Méthodologie de conception de système multi-robots : de la simulation à la démonstration

Pierre Kancir

THÈSE DE DOCTORAT DE

L'UNIVERSITE DE BRETAGNE SUD
COMUE **U**NIVERSITE **B**RETAGNE **L**OIRE

Ecole Doctorale N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Automatique, productique et robotique*
Par

« Pierre KANCIR »

« Méthodologie de conception de système multi-robots : de la simulation à la démonstration »

Thèse présentée et soutenue à LORIENT, le 11 Décembre 2018
Unité de recherche : Lab-STICC - UMR 6285
Thèse N° : 519

Rapporteurs avant soutenance :

Yannick Le Moullec, Professeur, Université de Technologie de Tallinn, Estonie

M. Sébastien Bilavarn, MC-HDR, Université de Nice, Laboratoire LEAT

Composition du jury :

Président :	Gilles Coppin, Professeur, IMT Atlantique, Lab-STICC, Brest
Examinateurs :	Eva Crück, Ingénieure de Recherche, DGA, Paris
Dir. de thèse :	Jean-Philippe Diguet, DR CNRS, Lab-STICC, Lorient
Co-dir. de thèse :	Marc Sevaux, Professeur, UBS, Lab-STICC, Lorient

Titre : Méthodologie de Conception de Système Multi-robots : de la Simulation à la Démonstration

Mot clés : Drone, MRS, Essaim, Simulation, Open Source

Resumé : Les systèmes multi-robots sont des systèmes complexes mais prometteurs dans de nombreux domaines, les nombreux travaux académiques dans ce domaine attestent de l'importance qu'ils auront dans le futur. Cependant, si ces promesses sont réelles, elles ne sont pas encore réalisées comme en témoigne le faible nombre de systèmes multi-robots utilisés dans l'industrie. Pourtant des solutions existent afin de permettre aux industriels et académiques de travailler ensemble à cette problématique. Nous proposons un état de l'art et les défis associés à la conception des systèmes multi-robots d'un point de vue académique et industriel. Nous présentons ensuite trois contributions pour la conception de ces systèmes : une réalisation d'un essaim hétérogène en tant que cas d'étude pratique afin de mettre en évidence les obstacles de conception. La modification d'un autopilote et d'un simulateur pour les rendre compatibles aux développements des systèmes multi-robots. La démonstration d'un outil d'évaluation sur la base des deux contributions précédentes. Enfin, nous concluons sur la portée de ces travaux et des perspectives à venir sur la base de l'open source.

Title : Multi-robot System Design Methodology : from Simulation to Demonstration

Keywords : Drone, MRS, Swarm, Simulation, Open Source

Abstract : Multi-robot systems are complex but promising systems in many fields, the number of academic works in this field underlines the importance they will have in the future. However, while these promises are real, they have not yet been realized, as evidenced by the small number of multi-robot systems used in the industry. However, solutions exist to enable industrialists and academics to work together on this issue. We propose a state of the art and challenges associated with the design of multi-robot systems from an academic and industrial point of view. We then present three contributions for the design of these systems : a realization of a heterogeneous swarm as a practical case study in order to highlight the design obstacles. The modification of an autopilot and a simulator to make them compatible with the development of multi-robot systems. Demonstration of an evaluation tool based on the two previous contributions. Finally, we conclude on the scope of this work and future perspectives based on open source.

Remerciements

Je voudrais tout d'abord remercier ceux qui m'ont encadré durant mes travaux de thèse : M. Jean-Philippe Diguet et M. Marc Sevaux.

Je tiens également à remercier les rapporteurs de cette thèse : M. Yannick Le Moullec, et M. Sébastien Bilavarn. Mes remerciements vont à Mme. Eva Crück et M. Gilles Coppin qui ont accepté de participer à ce jury.

Je souhaite également remercier M. Gilles Coppin pour sa collaboration à mes travaux sur le projet DAISIE.

Je remercie aussi les entreprises RetDTech France et Azurdrones pour leurs soutiens techniques, ainsi que la DGA pour sa patience et ses considérations vis-à-vis du déroulé de ma thèse.

J'adresse tous mes remerciements à l'équipe des développeurs d'ArduPilot en particulier M. Peter Barker et M. Randy Mackay pour leur expertise et l'aide qu'ils m'ont apporté durant les développements open source sur l'autopilote. Mais aussi à la communauté du projet qui a accueilli mes démonstrations avec enthousiasme et encouragement.

Enfin je remercie ma femme Amanda Abreu et ma famille pour leur soutien durant les difficultés que j'ai rencontrées.

Table des matières

Liste des tableaux

Table des figures

Glossaire

GLPv3 GNU General Public License version 3 : licence libre la plus utilisée. Elle se base sur la notion de copyleft qui permet d'utiliser, modifier, redistribuer librement les logiciels sous cette licence. Elle est compatible avec la vente de logiciel et de matériel. En effet, le principe de l'opensource n'impose aucunement la gratuité logicielle ou la redistribution systématique des modifications à la communauté.

GNSS Global Navigation Satellite System : sigle anglais désignant les systèmes de positionnement par satellites comme le GPS et GALILEO.

IHM Interface Homme Machine : sigle désignant les interfaces graphiques des logiciels.

Lipo Batterie Lithium Polymère. Ce type de batterie présente une plus grande densité énergétique que les batteries lithium-ion que l'on trouve couramment dans les batteries d'ordinateurs portables. Les Lipo sont donc un choix préférentiel pour les robots nécessitants des optimisations de poids comme les UAVs.

ROS Robot Operating System : intergiciel de référence pour les développements en Robotique.

RSSI Received Signal Strength Indication : sigle anglais désignant la mesure de la puissance en réception d'un signal reçu d'une antenne.

SAR Search And Rescue : sigle anglais désignant les opérations de recherche et sauvetage.

SITL Software In The Loop : simulateur du projet ArduPilot. C'est une simulation logicielle de l'environnement et du matériel.

UAV Unmanned Aerial Vehicle : sigle anglais désignant génériquement les robots mobiles aériens.

UGV Unmanned Ground Vehicle : sigle anglais désignant génériquement les robots mobiles terrestres.

Chapitre 1

Introduction

Commençons par un exemple parlant. En mars 2011, le Japon est touché par un important tremblement de terre qui déclenche un terrible tsunami. Malgré un bilan humain important, la communauté internationale se focalise sur le réacteur nucléaire de Fukushima car celui-ci est endommagé. À cause des radiations mortelles pour l'homme, la solution la plus évidente est de faire usage de la robotique pour contrôler l'état du réacteur. En dépit de dizaines d'années de recherche et développement en robotique, le réacteur devient un cimetière de robots, et la robotique manque de solutions déployables afin d'assister les opérateurs humains sur des solutions opérationnelles.

Ce manuscrit présente une analyse, d'un point de vue industriel, de la problématique de conception de solutions robotisées en groupe. En effet, les groupes de robots permettent de mettre en œuvre des méthodes efficaces de traitement distribué avec des propriétés particulièrement intéressantes de fiabilité par redondance et coopération, de coût réduit pour chaque entité et d'efficacité collective. Cependant, de par le nombre d'agents pouvant être utilisé, leurs conception et usage diffèrent des robots seuls, ce qui met en avant des problématiques à la fois pour leur conception, mais aussi pour l'évaluation de leurs performances. Le document suivant est donc restreint aux robots mobiles, en particulier les micros robots, qui sont des entités robotiques capables de se déplacer dans leur environnement et de tailles réduites. Cette définition englobe les robots roulants (Unmanned Ground Vehicles ou UGVs en anglais), volants (Unmanned Aerial Vehicles ou UAVs en anglais), navigants, etc., mais exclut les bras robots. La figure 1.1 présente divers robots mobiles.

Dans ce manuscrit, nous proposons un état de l'art et les défis associés à la conception des systèmes multi-robots. Nous présentons ensuite trois contributions pour la conception de ces systèmes : une réalisation d'un essaim hétérogène, la modification d'un autopilote et d'un simulateur pour les rendre compatibles aux développements des systèmes multi-robots et la démonstration d'un outil d'évaluation sur la base des deux contributions précédentes.

Le mémoire s'articule autour de quatre chapitres.

Le chapitre 1 présente un aperçu des systèmes multi-robots en explicitant les caractéristiques de ces systèmes et leurs applications possibles, puis une analyse de quelques contri-

FIGURE 1.1 – Exemple de différents micros robots. Copyright ArduPilot sous licence CC BY-SA 3.0

butions académiques est présentée afin de mettre en évidence les challenges de conception et d'expérimentation de ces systèmes. Enfin, une vision industrielle est présentée quant aux verrous d'exploitation des MRS.

Le chapitre 2 met en avant la difficulté de mise en oeuvre d'un système multi-robots au travers d'un projet de recherche multi partenaires académiques et industriels. Sur la base d'un état de l'art des outils disponibles, une méthodologie de conception est présentée afin de concevoir des MRS multivecteurs (roulant et volant), et un retour d'expérience met en évidence les principaux obstacles rencontrés.

Le chapitre 3 décrit les principales contributions qui ont été faites au niveau d'un *autopilote* et d'un simulateur associé afin de les rendre compatibles à une utilisation de systèmes multi-robots.

Le chapitre 4 propose un framework de mise en oeuvre et d'évaluation de MRS basé sur les retours des chapitres précédents. Diverses expérimentations sont présentées afin de montrer l'usage du framework.

Chapitre 2

Challenges des systèmes multi-robots

Les systèmes Multi-robots (Multi-robot systems ou MRS en anglais) sont des systèmes composés de plusieurs robots mobiles ou un large nombre de robots simples, par rapport à la complexité de la mission, est coordonné de façon à accomplir une tâche. Grâce à leur robustesse et modularité, ils peuvent réaliser des tâches complexes plus efficacement que des robots seuls [129]. Si les MRS sont étudiés intensivement aujourd'hui[127], [42] il y a encore que peu d'utilisation industrielle. Ce chapitre présente un aperçu des MRS, mettant en avant leurs avantages, leurs limites, les challenges associés à ces systèmes et une taxonomie. Il présente ensuite une analyse de quelques contributions académiques afin de souligner comment les MRS sont caractérisés et expérimentés et met en avant quels sont les principaux outils utilisés pour les travaux sur les MRS. Enfin, il conclut avec présentation des défis empêchant le déploiement de MRS dans plus d'applications industrielles.

2.1 Aperçu des MRS

Cette section présente un aperçu des MRS en mettant en avant leurs avantages, limites et principales applications actuels.

2.1.1 Présentation

Le terme multi-robot system est le terme générique pour un ensemble de robots évoluant dans le même environnement. Ce terme englobe donc différentes sous-catégories bien spécifiques. La section 2.2 détaillera les principales caractéristiques des MRS. Les types de MRS les plus courants sont :

Les essaims : ce sont des systèmes multi-robots décentralisés avec une communication explicite et une coordination dynamique souvent copiée sur la nature. Ils permettent avec un nombre d'unités simple d'effectuer des comportements collectifs complexes comme le flocking [106] (ou vol en nuée en français) présenté figure 2.1.

FIGURE 2.1 – Exemple de nuée d'oiseaux. Copyright Walter Baxter sous licence CC BY-SA 2.0

Les meutes : leur définition est similaire à celle des essaims, mais elles mettent en oeuvre un mécanisme d'élection de leader qui amène une prise de décision centralisée ou une détermination des rôles de chaque robot du groupe.

Les flottes de drones : c'est un MRS avec généralement des drones volants dont les propriétés ne sont pas définies par le nom. Ainsi, une flotte pourra être une meute ou un essaim. En France, le mot drone est généralement associé aux drones multi-rotors. L'article [53] présente un aperçu exhaustif des drones volants (exemple en figures 2.2 et 2.3) que nous ne redévelopperons pas ici. Cette appellation peut être trompeuse puisque le terme drone, en France, désigne en réalité un véhicule ou robot avec un certain degré d'autonomie (cf.2.2.8 pour la définition de l'autonomie). Ainsi, une flotte de drone pourra être constituée aussi bien de robots roulants (Unmanned Ground Vehicles ou UGVs en anglais) que volants (Unmanned Aerial Vehicles ou UAVs en anglais).

Leader-Follower ou leadeur-suiveur : ce système est un système complexe ou un nombre de suiveurs vont suivre un chef. Ils peuvent être implémentés de différente

FIGURE 2.2 – Exemple d'un quadrirotors DJI Phantom. Copyright Wikipedia sous licence CC BY-SA 3.0

FIGURE 2.3 – Exemple d'un octaquadrotors Skeyetech. Copyright AzurDrones

façon, à la fois avec des communications explicites ou implicites.

2.1.2 Avantages

Plusieurs études montrent que les MRS sont plus intéressants en théorie qu'un robot seul, et ce pour plusieurs raisons qui apparaissent comme des évidences dans [36], [18], [9] :

- Réduction des coûts en utilisant des robots plus simples, moins coûteux à construire ou utiliser qu'un robot classique. Par exemple, un robot seul, qui est capable de compléter une tâche par lui-même, doit être équipé de plusieurs capteurs pour atteindre son but. Mais plus il a de capteurs plus il est "efficace", mais aussi cher et complexe. À l'opposé, les MRS peuvent utiliser des robots plus simples, avec un seul capteur par exemple. Ainsi le coût global du MRS reste peu élevé par rapport à une solution avec un robot seul.

- Plus grande couverture d'espace. Bien que les robots seuls soient capables de bonnes performances sur certaines tâches, certaines missions leur sont impossibles pour des questions de complexité ou de performances comme la surveillance de grands espaces.

- Redondance. Les MRS permettent la mise en place d'usage non conventionnel des robots qui réduisent les erreurs (par la multiplication des sources d'information, comparaison des capteurs, etc.), les problèmes opérationnels (portée de communication et réserve d'énergie limitée, etc.), et les accidents de fonctionnement (pertes d'agent, pannes matérielles, etc.). En effet, en cas de perte de communication ou de faible batterie, un robot seul va, en général, arrêter les tâches en cours avant la fin de la mission afin de garantir sa sécurité et être "perdu". Dans la même situation, les MRS peuvent inclure ces problèmes dans la stratégie de la mission puisque d'autres robots restent actifs pour finir la mission.

- Possibilité d'action collaborative (Reconstruction 3D, recalibration des capteurs, etc.).

– Meilleure performance en théorie que les robots seuls sur certain type de mission (exploration, surveillance, etc.).

En raison de la complexité de ces systèmes et d'autres raisons qui seront expliquées plus tard, ces avantages, souvent présentés comme acquis, sont difficiles à démontrer. Et cela d'autant plus si l'avantage de coût est considéré. Néanmoins, grâce aux progrès technologiques des dernières années, les capacités des robots ont progressé et leur temps de développement a diminué significativement. Ces gains en capacité, coût, et temps de développement ont permis une allocation de temps plus importante sur les autres problèmes de la robotique comme le contrôle des MRS.

2.1.3 Limites

Si les attentes de ces systèmes sont nombreuses, leur utilisation est limitée par leur nature complexe. En effet, les MRS combinent les problèmes des robots seuls et ceux des groupes.

– **Coûts**. Si le coût des solutions MRS est souvent présenté comme un avantage dans les travaux académiques, le prix du groupe reste lié à celui de chaque unité. Mais, plus le robot est équipé, le plus son coût matériel et son coût d'utilisation est important. C'est pourquoi le coût de chaque unité a intérêt à rester aussi faible que possible afin de maximiser le rapport performance sur coût par rapport à une solution avec un robot seul. Cette contradiction sera développée plus en détail dans les chapitres suivants.

– **Gestion de l'énergie**. Les technologies de batteries embarquables actuelles ne permettent pas beaucoup d'autonomie, en particulier pour les drones volants. Ce facteur reste limitant pour les MRS même si des stratégies de gestion d'énergie de groupe peuvent être mises en place : recharge de certains robots pendant que d'autres continuent la mission, porteurs de batteries, etc.

– **Communication**. Dans les MRS, les robots peuvent être amenés à communiquer entre eux ou avec des systèmes extérieurs. En raison du nombre de communications possible, les MRS communicants sont sujets aux problématiques de saturation des communications, du partage des fréquences et bande passante, etc.

– **Détection et évitement**. Il n'existe actuellement aucun capteur qui soit capable de fonctionner dans toutes les situations. De plus, le nombre de robots dans les MRS devient un frein à leur usage puisque chaque unité du groupe devient un nouvel obstacle dynamique.

– Gestion du groupe. Il est difficile de créer des modèles de comportement de groupe fiables en tenant compte des comportements et caractéristiques de chaque membre du groupe. C'est pourquoi il est nécessaire de mettre en place une méthode de

coordination, souvent complexe, entre les robots afin de maximiser les performances et réussir la mission.

– **Interface avec l'opérateur**. Deux problèmes sont à considérer. Le premier est l'utilisation des MRS par les utilisateurs, le second est l'interaction du groupe avec les humains pendant une mission. À cause de la difficulté de compréhension des mécanismes d'interactions inter-agent et du nombre des données renvoyés par le système, le contrôle des MRS est une tâche difficile. De plus, les opérateurs humains doivent pouvoir être considérés dans les stratégies de contrôle des MRS afin de permettre des tâches collaboratives ou non. Ces deux aspects ne sont actuellement que faiblement étudiés, même si la gestion des données des MRS présente un verrou technologique important pour la maintenance et l'amélioration de ces systèmes.

2.1.4 Applications principales

Les principales applications actuelles des MRS sont liées aux domaines ayant des forts d'intérêt commercial à court terme. La liste suivante présente quelques exemples applications intensivement étudiées qui commencent à avoir de réels usages industriels :

– L'exploration et la cartographie de zone. C'est le domaine qui a le plus d'activités de recherche et d'applications industrielles. Grâce au nombre d'agents et la redondance permise par le groupe, les MRS sont les meilleurs candidats pour ces tâches e.g [29], [24], [45], [105], [56].

– Surveillance. L'utilisation de robots pour des patrouilles automatiques et non répétitives est particulièrement attendue par l'industrie de surveillance et les forces armée, e.g [23], [60], [100], [96], [97], [70].

– La recherche et le sauvetage de personne en zone accidentée ou dangereuse pour l'homme, e.g [27], [56], [74], [73], [71], [34].

– Le nettoyage industriel ou de zones polluées et les systèmes de maintenance, e.g [130], [1].

– La logistique, e.g [125], [41], [3].

– Le déploiement d'antennes relais, e.g [84].

– Le spectacle :[94], [59].

2.2 Définition des MRS : Taxonomie

2.2.1 Introduction

Une des principales difficultés du design de MRS est le nombre de possibilités envisageables et leurs complexités. C'est pourquoi il est nécessaire de comprendre la majorité

des configurations du système et pour cela la définition d'une taxonomie des MRS est essentielle. Plusieurs publications ont déjà traité du sujet, et notamment les suivantes :

- Dans [35] les auteurs ont abordé la question par une classification des groupes multi-robots qui doit permettre de trier et comparer les différents groupes de robots selon plusieurs critères : les communications et leurs caractéristiques, les capacités informatiques des robots, le nombre d'agents, la composition du système (homogène ou hétérogène) et la reconfigurabilité du système. Cette taxonomie est très axée sur les caractéristiques du MRS, cependant elle ne permet pas de classer clairement les différents domaines d'application des MRS et différentes approches associés aux MRS : swarm, colonie de fourmis, meute, etc.

- [18] se sont basés sur cinq axes de recherches pour la coopération dans les groupes de robots : l'architecture du groupe, les conflits de ressources, l'origine de la coopération, l'apprentissage du groupe vis-à-vis de la résolution des tâches et les problèmes géométriques liés aux robots (recherche de chemin par exemple). Leur publication présente aussi un aperçu des avancées des recherches sur la robotique mobile coopérative jusqu'à l'année 1995. Leurs travaux posent une base de réflexion sur les configurations possibles des MRS, mais les limites technologiques de l'époque jouent clairement un rôle sur les manques qu'ils trouvent dans les recherches sur le domaine.

- Dans [40] est présentée une taxonomie axée sur les mécanismes de coordination du MRS basés sur les travaux réalisés jusqu'au début des années 2000 dans le domaine de la coopération et la coordination. Ils utilisent une approche ascendante et descendante (top-down) pour définir les différents niveaux de représentation de la structure du système. Ces niveaux sont au nombre de 4 : le niveau de coopération, le niveau de perception du groupe, le niveau de coordination et le niveau d'organisation.

Ces taxonomies de référence permettent de définir des MRS selon certains critères. Dans le but de pouvoir comparer les MRS, il est important de pouvoir définir une taxonomie qui englobe l'intégralité du MRS sans entrer dans les spécifications techniques : communication pour Dudek and Jenkin, Michael, Milios, coordination pour Farinelli et al., coopération pour Cao et al.. La taxonomie suivante présente donc les caractéristiques générales des MRS qui doivent permettre d'établir une base de comparaison entre les différents MRS.

2.2.2 Simple vs multi-robots

En admettant qu'on puisse réaliser un super robot fonctionnel, comme Atlas [14] par exemple, les tâches requérant une distribution spatiale ne sont pas réalisables par celui-ci. Un exemple simple est la surveillance d'un bâtiment. La surveillance simultanée de deux entrées opposées ne pourra se faire que par l'utilisation d'au moins deux robots. Un robot seul est constitué de plusieurs capteurs et d'un système de contrôle qui lui permet de résoudre certaines tâches. Cependant, la complexité des tâches à résoudre peut-être

trop importante pour un robot seul. Par exemple l'exploration d'une zone donnée en temps limité. Les MRS sont constitués de plusieurs agents qui peuvent être les mêmes ou non. Leurs usages présentent plusieurs avantages vis-à-vis de l'usage d'un seul robot, notamment du fait de la présence de plusieurs agents. Ces avantages ont déjà été listés en Section 2.1.2 Les MRS se décomposent en deux groupes. Ils peuvent être homogènes, c'est-à-dire que l'ensemble des robots les composant ont des caractéristiques techniques identiques. Ces systèmes sont les plus étudiés dans la littérature, car la complexité de design du MRS est réduite. En effet, la gestion des déplacements et la distribution des tâches aux différents agents peuvent être identiques pour chacun puisqu'ils présentent les mêmes caractéristiques [29], [33], [20]. Ils peuvent aussi être hétérogènes, c'est-à-dire que les capacités des robots ne sont pas nécessairement les mêmes. Ainsi un MRS hétérogène pourra présenter des UAVs et UGVs par exemple. Cette hétérogénéité apporte une complexité supplémentaire pour son utilisation, mais permet aussi l'ajout de nouvelles compétences aux MRS. L'utilisation d'UAV comme relais de communication pour les UGVs permet des zones de couverture plus grande par exemple [56], [60], [71].

2.2.3 Comportement collectif

Comme tous les groupes travaillant sur une même tâche, les MRS se doivent de posséder un comportement collectif. Ce comportement est caractérisé par l'ensemble des actions et réactions du MRS en réponse à une situation donnée. À l'instar de l'homme, le comportement collectif de MRS peut être coopératif ou compétitif. Le comportement coopératif est caractérisé par une situation ou les différents agents vont interagir afin de résoudre une tâche ou une situation. Ce comportement vise en général à augmenter les capacités des robots seuls avec l'aide d'autre robot. La littérature sur les MRS est principalement pourvue de solutions présentant des comportements collectifs notamment sur les questions de localisation des MRS, où l'usage de plusieurs drones va permettre à certains de guider les autres [44], [33], ou d'exploration ou la répartition des tâches doit être faite de manière à ce que les robots repassent le moins possible dans des zones déjà explorées [60], [45], [70]. Le comportement compétitif est l'opposé du comportement coopératif, pour répondre à une tâche ou une situation, les robots vont agir de manière à ce que leur gain soit maximal. Cela peut se traduire par une consommation ou un déplacement minimal par exemple. Un bon exemple de comportement compétitif est celui des robots utilisés lors de la Robocup [66], [107]. Cependant, l'aspect coopératif met en avant la difficulté de convergence vers un consensus pour la résolution d'une tâche. Un exemple très classique de la littérature est le franchissement d'une porte par deux robots simultanément. Si les deux robots tentent de traverser en même temps, une collision risque de se produire. Or les robots sont en général programmés pour effectuer des actions ne devant pas toucher à leur intégrité intentionnellement. L'aspect collaboratif nécessite donc de la coordination, et généralement des communications pour faciliter celle-ci. Les comportements collectifs mettent en avant d'autres problématiques. Du fait de la présence de plusieurs robots sur une zone un problème de partage de ressource est inévitable. Le problème de par-

tage de ressource pour les MRS peut se caractériser de plusieurs façons. Le problème le plus fréquent, notamment avec un nombre important d'agents dans le MRS, est celui des communications. Les communications sont un aspect important des MRS puisqu'elles permettent un partage d'information directe entre les robots. Les limitations technologiques (bande passante, portée, etc.) ou environnementales (perturbation, nombre de robots, etc.) influent sur les capacités communicatives des robots et donc sur les performances du MRS. La deuxième ressource ayant le plus de problèmes de partage est l'espace de travail. En effet, la taille des robots et leur nombre vont influer sur les possibilités de déplacement des agents sans impliquer de collisions, de congestions, ou de deadlocks (lorsque que deux robots s'attendent mutuellement). D'autres problèmes existent comme la répartition de l'énergie disponible pour le MRS, les phénomènes d'interférences des différents capteurs, etc. Ce problème intrinsèque de partage de ressource met en avant une nécessité d'avoir une coordination des robots.

2.2.4 Coordination

La gestion de la coordination représente une tâche centrale des MRS, la qualité de la coordination influe directement sur les performances globales du système. Cette coordination peut être définie de deux façons

Statique : (offline) De façon statique ou hors-ligne, c'est-à-dire que les règles de coordination ont été prédéfinies avant de réaliser une tâche. Un exemple de règle simple est : les robots doivent rester à une distance de sécurité les uns par rapport aux autres. L'utilisation de coordination statique permet de définir des règles pour des tâches complexes, mais implique une latence dans le contrôle des robots puisqu'il y a besoin d'établir la coordination [28], [100].

Dynamique (online, réactif) La seconde manière de définir la coordination est de réaliser une coordination dynamique ou en ligne. Elle est basée sur l'analyse de la situation [60], [33][34]. Cette coordination peut elle-même être décomposée en deux sous-catégories. La coordination explicite : qui est défini par une communication intentionnelle et une synchronisation des comportements. Ce type de coordination est souvent associé à l'utilisation d'une méthode de communication explicite. La coordination implicite qui est émerge de la dynamique du groupe et de l'environnement afin de résoudre la tâche voulue. Contrairement à celle statique, la coordination dynamique présente une rapidité d'exécution, mais est difficile à mettre en place pour des tâches complexes. La nature des missions de MRS et leur environnement étant complexes, les deux méthodes de coordination sont généralement utilisées afin de répondre aux différentes tâches que devront résoudre les robots.

2.2.5 Communication

Bien que certains MRS n'aient pas besoin de communication pour fonctionner, par choix ou par limite technologiques (flotte de ROV par exemple [115]), la communication reste un élément central des MRS. C'est un mode d'interaction entre les robots qui leur permet d'échanger des informations comme leurs position, état, etc., mais aussi de mettre en place une coordination en partageant les intentions et objectifs courants des robots entre eux. La communication se distingue par deux catégories :

La communication explicite : Ce type de communication est défini par un échange direct d'informations. C'est le mode de communication prédominant dans la robotique, du fait des faibles coûts d'utilisation d'un module communicant par onde (WIFI, sonar, etc.). Il présente l'avantage d'obtenir les informations voulues sur demande [73], [71], [33]. La connexion au groupe peut être séparée en deux :

- Le groupe est *fortement connecté* : C'est-à-dire que le groupe travaille de façon qu'un ou des robots ne soient pas isolés des autres. En cas de perte de contact avec un robot, le groupe se réorganise. C'est le cas le plus courant dans les MRS ou la communication est explicite. En effet, le groupe va essayer de toujours garder le lien de communication avec tous les agents.

- Le groupe est *faiblement connecté* : C'est-à-dire que les agents peuvent travailler sans avoir la contrainte d'avoir toujours une connexion avec le reste du groupe.

La communication implicite : Ce type de communication peut être divisé en deux autres catégories : la communication implicite active et passive. Les actives sont définies par un échange d'information à travers l'environnement, avec la dépose d'une balise au sol par exemple. Ce mode de communication est principalement utilisé dans les MRS basés sur le biomimétisme comme les essaims avec l'usage de phéromones par exemple. La passive se base sur la perception des changements dans l'environnement par les capteurs du robot par exemple le placement des autres robots [56], [34], [103].

Les deux méthodes de communication sont aisées à mettre en œuvre de nos jours. Cependant, dans le contexte des MRS, l'augmentation du nombre de robots dans le MRS à un impact significatif sur le nombre d'informations échangées ce qui entraîne aussi bien une saturation des communications explicites (engorgement de bande passante, etc.) que l'implicite (saturation des marquages dans l'environnement, etc.) et qui conduit à une baisse des performances du MRS.

2.2.6 Planification

La planification pour la robotique est ce qui permet la construction d'une succession d'action devant mener à la réussite d'un objectif. Au niveau des MRS, cette tâche représente l'élément central du MRS. En effet, c'est l'action de planification qui va permettre

à chaque robot du groupe de se coordonner avec les autres afin d'accomplir la mission. La difficulté consiste à répartir X tâches sur Y robots en visant une minimisation de coût (qui dépendent de la mission voulue), avec X et Y des nombres entiers supérieurs ou égaux à un. Du fait de la complexité croissante avec le nombre de robots et l'hétérogénéité de ceux-ci, c'est un problème NP-difficile (non déterministe polynomial) pour les MRS. La tâche de planification doit donc trouver un compromis entre les performances et qualités des solutions trouvées et le temps de planification. La notion de temporalité est essentielle pour les MRS puisque leur environnement de déploiement est forcément de nature dynamique du fait des mouvements des différents agents. De nombreuses publications traitent déjà de la planification des tâches [48], [15], [50], un résumé des tâches essentielles à la compréhension des MRS est donné ici. Il peut être observé que les travaux actuels sont généralement divisés en deux sous-problèmes : la planification des tâches et la planification des mouvements. La planification des tâches est elle-même répartie en deux actions.

Décomposition des tâches : La décomposition des tâches qui a pour but de réduire une mission en un ensemble d'action simple pouvant être exécuté par un robot ou plusieurs robots [15, 118, 72]. La difficulté de la décomposition des tâches est, dans un premier temps, de définir clairement la tâche, le degré de décomposition voulu, et le résultat souhaité. La définition de la tâche est souvent complexe du fait de la nature dynamique de l'environnement des robots, des temps de réaction souhaités et de la façon dont les robots vont percevoir l'environnement. Le degré de décomposition des tâches dépend des caractéristiques et du degré d'autonomie des robots. En effet, plus le degré d'autonomie des robots va être élevé où leur équipement va leur permettre de réaliser des actions difficiles, moins la décomposition des tâches au niveau du MRS aura besoin d'être précise. Par exemple, dans le cas d'une mission d'exploration, un robot capable de se déplacer avec du SLAM (Simultaneous Localization And Mapping) pourra se contenter de tâches comme "explore la zone X", alors qu'un robot muni seulement d'un positionnement absolu aura besoin, en plus, d'un guidage jusqu'à la zone X et d'une définition de la zone X. La décomposition dépend aussi de la nature des tâches, selon celle-ci, un ou plusieurs agents peuvent lui être affectés.

Répartition des tâches : La deuxième action est la répartition des tâches aux différents robots. C'est un problème complexe qui est un problème d'affectation optimal bien étudié dans d'autres domaines scientifiques [50], [48], [93], [68]. La solution considérée influe directement sur les performances du MRS. Ce sujet est l'objet de nombreuses publications sur l'efficacité de différentes méthodes pour la répartition de tâches dans des missions robotiques [121]. Dans les cas des MRS, une distinction doit être faite entre les MRS homogènes et hétérogènes. En effet, l'hétérogénéité permet une répartition des tâches basées sur les caractéristiques individuelles des agents avec notamment des stratégies des spécialisations de certains robots.

Motion planning : La planification des mouvements est un grand sujet de recherche de la robotique. En effet, les mouvements des robots sont contraints par les caractéristiques de celui-ci, mais aussi par l'environnement et par la perception qu'a le robot de l'environnement. La planification des mouvements est donc une planification des tâches spécialisées seulement sur la gestion des mouvements, c'est-à-dire les déplacements d'un robot d'un point de l'espace à un autre. Dans les cas des MRS cette planification est complexifiée du faite la présence des autres robots. En effet, chaque robot est un obstacle dynamique pour les autres. L'espace de déploiement d'un MRS étant limité, chaque agent doit prendre en compte les actions des autres robots notamment à l'aide de la coordination. Les gestions des déplacements dans les MRS sont souvent héritées de celle des robots seuls à laquelle est ajoutée une notion de communication ou de coordination afin d'éviter des problématiques de partage de territoire entre différents agents. La gestion de la perception de l'environnement est un axe de recherche important qui influe directement sur la gestion des déplacements [113], [20], [47] .

2.2.7 Prise de décision

Dans le contexte robotique, la prise de décision est le processus qui amène à déterminer le plan d'action pour la tâche désirée. Cette prise décision peut être faite de trois manières : centralisée, décentralisée, hybride.

Centralisée : Dans un MRS, un agent ou une base est responsable de la gestion des ressources, du planning et de l'attribution des tâches entre les différents robots [105], [100], [74]. Cette gestion globale du groupe présente comme avantages d'être facilement mise en œuvre et produire une planification globalement optimale. Cependant du fait de la centralisation, elle souffre de certains désavantages :

- Elle repose essentiellement sur un mécanisme de communication explicite. Les problèmes liés aux communications entraînent donc une baisse importante des performances du MRS centralisé.
- Le nombre d'agents dans le MRS influe sur la charge de travail et de donnée que devra traiter l'unité centrale.
- Elle met en défaut une partie de l'avantage de redondance que possède le MRS. En effet, une défaillance de l'agent responsable de la prise de décision entraîne un arrêt de l'utilisation du MRS.

Décentralisée : De manière opposée, la prise de décision décentralisée ne nécessite pas d'unité coordinatrice centrale [60], [97], [34]. Chaque robot est responsable de son contrôle. Cette approche est plus complexe à mettre en œuvre et peut produire des moins optimaux que la solution centralisée du fait que la prise de décision est issue uniquement des informations que possède le robot et non plus de l'ensemble du groupe. Cependant, elle

présente comme avantage d'être moins dépendant des communications (il n'est plus nécessaire d'obtenir l'ensemble des informations du groupe pour avancer) et est plus robuste à la perte d'agents.

Hybride : L'architecture hybride ou hiérarchisée est une fusion des deux précédentes méthodes de prise de décision. En découpant le MRS en sous-groupe, de la même manière que des clusters en informatique, il est possible de créer des agents centraux locaux qui seront responsables de leur sous-groupe de son contrôle. Cette architecture reprend donc l'avantage de la concentration d'information en un agent de la méthode centralisé et la redondance de la méthode décentralisée [56].

2.2.8 Place de l'opérateur humain dans le MRS

Le problème des HRIs (Human Robot Interaction) peut être défini comme la compréhension et la mise en forme des interactions entre un ou des robots et un ou plusieurs opérateurs humains. Ces interactions sont toujours présentes, à plusieurs niveaux, ne serait-ce que pour des questions de sécurité dans l'usage des robots. L'opérateur humain peut aider le robot à résoudre des tâches complexes comme la planification de tâches, la reconnaissance d'objet. Actuellement, l'usage de robot mobile fait intervenir un ou plusieurs humains pour contrôler un robot. Cependant, l'usage de multiples robots met en avant de nouvelles contraintes à la fois pour le contrôle du groupe de robots, du retour d'information de ceux-ci et de l'interaction d'un MRS avec des agents qui ne sont pas robotiques [56], [105], [70], [43]. Le mode de téléopération classique n'est que très peu adapté aux groupes de robots, il faut donc donner d'autonomie aux robots et cela implique de définir le degré d'autonomie des robots. Selon Sheridan and Verplank, le niveau d'autonomie décrit jusqu'à quel degré le robot peut agir par lui-même. À cela s'ajoute une définition des différents niveaux d'autonomie possible [111].

1. Computer offers no assistance ; human decides everything
2. Computer offers a complete set of action alternatives
3. Computer narrows the selection down to a few choices
4. Computer suggests a single action
5. Computer executes that action if humanly approves
6. Computer allows the human limited time to veto before automatic execution
7. Computer executes automatically then necessarily informs the human
8. Computer informs human after automatic execution only if humanly asks
9. Computer informs human after automatic execution only if it decides to

10. Computer decides everything and acts autonomously, ignoring the human

Cette définition se révèle suffisamment précise pour décrire le niveau d'autonomie d'un robot du point de vue des interactions humains-robots indépendamment du type de robot.

La définition de l'autonomie des robots dans un MRS à une double importance. Elle va permettre de définir le niveau de contrôle des robots du groupe, mais aussi quelle est la place de l'homme dans le MRS. En effet, l'humain peut être intégré de deux façons dans le MRS, soit en tant que contrôleur en gérant le robot et leurs informations, soit en tant qu'agent en réalisant une mission avec le MRS. Ce cas est particulièrement présent dans les opérations de SAR. Cependant, si l'humain peut apporter de l'intelligence décisionnelle aux robots, son efficacité repose souvent sur l'expérience accumulée et non pas sur un processus rationnel ou mathématique. L'ajout de l'opérateur humain donc n'est pas synonyme de performance pour l'essaim du fait des erreurs pouvant être commises ou du manque d'intégration de la tactique de l'opérateur dans la stratégie choisie par l'essaim.

Tout en laissant libre cours à l'essaim, il peut être nécessaire d'avoir accès aux données de l'essaim en temps réel (affichage visuel, etc.) ou de pouvoir reprendre la main sur un robot pour l'assister ou confirmer un objectif. La place de l'opérateur humain dans la chaîne de supervision de l'essaim peut être clairement définie par les niveaux définis précédemment. Si le traitement des informations d'un robot peut être réalisé par un ou plusieurs opérateurs, le nombre de données envoyé par un MRS est trop important. Il est donc nécessaire d'avoir recours à de nouvelle interface homme-machine capable de synthétiser les informations et les intentions du MRS de façon exploitable par un opérateur humain. Ces interfaces présentent l'avantage de pouvoir traiter l'information pendant la mission au lieu d'analyser les informations enregistrées dans les robots après la mission.

2.3 Recherche sur les MRS

Les premiers MRS étaient basés sur des modèles bio-inspirés (essaim [13], meute, etc.), car ils étaient plus faciles à comprendre et plus simples à imiter. Cependant, les technologies de cette période se montraient insuffisantes pour répondre aux besoins complexes des MRS, notamment en matière de communication à distance, d'autonomie d'énergie, et de perception de l'environnement. Grâce aux progrès récents dans ces domaines, de nombreux travaux ont pu approfondir le sujet.

Les MRS étant des systèmes complexes et difficiles à implanter, l'analyse de leur performance est compliquée [42]. Afin de mettre en avant les principaux challenges et les verrous technologiques actuels des MRS, la section suivante propose donc une analyse globale de travaux académique sur les MRS, insistant sur trois points : i) la caractérisation des solutions MRS présentées, ii) l'analyse et la comparaison des résultats présentés, et iii) l'analyse des moyens de test des solutions en simulation et démonstration.

2.3.1 Caractérisation

Une des principales difficultés de la caractérisation des MRS est le nombre de possibilités et leur complexité. Les taxonomies de références définissent les MRS par rapport à certains critères : la communication pour Dudek and Jenkin, Michael, Milios, la coordination pour Farinelli et al. et coopération pour Cao et al.. Cependant, pour comprendre les propriétés d'une solution et avancer vers un usage industriel, il est plus important de définir certaines caractéristiques générales liées aux technologies robotiques. C'est pourquoi nous avons choisi d'utiliser les limitations présentées en 2.1.3 pour montrer comment elles sont adressées dans la littérature puisque que ces limitations sont liées à des solutions technologiques réelles. Le tableau 2.1 présente un ensemble de publications, qui sont représentatives des travaux du domaine, et analyse les caractéristiques de chacune des solutions présentées avec les limites de 2.1.3.

Tableau 2.1 – Principales limites des MRS présentées

Références	Coût	Gestion de l'énergie	Communication	Détection et évitement	Gestion du groupe	IHM
[56]	Yes	NS	Oui	Oui	Non	Oui
[44]	NS	NS	Non	Oui	Oui	NS
[20]	NS	NS	NS	Oui	NS	NS
[71]	NS	Oui	Oui	NS	NS	NS
[29]	NS	NS	Non	Oui	Non	Oui
[45]	NS	NS	Oui	Oui	Oui	Oui
[33]	NS	NS	Oui	Oui	Oui	NS
[60]	NS	NS	Oui	NS	Oui	NS
[34]	Non	NS	Oui	Oui	Oui	NS
[24]	NS	NS	NS	Oui	Oui	NS
[96]	NS	NS	NS	NS	Oui	NS
[103]	NS	NS	Oui	NS	NS	NS
[73]	NS	NS	NS	Oui	Oui	Non
[74]	NS	NS	Oui	NS	Non	NS
[105]	NS	NS	Oui	Oui	Non	Oui
[93]	NS	NS	NS	Oui	Non	NS
[97]	NS	NS	Oui	Oui	NS	NS
[100]	NS	NS	NS	NS	Non	NS
[32]	Oui	Non	Oui	None	Oui	NS

Dans un premier temps, il peut être remarqué que certains aspects ont été notés comme NS (Not Stated, non présenté). Cela est dû à un manque d'information dans les publications présentant quel choix technologique a été fait. Il est problématique pour la bonne compréhension, la répétabilité des résultats et la comparaison avec d'autres travaux que des choix importants dans le design des systèmes aient été oubliés. En effet, puisque toutes

les caractéristiques et les hypothèses faites n'ont pas été clairement précisées, reconstruire un système multi-robots sur ces travaux sera plus difficile. C'est pourquoi, à cause de ces manques d'informations, mais aussi du manque de relation entre les différents systèmes proposés, il n'est pas nécessaire d'entrer plus en profondeur des détails des publications pour notre analyse.

Si l'analyse du coût de la solution est généralement évitée, il n'en reste pas moins un des principaux avantages des MRS que les auteurs mettent en avant. Une solution de MRS fonctionnant avec un système de capture de mouvement et une autre reposant sur un guidage par GPS ne présentent évidemment pas le même coût ni les mêmes performances. Ainsi, une présentation du coût simpliste des ressources nécessaires au fonctionnement de la solution serait suffisante pour établir une classification et réellement pouvoir juger de l'avantage du MRS. De la même manière, l'analyse des aspects énergétiques est peu ou ne pas du tout étudier, alors que c'est un aspect critique de la robotique et spécialement dans des missions de patrouille. La plupart des métriques de performance analysées sont seulement tirées des objectifs de mission : temps d'exploration, taux de recouvrement, etc., mais ne sont pas liées aux caractéristiques du MRS.

