ing. Ordre des ingénieurs du Québec

Génie robotique

Guide pour le volet pratique de la formation en conception

Ce document a été préparé avec la collaboration :
du professeur **François Michaud, ing., Ph. D.**,
du professeur **François Ferland, ing., Ph.D.**,
de **Dominic Létourneau, ing., M.Sc.A.**,
de **Cédric Godin, ing., M.Sc.A.**
Département de génie électrique et de génie informatique de l'Université de Sherbrooke

Ainsi qu'avec la collaboration de :
David Brodeur, ing., M.Sc.A.
Productique Québec

Olivier Lecompte, étudiant à la maîtrise
Département de génie mécanique de l'Université de Sherbrooke

UNIVERSITÉ DE SHERBROOKE
Faculté de génie

Table des matières

Introduction

Pour réaliser un projet en conception, on doit se baser sur la méthodologie de la conception en ingénierie dont traitent les livres suggérés dans le volet théorique. Il faut aussi recourir à des outils appropriés aux approches et aux méthodes à mettre en œuvre. C'est précisément le but de ce document : guider le candidat et la candidate dans l'accomplissement d'un projet de conception en génie robotique.

Vous trouverez dans ce guide, d'une part, une description des principales étapes (ou phases) d'un projet de conception en génie robotique et, d'autre part, des renseignements sur les outils à utiliser et les approches ou techniques à appliquer.

En complément, nous reproduisons à titre d'exemple un projet élaboré par des étudiants.

Description d'un projet de conception en

génie robotique

Le génie robotique fait appel à des compétences en conception et en gestion de projets. Plus précisément, le génie robotique est une discipline à part entière qui fait constamment intervenir le génie mécanique, le génie électrique et le génie informatique en interaction les uns avec les autres. Cela requiert non seulement des compétences dans les domaines mentionnés (mécanique, électrique, informatique), mais également des compétences à l'intersection de ces savoirs disciplinaires. Ces compétences s'avèrent essentielles pour le développement de systèmes dans les secteurs manufacturier, de l'aérospatiale, de la santé, du transport, de la construction, de l'agriculture, de la surveillance, bref dans tous les domaines où des systèmes physiques doivent prendre des décisions à partir de capteurs et réaliser des actions dans des environnements réels.

Un projet de conception en génie robotique permet notamment d'améliorer une situation existante ou de concevoir un nouveau produit pour répondre à un nouveau besoin.

Un projet type est divisé en 5 étapes :

- l'analyse d'une problématique et la formulation d'un cahier des charges ;
- la conception préliminaire ;
- la conception détaillée ;
- la réalisation ;
- la validation.

Chacune de ces étapes consiste à accomplir une série d'actions, qui sont détaillés ci-dessous.

Étape 1 : Analyse d'une problématique et formulation d'un cahier des charges

1. Formulez le problème : décrivez la problématique ; déterminez les objectifs du projet et les résultats attendus.
   a. Faites la description détaillée de l'ensemble des fonctions et spécifications techniques.
   b. Indiquez les normes, lois et règlements devant être pris en compte dans le projet, en particulier pour ce qui concerne la santé-sécurité et les impacts potentiels sur l'environnement.
   c. Déterminez les critères d'évaluation et le barème qui permettent l'atteinte des objectifs, la cohérence avec les spécifications et le respect des restrictions.

2. Dessinez un arbre fonctionnel du projet, qui permet d'organiser chaque fonction dans plusieurs sous-projets. Précisez qui est responsable de chaque fonction.

3. Établissez le budget du projet, y compris une estimation des coûts et des revenus du projet.
   a. Justifiez les montants et faites la ventilation par sous-projets.
   b. Précisez à quel moment est attendu chaque flux monétaire.

4. Établissez une planification et un échéancier détaillés, présentés notamment sous la forme d'un diagramme de Gantt.

Étape 2 : Conception préliminaire

1. Cherchez des solutions.
   a. Procédez à des séances de remue-méninges (brainstorming).
   b. Éliminez rapidement les solutions les plus loin des objectifs par une étude de praticabilité.

2. Effectuez le processus de sélection.
   a. Décrivez les solutions restantes.
   b. Effectuez un dimensionnement grossier des différentes solutions.
   c. Préparez une étude comparative des solutions restantes pour obtenir la convergence vers la solution retenue.

3. Déterminez quelle est la solution retenue.

Étape 3 : Conception détaillée

Indiquez les étapes et les éléments de conception de la solution retenue, en précisant les techniques et les outils d'ingénierie utilisés (tableau 1).

Le tableau 1 présente des exemples d'outils généralement utilisés dans les différentes disciplines associées au génie robotique. Les outils informatiques servent à faire la programmation de systèmes ordinés utilisés pour le contrôle ou les interfaces du robot avec son environnement d'opération, ce qui peut inclure des humains ou d'autres machines ou dispositifs. Les outils mécaniques servent pour le dessin de plans de pièces et leur assemblage pour leur fabrication, ainsi que pour concevoir des outils de simulation et d'analyse. Les outils électriques sont destinés à la conception et la simulation de circuits imprimés ou de circuits logiques programmables. Enfin, les outils robotiques sont plus en lien avec des systèmes robotiques commerciaux, comme des bras industriels et des outils en automatisation industrielle, ou des environnements de programmation et de simulation pour la conception de robots.

Tableau 1 : Exemples d'outils utilisés en génie robotique selon les disciplines associées à différents projets.

Disciplines	Outils	Fonctions
Génie informatique	Arduino IDE / Android Studio Raspberry Pi	Environnement de programmation de systèmes ordinés
	C et C++ HTML (base) Matlab Python	Langages de programmation
	Geany GitHub PyCharm Visual Studio	Environnement de développement logiciel
Génie mécanique	AutoCAD Catia Impression 3D (ex. : Makerbot print) OnShape Shopturn Sketchup SolidWorks	Conception de pièces et d'assemblage 3D
	Automation Studio Motion Genesis Matlab / Simulink / LabView	Conception, simulation et analyses de systèmes mécatroniques
Génie électrique	Altium Cadence Eagle Ki-CAD LTSpice	Conception et simulation de circuits imprimés
	VHDL Xilinx ISE	Développement de circuits logiques programmables

Génie robotique	Bras Fanuc Bras UR	Bras robotiques industriels
	Ladder / Grafecet Handling Pro RobotStudio	Programmation d'automates et de bras robotiques industriels
	ROS	Intergiciel pour la conception de robots

En fonction des projets, les outils à utiliser changent selon les composants du projet. Le tableau 2 présente des exemples d'outils utilisés pour des projets en robotique industrielle.

Vous trouverez ensuite un exemple de projet de conception d'un véhicule robotique, une liste de projets de conception en génie robotique réalisés à la Faculté de génie de l'Université de Sherbrooke, puis un document précisant les éléments dont il faut tenir compte dans la conception robotique. Ainsi, qu'un projet de conception en robotique ayant été effectué par des étudiants.

Tableau 2 : Exemples d'outils utilisés dans différents projets en robotique industrielle.

Projets	Équipements	Logiciels	Langages de programmation et API
Automatisation d'une chaîne de production	• Automate programmable (Allen-Bradley, Beckoff, Omron, Siemens, etc.)	• Logiciel de programmation d'automate (Rockwell Studio 5000, etc.)	• Diagrammes à échelle, SFC, blocs de fonction, texte structuré
Programmation robotique	• Robot industriel (FANUC, ABB, etc.) • Robot collaboratif (UR, FANUC, Omron, etc.)	• Logiciel de programmation et de simulation de robot (Handling Pro, Robot Studio, Polyscope, RoboDK, etc.)	• Langage de programmation propriétaire : TP, Rapid, URScript, etc.
Cellule robotisée d'assemblage ou de manutention	• Préhenseur (Festo, OnRobot, Robotiq, etc.)	---	---
Cellule robotisée de soudure	• Torche à souder	• Logiciel de conception mécanique (AutoCAD, SolidWorks, etc.) • Logiciel de génération de trajectoires (Octopuz, etc.)	---
Cellule robotisée d'usinage	• Outil d'ébavurage (ATI Industrial Automation, etc.) • Compensation mécanique (FER Robotics, Pushcorp, etc.)	• Logiciel de conception mécanique (AutoCAD, SolidWorks, etc.) • Logiciel de génération de trajectoires (Octopuz, etc.)	
Cellule d'usinage CNC	• Machine à commande numérique pour le fraisage ou le tournage (EMCO, Mazak, etc.)	• Logiciel de simulation de machine CNC (Vericut, etc.)	• G-code
Interface humain-machine (HMI)	• Automate programmable (Allen-Bradley, Beckoff, Siemens, etc.)	• Logiciel de conception d'interface (InTouch, Ignition, etc.)	---

Inspection/localisation 2D	• Caméra d'acquisition 2D (Flir, The imaging source, etc.) • Caméra intelligente (Cognex, Keyence, Omron Microscan, Sick, etc.)	• Logiciel pour caméra intelligente (In-Sight, VisionPro, AutoVision, etc.)	• C/C++, OpenCV • Librairies commerciales (HALCON, etc.)
Inspection/localisation 3D	• Profilomètre (Cognex, Keyence, LMI, etc.) • Scanneur 3D (Creaform)	• Logiciel pour profilomètre (Gocator Emulator, etc.) • Logiciel d'analyse 3D (Polyworks, etc.)	• C/C++, OpenCV • Librairies commerciales (GoSDK, etc.)
Sécurisation d'une cellule automatisée (avec ou sans robot)	• Arrêt d'urgence • Automate de sécurité (Omron, etc.) • Cages de protection • Rideau virtuel (SICK, etc.) • Scanneurs de zone (SICK, etc.)	• Logiciel de programmation d'automate de sécurité	• Diagrammes à échelle, SFC, blocs de fonction, texte structuré

Conception d'un véhicule robotique : le RaceCAR-UdeS

Conception mécanique

Comme outils pour la conception mécanique assistée par ordinateur, SolidWorks, Inventor et OnShape ont été utilisés. La figure 1 montre les dessins réalisés avec OnShape. Il est à noter que la plateforme de développement OnShape s'avère intéressante, car elle favorise le travail d'équipe avec sa caractéristique de gestion des versions similaire à la plateforme de développement logiciel GitHub. De plus, les documents modélisés peuvent ensuite être partagés avec la communauté. En ce qui a trait à la fabrication, les outils principaux sont la découpeuse laser et l'impression 3D. La plateforme étant une modification d'un véhicule à échelle réduite 1 : 10, il faut fabriquer un ensemble réduit de pièces, ce qui fait que des outils de prototypage rapide sont adéquats. Advenant une production plus importante, on opterait de préférence pour des outils d'usinage standard.

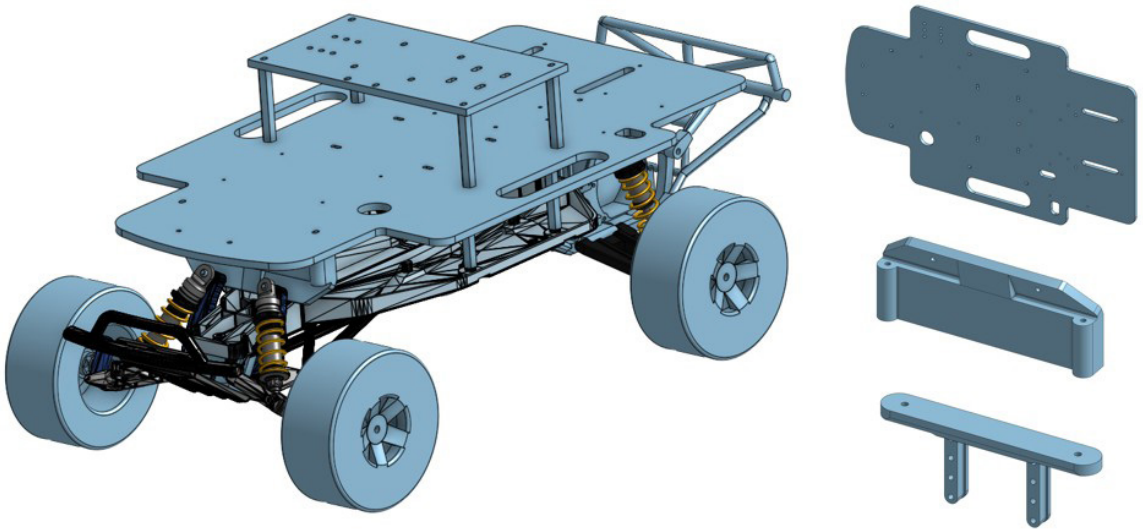

Figure 1 – Exemples d'assemblage et de pièces réalisés avec le logiciel OnShape.

Conception électronique

Pour la conception des cartes électroniques du RaceCAR-UdeS, le logiciel Eagle est utilisé pour dessiner le schéma électrique et le circuit imprimé. Toujours dans Eagle, les fichiers de fabrication (gerber) sont exportés avant d'être envoyés au fabricant sélectionné. La même chose peut être faite avec d'autres suites de logiciels, comme Altium ou Ki-CAD.

Conception informatique

La portion informatique comprend principalement les éléments nécessaires au contrôle du véhicule. Le contrôleur à haut niveau est réalisé dans l'environnement ROS (Robot Operating System) à partir de codes dans les langages Python et C++. Le lien entre le contrôle à haut niveau, effectué sur l'ordinateur de bord Raspberry Pi, et le contrôle à bas niveau, pris en charge par un microcontrôleur Arduino Mega, est réalisé par port série via Rosserial. L'ensemble du code nécessaire au fonctionnement du projet RaceCAR-UdeS se trouve sur la plateforme de développement logiciel GitHub.

Exemples de projets de conception en génie robotique

Il existe une très grande variété de projets d'ingénierie qui peuvent être associés au génie robotique. Le tableau 3 donne quelques exemples de projets réalisés dans le cadre des projets majeurs de conception à la Faculté de génie de l'Université de Sherbrooke.

Tableau 3 : Exemples de projets majeurs de conception réalisés à l'Université de Sherbrooke.

Projets	Milieux de pratique
Conception d'un robot cueilleur de tomates	Agricole
Conception et réalisation d'une plateforme de robotique d'essaim	Agricole
Conception d'un robot activateur de volaille dans un poulailler avec lectures des données de l'environnement	Agricole
Réalisation d'un système de caméras robotisées permettant la captation automatisée d'un sujet se déplaçant dans un environnement déterminé	Cinématographie
Conception d'un rover autonome pour la compétition Canadian International Rover Challenge 2022	Compétition robotique
Conception et fabrication d'un robot de combat pour participer à la compétition BattleBots 2022	Compétition robotique
Conception d'un robot capable d'interagir avec un humain par la voix et les gestes dans un environnement physique réel	Interaction humain-robot
Conception d'une plateforme robotique pouvant effectuer automatiquement des manipulations et des protocoles de laboratoire en biologie synthétique	Laboratoire en biologie
Conception d'une usine numérique collaborative	Manufacturier
Conception d'un système de trois chariots robotisés fonctionnant de façon concertée pour transporter des charges lourdes	Manufacturier
Conception d'un exosquelette pour les membres inférieurs avec des actionneurs compliants magnétorhéologiques	Santé
Conception d'un prototype de robot de sécurité abordable capable d'effectuer une patrouille dans un environnement institutionnel pendant laquelle il repère les intrus	Sécurité
Conception d'une preuve de concept d'une souffleuse électrique résidentielle autonome et sécuritaire avec un contrôle en temps réel	Service

Éléments à prendre en considération dans la conception robotique

Lorsqu'on conçoit un robot, il est bon de garder en tête les éléments suivants :

- La conception d'un robot implique de choisir des composantes matérielles (par exemple des capteurs et des actionneurs, assemblés à une structure mécanique) et logicielles (par exemple des systèmes informatisés, des mécanismes de contrôle et de traitement des données). Chacune des composantes (logicielles ou matérielles) augmente la complexité du robot et peut interagir avec d'autres, provoquant des fautes résultant de leur combinaison, fautes qui sont absentes lorsque ces composantes sont utilisées seules. Sous-estimer le temps requis pour l'intégration est l'erreur la plus courante en conception robotique. Si l'augmentation de complexité n'est pas justifiée par une augmentation similaire de la fonctionnalité, cela aura des conséquences sur le projet : le temps de conception, de réalisation et de déverminage est plus long, et le nombre de résultats imprévus augmente. Il est donc préférable d'éliminer les aspects qui ne sont pas utiles au robot pour effectuer sa tâche. Une fois une conception robuste réalisée, il est possible d'ajouter d'autres composantes et d'améliorer la fonctionnalité du robot de manière incrémentale. Ainsi, la conception d'un robot performant peut requérir plusieurs itérations de conception, prototypage et tests, augmentant graduellement les fonctionnalités du robot.
- Un robot a besoin de plusieurs composantes pour réaliser sa tâche de façon adéquate. Il faut éviter de porter une attention trop marquée pour une composante particulière au détriment du système en entier. Il faut aussi considérer une certaine redondance afin d'assurer la sécurité du robot dans son environnement d'opération.
- Un code bien conçu conceptuellement peut quand même générer des comportements non planifiés pour le robot. Ceci est habituellement causé par l'inhabilité du concepteur de bien prendre en considération certaines complexités du monde réel : les commandes du robot peuvent ne pas être exécutées à la perfection; les lectures des capteurs peuvent être imprécises, bruitées ou insuffisantes, etc. La programmation du robot doit prendre en considération ces ambiguïtés.

