ENSTA Bretagne
2, rue François Verny
29806 BREST cedex
FRANCE
Tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr

3.4&4.4

CI2020

19 mai 2019

Projet ERL2019

Rapport

C. JEGAT
A. SAAD
A. KLIPFEL
O. AL SHUAIBI

Table des matières

1 Contextualisation **7**
- 1.1 Le concours *ERL Emergency Service Robots* . . . 7
- 1.2 Travaux antérieurs . . . 7
- 1.3 Le concours 2019 . . . 7

2 Management de projet **10**
- 2.1 Répartition du travail . . . 10
- 2.2 Outils de projet . . . 10
- 2.3 Écarts de planification . . . 12

3 Vision globale du système **13**
- 3.1 Analyse du besoin . . . 13
  - 3.1.1 Définition du système . . . 13
  - 3.1.2 Exigences . . . 14
- 3.2 Analyse fonctionnelle . . . 14
  - 3.2.1 Analyse externe . . . 14
  - 3.2.2 Cahier des charges fonctionnelles . . . 16
  - 3.2.3 Analyse interne . . . 16
  - 3.2.4 Architecture fonctionnelle . . . 17
- 3.3 Analyse organique . . . 19
  - 3.3.1 Sous-systèmes . . . 19

4 Partie terrestre du système **21**
- 4.1 État de l'art des robots existants . . . 21
  - 4.1.1 Le robot six-roues . . . 21
  - 4.1.2 Le robot *char* . . . 24
  - 4.1.3 Le robot *buggy* . . . 24
- 4.2 Robot choisi . . . 25
- 4.3 Algorithme d'évitement d'obstacles . . . 25
  - 4.3.1 Suivi d'itinéraire . . . 26
  - 4.3.2 Évitement d'obstacles . . . 26
- 4.4 Imagerie *3D* . . . 31
  - 4.4.1 Contextualisation . . . 31
  - 4.4.2 Théorie . . . 31
  - 4.4.3 Pratique . . . 31
  - 4.4.4 Implémentation . . . 32

4.5 Cartographie . . . 32
4.5.1 Démarche . . . 32
4.5.2 Étude de l'existant . . . 32
4.5.3 ROS . . . 33
4.5.4 ROS et la cartographie . . . 36
4.5.5 Intégration sur le robot . . . 36
4.6 Dépôt de la trousse de soin . . . 38
4.7 Le robot six-roues . . . 38

5 Partie aérienne du système **41**
5.1 État de l'art des robots existants . . . 41
5.1.1 Robots existants . . . 41
5.1.2 Architecture d'un quadrirotor . . . 41
5.2 Robots choisis . . . 42
5.3 Travaux sur le *matrice 600* . . . 43
5.4 Modèle théorique d'un quadrirotor . . . 44
5.5 Choix de l'auto-pilote . . . 44
5.5.1 Ensemble des solutions . . . 44
5.5.2 Auto-pilote retenu . . . 45
5.5.3 Les fonctionnalités . . . 45
5.5.4 Configuration de l'auto-pilote . . . 45

6 Tests **47**
6.1 Tests unitaires . . . 47
6.2 Tests d'intégration . . . 47
6.3 Validation Fonctionnelle Usine . . . 48
6.3.1 Contextualisation . . . 48
6.3.2 Analyse des résultats . . . 48

A Cahier des charges **51**

B Cahier des charges fonctionnelles **55**

C Simulation du modèle d'un quadrirotor **65**

D Fiches de tests **67**
D.1 Fiche de test n°1: *lidar* . . . 67
D.2 Fiche de test n°2: *Pololu Maestro* . . . 68
D.3 Fiche de test n°3: *Kinect v2* . . . 69

E Fiche de VFU **70**

Bibliographie **74**

Remerciements

Nous tenons à remercier M. Le Bars pour ses conseils et son encadrement de qualité. Sa présence à chacune de nos séances de projet nous a permis d'échanger aussi régulièrement que possible. Par ailleurs, nous tenons également à remercier M. Dhaussy pour l'aide apportée sur la partie ingénierie système. N'oublions pas les doctorants et enseignant-chercheurs de l'ENSTA-Bretagne qui nous ont aidés dans notre travail, en réalisant une partie des fonctionnalités des robots aériens et terrestres. Citons notamment, M. Joris Tillet, M. Julien Damers, M. Thomas Le Mézo, et M. Alain Bertholom sans qui le succès à la compétition *ERL* n'aurait pu être ce qu'il était.

Résumé

L'objectif du projet *ERL2019* était de participer à l'édition 2019 du concours *ERL Emergency Service Robots*. Cette compétition rallie deux aspects fondamentaux: l'exploration et le sauvetage. Exploration, puisqu'il s'agit de cartographier et d'évoluer en toute autonomie, avec certaines phases manuelles, sur un terrain « dévasté» [1]. Sauvetage, puisque les robots doivent accomplir des missions, comme par exemple délivrer de l'aide à un blessé, cartographier la zone, et détecter des points d'intérêt [2]. Pour ce faire, des robots à la fois terrestre et aérien sont autorisés.

En cette année 2019, la compétition prenait place en Espagne, à Séville du 18 au 23 février. Afin de pouvoir réussir au mieux le concours, notre stratégie a été de concentrer davantage les efforts sur la partie terrestre de la compétition, qui est considérée comme plus importante par le concours. De l'équipe projet, *Corentin Jegat* et *Arnaud Klipfel* ont travaillé principalement sur la partie terrestre, et *Osama Al Shuaibi* et *Ali Saad* sur la partie aérienne [3]. En renfort, des doctorants et enseignant-chercheurs de l'*ENSTA-Bretagne* [4] ont apporté leur aide afin de réaliser certaines fonctionnalités du système *SIC* [5], et comportant une partie aérienne et terrestre.

Concernant le travail réalisé par l'équipe projet, avant et pendant le concours, c'est-à-dire pendant l'*uv 3.4* et le début de l'*uv 4.4*, le binôme chargé de le partie terrestre a principalement réalisé les algorithmes de suivi d'itinéraire, d'évitement d'obstacles, l'ingénierie système qui nous a permis de bénéficier d'une vision globale du système et de clarifier les exigences du concours, un sous-système pour le dépôt d'une trousse de soin, l'architecture électronique du robot terrestre *six-roues*, et son organisation. Les robots terrestres étaient opérationnels pour la compétition. Concernant la partie aérienne, *Osama Al Shuaibi* et *Ali Saad* ont travaillé sur le contrôle d'un robot aérien. Après le concours, ce qui correspond à la fin de l'*uv 4.4* [6], ayant constaté à quel point les algorithmes d'évitement d'obstacles et de suivi d'itinéraire étaient primordiales et que la cartographie *3D* était la seule fonctionnalité non implémentée pour la partie terrestre, *Corentin Jegat* et *Arnaud Klipfel* se sont focalisés sur ces problématiques ci. Parallèlement *Osama Al Shuaibi* et *Ali Saad* ont poursuivi leurs travaux sur la partie aérienne.

Finalement, comme le montre la fiche *VFU* [7], du système *SIC* seule la partie terrestre a été entièrement implémentée, et est presque entièrement fonctionnelle [8]. La partie aérienne doit être poursuivie. Néanmoins, l'objectif principal du projet a été atteint, puisque notre équipe s'est positionnée troisième. La base terrestre développée, principalement le robot *six-roues*, pourra être reprise pour les éditions futures.

1. avec des obstacles.
2. Entrées et sorties de bâtiments exemple.
3. les travaux des différents binômes sont présentés au 4 page 21 pour la partie terrestre, par *Corentin Jegat* et *Arnaud Klipfel* et au 5 page 41 pour la partie aérienne par *Osama Al Shuaibi* et *Ali Saad*.
4. cités page 4.
5. présenté au 3 page 13. C'est le système réalisé pendant ce projet.
6. deuxième moitié.
7. cf. E page 70.
8. Seule la cartographie *3D* doit encore être adaptée à la vitesse du robot terrestre.

Introduction

L'équipe de projet *ERL2019* constituée de Corentin Jegat, Ali Saad, Osama Al Shuaibi, et Arnaud Klipfel, encadrée par un professeur en robotique mobile de l'ENSTA bretagne, M. Le Bars, avait pour but de réaliser un système, le *Système d'Intervention et de Contrôle*[9], afin de contrôler des robots aériens et terrestres, de les faire collaborer, et de leur permettre de réaliser des missions de terrain. L'objectif de ce projet était notamment de participer à l'édition 2019 du concours *ERL Emergency Service Robots*. Finalement, le projet a déjà permis de produire une partie terrestre robuste et réutilisable pour les éditions futures. Notre équipe s'est placée troisième suite au concours.

Le projet se divise en deux fleurons, une partie consacrée aux travaux concernant les robots aériens, et une partie sur les robots terrestres. Ce rapport expose le travail effectué par les membres de l'équipe projet *ERL2019* tout au long des *uv 3.4 et 4.4*. Vous pourrez trouver une explication plus détaillée des enjeux du concours *ERL Emergency Service Robots* au 1 page 7. Une présentation de notre organisation au sein de l'équipe de projet est exposée en 2 page 10. Une définition du *SIC*, le système réalisé, ainsi qu'une étude en termes d'ingénierie système sont disponibles au 3 page 13. Par ailleurs, une présentation des robots terrestres candidats pour l'édition 2019, une explication des algorithmes d'évitement d'obstacles, les travaux d'imagerie et de cartographie *3D*, le sous-système de dépôt de trousses de soin, et le robot terrestre final *six-roues* sont présentés section 4 page 21. D'autre part, une présentation des robots aériens candidats pour l'édition 2019, une explication de la prise en main d'un modèle de drone quadrirotor, et le détail des travaux sur l'asservissement des robots aériens sont présentés section 5 page 41. Finalement, les tests unitaires, d'intégration, et la *validation fonctionnelle usine* sont exposés section 6 page 47.

Vous pourrez également vous-même remarquer, au cours de la lecture de ce rapport, que notre équipe de projet a pris conscience de l'importance de l'ingénierie système à la fois dans l'organisation de l'équipe et dans la définition du système à réaliser, ainsi que de l'importance des tests lors du développement d'un système technologique.

Mots clés

robotique mobile, autonomie, sauvetage, exploration, cartographie, collaboration inter-robots.

9. cf. 3.1.1 page 13.

Chapitre 1

Contextualisation

1.1 Le concours *ERL Emergency Service Robots*

Le concours *ERL*[1] *Emergency Robots* est un événement rassemblant différentes écoles d'ingénieurs et organismes liés au domaine de la robotique. En particulier, ce concours ci a pour but de développer les solutions robotiques existantes dans le cadre de situations d'urgence[2]. L'aspiration principale du concours est d'allier la collaboration de robots aériens, terrestres, et sous-marins. La compétition est composée de différentes missions de terrain, que les robots doivent réaliser en toute autonomie et en respectant certaines contraintes[3]. En outre, les robots terrestres et aériens doivent notamment être capables d'évoluer dans des bâtiments, ce qui rend l'utilisation des techniques de localisation instantanée par *GPS* obsolètes. Les exigences du concours 2019 sont présentées section 3.1.2, et notre cahier des charges en annexe A.1 page 51.

1.2 Travaux antérieurs

Ce n'est pas la première édition du concours *ERL Emergency Service Robots* à laquelle l'ENSTA Bretagne offre sa contribution, l'ENSTA Bretagne avait participé en 2015, et diverses travaux avaient déjà été réalisés à l'ENSTA Bretagne, comme le travail de stage assistant ingénieur de M.Benhnini [Ben17]. De ces innombrables travaux, il reste des robots aériens, terrestres, et sous-marins fonctionnels, ainsi que des techniques et stratégies d'approche de la compétition. Le rôle du projet *ERL2019* n'est donc pas de construire de nouveaux robots à mêmes d'accomplir les missions de terrain de la compétition, mais, en plus de devoir améliorer les robots existants et ainsi les adapter aux missions de l'édition 2019, il faudra construire un système qui permettra de contrôler les robots lors de leur intervention.

Plus précisément, il a fallu dans un premier temps choisir les robots de la compétition[4]. Les solutions des autres équipes[5] n'ont pas été étudiées pour le choix des robots, comme les dates du concours[6] ne nous laissaient pas le temps de développer nos robots. En revanche, les solutions adverses ont été étudiées, afin de trouver des solutions adéquates pour les différentes missions et améliorations.

1.3 Le concours 2019

Pour l'édition 2019 de la compétition, les robots aériens et terrestres concourraient afin de mener à bien une intervention sur une zone « dévastée ». Il n'était donc pas question d'utiliser

1. *European Robotic League.*
2. Telles, par exemple, des catastrophes naturelles, des accidents dans des zones difficiles d'accès ou à risques pour les secouristes.
3. Spécifiées par les organisateurs du concours, ces contraintes sont présentées dans le livret du concours [ERL18].
4. L'étude des robots existants et l'explication du choix sont présentées au 4.1 page 21 et 5.1 page 41.
5. Hors ENSTA.
6. Février 2019.

des robots sous-marins pour cette édition. Comme présenté en section 3.1.1, les robots sélectionnés dans le cadre de notre projet étaient: un robot terrestre [7], et un robot aérien [8]. Ces robots sont, comme précisé section 1.2, des systèmes existants.

En l'occurrence, trois missions étaient à réaliser par les robots aérien et terrestre. Premièrement, les robots, terrestre comme aérien, devaient être capables de produire une cartographie [9] intérieure, de certains bâtiments présents sur la zone d'exploration [10], et extérieure, sous-entendu de la zone à explorer. Deuxièmement, les robots, aériens comme terrestres, devaient être à mêmes de déposer une trousse de soin auprès d'un blessé présent sur la zone d'exploration. Troisièmement, mais non pas moins important pour autant, les robots, aussi bien aériens que terrestres, devaient pouvoir détecter des *OPI* [11], et devaient les reconnaître.

Figure *1.1 :* Image satellite de la zone du concours. Zone d'exploration en jaune. Séville, Espagne. [ERL18]

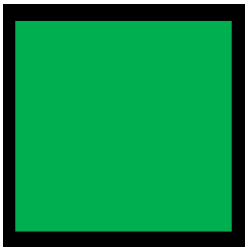


Figure *1.2 :* *OPI* indiquant les entrées autorisées. [ERL18]

7. Un robot terrestre quatre roues a été également sélectionné en secours, mais la mascotte de l'édition 2019 reste le robot six-roues, cf. 4.1.1 page 21.

8. Pour davantage de précisions, veuillez vous référer à la section 4.1 pour les robots terrestres, et à la section 5.1 pour le robot aérien.

9. Seule une cartographie *2D* avait été implémentée à l'époque du concours. La cartographie *3D* a été produite après la compétition, cf. 4.5 page 32.

10. Ce sont en réalité de grands chapiteaux.

11. *Object of Potential Interest*, en d'autres termes des symboles éparpillés sur la zone d'exploration.

Figure *1.3 :* *OPI* indiquant les entrées non-autorisées. [ERL18]

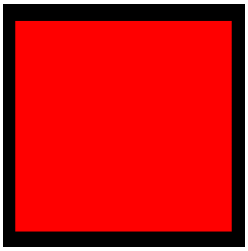


Figure *1.4 :* *OPI* indiquant les dommages bâtiments. [ERL18]

Par ailleurs, tout au long de l'intervention des robots, ces derniers devaient suivre un liste de *waypoints*[12], à la fois donnée par les organisateurs, mais également complétée par les équipes, qui leurs indiquaient quel chemin suivre dans la zone d'exploration, et surtout, où était à déclencher ou à effectuer telle mission.

12. Coordonnées géographiques.

Chapitre 2

Management de projet

2.1 Répartition du travail

L'organisation au sein de l'équipe de projet a été notre première préoccupation, puisqu'elle a permis, dès le début du projet, d'apporter efficacité [1]. L'équipe projet a été partitionnée en deux tandems: Corentin Jegat et Arnaud Klipfel, puis Osama Al Shuaibi, et Ali Saad. Ce choix n'était pas anodin. En effet, comme la compétition demande de travailler sur des robots terrestres et aériens, répartir un groupe [2] sur la gestion des robots terrestres [3], et un autre binôme [4] sur la partie aérienne [5], se montre plus efficace. La partie terrestre étant stratégiquement plus importante que la partie aérienne pour le concours, il était également crucial d'assurer la production d'un robot terrestre fiable. Par ailleurs, l'équipe projet a pu s'apercevoir de l'utilité de l'*ingénierie système* en termes de vision globale du système et de compréhension du système.

2.2 Outils de projet

En plus d'une répartition du travail, au sein de l'équipe, adaptée au projet conduit, l'organisation du projet a été simplifiée par l'utilisation d'un outil numérique en particulier [6], *Trello* ©. Cet outils numérique de gestion de projet permet de manager une équipe par le biais de « tableaux de bord ». Comme l'image 2.1 page 11 le montre, le tableau de bord de projet a été organisé par domaine de projet.

1. Chacun avait un travail qui lui était consacré, ou chaque tandem.
2. M.Jegat et M.Klipfel.
3. Les travaux sur la partie terrestre sont présents au 4.
4. M.Saad, et M.Al Schuaibi.
5. Les travaux sur la partie aérienne sont présents au 5.
6. Notamment, très prisé pour sa simplicité, et sa gratuité.

Figure 2.1 : *Tableau de bord* du projet *ERL2019*.

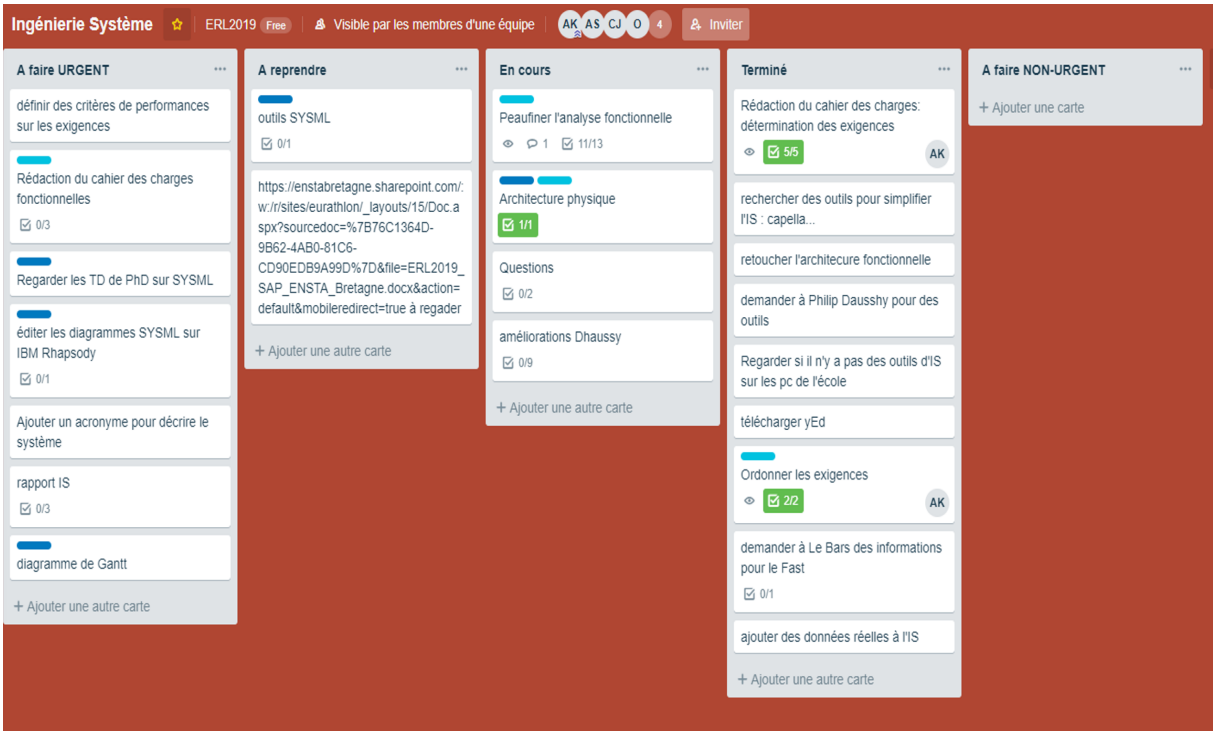


Figure 2.2 : *Tableau de bord* de la partie *Ingénierie Système*.

Par conséquent, il est notable de remarquer que le projet *ERL2019* se découpe en une partie aérienne et une partie terrestre, et que pour chacune de ces parties, des tâches en simulation et pratiques étaient à réaliser. Au sein d'une partie de projet, comme le montre l'image 2.2 page 11, les tâches sont réparties sous forme de liste. Cette dichotomie, émanant d'une analyse en amont de notre projet nous a permis d'organiser l'activité à long termes, de collaborer plus efficacement, et de définir des ordres de priorité.

2.3 Écarts de planification

Nous avions tout d'abord décidé d'un programme prévisionnel. Ainsi, la première phase du projet [7] devait consister, pour la partie terrestre, à réparer le robot *six-roues*, produire le premier modèle de notre système en termes d'ingénierie système, réorganiser son architecture électronique, travailler sur les algorithmes d'autonomie ou d'asservissement, et enfin à réaliser la cartographie . Au niveau de la partie aérienne, il était question de travailler sur les algorithmes d'asservissement et de régler les problèmes de stabilité en vol. La deuxième phase [8] devait consister à peaufiner certains aspects du système après bilan suite à la compétition.

Nous avons rapidement réalisé après quelques semaines, que nous ne pourrions jamais réaliser ce qui avait été initialement prévu. Certains doctorants et enseignants chercheurs nous sont venus en aide. Avant le concours, pour la partie terrestre, nous avons donc concentré nos efforts sur la réparation du robot terrestre, l'ingénierie système initiale, les algorithmes de suivi d'itinéraire, l'organisation du robot six-roues, et sur la réalisation d'un sous-système de dépôt de trousse de soin. Quant à la partie aérienne, les travaux se sont focalisés sur les algorithmes de contrôle. Après la compétition, mûris des expériences du terrain, nous avons amélioré nos solutions relatives aux algorithmes de suivi d'itinéraire, réaliser la cartographie *3D*, et développé l'architecture *ROS*[9] du robot six-roues. Les travaux de la partie aérienne se concentraient toujours sur le contrôle autonome du drone.

Nous avons néanmoins su adapter notre organisation à de nombreux imprévus, et avons finalement réussi à nous placer en troisième position le jour du concours.

7. Jusqu'au début du concours, soit jusqu'au 18 février 2019.
8. Jusque fin d'année.
9. cf. 4.5.3 page 33.

Chapitre 3

Vision globale du système

3.1 Analyse du besoin

3.1.1 Définition du système

Comme le présente le diagramme bête à cornes figure 3.1 page 13, le système réalisé à l'occasion du projet *ERL2019* est un système de contrôle et d'adaptation des robots existants, afin de permettre une réussite des interventions sur le terrain. Par « intervention », il faut comprendre une évolution autonome des robots sauveteurs durant laquelle ils devront accomplir trois types de mission [1]. Ainsi, avant le départ en intervention, les sauveteurs communiquent [2] aux robots sauveteurs une liste de *waypoints* [3], qui permettent aux robots sauveteurs à la fois de connaître leur itinéraire d'intervention, et où les missions doivent être exécutées. Pour la suite de ce rapport, l'acronyme *SIC* pour *Système d'Intervention et de Contrôle* fait référence au système réalisé dans le cadre du projet *ERL2019*.

