Ecole des mines d'Alès

Formation ingénieur généraliste

Mission R&D:

Conception d'une Plateforme mobile, prise en main de ROS

Rapport sur la documentation d'une Plateforme robotique mobile autonome sous ROS

Magnus Braun

Axel Billy

Travaux supervisé par:

Alexandre Meimouni

Sommaire

Sommaire **1**

I - Introduction **2**

Problématique 2

Démarche générale 2

II - Travail effectué **3**

Analyse bibliographique 3

Méthodologie 3

Résultats obtenus 3

Analyse de ces résultats 5

Conclusion **7**

Résumé **8**

Bibliographie **9**

I - Introduction

Problématique

Dans l'idée d'actualiser les plateformes robotique actuelles de l'école, le professeur A. Meimouni souhaite utiliser ROS (Robot Operating System), un système composé de plusieurs logiciels qui permettra de commander le robot.

Nous avons concentré notre travail sur la prise en main de ROS et sur l'implémentation sur les plateformes actuelles qui sont composées : d'un lidar, de cartes électroniques (Raspberry Pi 4, Arduino Mega), une motorisation et des capteurs. Puis, nous avons restitué ce travail sur un document que nous avons établi.

Démarche générale

Au départ, nous avons fait des recherches sur la manière d'implémenter ROS dans notre robot et sur la compréhension de son utilisation. C'est-à-dire, comment les différents logiciels de ROS s'interface entre eux, comment personnaliser notre modèle de robot et sur comment simuler la navigation autonome.

Après avoir effectué toutes ces recherches, nous avons suivi des tutoriels pour comprendre en détail la mise en oeuvre des différents aspects de ROS (simulation de l'environnement du robot, commandabilité du robot…).

Ces deux étapes nous ont permis de réunir dans une base de connaissances [1] toutes les étapes et démarches nécessaires pour pouvoir piloter la plateforme robotique de l'école via ROS.

II - Travail effectué

Nous vous invitons à lire le Wiki sur le GitHub de la plateforme robotique mobile qui vient compléter ce rapport comme convenu avec M. Meimouni :

https://github.com/Rattiputz/plateforme-robotique-mobile/wiki

Celui-ci est structuré en tutoriels qui ont 2 objectifs principaux :

- Apprendre à utiliser ROS sans connaissances préalables du système ;
- Transmettre tous les détails qui étaient nécessaires à la réalisation du projet pour faciliter la suite du développement.

Analyse bibliographique

Comme énoncé précédemment, nous avons utilisé de nombreuses sources (voir tableau), ainsi que le travail de Koji qui a établi les préliminaires de notre travail. Nous avons réuni toutes les informations utiles au bon fonctionnement de la plateforme sur Github [1].

Méthodologie

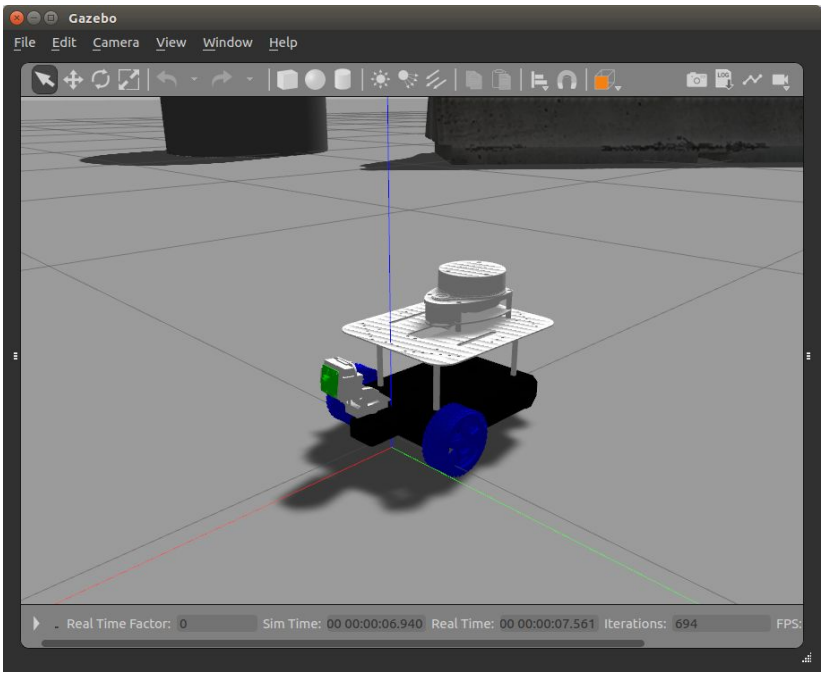
Après avoir compris comment installer ROS et établir le modèle de notre robot, nous avons suivi différentes étapes d'un blog [6] dans lequel nous avons compris comment :

