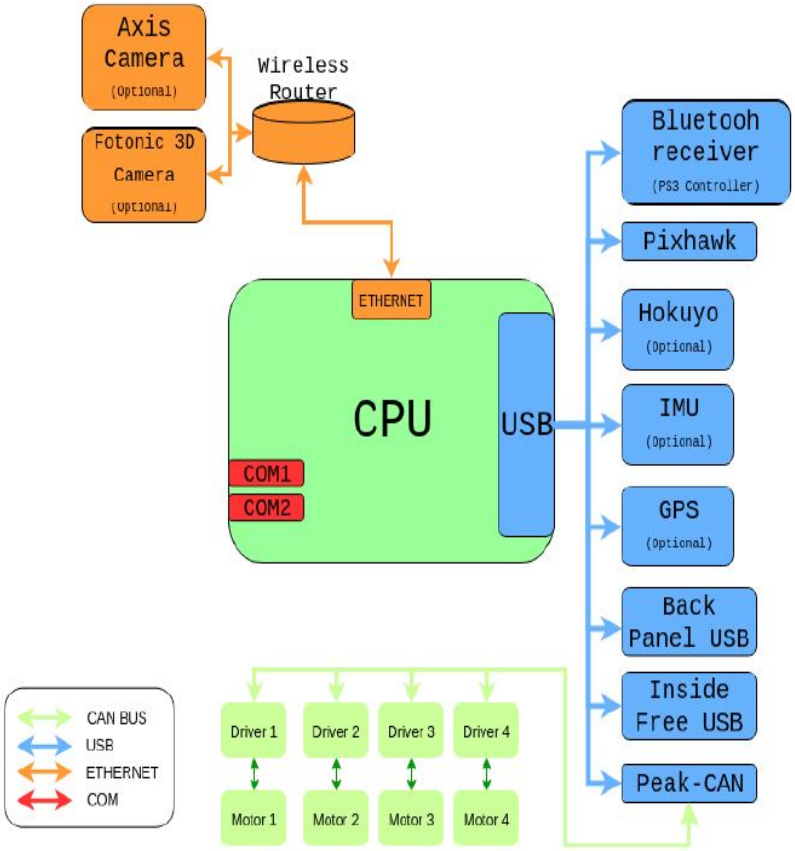


FIGURE 3.4 – Diagramme de communication

suivant :

- summit_xl_robot
  - summit_xl_bringup : contient les fichiers de démarrage
  - summit_xl_controller : contient le fameux ROS controller **??**
  - summit_xl_web : pour pouvoir commander le robot via le web.
- summit_xl_common
  - summit_xl_description : là où se trouve les modèle disponible du robot.
  - summit_xl_control
  - summit_xl_localization : contient plusieurs algorithmes de localization.
  - summit_xl_navigation : tous les configurations possible pour la navigation du robot.
  - summit_xl_pad : pour commander le robot via la manette du playstation
  - summit_xl_aubo_moveit_config
  - summit_xl_ur5_moveit_config
- summit_xl_sim
  - summit_xl_sim_brigup : les fichiers pour démarrer les simulations
  - summit_xl_gazebo pour simuller le robot sous gazebo.

3.1.3 Visualisation et simulation

Après l'installation du ROS correcte sur la machine et le téléchargement et la compilation des "pakages" nécessaires. On peut accéder à plusieurs fonctions dans le robot, il est encore possible de réaliser toutes ces fonctions sur un robot Summit XL réel, mais ce n'est pas le cas de notre travaille.

Etapes pour démarrer la simulation :
- Exécuter le maitre par l'instruction "roscore"

- Exécuter l'instruction de la figure 3.5

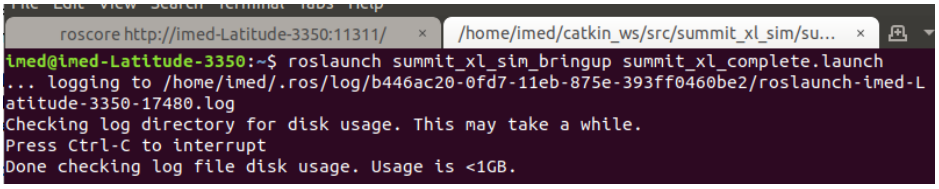


FIGURE 3.5 – commande pour la simulation complète

en fait le fonctionnement du robot est en mode différentiel par défaut ; cela est assuré en assemblant les roues de chaque coté comme dans la figure 3.6.

```
type        : "diff_drive_controller/DiffDriveController"
left_wheel  : ['summit_xl_a_front_left_wheel_joint','summit_xl_a_back_left_wheel_joint']
right_wheel : ['summit_xl_a_front_right_wheel_joint','summit_xl_a_back_right_wheel_joint']
```