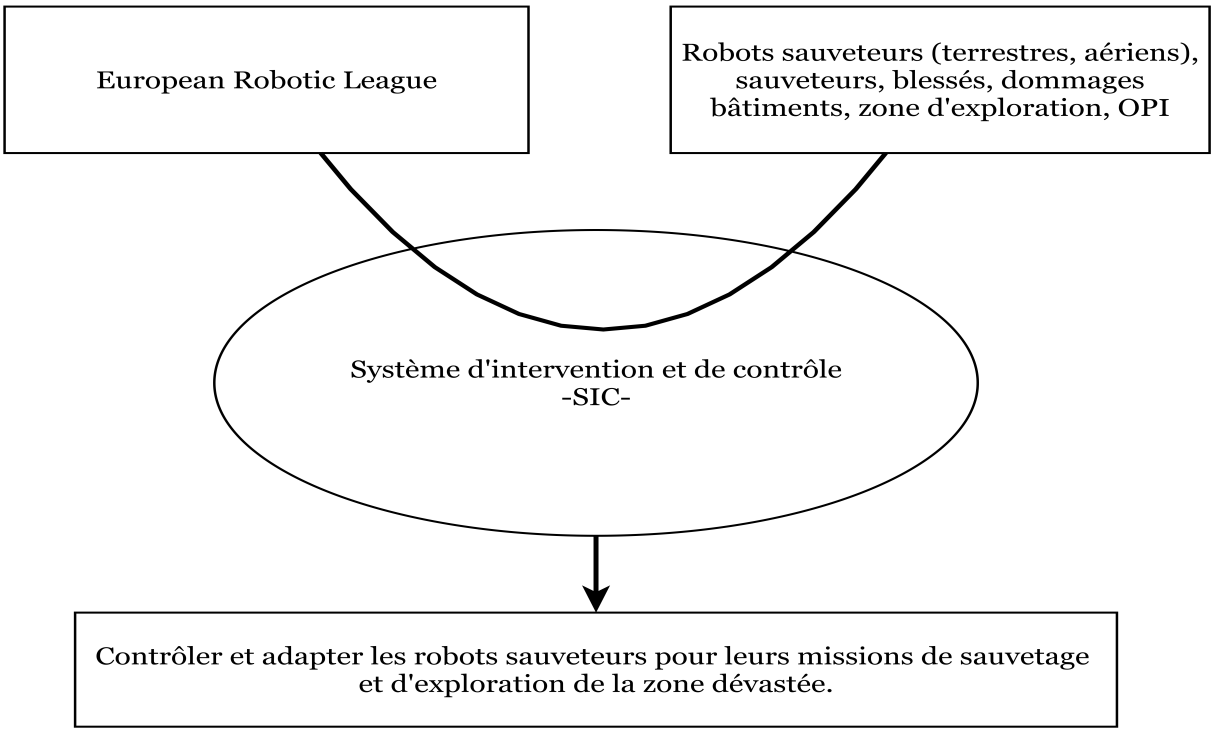


Figure *3.1 :* Diagramme de type *bête à cornes* pour l'analyse du besoin.

1. cf. section 1.1 page 7
2. La communication est réalisée par ordinateur. Les sauveteurs ont accès à l'ordinateur embarqué du robot par le biais d'un bureau à distance. L'architecture électronique d'un robot terrestre est présentée au 4.1.1.2 page 23, et celle d'un robot aérien au 5.1.2 page 41.
3. cf. note de bas de page 12 page 9.

3.1.2 Exigences

Afin de mieux cerner le *SIC*, il a fallu dans un premier temps identifier et reformuler les besoins du concours *ERL2019*[4] et les besoins de M. Le Bars. Pour ce faire, l'étude des règles du concours a été indispensable [ERL18], ce qui a permis, en plus de quelques requêtes de M. Le Bars de définir un cahier des charges pour le projet *ERL2019*, qui est présenté annexe A.1 page 51. Les critères principaux choisis pour les exigences sont: une description, un type[5], et sa raison d'être[6].

Par la suite, les exigences ont été regroupées en cinq *GLE*[7]. Le *GLE1 Analyse Environnement*[8] regroupe l'ensemble des exigences liées aux mesures, et aux capteurs. Le *GLE2 Autonomie* est composé des exigences relatives à l'évolution autonome des robots sauveteurs contrôlés par le *SIC* durant une intervention. Le *GLE3 Missions* traite de l'ensemble des exigences définissant le cadre des missions de terrain[9]. Le *GLE4 Sécurité* rassemble les exigences visant à assurer une évolution autonome en toute sécurité, et de ce fait de rendre l'évolution autonome un minimum contrôlable. Finalement, le *GLE5 Intégration* recense les exigences liées à l'intégrabilité du *SIC* à l'existant. La définition de ces groupes d'exigences à permis un classement des exigences par thématique.

3.2 Analyse fonctionnelle

3.2.1 Analyse externe

Afin de réaliser un cahier des charges fonctionnelles, et donc d'identifier fonctions principales et fonctions de contrainte, une analyse fonctionnelle partant des acteurs du milieu extérieur a dans un premier été menée. Pour simplifier le diagramme pieuvre fourni figure 3.2 page 15, une seule fonction principale *FP1* a été représentée à ce niveau de raffinement. Cette fonction principale a ensuite été décomposée en plusieurs sous fonctions principales, comme l'illustre la section 3.2.2 page 16. Pour une spécification des fonctions principales veuillez vous référer au cahier des charges fonctionnelles en annexe B.2 page 56.

4. Principalement, perçues comme des exigences.
5. Exigence de type contrainte ou fonction.
6. Seuls les critères les plus importants sont présentés dans ce rapport.
7. *Groupe Logique d'Exigences.*
8. cf. annexe A.1 page 51.
9. cf. 1.1 page 7.

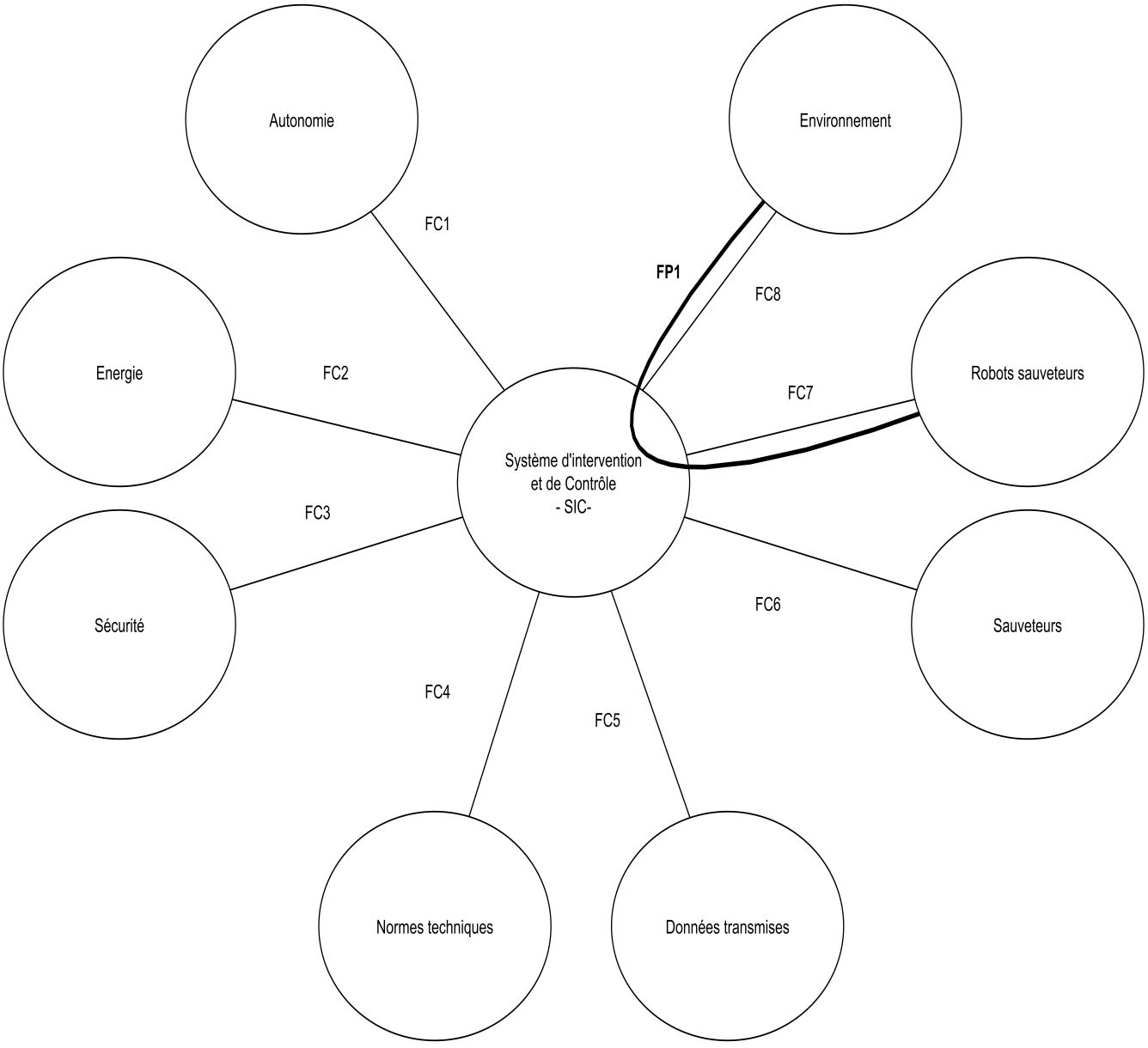


Figure 3.2 : Diagramme pieuvre du *SIC*. Les fonctions principales et de contrainte sont détaillées dans le cahier des charges fonctionnelles annexe B page 55.

3.2.2 Cahier des charges fonctionnelles

Après avoir identifié une fonction principale de haut niveau, *FP1*[10], celle-ci a été décomposée en plusieurs sous-fonctions principales. Dans ce processus, la correspondance avec les *GLE*[11] a été conservée. Par ailleurs, certaines fonctions principales se décomposent en deux fonctions principales liées au contrôle des robots terrestres pour l'une, et au contrôle des robots aériens pour l'autre. En effet, par exemple, la réalisation de la fonction principale *FP1.2.1* sur l'asservissement des robots sauveteurs ne sera pas identique pour un robot aérien et pour un robot terrestre.

De plus, certaines fonctions principales n'ont pas été spécifiées suivant les *trois axes*[12], puisque leur trop haut niveau d'abstraction rendait cette opération incohérente. Il a été préféré de spécifier suivant les trois axes les sous-fonctions principales de moins hauts niveaux d'abstraction. Le cahier des charges fonctionnelles pour sa partie contrainte est disposé en annexe B.1 page 55, et pour sa partie fonctionnalités du *SIC* est disponible en annexe B.2 page 56.

3.2.3 Analyse interne

Après avoir établi un cahier des charges fonctionnelles[13], il a fallu raffiner les fonctions principales en cours de développement pendant la présente phase du projet. Seul le raffinement par *FAST* de la fonction principale 1.2.1.2 concernant l'asservissement des robots terrestres dans leur suivi des *waypoints* est présenté dans ce rapport, étant le plus abouti. Ainsi, la figure 3.3 page 17 montre *comment* les robots terrestres contrôlés par le *SIC* peuvent être asservis[14]. Les robots terrestres doivent être commandés afin de les mener d'un *waypoint* à un autre. A cette fin, les données des capteurs des robots terrestres sont recueillies, analysées, et traitées. Ici, pour simplifier, seule l'analyse des données capteurs permettant de détecter les obstacles[15] a été représentée. Ainsi, après détection des obstacles, et connaissant l'itinéraire à suivre[16], il est possible à l'aide d'algorithmes[17] de calculer la commande à appliquer aux robots terrestres[18], afin de rallier le prochain *waypoint* sans encombres.

10. cf. 3.2.1 page 14.
11. Présentés au 3.1.2 page 14.
12. Temps, comportement, et données E/S.
13. cf. annexe B page 55.
14. Pour davantage de renseignements et de détails techniques sur l'asservissement des robots terrestres, veuillez vous référer au 4.3 page 25.
15. Objets naturels se trouvant entre le robot et son prochain *waypoint*.
16. Ensemble de *waypoints*.
17. présentés au 4.3 page 25.
18. tension des moteurs par exemple.

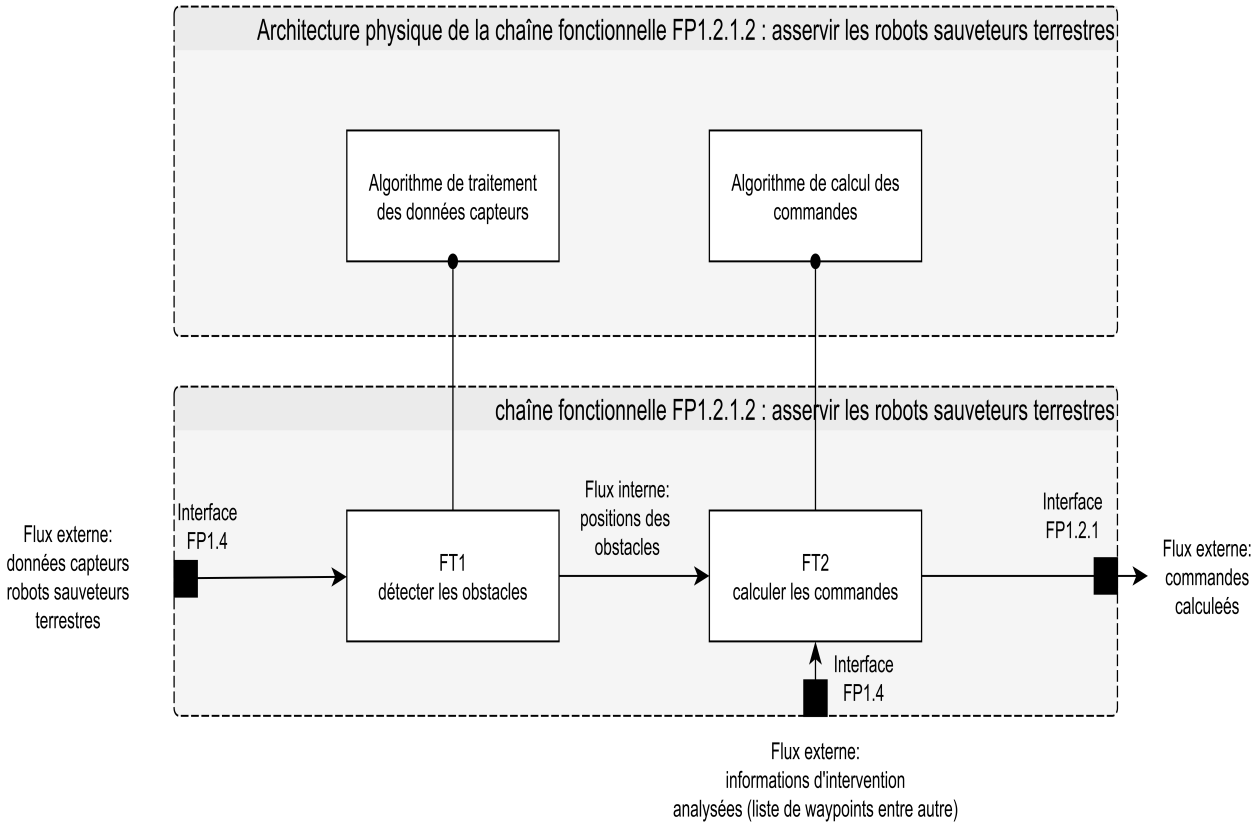


Figure *3.3 :* Raffinement par *FAST* de la fonction principale FP1.2.1.2 en bas (cf. cahier des charges fonctionnelles en annexe B.2 page 56); architecture physique de la chaîne fonctionnelle au-dessus.

3.2.4 Architecture fonctionnelle

Les fonctions principales du *SIC* étant identifiées [19], il est dès lors possible de s'intéresser aux interactions entre les fonctions principales du système.

Pour ce faire les flux entre les fonctions principales ont été étudiés, débouchant sur l'architecture physique présentée figure 3.4 page 18. De l'architecture fonctionnelle de la figure 3.4, il est possible de décrire le fonctionnement voulu du *SIC*. Ainsi, le sauveteur rempli au préalable un fichier contenant les ordres et informations d'intervention. Principalement, le sauveteur communique aux robots un ensemble de *waypoints*, qui constitue l'itinéraire de l'intervention et spécifie à quel moment dans l'intervention les missions doivent être lancées [20]. Dans le cas où les requêtes des sauveteurs sont valides [21], le *SIC* évolue en mode autonome, et par conséquent les robots sauveteurs également. Durant ce « mode autonome », les sauveteurs reçoivent des informations en temps réel [22] de l'état du *SIC*, et sont à mêmes d'arrêter l'intervention quand ils le jugent nécessaire, ainsi que de modifier les ordres d'intervention stipulés initialement.

Au cours de l'évolution autonome, le *SIC* analyse l'environnement à proximité des robots sauveteurs, veille à la sécurité des acteurs extérieurs [23], dirige les robots sauveteurs, et déclenche les missions adéquates en fonction des *waypoints* atteints par les robots sauveteurs. Le *SIC* permet de véhiculer les états et informations relatives aux robots sauveteurs entre eux.

19. cf. cahier des charges fonctionnelles en annexe B page 55.
20. Les trois missions sont rappelées figure 3.4 page 18.
21. Conformes avec les fonctionnalités des robots sauveteurs.
22. Par un ordinateur avec un bureau à distance connecté par *Wifi* sur les robots sauveteurs.
23. Les sauveteurs ou les blessés par exemple. Les acteurs extérieurs sont mentionnés dans le diagramme pieuvre figure 3.2 page 15.

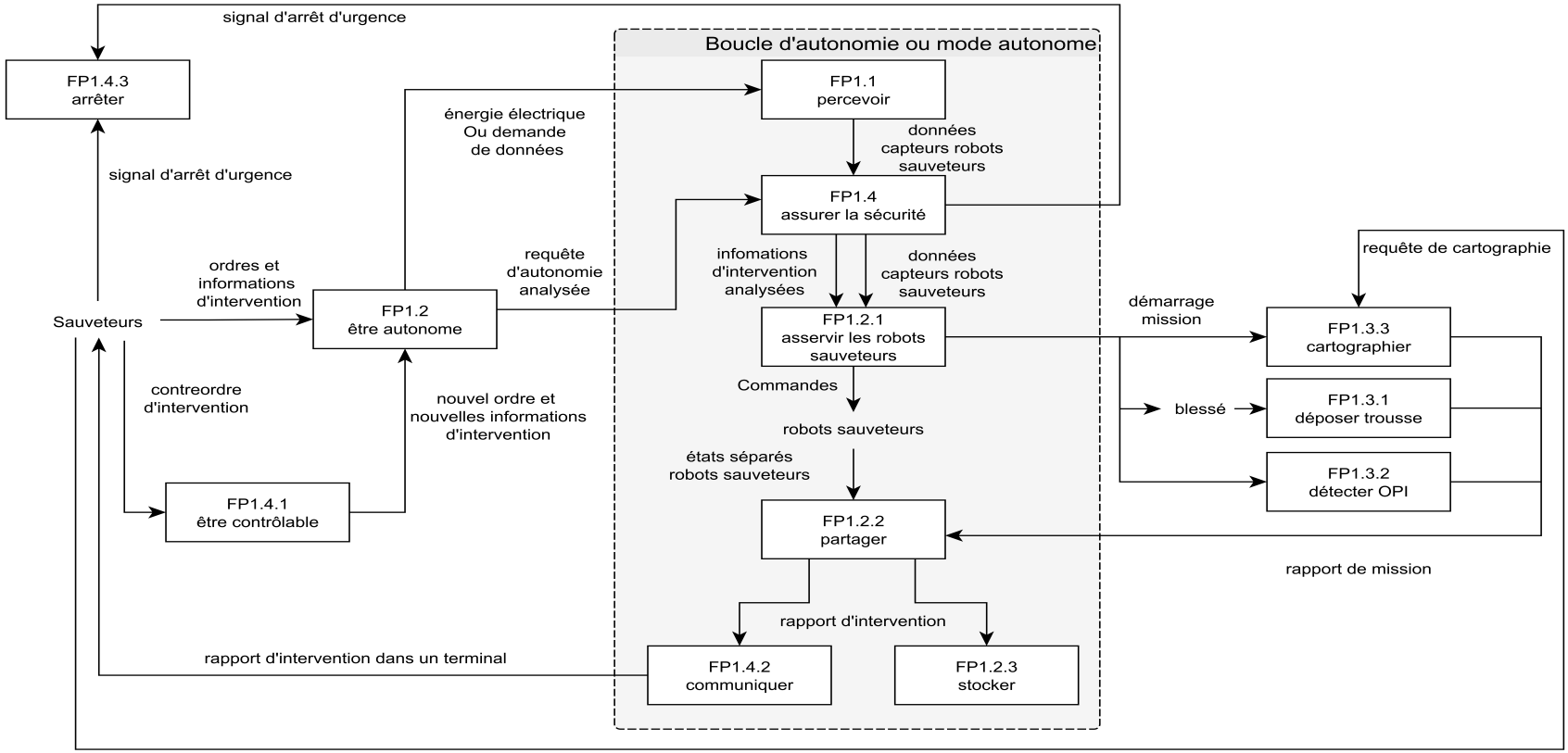


Figure *3.4 :* Architecture fonctionnelle du *SIC*.

3.3 Analyse organique

3.3.1 Sous-systèmes

Afin de déterminer l'architecture physique du *SIC*, le système a d'abord était décomposé en sous-systèmes. Puis, les interfaces physiques existant entre ces sous-systèmes ont été identifiées. Finalement, les fonctions principales on été associées aux différents sous-systèmes ce qui a permis de déboucher sur l'architecture organique présentée figure 3.5 page 20.

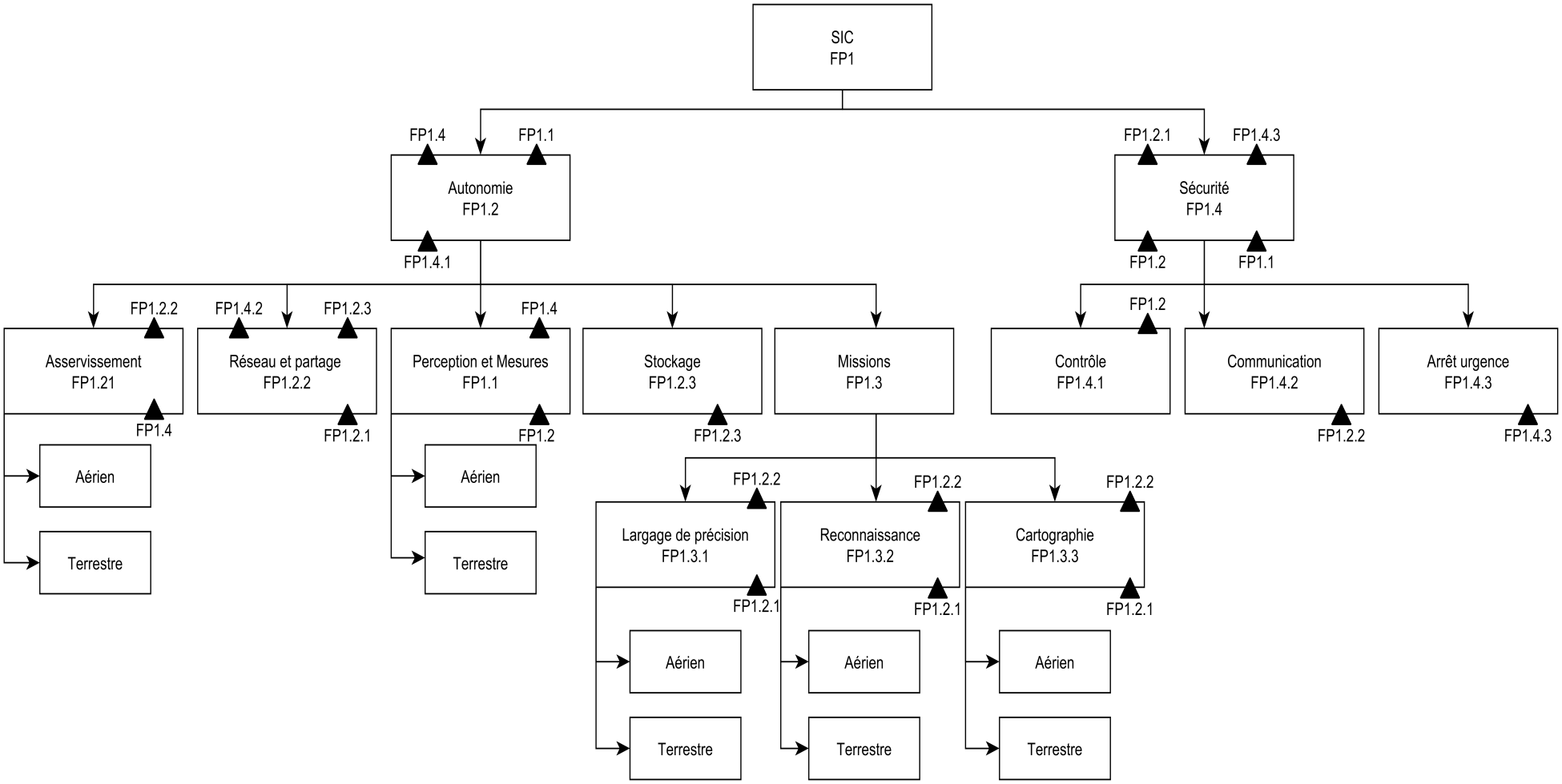


Figure ***3.5 :*** Architecture organique du *SIC*. Les triangles noirs symbolisent les interfaces entre deux fonctions principales.

