

Résumé

Pour répondre aux besoins qui motivent leur réalisation, les systèmes robotiques mobiles doivent s'appuyer sur une connaissance plus ou moins fiable de leur environnement. On distingue la perception de l'environnement externe du robot de la perception que le robot a de lui-même –en tant que plateforme mobile– au sein de cet environnement. Ces deux versants, qui tiennent une place prépondérante dans le domaine des véhicules autonomes ou intelligents, sont connus sous le nom de SLAM (Localisation et Cartographie Simultanées). Ce rapport aborde ces questions au travers de la spécification, de la conception et de l'implémentation de briques logicielles destinées à contrôler un robot mobile et rendre état de son environnement externe tout comme de sa localisation en temps réel. Le projet décrit dans ce document recouvre le système complet, allant des périphériques physiques d'acquisition de données jusqu'à la présentation des résultats au travers d'une IHM interactive.

In order to carry out specific tasks, robotic systems must rely on a consistent environment perception. Two types of environment perception can be distinguished. The first is about external environment, and the second is about the location of the mobile platform in this environment. These two complementary aspects are known as SLAM (Simultaneous Localisation And Mapping). This report addresses these issues through the specification, the conception and the implementation of software components.

Mots-clés

Cartographie et Localisation Simultanées

Développement logiciel

Temps-réel

Système embarqué

Robot Operating System

Remerciements

En prélude à ce rapport, je tiens à remercier M. David Daumand, ingénieur logiciel au sein de la société SII, qui a défini, tutoré et défendu ce projet. Merci à lui pour le suivi et l'implication inaltérables sur lesquels j'ai pu compter durant ces six mois, et ce malgré les efforts organisationnels qui lui incombaient. J'ai tout particulièrement apprécié la passion pour les sujets de robotique et de développement qu'il a su transmettre avec patience et surtout, avec un enthousiasme constant. Ce stage se concluant par une embauche, je ne pourrais que souligner la part importante qu'il a jouée dans le cheminement menant à cette situation et les conseils expérimentés qu'il a pu prodiguer à cet effet.

Je salue sincèrement M. Adel Hafiane, enseignant-chercheur et responsable du laboratoire de vision par ordinateur de l'INSA Centre Val-de-Loire, qui m'a assistée en qualité d'enseignant-référent. Je le remercie d'avoir veillé assidûment sur ce projet, tant dans les conditions matérielles de son déroulement, que dans ses orientations technique et pédagogique au regard de ma formation. J'exprime toute ma gratitude à son égard pour nous avoir, Alban Chazot et moi, orientés vers SII pour nos stages de fin d'étude avec la confiance qui était la sienne.

Enfin je remercie l'ensemble de l'équipe qui m'a accueillie avec bienveillance, sympathie et humanité au sein de l'agence de SII Bourges. Merci aux consultants, stagiaires, Directeur de site, collaborateurs qui ont jalonnés mon quotidien et enrichi mon expérience professionnelle nouvelle dans ses aspects techniques, organisationnels et bien sûr humains.

Table des matières

Remerciements	1
1 L'environnement du stage, la Société pour l'Informatique Industrielle	2
1.1 Introduction	2
1.2 Principaux repères et évolution de la structure SII	2
1.2.1 Signalétique générale : du groupe à l'agence parisienne	2
1.2.2 Évolution de SII en France et à Bourges	3
1.3 Une mission R&D pour bâtir de nouvelles offres	4
1.3.1 SII Research dans l'agence berruyère	4
1.3.2 Fruits du partenariat avec l'INSA Centre Val-de-Loire	5
2 Définition et mise en oeuvre d'un système de SLAM en temps-réel	6
2.1 De l'expression du besoin à une spécification complète	6
2.1.1 Vision et cas d'utilisation	6
2.1.2 Formalisation d'exigences	7
2.1.3 Matériel et Interfaces	8
2.2 Théorie et choix techniques	10
2.2.1 Cartographie et Localisation Simultanées	10
2.2.2 Le méta-système d'exploitation ROS	11
2.2.3 Hector SLAM	13
2.3 Réalisations et architecture logicielles	15
2.3.1 Le réseau ROS complet	15
2.3.2 Interface Homme-Machine avec Qt Creator	18
2.3.3 Éléments de modularité mis en place	22
3 Bilan et perspectives	25
3.1 Organisation du travail	25
3.1.1 Application du référentiel qualité interne	25
3.1.2 Vers une conduite AGILE adaptée	26
3.2 Résultats en vue d'un prolongement	28
3.2.1 Un module de SLAM en adéquation avec les attentes du projet . . .	28
3.2.2 Pourquoi continuer SRT2M	28
3.3 Retour d'expérience	28
3.3.1 Apports et difficultés du projet	28
3.3.2 Évolution personnelle au sein de la structure	29

Table des figures

1.1	BU actives en IDF	3
1.2	Répartition sectorielle de SII en IDF[2]	4
1.3	Wifibot embarquant un RPLidarA2 et une caméra Theta S	5
2.1	Diagramme de cas d'utilisation du système de cartographie et de localisation d'un robot mobile	7
2.2	Exigences de Spécification Technique du Besoin Logiciel	8
2.3	Architecture physique du projet	9
2.4	Architecture physique du projet	10
2.5	Logo du méta-système d'exploitation ROS (issu du kit de presse ROS[21]) .	11
2.6	Noeuds et topics d'intérêt et actifs lors de l'exécution d'Hector SLAM	13
2.7	Visualisation avec RViz d'un jeu de données pré-enregistrées	14
2.8	Réseau ROS intégré au système	15
2.9	Systèmes de coordonnées et transformations au sein du système	16
2.10	Définition du paquet générique de transmission de données à l'IHM	16
2.11	Machine à état de l'application Qt	19
2.12	Modélisation logicielle de l'Interface Homme-Machine	20
2.13	Rendu visuel de l'IHM en mode Réception	21
2.14	Rendu visuel de l'IHM en mode rejeu	22
2.15	Boîte de dialogue de configuration du réseau ROS depuis l'IHM	23
2.16	Contrôle du réseau ROS créé dynamiquement sur l'interface graphique	24
3.1	Entrées de la fonction <i>Transmettre : Formater un message</i>	25
3.2	Sorties de la fonction <i>Transmettre : Formater un message</i>	26
3.3	Politique de gestion de branches Git adoptée	27

Introduction

Issu d'un partenariat entre SII et l'INSA Centre Val-de-Loire, ce stage vise à la réalisation d'une application logicielle au sein de la société SII, sur le site de Bourges. Il s'inscrit dans le domaine du développement informatique appliquée à la robotique. Les briques logicielles implémentées devront permettre de cartographier l'environnement d'un robot mobile, ainsi que de le localiser en quasi-temps réel. Ces deux versants s'appuient prioritairement sur des mesures spatiales, fournies par un LIDAR embarqué sur le robot.

Ce stage s'inscrit dans un périmètre plus large qu'est la mise en œuvre d'un Système Robotique Tactique Multi-Missions pour la surveillance et l'aide à la prise de décisions dans les milieux à risques (SRT2M), adressé au secteur de la Défense. Il permet à un opérateur d'effectuer des missions de contrôle, de surveillance et de recherche en terrain dangereux ou potentiellement dangereux directement depuis un "shelter", à savoir une zone abritée. L'opérateur dispose d'une flotte de robots terrestres et aériens qu'il peut téléguider à distance. À mesure que l'exploration progresse, l'opérateur visualise d'une part les données cartographiques et la localisation des véhicules au sein de cette carte, et d'autre part, des flux vidéo émanant de caméras à 360 ° également embarquées. Ces flux vidéo sont soumis à des traitements qui permettent de détecter des objets d'intérêts et de les classifier de manière automatique. C'est Alban Chazot, étudiant ingénieur à l'INSA Centre Val-de-Loire qui a assuré, pendant son stage, la réalisation de cette fonctionnalité. Les classifications sont ensuite incrustées sur la carte générée, de manière à ce que l'utilisateur puisse visionner la classe, la position et la taille des entités détectées en une seule et même zone de rendu.

Le système ainsi mis en place ouvre le champ à divers scénarios d'utilisation. Ceux-ci trouvent leur pertinence dès lors que le recours à des opérateurs humains sur le terrain concerné présente un risque ou de fortes contraintes. Un scénario possible est l'exploration d'environnements accidentés (incendies, incidents nucléaires), où la localisation de victimes (USART) et l'estimation des dégâts sont réalisables de manière sécurisée.

Annonce du plan.

Chapitre 1

L'environnement du stage, la Société pour l'Informatique Industrielle

1.1 Introduction

Ce chapitre vise à apporter au lecteur une compréhension suffisante du contexte du stage pour en saisir les enjeux.

Il s'attache d'abord à décrire la structure d'accueil, à savoir le groupe SII, au sein duquel s'articulent les agences françaises et en particulier, l'agence Île-de-France. Cette dernière englobe la structure berruyère qui a hébergé le stage. À ce titre, sa portée et son organisation seront particulièrement détaillées.

Nous spécifierons ensuite les axes stratégiques qui justifient la mise en oeuvre du projet. Une section décrira l'apport souhaité du projet pour SII tandis que la section suivante reviendra sur un acteur particulier de l'écosystème de l'entreprise : l'INSA Centre Val-de-Loire, qui s'inscrit en partenaire financier, stratégique et organisationnel en amont de ce projet.

1.2 Principaux repères et évolution de la structure SII

1.2.1 Signalétique générale : du groupe à l'agence parisienne

Couramment nommée SII, la Société pour l'Informatique Industrielle est une SA à directoire et conseil de surveillance, aujourd'hui implantée dans dix-huit pays, sur quatre continents.

Sur l'exercice 2015/2016, le groupe SII enregistre un chiffre d'affaires consolidé de 360,1M€[1], pour un résultat net de 13.13M€[1], et compte sur un effectif moyen de 5 226 collaborateurs[1].

Depuis plus de 30 ans[1][2], SII œuvre pour s'inscrire en tant que partenaire technologique de choix auprès d'une clientèle professionnelle diversifiée. Le groupe –qui a consolidé son expérience dans quatorze secteurs d'activités distincts– propose des offres liées aux savoir-faire suivants :

- **L'informatique embarquée** incluant le développement de logiciels embarqués, de contrôle commande ou encore de bancs de tests
- **L'ingénierie scientifique** qui balaye les champs de l'électronique, du traitement du signal et de la mécanique

- **Les NTIC**, englobant les problématiques de téléphonie, de web, de mobilité ou d'infrastructures diverses
- **Les Systèmes d'Informations** qui recouvrent l'informatique décisionnelle, financière ou la sécurité du SI

Les secteurs d'activités s'organisent quant à eux en BU. Chacune de ces unités s'adresse à un groupe de clients majeurs et détient un organigramme interne, placé sous la responsabilité du Directeur de BU.

L'agence Île-de-France est une des neuf agences françaises de la société. Elle est placée sous la responsabilité du Directeur d'Agence, M. Didier Bonnet. Son administration est abritée sur le site de Kennedy¹ – principalement dédié aux opérations de gestions – et recouvre deux sites supplémentaires : celui du Dynasteur² et celui de Bourges³ au sein duquel s'est déroulé ce stage.

Les activités de réalisations se concentrent pour cette agence autour de quatre secteurs d'activités qui sont présentés figure 1.1.

BU actives en Ile-de-France
(2015/2016)



*Aero-Space Defense, **Energie Transport Retail, ***Banque Assurance Mutuelle

FIGURE 1.1 – BU actives en IDF

Pour ces secteurs, un portefeuille de dix clients est responsable de 80% du chiffre d'affaires annuel de l'agence. La figure 1.1 donne la nomenclature des BU définies par l'entreprise, et illustre la clientèle française qui leur est associée.

1.2.2 Évolution de SII en France et à Bourges

SII a été créée à Paris en 1979 par un ingénieur, Bernard Huvé, qui en est aujourd'hui le Président du Conseil de Surveillance. S'en suivra la création de multiples agences provinciales, jusqu'en 1999 où, forte de 400 collaborateurs et d'une croissance soutenue, la société s'introduit en bourse. On peut relever l'internationalisation de SII, qui intervient en 2006 avec l'ouverture d'une filiale en Pologne, pour atteindre aujourd'hui un total de 17 filiales à l'étranger[1].

Dans ce qui va suivre, nous nous concentrerons uniquement sur l'implantation française de SII, qui se retrouve à présent à travers 22 sites sur le territoire.

Aujourd'hui, les secteurs d'activités clés de l'entreprise peuvent être classés selon deux catégories[2], que la figure 1.2 permet de justifier :

- **Les secteurs stratégiques**, que sont ASD et Télécoms. Etant des générateurs forts de valeur, ils concernent des marchés mûrs où SII a su établir des relations pérennes avec ses clients grand-compte.

