École des Mines d'Alès
Formation d'ingénieur généraliste

Plateforme Robotique

Documentation de la Plateforme robotique mobile autonome

Documentation réalisée dans le cadre
d'un monitorat recherche (travaux d'étudiant de 1A)

Andriamahery Koji

Travaux supervisés par
Couturier Pierre
Meimouni Alexandre

Résumé

Ce dossier consiste en une documentation technique de la plateforme robotique mobile utilisée dans le cadre des cours d'IoT, d'architecture de microcontrolleurs, de robotique et de systèmes embarqués. Il détaille les résultats des travaux effectués dans le cadre de la période de stage ouvrier consacrée à la poursuite du monitorat recherche. Il fait directement suite au Document de Synthèse sur la Plateforme Robotique mobile : un prototype a été créé suite aux propositions d'architecture décrites dans le document de synthèse, et cette documentation permet de décrire le fonctionnement de l'architecture retenue et des outils mis en place autour de celle-ci.

Pour rappel, voici les principaux points que devait respecter le prototype robotique :

- le robot doit consister en un outil pédagogique permettant la sensibilisation aux élèves des Mines d'Alès autour des thématiques de la robotique mobile, tels que : les systèmes embarqués, les objets connectés et l'IOT, la modélisation numérique et le contrôle commande, … ;
- le robot doit évoluer sur un terrain plat comportant peu ou pas d'obstacles ;
- le robot doit présenter une architecture modulaire ;
- l'autonomie du robot doit être supérieure à 2 heures de fonctionnement sans charge.

Cette documentation, susceptible d'évoluer avec le temps, décrit en une approche mécatronique le système : les aspects mécaniques, électroniques et informatiques sont tous traités. La partie électronique s'appuie notamment sur divers tutoriels et manuels de prise en main du projet open source ROS (pour Robot Operating System), plate-forme de développement logiciel sur lequel est basé la plateforme robotique.

Table des matières

Introduction

La robotique est un domaine particulièrement en essor ces dernières années, notamment dans le cadre du développement de l'industrie 4.0. Ainsi, la *robotique mobile,* branche spécialisée dans la navigation et la localisation d'un système autonome dans un environnement plus ou moins inconnu et/ou complexe, s'est naturellement développée au sein de plusieurs centres de recherche afin de répondre aux problématiques nouvelles : mobilité urbaine (véhicules autonomes, drones, …), exploration de milieux hostiles à l'homme, télésurveillance et sécurité connectée, …

Bien que les problématiques d'ordre mécanique sont également soulevées par la robotique mobile, le plus gros à traiter dans cette thématique reste de l'ordre du traitement en temps réel des informations sur l'environnement et de la planification des actions en conséquence. Ainsi, le choix d'une architecture, plus particulièrement le choix d'un ensemble de capteurs et de CPU's, répond à un problème - plus ou moins large mais surtout plus ou moins complexe - principalement défini par un type d'environnement (connu/inconnu, complexe/simple, étendu/restreint, stable/instable, intérieur/extérieur…). Une plateforme robotique mobile a donc été créée à l'École des Mines d'Alès afin de sensibiliser les étudiants à ces thématiques. Se voulant comme support de TD et TP pour l'approche des problématiques modernes, elle a été mise à jour dans le cadre d'un monitorat recherche en juillet 2019 afin d'intégrer les outils et capteurs modernes (dont notamment le LiDAR et la caméra comme capteurs de navigation et de mapping, couplés à la couche logicielle ROS).

Ce document se veut donc comme une documentation technique visant à appréhender les techniques robotiques et IoT considérées comme modernes à l'heure où sont écrites ces lignes. Parce qu'il est toujours possible d'aller plus loin en terme de développement logiciel, il est proposé aux plus curieux de consulter la bibliographie ainsi que les documents et cités dans les notes en bas de page afin de mieux s'approprier les outils utilisés et de rendre la plateforme la plus autonome possible.