Chapitre 4

Partie terrestre du système

4.1 État de l'art des robots existants

Le système *SIC*[1] développé à l'occasion de ce projet doit contrôler des robots terrestres et aériens lors de leur intervention de terrain. Avant de commencer tout travail sur la programmation des robots et leur asservissement, comme présenté au 4.3 page 25 et au 5.4 page 44, il faut d'abord choisir les robots terrestres et aériens à contrôler. L'étude du choix pour le robot terrestre est présenté au 4.1, dans cette section. Les conclusions relatives au robot aérien choisi, quant à elles, sont exposées au 5.1 page 41.

Afin de guider notre décision, les architectures électroniques et mécaniques des robots, si elles présentaient des singularités, ont été analysées.

4.1.1 Le robot six-roues

4.1.1.1 Architecture mécanique

Figure *4.1 :* Robot six-roues utilisé pour l'édition 2015 du concours *ERL*.

Le robot, figure 4.1, est constitué de six roues, trois à droites et trois à gauche. La chaîne de puissance mécanique est constituée d'un moteur par côté qui transmet la rotation aux trois roues via des chenilles et des barbotins. Une vue des chaînes de transmissions de puissance du robot est à disposition figure 4.2 page suivante.

Les moteurs choisis sont des *motoréducteurs*, ils sont assez long par rapport à la largeur du

1. cf. 3.1.1 page 13 pour une définition.

robot. De ce fait le moteur qui entraîne le côté droit et celui qui entraîne le côté gauche ne peuvent pas se placer en regard. Le parti pris a été de placer le moteur droit sur la roue avant et le moteur gauche sur la roue du milieu. Cela ne change en rien la cinématique du robot, tout laisse à penser que la contrainte d'espace a guidé ce choix [2].

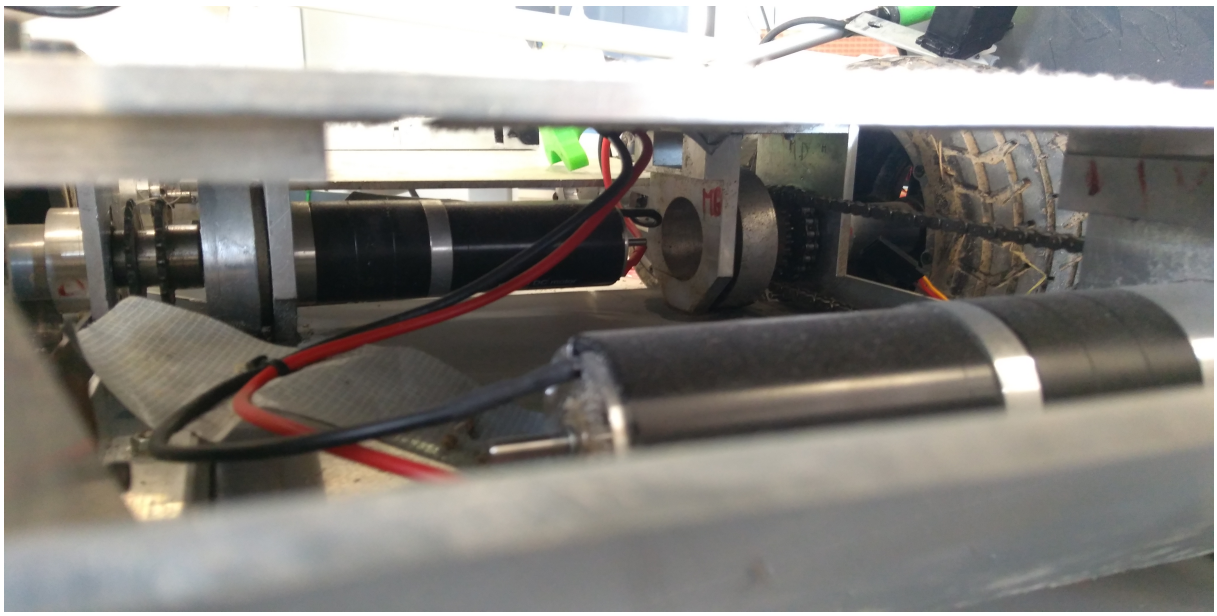


Figure 4.2 : Chaînes de transmission de puissance du robot six-roues.

Le robot étant fait pour s'aventurer sur plusieurs types de terrain, notamment sur des zones non plates, il est donc articulé par une liaison pivot entre son centre et son arrière. Le schéma cinématique figure 4.3 page 22 le met en évidence.

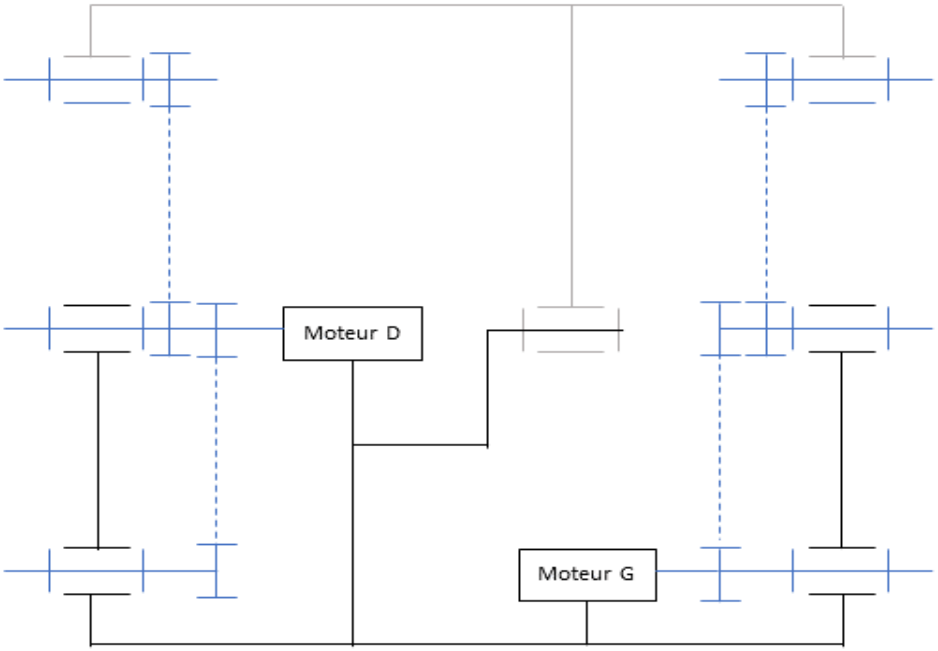


Figure 4.3 : Schéma cinématique du robot six-roues.
Légende: moteur D correspond au moteur droit, et moteur G au moteur gauche.

2. Ce robot n'a pas été développé durant ce projet.

4.1.1.2 Architecture électronique

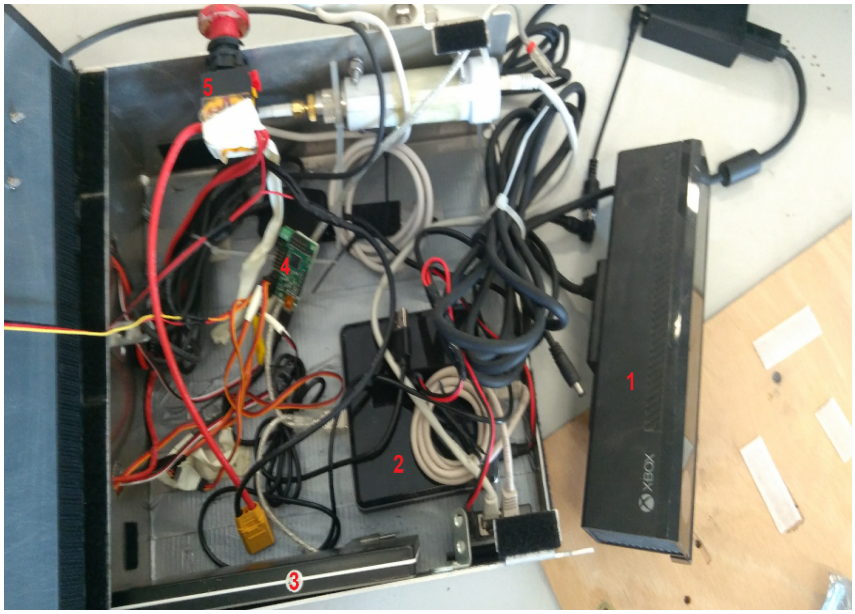


Figure 4.4 : Électronique du robot six-roues.
Caméra (1), Ordinateur embarqué (2), Batterie (3), Carte de contrôle des moteurs (4), Bouton d'arrêt d'urgence (5).

Les numéros figurant sur les sections suivantes font références aux numéros de légende de la figure 4.4. Cela permet de mieux visualiser l'interconnexion des équipements électronique.

Caméra (1) [3] La caméra utilisée est une Kinect. Initialement prévue comme accessoire de la console de jeu XBOX, la Kinect possède de nombreux avantages qui font d'elle un excellent capteur en robotique. Premièrement, elle possède une caméra couleur haute définition (1920×1280 pixels avec une vitesse de transmission de 30 images par seconde). On peut alors utiliser cette vidéo pour faire du traitement d'image en temps réel. Mais la Kinect possède également une caméra en nuance de gris qui détecte les objets devant elle et mesure la distance la séparant de l'objet. En combinant ces deux flux d'informations, on peut reconstituer un nuage de point coloré en 3D .

La présence de ce capteur *Kinect* est notamment très pratique pour réaliser la fonction principale *FP1.3.3.1* relative aux missions de cartographie [4].

Ordinateur embarqué (2) C'est le cœur de l'électronique de ce robot. Ce mini-ordinateur, l'*Intel NUC*, va centraliser toutes les données pour ensuite exécuter les différents codes programmés. Il possède tous les composants d'un ordinateur normal (carte graphique, processeur, sortie USB ...) et sa petite taille, $10 \times 10cm$, fait de lui un excellent ordinateur embarqué.

Batterie (3) Elle sert à alimenter les divers équipements du robot. Attention tout de même à son utilisation, car les une batterie Li-Po sont très fragiles.

Carte de contrôle des moteurs (4) La carte *Polulu Maestro* constitue l'intermédiaire entre les commandes envoyées par l'ordinateur et les moteurs. Pour les servomoteurs elle convertit l'angle souhaité du moteur en impulsions électroniques. Pour les moteurs de puissance, elle convertit la vitesse souhaitée en impulsions PWM [5].

3. cf. figure 4.4.
4. cf. cahier des charges fonctionnelles annexe B page 55.
5. Pulse-width modulation.

Bouton d'arrêt d'urgence (5) En cas de problèmes, il faut toujours garder une sécurité manuelle[6]. C'est un composant incontournable de tout projet robotique pour éviter des mésaventures désagréables.

Centrale inertielle (non-représentée sur la figure) Sur le capot du robot se trouve une centrale inertielle. Ce composant permet de connaître, après un traitement informatique des données recueillies, la position et l'orientation du robot. C'est un composant très utilisé pour la localisation, que ce soit pour un robot terrestre, aérien ou sous-marin.

4.1.2 Le robot *char*

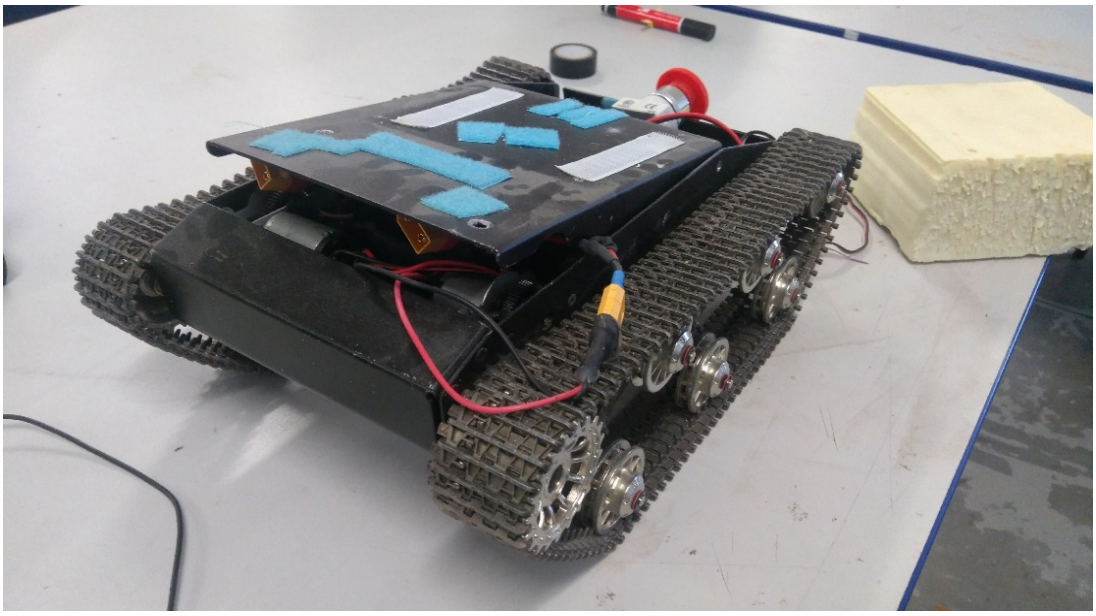


FIGURE *4.5 :* Robot char de l'*ENSTA Bretagne*.

4.1.2.1 Architecture mécanique

Le déplacement du robot-char, figure 4.5, est différent de celui d'une voiture classique avec des roues directrices. On a ici un moteur par chaîne, respectivement *M1* et *M2*. Pour avancer en ligne droite *M1* tourne dans le même sens que *M2*, pour reculer les deux sont en sens inverse, et pour tourner *M1* tourne dans un sens et *M2* dans l'autre. Ce type de véhicule est pratique, car créer une commande pour le déplacer est relativement simple.

4.1.2.2 Architecture électronique

Les composants électroniques et leurs agencements sont les mêmes que pour le robot six-roues présenté en 4.1.1 page 21. Veuillez donc vous référer au 4.1.1.2 page précédente pour davantage de détails sur l'architecture électronique d'un robot terrestre.

4.1.3 Le robot *buggy*

6. Ce qui permet de réaliser en partie la fonction principale *FP1.4.3* du cahier des charges fonctionnelles annexe B page 55.

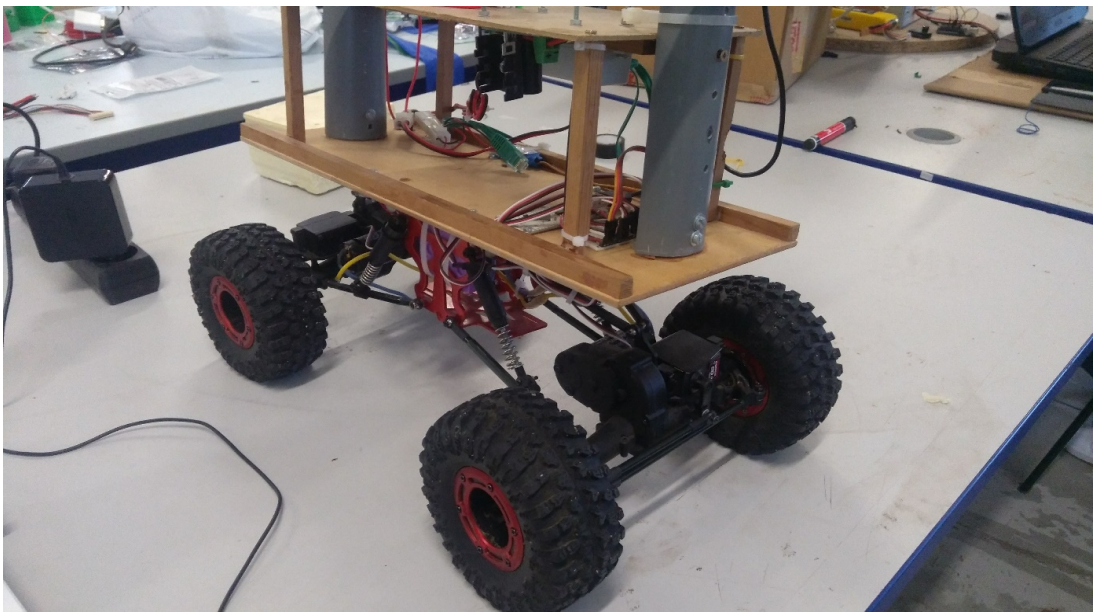


Figure *4.6 :* Robot de type « buggy » de l'*ENSTA Bretagne*.

4.1.3.1 Architecture mécanique

Le buggy figure 4.6 est composé de châssis complètement suspendus. Chaque châssis est composé d'un moteur pour la propulsion et d'un servomoteur pour la direction. La suspension pendulaire permet une grande mobilité du châssis pour passer les différents obstacles qui peuvent exister sur le trajet. Les quatre roues sont motrices et directrices, ce buggy se pilote donc comme une voiture, avec un angle de direction.

4.1.3.2 Architecture électronique

Les composants électroniques et leurs agencements sont les mêmes que pour le robot six-roues présenté en 4.1.1 page 21. Veuillez donc vous référer au 4.1.1.2 page 23 pour davantage de détails sur l'architecture électronique d'un robot terrestre.

4.2 Robot choisi

Des robots présentés au 4.1 page 21 le robot six-roues a été retenu pour sa robustesse. En effet, il est plus imposant que le buggy 4.6, et ne possède pas de chenilles comme le robot char 4.5 page ci-contre. Par ailleurs, ses grandes dimensions lui permettent de transporter davantage de capteurs, ce qui sera notamment souhaitable pour la réalisation des missions de détection d'*OPI*, et de cartographie, respectivement *FP1.3.2.1* et *FP1.3.3.1* du cahier des charges fonctionnelles en annexe B page 55. La possible installation d'un sous-système pour le dépôt d'une trousse de soin auprès d'un blessé, *FP1.3.1.1* du cahier des charges fonctionnelles en annexe B page 55, sera notamment facilitée sur le robot analysé au 4.1.1 page 21, sur le robot six-roues.

4.3 Algorithme d'évitement d'obstacles

Afin d'asservir le robot six-roues, et donc de réaliser la *FP1.2.1.2* du cahier des charges fonctionnelles en annexe B page 55, il a fallu développer et tester des algorithmes et des techniques d'asservissement dans un premier temps. Le but reste toujours le même. Á savoir, le robot doit pouvoir suivre en toute autonomie un itinéraire constitué de *waypoints*, et éviter les obstacles sur son chemin. Pour avoir un détail de l'architecture physique de la chaîne fonctionnelle liée à l'asservissement du robot terrestre, veuillez vous référer à la figure 3.3 page 17.

4.3.1 Suivi d'itinéraire

Le suivi d'itinéraire est un point essentiel de ce projet. Le robot doit être capable de rallier une position demandée de manière autonome. Pour cela, il doit être capable de suivre une consigne donnée par les programmes des cartes de contrôle [7]. A tout instant le robot va chercher à aligner sa direction avec un vecteur directeur qui fera office de consigne. Pour l'instant l'asservissement ne se fait que par un correcteur proportionnel.

4.3.2 Évitement d'obstacles

Lorsque notre robot se déplace d'un point à un autre, il se peut [8] qu'il y ait un ou des obstacles sur le chemin. C'est pourquoi, il faut implémenter dans le robot un algorithme d'évitement d'obstacles.

4.3.2.1 Les champs de potentiel

La première technique utilise les champs de potentiel [Jau15]. L'objectif est simple: à chaque point de l'espace on associe un vecteur qui pointe vers le waypoint, point en vert sur la figure 4.7 . Par exemple au point bleu [9] on associe le vecteur unitaire noir qui pointe vers le point vert. Ce vecteur sera un repère pour asservir le robot [10].

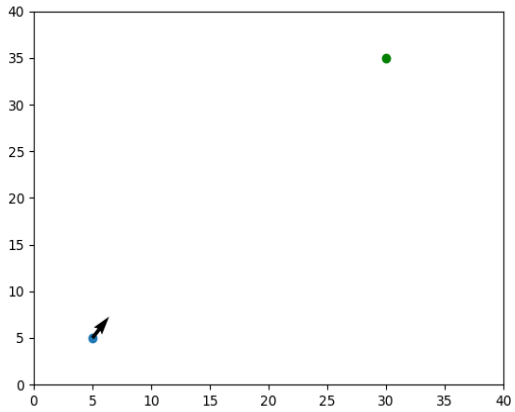


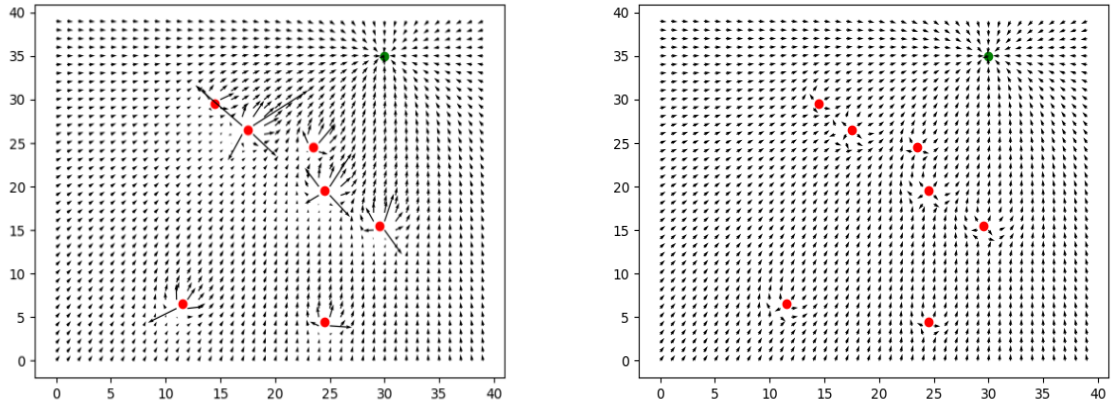
Figure 4.7 : Robot (en bleu) pointant dans la direction de son prochain *waypoint* (en vert).