1. 104 Avenue du Président Kennedy, 75016 Paris

2. 6, 12, 10 Rue Andras Beck, 92360 Meudon

3. 14, allée Charles Pathé, 18000 Bourges

Répartition sectorielle de SII en Ile-de-France (2015/2016)



FIGURE 1.2 – Répartition sectorielle de SII en IDF[2]

- **Les secteurs de conquête**, inclus dans les BU ETR et BAM, amenés à se digitaliser massivement

Dirigée par M. Fabrice Bosch, l'agence berruyère compte quant à elle une cinquantaine de collaborateurs répartis sur les secteurs d'activités ETR et ASD. Elle héberge également des activités de QHSE sous la responsabilité de M. Jean-Sébastien Salis.

Les activités ETR se concentrent sur le développement de solutions billettiques pour des acteurs majeurs du transport de voyageurs. En parallèle, les activités ASD sont tournées vers un ensemble d'acteurs du secteur de l'armement et de la Défense historiquement implantés à Bourges et, aujourd'hui encore, essentiels à l'économie locale[4]. Parmi ces clients on peut citer la filiale missilière du groupe Airbus MBDA, ou encore Nexter Munitions dont le site de Bourges accueille les ingénieurs et cadres affiliés aux missions de R&D. Enfin, et au regard du caractère critique des missions relevant de la défense nationale ou européenne, on notera que le site de Bourges dispose d'une habilitation à mener au sein de son infrastructure des projets classés Confidential Defense.

1.3 Une mission R&D pour bâtir de nouvelles offres

1.3.1 SII Research dans l'agence berruyère

La mission qui m'a été confiée durant ce stage est placée sous l'égide de l'entité SII Research. Sous la responsabilité du Pôle Innovation, cette structure permet de mener à bien des projets transverses aux différents secteurs d'activités, financés sur les fonds propres de l'entreprise.

Typiquement, ce type de projet sera qualifié de POC ou démonstrateur. Pour l'entreprise d'accueil, les réalisations sont menées avec les objectifs suivants :

- Valoriser l'expertise de l'entreprise dans des domaines techniques précis et actuels
- Elargir les connaissances internes en communiquant autour des savoir-faire sollicités
- Bâtir de nouvelles offres à partir de tout ou partie du POC

Ces réalisations permettent de communiquer techniquement avec des acteurs internes et des parties prenantes externes (clients, prospects ou étudiants) –notamment par le biais de médias sociaux– sans compromettre la confidentialité de projets clients.

Ce projet s'inscrit dans la mise en œuvre d'un Système Robotique Tactique Multi Missions pour le Surveillance et l'Aide à la Prise de Décisions dans les Milieux à Risques, adressé au secteur de la Défense et, plus particulièrement, aux clients berruyers de la BU ASD. Dans cette optique, le stage s'est d'emblée accompagné de l'objectif ambitieux d'être véhiculé auprès de représentants de l'entreprise Nexter Systems, à son terme ou durant des stages suivants. Ainsi, plusieurs jalons ont pu être posés avec succès dans le respect des procédures internes de validations hiérarchiques. Il a notamment pu être présenté à MM.

Giraud-Sauveur et Boucher respectivement commercial / chargé d'affaires sur le périmètre MBDA / Nexter et Directeur de projets Nexter. Accompagné d'une démonstration, cet entretien s'est soldé positivement et a permis au POC d'être relayé par la suite auprès des instances susceptibles d'assurer son pilotage futur.

1.3.2 Fruits du partenariat avec l'INSA Centre Val-de-Loire

Ce stage –comme celui d'Alban Chazot– s'effectue dans le cadre d'un partenariat entre SII et l'INSA Centre Val-de-Loire. La relation établie entre ces deux entités découle de la convention "Carré des partenaires" acceptée par les deux parties en 2015. Quentin Ménart, aujourd'hui ingénieur logiciel au sein de l'agence SII Bourges nous a précédés en expérimentant l'année dernière un stage découlant du partenariat.

Cette relation privilégiée mène à retrouver SII en qualité d'acteur ou de sponsor de certains évènements majeurs de la vie étudiante : Nuit de l'Info, Forum des entreprises, conférences techniques. Par ailleurs, elle permet de favoriser l'accueil de stagiaires et d'alternants en provenance de l'institut.



FIGURE 1.3 – Wifibot embarquant un RPLidarA2 et une caméra Theta S

En ce qui nous concerne, cette collaboration a impacté nos stages de manière visible et fructueuse. En effet, elle a permis le prêt d'un robot mobile par M. Benali Abderraouf, enseignant-chercheur en robotique à l'INSA Centre Val-de-Loire. Ce robot qui a constitué la base du projet est présenté sur la figure 1.3. Dans ce cadre, l'INSA Centre Val-de-Loire a également financé pour moitié le matériel nécessaire au projet, à savoir une caméra Theta S à 360 ° (cf. figure 1.3).

Ce partenariat a aussi donné lieu à des réunions régulières de co-pilotage de projet réunissant MM. Hafiane et Bosch. Elles nous ont permis de réfléchir ensemble au cap à donner, au matériel à acquérir ou encore aux techniques à employer. Enfin, nous avons pu évoluer sereinement en qualité de stagiaires, en comptant sur le suivi fréquent de M. Adel Hafiane, tant dans la mise en place du projet, que durant les phases plus tardives de développement.

Chapitre 2

Définition et mise en oeuvre d'un système de SLAM en temps-réel

2.1 De l'expression du besoin à une spécification complète

2.1.1 Vision et cas d'utilisation

Les réalisations effectuées ont été guidées par une phase d'analyse du besoin, menée de concert avec les membres de l'équipe projet. En qualité de tuteur de stage, M. David Daumand a guidé ce projet afin que le système final implique des perspectives commerciales concrètes. Dans cette optique, il a apporté son expertise sur ce que seraient les besoins de clients potentiels, en orientant la démarche vers une application militaire. Il a ainsi défini la dénomination du système SRT2M, permettant à elle seule d'exprimer la fonction, le contexte et le secteur visé par le produit.

La preuve de concept réalisée durant ce stage est un sous-système de SRT2M. Afin d'en exposer clairement le périmètre, la figure 2.1 définit sa nomenclature et en donne les principaux cas d'utilisation.

Ainsi, ce *sous-système* doit permettre d'effectuer une cartographie de l'environnement d'un robot mobile, tout en donnant sa localisation. L'acteur primaire du système pourra visualiser une carte, soit par le biais de données préalablement sauvegardées au sein de ce même système, soit grâce à des données acquises en temps réel par des périphériques dédiés. Cette visualisation implique la représentation d'obstacles statiques dans l'espace exploré, mais aussi la position et la trajectoire du robot en tout temps. Le téléguidage du robot par l'utilisateur doit également être assuré. Un opérateur –typiquement un contributeur du système– pourra jouer sur les paramètres de présentation des résultats, de la mise en réseau des périphériques ou de la définition de ces périphériques.

Les périphériques matériel sont à ce stade réunis en un seul acteur “Périphérique matériel”. Il inclut le LIDAR et le robot qui permettent respectivement d'acquérir les mesures de distances aux obstacles et de déplacer la plateforme. On souligne aussi la présence d'un “Composant logiciel” qui interagit avec le système. Ce composant est vu comme une boîte noire qui fournit en sortie les résultats de détection et de classification d'objets d'intérêt rencontrés par le système. Dans les faits, ces résultats correspondent à la position, la taille et la classe de chaque objet, ce qui nous permettra de les représenter sur la carte générée (cf. fonction *Voir objets détectés()*).

LOCALISATION ET CARTOGRAPHIE DE L'ENVIRONNEMENT D'UN ROBOT MOBILE

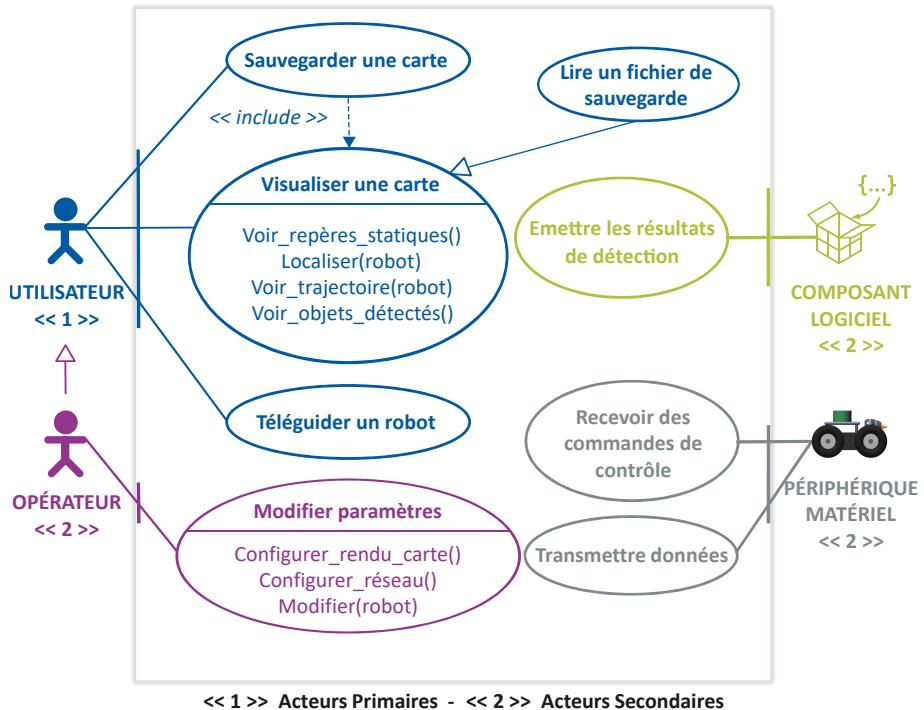


FIGURE 2.1 – Diagramme de cas d'utilisation du système de cartographie et de localisation d'un robot mobile

2.1.2 Formalisation d'exigences

Les réalisations s'articulent autour d'exigences formalisées au sein d'un document de spécification du besoin logiciel (STBL) dont le but et la portée seront décrits dans la section Organisation du travail. La figure 2.2 reprend les déclarations de haut-niveau qui permettent de cerner les principaux axes de l'analyse fonctionnelle. Les exigences exposées s'appliquent à un ou plusieurs composants du système¹.

Chaque exigence est une entrée du tableau, que l'on associe à l'une des catégories suivantes :

- **fonctionnelle** réuni les besoins métiers qui justifient la réalisation du logiciel
- **performance** regroupe les contraintes spécifiques liées au temps d'exécution ou à l'utilisation des ressources
- **design et conception** donne les impératifs en termes de rendu visuel
- **opérationnelle** traite des interactions survenant pendant l'exploitation du système

On attribue une référence unique permettant le suivi unitaire des exigences lors des différentes phases du projet, une description textuelle non ambiguë ainsi que le niveau de criticité de l'exigence : “C” signifie critique, “I” important et “S” souhaitable.

À ce niveau, les exigences ne suggèrent pas d'implémentation ou de choix techniques. Par contre, elles guident l'ensemble de la réalisation, depuis la conception jusqu'aux tests, en passant par l'estimation de la satisfaction du client. Chacune d'entre-elles s'accompagne donc d'un paragraphe servant à la détailler, à en donner les pré-requis et les cas de validation. Par exemple, on précisera que la FNC001 nécessite la possession d'un matériel d'acquisition de mesures spatiales ou, à défaut, de données de test simulant cette acquisition.

Dans ce qui va suivre, nous nous concentrerons sur les exigences qui apportent la valeur

1. Par opposition aux exigences unitaires –allouées à un et un seul composant– abordées au chapitre Bilan et perspectives

Fonctionnelles		
Référence	Description	Valeur
FNC001	Visualisation en 2D de l'environnement du robot (cartographie)	C
FNC002	Localisation du robot sur la carte	C
FNC003	Visualisation de la trajectoire du robot	C
FNC004	Commande des périphériques à distance	C
FNC005	Téléguidage du robot au clavier et joystick virtuel	I
FNC006	Journalisation selon plusieurs niveaux de criticité	I
FNC007	Sauvegarde des données de cartographie et de localisation	S
FNC008	Rejet de données sauvegardées	S
FNC009	Représentation des objets détectés et classifiés sur la carte, à l'échelle	S
FNC010	Centralisation du contrôle de l'ensemble des briques logicielles dans l'IHM	S
FNC011	Modularité des informations réseau pour les périphériques	S
Performance		
PRF001	Rendu visuel en quasi temps réel	C
PRF002	Navigation fluide dans le contexte graphique	C
Design et Conception de l'IHM		
DCC001	IHM ergonomique et intuitive	C
DCC002	Représentation indépendante des éléments cartographiques au regard du flux vidéo	C
DCC003	Présence d'un panneau de contrôle des modules	C
DCC004	Présence de témoins de statut des modules	I
DCC005	Présence d'une console de journalisation dans le panneau de contrôle	I
DCC006	Modification des modules (nom, exécutable, arguments) depuis l'IHM	S
DCC007	Rendus visuels aux 3 ^{ème} et 1 ^{ère} personnes	S
Opérationnelle		
OPE001	Communication sans fil entre le robot et le poste de travail	C
OPE002	Sécurisation des communications avec les éléments distants	S
OPE003	Stabilité des périphériques sur la plateforme mobile	S

FIGURE 2.2 – Exigences de Spécification Technique du Besoin Logiciel

intrinsèque de l'ouvrage, en mettant l'accent sur les niveaux de criticité élevés : à savoir celles qui permettent d'établir et de représenter une cartographie de l'espace et d'y localiser le robot en tout temps. La définition de ces exigences a permis d'entamer une phase de définition des architectures physique et logicielle sereinement, sans perdre de vue l'ensemble des fonctionnalités à assurer et leur priorité respective.