- Simuler le modèle de notre robot dans un environnement virtuel sur notre ordinateur ;
- Simuler virtuellement l'utilisation de différents capteurs (lidar, infrarouges…) ;
- Etablir la connection entre notre ordinateur et la carte électronique (Raspberry Pi 4) qui commande les différents éléments (Actionneurs et capteurs) ;
- Effectuer la navigation autonome de notre robot via ROS avec deux procédés de cartographie de l'environnement (Hector Slam et Gmapping).

Cela nous a permis de généraliser ces méthodes et d'établir des explications à la fois faciles à comprendre et détaillées, afin de servir à la fois de support pédagogique pour les élèves et d'outil pour le développement des robots à l'école.

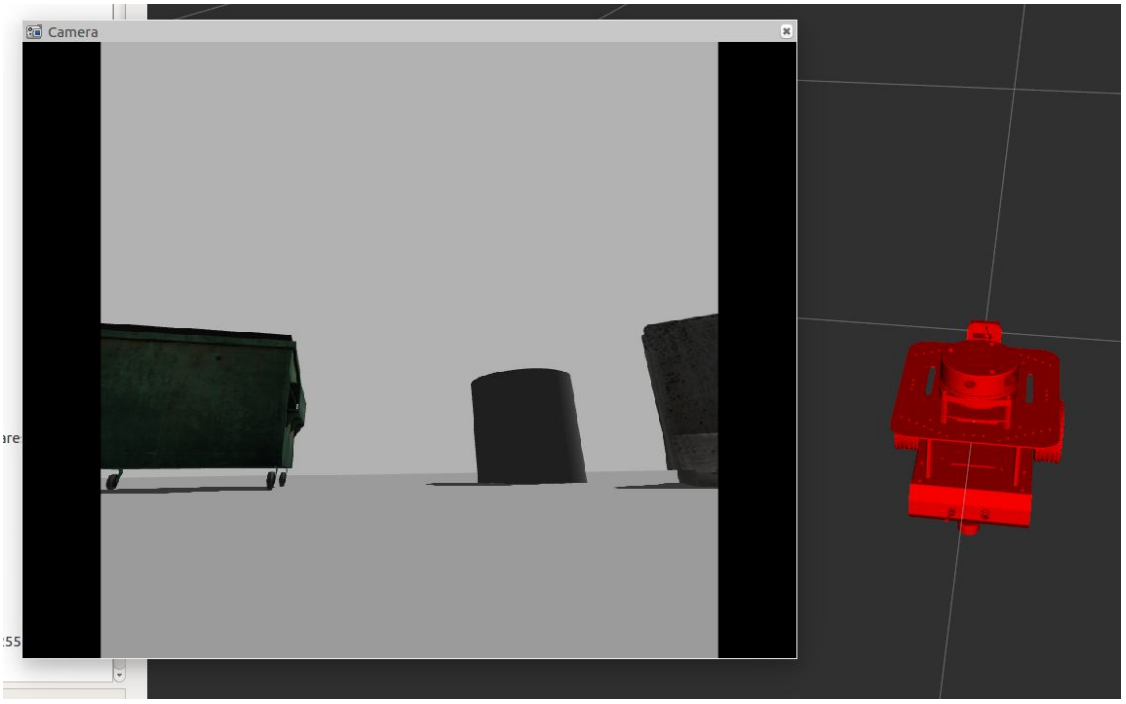
Résultats obtenus

En premier lieu, nous avons pu simuler notre robot virtuellement dans Gazebo qui est un logiciel de simulation. On peut voir ci-dessous une image de la simulation en question.