1. L'architecture

Comme évoqué précédemment, ce robot à base mobile consiste principalement en une mise à jour de la plateforme. Ainsi, ce document se base principalement sur les documents techniques écrit sur la version antérieure de la plateforme. L'objet de cette première partie est de décrire de façon exhaustive l'architecture du système en insistant sur les changements liés à cette « nouvelle » version.

1.1. Les constituants

1.1.1.Les constituants mécaniques

Afin de permettre le plus important développement électronique possible, qui rappelons-le constitue la capacité du « cerveau » du robot, peu de changement ont été effectués du côté mécanique. Ainsi, le châssis est le même que celui utilisé précédemment[1]. Les passages de câbles ont été pensés différemment afin d'accueillir mécaniquement les nouveaux composants électroniques et d'optimiser au mieux leur fonctionnement (pas de câbles ou de partie parasite d'objets obstruant le champ de vision de la camera ou le faisceau laser du LiDAR, pas de partie chauffant inutilement les cartes électroniques, etc.). La seule véritable modification mécanique a été le remplacement du modèle cinématique : en effet, la plateforme possède originellement une base mobile à 4 roues motrices. Le modele cinématique du robot a été ajusté afin de ne présenter que deux roues motrices à l'avant, et une roue folle à l'arrière. Egalement, nous pouvons noter le retrait de la pince de contrôle et l'ajout d'une tourelle « pan-tilt » permettant les rotations autour de z et de x de la caméra du robot[2]. Nous pouvons donc dresser la liste des constituants mécaniques de la manière suivante :

- Châssis en aluminium (150 mm X 200mm).
- 2 roues en caoutchouc (Diam.: 65mm, Epaisseur:26 mm).
- 2 motoréducteurs CC (5 à 12V).
- Roue folle.

Le modèle 3D (format .step) de la plateforme est disponible sur le Github ici.

[1] Chassis à 4 roues motrices de DFRobot, https://www.dfrobot.com/product-97.html

[2] Tourelle « Pan-Tilt » pour caméra Raspberry Zero, https://www.robotshop.com/eu/fr/kit-tourelle-pan-tilt.html

1.1.2.Les constituants électroniques

C'est par l'intégration de meilleurs composants électroniques que la plateforme a essentiellement été améliorée. On note principalement l'ajout d'un capteur laser type LiDAR[3], le passage de la caméra v1 à la caméra v2 de la Raspberry Pi et l'utilisation du modèle à 2GB de la Raspberry Pi 4 B+. Cette dernière fonctionne sur Raspbian Buster afin de permettre l'utilisation la plus complète et la plus simple de la Raspberry - on notera néanmoins que ROS n'est officiellement supporté que par Ubuntu, mais que par soucis de simplicité il est préférable d'utiliser l'OS natif de la framboise, sur lequel de nombreux robots de la communauté ROS sont développés. Par rapport à la version précédente, on pourra noter que les capteurs ultrasons ont été délaissés car il est inutile de les coupler à un capteur LiDAR. Néanmoins, de nouveaux capteurs infrarouges analogiques, plus performants, ont été placés afin de s'assurer de la détection d'obstacle de courte portée (0-30 cm). En effet, la technologie ultrason ne permet pas la détection sur de la courte distance (cf Document de Synthèse sur la Robotique Mobile). Une carte IMU intégrant gyroscope, accéléromètre, magnétomètre (3 axes pour chacun) et baromètre a également été intégrée pour permettre l'odométie du robot. L'alimentation a été choisie pour améliorer l'autonomie du robot : tout en restant de 11.1V pour permettre de rester sur les mêmes adaptateurs de charge, la nouvelle batterie embarque une autonomie de 4000mAh. Les composants non cités restent inchangés. La liste des constituants électroniques peut donc être dressée comme ce qui suit :