FIGURE 3.6 – Le mode differentiel

Résultats

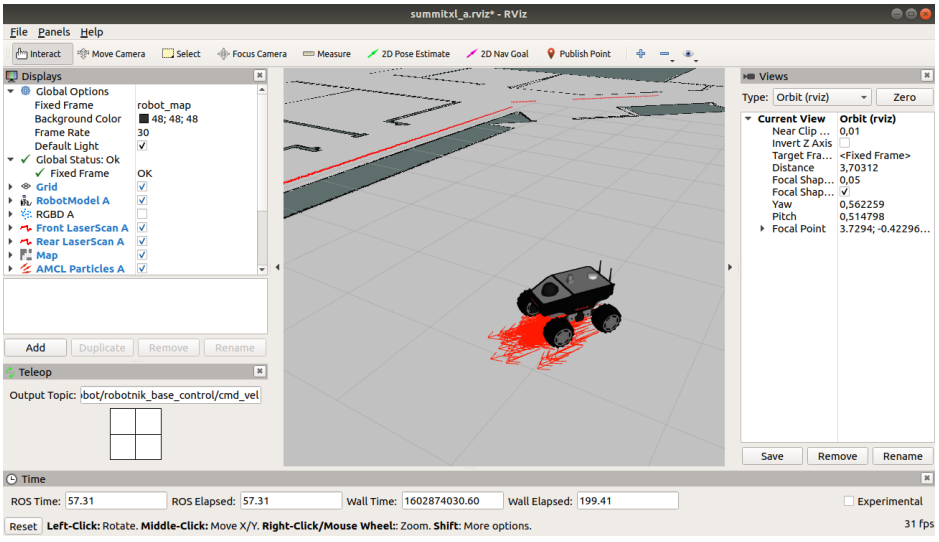


FIGURE 3.7 – La visualization sur Rviz

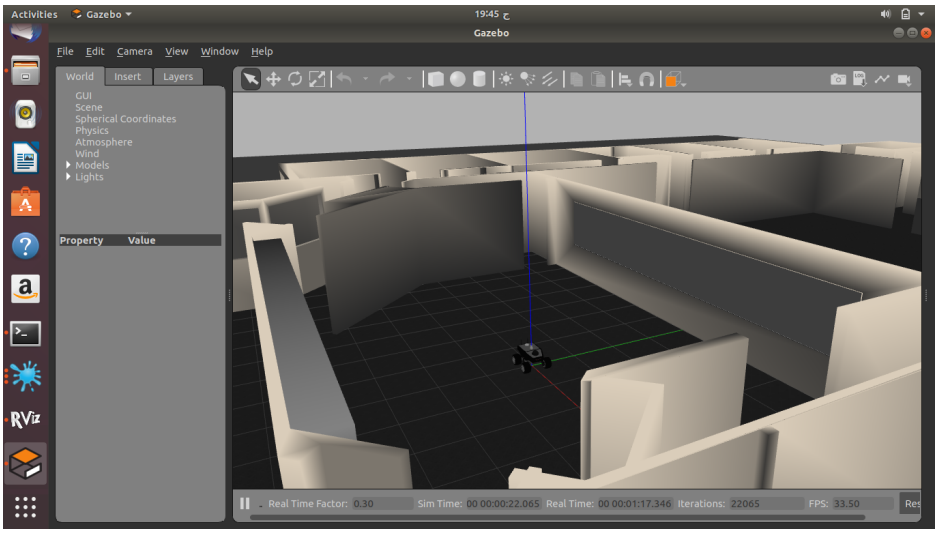


FIGURE 3.8 – La simulation sur gazebo

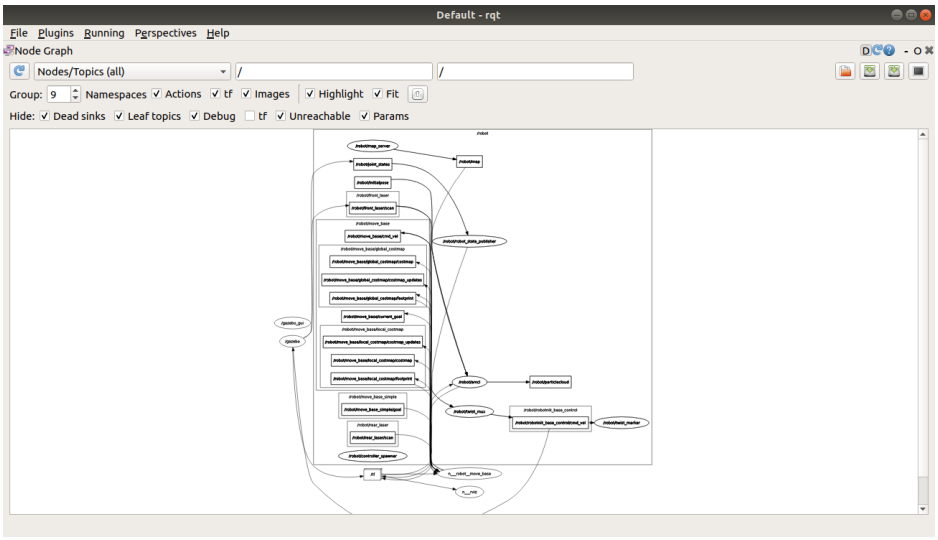


FIGURE 3.9 – Le graphe des nœuds actifs lors de la simulation

Conclusion générale

Conclusion générale

Le but de notre projet de fin d'études était d'appliquer la commandetolérante aux fautes sur le Summit-XL tout en considérant la saturation des actionneurs. Ce travail entre dans le cadre de collaboration entre le laboratoire LAIG (Laboratoire d'Automatique et informatique de Guelma) et le CRIStAL (Centre de Recherche en Informatique Signalet Automatique de Lille).

Les choses ne vont pas comme on veut, à cause de la pandémie COVID19, mon stage au sein de CRIStAL a été interrompu et avec le temps, mon rêve de réaliser ce projet s'est évaporé.

Et malgré,toutes les difficultés qu'a vécues le monde entier qui ont fortement affecté mon travail, ce modeste mémoireest le fruit d'un défi et de la patience en cette période.

En fait, plusieurs objectifs ont été accomplies lors de mes recherches ; la maitrise du système d'exploitationLinux et le ROS, que je ne connaissais du tout avant. Il est important aussi de noter la puissance du ROSpour développer des systèmes en robotique, plus précisément en robotique mobile ; la disponibilité desbibliothèques, la communauté vibrante, le cumul des compétences... tous ces critères permettent au ROSde jouer un rôle très important dans la recherche et l'innovation. La maitrise du ROS et les différentesnotions de la robotique mobilesont des pistes vierges à exploiter dans le cadre des projets recherches dans le domaine.