Dans le cas des communications, de l'évitement, et de la gestion du groupe, le fait de ne pas avoir les choix technologiques rend difficiles la compréhension du système et la détermination du degré d'autonomie [111] atteint par la solution. Cela amène aussi des contradictions de démonstration ou des algorithmes de gestion de groupe présenté comme décentralisés sont utilisés sur un serveur et donc en réalité centralisés, et ne reflètent pas les difficultés liées à la physique des robots. Dans d'autres cas, des unités sont présentées comme autonomes, mais ont leurs positions fournies par des systèmes de capture de mouvements [83] [110]. Ces choix technologiques doivent être clairement présentés, car ils ont un impact très important sur les autres caractéristiques du système. En effet, fournir la position où les ordres de trajectoire aux robots vont influencer leur comportement puisque l'usage de communication se traduit par un usage d'énergie plus important, des problèmes de timing ou de bande passante, etc. Ainsi un système avec un moyen de localisation implémenté dans chaque unité et un autre avec localisation extérieure vont produire des résultats différents.

Enfin, la gestion du retour de données et les interactions humain-machine sont généralement ignorées alors que c'est un aspect essentiel au déploiement industriel.

Pour conclure, nous pouvons remarquer que les travaux sur les MRS donnent seulement un aperçu partiel des caractéristiques des solutions proposées puisque les choix technologiques essentiels ne sont pas considérés. Comme chaque sous-problème des MRS est complexe en soit, mais aussi lié aux autres, cette caractérisation partielle est un facteur limitant pour la poursuite des travaux et la répétabilité des travaux et potentielles avancées. Il est un fait que le but des publications n'est pas d'être exhaustif sur les caractéristiques de robots, mais ce manque d'informations simples est un frein à la mise en œuvre concrète et donc à aux progrès dans le domaine et au final à leur éventuelle adoption . L'analyse que nous avons donnée ici peut être vue comme naïve, mais elle met en avant un grand défaut des travaux en robotique en général qui est l'oubli de ce qu'est un

robot. En effet, un robot reste une entité physique réelle avec des caractéristiques (poids, taille, etc.) qui ne peuvent être réduites à une variable dans un algorithme, ou un point sur un graphe. Il ne serait pas réaliste que toutes les publications soient conclues par une expérimentation sur plateforme réelle puisque différents niveaux de recherche sont nécessaires et tous ne nécessitent pas une mise en œuvre concrète. Mais il parait essentiel que les démonstrations existent afin de faire vivre les contributions académiques. Cela nous amène à analyser comment les résultats des publications sont analysés.

2.3.2 Études des résultats

En définissant un système multi-robots comme un ensemble de robots capable d'accomplir une tâche dans un environnement, les publications du tableau 2.2 ont été analysées pour montrer comment les auteurs ont évalué leurs résultats et les limites associés.

Tableau 2.2 – Analyse des résultats des publications

Références	Variation du nombre de robots	Variation type de mission	Variation type de terrain	Variation type de robot	Comparaison avec autres solutions
[56]	Non	Oui	Non	Oui	Non
[20]	Oui	Non	Non	Non	Non
[44]	Oui	Non	Oui	Non	Non
[71]	Non	Non	Non	Oui	Non
[45]	Oui	Non	Non	Non	Non
[29]	Non	Non	Non	Non	Non
[33]	Oui	Non	Non	Non	Non
[60]	Oui	Non	Non	Oui	Non
[96]	Oui	Non	Non	Non	Oui
[24]	Oui	Non	Non	Non	Oui
[34]	Oui	Non	Non	Oui	Oui
[73]	Non	Oui	Non	Non	Non
[103]	Non	Non	Non	Non	Non
[74]	Non	Non	Non	Non	Oui
[105]	Non	Non	Oui	Non	Non
[93]	Oui	Non	Non	Non	Oui
[97]	Oui	Non	Non	Non	Oui
[100]	Oui	Non	Non	Non	Oui
[32]	Oui	Non	Oui	Non	Non

Un des grands challenges actuels des MRS est la validation réelle des résultats. Il peut être observé dans les travaux précédents que :

- La plupart des solutions sont seulement testées sur une seule plateforme avec un seul jeu de capteurs et d'actionneurs. De plus, il y a peu de MRS hétérogènes.

- Un seul scénario est envisagé pour la validation.

- Les solutions sont, pour la plupart, testées en intérieur et sans obstacle dynamique.

- Il y a peu de comparaison avec d'autres algorithmes ou solutions pour le même type de mission ou d'action.

De plus, deux importants manques peuvent être soulignés. Premièrement, il n'y a pas d'analyse globale des systèmes, cela signifie que seules les métriques associées à la mission sont présentées sans analyse par rapport à la performance globale du groupe en temps que MRS. En effet, l'analyse de performance la plus utilisée est la variation du nombre de robots dans le groupe. Cependant, la seule analyse de l'influence du nombre de robots dans le groupe est incomplète, car elle adresse seulement un cas particulier du système présenté. De plus, la comparaison des performances d'un robot seul par rapport à celui d'un groupe du même robot n'est pas suffisante pour montrer l'avantage du MRS. En effet, le robot seul présente l'avantage d'être plus simple et moins cher que le groupe. Comme le coût des MRS est systématiquement présenté comme un avantage, il devrait être démontré en même temps que les performances de la solution avec de nouvelles métriques génériques comme le ratio performance sur coût. Ce type de métriques permettrait de faciliter la comparaison de différentes solutions, mais aussi de présenter des informations plus générales sur l'utilité de la solution. Par exemple, prenons un robot de 2k€ capable d'effectuer la cartographie d'une zone en 100 secondes et un MRS de 10 robots capable de la même tâche en 80 secondes. Pour un gain de performance de 20% en utilisant le groupe, un investissement de 18k€ supplémentaire est nécessaire. Si on considère uniquement le gain de performance de mission, le groupe est plus performant, cependant, si l'on considère le MRS globalement, il devient plus difficile de juger de l'avantage du groupe puisqu'à prix égal de nombreux robots peuvent opérer seuls et avoir les mêmes bénéfices que le groupe, mais sans la complexité. D'un autre côté, si l'on compare les performances du groupe avec un robot de 20k€, l'avantage du MRS devient évident. De la même manière, les comparaisons d'autonomie (en énergie) doivent aussi être faites avec d'autres types de robots pour montrer des gains réels.

Deuxièmement, les comparaisons avec d'autres travaux du domaine ne sont pas systématiques. Si certains domaines "classiques" comme l'exploration ou les algorithmes de contrôle centralisés représentent une part importante des travaux, ceux utilisant pleinement tous les avantages des groupes comme le contrôle hybride (mélangeant mécanisme de contrôle centralisé et décentralisé) ou les systèmes hétérogènes sont peu étudiés. La multiplication des travaux sur les mêmes problématiques et les différentes qualités des publications deviennent problématiques dans le sens qu'il est difficile de réaliser des comparaisons entre eux, mais aussi d'avoir une vision claire de la recherche dans le domaine afin de pouvoir choisir les meilleures solutions pour une situation donnée.

Nous sommes conscients du fait que la recherche dans les MRS est un domaine difficile et que de nombreux travaux ne restent que théoriques ou des preuves de concept et sont loin d'applications concrètes. Les manquements présentés mettent en avant la nécessité de revenir à la base de la robotique et proposer des analyses globales des solutions à l'aide de métriques globales dédiées aux MRS afin de faciliter les comparaisons entre les différentes solutions, mais aussi permettre une classification efficace des MRS. Si nous ne pouvons pas demander une analyse globale des performances et comparative dans chacune des publications du domaine, car elles doivent généralement rester focalisées sur une problématique, une formulation globale du problème associé au MRS et une explication des hypothèses posées sont nécessaires à la bonne compréhension.

Enfin, cette analyse globale ne sera possible que si des outils performants et plateformes robotiques sont disponibles. Les travaux de cette thèse ont été menés afin de contribuer la création de ces outils. Ces contributions seront détaillées dans les prochains chapitres.

2.3.3 Démonstration des résultats

La complexité de ces systèmes rend l'évaluation analytique des algorithmes quasiment infaisables. C'est pourquoi l'approche privilégiée est de réaliser une évaluation empirique à l'aide de simulateurs et d'expérimentations réelles. Le tableau 2.3 présente les simulateurs et les plateformes utilisés dans les publications précédemment mentionnés.

Dans le tableau 2.3, trois points peuvent être mis en avant. Premièrement, de nombreuses études ne présentent pas d'expérimentations réelles et restent sur des simulations seulement. Si le coût important du déploiement d'un MRS peut être mis en avant comme une excuse pour limiter les démonstrations réelles, c'est le même coût qui est aussi présenté comme un avantage de ces systèmes, créant ainsi une importante incohérence. Néanmoins, il reste plus simple, dans la majorité des cas, de faire des simulations si on prend en compte le coût et l'obsolescence du matériel. Les expérimentations réelles sont cependant nécessaires pour dépasser le "Simulation Gap" et tester les solutions face aux événements non attendus ou des pannes (problèmes de batteries, robot bloqué face à un mur, etc.).

Deuxièmement, on peut observer que la plupart des publications utilisent un simulateur différent ce qui empêche les répétitions comparaisons de résultats. De plus, l'utilisation des simulateurs soulève quelques questions :

- Quelles sont les limites de la simulation (conditions de départ et de fin, environnement, etc.) ?
- Quelles sont les limitations de la simulation (modèles, capteurs, etc.) ?
- Comment sont définis les robots dans le simulateur (simple point, corps rigide, etc.) ?
- Comment sont contrôlés les robots simulés (une variable dans l'algorithme, distance parcourue fixe à chaque pas de temps, etc.)

Dans de nombreux cas, ces simples questions ne trouvent pas de réponses alors qu'elles sont essentielles à la compréhension des relations entre les différentes parties du système utilisé.

Tableau 2.3 – Simulateurs et expérimentations

Références	Simulateur	Réalisme	Type de robot	Tests réels
[56]	Stage	Bon	Pioneer based	Oui
[44]	Move3D	Faible	MagellanPro	Non
[20]	NS	NS	Pioneer 3-DX	Oui
[71]	NS	NS	NS	NS
[29]	MRESIM	Moyen	NS	Non
[45]	Move3D	Faible	Khepera	Oui
[33]	ARGoS	Bon	marXbot	Oui
[60]	Stage	Bon	Erratic robot	Oui
[96]	Stage	Bon	NS	Non
[24]	NS	NS	NS	Non
[34]	ARGoS	Bon	Swarmoid	Oui
[103]	NS	NS	Khepera III	Oui
[73]	Stage	Bon	Roomba	Oui
[74]	Pas de simulation	Pas de simulation	AR drone	Oui
[105]	Stage	Bon	Pioneer	Oui
[93]	Stage	Bon	Turtlebot	Oui
[97]	Stage	bon	Pioneer	Oui
[100]	NS	NS	NS	Non
[32]	JBotEvolver	Moyen	bateaux	Oui

De plus, la répétabilité et la réutilisation des démonstrations faites sur des simulateurs développés par des laboratoires est impossibles si aucune information sur ceux-ci n'est donnée, et d'autant plus si ces simulateurs ne sont pas disponibles publiquement.

Le troisième problème est lié à l'utilisation de trop nombreuses plateformes. Il n'y a à ce jour aucune plateforme de référence robotique ni de logiciel standard dans ce domaine. C'est pourquoi chaque expérimentation est effectuée avec des équipements, matériels, et logiciels spécifiques rendant de futures comparaisons difficiles. De plus, les solutions robotiques actuelles sont chères, peu modulaires, et généralement uniquement capables de travailler dans un seul environnement ce qui est loin d'un réalisme industriel et réduit les capacités d'expérimentation. D'une part, nous pouvons aussi observé que de nombreuses démonstrations basées sur des micro-plateformes comme e-puck ([82]) ou Khepera III ([61]) sont limitées uniquement aux développements de certains types d'algorithmes puisque ces robots ne peuvent fonctionner que sur des petites surfaces d'environnement de laboratoire. Il est important de souligner que l'AR Drone ([91]), qui est une plateforme jouet détourné par le monde académique, a déclenché un saut technologique. En effet, son design a amené de nombreuses améliorations par rapport aux autres plateformes robotiques : bas prix (250€), technologies à l'état de l'art, etc., ce qui en a fait un choix préférentiel pour les études sur les drones aériens. Cependant, l'AR Drone n'a pas été conçu pour cet usage et manque donc de modularité afin de pouvoir recevoir de nouveaux capteurs ou logiciels. Comme pour l'e-Puck, il est aussi limité en capacité de calcul et de technologie de localisation. C'est ce qui amène des expérimentations d'algorithmes décentralisé, mais exécuté sur des ordinateurs déportés. Soit les processeurs embarqués ne peuvent offrir les performances souhaitées pour les algorithmes retenus ou proposés, ou alors leurs mises en œuvre sous une forme distribuée s'avèrent trop complexes en pratique, ou encore ceux-ci deviennent inefficaces en raison de l'usage intensif de réseaux de caméra, de captures de mouvement qui sont extrêmement chers et inutilisables en dehors d'un environnement de laboratoire. En raison de leurs caractéristiques, ces plateformes mettent en avant leurs défauts récurrents dans les travaux académiques qui sont le manque d'autonomie énergétique, de capacité de déplacement et de calcul.

D'autre part, nous pouvons aussi trouver dans la littérature des MRS avec des super robots, c'est à dire des robots équipés avec les capteurs et actionneurs les plus chers du marché comme des caméras stéréoscopiques HD, des radars longs portés, etc. Ce type de démonstration met à mal l'avantage de coût des solutions MRS face à un robot seul.

2.3.4 Conclusion

Ces trois problèmes soulèvent la question des difficultés de validation des résultats, mais aussi de la répétabilité et de comparaison. Si les modèles numérique et analytique sont nécessaires, les expérimentations réelles sont essentielles pour la validation des résultats en robotique. L'intérêt est double. D'une part, elles permettent la validation, ou non, du modèle de MRS concerné et permet d'améliorer les modèles de simulation afin de réduire le simulation gap. D'autre part, elles valident la qualité et les performances des

solutions proposées. La diversité dans les MRS est une bonne chose, mais la création de bases communes et de méthode d'analyse globale des solutions permettra plus de travaux compétitifs et d'objectifs clairs à dépasser, à la fois académiquement, mais aussi industriellement. Dans l'analyse des publications précédentes, nous avons pu mettre en avant des problématiques qui peuvent être résumées par la liste suivante :

- Les limites globales des MRS ne sont pas considérées : on reste sur un type de robot, une mission, un environnement.
- Les travaux couvrent seulement une partie des challenges des MRS et ignorent les autres sans donner d'explication sur les hypothèses posées.
- Il y a peu de comparaison entre les différents travaux ou solution MRS.
- De nombreuses simulations sont seulement faites sur des simulateurs limités.
- Il y a peu de validations expérimentales.
- Les plateformes robotiques actuelles sont peu adaptées aux caractéristiques des MRS.
- Il y a peu de travaux sur les MRS tenant compte de tous les avantages offerts (hétérogénéité, contrôle hybride, etc.)
- Il y a peu de travaux répétables.

2.4 Les défis de la conception entre simulation et réalité

Dans les sections précédentes, nous avons présenté ce que sont les MRS, comment les définir globalement, et mise en avant certaines problématiques issues du travail académique dans le domaine. Bien que les MRS soient prometteurs de par leurs capacités, l'industrie ne les utilise que pour un faible nombre d'applications. La section suivante développe les disparités entre les contraintes industrielles et les solutions académiques courantes.

2.4.1 Disparités entre solutions académiques et attentes industrielles

Si de nombreux travaux académiques touchent tous les domaines des MRS, leur présence dans l'industrie est plus mitigée. Excepté pour les tâches d'aides en logistique [3], les MRS restent encore à l'état d'expérimentations e.g. [1], [86]. En effet, par nature, les publications scientifiques sont principalement dans une démarche de questionnement plutôt que dans la mise en place de solution effective à une problématique. Pourtant, les

performances des MRS en font des outils plus intéressants industriellement parlant que les robots seuls. Plusieurs raisons peuvent expliquer ce manque d'utilisation.

Les travaux académiques du point de vue des industriels

Dans un premier temps, si on regarde les contributions académiques, nous pouvons observer qu'elles ne sont pas en phase avec les solutions technologiques récentes et encore moins avec les directions industrielles. En effet, dans la plupart des travaux avancés deux solutions existent pour réaliser des démonstrations : réutiliser du matériel déjà disponible, ou acheter du matériel sur dimensionné afin de parer à d'éventuelles faiblesses d'implémentations. Dans les deux cas, il en résulte des solutions chères qui ne prennent pas en compte les questions de viabilité dans le temps et traite seulement un seul aspect des MRS comme montré précédemment. Il est donc difficile des baser un travail industriel sur des travaux dont la portée n'est pas clairement définie ou analysée et qui restent loin des conditions d'utilisations industrielles. Il y a un décalage clair entre les travaux académiques et les réalités industrielles qui conduit à des dysfonctionnements. L'utilisation de flotte de robot comme relais de communication est un sujet bien étudié dans la littérature, cependant, l'arrêt récent du Projet Titan de Google, projet qui visait à utiliser une flotte de drones solaires pour amener internet est un bon exemple de ce décalage. Un autre point de considération est l'obsolescence des publications. Si on regarde les publications de l'état de l'art des MRS, on peut observer qu'elles commencent à être datées (> 5 ans). C'est problématique du fait que la plupart des outils et plateformes utilisés ne sont plus disponibles de nos jours et empêchent la reproduction des résultats et les améliorations. Enfin, un grand point bloquant le déploiement des MRS dans le domaine industriel est le manque de visibilité des différents travaux. En effet, il y a eu plusieurs milliers de publications sur les MRS portant sur leurs nombreux aspects. Si on ajoute à cela le manque de base de référence ou de comparaison des travaux portant sur les MRS, mais aussi le manque d'expérimentations réelles ou de partenariat laboratoire – industrie, il devient alors difficile pour des industriels (mais aussi les chercheurs) de trouver les travaux adéquats et vérifier leurs validités en vue de l'utilisation d'un MRS ou de reprise des travaux. Cette faible visibilité pose à la fois des problèmes pour l'innovation, mais aussi une perte de temps et d'argent puisque beaucoup de travaux académiques seront perdus, car non trouvés. Pire encore, les industriels préféreront redévelopper leurs solutions plutôt que de se baser sur des travaux publiés, mais dont les hypothèses n'auront pas été correctement énoncées et la portée vérifiée.

Les blocages industriels

Afin de comprendre cette inadéquation entre le travail académique et industriel, voyons plus précisément quelles sont les contraintes industrielles des MRS. Plutôt discrètes ces dernières années, les usages des robots autonomes apparaissent avec la mode portée par les drones volants. En effet, ils ont poussé le marché de la microélectronique à produire des capteurs de plus en plus performants pour des coûts de plus en plus faibles (une IMU

(Inertial Measurement Unit) ne coûte plus que quelques dollars). Cette baisse de prix combinée avec les avancées technologiques récentes (Transmission HD, Lidar, etc.) profite à tous les domaines de la robotique et donc aux MRS. Cependant, les robots actuels ne sont pas équipés pour répondre aux fonctionnements des MRS puisqu'ils manquent d'autonomie décisionnelle et ont peu d'interactions avec leurs environnements, que ce soit directement ou indirectement. Par exemple, les robots aspirateurs commencent tout juste fin 2017 à pouvoir se connecter aux box domotiques afin de savoir quand effectuer le nettoyage, etc. Il faut donc encore inventer des usages à la robotique et un travail industriel important afin de pouvoir faire une place pour les MRS dans l'industrie. Cependant, cela nécessitera une standardisation matérielle et logicielle que nous avons déjà signalé, mais aussi une phase de recherche académique qui nécessitera des outils performants et reconnus.

Nous pouvons aussi observer le manque de communication sur les travaux industriels dans les MRS, peu de Startup ou d'industriels communiquent des avancés dans ce domaine. Durant cette thèse, nous avons rencontré de nombreux industriels et laboratoires qui nous ont permis d'identifier deux causes à ce manque de communication. Premièrement, même si les robots autonomes ont un côté attractif, la robotique en général a plutôt une mauvaise réputation auprès des populations. Cette réputation s'est bâtie sur deux facteurs : l'appréhension des systèmes robotiques qui pourraient avoir un degré d'autonomie tel qu'ils deviendraient incontrôlables, et l'impact de la robotisation sur l'emploi. Le 20e siècle a été très marqué par l'arrivée des robots dans les usines qui a conduit au remplacement d'une partie de la force de travail humaine. Cette révolution robotique est encore très présente dans les esprits, et la proposition d'utiliser des robots sans forts arguments sur les avantages apportés est systématiquement perçue comme une menace pour l'emploi. Deuxièmement, les MRS représentent une nouvelle rupture avec à la fois les marchés du travail traditionnels et robotiques du fait de leurs promesses et avantages face aux outils précédents. Ainsi le manque de communication et d'intérêt industriel pour les MRS est aussi dû à des considérations économiques. Les grands acteurs industriels du marché ont investi beaucoup de temps et de ressources dans des produits traditionnels et ne sont pas en faveur de l'avènement de ces nouveaux systèmes qui remettent en cause les performances de leurs produits et donc leurs investissements.

Les défis industriels

Les difficultés pour arriver à avoir des MRS fonctionnels et déployés sont multiples : algorithmes non testés, peu de dialogue entre industriels et laboratoires, etc. À cela s'ajoutent les difficultés de construction des robots devant composer le système. Selon les caractéristiques des MRS, ils doivent être moins coûteux qu'un robot traditionnel tout en ayant suffisamment de capacité pour fonctionner en groupe. Si certaines expérimentations sont réalisées sur des robots réels (avec ou sans les problèmes expliqués précédemment), les milieux industriels ajoutent d'autres contraintes. Les paragraphes suivants présenteront les problématiques essentielles à la bonne compréhension de cette thèse en concentrant les principaux exemples sur le cœur des robots qui est leur carte électronique principale,

bien souvent un micro-ordinateur.

L'augmentation des travaux sur les MRS est liée aux progrès technologiques à la fois dans le matériel et les logiciels associés. La disponibilité croissante de nouveaux capteurs complexes comme les sonars, les Lidars, et les caméras, a favorisé le développement de plateforme robotique toujours plus performante. À cela s'ajoute, les progrès réalisés dans l'utilisation de ces matériels afin de permettre aux robots de réaliser des tâches simples, mais complexes comme l'évitement d'obstacles, la détection d'objet, etc. L'usage des multiples capteurs du robot pour la réalisation de tâche complexe à nécessité une augmentation de leur intelligence et donc de leur puissance de calcul. Si l'évolution des performances des systèmes embarqués à suivi celle de l'informatique classique, l'année 2010 représente un tournant pour les systèmes robotiques. En effet, elle correspond au début de la révolution des smartphones qui a amené une réduction importante des coûts de l'informatique embarquée nécessaires pour une nouvelle Robolution et le déploiement des MRS. Depuis 2010, chaque année voit l'apparition de nouveaux processeurs embarqués plus puissants pour des prix moindres que l'année précédente. Il devient actuellement aisé de trouver une carte embarquée avec un ratio performance/consommation satisfaisant pour moins de 100€.

Tableau 2.4 – Comparaison prix, puissant, consommation entre SOC ARM

Carte SOC	Dragonboard 410c	ODROID-C1+	Raspberry Pi 3
Processeur	Qualcomm Snapdragon 410 1.2GHz quad core	Amlogic S805 1.5Ghz quad core	Broadcom BCM2837 1.2GHz quad core
Consommation électrique	24 W	5W	5W
Prix	75$	35$	40$

Parallèlement à l'amélioration des processeurs, les technologies de télécommunication ont largement évolué depuis 2010. L'arrivée des standards 802.11n , 4G et Bluetooth 4.0, qui permettent des transmissions de données toujours plus importantes à des distances en augmentation, a permis le déploiement de nouveaux robots à moindre coût comme l'AR.Drone ou NAO[51]. Outre l'aspect coût, qui impacte largement la robotique, la démocratisation de ces technologies a permis des gains de temps considérables dans le développement des robots. En effet, le fait d'utiliser des micro-ordinateurs au lieu de micro contrôleurs permet un développement plus souple en termes de programmation et met à disposition des robots la plupart des logiciels et interfaces matériels grand public. Ce gain de temps sur le développement de la plateforme se traduit par une allocation de temps plus importante sur les autres problèmes de la robotique comme le contrôle des MRS.

Obsolescence HW/SW Le mode de fonctionnement des laboratoires est souvent un fonctionnement par projet. À ce titre, la construction des plateformes robotiques se fera

de façon à pouvoir démontrer la faisabilité du projet durant le temps de projet prévu. Les robots sont bien souvent équipés du meilleur matériel possible pour un gain de temps dans le déploiement du projet. Si en début de projet, le matériel semble surdimensionné, il est souvent dépassé en fin de projet. En effet, l'obsolescence des produits aujourd'hui est un vrai problème que ce soit en matière de logiciel ou de matériel : par exemple, la Raspberry Pi 1 sortie en 2012 est déjà remplacée par la Raspberry Pi 2 sorti en 2015 et suivi de la Raspberry Pi 3 en 2016 qui change complément d'architecture (armv7 vers armv8). Le monde académique ne tient pas compte de ces contraintes qui sont importantes pour le monde industriel qui doit assurer la pérennité dans le temps de son MRS, à la fois fonctionnellement et financièrement. Actuellement, la longévité moyenne d'un système industrielle est de 5 ans avec des systèmes plus spécifiques ayant des durées de vie de 10 ans ou plus. Cependant, même les plateformes ARM, qui dominent dans le monde embarqué, ont une durée de vie qui décroît du fait de la compétitivité du marché. Actuellement, certaines plateformes ne dépassent même pas les 3 ans de production [58]. Cette problématique est une contrainte majeure pour la robotique. En effet, un changement majeur de plateforme reste très coûteux, d'autant plus si les développements logiciels précédents ne peuvent pas être utilisés. Beaucoup de projets basés sur des cartes ARM très grand public comme la Raspberry Pi ou la Beaglebone black ne peuvent pas être considérés comme stables dans le temps. En effet, le choix d'acheter ce type de carte déjà assemblé à un faible coût présente le risque que la solution ne soit pas disponible sur le marché pendant 10 ans [55]. On ajoutera aussi le surdimensionnent du système avec de nombreux périphériques inutilisés. Le prix ne peut pas être le seul facteur de décision pour la construction d'un système autour d'un équipement. Des stratégies comme la standardisation des périphériques, le placement des interfaces, permettent cependant une portabilité dans le temps de la plateforme développé en changent une partie du système pour un équivalent plus récent. C'est le cas des cartes Raspberry Pi qui, à travers les différentes révisions, ont conservé le même form factor et connectique. Sans compter que la plus grande force de ce type de carte reste leur communauté bâtie sur la base de licence open source du matériel. En effet, les communautés d'utilisateurs sur ce type de système fournissent une grande partie du support technique. Ces communautés sont d'une grande utilité puisqu'elles permettent en plus de support technique, des remontées d'informations, d'usage et de retour d'expérience importants que les industrielles peinent à obtenir de façon traditionnelle. Cependant, ces communautés d'utilisateur ne sont que temporaires puisqu'elles s'orientent rapidement sur de meilleures cartes au fil du temps, rendant la maintenance du système plus complexe dans le temps si le système n'est pas pensé pour tenir compte cette obsolescence. Enfin, l'obsolescence logicielle est souvent associée à l'obsolescence matérielle. En effet, dans le mode de construction classique d'un robot, on fixe une plateforme matérielle et on développe les logiciels nécessaires au fonctionnement du robot. Si l'on change de plateforme, on redéveloppe un logiciel. La tendance actuelle est de bâtir des solutions robotiques sur des bases logicielles plus génériques et souvent open source. L'usage de cartes ARM, notamment, qui sont de réels micro-ordinateurs, permet de réaliser des robots plus simplement en basant les développements sur des briques logicielles déjà existantes sur

les ordinateurs ou les OS. Cependant, ce nouveau type de développement est lui aussi contraint par l'obsolescence. En effet, les distributions Linux qui composent la majorité des systèmes embarqués robotiques actuelles sont sujettes à peu d'évolutivité logicielle. Ce manque impose soit une non-mise à jour des logiciels avec les risques de pannes ou sécuritaires associés [123]. Dans son article Tashkinov présent en détails les nombreux problèmes logiciels associés aux distributions Linux. L'obsolescence logicielle est particulièrement contraignante de nos jours pour la robotique. En effet, l'absence de mise à jour de robot, en plus de considération sécuritaire, ne permet pas la possibilité d'améliorer ou rendre compatible le robot avec son nouvel environnement par l'ajout de nouveaux capteurs et/ou moyen de communication par exemple. L'usage de l'open source peut être une bonne réponse à la problématique de l'obsolescence. Néanmoins, l'usage de l'open source reste un risque puisque comme pour l'open hardware, il est dépendant la bonne santé du projet [38] et des personnes qui y contribuent. Malgré une popularité et un large usage de l'open source, ces projets souffrent du manque de contributions et de moyen qui leur est alloué [116]. Enfin, une dernière contrainte liée à l'obsolescence des MRS reste l'émergence des nouveaux marchés. En effet, nous sommes dans une ère de développement rapide des solutions informatiques et automatisées (un appui sur un bouton suffit à déclencher une multitude d'actions, les Google Cards permettant avec son smartphone de régler sa maison, etc.). Nous voyons aujourd'hui arriver l'internet des objets 10 ans à peine après les premiers smartphones. Cette croissance technologique pose de nouveaux problèmes de déploiement pour les MRS. En effet, les travaux sur les MRS ont été construits sur les bases de la robotique, qui consiste à utiliser uniquement les capteurs présents sur le robot. Cependant, la multiplication des capteurs intelligents dans l'environnement doit pouvoir permettre d'augmenter l'autonomie des solutions robotiques à condition que robot en tiennent compte et soit compatible. Cela demande une modularité logicielle et matérielle qui doit pouvoir être prise en compte dans le design du MRS afin d'assurer la pérennité dans le temps du groupe.

Fiabilité et coûts Une des grandes caractéristiques souvent mises en valeur des MRS est d'être composés de robots à plus faible coût qu'un robot seul, mais capable en groupe de performances supérieures. Une grande problématique est d'arriver à réduire les coûts de production de ces robots tout en gardant une fiabilité importante. Cette recherche de baisse des coûts amène à de nouveaux modes de conception. En effet, la conception et la fabrication de système électronique sont très coûteuses et demandent des investissements à long terme. De plus, la robotique est un domaine en pleine extension ou les technologies évoluent vite. Afin de pouvoir suivre les évolutions technologiques à moindres coûts, plusieurs industriels font le choix de réduire leurs coûts de conception par l'utilisation de ressource générique, modulaire et bien souvent open source. Ainsi les cartes ARM comme les Raspberry Pi se retrouvent souvent comme le cœur de solution robotique. Ces cartes présent l'avantage d'être directement assemblé avec une multitude de périphériques et disponible à des coûts faibles. Leurs caractéristiques techniques répondent bien à un usage robotique modulaire et générique. Cependant, elles présentent aussi de nombreux

défauts qui n'en font pas les choix parfaits pour la construction d'un robot. On peut remarquer aujourd'hui que le marché est envahi de cartes ARM ou de capteurs provenant la plupart du temps de Chine. La page Wikipédia *Comparison of single-board computers* [124] recense 113 cartes ARM, 47 sont issues de concepteurs chinois. Malgré un avantage certain en termes d'approvisionnement et de prix, le coût en test et en temps de développement sur ce matériel peut être plus élevé que sur des solutions développées en interne ou achetées auprès de grands acteurs ayant des contrôles qualité de niveaux plus élevés, mais avec un coût d'achat plus élevé aussi. Ce type de matériel présente des qualités et performances souvent exagérées ou non vérifiables, ou encore d'innombrables défauts (inversion de câble, etc.) dans le cas de matériel électronique. La question de la maintenance sur ces cartes est aussi discutable. Si beaucoup présentent des communautés d'utilisateurs, le support industriel est bien souvent absent. On pourra aussi noter un manque de suivis logiciel pour les cartes ARM qui sont souvent livrées avec des fichiers binaires compilés, qui ne permettent pas l'ajout de fonction logicielle non prévue initialement et sans documentations. Ou encore la violation des licences open source en vigueur par la non-mise en disposition des codes sources libres. Ces problèmes rejoignent le problème de l'obsolescence de ce matériel (HW/SW) qui ne permettront pas suivre les évolutions dans le temps comme l'ajout de nouveaux capteurs, d'amélioration des performances, etc. La conception des MRS nécessite des industriels des choix importants entre coût, fiabilité, et durée dans le temps, choix qui ne sont, généralement, pas pris en compte dans les milieux académiques. Ces choix reflètent pourtant un des aspects intéressant des MRS qui est l'utilisation des robots à moindres coûts ou à performances dégradés.

Protocoles Un autre point d'importance dans les déploiements des MRS est la multitude des protocoles et normes disponibles pour les communications. En effet, il existe de nombreuses normes wifi : 802.11 a/b/c/../ac, 3G ,4G pour les communications, sans parler des bus de terrain (CAN, I2C, UART, etc.) ou des protocoles logiciel (http, websocket, protobuf etc. . .).

Tableau 2.5 – Principaux protocoles utilisés dans les MRS selon le modèle OSI

Layer 7 Application Layer	DDS, SSH, RTSP, RPC, Modbus, http
Layer 5 Session Layer	RPC, SOCKS, SDP
Layer 4 Transport Layer	TCP, UDP, DHCP
Layer 3 Network Layer	IPv4/IPv6
Layer 2 Data Link Layer	IEEE 802.11, I²C, Ethernet, CAN
Layer 1 Physical Layer	Can bus, I²C, I²S, Ethernet physical layer, USB, IEEE 802.11

Même si l'on observe une certaine uniformisation notamment autour du wifi, TCP/IP, bus CAN, il est souvent difficile de faire interagir plusieurs systèmes entre eux ce qui ralentit d'autant plus la construction des systèmes robotiques et la mise en œuvre de MRS. À cela s'ajoutent des considérations de sécurité. En effet, en général les MRS utilisent des

moyens de communication et de par leur nature, ils utilisent d'autant plus de communication qu'il y de robot dans le groupe. L'usage important des communications dans le MRS augmente donc la vulnérabilité du groupe face à un piratage qui pourrait avoir des conséquences fâcheuses lors de certaines tâches. Les protocoles doivent donc être dimensionnés afin de pouvoir être utilisés par un groupe important de robot, sécurisé afin de limiter l'exposition du MRS au piratage et améliorable afin de pouvoir combler d'éventuel défaut ou problème de sécurité.

2.4.2 Verrous à l'exploitation réelle de MRS

Encore discret il y quelques années, les robots autonomes tendent à se déployer avec l'intérêt porté sur les drones. Avec, en 2015, plus de 1500 entreprises homologuées par la direction générale de l'Aviation civile (DGAC) [54] et autorisées à faire voler des drones à des fins professionnelles, les drones volants occupent une part importante de cette nouvelle révolution. Bien que quelques MRS soient déjà fonctionnels, la plupart restent à l'état de prototype ou d'algorithmes. Il est donc important d'identifier quels sont les verrous technologiques empêchant leur émergence.

Dans [90], les auteurs font référence aux challenges qui n'ont pas été adressés en 2013. En 2017, quatre d'entre eux sont encore peu ou pas abordés, en dépit des avancées faites dans les MRS.

1. Comment identifier et quantifier les caractéristiques et avantages fondamentaux des systèmes multi-robots ?

2. Comment rendre le contrôle des équipes multi-robots plus facile ?

3. Peut-on généraliser les démonstrations impliquant plus de 12 robots ?

4. Comment la complexité de la mission et l'environnement affectent le design des systèmes multi-robots ?

Le premier et le dernier point recoupent les limites que nous avons identifiées précédemment. Le second point est toujours problématique. En effet, comme préalablement présenté, la majorité des travaux du domaine se concentrent sur les méthodes algorithmiques de contrôle, mais pas de l'aspect de contrôle haut niveau avec des IHMs. C'est une problématique importante pour un usage courant des MRS puisque leur complexité nécessite des modes de contrôle et commande clairs. Enfin le dernier point de leur analyse, le nombre de robots dans les démonstrations, soulève deux problèmes que sont : le problème de dimensionnement en coût des solutions et la portée des démonstrations. En effet, en raison du coût des matériels engagé les laboratoires sont limités sur le nombre de robots qu'ils peuvent mettre en oeuvre, ce qui affecte les résultats et les types de démonstration expérimentale qu'ils peuvent mettre en oeuvre.

Dans [131], les auteurs ont listé les trois challenges suivants pour les essaims en 2008 :

1. Le design des algorithmes de contrôle des essaims : les agents de l'essaim et leurs comportements individuels doivent être conçus pour faire émerger un comportement de groupe. Il n'existe actuellement aucune méthode générique pour passer d'un comportement individuel à celui de groupe.

2. L'implémentation et le test de telles solutions nécessitent des infrastructures et des ressources que les laboratoires ne possèdent pas en général.

3. Les essaims sont par définition stochastiques (i.e la même cause n'induit pas le même effet)[12] et non linéaires, la création de modèles mathématiques pour l'analyse et la validation est donc difficile. Cependant, ces modèles sont nécessaires pour créer des méthodes sures et sécurisés pour l'utilisation d'essaims.

Les analyses de Sahin et Winfield paraissent encore d'actualité au vu du bilan porté en section 2.3 à l'exception du second point. Comme expliqué précédemment, les technologies ont considérablement évolué et avec elles leurs coûts. Le coût de création d'un essaim ou de MRS est maintenant réduit et accessible aux laboratoires sans demander des investissements prohibitifs. Il existe de plus de nombreux outils accélérant le développement des solutions robotiques comme ROS (Robot Operating System) [104], ArduPilot [6] et Gazebo [67]. Si les travaux actuels se concentrent toujours sur le nombre de robots dans les MRS où le succès de la mission, l'analyse des caractéristiques globales et fondamentales des MRS permet de juger de l'utilité du système dans une situation et un terrain donnés. Étant donné la complexité des robots, les groupes de robots et leurs environnements de déploiements ne permettent pas la mise en place d'évaluation purement mathématique des algorithmes mis en pratique. En effet, trop de paramètres sont à considérer comme le nombre de robots, l'homogénéité du groupe, le type de capteur utilisé, l'environnement, etc. Ces limitations ajoutées à celles de la section 2.3 mettent en évidence l'importance du problème de la comparaison entre les différentes solutions existantes, mais aussi les difficultés de compréhension de l'intérêt des MRS face aux robots seuls. En effet, il existe de nombreux travaux traitants des groupes de robots, il est donc difficile de choisir une solution pour une mission donnée sans avoir une vue globale des capacités, performances et du coût du système. Le coût de la solution devant être présenté afin de faciliter les comparaisons d'une part, et d'autre part de mettre en avant les avantages des MRS face aux solutions avec un seul robot. Autrement, le déploiement d'un robot sera toujours un choix plus simple. Les conclusions de [42] souligne aussi les mêmes challenges. Il est surprenant, dans un domaine actif comment les MRS, de voir qu'entre 2003 et 2017 ces problématiques sont toujours entièrement ouvertes. Nous croyons que les trois points suivants sont essentiels pour permettre l'émergence de plus de solutions de MRS dans le futur :

– Mise place d'une plateforme matérielle et logicielle de référence ouverte (Open Source et Open Hardware).

– Mise en place d'une méthode d'analyse globale et d'outils adaptés aux MRS.

– Déploiement d'une base de données de benchmark afin de pouvoir comparer les différents résultats entre eux.

Enfin, nous croyons que ces points ne sont pas réservés aux seuls travaux des laboratoires de recherche. Ils devront être le fruit d'une collaboration entre industriels et académiques afin de mutualiser à la fois le travail et les résultats. Pour cela l'open sourcing apparaît comme le candidat idéal. Ainsi les contributions de cette thèse ont porté sur ces points tout en restant libres afin de maximiser leurs portées.

2.4.3 Conclusion

Dans ce chapitre, nous avons présenté un aperçu des MRS et des travaux académiques dans ce domaine afin de comprendre quels sont les challenges et limites associés à ces systèmes. Il est évident que des progrès ont été faits sur un certain nombre de questions importantes comme la localisation, la cartographie et l'exploration. Ainsi, plusieurs aspects des MRS ont été intensivement étudiés et montrent des résultats importants. Cependant, d'autres aspects demeurent trop peu étudiés comme les interactions entre les MRS et les opérateurs, l'utilisation des solutions sur plusieurs scénarios, et les limites réelles dans des applications réelles. La caractérisation globale de ces systèmes (coût, autonomie, communication, etc.) est aussi un point généralement sous-estimé. À cause du manque d'informations fondamentales sur les solutions et l'absence de plateforme commune ou de référence, il est devenu difficile de comparer les solutions afin de définir laquelle est la plus appropriée pour chaque situation. Afin de permettre des analyses plus poussées des solutions et de simplifier le portage des résultats des travaux académiques dans des réalisations réelles, trois contributions ont été proposées. La suite de ce document présentera donc une proposition de plateforme, qui est utilisée dans le cadre d'une expérimentation réelle d'essaim, ainsi que le début de la création d'outils de dimensionnement, d'analyse et comparaison des MRS.

Chapitre 3

Création d'un essaim : de la simulation à l'évaluation

Les MRS sont complexes à modéliser à la fois analytiquement, mais aussi matériellement et logiciellement. L'analyse précédente a montré la multitude d'outils utilisés dans leur mise en place, mais aussi les limites des solutions et outils proposés. Pourtant, le monde robotique présente des outils puissants et reconnus académiquement comme industriellement. Ce chapitre fait un état de l'art des outils actuels pour la conception des MRS. Puis sur la base d'un projet de recherche, présente une méthodologie pour la construction d'un système multi-robots et une solution de plateforme bas coût et générique pour les MRS.

3.1 Outils pour la conception de MRS

3.1.1 Outils de simulation

Lorsqu'on fait un point sur les travaux réalisés sur les MRS, on peut remarquer qu'il existe une multitude d'outils de simulation. En effet, quasiment chaque laboratoire utilise son propre simulateur ou redéveloppe un outil de simulation pour démontrer la viabilité de sa solution [16], [29]. L'utilisation de tels simulateurs peut représenter un frein au développement des MRS du fait des trop grandes approximations qui sont permises dans les modèles des composants des robots du MRS voir 2.3.3. Beaucoup des publications liées aux MRS basés uniquement sur une simulation sont donc imparfaites. De plus, ce type de publication suit une trame bien connue : proposition d'un nouvel algorithme ou d'une amélioration de l'algorithme, test avec un simulateur, présentation des résultats, conclusion de la viabilité par rapport aux résultats obtenus. Cette trame met en avant d'importantes questions :

– Quel simulateur a été utilisé ? Est-ce un simulateur de référence en robotique (Gazebo, Stage) ou un simulateur développé pour l'algorithme ?

- Comment a été validée la viabilité du simulateur ? Quel est l'impact de la simulation sur les résultats ? Comment fonctionne le simulateur ? Avec un modèle de capteur ? Avec un modèle physique 3D ?

- Quels robots ont été utilisés dans le test ? Des robots réels (Pioneer, e-pucks) ou des robots virtuels ? Dans ce dernier cas, comment sont-ils définis ? Capteurs ? Actionneurs ? Taille ? Comment interagissent-ils avec les autres robots et l'environnement ?