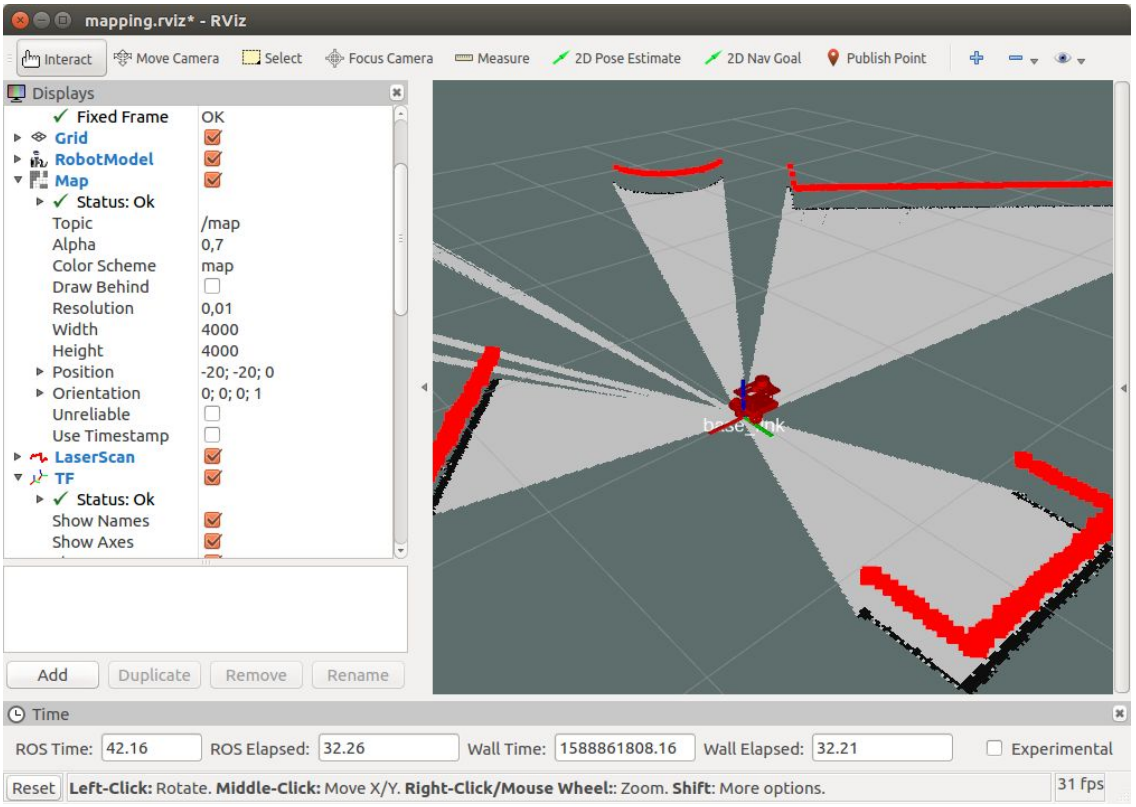
Suite à ces premières simulations, nous avons remarqué certaines imperfections (inertie du robot mal calibrée, ordre de définition des pièces dans le modèle urdf) qui nous ont poussé à modifier le modèle du robot que Koji avait établi.

Nous avons pu tester l'utilisation de la caméra et du lidar dans un environnement composé de plusieurs obstacles. Ainsi, nous avons pu commander notre robot dans cet environnement et voir comment la navigation est établi. Ci-dessous on peut voir l'image renvoyée par la caméra sur RViz, logiciel qui nous permet de gérer les informations des différents capteurs :



Malgré le fait que notre robot ne soit pas monté (certaines pièces n'étaient pas encore arrivées à temps), nous avons pu connecter notre carte électronique, la Raspberry Pi 4, à l'ordinateur via une connection SSH. Ce protocole nous a permis de transmettre les informations de manière bidirectionnel, c'est à dire que nous avons pu envoyer des ordres à la carte électronique et recevoir les données des capteurs liés à la carte. Disposant du rplidar A1 de Slamtec, nous avons pu le connecter à la carte électronique, la cartographie de l'environnement réel était donc possible.

Ci-dessous la visualisation des données capteurs (simulation ici) sur RViz :



L'affichage des données est possible grâce à RViz qui publie les données du lidar. Tous les points rouges sont les obstacles que le lidar a détecté autour de lui et RViz cartographie : les points gris qui forment l'espace libre, les points noirs les obstacles.