- Alimentation (batterie LIPO 11.1V ou Alim. de Laboratoire).
- 4 capteurs infrarouges analogiques (d < 30 cm) .
- 1 carte IMU (accéléromètre 3 axes, gyroscope 3 axes, compas 3 axes, baromètre).
- 1 module LiDAR (modèle A1 de chez Slamtec).
- 1 module GPS.
- 1 caméra v2 pour Raspberry Pi.
- 1 carte d'alimentation (5V).
- 1 carte de puissance pour les 2 motoréducteurs CC.
- 1 carte de commande pour servomoteurs (jusqu'à 8 servos).
- 1 carte Raspberry Pi 4 B+ (embarquant Linux - distribution Raspbian, version de Debian spécifiquement développée pour carte Raspberry Pi).
- 1 carte CPU PIC18F46K22.
- 1 carte RF 433MHz.

[3] RPLiDAR A1 de Slamtec, https://www.slamtec.com/en/Lidar/A1

1.2. Le microcontrolleur

Tout comme sa version précédente, la plate-forme robotique mobile intègre un microcontrolleur d'architecture PIC. Comme certains capteurs ont été enlevés et d'autres retirés, il a fallu modifier l'architecture de la carte PCB du microcontrolleur. La nouvelle conception a été réalisée à l'aide du logiciel Eagle.

1.3. Le schéma fonctionnel

Le schéma fonctionnel de la plateforme (voir ci-dessous) correspond plus ou moins à ce qui a été fait précédemment.