Maintenant on s'intéresse au cas où on détecte un obstacle. Chaque obstacle sera représenté par un point dit « répulsif ». Ils sont appelés ainsi, car leur rôle est de faire varier l'inclinaison des vecteurs afin que le robot puisse éviter les obstacles. On veut également que plus le robot soit proche d'un point répulsif, plus l'influence de ce dernier soit forte sur le vecteur directeur. Mathématiquement ce problème se traduit de la manière suivante :

1. On crée un *waypoint* à suivre.
2. A tout point de l'espace on associe un vecteur unitaire qui pointe vers le *waypoint*.
3. A tout point répulsif on associe un champ de potentiel de norme égale à l'inverse de la distance entre le robot et le point répulsif et de direction radiale.
4. On ajoute les deux champs.

Avec cette méthode on obtient bien des vecteurs qui tendent à éviter les obstacles autour des points répulsifs et des vecteurs loin des points répulsifs qui pointent vers le waypoint à atteindre. Le résultat brut et le résultat avec le champ normé sont présents respectivement figure 4.8a et figure 4.8b page ci-contre.

7. Typiquement, les ordinateurs embarqués présents sur les robots, cf. 4.1.1.2 page 23.
8. Et dans le cadre du concours c'est même certain.
9. cf. figure 4.7. Ce point modélise la position du robot.
10. voir 4.3.1.



(a) Champ de potentiel brut

(b) champ de potentiel normé

Figure 4.8 : Champ de potentiel généré à partir de points répulsifs, les obstacles (en rouge), et du waypoint (en vert).

4.3.2.2 Simulation d'un lidar

Maintenant qu'une technique d'évitement a été mise au point on va chercher à simuler notre lidar [11]. En effet, il est bien plus aisé de tester un algorithme dans plein de situations générées numériquement, que de prendre du temps pour créer un environnement physique réaliste et voir si le robot réel réagit comme prévu. On simule d'abord une pièce ou un environnement dans lequel évolue le robot. Après un traitement mathématique (Tiré d'un exercice de robotique de Luc Jaulin [Jau14]) , on acquiert la position des murs [12]. Le résultat obtenu avec 150 points d'acquisition est présent 4.9.

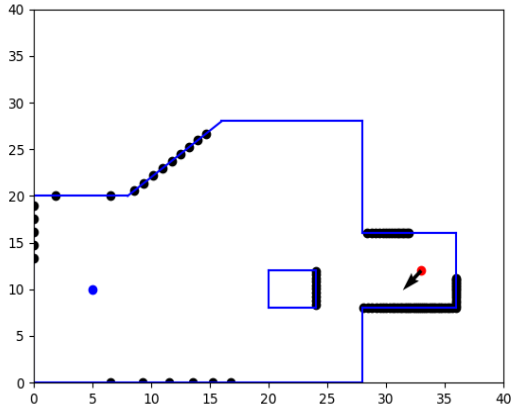


Figure 4.9 : Simulation du lidar (les points noirs sont les points qui simulent les données traitées d'un lidar).

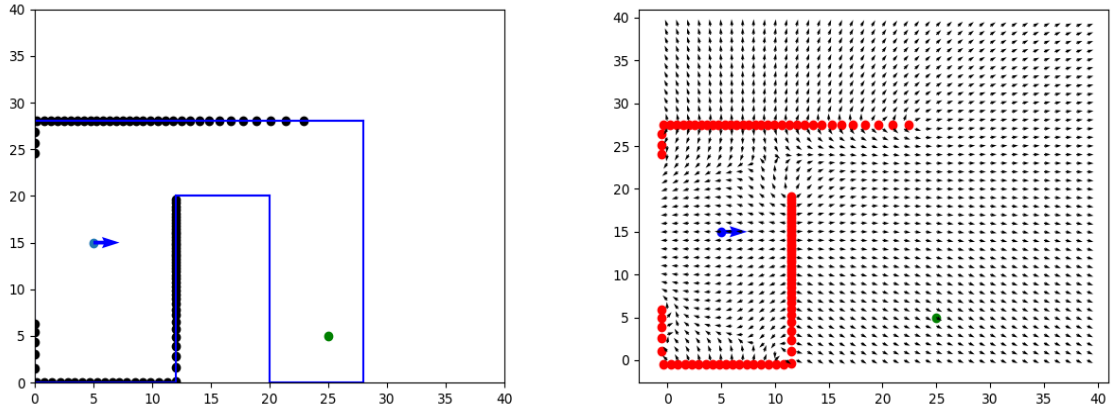
On remarque que plus les murs sont loin plus les espacements entre les points noirs, images du mur, sont importants. Cela est tout à fait normal car le lidar prend une valeur pour chaque pas angulaire. En faisant varier ce pas on augmente et on diminue le nombre de points obtenus. En simulation, on ne dépassera pas 200 points pour avoir un temps raisonnable de calcul.

11. Un lidar, ou *light detection ranging*, est est un capteur qui permet de détecter des obstacles en envoyant un faisceau laser directif.

12. cf. figure 3.3 page 17 pour un détail de l'architecture physique du calcul de l'asservissement effectué par les algorithmes de traitement des données capteurs, ici les données du lidar, et de calcul des commandes connaissant la position des obstacles.

4.3.2.3 Mise en évidence des limites des champs de potentiel

Reprenons notre simulation avec une salle différente, où le vecteur bleu représente la direction du robot, comme présenté figure 4.10. On applique donc notre algorithme [13]et on obtient la figure 4.10.



(a) Vue omnisciente

(b) Vue du robot

Figure ***4.10 :*** Simulation de l'algorithme d'évitement d'obstacles par champ de potentiel du 4.3.2.1 page 26 avec simulation de données *lidar*, comme développé au 4.3.2.2 page précédente.

Le problème est le suivant : le champ de vecteur résultant tend à faire rebrousser le robot en arrière. Or, dans certains cas, il y a aussi un mur derrière le robot. De ce fait, le mur est également situé dans l'angle mort du robot [14]. Le robot va donc osciller et faire des demi-tours intempestifs. Ce qui n'est pas le comportement désiré.

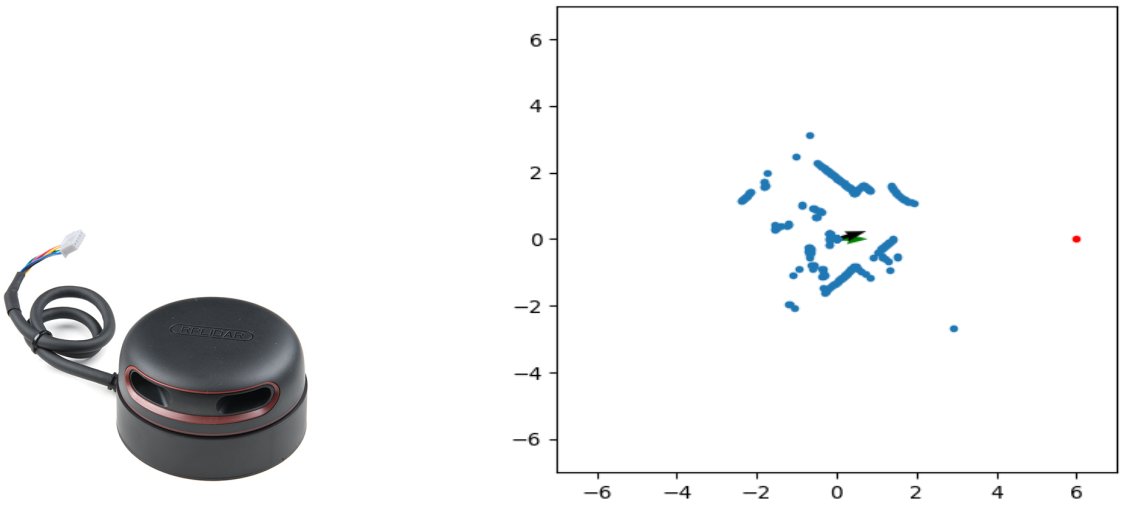
Afin de remédier à ces imprévus, pour la suite du projet, il est envisagé d'implémenter un autre type d'algorithme, un algorithme de type *SLAM*[15].

4.3.2.4 Seconde approche

4.3.2.4.1 Contextualisation Après avoir mis en place la technique des champs de potentiels et du *SLAM*[16], le robot s'est bien comporté pendant la compétition 2019 [17]. Cependant, avec l'évitement d'obstacles implémenté, le robot ne cherche pas à emprunter la trajectoire optimale [18]. Ayant eu un cours sur le sujet , il était alors plus aisé d'aborder ce type d'évitement.

Le point de départ est le même que pour l'évitement via la méthode des potentiels : l'algorithme d'évitement possède en entrée une mesure retournée par un *lidar* présenté figure 4.11a page ci-contre. La figure 4.11b page suivante montre les mesures considérées ici. L'orientation du robot est représentée par la flèche verte, le point à joindre est en rouge, et la flèche **noire** représente l'orientation que doit prendre le robot après le traitement de l'algorithme décrit dans au 4.3.2.4.2 page ci-contre.

13. Développé en 4.3.2.2 page précédente et en 4.3.2.1 page 26.
14. Le robot ne peut donc pas détecter ce mur. Tout se passe comme si il l'ignorait.
15. *Simultaneous Localization And Mapping.*
16. cf. 4.3.2.1 page 26.
17. En Espagne, du 18 au 23 février 2019.
18. Le plus court chemin.



(a) *RP Lidar A2* utilisé pour la compétition 2019.

(b) Données du lidar

Figure *4.11 :* Données recueillies par le *lidar*.

4.3.2.4.2 Filtrage, discrétisation, et recherche de plus court chemin Tout d'abord un filtre est appliqué sur les mesures du *lidar*. Les points trop près [19] du robot sont enlevés pour éviter que le robot ne voit certaines de ses antennes ou les faisceaux qui n'ont rien retourné [20]. Ainsi, l'amas de points au centre [21] est supprimé. Ensuite tous les points qui sont situés derrière le robot sont enlevés, car ils ne sont pas utiles.

Maintenant que les données sont correctement filtrées, l'espace est discrétisé à l'aide d'une grille, c'est-à-dire un maillage plus ou moins fin qui est appliqué sur la figure 4.11b filtrée. Ainsi, chaque maille qui contient un point *lidar* prend la valeur 1, les autres prennent la valeur 0. La figure 4.12 page suivante est le résultat des étapes précédentes. Cette figure contient exactement les points situés devant le robot et qui ont été convertis en blanc. Reste à appliquer un dernier traitement sur la matrice représentée par la figure 4.12 page suivante. En effet, le robot possède une certaine longueur et largeur et les obstacles sont représentés par des points de la taille d'une maille. C'est pour cela qu'il est alors nécessaire d'élargir les obstacles [22], afin de condamner une zone autour de chaque point. La dimension de cette dilatation [23] est donnée par la relation 4.1.

$$d = \frac{p \times l}{2} \tag{4.1}$$

où d est la taille de la dilatation
p le pas de grille ou du maillage
l la dimension maximale du robot.

Ainsi, si deux obstacles sont distants de moins de la largeur du robot, alors leurs deux disques de dilatation se joignent , et la possibilité pour le robot de passer entre ces deux points est nulle [24].

Ayant des données adéquates, il faut alors trouver le plus court chemin ralliant le point de départ et le point d'arrivée du robot. Il existe plusieurs algorithmes de ce type, ici l'algorithme

19. distance inférieure à $0.2m$.
20. Pour des raisons variées: problème de circuit, *hardware* ... Ces mesures prennent alors une distance de zéro par défaut.
21. en $(x, y) = (0, 0)$, cf. figure 4.11b.
22. Points en blanc.
23. Tel est le nom de cette transformation en traitement d'image.
24. cf. figure 4.12 page suivante.

Astar[25] a été retenu, et qui est d'ailleurs très utilisé dans le milieu de la robotique pour la recherche de plus court chemin [Les05]. Après un cours temps d'exécution, l'algorithme a trouvé dans la grille un chemin. Ce chemin est présenté figure 4.12.

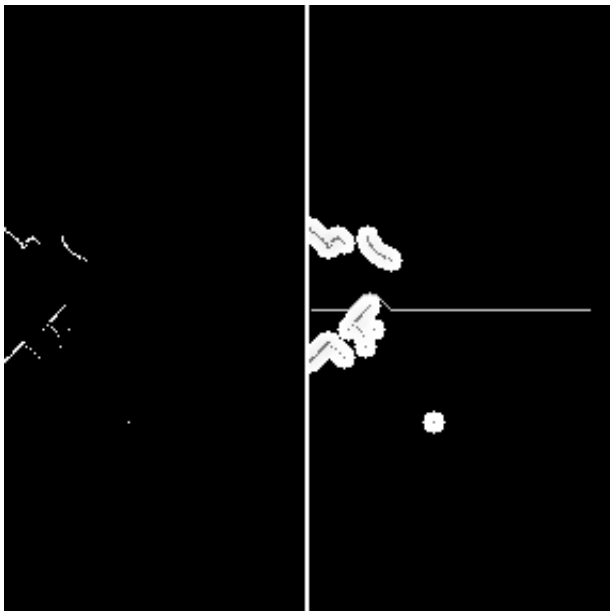


Figure *4.12 :* Résultat de l'algorithme *A-STAR* sur un maillage *lidar*. Le chemin trouvé est tracé en blanc (et est ici horizontal). La partie gauche de l'image représente les mesures lidar devant le robot et maillées. La partie droite de l'image est issue de la dilatation de la partie gauche, et contient le plus court chemin trouvé par l'algorithme.

4.3.2.4.3 Analyse des résultats L'algorithme renvoie bien un chemin qui est possible, mais est-il le plus court des chemin? Il peut paraître logique d'aller en ligne droite vers la cible et d'éviter un obstacle dès qu'il s'en présente un, cf. figure 4.12. Cependant, le chemin formé par un segment tangent entre le point de départ et le haut du mur [26] est plus court. Cela provient de la grille de discrétisation. Chaque case de la grille possède huit voisins, et il n'existe pas d'autre possibilité. Cette perte de continuité induit donc une perte d'information. L'algorithme renvoie bien un plus court chemin, mais qui est plus court au regard de sa métrique.

C'est pourquoi un post-traitement du chemin retourné par l'algorithme est requis, afin de récupérer le « chemin » orange de la figure 4.13. Ainsi, on obtient un nouveau cap qui s'appuie sur ce chemin, symbolisé par la flèche **noire** de la figure 4.11b page précédente.

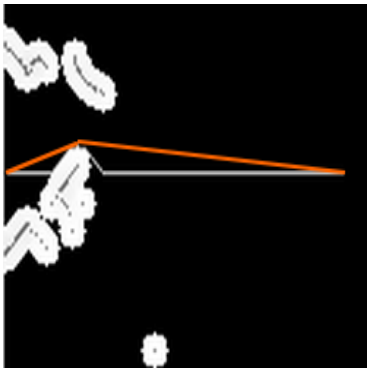


Figure *4.13 :* Plus court chemin en orange.

25. aussi appelé *A étoile* en français.
26. cf. figure 4.13 en orange, image 4.12 agrandie.

4.4 Imagerie *3D*

4.4.1 Contextualisation

Dans le but de réaliser la *FP1.3.3*[27] quant à la cartographie *3D et 2D*[28] il a fallu choisir un composant permettant au robot de visualiser l'environnement et d'en tirer les informations adéquates afin de synthétiser en aval une connaissance tridimensionnelle de l'environnement.

Afin de pouvoir le piloter manuellement à distance, le robot terrestre[29] possède une *Kinect v2*[30]. Néanmoins cette caméra est encombrante. La caméra *stereo*[31] *ZED* de *Stereolabs* a été étudiée et utilisée afin de voir si celle-ci ne constituerait pas un bon substitut de la *Kinect v2*. La caméra *ZED* est présentée figure 4.14

Cette caméra a été choisie pour ses caractéristiques: haute résolution, fréquence élevée de rafraichissement, perception en profondeur de l'environnement, portée de $50cm$ à $20m$, et possibilité de cartographie *3D*. De par sa taille, la caméra *ZED* est plus simple à intégrer sur le robot terrestre. Par ailleurs, elle possède directement une sortie *USB*. Cependant, le grand inconvénient de cette caméra est son *software*. Le logiciel requiert une carte graphique[32]. Il a donc fallu chercher un moyen de contourner cet imprévu afin d'acquérir les deux images fournies par les deux objectifs de la caméra *ZED* et de les traiter avec un programme informatique.

Figure *4.14 :* Caméra *stereo ZED* de *Stereolabs*.

4.4.2 Théorie

Pour reconstituer un nuage de points *3D* avec deux images *2D*, il faut comparer les deux images. Soient deux images, *image 1* et *image 2*. Pour tout point de l'*image 1*, il faut trouver à droite ou à gauche le même pixel dans l'*image 2*. Connaissant ce décalage et l'écartement des deux objectifs on peut en déduire la distance entre l'objet et la *ZED caméra*.

4.4.3 Pratique

En pratique l'implémentation de ce concept assez simple est bien plus complexe. En effet il est rare qu'un pixel d'intérêt soit parfaitement positionné sur la même ligne dans l'*image 1* et l'*image 2*. Il faut donc définir non plus une ligne mais bien un pavé de recherche. De plus, l'intensité lumineuse peut très légèrement varier entre les deux images. Á l'œil nu, ce n'est

27. cf. B.2 page 56.
28. Les travaux présentés concernant la cartographie ont été testés sur le robot terrestre six-roues 4.1.1 page 21 (cf. 4.5 page suivante pour la cartographie *3D* et 4.4 pour l'imagerie *3D*). Les outils développés sur le robot terrestre peuvent néanmoins être adaptés aux robots aériens. Ceci n'a pas pu être testé pendant le projet.
29. cf. 4.1.1 page 21
30. cf. 4.1.1.2 page 23.
31. cf. note de bas de page 45 page 33.
32. Le robot terrestre n'a pour l'instant que la carte graphique de son processeur. L'ordinateur de bord *NUKE* cf. 4.1.1.2 page 23 ne possède pas de carte graphique annexe.

parfois pas perceptible, mais un écart d'une unité sur une plage de 255 valeurs fait échouer un test d'égalité strict. Il faut donc là aussi prévoir un seuillage [33].

4.4.4 Implémentation

La bibliothèque *OpenCV* de Python a été utilisée pour réaliser les traitements sur les images de la caméra *ZED*, puisqu'elle possède un module pour faire ces types de traitement. Malgré divers tests avec plusieurs paramètres différents, les résultats ne sont pas concluants comparés à ceux de la *Kinect v2*. Une explication possible provient de la fabrication de la caméra *ZED*. Les deux images renvoyées par la ZED caméra ne sont pas à la même hauteur : il y a un décalage d'une vingtaine de pixels selon l'axe vertical de l'image. Une recalibration permettrait sûrement d'obtenir de meilleurs résultats, mais cette piste n'a pas pu être explorée.

La *Kinect v2* a donc été conservée pour les travaux de cartographie, et l'utilsation du *Middleware ROS* a permis d'obtenir des résultats convaincants. Pour davantage de précisions sur la cartographie veuillez vous reporter à la section 4.5.

4.5 Cartographie

4.5.1 Démarche

Avant le concours *ERL* [34] seule une cartographie *2D* par *lidar 2D* avait été réalisée, ce qui répondait partiellement à la fonction principale *FP1.3.3* [35]. L'objectif suivant était alors d'implémenter et d'intégrer un système de cartographie *3D* à la fois pour un milieu intérieur et extérieur.

Afin de réaliser la cartographie *3D*, les solutions technologiques à disposition et usuellement utilisées pour la cartographie ont été étudiées sous forme d'une étude de l'existant dont les conclusions sont présentes en 4.5.2. La solution technologique alors choisie n'a pas été directement testée sur le robot terrestre [36], mais sur un ordinateur possédant une carte graphique suffisamment puissante [37], servant de test unitaire. Des tests d'intégration on alors été réalisés afin de comprendre les qualités et défauts du système mis en place, pour alors l'adapter.

4.5.2 Étude de l'existant

Réaliser une cartographie intérieure ou extérieure consiste à prendre des photographies de l'environnement à intervalle de temps régulier, et de les positionner sur une carte [38]. Pour ce faire il est indispensable de disposer de la position du robot à chaque instant et de données en profondeur [39]. Les images peuvent ainsi être positionnées par rapport au robot grâce au données *en profondeur*, et le robot ensuite positionné par rapport à la carte construite précédemment par le biais de la position du robot. Une carte *3D* de l'environnement peut ainsi être réalisée.

Cependant, la position *GPS* n'est pas disponible à l'intérieur d'un bâtiment [40]. De même, l'estimation de la position renvoyée par la centrale inertielle de type *SBG Ellipse 2A* n'est que très imprécise et souffre d'incertitudes issues des intégrations successives provenant des accélérations mesurées.

Pour palier à cette absence de localisation directe du robot [41], une autre technique peut être

33. ou une plage d'acceptabilité.
34. Semaine du 18 au 23 février 2019.
35. cf. B.2 page 56.
36. Le robot terrestre ayant participé à la compétition 2019, à savoir le robot six-roues comme présenté en 4.1.1 page 21.
37. Hardware: *Nvidia GEFORCE GTX 1080 Ti* sur un ordinateur *Asus G703Vi*.
38. Ou plutôt par rapport à un repère galiléen.
39. Ou en *depth*.
40. Sinon, très imprécise et donc inutilisable.
41. C'est-à-dire utilisant des capteurs délivrant la position recherchée, comme un *GPS*, filtre de *Kalman*, ou encore une centrale inertielle.

utilisée: le *SLAM*[42]. Le concept est d'utiliser la connaissance progressive de l'environnement du robot pour le localiser. Dans le cas de la cartographie *3D*, une approche très efficace est l'*odométrie visuelle*. Différentes implémentations de l'*odométrie visuelle* ont été développées [CVG ; ML13]. Ces approches fusionnent ou non des données de plusieurs capteurs, comme le *lidar*[43], la centrale inertielle ou *imu*[44], des caméras *stereo*[45], des odomètres, ou encore des caméras *RGB-D*[46]. Il s'agit alors d'utiliser les données capteurs pour cartographier et localiser le robot. Quelques configurations usuelles sont par exemple:

- Un *lidar 3D*: la position du robot est alors obtenue en comparant les nuages de points obtenues successivement.
- Une caméra *RGB-D* ou une caméra *stereo*: il s'agit alors de comparer les images couleurs et images en profondeur obtenues successivement.
- Une caméra *RGB-D* ou une caméra *stereo* couplée avec un *lidar* ou bien une centrale inertielle, pour améliorer la localisation.

La *Kinect V2*[47] a été choisie pour la qualité des informations[48] délivrées. La *Kinect* étant une caméra *RGB-D*[49], la cartographie *3D* a donc été réalisée par *SLAM RGB-D*[ML13].

Néanmoins, obtenir un algorithme de *SLAM RGB-D* est complexe, et excessivement chronophage. Un outils adapté à la robotique et pour de tels développement, appelé *ROS* pour *Robot Operating System* permet d'installer sur n'importe quel robot possédant le *Middleware ROS* et ayant le *Hardware* adéquat certains *packages*[50]. *ROS* a donc été utilisé sur ordinateur[51] afin de réaliser une cartographie *3D* de l'environnement. La section 4.5.3 présente *ROS* plus précisément.

