IstiABot ou la Conception d'un Robot Libre pour l'Éducation et la Recherche

Rémy Guyonneau, Franck Mercier

IstiABot ou la Conception d'un Robot Libre pour l'Éducation et la Recherche

Rémy Guyonneau et Franck Mercier*

17 juin 2021

Résumé

L'IstiABot est un robot mobile terrestre réalisé dans un but conjoint de pédagogie et de recherche. Il vise à être modulaire, simple à modifier et utilisable autant par des étudiants de première année du cycle préparatoire que par des étudiants en dernière année du cycle ingénieur et des chercheurs. Pour pouvoir satisfaire ces objectifs, le robot repose sur l'utilisation d'un bus CAN (Controller Area Network). Cet article présente le robot ainsi que les raisons qui ont mené à sa fabrication. Il détaille aussi deux applications de la plate-forme dans un cadre pédagogique (développement d'une carte électronique et mise au point d'un contrôleur PID - Proportionnel, Intégral, Dérivé - pour une régulation de vitesse) et une application dans le cadre de recherches (expérimentation de cartographie et localisation simultanées). Finalement, sont aussi présentés deux autres robots conçus sur la même base que l'IstiABot. Le robot IstiABot étant pensé avec une philosophie libre, tous les codes sources, modèles 3D, schémas des cartes... sont disponibles en accès libre.

Mots-clés Robotique mobile, retour d'expérience, pédagogie, recherche, SLAM

*Polytech Angers (anciennement "IstiA"), Université d'Angers, France. `prenom.nom@univ-angers.fr`, auteur destinataire de la correspondance: Rémy Guyonneau

Article publié par EDP Sciences et disponible sur le site https://www.j3ea.org ou https://doi.org/10.1051/j3ea/20210002

1 Introduction

1.1 Contexte

La robotique demande des connaissances en électronique, en mécanique et en informatique. C'est un vaste domaine au sein duquel il est possible d'appliquer concrètement des notions théoriques liées à des problématiques de recherche. En ajoutant le fait que les étudiants sont généralement motivés quand il s'agit de travailler sur des robots, la robotique est devenue logiquement un outil pédagogique répandu [4, 5, 6].

Polytech Angers (anciennement IstiA, d'où le nom du robot) est l'école d'ingénieurs de l'Université d'Angers (France). Entre autres, elle forme les étudiants au métier d'ingénieur en automatique et génie informatique. La robotique étant de plus en plus présente dans le monde industriel, c'est tout naturellement qu'elle fait partie intégrante de la formation. Il en résulte le besoin d'avoir des plate-formes expérimentales pour les étudiants. C'est la raison première de la réalisation des IstiABots.

En plus des applications pédagogiques, l'ambition pour la plate-forme était qu'elle puisse aussi servir des intérêts recherche. En effet, la plupart des enseignants de l'école sont enseignants-chercheurs et la robotique mobile fait partie d'un des nombreux axes de recherches [7, 8]. Mutualiser une plate-forme entre la pédagogie et la recherche permet notamment de sensibiliser les étudiants au monde de la recherche.

1.2 Développer un Nouveau Robot

Malgré la pléthore de robots commerciaux disponibles, il a été décidé de réaliser un nouveau robot "fait maison". Ce choix a été motivé par plusieurs raisons.

La première est le respect des contraintes suivantes :

— La plate-forme a pour vocation à être utilisée par des étudiants, de la première année du cycle préparatoire à la dernière année du cycle ingénieur, ainsi que par des chercheurs. Ceci étant, elle doit être facile à utiliser tout en permettant des développements complexes.

— La plate-forme doit être facile à modifier. Les étudiants doivent être capables de développer leurs propres modules additionnels (mécanique ou électronique). Ils doivent par exemple être en mesure d'ajouter des capteurs/actionneurs sur le robot.

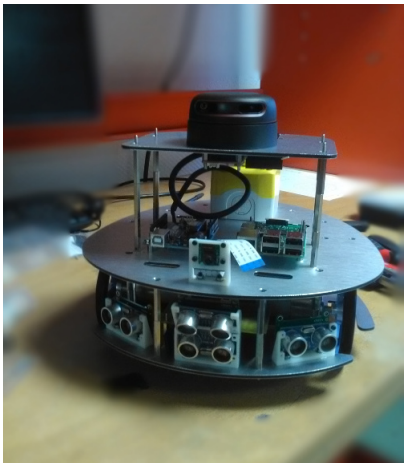


FIGURE 1 – Un robot IstiABot.

— La plate-forme doit s'intégrer autant que possible dans une démarche libre (d'un point de vue logiciel comme matériel).

La deuxième raison mise en avant est l'opportunité d'utiliser les moyens de fabrication présents à l'école et de mettre en avant ce qui peut être réalisé en interne.

Enfin, réaliser un robot permet surtout une montée en compétences tout en impliquant les étudiants dans le processus.

Voici un résumé du retour d'expérience sur le choix de faire un robot en interne :

— Inconvénients :
  — Développer un robot est très consommateur en termes de temps à consacrer au projet. Il a fallu plus d'un an pour avoir une première version de la plate-forme (comprenant la conception, la réalisation et les premières corrections...) ;
  — La documentation est à la hauteur du temps consacré... Il n'y a pas de service après-vente ni de FAQ [1] sur le fonctionnement du robot. La documentation doit donc être rédigée pendant la réalisation du robot, de façon la plus complète possible.
— Avantages :
  — Le robot répond parfaitement aux besoins. Il remplit toutes les contraintes définies précédemment ;

1. Foire Aux Questions

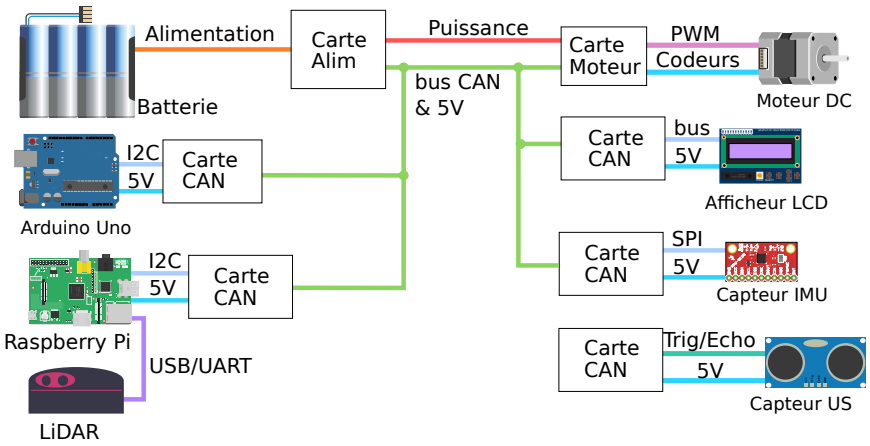


FIGURE 2 – Architecture d'un IstiABot.

— La plate-forme est parfaitement maîtrisée : elle peut donc facilement être mise à jour, modifiée, réparée... selon les besoins ;
— Tous les aspects matériels et logiciels sont en libre accès. Plusieurs tutoriels ont aussi été rédigés, principalement pour les étudiants et sont disponibles sur internet.

Cet article présente la plate-forme réalisée (visible sur la figure 1) en expliquant les choix techniques adoptés et en présentant les applications du robot. La section 2 détaille la plate-forme ainsi que les cartes "faites maison". Les sections 3 et 4 présentent des applications de cette plateforme dans un contexte pédagogique et de recherche. La section 5 présente deux autres robots réalisés sur la même base que l'IstiABot et finalement la section 6 conclut cet article.