2.1.3 Matériel et Interfaces

Les fonctionnalités principales de l'application dépendent de l'acquisition de mesures spatiales. Ainsi, nous devions nous doter d'une dispositif capable de prendre ces mesures et de les relayer en quasi temps-réel jusqu'à un module de traitement, encore à définir.

Le choix du matériel retenu a fait l'objet d'une veille technologique visant à comparer les différents LIDAR présents sur le marché. Ceux-ci ont pu être classés, outre selon leur prix, selon les critères suivants :

- la technologie employée (1D, 2D, 3D)
- la documentation disponible (notamment en ce qui concerne l'utilisation de la SDK)
- la précision angulaire

- la vitesse de calcul et de transfert des mesures
- la pérennité du produit et/ou de la marque en gage de qualité
- la disponibilité du produit en France, selon des délais serrés

Présenté en **Annexe ? ?**, le résultat de ce comparatif a débouché sur l'acquisition du RPLidar A2 représenté . Celui-ci est équipé d'un seul faisceau laser qui a la particularité de tourner sur 360 ° . Cet équipement est donc adapté à la mesures des distances dans un espace plan, tout en impliquant une dépense acceptable. La possibilité d'équiper le robot de servo-moteurs et d'un LIDAR à faisceau unique fixe moins onéreux a également été explorée, mais n'a pas été retenue au regard de la compléxité de mise en œuvre d'un tel dispositif, du coût cumulé du LIDAR et des servo-moteurs et enfin, des fortes imprécisions potentiellement engendrées lors du mouvement du robot.

Par ailleurs, le robot mis à disposition du projet est un Wifibot Lab v3, muni d'une carte mère Windows Embedded.

Enfin, un nano-ordinateur Raspberry Pi 3 modèle B est utilisé à des fins de communication entre les briques embarquées et une station de travail qui réuni les modules de calcul et l'IHM.

Nous obtenons ainsi l'archctecture physique schématisée sur la figure 2.3, au travers de laquelle sont définis quatre modules principaux, notés M_i .

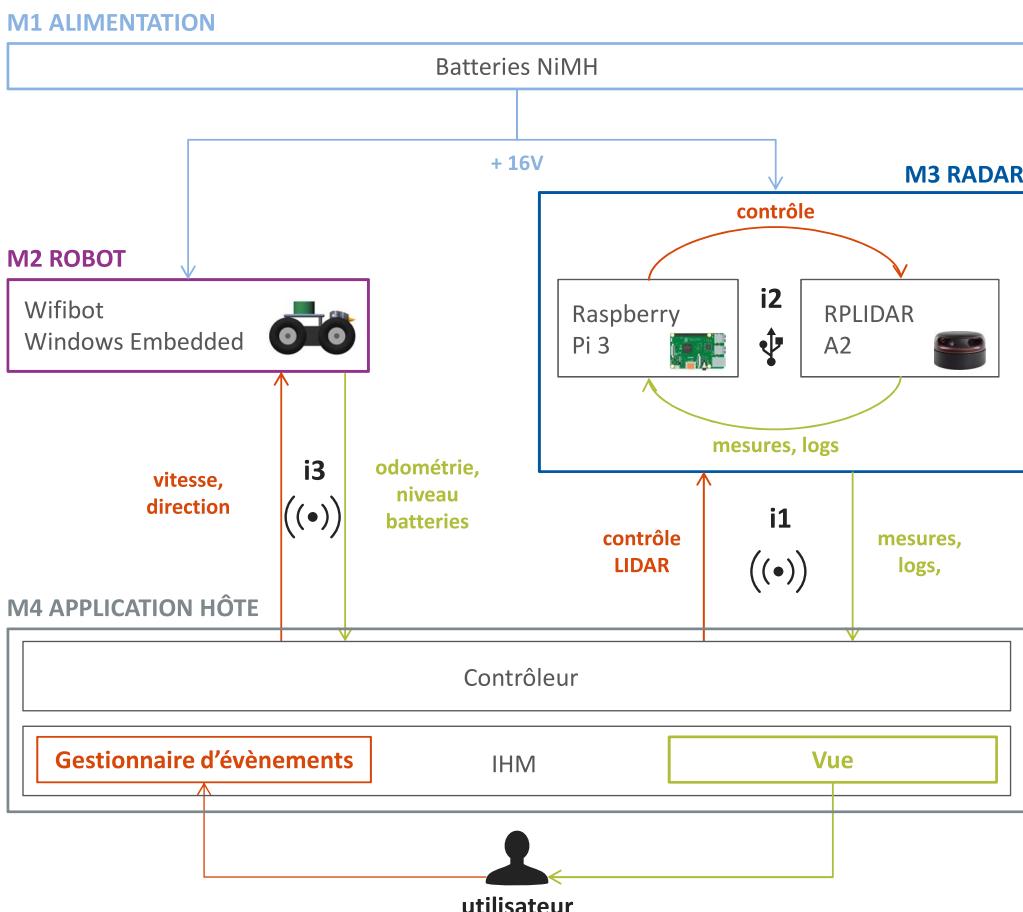


FIGURE 2.3 – Architecture physique du projet

M_1 est dédié à l'alimentation électrique du robot, du LIDAR et du Raspberry Pi. M_2 correspond au Wifibot, M_3 regroupe le LIDAR et le Raspberry Pi et M_4 symbolise l'application hôte sur une station de travail Linux².

2. Ce dernier module présente un intérêt seulement en termes de transmission de données avec les périphériques matériels

Les interactions entre les briques matérielles sont représentées par des flèches d'entrées / sorties ou internes aux modules, représentées de la couleur orange quand il s'agit de requêtes et de la couleur verte pour les réponses afférentes. Par ailleurs, des interfaces de communications inter-modules notées i_i apparaissent en noir et définissent les canaux au travers desquels requêtes et réponses sont transmises.

L'application hôte est responsable de l'émission de commandes de contrôle du LIDAR au travers d'une interface WiFi³ i_1 . Ces commandes transitent par le Raspberry Pi 3 qui les formattera et les transmettra au LIDAR via la liaison USB i_2 . Ce cheminement de données peut être parcouru dans le sens inverse pour remonter les données acquises par le LIDAR à savoir, des mesures de distances spatiales, communiquées tous les 360° , pouvant être perçues comme des nuages de points.

Parallèlement, M_4 communique directement des ordres de vitesse et de direction au Wifibot par le biais de l'interface WiFi i_3 . Ces commandes de direction sont quasi-instantanément suivies de réponses contenant les relevés odométriques des encodeurs du Wifibot, son niveau de batterie et le relevé de capteurs infrarouges présents à l'avant du robot. Les réponses empruntent là encore le chemin inverse par le biais de la même interface.

2.2 Théorie et choix techniques

2.2.1 Cartographie et Localisation Simultanées

Le SLAM, pour Simultaneous Localization And Mapping, est un ensemble de méthodes permettant à un robot mobile d'établir une cartographie de son environnement et de se repérer de manière fiable dans cette même carte.

De manière générale, on peut dégager un schéma autour duquel s'articulent ces techniques, dont la figure 2.4 donne les grandes lignes.

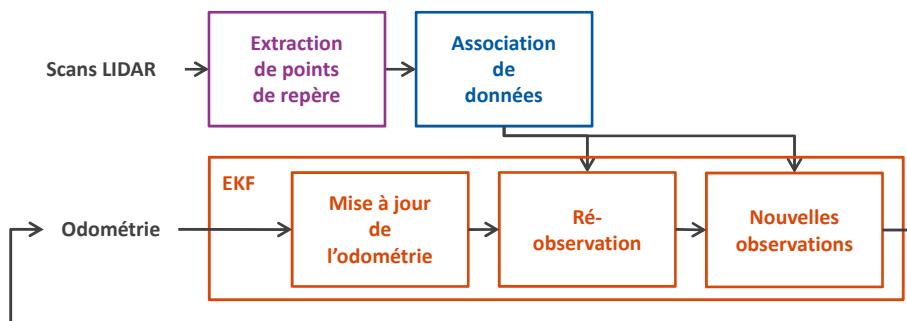


FIGURE 2.4 – Architecture physique du projet

La première étape d'un processus de SLAM consiste à récupérer les données numériques issues d'un laser. Généralement, on peut décrire de telles données en termes de précision, de champs de capture, de longueur du faisceau ou encore de résolution verticale. Le RPLidar A2 dispose d'un champ de 360° sur l'axe horizontal (scan plan), d'une fréquence de rotation typique autour de 600 rpm, d'une portée comprise entre 0.15 et 6m et pour finir, d'une résolution angulaire comprise entre 0.45° et 1.35° .

Les données odométriques (ou odométrie) permettent la prédiction de la position du robot dans le plan. Ce calcul se base généralement sur les relevés de capteurs internes au robot (différentiel de vitesse entre les deux roues motrices, donné par des encodeurs internes). La difficulté réside lorsqu'il s'agit de corrélérer dans le temps et avec précision ces données, avec

3. Le Raspberry Pi 3 est nativement doté d'une carte et d'un émetteur WiFi

les mesures en sortie du laser. Pour pallier ce problème, l'odométrie peut être extrapolée de l'instant précédent dans une démarche prédictive.

Une fois les données externes collectées, il convient de les traiter pour en extraire des points de repères spatiaux. Cette étape, représentée en violet sur la figure 2.4, consiste à caractériser les points de repère pertinents. Ils sont ensuite gardés en mémoire pour établir la cartographie de l'espace et également en vue d'être discriminés ou assimilés ultérieurement. Plusieurs algorithmes, tels que Spike ou RANSAC, proposent ces fonctionnalités avec des approches diverses qu'il convient d'adapter à une application donnée[5].

L'étape suivante, représentée en bleu sur la figure 2.4, vise à associer les données collectées, à savoir reconnaître les points de repère qui arrivent dans le champs d'observation du robot à plusieurs reprises. Le problème de l'association de données vise à prendre en compte du mieux possible les risques de confusion ou de non reconnaissance de points de repères au travers d'une politique de validation qui encadre la fusion de données. À cet effet, on peut appliquer l'algorithme KNN implémentant des mesures de distances diverses comme la distance Euclidienne ou la distance de Mahalanobis.

Enfin, le filtre de Kalman étendu est utilisé pour estimer un vecteur d'état du robot (à savoir sa position et son orientation). Dès lors que les briques d'extraction de points de repère et d'association de données sont mises en place, un processus de SLAM peut être considéré en trois étapes, présentées en orange sur la figure 2.4. Le premier point consiste simplement à ajouter les relevés odométriques du robot à l'ancien vecteur d'état. Le deuxième point apporte une correction au vecteur d'état et à l'état du système de façon globale. En effet, au vu de l'état courant, on peut estimer la position des points de repère précédemment sélectionnés. Si un décalage spatial a lieu, on l'appellera innovation, c'est-à-dire la différence entre la position estimée du robot à l'étape 1 et sa position basée sur la perception de son environnement. De plus, cette étape nous permettra d'affiner la confiance accordée dans notre connaissance de l'environnement. Une innovation faible va entraîner une confiance accrue quant aux positions des points de repère connus et visible, tandis qu'une innovation forte aura l'effet inverse. Le troisième point permet d'ajouter à notre système les points de repère détectés ayant été discriminés par la politique d'association mise en place.

La définition et l'implémentation d'un algorithme de SLAM est un processus fastidieux qui repose sur des fondements théoriques complexes. La diversité de ces algorithmes[6] dénote l'ampleur de la tâche qui consiste à les recenser, à en comprendre tous les rouages pour proposer des améliorations et finalement passer à l'implémentation. Ainsi il n'était pas envisageable de développer un algorithme de SLAM dédié au projet. Par contre, l'utilisation de tels algorithmes paraissait incontournable au regard des exigences du projet. Un état de l'art sur le sujet a donc été mené, permettant de choisir l'algorithme à intégrer, présenté à la section 2.2.3.

2.2.2 Le méta-système d'exploitation ROS



FIGURE 2.5 – Logo du méta-système d'exploitation ROS (issu du kit de presse ROS[21])

Robot Operating System, ou ROS, est un middleware pour systèmes Unix destiné à la réalisation de projets robotiques écrits en C++ ou en Python. Il offre des fonctionnalités dédiées à la robotique –notamment par le biais de bibliothèques– mais aussi des moyens de communication et des outils facilitant le développement et le déploiement de tels systèmes. Il a ainsi été rapidement étudié puis adopté pour ce projet sous sa version Kinetic. Notre utilisation de ROS couvre les briques logicielles embarquées sur le Raspberry Pi et des

briques présentes localement sur le poste de travail de l'utilisateur final. Son fonctionnement passe par la création d'un réseau de noeuds[8], c'est-à-dire des exécutables dans l'écosystème ROS, organisés autour d'un noeud central appelé ROS Master[7]. Cette entité enregistre les noeuds, leur fourni un nom et instancie un annuaire qui leur permet d'échanger des données. Il s'appuie en partie sur le protocole XML-RPC.