Il faut toujours se rappeler que les actions du robot sont fonction de ses caractéristiques physiques, comme sa masse, la force de ses moteurs, la traction de ses roues, le temps de cycle de traitement, etc. Un programme qui effectue un traitement en 0.1 msec n'est pas mieux qu'un autre qui le fait en 1 msec si le robot a besoin de 0.1 sec pour effectuer une action. Cependant, le traitement que doit effectuer un robot doit se faire suffisamment rapidement pour que le robot pose des actions adéquates en fonction de la dynamique de l'environnement.

Projet :

Projet Securbot

Ce projet, préparé par des étudiants, est un bon exemple d'application de la méthodologie en conception et de présentation des outils utilisés.

Faculté de génie
Département de génie électrique et génie informatique

SecurBot

Rapport final de conception

Projet majeur de conception en génie électrique/génie informatique
GEL801/GIF801

Présenté à
Dominic Létourneau et François Michaud,

pour le groupe IntRoLab

Présenté par
Olivier Belval - belo1601
Édouard Dénommé - dene2303
Valérie Gauthier - gauv2903
Cédric Godin - godc2305
Alexandre Guilbault - guia2812
Édouard Légaré - lege2104
Philippe Marcoux - marp1824
Anthony Parris - para2709

Sherbrooke - 16 décembre 2019

Table des matières

1 Mise en Contexte

1.1 Résumé du projet

Le but du projet SecurBot est de concevoir un prototype de robot de sécurité abordable utilisant des technologies disponibles gratuitement en ligne. Le robot a pour but d'effectuer une patrouille dans un environnement institutionnel pendant laquelle il repère les intrus. En plus d'avoir développé l'architecture logicielle permettant à un robot d'effectuer cette tâche, l'équipe a conçu une interface utilisateur permettant de suivre et de contrôler le robot, un serveur permettant d'enregistrer les intrus, une armure mécanique permettant de protéger le système de son environnement et un système d'alimentation personnalisée permettant au robot d'aller se recharger par lui-même.

Mots-clés : Robotique, Sécurité, Architecture par service, Interface graphique, Cartographie automatique, Navigation autonome, Reconnaissance visuelle, Circuit imprimé, ROS, Code ouvert.

1.2 Introduction et motivation

Le Laboratoire de robotique intelligente, interactive, intégrée et interdisciplinaire (IntRoLab) a développé, au fil des années, plusieurs technologies adaptées aux robots mobiles et aux robots de téléprésence. Par exemple, RTAB-Map, un outil de cartographie et de navigation, ODAS, un logiciel de détection de sources sonores ainsi que Tera-Vigil, une interface pour la téléprésence robotique et l'assistance aux soins à domicile.

Constatant l'engouement actuel pour les robots de sécurité, le laboratoire IntRoLab a décidé de concevoir une preuve de concept d'un de ces robots qui utilise les technologies du laboratoire. Le laboratoire n'ayant pas les ressources humaines nécessaires pour réaliser le projet, l'équipe SecurBot, composée de 8 étudiants en génie électrique, a été mandatée pour faire la conception et le développement de ce prototype dans le cadre de leur projet de fin de baccalauréat.

1.3 État de l'art

Dans le but de réaliser un produit de sécurité moderne et pertinent aux contextes actuels, SecurBot se devait d'intégrer des technologies récentes afin de pouvoir apporter une addition authentique et une contribution technologique significative. Dans cette optique, il était important de considérer ce qui se faisait sur le marché à cette heure. Dans cette section, l'état de l'art en ce qui concerne les robots de sécurité est examiné. Alors que le rapport fourni à cet égard dans la deuxième section du MIP se concentrait plus sur les solutions finies et les solutions de sécurité, la présente analyse explore également les différents modules technologiques implémentés dans le robot SecurBot en les comparant à des technologies similaires.

Crosswing

La compagnie Crosswing [1] conçoit le VirtualME [2], un robot destiné au grand public ainsi qu'un robot de sécurité nommé Bishop. Ces robots sont basés sur une plateforme commune appelée nav2. Celle-ci est conçue pour la navigation rapide dans des espaces intérieurs. Elle comporte un système de traction omni-directionnelle permettant un positionnement idéal [3]. De plus les robots de Crosswing sont équipés d'une série de périphériques similaires à ceux qui sont présents sur Securbot : une caméra RGBD, des capteurs de distance ultrasoniques, un essaim de microphones pour identifier les sources de son et un écran LCD pour

permettre de communiquer avec un utilisateur.
Atouts et avantages des robots de Crosswing :

- Grâce à son système de motorisation omnidirectionnel, le nav2 est plus mobile dans les environnements intérieurs sur plancher plat. SecurBot ne prévoit pas intégrer de telles méthodes de locomotion au cours de ce projet.
- Le modèle Bishop est équipé d'une caméra panoramique qui permet de récolter un flux vidéo 360° d'autour du robot. Les angles de vue récoltés sont utiles dans une application de sécurité pour récolter un maximum de données.
- Utilisation de la pile WebRTC [4], afin de pouvoir acheminer des flux vidéos vers des appareils de consommation. Cette technologie permet d'acheminer des vidéos de haute qualité vers la plupart des navigateurs modernes. Il est également prévu que SecurBot incorpore le même genre de technologie.
- Les modèles VirtualME sont capables d'établir un lien de téléprésence par vidéoconférence.
- Le robot virtualMe a démontré des capacités de suivi de trajectoire ainsi que des capacités d'escorte personnelle. Il peut donc repérer, identifier et accompagner un humain dans son champ d'action. Cela ouvre plusieurs possibilités d'interaction avec l'utilisateur dans un environnement de déploiement.

Limitations des robots de Crosswing :

- Les robots nav2 semblent peu adaptés aux surfaces accidentées. La taille, les proportions et l'assise sur trois roues font en sorte que le robot serait plus risqué à exploiter sur des surfaces qui ne sont pas complètement planes.

FIGURE 1 – Robot VirtualME et représentation explosée du Bishop de Crosswing

Knightscope

Knightscope [5], une compagnie américaine offre des solutions de robots de sécurité. La compagnie a déjà développé plusieurs robots qui sont en déploiement chez leurs clients. La compagnie offre des solutions de sécurité très intégrées et opère sous un modèle de service. Leurs robots sont déployés chez les clients qui sont facturés à l'heure. Ce modèle met en perspective le rôle qu'auront les robots dans le remplacement de la main-d'œuvre dans un avenir proche. Knightscope a développé plusieurs robots de différentes tailles pouvant naviguer sur des terrains de toutes sortes, incluant un modèle stationnaire qui peut être déployé près d'infrastructures critiques telles que des points de vente ou les points d'entrée et sortie d'un bâtiment.
Atouts et avantages des robots de Knightscope :

- La compagnie compte maintenant quelques années d'expérience avec des systèmes robotiques qui sont déployés en production. Cette expérience leur confère la boucle de rétroaction nécessaire avec le client pour valider leur technologie et itérer sur les points critiques de la technologie. Dans une entrevue audio [6], le CEO de Knightscope, William Santana Li, explique les difficultés rencontrées sur le terrain lors de déploiements.
- Les robots Knightscope intègrent une grande variété de capteurs comme des caméras, des Lidars et des encodeurs. Ils s'en servent pour reconstruire leur environnement virtuellement et pour s'y positionner.

Limitations des robots de Knightscope :

- Robots de taille imposante qui peut affecter la perception publique du robot. Plusieurs sources de nouvelles, ayant couvert les déploiements d'appareils Knightscope, rapportent que ceux-ci attirent énormément l'attention. Cela représente un désavantage dans la mesure que le robot peut être impacté dans ses fonctions de surveillance s'il est le centre de l'attention.
- L'environnement Knightscope est entièrement fermé et contrôlé par la corporation Knightscope. Cela limite l'interopérabilité et empêche le développement par des tierces parties.

FIGURE 2 – Knightscope K5

Segway Loomo

La gamme de produits Segway a gagné beaucoup de popularité comme véhicule électrique personnel. La compagnie s'est également lancée dans la conception de plateformes robotiques [7]. Leur produit vedette est le Loomo. Il s'agit d'un robot se tenant debout en équilibre sur deux roues comme le véhicule électrique Segway. Il est muni de caméras et d'un ordinateur de bord.

Atouts et avantages du Segway Loomo :

- Haute vitesse de déplacement et grande charge utile. Le Segway est équipé de moteurs puissants lui conférant un bon ratio puissance/masse.
- Peut dédoubler en véhicule électrique personnel. Le Loomo peut également être utilisé comme un Segway traditionnel, ce qui peut diversifier ses utilisations. Par exemple, un gardien de sécurité pourrait s'en servir pour accéder à un lieu d'intervention ou comme véhicule de patrouille.
- Segway offre une interface de programmation (API) [8] pour les développeurs qui veulent étendre les fonctionnalités du Loomo. Grâce à ces efforts, certaines équipes indépendantes ont pu expérimenter le développement de plateformes de navigation [9].

Limitations du Segway Loomo :

- La durée de vie des batteries du Loomo est limitée relativement à celle des autres plateformes robotiques énumérées dans cette liste. La durée d'un cycle de recharge peut varier beaucoup en fonction de l'utilisation et de s'il est utilisé comme véhicule. Lorsqu'il est utilisé comme véhicule, il a une autonomie de moins d'une heure.

Robotex Avatar III

Diverses compagnies de robotique produisent des produits de surveillance. L'une de ces compagnies est Robotex [10]. Robotex fabrique des robots ayant des capacités de téléopération. Leurs plateformes semblent viser les situations de déploiements en conditions d'urgence où le robot peut être utilisé pour établir un contact visuel à distance avec un intrus ou un danger. Leurs robots semblent être conçus pour leur robustesse et la capacité de surmonter une multitude d'obstacles. Pour cette raison, ils sont équipés de chenilles ajustables pouvant gravir des escaliers et d'autres types de terrain accidenté [11].
Atouts et avantages du Robotex AvatarIII :

- Construction robuste, faite pour les terrains accidentés, pouvant être utilisée autant à l'intérieur qu'à l'extérieur.
- Avatar III comprend une caméra PTZ qui peut être orientée vers une cible d'intérêt. Ce genre de technologie, si intégrée sur le SecurBot, permettrait de capturer des angles de vidéo intéressants.

Limitations du Robotex AvatarIII :

- Robotex semble offrir des solutions plus adaptées aux interventions tactiques dans les cas d'urgence. Ne possédant pas de mode autonome, ce robot téléopéré est moins utile pour la surveillance.

Cobalt

Cobalt Robotics [12] a mis au point un robot de surveillance autonome destiné au déploiement dans un environnement de bureau. Tout comme SecurBot, Cobalt est un robot autonome basé sur système ROS [13] [14]. De toutes les solutions présentées dans cette section, Cobalt est la plus similaire de par la gamme de capteurs intégrés et de par les fonctionnalités clé. Tout comme SecurBot, Cobalt possède une caméra RGBD qui lui permet de cartographier son environnement et y détecter des anomalies.
Atouts et avantages du robot Cobalt :

- Navigation autonome de son environnement.
- Cobalt peut détecter de façon automatique des environnements troubles ou des anomalies dans son milieu et lancer une alerte vers l'opérateur.
- Téléopération : Le robot cobalt peut être téléopéré et offre un lien vidéo bidirectionnel entre l'opérateur et les individus à proximité de Cobalt.

Limitations du robot Cobalt :

- Solution à source fermée

FIGURE 3 – Robot de sécurité Cobalt

Tableau 1 – Tableau récapitulatif des concurrents

Nom de la solution concurrente	Points forts ou à intégrer à SecurBot
CrossWing virtualMe et Bishop	+Téléopération et vidéoprésence +Mouvement omnidirectionnel +Escorte automatique
Knightscope	+Expérience de déploiement en milieu institutionnel. +Navigation automatique sous forme de trajectoire. +Application de gestion de sécurité intégrée avec interface graphique.
Segway Loomo	+Capcité de dédoubler en véhicule. +Robot rapide et robuste. +API pour en étendre la fonctionnalité au-delà de sa fonction primaire
Robotex Avatar III	+Capacité de gravir des obstacles comme des escaliers. +Caméra PTZ offrant plus d'angles de vue.
Cobalt	+Adapté pour opération en milieu de bureau. +Détection d'anomalies automatique +Patrouille automatique

2 Description du système

2.1 Cahier des charges

Lors des premières rencontres avec le client, une série de buts à atteindre pour le projet ont été identifiés. Le système livré devait :

— Pouvoir effectuer une ronde de sécurité pendant une heure sans intervention humaine ;

— Être capable de cartographier en 3D une zone prédéterminée en moins d'une heure ;

— Pouvoir identifier des situations problématiques pour la sécurité de l'environnement ;

— Permettre une communication bidirectionnelle entre l'opérateur et le robot à l'aide d'une interface graphique ;

— Afficher constamment les informations transmises par le robot et la réponse du robot lorsqu'une commande lui est envoyée.

Les indicateurs présentés au tableau 2 ont été établis pour valider l'atteinte des buts.

Tableau 2: Indicateurs de performance et moyens de validation

Indicateur objectivement vérifiable	Moyen de vérification
Toutes les composantes électroniques sont fermement fixées au robot et ce dernier doit pouvoir se déplacer de façon agile et stable.	Les composantes demeurent fixes lorsque le robot est placé en angle et lorsque celui-ci est soumis aux accélérations typiques de ses fonctions, qu'il peut effectuer sans tomber.
Le robot peut effectuer ses tâches pendant au moins une heure et être en mesure de se recharger en moins de deux heures.	Enregistrer avec le robot le temps d'autonomie et le temps de recharge de chaque ronde afin d'évaluer son autonomie et comment elle évolue dans le temps.
L'interface se connecte au robot et permet de voir la carte du bâtiment, de téléopérer le robot avec un flux vidéo et permet de planifier des trajets pour une future ronde de sécurité.	Il est possible de situer le robot sur la carte, le flux vidéo est d'au moins 20 images par secondes avec une latence maximale de 100 msec et le planificateur de trajet permet de créer et de planifier une ronde de sécurité.
Le robot et l'opérateur sont capables de se connecter au serveur. Ce dernier conserve un registre des alertes générées par le robot.	La connexion de l'opérateur au robot est fonctionnelle et le registre est accessible.
Le robot est en mesure d'identifier 70% des alertes sonores préprogrammées dans l'interface utilisateur (niveau sonore minimal de l'alerte de 50 dB avec des perturbations sonores de maximum 60 dB).	Le robot sera testé sur le même trajet qu'un humain avec des évènements simulés et les performances de chacun seront comparées.
Le robot est en mesure d'identifier 70% des évènements visuels rencontrés le long du trajet.	Le robot sera testé sur le même trajet qu'un humain avec des évènements simulés et les performances de chacun seront comparées.
Le robot planifie ses actions selon son niveau de charge et est en mesure de changer son mode contrôle selon la situation.	Le robot retourne à sa station de charge avant de manquer de batterie. Il est possible de le téléopérer pour ensuite le renvoyer en mode autonome.
Projet en ligne sur *GitHub*	Sont disponibles : -documentation -tests -Fichier de conception pour tous les extrants

L'implémentation logicielle de la reconnaissance d'évènements sonores a été écartée un peu après le début du projet sous la recommandation du client. Une autre équipe de projet travaillant déjà sur la reconnaissance sonore avec les outils du laboratoire, ce dernier a recommandé à l'équipe SecurBot de concentrer ses énergies sur d'autres tâches et de seulement prévoir l'intégration matérielle de la carte de son et des micros sur la

structure mécanique du robot. Ce requis a donc été éliminé.

2.2 Architecture globale

Pour répondre aux demandes du client, l'équipe SecurBot a conçu et fabriqué un système logiciel et matériel permettant de transformer une plateforme robotique abordable en robot de sécurité.