2 PRESENTATION DU ROBOT

2.1 L'Architecture Générale

L'IstiABot est un robot à deux roues motrices différentielles (avec une roue folle pour la stabilité). Cette configuration permet une modélisation et une commande relativement simples et par conséquent est bien adaptée aux étudiants débutants.

La figure 2 présente l'architecture générale de l'IstiABot. Comme le robot a vocation à être utilisé par des étudiants de tous niveaux, il a été

jugé intéressant qu'il puisse être piloté par une carte type Arduino ou une Raspberry Pi en fonction des applications. Ces deux cartes sont largement utilisées dans un cadre pédagogique [10, 11]. Bien évidemment, l'utilisation de l'une ou de l'autre doit se faire de façon transparente pour les étudiants (sans nécessiter de câblage, démontage...). De plus, comme mentionné précédemment, il est important que les étudiants puissent aisément développer leurs propres modules sur le robot. Pour ces raisons, le robot a été conçu en s'appuyant sur un bus de communication CAN [9]. Ce bus permet une certaine flexibilité au regard des éléments (nœuds) qui sont connectés, car seuls les messages circulant sur le bus sont identifiés, pas les nœuds. En d'autres termes, il est possible de définir un message signifiant "tourner le moteur droit dans le sens horaire", et peu importe quel module envoie le message (Arduino ou Raspberry Pi). De plus, le bus CAN est un bus répandu dans le monde industriel. L'utilisation du bus CAN sur le robot permet donc un support supplémentaire pour l'étude de ce réseau par les étudiants.

Afin de limiter la quantité de fils présents dans le robot, il a été décidé d'utiliser une nappe de quatre fils permettant de faire circuler les signaux CANH et CANL (nécessaires au bus CAN) mais aussi la masse et le 5V permettant d'alimenter toutes les cartes électroniques au sein du robot.

2.2 Aspects matériels

Le robot est équipé de deux moteurs à balais de 12V. Ces derniers sont équipés de réducteurs 30 : 1 et intègrent des codeurs à quadrature qui fournissent une résolution de 64 tics pour un tour de l'arbre moteur. Un tutoriel complet pour l'assemblage du robot peut être téléchargé au lien suivant [2]. Ce tutoriel a initialement été rédigé pour les étudiants, mais est accessible à tous. L'objectif est de sensibiliser les étudiants aux différentes parties nécessaires à la réalisation d'un robot

La puissance du robot est fournie par une batterie lithium-ion 4S (environ 14.4V).

Toutes les parties structurelles (en Panneaux Composite Aluminium - PCA) du robot ont été usinées à l'aide d'une machine CNC [3]. Les parties plastiques ont été imprimées à l'aide d'imprimantes 3D. Tous les modèles 3D

2. `https://github.com/PolytechAngersMecatroniqueClub/Tutorials/blob/master/Tuto_assembling/tutoIstiaBot_montage.pdf`

3. Computer Numerical Control

des parties mécaniques sont disponibles au lien suivant [4].

Sur la figure 2 on notera que toutes les images "graphiques" correspondent à des éléments achetés tels quels (cartes Ultra-sons, Arduino...) tandis que les "rectangles" (carte d'alimentation, carte moteur...) ont été réalisés en interne (conception et réalisation). Les sous-sections suivantes présentes les caractéristiques de ces cartes "faites maison".

2.3 La Carte d'Alimentation

L'objectif de la carte d'alimentation est de convertir la puissance de la batterie en plusieurs alimentations pour les différents éléments du robot. Pour l'IstiABot, seulement deux alimentations différentes sont nécessaires : la puissance pour les moteurs (12V) et l'alimentation de toutes les cartes électroniques (5V). Les 12V sont directement envoyés aux cartes moteurs, tandis que les 5V circulent dans le robot à l'aide de la nappe 4 fils (5V, GND et les deux fils du bus CAN), visible sur la figure 2.

La carte d'alimentation possède aussi un capteur de tension sur son entrée. Elle utilise le bus CAN pour transmettre la tension courante de la batterie. Pour l'IstiABot, cette information de tension est traitée par l'Interface Homme/Machine (IHM), équipée d'un avertisseur sonore, qui émet un signal quand la tension est jugée trop faible. Ce signal permet de limiter la dégradation de la batterie pour cause de sous-tension.

Les schémas et les codes sources de cette carte sont disponibles à l'adresse suivante [5].

2.4 La Carte Moteur

Cette carte, schématisée sur la figure 3, permet de contrôler un moteur à courant continu selon une consigne de vitesse. Elle nécessite deux alimentations différentes : 5V pour l'électronique et 12V pour le moteur. Elle reçoit la commande de vitesse via le bus CAN. Elle génère ensuite un signal MLI [6] qui est envoyé aux moteurs par l'intermédiaire d'un pont en H. Afin de contrôler la vitesse du moteur, les valeurs du codeur sont traitées et une régulation de la vitesse est effectuée à l'aide d'un régulateur PID [7]. Ce régulateur est décrit

4. `https://github.com/PolytechAngersMecatroniqueClub/Frame`
5. `https://github.com/PolytechAngersMecatroniqueClub/Power_Board`
6. Modulation de Largeur d'Impulsion - Pulse Width Modulation
7. Proportionel Integral Derivé

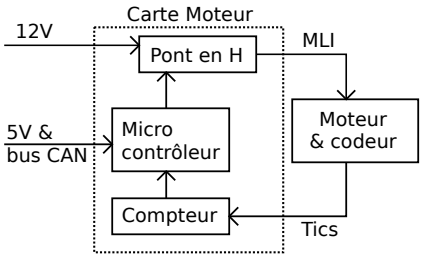


FIGURE 3 – Principe de fonctionnement de la carte moteur. Par le biais du bus CAN, sont envoyés des messages contenant une commande de vitesse pour les moteurs (les valeurs sont en $rad.s^{-1}$).Un régulateur PID est implémenté dans le micro-contrôleur pour réguler la vitesse du moteur.

en section 3.2.

Cette carte intègre également un capteur de courant. Ses données peuvent être traitées pour détecter les pics de courant pouvant survenir lorsque le robot est bloqué pendant d'une tentative de déplacement par exemple.

Elle a été conçue à l'origine pour la technologie brushless et possède trois canaux de sortie correspondant à chaque bobine du moteur et trois capteurs à effet Hall sont câblés pour le retour. Cette carte est optimisée d'un point de vue électronique pour le moteur Maxon EC45 sans balais [8], mais il était également prévu d'utiliser la technologie à courant continu avec rétroaction du codeur via un compteur en quadrature LS7366R [9].

Tous les schémas et codes sources de cette carte se trouvent dans le dépôt github suivant [10].

2.5 Les Cartes CAN

L'inconvénient d'un robot modulaire basé sur un bus CAN est que chaque composant du robot doit être associé à une interface CAN (capteurs, IHM [11], etc.). *A contrario*, cela permet de changer facilement les composants et modules du robot, en supposant que les messages CAN soient correctement envoyés et reçus. Par exemple, cela permet au robot d'être contrôlé par le Raspberry Pi ou l'Arduino sans avoir besoin de modifier le câblage.

8. `https://www.maxonmotor.com/maxon/view/product/397172`
9. `https://lsicsi.com/datasheets/LS7366R.pdf`
10. `https://github.com/PolytechAngersMecatroniqueClub/Motor_Board`
11. Interface Homme Machine

Néanmoins, une connectivité bus CAN doit être développée pour chaque sous ensemble du robot. La chaîne connecteur / transceiver (MCP2551 [12] / micro-Contrôleur Atmega32m1 [13]) a été respectée sur toutes les cartes électroniques. L'uniformité a été prise en compte également pour les connecteurs afin de faciliter l'ajout de nouveaux éléments.