4.5.3 ROS

A l'instar d'un *système d'exploitation*, un robot peut être vu comme un ensemble de tâches informatiques ou de *processus*[52] qui interagissent les uns avec les autres. Pour permettre une gestion plus opitmale du *Hardware* du robot des *Middlewares* sont utilisés. *ROS* est un *Middleware*. Ainsi, *ROS* permet de créer des *topics* servant de canaux de communication où chaque processus, en language *ROS* appelé *noeud*, peut publier des données[53]. Il existe également des relations *service-client* entre le noeuds qui permettent de fluidifier les échanges d'informations au sein du robot. Par ailleurs, l'un des grands avantages de *ROS* est de pouvoir bénéficier des programmes, techniques,et astuces utilisés par la communauté *ROS*.

La mise sous *ROS* du robot présente donc un enjeu important, puisqu'elle sera gage d'une plus grande évolutivité et durabilité[54]. Développer un système c'est aussi penser à toutes ses phases de vie, et notamment à son maintien. Il est important aussi de préparer et faciliter les travaux futurs sur le robot terrestre[55].

Dans l'immédiat, *ROS* a permis de développer et de mettre en place une solution de cartographie *3D*. [Gar ; Lum15]

Néanmoins, une architecture *ROS* du robot six-roues[56] a été pensée, et dont seules les librairies indépendantes et *drivers* des différents capteurs ont pu être installées. Il reste notamment

42. Simultaneous Localization And Mapping.
43. cf. note de bas de page 11 page 27.
44. *Inertial Measurement Unit*.
45. Une caméra avec plusieurs objectifs qui permet d'avoir une vision tridimensionnelle de l'environnement.
46. *RGB* pour la partie caméra couleur, et *D* pour la partie vision en profondeur.
47. Développée par *Windows* pour la *Xbox One*.
48. images ...
49. Renvoyant des informations en images couleurs et en profondeur.
50. Ou librairies, ensembles de programmes déjà développés, qui permettent d'optimiser le fonctionnement du robot. Les librairies partagées sont très bien documentées pour la plupart et ont été rigoureusement testées.
51. Pas sur le robot puisque, le robot fonctionne actuellement sous *Windows* et non sous une architecture *Linux*. *ROS* a été principalement développé sous *Linux* et notamment sous *Ubuntu*.
52. Un processus par capteurs ...
53. Des données capteurs par exemple.
54. Un robot sous *ROS* est plus simple à maintenir, qu'un robot entièrement programmé de A à Z.
55. Robot six-roues.
56. cf. 4.1.1 page 21.

à faire fonctionner la carte *Pololu Maestro* sous *ROS*, mais le bas niveau de ce microcontrôleur est déjà implémenté et garder le bas niveau en l'état peut très bien coexister avec une architecture *ROS*. Il reste alors à faire communiquer les différents capteurs par le biais de *topics ROS*. L'architecture *ROS* est présentée figure 4.15 page suivante.

La partie supérieure « Capteurs » regroupe les *drivers* des capteurs gérés par des nœuds *ROS*[57], et les nœuds chargés de formater[58] et de filtrer les données des capteurs. La partie inférieure est composée d'un ensemble « Contrôleur ». Cet ensemble de nœuds est chargé à l'aide des données capteurs, et de consignes ou d'ordres de mission[59], d'envoyer les commandes à la carte *Pololu Maestro* chargée de contrôler les moteurs du robot. Plus précisément, l'ensemble « Contrôleur » est constitué des différents noeuds permettant de gérer les missions du robot: *la cartographie*, *la détection des OPI*, *le suivi d'itinéraire*, et *le dépôt de la trousse de soin.* De cette architecture *ROS*, seules les drivers des capteurs et les *topics* des données capteurs brutes ont été mis en place, ainsi que l'ensemble des noeuds de la cartographie[60].

De l'architecture *ROS* présentée figure 4.15 page suivante la section suivante expose l'ensemble de la *cartographie*.

57. des processus.
58. En disposant les informations dans une structure de données plus adaptée.
59. Générés par le noeud « automate ».
60. Pour plus de détails sur la cartographie, voir 4.5.4 page 36.

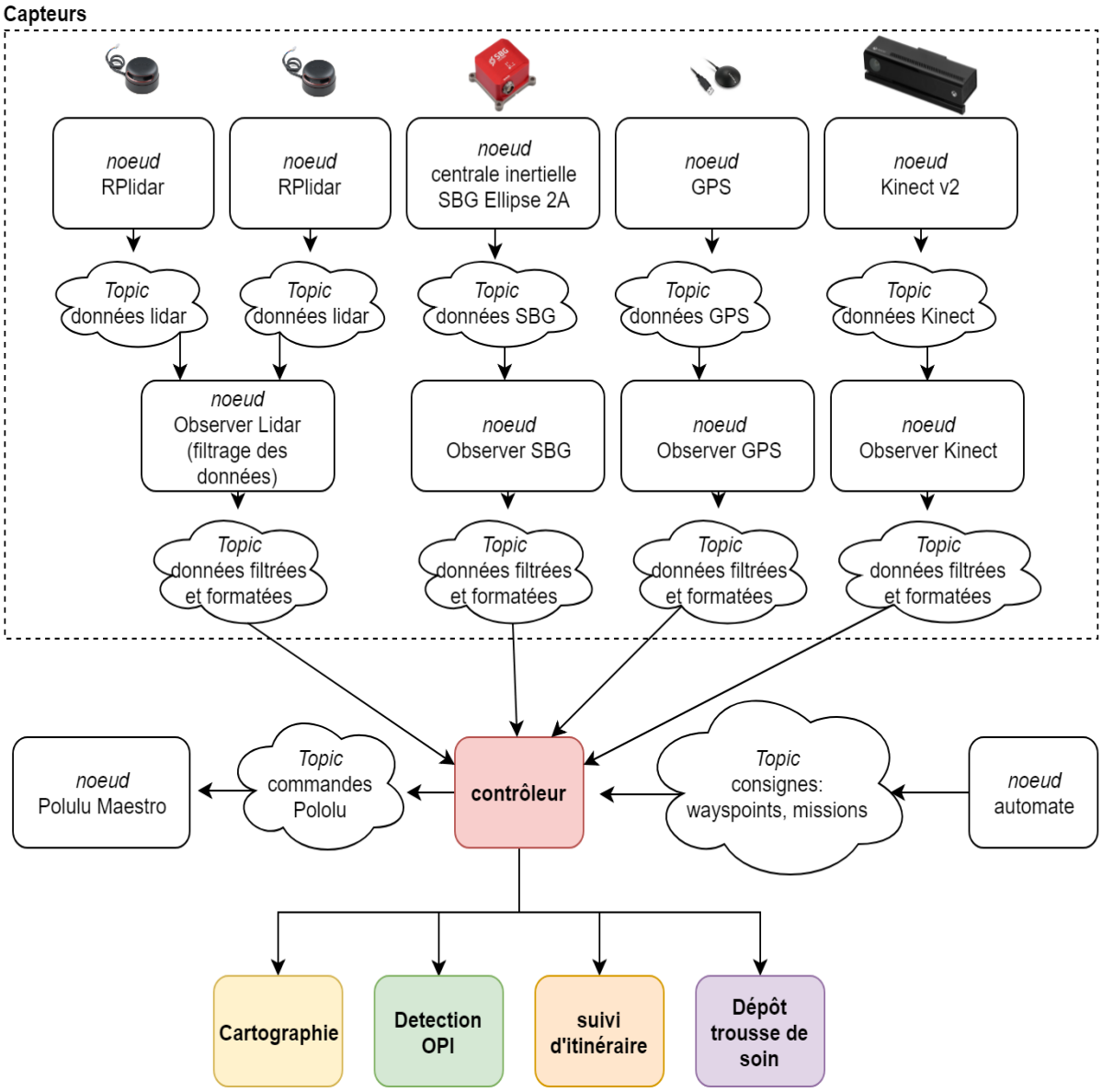


Figure 4.15 : Architecture *ROS* du robot six-roues.

4.5.4 ROS et la cartographie

Les tests exposés dans cette section ont été réalisés sur un ordinateur séparé [61] avec une carte graphique suffisamment puissante [62] pour les traitements d'images copieux dans les algorithmes de *SLAM RGB-D*. Liste du matériel:

- Ordinateur *Asus G703Vi*.
- Carte graphique: *Nvidia GeForce 1080 Ti*.
- Caméra *RGB-D*: *Kinect v2*.

Parmi les nombreuses solutions offertes par *ROS* en termes de *SLAM RGB-D*, la solution appelée *rtabmap* pour *Real-Time Appearance-Based Mapping* a été retenue pour les possibilités d'évolution et de réglage[3IT ; ML13]. Par ailleurs, il est également possible de sauvegarder les données de la cartographie dans le format *.ply* [63].

Après avoir installé les *drivers* [64] de la *Kinect v2*, et avoir adapté la librairie *rtabmap* au *Hardware* de l'ordinateur, les résultats obtenus sont présentés figure 4.16.

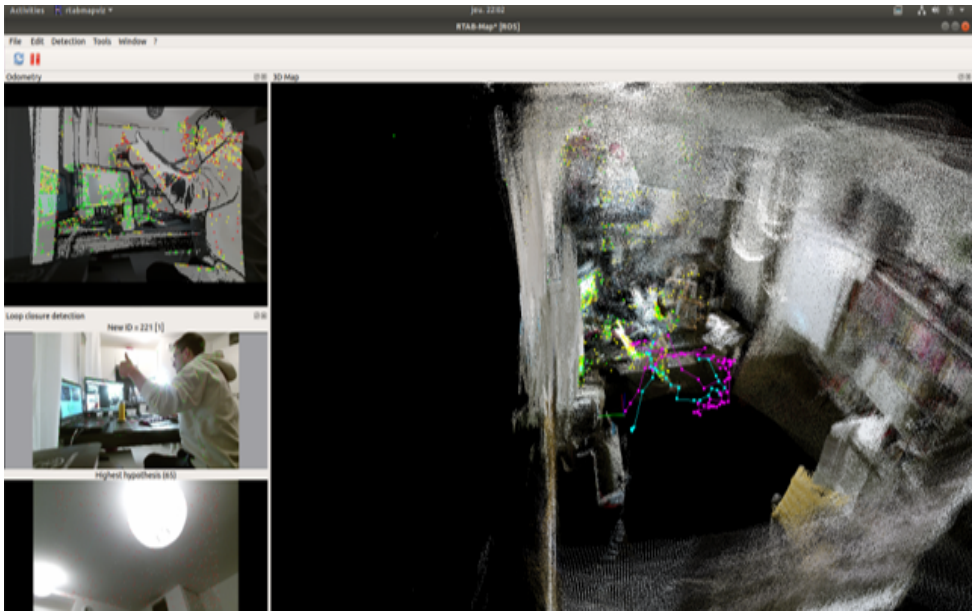


Figure *4.16 :* Capture d'écran du résultat de la cartographie intérieure avec *rtabmap*.

Il était possible de cartographier une pièce en intérieur et un milieu extérieur avec l'ordinateur *G703Vi* et la *Kinect v2*. Il était alors nécessaire d'intégrer le système de cartographie sur notre robot.

4.5.5 Intégration sur le robot

Tout d'abord le premier test a été de positionner la *Kinect v2* sur le robot afin d'analyser le comportement du programme de *SLAM RGB-D* en situation réelle. Le programme de *SLAM RGB-D* était donc lancé sur l'ordinateur séparé. Il semblerait que la vitesse de calcul, de traitement des images, et d'estimation de la position du robot par *SLAM RGB-D* est encore trop lente pour la vitesse de déplacement du robot, principalement lors des rotations. Après avoir modifié certains paramètres, l'odométrie [65] se montre plus fiable et plus performante. L'odométrie n'est cependant toujours pas suffisamment rapide lors des rotations du robot [66].

61. Pas sur l'ordinateur du robot directement.
62. Le *hardware* utilisé pour les tests est détaillé note de bas de page 37 page 32.
63. C'est un format qui permet de représenter des nuages de points en *3D*. Correspond à l'exigence *3.6* du cahier des charges présenté annexe A.1 page 51.
64. Programmes bas-niveau permettant de faire transiter les informations émises par un périphérique quelconque vers un ordinateur.
65. Ou localisation.
66. Virage brusque à *90°*.

Quoiqu'il en soit, ce test d'intégration a permis de comprendre à quel point la puissance de calcul de la carte graphique est essentielle. En effet, l'ordinateur *Intel NUKE*[67] présent sur le robot ne présente pas de carte graphique séparée autre que celle du processeur, contrairement à l'ordinateur de test. En vue d'une intégration sur le robot il sera donc sans doute indispensable d'ajouter un ordinateur embarqué de type *Jetson TX1*[68] pour les traitement d'images gourmants du *SLAM RGB-D*. Pour l'ordinateur de test *Asus G703Vi* il a notamment était possible d'installer sous *Ubuntu 18.04* un utilitaire *CUDA* qui permet d'accélérer les traitements de la carte graphique.

Néanmoins, la puissance de calcul ne fait pas tout. Même si une carte graphique plus puissante est requise pour l'intégration du système de cartographie sous *ROS*[69] sur le robot, la perte de la localisation du robot du fait de ses mouvements trop brusques ne sera pas solutionnée. Le problèmes provient principalement du champ de vision réduit de la *Kinect v2*. En effet, un algorithme de *SLAM RGB-D* estime la position du robot en comparant successivement les images prises. En comparant deux images, et en extrayant les points communs[70], et les distinctions[71] entre deux images, il est possible de retrouver le mouvement de la caméra qui a permis de passer d'une image à une autre. Cependant, si les images n'ont rien en commun ou que très peu d'*inliers*, ce qui ce produit quand le robot tourne très rapidement, il est impossible de retrouver le mouvement du robot, et donc de le localiser. Il est alors possible de coupler un *lidar 2D*[72] avec la *Kinect v2* afin d'améliorer la localisation. Le *lidar* ayant un champ de vision de presque *360°* va permettre de bénéficier d'une odométrie plus précise en cas de mouvement brusque et notamment pendant les rotations.

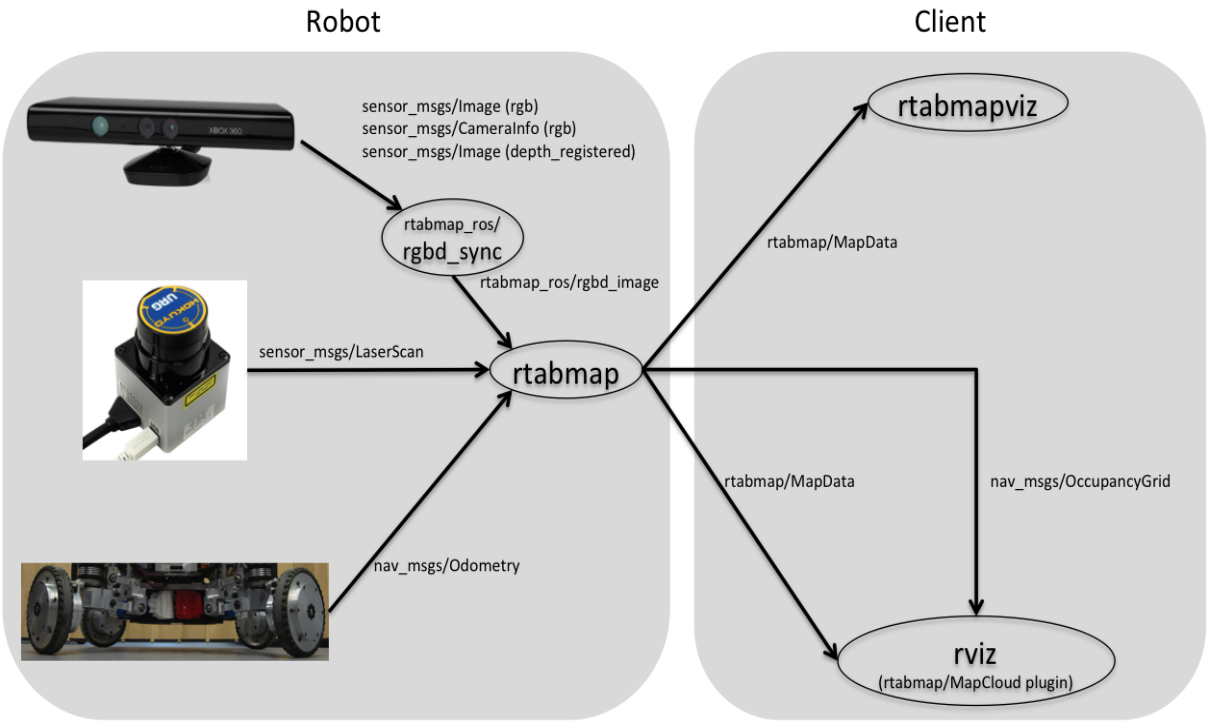


Figure 4.17 : Il s'agit d'adapter la configuration de cette figure à la configuration envisagée pour le *SLAM RGB-D* avec *rtabmap* sur le robot *six-roues*: *Kinect v2* (au-dessus), *lidar* (au milieu), et *Centrale inertielle SBG* (en bas). Extrait de [Wik].

Comme l'appuie la figure 4.17, il sera également possible d'ajouter les données de la *centrale inertielle SBG Ellipse A2* afin de permettre une estimation plus précise des angles d'*Euler*. Ce

67. cf. 4.1.1.2 page 23.
68. Sur le site du fabricant *nvidia*: https://developer.nvidia.com/embedded/buy/jetson-tx1.
69. Sous *Linux* forcément.
70. inliers.
71. outliers.
72. *RP Lidar A2*.

qui fournira une odométrie plus précise.[ML13]

4.6 Dépôt de la trousse de soin

Afin de répondre à la fonction principale *FP1.3.1.1*[73] étant le dépôt d'une trousse de soin auprès d'un blessé[74], la solution initiale était de construire un bras robotisé. Finalement, une benne en bois a été fabriquée, et munie d'un servomoteur afin de pouvoir la basculer. Cette solution simple s'est notamment montrée adaptée à la compétition[75]. Une photo du système réalisé est présenté figure 4.18. Un bras robotisé pourrait cependant permettre de récupérer la trousse de soin une fois déposée, dans le cas où par exemple la trousse de soin a été déposée trop loin du blessé.

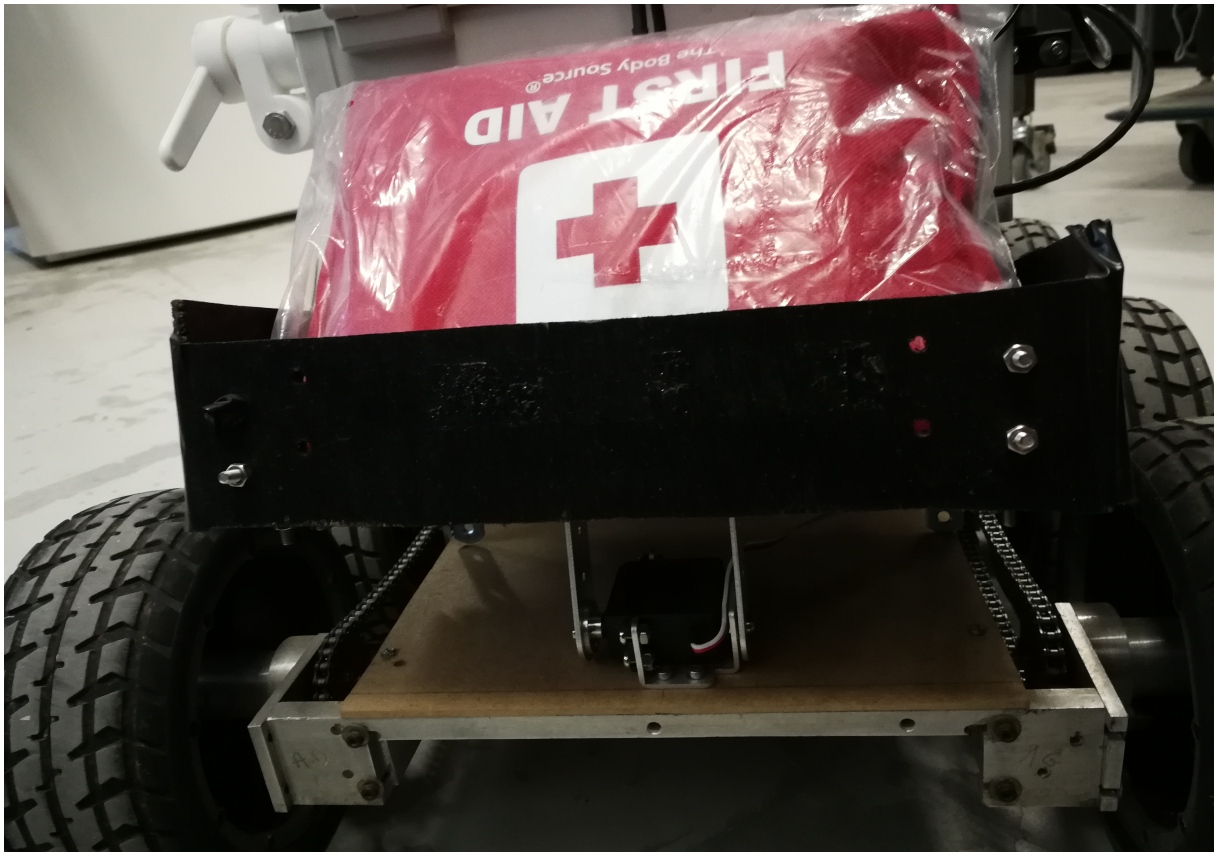


Figure *4.18 :* Système réalisé pour le dépôt de la trousse de soin.

4.7 Le robot six-roues

Á la suite du projet, le robot terrestre principal, le robot six-roues[76] est opérationnel et sa robustesse a été prouvée lors du concours *ERL* 2019. Cette base fiable pourra notamment être réutilisée pour les éditions futures de la compétition *ERL2019*. Cette section présente l'architecture matérielle[77] du robot six-roues, figure 4.19 page suivante, accompagnée de photos.

73. cf. cahier des charges fonctionnelles B page 55.
74. Mannequin orange.
75. cf. 6.3 page 48.
76. cf. 4.1.1 page 21.
77. Capteurs, actionneurs ...