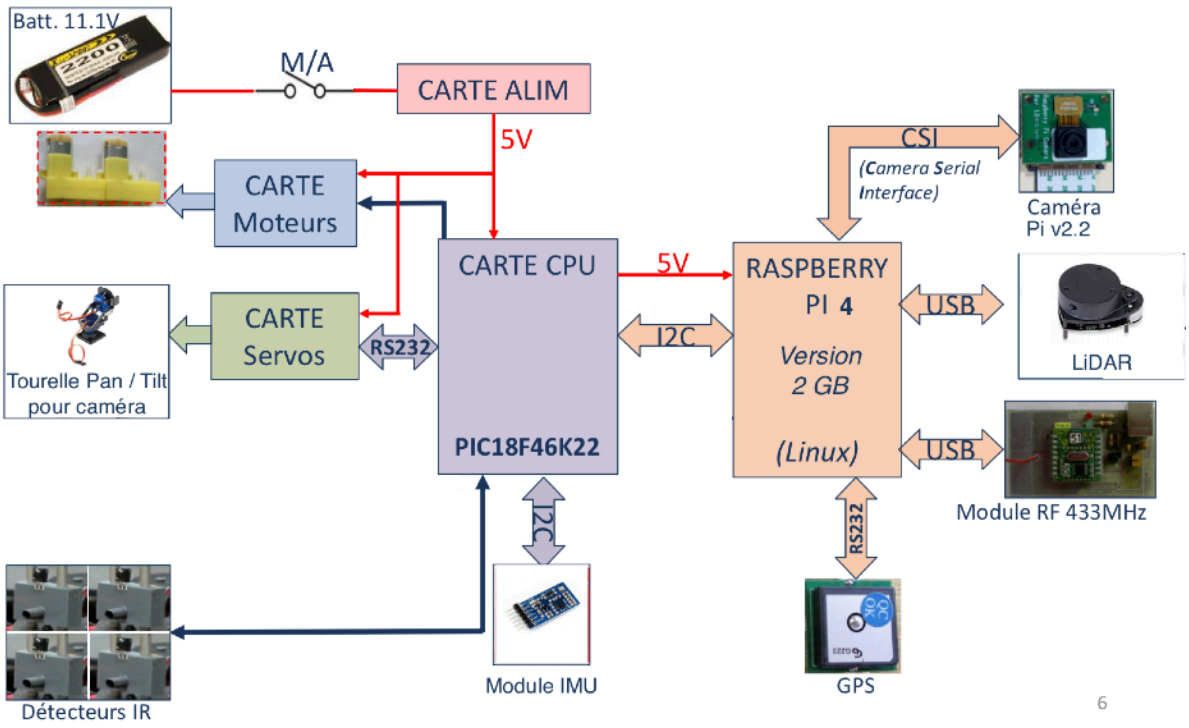

FIGURE 1. SCHEMA FONCTIONNEL DE LA PLATEFORME ROBOTIQUE MOBILE

2. Le fonctionnement

Le but du robot est de se repérer et de naviguer de manière autonome dans un environnement inconnu, dont la carte se dessinera à l'aide du déplacement du robot. Il existe plusieurs algorithmes de navigation autonome, dont principalement le SLAM (Simultaneous Localization And Mapping - cf Document de Synthèse sur la Robotique Mobile). Il devra être possible également de télé-opérer le robot afin de le faire déplacer dans son environnement. Ainsi, après avoir comparé plusieurs architectures système des couches logicielles existantes traitant de la robotique, le choix le plus prometteur s'est porté sur ROS - pour *Robot Operating System* - pour *sa communauté grandissante*, *ses possibilités technologiques*, *son aspect d'intéroperabilité* pratique déjà déployé sur plusieurs robots ainsi que *son essence open source* qui motive son utilisation courante dans les plus grands centres de recherche robotique à travers le monde.

2.1. ROS - *Robot Operating System*

2.1.1.Le principe de ROS

ROS est l'acronyme de *Robot Operating System*. C'est le seul projet open source à échelle mondiale qui tente d'unifier le fonctionnement d'un robot : en proposant une couche logicielle et un ensemble de librairies et bibliothèques numériques accessible à tous, ROS met à disposition de tous les plus récentes avancées en terme de robotique. Toutes les thématiques de la robotique y sont abordés : la robotique industrielle, la cobotique, la navigation autonome terrestre, sous-marine et volante, … l'un des points forts de ROS est justement l'interoperabilité entre système : il est possible de faire tourner une simulation numérique se rapprochant énormément de la réalité sur un poste d'ordinateur fixe, tout en exécutant en temps réel le système physique sur la carte embarqué du robot qui serait lié au poste. Il est dans ce cas extrêmement interessant de remarquer les différences entre les jumeaux numériques, et de les ajuster afin d'améliorer la simulation sur poste fixe, tout en traitant les données réelles reçues. De plus, il est de cette manière très simple et évidente de faire dialoguer plusieurs systèmes embarqués entre eux. Assez peu facile à prendre en main, ROS permet néanmoins de réaliser pour très peu un système intelligent performant. Vous pouvez retrouver ici[4] une histoire et un « overview » plus détaillé sur les fonctionnalités et possibilités que propose ROS.

[4] ROS - Robot Operating System, Generation Robots, https://www.generationrobots.-com/blog/fr/ros-robot-operating-system-3/

2.1.2.L'installation sur poste fixe

Afin de permettre une prise en main plus rapide et plus efficace, un répertoire Github a été créé. Il recense tous les répertoires et éléments nécessaires à la simulation du robot sur ordinateur. Avant d'installer ceux-ci, il faut installer une distribution de ROS sur votre ordinateur fixe. Bien que le site wiki de ROS soit très utile, il n'est que partiellement traduit en français. Il est recommandé que l'installation s'effectue sur Ubuntu 16.4 puisque c'est la version officiellement compatible avec ROS. D'ailleurs, il est recommandé d'installer l'avant dernière distribution de ROS (soit Kinetic à l'heure où ces lignes sont écrites), puisqu'elle est à la fois la plus stable et la plus complète en terme de fonctionnalités. Cependant, bien qu'il y ait un planning des sorties de distributions, ROS est environnement soutenu activement et régulièrement par des contributeurs dynamiques, ainsi il est toujours préférable de suivre les recommandations de ROS et de sa communauté.

Note : Il existe plusieurs liens de prise en main de ROS en français[5], mais sachez que les avancées les plus récentes sont d'abord communiquées (et bien souvent également d'abord exploitées) en anglais.