Plus spécifiquement, le rôle de l'équipe fut d'intégrer sur deux plateformes robotiques du laboratoire IntRoLab un système de contrôle, une structure mécanique pour les composantes essentielles du système (capteurs, électronique et autres) et un système d'alimentation. Ces modules devaient être interchangeables entre les deux plateformes robotiques fournies par le client, soient un TurtleBot et un Pionner 2. L'utilisation de ces deux plateformes devait permettre de démontrer la portabilité des technologies intégrées par l'équipe.

Les spécifications sur l'alimentation électrique des deux plateformes étant différentes, le système d'alimentation devait être suffisamment flexible pour fournir les alimentations requises. Le système conçu par l'équipe SecurBot comporte donc des régulateurs de tensions variées. Il est aussi en mesure de déterminer lorsque le robot doit aller se recharger. Enfin, l'équipe a fabriqué une station de charge compatible avec les deux plateformes robotiques pour assurer l'autonomie complète des robots.

Les autres modules représentent la partie de développement logiciel du projet et sont représentés à la figure 4.

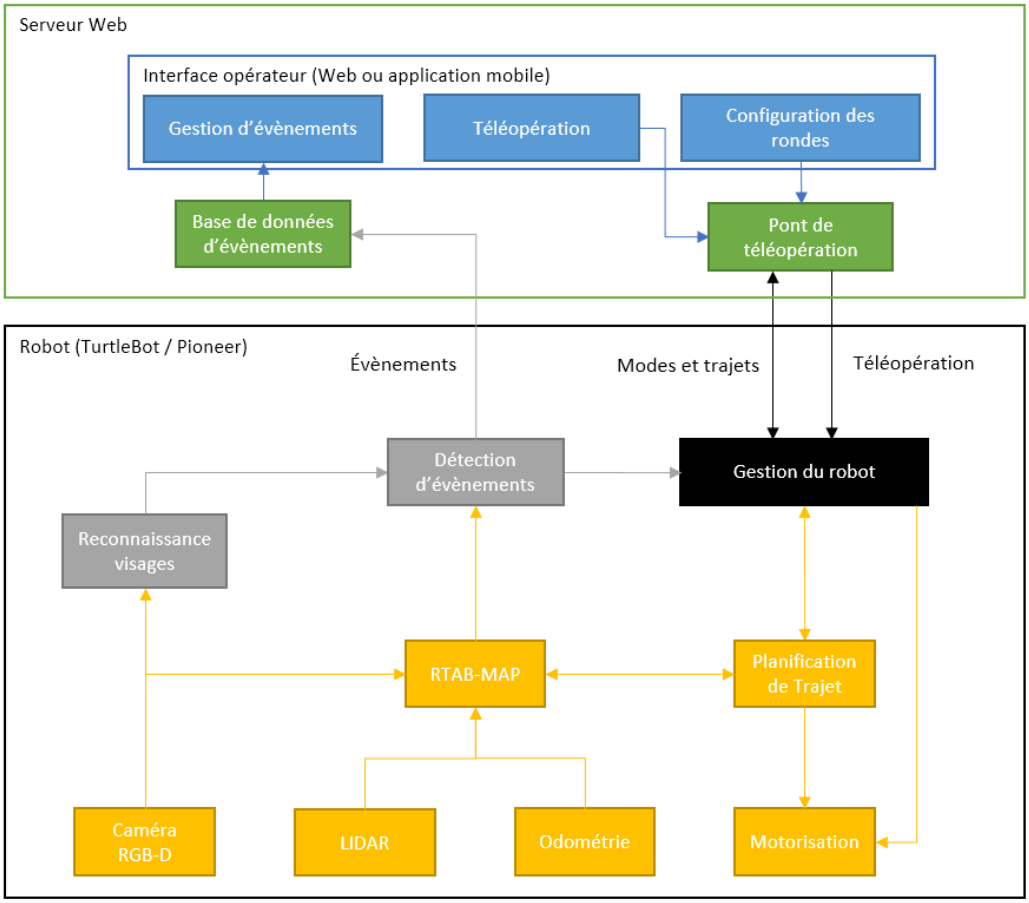

FIGURE 4 – Architecture globale logicielle du projet SecurBot

Afin que le robot se déplace de manière autonome, SecurBot a intégré un système de navigation et une architecture de gestion des modes d'opération. Le module de navigation, représenté en jaune dans le schéma, a pour objectif de permettre au robot de cartographie son environnement et de s'y déplacer de façon sécuritaire pour le public. Le module de gestion, représenté en noir, assure la communication entre le robot et le serveur et permet au robot de décider quelle action entreprendre.

Les fonctionnalités de sécurité du robot sont assurées par le module de détection d'évènements, représenté en gris dans le schéma. Ce module détecte les personnes dans son champ de vision et les enregistre dans sa carte comme étant des intrus.

L'équipe a aussi développé un serveur, représenté en vert, qui permet d'offrir à un utilisateur une interface de contrôle du robot et de faire le lien entre cette interface et le robot. Le serveur est aussi utilisé pour enregistrer les intrus trouvés par le robot. Cette interface utilisateur, représentée en bleu, permet à un utilisateur de téléopérer le robot, de planifier des patrouilles et de visionner les évènements détectés lors d'une patrouille.

2.3 Description des modules

2.3.1 Mécanique

Pour la structure mécanique, les technologies furent choisies en gardant en tête les philosophies "Open Source", donc une conception mécanique facilement transposable et accessible à un grand public. Il fut donc misé sur l'utilisation d'imprimante 3D et de découpeuse laser, deux outils relativement accessibles pour construire un prototype. Également, les plans mécaniques des armures et de la station de charge furent effectués sur SolidWorks, un logiciel de conception très utilisée dans le domaine de la robotique. L'équipe mécanique a misé sur une technique de conception par itération afin de pallier au manque d'expérience de l'équipe en conception mécanique. Grâce aux designs itératifs, les risques liés à l'intégration mécanique furent mitigés. Les pièces étant facilement modifiables dans SolidWorks apportaient un net avantage au processus de conception itératif. Aussi, les pièces modélisées peuvent être facilement intégrées dans un assemblage afin de diminuer les risques d'erreur de conception. De plus, le logiciel offrait des tutoriels gratuits et des vidéos éducatives sur leur site Web de façon abondante ce qui rendait l'apprentissage de ce logiciel moins risqué pour le projet.

Le matériel choisi pour faire les pièces imprimées en 3D est le PLA, soit le matériel de base utilisé généralement dans les imprimantes 3D. De plus, la précision obtenue par une imprimante 3D de qualité moyenne avec ce matériau était plus que satisfaisante pour obtenir de bons résultats. Outre le PLA, de l'acrylique fut aussi utilisé comme matériel de base pour les éléments mécaniques. L'acrylique est abordable, malléable et généralement compatible avec les découpeuses laser destinées au bois. Pour l'équipe, il fut facile de travailler avec ce matériel, car une découpeuse laser était disponible au Studio de création et au 3IT. Ces découpeuses laser offraient aussi une précision satisfaisante.

Comme mentionné plus haut, le processus de conception mécanique intégrait une portion itérative en prévision que l'équipe mécanique devrait faire plus d'un prototype pour certains modules. Ayant accès à une imprimante 3D, il était simple de produire divers prototypes et de les tester avant de choisir le modèle final. Le processus de conception mécanique fut le suivant : faire un design initial selon les requis, imprimer une pièce, vérifier si la pièce satisfait les critères, tester sa rigidité, faire les modifications dans le logiciel et recommencer le processus jusqu'à tant que la pièce soit satisfaisante. Ensuite, il s'agissait de faire l'intégration et l'ajustement des pièces si nécessaire.

Pour l'armure, les défis de conception étaient multiples, mais le plus important fut la conception d'une structure étant facile d'intégration sur diverses plateformes robotiques existantes. Pour ce faire, il était important d'ajouter comme critère de conception d'utiliser les bases robotiques sans les modifier, ce qui a évidemment ajouté un défi considérable de conception. La conception finale des armures permet une modification très simple des dimensions originales afin de l'adapter en hauteur, longueur et largeur. Il était aussi important de considérer l'intégration des éléments externes qui seraient ajoutés, soit les capteurs et les cartes électroniques. La position des capteurs, comme le Lidar et la caméra, avait des besoins particuliers relatifs à leurs champs de vision ou de mesure qui ne devaient pas être obstrués, nécessitant un questionnement sur la hauteur la plus judicieuse.

FIGURE 5 – Design de l'armure dans le logiciel SolidWorks

La seconde étape de la conception de l'armure était d'imprimer chaque type de pièce qui connecte les panneaux d'acrylique pour vérifier si leurs dimensions offrent un espacement pour la colle. Si leurs dimensions ne satisfaisaient pas cette contrainte, elles devaient être modifiées. La figure 5 permet de visualiser ces pièces en gris sur la figure. Il existe cinq types de pièces sur l'armure : Les pièces du bas du corps, les pièces du haut du corps, les pièces du bas de la tête, les pièces du haut de la tête et les équerres entre l'armure et la base du robot.

L'étape finale était d'assembler l'armure et de monter les contacts pour la recharge. Pour ce faire, tous les liens furent imprimés et les panneaux découpés. Les différents morceaux ont, par la suite, été collés ensemble avec de l'époxy à plastique. Les contacts sur les robots sont des morceaux d'aluminium machinés à la main. Les trous pour monter les contacts sur l'armure ont été pensés sous forme de fentes afin de pouvoir ajuster leur hauteur avec la station de recharge. Il y avait aussi les équerres qui contenaient le même type de fente pour ajuster l'armure à la base.

Pour la station de charge, les premiers éléments conçus furent les contacts. Comme le reste de la structure devait être simple, la priorité fut mise sur ceux-ci. Plusieurs itérations ont été réalisées avant d'obtenir un design fonctionnel avec des pièces solides. Ce design peut être observé à la figure 6.

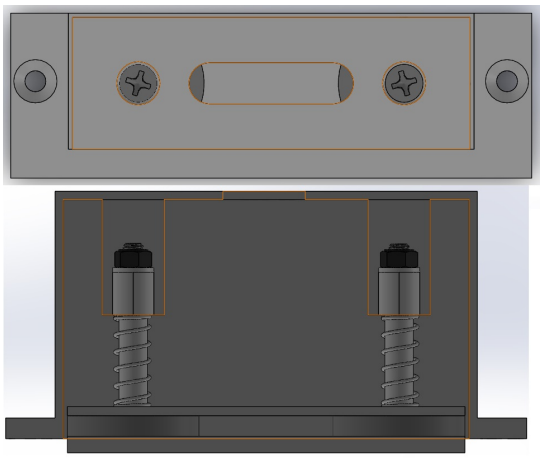

FIGURE 6 – Design de la boite de contact dans le logiciel SolidWorks (vue de devant et du dessus)

Après à la première itération, les principales modifications des contacts visaient la solidité, la force des aimants et la force des ressorts. Les aimants étant trop fort et les ressorts trop raides, ceux-ci étaient problématiques. Toutefois, après quelques ajustements, ces aspects furent grandement améliorés. Ensuite, comme certaines pièces de la boite de contact se cassaient facilement, il a été requis d'améliorer leur rigidité. Lorsque la dimension des pièces et leurs propriétés furent satisfaisantes, l'étape suivante fut l'assemblage des contacts. Suite à l'assemblage, la conception de la boite de la station de charge a débuté. Le design de la station s'est inspiré de l'armure avec l'utilisation de pièces imprimées en 3D servant à soutenir les morceaux d'acrylique. La figure 7 permet d'observer les différentes parties de la station de charge.

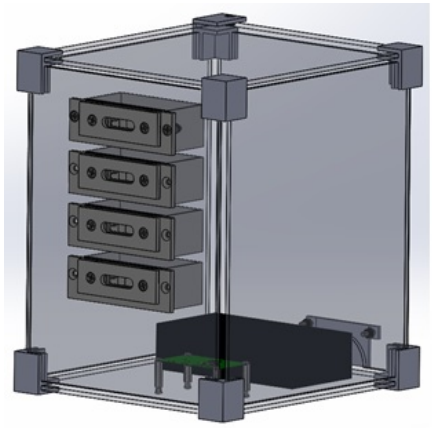

FIGURE 7 – Design de la station de charge dans le logiciel SolidWorks (vue isométrique)

Pour la partie mécanique, deux membres de l'équipe ont effectué la majeure partie de la conception en plus d'une personne pour aider comme regard externe et critiquer le travail effectué. La période de travail est répartie principalement sur la deuxième session. Le travail a été fait par Olivier Belval et Valérie Gauthier. Valérie a travaillé principalement sur l'armure et Olivier sur la station de charge et les contacts. Édouard Denommée fut la personne ressource pour le retour sur le travail de conception et l'opérateur expert pour les impressions de pièces 3D.

Pour valider la partie mécanique, la plupart des pièces furent testées au moment de leur acquisition. Cela signifie qu'avant de faire le collage permanent des pièces, un assemblage temporaire était fait pour permettre de valider le design et s'assurer que des modifications pouvaient facilement être apportées. Au niveau de l'armure, celle-ci remplit sa tâche de protéger les composantes en agissant comme barrière avec le monde extérieur en plus de supporter les capteurs. De plus, la rigidité de l'armure fut testée à maintes reprises, par exemple lors du développement logiciel où certains problèmes de navigation ont engendré des impacts et collisions ou encore lors de déplacement en situation réelle durant MégaGÉNIAL. Suite à ces évènements, il fut observé que l'armure tient toujours et qu'aucun dégât ne peut être observé sur celle-ci. De plus, grâce à SolidWorks, il fut possible de valider à même les dessins que l'armure serait en mesure d'intégrer les capteurs et permettrait de contenir les différentes cartes électroniques. Le tableau 3 contient les fonctionnalités et les moyens de vérification de la partie mécanique du projet. Il est à noter que les fonctionnalités incluent dans le tableau diffèrent de ceux du MAP, car la partie mécanique fut repensée en début de S8.

Tableau 3: Performances mécaniques

Fonctionnalités	Test	Conditions	Résultat Attendu	Résultats
Contenir les composantes du robot.	Les composantes sont stables lorsque le robot bouge.	Le robot effectuera des mouvements droits et tournera sur lui-même.	Les composantes seront dans un rayon de 5 cm de leur état initial.	Les composantes n'ont pas bougé ($<$ 1cm) lors du test.
Protéger les composantes.	Doit être capable d'endurer un choc léger.	Le robot recevra un choc léger de l'ordre d'une collision avec une chaise.	Aucune partie de la structure ne présente des signes de dommages intenses comme des fissures ou des morceaux manquants.	Aucun dommage critique visible après plusieurs chocs.
Les composants seront inaccessibles au public.	Le public ne doit pas être en mesure d'accéder aux composants des robots et de la station charge.	Le robot sera stationnaire et un membre de l'équipe tentera d'accéder aux composants sans retirer de morceaux/pièces ou détruire l'armure.	Il n'est pas possible d'accéder à l'intérieur de l'armure et station de charge.	Aucun membre n'a réussi à accéder à l'intérieur de l'armure et station de charge.

Tableau 3: Performances mécaniques

Fonctionnalités	Test	Conditions	Résultat Attendu	Résultats
Permettre aux composantes de réguler efficacement leur température.	Prendre une mesure de la température dans la coque du robot lors de son fonctionnement.	Après une heure de fonctionnement.	50 °C de température ambiante maximale.	La température à l'intérieur de l'armure était équivalente à la température ambiante de la pièce.
S'adapte à plusieurs robots	Faire une armure pour les deux robots	Les pièces qui tiennes les panneaux doivent être les mêmes. Seulement les dimensions des panneaux doivent changer.	L'armure est convenable pour les deux bases.	L'armure convient aux deux bases et les capteurs nécessaires peuvent être intégrés, peu importe la base.
La station de charge et l'armure doivent être compatibles pour la recharge.	Lors de la séquence de recharge, tous les contacts du robot arrivent sur ceux de la station de charge et un contact électrique est créé.	Lorsque la séquence de recharge est fonctionnelle	Les contacts du robot touchent ceux de la station de charge sans que le robot aille à pousser.	Le contact électrique se fait, cependant le robot doit pousser pour assurer un bon contact.

2.3.2 Système d'alimentation

Le système d'alimentation de SecurBot permet de fournir la tension requise à chaque appareil électronique du robot. La source d'énergie est une batterie, que le robot est en mesure de charger automatiquement en se stationnant à sa borne de recharge. Il est composé de trois cartes électroniques : la carte de charge, la carte de batteries et la carte d'alimentation. Cette architecture est présentée à la figure 8. La portion sur le robot robot a été séparée en 2 circuits imprimés pour avoir une plus grande modularité au cas où les capteurs devaient changer ou si un plus grand module de batterie était nécessaire. Cela simplifiait aussi les requis et facilitait la gestion du travail avec l'horaire des deux sessions.