Bibliographie

[1] COIFFET Philippe, Robots : définitions et classification. technique de l'ingénieur. Code du document R 7 700.

[2] Le robot mobile "Sejourner" utilisé pour la mission pathfinder de la NASA, article sur site web www.robocup2014.org, 2014.

[3] BENMACHICHE Abdelmadjid, Approche de Navigation Coopérative et Autonome des Robots Mobiles.Thèse de doctorat : Université BADJI MOKHTAR ANNABA. 115 pages.

[4] Debra F Laefer, Siyuan Chen, Eleni Mangina. State of Technology Review of Civilian UAVs. 2016. Article sur researchgate

[5] Lionel Geneve, Système de déploiement d'un robot mobile autonome basé sur des balises. Thèse de doctorat soutenue le 25 Septembre 2017 université de STRASBOURG. 198 pages.

[6] NADER Joudy,Commande et fusion des données tolérantes aux fautes pour la navigation autonome d'un système multi-robot. DIPLÔME D'INGÉNIEUR DE L'UNIVERSITÉ LIBANAISE, Soutenue le 25 Juillet 2019. 60 pages.

[7] Giuditta De Prato, Annike Thierry, Une technologie de base : l'intergiciel (middleware)

[8] YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim. ROS Robot Programming.

[9] Hugo ANDRIANASOLO Yoann AFFLARD Mise en place d'une architecture décentralisée pour la commande en quatre roues motrices indépendantes. Mémoire de Master à l'université de Lille 1. 2019-2020.45 pages.

[10] Jérôme Laplace, Présentation de Robot Operating System

[11] Keenan Wyrobek,The Origin Story of ROS, the Linux of Robotics,journal IEEE.

[12] Evan Ackerman, Erico Guizzo, Wizards of ROS : Willow Garage and the Making of the Robot Operating System, journal IEEE.

[13] Fiches techniques fournies par Robotnik Summit_xl Manuals