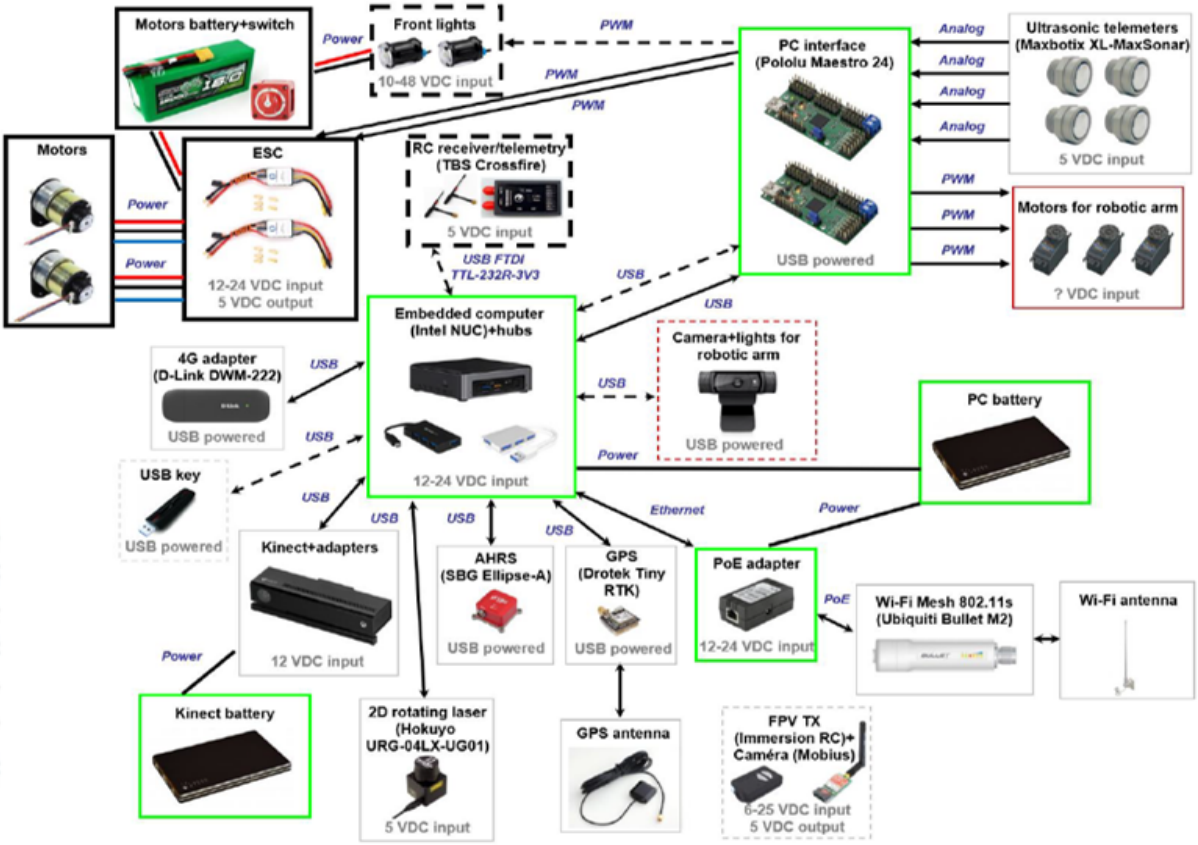


Figure 4.19 : Architecture matérielle. [Bar19]

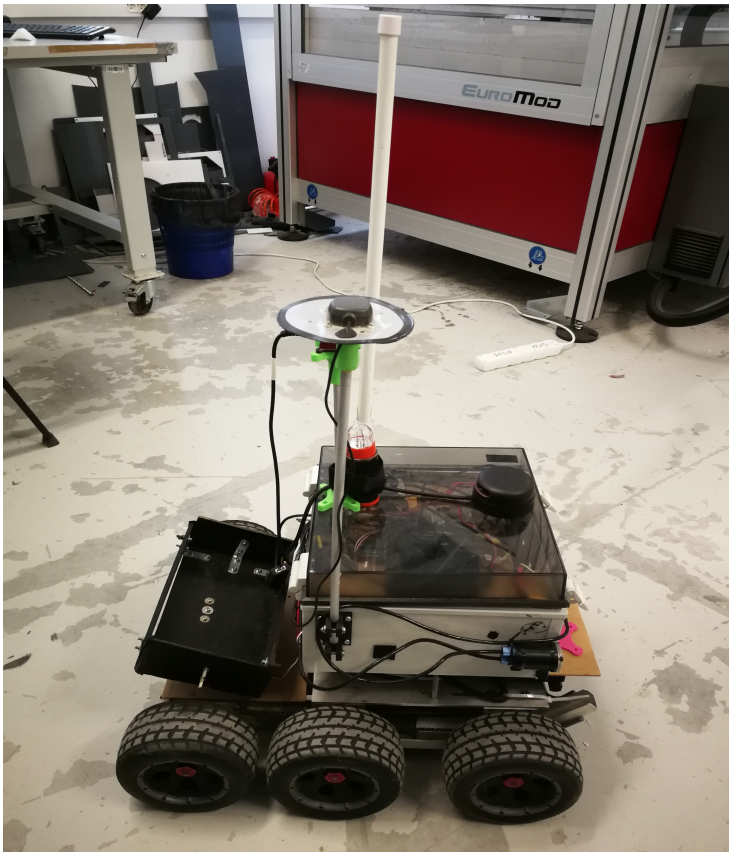


Figure 4.20 : Robot six-roues final.

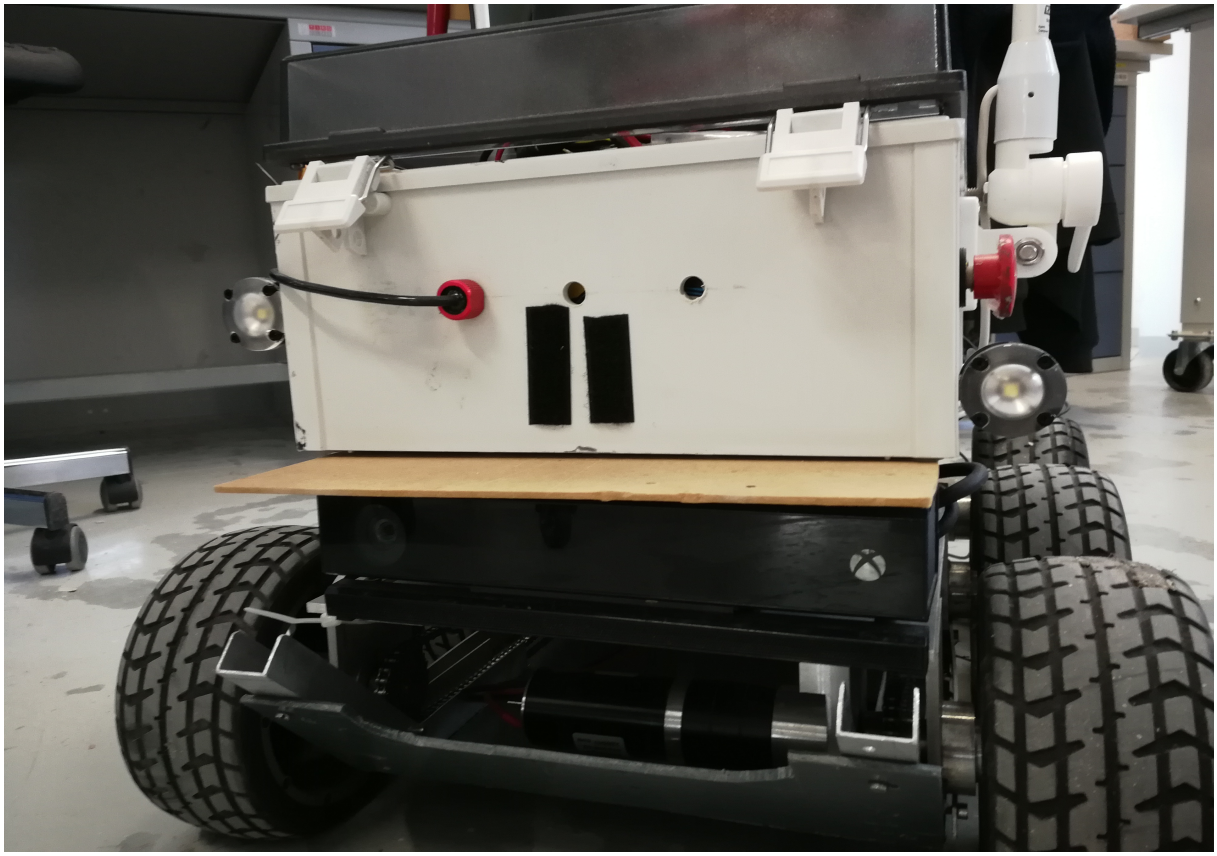


Figure *4.21 :* Avant du robot six-roues et sa *Kinect v2*.

Chapitre 5

Partie aérienne du système

5.1 État de l'art des robots existants

5.1.1 Robots existants

Plusieurs drones ont déjà été construits à l'Ensta-Bretagne, dont les trois présentés figure 5.1, que ce soit par des professeurs ou bien par des élèves. Comme nous devons utiliser un drone dans le cadre de la compétition 2019, il a donc fallu étudier les différentes solutions, peser le pour et le contre, et décider. Même si durant la compétition un seul drone aérien devra être utilisé, il faut tout de même prévoir l'éventualité d'une panne ou d'un problème technique sur le drone [1]. Plusieurs drones sont donc à retenir [2]. Néanmoins, un seul drone sera effectivement désigné comme *drone principal* [3].

(a) *(b)* *(c)*

Figure 5.1 : Drones de l'ENSTA Bretagne étudiés.

5.1.2 Architecture [4] d'un quadrirotor [5]

Afin de choisir le *drone principal*, des études détaillées des différents drones présents figure 5.1 ont été conduites. Le but était de déterminer les éléments constitutifs d'un drone afin de guider le choix du *drone principal*. Comme les architectures des drones figure 5.1 sont fortement similaires, seule l'architecture du drone image 5.1c est détaillée dans ce rapport.

Moteur Le moteur convertit l'énergie électrique délivrée par la batterie en énergie mécanique de rotation transmise aux pales. La figure 5.2a page suivante en donne un aperçu.

Carte électronique C'est le « cerveau » du drone. Elle acquiert les données renvoyées par les capteurs et module la tension envoyée aux moteurs pour les faire tourner plus ou moins rapidement. La figure 5.2b page suivante en donne un aperçu.

1. Les robots terrestres se montrent plus robustes. Nous n'avons donc pas choisi de robots terrestres de secours.
2. Les drones de secours seront les drones des images 5.1a et 5.1c.
3. cf. drone de l'image 5.1b.
4. Ensemble des éléments constitutifs d'un drone.
5. Drone ayant quatre rotors horizontaux comme présenté figure 5.4 page 44.

Batterie Elle fournit l'énergie pour alimenter tous les composants du drone, que ce soit aux moteurs ou aux autres composants électroniques.

Hélice Les hélices d'un drone sont formées de deux ou trois pales. Ces pales possèdent un pas, qui représente l'angle de rotation des pales, et qui a une influence sur la portance générée [6]. Chaque hélice est encastrée sur un arbre moteur, afin de pouvoir mouvoir le drone. La figure 5.2c en donne un aperçu.

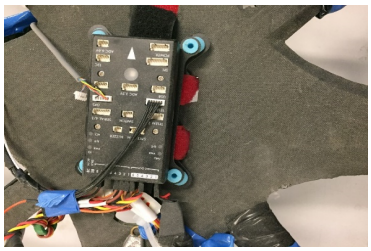
Capteur On peut trouver plusieurs types de capteurs sur le drone. Par exemple : le **capteur de GPS** qui permet connaître la position exacte du drone, le **capteur ultrasonore** qui permet de mesurer la hauteur entre le drone et le sol, le **Gyroscope** qui stabilise le drone horizontalement. Les figures 5.2d et 5.2e en donne un aperçu.

Caméra *flow optique* On peut trouver deux types de caméra sur un drone. Une caméra de *résolution* qui se trouve généralement à l'avant du drone, et une caméra située en-dessous du drone qui est utilisée pour la *stabilité* du drone.

Main switch [7] C'est un commutateur électronique conçu principalement pour la sécurisation de l'alimentation principale d'un modèle radio-commandé. Cette partie permet de remplir la fonction d'arrêt d'urgence.

ESC [8] L'*ESC* est un composant qui converti une consigne de vitesse de rotation provenant de la carte électronique, image 5.2b, en une tension d'alimentation variable du moteur électrique, image 5.2a.

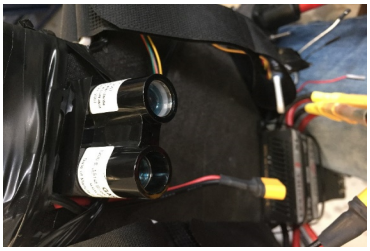
(a) Moteur



(b) Carte électronique

(c) Hélices

(d) *GPS*



(e) Transducteurs ultrasonores

Figure *5.2 :* Éléments principaux constitutifs d'un quadrirotor.

5.2 Robots choisis

Chacun des drones présents figure 5.1 page précédente a des avantages. Il a fallu trancher au vu de nos exigences [9].

6. La force qui permet de faire voler le drone, de le soulever dans les airs.
7. Interrupteur principal.
8. *Electronic Speed Control.*
9. cf. exigence E2.11 du cahier des charges au A.1 page 51.

Le drone image 5.1b page 41 a l'avantage de sa petite taille. Cela lui permet de passer dans un bâtiment par les portes et les fenêtres, de dépasser les obstacles facilement tout en ayant un cadre qui protège ses hélices. Néanmoins, le grand défaut de ce drone réside dans son autonomie. En effet, du fait de ses dimensions réduites, et de ses moteurs moins puissants que ceux des drones aux images 5.1a page 41 et 5.1c page 41, le drone (5.1b page 41) est incapable de transporter une batterie de trop grande capacité [10]. Nous ne pouvons donc pas utiliser le drone sur une longue période.

Les drones des images 5.1a page 41 et 5.1c page 41 ont l'avantage de leurs grandes dimensions. Ils peuvent ainsi transporter davantage de capteurs, et possèdent un temps d'utilisation plus long. La construction du châssis en carbone les rend également plus résistants aux chocs que le drone en 5.1b page 41. Cependant, d'un autre côté, de telles dimensions les rendent beaucoup moins adaptés à des environnements confinés, tels l'intérieur d'un bâtiment, ou d'un chapiteau [11].

Pour toutes ces raisons, et comme la contrainte de taille est la plus significative, le drone présenté figure 5.1b page 41 a été retenu. Les deux autres drones seront employés en cas d'urgence.

5.3 Travaux sur le *matrice 600*

Figure 5.3 : Drone *matrice 600*. [Gaz16]

Le drone *principal* étant choisi [12], en même temps que sa réparation, nous avons pris en main un drone opérationnel, le *matrice 600*. En effet, ce premier contact avec la technologie des quadrirotors a permis de découvrir les problèmes récurrents des drones, notamment concernant la stabilité, et de se familiariser avec les capteurs embarqués que possèdent les drones.

D'une part, nous avons travaillé sur une technique de stabilisation horizontale [13] par caméra. Il a donc était question d'installer une caméra [14] en-dessous du drone, afin de stabiliser le drone à l'aide d'un logiciel présent sur le *matrice 600*.

D'autre part, nous avons pris en main un *GPS*. Le *GPS* du *matrice 600* a donc été calibré, ce qui a permis de comprendre le fonctionnement d'un *GPS*.

Après avoir fait nos premiers pas dans la technologie des quadrirotors, il était alors nécessaire de chercher davantage à comprendre le modèle théorique d'un quadrirotor, ce qui fait l'objet du 5.4 page suivante.

10. Car trop lourde.
11. cf. note de bas de page 10 page 8.
12. cf. 5.2 page ci-contre.
13. en roulis, afin d'éviter les oscillations du drone.
14. La caméra « Zenmuse Z3 ».

5.4 Modèle théorique d'un quadrirotor

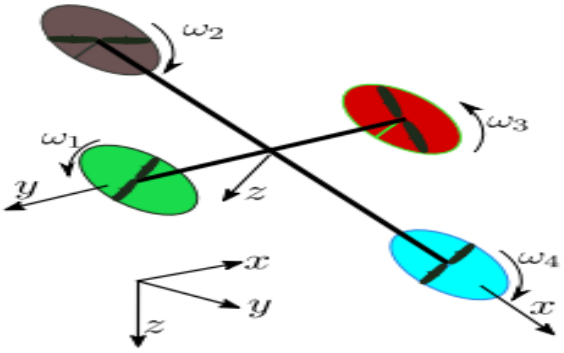
Dans un souci de compréhension du fonctionnement d'un quadrirotor, mais également parce que l'objectif final est de rendre le drone 5.1b page 41 autonome [15], le modèle théorique du quadrirotor présenté figure 5.4 a été étudié.

Le quadrirotor est un drone volant propulsé par quatre moteurs. Chaque moteur met en mouvement une hélice. Les moteurs sont commandés de telle sorte que les hélices positionnées deux à deux face à face [16] aient le même sens de rotation, dans le sens direct [17] ou dans le sens opposé [18]. Le quadrirotor va donc de ce fait pouvoir être incliné et avancer. Après avoir cherché les équations d'état du système, présentées figure 5.4a avec un schéma illustratif présentant les conventions figure 5.4b, une simulation sous *Python* a été réalisée afin de tester divers codes de régulation, le but étant au final de comprendre comment fonctionne un quadrirotor afin de s'intéresser à l'asservissement d'un quadrirotor dans un second temps. Inspiré par [Jau15], le code présenté en annexe au C page 65, permet de simuler un asservissement du drone en position. Le quadrirotor va donc stabiliser puis conserver une position donnée.

Après avoir simulé un asservissement en position d'un quadrirotor, il serait alors intéressant d'adapter les algorithmes d'évitement d'obstacles et de suivi de *waypoints* par champ de potentiel développés au 4.3 page 25 à l'asservissement d'un drone aérien de type quadrirotor. Les problèmes alors rencontrés pour les robots terrestres pourraient ne pas se poser pour les robots aériens.

$$
\begin{cases}
\dot{\mathbf{p}} &= \mathbf{R}(\varphi,\theta,\psi)\cdot\mathbf{v}_r \\
\begin{pmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix} &= \begin{pmatrix}1 & \tan\theta\sin\varphi & \tan\theta\cos\varphi\\ 0 & \cos\varphi & -\sin\varphi\\ 0 & \frac{\sin\varphi}{\cos\theta} & \frac{\cos\varphi}{\cos\theta}\end{pmatrix}\cdot\boldsymbol{\omega}_r \\
\dot{\mathbf{v}}_r &= \mathbf{R}^T(\varphi,\theta,\psi)\cdot\begin{pmatrix}0\\0\\g\end{pmatrix}+\begin{pmatrix}0\\0\\-\frac{\tau_0}{m}\end{pmatrix}-\boldsymbol{\omega}_r\wedge\mathbf{v}_r \\
\dot{\boldsymbol{\omega}}_r &= \mathbf{I}^{-1}\cdot\left(\begin{pmatrix}\tau_1\\ \tau_2\\ \tau_3\end{pmatrix}-\boldsymbol{\omega}_r\wedge(\mathbf{I}\cdot\boldsymbol{\omega}_r)\right)
\end{cases}
$$

(a) Équations d'états d'un quadrirotor



(b) Quadrirotor

Figure 5.4 : Modèle théorique d'un quadrirotor selon [Jau15].

5.5 Choix de l'auto-pilote

5.5.1 Ensemble des solutions

Afin de réaliser la fonction principale *FP1.2.1.1* [19] le drone quadrirotor 5.1b page 41 doit être asservi. Pour ce faire, un composant appelé *auto-pilote* est nécessaire.

Un *auto-pilote* est une carte électronique qui permet de centraliser et de traiter certaines données capteurs [20]. Il est programmable avec certaines librairies et peut gérer l'évolution autonome du drone et donc son asservissement une fois correctement programmé. [Pro18]

Afin de choisir un *auto-pilote*, nous avons regardé ceux proposés sur le marché, comme par exemple *OpenPilot*, *NuttX*, ou encore *Auto-Tune*. Néanmoins, l'ancienneté, la fiabilité, et la

15. Exigence *E2.03* du cahier des charges au A.1 page 51.
16. La disposition des hélices est représentée figure 5.4b.
17. clockwise.
18. counterclockwise.
19. cf. cahier des charges fonctionnelles au B page 55.
20. Comme par exemple celles des *GPS*.

simplicité plus prononcée de prise en main de l'auto-pilote [21] *APM* [22] d'*Ardupilot* l'ont presque imposé comme unique alternative au regard de nos exigences.

5.5.2 Auto-pilote retenu

L'auto-pilote *APM* présenté figure 5.5 contient des capteurs principaux comme des accéléromètres qui mesurent l'accélération, des gyroscopes qui mesurent la position et des magnétomètres qui mesurent le cape. Néanmoins, sa grande adaptabilité permet d'ajouter multitude de capteurs annexes comme un *GPS*, télémètre ultrasonore ou laser (lidars), ... Ce qui est utile pour réaliser les missions de cartographie [23] et de détection des *OPI* [24].

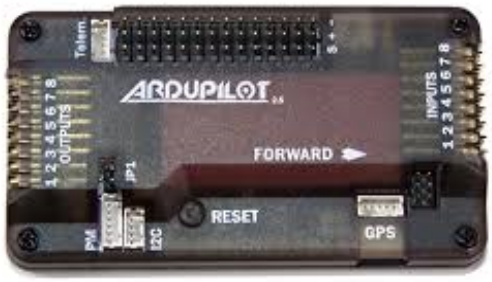


Figure 5.5 : Auto-pilote *Ardu Pilot Mega*. [Pro18]

5.5.3 Les fonctionnalités

Comparé aux autres solutions étudiées, et mentionnées au 5.5.1 page ci-contre, l'autopilote *APM* possède notamment les avantages suivant: [Pro18]

- Il permet de stabiliser le drone contre des rafales de vent.
- Il renvoie la position du drone où la connexion a été perdue.
- Il contient un système de sécurité au décollage et à l'atterrissage.
- Il est opérationnel pour différents modèles de robots aériens [25].

5.5.4 Configuration de l'auto-pilote

Pour commencer à utiliser l'auto-pilote *APM*, présenté figure 5.5, il est nécessaire de le configurer. Pour ce faire, *Mission Planner* a été retenu. La figure 5.6 page suivante montre la configuration de l'*APM*.

Le choix de ce gestionnaire d'auto-pilote a notamment été guidé par le fait, que *Mission Planner* détecte les anomalies de fonctionnement des drones, et met à disposition un tutoriel détaillé pour sa prise en main, ce qui est une différence notable par rapport aux autres solutions. Ce logiciel est également préconisé par les constructeurs de l'*APM*. Par ailleurs, *Mission Planner* affiche toutes les mesures des capteurs qui équipent l'*APM*.

21. Notamment pour sa programmation postérieure.
22. *Ardu Pilot Mega*.
23. *FP1.3.3.1* dans le cahier des charges fonctionnelles annexe B page 55.
24. cf. note de bas de page numéro 11 page 8 pour une définition. Cette mission fait référence à la *FP1.3.2.2* du cahier des charges fonctionnelles annexe B page 55.
25. Avions, quadrirotors ...

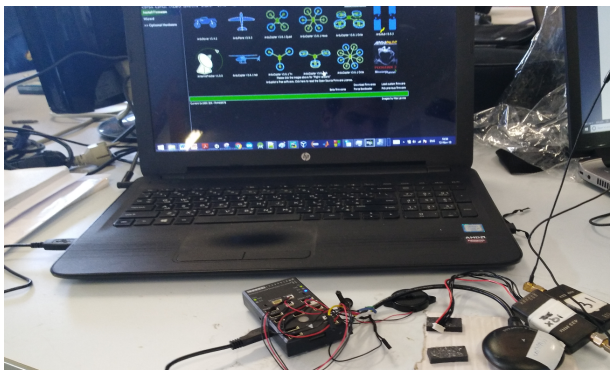


Figure *5.6 :* Configuration de l'auto-pilote avec *Mission Planner*.

Pour la configuration de *Mission Planner*, il faut dans un premier temps choisir le type de drone, comme le montre la figure 5.6. *Mission Planner* gère alors le téléchargement des librairies, qui contiennent toutes les informations pour le modèle de drone choisi. Ensuite, il allume les moteurs, s'assure que le branchement est correcte, et les calibre. Enfin, *Mission Planner* paramètre la télécommande par laquelle l'utilisateur pourra commander son drone [26]. Finalement, le drone est prêt à décoller.

26. L'évolution autonome est à programmer

Chapitre 6

Tests

L'une des leçons de ce projet peut se formuler en la maxime suivante: « ne jamais commencer à développer un système sans avoir une idée précise de comment tester le système.» Cette partie expose les différents tests réalisés au cours du projet.