La station de charge prend en entrée du 120 VAC et fourni une tension de 19.5 VDC ayant une puissance

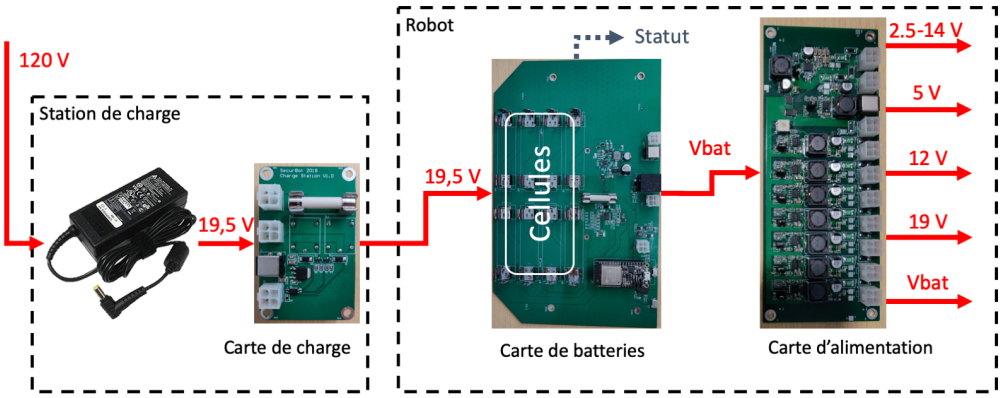

FIGURE 8 – Architecture matérielle de SecurBot

maximale de 180 W. Il fut décidé de prendre un bloc d'alimentation déjà sur le marché puisque celui-ci était déjà approuvé CSA. Cela a permis d'économiser beaucoup de temps à l'équipe de conception électrique. De plus, il fut choisi de concevoir un circuit imprimé pour créer une interface électrique entre le bloc d'alimentation et les contacts métalliques de la station de charge afin de rendre la station plus sécuritaire. Ce circuit connecte les contacts de charge seulement lorsque le robot applique une tension sur les contacts de sécurité. Il possède un photo-coupleur activé par le 5 V provenant du robot qui active la sortie du 19,5 V pour permettre la recharge. Ce circuit possède aussi un régulateur de tension 5 V afin d'alimenter les transistors. Le schéma de ce circuit peut être trouvé dans la figure 9.

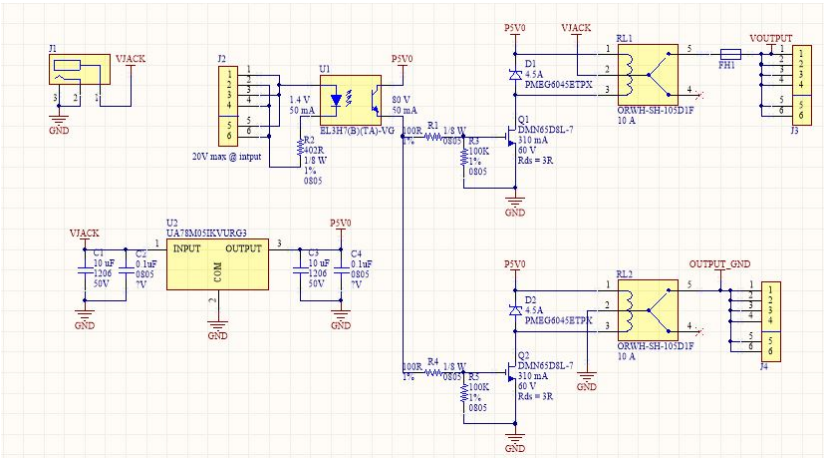

FIGURE 9 – Schéma du circuit de la station de charge

La carte de batteries possède deux système majeurs : le système de surveillance des cellules réalisé avec un BQ76925PWR et le système de chargement réalisé avec un BQ24725A. Ensembles, ils permettent de déterminer le niveau de charge actuel et d'estimer l'autonomie restante. Ces deux circuits intégrés ont été sélectionnés parce que ceux-ci étaient compacts, respectaient le budget d'alimentation et étaient capables de gérer la quantité de cellules prévues, soit quatre en séries et deux en parallèles pour un total de huit cellules. Le nombre de cellule a été choisi pour être capable de fournir une alimentation stable de 14,25 V avec une capacité de 7 Ah. L'équipe SecurBot a aussi décidé que la carte de batteries accueillerait les cellules à même le circuit imprimé. Cette décision a été prise afin de ne pas avoir à souder les cellules ensembles ce

qui représentait un risque de sécurité important et un long travail pour préparer les procédures nécessaires.

L'équipe SecurBot a utilisé un ESP32 pour communiquer avec les différents circuit intégrés. Ce microcontroleur a été choisi pour son faible coût et sa simplicité de programmation. En effet, il possède un programmeur intégré et un membre de l'équipe avait déjà une bonne expérience avec la programmation d'un micrologiciel de surveillance sur ESP32. De plus, ce microcontroleur était un *System On Chip* ce qui le rendait très simple à intégrer.

Le système de surveillance permet de mesurer la tension, la température et le courant de chaque cellule. Ces données sont ensuite envoyé vers le ESP32 pour être interprétées. Le schéma de ce circuit est présenté à la figure 10. Pour contrôler le chargement des cellules et gérer les alimentations lorsque le robot est en charge, l'équipe a choisi un BQ24725A. Celui-ci est en mesure de couper l'alimentation en cas de problème. De plus, une commande faite par le ESP32 permet de déconnecter les cellules du circuit en cas de problème. Le schéma de ce circuit est présenté à la figure 11.

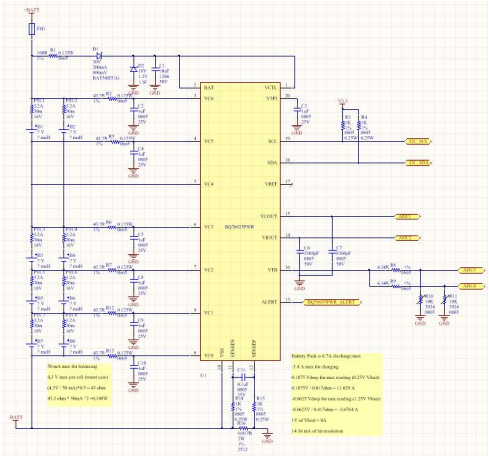

FIGURE 10 – Schéma du circuit pour les BQ76925PWR

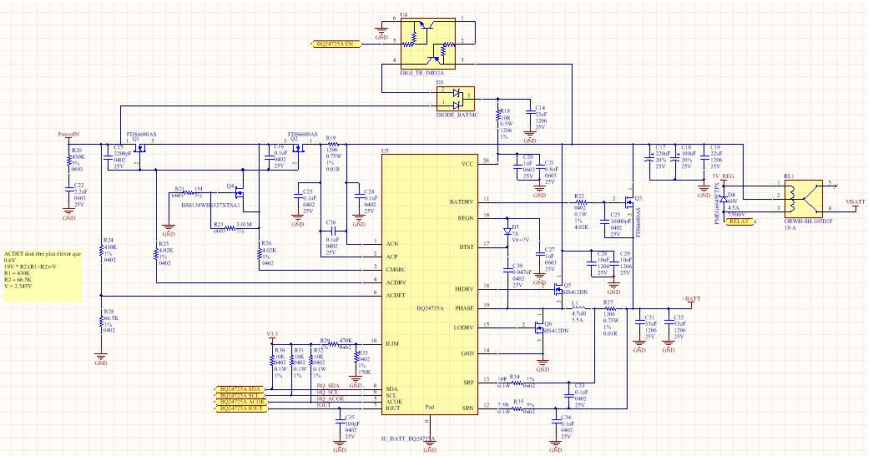

FIGURE 11 – Schéma du circuit du chargeur de batterie

Pour le PCB de distribution d'alimentation, le circuit intégré RT8284N à été choisi pour faire les conversions

de tension parce que celui-ci pouvait créer toutes les tensions nécessaires pour les besoins des robots réduisant le temps de développement des schémas et de l'implémentation physique pour l'équipe électrique. L'arbre d'alimentation possède 8 RT8284N pour créer 4 x 5 V et 2 x 12 V et un 15 V pour les différents capteurs, un 19 V afin d'alimenter le Jetson TX2 et une alimentation variable de 2,5 à 14 V afin d'alimenter les différentes bases robotiques. Les schémas des alimentations avec le RT8284N et le buck-boost 19 V sont présentées dans les figures 12 et 13.

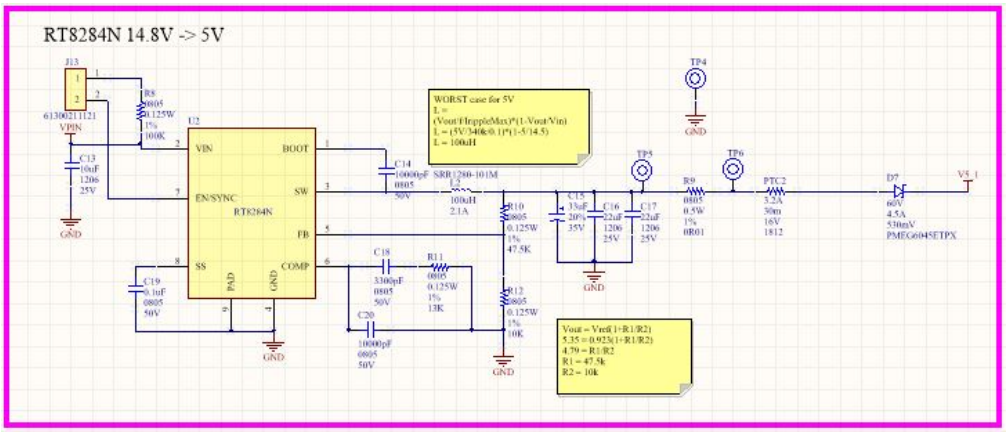

FIGURE 12 – Schéma du circuit pour les RT8284N

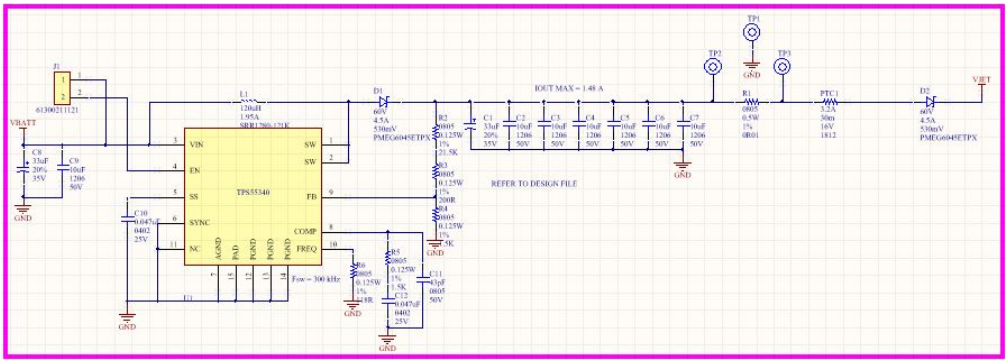

FIGURE 13 – Schéma du circuit du Buck-Boost 19V

Afin de réussir à concevoir, assembler et tester ces trois circuits, l'équipe SecurBot a décidé d'avoir des échelons précis pour les différents circuits étalés sur les deux sessions et planifiés de façon à avoir le temps de faire une deuxième itération des circuits imprimés si nécessaire. La première étape de conception était de déterminer les requis des différents circuits et de trouver les composants ayant les spécifications nécessaires. Si la hiérarchie du circuit à faire n'était pas familière aux membres de l'équipe, celle-ci faisait de la recherche pour considérer les composants existants capable de rencontrer les spécifications. Lorsque les pièces étaient sélectionnées, les fiches techniques de chacun des circuits intégrés étaient consultées pour savoir comment implémenter ces composants. Après cette étape, l'équipe préparait les schémas électriques du circuit qui étaient ensuite revus par un autre membre de l'équipe pour éviter les erreurs. Une fois le circuit révisé, la disposition de la carte était élaborée. Une fois terminée, la disposition des composants était inspectée une seconde fois par un autre membre de l'équipe puis envoyée en fabrication. Après avoir reçu le circuit imprimé, les pièces étaient soudées

sur celui-ci. Le circuit était ensuite testé dans un local approprié en suivant un protocole de test approuvé. Si des erreurs étaient présentes, elles étaient corrigées physiquement et les modifications étaient ajoutées sur les schéma et la disposition. Cette démarche a été utilisée pour les trois circuits imprimés.

Afin de faire la conception électrique de SecurBot, Philippe Marcoux et Olivier Belval ont été assigné à faire l'arbre d'alimentation, la conception schématique et la disposition initiale du circuit de batterie lors de la première session. Lors de la deuxième session, Philippe a terminé le circuit de batterie et Olivier a conçu le circuit pour la station de charge. Cédric Godin s'est chargé du développement du micrologiciel de gestion des batteries qui s'exécute sur le microcontroleur ESP32. Ils ont par la suite procédé aux tests du circuit de batterie et à son déverminage.

Le tableau 4 illustre les différents tests effectués sur les circuits de SecurBot et leurs résultats. Les protocoles de tests peuvent être retrouvés dans l'archive.

Tableau 4: Performances électroniques

Fonctionnalités	Test	Conditions	Résultats attendus	Résultats
Fournir 19.5V 3A a SecurBot.	Mesurer la tension et le courant généré.	À l'aide d'une charge de test représentant l'impédance d'entrée du circuit, il faut mesurer le courant/la puissance fournie.	19 ± 0.5 V 3 ± 0.3 A	À l'aide d'une résistance 5ohm, la tension était de 19.5V et 3.9A
Mesurer la dissipation de chaleur.	Avec une charge de test qui représente l'impédance d'entrée de la recharge et une source de potentiel qui simule la batterie, il faut mesuré avec une caméra thermique la température des composantes	Chargement à 19.5V - 2A durant 2h	T<=70 ± 1 °C 19.5 ± 0.5 V 2 ± 0.1 A	La température durant l'heure de charge n'a jamais excédé 45°C
Recharger la batterie en 2 h.	Mesurer le temps de chargement .	Avec une entré de tension directe vers dans le chargeur de batterie, chronométrer le temps de recharge.	2 h ± 10 min	Moyenne de 3h pour chaque charge avec le balancement
Être capable d'arrêter le chargement.	Rendre les contacts hors tension à la réception d'un signal. Déconnecter le robot de la station de Charge et vérifier que les contacts sont hors tension.	Connecter le robot, vérifier que celui-ci se recharge puis le déconnecter et vérifier que les contacts sont hors tension	Les contacts doivent être mis hors tension 99 % du temps.	Les contacts ont été hors tensions 100% du temps
Fournir 5V 1.5A au LIDAR	Vérifier la tension fournie par le régulateur.	Vérifier à l'aide d'un Voltmètre la tension lue sur les différents rails d'alimentation lorsque les périphériques sont branchés.	La tension doit être de 5 ± 0.3 V	La tension était entre 4.73V et 5.17 V avec un courant de 1.56 A
Fournir 12V 1.5A Aux.	Vérifier la tension fournie par le régulateur.	Vérifier à l'aide d'un Voltmètre la tension lue sur les différents rails d'alimentation lorsque les périphériques sont branchés.	La tension doit être de 12 ± 0.3 V	La tension était entre 12.75V et 12.20 V avec un courant de 1.59 A
Fournir 18.5V 1.6A à l'ordinateur embarqué.	Vérifier la tension fournie par le régulateur.	Vérifier à l'aide d'un Voltmètre la tension lue sur les différents rails d'alimentation lorsque les périphériques sont branchés.	La tension doit être de 18.5 ± 0.3 V	La tension était entre 18.45V et 18.85 V avec un courant de 1.74 A

Tableau 4: Performances électroniques

Fonctionnalités	Test	Conditions	Résultats attendus	Résultats
Alimenter les modules du robot pour une ronde de 1 h.	Calculer l'autonomie du robot.	Faire fonctionner le robot avec des tâches standards pendant le plus longtemps possible jusqu'à ce que la batterie nécessite une recharge.	Durée de 1 h ± 15 min	Ce test n'a pas été effectué, mais devrait être environ 1.5h selon les calculs
Fournir de la puissance aux composantes même lors de la recharge.	Vérifier la puissance fournie lors de la recharge.	Vérifier à l'aide d'un voltmètre et d'un ampèremètre la puissance en sortie de chacun des rails avec des charges tests lorsque les batteries sont en chargement.	La puissance doit être de l'ordre de ± 0.5 W	Le PCB est capable de fournir 19V 2A au robot lors du chargement
Être capable de balancer les cellules lors de la charge	Vérifier la tension à chaque cellule lors de la recharge	Charger les cellules à 19V 1A et laisser l'algorithme balancer les cellules jusqu'à la charge complète et mesurer chaque cellule à l'aide d'un voltmètre et du BQ76925PWR	La différence de tension entre chaque cellule doit être plus petite que 0.1V	La différence de tension entre chaque cellule est plus petite que 0.1V

Malheureusement, puisque la carte de batteries a nécessité un mois de déverminage, le système d'alimentation n'a pas pu être intégré sur les robots. Le bon fonctionnement de chaque carte a tout de même été confirmé en laboratoire. Certaines modifications pourraient être faites au niveau du circuit de batterie afin de le rendre plus modulaire. En effet, avec une conception où les batteries ne seraient pas directement sur le circuit imprimé, il serait possible de réduire le coût du circuit et de permettre une plus grande modularité au niveau des cellules.