- Comment est intégré le robot dans le simulateur ? Comment le robot est-il contrôlé par le simulateur ?

- Comment les limites du simulateur, de l'environnement, et du modèle de robot influent-elles sur les résultats ?

- Comment rejouer les résultats ? Quelles versions des simulateurs, modèle de robot ont été utilisés ?

Si les simulateurs robotiques modernes peuvent se montrer très performants, ils n'en sont pas moins dangereux. Deux aspects des simulateurs sont considérés comme dangereux. Le premier est le Reality Gap qui est le niveau de corrélation entre les résultats des simulations et ceux réels. Aussi puissant qu'ils soient la plupart des simulateurs pour la robotique ne sont pas capables de simuler tous les aspects d'un robot : électronique, mécanique, logiciel. Ainsi les résultats de simulation s'approcheront seulement au mieux de la réalité. Le second danger est la visualisation. En effet, le réalisme visuel (ombres, jeu de lumière, etc.) renforce l'illusion que ce qui est vu est réel et fonctionnel. Mais seule l'analyse des données brutes montre les résultats des tests et validations pendant une simulation et pas l'animation qui en a été faite.

Le simulateur parfait n'existe pas encore, mais il en existe certains qui sont capables d'approcher la réalité robotique actuelle afin de démontrer et comparer des travaux avec d'autres solutions existantes. Parmi les plus connus se trouvent :

- La suite Player/Stage [49] : C'est un simulateur robotique open source 2D. Il prend en charge nativement les systèmes multi-robots avec une large base de données de capteurs. La suite est aussi compatible avec le middleware ROS afin d'étendre ses capacités de simulation. Cependant, les modèles utilisés accusent leurs âges (pas de drone, pas de réseaux sans fil, etc.) et le projet n'est plus maintenu que pour maintenir la compatibilité avec ROS.

- Gazebo [67] : C'est un simulateur open source 3D, uniquement disponible sous Linux. C'est la référence des simulateurs robotiques open source 3D.

    - Capacité de simuler avec précision des robots complexes comme Atlas ou Valkyrie ou des drones comme le 3DR Iris.

  - Gazebo est flexible. Il est possible de simuler un robot complexe comme une nuée de drones par reconfiguration. À cela s'ajoute une multitude de plug-ins augmentant les possibilités déjà présentes (capteur, modèle physique, etc.). Cependant, la phase de configuration reste complexe et la puissance de calcul nécessaire à la simulation augmente grandement en fonction du nombre de robots.

  - Intégration avec ROS. Les robots peuvent être simulés avec le code réel qui leur sera embarqué.

  - Les simulations réalisées sur Gazebo avec ROS peuvent être rejouées grâce au logging.

  - Une forte communauté.

- Webots[25] : un simulateur 3D réaliste, mais payant qui permet des simulations multi-robots. C'est un simulateur multiplateforme Linux, Mac OSX, Windows. Il supporte de nombreux types de capteurs comme les caméras, ultrason, etc. Les modèles simulés peuvent avoir plusieurs attributs comme des masses, formes, etc., ce qui permet une simulation basée sur des modèles de corps dynamique en 3D. Cependant, ses performances se dégradent rapidement dans le cas où un nombre important d'agents serait déployé. Il comporte aussi une intégration de ROS

- V-Rep [108] : Un simulateur semblable à Gazebo, mais payant, avec licence gratuite dans le cadre académique. C'est donc un simulateur 3D multiplateforme. L'avantage de V-Rep sur Gazebo est son support professionnel ainsi que sa prise en main plus simple. L'utilisation de V-Rep en combinaison avec des robots réels ne peut se faire qu'à travers l'utilisation de ROS.

- MORSE [37] : C'est un simulateur construit sur un ensemble de librairies Python qui utilise le moteur du logiciel Blender pour interfacer un environnement 3D virtuel avec des logiciels externes. Open source et multiplateforme, il permet de simuler un grand nombre de capteurs et robots. Son avantage repose sur sa modularité et son implémentation en python qui permet de le prendre en main rapidement.

Tableau 3.1 – Simulateurs de référence

	Stage / Player	Gazebo	Webots	V-Rep	MORSE
OS	Mac, Windows, Linux	Linux	Mac, Windows, Linux	Mac, Windows, Linux	Mac, Windows, Linux
Type	2D	3D	3D	3D	3D
Portabilité sur plateforme réelle	oui	oui	oui	Avec ROS seulement	oui
Environnement réaliste	Non	Oui	Oui	Oui	Oui
Object dynamique	Non	Oui	Oui	Oui	Oui
Ressources nécessaires	Faibles	Normales	Normales	Normales	Normales
Open Source	Oui	Oui	Non	Non	Oui
Courbe d'apprentissage	Dure	Dure	Moyenne	Moyenne	Moyenne
Actualisation	Non	Forte	Moyenne	Moyenne	Faible

Une des faiblesses dans la recherche sur les MRS est le manque de base d'évaluation des performances et d'unification des simulations afin de pouvoir juger des solutions proposées. Les principaux paramètres d'un MRS que le simulateur doit pouvoir mettre en œuvre sont :

- Pouvoir avoir un nombre de robots variable
- Pouvoir utiliser différents types de robots : roulant, volant, navigant, etc.
- Pouvoir représenter plusieurs caractéristiques des robots : tailles, poids, capteurs, etc.
- Pouvoir être utilisable avec plusieurs niveaux de réalisme : position parfaite, avec perturbation du positionnement, etc.
- Pouvoir faire varier l'environnement de la mission : toute la difficulté de l'utilisation d'un MRS repose sur son organisation dans l'environnement, le simulateur doit pouvoir simuler différents environnements avec des caractéristiques physiques : bâtiment, élévation de terrain, etc.
- Avoir un système d'enregistrement des données de simulation afin de pouvoir les rejouer.

– Être libre d'accès et participatif afin de maximiser les réutilisations et partager les expériences sur les MRS.

Une unification autour d'un seul simulateur comme Gazebo, puisqu'il est gratuit et open source et répond parfaitement aux caractéristiques demandées. Cependant, du fait que ces simulateurs sont relativement complets, leur utilisation reste complexe. En effet, ils ont un temps d'apprentissage long du fait du manque d'exemple concret surtout dans le domaine des simulations multi-robots.

3.1.2 Plateforme matérielle

Lorsqu'on compare les différents travaux réalisés sur les MRS, on peut observer que la plupart des expérimentations ont été réalisées sur des plateformes différentes tableau 3.2. Il existe en effet une multitude de plateformes de référence en robotique permettant la réalisation d'expérimentation en environnement contrôlé, comme celles présenté figure 3.1.

Tableau 3.2 – Plateformes de référence

Robot	Pioneer 3-DX	Pioneer 3-AT	epuck	ARdrone
Longueur	45,5 cm	50 cm	70 mm	45 cm
Largeur	38,1 cm	49 cm	70mm	29 cm
Hauteur sans accessoire	23,7 cm	27,7 cm	50mm	N/A
Poids	9 kg	12 kg	200g	420g
Environnement et terrain	intérieur	extérieur	intérieur	intérieur
Autonomie batterie	2-4h	2-4h	2h	12min
Motorisation	2-wheel differential drive	4 wheel skid steer	2 wheels	4 rotors
Vitesse maximale	1.2 m/s	0,7 m/s	13 cm/s	5 m/s
Franchissement maximal	2.5 cm	10 cm	0	N/A
Communication	RS-232 serial or USB-Serial	RS-232 serial or USB-Serial	Infrarouge, bluetooth, RS-232	Wifi
Framework	Oui	Oui	Oui	Oui
ROS	Oui	Oui	Oui	Oui
Capteurs	8 sonars front	8 sonars front	led, micros, IR, camera	Sonar, camera
Extension pour ordinateur	Oui	Oui	Non	Non
Prix du kit	>4000€	>8000€	>1000€	250€

Chacune de ces plateformes présente des caractéristiques différentes, des modes de contrôles différents et un coût important pour des fonctions basiques excepté l'ARdrone. En effet, en 2017 une carte Raspberry Pi avec un module wifi portant à 50m coûte environ 50€, mais des robots à 4000€ n'ont pas ses capacités calculs ni de communications ou d'interfaçage. Les capacités de déplacements sont aussi critiquables puisqu'elles restent

faibles, limitant ainsi les usages en environnement intérieur. Ces différences entre plateformes sont cependant nécessaires afin de pouvoir tester des comportements de MRS sur différents types de robot. Mais d'un autre côté, cela rend les comparaisons entre expérimentations difficiles. La difficulté est d'autant plus grande que ces plateformes sont souvent modifiées afin d'intégrer de nouveaux capteurs ou moyens de communication. Il manque donc une base de référence matérielle et logicielle alors que des travaux ont déjà proposé des solutions pour certains aspects des MRS : [17] [22] Compte tenu de ces caractéristiques, il est difficile de concevoir des MRS simplement puisque de nombreuses modifications sont nécessaires (ajout de système de calcul, capteur, etc.) et demandent des coûts monétaires et en temps ingénieux important. La création d'une base de référence présente donc plusieurs avantages en plus du gain de temps de développement comme permettre des comparaisons entre différents travaux, fixer une référence en termes de performance pour le développement d'autres plateformes ou encore favoriser les échanges entre laboratoires ou industrielles. Les principales caractéristiques que doit présenter une plateforme de référence pour les MRS seraient donc :

- Capacité de déplacement tout terrain
- Capacité de calcul embarqué native
- Capacité de déplacement en autonomie
- Capacité de communication native
- Interfaces pour nouveaux capteurs
- Bas coût

FIGURE 3.1 – Illustration des plateformes présentées en 3.2

3.1.3 Plateforme logicielle

Il y a quelques années, la conception d'un système robotisé passait par la conception du robot au niveau matériel et logiciel. Il fallait donc avoir des connaissances en mécanique, mais aussi en électronique et logiciel, être capable de concevoir une carte électronique et de la programmer. Le roboticien passait donc la plupart de son temps à faire

de l'électronique. Il était alors très compliqué de réutiliser du code, car celui-ci était très lié au matériel utilisé et l'ajout de nouvelle fonctionnalité au robot était compliqué. Les middlewares de robotique ont donc été créés afin de permettre d'avoir un code réutilisable et d'éviter d'avoir à repartir de zéro à chaque changement de matériel. Un middleware est une suite logicielle basée sur deux fonctions : créer un réseau d'échange d'informations entre différentes applications informatiques et standardiser les échanges d'informations entre toutes les applications impliquées. Chaque entreprise/université créatrice de robot a donc commencé à créer son OS robotique, son propre Framework adapté à son utilité. Si les middlewares de robotique se développaient en 2013, on constate qu'en 2017, c'est ROS (Robot Operating System) [104] qui s'est imposé comme la référence dans le monde robotique aussi bien au niveau académique que industriel. ROS sert donc au développement de robot et il est open source sous licence BSD. Il est donc destiné aux entreprises créatrices de systèmes robotisés, mais aussi aux chercheurs, étudiants ou simples amateurs. Ainsi dans la liste des utilisateurs de ROS, on retrouve Aldebaran, Shadow's robotic, Clearpath, Amazon, mais aussi des universités comme le MIT, Telecom Bretagne, le LIRMM, le LAS, etc. ROS étant open source et généraliste, une large communauté s'est logiquement créée autour de ce middleware. En 9 années d'existence, ROS s'est imposé comme la référence des middlewares pour la robotique. On recense plus de 3500 citations de l'article d'introduction de ROS « ROS : An Open-Source Robot Operating System » et plus de 2000 contributeurs pour les 10 millions de lignes de code que comporte le projet, avec en moyenne 20 commits par jour. Cette popularité sur le plan académique s'est aussi transmise au niveau des industriels qui y voient une opportunité technologique et économique d'augmenter leur capacité robotique. Ainsi plusieurs grands industriels comme Boeing, Siemens, PSA Peugeot Citroën utilisent déjà ROS. La philosophie de ROS peut être décrite en 4 points :

- **Peer-to-Peer** : Lorsqu'un robot devient complexe, ce qui est rapidement le cas, il se peut que plusieurs « entités » aient à dialoguer avec les différents organes du système par exemple une carte embarquée sur le système et un ordinateur externe pour faire de plus gros calculs. ROS, avec son architecture basée sur la couche TCP/IP et une sérialisation de donnée efficace, permet de réaliser des communications synchrones et asynchrones entre application et entre différente entité du robot. Le tout est coordonné par un service maître.

- **Basé** sur des outils : Dans le but de rendre ROS moins compliqué au niveau de sa structure et plus flexible, le middleware n'a pas été codé de manière à ne former qu'un seul exécutable. Les développeurs ont plutôt choisi un design modulaire utilisant une multitude d'outils servant à lancer les différents composants de ROS. Ainsi s'il y a problème sur une des applications du robot, les autres ne seront pas touchées et la panne peut être localisée rapidement. Cette structure permet de gagner en stabilité et de perdre en complexité de fonctionnement. Ces différents outils sont séparés en 3 catégories :

  - Les outils d'aide au développement : ils aident à la création d'application uti-

lisant ROS accélérant ainsi le développement.

- Les outils d'analyse : ils permettent de gérer les messages, de visualiser des graphiques reliant les différents nœuds et messages des projets, de visualiser les informations de débogage ou encore de paramétrer ROS
- Les outils fonctionnels : Interface capteurs, gestion de la planification de trajectoire, simulateurs, etc.

- **Multi langage** : Lorsque l'on réalise un programme, chaque concepteur à sa préférence quant au langage utilisé. Ces préférences résultent de la syntaxe de chaque langage, du mode de débogage ou tout simplement du fait qu'une personne peut ne pas connaitre tel ou tel langage. C'est pourquoi ROS a été créé de façon à supporter différents langages de programmation. Ainsi un développeur peut coder sous ROS en C++, Python, LISP, Java, JavaScript, etc. En fait, la connexion Peer-to-Peer utilise le XML-RPC qui existe dans la plupart des langages ou par des ponts de communication, rosbridge pour le JavaScript par exemple. De cette façon, une application codée en C++ peut laisser des informations dans un message codé en XML-RPC qui pourra alors être lu par une autre application codée en Python. ROS est donc complètement neutre du point de vue du langage, les langages peuvent être mixés dans un seul et même projet.

- **Léger** : Les développeurs de ROS ont voulu un middleware qui puisse être réutilisable, mais ils ne pouvaient pas inclure tous les drivers ou algorithmes déjà implémentés, cela aurait rendu ROS beaucoup trop lourd. Il a donc été décidé de privilégier la construction de bibliothèques. Ainsi chaque utilisateur peut télécharger les outils qui lui sont utiles, de cette manière, le roboticien peut avoir un système suffisamment léger pour pouvoir l'inclure sur une carte embarquée et il n'a pas un système inutilement complet : inutile pour une personne développant un système sans dispositif de vision d'avoir une bibliothèque de traitement d'image incluse dans sa carte embarquée. De plus, ce système permet de simplifier le développement, chaque driver est contenu dans un exécutable, ce qui facilite les tests unitaires par exemple.

- **Gratuit et open source** : ROS est sous la licence BSD qui permet l'utilisation de celui-ci dans des projets à but commercial ou non. ROS a été créé pour éviter d'avoir à de recommencer le développement d'un projet à zéro, il parait alors logique de laisser un plus grand nombre utiliser cette plateforme et ainsi permettre de multiplier les bibliothèques disponibles.

Étant la référence pour les développements de robot, ROS est aussi la référence pour les MRS. Si tous les travaux sur les MRS ne l'utilisent pas, la plupart proposant des expérimentations sont basés surs ROS afin de profiter d'écosystème permettant la réutilisation du code utilisé en simulation sur les plateformes réelles. Cependant, ROS présent aussi de nombreux désavantages qui n'en font pas le choix parfait pour les MRS.

Initialement prévu pour fonctionner sur le robot PR2 de Willow Garage, ROS s'est répandu au sein d'autres robots et d'autres applications dont les usages n'avaient pas été prévus. Malgré ses nombreux atouts, ROS présent aussi certains désavantages comparés à d'autres middlewares ou logiciels :

- ROS n'est pas temps réel.
- Il n'y a pas de suivi des mises à jour détaillé des sources (changement des API, protocole).
- En fonction des librairies installées, l'espace de stockage nécessaire peut se révéler important.
- Pas de déploiement sur de très petites plateformes (AVR, etc.)
- Pas de prise en compte de la qualité du réseau : QoS, perte de paquet, etc.
- **Pas de standard pour l'approche multi robot**
- Communications centrées sur le protocole TCP et pas de support pour des protocoles broadcast
- Peu d'interaction avec les autres écosystèmes : IoT etc. [10] [57] [31]

Ainsi, l'utilisation de ROS ne donne aucune garantie sur le bon fonctionnement d'une solution et manque de performance. C'est un outil qui doit rester dans un usage de démonstration. Afin de combler ces manques, une nouvelle version de ROS nommée ROS 2.0 est en cours d'élaboration. Elle amènera, en plus de la résolution des désavantages cités précédemment, un nouveau niveau d'abstraction permettant d'utiliser les nouvelles technologies actuelles (zeroconf, ProtoBuf, etc.) largement utilisées dans d'autres domaines et renforçant ainsi la modularité du middleware.

Si la modularité et la popularité de ROS ont permis son énorme développement, cela pose aussi des nombreux problèmes pour un usage plus poussé que le prototypage. En effet, la qualité des différents outils proposés par ROS n'est pas la même pour tous. Certains outils manquent clairement de tests et de vérifications ce qui en plus d'amener des retards dans les temps de développement, entraîne aussi des problèmes de fonctionnement et fiabilité. De plus, la majorité des contributions provenant de laboratoires ou d'amateurs en robotique, les librairies proposées manquent de qualité et d'optimisation pour un usage en système embarqué (utilisation de librairies logicielles expérimentales, peu de protection contre les dépassements de mémoires ou division par zéro, etc.). Initialement prévu pour des robots terrestres, ROS manque aussi d'interface et de librairies standard pour les autres types de robots notamment les volants. Enfin, il ne propose aucune librairie pour les actions de sécurité lors de problèmes malgré la présence d'un outil d'auto-diagnostic. Ainsi, ROS est une solution intéressante pour les MRS, mais pas suffisante seule.

3.2 Contribution : création de la plateforme

Participant projet DAISIE, il a été proposé une solution de plateforme matérielle et logicielle pour les essaims. Le projet DAISIE consistait à mettre au point, sur la base des simulations proposées par le projet SUSIE, un démonstrateur d'essaims de robots terrestres et aériens composés de dix UAVs et six UGVs qui devaient permettre d'évaluer les capacités de tels systèmes en grandeur réelle. DAISIE est entièrement financé par la Direction Générale pour l'Armement (DGA) à travers le programme d'Accompagnement Spécifique des Travaux de Recherches et d'Innovation Défense : Maturation et valorisation (ASTRID-Maturation). La durée du projet était initialement de 24 mois (mars 2014 - mars 2016). Il a déjà été abordé précédemment le manque de références logicielles et matérielles pour la construction d'essaim. Avec un temps, un budget restreint et des équipes non spécialistes en robotique, il était donc important de construire une plateforme générique et simple qui répondait aux besoins des missions du projet DAISIE, mais aussi de montrer que l'on pourrait couvrir un plus grand champ de missions. Cela a permis de fixer les caractéristiques principales que devait présenter la plateforme :

- Plateforme simple à mettre en œuvre et commune pour les différents types de véhicules.
- Utilisable sur plusieurs types de terrains.
- Utilisable sur plusieurs missions.
- Bas coût afin d'avoir le plus grand nombre de robots possible.

À cela, il a été ajouté d'autres caractéristiques :

- Conforme aux réglementations d'usage des véhicules autonomes (redondance des communications, protocoles d'action de sécurité, formation, etc.).
- Open source.
- Portabilité entre simulation et robot réel.
- Évolutive.
- Acquisition et traitement vidéo simple embarqué
- Transmission radio et vidéo

3.2.1 Méthodologie de la conception du système

La création de la plateforme représentait un défi technologique. En effet, en 2013, les technologies embarquées restaient coûteuses et les drones volants n'étaient qu'au début de leur industrialisation. L'essaim de DAISIE devait être capable de réaliser des missions

surveillance de zone à l'aide d'un algorithme basé sur l'évaporation de phéromones virtuelles (EVAP). Les UGV devaient être équipés de caméras et de capteurs ultrason pour la reconnaissance. Dans le cas d'une détection, ils devaient transmettre un flux vidéo à l'opérateur qui devait alors valider la détection. Du fait de leur temps de vol limité, les drones volants devaient être utilisés comme relais de communication dans le cas d'une détection ou afin de renforcer le lien radio dans l'essaim. Objectifs prévus :

- Démonstration d'algorithmes de déplacement en essaim par carte de phéromones
- Patrouille automatique
- Détection d'objets (choix de nains de jardin)
- Reconfiguration de l'essaim
- Retransmission de flux vidéo

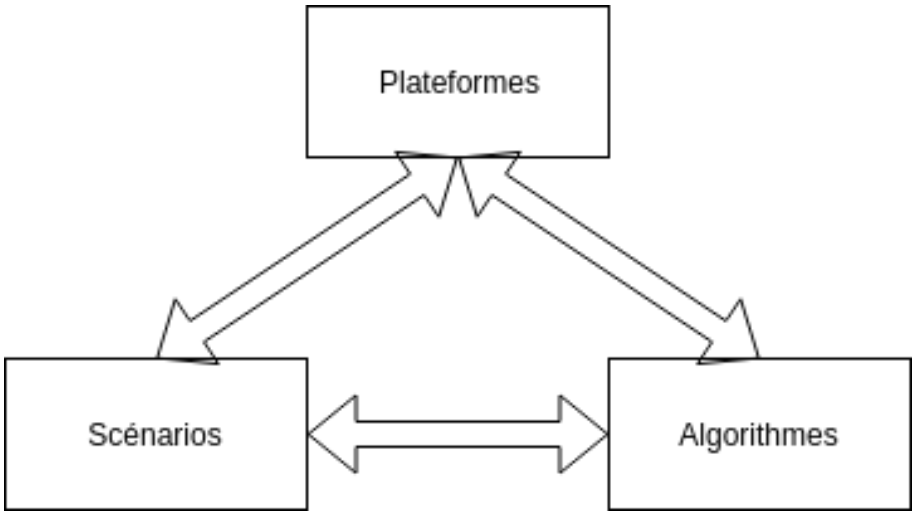


FIGURE 3.2 – Diagramme de compromis de construction d'un MRS

La figure 3.2 met en avant le challenge de la construction des MRS. Celui-ci repose sur l'équilibre entre 3 facteurs : les missions prévues, les logiciels utilisés et le matériel choisi, chacun influençant les autres. Un des objectifs du projet autre que la faisabilité technique est de montrer l'intérêt et l'efficacité réelle d'un tel système ce qui n'a que peu été réalisé. Pour ce faire, la phase de design de la plateforme est passée par plusieurs séquences de tests afin de juger de la pertinence des choix et évaluer les performances de la plateforme drones. Le diagramme 3.3 suivant illustre la méthodologie utilisée afin de créer une plateforme pour les MRS.

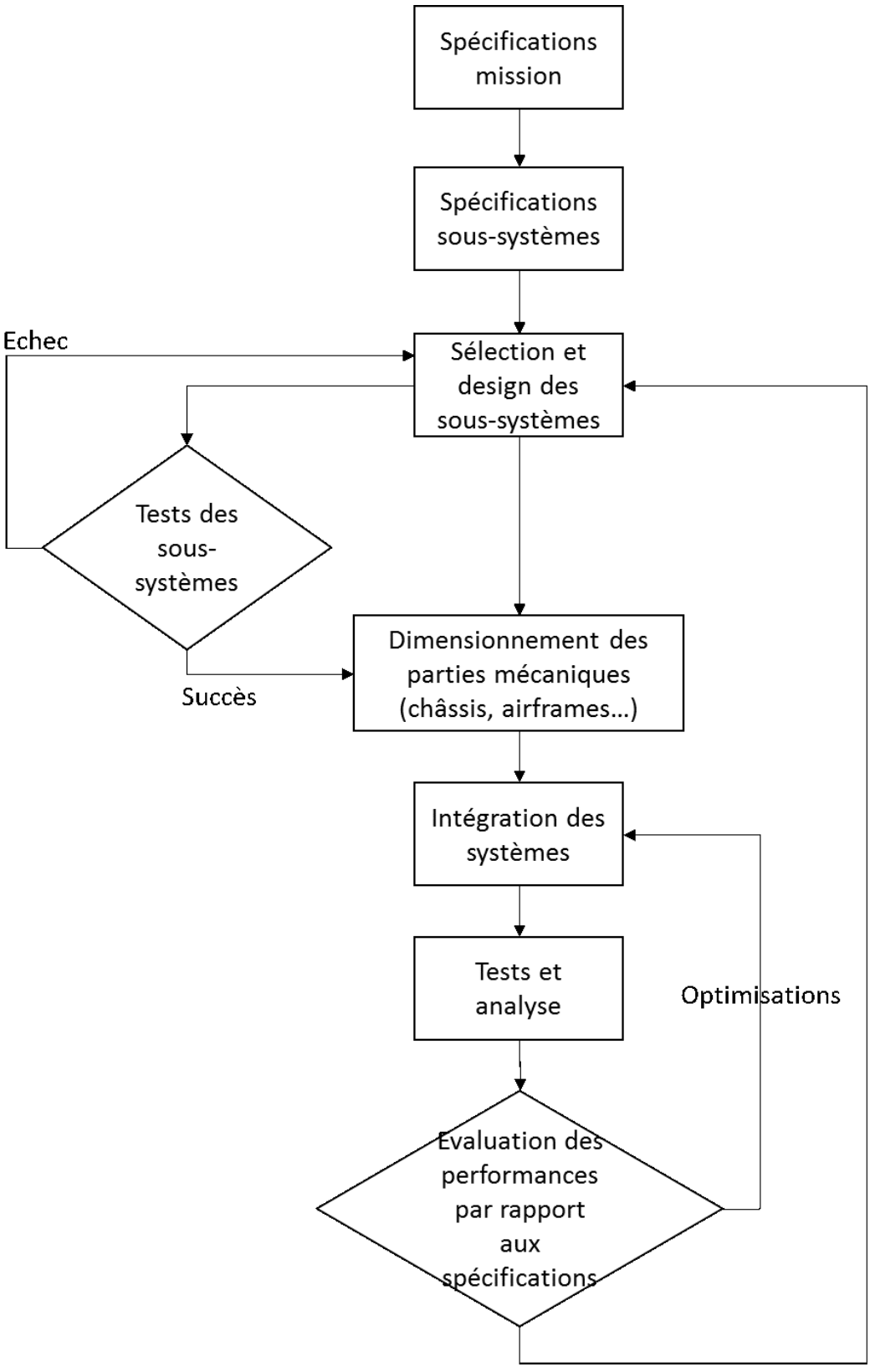


FIGURE 3.3 – Diagramme de sélection des composants

La gestion "complète" d'un essaim de drones pose de nombreux problèmes tels que la prise en compte des questions d'énergie (temps d'autonomie et recharge de batteries), difficultés de localisation précise des drones, prise en compte de conditions météo, gestion de la charge utile distribuée (gestion des capteurs embarqués), etc. C'est pourquoi, la conception de véhicule autonome est souvent associée au développement logiciel, mais à cause du manque de plateformes matérielles, la complexité de conception de ces dernières est souvent la tâche la plus longue. En effet, il faut pouvoir s'assurer que chacun des robots fonctionne correctement avant d'utiliser des algorithmes de fonctionnement en groupe. Cette tâche requiert un temps d'autant plus important que les agents sont de natures différentes comme l'usage d'UAV et UGV. La création d'un MRS est donc complexe à la fois par la définition de son modèle, mais aussi dans sa mise en œuvre réelle. Cette difficulté intrinsèque explique en partie le faible nombre de démonstrateurs d'essaims allant au-delà des simulations.

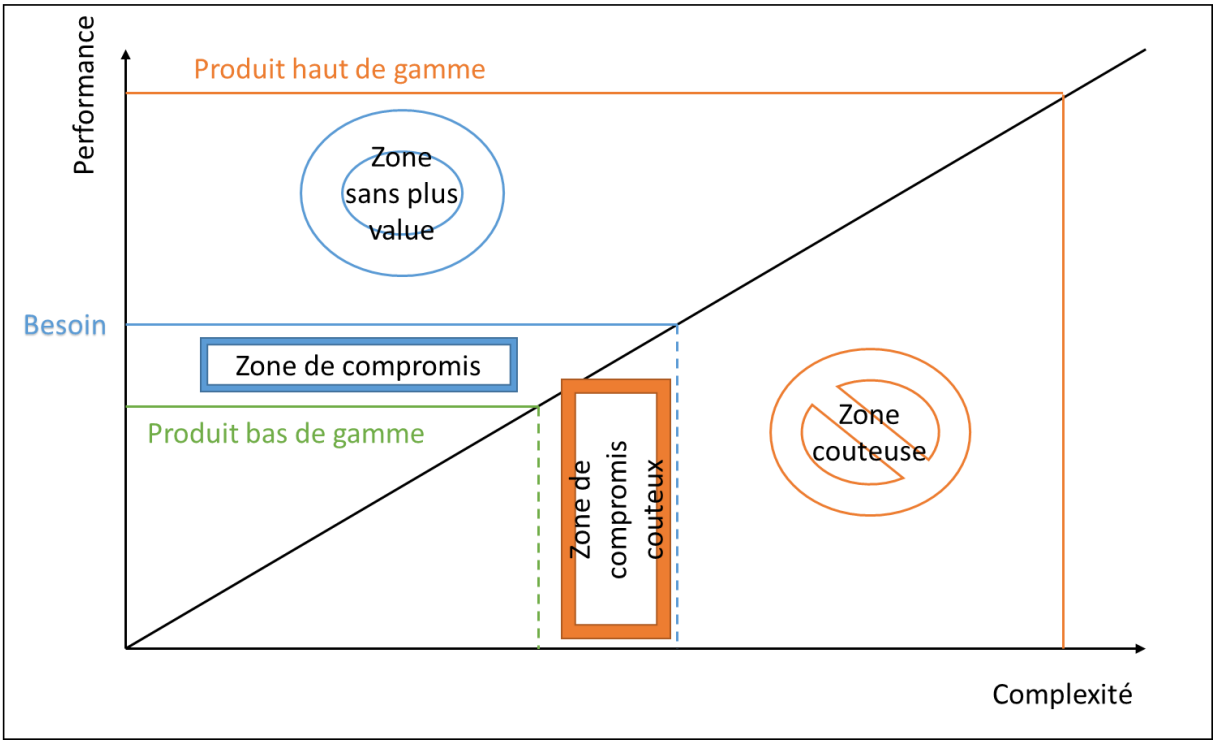


FIGURE 3.4 – Compromis entre performance et complexité

Le graphique du compromis entre performance (résultat obtenu) et complexité (niveau de difficulté de conception ou d'utilisation), figure 3.4 est applicable à l'ensemble de la sélection des composants des MRS. Ainsi un ARdrone ou un Bebop de Parrot sont des bons compromis : ils ont des performances satisfaisantes au regard de leurs capacités de vol et de traitement embarqué. À l'inverse, un robot comme Atlas permet des travaux avancés en robotique, mais au prix de complexe développements et coûteux essais. Dans

les parties précédentes, nous avons montré que les plateformes actuelles étaient souvent soit surdimensionnés : dans la zone sans plus value, soit sous-dimensionnés : dans la zone de compromis coûteux, avec des plateformes spécifiques, coûteuses et avec peu de performance globales. Nous pensons que le moyen de concevoir des MRS les plus génériques possible est de se concentrer sur la simplicité de construction et de mise en œuvre. En effet, nous verrons dans les prochaines sections que faire le choix de la simplicité sur la performance peut être gagnant sur le long terme et lorsque le temps et le budget de développement sont réduits.

Les sections suivantes présenteront les choix effectués afin de concevoir la plateforme en privilégiant la simplicité d'utilisation et le bas coût de la solution.

3.2.2 Choix des logiciels

La caractérisation des MRS dans la section 2.3.1, nous a permis de mettre en avant 5 challenges : gestion de l'énergie, communication, détection et évitement, gestion du groupe, interface avec les opérateurs. À cela, nous devons ajouter la gestion de la localisation. Le choix des logiciels utilisés pour les MRS doit donc répondre à ces challenges. Ainsi les choix logiciels utilisés pour le projet DAISIE ont été fixés en tenant compte des caractéristiques que devaient présenter les drones de l'essaim présenté précédemment.

Choix de l'OS

Il existe de nombreux OS dédiés aux systèmes embarqués : Linux, Android, RTOS, Nuttx, Yocto, Chibios pour les plus connus. Chacun présentant des avantages et inconvénients. Cependant, le choix de l'OS reste un élément critique puisque tous les développements vont être faits sur cette base. Nous pensons que le développement des MRS se fera à travers la modularité de la plateforme et la simplicité de celle-ci, ainsi, nous visons un choix dans la zone de compromis. La simplicité de mise en place, de développement et d'utilisation sera un avantage face à un gain de performance. C'est pour cela que nous avons choisi de baser la plateforme sur l'OS Linux en particulier sur la distribution Ubuntu [122]. En effet, comme il a été présenté précédemment, les cartes embarquées munies de processeurs puissants et à bas coûts sont nombreuses en 2017. Cela permet une simplification des développements, mais aussi une relative portabilité dans le temps puisque le développement logiciel n'est plus directement lié au matériel. Le choix de la distribution Ubuntu permet un bon compromis entre simplicité, performance, sécurité. La distribution possède de nombreux logiciels de développement et utilisables en robotique déjà compilée dans des versions récentes : serveur web, gestion des périphériques, accélération matérielle, etc. Cette distribution possède aussi une bonne base documentaire et un support longue durée ce qui permet la création de systèmes viables et actualisables dans la durée. De plus, Ubuntu est proposé par nombreuses plateformes matérielles ce qui simplifie l'intégration. Enfin, c'est aussi la distribution de référence pour le middleware ROS qu'il est indispensable d'utiliser en 2017. Le choix de ROS fait sens dans un système complexe. Basé sur un système de messagerie efficace, une abstraction matérielle et de

nombreuses librairies, ce middleware permet de faire fonctionner ensemble les différents composants d'un robot. Open source et possédant une forte communauté avec un soutien industriel, son usage permet un gain de temps notable dans le développement logiciel et le développement d'algorithmes pour les robots et les véhicules autonomes. En effet, plusieurs librairies sont disponibles pour une large gamme de capteurs (caméra, LIDAR), mais aussi pour des fonctions de navigation et de communication. Son utilisation permet à la fois la planification de trajectoire, la gestion de l'évitement d'obstacle, le mapping, la reconnaissance d'objet, la communication inter agents et la création d'interfaces utilisateur. Cependant, il présente aussi de nombreux défauts présentés précédemment, notamment au niveau de la sûreté de fonctionnement et du temps réel (respect des performances minimum) qui sont indispensable pour un déploiement de MRS : redondance des contrôles, actions automatisées sur défaillance matérielle comme les problèmes GPS ou de batterie, toujours au moins un contrôle direct sur l'agent, etc.

Choix de l'autopilote

Tableau 3.3 – Comparaison des Autopilotes

Autopilote	ArduPilot [6]	PX4 [78]	DJI [30]	PaparazziUAV [89]
Open-source logiciel et matériel	Oui	Oui	Non	Oui
Véhicule supporté	Copter, Avion, Roulant, Navigant	Copter, Avion, Roulant [1]	Copter	Copter, Avion
Gamme de prix	200€	200€	400€	200€
MRS compatible	Oui	Oui	Non	Oui
Simulation complète	Oui	Oui	Non	Non
Compatible industrie	Oui	Oui	Oui	ind.

L'utilisation de drones volants demande un contrôleur de vol fiable et éprouvé capable de s'occuper de la stabilisation de l'appareil pendant le vol, chose que ne permet pas ROS. C'est pourquoi il existe des autopilotes pour drone volant qui possède une plateforme matériel et logiciel capable de gérer les fonctions bas niveau du drone. Dans le cas de notre essaim, l'autopilote doit pouvoir s'adapter à des drones volants et roulants. Le développement d'un autopilote complet nécessitant des connaissances avancées en électronique

1. support expérimental en Aout 2017.

et logiciel en plus de connaissance sur le type de robot voulu (volant, roulant, etc.) et un temps de développement long et coûteux, il est donc préférable d'utiliser une solution existante. Plusieurs projets de recherche ont abouti à la création d'autopilotes. Si ces logiciels présentent des fonctionnalités permettant leur utilisation dans le cadre universitaire [79], ils possèdent aussi de nombreux désavantages. Étant développé initialement pour un projet, il est fort probable que le logiciel ne soit plus maintenu ce qui nécessitera une adaptation aux outils actuels. À cela s'ajoute la qualité du développement, initialement prévu pour une plateforme et une mission, la fiabilité de l'autopilote n'aura pas été prouvée sur une autre plateforme ou mission. Il a été choisi d'utiliser un autopilote Commercial Off The Shelf (COTS). Devant la multitude de choix sur le marché, il a été décidé de se baser sur un autopilote open source afin de profiter du support d'une large communauté, mais aussi d'un remplacement facile du matériel associé en cas de défaut matériel ou dégâts pendant les tests. Enfin, l'autopilote devait être facilement disponible et bas coût.

Sur ces contraintes, la plateforme ArduPilot a été sélectionnée comme elle est la seule à proposer un support à la fois pour les véhicules volants et roulants. La figure 3.5 présent un aperçue des différents types de véhicules supportés.

FIGURE 3.5 – ArduPilot plateformes ArduPilot sous licence CC BY-SA 3.0

La plateforme ArduPilot présente l'avantage d'être bas coût, open source avec une forte communauté et des industrielles : NASA, Parrot, DARPA. Cette plateforme est largement utilisée dans le monde et a prouvé son efficacité à plusieurs reprises : ArduRover (version UGV) a remporté les sessions 2013 et 2014 de la compétition Sparkfun Autonomous Vehicle Competion, ArduPlane (version drone voilure fixe) a remporté les sessions 2014 et 2016 de l'Outback Challenge [19] en Australie et ArduCopter (version UAV) la AUSVI SUAS Competition 2017 et 2018 [117].

Initialement lié à une plateforme matérielle, son support s'est largement étendu avec son utilisation sous Linux avec des cartes comme la référence Raspberry Pi. Ces caractéristiques font de cette plateforme un excellent choix pour les MRS. En effet, la plateforme est suffisamment générique pour s'adapter à un grand nombre de véhicules et possède grâce à sa licence open source et son support d'industriel, suffisamment de tests et de fonctionnalités pour être conforme avec les lois d'utilisation des véhicules autonomes dans de nombreux pays. Du fait du développement de drones volant en mode automatique, une attention particulière a été apportée aux mesures de sécurité : redondance des contrôles, actions automatisées sur défaillance matérielle, briques technologiques qui sont absentes de ROS. Enfin, la plateforme de l'autopilote ArduPilot est utilisable avec ROS grâce à de nombreuses librairies capables d'effectuer la conversion entre les différents protocoles de communication utilisés et possède un simulateur SITL (Software In The Loop) permettant de simuler l'intégralité des fonctionnalités. Plus de détails sur l'architecture d'ArduPilot seront donnés dans les chapitres suivants.

Ainsi l'autopilote ArduPilot a été choisi comme base pour les robots terrestres et aériens. Il permet de les mettre en oeuvre par simple utilisation du logiciel et fournit plusieurs IHM de configuration. C'est une force par rapport à ROS qui nécessite une configuration avancée pour chaque robot. Les fonctionnalités de l'autopilote sont suffisantes pour un déplacement en suivant des coordonnées point de passage GPS et permettre la communication inter-drones. D'autres parts, ROS possèdent d'importantes librairies pour le traitement d'image et d'algorithme de navigation complexe. Il a donc été choisi de séparer le contrôle des robots en deux parties : la partie autopilote gère les fonctionnalités de base (gestion des capteurs, contrôle moteur, sécurité, etc.) et ROS gère les algorithmes de groupes et avancés (navigation, échange de cartes de phéromones, gestion de la vidéo, etc.).

Malgré une grande praticité d'usage, le projet ArduPilot, en 2013, était insuffisamment abouti pour permettre son usage direct avec des MRS tant au niveau simulation que mis en oeuvre. Un travail d'améliorations a donc été entrepris et sera présenté dans le chapitre 4

Choix des protocoles de communication

Les communications dans les MRS mettent en avant deux problématiques : les protocoles et les transmetteurs. En effet, la gestion des communications dans les MRS impacte fortement leurs comportements : plus les communications sont nombreuses plus il y aura d'énergie utilisée et plus les problèmes d'engorgements des communications seront présents. Le type de système multi-robots définit la fréquence de communication et le modèle utilisé : graphe complet, système centralisé, etc. Cependant, cette définition doit être liée à un protocole de communication et une interface matérielle. La solution la plus simple en 2017 reste l'usage de transmetteur WiFi (tableau 3.4).

Tableau 3.4 – Comparaison des technologies de transmission sans fils les plus courantes.

Technologies	Portée	Poids	Complexité	Coût	Consommation énergétique
WiFi	Moyenne	Moyen	Moyenne	Moyen	Moyenne
Zigbee	Longue	Léger	Faible	Faible	Faible
Bluetooth	Faible	Léger	Moyenne	Faible	Faible
Cellulaire	Très Longue	Léger	Haute	Haut	Moyenne
Acoustic	Faible	Lourd	Haute	Haut	Haute

En effet, ceux-ci arrivent à un bon compromis entre bande-passante, gestion de la congestion, et portée. Néanmoins, nous verrons dans la partie matérielle que l'usage de WiFi n'est pas simple de mise en œuvre.