Les noeuds disposent de plusieurs moyens de communication, disponibles à travers une librairie cliente fournie par ROS : les topics[9] auxquels il est possible de s'abonner, les services[10] qui fonctionnent sur le modèle requête / réponse, et des valeurs stockées par un serveur de paramètres géré par le ROS master et qui peuvent être récupérés par les noeuds durant leur exécution. On notera que, dans le cas des topics, le format des données transmises correspond généralement à des types de messages définis par ROS dans ses packages internes⁴. Par exemple, le type de données `sensor_msgs/LaserScan` dispose d'un fichier de définition du même nom, portant l'extension `.msg` et d'un fichier d'en-tête qui permet la publication et l'abonnement aux topics de ce type. **À titre d'exemple, le fichier `sensor_msgs/LaserScan.msg` est présenté en annexe X.**

Le framework s'accompagne de nombreux utilitaires en ligne de commande qui permettent à l'utilisateur de configurer le réseau, de le sonder et de créer des noeuds à la volée pour diverses utilisations.

ROS dispose également d'un outil de gestion du système de fichiers et de chaîne de compilation nommé Catkin. Ce dernier permet la mise en place d'un format d'arborescence facilitant la production et la gestion de packages. Ces packages sont définis comme des agglomérats logiques de noeuds et possèdent une structure spécifique au sein de l'environnement de travail Catkin, tel que représenté ci-dessous.

```
catkin_workspace/ ..... racine de l'environnement catkin
  └── src/ ..... emplacement des codes sources
      ├── CMakeLists.txt ... invoqué par CMake lors de la configuration de l'environnement
      ├── package_1/
      │   ├── CMakeLists.txt ... invoqué par CMake lors de la compilation de package_1
      │   ├── package.xml ..... manifeste du package_1
      │   ├── default.launch ..... fichier de lancement des noeuds du paquet
      │   └── src/ .. contient les sources du paquet, implémentant un ou plusieurs noeuds
      ├── package_2/
      │   ├── CMakeLists.txt
      │   ├── package.xml
      │   └── src/
```

Le fichier `package.xml`[11] est le manifeste des paquets. Placé à la racine de chacun d'entre-eux, il en donne la définition (nom, version, auteur, license) ainsi que les dépendances. Les fichiers `CMakeLists.txt`[12] sont les entrées du système de build CMake, invoqués dans la chaîne de compilation de catkin. C'est ici que l'on définit les noms et fichiers sources associés à chaque noeud du paquet, les sources étant généralement stockées sous le répertoire `catkin_workspace/src/<package_name>/src/`. **Les annexes X et Y donnent des exemples correspondant à ces deux fichiers.**

Les deux éléments précédents sont indispensables à la création de paquets dans l'environnement catkin. Parallèlement, on relève la présence d'un fichier portant l'extension `.launch`, appelé launchfile. Ce fichier optionnel permet d'exécuter, via un utilitaire en ligne de commandes, plusieurs noeuds en une seule fois. Pour parser et exécuter le launchfile présent dans notre exemple on saisira simplement :

4. Lors d'une installation standard, ces paquets se situent sous `/opt/ros/<ROS-version>/share/` pour les messages et `/opt/ros/<ROS-version>/include/` pour les fichiers d'en-tête

```
1 | > roslaunch package_1 default.launch
```

Les briques logicielles implémentées avec ROS présentent intrinsèquement un fort potentiel de modularité et de maintenabilité. En effet, chaque package et chaque noeud observent des cycles de vie indépendants et n'interagissent les uns avec les autres qu'en cas d'échange d'informations utiles. Cet aspect modulaire, sur lequel nous reviendrons, a fortement participé à l'adoption de ROS et de ses outils. De plus, ROS est un outil entretenu par une large communauté de chercheurs et développeurs qui alimentent fréquemment l'écosystème par de nouveaux paquets, généralement mis à l'épreuve lors de compétitions robotiques internationales. Ainsi, la problématique de SLAM est largement couverte par ROS, tant et si bien que les librairies et outils publics dédiés au SLAM se retrouvent presque exclusivement sous la forme de paquets ROS.

2.2.3 Hector SLAM

Dans le cadre de ce projet un état de l'art a été entrepris, visant à recenser les principaux algorithmes de SLAM intégrables à notre système logiciel. Les algorithmes majeurs du domaine ont donc été étudiés et qualifiés notamment au travers des applications qu'ils visent, de leur principe de fonctionnement, des possibles implémentations qu'elles connaissent et de leurs limites. Suite à cette étude, le projet Hector SLAM a été retenu. Notons qu'il profite d'une notoriété acquise depuis plusieurs années, lui permettant d'être aujourd'hui encore largement actif et reconnu[14][15].

Hector SLAM est un métapackage⁵ ROS[16] qui inclut nativement trois packages principaux : `hector_mapping`, `hector_geotiff` et `hector_trajectory_server`. Nous ne reviendrons pas sur `hector_geotiff` responsable de l'enregistrement des données au format GeoTIFF qui a été écarté du système final. `hector_mapping` est le moteur de SLAM intégré par Hector. Il a la particularité de ne pas requérir de données odométriques pour fonctionner. Cela sous-entend que l'on peut disposer de résultats de SLAM en tenant simplement le LIDAR dans ses mains. Ce point a été déterminant dans la sélection d'Hector SLAM puisque d'une part l'acquisition du robot s'est faite tardivement, et d'autre part, ce dernier a été fourni sans documentation permettant de s'assurer de la précision des encodeurs. Nous avons donc préféré un système de SLAM qui favorise les données issues du LIDAR fraîchement acquis. Par ailleurs, Hector SLAM s'adapte à une utilisation sur des drones par le biais de données IMU. Dans l'optique d'assurer une continuité au projet en l'orientant vers le secteur de la défense, la possibilité d'adapter le système à des dispositifs aériens semblait particulièrement intéressante. Enfin, le noeud `/hector_trajectory_server` est utilisé pour traiter la succession de positions de la plateforme mobile depuis le début de l'acquisition.

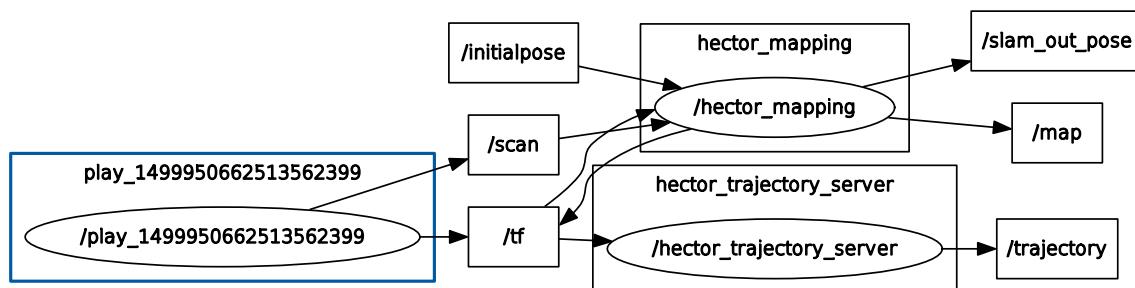


FIGURE 2.6 – Nœuds et topics d'intérêt et actifs lors de l'exécution d'Hector SLAM

La figure 2.6 expose les nœuds et topics actifs lors d'un l'utilisation d'Hector SLAM à partir

5. Un métapackage est simplement pourvu d'un manifeste qui permet de référencer des paquets regroupés logiquement mais faiblement couplés d'un point de vue fonctionnel

d'un jeu de données pré-enregistrées dans un fichier bagfile. Les nœuds actifs sont présentés au sein d'ellipses, les topics au sein de rectangles sous la forme `/<topic-name>` et les noms des packages auxquels les nœuds appartiennent figurent au dessus de chacun d'entre-eux. Enfin, les flèches signalisent quels sont les nœuds responsables de la publication des topics, et quels sont les nœuds qui s'y sont inscrits. Le package généré par la lecture du bagfile est représenté en bleu.

On s'intéresse particulièrement aux topics en sortie d'Hector SLAM. `/slam_out_pose` donne la localisation du robot en terme de position et d'orientation, `/trajectory` définit la trajectoire parcourue par le robot et `/map` donne la représentation de son environnement. Ce dernier topic correspond à une grille d'occupation, à savoir une discréétisation de l'espace sous forme de grille où chaque case porte la probabilité qu'elle soit un obstacle. Dans la pratique la carte est de dimensions 2048×2048 avec une résolution de $0.05m$. Les résultats issus de l'approche probabiliste s'expriment selon trois valeurs : 0 pour une case vide, 1 pour une case occupée et -1 pour une case inconnue. Chaque nouvelle acquisition attestant l'occupation d'une case augmente sa probabilité, et inversement lorsqu'une acquisition démontre que la case est libre. Lorsque la probabilité portée par une case dépasse un seuil interne à Hector SLAM, la case sera considérée comme occupée et passera à la valeur 1^6

L'intégration d'Hector SLAM au projet s'est déroulée en plusieurs temps. Nous avons d'abord pu prendre l'outil en main avant l'achat du LIDAR grâce à l'utilisation de données de test contenues par des bagfile. Durant cette étape, les résultats pouvaient être visualisés grâce à RViz, un outil graphique inclu dans ROS⁷. La figure 2.7 illustre le résultat obtenu pour un jeu de test issu de la RobotCup German Open 2011. Dans un deuxième temps, il a fallu adapter Hector SLAM pour recevoir en entrée les données du LIDAR, ce qui revient principalement créer des fichiers de lancement (*launchfile*) adaptés au périphériques d'acquisition. Dans notre cas nous avons créé deux fichiers distincts : l'un pour le mode d'exécution standard et l'autre pour un mode "debug". Nous avons également mis en place des éléments qui définissent les transformations spatiales entre les différents périphériques matériels et qui seront évoqués dans la partie suivante.

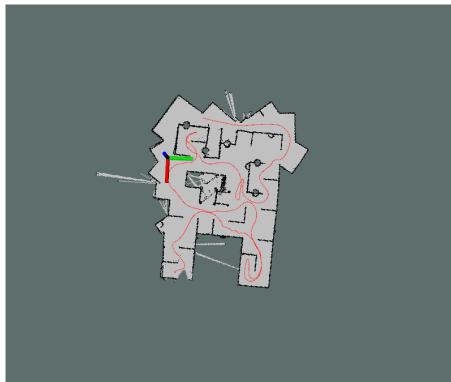


FIGURE 2.7 – Visualisation avec RViz d'un jeu de données pré-enregistrées

Ces différents composants et la manière dont ils ont été réalisés font l'objet de la partie 2.3.1.

6. Ce seuil est fixé à 0.5

7. Pour la distribution Kinetic de ROS, RViz est présent dans les paquets d'installation dits "Desktop"[17].

2.3 Réalisations et architecture logicielles

2.3.1 Le réseau ROS complet

Hector SLAM a été intégré à un réseau plus large prenant en compte les spécificités de notre application. Il était entendu que l'IHM ne serait pas inclue dans ce réseau et serait développée avec Qt Creator (voir partie 2.3.2).

La définition du réseau doit alors s'inscrire comme une solution aux problématiques suivantes :

- le contrôle du LIDAR ainsi que la lecture et l'émission des nuages de points calculés
- l'intégration d'Hector SLAM
- la communication des résultats de cartographie et de localisation vers l'IHM

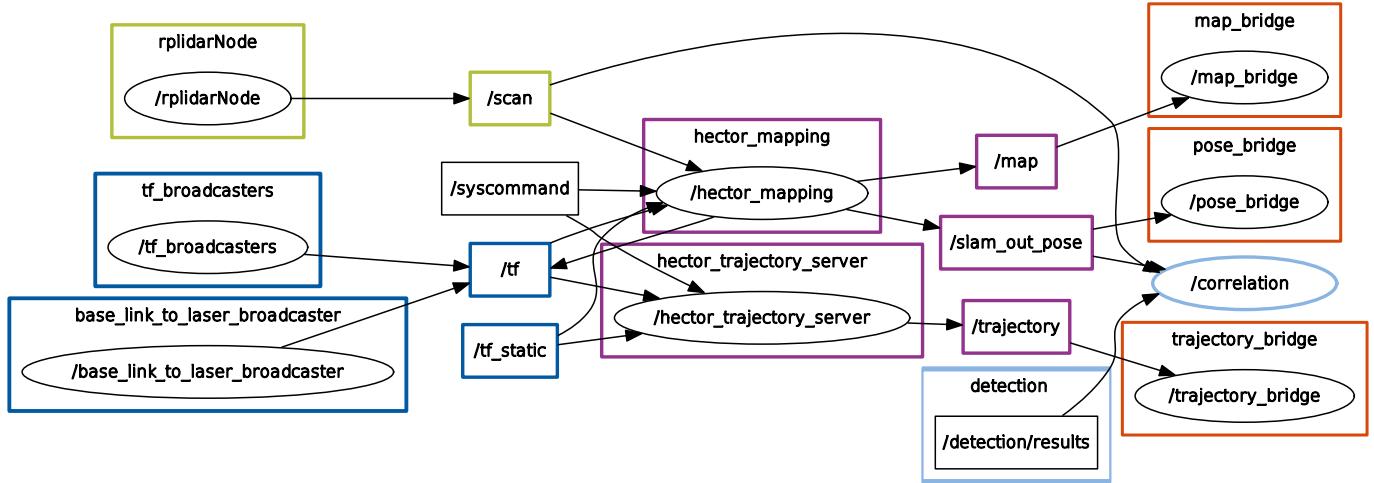


FIGURE 2.8 – Réseau ROS intégré au système

Le premier point, représenté en vert sur la figure 2.8, est assuré par un noeud embarqué sur le Raspberry Pi régissant la communication avec le LIDAR. Il s'interface avec plusieurs fichiers d'en-tête qui constituent la SDK du LIDAR. Cet élément du réseau ROS est fourni par le constructeur du LIDAR dans son kit de développement. Sa mise en œuvre a demandé de comprendre les principales méthodes de la SDK et la portée fonctionnelle de chaque fichier d'en-tête. Par ailleurs, outre l'installation de ROS sur Raspberry, la mise en place du matériel a nécessité l'installation d'un driver permettant la conversion des données depuis un l'UART vers l'USB, ainsi que la définition de règles d'utilisation du port série sur lequel est raccordé le LIDAR. En sortie, ce noeud publie le topic `/scan`, correspondant au format de messages `sensor_msgs/LaserScan` précédemment évoqué.