Tous les schémas et codes sources de ces cartes se trouvent dans des dépôts github [14]. Un tutoriel pour configurer la Raspberry Pi a également été écrit afin de pouvoir programmer les micros-contrôleurs avec cette carte. Ce tutoriel se trouve également sur le dépôt [15]. Enfin, un tutoriel visant à faciliter la programmation de l'interface CAN pour le micro-contrôleur a également été rédigé [16].

L'utilisation d'une interface CAN pour tous les sous-ensembles permet également d'être indépendant quand au langage de programmation : tant qu'il est possible de lire/écrire un message CAN, il est possible d'accéder à tous les éléments du robot.

3 APPLICATIONS PÉDAGOGIQUES

Comme mentionné précédemment, le robot est utilisé aussi bien par des étudiants débutants que par des étudiants plus expérimentés. L'objectif est que les étudiants puissent progresser avec la même plate-forme tout au long de leur processus d'apprentissage. Ceci est principalement permis par le bus CAN et le fait qu'une carte Arduino ou une carte Raspberry Pi puisse être utilisée pour contrôler le robot.

Voici un exemple de progression :

— En supposant le robot fonctionnel, il est aisé de le faire se mouvoir par programmation Arduino. L'utilisation de Scratch pour Arduino permettra le contrôle du robot même par des élèves n'ayant jamais appris la programmation. Dans la même philosophie, il est possible d'obtenir facilement les valeurs des capteurs et de commencer à développer des algorithmes de déplacements simples ;

12. https://www.microchip.com/wwwproducts/en/en010405
13. https://www.microchip.com/wwwproducts/en/ATmega32M1
14. https://github.com/PolytechAngersMecatroniqueClub
15. https://github.com/PolytechAngersMecatroniqueClub/Tutorials/blob/master/tutoIstiABot_raspberry.pdf
16. https://github.com/PolytechAngersMecatroniqueClub/Tutorials/blob/master/tutoIstiABot_CAN_ATMEGA.pdf

— L'introduction de boucles de régulation et de programmation embarquée peut être effectuée par une programmation bas niveau sur les micro-contrôleurs embarqués dans les cartes. Par exemple en configurant le bus CAN ou en implémentant une régulation de vitesse sur les cartes moteurs ;
— En utilisant le Raspberry Pi et le capteur LiDAR, il est alors possible de s'essayer à des algorithmes plus complexes tels que la cartographie ou la planification de trajectoire... A ce stade de la progression, il est possible d'introduire des problèmes liés à la recherche scientifique et au développement d'algorithmes complexes.

Dans la partie suivante de ce document, deux exemples de résultats pédagogiques sont détaillés :

— Le développement d'un nouveau module pour le robot (une carte interface homme machine (IHM)) ;
— L'implementation d'un régulateur PID.

3.1 Développement d'une carte Interface Homme/Machine (IHM)

L'un des derniers modules ajoutés à l'IstiABot est l'IHM. Initialement, les robots ne possédaient aucune interface utilisateur : ils manquaient de retour visuel et étaient incapables de fonctionner sans ordinateur distant (connexion SSH).

C'est pourquoi les étudiants ont été invités à créer une carte interface. Selon les spécifications, la carte devait avoir : un écran LCD, des boutons-poussoirs, des LEDs pour le retour visuel et un avertisseur sonore.

Une carte afficheur LCD 2*16 caractères commerciale standard a été utilisée comme point de départ. Ensuite, elle a été intégrée sur une carte électronique conçue par les élèves, avec l'ajout de fonctionnalités CAN pour pouvoir connecter la carte au reste du robot. Voici les différentes étapes du projet :

— Lecture de la documentation de la carte LCD achetée afin de définir comment la contrôler avec un micro-contrôleur ;
— Conception d'un schéma incluant le LCD, le micro-contrôleur et les fonctions périphériques (interface CAN, bouton-poussoir, avertisseur sonore, led) ;
— Réalisation de la nouvelle carte et tests ;

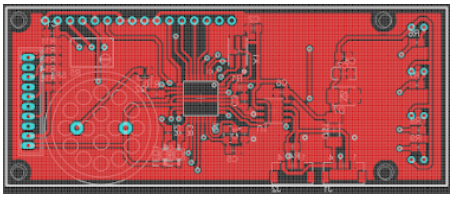


FIGURE 4 – Vue du dessous de la carte IHM développée.

— Programmation du micro-contrôleur pour gérer le bus CAN et traiter les entrées/sorties.

Ce projet est assez complet, car il nécessite des étapes de lecture de documentation technique, de conception, de réalisation et de programmation. Les ressources nécessaires à la réalisation de cette IHM sont un logiciel de CAO [17] pour la conception de la carte, une machine de gravure mécanique permettant de produire la carte et bien sûr, le moyen de souder des composants électroniques.

Les difficultés de ce projet sont nombreuses... Pour le schéma par exemple, une contrainte a été donnée aux étudiants : l'utilisation du micro-contrôleur ATmega32M1. Par conséquent, un câblage approprié des ports d'entrée/sortie du micro-contrôleur est nécessaire. Il est important de comprendre et d'anticiper le contrôle 4 bits de l'écran LCD, l'utilisation des entrées d'interruption pour les boutons-poussoirs, les sorties pour le bus CAN, etc... Cette étape schématique validée, vient la phase de routage pendant laquelle les étudiants doivent veiller à l'optimisation du placement, à la CEM [18] de la carte et doivent anticiper le processus de réalisation pour s'assurer de sa faisabilité.

Cette carte IHM est maintenant opérationnelle sur les IstiAbots.

3.2 Implémentation d'un asservissement de vitesse

Initialement, le contrôle de la vitesse du robot était en "boucle ouverte" : des commandes étaient envoyées aux moteurs sans vérifier la vitesse effective. Comme cela était prévisible, il en résultait des problèmes de direction et une vitesse non homogène. Un projet étudiant a donc été mis en place pour résoudre ce problème.

17. Conception Assistée par Ordinateur
18. Compatibilité Électro-Magnétique

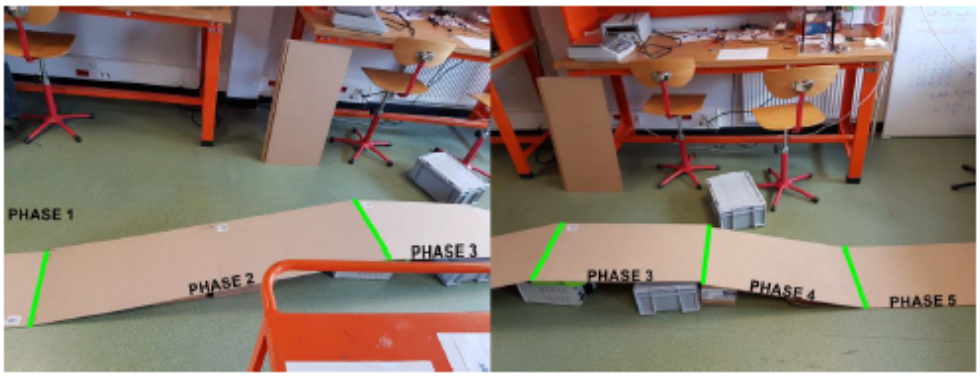


FIGURE 5 – Configuration expérimentale : la trajectoire attendue est une ligne droite passant par cinq phases avec des pentes différentes. Le robot doit se déplacer sur le chemin à vitesse constante.