Les protocoles de communication pour la robotique sont nombreux. Cependant, comme ROS est utilisé il fait sens d'utiliser le protocole de ROS. Le protocole de ROS est un protocole de donnée sérialisée semi-décentralisée. En effet, celui-ci repose sur le mécanisme de publish-subscribe basé sur le généralement TCP. Le TCP a l'avantage d'être relativement simple à mettre en œuvre, de proposer des communications fiables et les paquets TCP sont garantis d'arriver dans l'ordre d'envoi. Les données transmises par ROS ne transitent pas par le Master et les données ne sont pas transmissent en XMLRPC. L'XMLRPC et le Master sont uniquement utilisés pour la phase de négociation des connexions de données afin de permettre la mise en liaison des publishers et subscribers et gérer la liste des publishers (gestion des conflits, déconnexions, etc.). Les données sont donc directement transmises de façon sérialisée entre les *publishers* et les *subscribers* comme résumé sur la figure 3.6

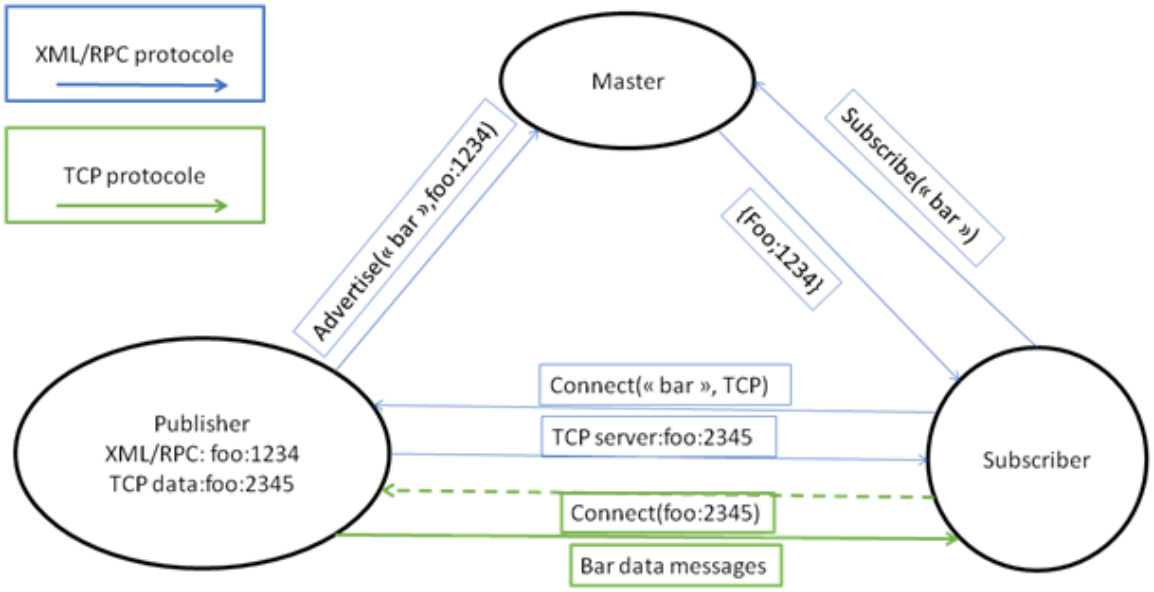


FIGURE 3.6 – Protocole ROS

Si ces caractéristiques sont efficaces en connexions câblées, la plupart des communications des MRS reposent sur des transmissions non filaires sujettes à des pertes d'informations ou de retard comme le WiFi et les réseaux mobiles. C'est pourquoi ROS ne répond que peu à l'usage des systèmes décentralisés ou devant fonctionner en conditions dégradées, même si différentes options existent pour corriger ce défaut : rosserial, rosbridge, etc. La sortie de ROS2.0 [87] est encore en phase d'évaluation, comblera en partie ces manques en basant son architecture sur un DDS (Data Distribution Service) et la non-dépendance à un centralisateur comme Master. Ceci est problématique pour un aspect de fiabilité des MRS puisque l'usage d'un centralisateur pose un problème évident d'engorgement des communications, mais possibilité de perte totale du contrôle du système en cas de défaillance du centralisateur ou de perte de communication avec celui-ci. L'autopilote ArduPilot utilise des connexions avec le protocole MAVLink [76]. C'est un protocole léger et open source, basé sur des structures de données binarisées utilisables par des systèmes avec de faibles ressources et des transmissions avec peu de bande passante. Il repose lui aussi sur un mécanisme de publish-subcribe et de point-à-point. MAVLink est construit de façon à être utilisé avec des flux de données à faible débit : position du drone, altitude, etc., transmis en multicast. Le but étant de transmettre l'information à ceux qui peuvent la recevoir, mais sans garantis de réception. Le point-à-point étant plutôt réservé aux communications nécessitant un acquiescement et une garantie de délivrance comme les reconfigurations, ordre d'urgences, etc. Enfin, il permet un cryptage des données, un adressage des messages et un reroutage des messages par adressage permettant à la fois la sécurisation des données et la retransmission des données sur de longues distances en passant par plusieurs relais, deux caractéristiques pouvant être importantes pour les MRS. Fort de nombreux messages standard et de support industriel, ce protocole s'est

imposé comme une référence dans les autopilotes pour UAV, mais aussi au niveau des UGV puisque le récent Turtlebot 3 [109] basé sur ROS l'utilise. La figure 3.7 schématise l'utilisation du protocole MAVLink dans l'essaim.

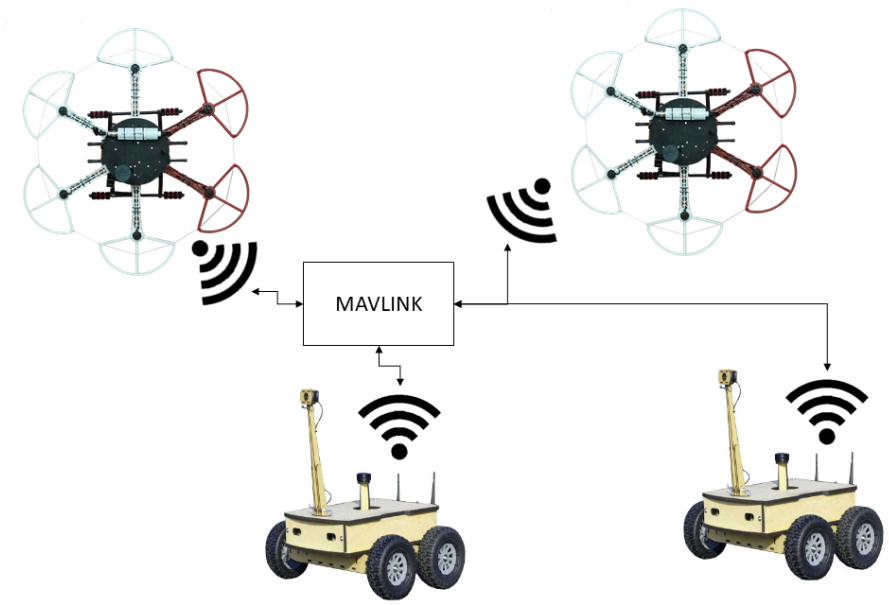


FIGURE 3.7 – Représentation de l'usage du protocole MAVLink dans l'essaim

Conclusion

Il a été présenté ici une sélection de logiciel et protocoles devant servir de base à la gestion des robots dans un MRS générique et résumé figures 3.8 et 3.9. En effet, ces choix présentent chacun des avantages et inconvénients qui ont été présentés. Leurs plus grands avantages sont leurs non-dépendances à une plateforme matérielle spécifique, leur standardisation qui permet un usage industriel et une suite de simulation permettant à la fois des tests, mais aussi le portage immédiat des simulations dans le réel. Il a été choisi ici de séparer les fonctions haut niveau (calcul de trajectoire, évitement d'obstacle, SLAM, gestion de la vidéo, etc.) basées sur ROS et les fonctions critiques comme la gestion des actionneurs (moteurs), des pannes et de la navigation basique basés sur l'autopilote. Cette combinaison s'est révélée la plus simple et la plus robuste lors des phases de tests, et elle répond aux problématiques de sécurité dont ont besoin les industriels. Si ROS pourrait répondre entièrement à ce besoin, il n'existe actuellement pas de solution aussi complète et standardisée, les solutions basées sur ROS utilisant systématiquement un contrôleur bas niveau comme une carte Arduino, l'usage d'un autopilote permet un gain important en capacités.

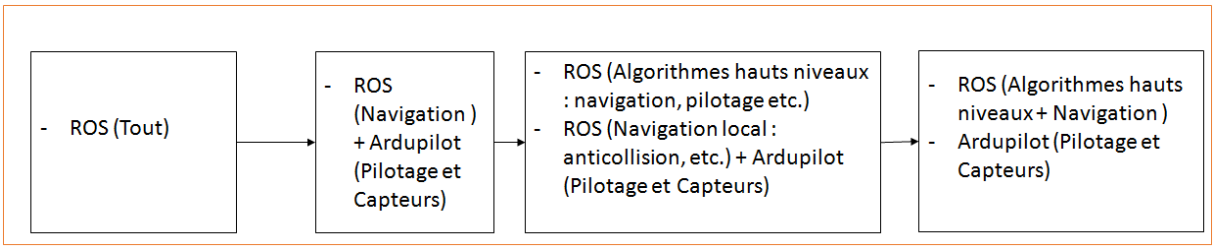


FIGURE 3.8 – Évolution du choix des logiciels de 2013 à 2017

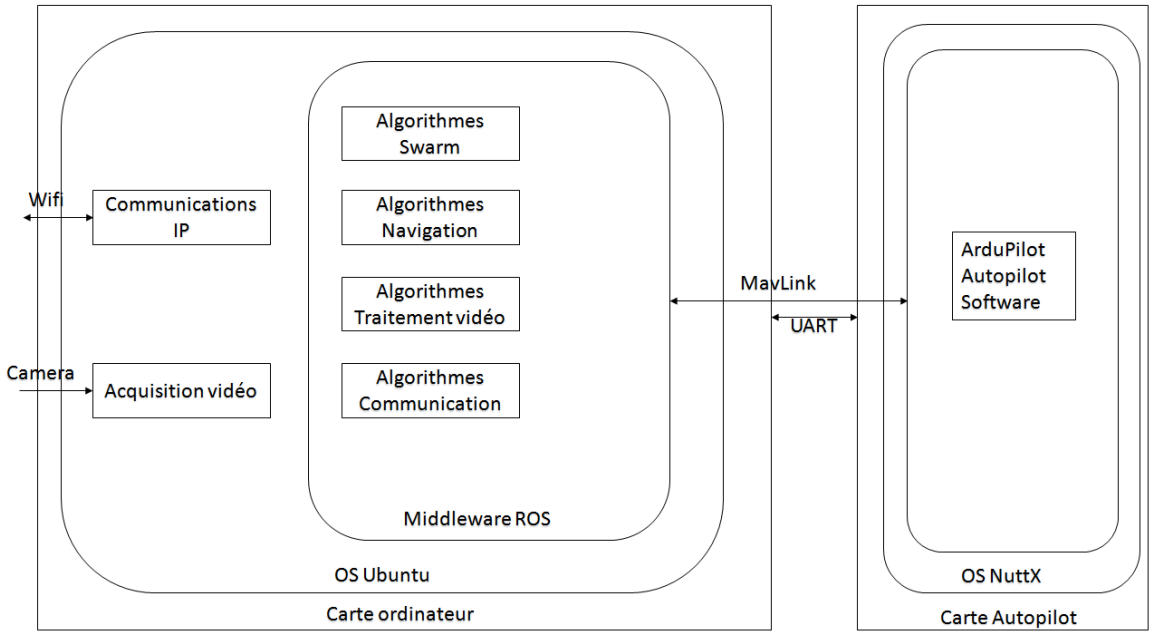


FIGURE 3.9 – Diagramme logiciel de la plateforme

Nous remarquerons que le détail de tous les logiciels et librairie devant être utilisés n'ont pas été donnés à des fins simplificatrices. En effet, nous cherchons à montrer ici l'aspect global de la plateforme générique, ainsi il serait superflu de donner le détail des implémentations des librairies de capteurs ou de gestion des MRS. La contribution porte ici sur une analyse puis une exploitation des outils les plus utilisés et utiles pour une plateforme pour MRS de façon à joindre les milieux académiques et industriels.

3.2.3 Choix du matériel

Comme présenté précédemment, le matériel pour la robotique est en constante évolution ainsi, en 4 années de thèse, le matériel de la plateforme pour MRS que nous avons

conçue a évolué et suivi les innovations technologiques chaque année. Comme expliqué précédemment, ce choix d'évolutivité est vital dans un domaine comme les MRS et la robotique en général.

Carte embarquée : Processeur principal

Au vu du nombre de cartes et ordinateurs disponibles sur le marché, un choix sur des critères nous paraissant important a été fait (notamment le prix). Pour l'évaluation globale, les critères de choix ont été affectés d'un coefficient par ordre d'importance allant de 1 à 3 suivant le tableau suivant :

Tableau 3.5 – Critère de sélection des cartes embarquées

Critère	Coefficient
SOC	3
Connectivité	3
Encombrement	1
Open Source	2
Communauté et industriels	2
Rapport performances, qualité sur Prix	2

Justifications :

- SOC : Au vu des algorithmes qui allaient être utilisés et du peu de temps qui serait alloué aux optimisations, la puissance de calcul du SOC était un paramètre primordial.

- Connectivité : Les drones devant pouvoir servir dans la durée et être mis à jour avec différentes configurations ou capteurs, la présence de plusieurs interfaces de connectique déjà présente était un plus.

- Encombrement : Les systèmes visant à être embarqués, le gain de poids et d'encombrement est un plus.

- Le critère open source était nécessaire pour pouvoir utiliser et modifier la configuration de la carte sans avoir de contraintes industrielles (NDA, attente des mises à jour, etc.).

- Communauté et industriels : Afin de bénéficier de support technique et d'une mise à jour dans le temps, une bonne communauté ou un support d'entreprise est nécessaire.

- Rapport performances, qualité sur Prix.

Tableau 3.6 – Sélection des cartes embarquées

Cartes	BeagleBone Black	Raspberry Pi 2 Model B	Radxa Rock	Cubie Board 2	Odroid C1
SOC					
SOC Type	TI AM335x	Broadcom BCM2836	Rockship RK3188	All winner A20	Amlogic S805
Core Type	ARM Cortex A8	ARM Cortex A7	ARM Cortex A9	ARM Cortex A7	ARM Cortex A5
# Coeurs	1	4	4	2	4
Fréq. CPU	1 GHz	900 MHz	1.6 GHz	1 GHz	1.5 GHz
Choix	6	7	9	5	8
Connectivité					
USB 2.0	1	4	2	2	4
GPIO	2*46	40	2*40	2*48	40
Port Série	Oui	Oui	Oui	Non	Oui
WIFI	Non	Non	Oui	Non	Non
Choix	7	5	9	7	6
Encombrement					
Dimensions (cm)	86.4 * 54.6 * 19	85.6 * 54 17	100 * 80 * 12	100 * 60 * 20	85 * 56 * 18
Poids (g)	40	45	60	140	40
Choix	8	8	7	5	8
Open Source					
Hardware	Oui	Non	Oui	N/A	Oui
Software	Oui	Oui partiel	Oui partiel	Non	Oui
Choix	9	6	8	4	8
Communauté et entreprise					
Indice fiabilité	8	8	6	3	7
Rapport performances, qualité sur prix					
Prix moyen	44 €	30 €	56 €	65 €	30 €
Indice rapport	7	7	8	5	8
Évaluation globale					
Total	95	86	105	65	98

Notre plateforme devant être générique et adaptée aux systèmes volants où la taille, le poids et la consommation sont des critères très importants, les ordinateurs classiques n'étaient pas sélectionnables. De plus, ne pouvant nous permettre un développement de plateforme spécifique, il a été choisi de sélectionner une plateforme SOC de type Raspberry Pi open source afin de profiter de généricité et d'évolutivité, de performance et de nombreuses interfaces matérielles à prix réduit. La tableau 3.6 présente la liste des cartes évaluées en 2014.

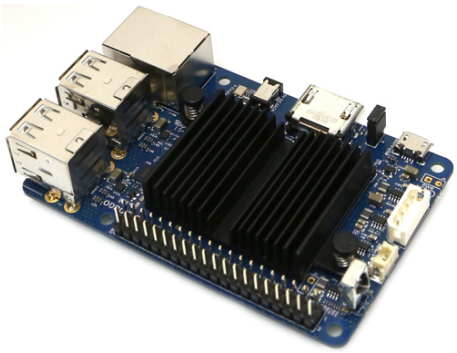


FIGURE 3.10 – Carte retenue : Odroid C1+

Les indices de choix ou de qualité sont classés de 1 à 10 dans l'ordre croissant d'importance (de 1=médiocre à 10=excellent). Les indices de choix ont été fixés par rapport aux caractéristiques, à l'étude et aux tests des cartes en 2014. Suite aux tests, la carte Radxa, qui avait été sélectionnée en premier lieu, a été changée pour une carte Odroid C1+. En effet, de nombreux problèmes logiciels ont été rencontrés et le support sur la carte était très faible, voire inexistant. Nous avons donc sélectionné une carte Odroid C1+, figure 3.10, qui est plus stable et entièrement fonctionnelle malgré la perte de quelques fonctionnalités. L'article plus récent [92] a conforté et détaillé les performances de certaines de ces cartes. La carte Odroid C1+ se détache clairement comme la carte ayant le meilleur ratio qualité, performance sur prix de sa catégorie. Lors de phase de conception de notre plateforme, un choix s'est posé : utiliser une seule carte pour l'autopilote et les fonctions haut niveau ou utiliser deux cartes séparées ? Un premier choix avait été porté sur le portage Linux de l'autopilote avec une carte Erlebrain (Beaglebone black + shield capteurs) illustrée figure 3.11. Le matériel était intéressant du fait de pouvoir utiliser une carte Linux pour l'autopilote et rajouter des capteurs sans contrainte de puissance ou connectique. Encore une fois, plusieurs problèmes sont apparus et toutes les fonctionnalités (et la stabilité et fiabilité) d'ArduPilot sur carte Pixhawk n'étaient pas présentes. Nous sommes donc retournés vers la carte Pixhawk pleinement fonctionnelle pour l'autopilote et donc vers une séparation de l'autopilote et des fonctions haut niveau sur deux cartes différentes.

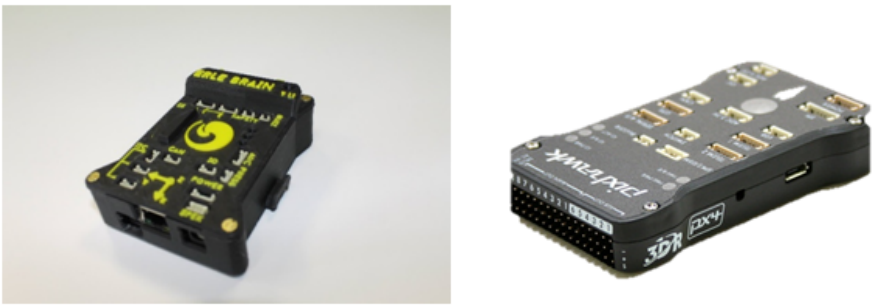


FIGURE 3.11 – Carte Erle-Brain et Pixhawk

Bien que les deux solutions étaient viables, la solution avec une carte Pixhawk moins puissante, mais plus fiable a amené un aspect sécuritaire puisque c'est sur cette carte que le code ArduPilot est activement utilisé par toute la communauté de l'autopilote ce qui en fait un système testé et fiable. Ce choix est représentatif du challenge de conception de plateforme robotique puisqu'un compromis doit être trouvé entre avancé technologique, fiabilité et usage.

Comme expliqué précédemment, le rôle de cette carte embarquée est d'être le « cerveau du drone » et d'utiliser ROS afin gérer les fonctions haut niveau du robot. Si toutes ces cartes présentent des performances importantes pour un petit prix et une petite taille, elles sont cependant mal adaptées à un usage pour la robotique ou elles doivent contrôler plusieurs capteurs ou générer plusieurs signaux simultanément. En effet, les tests de cartes montrent leur faiblesse pour la génération de signaux ou du respect du timing de certains protocoles comme l'I2C. Par exemple, une des cartes embarquées de référence en haut de gamme : la Nvidia Jetson , recommande l'utilisation de carte externe pour la génération de signaux PWM, [85] car le processeur n'est pas capable de gérer avec suffisamment de précision les interruptions pour créer des signaux PWM propres comme illustrés figure 3.12.

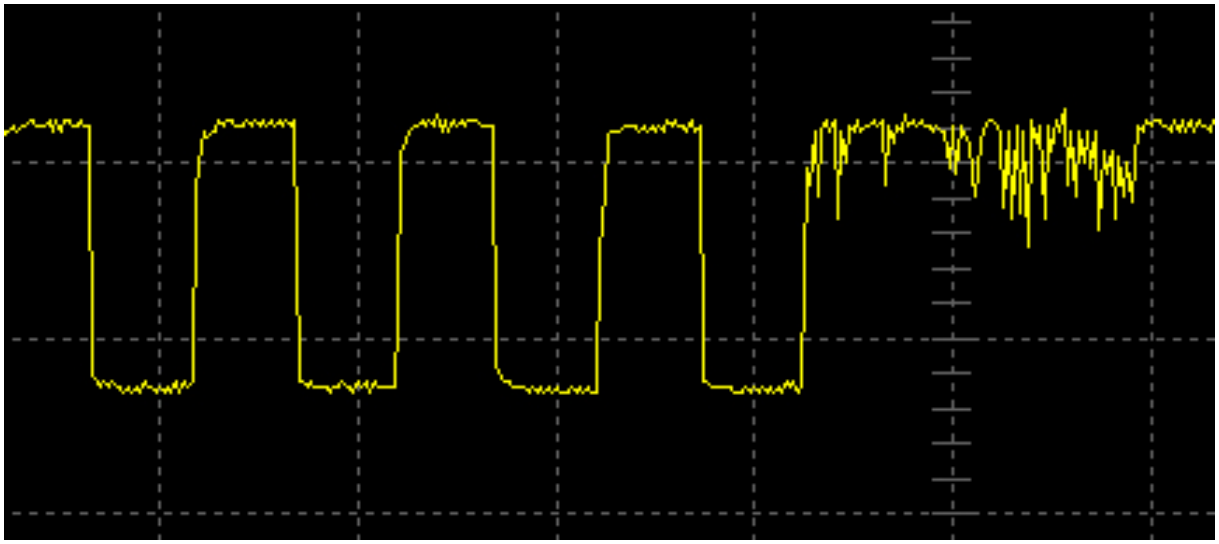


FIGURE 3.12 – Exemple de signal PWM bruité généré par une carte Linux.

Cette faiblesse matérielle rejoint le choix logiciel de séparer les parties haut niveau et bas niveau afin de pallier les problèmes de performance et de fiabilité. La figure 3.13 schématise la connexion entre la carte Odroid et la carte Pixhawk.

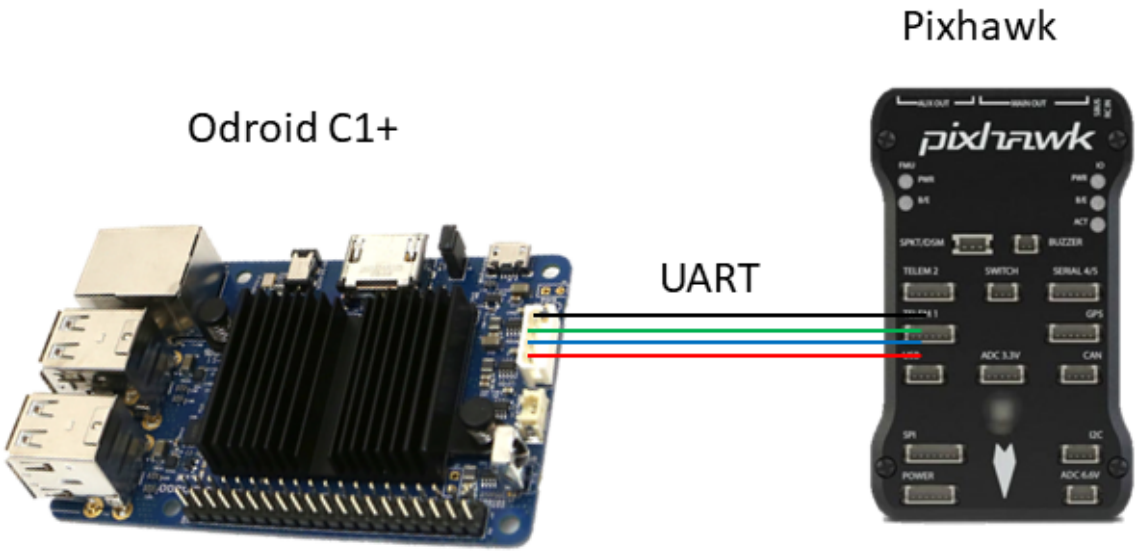


FIGURE 3.13 – Connexion entre la carte Odroid et Pixhawk

La carte Pixhawk est le support principal de l'autopilote ArduPilot. C'est une carte basée sur un processeur STM32 à 168MHz Cortex M4F CPU (256 KB RAM, 2 MBFlash) et des capteurs utilisés pour rendre la carte autonome :

- Gyroscope 3 Axes, Accéléromètre 3 Axes, Compas Magnétique 3 Axes pour l'orientation
- Baromètre pour l'altitude
- Plusieurs I/O (ADC, bus CAN, etc.)
- Carte SD pour le logging des vols
- Connectique USB pour la programmation
- LED de fonctionnement (Alimentation, santé, interface utilisateur)

Cette carte possède aussi l'avantage d'avoir un form factor petit qui lui permet de s'adapter à de nombreux châssis de robotique pour un prix avoisinant les 100€.

À cela, nous avons ajouté :

– Un GPS multi constellation et compatible DGPS afin d'avoir une référence de positionnement.

– Un compas magnétique 3 axes. Ce compas est déporté à proximité de l'antenne GPS afin de limiter les perturbations magnétiques et électromagnétiques.

– Une connectivité 433 MHz bas débit, mais longue portée afin de pouvoir envoyer des ordres d'urgence/sécurité à l'autopilote.

Ces ajouts sont schématisés figure 3.14.

Cette carte présente aussi la particularité de pouvoir être utilisé avec l'autopilote PX4 [78] qui est un autre autopilote pour drone. Produite en grand nombre, cette carte électronique a été éprouvée dans de nombreuses conditions d'utilisations et possède une excellente fiabilité. Il existe aujourd'hui en 2017, des nouvelles versions de la carte Pixhawk qui intègrent plus de capteurs, des capteurs plus récents, une redondance, etc.

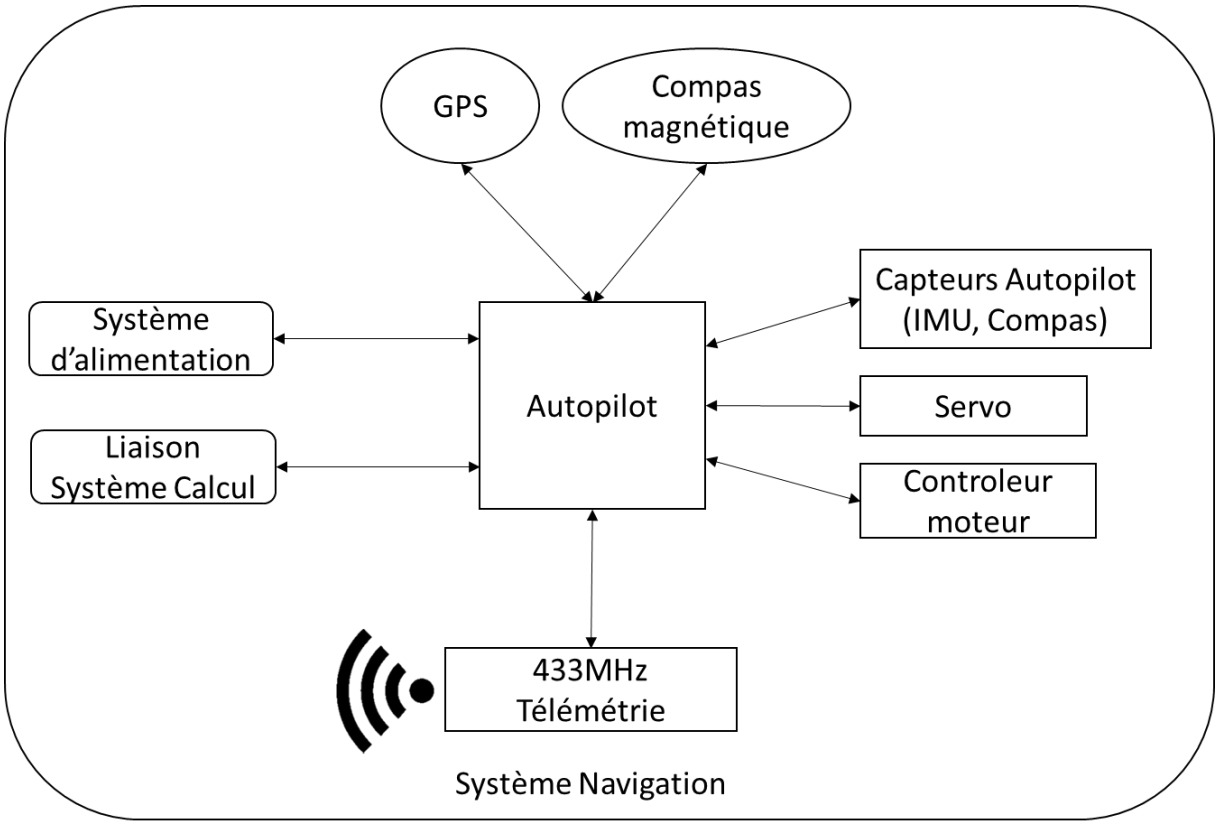


FIGURE 3.14 – Diagramme matériel de la carte Pixhawk

Communications

Les communications au sein des MRS, mais aussi avec les MRS ne sont pas obligatoires (c.f. 2.2), cependant dans le cadre de développements et pour des raisons de sécurité évidentes, il est nécessaire d'avoir au moins un moyen de communication avec le groupe de robot. Nous avons vu dans la taxonomie qu'il était possible d'utiliser des moyens de communication implicites. Cependant, l'usage des drones volants nécessite obligatoirement au moyen de communication explicite au niveau réglementaire. Dans le cadre du développement de notre plateforme, nous sommes allés plus loin que les exigences réglementaires.

Les missions devant faire passer un ou plusieurs flux vidéo depuis un ou des drones jusqu'à la station de base, seule la technologie Wifi (802.11 a/b/g) nous permettait des répondre, à budget limité, à cette contrainte. En effet, les transmetteurs analogiques sont chers et présentent de nombreuses contraintes en utilisation liées aux perturbations dont ils peuvent être sujets. Pour la technologie Wifi, de nombreux choix sont possibles. Avec notre contrainte de plateformes génériques, nous nous sommes concentrés sur des transmetteurs Wifi embarquables. Deux choix se sont présentés : utiliser une carte Wifi par USB ou utiliser un routeur. La première solution est la moins chère en termes de matériel, mais les problèmes logiciels associés à ces cartes (nécessite une compilation du driver sur Linux, perturbations électroniques, manque de documentations, etc.) et leurs faibles performances n'en font pas des bons candidats. Le choix s'est donc porté sur des routeurs longue portée et embarquable. Il en existe plusieurs et après plusieurs tests le choix s'est porté sur les transmetteurs Ubiquiti Bullet, illustré figure 3.15. Ces petits routeurs sont compatibles avec les normes 802.11 a/b/g (voir n ou ac pour les dernières versions) et possèdent, avec le récepteur associé, des portées supérieures à 100 m avec un débit suffisant pour transmettre des vidéos avec une faible latence (<100ms). La gestion des transmissions est un sujet de recherche très actif en robotique : gestion de l'énergie, maintien de la communication, trajets de données optimisés, etc. Ces transmetteurs disposent de simple API afin être reconfigurés pour réduire leur consommation par exemple. Dans notre cas, afin de pouvoir garantir une liaison permanente de chaque drone avec la base sol, nous avons choisi de les utiliser en mode répéteur Point d'Accès. Cela nous permet de déployer un réseau de type "Mesh" simplement, les routeurs se chargeant d'effectuer la retransmission si besoin. Le router possédant aussi une mesure du RSSI, cela nous a permis aussi de savoir ou placer les drones afin de maximiser les communications dans l'essaim. Ce type de transmetteurs est intéressant académiquement, car il permet de travailler sur les aspects communication tout en gardant le système simple. Du fait la bande passante disponible et de légèreté du protocole MAVLink, le Wifi peut aussi être utilisé pour envoyer des ordres directement à l'autopilote. Enfin, le choix s'est porté sur la version 5GHz qui est moins encombrée et sujette à des interférences.

FIGURE 3.15 – Ubiquiti Bullet M5

Afin d'avoir une redondance des communications, nous avons aussi ajouté deux autres liens radio. En effet, notre essaim de drones devait pouvoir à tout moment faire communiquer les drones entre eux pour des échanges d'informations, mais aussi communiquer avec la station de base. Le premier est un lien de télémétrie à 433MHz branché uniquement sur l'autopilote et à destination de la base sol. C'est un lien de sécurité. En effet, ce lien envoi très peu d'information (position, état) du drone, mais permet de « sauter » la carte embarque pour donner des ordres directement à l'autopilote pour effectuer des manœuvres de sécurité voire d'urgence (arrêt moteur, etc.). Le lien de télémétrie utilise le protocole MAVLink pour l'envoi des informations. À l'usage, ce lien ne s'est pas montré très fiable, car il a fait face à des problèmes de portée. Cependant, il existe aujourd'hui en 2017 des transmetteurs en 868MHz capable de bien meilleure performance et de la gestion de transmission multipoint. Ces transmetteurs n'ont pas eu le temps d'être testés dans le cadre du projet DAISIE. Le second lien de télémétrie utilise aussi le protocole MAVLink, mais à travers le réseau Mesh DigiMesh de chez DIGI. Ce réseau est un réseau Mesh auto-configuré qui permet au drone de s'échanger des informations, mais aussi de garder un lien avec la base sol. Le choix s'est porté sur cette technologie, car elle était simple à mettre en œuvre et avec suffisamment de bande passante et de portée pour faire fonctionner un essaim de 15 drones. En effet, le protocole employé par ces transmetteurs permet de partager la bande passant intelligemment afin que plusieurs modules puissent communiqué en même temps sans risque de collision de frames radio.

L'utilisation de 3 moyens de communication sur des fréquences différentes permet une redondance fiabilité avec tous les drones de l'essaim. En effet, une de problématique des MRS est l'aspect sécuritaire. Dans le cas de la plateforme présentée ici, nous avons maximisé les technologies : liaison sur différente fréquence, communication point-à-point ou mesh, etc., afin de parer à d'éventuels problèmes et garantir de pouvoir garder le contrôle de l'essaim même en cas de défaillance d'un système de communication. La figure 3.16 donne une vue d'ensemble de la plateforme matérielle.

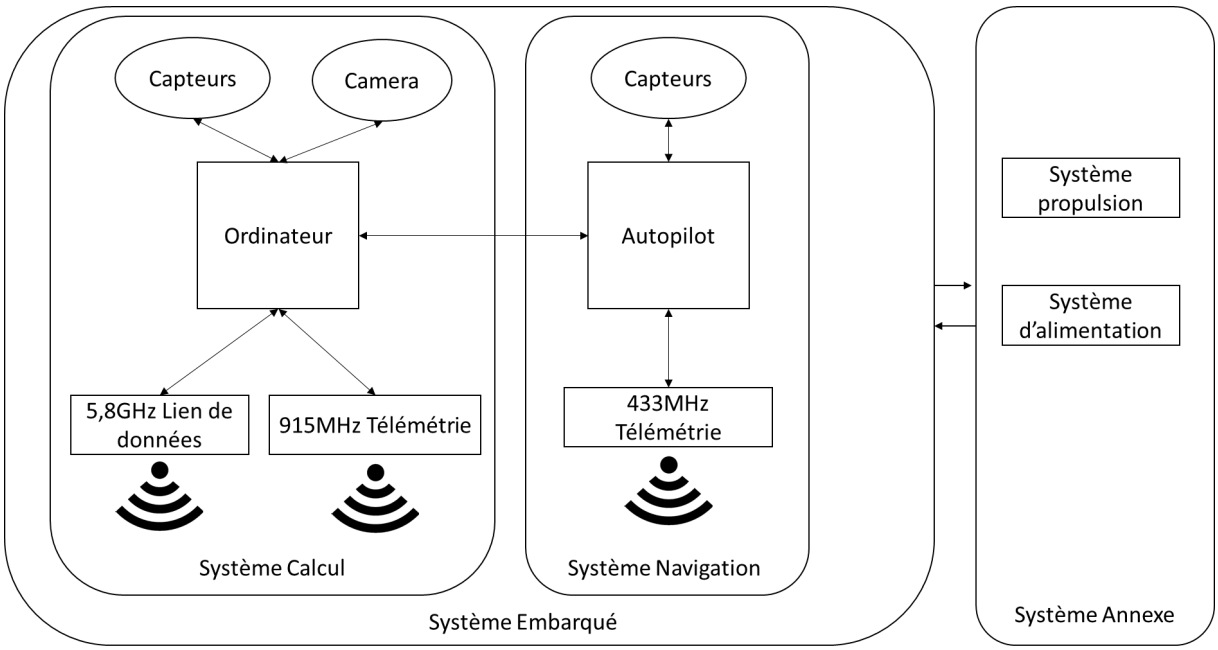


FIGURE 3.16 – Diagramme matériel de la plateforme

Autres capteurs

La plateforme ainsi conçue ne présente pas un intérêt industriel majeur alors qu'elle est riche académiquement puisqu'elle permet de travailler sur la plupart des domaines des MRS. Cependant, l'intérêt industriel vient d'un capteur mission.

Dans le cadre du projet DAISIE, les missions de patrouille nécessitaient un capteur optique. Un choix de caméra simple a été fait, en effet l'utilisation des caméras et de vidéo est généralement source de problèmes en robotique (problème driver, perturbations, etc.). C'est donc naturellement que nous avons sélectionné la caméra Logitech C920 qui est une caméra USB bas coût bien connu dans le monde académique et qui possède des performances suffisantes dans le cadre d'expérimentation. Il est à noter que cette caméra est capable de réaliser directement l'encodage vidéo H264. Cela nous permet de récupérer un flux vidéo déjà encodé et léger que l'on peut traiter rapidement avec une librairie comme OPENCV et retransmettre directement pour être affiché sur la base sol. En limitant ainsi le développement de driver, et de capture vidéo, cette caméra permet un gain de temps pour les réalisations concrètes de l'essaim.

Enfin, les robots roulants nécessitent des capteurs de proximité afin d'éviter les collisions. Le prix des solutions LIDAR n'étant pas adapté à un essaim bas coût, nous avons utilisé un réseau de sonars sur chaque robot malgré les limites de cette technologie : faux écho, faible portée, perturbations.

Conclusion

Il a été présenté ici une sélection matérielle devant servir de base aux robots dans un MRS générique. En effet, ces choix présentent chacun des avantages et inconvénients, mais aussi les raisonnements et les justifications des choix technologiques faits sur la base de tests dont l'évolution est illustrée figures 3.17 et 3.18. Durant les années de thèse, les choix initiaux ont largement évolué pour certaines parties. Cependant, la bonne conception du système a permis une actualisation rapide et simple des parties défaillantes ou obsolètes ce qui était un des buts de la plateforme : être générique.

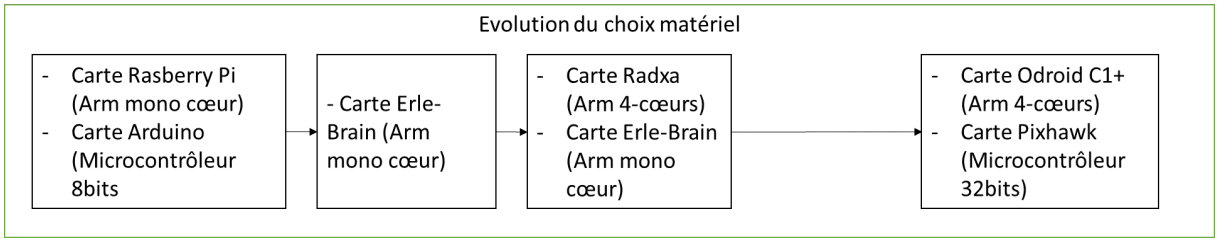


FIGURE 3.17 – Évolution du choix de matériel de 2013 à 2017

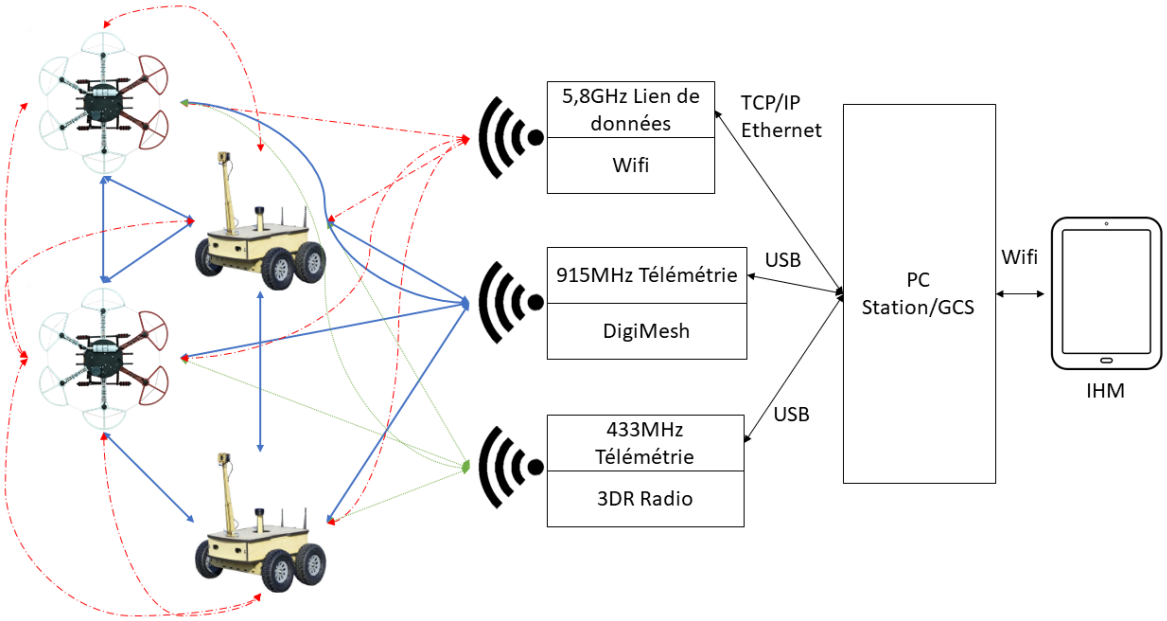


FIGURE 3.18 – Schéma global des communications dans DAISIE

Il n'a été présenté ici que les parties essentielles des robots, d'autres choix importants

comme la gestion des connectiques (pouvant être des perturbations électromagnétiques), de l'énergie n'ont pas été présentés (batterie, système de filtrage, etc.) et la mécanique, car ils relèvent de l'ingénierie et ne sont pas en de domaine de recherche du projet.

3.3 Résultats obtenus

Si l'architecture matérielle ou logicielle ne semble plus innovante en 2018, elle a présenté une innovation de conception au démarrage du projet puisque ce type de plateforme n'existait pas et aucune solution du commerce n'était adaptée aux essaims. On peut donc observer de nos jours d'autres plateformes semblables [114], [69] du fait de la convergence vers des solutions fonctionnelles et des travaux effectués et publiés en temps qu'open source dans le cadre de cette thèse. Cette plateforme est largement basée sur du logiciel portable notamment grâce à ROS. Cela permet aux développements effectués de ne pas être complément dépendant du matériel et donc d'être portable dans le temps en changement par exemple l'ordinateur. Ces choix ont résulté de plusieurs tests de matériel et d'un suivi des innovations dans le domaine afin de cibler le meilleur matériel en fin de projet.

Avec ce matériel, nous avons construit une plateforme pouvant embarquer suffisamment d'intelligence pour se déplacer de façon quasi autonome et répondre à des problématiques d'essaims (déplacement en groupe, gestion des communications, etc.). Les travaux visant à réaliser les contrôles de l'essaim par phéromones ont pu démarrer pendant la construction de la plateforme. En effet, l'utilisation de ROS permet de se servir des simulateurs de robotique de référence comment GAZEBO et STAGE. À cela s'ajoute que l'autopilote ArduPilot possède aussi un simulateur SITL (Software In The Loop).