L'intégration d'Hector SLAM demande la définition de plusieurs systèmes de coordonnées correspondant au divers composants du système. Les transformations spatiales entre ces systèmes étant également requises par Hector SLAM, elles sont publiées au travers d'un package appelé `tf`[18][19]. Les noeuds et topics impliqués dans ce processus sont représentés en bleu foncé sur la figure 2.8. `tf` distingue deux types de transformations : les transformations statiques et les transformations dynamiques. Dans le premier cas, il suffit de définir l'identifiant d'un repère de référence (`frame_id`) et d'un repère fils (`child_frame_id`) ainsi que la transformation qui subit ce dernier, en s'assurant qu'il ne sera jamais amené à changer. Dans le second cas, il faut généralement passer par la création d'un noeud publiant la valeur de ce décallage régulièrement durant l'exécution. Pour mieux saisir les enjeux soulevés par l'intégration d'Hector SLAM, la figure 2.9 donne les systèmes de coordonnées définis, ainsi que les noeuds implémentés (les *broadcaster*) responsables de la publication des transformations.

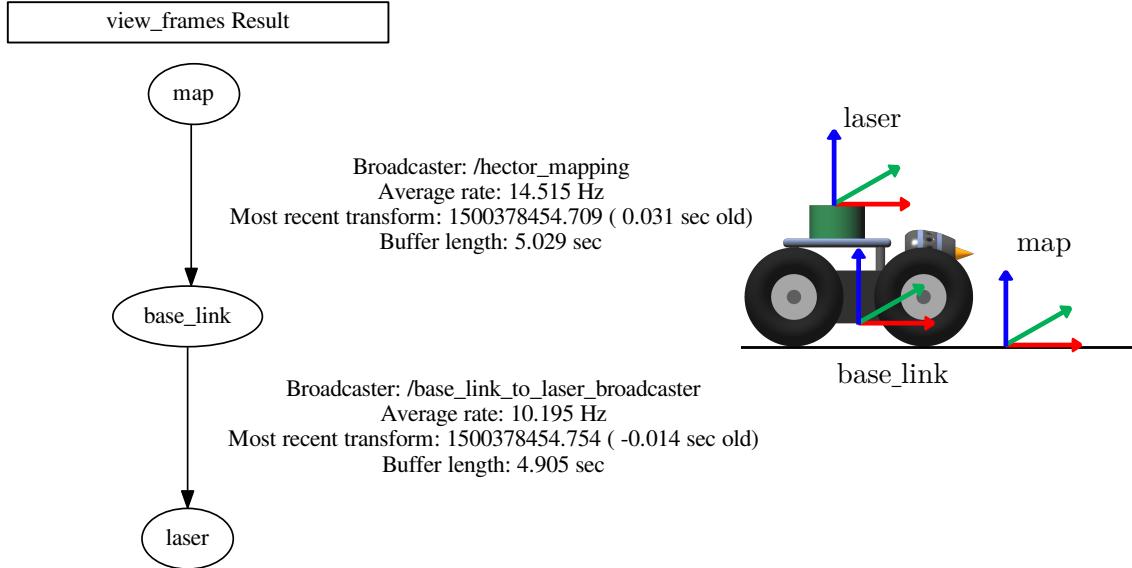


FIGURE 2.9 – Systèmes de coordonnées et transformations au sein du système

Le système *map* est un repère fixe, ancré au sol, qui sert de référence au positionnement du robot sur le long terme[20]. *base_link* est un repère mobile par rapport à *map* attaché sous la plateforme mobile en son centre. Une première transformation entre *map* et *base_link* est requise par Hector SLAM. Elle est calculée par le noeud */tf_broadcaster* (voir figure 2.8) à partir des relevés odométriques fournis par le robot. Une fois intégrée par Hector SLAM, cette transformation est corrigée puis ré-émise par */hector_mapping* : c'est le topic */tf* de la figure 2.8. Enfin, le noeud */base_link_to_laser_broadcaster* publie régulièrement la transformation entre *base_link* et *laser*, c'est-à-dire le différentiel de position entre la base du robot et le LIDAR. Celui-ci n'étant jamais amené à changer nous sommes dans le cas d'une transformation statique définie de la manière suivante dans le launchfile d'Hector SLAM en mode d'exécution standard⁸ :

```

1 <!-- Laser is set 25cm above robot base. Node arguments must be :
2 dx dy dz yaw pitch roll frame_id child_frame_id publishing_frequency_in_ms
3 -->
4
5 <node pkg="tf" type="static_transform_publisher"
6   name="base_link_to_laser_broadcaster"
7   args="0 0 0.25 0 0 0 base_link laser 100" />

```

Nous obtenons ainsi un moteur de SLAM fonctionnel à partir duquel nous devons construire une IHM. Pour ce faire, nous avons défini trois packages appelés *bridges* représentés en orange sur la figure 2.8. Chacun d'entre-eux s'abonne à l'un des topics en sortie d'Hector SLAM. Ensuite, les données utiles sont formatées par appel à une classe statique nommée **Serializer** et communiquées à l'IHM par le biais d'un client TCP implémenté avec l'API socket. La sérialisation répond à un schéma générique composé d'une en-tête (*header*) et d'une charge utile (*payload*). Il est représenté sur la figure 2.10.



FIGURE 2.10 – Définition du paquet générique de transmission de données à l'IHM

Les éléments contenus par le header sont **tID**, un identifiant donnant le type de la donnée transmise, **pSize**, la taille du payload et un **padding** à 0 qui vise à ce qu'il atteigne

8. En mode “debug” cette transformation est non avenue puisqu’elle est déjà comprise dans les données à rejouer, à savoir le bagfile

20 octets. La formation du **payload** dépend quant à elle des données à transmettre. La principale difficulté réside en la transmission de la grille d'occupation contenue par le topic **/map**. Celle-ci contenant plus de 4000000 de valeurs entières, nous nous sommes focalisé sur l'émission de mises à jour locales à chaque acquisition, plutôt que globales⁹. Ainsi, le **payload** se construit de la manière suivante afin de désencombrer la bande passante et réduire la taille de la structure de données qui l'encapsule :

```

1 // Fill a buffer with occupancy grid local updates
2 int Serializer::serializeMap( const nav_msgs::OccupancyGrid& map,
3                               const nav_msgs::OccupancyGrid& old_map,
4                               std::string *buffer ) {
5
6     std::string header, payload;
7
8     // Check data consistency
9     if( map.info.width != old_map.info.width ||
10        map.info.height != old_map.info.height ) {
11         std::cout << "Serialization error : Map and old_map with different
12             dimensions" << std::endl;
13         return -1;
14     }
15
16     // Fill payload with updated cases' value and index
17     for( int i = 0; i < map.info.width * map.info.height; ++i ) {
18         if( map.data[i] != old_map.data[i] )
19             payload.append( std::to_string(i) + ","
20                             + std::to_string( map.data[i] ) + "," );
21     }
22     payload.append( "EOP" );
23
24     // Fill header, return error if its size exceeds 20 bytes
25     header.append( MAP_ID );
26     header.append( "," + std::to_string( payload.length() ) + "," );
27     if( Serializer::addZeroPadding( &header ) < 0 ) return -1;
28
29     // Fill buffer with header and payload
30     buffer->append( header );
31     buffer->append( payload );
32     return 0;
33 }
```

Enfin, la figure 2.8 fait également état du nœud **/correlation** abonné aux résultats de détection d'objets d'intérêt et au topic **/scan**. Il est responsable du croisement de ces données afin de déterminer, quand cela est possible, les coordonnées d'un objet détecté à partir du flux vidéo. La technique employée ici consiste à trouver pour chaque résultat de détection, la position du robot à l'instant de la détection ainsi que ou les faisceaux du LIDAR susceptibles d'avoir atteint l'objet. Lorsque ces-derniers existent¹⁰ nous devons

9. Il en va de même pour la trajectoire où, dans un même souci d'optimisation, nous effectuons un différentiel entre les valeurs précédemment acquises et la nouvelle acquisition

10. Le plan parcouru par les faisceaux du LIDAR doit être compris entre les extrémités haute et basse de l'objet.

transformer la distance et l'angle associés dans le système de coordonnées de la carte.

Algorithme 1 : Algorithme de calcul des résultats de corrélation

Entrées : R les derniers résultats de détection reçus sous la forme : dimensions plane de l'objet (w, h) et ses coordonnées sphériques (ϕ, θ)
 t un entier non signé, estampille de R

Données : S l'ensemble des scans n'ayant jamais été corrélés
 P l'ensemble de positions du robot n'ayant jamais été corrélées
 $scan$ un timestamp et un ensemble $coord$ de paires de coordonnées polaires (θ, r)
 $pose$ un timestamp, une position $p = (x, y, z)$ et une orientation $q = (pitch, roll, yaw)$
 $angle$ l'angle auquel se trouve l'objet dans le système de coord. du scan
 $matched$ couple (θ, r) caractérisant la position de l'objet détecté

début

```

si  $S = \emptyset \vee P = \emptyset$  alors
|   retourner -1
sinon
|   /* find temporally nearest scan and pose with respect to  $t$  */
|    $scan \leftarrow findNearestScan(t)$ 
|    $pose \leftarrow findNearestPose(t)$ 
|   si  $scan = 0 \vee pose = 0$  alors
|   |   retourner -1
|   sinon
|   |   pour  $r \in R$  faire
|   |   |   /* check if current object  $r$  crosses LIDAR laser */
|   |   |   si  $(r.\theta - \frac{r.h}{2} < \frac{\pi}{2}) \wedge (r.\theta + \frac{r.h}{2} > \frac{\pi}{2})$  alors
|   |   |   |    $angle \leftarrow r.\phi$ 
|   |   |   pour  $s \in scan.coord$  faire
|   |   |   |   si  $s.\theta > angle$  alors
|   |   |   |   |   break
|   |   |   |   |    $matched \leftarrow s$ 
|   |   |   |   /* add object position and dimension in result structure */
|   |   |   |   /*
|   |   |   si  $matched.r \neq \infty$  alors
|   |   |   |    $r.x \leftarrow pose.p.x + matched.r \times \cos( matched.\theta + pose.yaw )$ 
|   |   |   |    $r.y \leftarrow pose.p.y + matched.r \times \sin( matched.\theta + pose.yaw )$ 
|   |   |   |    $r.z \leftarrow LIDAR\_LENGTH + matched.r \times \tan( \frac{\pi}{2} - r.\theta )$ 
|   |   |   |    $r.w \leftarrow 0.5 \times 2 \times matched.r \times \tan( \frac{r.w}{2} )$ 
|   |   |   |    $r.h \leftarrow 0.5 \times 2 \times matched.r \times \tan( \frac{r.h}{2} )$ 
|   |   |   
```

2.3.2 Interface Homme-Machine avec Qt Creator

L'Interface Homme-Machine de l'application a été implémentée en C++ avec l'API Qt, facilitant notamment la réalisation de l'interface graphique au moyen de l'environnement de développement Qt Creator. Dans l'API Qt, les éléments graphiques sont appelés *widgets* et dérivent de classes définies dans les bibliothèques internes, telles que `QWidget`, `QOpenGLWidget` ou encore `QDialog` pour ne citer que celles qui ont été utiles au projet. Qt fourni également un mécanisme de communication inter-classes thread-safe et type-safe,

par le biais d'éléments appelés signaux et slots¹¹. Ceux-ci ont la particularité d'occuper la démarche de liaison entre les parties communicantes et ne nécessitent pas de classe dédiée à cet effet. Ils permettent de mettre en place simplement des connexions *one-to-many*, *many-to-one* ou encore *many-to-many*. Un signal est une signature de fonction membre d'une classe, qui pourra être émis par ses instances. Les slots sont quant à eux des fonctions membres désignées par la macro Q_SLOTS. Ce mécanisme est illustré dans l'exemple suivant, tiré du code source du projet à mettre en annexe plutôt....

```

1 // sensorDataAvailable signal connection to setWifibotInfo slot
2 QObject::connect( wc_, &WifibotClient::sensorDataAvailable,
3                   w_, &MainWindow::setWifibotInfo );
4
5 // signal emission in WifibotClient class
6 emit sensorDataAvailable(robotSensors);
7
8 // data computing in MainWindow callback
9 void MainWindow::setWifibotInfo(SensorData sd)
10{
11    // dispatch info to sensor widgets through other signals and slots
12    // mechanisms
13    emit setIRLeftValue(sd.IRLeft);
14    emit setIRRRightValue(sd.IRRRight);
15    emit setBatteryValue(sd.batVoltage);
16    emit setOdomLeftValue(sd.odometryLeft);
17    emit setOdomRightValue(sd.odometryRight);
18    emit setSpeedLeftValue(sd.speedFrontLeft);
19    emit setSpeedRightValue(sd.speedFrontRight);
20}
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
678
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
746
746
747
748
748
749
749
750
751
752
753
754
755
756
756
757
758
758
759
759
760
761
762
763
764
765
766
766
767
768
768
769
769
770
771
772
773
774
775
776
776
777
778
778
779
779
780
781
782
783
784
785
785
786
787
787
788
788
789
789
790
791
792
793
794
795
795
796
796
797
797
798
798
799
799
800
801
802
803
804
805
805
806
806
807
807
808
808
809
809
810
811
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
```

La figure 2.12 expose quant à elle les classes et packages développés avec Qt pour répondre aux exigences fonctionnelles et d'interface précédemment formulées. Sont également représentés les moyens de communications mis en place entre l'IHM et les autres éléments constituant le projet : le réseau ROS et la couche d'exploitation du Wifibot.