2.3.3 Interface utilisateur

En ce qui a trait à l'interface utilisateur, celle-ci est une application Web bâtie à l'aide du *Framework* JavaScript Vue. L'architecture Web fut choisie pour obtenir une application multiplateforme facilitant ainsi son utilisation et accessibilité pour les utilisateurs. Vue est un *Framework* à popularité grandissante qui, comme ses principaux compétiteurs, est conçu pour être développé et intégré de façon continue et itérative. Ce dernier se distingue toutefois de ses compétiteurs par sa facilité à prendre en main et à intégrer à d'autres bibliothèques en plus d'être très flexible dans son implémentation. Utilisant la méthode de gestion agile et en prenant en considération que plusieurs membres de l'équipe n'avaient jamais programmé pour le web, Vue fut un choix simple à faire. En concert avec Vue, plusieurs autres bibliothèques et outils furent utilisés pour créer l'interface utilisateur. L'interface EasyRTC fut utilisée pour intégrer le protocole WebRTC permettant le transfert de flux vidéo et la mise en place de canal d'échange de données entre utilisateurs et robots. La boîte à d'outils (*toolkit*) Bootstrap/Bootstrap-Vue fut utilisée pour uniformiser le style de l'interface utilisateur et ses composants. Deux autres bibliothèques furent utilisées pour le développement soit Vue-Router, pour la navigation entre pages et composants, et Vuex, un outil agissant comme banque de données centralisée à l'intérieur de l'application. Un autre outil appelé Webpack, qui est intégré à Vue, permettait de compiler et minimiser l'interface pour un environnement de production se prêtant à une architecture par services.

Suivant le principe de développement par itération, les fonctionnalités de l'interface utilisateur furent développées selon les besoins de l'équipe et les désirs du client. L'objectif étant de simplifier l'utilisation des robots à un utilisateur, l'interface comporte quatre 'pages' pour couvrir tous les besoins. La première page est celle de téléopération. Elle permet de visualiser le flux vidéo de la caméra du robot et de sa carte, de téléopérer

le robot à l'aide des touches du clavier (flèches ou *WASD*) ou d'un *joystick* affiché à l'écran, de donner un objectif sur la carte (*goto*) et de commander au robot d'aller se recharger. L'interface, essayant aussi d'être le plus conviviale possible pour un utilisateur, offre des options de zoom pour la carte, un contrôle de la vitesse pour la téléopération et d'autres options d'affichage pour les flux vidéo.

Les deuxième et troisième pages fonctionnent conjointement pour la planification des patrouilles. L'une des pages (planification) permet d'ajouter des points sur la carte pour créer un parcours que le robot devra accomplir pour compléter une patrouille. Les points peuvent être nommés, retirés et réorganisés selon les besoins de l'utilisateur. L'autre page (configuration) permet de nommer et décrire une patrouille pour ensuite la sauvegarder sur la base de données permettant ainsi de la réutiliser. La patrouille peut aussi être envoyée directement au robot, être réinitialisée et même supprimée de la base de données. Lorsqu'une patrouille est enregistrée sur la base de données, un horaire peut lui être attribué. Les horaires permettent à un utilisateur de programmer quand le robot doit accomplir une patrouille à l'aide du format cron. Un utilisateur a donc l'option de programmer un horaire pour chaque heure, jour, semaine, mois ou année. L'utilisateur peut aussi programmer le nombre de fois que la patrouille doit être répétée pour compléter un horaire et donner un temps limite à l'achèvement, sans quoi le nombre supplémentaire d'exécutions seront ignorées.

La dernière page permet de filtrer et visualiser les évènements émis par les robots sur la base de données. L'utilisateur peut filtrer les évènements affichés par robot, par date (avant et après), par l'inclusion et exclusion d'étiquette, par recherche textuelle personnalisée, par notification et par nouvel évènement seulement. Plusieurs filtres prédéfinis sont aussi présents. L'affichage des évènements se fait à l'intérieur d'un tableau qui peut être défilé et trié selon certaines des informations affichées. Seulement les informations utiles à un utilisateur sont montrées, soit la date, le robot, l'objet/type, le contexte, la description, les étiquettes et image prise de l'évènement. La plupart des informations sont montrées à même le tableau, sauf la description et l'image. Dans le cas des descriptions, comme celles-ci peuvent être longues, un bouton indiquant si une description est disponible les remplace. Si une description est disponible, un clic sur le bouton de description fait afficher une fenêtre sous la rangée de l'évènement qui affiche la description. Dans le cas des images, pour faciliter leur visualisation, celles-ci sont aussi remplacées par des boutons. Dans le cas des images, toutefois, cliquer sur le bouton fait afficher l'image en premier plan sur la page web avec un fond flou, cliquer sur le '**X**' en haut à droite de l'image ou dans le flou fait disparaître l'image et remet l'accent sur la page Web contenant le tableau.

Pour fournir un contexte du *Framework* Vue, la description suivante peut être obtenue sur leur site Web :

> **Une application Vue se compose d'une instance de racine Vue créée avec une *new Vue*, éventuellement organisée en une arborescence de composants réutilisables imbriqués.**

Toutes les pages furent développées en suivant cette description soit une instance racine à laquelle plusieurs composants sont imbriqués les uns dans les autres pour former un tout représentant l'interface. Plus en détail, le principe de composant à fichier unique (Single File Component) fut utilisé. Ce principe consiste à regrouper tous les éléments d'un composant (HTML, JavaScript, CSS) dans un seul fichier agissant comme une classe/objet réutilisable et indépendant. Ce principe d'indépendance signifie qu'un composant nécessite seulement ce qui est contenu dans son fichier pour fonctionner correctement. Cet aspect permet d'obtenir une interface utilisateur très modulaire.

Toutefois, cette indépendance engendre aussi le fait que, sauf en utilisant des méthodes déconseillées, les données d'un composant sont uniques à celui-ci et inaccessibles par les autres. Bien que l'interface suit le

principe que les données d'un composant doivent être remplaçables (*stateless*), il devient inconvénient de perdre les états de ceux-ci lorsqu'un composant est retiré du *Document Object Model* (DOM), ce qui arrive lorsqu'un utilisateur navigue dans l'interface. Pour pallier à ce problème, et plusieurs autres, Vuex fut utilisé. Vuex est un modèle de gestion d'états et bibliothèques (*datastore*) qui est injecté à tous les composants de l'application. Ce modèle permet de centraliser dans un même endroit tous les états nécessaires à l'application, tout en permettant aux composants de garder des états locaux. La méthode de gestion des états implémentée par Vuex est simple, mais rigoureuse. La méthode rend tous les états contenus dans sa banque disponible en lecture seulement, pour modifier un état, celle-ci doit être faite à l'intérieur de la banque à l'aide de concepts nommés mutations et actions. Une mutation est synchrone et ne devrait modifier qu'un seul état, alors qu'une action est asynchrone et peut utiliser les mutations pour modifier plusieurs états. Cette méthode 'garantit' l'intégrité des données de la banque en faisant prendre conscience à tous les composants qui utilisent un état que celui-ci a été modifié. En utilisant les outils de développeur fourni par Vue, cette méthode permet aussi de garder un historique des changements encourus lors de l'exécution de l'application ce qui est très utile pour déverminer cette dernière.

De plus, Vuex supporte l'intégration de module représentant une sous-banque à l'intérieur de la banque racine. Cet aspect permet de rendre l'application encore plus modulaire. En concentrant l'utilisation d'une bibliothèque à l'intérieur d'un module, ce dernier peut facilement être remplacé par un module semblable employant une autre bibliothèque.

Plus de la moitié des membres de l'équipe ont participé au développement de l'interface utilisateur, mais selon les périodes le nombre de personnes travaillant simultanément sur celle-ci varie. Au début de S7, la moitié des membres de l'équipe participaient au développement, Édouard Légaré, Valérie Gauthier et Anthony Parris, mais par manque de ressources sur d'autres éléments du projet ce nombre a diminué à un seul membre, Édouard Légaré, vers la fin de la S7. Pour la majeure partie de la S8, cette même personne travaillait sur l'interface pour répondre aux différentes demandes du client et améliorer plusieurs aspects de l'interface. Vers la fin de la S8, une seconde personne, Cédric Godin, a participé au développement.

Bien que des bogues soient encore présents dans l'interface, ceux-ci n'empêchent et ne dérangent pas son utilisation. De ce fait, en fin de S8, l'interface utilisateur est totalement fonctionnelle et couvre toutes les fonctionnalités désirées. De plus, celle-ci a été testée extensivement durant les deux jours de l'exposition MégaGÉNIALE sans problèmes et est donc considérée fiable. Le seul problème d'importance à relever est la latence dans les flux vidéo, un problème essentiellement hors du contrôle de l'interface. Ce problème provient de deux sources :

- Deux flux vidéo sont transférés en **Wi-Fi** des robots.
- EasyRTC n'a pas été mise à jour récemment et son implémentation de WebRTC est donc presque désuète par rapport aux nouvelles versions des navigateurs Web ce qui rend son utilisation problématique.

2.3.4 Serveur

Une application de sécurité comme SecurBot demande l'archivage des données collectées par les capteurs du robot. En effet, que ce soit pour faire la revue d'évènements ou pour auditer des périodes critiques, SecurBot a besoin d'une centralisation de ses données. Afin de sauvegarder ces données, une base de données centrale à l'extérieur des unités de patrouille est de mise. De plus, puisque l'interface utilisateur de SecurBot est une application Web, il fut nécessaire de penser à une architecture de serveur permettant de répondre aux besoins

de services en lignes des robots et des utilisateurs. Cette section portera donc sur les opérations "Serveur" SecurBot.

Après avoir identifié deux services essentiels au fonctionnement de SecurBot (desservir l'application Web et archiver les évènements), il fut nécessaire de choisir un type d'infrastructure approprié à cette réalisation. Pour desservir l'application, le choix a été fait d'utiliser un serveur HTTP générique. Les deux candidats ayant été retenus dans cette catégorie ont été Apache et Nginx. Apache fut le premier choix, car les membres de l'équipe avaient de l'expérience avec cette plateforme et qu'il s'agit d'un standard industriel depuis plusieurs années. Cependant, en poussant les recherches vers la fin de la S7, la décision a été prise de passer à Nginx qui possède de meilleures performances, des fichiers de configuration plus lisibles, un développement plus actif et une communauté grandissante. Une fonctionnalité additionnelle du serveur HTTP est de servir de *proxy*-inverse. Ce dernier permet de se connecter à toutes les ressources de SecurBot sous un même port au domaine de l'équipe. Le trafic destiné aux autres services comme la base de données est redistribué par le *proxy*-inverse au port correspondant.

Pour ce qui est de la base de données, le plus grand choix technique fut de décider entre une structure par table ou par document. Pour les besoins de SecurBot, les structures par document semblaient plus appropriées. Un des avantages du modèle par document est qu'il laisse une plus grande marge de manoeuvre pour ce qui est de changements à apporter au modèle de données après la conception initiale. Lors de la conception de la base de données, tous les champs qui seraient nécessaires a l'opération de SecurBot n'étaient pas encore définis. Cette flexibilité a permis de les adapter plus rapidement quand des requis de bases de données inattendus sont survenus. En ayant défini le type de base de données, MongoDB a été retenu comme meilleur candidat. Il s'agit de la base de données par document la plus populaire dans l'industrie et avec laquelle les membres de l'équipe ont de l'expérience.

Afin de mieux restreindre les accès à la base de données et en contrôler l'utilisation, un service REST a été créé afin de servir de pont vers la base de données. C'est grâce à cette interface que l'on définit clairement les ressources disponibles à la base de données et à l'interface utilisateur. Dans la documentation de cette interface, les points d'accès ainsi que les modèles de données des documents de la base de données sont énumérés. Cette interface aide également à mieux implémenter du contrôle d'accès plus spécifique. Par exemple, les robots sont autorisés à ajouter des évènements à la collection, mais ne sont pas autorisés à en enlever. Ce genre de contrôle permet de mieux restreindre le flot des données sensibles.

Le trafic vidéo provenant des robots vers l'interface client doit également être routé à travers du serveur. Pour ce faire, le serveur doit négocier la connexion vidéo entre les clients. Puisque l'interface avec WebRTC utilisée par l'interface utilisateur est EasyRTC, ce serveur de communication vidéo est tout simplement une instance d'un serveur EasyRTC. Avec tous ces services en opération pour faire fonctionner SecurBot, une architecture par service a été choisie pour l'aspect serveur du projet. Nginx, MongDB, l'interface REST, et le serveur EasyRTC s'exécutent donc tous dans des conteneurs Docker afin de pouvoir gérer une future infrastructure infonuagique.

Le développement des requis nécessaires au serveur s'est fait au fur et à mesure du projet. Par exemple, lorsque le besoin de desservir une interface utilisateur s'est présenté, il fut nécessaire d'explorer les options possibles pour supporter ce besoin. Cette approche permit de s'assurer d'une intégration en continu et laisse le moins de surprises. Le serveur a requis le travail intermittent de deux personnes dans le projet. Deux personne, Édouard Denommée et Édouard Légaré, se sont impliqués dans cette partie du projet à cause de

leurs connaissances préalables. En termes de charge de travail, une personne plus dédiée aurait été amplement suffisante.

Pour conclure, les performances du serveur que l'on visait à concevoir initialement furent évaluées. Les résultats des test sont présentés dans le tableau 5.

Tableau 5: Tests du serveur

Fonctionnalités	Test	Conditions	Résultat Attendu	Résultat obtenu
Relayer l'information entre le robot et l'interface web.	Vérifier que le système produit une carte de l'environnement du robot.	Planifier une ronde de sécurité qui traverse une zone où la réception wifi est mauvaise.	La ronde se complète comme prévu.	Interruption de la ronde.
Stocker les évènements et les trajets.	Les évènements et les points de cheminements sont stockés dans la base de données.	Soumettre des évènements et des points de cheminement avec des requêtes de test, puis valider l'enregistrement avec d'autres requêtes.	Les requêtes sont complétées en moins de 0,1 s.	$<$ 200 ms

2.3.5 Système de navigation

Deux technologies principales sont utilisées pour le système de navigation. D'abord, RTAB-Map permet de produire la carte de l'environnement du robot à partir de ses capteurs et d'obtenir la position du robot dans la carte générée. C'était un choix évident puisque RTAB-Map, en plus d'être développé par le client, représente l'état de l'art de la cartographie et de la localisation simultanée dans ROS ce qui assurait à l'équipe un excellent support. Ensuite, la pile de navigation ROS est utilisée pour la navigation, qui consiste à calculer la vélocité que le robot doit avoir pour atteindre une coordonnée donnée sur la carte. Les planificateurs par défaut, soient NavfnROS et *Base Local Planner*, ont été conservés. Cela permettait une performance adéquate, étant donné la cinématique simple des bases robotiques, et maximisait la documentation disponible. La figure 14 présente l'architecture du système de navigation de SecurBot. Les rectangles aux coins arrondis sont les modules, tandis que les rectangles aux coins carrés sont les informations qui transitent par le lien illustré.