3.2.1 Présentation du problème

La figure 5 montre une expérimentation simple qui illustre le problème : le robot doit effectuer un trajet en ligne droite, à vitesse constante. Le chemin est divisé en cinq phases avec des pentes différentes. Tout d'abord, l'expérimentation est effectuée à l'aide d'une commande en boucle ouverte : une commande MLI est appliquée à chaque moteur. En utilisant les mesures des codeurs des roues envoyées par les cartes moteurs sur bus CAN, il est assez facile d'enregistrer la vitesse de chaque moteur. Les résultats de cette première expérimentation sont présentés sur la figure 6.

Ces courbes illustrent deux problèmes :

— La vitesse n'est pas constante sur la trajectoire. Elle diminue lorsque la pente est positive et inversement.

— Le robot n'avance pas droit. Ceci est déduit du fait que la roue gauche n'a pas la même vitesse que la droite.

Cette expérimentation simple permet de présenter le problème aux étudiants et de définir le comportement idéal : les courbes doivent être identiques et égales au point de consigne. Les étudiants sont maintenant en mesure d'expliquer l'objectif du projet et, plus tard, de comparer leurs résultats avec cette référence pour quantifier le gain de la correction.

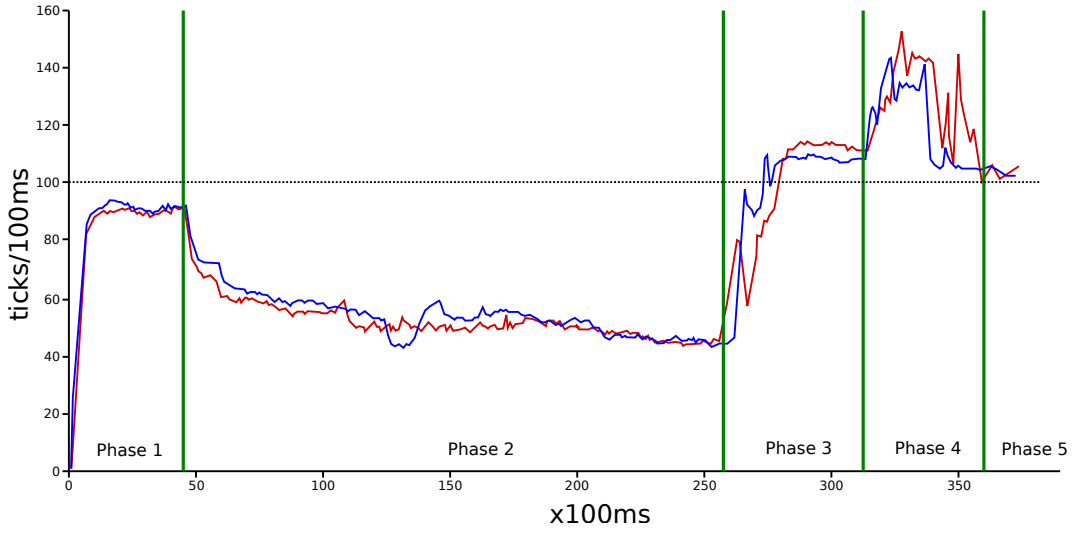


FIGURE 6 – Résultats d'une commande de boucle ouverte. Le graphique indique l'évolution de la vitesse des moteurs (nombre de tics des codeurs toutes les 100 ms) au passage des différentes phases. La courbe rouge correspond au moteur droit et la bleue au moteur gauche. La consigne théorique de chaque moteur est de 100 tics/100 ms, ce qui équivaut à 0,52 tour/s dans ce cas.

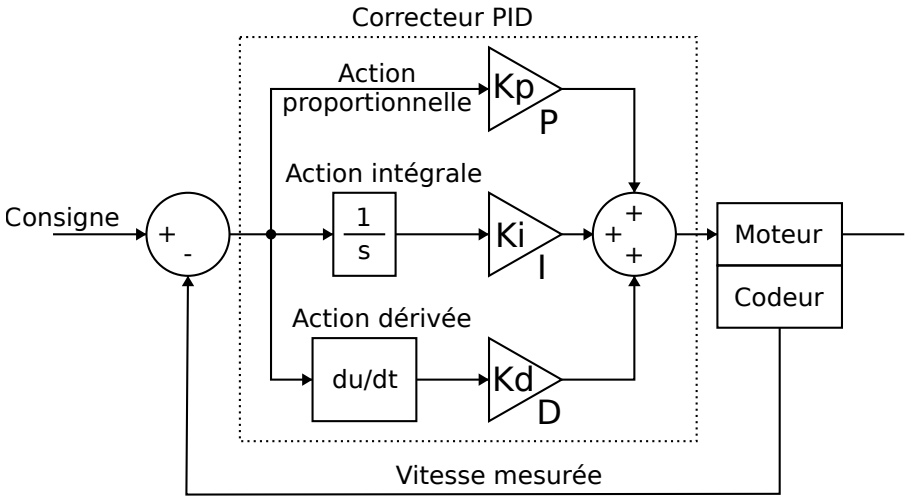
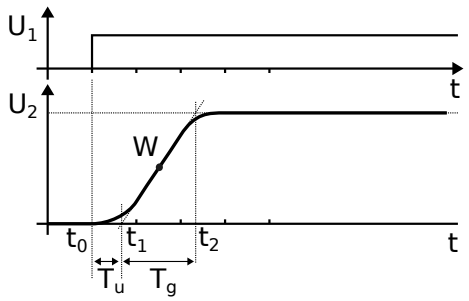


FIGURE 7 – Présentation du correcteur PID.



(a) Paramètres CHR à relever expérimentalement sur la réponse indicielle en boucle ouverte.

Type de régulation	Ajustement des paramètres Chien, Hornes et Reswick (CHR)		
$PID(K_p;T_n;T_v)$	$K_p = \frac{0.6T_g}{T_u}$	$T_n = T_g$	$T_v = 0.5T_u$

(b) Table de correspondance CHR.

FIGURE 8 – Méthode d'identification CHR.

3.2.2 Méthodologie adoptée

La mise en œuvre d'une boucle de régulation, avec correcteur PID dans notre cas, est nécessaire pour maintenir à la fois la vitesse et la direction. Ce contrôleur est illustré sur la figure 7. Il est ajouté dans un système bouclé et fonctionne sur un signal d'erreur (la différence entre la consigne et le signal mesuré). Il existe plusieurs structures possibles pour les contrôleurs PID. Celle considérée pour l'IstiABot est définie par l'équation 1.

$$C_{pid} = K_p \times \left(ec + \frac{1}{T_n} \int ec \times \mathrm{d}t + T_v \frac{\mathrm{d}(ec)}{\mathrm{d}t} \right), \tag{1}$$

avec $K_i = \frac{K_p}{T_n}$, $K_d = K_p \times T_v$, C_{pid} la commande (sortie du correcteur PID) et ec l'erreur (différence entre la vitesse attendue - la consigne - et la vitesse mesurée).

La principale difficulté pour l'implémentation d'un PID consiste à ajuster les valeurs des coefficients K_p, K_i and K_d. Il existe beaucoup de méthodes pour évaluer ces paramètres. Celle choisie ici est la méthode CHR [19] [13]. Cette méthode explique comment évaluer chaque coefficient à partir de la réponse indicielle du système.