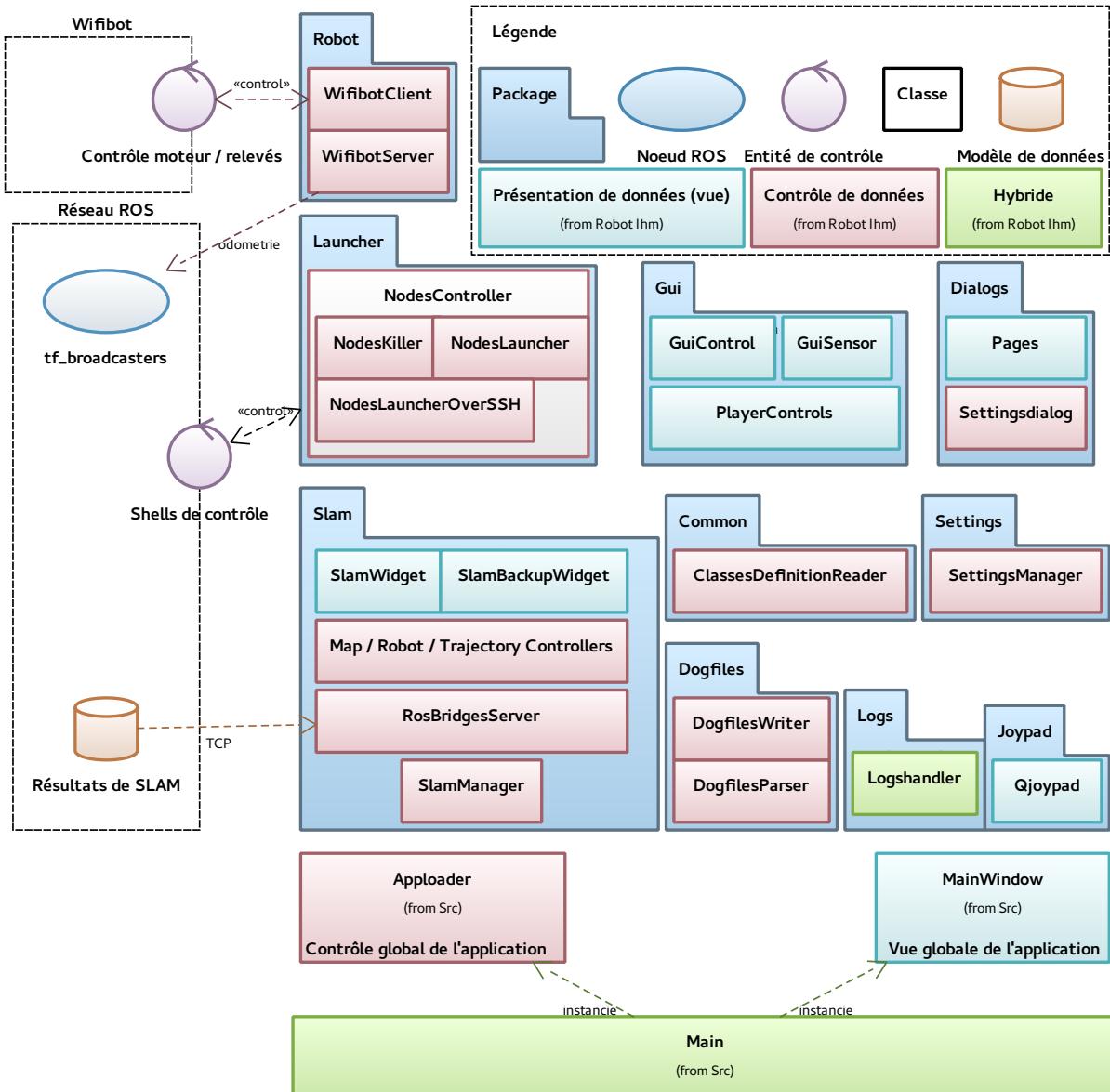


FIGURE 2.12 – Modélisation logicielle de l'Interface Homme-Machine

Le point d'entrée de l'application est la classe `main.cpp` qui instancie la fenêtre principale de l'application `MainWindow`[22] et une classe `Apploader` spécifique à notre architecture. Cette dernière maintient et met à jour la machine à états présentée en 2.11 gérant ainsi le cycle de vie de toutes les autres instances.

Le package *SLAM* traite les données en sortie du réseau ROS. La classe `RosBridgesServer` instancie un serveur TCP consacré à la communication avec les noeuds de bridges décrits dans la partie 2.3.1. Ces données sont ensuite transmises à des contrôleurs dédiés à chaque type de données (carte, position ou trajectoire) qui assurent le parsing des paquets reçus puis émettent des signaux aux *wIDGETS* de rendu. Dans notre cas, deux *wIDGETS* sont susceptibles de rendre ces données : `SlamWidget` ou `SlamBackupWidget` respectivement associés au mode *Réception* ou *Backup*. Ces *wIDGETS* héritent de la classe Qt `QOpenGLWidget` fournissant les fonctionnalités de rendu graphique d'OpenGL dans un contexte Qt. La figure 2.13

donne un aperçu de l'IHM en mode *Réception* afin de présenter les différents éléments de l'application. La fidélité des résultats sera quant à elle discutée dans la partie 3.

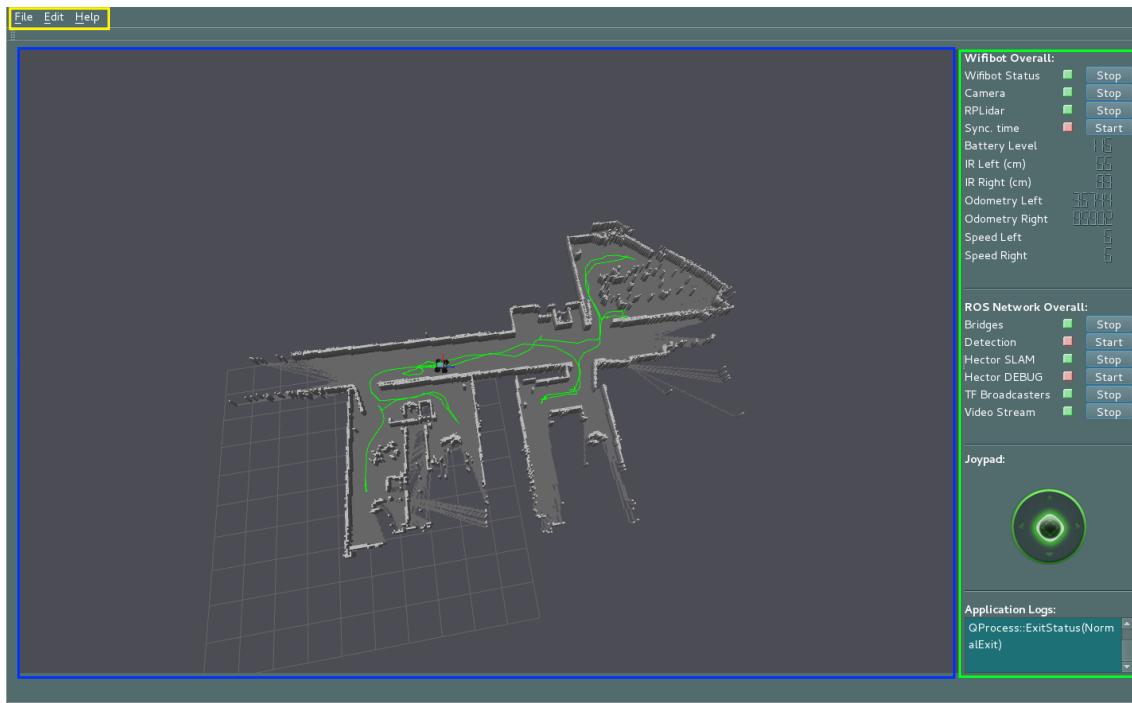


FIGURE 2.13 – Rendu visuel de l'IHM en mode Réception

Nous pouvons visualiser la fenêtre principale (`MainWindow`) complétée d'instances de *widgets* responsables du rendu d'éléments d'intérêt :

- **un widget de rendu OpenGL**, en bleu, fourni une représentation de la carte, du robot par le biais d'un modèle 3D et de sa trajectoire (position et orientation en tout temps)
- **un panneau de contrôle**, en vert, est découpé en quatre sections : le Wifibot, le réseau ROS, un joystic virtuel et une console de logs
- **une barre de menu**, en jaune, permet la sauvegarde de jeux d'acquisition, le chargement d'une sauvegarde pour la rejouer, l'édition de paramètres de l'application ou l'affichage du manuel utilisateur

La figure 2.14 expose le rendu global en mode *Backup*. Celui-ci se distingue particulièrement du mode d'acquisition par la présence –au sein du panneau de contrôle– d'un *player* offrant les fonctionnalités d'un lecteur multimédia classique.

À cet effet, un format d'enregistrement des données propre à l'application a été défini en XML. Ce format, appelé *DOG*¹² est géré par les classes du package `DogFiles` : `DogFilesWriter` pour l'écriture et `DogFilesParser` pour la lecture. Des techniques de compressions ont été mises en œuvre pour les données de taille critique devant être sauvegardées. Ainsi, les données décrivant la carte sont traitées par un algorithme d'encodage par répétition (Run Length Encoding) tandis que toutes les composantes des vecteurs de trajectoire du robot sont gérées par la méthode suivante :

```

1 // Multiply entry by 100, truncate it and returns its hexadecimal value.
// Negative number are concatenated with '-' char and encoded like
// positive values.
2 QString SlamWidget::floatToHex(float v)
3 {
4     return (int)(v * 100) >= 0.0f
5         ? QString::number( (int)(v * 100), 16 )
6         : QString('-' + QString::number( (int)-(v * 100), 16 ));
```

12. Pour Detection and Occupancy Grid

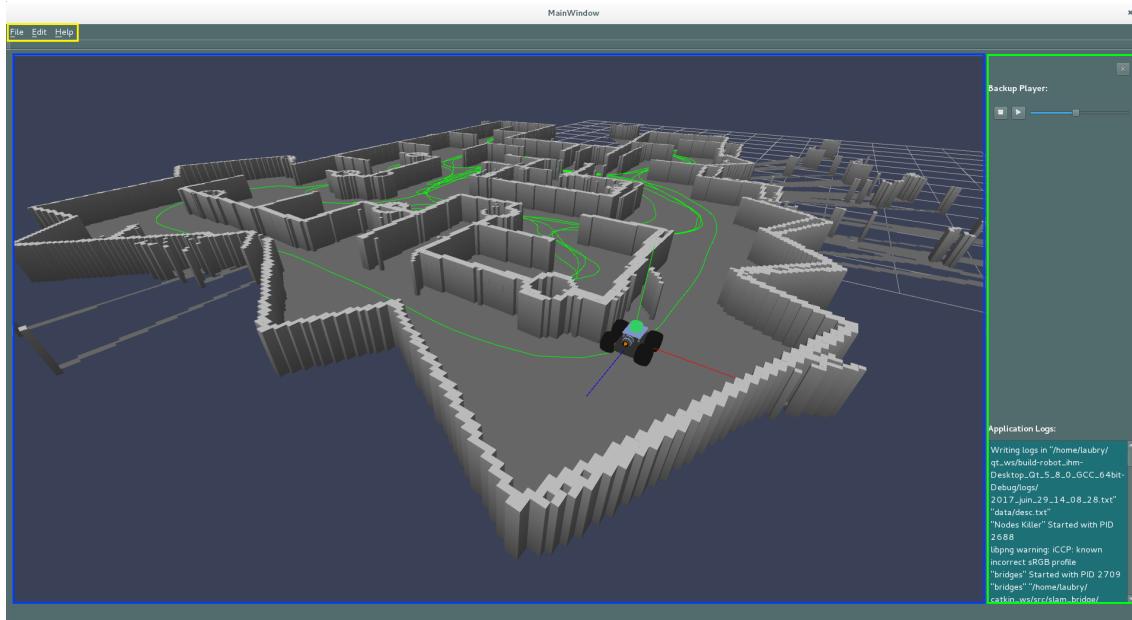


FIGURE 2.14 – Rendu visuel de l’IHM en mode jeu

7 | }

Un algorithme similaire a été appliqué pour l’ensemble des orientations du robot. Ces techniques ont permis de réduire significativement les données à sauvegardées à l’aide d’une compression sans perte dans un premier cas et une perte de l’ordre du centième de case (soit 0.0005m) dans un second cas.