La conception s'est déroulée en quatre étapes. La première étape était l'intégration matérielle des plateformes robotiques. L'ordinateur de bord, les capteurs et le contrôleur moteur devaient être installés et connectés sur chaque robot. Cette étape s'accompagnait également de tests unitaires du fonctionnement de chaque élément dans ROS. C'est à ce moment que le noeud ROS de chaque composant était identifié, installé et testé. La deuxième étape consistait à compiler RTAB-Map sur l'ordinateur de bord et à le configurer pour utiliser les mesures en provenance des capteurs du robot. À ce moment, le robot a été piloté en mode téléopéré avec la télécommande clavier de ROS et la qualité de la carte et de la localisation a été vérifiée. La configuration de la pile de navigation ROS s'est faite lors de la troisième étape. En plus de configurer les capteurs à utiliser, il était nécessaire de spécifier les vitesses admissibles de la base robotique ainsi que l'importance à accorder aux

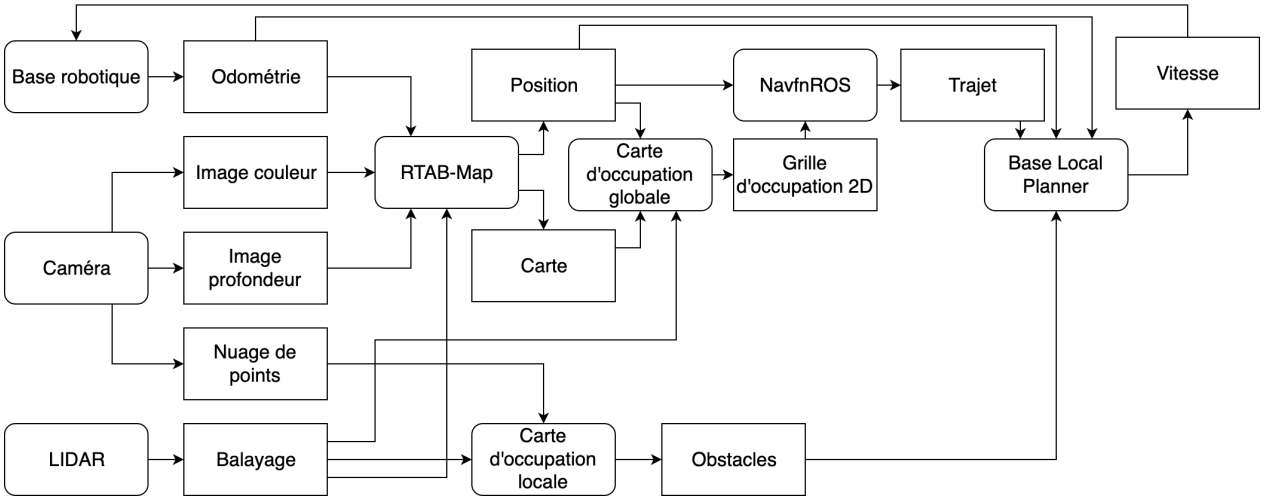

FIGURE 14 – Architecture du système de navigation

différents termes de la fonction de coût du planificateur de trajet. Le robot était alors capable de rejoindre un point spécifié sur sa carte de manière autonome. La quatrième étape était celle des tests et de l'ajustement des paramètres. Le robot a été soumis à diverses conditions et un ensemble de paramètres idéaux a été sélectionné en fonction des observations effectuées lors des essais.

La navigation a occupé deux membres de l'équipe, Alexandre Guilbault et Cédric Godin, à temps plein pendant la première session pour l'implémentation et la configuration initiale. Elle a également nécessité l'implication à temps partiel des deux mêmes membres pendant la deuxième session pour la correction des problèmes et l'ajustement des paramètres. Du côté client, l'équipe pouvait compter sur le support de Mathieu Labbé, développeur de RTAB-Map, pour la configuration et l'optimisation de RTAB-Map.

Le fonctionnement du système de navigation a été validé à l'aide des tests qui avaient été identifiés lors de la rédaction du MAP. Ces tests et leur résultat sont présentés dans le tableau 6. Toutes les métriques de performance ont été atteintes ou dépassées.

Une évaluation qualitative en continu a également permis d'optimiser les paramètres de la pile de navigation ROS afin que les déplacements du robot soient fluides et naturels. Les principaux critères d'observation étaient : l'atteinte de l'objectif ; la vitesse de déplacement ; le positionnement du robot par rapport aux obstacles ; l'oscillation perçue du robot lorsque l'espace est suffisant pour avancer en ligne droite. Les paramètres étaient également ajustés pour que ni le taux de rafraîchissement de la carte, ni de la commande, ne descendent sous le seuil spécifié dans la configuration de la pile de navigation lors de l'opération normale du robot. Afin de confirmer l'atteinte des taux de rafraîchissement voulus, la pile de navigation ROS affiche un message d'erreur chaque fois qu'une mise à jour prend trop de temps.

Tableau 6: Tests du système de navigation

Fonctionnalités	Test	Conditions	Résultat Attendu	Résultat obtenu
Générer la carte 3D de l'environnement.	Vérifier que le système produit une carte de l'environnement du robot.	Piloter le robot de manière à passer une fois à chaque endroit de la zone d'intérêt. Confirmer que la carte produite soit adéquate.	Un objet de la taille d'un extincteur est identifiable sur la carte. Le plan des corridors est reconnaissable.	Les pattes de table sont identifiables sur la carte. Le plan des corridors correspond au plan de l'étage.
Suivre un trajet programmé.	Valider que le robot puisse suivre un trajet programmé.	Télécharger une liste de points de cheminement sur le robot et démarrer le trajet. Confirmer que le robot suit les points de cheminement.	100 % des points de cheminement sont rejoints dans un rayon de 1 m.	100 % des points de cheminement sont rejoints dans un rayon de 0,2 m.
Planifier le trajet jusqu'à un emplacement.	Vérifier que le système de navigation trouve un chemin vers un objectif.	Commander une destination sur la carte au robot. Confirmer que l'algorithme de planification de trajet trouve un trajet si un trajet est possible.	L'algorithme trouve un trajet. Le robot ne bouge pas si aucune destination n'est trouvée.	L'algorithme trouve un trajet s'il y a au moins 20 cm d'espace de chaque côté du robot. Le robot tourne trois fois sur lui-même puis s'arrête si aucun trajet n'est trouvé.

Le système de navigation devait également être en mesure de stationner le robot à sa station de charge. Avec les spécifications de la station de charge définies par l'équipe mécanique, les tolérances sur la position du robot au moment du chargement sont limitées à une dizaine de centimètres dans l'axe des y (gauche à droite), 2 centimètres dans l'axe des x(avant et arrière) et une vingtaine de degrés en rotation. Afin de se localiser, le robot utilise principalement RTAB-Map. Cependant, la précision donnée par RTAB-Map relativement à la station n'est pas suffisante pour effectuer des arrimages automatiques. Pour cette raison, la station de charge a été équipée d'un marqueur *Apriltag*. Ces symboles carrés permettent à la caméra frontale des robots de positionner la station en 3D avec une précision suffisante. Ces marqueurs sont optimisés pour leur facilité de détection et la localisation précise de leur système d'axes. De plus, ils portent un numéro d'identification permettant de viser un marqueur particulier.

La deuxième étape vers la réalisation du système de stationnement est le choix de la trajectoire. Il faut choisir une trajectoire que le robot peut suivre facilement et qui se termine sur un angle perpendiculaire au connecteur

de la station de charge. Idéalement, cette trajectoire devrait également garder le marqueur *Apriltag* dans la mire de la caméra tout au long de la procédure d'approche. Durant notre recherche de trajectoire adéquate, une proposition a été reçue de s'inspirer du mémoire de maitrise de Sébastien Laniel[15]. La trajectoire utilisée dans cet ouvrage est une courbe polynomiale de degré 2. Cette courbe permet de minimiser les corrections vers la fin de l'approche tout en proposant une vitesse de rotation continue. Cette trajectoire a été mise à l'épreuve sur le Turtlebot afin de la valider. La trajectoire a commandé une approche qui satisfait les capacités des robots en termes de guidage, cependant, dépendant de l'angle d'approche initiale, il se peut que le marqueur *Apriltag* soit perdu lors de l'approche finale. Il était donc nécessaire d'implémenter un algorithme de guidage pouvant suivre une trajectoire même après avoir perdu le contact visuel.

L'étape de conception suivante fut de développer le logiciel de guidage pouvant suivre la trajectoire jusqu'à la station. L'idée initiale fut d'essayer d'utiliser le planificateur local de la pile de navigation ROS. Ce planificateur prend en entrée une trajectoire et envoie les commandes aux roues permettant de suivre ladite trajectoire. Cependant, après une étude des planificateurs existants, il fut décidé d'implémenter notre propre algorithme de guidage. Le planificateur local était tout simplement trop intégré et interdépendant à la pile de navigation ROS.

Le développement a occupé Édouard Denommée pendant une bonne portion de la S8. Le tableau 7 présente les résultats des tests de stationnement effectués avec le TurtleBot. Le stationnement a également été démontré à de nombreuses reprises lors de l'Expo MégaGÉNIALE.

Tableau 7: Tests de l'approche du chargeur

Fonctionnalités	Test	Conditions	Résultat attendu	Résultat obtenu
Localiser la station de charge	S'assurer que les marqueurs soient repérés.	Imprimer des marqueurs et les mettre dans le champ de vision de la caméra.	Les marqueurs sont repérés et leurs positions sont à 10cm près de la mesure attendue.	Marqueurs détectés, erreur de 6 cm à 2 m de distance.
Effectuer une approche vers la station de charge	Le robot détecte le marqueur et se dirige vers la station	Présenter un *AprilTag* au robot.	Le robot complète l'approche 20 fois sans échec	20 approches en lignes sont un succès.

2.3.6 Système de surveillance de l'environnement

Le module de reconnaissance visuelle reçoit chaque image de la caméra pour effectuer des traitements d'image en utilisant la bibliothèque YOLO de Darknet et envoie des informations sur les objets détectés dans le système ROS du robot. La bibliothèque YOLO permet d'entraîner des réseaux de neurones spécifiquement pour la reconnaissance visuelle et elle permet de mettre en marche le traitement visuel en utilisant un réseau de neurones spécifié. Le choix de cette technologie repose d'abord sur la recommandation du client à explorer cette dernière, mais aussi sur sa popularité. L'expérience et la documentation disponible grâce à la communauté en ligne des logiciels libres (c.-à-d. Github et forums internet) étaient également des avantages. *Darknet* proposait également un noeud ROS qui permettait de l'intégrer facilement au reste du logiciel du robot. Un noeud ROS, programmé par l'équipe, permet également de filtrer les éléments visuels détectés par

Darknet et de construire les évènements de sécurité en incluant la date, l'heure, l'image et l'emplacement du robot lors de la détection.

La première étape a été d'effectuer la preuve de concept que YOLO et *Darknet* étaient des bibliothèques envisageables pour la reconnaissance visuelle d'objets d'intérêt. Pour ce faire, YOLO a été compilé et testé séparément du reste du système dans un environnement de test.

Ensuite, le noeud ROS *Darknet* a été testé avec une caméra intégrée à la machine et le résultat du traitement d'image satisfaisait les spécifications recherchées. À ce stade de conception, il était donc possible d'exécuter YOLO dans l'environnement ROS afin d'exploiter et de substituer différents réseaux de neurones préentraînés. Cette approche a permis de reconnaître divers objets d'intérêts et de valider la preuve de concept.

Les visages ont la priorité la plus haute parmi les objets à reconnaître dans le cadre du projet. Une banque de données d'environ 12 000 photos de visages annotées a été trouvée sur internet. L'annotation a cependant dû être ajustée pour respecter les spécifications de YOLO. Il a ensuite été possible d'entraîner un réseau de neurones pour la reconnaissance faciale avec succès. Une fois le réseau de neurones fonctionnel, les noeuds de filtrage et d'envoi vers la base de données ont pu être programmés afin d'obtenir la chaîne de détection complète. Cette dernière a pu être intégrée au robot au travers du système de gestion de robot afin de valider la performance du système complet et la quantité de ressources nécessaires à l'exécution de tous les modules simultanément.

Finalement, le deuxième objet d'intérêt pour la reconnaissance visuelle dans le cadre du projet était un extincteur. Puisqu'il n'existait pas de banque de données d'images d'extincteurs disponible en ligne, environ 600 images d'extincteurs ont été accumulées puis annotées par l'équipe. Toutefois, le nombre d'images étant insuffisant, l'entraînement fut un échec. La fonctionnalité de reconnaissance des extincteurs a donc été mise à l'écart pour le cadre du projet.

Anthony Parris s'est occupé de *Darknet* et du noeud de filtrage des détections. Cela inclut l'annotation ou le traitement de l'annotation des banques de données d'entraînement, la preuve de concept, les entraînements de réseau de neurones et la configuration des paramètres du module de reconnaissance visuelle. Pour ce qui est de l'intégration, Anthony et deux autres membres (Édouard Denommée, Édouard Légaré) se sont entraidés pour établir la chaîne complète de détection des évènements.

La performance de la chaine de détection d'évènements a été évaluée lors des patrouilles dans la faculté de génie pendant l'Expo MégaGÉNIALE. Les résultats obtenus sont présentés dans le tableau 10. Outre la consommation importante de ressources qui a nécessité une réduction de la fréquence des images traitées, la chaine de détection performe très bien et rencontre facilement les requis du projet.

2.3.7 Système de gestion du robot

Pour le module de gestion du robot, l'équipe a décidé d'utiliser HBBA (Hybrid Behavior-Based Architecture), un programme développé par le client et disponible gratuitement en ligne. Cette technologie a été sélectionnée à la demande du client, puisqu'elle facilitera l'intégration future d'autres technologies développées par le laboratoire IntRoLab.

Le principe de fonctionnement de HBBA est de filtrer les messages entre les différents noeuds ROS et de seulement laisser passer les messages pertinents. Pour ce faire, HBBA crée un registre de désirs que le robot veut accomplir et un registre de stratégies qui définissent comment accomplir ces désirs. Ces stratégies

précisent quels noeuds ROS doivent être activés ou non pour effectuer l'action qui accomplie le désir. Ainsi, lorsqu'un désir est ajouté dans le registre par un noeud ROS, le robot active la stratégie appropriée, jusqu'à ce que le désir soit retiré du registre par le noeud qui l'avait initialement ajouté.

La deuxième fonctionnalité du système de gestion du robot est d'assurer l'interprétation des données provenant du serveur. Pour ce faire, l'équipe a développé différents noeuds ROS de communication et de traduction qui permettent de relier le serveur à HBBA. L'un de ceux-ci est le noeud Electron. Ce noeud implémente EasyRTC et permet, à partir de l'interface Web, de se connecter aux robots. C'est aussi ce noeud qui achemine les flux vidéos des robots vers le serveur et qui agit comme interface avec les canaux de données de EasyRTC et le réseau ROS. Un noeud de téléopération est aussi présent. Ce noeud fait la conversion entre les commandes reçues de l'interface utilisateur et les commandes attendues par la pile de navigation ROS. Les deux autres noeuds se rapportent aux patrouilles. Un noeud d'exécution procède à la conversion des points de cheminement d'une patrouille reçue en objectifs de navigation et à leur acheminement ordonné vers la pile de navigation. Le second noeud, le planificateur, interagit avec la base de données sur le serveur pour obtenir les horaires associés au robot et fournir la patrouille au noeud d'exécution lorsqu'elle doit être exécutée.

Pour intégrer HBBA, il a tout d'abord fallu déterminer quels types de désirs le robot pouvaient avoir et quels noeuds ROS devaient être utilisés dans la stratégie associée. La liste finale est la suivante :

- Téléopération : Ce désir permet à l'opérateur de contrôler directement le robot. Pour ce faire, la stratégie associée laisse passer les commandes de téléopération venant de l'interface utilisateur directement jusqu'au noeud de contrôle des moteurs.
- Objectif : Ce désir permet à l'opérateur d'envoyer le robot à un objectif précis dans sa carte. Pour ce faire, la stratégie associée active le noeud de planification de trajet et lui envoie l'objectif qui vient de l'interface utilisateur.
- Patrouille : Ce désir envoie le robot sur une patrouille. Pour ce faire, la stratégie associée active le noeud de planification de trajet et le noeud de détection d'évènements. Les commandes envoyées au générateur de trajet proviennent du noeud de gestion de patrouille, qui est aussi activé par la stratégie.
- Stationnement : Ce désir permet au robot de retourner à sa station de charge. Pour ce faire, la stratégie associée active la détection des AprilTags et le noeud de génération du trajet de stationnement.

L'autre section importante à intégrer dans l'architecture de SecurBot est la couche de motivation de HBBA, soit le code qui ajoute et retire les désirs du registre des désirs actifs au moment approprié.