19. Chien, Hornes et Reswick.

Cette approche permet une estimation théorique des paramètres PID. Mais pour que le contrôleur ait le comportement attendu, il est nécessaire d'affiner ces valeurs expérimentalement.

Voici les différentes étapes pour affiner les paramètres PID expérimentalement :

— En boucle ouverte, obtenir la réponse indicielle du système ;
— Selon l'approche CHR (Figure 8) et la réponse indicielle, calculer les valeurs théoriques pour K_p, K_i et K_d ;
— Passer en boucle fermée et régler K_p à sa valeur théorique, K_i et K_d à zéro. Obtenir la réponse indicielle du système ;
— Comme indiqué sur la figure 9 régler K_p à sa valeur finale et ne plus la changer à partir de maintenant ;
— Régler K_i à sa valeur théorique et K_d à 0. Obtenir la réponse indicielle du système ;
— Comme indiqué sur la figure 9 régler K_i à sa valeur finale et ne plus la changer ;
— Régler K_d à sa valeur théorique et obtenir la réponse indicielle du système ;
— Comme indiqué sur la figure 9 régler K_d à sa valeur finale, Le correcteur PID est réglé.

3.2.3 Résultats de l'implémentation du correcteur PID

En comparaison des courbes de la figure 6, la même expérimentation a été effectuée et les résultats sont présentés sur la figure 10.

Comme on pouvait l'espérer, les courbes sont bien meilleures : le point de consigne est atteint pour chaque phase et le comportement des moteurs gauche et droit est identique. En d'autres termes, la vitesse reste constante et le robot se déplace en ligne droite quelle que soit la pente.

3.2.4 Conclusion du projet

Ce projet permet aux étudiants de travailler sur un problème d'automatique récurrent (la régulation) via une application pratique. Grâce au robot, ils ont une bonne compréhension du problème, des conséquences et des résultats attendus. Ce projet permet de mélanger approche théorique avec expérimentation et programmation. De plus, le fait que le robot se comporte comme prévu semble être une récompense aussi importante que la note...

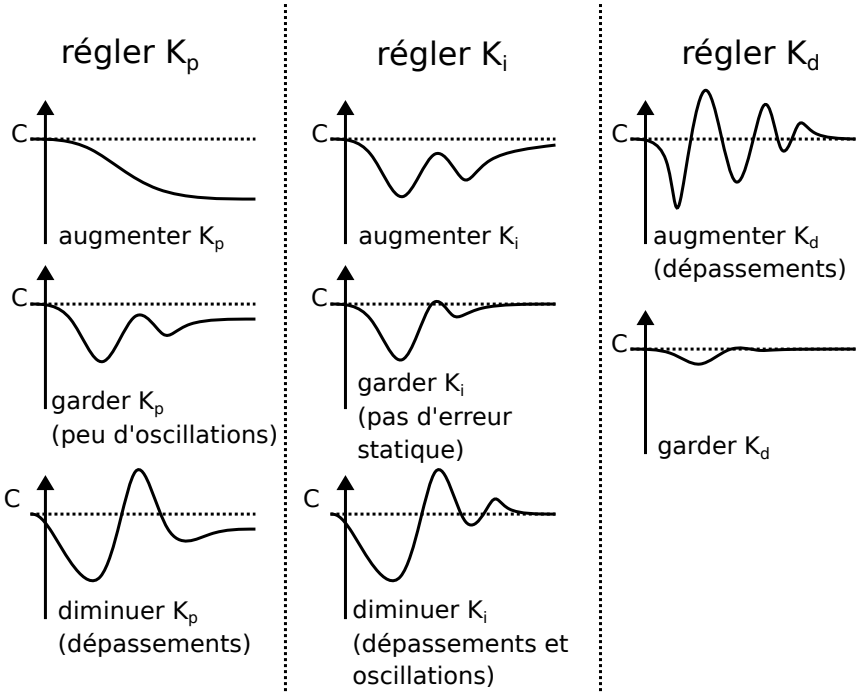


FIGURE 9 – Réglage expérimental. Les graphiques correspondent à des réponses indicielles, C étant la consigne.

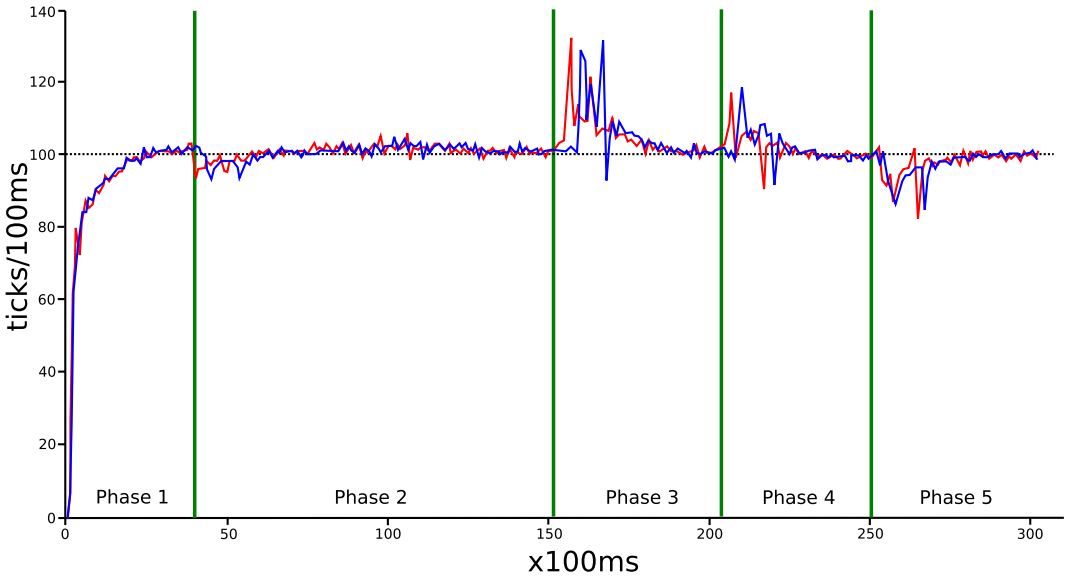


FIGURE 10 – Résultats de l'expérimentation après la mise en œuvre et le réglage du contrôleur PID.

4 APPLICATION EN RECHERCHE

4.1 Problématique du SLAM

Pour qu'un robot mobile soit totalement autonome, il doit pouvoir évaluer sa position à tout moment. Afin de pouvoir définir sa position, il doit disposer d'une représentation connue de son environnement : une carte. Dans de nombreuses situations, la carte n'est pas connue *a priori*. Dans ce cas, le robot doit construire la carte et se localiser en même temps. La difficulté étant qu'il a besoin de son emplacement pour construire la carte et il a besoin d'une carte pour se localiser... Ce problème est appelé SLAM [20]. Un algorithme de SLAM, développé au sein du laboratoire [12], a été implémenté et testé à l'aide un IstiABot.