2.3.3 Éléments de modularité mis en place

La réalisation de l’application a d’emblée été associée à un objectif de modularité afin d’appréhender au mieux la perspective de stages futurs et / ou d’une possible adaptation industrielle du projet. L’utilisation de ROS allant dans ce sens, il paraissait intéressant d’appliquer cet état d’esprit à l’IHM, tant dans l’architecture du logiciel que dans les fonctionnalités proposées à l’utilisateur final.

Le premier point se retrouve principalement dans l’implémentation de l’**Uploader** qui gère le cycle de vie des instances d’une manière communément compréhensible : création, mise-à-jour puis destruction. D’autre part, l’interface graphique et les éléments de communication avec les noeuds peuvent eux-même être modifiés à convenance par le biais de l’IHM. Nous exposons ici les principes qui sous-tendent cette démarche, nous amenant à balayer les points suivants :

- le paramétrage modulaire du réseau ROS
- les éléments graphiques attestant l’état de chaque noeud
- le contrôle du réseau ROS depuis l’IHM

Qt offre un système de paramétrage des applications par le biais de fichiers texte et d’une classe **QSettings** indépendante de la plateforme utilisée¹³. Cette dernière permet l’écriture de paramètres selon le paradigme clé / valeur et hiérarchisés sous forme de groupe. Dans notre IHM, une classe statique **SettingsManager** est utilisée pour écrire de tels paramètres, soit à l’initialisation de l’application soit lors de la configuration de celle-ci par l’utilisateur. Nous avons défini les groupes suivants relativement au réseau ROS :

13. En l’occurrence, sur un système Unix, de tels fichiers seront stockés au format .ini sous ~/.config/<filename>.ini

```

ros ..... groupe de paramétrage du réseau ROS
  └─<ID> ..... identifiant du groupe de paramètres
    ├─package
    │  └─<nom du package> ..... nom du package à invoquer
    ├─launchfile
    │  └─<launchfile>.launch..... nom du launchfile à invoquer
    ├─islocalnode
    │  └─<bool>..... indique si le noeud est local ou embarqué
    ├─label
    │  └─<GUI label>..... label à afficher sur l'interface
    ├─nodes
    │  └─node
    │     └─<liste de noeuds>..... liste de noeuds à stopper

```

L'utilisateur peut mettre à jour ou supprimer les valeurs par défaut et créer de nouvelles entrées directement depuis l'application sous **Edit > Settings > ROS** (voir figure 2.15).

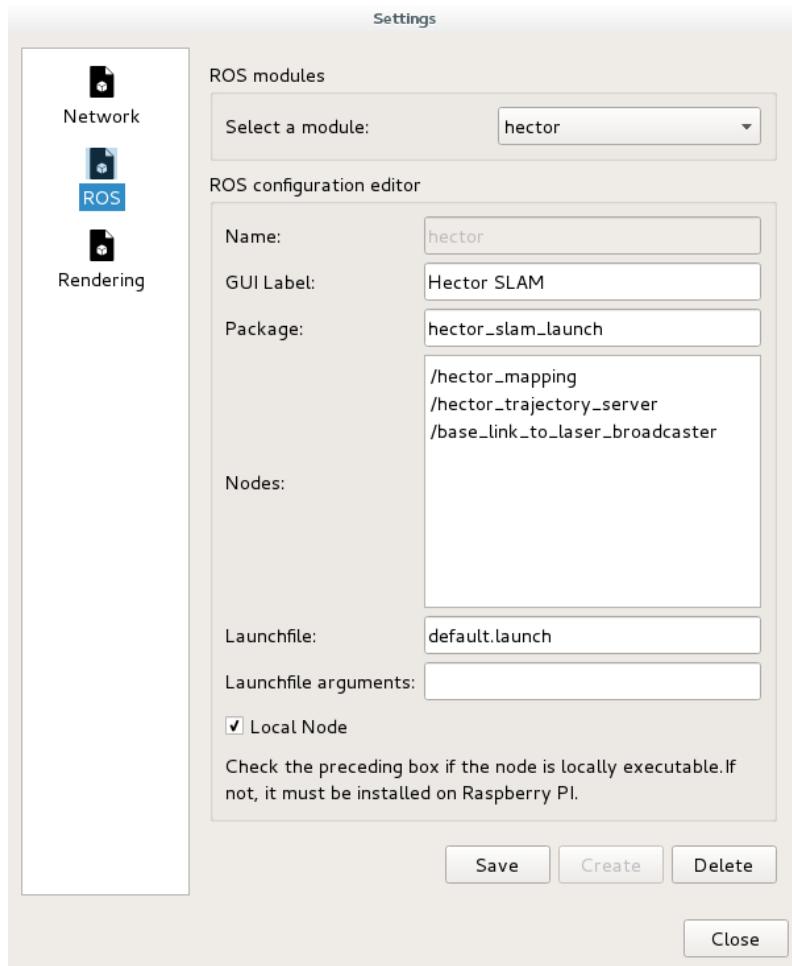


FIGURE 2.15 – Boîte de dialogue de configuration du réseau ROS depuis l'IHM

Ces paramètres sont ensuite parsés, permettant la présentation et l'utilisation de chaque module ROS de manière uniforme. D'un point de vue graphique, la classe **GuiControl** est responsable de la création dynamique des éléments de l'interface représentant les modules ROS, à partir de la clé de configuration **ros/<ID>/label/<GUI label>** tels que représentés figure 2.16.

Parallèlement, chaque nouvel identifiant sous **ros/<ID>** entraîne l'instanciation de la classe **NodesLauncher** ou **NodesLauncherOverSSH** –en fonction du caractère local ou non du package– et de la classe **NodesKiller**. Ces objets exécutent d'un shell Linux (**/bin/bash**)

ROS Network Overall:		
Bridges		Stop
Detection		Start
Hector SLAM		Start
Hector DEBUG		Stop
TF Broadcasters		Stop
Video Stream		Start

FIGURE 2.16 – Contrôle du réseau ROS créé dynamiquement sur l’interface graphique

au sein d’un membre Qt de type **QProcess** permettant l’exécution du launchfile afférent lorsque l’utilisateur presse le bouton “Start” du module. Ils sont aussi responsables de la redirection des sorties du processus vers les sorties dédiés à la journalisation des évènements de l’application¹⁴. La classe **NodesKiller** va quant à elle émettre des commandes système vers le ROS Master afin de tuer les noeuds d’un package lorsque l’utilisateur souhaitera mettre fin à leur exécution.

Pour résumer, un utilisateur voulant ajouter un nouveau package ROS à l’application devra le compiler de manière classique dans son workspace catkin, éditer un launchfile permettant d’executer le ou les noeuds afférents puis simplement renseigner les informations requises depuis l’IHM. Sous la section **Edit > Settings > Network** l’utilisateur pourra également configurer les noms d’hôtes et adresses IP distants, adaptant ainsi le système à d’autres modèles ou instances de robots. Si l’opérateur venait à utiliser une plateforme dotée de capteurs communiquant leurs résultats à un package ROS dédié, le contrôle de ces nouveaux exécutables pourrait être couvert par l’application en quelques manipulations seulement.

14. La classe **LogsHandler** est utilisée pour journaliser les messages d’une part dans des fichiers datés du répertoire d’exécution de l’application et d’autre part, dans la console du panneau de contrôle

Chapitre 3

Bilan et perspectives

3.1 Organisation du travail

3.1.1 Application du référentiel qualité interne

Nous exposons ici les éléments fourni par SII ayant guidé de manière significative la conduite du projet. Ceux-ci seront plus ou moins détaillés notamment au temps ayant été consacré à leur mise en place.

Les premiers éléments organisationnels qui ont été formalisés s'incarnent par deux documents de spécification appelés STBL et DCL. Le premier vise, comme son nom l'indique, à exprimer le plus objectivement possible le besoin qui justifie la réalisation logicielle, notamment au moyen d'exigences représentées figure 2.2. Cette énumération haut-niveau permet d'appréhender simplement et rapidement le périmètre du logiciel ciblé. Cependant, la réalisation de ce document en bonne et dûe forme nécessite une liste exhaustive d'exigences fonctionnelles vues comme des ensembles de fonctions qu'il convient de décrire en terme de rôle, d'entrées, de sorties et de traitements. Dans cadre de ce stage, nous définissons des fonctions comprises dans des modules distincts et étant reliées de la manière suivante : **relation entre fonctions + légende incorporée dans le schéma**

Puis, pour chacune des fonctions, on va définir des sous-fonctions, idéalement jusqu'à un niveau de granularité maximal où tous les types de données sont énumérés. Par exemple, on établit que la fonction *Transmettre* du module **Map_bridge** regroupe les quatre sous-fonctions *Recevoir un message*, *Formater un message* et *Émettre un message*. Afin d'illustrer cette démarche, nous donnons ici pour exemple la spécification de la fonction *Transmettre : Formater un message*.

1. Rôles

Cette fonction est activée à l'arrivée d'une demande de traitement de message sur la liaison SLAM \iff MAP_BRIDGE. Elle intervient après que la fonction de réception ait attesté de la consistance du message reçu. Elle permet d'extraire la charge utile du message reçu et d'en réduire significativement le volume avant envoi. Les données en sorties sont sérialisées. Cette extraction s'effectue en comparant OCCUPANCY_GRID avec une copie locale de la dernière carte reçue OLD_OCCUPANCY_GRID.

2. Entrées

Désignation	Type de données
Message OCCUPANCY_GRID	OccupancyGrid

FIGURE 3.1 – Entrées de la fonction *Transmettre : Formater un message*

3. Sorties

Désignation	Type de données
Message MAP	OccupancyGridUpdate
MessageINI_MAP	IniOccupancyGrid

FIGURE 3.2 – Sorties de la fonction *Transmettre : Formater un message*

Notons que la spécification des types de données d’entrées-sorties est donné dans un document externe. Ceux relatifs à l’exemple ci-dessus sont décrits en **Annexe XX**.

4. Traitements

NB : Les traitements sont identifiés de manière unique au sein d’un document, ils satisfont également des règles permettant d’être traités automatiquement par des logiciels de suivi de projet. Nous donnons ici un exemple succinct permettant d’en saisir le sens. Dans la pratique une sous-fonction donne lieu à plusieurs traitements.

[REQ_STBL_MAP_BRIDGE_FORM_1]

Si OLD_OCCUPANCY_GRID existe :

On compare le champ “data” de OCCUPANCY_GRID et de OLD_OCCUPANCY_GRID.

On crée la structure MAP à partir des valeurs de “data” qui diffèrent.

MAP est une chaîne de caractères dont les valeurs sont séparées par des ”, “.

[FIN_REQ]

Le Document de Conception Logiciel est quant à lui intervenu plus tard dans l’avancée du projet. Il consiste ne la formalisation des éléments conceptuels, en terme d’architecture physique et logicielle du projet. Ces briques ayant été largement explicitées au long de ce rapport nous n’attayerons pas d’avantage ce point.

Par ailleurs, un diagramme de Gantt présentant une granularité hebdomadaire a été effectué au mois de mars, ce document est présenté figure X. Cette réalisation vise à définir et ordonner les tâches à réaliser et, d’autre part, à estimer l’impact temporel de chacune d’entre-elles. Ce document a été réalisé dans une optique d’organisation personnelle mais a également constitué un outil d’évaluation et de communication avec M. Daumand qui a été en grande partie affecté en missions hors de l’agence. Il s’agissait donc pour nous de fixer les jalons clés, les *dead-lines* et les versions du logiciel à produire afin qu’il suive de manière pragmatique l’avancée du travail.