- Téléopération : Le noeud de motivation de téléopération est le plus simple. Lorsque l'interface utilisateur active le mode de téléopération, le désir est ajouté. Lorsque le mode de téléopération est désactivé, le désir est retiré.
- Objectif : Le noeud de motivation de ce désir l'ajoute lors que la commande est reçue de l'interface utilisateur. Par la suite, il compare la position du robot à celle dans le désir et retire le désir lorsque le robot se trouve à cette position, avec une précision de 20cm.
- Patrouille : Le noeud de gestion de patrouille est un noeud développé par l'équipe qui envoie le robot à différents objectifs sur sa carte dans un ordre précis. Lorsque le noeud reçoit une patrouille venant du serveur, il envoie le robot au premier point de cheminement en utilisant un désir de type objectif. Cependant, ce désir sera moins prioritaire qu'un désir objectif manuel, donc le robot pourra dévier d'une patrouille si l'opérateur lui demande d'aller ailleurs. Une fois que l'objectif de la patrouille est atteint,

une confirmation est envoyée au noeud d'exécution de patrouille et la commande pour le prochain point de cheminement est envoyée.

- Stationnement : Le noeud de motivation du stationnement s'occupe de créer les désirs pour envoyer le robot à sa station de recharge. Pour ce faire, il envoie tout d'abord le robot en face de la station avec un désir de type objectif. Une fois que ce désir est accompli, le désir de stationnement est ajouté. Si jamais le robot active une autre stratégie avant d'avoir accompli le désir de stationnement, le robot sera renvoyé en face de la station de charge avec un désir objectif avant que la stratégie de stationnement soit réactivée. Une fois que le stationnement est terminé, le robot tombe dans un état d'attente pour charger sa batterie sans qu'un autre désir puisse le faire bouger. Lorsque la stratégie d'attente est désactivée, le robot recule automatiquement pour ne plus toucher à la station de recharge.

La figure 15 représente l'architecture HBBA développé pour le projet. Chaque boite représente un noeud ROS et chaque ligne un lien de communication entre les noeuds. Les valves vertes représentent des filtres que HBBA peut activer ou désactiver selon la stratégie active.

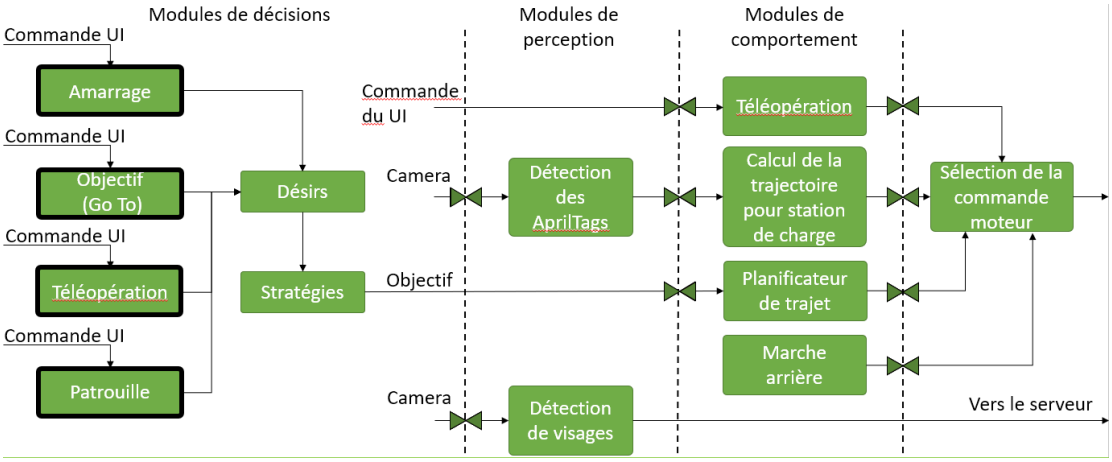

FIGURE 15 – Architecture ROS avec HBBA

Les modules qui peuvent être activés sont séparés en deux catégories : les modules de perception et les modules de comportement. Les modules de perception sont des blocs qui transforment les données des capteurs du robot, comme une caméra, en informations utilisables par le robot, comme la position d'un AprilTag. Les modules de comportement sont des modules qui envoient des commandes aux moteurs du robot. Ainsi, si plusieurs modules de comportement peuvent envoyer des commandes aux mêmes moteurs, comme dans le cas du projet SecurBot, un noeud est automatiquement généré par HBBA pour sélectionner la commande d'un seul module de comportement et l'envoyer aux moteurs. Cette sélection est faite en fonction d'un ordre de priorité prédéfini.

L'implémentation de cette section du projet a demandé la participation de deux membres de l'équipe, soit Alexandre Guilbault et Cédric Godin. L'équipe a aussi reçu l'aide de François Ferland, le créateur de HBBA, pour intégrer l'outil au projet SecurBot. Son aide a permis de corriger les problèmes rencontrés et a aussi permis d'apporter quelques améliorations et corrections à HBBA.

Pour valider la performance du système de gestion du robot, les tests effectués ont été divisés en deux catégories. La première visait à confirmer que les stratégies implémentées dans HBBA activaient les bons

modules de l'architecture ROS. La deuxième avait pour rôle de confirmer que la bonne stratégie était activée si le robot avait plusieurs désirs au même moment.

La procédure utilisée pour la première partie des tests a été de manuellement ajouter un désir à tester dans le registre des désirs actifs et par la suite regarder quelle stratégie est activée par HBBA. Des outils fournis avec HBBA permettent de facilement réaliser ces deux étapes. Le comportement du robot pouvait alors être observé pour confirmer que la stratégie activait les bons modules.

Tableau 8 – Test de fonctionnement des désirs et stratégies

Désir ajouté	Stratégie activée	Comportement observé
Objectif	Objectif	Le robot calcule une trajectoire jusqu'à l'objectif. Le robot se déplace jusqu'à l'objectif. Le robot arrête de bouger.
Stationnement	Stationnement	Le robot trouve le AprilTag. Le robot avance vers la station de recharge et fait contact.
Patrouille	Patrouille	Le robot se dirige vers chaque point de la patrouille en ordre. Le robot arrête.
Attente	Attente	Le robot ne fait rien.
Téléopération	Téléopération	Le robot ne bouge pas. Lorsqu'une commande est envoyée avec l'interface, le robot se déplace.
Rien	Marche arrière	Le robot recule lors de la publication d'un temps de marche arrière, sinon il est immobile.

Le tableau 8 montre le résultat de la première partie des tests. Dans chaque cas, la stratégie est la bonne et le comportement respecte les attentes. La publication manuelle d'un temps vers le module de marche arrière permet de valider les deux états possibles de cette stratégie. Ainsi, il est possible de vérifier son fonctionnement sans recourir au module de stationnement.

Pour la deuxième partie des tests, tous les types de désirs sont ajoutés au registre de désirs. La stratégie activée et le comportement du robot sont observés pour s'assurer qu'ils correspondent aux résultats attendus. Le désir dont la stratégie a été activée est alors enlevé et le test recommence.

Tableau 9 – Test de fonctionnement des désirs et stratégies

Désirs ajoutés	Stratégie activée	Comportement observé
Objectif Stationnement Patrouille Attente Téléopération	Téléopération	Le robot ne bouge pas. Lorsqu'une commande est envoyée avec l'interface, le robot se déplace.
Objectif Stationnement Patrouille Attente	Objectif	Le robot calcule une trajectoire jusqu'à l'objectif. Le robot se déplace jusqu'à l'objectif. Le robot arrête de bouger.
Stationnement Patrouille Attente	Stationnement	Le robot trouve le AprilTag. Le robot avance vers la station de recharge et fait contact.
Patrouille Attente	Attente	Le robot ne fait rien.
Patrouille	Patrouille	Le robot se dirige vers chaque point de la patrouille en ordre. Le robot arrête.
Rien	Marche arrière	Le robot ne bouge pas.

Le tableau 9 montre le résultat de la deuxième partie des tests. Comme lors des derniers tests, une seule stratégie est activée à un moment donné, selon l'ordre d'importance de ces stratégies. Les stratégies apparaissent

bien dans l'ordre désiré pour l'application du projet SecurBot. Le test montre aussi que le désir d'attente est utilisé seulement pour empêcher le robot de partir en patrouille lors de la recharge.

2.4 Performances du système et de l'intégration des modules

L'utilisation du robot pendant les dernières semaines du projet a permis de valider l'atteinte des indicateurs identifiés à la section 2.1. Les fonctionnalités de patrouille autonome ont d'ailleurs été démontrées en continu pendant deux jours lors de l'Expo MégaGÉNIALE. Les résultats obtenus sont présentés dans le tableau 10. Le seuil désiré pour tous les indicateurs a au moins été partiellement atteint.

Tableau 10: Indicateurs de performance obtenus

Indicateur objectivement vérifiable	Résultat	Atteint
Toutes les composantes électroniques sont fermement fixées au robot et ce dernier doit pouvoir se déplacer de façon agile et stable.	Le robot demeure stable jusqu'à sa vitesse maximale de 1 m/s. Il est resté intact lors du transport entre le 3IT, la faculté de génie et le centre culturel.	Oui
Le robot peut effectuer ses tâches pendant au moins une heure et être en mesure de se recharger en moins de deux heures.	Le robot peut patrouiller pendant 3h30. Le changement de batterie dure 5 minutes.	Partiel
L'interface se connecte au robot et permet de voir la carte du bâtiment, de téléopérer le robot avec un flux vidéo et permet de planifier des trajets pour une future ronde de sécurité.	La position du robot s'affiche sur la carte avec une précision 30 cm. Le flux vidéo est d'environ 2 images par seconde avec une latence entre 1 et 3 secondes.	Partiel
Le robot et l'opérateur sont capables de se connecter au serveur. Ce dernier conserve un registre des alertes générées par le robot.	2 robots et 2 opérateurs peuvent utiliser le serveur simultanément. Le robot a enregistré 895 alertes sur le serveur pendant le projet.	Oui
Le robot est en mesure d'identifier 70% des évènements visuels rencontrés le long du trajet.	39 personnes sur 40 rencontrées sont identifiées. Il y un faux positif sur 80 images.	Oui
Le robot planifie ses actions selon son niveau de charge et est en mesure de changer son mode contrôle selon la situation.	Les patrouilles démarrent au maximum à une minute de l'heure prévue. La patrouille peut être interrompue par un objectif manuel ou la téléopération puis se poursuivre.	Partiel
Projet en ligne sur *GitHub*	Les documents suivants sont en ligne : -documentation -tests -Fichier de conception pour tous les extrants	Oui

Le premier manque se situe au niveau de la gestion de l'énergie par le robot. En effet, puisque le déverminage de la recharge ne progressait pas, les ressources de l'équipe logicielle ont été distribuées ailleurs dans le projet. Ainsi, lorsque la carte de batteries fût fonctionnelle, le noeud ROS qui devait transmettre le niveau de charge mesuré par le microcontrôleur de la carte de batterie à HBBA pour ajouter le désir de recharge n'était pas prêt. Le micrologiciel de la carte n'était pas non plus adapté à la communication avec le robot à ce moment, le port série étant utilisé pour imprimer des informations de débogage lisibles par un humain. La fonctionnalité a donc été mise de côté quelques jours avant l'Expo MégaGÉNIALE. Les batteries commerciales utilisées pendant le développement sont donc demeurées en utilisation puisque le robot ne pouvait pas encore se recharger par lui même à la station de charge. Il est toutefois en mesure de s'y stationner si l'utilisateur démarre la séquence de stationnement à partir de l'interface de téléopération.

Ensuite, au niveau de l'intégration logicielle, l'utilisation de tous les modules simultanément ne peut pas se faire sur les deux robots. Seul le Jetson TX2 installé sur la base Pioneer offre une performance suffisante pour le faire. Le processeur et la mémoire sont alors saturés à respectivement 100% et 80%. L'utilisation d'un disque dur supplémentaire est également conseillée au client, les cartes produites par RTAB-Map atteignant rapidement plusieurs gigaoctets. Sur le Jetson Nano, installé sur la base TurtleBot, la détection d'évènement a été omise, la mémoire vive étant insuffisante.

Enfin, les performances des flux vidéos vers l'interface ne rencontrent pas les requis fixés par l'équipe pour une expérience de pilotage à distance agréable. Toutefois, elles sont suffisantes pour bien observer l'environnement du robot et lui envoyer des cibles vers lesquelles le robot navigue de manière autonome. Cela s'explique par les limitations au niveau de la bande passante, mais surtout par la saturation de l'ordinateur de bord. Aussi, les notifications poussées d'évènements n'ont pas été réalisées. Le niveau de complexité était assez grand puisque la mise en place des notifications impliquait l'ajout d'un service au serveur et l'utilisation de technologies non familières et nouvelles dans le projet. Une solution de contournement, soit un bouton de rafraichissement sur la page de l'interface qui affiche les évènements, a été préférée vu sa grande simplicité d'implémentation et la faible valeur ajoutée des notifications poussées pour l'atteinte du but du projet.

3 Bilan sur la gestion du projet

3.1 Bilan financier

Tableau 11 – Budget du projet SecurBot

Catégorie	Coût prévu ($)	Dépense ($)	Restes ($)
PCB	400.00	315.30	84.70
Électronique	800.00	1173.06	-373.06
Batteries	250.00	243.89	6.11
Navigation	3000.00	2290.90	709.10
Mécanique	700.00	818.91	-118.91
Équipement de protection	200.00	202.89	-2.89
Total	5350.00	5044.95	305.05

Comme le présente le tableau 13, le budget global du projet a été respecté. Il reste 305$ sur ce qui était le total prévu de 5350$. Toutefois, certains écarts entre la planification et les dépenses réelles sont considérables. Pour la section de navigation, le surplus de 709.10$ est dû au fait que notre client nous a demandé d'utiliser

une version allégée de l'ordinateur de bord prévu à l'origine, afin de tester les capacités de celui-ci et sa compatibilité avec notre système.

Le dépassement de coût pour la portion électronique s'explique principalement par l'achat de pièces supplémentaires pour le déverminage des cartes et la réalisation des correctifs identifiés lors des tests. Il a notamment fallu ajouter un circuit imprimé pour corriger l'empreinte sur la carte de l'isolateur qui permet la connexion USB sécuritaire entre l'ESP32 et l'ordinateur de bord et remplacer certaines composantes endommagées lors de l'assemblage.

Le dépassement de coût pour la section de mécanique est dû à la modification des objectifs de cette partie. En effet, une seule structure mécanique était prévue lors de l'élaboration du budget au début du projet. Toutefois, avec les surplus disponibles et afin de démontrer la facilité d'adaptation de la conception mécanique des armures d'un robot à l'autre, une deuxième armure fut fabriquée.

3.2 Bilan en temps

En ce qui a trait à la contribution en temps investie dans le projet par membre d'équipe, on peut constater à la table 12 que chaque membre a consacré dans les alentours de 18h par semaine au long des 30 semaines de projet. L'équipe considère que tous les membres ont investi un nombre acceptable d'heures et que tous ont contribué au projet de façon efficace. Il est à noter que certains membres ont fait preuve d'altruisme et ont investi plus d'heures que le 18h par semaine établi dans le contrat d'équipe en début de projet. Cela nous a permis d'avoir un produit plus complet à la fin du projet, donc une navigation très réactive et une interface utilisateur intuitive et facile d'utilisation.

Tableau 12 – Heures investies par membre de l'équipe

Membres	**Heures travaillées (h)**	**Moyenne d'heures pour 30 semaines (h)**	**Rôle de gestion**
Olivier Belval	551.50	18.38	Trésorier
Édouard Denommée	508.83	16.96	Responsable assurance qualité
Valérie Gauthier	541.58	18.05	Chef de projet
Cédric Godin	659.08	21.97	Directeur technique
Alexandre Guilbault	542.83	18.07	Responsable des ressources humaines
Édouard Légaré	714.00	24.70	Secrétaire
Philippe Marcoux	504.00	16.80	Scrum Master
Anthony Parris	519.16	17.30	Responsable documentation et standard

La figure 16 illustre la répartition des heures en fonction des tâches effectuées par chaque membre de l'équipe. On observe que chaque membre a investi entre 40 à 60% dans des tâches non explicitement techniques soit les livrables, la préparation d'audit et la gestion du projet. Conséquemment, chaque personne a pu contribuer à des tâches techniques de façon équitable. Dans la figure, les tâches en jaune sont les tâches majeures dans lesquelles chaque membre a investi la majorité de son temps. Olivier Belval et Philippe Marcoux se sont investis majoritairement dans les tâches reliées au module d'électronique, Cédric Godin et Édouard Denommée au module de navigation, Édouard Légaré à l'interface utilisateur, Valérie Gauthier à la mécanique, Alexandre Guilbault à l'application robot et finalement Anthony Parris à la détection d'évènement. On en conclut donc

que la méthode d'attribution des tâches, c'est-à-dire selon la volonté de chaque membre, fut efficace et que chacun des membres avait des tâches balancée entre le volet technique et le volet planification et gestion du projet.