4.2 Vue d'ensemble de l'algorithme

L'objectif est de construire une grille 2D pour représenter l'environnement. Dans chaque cellule de cette grille, on calcule la probabilité de la

20. Simultaneous Localization And Mapping - Localisation Et Cartographie Simultanées

présence d'un obstacle. Pour construire cette carte, l'algorithme utilise uniquement des données LiDAR [21]. Une mesure LiDAR correspond à la distance entre le robot (le capteur) et l'obstacle le plus proche dans une direction donnée. Le LiDAR utilisé est un LiDAR 2D [22] permettant, pour chaque jeu de mesures, d'acquérir 360 distances/angles tout autour du robot. En utilisant ces données, l'idée est de construire la carte de manière itérative en ajoutant au fur et à mesures les obstacles détectés. Les étapes pour construire cette carte sont illustrées sur la figure11. Voici son fonctionnement général :

- **Acquérir des données LiDAR** : Avec le RPLIDAR A2 installé sur les IstiABots, un jeu de données correspond à 360 mesures autour du capteur (1 degré entre chaque mesure) sur une portée de cinq mètres. Une mesure est donc définie par une distance et une direction : la distance entre le capteur et l'obstacle le plus proche dans une direction donnée (coordonnées polaires).
- **Initialiser la carte avec des obstacles et les zones libres** : À l'instant t_0 le robot se trouve dans sa posture initiale (posture : position et orientation, $(0, 0, 0)$ par exemple). Il est donc possible de calculer toutes les positions (et donc les cellules de la grille correspondantes) des obstacles détectés par le capteur. En faisant cela, une probabilité pour chaque cellules peut être initialisée avec ce premier jeu de données.
- **Estimer la transformée des données LiDAR** : À ce stade, nous avons une carte partielle (tout juste initialisée ou résultante des itérations précédentes) et un nouveau jeu de mesures. L'objectif est alors de trouver la transformation (rotation et translation) des nouvelles mesures pour quelles soient cohérentes avec la carte actuelle. Pour cela on cherche à minimiser le coût de l'ensemble des mesures (la somme des distances entre les nouveaux obstacles, i.e. les mesures, et les obstacles connus, i.e. la carte). Pour résoudre ce problème d'optimisation, nous utilisons l'algorithme de Nelder et Mead [14].
- **Mettre à jour la carte avec les obstacles et les zones libres** : une fois la meilleure posture trouvée (qui minimise le coût des mesures), nous pouvons mettre à jour la carte avec les nouveaux obstacles et espaces libres détectés.

21. Light detection and ranging - Détection et télémétrie par lumière
22. Un RPLIDAR A2

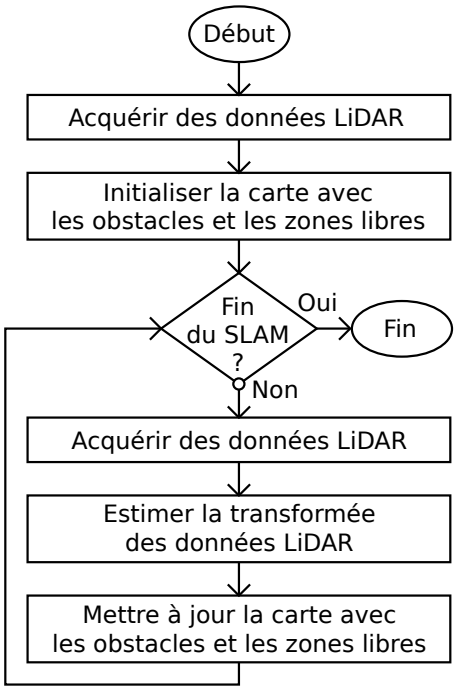


FIGURE 11 – Principe de l'algorithme SLAM.

4.3 Implementation et resultats

L'algorithme de SLAM a d'abord été implémenté sur l'IstiABot par un étudiant en master. Cette première implémentation a été réalisée à l'aide du framework ROS [23]. Ce framework peut sembler difficile au premier abord, mais une fois le premier pas franchi, il permet des développements efficaces et propose de nombreux outils. Quelques avantages (parmi d'autres) :

- C'est un framework Open Source ;
- Le principe de base consiste à développer des nœuds (programmes) qui communiquent entre eux par l'intermédiaire de messages. C'est une architecture qui permet une grande modularité ;
- Des "logs" peuvent facilement être enregistrés en utilisant des *rosbags* : il est possible de sauvegarder toutes les informations d'un robot et les rejouer *a posteriori* ;
- Un nœud peut être implémenté en C++ ou en python ;
- ROS est basé sur une communication TCP / IP afin que les nœuds puissent être exécutés localement, mais également sur des ordinateurs distants, ceci sans modifier le code ;
- Il existe un logiciel de visualisation (*rviz*) interfacé avec ROS qui permet d'afficher des cartes, des mesures de capteurs, des trajectoires, etc... De plus à l'aide de la communication TCP/IP, rviz peut être exécuté localement ou sur un ordinateur distant.
- Les capteurs de l'IstiABot ont tous des pilotes ROS et des exemples documentés (le LiDAR utilisé sur l'IstiABot par exemple) ;
- Il y a une grande communauté et beaucoup de nœuds sont librement disponibles.

Pour toutes ces raisons, la programmation de haut niveau de l'IstiABot est basée sur ROS. Une version libre du SLAM développé peut être téléchargée ici [24].

En raison de la puissance de calcul limitée du Raspberry Pi, pour pouvoir cartographier un grand environnement avec une précision acceptable, il est recommandé de traiter l'algorithme de SLAM sur un ordinateur plus puissant. Cela peut être fait en ligne en exécutant le nœud SLAM sur un ordinateur distant tout en exécutant le nœud LiDAR sur l'IstiABot, ou hors ligne en enregistrant toutes les données (rosbag) et en les traitant ultérieurement. Les résultats présentés dans la figure 12 ont été traités hors ligne, à l'aide