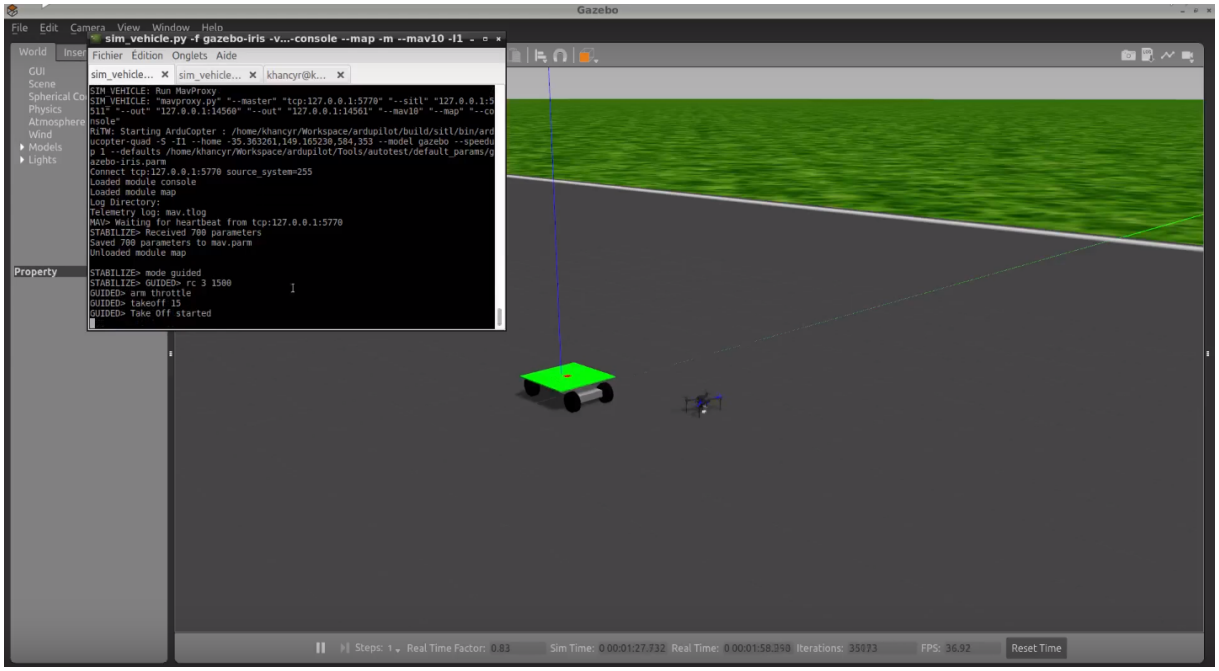


FIGURE 3.19 – Test d'un robot roulant et volant dans GAZEBO

Ainsi on peut réaliser des simulations avec le code complet qui sera ensuite embarqué directement sur les drones comme présentés figure 3.19. Par manque de temps, le portage complet de la plateforme dans le simulateur GAZEBO n'a pu être testé. Ces solutions permettent un gain important de temps en limitant les tests réels en faisant utilisant des simulations pour tester le comportement de l'essaim de façon sécurisée et rapide avant même d'avoir fini la plateforme. Ce processus permet aussi une aide sur le design de la plateforme en montrant une partie des ressources et performances que doit posséder la plateforme. Du fait de l'utilisation d'outil et bibliothèque logiciels standard, le changement de composant en cours de projet n'entraîne pas une remise à zéro des développements.

Ce processus permet aussi une aide sur le design de la plateforme en caractérisant une partie des ressources et des performances que doit posséder la plateforme. L'usage du simulateur SITL permet de faire fonctionner le code ArduPilot, sans besoin de matériel spécifique, avec des données de capteurs simulés provenant de modèle issu des capteurs réels ou d'autres simulateurs comme Gazebo. Ainsi il est possible d'observer le comportement du drone comme son déplacement, mais aussi la puissance théoriquement transmise aux moteurs par exemple. Dans le projet Daisie, SITL a permis de :

– Tester et comprendre les modes de fonctionnement de l'autopilote (phase décollage, atterrissage, suivis de waypoints, etc.).

– Simuler le bon fonctionnement des algorithmes du projet avant les tests réels.

– Tester certains composants réels dans le simulateur comme les capteurs ultrasons, le transfert de carte de phéromone, etc.

– Tester sur les cartes réelles des algorithmes de reconnaissance d'objets et d'évitement dans une simulation.

L'exemple le plus parlant sur l'intérêt de l'usage SITL est le développement du système anticollision. Celui-ci repose sur des capteurs ultrasons qui lors de la détection d'un obstacle doit procéder à une manœuvre du robot terrestre. Afin de tester en sécurité le bon fonctionnement des capteurs et l'algorithme d'anticollision, SITL a été installé sur les robots afin de simuler les déplacements du robot. La figure 3.20 schématise l'utilisation de SITL avec du matériel réel.

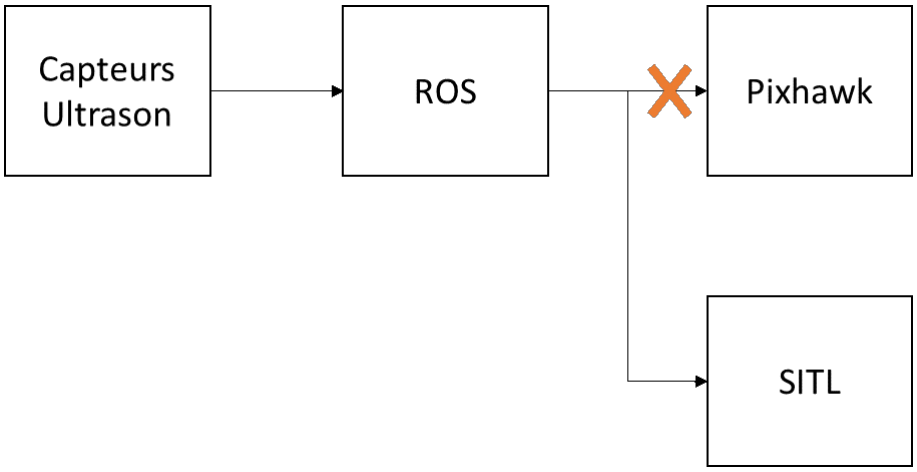


FIGURE 3.20 – Schéma d'utilisation de SITL avec les robots réels

Ainsi la prise en compte de la détection et les temps de réaction de l'algorithme ont pu être observés dans SITL avant d'être vérifiés en utilisant la plateforme complète. Ce mode de fonctionnement hybride, combinant matériel réel, code réel, et données simulées, présente un gain de temps dans le développement de plateforme robotique, mais aussi de sécurité à la fois pour les personnels, mais aussi pour les plateformes.

Une des grandes difficultés des MRS est leur IHM. En effet, il est difficile pour des opérateurs humains de comprendre les différentes données que renvoie le groupe et de le gérer efficacement. Faisant suite aux travaux du projet SUSIE, l'IHM de l'essaim de DAISIE est simplifié au maximum afin de n'afficher que la position et l'état des drones. Cette configuration permet à l'opérateur de ne pas interférer avec l'auto organisation de l'essaim. Il est aussi possible d'utiliser les IHM (Mission Planner, QGC, etc.) du projet ArduPilot afin de contrôler individuellement chacun des robots. De plus, l'utilisation de ROS permet un logging efficace des informations des capteurs et des communications afin de pouvoir faire une analyse post mission du comportement du groupe. Le logging de l'autopilote permettant de vérifier le bon comportant physique des agents.

En résumé nous avons développé une plateforme répondant à ces critères :

- Plateforme de drone bas coût et ouverte : carte Linux, transmetteur COTS, auto-pilote ArduPilot.
- Plateforme commune pour des robots roulants et des drones volants.
- Facilité de mise à jour et d'amélioration dans le temps.
- Utilisable pour un grand nombre de missions de MRS : exploration, surveillance, etc.
- Suite logicielle complète, documentée et mise à jour.
- Suite de tests et de simulation.
- Un système de logging.
- Des IHMs

La fin du projet a été contrariée par les difficultés de l'entreprise partenaire R&DTECH et il n'a pas été possible de prendre part aux tests finaux du projet DAISIE et donc présenter les résultats du projet. Cependant, le démonstrateur a été mis en oeuvre ce qui valide les choix effectués précédemment. Les figures 3.21, 3.22, 3.23 présent un aperçu du fonctionnement de l'essaim.

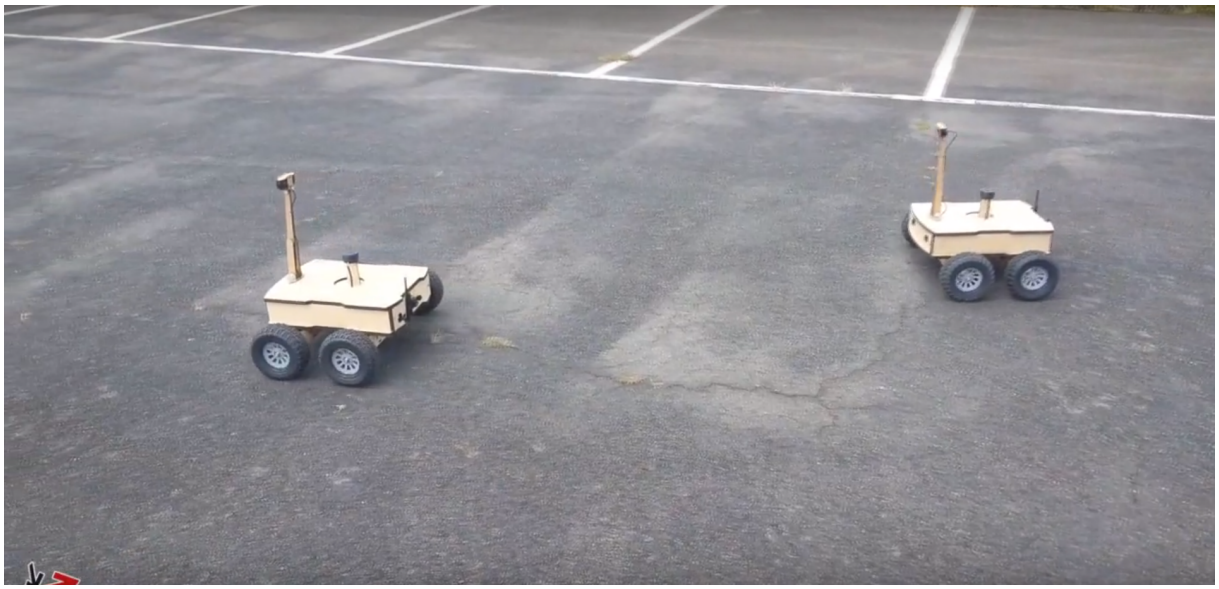


FIGURE 3.21 – Test de deux robots roulants

FIGURE 3.22 – Essaim Daisie

FIGURE 3.23 – IHM montrant le flux vidéo d'un robot film un autre

La création de cette plateforme permet de mettre en œuvre un essaim réel pouvant démontrer la plupart des caractéristiques et comportements des MRS. Cette plateforme est prête pour permettre de tester en condition réelle des algorithmes de déplacement par phéromones et de juger de leur efficacité et intérêt pour une utilisation réelle. Ces résultats pourront être mis en comparaison avec les résultats théoriques. Ce type de plateforme doit être mis en avant afin de prouver qu'il est possible de réaliser des MRS réels avec des coups limités. L'utilisation de ressources open source comme ROS ou ArduPilot permet à la fois de réduire les coûts de création d'un MRS, mais permet aussi de renforcer les collaborations entre différent laboratoire et industrielle. En effet, la plateforme développée ici est générique et démontre l'usage d'agents hétérogènes. Nous pourrions ajouter à notre essaim, les robots d'autres laboratoires utilisant les mêmes caractéristiques pour étendre les capacités de l'essaim, mais aussi vérifier les comportements réels à plus grand nombre, et cela sans grande difficulté puisque les protocoles de communication sont standardisés avec ROS et MAVLink. À cela s'ajoute la possibilité à d'autres équipes de reproduire nos résultats ou les mettre en concurrence avec les leurs, choses qui manquent énormément dans le domaine de MRS.

Chapitre 4

Amélioration d'un autopilote et d'un simulateur pour être compatible au développement de MRS

La création de l'essaim de l'ASMAT DAISIE a mis en avant plusieurs difficultés à la fois dans la création d'une plateforme, mais aussi dans la simulation des différentes parties de la démonstration. Malgré l'utilisation d'outils de références, ceux-ci se sont révélés insuffisant techniquement et trop complexe à mettre en oeuvre dans le cadre d'un essaim retardant ainsi la mise en application d'algorithmes de déplacement par phéromones, qui étaient le coeur du projet, au profit de la résolution de problématique de robotique. Le chapitre suivant présente les principales contributions techniques effectuées afin de mettre en place une plateforme de robotique adaptée aux travaux sur les MRS en permettant à la fois des simulations simples et complexes et un déploiement réel simplifié.

4.1 Autopilote générique pour rover

Afin de pouvoir utiliser des robots mobiles en autonomie en groupe, il est nécessaire d'avoir des solutions fonctionnelles individuellement, et pour cela, plusieurs solutions existent déjà. Il est possible de construire une plateforme basée sur ROS, mais avec les défauts précédemment cités, ou utiliser une solution intégrant une plateforme matérielle et logicielle. Cette section présentera une analyse de plusieurs plateformes de robots roulants, soulignant pourquoi elles ne sont pas adaptées aux MRS, puis présentera en détail l'autopilote open-source ArduPilot et les modifications qui lui on était apporté afin d'en faire une plateforme utilisable pour les MRS.

4.1.1 Solutions robotiques existantes

Il est difficile de trouver une plateforme robotique roulante capable de fonctionner en extérieur, à bas coût et étant équipé pour fonctionner en groupe. Si des solutions bas

coûts basés sur des cartes comme les Rasberry Pi et ROS sont disponibles et nombreux sur le marché, elles se révèlent trop limitées logiciellement et matériellement pour être véritablement utilisables sans lourd investissement de développement. Il existe cependant d'autres solutions plus complètes, issues de projets académiques, pouvant servir de base aux projets sur les MRS. Ces plateformes diffèrent de celles présentées dans la section 3.2 du fait qu'elles proposent une suite matérielle et logicielle permettant un fonctionnement en autonomie en intérieur comme en extérieur, ce qui est la base nécessaire aux travaux des MRS, et reste relativement bas coût ($< 10.000€$).

- Racecar : C'est un projet de rover de type voiture RC open source, équipé d'une carte Nvidia Jetson TX1, d'une IMU, d'un lidar, de caméra et d'un contrôleur moteur intelligent. La solution logicielle propose un code de contrôle utilisant la géométrie d'Ackermann et quelques codes d'exemples pour le déplacement autonome. Le tout est basé sur ROS. Le projet reste restreint aux démonstrations proposées et peu de contributions extérieures sont à noter sur ce projet.

- AutoRally : La plateforme est chère ($>10.000€$), mais probablement sur dimensionnée par rapport aux objectifs du projet. La solution est encore un rover mais basé sur un ordinateur classique, d'une IMU, d'un GPS RTK, de caméra et d'un ensemble de transmetteurs sans fils. La solution logicielle est complète : elle propose un code de contrôle, une gestion de l'état du robot, une station de contrôle, le tout étant basé sur ROS. Le projet est suffisamment générique pour être utilisé dans d'autres applications que celles prévues initialement, cependant le prix de la solution semble restreindre les contributions. On notera une contribution en janvier 2018 visant un usage en groupe de la plateforme.

- Swarmathon : La plateforme est plus traditionnelle, un modèle de type tank, un ordinateur pour l'intelligence embarquée, une IMU, une caméra, et des sonars composent la plateforme. La solution logicielle repose sur ROS et propose en plus de nombreux exemples et codes de contrôle une solution d'utilisation en groupe et une station de contrôle. Malgré la qualité de la plateforme, il ne semble pas y avoir de communauté créée autour du projet.

Tableau 4.1 – Résumé des caractéristiques attendues pour un usage industriel de MRS

Plateforme	racecar	autorally	Swarmathon
Couts < 10.000€	OUI	NON	OUI
Compatibilité ROS	OUI	OUI	OUI
Fonctionnement sans ROS	NON	NON	NON
Amélioration matérielle possible	NON	NON	NON
Systèmes de sécurité matérielle	NON	OUI	OUI
Systèmes de sécurité logiciel	NON	NON	NON
IHM	NON	NON	OUI
Simulation	OUI	OUI	OUI

Dans le tableau précédent, les systèmes de sécurité sont les systèmes matériels ou logiciels permettant d'éviter les dommages sur le système des comportements dégradés, et garantissant un contrôle permanent sur le robot : capteur de batterie, lien permettant la coupure de l'alimentation des moteurs, géofencing, etc.

Les travaux sur la robotique mobiles sont nombreux aujourd'hui et les tests en conditions réelles ont déjà commencé partout dans le monde. Du fait de l'usage de ces robots avec des hommes à proximité, il est important de multiplier les tests réels dans le maximum de situation possible. Les trois projets présentés ici proposent des solutions pour l'étude de certaines composantes des véhicules autonomes. Cependant, de par leur construction aucun de ces projets n'est assez générique pour répondre à la diversité des problématiques des véhicules autonomes en groupes.

Certes, chacun est basé sur ROS et open-source mais aucune communauté ne s'est formée pour étendre ces projets au-delà de leurs objectifs initiaux, alors que les plateformes le permettraient. L'existence d'une telle communauté est indispensable et le travail justement y contribue. Ces problématiques ont déjà été abordées dans le Chap. 2.

4.1.2 ArduPilot : présentation

Les drones actuels ont de plus en plus de capacités grâce aux autopilotes qui les dirigent. Cependant, le développement d'un autopilote complet nécessite de très bonnes connaissances en électronique et logiciel embarqué en plus de la dynamique des robots utilisés (multirotors, avion, rover, etc.). C'est une tâche longue et fastidieuse. Il est donc préférable de se baser sur une solution existante. La plateforme proposée dans le cadre de cette thèse a été construite sur la base d'ArduPilot. Si l'autopilote est bien connu dans le domaine des drones volants, son usage en groupe et avec des robots terrestres l'est beaucoup moins. Il a l'avantage d'être bas coût, open source et dispose d'une

forte communauté et d'industriels partenaires : `http://ardupilot.org/copter/docs/common-partners.html`.

L'état de l'art présenté dans les chapitres précédents à montrer que si ROS est devenu une référence dans le domaine de la robotique, il n'existe toujours pas de plateforme robotique générique à prix accessible (c.-à-d. $< 10.000€$). C'est un problème important puisque les nouveaux développements vont nécessiter du temps sur la plateforme aux dépens du travail sur des questions plus académiques (génération de trajectoire, contrôle de swarm, etc.). Cette partie présente des contributions au projet ArduPilot afin d'en faire une solution capable de réduire le temps de construction d'une plateforme en fournissant un autopilote robuste, maîtrisé et sécurisé pour tous les types de véhicules, de la même manière que ROS permet de réduire le temps de développement logiciel pour les projets de robotique.

Contrairement à ROS, ArduPilot n'est pas un middleware pour la robotique, mais un autopilote pour micro véhicule autonome. Il supporte donc les 3 principaux types robots : les roulants, les volants et les navigants. Le code est basé sur une architecture monolithique (c.-à-d. un seul programme) et uni-langage (C++) et est disponible sous licence GLPv3. L'architecture logicielle diffère largement de celle de ROS puisqu'elle est basée sur le modèle des architectures d'informatique embarquée qui tiennent compte des problèmes de ressource limitée de la criticité des opérations, des contraintes temps réels et d'un impératif de fiabilité. ROS étant basé sur l'architecture logicielle des ordinateurs de bureau. Le développement initial d'ArduPilot a été fait par des amateurs et a évolué aujourd'hui en un autopilote reconnu pour sa fiabilité dans le monde par les acteurs académiques, industriels et étatiques tout en gardant une forte communauté de passionné active. Afin de viser un niveau de qualité proche de la certification pour le transport de personne (les Chinois le font déjà avec ArduPilot cf.Ehang), une équipe d'une vingtaine de développeurs gère le projet et s'astreint à un processus d'analyse, de corrections, et de tests des contributions.

Le processus de développement d'ArduPilot est classique pour les développements informatiques open source. Il s'appuie sur un système de vérification de code ouvert qui suit le déroulement présenté sur la figure 4.1

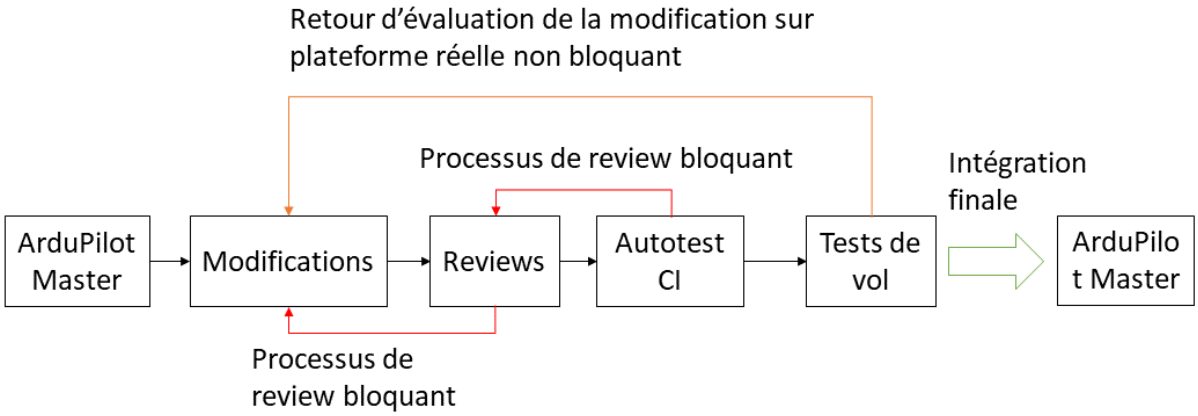


FIGURE 4.1 – Flux de travail d'ArduPilot

ArduPilot Master : Branche principale du code de l'autopilote open-source ArduPilot `https://github.com/ArduPilot/ardupilot`. C'est la branche qui reçoit les contributions et améliorations. La branche Master est ensuite forker (copier) sur une revision fixée afin de bloquer une version.

Modifications : Demande de modification d'une partie de l'autopilote pour ajouter des fonctionnalités ou corriger un bug ou un comportement. Le projet étant open-source toute personne est capable de proposer des modifications dans le code. Chaque modification doit être expliquée et justifiée.

Reviews : Analyse du code de la modification par au moins une personne autre que la personne ayant fait la modification. Le reviewer, personne qui réalise la review, analyse la viabilité des modifications, les implications de la modification et si besoin demande des corrections. Toute personne est autorisée à faire des reviews, celles-ci étant ouvertes. La gestion des reviews est réalisée grâce au logiciel Github qui est une interface web pour le gestionnaire de version Git. Github assure la traçabilité des reviews. Git assure la traçabilité des modifications et la gestion des versions des fichiers. Seule l'équipe de développeurs du projet est capable de valider une demande de modification. Seul le responsable du véhicule impacté par les modifications est autorisé à intégrer les modifications dans ArduPilot Master.

Gestions des retours utilisateurs : l'interface Github permet à tout utilisateur de publier un cas d'erreur qu'il aurait rencontré ou une amélioration à faire sur le code qui peut ensuite être étudié, discuté et faire l'objet d'une modification par la suite. Cela permet de mutualiser les retours d'expérience et d'avoir une traçabilité des problèmes rencontrés.

Autotest CI : Réalisation en simulation d'un jeu de tests critiques. Ceux-ci sont automatiquement déclenchés sur toute demande de modification et relancés lorsque la demande est modifiée. Chaque test produit un résultat de validation et un log complet de la simulation est produit pour analyse ultérieure si besoin.

Tests de vol : Réalisation de vol de contrôle par des utilisateurs volontaires. Ceux-ci ont pour tâche de valider les modifications proposées.

Cette équipe n'est pas rémunérée et travaille largement pour le plaisir d'aider et de proposer une solution innovante et fonctionnelle gratuitement au plus grand nombre. Cette ouverture a permis l'émergence d'une grande communauté d'utilisateurs qui par sa diversité, et ses tests permet la notification de bugs, la correction de ceux-ci, et les avancées dans l'usage des drones que nous connaissons aujourd'hui. L'aspect open source n'est pas contradictoire avec le développement industriel. Bien au contraire, il est une force qui permet de regrouper le plus grand nombre avec des objectifs communs et avancer ensemble dans la même direction chacun avec son marché. Enfin, la plateforme de l'autopilote ArduPilot est utilisable avec ROS grâce à de nombreuses librairies dont notamment Mavros [77]. Cette application ROS permet de le faire le pont entre le protocole MAVLink utilisé par ArduPilot et ROS.

Afin de pouvoir être utilisée par un grand nombre de véhicules différents l'architecture d'ArduPilot, schématisé en figure 4.2, est divisée en 3 parties :

- Code véhicule : C'est le code principal de chaque type de véhicule. Cette partie implémente les différents algorithmes de gestion de l'état et des déplacements des véhicules.

- Librairies partagées : Ces librairies sont tout ce qui peut être partagé entre les différents types de robots : capteurs, estimateurs d'attitude et position, contrôleurs PID, etc.

- HAL (Hardware Abstraction Layer) : C'est la couche responsable de la portabilité d'ArduPilot sur différentes plateformes matérielles définissant les capteurs utilisés, les bus de terrain, les IO, etc.

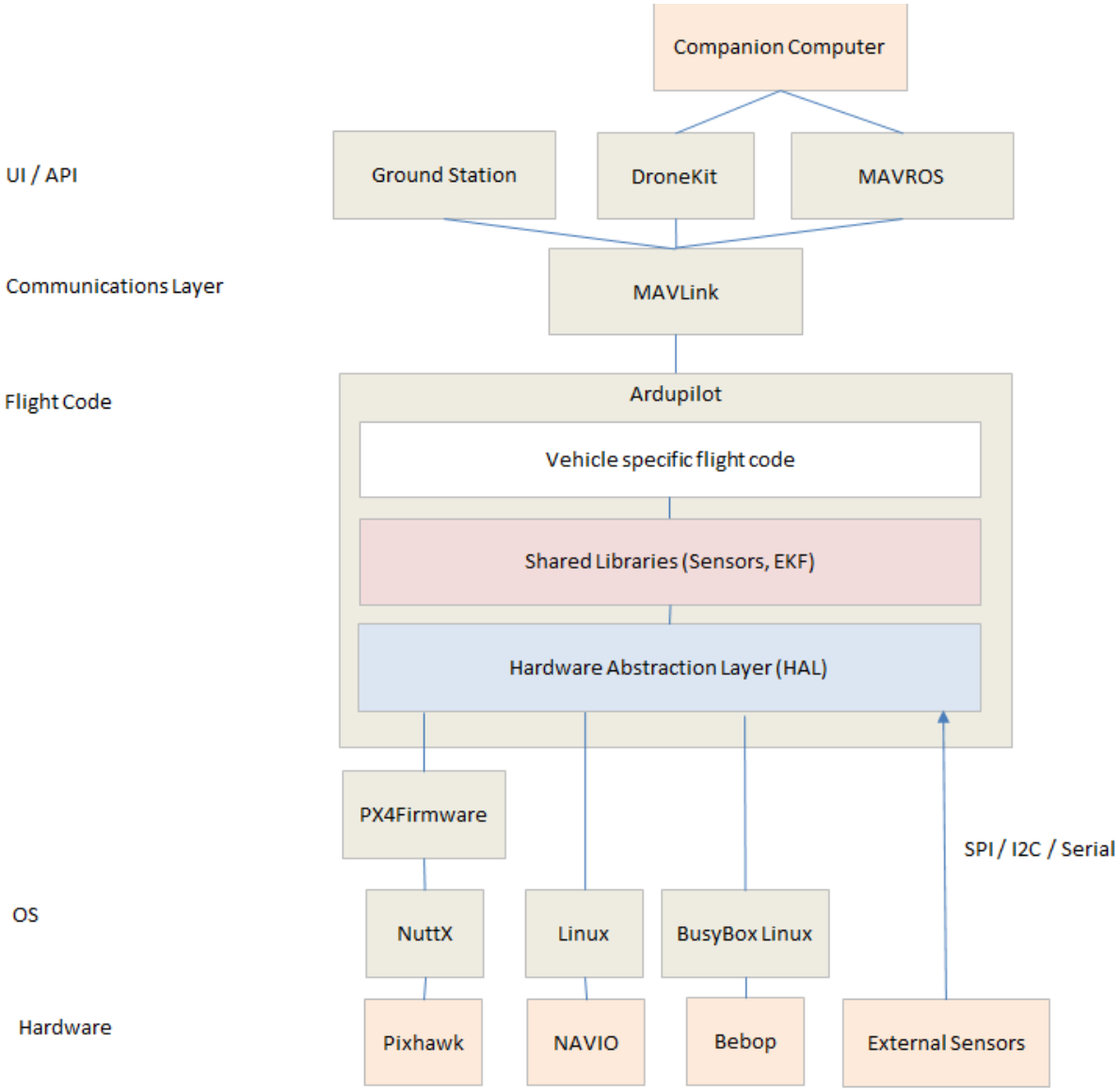


FIGURE 4.2 – Schéma d'architecture d'ArduPilot sous licence CC BY-SA 3.0

Le projet ArduPilot est divisé en plusieurs sous-projets correspondant aux différents types de véhicules utilisables :

– ArduCopter ou ArduPilot : Copter : c'est la version pour multirotors et hélicoptère d'ArduPilot.

– ArduPlane ou ArduPilot : Plane : c'est la version pour les voilures fixes ou VTOL (vertical take-off and landing), c'est dire des avions pouvant effectuer des décollages verticaux.

- ArduRover ou ArduPilot : Rover : c'est la version pour les rovers, voitures et autre véhicule terrestre, mais aussi les véhicules de surface comme les bateaux.
- ArduSub ou ArduPilot : Sub : c'est la version pour les sous-marins du projet.
- AntennaTracker : c'est le sous projet pour un traquer de véhicule afin d'orienter les antennes du sol dans sa direction et maximiser la qualité de la réception radio.

De par son architecture actuelle, ArduPilot est encore limité à un usage en extérieur. En effet, le jeu de capteurs nécessaire au bon fonctionnement de la plateforme est une IMU, un GPS, un compas magnétique. Cela restreint son utilisation en extérieur. Cette problématique est liée au cœur d'ArduPilot qui est à la fois sa force et sa faiblesse : l'algorithme d'estimation d'état. Dans les premières versions de la plateforme, des filtres complémentaires [101] étaient utilisés. Avec l'arrivée de nouvelles plateformes matérielles plus puissantes (micro contrôleurs STM32F4xx), des algorithmes d'estimations d'états plus avancés ont été développés pour ArduPilot. La solution actuelle repose sur un EKF (Extended Kalman Filter) à 24 états (de base) qui incluent la position, la vitesse et l'orientation angulaires du véhicule, en plus des biais des IMU et de l'état du champ magnétique par exemple. L'EKF d'ArduPilot est une implémentation classique d'un EKF en termes d'équation de prédiction et de mise à jour des matrices de covariances pour le calcul des gains et la mise à jour des états. Au niveau des véhicules, l'algorithme de l'EKF d'ArduPilot fonctionne de la même manière qu'un filtre de Kalman traditionnel. Les données des IMU sont lues périodiquement et utilisées pour la prédiction des variables d'états, qui incluent la position, vitesse et l'orientation du véhicule. De plus, la matrice de covariance d'état est aussi prédite. Les mesures d'autres capteurs comme le GPS sont fusionnées pour corriger les états et les matrices de covariance, l'innovation (différence entre la valeur de l'état prédite et la valeur mesurée) et le gain de Kalman sont aussi calculés. L'état est ainsi mis à jour grâce au gain de Kalman et à l'innovation en utilisant les équations de mise à jour d'états standards des EKF.

Cette EKF est au cœur de solution basée sur ArduPilot est dépend malheureusement encore d'un jeu de capteur fixé. Cependant, les récents développements, notamment grâce aux travaux de cette thèse, ont permis de dépasser ces limitations et l'intégration de données d'odométrie ou d'asservissement visuel par exemple, au sein de l'EKF. Ces ajouts autorisent une utilisation d'ArduPilot sans données GPS.

Détail des avantages de la plateforme avant les contributions de thèse :

- Contrôle en repère absolu (WGS 84, NED)
- Contrôle en repère relatif (NED relatif à la position de départ, etc.)
- Contrôle en vitesse et position (ArduCopter seulement)
- Station de contrôle préexistante
- Évitement d'obstacles (basique)

– Suivis de mission préprogrammée

– Asservissement par contrôleurs internes ou externes

– Gestion des pannes

– Sécurité du logiciel et du matériel

– Mode de contrôle identique pour les différents véhicules

En dépit de ces avantages, la plateforme présentait aussi de grandes limitations, surtout sur la partie robot roulant, qui ne permettait pas son usage simple dans les MRS. À l'instar d'ArduCopter, la version rover était capable de suivre une trajectoire de point GPS préprogrammé. Cependant, l'architecture logicielle était très dépendante du code originel et ne permettait qu'un fonctionnement basique de suivis de trajectoire pour des véhicules roulants de type rover. Cette configuration est basée sur la géométrie d'Ackermann, figure 4.3, en utilisant un servo moteur pour contrôler la direction et un moteur asservis en vitesse pour la vitesse du robot. Cette solution est similaire à celle utilisée dans le projet Autorally. Si elle est fonctionnelle, cette architecture est limitante pour un usage en groupe et sur différente plateforme, car toute l'interface de navigation est basée sur la seule configuration d'Ackermann, ou le modèle des voitures. Cette dépendance importante à un seul type de contrôle empêchait l'usage correct de robot de type tank (les moteurs du côté droit et du côté gauche sont contrôlables indépendamment l'un de l'autre).

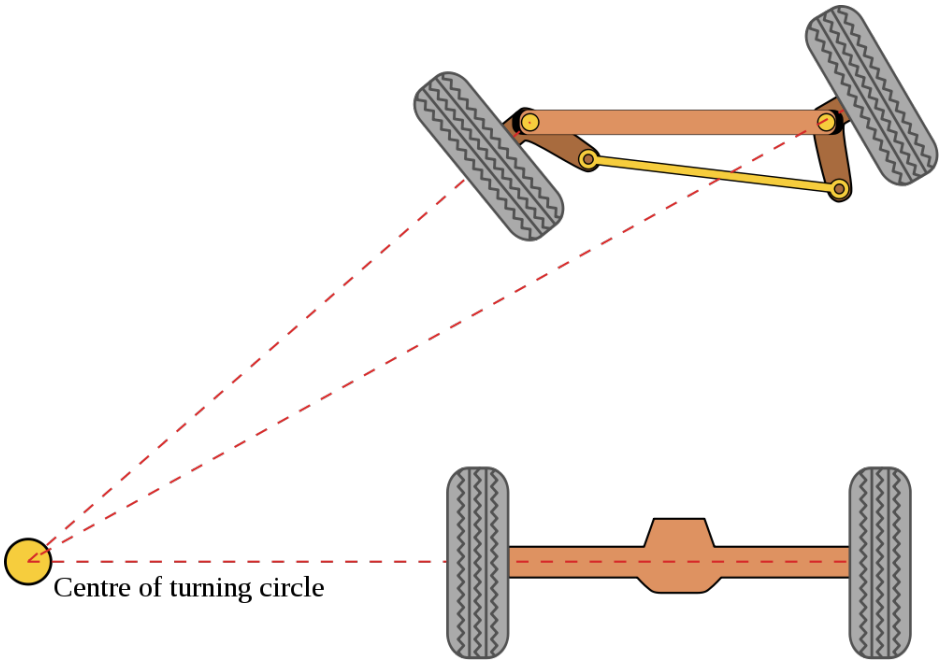


FIGURE 4.3 – Schéma de principe de la géométrie d'Ackermann. Copyright Wikipedia sous licence CC BY-SA 3.0

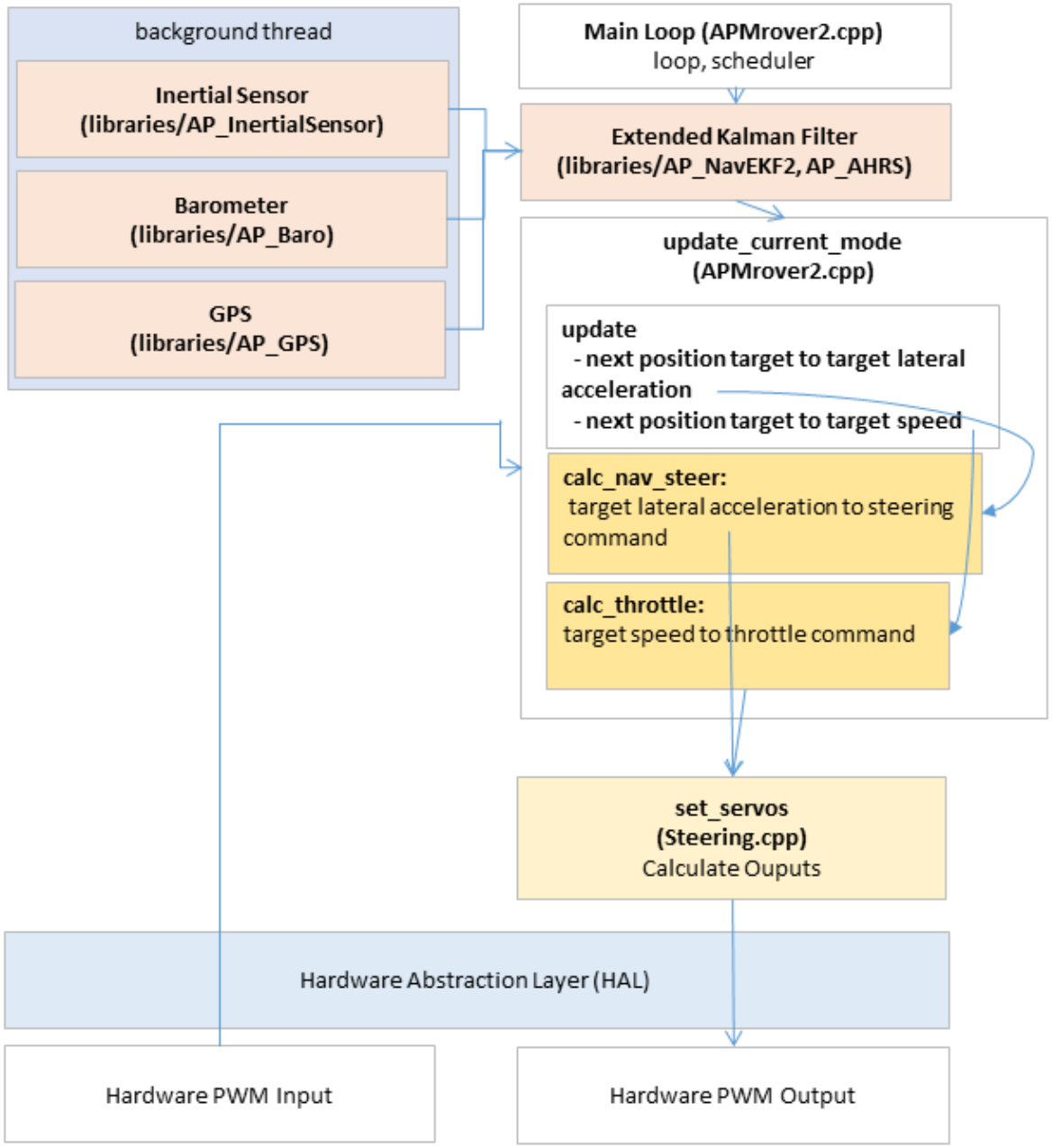


FIGURE 4.4 – Schéma de l'ancienne architecture du code d'Ardurover

Ainsi ArduRover était en retrait par rapport à ArduCopter et limité en utilisation hétérogène. En effet, seul un évitement d'obstacle basic était implémenté (rotation de 45 degrés en cas de détection d'obstacle, pas de gestion des lidars 360 degrés ou système de caméra), peu de gestion de pannes et problèmes était présents, etc. . Certaines fonctionnalités propres aux robots terrestres étaient aussi manquantes : rotation sur place,

déplacement en marche arrière, etc. Ainsi qu'une gestion correcte des moteurs. En effet, seule la gestion de deux moteurs avec des PWM (Pulse-Width Modulation ou modulation en largeur d'impulsion) RC (Radio Command) était disponible. La PWM RC se caractérise par un signal haut durant 1 à 2ms toutes les 20ms. Ce mode de contrôle vient des transmetteurs radio RF qui pouvait transmettre ce type de signal simplement sur plusieurs canaux de fréquence afin de contrôler plusieurs servo moteurs. Ce type PWM, illustré figure 4.5, est très utilisé dans les contrôleurs moteurs de drones volants ce qui explique son usage dans ArduPilot.

FIGURE 4.5 – Schéma de principe de PWM RC. Copyright Wikipedia sous licence CC BY-SA 3.0

Une PWM traditionnelle, figure 4.6, permet de générer un signal analogique à partir d'une source numérique, c'est le mode de contrôle principal des moteurs à courant continu. Elle est caractérisée par une fréquence et un rapport cyclique (duty cycle en anglais). La fréquence détermine la vitesse à laquelle la PWM effectue un cycle, c'est-à-dire le nombre de fois pour laquelle le signal passe d'un niveau haut à un niveau bas et vis et versa durant une seconde. Le rapport cyclique détermine le temps que le signal va passer au niveau haut en pourcentage du temps du cycle.

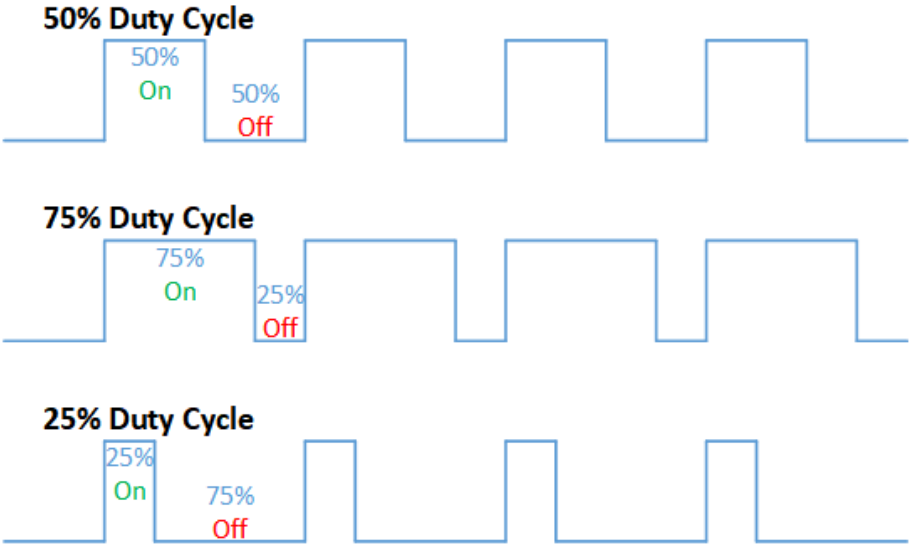


FIGURE 4.6 – Schéma de principe de PWM traditionnelle. Copyright Wikipedia sous licence CC BY-SA 3.0

Parmi les limitations générales les plus importantes se trouvaient :

- Les limites du modèle d'ArduRover : basé sur le modèle des voitures RC avec une architecture figé, voir 4.4.