3.1.2 Vers une conduite AGILE adaptée

Au référentiel interne de qualité du logiciel se sont ajoutées à la gestion de ce projet certaines bonnes pratiques issues de la formation en Architecure et Sécurité du Logiciel prodiguée à l’INSA Centre Val-de-Loire qui ont pu être exploitées dès lors qu’Alban Chazot et moi-même avons été amenés à travailler de concert. Cette phase est intervenue dès que nos fonctionnalités respectives, décrites dans le cadre de nos sujets de stage étaient opérationnelles. En particulier, nous avons utilisé le gestionnaire de versions Git au travers du système de gestion de dépôts (forge) GitLab auquel nous avons appliqué une stratégie de création de branches appelée ”Git Flow“. Ce modèle d’utilisation de Git vise à minimiser les conflits et les régressions, accentuer la visibilité des tâches en cours ou terminées et de scinder clairement les phases de développement du logiciel. À cet effet nous avons adopté la démarche illustrée sur la figure 3.3 qui se lit de bas en haut et où les commits sont représentés par des points. Cela consiste à travailler sur une branche de développement (**develop**), à partir de laquelle nous créons une branche par fonctionnalité (**feature**),

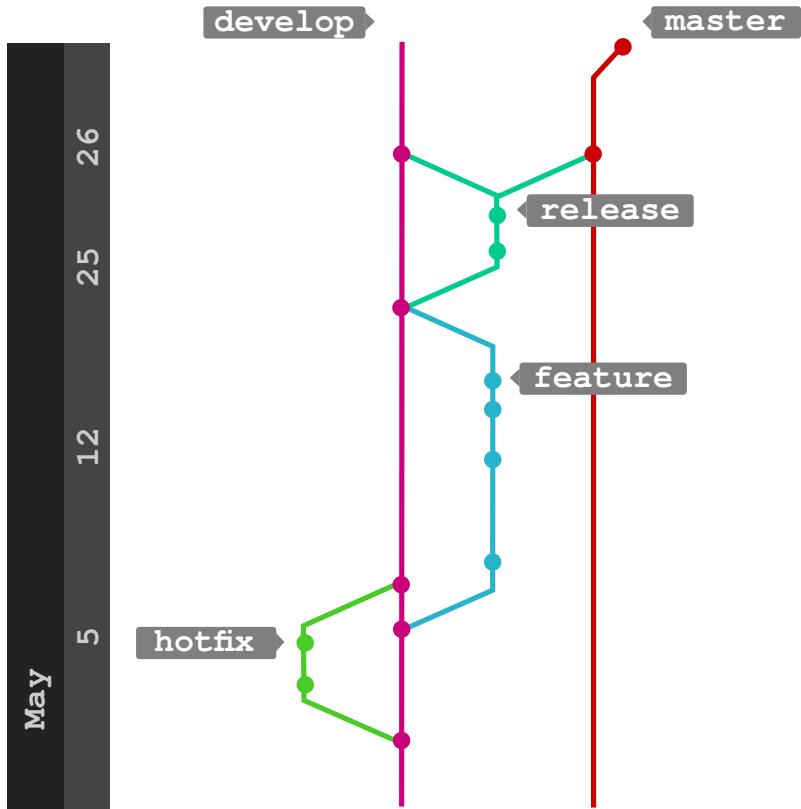


FIGURE 3.3 – Politique de gestion de branches Git adoptée

éventuellement une branche **hotfix** pour une correction de bug et, enfin, une branche **realease** qui va aboutir sur la dernière version stable du logiciel. Cette branche est ensuite répercutée sur **develop** et **master**, dans le premier cas pour continuer les actions de développement et dans le second, afin de permettre la récupération des sources ou son déploiement.

Aussi, l'**Uploader** a constitué une refonte majeure de l'architecture du logiciel qui a été conceptualisée et développée en équipe. Afin de répartir les tâches que nécessitait son implémentation, nous avons pu expérimenter l'utilisation d'un Kanban recensant :

- les tâches à effectuer, par exemple *Enregister les données de cartographie dans le format DOG*
- la complexité relative de chacune d'entre-elles
- les dépendances de certaines tâches les unes par rapport aux autres

Définir la "complexité" d'une tâche revient à lui attribuer un score en se référant aux scores attribués pour les tâches précédentes. Les méthodes AGILE –notamment SCRUM– préconisent le recours à des échelles de quantification relatives plutôt qu'à des jours hommes ou d'autres échelles se voulant précises. Concrètement on peut utiliser les valeurs de la suite de Fibonacci qui suivent ce que l'on appelle la courbe d'incertitude, à savoir qu'au plus une valeur est élevée, au plus l'écart avec la valeur suivante sera grande. Nous avons estimé la complexité des unités de travail à réaliser selon une méthode également empruntée à SCRUM, appelée *planning poker*. Tous les participants disposent d'un jeu de carte qui, dans notre cas, comporte les nombres $\frac{1}{2}, 1, 2, 3, 5, 8, 13, \infty$. L'effort de réalisation est ensuite estimé en même temps pour une tâche donnée, puis soumis à discussion dans le cas de désaccord. Cette pratique est à la fois rapide, ludique et présente l'intérêt de dénuder d'emblée la quantification de l'influence mutuelle des participants.

Ce Kanban a ainsi été adopté dans une optique d'ordonnancer les tâches et, accessoirement,

de les répartir entre les membres de l'équipe. Il a aussi permis d'attester visuellement de nos avancées respectives et du nombre de tâches en cours permettant un allègement ou une répartition de la charge si besoin. La consultation du nombre de tâches restantes a également joué dans nos décisions de poursuivre telle ou telle fonctionnalité au profit d'autres par exemple.

3.2 Résultats en vue d'un prolongement

3.2.1 Un module de SLAM en adéquation avec les attentes du projet

Nous nous intéressons dans un premier lieu à la fidélité des résultats de SLAM et dans un second temps, à une appréciation de l'ensemble du projet, incluant également les travaux d'Alban Chazot et bien sûr la vision de M. Daumand.

La figure suivante donne un aperçu de la précision des résultats de SLAM. Elle met en parallèle les résultats cartographiques du système avec un plan d'évacuation des locaux dans lesquels l'acquisition a pu être menée. Pour des raisons pratiques, toutes les pièces n'ont pu être scannées, puisque celles-ci hébergent des collaborateurs ou directeur d'agence menant leurs activités.

Enfin, la réussite du projet dans sa globalité a pu être attestée par les résultats d'une présentation devant des collaborateurs commerciaux et directeurs de projet qui est intervenue le 13 juin 2017. Cette étape a permis de recueillir des avis extérieurs sur le projet lui-même et d'estimer sa potentielle viabilité commerciale et technique. Ayant été positivement perçu, le système démonstrateur réalisé a été présenté brièvement auprès de Nexter System, enclin à organiser une rencontre à cet effet au mois de septembre.

3.2.2 Pourquoi continuer SRT2M

La présentation du projet à un client tel que Nexter Systems étant un objectif établi dès les premières phases du projet, nous avons veillé à maximiser le potentiel d'appropriation de ce dernier par de tierces personnes. Cette volonté s'est incarnée par la mise en place des éléments suivants :

- un manuel utilisateur interactif et ergonomique présenté au sein d'une interface web
- un manuel d'installation, agrémenté de résolutions problèmes pouvant être rencontrés lors de cette phase
- la création d'une documentation automatique et, là aussi, intercative grâce au logiciel Doxygen
- l'automatisation de l'installation pour la distribution Debian 8, notamment au moyen du gestionnaire de paquets apt, et la gestion des dépendances manquantes sur la station hôte
- l'adoption de ROS répondant à un critère de haute flexibilité fonctionnelle et matérielle
- l'adaptation des possibilités de l'IHM au caractère modulaire des éléments du middleware ROS

3.3 Retour d'expérience

3.3.1 Apports et difficultés du projet

Ce projet s'est trouvé complexe et stimulant à tous les niveaux.

Les phases de spécification du besoin ont été fastidieuses tant du fait de la rigueur des documents requis qu'au regard de mon manque de vision intial et de ma difficulté à me représenter ce que serait le système en bout de stage. Cela s'explique notamment par une méconnaissance intiale du domaine de la robotique et du secteur de la défense. Les aides conjointes de MM. Daumand et Hafiane, de vastes recherches documentaires et un intérêt certain pour ce sujet où tout était à construire ont fort heureusement motivé une rapide appropriation des problématiques et enjeux afférents.

Les outils utilisés tels que ROS et à plus faible mesure Qt, ou les fondements théoriques du SLAM puis d'Hector SLAM ont également nécessité des temps de compréhension et d'assimilation conséquents. Bien que n'ayant pas outrepassé le temps imparti à la recherche de solutions techniques, il est assez compliqué de passer plusieurs semaines à approfondir des connaissances très spécifiques –comme dans le cas de ROS– sans avoir la moindre idée de ce que donneraient les débuts de l'implémentation. Cet effet tunnel s'est trouvé renforcé du fait des délais d'acquisition du matériel et des incertitudes liées au fonctionnement du robot.

Aussi, nous avons eu la chance d'appréhender la réalisation d'un système à naître, sans qu'aucune étude préalable n'ai été menée. Ce point a à la fois constitué un atout majeur dans le choix de ce stage, mais également un point d'incertitude indéniable qui complique la visualisation d'un système final. Ces difficultés ont peut-être avant tout été surmontées par la communication au sein de l'équipe, permettant de désamorcer les situations de doutes.

Enfin, j'ai particulièrement apprécié l'alternance entre une période assez longue de travail individuel, et un travail en équipe –certes restreinte– dans une seconde phase du développement. Dans le premier cas, cela force à une organisation, une prise de décisions et de responsabilités individuelles accrues qui n'a que très peu été expérimentée lors de projets scolaires au long-cours.

3.3.2 Évolution personnelle au sein de la structure

Bibliographie

- [1] SII Siège Social - Paris, *Document de référence incluant rapport annuel financier. Exercice 2015/2016*, 172 p.
- [2] SII Ile de France - Paris, *Memeto Agence Ile-de-France. Missions, Organisation, Processus*, Version 11, 102 p, Mai 2017.
- [3] Wikipédia, *Société Anonyme*, https://fr.wikipedia.org/wiki/Soci%C3%A9t%C3%A9_anonyme 21 juin 2017.
- [4] Frank Canton relayé par le Ministère de l'intérieur, *Armement et défense, traditions du Cher*, <https://www.interieur.gouv.fr/Archives/Archives-des-dossiers/2016-Dossiers/Le-Cher/Armement-et-defense-traditions-du-Cher> janvier 2016.
- [5] Cognitive Robotics, Massachusetts Institute of Technology, Søren Riisgaard and Morten Rufus Blas *SLAM for Dummies A Tutorial Approach to Simultaneous Localization and Mapping*, https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam blas_repo.pdf Spring 2005
- [6] Cyrill Stachniss, Udo Frese, Giorgio Grisetti, *Give your algorithm to the community*, <https://www.openslam.org> Spring 2005
- [7] ROS Documentation, *Master* <http://wiki.ros.org/Master>
- [8] ROS Documentation, *Nodes* <http://wiki.ros.org/Nodes>
- [9] ROS Documentation, *Topics* <http://wiki.ros.org/Topics>
- [10] ROS Documentation, *Services* <http://wiki.ros.org/Services>
- [11] ROS Documentation, *catkin/package.xml* <http://wiki.ros.org/catkin/package.xml>
- [12] ROS Documentation, *catkin/CMakeLists.txt* <http://wiki.ros.org/catkin/CMakeLists.txt>
- [13] S. Kohlbrecher and J. Meyer and O. von Stryk and U. Klingauf, *A Flexible and Scalable SLAM System with Full 3D Motion Estimation* http://www.sim.informatik.tu-darmstadt.de/publ/download/2011_SSRR_KohlbrecherMeyerStrykKlingauf_Flexible_SLAM_System.pdf 2011
- [14] Stefan Kohlbrecher, Christian Rose, Dorothea Koert, Paul Manns, Florian Kunz, Benedikt Wartusch, Kevin Daun, Alexander Stumpf and Oskar von Stryk, *RoboCup Rescue 2016 Team Description Paper Hector Darmstadt* http://www.robocup2016.org/media/symposium/Team-Description-Papers/RescueRobot/RoboCup_2016_RescueR_TDP_HectorDarmstadt.pdf 2016
- [15] Site officiel de l'équipe de développement et de maintien d'Hector SLAM, <http://www.teamhector.de/>
- [16] Stefan Kohlbrecher, *Hector SLAM Git repository* https://github.com/tu-darmstadt-ros-pkg/hector_slam 29 mars 2011
- [17] ROS Documentation, *Debian install of ROS Kinetic* <http://wiki.ros.org/kinetic/Installation/Debian>

- [18] ROS Documentation, *tf* <http://wiki.ros.org/tf>
- [19] Foote, Tully, in Technologies for Practical Robot Applications (TePRA), *tf : The transform library* http://wiki.ros.org/Papers/TePRA2013_Foote?action=AttachFile&do=get&target=TePRA2013_Foote.pdf avril 2016
- [20] Wim Meeussen, *Coordinate Frames for Mobile Platforms* <http://www.ros.org/reps/rep-0105.html> 27 octobre 2010
- [21] *ROS Press Kit* <http://www.ros.org/press-kit/>
- [22] Qt Documentation, *QMainWindow Class* <http://doc.qt.io/qt-4.8/qmainwindow.html#details>