2.1.3.L'installation sur Raspberry Pi

La Raspberry Pi, bien que performante (la version 4 particulièrement puisqu'elle est capable d'embarquer jusqu'à 4GB de RAM), n'est pas optimisée pour une utilisation « desktop » - c'est-à-dire graphique - de ROS. De plus, comme évoqué précédemment, ROS n'est officiellement supporté que sur Ubuntu. Or bien qu'il soit possible d'installer Ubuntu sur Raspberry Pi, il est préférable pour une question de fluidité et d'optimisation de fonctionnement que la Raspberry fonctionne sur Raspbian OS. Ainsi, nous recommandons l'installation de Raspbian Buster (ou version ultérieure) en configuration IoT (c'est-à-dire sans outils graphiques), et l'installation des dépôts de ROS qui ne comprenne aucune version graphique. Les outils de simulation seront donc utilisés sur ordinateur fixe, et nous lierons la version simulée sur ce poste fixe à la version réelle du robot sur le système embarqué par le biais de l'outil d'interopérabilité entre machines de ROS. Le tutoriel d'installation (en anglais) se trouve ici.

[5] Documentation sur le site Ubuntu fr, https://doc.ubuntu-fr.org/ros

2.1.4.La liaison entre systèmes

Comme évoqué précédemment, ROS s'avère particulièrement efficace lorsqu'il s'agit de relier plusieurs systèmes, qu'importe leur architecture physique.

2.2. Le Workspace - prm_ws

Le but de ce document n'est pas de décrire de façon théorique les algorithmes et architectures des programmes de ROS, bien que la compréhension de ces principes constituent une étape nécessaire pour l'utilisation de la plateforme. Il est ici question de comprendre l'architecture spécifique du robot à base mobile et les outils ROS qui ont été déployés afin de définir son fonctionnement.

Tous les fichiers sont commentés afin de permettre une meilleure compréhension des outils.

On notera que les packages ROS de la plateforme sont principalement inspirés de projets universitaires ou open source sur la robotique mobile[6] [7].

2.2.1.prm_description

Ce package est le package principal, puisqu'il décrit la composition hardware (tant composants mécaniques que capteurs électroniques du robot).

Le dossier */urdf* contient les fichiers .urdf du robot. Un fichier .urdf est un fichier lu et compris par ROS permettant une représentation en 3D du robot. Ce format est presque similaire au .xml. Le fichier « macros.xacro »spécifie des raccourcis de calcul d'inertie pour des géométries basiques. Le fichier « materials.xacro » précise les couleurs et matériaux des fichiers (exemple : bleu = code rgba 0/0/0.8/1). Le fichier « prm.gazebo » spécifie les paramètres du robot dans son environnement avec l'outil de simulation Gazebo (voir plus loin). Enfin, le fichier « prm.xacro » lie les fichiers .stl entre eux en spécifiant leurs positions absolues et relatives ainsi que leurs liaisons mécaniques.

Note : le format .xacro est en fait un fichier .urdf : l'outil xacro permet juste une manipulation plus simple des paramètres et spécificités d'un fichier .urdf.

Le dossier */meshes* contient les fichiers .stl de chaque composant du robot. Ceux-ci sont chacun assigné à un objet de type « link » décrit dans le fichier .urdf du robot.

---

[6] Moore Robotics, http://moorerobots.com

[7] Introduction à la navigation avec ROS, https://github.com/richardw05/mybot_ws

Le dossier */rviz* contient le fichier de paramètres de lancement du simulateur 3D Rviz intégré à ROS.

Le dossier */launch* contient les fichiers de lancement du modèle du robot.