- Contrôleurs moteurs limités aux seuls controleurs du monde RC.

- Le manque de capacité de contrôle d'ArduRover face à ArduCopter : seul un contrôleur en position ou en commande manuelle était disponible, pas géo-fencing, etc.

- Peu de compatibilité avec ROS.

- Simulations limitées et difficilement utilisables pour une mise en oeuvre d'un groupe de robots.

Ainsi un travail de contribution au projet a été entrepris afin de corriger ces limitations et faire profiter au plus grand nombre ces améliorations.

4.1.3 Principales modifications effectuées

Afin de pouvoir proposer cette plateforme dans des MRS hétérogène ou non, un travail d'ingénierie logicielle a été mis en œuvre afin d'améliorer la structure globale d'ArduRover. Les améliorations principales ont été les suivantes :

- Création d'un nouveau modèle d'architecture par mode pour séparer les contrôles hauts niveaux (navigation, évitement, etc.) de ceux bas niveaux (contrôle moteurs, rayon de braquage, etc.) voir figure 4.7 et des caractéristiques des différents types de véhicules : voiture, tank, bateau pour l'instant.

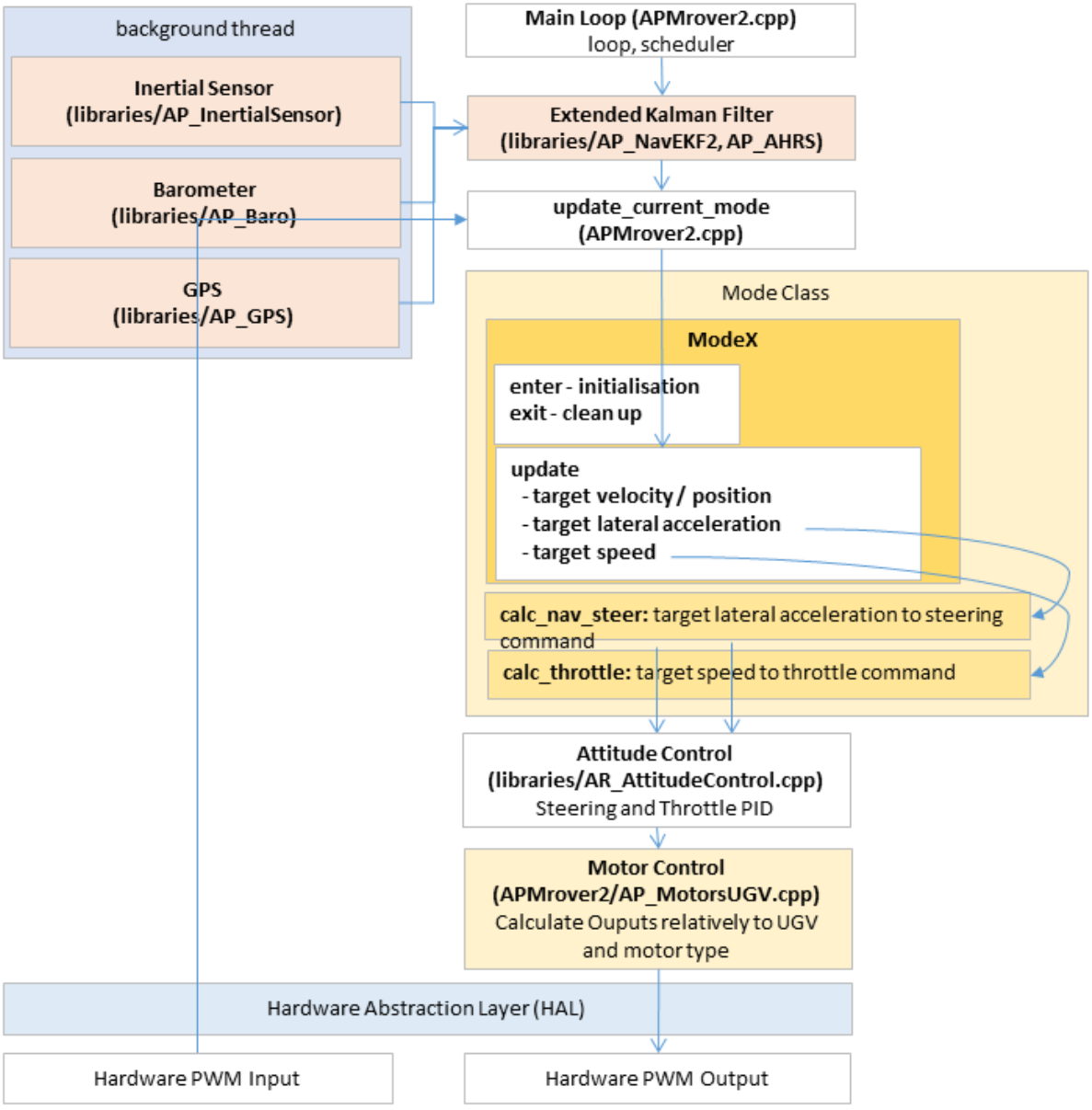


FIGURE 4.7 – Schéma de la nouvelle architecture du code d'Ardurover

– Implémentation de nouveaux modèles de contrôle et d'asservissement. ArduRover est passé d'un simple contrôleur PID en vitesse d'avance à un contrôleur d'attitude complet : PID pour la vitesse d'avance et de braquage ou de rotation, gestion du frein moteur et de la marche arrière, correction de l'effet centrifuge à haute vitesse, etc. La figure 4.8 montre l'amélioration significative dans le guidage qu'a apporté ce travail.

FIGURE 4.8 – Comparaison ancien contrôleur (en rouge) et nouveau contrôleur (en violet) sur un suivi de waypoints GPS (en blanc)

- Nouveau type de moteur supporté : support de la commande de contrôleur moteur en PWM RC (50Hz), PWM (50Hz à 20kHz), UAVCAN.
- Ajout d'un contrôleur en vitesse : précédemment, seul le contrôle automatique en position était possible, le contrôle en vitesse ne se faisait que par contrôle radio.
- Ajout de sécurité matérielle : un problème récurrent dans les robots est leur manque de protocole de sécurité. Il a donc été ajouté une gestion de l'état de la batterie. Cette gestion permet d'éviter des dégâts dus aux décharges profondes des batteries soit en stoppant le véhicule soit en faisant retourner le robot au point de départ avant le seuil de décharge critique. Cela permet également d'empêcher un effondrement de la tension de la batterie lors d'un changement immédiat de vitesse des moteurs et/ou limiter la tension dans les moteurs (Alimentation de moteurs 6V avec une batterie 12V par exemple). À cela, s'est ajoutée une fonctionnalité de détection de situation de blocage du véhicule afin de prévenir les dégâts sur les accouplements moteurs-roues.
- Adaptation et intégration de nouvelles librairies héritées du code des autres véhicules ArduPilot : Géo-fencing, évitement d'obstacles, retour intelligent au point de départ.
- Amélioration de la compatibilité avec ROS grâce à la nouvelle architecture et au contrôleur en vitesse (figure 4.9). En effet, le stack de navigation de ROS ne permet qu'un contrôle en vitesse des robots. Les améliorations ont porté sur la commande depuis ROS et le retour d'informations des capteurs présent sur la plateforme vers ROS.

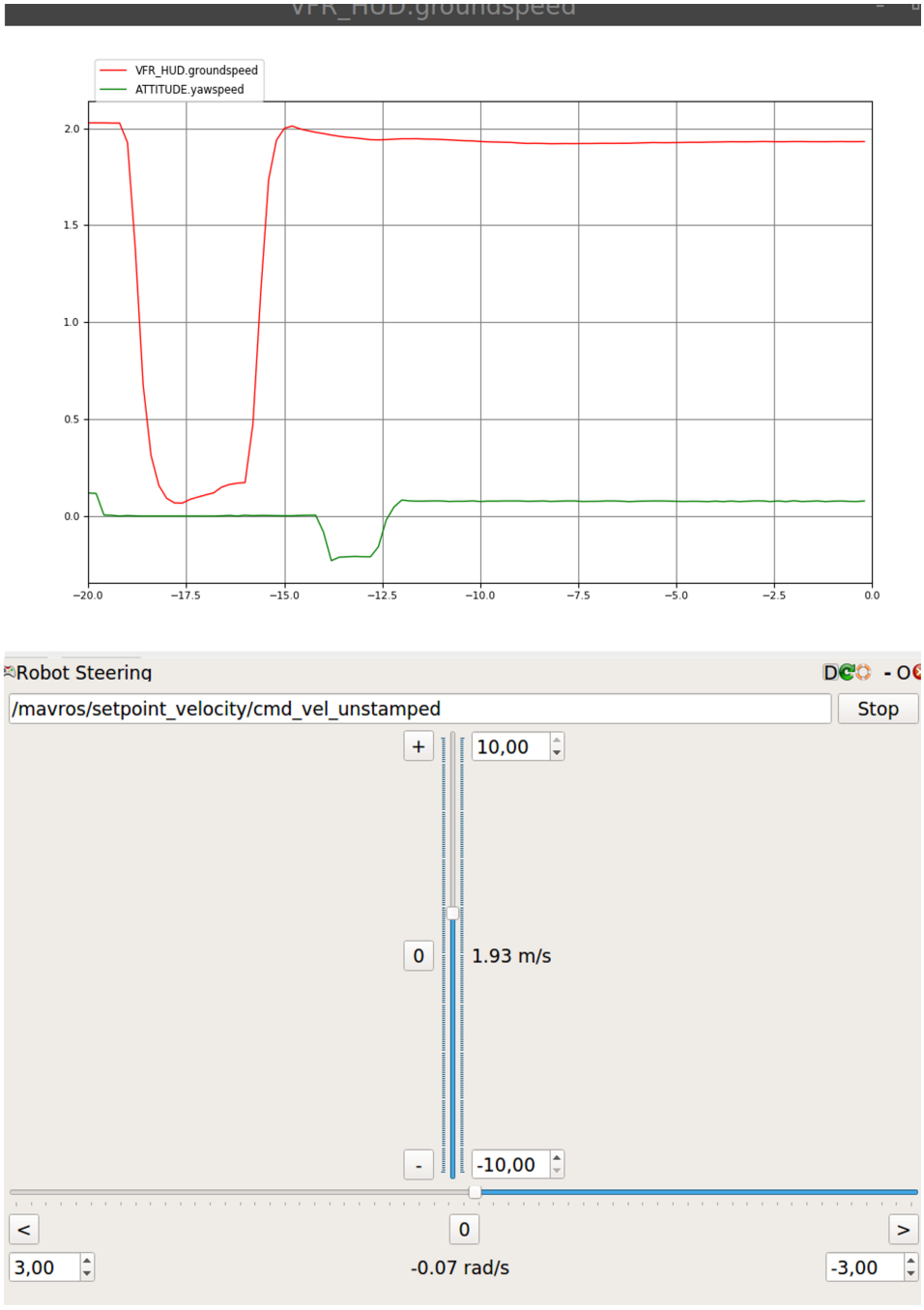


FIGURE 4.9 – Contrôle d'ArduRover en vitesse avec ROS via Mavros

– Renouvellement de la simulation SITL avec une amélioration des caractéristiques et prises en compte des deux modèles de robot roulant supporté et mise en place de fonctions d'autotest. Ces fonctions permettent, sur la base de simulation principalement, de vérifier automatiquement que chaque contribution au projet ArduRover

n'entraine pas de régression.

– Mise en place d'une nouvelle méthode d'autotest du code d'ArduPilot afin de tester en simulation la bonne mise en oeuvre des différents types de véhicules : vérification de la navigation, des protocoles sécurité, des réponses aux différents moyens de guidage, etc. Cet autotest est effectué sur chaque modification du code d'ArduPilot afin de valider (en plus de la review de code et des tests réels) qu'aucune régression n'est introduite dans les nouvelles versions et chaque test est enregistré sur les serveurs du projet ArduPilot [5]. C'est une nécessité pour les industriels d'avoir une preuve visible du bon fonctionnement de la plateforme. Les auteurs de [120] soulignent l'importance et l'utilité de ces tests, en dépit de leur analyse sur l'ancienne version de l'autotest d'ArduPilot. La nouvelle version étant passée d'une quarantaine de tests à plus d'une centaine par véhicule.

Durant le développement sur ArduRover est sortie la nouvelle version d'un robot de référence de ROS et qui est largement utilisé pour des expérimentations de MRS : TurtleBot 3. Une comparaison entre la plateforme mise en avant par cette thèse et une plateforme de référence de ROS est intéressante.

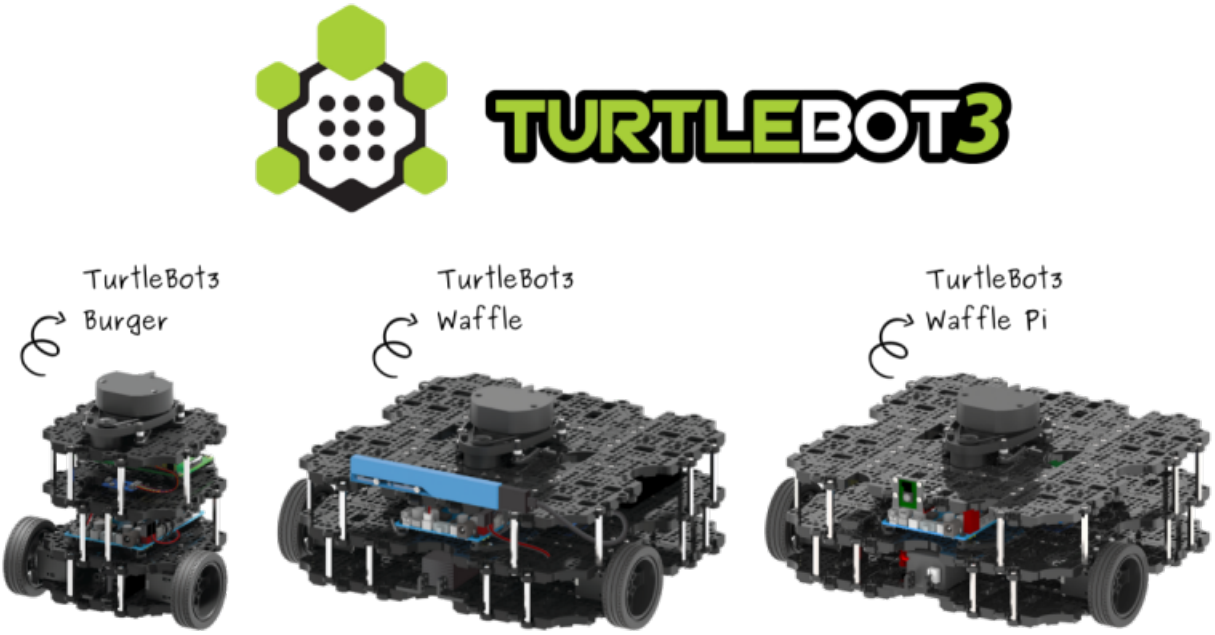


FIGURE 4.10 – Turtlebot3, Copyright ROBOTIS

Le tableau 4.2 permet de conforter les choix qui ont été faits dans ArduPilot. En effet, en utilisant un robot basé sur cette autopilote, on profite des avantages de l'éco système d'ArduPilot qui est beaucoup plus complet en termes de navigation native (Mode GPS, asservissement position local, etc.) et de sécurité (gestion des pannes matérielles, gestion

de la batterie, etc.) et un ensemble d'IHM compatible capable de programmé des missions complexes et fournir une analyse des performances post missions. Enfin, la compatibilité avec ROS reste présente sur la plateforme ArduPilot ce qui lui permet d'avoir les mêmes capacités que Turtlebot3 à ce niveau.

Tableau 4.2 – Comparaison ArduRover et TurtleBot3

Plateforme	ArduRover	TurtleBot3
Couts	< 1500 €	< 1500 €
Compatibilité ROS	OUI	OUI
Fonctionnement sans ROS	OUI	NON
Plusieurs types de véhicules possible	OUI	OUI
Utilisation en intérieur	OUI	OUI
Utilisation en extérieur	OUI	Peu
Systèmes de sécurité	OUI	NON
Gestion des pannes	OUI	NON
Durée de vie	Longue	Faible (la carte Intel Joule n'est déjà plus en vente)
IHM	OUI	NON
Simulation	OUI	OUI via ROS seulement
MRS compatible	OUI	OUI via ROS seulement
Amélioration matérielle possible	OUI	OUI via ROS

4.1.4 Conclusion

Les travaux présentés ici ont permis l'émergence d'une nouvelle plateforme de robot mobile terrestre et de surface open-source, générique, et à moindre coût, ce qui été les prérequis à l'utilisation dans les MRS. Bien que d'autres solutions existaient déjà, celle présentée ici s'est révélée plus complète et possède un taux d'adoption croissant aussi bien au niveau des industriels (Aion Robotic, Texys Marine, AzurDrones) qu'académique (UBS, University of Sarajevo, ETH). Cette adoption est notamment due à la robustesse et la fiabilité de la plateforme, mais aussi à la complémentarité avec ROS. En effet, ArduPilot possède les briques logicielles et matérielles manquantes par défaut dans ROS. Enfin, ces modifications ont permis une convergence des codes des rovers et copters dans ArduPilot, permis des démonstrations en groupe plus simplement [114], [112], et l'émergence de nouveau projet comme le support de robot omnidirectionnel ou à balancement [8] [4].

4.2 Outils pour la simulation

Plusieurs simulateurs ont déjà été présentés, cependant leur complexité et leur lourdeur de mise en application en configuration multi-robots sont aussi des freins à leur usage courant et pour des tests simples comme la gestion des déplacements ou gestion des communications. Cette section présente donc de nouveaux outils bâtis pour rendre ces tests plus accessibles.

4.2.1 ArduPilot SITL : présentation

Si les simulateurs exposés précédemment sont capables de répondre à ces caractéristiques, leur utilisation reste complexe et lourde à mettre en œuvre. La figure 4.11 présente l'exemple du code nécessaire pour mettre en oeuvre une canette statique dans Gazebo. Dans celui-ci nous pouvons voir le type de paramètres qu'il est nécessaire de savoir réglé afin de mettre en oeuvre un solide simple qui aura un contact avec le sol. La complexité de ce type de paramétrage freine l'usage de Gazebo, car ils sont complexes à comprendre et sans un bon paramétrage, la simulation ne fonctionnera pas.

En effet, les simulateurs comme Gazebo ont un temps d'apprentissage long du fait du manque d'exemples concrets surtout dans le domaine des simulations multi-robots et nécessitent d'importantes puissances de calculs. C'est pourquoi l'utilisation d'une simulation plus simple à prendre est généralement mise en oeuvre aux dépens du réalisme. Afin de parer ce défaut, il a été mis en oeuvre une nouvelle solution de simulation avec plusieurs niveaux de réalisme et surtout simple de prise en main pour des tests simples (vérification des mouvements, etc.). Cette nouvelle simulation se base sur le simulateur SITL du projet ArduPilot. SITL est capable de simuler les différents véhicules supportés et un ensemble de capteurs. La simulation des drones est complète dans le sens ou le code simulé est le même que celui qui sera embarqué sur les plateformes réelles.

```
<?xml version="1.0" ?>
<sdf version="1.5">
  <model name="coke_can">
    <link name="link">
      <inertial>
        <pose>-0.01 -0.012 0.15 0 0 0</pose>
        <mass>0.390</mass>
        <inertia>
          <ixx>0.00058</ixx>
          <ixy>0</ixy>
          <ixz>0</ixz>
          <iyy>0.00058</iyy>
          <iyz>0</iyz>
          <izz>0.00019</izz>
        </inertia>
      </inertial>
      <collision name="collision">
        <pose>0 0 -0.46 0 0 0</pose>
        <geometry>
          <mesh>
            <uri>model://coke_can/meshes/coke_can.dae</uri>
          </mesh>
        </geometry>
        <surface>
          <friction>
            <ode>
              <mu>1.0</mu>
              <mu2>1.0</mu2>
            </ode>
          </friction>
          <contact>
            <ode>
              <kp>10000000.0</kp>
              <kd>1.0</kd>
              <min_depth>0.001</min_depth>
              <max_vel>0.1</max_vel>
            </ode>
          </contact>
        </surface>
      </collision>
      <visual name="visual">
        <pose>0 0 -0.46 0 0 0</pose>
        <geometry>
          <mesh>
            <uri>model://coke_can/meshes/coke_can.dae</uri>
          </mesh>
        </geometry>
      </visual>
    </link>
  </model>
</sdf>
```

FIGURE 4.11 – Exemple du code nécessaire pour créer une canette dans Gazebo [88]

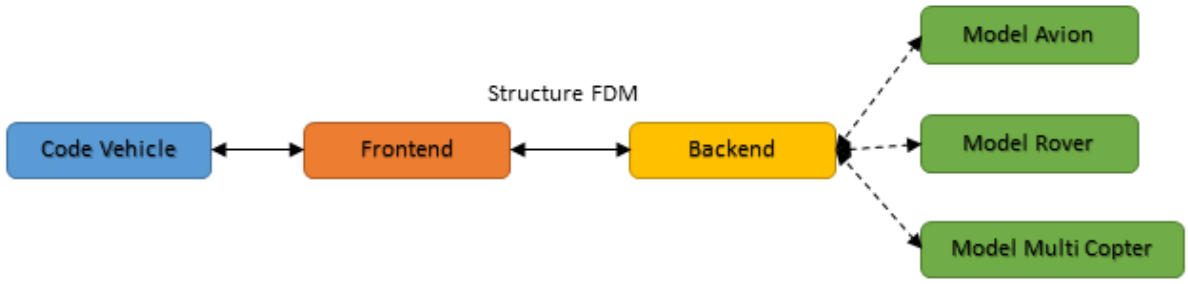


FIGURE 4.12 – SITL architecture

SITL repose sur des modèles de véhicules simples (poids, dimensions, etc.) et de capteurs simulés. L'architecture est séparée en 3 parties : le véhicule, le *frontend* et *backend*, schématise figure 4.12.

Véhicule : La partie véhicule simulée utilise le code complet d'ArduPilot et les données des capteurs simulés correspondant à celles des plateformes supportées : IMU, GPS, baromètre, etc. Grâce aux données des capteurs, le code réalise les tâches de stabilisation, navigation, évitement, etc., du véhicule et renvoi au *frontend* les sorties des actionneurs (moteurs, LEDs, etc.) calculées.

Frontend : La partie *frontend* fournit à l'autopilote les données simulées provenant du *backend*, et renvoie au *backend* les sorties de l'autopilote qui seront utilisées pour mettre à jour les modèles dans la simulation. C'est cette partie qui est responsable de la gestion de l'horloge de la simulation. L'horloge de la simulation est séparée de l'horloge de celle de l'environnement matériel. Cela permet d'accélérer le temps simulé ou de le stopper pour effectuer du débogage par exemple.

Backend : La partie *backend* définit l'environnement simulé avec les caractéristiques techniques des véhicules et des capteurs (poids, dimension). Chaque véhicule possède un lot de caractéristiques et plusieurs véhicules par défaut correspondant aux véhicules les plus utilisés sont proposés. À chaque pas de temps simulé, le *frontend* met à jour le *backend* qui envoie à l'autopilote les données de capteurs et reçoit les données des actionneurs qui feront évoluer la simulation.

La séparation en *frontend* et *backend* du simulateur est efficace pour permettre l'utilisation de différente *backend*. Si le *backend* par défaut reste simple, il permet de simuler rapidement des robots avec un jeu de capteurs simple, mais ne possède pas de modèle de collision autre qu'avec le sol.

Durant le projet Daisie, il a été remarqué trois problèmes majeurs à SITL. Si la simulation était aisée pour un seul drone, la configuration de SITL pour utiliser plusieurs véhicules en parallèle se révélait compliquée. De plus, l'architecture de SITL ne permettait

pas l'ajout simple de nouveau capteur du côté de la simulation alors que le code du véhicule le permettait. Enfin, le dernier problème relevé était le manque de compatibilité avec un environnement 3D ou l'ajout de donnée provenant d'autres simulateurs pour augmenter les modèles présents. La section suivante présentera donc les contributions qui ont été développées afin de proposer des solutions à ces problèmes.

4.2.2 Contributions

Afin de pouvoir proposer des simulations aussi bien simples que complexes, plusieurs contributions ont été apportées à SITL dans le but d'en faire un simulateur multi-niveaux. L'objectif est de pouvoir simuler rapidement des groupes de véhicules avec une courbe d'apprentissage du simulateur légère, mais sans tomber dans le piège des simulations trop simple ou le modèle du véhicule ne possède aucune physique, mais seulement une variable informatique. Mais aussi de permettre des niveaux de simulation avancés ou combinés avec la réalité.

Amélioration générale

Un travail important d'amélioration de SITL a été entrepris afin de permettre l'ajout plus simple de *backend* et de capteur depuis les différents *backend* et une interface de lancement de la simulation permettant de créer simplement plusieurs instances de SITL avec des véhicules configurés pour fonctionner en groupe illustré figure 4.13.

L'utilisation d'autre *backend* permet donc d'augmenter les modèles et les niveaux de simulation. Ainsi il est désormais possible de connecter SITL à Gazebo comme simulateur afin de profiter de modèle de collision et d'environnement de simulation plus complet (obstacles, contraintes mécaniques, etc.). La figure 4.14 montre l'ensemble des nouvelles configurations qui sont possibles avec SITL. Les différentes nouvelles parties seront présentées dans la suite de cette section.

L'usage de SITL permet désormais des simulations de MRS simples, mais rapides avec le simple ajout de mécanismes de comportement collectif ou de coordination soit directement dans le code du véhicule soit avec une connexion extérieure comme ROS. En effet, les avancés durant le projet DAISIE ont mis en avant la nécessité de pouvoir utiliser une simulation rapide, mais proche d'une réalité robotique afin de travailler sur des comportements collectifs. Dans le cas de SITL, la simulation de base est simplifiée puisque les modèles de véhicule utilisés sont basiques : modèle de capteurs et actionneurs simples, pas de système de collision, pas de problématiques mécaniques. Mais ils simulent de façon réaliste les déplacements et comportements des véhicules puisque le code entier de l'autopilote est utilisé. Cette simulation faible en ressource (1-2% d'un processeur i5 de 2017 pour une instance de véhicule) permet des avancés rapides dans les validations de comportements collectives avec un simple apprentissage d'usage de la plateforme et des notions de sécuritaire associé comme la gestion de la batterie.

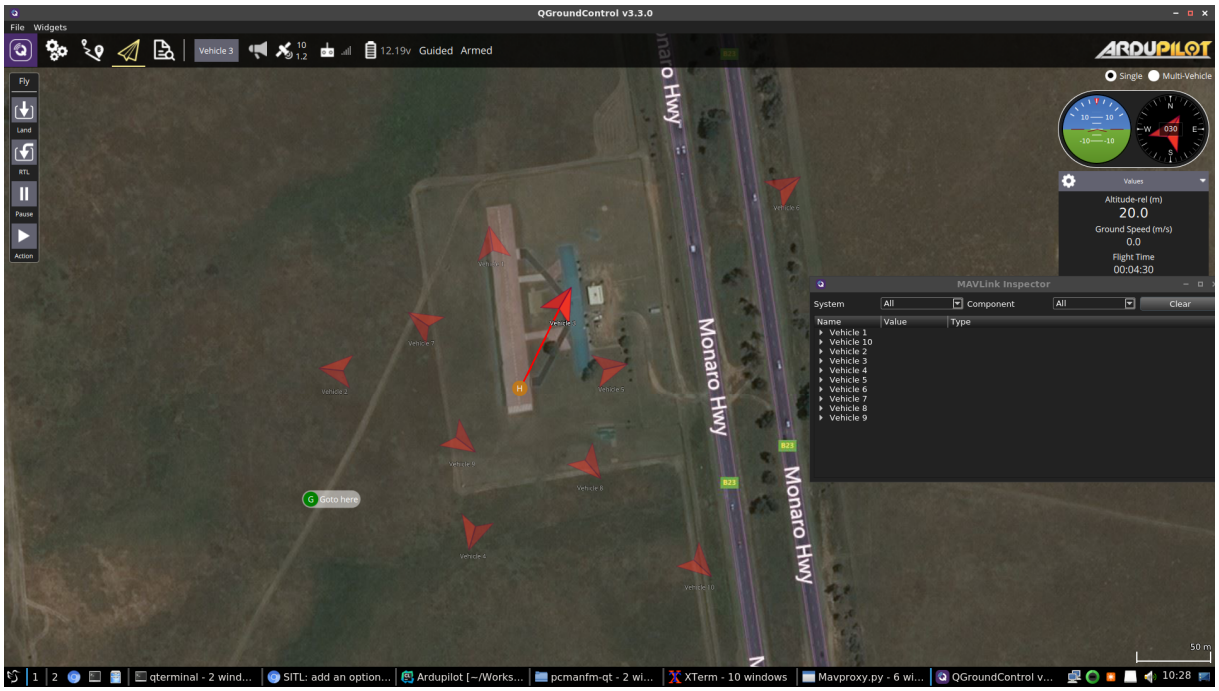


FIGURE 4.13 – Simulation multi véhicules à l'aide de SITL

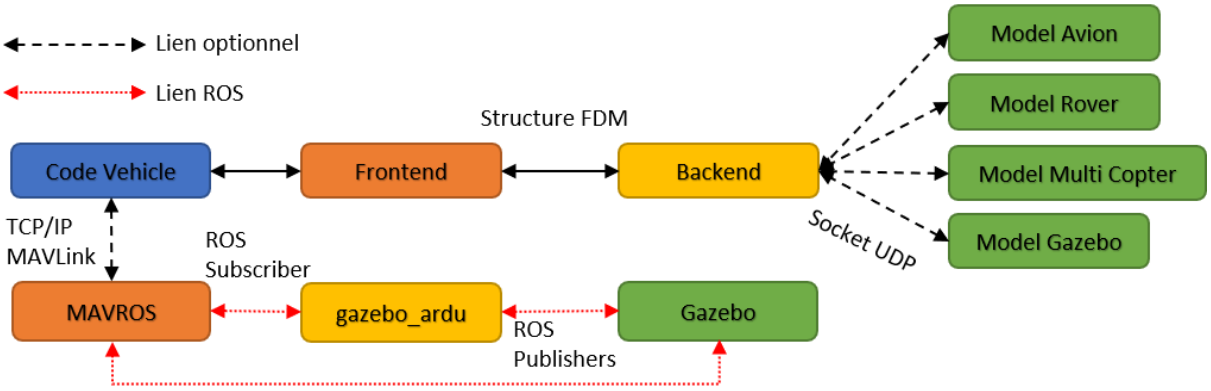


FIGURE 4.14 – Schéma des nouvelles configurations possibles de SITL

Augmentation des capacités avec ajout de la 3D

Afin d'étendre les possibilités d'usage de SITL, il a été développé un simple plugin de "plotting" dans Gazebo [63]. Ce plugin permet de charger un modèle 3D de véhicule et de capteurs externes (lidar, sonar, caméra, etc.) dans Gazebo et de déplacer le modèle avec les données de position et d'orientation de provenant de SITL.

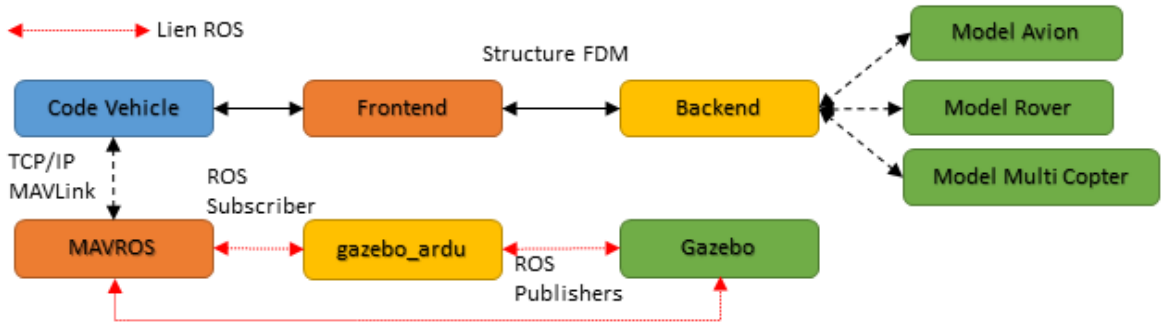


FIGURE 4.15 – Schéma de connexion entre SITL et le "plotteur"

Ainsi Gazebo sert de simulateur pour les capteurs et de moteurs 3D pour les robots dont les données simulées proviennent de SITL. Cette architecture est la même que celle utilisée dans les drones utilisant des autopilotes : l'autopilote gère les parties bas niveau et un autre contrôleur (généralement un micro-ordinateur) gère les tâches hauts niveaux avec ROS (capteur nécessitant des traitements type Lidar 360, path planning, etc.). Pour cela, ROS est utilisé. Un plugin Gazebo aurait pu être développé, mais il était plus utile et simple d'utiliser ROS ici. En effet, les capteurs simulés dans Gazebo s'interfacent avec ROS directement et avec ArduPilot via Mavros, voir figure 4.15. Ainsi il est possible de combiner les trois logiciels, pour faire des simulations rapides avec des simulations de capteurs avancés et une gestion des obstacles et collisions simple au prix d'une perte de réalisme. Du fait de l'usage de Gazebo, cette simulation est plus lourde que celle fournie par SITL seul tant au niveau puissance de calcul et de mise oeuvre. En effet, l'utilisation du moteur graphique de Gazebo reste coûteuse et nécessite à minimal de savoir créer un modèle 3D dans Gazebo. Néanmoins, il n'est plus nécessaire de paramétrer le modèle pour tirer parti du moteur de physique de Gazebo, ce qui est un gain de temps important. Cette solution apporte une gestion des collisions et de capteurs avancés sans la gestion des modèles cinématiques et dynamiques avancés habituellement utilisés avec Gazebo et qui sont difficiles à mettre en œuvre. Ce type de simulation est une solution pour tester des stratégies de localisation ou d'évitement qui seront ultérieurement optimisées pour être embarquées sans les désavantages d'une simulation trop lente et difficile à configurer.

La figure 4.16 présente une visualisation d'un comportement de leader-followers réalisé directement dans ArduPilot (vidéo disponible à [64]), le drone encadré est le leader qui envoie sa position courante et son vecteur vitesse aux deux followers qui doivent maintenir

une distance constante et une position donnée avec le leader. Pour cela, 3 instances de SITL sont lancées 4.17, chacune est connectée aux deux autres afin de simuler une liaison radio entre les drones, et chacune lance une instance de Mavros et de gazebo_ardu. Chaque instance de gazebo_ardu met à jour le bon modèle de drone dans Gazebo.

FIGURE 4.16 – Simulation de Leader-Follower avec Gazebo et SITL

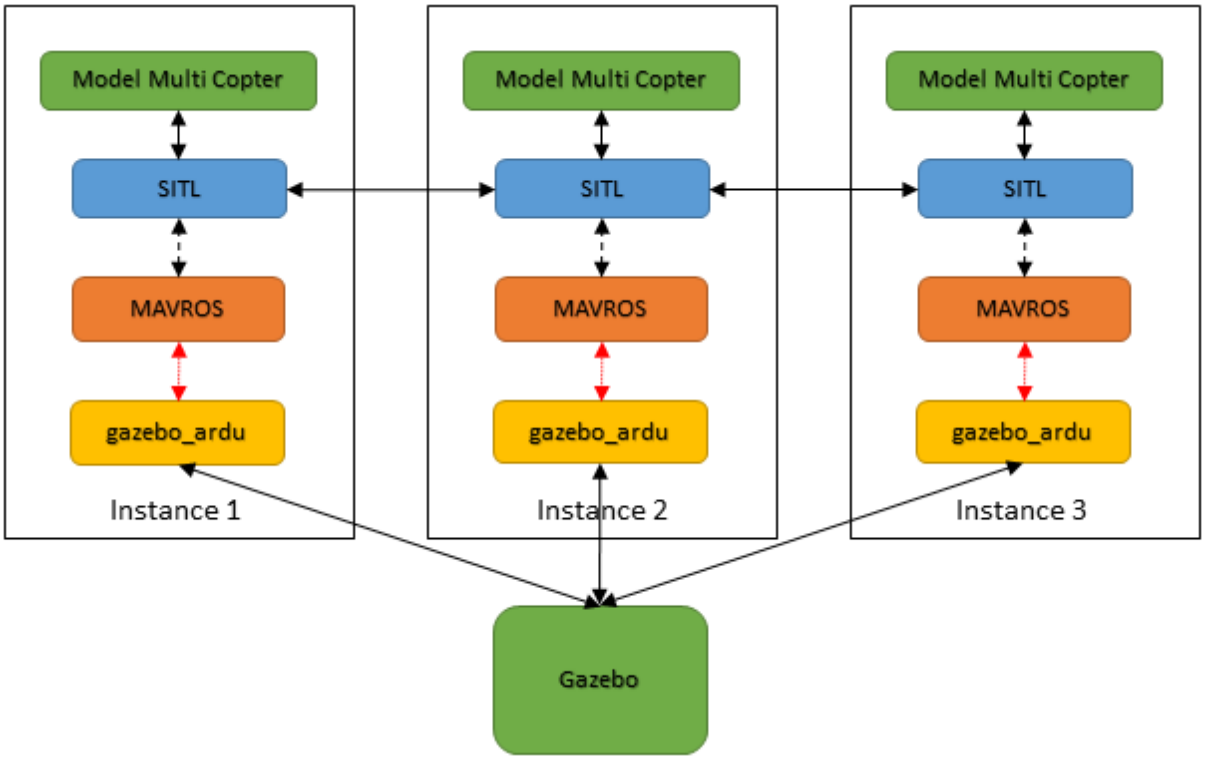


FIGURE 4.17 – Architecture de la simulation de Leader-Follower avec Gazebo et SITL

Ajout d'un backend pour Gazebo

Enfin, un nouveau plugin Gazebo a été créé [62]. Celui-ci permet cette fois d'utiliser Gazebo comme backend de SITL afin de profiter de modèle de collision et d'environnement de simulation plus complet (obstacles, contraintes mécaniques, etc.). C'est-à-dire que les toutes données fournies à SITL proviennent des modèles chargés dans Gazebo. Cela permet de profiter des modèles avancés de capteurs de Gazebo directement dans l'autopilote. Il n'est plus ici nécessaire d'utiliser ROS comme passerelle.

En effet, même si ArduPilot n'est pas aussi complet de ROS en termes d'algorithmes de robotique (path planning, interface captors, etc.), il l'est suffisamment pour permettre des déplacements en autonomie (navigation sur trajet par point de passage GPS et évitement d'obstacle simple). Ce nouveau plugin Gazebo n'inclut pas ArduPilot dans celui-ci, mais permet une connexion directe avec SITL qui fonctionne sur tous les types de véhicules : volant, roulant et navigant et supporte les configurations multi-robots. La connexion se fait à travers deux sockets UPD avec un deux messages prédéfinis dont le contenu est synthétisé dans les figures 4.19 et 4.18.

Le premier socket permet à Gazebo d'envoyer à SITL plusieurs données :

– L'horloge de la simulation : c'est maintenant Gazebo qui sert de serveur temps pour la simulation.

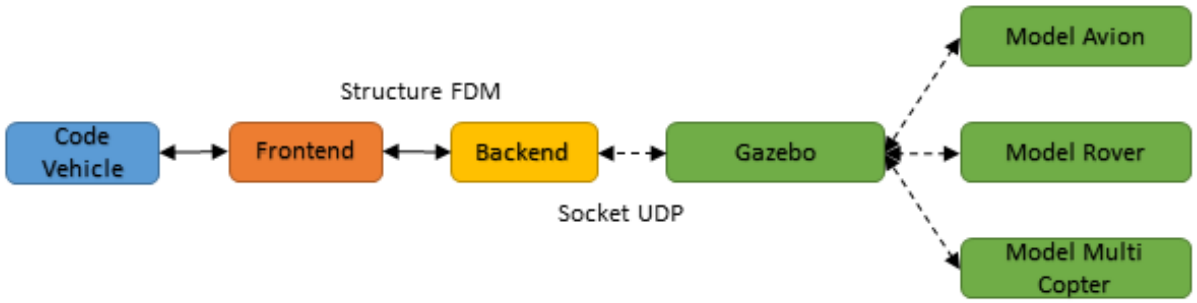


FIGURE 4.18 – Schéma de connexion entre SITL et Gazebo

- Les données de position du modèle dans Gazebo : position x,y,z en mètre dans le repère de la simulation.
- Les données de vitesse du modèle dans Gazebo : vitesse x,y,z en mètre par seconde dans le repère de la simulation.
- Les données de l'IMU du modèle dans Gazebo : accélération, quaternion, vitesse angulaire.
- Les données de position GPS en coordonnées WGS84.
- Les données du capteur de distance pour l'altitude : sonar ou lidar
- Les données du capteur de pression barométrique et d'anémométrie.

Ces données permettent de faire fonctionner la simulation sous SITL en les fournissant au code du véhicule. Avec ces données simulées, le code du véhicule va mettre à jour son attitude et renvoyer à SITL la commande pour chaque moteur du véhicule. SITL va renvoyer ces commandes à Gazebo à travers le deuxième socket afin que le modèle dans Gazebo simule les moteurs et mette à jour le modèle.