23. Robot Operating System
24. `https://github.com/PolytechAngersMecatroniqueClub/IstiaSLAM`

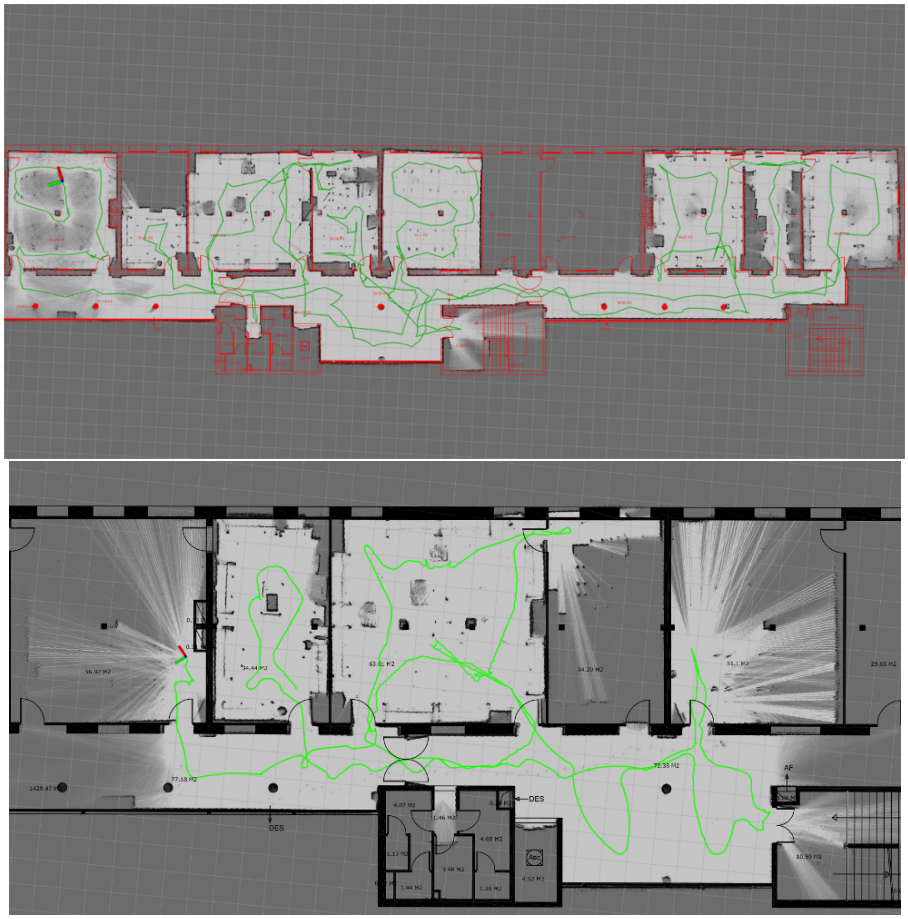


FIGURE 12 – Cartes obtenues avec un IstiABot et notre algorithme de SLAM. Il s'agit d'une représentation vue du dessus d'un environnement intérieur : la zone grise correspond à la zone de l'environnement que le robot n'a pas explorée, la zone blanche correspond à la zone détectée par le robot comme espace libre d'obstacles, les lignes noires et rouges représentent la carte d'architecte des locaux (pour vérifier la cohérence de la cartographie avec une vérité terrain), et la ligne verte qui "serpente" représente la trajectoire calculée du robot.

de fichiers rosbag acquis par un IstiABot. Les fichiers rosbag en question peuvent être téléchargés sur le dépôt [25].

5 EVOLUTIONS

Comme mentionné dans l'introduction (section 1), la plate-forme IstiABot a été conçue pour être facilement modifiable. Le bus CAN est l'élément clé de la polyvalence des robots. Avec exactement les mêmes cartes et moteurs, il a été possible de concevoir d'autres robots dotés de capacités différentes. Deux de ces robots sont illustrés sur la figure 13 et sont rapidement présentés ci-dessous.

5.1 Self-Balancing Robot

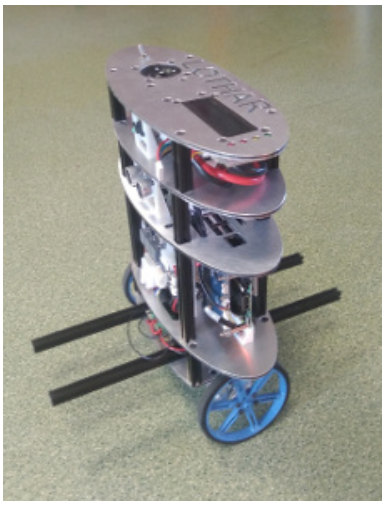
Ce robot (figure 13a) est basé sur le principe du pendule inversé. Il ne dispose que de deux roues et est donc naturellement instable. L'enjeu d'un tel robot est de mettre en place une commande qui le maintienne en équilibre tout en lui permettant de se déplacer (tourner, reculer, avancer, etc.).

Pour construire ce robot, tous les sous-ensembles et les résultats des projets pédagogiques présentés précédemment ont été nécessaires. Le correcteur PID a été mis en œuvre sur chaque carte moteur : ainsi lorsqu'une commande de vitesse est envoyée via le bus CAN, elle est supposée respectée par les moteurs.

Un deuxième correcteur a du être mis en place : non plus sur la vitesse des roues mais pour faire tenir le robot à la verticale. Une fois encore, un correcteur PID a été choisi pour cette tâche. Un capteur IMU [26] permet de mesurer les orientations du robot, et donc de calculer une erreur par rapport à la consigne (robot à la verticale). Cette erreur est donc mise en entrée de notre correcteur qui en sortie nous donne la commande à appliquer aux moteurs afin de la corriger. Cette boucle de rétroaction a été implémentée en C++ sur une carte Arduino.

Une vidéo du résultat obtenu est disponible ici [27], et la simulation d'un self-balancing robot (utilisée pour des résultats préliminaires) est disponible

25. `http://perso-laris.univ-angers.fr/~r.guyonneau/rosbags/`
26. Inertial Mesurement Unit - Centrale Inertielle
27. `https://youtu.be/G2n9_4nhEaQ`



(a) Self-Balancing Robot.

(b) Ball-Balancing Robot.

FIGURE 13 – Robots basés sur l'architecture des IstiABots.

ici [28] (Code Python3 utilisant OpenGL pour le rendu).

5.2 Ball-Balancing Robot

Ce robot correspond à une version complexifiée du robot précédent. Ici aussi le robot est par nature instable, mais il doit cette fois se maintenir en équilibre sur un ballon. Pour cela il dispose de trois roues omni-directionnelles réparties en triangle (Figure 13b). Malgré ce changement de roues, les composants du robot restent les mêmes que ceux des IstiABots et du self-balancing robot.

Une fois le robot capable de se maintenir en équilibre, l'objectif est de pouvoir le déplacer en faisant rouler le ballon.

Ce projet a démarré en 2018. Au moment de la rédaction de cet article (début 2020), la partie "hardware" a été réalisée et les étudiants travaillent sur la modélisation et la programmation du robot.

6 Conclusion

Comme il a été présenté dans cet article, l'IstiABot a permis de répondre à toutes les contraintes définies concernant la plate-forme : elle est modulaire, nous en avons une maîtrise complète (matériel et logiciel), elle peut être utilisée par les étudiants de la première à la dernière année, elle est utilisée pour des travaux de recherche...

Les IstiABots autorisent un large panel de projets d'étudiants (mécanique, électronique, informatique, etc.). De plus la possibilité de tester des algorithmes de recherche est très confortable, compte tenu de la possibilité de personnaliser ce robot en ajoutant et en supprimant des capteurs en fonction des besoins.

Concernant notre retour d'expérience sur le fait de réaliser nos robots en interne : le principal inconvénient de ce choix est le temps à consacrer à la réalisation des robots. En effet, il nous a fallu plus d'un an pour avoir la première version du robot. De plus les IstiABots ne sont pas parfaits (connecteur parfois capricieux, batterie difficilement accessible)... Mais ces inconvénients sont au final largement compensés par les avantages à réaliser nos robots :

28. `https://github.com/rguyonneau/Self-Balancing-Robot-Python-Simulator`

FIGURE 14 – Exemples de robots réalisés à Polytech Angers.

— Gain d'expérience concernant la réalisation de robots. Il s'avère que depuis le premier IstiABot, nous avons largement augmenté la quantité de robots conçus et réalisé en interne (figure 14). Nous sommes de plus en plus efficaces dans leurs réalisations ;
— Les étudiants sont impliqués dans le processus. L'objectif de réalisation d'un robot s'avère être moteur concernant les motivations des étudiants.
— Valorisation des outils de productions disponibles au sein de l'école ;
— Mise à disposition de démonstrateurs pour présenter l'école (journées portes ouvertes, salons étudiants...) mais aussi pour vulgariser la recherche (nuit des chercheurs, fête de la science...) ;
— Enfin, comme mentionné plus haut, nous avons une plate-forme adaptée à nos besoins, et toutes les compétences pour en refaire une si les besoins changent ;

Si nous devions refaire ce choix (acheter des plate-formes ou les faire nous-même), nous repartirions sur la réalisation d'un robot en interne sans hésiter.

Finalement, un effort a été fait pour développer ce robot (matériel et logiciel) avec une philosophie de licence libre. Nous espérons que cela permettra à tout un chacun de dupliquer et de personnaliser ce robot. *A minima* nous espérons que ces ressources seront utiles pour celles et ceux qui souhaitent développer leurs propres robots.