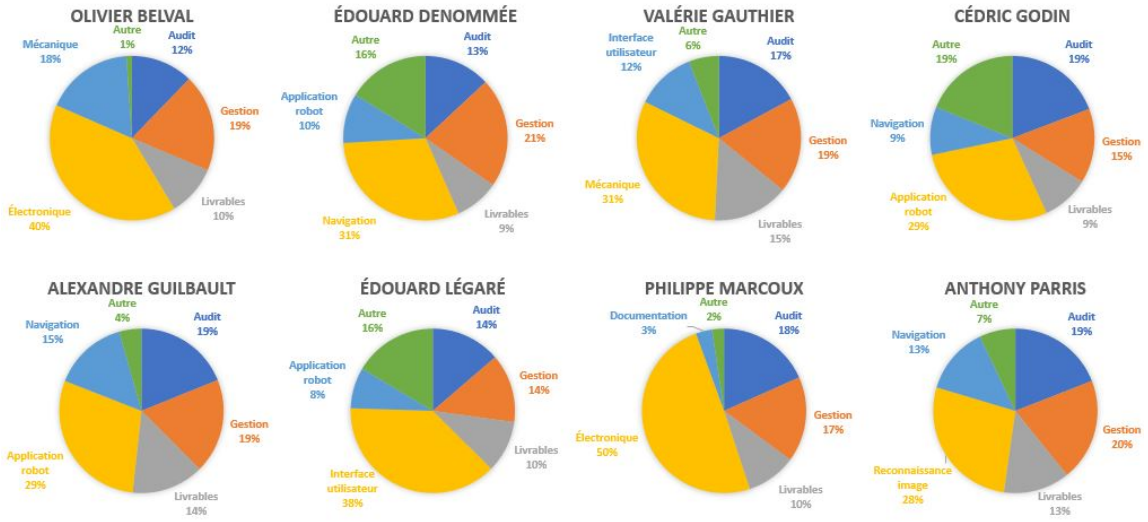

FIGURE 16 – Répartition des heures investies par membre de l'équipe

D'un point de vue plus global, le tableau 13 permet la comparaison entre les heures par module prévues au début du projet, dans la phase de conception, et les heures investies réellement. Pour les modules d'électronique, de navigation et d'application robot, les estimations se sont révélées être plutôt précises.
Dans le cas de la mécanique, le temps réel investi est bien en dessous de ce qui fut planifié. L'équipe n'ayant aucun membre avec une formation en génie mécanique a prévu plus de temps en prévision que le manque d'expertise allait ralentir l'avancement du module. Il s'est toutefois avéré qu'avec l'aide de logiciel de conception mécanique, il fut possible d'effectuer le travail plus rapidement et de diminuer les risques associés à une mauvaise intégration mécanique.
Pour l'interface utilisateur, le temps investi se retrouve bien en dessous de l'estimation originale, cela est dû principalement à la redistribution des membres de l'équipe pour combler les différents besoins du projet. Il était initialement prévu que plusieurs membres devaient travailler sur l'interface, dont certains n'ayant pas d'expérience en programmation Web. L'estimation prenait en compte ces deux éléments (plusieurs personnes + apprentissage de la technologie) en plus d'une marge, mais, comme mentionné précédemment dans la description du système, une seule personne a finalement travaillé sur ce module ce qui a diminué les heures nécessaires à son accomplissement.
Pour le module de détection d'évènement, plusieurs éléments ont mené à la surestimation du temps nécessaire à l'accomplissement de la tâche. Premièrement, l'intention originale de l'équipe pour la détection d'évènement était d'inclure la cartographie de données environnementales. L'intégration de cette fonction n'était toutefois pas dans les priorités de l'équipe et elle a donc été mise de côté afin de favoriser d'autres tâches comme le développement de l'application robot. Deuxièmement, l'intégration de la technologie ODAS fut d'abord considérée comme une tâche nécessaire, mais le client a par la suite communiqué qu'il ne désirait pas inclure cette tâche dans notre mandat.
Pour le serveur, un manque de connaissance sur les technologies Web à emprunter a mené à une surestimation des heures à investir dans ce domaine. Les tâches serveur consistent principalement à créer l'infrastructure

back-end pour la page Web servant d'interface utilisateur ainsi que de cataloguer les données collectées par les robots. Après avoir choisi la technologie WebRTC pour acheminer les flux vidéo des robots à l'interface utilisateur, il a été estimé que l'infrastructure requise du côté serveur aurait demandé plus de travail. Il s'est avéré qu'en utilisant la librairie EasyRTC, plusieurs étapes de configuration serveur ont été simplifiées. En ayant plus d'expérience avec les technologies back-end et la pile WebRTC, l'estimation des tâches "Serveur" aurait pu être amélioré.

Tableau 13 – Comparaison des heures investies par module en fonction de la planification initiale

Modules	**Heures prévues (h)**	**Heures investies (h)**	**Erreur sur la prévision**
Gestion	-	1164.73	-
Livrables	-	931.50	-
Interface utilisateur	600.00	470.50	28%
Électronique	600.00	551.33	9%
Navigation	475.00	507.08	-6%
Mécanique	400.00	291.08	37%
Détection d'évènement	400.00	156.58	155%
Application robot	300.00	357.00	-16%
Serveur	300.00	94.75	219%

La figure 17, illustre la répartition du temps investi dans chaque sous-module du projet. Un peu moins de 50% des efforts de l'équipe furent investis dans la gestion et les livrables. Les tâches techniques représentent l'autre moitié des heures du projet total. Les modules les plus importants en heure furent l'électronique (12%), la navigation (11%) et l'interface graphique (10%). Il reste que la majorité des modules sont de nature très diverse et qu'ils représentent chacun une part non négligeable du travail. La figure nous permet donc de bien apprécier la complexité du projet et la variété des tâches qui furent accomplies.

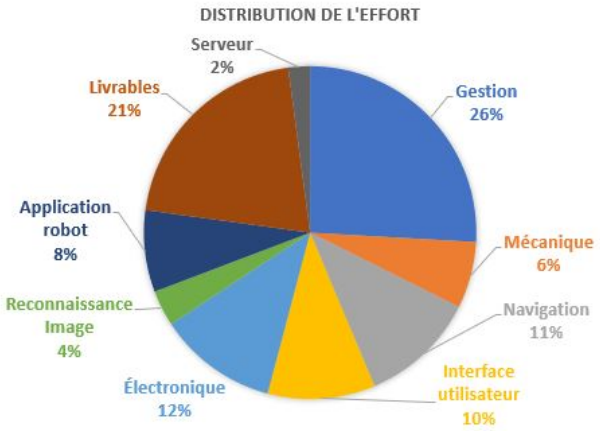

FIGURE 17 – Répartition des heures investies dans le projet

L'équipe SecurBot a décidé d'utiliser une méthode agile comme méthode de gestion pour mener à terme le projet. La méthode agile conventionnelle fut adaptée avec des sprints de 2 semaines qui permettaient des rencontres et un suivi avec le client 2 fois par mois comme illustré à la figure 18. Des sprints de deux semaines avec une rencontre à chaque semaine s'agençaient mieux avec l'horaire de chacun des membres.

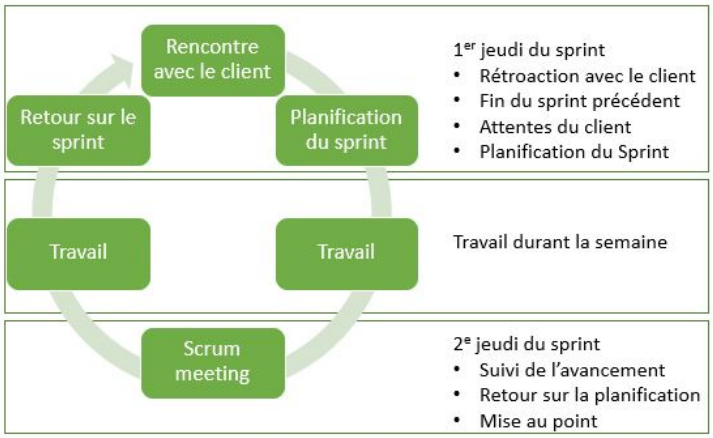

FIGURE 18 – Méthode de gestion par sprint pour SecurBot

À la figure 19, le graphique du taux d'accomplissement illustre la valeur acquise en heure sur le temps prévue pour chaque sprint. Cela donne donc une métrique pour évaluer si les tâches prévues dans le sprint ont été accomplies. Le deuxième graphique permet d'évaluer l'erreur sur l'estimation du travail afin de vérifier que le temps investi sur une tâche était bien ce qui était prévu. Pour les 5 premiers sprints du projet (S7), l'équipe a réussi a complété environ 90% du travail planifié, ce qui est très bien considérant que l'équipe a parfois sous-estimé (jusqu'à 40%) le temps nécessaire à accomplir une tâche. On en déduit donc que l'équipe a été en mesure d'investir plus d'heures que prévu pour atteindre les exigences. Pour la deuxième session (S8), on remarque que l'estimation du temps nécessaire pour effectuer les tâches est beaucoup moins précise. Ce problème est relié à la nature des tâches qui était beaucoup plus technique en S8 qu'en S7. Avec des tâches techniques, il est plus difficile de prévoir les problèmes ainsi que d'estimer le temps nécessaire pour les régler. Un exemple bien précis de ce phénomène fut d'installer HBBA, une tâche qui fut estimée à 10 heures qui au final en aura nécessitée près d'une centaine (sprint 7). L'estimation avait été faite en considérant que l'outil était prêt à être utilisé, mais certaines dépendances que HBBA utilise pour fonctionner n'était pas utilisable sur une architecture ARM64. L'équipe a donc investir du temps supplémentaire pour recompiler ces bibliothèques et modifier HBBA pour fonctionner sur l'architecture ARM64.

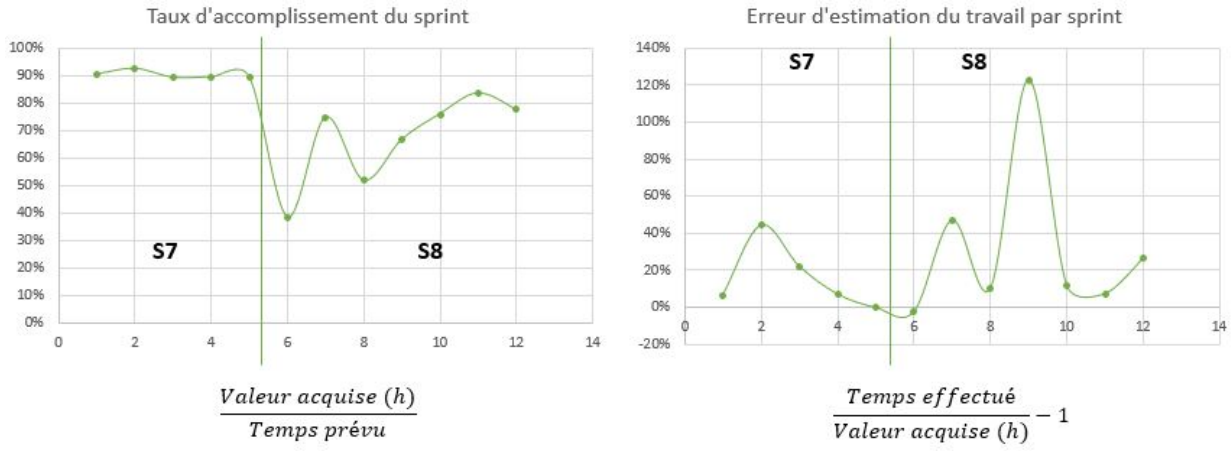

FIGURE 19 – Métriques d'analyse de l'efficacité des sprints

3.3 Bilan global sur la satisfaction des attentes

Le but du projet était que le prototype conçu agisse comme une preuve de concept capable de convaincre des investisseurs potentiels qu'un robot de surveillance est réalisable, à coût réduit, avec les technologies d'IntRoLab. Le produit final permet effectivement de faire la preuve qu'avec un investissement modeste on obtient un produit avec les fonctionnalités de base d'un robot de sécurité commercial. Plus précisément, le prototype :

- peut effectuer une ronde de sécurité durant plusieurs heures sans intervention humaine ;
- est capable de cartographier en 3D une zone prédéterminée, ici, le quatrième étage de la faculté de génie, en moins d'une heure ;
- identifie la présence d'individu dans son environnement de façon très fiable ;
- communique avec l'opérateur et est contrôlé à l'aide d'une interface graphique intuitive ;
- utilise la technologie d'IntRoLab HBBA pour la gestion globale du robot ;
- utilise la technologie d'IntRoLab RTAB-Map pour la cartographie et la navigation du robot dans son environnement ;
- intègre mécaniquement la technologie d'IntRoLab 16SoundsUSB pour l'implémentation de la technologie d'IntRoLab ODAS pour de l'identification de sources sonores.

L'équipe a confiance que les spécifications initiales furent complétées et adaptées de façon à répondre à la demande du client.

Pour la suite du projet, le premier point critique à adresser est l'intégration des cartes électroniques dans les robots. Leur fonctionnement a été vérifié au laboratoire, il reste l'installation. Le noeud ROS pour la publication du statut de la batterie vers HBBA est également à faire.

Le deuxième point est l'optimisation des performances lorsque tous les modules sont actifs. L'équipe a identifié plusieurs pistes à ce niveau. D'abord, l'utilisation d'une caméra officiellement supportée sur le Jetson afin d'exploiter la carte graphique pour libérer le processeur. Aussi, utiliser ROS bridge plutôt que WebRTC pour envoyer la vidéo et les données vers l'interface utilisateur ce qui permettrait d'éliminer Electron et de grandement réduire la charge du processeur pour l'encodage des flux vidéo et l'utilisation de la mémoire vive.

L'équipe croit que le robot livré pourrait être utilisé pour l'optimisation de bâtiments en plus de la surveillance. L'installation de capteurs environnementaux couplée aux patrouilles programmables permettrait, par exemple, de dresser la carte de la température sur un étage en fonction de l'heure de la journée et de l'achalandage afin d'ajuster le système de ventilation.

L'ajout d'un mode d'exploration automatique serait également intéressant. Cela permettrait d'éviter le processus de cartographie manuel pendant lequel l'opérateur doit téléopérer le robot dans tout l'espace à couvrir. L'équipe croit qu'une solution existante dans ROS serait triviale à intégrer puisque la navigation de SecurBot utilise déjà la pile de navigation ROS.

Références

[1] CrossWing, "HOME | Crosswing." https://www.crosswing.com. Page consultée le 2019-02-19.

[2] S. Sutherland, S. Coulombe, and D. Wick, "Customizable robotic system," Mar. 31 2015. US Patent 8,994,776.

[3] S. B. Sutherland and D. Wick, "Control system for mobile robot," Jan. 19 2017. US Patent App. 15/255,935.

[4] WebRTC, "WebRTC." https://webrtc.org. Page consultée le 2019-01-23.

[5] Knightscope, "Knightscope." https://www.knightscope.com/. Page consultée le 2019-02-20.

[6] A. Tabibian, "Tech. Cars. Machines.." https://gtkpartners.com/knightscope/. Page consultée le 2019-02-20.

[7] Segway, "Segway Robotics | Invent. Inspire. Empower.." https://www.segwayrobotics.com/. Page consultée le 2019-02-20.

[8] Segway, "Segway Robot | Developper." https://developer.segwayrobotics.com/developer/overview.html. Page consultée le 2019-02-20.

[9] M. Steiner, "ROS Navigation Stack On A Loomo Segway robot." https://karriere.iteratec.de/fileadmin/Karriere/PDF/Abschlussarbeiten/ROS_Anbindung_fuer_Loomo.pdf, 2018. Page consultée le 2019-01-20.

[10] RoboteX, "RoboteX | Robot Technology Solutions." https://robotex.com/. Page consultée le 2019-02-20.

[11] RoboteX, "Avatar III Security Robot | RoboteX." https://robotex.com/avatar-security-robot/. Page consultée le 2019-02-20.

[12] Cobalt, "Cobalt Robotics | Robots and people working together to secure the modern workplace." https://cobaltrobotics.com/. Page consultée le 2019-02-20.

[13] "Cobalt," *IEEE Robots*. https://robots.ieee.org/robots/cobalt/ Page consultée le 2019-01-20.

[14] O. S. R. Foundation, "ROS Powering the world's robots." http://www.ros.org/, 2019. Page consultée le 2019-01-22.

[15] S. Laniel, "Architecture de contrôle d'un robot de téléprésence et d'assistance aux soins."

oiq.qc.ca