L'utilisation de deux sockets UDP peut être limitante, mais elle apporte une solution fonctionnelle pour des simulations. Une des difficultés rencontrées a été la différence entre les deux projets : Gazebo et ArduPilot. En effet, même si les deux projets sont open source et permettent des usages avancés de la robotique, ils n'ont pas de synergie entre eux. La problématique a été de trouver une solution logicielle qui fonctionne pour les deux projets. En effet, Gazebo propose des API de programmation, les utiliser impliqués d'ajouter de lourdes dépendances logicielles au projet ArduPilot alors que peu de gens utiliseront Gazebo. De l'autre côté, ArduPilot est liée au protocole MAVLink, mais si le protocole des trames MAVLink est versionné, les différents types de messages utilisés évoluent encore rapidement et ne permettent pas un versionning, ce qui n'aurait pas été compatible avec le cycle de sortie des versions de Gazebo. Ainsi le compromis le plus

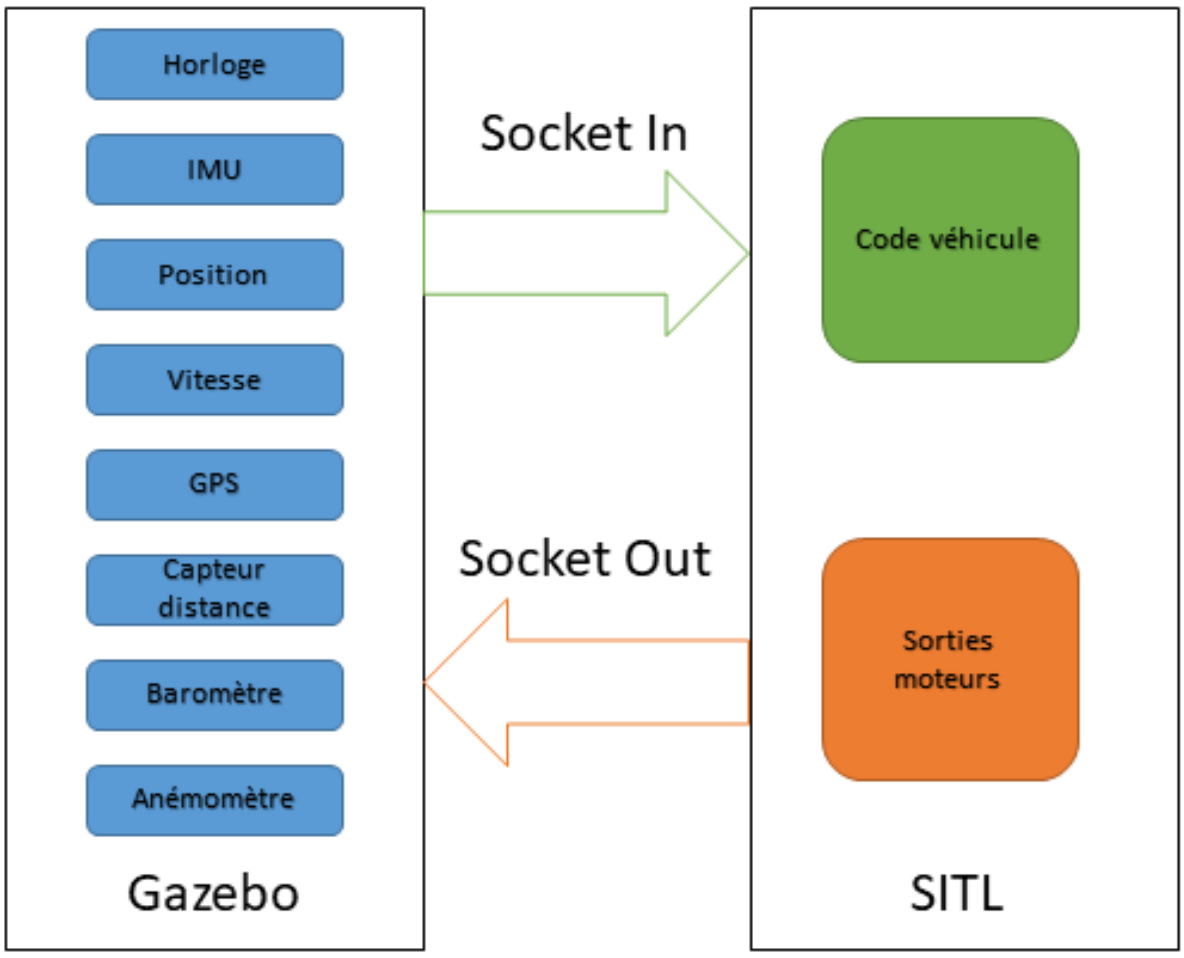


FIGURE 4.19 – Schéma de fonctionnement du plugin SITL pour Gazebo

efficace a été de définir un nouveau type de message pour les backends de SITL, malgré la limite d'évolutivité de celui-ci.

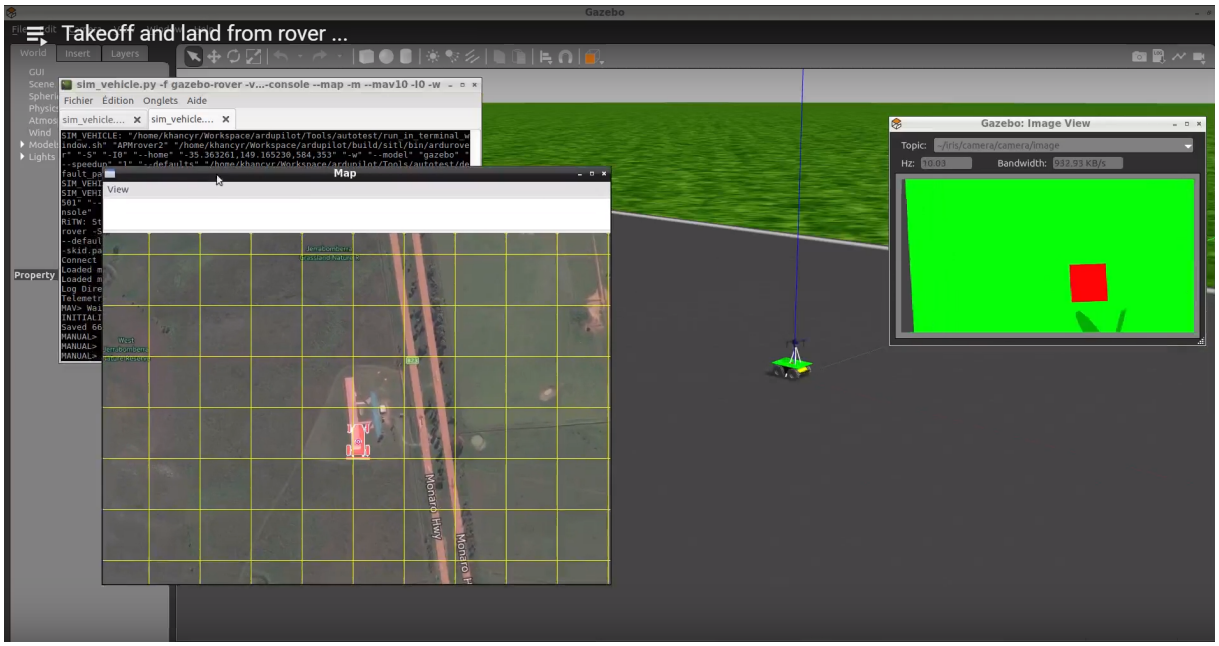


FIGURE 4.20 – Simulation d'un rover et un quadcopter réalisant un atterrissage de précision sur le rover avec le code ArduPilot.

La figure 4.20 présente la simulation d'un rover et d'un quadcopter sous ArduPilot dans Gazebo (vidéo disponible à [65]). Grâce aux données de la caméra, le quadcopter est capable d'effectuer le suivi du rover et se poser avec précision dessus lors de mouvements. Le code pour l'atterrissage de précision est disponible dans ArduPilot [7], Gazebo à permet ici de vérifier son bon fonctionnement avec l'ajout du vent et différentes vitesses de déplacement du rover.

4.2.3 Conclusion

Cette section a présenté une plateforme d'autopilote complémentaire à ROS et trois nouvelles méthodes de simulation devant permettre de simplifier les développements de MRS en particulier ceux hétérogènes, c'est dire comportant plusieurs types véhicules différents. Ces outils ont été mis à disposition de la communauté scientifique en tant de logiciels Open Source [63] et [62] un nombre important de contributions ont été faites directement sur ArduPilot et Gazebo. Cette plateforme est compatible avec ROS, mais répond à des contraintes de fonctionnements plus proches des solutions attendues par les industriels, et fournira un framework générique pour des solutions MRS encore plus avancées. Les résultats de ce travail ont été nombreux. Ils sont à la base des simulations réalisées pour le Service Academies Swarm Challenge Live-Fly Competition [21] ou deux

groupes de 40 UAV (20 ailes volantes et 20 quadrotors) s'affrontent dans un match de capture de drapeaux (vidéo disponible à [26]). Deux projets de doctorants ont, durant le GSOC 2018, utilisés ces outils pour fournir des simulations de suivis de chemin avec réseaux de neurones [81] (figure 4.21) et d'évitement d'obstacles avec des drones volants [46] (figure 4.22).

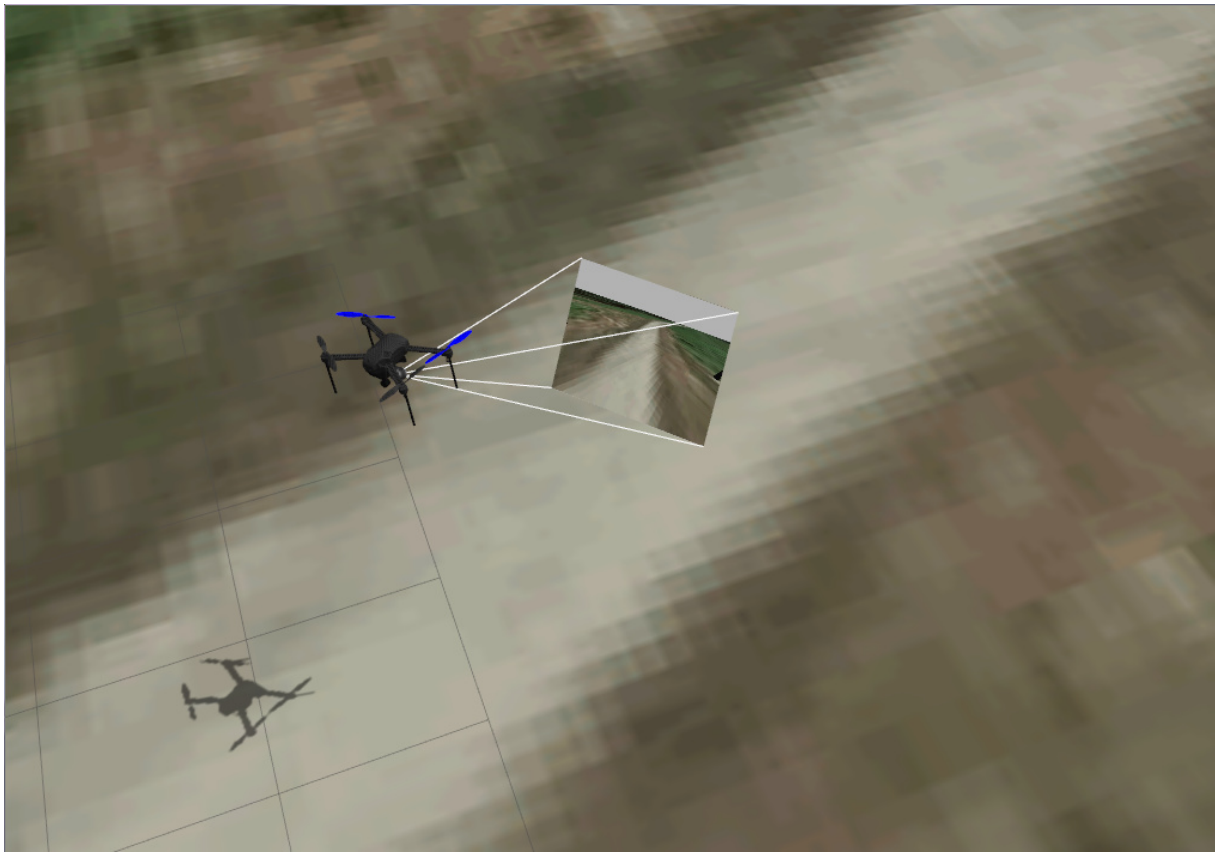


FIGURE 4.21 – Simulation d'un quadcopter réalisant un suivi de chemin dans Gazebo.

Dans les trois outils de simulations développées, une attention a été portée afin de fournir le maximum d'autoconfigurations possible pour les utilisateurs qui ne sont pas du domaine, une documentation des outils, et des exemples d'utilisations.

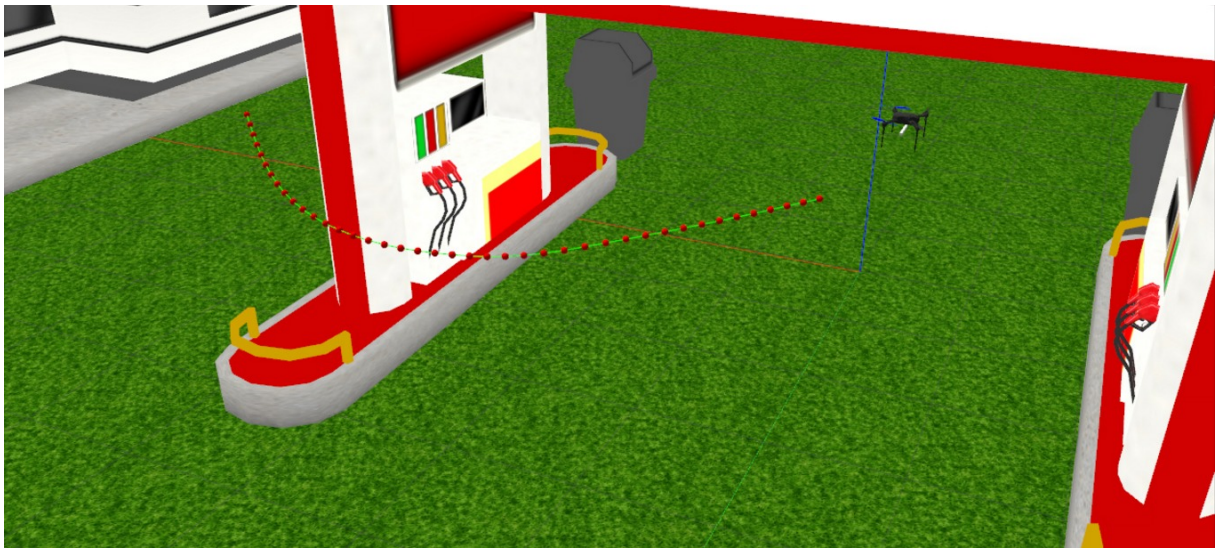


FIGURE 4.22 – Simulation d'un quadcopter réalisant un évitement d'obstacle dans Gazebo.

Chapitre 5

Proposition d'un outil complet de simulation et dimensionnement

Les chapitres 2 et 3 ont permis de mettre en avant les difficultés à la fois de dimensionnement, mais aussi d'évaluation de l'efficacité d'un système multi-robots. Dans l'objectif d'un déploiement industriel de telles solutions, l'évaluation des performances d'un système doit pouvoir être démontrée expérimentalement afin de dépasser les modèles mathématiques théoriques qui ne prennent pas en compte la nature matérielle des robots, leurs environnements de travail et les pannes de ces systèmes. Cependant, au regard du prix des solutions robotiques actuelles, une expérimentation sur les performances sans dimensionnement préalable serait extrêmement coûteuse. Le chapitre suivant propose un framework, sur la base des travaux présentés précédemment, permettant de faire des évaluations de performance au regard du coût de la solution et ainsi établir les bases pour le dimensionnement de MRS.

5.1 Proposition d'un framework générique

Il a été montré dans les chapitres précédents qu'il existe de nombreux outils pour la robotique notamment avec ROS et Gazebo, mais aussi que ceux-ci se révèlent difficile d'accès et incomplet pour un usage multi-robot à la fois pour des simulations ou des utilisations réelles. Les auteurs de [52] propose 5 caractéristiques pour de design d'un simulateur pour MRS :

- *Design and development of algorithms should be straight forward, reduce effort and involve minimal learning curve.*
- *Should support multiple entities, i.e. simulation of swarms, rather than isolated individuals.*
- *Should support entity-entity communication because communication and interaction between swarm individuals causes the emergence of swarm behavior.*

– *Entities in the simulator should have a construction similar to real robots. This enables algorithm portability and supports the direct translation of algorithms from the simulator, to a hardware platform.*

– *The simulation platform must run on a standard desktop computer.*

Sur la base de ces caractéristiques attendues, le facteur le plus important pour un bon framework, après le réalisme, est la simplicité d'utilisation. Ainsi l'utilisation des références actuelles comme base de travail de l'outil est une nécessité. La solution qui va être présentée est basée sur ROS, Gazebo et ArduPilot. En effet, les travaux sur ces trois projets représentent un temps de développement important et une maturité d'utilisation qu'il serait difficilement évitable afin de proposer une solution crédible. Devant d'être accessible au plus grand nombre, l'usage de l'open source comme base est tout à fait adapté. Le but étant de se construire sur des bases solides et de tirer parti des communautés, aussi bien amateurs, académiques ou industrielles, pour multiplier les tests, les usages et comparer les solutions obtenues afin d'améliorer continuellement l'outil.

L'outil doit permettre de répondre à chaque question de la méthode QQOQCCP (Qui ? Quoi ? Où ? Quand ? Comment ? Combien ? Pourquoi ?) indépendamment des autres :

– Qui ? : quel type de robots et combien ?

– Quoi ? : le type de mission parmi un ensemble prédéfini (p. ex. exploration, patrouille, etc.)

– Où ? : le terrain de la mission (dimensions, obstacles, etc.)

– Quand ? : la durée de la mission

– Comment ? : l'algorithme qui gère le MRS

– Combien ? : le coût de la solution (coût du drone, des capteurs et de l'énergie nécessaire à la mission, etc.)

La réponse au "combien" coûtent la solution est rarement discutée dans les publications académiques alors qu'une simple évaluation de celle-ci au regard des performances donne une métrique importante sur le dimensionnement de la solution. Le framework de l'outil est composé de 5 parties :

1. La partie bas niveau avec ArduPilot présenté en cf.4.1.2 : cette partie est responsable du contrôle des robots, de la navigation de base, de la communication inter robot via MAVLink, et de récolte des données de la plupart des capteurs.

2. La partie haut niveau avec ROS cf.3.1.3 : cette partie est responsable des algorithmes avancés des robots comme l'évitement d'obstacle, la collaboration, etc..

3. La partie simulation : combinaison de Gazebo, ROS et SITL cf.4. Elle permet la simulation sur plusieurs résolutions : faible avec SITL, et forte avec Gazebo. La combinaison ROS et Gazebo permet la simulation de capteur avancé comme des caméras.

4. La partie IHM : elle permet de faire fonctionner le groupe de robot et de surveiller l'état de chaque membre.

5. Un script d'évaluation des performances : sur la base des robots simulés, il évalue les performances de mission, mais aussi individuelles de chaque unité et réalise une évaluation globale de la solution.

6. Un script de lancement des expérimentations. Celui-ci est responsable de la configuration de l'expérimentation en utilisant soit la partie simulation soit des robots réels. Il permet en fin de mission la récolte des différentes données qui seront fournies au script précédent 5. Une IHM pour est en cours de réalisation pour facilité le lancement des expérimentations et l'affichage des données, mais ne sera pas présenté dans ce document.

Ainsi composé, le framework permet de parer aux défauts individuels de chacune de ses parties. ArduPilot par son usage simple, mais sur de nombreux véhicules permet de diminuer la difficulté d'utilisation de ROS avec des véhicules non terrestres et en groupe. ROS étend les capacités d'ArduPilot avec des algorithmes plus avancés. SITL permet des simulations plus simples que Gazebo, mais celui-ci permet un usage de la 3D et des gestions de collisions. ArduPilot permet aussi l'utilisation de nombreuse IHM qui est nécessaire à la fois à la bonne simulation, mais aussi aux usages réels. En effet, le framework peut être utilisé en simulation aussi bien que sur des robots réels. La combinaison de ROS et ArduPilot permettent la création de robots réelle avec le même code qui a été simulé. De plus, la non-dépendance du code fonctionnel aux parties de simulations permet un mixage de réalité et réalité virtuelle afin d'expérimenter de futures solutions. L'intérêt de cet outil repose dans la possibilité de comparer des résultats d'estimation à haut niveau : temps de simulation court et modèles simplifiés à des estimations à bas niveau avec une simulation plus complète, et à une évaluation du code sur les vraies machines.

La partie 4 sur l'IHM ne sera pas abordée ici. En effet, il existe déjà de nombreuse IHM capable de communiquer avec le protocole MAVLink : QGroundControl [102], MissionPlanner [80], UGCS [39] et de gérer une flotte de robots.

La partie 5 donne une évaluation du groupe. Pour cela, plusieurs bibliothèques de modèles génériques sont disponibles, à la fois sous ROS, et sur ArduPilot. Le choix du modèle de robots et des capteurs embarqué est important, car il va permettre de fixer les performances optimales attendues pour chaque unité du groupe, mais aussi le coût de la solution. L'évaluation du groupe se fait ensuite avec plusieurs métriques :

– QoS Mission : elle est adaptative en fonction de la mission : taux de couverture pour les patrouilles, temps d'exploration pour l'exploration, etc.

– Consommation moyenne de batterie : elle est calculée sur la base des consommations individuelles. Cette donnée est importante pour un dimensionnement réel. En effet, la plupart des solutions robotiques sont énergivores et la recharge d'une batterie nécessite un temps d'autant plus important qu'elle a une réserve d'énergie importante. Plusieurs cas peuvent se présenter, les principaux sont les suivants :

  1. Les robots retournent à la base et rechargent leur batterie. En fonction du nombre de bases de chargement, cela nécessite une stratégie rotative à la fois pour la gestion des stations de recharge qui sont partagées. Mais aussi pour éviter que tous les robots soient en chargement ce qui signifie un arrêt temporaire de la mission.
  2. Les robots sont capables d'effectuer un échange de batterie. Cela nécessite une réserve de batterie suffisante pour couvrir toute la durée de la mission.

  Dans les deux cas, le choix de la stratégie de recharge à une importance sur le dimensionnement de la solution puisqu'elle agit directement sur la disponibilité du groupe, mais aussi sur le coût d'usage de la solution multi-robots. En effet, le coût d'une station, d'une batterie de rechange ou de la charge énergétique d'une batterie devient vite conséquent avec le nombre de robots mise en oeuvre.

– Distance parcourue moyenne : elle est aussi calculée sur la base des distances parcourues individuellement. Cette donnée à une double utilité. En effet, elle va permettre le dimensionnement en nombre du groupe. En fonction de la mission, le nombre de robots en mission va diminuer la charge sur chaque robot jusqu'à que l'augmentation du nombre de robots dans le groupe deviennent contre-productif. Cette métrique est aussi utilisée pour la maintenance des robots. En effet, combiné avec le temps d'activité des robots, elle permet d'estimer l'usure des actionneurs des robots et donc la durée de vie des unités.

– Temps de fonctionnement : cette métrique permet d'évaluer la charge de travail des robots.

– Coût de la solution : cette métrique est évaluée par rapport aux modèles de robots choisis et des métriques précédentes.

5.2 Cas d'étude et validation

Afin de montrer l'intérêt de l'approche, il a été réalisé des expérimentations en patrouille avec différents types de robots. Malheureusement, le framework n'était pas disponible pour la réalisation du projet DAISIE cf.3.2, il n'a donc pas pu servir ni pour la phase de simulation et dimensionnent, ni pour l'expérimentation.

Dans [99], les auteurs définissent la patrouille comme "l'activité de contourner ou de traverser une zone à intervalles réguliers à des fins de sécurité". Cette tâche est complexe,

mais représente un challenge ou les MRS montrent leurs avantages en proposant une performance optimale dans la surveillance des zones voulues. De nombreuses applications sont possibles comme la supervision de l'environnement, la récolte d'informations, la détection d'anomalies, etc. L'utilisation d'équipes de robots pour des tâches de surveillance active présente plusieurs avantages surs, par exemple, un système de surveillance passive par caméra, qui ont déjà été exposés au chapitre2.1.2.

Récemment, des industriels [11], [2] et des travaux académiques[98] [126] se sont intéressés à la mise en oeuvre de patrouille par flotte automatisée. Cela a mis en avant plusieurs variations dans les stratégies de patrouilles multi-robots et notamment dans la manière de juger de l'efficacité de la patrouille. Cependant, la métrique la plus utilisée dans ce type de travaux est l'*idleness*, c'est-à-dire le temps pendant lequel un point dans l'environnement passe sans être visité par un robot.

5.2.1 Système expérimental

Cette section présentera donc une application du framework présenté précédemment accès sur la mise en oeuvre puisque la mise en oeuvre en simulation est la même que celle pour des robots réels. Seules trois stratégies connues seront présentées afin de réaliser une patrouille sur une zone donnée, les différentes problématiques telles que décompositions des tâches, la planification ne seront pas abordées. La figure 5.1 présent le schéma logiciel global de la solution.

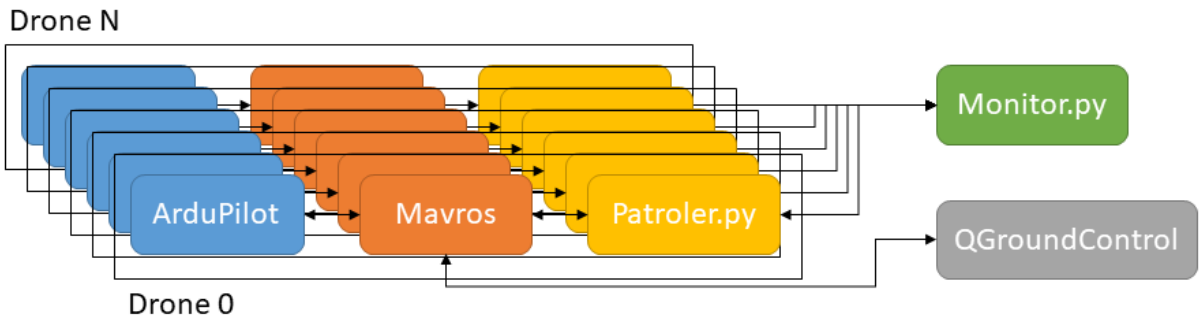


FIGURE 5.1 – Schéma global du système expérimental.

Robots

Le système mettra en oeuvre trois types de systèmes :

1. Une flotte de robots roulants.
2. Une flotte de robots volants de type multirotors.
3. Une flotte de robots avec le même nombre de robots roulant que volant.

Les robots sont tous équipés de l'autopilote ArduPilot et du matériel associé ainsi que d'un micro-ordinateur équipé de ROS. Ainsi grâce à l'autopilote et le GNSS, chacun des robots est capable de se déplacer correctement sans mettre en oeuvre de technique lourde comme du SLAM. Ce choix se justifie dans ce système, car les déplacements se feront dans un environnement connu et en extérieur. Si le SLAM à base de Lidar 306 ° se révèle efficace en intérieur, son usage en extérieur fait face à de nombreuses problématiques comme la faible portée des rayons lasers et les problèmes de réflexivités associés (par exemple dans le brouillard), mais aussi algorithmiques puisque la majorité des algorithmes de SLAM ont besoin de repères dans l'espace pour fonctionner. Le fonctionnement en terrain extérieur plat est donc compliqué. Contrairement aux démonstrations des équipes de Portugal and Rocha avec leur simulateur *patrolling_sim* [95], les patrouilles auront lieu sur de grandes distances et avec des vecteurs aériens dont les outils de SLAM ne sont encore que faiblement développés pour une application réelle. Dans le cas où le positionnement par GNSS ne serait pas désirable, l'autopilote est aussi capable d'utiliser les données odométriques des roues `http://ardupilot.org/rover/docs/wheel-encoder.html`, le SLAM provenant de ROS `http://ardupilot.org/dev/docs/ros-slam.html`, ou des balises de localisation `http://ardupilot.org/rover/docs/common-marvelmind.html`.

La vitesse maximale des drones volants a été fixée à 10m/s, soit 36km/h. Celle des drones terrestres est de 5m/s, soit 18km/h.

Afin de pouvoir réellement effectuer une mission de patrouille, chaque unité est équipée d'une caméra. Afin de pouvoir transmettre le flux vidéo chaque robot possède un transmetteur radio fréquence longue portée.

Afin de pouvoir effectuer un évitement d'obstacle, chaque robot est équipé d'un Lidar frontal permettant à l'autopilote d'effectuer une manoeuvre d'évitement. La stratégie d'évitement est simple : si un obstacle est détecté, on tourne à droite jusqu'à la fin de détection et on ajuste la trajectoire.

Ainsi équipé, le prix d'un drone volant est estimé à 4000€ et celui-ci d'un robot terrestre à 2500€.

Modification pour les simulations

La gestion de l'énergie consommée dans les simulations est souvent approximée à la consommation des moteurs durant les déplacements. En effet, la consommation des moteurs est généralement bien plus importante que celle de l'électronique embarquée. Cependant, afin d'avoir une estimation de la consommation, une simple estimation linéaire entre la consommation et la distance parcourue est utilisée [128]. Cette approximation est valide dans le cas des robots terrestres ou une situation d'attente se traduit par une consommation des moteurs nulle, mais n'est pas applicable dans le cas des drones volants où l'attente est consommatrice. Ainsi, l'estimation de consommation de batterie repose ici sur une loi de consommation linéaire relative à la puissance moteur (5.1) et (5.2). On considère une consommation de 50A à la vitesse maximale.

Drone volant

$$consommation = 50.0 * \mid vitesse \mid \tag{5.1}$$

Drone terrestre

$$consommation = 25.0 * \mid vitesse \mid \tag{5.2}$$

Le choix du changement de batterie est fait ici à des fins simplificatrices. Si le temps de fonctionnement d'un robot terrestre de taille normal peut aisément dépasser une heure de fonctionnement, un drone multirotors avec une charge utile, c.-à-d. capteur, ne dépasse généralement pas une vingtaine de minutes de vol. Les drones possèdent donc initialement une batterie Lipo 4 cellules (4S) d'une capacité de 12000mAh. Ce type de batterie est standard dans le domaine des drones volant. En effet, elles ont un rapport énergie sur prix intéressant et permettent des usages à plus fort courant que des batteries de type Li-ion. Jusqu'à 2018, ce type de batterie était nécessaire pour les drones volants du fait du manque d'optimisation des moteurs et contrôleurs moteurs brushless utilisés. Cependant, les nouvelles technologies de moteurs ont permis une diminution significative de leur consommation. Les drones volants effectueront donc des aller-retour à la base pour un changement de batterie lorsqu'ils auront consommé 9000mAh d'énergie.

La plupart des simulations ne prennent pas en compte les pannes possibles dans les systèmes autres qu'un problème dans l'algorithme de localisation. Il est souvent défini un modèle probabiliste pour les pertes de données lors de communications, mais pas de prise en compte de perte matérielle de robot autre que lors d'une erreur de navigation. Il a donc été ajouté une fonction probabilité afin de simuler un problème matériel sur un robot. Afin de pousser le réalisme de la simulation, une fonction simulation de panne a été ajoutée. Celle-ci utilise un générateur de nombre pseudo-aléatoire (RAND) avec les taux probabilités de pannes définis comme il suit :

- Problème système moteur (contrôleur, moteur) : 20%
- Problème système navigation (IMU, compas, GPS) : 10%
- Problème batterie (connexion, régulateur, vieillesse) : 25%
- Problème électronique (Panne processeur, câblage) : 5%
- Problème système de communication : 20%
- Problème capteur mission : 20%

Cette fonction est exécutée chaque seconde avec une probabilité de 1 / 3600000 de tirer une panne. Ce qui équivaut à une panne pour 41 jours de fonctionnement. Cette valeur est issue d'une moyenne sur les durées de vie des moteurs de qualité pour les drones volants.

Les simulations sont réalisées sur un temps de 60 minutes avec un rapport de temps simulé sur temps réel de 10. Ce qui équivaut à 10s de temps simulé pour 1s de temps réel. Cela permet des simulations rapides, malgré l'augmentation de la charge pour l'ordinateur de simulation.

Stratégies de patrouille

Trois modes de patrouilles simples ont été implémentés. En effet, le but de la démonstration est de montrer l'intérêt du framework plus que les résultats opérationnels. Ainsi, les algorithmes utilisés sont simples. Ceux-ci ont été extraits de [95]. La patrouille s'effectue sur un nombre de points fixé à 12 et préalablement connu cf 5.2.

- Patrouille aléatoire (RANDOM). C'est un algorithme décentralisé. Chaque robot possède la liste des points de patrouille et décide aléatoirement du point à visiter. La fonction aléatoire est ici basée sur la fonction *Random* du langage Python. Cette fonction donne une distribution gaussienne basée sur l'algorithme de *Mersenne Twister* [75].

- Patrouille réactive (HIGHEST). C'est un algorithme centralisé. À l'arrivée à destination, chaque robot notifie la station de contrôle qui lui renvoie le point suivant à patrouiller. Celui-ci étant le point avec la plus forte *idleness* et n'ayant pas de drone attribué. Si aucun point n'est disponible, la station tire un point au hasard avec le même algorithme que pour la patrouille aléatoire. Une autre solution aurait pu être de mettre le drone en attente d'un point, mais cela aurait posé une limite au système lorsqu'il y a plus de drones que de points à patrouiller et limité l'aspect *imprévisible* de la patrouille.

- Patrouille cyclique (CYCLE). C'est un algorithme décentralisé. Au démarrage de la patrouille, chaque robot crée un cycle de patrouille aléatoire avec la liste des points à patrouiller. Ainsi, chaque robot parcourt les mêmes points, mais dans un ordre différent.

Algorithmes

Les robots sont donc programmés pour fonctionner en mode automatique avec ArduPilot. L'autopilote est donc en attente permanente d'instructions de la part de la partie ROS. La partie ROS contient la machine d'état principale. Celle-ci possède 6 états :

1. Initialisation : le drone reste au sol et attend l'ordre de départ de la patrouille.

2. Décollage : le drone effectue son auto-diagnostic et décolle à l'altitude de démarrage en cas de succès. Afin d'éviter les collisions entre les drones volants, chaque drone volant possède son altitude de démarrage (en mètres) qui est calculé avec la règle suivante :
$$altitude = 15.0m + numero\_drone * 2.0m \tag{5.3}$$

3. Mission : en fonction de la stratégie de mission choisie, le drone se dirige vers le point de patrouille tiré. Quand il arrive au point voulu, il reste sur zone 20s puis notifie la station de contrôle avant d'aller vers le point suivant calculé selon le type de patrouille.

4. Changement de batterie : il a été fait ici le choix d'utiliser la solution de remplacement de batterie. Ainsi, lorsque le drone a consommé 9000mAh d'énergie, il retourne à la base, se pose, et attend 60s avant de repartir avec une batterie pleine.

5. Fin de mission : lorsque le timer de fin de mission se déclenche, chaque drone rentre à la base, s'arrête et envoie au moniteur ses statistiques de mission.

6. Panne : lorsqu'un problème survient, le drone essaye de rentrer la base. S'il ne peut pas, il tente un atterrissage sur zone. S'il ne peut pas, il déclenche son parachute. Dans les trois cas, le drone est considéré comme perdu puisqu'il nécessitera une maintenance active.

Cette partie est aussi en charge de la communication avec la base sol pour reporter l'arrivée sur un point de patrouille.

Moniteur

Le moniteur est une application ROS qui monitor l'avancement de la patrouille. C'est le moniteur qui déclenche le départ et la fin de la patrouille. À chaque fois qu'un drone a fini son attente sur point, il reporte au moniteur le point patrouillé, et si le mode le nécessite, attend le prochain point de patrouille. À la fin du temps de patrouille allouée, le moniteur attend que chaque robot soit rentré et posé à la base afin de récolter les statistiques d'usages et écrire le rapport dans un fichier texte.

Terrain de patrouille

Le terrain de patrouille est situé sur la zone de vol étendu du CMAC à Canberra. Ce terrain est le terrain de démarrage par défaut de SITL qui se situe aux coordonnées GPS : *-35.363261,149.165230,584,353*. La figure 5.2 présente la zone de patrouille avec un drone posé sur le point de démarrage.

La température est fixée à 25 ° ce qui permet un usage optimal des batteries. Dans ces simulations, il n'a pas été fixé de vent.

Plateforme matérielle de simulation

La plateforme matérielle utilisée pour les simulations est un PC portable équipé d'un processeur Intel Core^{TM} i5-7200U CPU @ 2.50GHz (2 coeurs) et 8Go de mémoire RAM. Il est équipé d'une carte graphique Nvidia GeForce 940MX qui n'est pas utilisée ici. Sur ce faible matériel, les simulations en vitesse x10 sont possibles jusqu'à 20 drones en un temps raisonnable. En effet, au-delà, la charge sur les processeurs est telle que la simulation n'est plus exécutée à la vitesse x10. Ainsi, sur ce matériel, il est plus rentable de diminuer l'accélération de la simulation à plus de 20 drones.

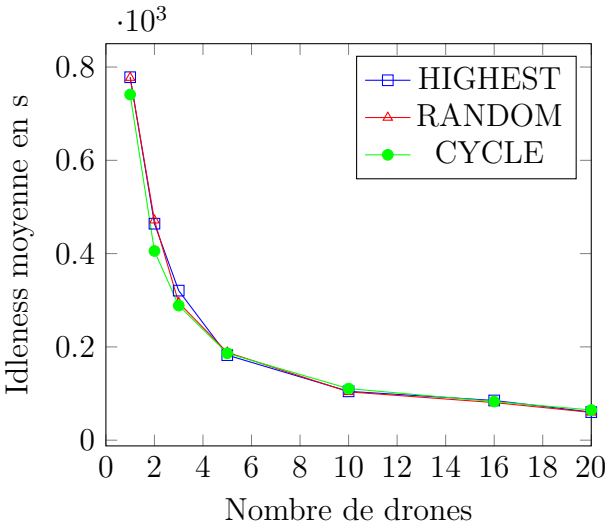
FIGURE 5.2 – Carte des points de patrouilles pour les simulations.

5.2.2 Résultats

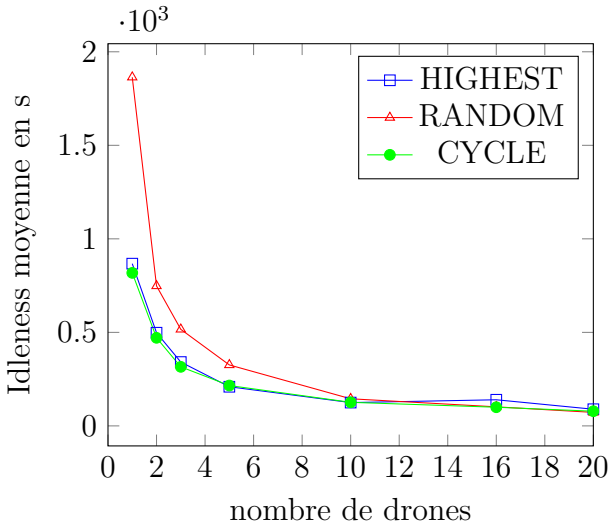
Cette partie présente les résultats obtenus sur de nombreuses simulations en faisant varier le nombre de drones dans la flotte.

Résultats drones volants

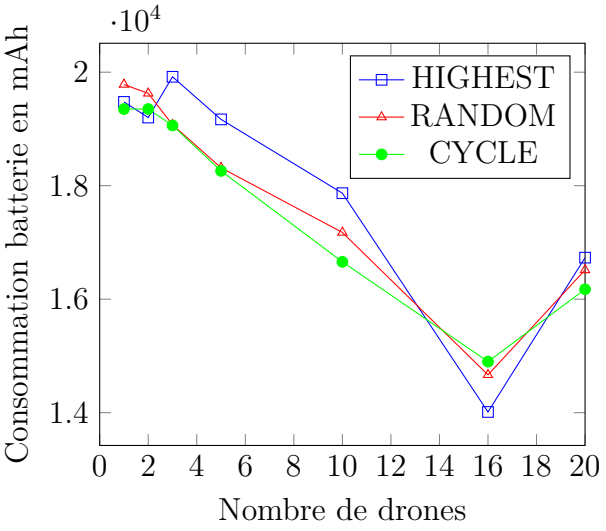
Idleness moyenne
en fonction du nombre de drones



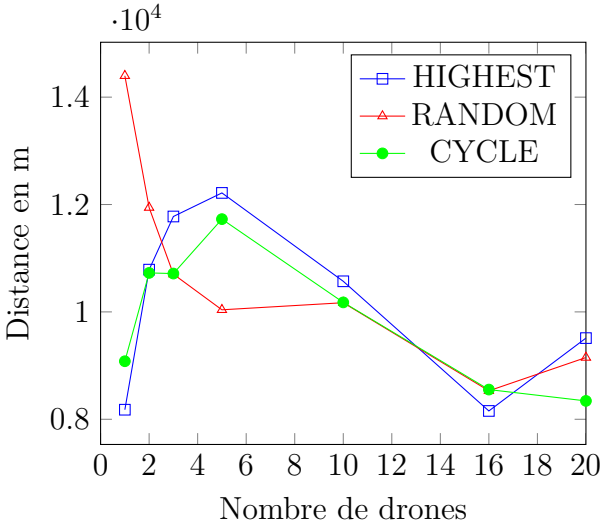
Plus mauvais idleness moyenne
par point de patrouille en
fonction du nombre de drones



Consommation batterie moyenne



Distance parcourue moyenne par drone

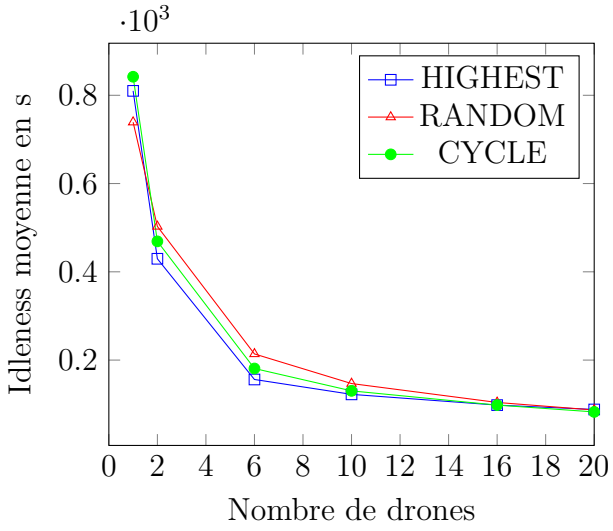


Les résultats des simulations montrent que les trois algorithmes donnent le même type de résultat avec des drones volants au niveau de l'efficacité de la patrouille, quel que soit le nombre de drones. Dans les trois cas, nous observons une nette diminution de l'idleness jusqu'à 10 drones. L'amélioration de la performance est beaucoup moins importante en-

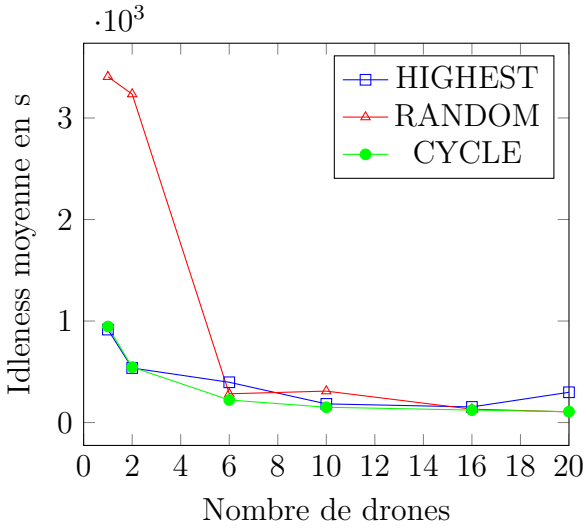
suite. Cependant, si on tient compte du facteur de consommation de batterie, on observe un regain de consommation à plus de 16 drones. Cela est dû au mode de lancement des drones qui ont un temps d'attente avant chaque décollage afin de prévenir les collisions. Si l'on tient compte de l'augmentation de 22% de performance entre 10 et 16 drones avec 27% d'économie de batterie, mais pour un coût 60% plus élevé, la performance de 16 drones reste la meilleure.