6.1 Tests unitaires

Lors du développement des fonctionnalités d'un système, il est crucial de commencer par essayer les solutions trouvées indépendamment du système réalisé [1]. Après avoir mis en place une solution technologique répondant à une certaine exigence, des tests *unitaires* doivent être organisés.

Tout au long du projet, notre démarche projet a été de tester immédiatement les solutions proposées sur un ordinateur ou en simulation, mais dans un premier temps, pas sur les robots. Des fiches de tests sont disposées en annexe D page 67. Voici une liste des tests unitaires menés:

- Tests des différents capteurs: *Kinect v2*, *ZED caméra*, *GPS*, *Lidar Hokuyo* et *RP Lidar A2*, *centrale inertielle SBG*.
- Tests des différents éléments des robots: moteurs, *Pololu Maestro*,*Ardu Pilot Mega*, chaînes [2], lampes, servomoteurs, batteries ...
- Tests de la benne [3].
- Tests logiciels: *Ardupilot*, *Mission Planner*.
- Tests des programmes de cartographie.
- Tests des programmes d'évitemment.

6.2 Tests d'intégration

Une solution requiert un test unitaire si derrière cette solution technologique va potentiellement être intégrée au système conçu. Tout test unitaire doit donc être suivi d'un test d'intégration au cours duquel la solution technologique alors mise en place est ajoutée au développement validé jusque là. Des fiches de tests sont disposées en annexe D page 67. Voici une liste des tests d'intégration menés [4]:

- Tests des différents capteurs: *Kinect v2*, *ZED caméra*, *GPS*, *Lidar Hokuyo* et *RP Lidar A2*, *centrale inertielle SBG*.
- Tests des différents éléments des robots: moteurs, *Pololu Maestro*,*Ardu Pilot Mega*, chaînes, lampes, servomoteurs, batteries ...
- Tests de la benne.
- Tests logiciels: *Ardupilot*, *Mission Planner*.

1. Dans un cadre plus restreint, qui permet de maîtriser le cadre du développement, et surtout de ne pas faire régresser l'état d'avancement du système.
2. qui ont du être réparées.
3. cf. 4.18 page 38
4. Ce sont les mêmes que les tests unitaires.

- Tests des programmes de cartographie.
- Tests des programmes d'évitemment.

6.3 Validation Fonctionnelle Usine [5]

6.3.1 Contextualisation

La vérification fonctionnelle usine du système réalisé lors du projet *ERL2019* s'est tenue durant la semaine de concours du 18 au 23 février en Espagne, à Séville. Chaque jour les robots devaient réaliser certaines missions, à la suite desquelles notre prestation était notée. Des points étaient attribués par mission accomplie. Une fiche type de notation est à disposition en annexe E page 70. Cette fiche est la fiche du dernier jour de la compétition qui constitue la fiche de la *vérification fonctionnelle usine*. Chaque *fonction principale* est listée et une croix indique que l'implémentation de la fonctionnalité était satisfaisante pour le jury.

6.3.2 Analyse des résultats

6.3.2.1 Partie terrestre

Au niveau du robot terrestre, la plupart des fonctionnalités exigées par le concours étaient implémentées et satisfaisantes. Nous n'avons pas implémenté toutes les fonctionnalités entièrement nous mêmes. Certains doctorants et enseignant-chercheurs de l'*ENSTA-Bretagne* nous sont venus en aide [6]. Ne sont présentées dans ce rapport que les fonctionnalités que nous avons entièrement réalisées. Néanmoins, de toutes les fonctionnalités développées durant ce projet, seule la cartographie *3D* n'était à l'époque du concours pas implémentée [7]. L'évolution autonome du robot terrestre [8] était performante. Nous avons alors réalisé à quel point cette partie était cruciale pour le concours, ce que nous avons encore cherchés à améliorer après la compétition, cf. 4.3.2.4 page 28. Par ailleurs, certains problèmes de connexions *Wifi* et des branchement défaillants avec les batteries nous ont coûtés des points. Il est important de préciser que la partie terrestre était très fiable et nous a permis de nous positionner troisième, et même second en fin de compétition.

6.3.2.2 Partie aérienne

Le mode manuel du robot a finalement réussi à bien marcher le dernier jour du concours. Il y a eu pendant la semaine des problèmes de stabilité, de régulation, et de manœuvrabilité [9] des drones, ce qui a coûté la vie à l'un d'entre eux. Compte tenu de ces soucis, la partie d'évolution autonome n'a pas pu être abordée. Pourtant, avec un ou deux jours de plus, en suivant l'amélioration faite de jours en jours, il est presque certain que le drone aérien aurait été opérationnel et prêt à marquer plus de points. Cette partie reste à développer davantage.

6.3.2.3 Interaction entre les deux parties

Le temps nous a manqué pour pouvoir mettre en place une interaction entre les robots aérien et terrestre. Nous avons préféré nous consacrer dans un premier temps à faire fonctionner les robots indépendamment. L'interaction entre robots n'était pas demandée par le concours et relevait plus de l'optionnel, même si elle peut être intéressante dans le cadre de la compétition.

6.3.2.4 Analyse globale

Globalement, le système ne répond pas à l'entièreté du cahier des charges décrit A.1 page 51 et n'est qu'un prototype non achevé du système *SIC* décrit au 3 page 13. Néanmoins, dans le temps

5. VFU
6. Ils sont listés page 4.
7. Elle a été développée après le concours, cf. 4.5 page 32
8. Evitement d'obstacle ...
9. Principalement dûs à l'autopilote.

imparti prévu pour ce projet, et prenant en compte la date du concours qui était positionnée mi-projet, l'objectif principal a été réalisé: nous sommes arrivés troisième du concours, talonnant le second. La base terrestre développée est notamment robuste et fonctionnelle, et pourra être reprise pour les éditions suivantes de la compétition *ERL*.

Conclusion

Analysant le travail et les résultats fournis par l'équipe projet *ERL2019*, du système *SIC*, une partie terrestre fiable et robuste a été produite. L'équipe projet a choisi de se concentrer sur la base terrestre à des fins stratégiques. Par ailleurs, l'objectif principal du projet, celui de participer à la compétition, à été atteint avec succès puisque l'*ENSTA-Bretagne* est arrivée en troisième position face à des équipes bien plus expérimentées.

Des conseils pour les éditions futures et donc les étudiants qui reprendront ce projet sont bien entendu de rigueur. Il est primordial de disposer d'une base terrestre très fiable pour espérer obtenir des résultats convenables. La base terrestre développée au cours de ce projet pourra être reprise et la partie aérienne pourra alors davantage être développée, notamment en ce qui concerne l'asservissement. Les travaux de la partie terrestre concernant la cartographie et les algorithmes d'autonomie développés dans le cadre de ce projet pourront être adaptés aux cas aérien. Par ailleurs, nous conseillons de poursuivre la mise sous *ROS* des robots, notamment du robot terrestre, ainsi que des travaux de cartographie dont les derniers résultats se sont montrés prometteurs. Nous faisons confiance à M. Le Bars pour bien sûr continuer son travail d'encadrement qui nous a amplement aidé.

Annexe A

Cahier des charges

Nom	ID	Description	Type
Analyse environnement	**GLE1**		
Capteurs	**1,01**	Le système doit pouvoir traiter des données renvoyées par les capteurs des robots.	Fonction
Format Navigation	**1,02**	Le système doit stocker les données de navigation en KML (Keyhole Markup Language).	Contrainte
Mat-Sonar	**1,04**	Le système doit utiliser un mat de sonars.	Contrainte
Autonomie	**GLE2**		
Communication Entre Robots	**2,01**	Le système doit permettre la communication entre les robots terrestres et aériens existants.	Fonction
Intinéraire	**2,02**	Le système doit pouvoir permettre au robot la suivi d'un itinéraire constitué d'une succession de coordonnées géographiques (waypoints).	Fonction
Mode-Autonome	**2,03**	Si le système est en mode autonome, il doit pouvoir fonctionner en toute autonomie.	Fonction
Contrôle-Aérien	**2,04**	Le système doit contrôler un quadrirotor via Ardupilot.	Contrainte
Contrôle	**2,05**	Le système doit contrôler les robots terrestre et aériens.	Fonction
RelaisWifi	**2,06**	Le système doit utiliser un robot de relais wifi pour assurer la communication via Wifi entre l'opérateur et les robots, mais également entre les robots.	Contrainte

tournez la page S.V.P

Nom	ID	Description	Type
Obstacle-Evitement	**2,07**	Les système doit permettre aux robots d'éviter les obstacles de leur environnement.	Fonction
Sauvegarde Données	**2,08**	Le système doit stocker les données d'intérêt sur un clé USB au format NTFS. A savoir, les données de navigation en KML, données d'avancement de mission, données de cartographie, données de reconnaissance d'objet, données de communication inter-robots.	Contrainte
Portes	**2,09**	Le système doit permettre aux robots de franchir des portes ou des excavations.	Fonction
Stabilisation-Aérienne	**2,10**	Le système doit stabiliser horizontalement les robots aériens.	Fonction
Intérieur et Extérieur	**2,11**	Le système doit permettre aux robots aériens et terrestres de fonctionner en intérieur (bâtiment) et en extérieur.	Fonction
Trouver Entrée	**2,12**	Le système doit permettre aux robots (aériens et terrestres) de trouver une entrée pour accéder à l'intérieur du bâtiment. Les entrées sont indiquées par un marqueur de couleur verte.	Fonction
Waypoint atteint	**2,13**	Le système doit atteindre un waypoint à 3 m de précision.	Contrainte
Obstacle Naturel	**2,14**	Le système doit détecter des obstacles naturels.	Fonction
Format Waypoints	**2,15**	Le système doit stocker les points d'itinéraire au format KML.	Contrainte
Missions	**GLE3**		
Trousse-de-soin	**3,01**	Le système doit permettre aux robots de déposer une trousse de soin de moins de 0.5 kg.	Fonction
Radius Aérien	**3,02**	Le système doit permettre aux robots aériens un dépôt de la trousse de soin dans un périmètre de 2 m autour du blessé.	Contrainte
Bras-Robotisé/Benne	**3,03**	Le système doit permettre aux robots terrestres de déposer la trousse de soin via un bras robotisé ou une benne.	Contrainte
aide Missing Worker	**3,04**	Le système doit pouvoir permettre aux robots de déposer précisément une trousse de soin auprès d'un mannequin.	Fonction

tournez la page S.V.P

Nom	ID	Description	Type
Radius Terrestre	**3,05**	Le système doit permettre aux robots terrestres un dépôt de la trousse de soin dans un périmètre de 1 m autour du blessé.	Contrainte
Missing-Worker	**3,06**	Le système doit détecter les blessés. Les blessés sont des mannequins de couleur orange.	Fonction
OPI (object of potential interest)	**3,07**	Le système doit détecter et reconnaître les OPI, en milieu extérieur comme en milieu intérieur.	Fonction
Dégâts Bâtiment	**3,08**	Le système doit identifier les dégâts du bâtiment. Les dégâts sont signalés par des marqueurs de couleur rouge.	Fonction
Format OPI	**3,09**	Le système doit stocker les données des OPI au format KLM.	Contrainte
Données OPI	**3,10**	Le système doit renseigner pour un OPI: position de l'OPI, image de l'OPI.	Contrainte
Cartographie-2D	**3,11**	Le système doit permettre une cartographie 2D du bâtiment à explorer.	Fonction
Kinect	**3,12**	La système doit utiliser les informations renvoyées par la Kinect pour cartographier son environnement.	Contrainte
Cartographie-3D	**3,13**	Le système doit permettre une cartographie 3D du bâtiment à explorer.	Fonction
Cartographie Extérieur bâtiment	**3,14**	Le système doit permettre aux robots terrestres et aériens de cartographier l'extérieur du bâtiment.	Fonction
Cartographie-Post-traitement	**3,15**	Le système doit générer les cartographies de son environnement en post-traitement.	Contrainte
Format Carte	**3,16**	Le système doit stocker les données de la cartographie en image, ou bien des données traitables avec MeshLab (.ply).	Contrainte
Sécurité	**GLE4**		
Arrêt-Urgence	**4,01**	Le système doit permettre l'arrêt des robots aériens et terrestres en cas d'urgence.	Fonction
Mode-Manuel	**4,02**	Si le système est en mode manuel, il doit être commandable par un opérateur humain.	Fonction

tournez la page S.V.P

Nom	ID	Description	Type
voyant d'alerte	**4,03**	Le système doit permettre aux robots de signaler leur présence par des signaux lumineux.	Contrainte
Mode-Semi-Autonome	**4,04**	Si le système est en mode semi-autonome, il doit pouvoir prendre en compte des commandes données par un opérateur humain tout en ayant un fonctionnement autonome.	Fonction
Masse	**4,05**	Le système ne doit pas faire dépasser 25 kg aux robots aériens, et 350 kg aux robots terrestres.	Contrainte
Intégration	**GLE5**		
Intégration-Existant	**5,01**	Le système doit s'intégrer sur les robots et l'existant.	Contrainte
Simplifié-Intégration	**5,02**	Le système doit simplifier son intégration à l'existant.	Fonction

TABLE *A.1:* Cahier des charges du *SIC*.

Annexe B

Cahier des charges fonctionnelles

TYPE	ID relatif	Expression ou identification	Exigences relatives ou Traçabilité	Critères
FC	**1**	posséder une autonomie suffisante.	GLE 2	temps des interventions: autonomie -en action aérien: 5 - 10 min et terrestre: 1h. -au repos: aérien:2h et terrestre:2h.
FC	**2**	utilser les sources d'énergie présentes.	GLE 2	électrique, batteries embarquées, moteurs Ex: aérien: moteur de 100 A pic et 50 A, poids (poussée de 8kg minimum).
FC	**3**	satisfaire aux normes de sécurité.	GLE 4 E4.05	réglementation du concours (liaison d'urgence): 1 bouton d'arrêt d'urgence. Masses limites.
FC	**4**	satisfaire aux normes techniques.	GLE 4	ne pas communiquer en 2,4 Ghz (aérien)
FC	**5**	utiliser les données transmises en début de mission ou en cours de mission.	GLE 3	format de données CSV
FC	**6**	recevoir ses ordres d'intervention des sauveteurs.	GLE 4	accès par un ordinateur à distance.
FC	**7**	résister à l'environnement et être opérationnel pour un tel environnement.	GLE 2	pluie (cage IP65), gravier(roues larges)
FC	**8**	s'intégrer et s'adapter à l'existant.	GLE 5 E5.01 - E5.02	utiliser du C++, OpenCV

Table *B.1:* Cahier des charges fonctionnelles: fonctions de contrainte du *SIC*.

Table *B.2:* Cahier des charges fonctionnelles: fonctions principales du *SIC*.

TYPE	ID relatif	Expression ou identification	Exigences relatives ou Traçabilité	Données E/S	Comportement	Temps
FP	**1**	contrôler et adapter les robots sauveteurs.	GLE 1- 5 Haut niveau E2.05			
FP	**1.1**	percevoir le mileu environnant.	GLE 1 E1.01 - E1.02 - E1.03	E:Energie électrique, alimentation, ou demande de données. S: données des capteurs présents sur les robots sauveteurs. (données brutes, le format dépend du capteur)	Retourner les données brutes acquises par les capteurs présents sur les robots sauveteurs.	Exécutée à intervalle de temps régulier (défini par l'horloge interne des robots sauveteurs).
FP	**1.1.1**	percevoir l'environnement à porixmité des robots sauveteurs terrestres.	GLE 1 E1.01 - E1.02 - E1.03			
FP	**1.1.2**	percevoir l'environnement à porixmité des robots sauveteurs aériens.	GLE 1 E1.01 - E1.02 - E1.03			

Tournez la page S.V.P

TYPE	ID relatif	Expression ou identification	Exigences relatives ou Traçabilité	Données E/S	Comportement	Temps
FP	**1.2**	évoluer en toute autonomie en mode autonome.	GLE 2 E2.03	E: Si les sauveteurs émettent un contreordre : ordre d'intervention autonome (signal du sauveteur, et remplissage d'un fichier d'intervention donné aux robots sauveteurs: par exemple une liste de waypoints à suivre). Le signal est envoyé aux robots sauveteurs. S: signal d'alimentation électrique des robots sauveteurs, et requête d'autonomie analysée (liste des waypoints d'intervention, et informations d'intervention).	Emission d'une requête d'intervention et attente de validation par les robots sauveteurs.	Exécuté à chaque fois qu'un sauveteur donne un ordre au SIC.
FP	**1.2.1**	asservir les robots sauveteurs.	GLE 2 E2.02 - E2.05 - E2.06 - E2.07 - E2.09 - E2.11 - E2.12 - E2.13 - E2.14 - E2.15	E: données capteurs, données liées à l'environnement et à l'état du robot, et informations d'intervention analysées. S: commandes envoyées aux robots (commandes des moteurs par exemple) et gestion des déclenchements des missions de terrain: cartographie, OPI, ou sauvetage d'un blessé (signal d'activation).	Pilotage des robots sauveteurs de manière autonome et gestion du démarrage des missions de terrain.	Exécutée à intervalle de temps régulier (défini par l'horloge interne des robots sauveteurs).

Tournez la page S.V.P

TYPE	ID relatif	Expression ou identification	Exigences relatives ou Traçabilité	Données E/S	Comportement	Temps
FP	**1.2.1.1**	asservir les robots sauveteurs aériens.	GLE 2 E2.04 - E2.10			
FP	**1.2.1.2**	asservir les robots sauveteurs terrestres.	GLE 2 E2.02 - E2.05 - E2.06 - E2.07 - E2.09 - E2.11 - E2.12 - E2.13 - E2.14			
FP	**1.2.2**	partager les informations entre les robots sauveteurs.	GLE 2 E2.06	E: états de chaque robot sauveteur collectés de manière indépendante (poistion. . .), et rapports de mission pour chaque robot sauveteur (succés ou échec, images). S: états et rapports collectés des différents robots sauveteurs.	Partage des données sur l'évolution des robots sauveteurs entre eux. (interne au SIC)	Exécutée à intervalle de temps régulier (défini par l'horloge interne des robots sauveteurs).

Tournez la page S.V.P

TYPE	ID relatif	Expression ou identification	Exigences relatives ou Traçabilité	Données E/S	Comportement	Temps
FP	**1.2.3**	stocker les données de l'évolution autonome.	GLE2 E2.08	E: états des robots et rapports de mission. S: rien.	Stockage des données de mission et des informations sur l'évolutions des robots au cours de leur intervention sur une clé USB dans le format adéquat.	Exécutée à intervalle de temps régulier (défini par l'horloge interne des robots sauveteurs).
FP	**1.3**	réussir les missions de terrain.	GLE 3			
FP	**1.3.1**	déposer une trousse de soin.	GLE 3 E3.01 - E3.02 - E3.03 - E3.04 - E3.05 - E3.06	E: ordre de mission. Déclenchement du largage de la trousse de soin. S: rapports ou résultats de mission. Typiquement, des images ou une vidéo de la mission.	Largage de la trousse de soin à proximité du blessé. La position du blessé sera donnée au préalable.	Exécutée si requête de mission.
FP	**1.3.1.1**	déposer une trousse de soin par l'intermédiaire des robots sauveteurs terrestres.	GLE 3 E3.01 - E3.02 - E3.03 - E3.04 - E3.05 - E3.06			

Tournez la page S.V.P

TYPE	ID relatif	Expression ou identification	Exigences relatives ou Traçabilité	Données E/S	Comportement	Temps
FP	1.3.1.2	déposer une trousse de soin par l'intermédiaire des robots sauveteurs aériens.	GLE 3 E3.01 - E3.02 - E3.03 - E3.04 - E3.05 - E3.06			
FP	1.3.2	détecter des OPI.	GLE 3 E3.07 - E3.08 - E3.09 - E3.10	E: ordre de mission. Déclenchement de la détection des OPI (objets d'intérêt). S: rapports ou résultats de mission. Typiquement, une image de l'OPI et l'interprétation des robots sauveteurs.	Détection des OPI. Les positions des OPI seront connues avant le lancement de l'intervention. Les positions seront rentrées dans un ensemble de waypoints.	Exécutée si requête de mission.
FP	1.3.2.1	détecter les OPI par l'intermédiaire des robots sauveteurs terrestres.	GLE 3 E3.07 - E3.08 - E3.09 - E3.10			
FP	1.3.2.2	détecter les OPI par l'intermédiaire des robots sauveteurs aériens.	GLE 3 E3.07 - E3.08 - E3.09 - E3.10			

Tournez la page S.V.P

TYPE	ID relatif	Expression ou identification	Exigences relatives ou Traçabilité	Données E/S	Comportement	Temps
FP	**1.3.3**	cartographier en extérieur et en intérieur.	GLE 3 E3.11 - E3.12 - E3.13 - E3.14 - E3.15 - E3.16	E: demande ou requête de cartographie, signal de commande qui permet de démarrer la cartographie. S: résultats ou rapport de la cartographie. Typiquement, les images de la cartographie, les données des capteurs ayant permis la cartographie (lidar), et un fichier Meshlab avec la cartographie 3D et 2D.	Cartographie sur demande en post-traitement ou en direct (pendant) la mission. Les positions où lancer les cartographie seront données avant le lancement de l'intervention.	Exécutée sur demande d'un sauveteur en post-traitement (retour de mission) ou bien déclenchée par le robot en cours de mission (à un instant précis de la mission).
FP	**1.3.3.1**	cartographier en extérieur et en intérieur le milieu environnant des robots sauveteurs terrestres.	GLE 3 E3.11 - E3.12 - E3.13 - E3.14 - E3.15 - E3.16			
FP	**1.3.3.2**	cartographier en extérieur et en intérieur le milieu environnant des robots sauveteurs aériens.	GLE 3 E3.11 - E3.12 - E3.13 - E3.14 - E3.15 - E3.16			

Tournez la page S.V.P

TYPE	ID relatif	Expression ou identification	Exigences relatives ou Traçabilité	Données E/S	Comportement	Temps
FP	**1.4**	assurer la sécurité des acteurs: robots sauveteurs, sauveteurs, blessés, infrastructures.	GLE 4	E: requête de déclenchement d'intervention si ordre antérieur d'un sauveteur. (données relatives à l'intervention; liste des waypoints d'intervention) et données retournées par les capteurs des robots sauveteurs (positions des robots sauveteurs) S: données capteurs afin de commencer ou de poursuivre l'évolution autonome. Si la sécurité de l'intervention ne peut pas être assurée, un signal d'urgence est envoyé. Sont aussi envoyées les informations d'intervention analysées (liste des waypoints).	En présence d'un ordre d'intervention d'un sauveteur, le SIC doit vérifier la consistance de la requête émanante, qu'il peut évaluer à partir des données des capteurs. Si la requête est impossible à exécuter en l'état car elle ne permet pas d'assurer la sécurité, l'arrêt d'urgence est enclenché.	Exécutée à intervalle de temps régulier (défini par l'horloge interne des robots sauveteurs).

Tournez la page S.V.P

TYPE	ID relatif	Expression ou identification	Exigences relatives ou Traçabilité	Données E/S	Comportement	Temps
FP	**1.4.1**	être programmable et contrôlable par un sauveteur.	GLE 4 E4.02 - E4.04	E: contrordre d'intervention. Ce qui peut se traduire par la modification d'un waypoint dans l'itinéraire d'intervention. S: nouvel ordre et informations d'intervention (signal du sauveteur, et remplissage d'un fichier d'intervention donné aux robots sauveteurs: par exemple une liste de waypoints à suivre).	Modification des ordres d'intervention et prise en compte pour la suite de l'intervention. Cette fonctionnalité permet l'implémentation du mode semi-autonome et manuel. Des ordres sont alors donnés à intervalle régulier.	Exécuté sur requête des sauveteurs.
FP	**1.4.2**	communiquer en temps réel avec les sauveteurs.	GLE 4 E4.03	E: états des robots sauveteurs, rapports de mission, données capteurs adaptées. Typiquement, données adaptées pour un retour en ligne de commande dans un terminal. S: affichage des données.	Communication de l'état de l'intervention aux sauveteurs et en temps réel.	Exécutée à intervalle de temps régulier (défini par l'horloge interne des robots sauveteurs).