Remerciements

Merci à Philippe Lucidarme pour son aide lors de la conception du robot et pour ses schémas de cartes. Merci à Baptiste Hamard pour son travail sur l'implémentation du PID et à Vincent Cueille pour la première implémentation de l'algorithme de cartographie basée sur le framework ROS. Enfin, merci au département SAGI (Systèmes Automatisés et Génie Informatique) de Polytech Angers qui a subventionné les robots.

Références

[1] L. Lindner, O. Sergiyenko, J. C. Rodriguez-Quinonez, M. Rivas-Lopez, D. Hernandez-Balbuena, W. Flores-Fuentes, F. Natanael Murrieta-Rico and V. Tyrsa, Mobile robot vision system using continuous laser scanning for industrial application, Industrial Robot : An International Journal, 2016, Vol. 43 Issue : 4, pp.360-369.

[2] I. Nielsen, Q-V. DangEmail, G. Bocewicz and Z. Banaszak, A methodology for implementation of mobile robot in adaptive manufacturing environments, Journal of Intelligent Manufacturing, June 2017, Volume 28, Issue 5, pp 1171-1188.

[3] C. SprunkEmail, B. Lau, P. Pfaff and W. Burgard, An accurate and efficient navigation system for omnidirectional robots in industrial environments, Autonomous Robots, February 2017, Volume 41, Issue 2, pp 473-493.

[4] F. M. Lopez-Rodriguez, F. Cuesta, Andruino-A1 : Low-Cost Educational Mobile Robot Based on Android and Arduino, Journal of Intelligent and Robotic Systems, 2015, 81 :63-76.

[5] B. Curto and V. Moreno, Robotics in Education, Journal of Intelligent and Robotic Systems, 2016, vol. 81, no 1, p. 3.

[6] J. M. Gomez-de-Gabriel , A. Mandow, J. Fernandez-Lozano and A. Garcia-Cerezo, Mobile robot lab project to introduce engineering students to fault diagnosis in mechatronic systems, 2015, IEEE Transactions on Education, 58(3), 187-193.

[7] M. Mustafa, A. Stancu, N. Delanoue and E. Codres, Guaranteed SLAM - An interval approach, Robotics and Autonomous Systems. 2018. Vol. 100 p. 160-170.

[8] R. Guyonneau, S. Lagranges, L. Hardouin and P. Lucidarme, Guaranteed Interval Analysis Localization for Mobile Robots, Journal on Advanced Robotics, 2014, Vol. 28 n 16 p. 1067-1077.

[9] Robert Bosch, Specification, C.A.N. (1991). Bosch. GmbH, Postfach, 50.

[10] P. Jamieson and Jeff Herdtner, More missing the Boat-Arduino, Raspberry Pi, and small prototyping boards and engineering education needs them, 2015, IEEE Frontiers in Education Conference (FIE).

[11] J. Sobota et al., Raspberry Pi and Arduino boards in control education, IFAC Proceedings Volumes, 46.17 (2013) : 7-12.

[12] A. Bautin, P. Lucidarme, R. Guyonneau, O. Simonin, S. Lagrange, N. Delanoue and F. Charpillet, Cart-O-matic project : autonomous and collaborative multi-robot localization, exploration and mapping, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, 2013, Tokyo.

[13] K. J. Astrom and T. Hagglund, PID controllers : theory, design, and tuning. Research Triangle Park, NC : Instrument society of America, 1995.

[14] J. E. Dennis and D. J. Woods, Optimization on microcomputers : The Nelder-Mead simplex algorithm, New computing environments : microcomputers in large-scale computing, 1987, vol. 11, p. 6-122.

Webographie

— Le github de Polytech Angers Mecatronique Club
  — `https://github.com/PolytechAngersMecatroniqueClub/pa_slam` : l'algorithme de SLAM présenté dans l'article, ce dépôt contient une démonstration qui devrait fonctionner "out of the box"
  — `https://github.com/PolytechAngersMecatroniqueClub/IBallBBOT` : une simulation d'un ball balancing robot en python
  — `https://github.com/PolytechAngersMecatroniqueClub/IBBOT` : le code Arduino de notre robot "pendule inversé"
  — `https://github.com/PolytechAngersMecatroniqueClub/Tutorials` : Plusieurs tutoriels (un tutoriel pour l'assemblage du robot, une aide pour programmer le bus CAN sur l'ATMega et un tutoriel pour configurer et utiliser la Raspberry Pi)

- https://github.com/PolytechAngersMecatroniqueClub/HMI_Board : les informations relatives à la carte d'interface (les schémas, la documentation, le code de l'ATMega)
- https://github.com/PolytechAngersMecatroniqueClub/Frame : les modèles 3D du chassis du robot, des supports caméra et des supports des capteurs ultra-sons.
- https://github.com/PolytechAngersMecatroniqueClub/Motor_Board : les informations concernant les cartes moteurs (les schméas, la documentation, le code de l'ATMega)
- https://github.com/PolytechAngersMecatroniqueClub/ibot_can_interface : Le code du nœud ROS permettant d'accéder aux informations du bus CAN
- https://github.com/PolytechAngersMecatroniqueClub/Power_Board : les informations concernant la carte d'alimentation (les schémas, la documentation, le code de l'ATMega)
- https://github.com/PolytechAngersMecatroniqueClub/US_Board : les informations concernant la carte du capteur ultra-son (les schémas, la documentation, le code de l'ATMega)
- https://github.com/PolytechAngersMecatroniqueClub/IMU_Board : les informations concernant la carte de la centrale inertielle (les schémas, la documentation, le code de l'ATMega)
- https://github.com/PolytechAngersMecatroniqueClub/pa_mobile_robot : la dernière version de notre algorithme de SLAM, avec l'ajout d'un algorithme d'exploration. Nous avons considéré la possibilité de traiter les informations venant d'un IstiABot (cependant la carte n'est plus faite en locale sur le robot mais sur un PC distant en temps réel)
- https://github.com/rguyonneau/Self-Balancing-Robot-Python-Simulator : Une simulation python pour un robot "pendule inversé"

— Quelques vidéos
  - https://youtu.be/TZoq1NsQ8PM : une vidéo de démonstration du SLAM
  - https://youtu.be/G2n9_4nhEaQ : une vidéo de démonstration du balancing bot (robot pendule inversé)

Table des figures

Auteurs et biographies

Rémy Guyonneau
— remy.guyonneau@univ-angers.fr
— Rémy Guyonneau, Polytech Angers, Bureau 312, 62 avenue Notre Dame du Lac, 49000 Angers, France
— Enseignant chercheur à Polytech Angers (l'école d'ingénieurs de l'Université d'Angers), il est notamment responsable d'un cours de robotique mobile avec les étudiants de 4ème année du cursus Systèmes Automatisés et Génie Informatique. Dans le cadre de sa recherche il s'intéresse aux problématiques de localisation et cartographie simultanées pour des robots mobiles.

Franck Mercier
— franck.mercier@univ-angers.fr
— Franck Mercier, Polytech Angers, 62 avenue Notre Dame du Lac, 49000 Angers, France
— Franck Mercier est Ingénieur de recherche au sein du LARIS [29] depuis 2014. De formation dans l'électronique, il à débuté sa carrière au CNRS dans la conception des contrôles commandes d'expérimentations de grandes envergures dédiés à la propulsion spatial. Par connivence avec son domaine d'expertise, il s'est rapidement adapté à la robotique mobile, pour participer dorénavant activement aux recherches menés au sein de l'équipe robotique. Franck Mercier enseigne par ailleurs l'électronique numérique à Polytech Angers

29. Laboratoire Angevin de Recherche en Ingénierie des Systèmes