Résultats drones terrestres

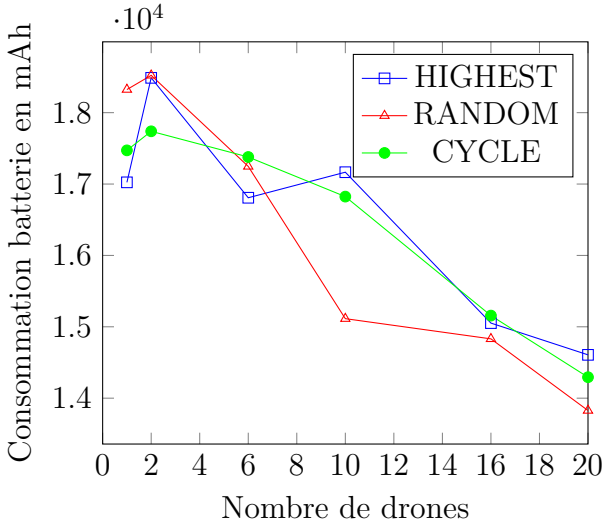
Idleness moyenne
en fonction du nombre de drones



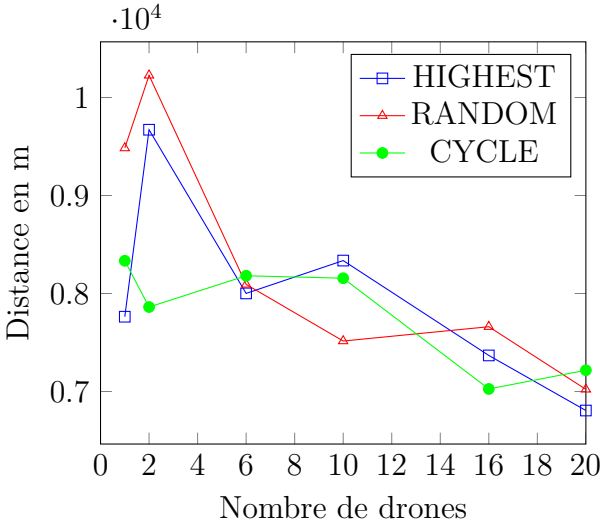
Plus mauvais idleness moyenne
par point de patrouille en
fonction du nombre de drones



Consommation batterie moyenne



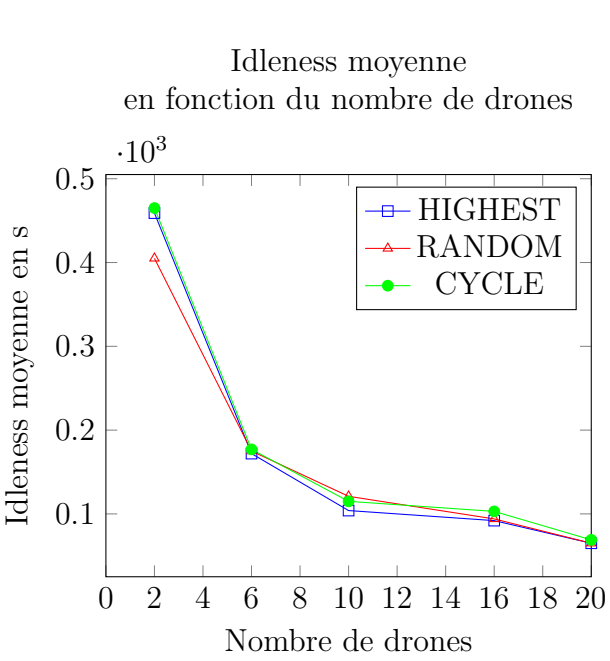
Distance parcourue moyenne par drone

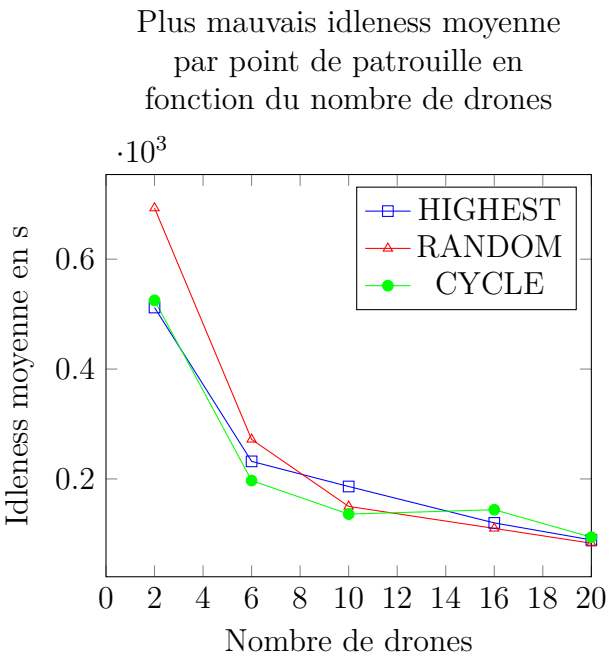


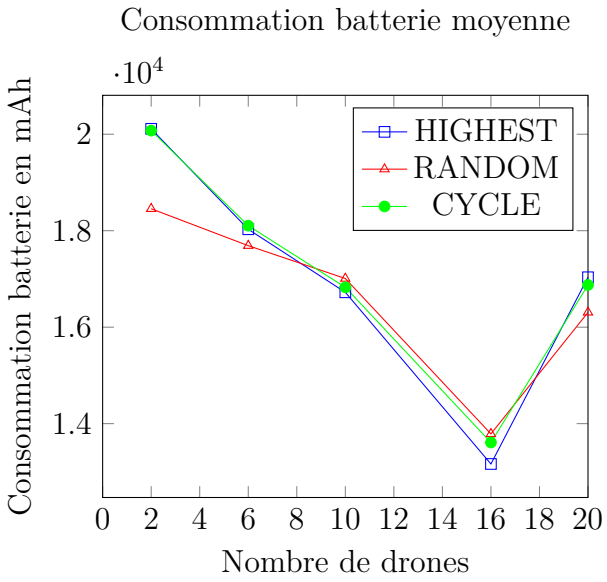
Les résultats des simulations montrent que les trois algorithmes donnent, encore une

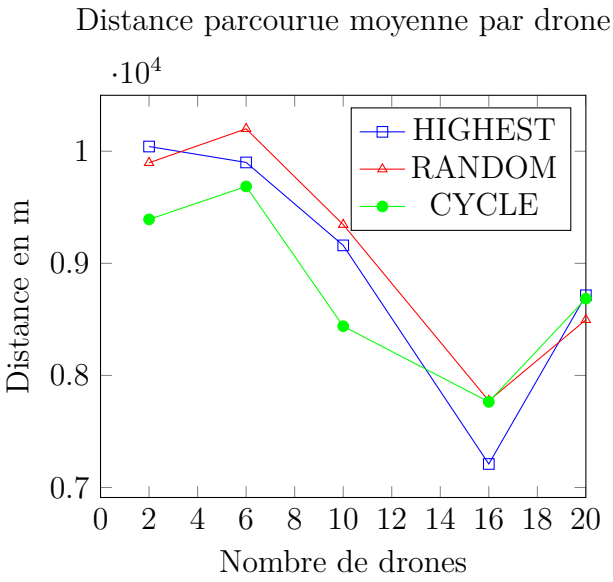
fois, le même type de résultat avec des drones terrestre au niveau de l'efficacité de la patrouille, quel que soit le nombre de drones. Dans les trois cas, nous observons une nette diminution de l'idleness jusqu'à 10 drones. L'amélioration de la performance est beaucoup moins importante ensuite. Contrairement à la démonstration précédente, la diminution de la consommation de batterie reste constante avec l'augmentation du nombre de drones. Il en est de même avec la distance parcourue avec plus de 10 drones. Dans le cas de la stratégie HIGHEST, la redistribution des robots sur le point le moins patrouillé montre ses limites puisqu'on observe après 16 drones, une augmentation de la pire idleness moyenne par point. En effet, les robots terrestres sont plus lents que les aériens, une redistribution abusive sur un point éloigné n'améliorera pas les performances. Au vu des performances croissantes, et la diminution à la fois de la consommation et de la distance parcourue par chaque unité, la solution de 20 drones est la meilleure.

Résultats drones combiné

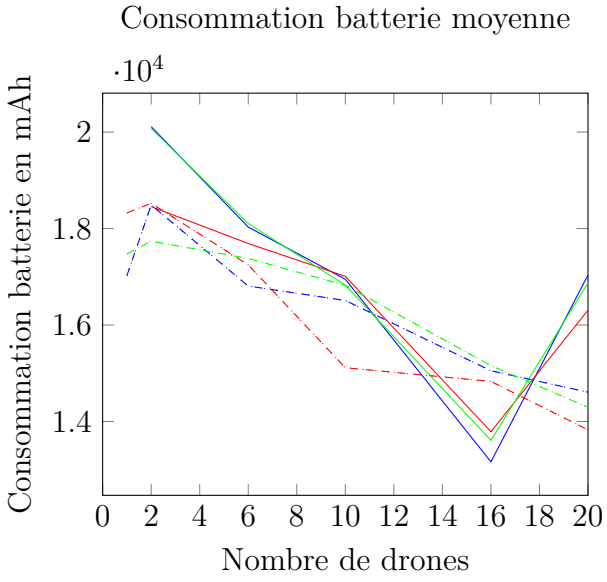


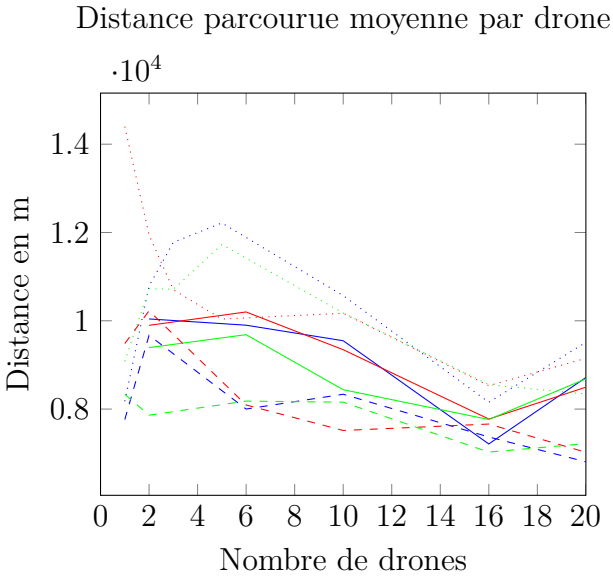


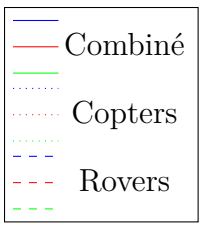


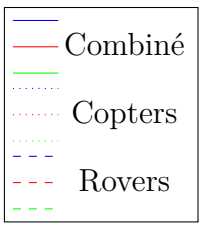


La solution avec un mélange de drones volants et roulants présente des résultats similaires aux précédents avec des performances proches de celle des drones volants seuls, qui étaient les meilleures.









Si l'on regarde plus en détail les performances globales par rapport à l'utilisation des drones, on peut remarquer clairement un gain dans la consommation moyenne de bat-

terie et une diminution significative de la distance parcourue par chaque drone. Ainsi, l'usage combiné de drones volants et terrestres se révèle plus avantageux qu'une solution purement volante ou roulante. En effet, les performances en combiné sont similaires à celles des drones volants, mais avec un coût bien moins important. Ceci est dû à la fois au fait que les drones roulants sont deux fois moins chers que les volants, mais aussi à la consommation du groupe qui sera moins importante et donc moins coûteuse au niveau de la recharge des batteries. Enfin, du fait de la diminution d'utilisation de chaque drone, leur durée de vie en sera augmentée.

5.3 Conclusion

La solution présentée ici de réaliser de nombreuses simulations avec différents types de robots et différents algorithmes. Le code de la simulation et les résultats textes sont disponibles à l'adresse : Github : patrol. Une vidéo du déroulement d'une simulation avec 5 robots roulants et 5 drones volants est aussi disponible à cette adresse : youtu.be/YLDb-ti9bjU. Les simulations réalisées ici pourraient être rejouées sur des robots réels afin de confronter les résultats à la réalité. En effet, si la navigation simulée des drones volants est relativement proche de la réalité, l'état du sol d'un vrai terrain pourrait changer radicalement les résultats des drones roulants. Enfin, si la partie simulation est opérationnelle, la mise en oeuvre de celle-ci n'est pas optimale puisqu'elle manque d'interface graphique et de traitement automatisé des résultats. En effet, il est nécessaire de configurer le fichier de lancement pour chaque type de simulation et les résultats ne sont compilés que pour une simulation. De futurs travaux permettront l'automatisation de l'extraction des résultats afin d'établir plus simplement des statistiques pour un grand nombre de simulations.

Chapitre 6

Conclusion générale

Les systèmes multi-robots sont des systèmes complexes, mais prometteurs dans de multiples domaines. Les nombreux travaux académiques sur ce thème attestent de l'importance qu'ils auront dans le futur. Cependant, si ces promesses sont réelles, elles ne sont pas encore réalisées comme en témoigne le faible nombre de systèmes multi-robots utilisés dans l'industrie. Pourtant des solutions existent afin de permettre aux industriels et académiques de travailler ensemble à cette problématique.

Nous avons présenté dans nos travaux une nouvelle approche du problème. Nous avons choisi une approche partant de l'existant et de problématique d'ingénierie plutôt que par des aspects théoriques. À l'issue de ce travail de thèse, nous avons contribué à proposer une analyse critique des méthodes, outil et prototype dans le domaine des MRS et mise en évidence des limitations qui freinent l'adoption par les industriels des solutions MRS développées par les académiques. Cette analyse mise au regard des attentes industrielles nous a permis de souligner l'inadéquation entre le positionnement du problème de conception par les académiques et les réels obstacles auxquels sont confrontés les industriels.

La réalisation d'un essaim hétérogène en partant de l'aspect théorique, puis en passant par la simulation pour finir par des démonstrations réelles a permis de mettre en évidence les véritables obstacles de conception et d'usage des MRS. En effet, les solutions actuelles se révèlent souvent trop coûteuses matériellement et trop complexes à mettre en oeuvre ce qui empêche les validations expérimentales complètes des algorithmes issus des laboratoires.

Sur la base des travaux du démonstrateur d'essaim hétérogène, nous avons réalisé de nombreuses contributions sur un autopilote open source pour micro véhicules. Ces contributions ont eu pour but d'étendre ses capacités pour le rendre compatible avec l'utilisation en système multi-robots et avec le framework de référence en robotique ROS. En effet, l'utilisation de cet autopilote permet une simplification des fonctions de base attendue des robots afin de pouvoir les utiliser en groupe. Associés à ces contributions, le simulateur associé a aussi été modifié d'une part pour des simulations multi véhicules et d'autre part pour être utilisé afin d'étendre les capacités d'un simulateur de référence en robotique Gazebo.

En utilisant les réalisations précédentes, nous avons contribué à créer une nouvelle

plateforme ouverte à la fois pour les simulations et les démonstrations réelles devant permettre des travaux plus faciles sur les MRS en particulier ceux avec plusieurs types de robots comme des drones volants. Si ces contributions sont prometteuses, elles souffrent néanmoins de quelques faiblesses d'utilisations. En effet, du fait du peu de temps ingénieux qui a pu leur être accordé, les solutions proposées ne sont pas complètes au niveau documentation et mise à jour. Elles manquent aussi d'IHM pour la partie simulation. Nous espérons pouvoir continuer à développer ces logiciels afin de les améliorer et contribuer à leur adoption.

Enfin, nous avons montré l'intérêt de nos contributions logicielles par une démonstration de patrouilles multi-robots en mettant en avant les performances au regard du coût des solutions envisagées.

En dépit des difficultés rencontrer avec le partenaire industriel, qui ne nous a pas permis de validation expérimentale, nous avons pu présenter nos travaux sur plusieurs conférences. Cependant, les travaux réalisés ont été proposés au plus grand nombre sous licences open source, ce qui a permis leur fort usage à la fois académique et industriel depuis leurs publications. Ce qui montre l'intérêt et d'attente qu'ils suscitaient.

Perspectives

Nous n'avons pu mettre en oeuvre les solutions que sur un petit nombre de robots avec des algorithmes simples. Il serait donc intéressant de reprendre les algorithmes de référence dans plusieurs domaines d'application des MRS pour les tester sur la solution proposée.

Il a pu être remarqué que nos travaux n'ont principalement parlé que de robots roulants et multirotors. En effet, le manque d'expérience sur les autres types de plateformes a empêché de présenter leur usage. Cependant, du fait de l'utilisation de l'autopilote ArduPilot, il est possible de les inclure. Ainsi, des travaux futurs pourront porter sur la mise en oeuvre de nouveau type de vecteurs dont les mélanges peuvent faire la force des MRS.

Nous nous sommes beaucoup appuyés sur des solutions existantes. Cependant, nous avons pu remarquer que les simulations complètes sont peu nombreuses. En effet, elles manquent de modèles sur de nombreux capteurs ou actionneurs. Dans notre cas, les modèles de batterie et de transmission sont réduits au plus simple et les stratégies de gestion des MRS adaptés en conséquence. Nous pensons la mise à disposition de nouveaux modèles permettra des simulations plus fidèles à la réalité et permettant l'exploration de nouveaux algorithmes pour la gestion des MRS.

Enfin, les solutions proposées permettent d'utiliser les mêmes logiciels et commandes pour la simulation et les robots réels. Nous espérons voir de futurs travaux sur la combinaison des deux par mixage de la simulation et la réalité. En effet, cette approche permettra d'augmenter les modèles simulés avec des données réelles, mais aussi une réduction des coûts de démonstration tout en permettant d'atteindre un MRS de taille conséquente.

Bibliographie

[1] Evan Ackerman. SPIDER, 2016. URL http://spectrum.ieee.org/automaton/robotics/industrial-robots/how-lockheed-martin-spider-blimp-\fixing-robot-works.

[2] airoboticsdrones. airoboticsdrones, 2018. URL "https://www.airoboticsdrones.com/".

[3] Amazonrobotics. Amazonrobotics, 2015. URL https://www.amazonrobotics.com/.

[4] Ammarf. omnirover, 2018. URL "https://www.youtube.com/watch?v=GqyqBl8X9GQ&feature=youtu.be".

[5] ArduPilot. Autotest servers, 2018. URL "http://autotest.ardupilot.org/".

[6] Ardupilot. Ardupilot, 2018. URL http://ardupilot.org/ardupilot/index.html.

[7] ArduPilot. Ardupilot precision landing, 2018. URL "http://ardupilot.org/copter/docs/precision-landing-with-irlock.html".

[8] ArduPilot. Balancebot, 2018. URL "https://discuss.ardupilot.org/t/gsoc-2018-balance-bot-part-ii/31934".

[9] Ronald C Arkin, Tucker Balch, et al. Cooperative multiagent robotic systems. *Artificial Intelligence and Mobile Robots*, pages 277–295, 1998.

[10] R Arumugam, V R Enti, L Bingbing, W Xiaojun, K Baskaran, F F Kong, A S Kumar, K D Meng, and G W Kit. DAvinCi : A cloud computing framework for service robots. *2010 IEEE International Conference on Robotics and Automation, ICRA 2010*, pages 3084–3089, may 2010. ISSN 10504729. doi : 10.1109/ROBOT.2010.5509469. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-77955831617{&}partnerID=40{&}md5=475cec250602b92879b9751beb70f40b.

[11] AzurDrones. Skeyetech, 2018. URL "http://dronesguard.azurdrones.com/skeyetech/".

[12] Nikolai WF Bode, Daniel W Franks, and A Jamie Wood. Making noise : emergent stochasticity in collective motion. *Journal of theoretical biology*, 267(3) :292–299, 2010.

[13] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence : from natural to artificial systems*. Oxford university press, 1999. ISBN 0195131584.

[14] Boston Dynamics. Atlas - The Agile Anthropomorphic Robot, 2016. URL `http://www.bostondynamics.com/robot{_}Atlas.html`.

[15] S.C. Botelho and R. Alami. M+ : a scheme for multi-robot cooperation through negotiated task allocation and achievement. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 2(May) :1234–1239, 1999. ISSN 1050-4729.

[16] Peter Brass, Flavio Cabrera-Mora, Andrea Gasparri, and Jizhong Xiao. Multirobot tree and graph exploration. *IEEE Transactions on Robotics*, 27(4) :707–717, 2011. ISSN 15523098.

[17] Axel Bürkle, Florian Segor, and Matthias Kollmann. Towards autonomous micro UAV swarms, 2011. ISSN 09210296.

[18] Y.U. Cao, A.S. Fukunaga, A.B. Kahng, and F. Meng. Cooperative mobile robotics : antecedents and directions. *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, 23 :226–234, 1997. ISSN 0929-5593.

[19] UAV Challenge. Uav challenge, 2018. URL `"https://uavchallenge.org/"`.

[20] H. Jacky Chang, C. S George Lee, Y. Charlie Hu, and Yung Hsiang Lu. Multi-robot SLAM with topological/metric maps. *IEEE International Conference on Intelligent Robots and Systems*, pages 1467–1472, 2007.

[21] T. H. Chung, M. R. Clement, M. A. Day, K. D. Jones, D. Davis, and M. Jones. Live-fly, large-scale field experimentation for large numbers of fixed-wing uavs. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1255–1262, May 2016. doi : 10.1109/ICRA.2016.7487257.

[22] Matthew Coombes, William Eaton, Owen McAree, and Wen Hua Chen. Development of a generic network enabled autonomous vehicle system. *2014 UKACC International Conference on Control, CONTROL 2014 - Proceedings*, pages 621–627, 2014. doi : 10.1109/CONTROL.2014.6915211.

[23] Gilles Coppin. Controlling Swarms of Unmanned Vehicles through User-Centered Commands. In *AAAI Fall Symposium : Human Control of Bioinspired Swarms*, pages 21–25, 2012.

[24] Micael S. Couceiro, Rui P. Rocha, and Nuno M. F. Ferreira. A novel multi-robot exploration approach based on Particle Swarm Optimization algorithms. *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 327–332, 2011.

[25] Cyberbotics. Webots, 2018. URL "http://www.cyberbotics.com/".

[26] DARPAtv. Service academies swarm challenge live-fly competition, 2017. URL "https://www.youtube.com/watch?v=RZ-CKA4fUhg".

[27] Geert De Cubber. ICARUS Consortium - Providing Unmanned Search and Rescue Tools. *Remotely Piloted Aircraft Systems - The Global Perspective - Yearbook 2013/2014*, pages 133–134, 2013.

[28] Julian De Hoog. *Role-Based Multi-Robot Exploration*. PhD thesis, University of Oxford, 2011.

[29] Julian De Hoog, Stephen Cameron, and Arnoud Visser. Role-based autonomous multi-robot exploration. In *Computation World : Future Computing, Service Computation, Adaptive, Content, Cognitive, Patterns, ComputationWorld 2009*, pages 482–487, 2009. ISBN 9780769538624.

[30] DJI. DJI, 2017. URL http://www.dji.com/fr.

[31] Rajesh Doriya, Pavan Chakraborty, and G. C. Nandi. 'Robot-Cloud' : A framework to assist heterogeneous low cost robots. *Proceedings - 2012 International Conference on Communication, Information and Computing Technology, ICCICT 2012*, pages 1–5, oct 2012. doi : 10.1109/ICCICT.2012.6398208. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6398208.

[32] M Duarte, V Costa, J Gomes, T Rodrigues, F Silva, S M Oliveira, and A L Christensen. Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots. *PLOS ONE*, 11(3) :1–40, mar 2015. ISSN 1932-6203.

[33] Frederick Ducatelle, Gianni A. Di Caro, Carlo Pinciroli, Francesco Mondada, and Luca Gambardella. Communication assisted navigation in robotic swarms : Self-organization and cooperation. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4981–4988, 2011. ISSN 2153-0858.

[34] Frederick Ducatelle, Gianni A. Di Caro, Carlo Pinciroli, and Luca M. Gambardella. Self-organized cooperation between robotic swarms, 2011. ISSN 1935-3812.

[35] Gregory Dudek and Evangelos Jenkin, Michael, Milios. A Taxonomy of Multirobot Systems. *Autonomous Robots*, 3(4) :375—-397, 1996.

[36] Gregory Dudek, Michael R M Jenkin, Evangelos Milios, and David Wilkes. A Taxonomy for Multi-Agent Robotics. *Autonomous Robots*, 397 :375–397, 1993. ISSN 09295593.

[37] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan. Modular openrobots simulation engine : Morse. In *Proceedings of the IEEE ICRA*, 2011.

[38] Nadia Eghbal. Methodologies for measuring project health, 2018. URL `https://nadiaeghbal.com/project-health`.

[39] SPH Engineering. "ugcs", 2018. URL `"https://www.ugcs.com/"`.

[40] Alessandro Farinelli, Alessandro Farinelli, Luca Iocchi, Luca Iocchi, Daniele Nardi, and Daniele Nardi. Multi Robot Systems : A Classification Focused on Coordination. *October*, 34(5) :2015–2028, 2004.

[41] Alessandro Farinelli, Nicolo Boscolo, Elena Zanotto, and Enrico Pagello. Advanced approaches for multi-robot coordination in logistic scenarios. *Robot. Auton. Syst.*, 90(C) :34–44, April 2017. ISSN 0921-8890. doi : 10.1016/j.robot.2016.08.010. URL `https://doi.org/10.1016/j.robot.2016.08.010`.

[42] Gianpiero Francesca and Mauro Birattari. Automatic design of robot swarms : achievements and challenges. *Frontiers in Robotics and AI*, 3 :29, 2016.

[43] Antonio Franchi. *Human-Collaborative Schemes in the Motion Control of Single and Multiple Mobile RobotsMobile robot*, pages 301–324. Springer International Publishing, Cham, 2017. ISBN 978-3-319-40533-9. doi : 10.1007/978-3-319-40533-9_13. URL `http://dx.doi.org/10.1007/978-3-319-40533-9_13`.

[44] Antonio Franchi, Luigi Freda, Giuseppe Oriolo, and Marilena Vendittelli. A randomized strategy for cooperative robot exploration. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 768–774, 2007. ISBN 1424406021.

[45] Antonio Franchi, Luigi Freda, Giuseppe Oriolo, and Marilena Vendittelli. The sensor-based random graph method for cooperative robot exploration. *IEEE/ASME Transactions on Mechatronics*, 14(2) :163–175, 2009. ISSN 10834435.

[46] Ayush Gaud. Gsoc 2018 : Realtime mapping and planning for collision avoidance, 2018. URL `"https://discuss.ardupilot.org/t/gsoc-2018-realtime-mapping-and-planning-for-collision-avoidance/29864"`.

[47] Russell Gayle, Avneesh Sud, Ming C. Lin, and Dinesh Manocha. Reactive deformation roadmaps : Motion planning of multiple robots in dynamic environments. *IEEE International Conference on Intelligent Robots and Systems*, pages 3777–3783, 2007.

[48] B. P. Gerkey. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, 23(9) :939–954, 2004. ISSN 0278-3649.

[49] Brian Gerkey, Richard T Vaughan, and Andrew Howard. The player/stage project : Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th international conference on advanced robotics*, volume 1, pages 317–323, 2003.

[50] Brian P Gerkey and Maja J Mataric. Murdoch : Publish / Subscribe Task Allocation for Heterogeneous Agents. *International Conference on Autonomous Agents*, pages 203–204, 2000.

[51] David Gouaillier, Vincent Hugel, Pierre Blazevic, Chris Kilner, Jerome Monceaux, Pascal Lafourcade, Brice Marnier, Julien Serre, and Bruno Maisonnier. Mechatronic design of NAO humanoid. In *2009 IEEE International Conference on Robotics and Automation*, pages 769–774, 2009. ISBN 978-1-4244-2788-8. doi : 10.1109/ROBOT.2009.5152516. URL `http://ieeexplore.ieee.org/document/5152516/`.

[52] M. Hamer and C. Ortega-Sanchez. Simulation platform for the evaluation of robotic swarm algorithms. In *TENCON 2010 - 2010 IEEE Region 10 Conference*, pages 1583–1588, Nov 2010. doi : 10.1109/TENCON.2010.5686044.

[53] Mostafa Hassanalian and Abdessattar Abdelkefi. Classifications, applications, and design challenges of drones : a review. *Progress in Aerospace Sciences*, 91 :99–131, 2017.

[54] HelicoMicro. ventes-drones-2016, 2016. URL `https://www.helicomicro.com/2016/03/23/ventes-drones/`.

[55] William Hermans. Beaglebone black, 2015. URL `"https://groups.google.com/forum/#!topic/beagleboard/I2C-34ISae0"`.

[56] a. Howard. Experiments with a Large Heterogeneous Mobile Robot Team : Exploration, Mapping, Deployment and Detection. *The International Journal of Robotics Research*, 25(5-6) :431–447, 2006. ISSN 0278-3649.

[57] Dominique Hunziker, Mohanarajah Gajamohan, Markus Waibel, and Raffaello D'Andrea. Rapyuta : The RoboEarth cloud engine. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 438–444, 2013. ISSN 10504729. doi : 10.1109/ICRA.2013.6630612.

[58] Intel. Inteljoule discontinued, 2017. URL `"https://communities.intel.com/docs/DOC-112092"`.

[59] Intel. Intel Shooting Star, 2018. URL `https://www.intel.fr/content/www/fr/fr/technology-innovation/aerial-technology-light-show.html`.

[60] Luca Iocchi, Luca Marchetti, and Daniele Nardi. Multi-robot patrolling with coordinated behaviours in realistic environments. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2796–2801, 2011. ISBN 9781612844541.

[61] K-Team. khepera-iii, 2007. URL `http://www.k-team.com/mobile-robotics-products/old-products/khepera-iii`.

[62] Pierre Kancir. ardupilot_gazebo, 2017. URL `https://github.com/khancyr/ardupilot_gazebo`.

[63] Pierre Kancir. gazebo_ardu, 2017. URL `https://github.com/khancyr/gazebo_ardu`.

[64] Pierre Kancir. Ardupilot leader follower video, 2018. URL `"https://www.youtube.com/watch?v=ew0ImLIQwNY&list=PL6sCNLbHuYxYyFWLTq3E-R2cV1CrEoYOe&t=0s&index=15"`.

[65] Pierre Kancir. Ardupilot precision landing on rover video, 2018. URL `"https://www.youtube.com/watch?v=qEO6ef53Pyw&list=PL6sCNLbHuYxYyFWLTq3E-R2cV1CrEoYOe&t=0s&index=8"`.

[66] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup, 1997.

[67] N. Koenig and a. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 3 :2149–2154, 2004.

[68] G Ayorkor Korsah, Anthony Stentz, and M Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12) :1495–1512, 2013. ISSN 0278-3649. URL `http://ijr.sagepub.com/content/32/12/1495.abstract`.

[69] A. R. Kuroswiski, N. M. F. de Oliveira, and É. H. Shiguemori. Autonomous long-range navigation in gnss-denied environment with low-cost uav platform. In *2018 Annual IEEE International Systems Conference (SysCon)*, pages 1–6, April 2018.

[70] Francois Legras, Arnaud Glad, Olivier Simonin, and François Charpillet. Partage d'autorité dans un essaim de drones auto-organisé. In *16es Journées Francophones des Systèmes Multi-Agents-JFSMA'08*, 2008.

[71] Yong Li, Shufei Du, and Younghan Kim. Robot swarm manet cooperation based on mobile agent. In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pages 1416–1420. IEEE, 2009.

[72] Tim C Lueth and Thomas Laengle. Task Description, Decomposition, and Allocation in a Distributed Autonomous Multi-Agent Robot System. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and The Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, pages 1516–1523, 1994. ISBN 0-7803-1933-8.

[73] Ali Marjovi and Lino Marques. Multi-robot topological exploration using olfactory cues. *Springer Tracts in Advanced Robotics*, 83 STAR :47–60, 2012.

[74] M. Anwar Ma'sum, Grafika Jati, M. Kholid Arrofi, Adi Wibowo, Petrus Mursanto, and Wisnu Jatmiko. Autonomous quadcopter swarm robots for object localization and tracking. *Mhs2013*, pages 1–6, November 2013.

[75] Makoto Matsumoto and Takuji Nishimura. Mersenne twister : a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1) :3–30, 1998.

[76] MAVLink. MAVLink, 2017. URL `https://mavlink.io/en/`.

[77] Mavros. Mavros, 2018. URL `https://github.com/mavlink/mavros`.

[78] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. PX4 : A node-based multithreaded open source robotics framework for deeply embedded platforms. *2015 IEEE International Conference on Robotics and Automation*, pages 6235–6240, 2015. ISSN 1050-4729. doi : 10.1109/ICRA.2015.7140074. URL `http://www.inf.ethz.ch/personal/lomeier/publications/px4{_}autopilot{_}icra2015.pdf`.

[79] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, and Oskar von Stryk. Comprehensive simulation of quadrotor uavs using ros and gazebo. In *3rd Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, page to appear, 2012.

[80] M.Oborne. "mission planner", 2018. URL `"http://ardupilot.org/planner/"`.

[81] Sepehr MohaimenianPour. Gsoc 2018 : Complex autonomous tasks onboard a uav using a monocular camera (nvidia redtail), 2018. URL `https://discuss.ardupilot.org/t/gsoc-2018-complex-autonomous-tasks-onboard-a-uav-using-a-monocular-camera-nvidia-redtail/31933`.

[82] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, 1(1) :59–65, 2009.

[83] Yash Mulgaonkar, Gareth Cross, and Vijay Kumar. Design of small, safe and robust quadrotor swarms. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2208–2215. IEEE, 2015.

[84] Thomas Nestmeyer, Paolo Robuffo Giordano, Heinrich H. Bülthoff, and Antonio Franchi. Decentralized simultaneous multi-target exploration using a connected network of multiple robots. *Autonomous Robots*, 41(4) :989–1011, 2017. ISSN 1573-7527. doi : 10.1007/s10514-016-9578-9. URL http://dx.doi.org/10.1007/s10514-016-9578-9.

[85] NVIDIA. Jetson pwm, 2018. URL "http://elinux.org/Jetson/PWM".

[86] ONR. Navy Boat, 2014. URL http://www.onr.navy.mil/en/Media-Center/Press-Releases/2014/autonomous-swarm-boat-unmanned-caracas.aspx.

[87] OSRF. ROS2.0, 2017. URL http://design.ros2.org/.

[88] OSRF. Gazebo can model, 2018. URL "https://bitbucket.org/osrf/gazebo_models/src/9533d55593096e7ebdfb539e99d2bf9cb1bff347/coke_can/model.sdf?at=default&fileviewer=file-view-default".

[89] PaparazziUAV. PaparazziUAV, 2018. URL https://github.com/paparazzi/paparazzi.

[90] Lynne E Le Parker. Current research in multirobot systems. *Artificial Life and Robotics*, pages 1–5, 2003. URL http://link.springer.com/article/10.1007/BF02480877.

[91] Parrot. ARDrone, 2012. URL https://www.parrot.com/fr/drones/parrot-ardrone-20-elite-edition.

[92] phoronix. 8-Way ARM Board Linux Benchmark Comparison From The Pi Zero and ODROID To Tegra, 2016. URL "https://www.phoronix.com/scan.php?page=article&item=8way-arm-sbc&num=1".

[93] Charles Pippin, Henrik Christensen, and Lora Weiss. Performance based task assignment in multi-robot patrolling. *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, page 70, 2013.

[94] Pixiel. drones-puy-du-fou-neopter, 2016. URL http://www.pixiel.com/drones-puy-du-fou-neopter.

[95] David Portugal. patrolling_sim, 2018. URL "https://github.com/davidbsp/patrolling_sim".

[96] David Portugal and Rui P. Rocha. On the performance and scalability of multi-robot patrolling algorithms. In *9th IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2011*, pages 50–55, 2011. ISBN 9781612847696.

[97] David Portugal and Rui P. Rocha. Distributed multi-robot patrol : A scalable and fault-tolerant framework. *Robotics and Autonomous Systems*, 61(12) :1572–1587, 2013. ISSN 09218890.

[98] David "Portugal and Rui P." Rocha. "cooperative multi-robot patrol with bayesian learning". *"Autonomous Robots"*, "40"("5") :"929–953", "Jun" "2016". ISSN "1573-7527". doi : "10.1007/s10514-015-9503-7". URL "https://doi.org/10.1007/s10514-015-9503-7".

[99] David "Portugal, Luca Iocchi, and Alessandro" Farinelli. *"A ROS-Based Framework for Simulation and Benchmarking of Multi-robot Patrolling Algorithms"*, pages "3–28". "Springer International Publishing", "2019". ISBN "978-3-319-91590-6". doi : "10.1007/978-3-319-91590-6_1". URL "https://doi.org/10.1007/978-3-319-91590-6_1".

[100] David Portugal, Charles Pippin, Rui P Rocha, and Henrik Christensen. Finding optimal routes for multi-robot patrolling in generic graphs. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 363–369. IEEE, 2014.

[101] William Premerlani and Paul Bizard. Direction cosine matrix imu : Theory. *Dcm-Imu-Theory*, pages 1–30, 2009. ISSN 0196-6553. doi : 10.1002/ejoc.201200111.

[102] QGROUNDCONTROL PROJECT. "qgroundcontrol", 2018. URL "http://qgroundcontrol.com/".

[103] Amanda Prorok, Alexander Bahr, and Alcherio Martinoli. Low-cost collaborative localization for large-scale multi-robot systems. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4236–4241, 2012. ISSN 10504729.

[104] Morgan Quigley, Ken Conley, Brian Gerkey, Josh FAust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Mg. ROS : an open-source Robot Operating System. *Icra*, 3(Figure 1) :5, 2009. ISSN 0165022X.

[105] Robert Reid, Andrew Cann, Calum Meiklejohn, Liam Poli, Adrian Boeing, and Thomas Braunl. Cooperative multi-robot navigation, exploration, mapping and object detection with ROS. *2013 IEEE Intelligent Vehicles Symposium (IV)*, (Iv) : 1083–1088, June 2013.

[106] Craig W Reynolds. Steering behaviors for autonomous characters. In *Game developers conference*, volume 1999, pages 763–782. Citeseer, 1999.

[107] RoboCup. Robocup, 2017. URL "http://www.robocup.org/".

[108] Coppelia Robotics. V-rep, 2018. URL "http://www.coppeliarobotics.com/".

[109] robotis. "turtlebot 3", 2018. URL "http://www.robotis.us/turtlebot-3/".

[110] Martin Saska, Jan Chudoba, Libor Precil, Justin Thomas, Giuseppe Loianno, Adam Tresnak, Vojtech Vonasek, and Vijay Kumar. Autonomous deployment of swarms of micro-aerial vehicles in cooperative surveillance. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 584–595. IEEE, 2014.

[111] Thomas B Sheridan and William L. Verplank. Human and Computer Control of Undersea Teleoperators. Technical report, ManMachine Systems Lab Department of Mechanical Engineering MIT, 1978.

[112] Glen A Simon, Jared M Moore, Anthony J Clark, and Philip K McKinley. Evoros : integrating evolution and the robot operating system. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1386–1393. ACM, 2018.

[113] Rajnesh Kumar Singh and Neelu Jain. Comparative Study of Multi-Robot Area Exploration Algorithms. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(8) :778–786, 2014. URL http://www.ijarcsse.com/docs/papers/Volume{_}4/8{_}August2014/V4I7-0259.pdf.

[114] A. Sinisterra, M. Dhanak, and N. Kouvaras. A usv platform for surface autonomy. In *OCEANS 2017 - Anchorage*, pages 1–8, Sept 2017.

[115] Thomas Sousselier and Christian Prins. *Conception et validation d ' un algorithme de mise en formation d ' essaim de Application à l ' exploration de zone en guerre des mines*. PhD thesis, UBS, 2013.

[116] SPI. Software in the public interest, inc. 2017 annual report, 2017. URL https://www.spi-inc.org/corporate/annual-reports/2017.pdf.

[117] AUVSI SUAS. Auvsi suas, 2018. URL "http://www.auvsi-suas.org/".

[118] Fang Tang and Lynne E. Parker. ASyMTRe : Automated synthesis of multi-robot task solutions through software reconfiguration. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2005, pages 1501–1508, 2005. ISBN 078038914X.

[119] Artem S. Tashkinov. Major linux problems on the desktop, 2014. URL "http://itvision.altervista.org/why.linux.is.not.ready.for.the.desktop.current.html".

[120] Christopher Steven Timperley, Afsoon Afzal, Deborah S Katz, Jam Marcos Hernandez, and Claire Le Goues. Crashing simulated planes is cheap : Can simulation detect robotics bugs early ? In *Software Testing, Verification and Validation (ICST), 2018 IEEE 11th International Conference on*, pages 331–342. IEEE, 2018.

[121] Matthew Turpin, Nathan Michael, and Vijay Kumar. Capt : Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1) :98–112, 2014. doi : 10.1177/0278364913515307.

[122] Ubuntu. robot-operating-system-choice, 2018. URL `https://www.ubuntu.com/engage/robot-operating-system-choice?utm_source=twitter_social&utm_medium=social&utm_campaign=FY19_IOT_Robotics_Whitepaper_OSconsiderations`.

[123] Wikipedia. Heartbleed, 2014. URL `"https://en.wikipedia.org/wiki/Heartbleed"`.

[124] Wikipedia. Comparison of single-board computers, 2016. URL `"https://en.wikipedia.org/wiki/Comparison_of_single-board_computers"`.

[125] Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. In *Proceedings of the 19th National Conference on Innovative Applications of Artificial Intelligence - Volume 2*, IAAI'07, pages 1752–1759. AAAI Press, 2007. ISBN 978-1-57735-323-2. URL `http://dl.acm.org/citation.cfm?id=1620113.1620125`.

[126] Chuanbo Yan and Tao Zhang. Multi-robot patrol : A distributed algorithm based on expected idleness. *International Journal of Advanced Robotic Systems*, 13(6) : 1729881416663666, 2016.

[127] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10, 2013. ISSN 17298806. doi : 10.5772/57313.

[128] Zhi Yan, Luc Fabresse, Jannik Laval, and Noury Bouraqadi. Team size optimization for multi-robot exploration. In Davide Brugali, Jan F. Broenink, Torsten Kroeger, and Bruce A. MacDonald, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, pages 438–449, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11900-7.

[129] Zhi Yan, Luc Fabresse, Jannik Laval, and Noury Bouraqadi. Metrics for performance benchmarking of multi-robot exploration. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem :3407–3414, 2015. ISSN 21530866. doi : 10.1109/IROS.2015.7353852.

[130] Emaad Mohamed H Zahugi, Mohamed M Shanta, and T V Prasad. AISC 178 - Oil Spill Cleaning Up Using Swarm of Robots. In *"Advances in Computing and Information Technology"*, pages 215–224. Springer, 2013.

[131] Erol Şahin and Alan Winfield. Special issue on swarm robotics. *Swarm Intelligence*, 2(2-4) :69–72, 2008. ISSN 19353812.