2.2.2.prm_gazebo

C'est le package qui permet la liaison avec l'outil de simulation Gazebo. Gazebo est un outil permettant la modélisation complète d'un robot (quelque soit son type : mobile, industriel, drone, …) et de son environnement. Il permet la modélisation des capteurs qu'il intègre et des interactions et comportements associés. Ce package comprend donc les différents paramètres de la simulation Gazebo. Le dossier */worlds* contient les fichiers de modélisation d'environnements dans Gazebo, et le dossier */launch* contient les paramètres de lancement de Gazebo.

2.2.3.prm_control

Notre robot nécessite un modèle cinématique afin d'être simulé. Afin de ne pas effectuer des calculs déjà réalisés pour les modèles les plus basiques, Gazebo propose des « controllers » qui sont en fait des modèles cinématiques et dynamiques préétablis. Il suffit alors de spécifier les différentes caractéristiques du modèle dans les fichiers « prm.xacro » et « prm.gazebo » (couple, nombre de roues, nombre de roues motrices, inerties, …).

2.2.4.prm_navigation

Comme évoqué précédemment, il est inutile de recalculer le modèle cinématique lors de la création d'un robot sur ROS. Il en est de même pour les algorithmes de navigation autonome (GMapping, SLAM, …) : des librairies et modèles sont déjà établis dans les bases de bibliothèque de ROS et de Gazebo. Ainsi, les fichiers présents dans les dossiers */config* et */launch* spécifient les comportements liés à la navigation.

Conclusion

Pour un problème donné, plusieurs architectures peuvent correspondre et répondre de manière convenable aux besoins du cahier des charges imposé. Il est donc intéressant de proposer une architecture facilement modulable en fonction des besoins. Il est également important d'imaginer les perspectives futures du projet robotique ainsi que l'évolution éventuelle du cahier des charges initial, car cela peut facilement et rapidement influer sur le choix d'une architecture au profit d'une autre. En effet, plusieurs fonctionnalités peuvent facilement s'ajouter pour un même cahier des charges, et en cas de limitation par l'architecture robotique, un autre système devra être imaginé. Ceci étant dit, il est intéressant de noter que les améliorations d'un système se fait dans la majeure partie au niveau du traitement de l'information, ce qui bien sûr induit que l'efficacité d'un système autonome est fortement liée à son système de capteurs. Ce sont donc eux qui vont dimensionner le système robotique. Ainsi, si l'efficacité d'un robot se mesure à ses capteurs, sa complexité se mesure par sa capacité à traiter l'information de façon instantanée comme sur le long terme. Rendre un système mécanique intelligent et autonome, c'est le rendre apte à s'adapter de manière fiable sur un environnement non structuré, c'est à dire rendre la phase de décision de son cycle la plus performante, qui sera bien sûr la plus effective possible si la phase de perception est efficacement effectuée en amont.

Bien sûr, cette base de connaissances n'est pas exhaustive, et est constituée à une échelle d'un problème relativement simple, puisque l'environnement imposé et la distance à parcourir ne sont pas particulièrement complexe. Mais puisque le projet est avant tout à visée pédagogique, la plateforme robotique est naturellement amenée à développer de nouvelles fonctionnalités et à adopter les technologies les plus récentes. Ainsi, il est important de conserver et de mettre à jour ce document de synthèse.

Bibliographie

Notes de bas de page :

[1] Chassis à 4 roues motrices de DFRobot, https://www.dfrobot.com/product-97.html

[2] Tourelle « Pan-Tilt » pour caméra Raspberry Zero, https://www.robotshop.com/eu/fr/kit-tourelle-pan-tilt.html

[3] RPLiDAR A1 de Slamtec, https://www.slamtec.com/en/Lidar/A1

[4] ROS - Robot Operating System, Generation Robots, https://www.generationrobots.com/blog/fr/ros-robot-operating-system-3/

[5] Documentation sur le site Ubuntu fr, https://doc.ubuntu-fr.org/ros

[6] Moore Robotics, http://moorerobots.com

[7] Introduction à la navigation avec ROS, https://github.com/richardw05/mybot_ws

Hors bas de page :

- wiki.ROS.org, <wiki.ros.org>