Tournez la page S.V.P

TYPE	ID relatif	Expression ou identification	Exigences relatives ou Traçabilité	Données E/S	Comportement	Temps
FP	**1.4.3**	arrêter l'intervention en cours en cas d'urgence.	GLE 4 E4.01	E: signal d'arrêt d'urgence ou signal électrique. (PWM) S: voyants d'arrêt d'urgence enclenchés.	Arrêt des robots sauveteurs en cas de dysfonctionnement. Les robots sauveteurs sont arrêtés (plus de mouvement), mais les sauveteurs ont toujours accés au SIC afin de traiter des données ou palier le problème.	Exécutée si requête des robots sauveteurs ou des sauveteurs.

Annexe C

Simulation du modèle d'un quadrirotor

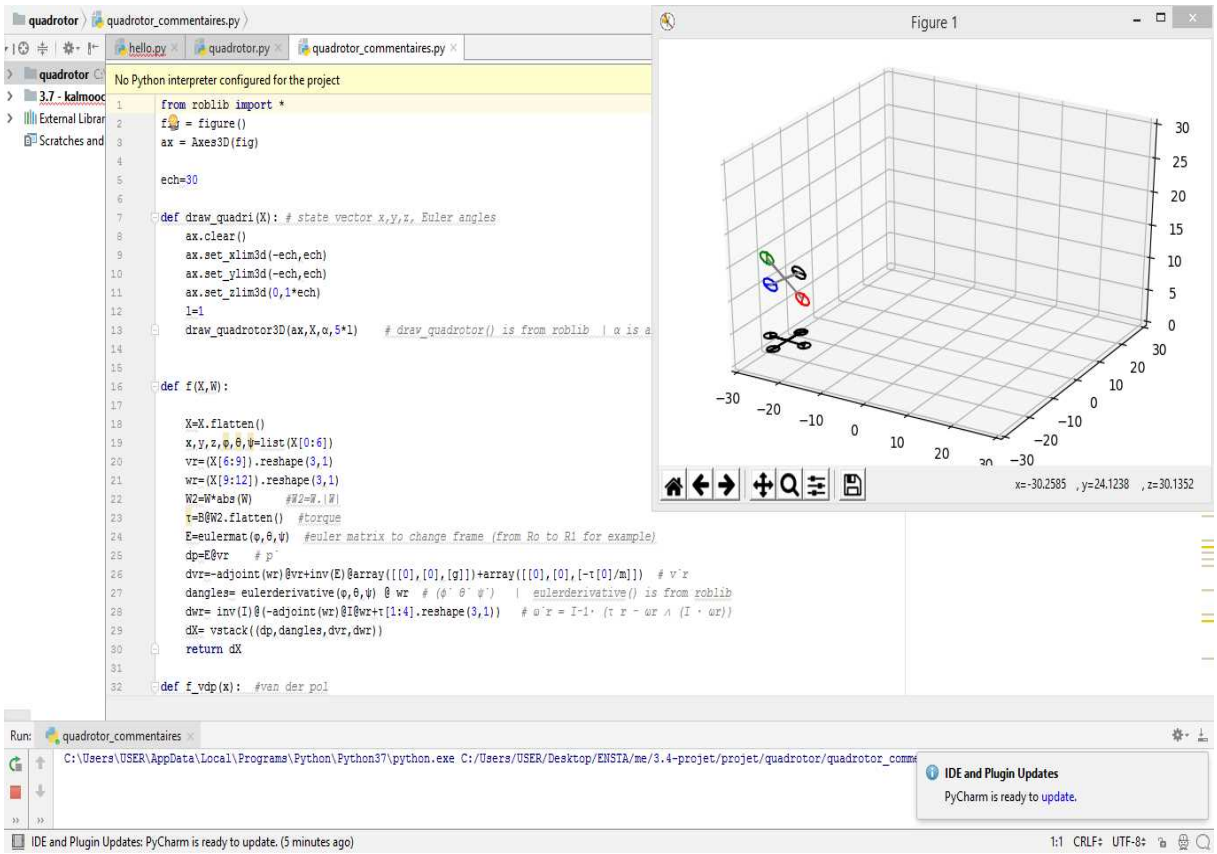


Figure *C.1 :* Algorithme de simulation d'un modèle de quadrirotor. (1/3)

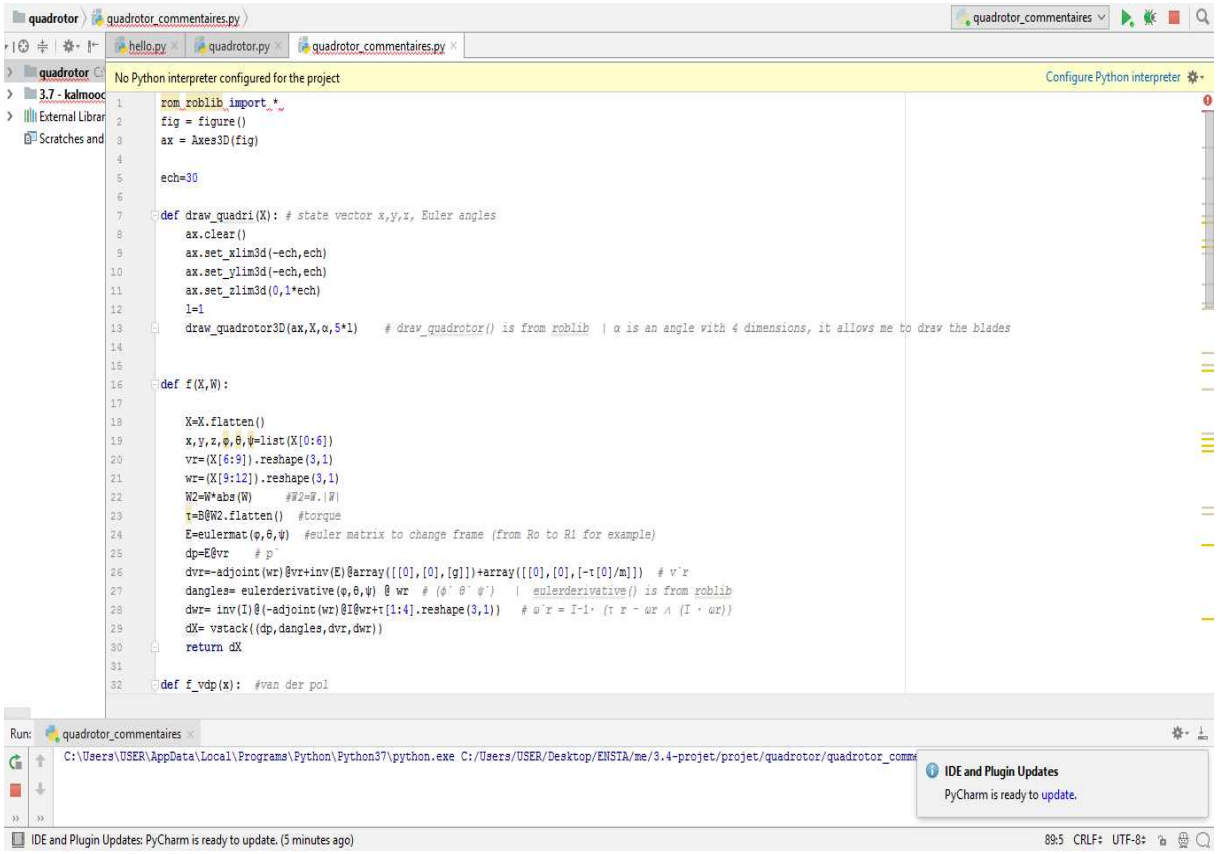


FIGURE *C.2 :* Algorithme de simulation d'un modèle de quadrirotor.(2/3)

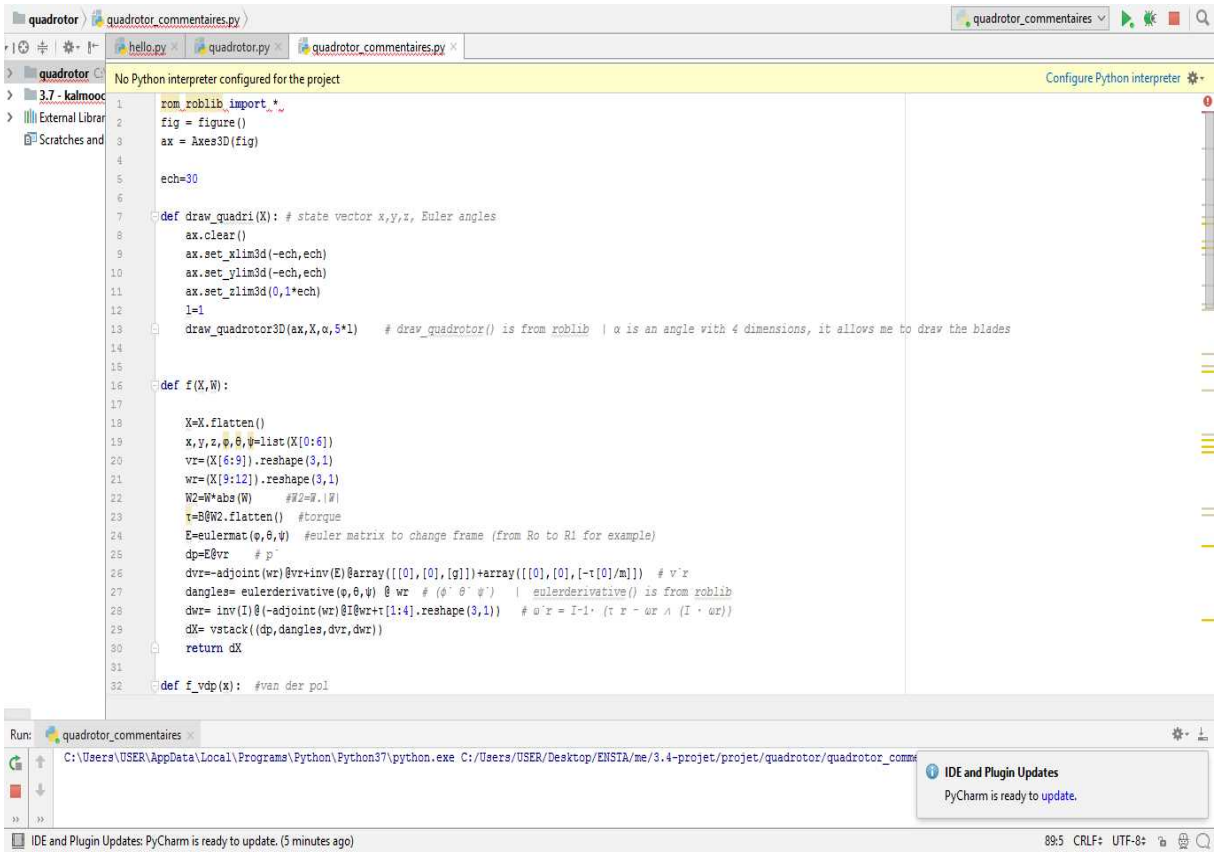


FIGURE *C.3 :* Algorithme de simulation d'un modèle de quadrirotor.(3/3)

Annexe D

Fiches de tests

Cette annexe présente une partie des tests d'intégration effectués lors du projet. Seuls les tests les plus intéressants ont été sélectionnés. Une liste des autres tests effectués est présentée au 6 page 47.

D.1 Fiche de test n°1: *lidar*

Description

Le *lidar* est un capteur qui permet de scanner son environnement. Qu'il soit conçu pour un balayage *3D* ou bien *2D*, le *lidar* émet des faisceaux infrarouges et en reçoit les échos [1]. Ainsi, ce capteur renvoie un point [2] dans l'espace environnant, qui indique la présence d'un obstacle.

Tests unitaires

N°	Action	Attendu	Résultat
1	Connecter le Lidar à un ordinateur	Le périphérique USB est bien reconnu	✓
2	Lancer le Lidar	Le lidar s'allume	✓
3	Acquérir des valeurs	Un nuage de points s'affiche	✓
4	Avoir des valeurs cohérentes	Les valeurs correspondent aux obstacles	✓

Tests d'intégration

N°	Action	Attendu	Résultat
1	Réaliser un test sur l'ordinateur embarqué du robot	Le test doit être concluant	✓
2	Fixer le capteur sur le robot	Le capteur est solidaire du robot	✓
3	Acquérir les valeurs du Lidar via le programme principal	Le programme principal peut exploiter le *Lidar*	✓

Remarques

Pendant le concours *ERL2019*, nous avons remarqué qu'un énorme bruit était présent sur les données renvoyées par le capteur, phénomène jamais observé à *Brest* lors des divers tests. Nous observions des points qui n'étaient pourtant pas des obstacles. Le *Lidar* était en fait perturbé

1. Réfléchis sur des obstacles.
2. En général, une coordonnée (x, y) dans le repère du *lidar*.

par les rayonnements infrarouges du soleil. En prenant un capteur similaire d'une autre marque, *RP lidar*[3], nous avons résolu ce problème.

D.2 Fiche de test n°2: *Pololu Maestro*

Description

La *Pololu Maestro* est un microcontrôleur qui permet de générer des impulsions *PWM*[4]. Cette modulation de fréquence permet de faire varier la tension moyenne envoyée à un autre système du robot, comme par exemple les moteurs ou les servomoteurs.

Tests unitaires

N°	Action	Attendu	Résultat
1	Connecter la Polulu à un ordinateur	Le périphérique USB est bien reconnu	✓
2	Reconnaître la carte dans le logiciel Polulu Maestro Centre	Le logiciel indique que la carte est connectée	✓
3	Envoyer une impulsion PWM à un moteur de test	Le moteur tourne	✓

Tests d'intégration

N°	Action	Attendu	Résultat
1	Réaliser un test sur l'ordinateur embarqué du robot	Le test doit être concluant	✓
2	Fixer le capteur sur la boîte étanche du robot	Le capteur ne sort pas de la boîte du robot	✓
3	Faire tourner les moteurs droit et gauche dans les deux sens	Les moteurs répondent bien aux commandes	✓

Remarques

La *Pololu Maestro* est très simple d'utilisation. Il n'a fallu que très peu de temps pour la prendre en main et réaliser les divers tests. Elle a très bien fonctionné lors du concours.

3. cf. 4.11a page 29.
4. *Pulse Width Modulation.*

D.3 Fiche de test n°3: *Kinect v2*

Description

La *Kinect v2* est un capteur développé par Microsoft pour la console de salon *XBox One*, qui comprend une caméra couleur ou *RGB* et une caméra infrarouge qui renvoie des distances [5].

Tests unitaires

N°	Action	Attendu	Résultat
1	Connecter la Kinect à un ordinateur	Le périphérique USB est bien reconnu	✓
2	Lancer le SDK	La Kinect est reconnue	✓
3	Acquérir les images	Trois images s'affichent : l'image couleur, celle de profondeur et celle des infrarouges brutes	✓

Tests d'intégration

N°	Action	Attendu	Résultat
1	Réaliser un test sur l'ordinateur embarqué du robot	Le test doit être concluant	✓
2	Fixer le capteur sur le robot	Le capteur est solidaire du robot et placé à l'avant du robot	✓
3	Acquérir les valeurs de la Kinect via le programme principal	Le programme principal peut exploiter les données de la Kinect	✓, mais il faut passer par un programme qui convertit les données brutes de la Kinect en trois images: couleur, profondeur, et infrarouge brute

Remarques

La *Kinect v2* est assez volumineuse et prend beaucoup de ressources, mais elle constitue un atout inestimable pour réaliser de la cartographie *3D*, cf. 4.5 page 32.

5. Des données en profondeur. (cf. 4.5 page 32 pour plus de détails)

Annexe E

Fiche de VFU [1]

1. Vérification fonctionnelle usine

TBM 2: Emergency in a building (Land + Air)

Team name: ENSTA BRETAGNE
Referee I (Land): MARIA, Referee II (Land): VICTOR
Referee I (Air): PACO, Referee II (Air): CARLOS
Date (DD/MM/YYYY): 23/02/2019, Time (24:00): 10:40
Duration: _____ (Max. 90 min) ☐ Timeout

Achievements

A: 25 (24)
PB: 2

Set A1: Outdoors

An **aerial robot** reaches WP1, WP2 and WP3 within 3m, avoiding obstacles along the route.	A1.1 WP1 A ☒ A1.2 WP2 A ☒ A1.3 WP3 A ☒
An **aerial robot** autonomously avoids obstacles 1, 2 and 3.	A1.4 ☐ A1.5 ☐ A1.6 ☐
A **land robot** reaches WP1 within 3m, avoiding obstacles along the route.	A1.7 WP1 L ☒
A **land robot** reaches autonomously WP2 and WP3 within 3m, avoiding obstacles along the route.	A1.8 WP2 L ☒ A1.9 WP3 L ☒
Within **30 minutes** of start of the run, a robot reports the correct location (within radius 5 m) of the **missing worker** outside the building.	A1.10 ☒
The missing worker is detected by the robot in real-time in an **automatic** way.	A1.11 ☒
An **aerial robot** deploys the first-aid kit within a radius of 2 m from the worker found outside the building.	A1.12 ☐
A **land robot** deploys the first-aid kit within a radius of 1 m from the worker found outside the building.	A1.13 ☒
Robots recognise the **damages** in the area outside the building *(each damage can only be scored once).*	A1.14 D1 ☒ A1.15 D2 ☒ A1.16 D3 ☒
Robots localise the **unobstructed entrance** in real-time in an automatic way.	A1.17 ☒
Robots localise the **obstructed entrances** *(each unobstructed entrance can only be scored once),*	A1.18 E1 ☒ A1.19 E2 ☒ → detection ?

Set A2: Indoors

A **land robot** enters the building through the unobstructed entrance.	A2.1 ☒
An **aerial robot** enters the building through the unobstructed entrance.	A2.2 ☒
A **land robot** enters the building autonomously through the unobstructed entrance after trying to complete the outdoor achievements.	A2.3 ☒
An **aerial robot** enters the building autonomously through the unobstructed entrance after trying to complete the outdoor achievements.	A2.4 ☐
The land robot(s) builds a geometric indoor map of the building *(use the best map or a combination of land robots maps).*	A2.5 ☐ (incomplete)
The land robot(s) builds the map on-board during the operation. *(the map must be shown to the referees just after the run finishes).*	A2.6 ☐
The aerial robot builds a geometric indoor map of the building.	A2.7 ☐
The aerial robot builds the map on-board during the flight. *(the map must be shown to the referees just after the run finishes).*	A2.8 ☐
Robots recognise the **damages** in the area inside the building. *(each damage can only be scored once).*	A2.9 D1 ☒ A2.10 D2 ☒
Within **30 minutes** of start of the run, a robot reports the correct location (within radius 5 m) of the **missing worker** inside the building.	A2.11 ☐
The missing worker is detected by the robot in real-time in an **automatic** way.	A2.12 ☒
An **aerial robot** deploys the first-aid kit within a radius of 2 m from the worker found inside the building.	A2.13 ☐
A **land robot** deploys the first-aid kit within a radius of 1 m from the worker found inside the building.	A2.14 ☐

Set A3: General

The **land robot(s)** returns to the starting area once all the tasks have been done.	A3.1 ☒
The **aerial robot(s)** returns to the take-off/landing area once all the tasks have been done.	A3.2 ☒
The **land robot(s)** transmits live position and images/video to the control station during the run.	A3.3 ☒
The **aerial robot(s)** transmits live position and images/video to the control station during the run.	A3.4 ☒

Penalised Behaviours

The robot needs manual intervention during a run (e.g. the robot is stuck):	
Aerial robot	PB1 ☐ (max. 1)
Land robot 1	PB2 ☒ ☒ (max. 2)
Land robot 2	PB3 ☐ ☐ (max. 2)
The land robot leaves the operating area.	PB4 ☐ (max. 1)
The land robot-1 hits the obstacles.	PB5 ☐ ☐ ☐ ☐
The land robot-2 hits the obstacles.	PB6 ☐ ☐ ☐ ☐
The land robot changes batteries or is refuelled during the run.	PB7 ☐ (max. 1)

Disqualifying Behaviours

A robot damages competition arena (including the obstacles).	DB1 ☐
A robot does not conform to safety requirements for the competition.	DB2 ☐
The aerial robot leaves the flight volumes defined by the organisation.	DB3 ☐
The team does not provide the data after the required time.	DB5 ☐

Comment:

WARNING: A disqualifying behaviour discards all other achievements in the current task. Use it only when it is really necessary (e.g. cheating).

Benchmarking data delivered appropriately: ☒ yes / ☐ no

(Time is 60 min after the end of the team's time-slot, formats as described in the TBM)

Team leader signature:

Referee signature:

Bibliographie

[3IT] Introlab 3IT. *Rtabmap*. http://introlab.github.io/rtabmap/ (cf. p. 36).

[Bar19] Fabrice Le Bars. *Cours de robotique pratique*. http://www.ensta-bretagne.fr/lebars/robotique_pratique.pdf. 2019 (cf. p. 39).

[Ben17] Yacine Benhnini. *ERL Rapport de Stage*. Sept. 2017 (cf. p. 7).

[CVG] TUM Department of Informatics Technical University of Munich Computer Vision Group. *Visual SLAM*. https://vision.in.tum.de/research/vslam/rgbdslam (cf. p. 33).

[ERL18] ERL. *ERL Eemergency Service Robots 2019 - Rulebook*. https://drive.google.com/file/d/1-yI6JZduFEHi26yvUDDo1IMMyr5E7X62/view. Déc. 2018 (cf. p. 7-9, 14).

[Gar] Willow Garage. *ROS : an open-source Robot Operating System*. http://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf (cf. p. 33).

[Gaz16] Drone Gazette. *The all new Matrice 600 with A3 controller*. http://www.dronegazette.com/new-matrice-600-m600-a3-controller/. Juin 2016 (cf. p. 43).

[Jau14] Luc Jaulin. *Algorithme du recuit simulé*. https://www.youtube.com/watch?v=oHbTrxpnOHo&feature=youtu.be. exercice 6.6. Oct. 2014 (cf. p. 27).

[Jau15] Luc Jaulin. *La robotique mobile*. ISTE Editions. 2015 (cf. p. 26, 44).

[Les05] Patrick Lester. *A-STAR Pathfinding for Beginners*. http://csis.pace.edu/~benjamin/teaching/cs627/webfiles/Astar.pdf. Juil. 2005 (cf. p. 30).

[Lum15] Joshua S. Lum. *Utilizing Robot Operating System (ROS) in robot vision and control*. https://calhoun.nps.edu/bitstream/handle/10945/47300/15Sep_Lum_Joshua.pdf?sequence=1&isAllowed=y. Sept. 2015 (cf. p. 33).

[ML13] François Michaud Mathieu Labbé. *RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation*. https://introlab.3it.usherbrooke.ca/mediawiki-introlab/images/7/7a/Labbe18JFR_preprint.pdf. 2013 (cf. p. 33, 36, 38).

[Pro18] Ardupilot Mega Project. *Ardupilot Mega*. https://www.ardupilot.co.uk/. 2018 (cf. p. 44, 45).

[Wik] ROS Wiki. *Rtabmap ROS*. http://wiki.ros.org/rtabmap_ros (cf. p. 37).