De toutes ces opérations, il en résulte le document [1] qui récapitule toutes les étapes pour effectuer ce que nous avons pu tester (navigation autonome, utilisation de la caméra, du lidar, simulation…)

Analyse de ces résultats

Notre étude nous a mené à la navigation autonome du robot grâce notamment au Lidar qui détecte l'environnement. Pour effectuer la navigation, l'algorithme utilisé est le

SLAM (Simultaneous Localization and Mapping) qui permet de cartographier l'environnement pour détecter les obstacles et de se diriger sur cette carte en fonction des ordres donnés par l'opérateur. Pour cartographier l'environnement, nous avons testé deux procédés :

- Hector Slam, qui utilise l'environnement pour placer la carte. Ce qui permet d'avoir une meilleure capacité pour extrapoler les informations de la carte existante.
- Gmapping, qui utilise l'odométrie pour se repérer sur la carte existante et l'agrandir. Cela induit une plus forte probabilité d'erreur dans la détection mais permet une plus grande précision.

Les changements dans la modélisation du robot permettront également de modifier plus rapidement le robot, et de manière systématique. Une description complète de ce procédé est disponible sur Github (Tutoriel 3).

Conclusion

Pour conclure, nous avons grâce à cette mission identifié la structure et la manière d'implémenter ROS, simulé l'utilisation d'un lidar et d'une caméra via RViz et simulé la navigation autonome du robot.

Pour la suite, nous pensons qu'il est nécessaire de:

- Finaliser la construction du robot physique ;
- Actualiser le modèle du robot sur l'ordinateur pour représenter le robot réel correctement ;
- Permettre la collaboration de plusieurs robots ;
- Mettre en oeuvre le micro contrôleur indépendamment de ROS pour que le robot puisse fonctionner sans ordinateur supplémentaire.

Résumé

Nous avons mené une étude sur l'utilisation de ROS (Robot Operating System), un système complémenté par plusieurs logiciels de simulation pour commander un robot. Durant cette étude, nous avons compris la structure et l'implémentation de ROS. Par la suite, il nous a été possible d'effectuer des simulations des capteurs et de la navigation autonome (se basant sur l'algorithme SLAM) du robot. Des essais réels n'ont pas été possible à cause du confinement mais nous avons pu cartographier un environnement réel et implémenter la transmission d'informations entre l'ordinateur et la carte électronique qui doit commander les capteurs et les actionneurs. Toutes les étapes de notre étude ont été résumées sur un Wiki [1].

We have conducted a study on the use of ROS, a system used in conjunction with software to control a robot. During this study, we have understood how to implement ROS and how it's built. Furthermore, we were able to simulate the use of sensors and the autonomous navigation (based on the SLAM algorithm) of the robot. We could not carry out tests on the real robot because of the containment but we were able to map a real environment and implement the SSH communication between the computer and the electronic card which is used to control the actuators and sensors. All the steps of our study have been summarized on a Wiki [1].

Bibliographie

Numéro	Titre	Liens/ Référence
1	Github prm	https://github.com/Rattiputz/plateforme-robotique-mobile/wiki
2	Robot Operating System	Livre de Olivier Stasse de 2016 publié par Publisher
3	Direct Raspberry Pi to computer Ethernet control,	https://www.youtube.com/watch?v=8qleH35Kgjk
4	Guide d'installation de ROS Kinetic sur Ubuntu	http://wiki.ros.org/kinetic/Installation/Ubuntu/
5	Installation de Gazebo 8/9 sur ROS Kinetic	https://medium.com/@abhiksingla10/setting-up-ros-kinetic-and-gazebo-8-or-9-70f2231af21a
6	Prise en main de navigation robotique	http://moorerobots.com/blog
7	Ros Tutorials	https://husarion.com/tutorials/
8	Introduction to Robotics	https://u.cs.biu.ac.il/~yehoshr1/89-685/
9	ROS- Robot Operating System	https://www.generationrobots.com/blog/fr/ros-robot-operating-system-3/
10	Is there a Ubuntu Mate for Raspberry Pi 4?	https://ubuntu-mate.community/t/is-there-a-ubuntu-mate-for-raspberry-pi-4/19943/4/