POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

Département de génie informatique et génie logiciel

INF3995

Projet de conception d'un système informatique

Documentation du projet répondant à l'appel d'offres no. H2025-INF3995 du département GIGL.

Conception d'un système d'exploration multi-robot

Équipe No ***102***

Zerouali, Amine

Gratton Fournier, Kevin Santiago

Haddadi, Issam

Hachemi Boumila, Rafik

Abassi,Yassine Mohamed Taha

Milord, Mario Junior

février 2025

1. Vue d'ensemble du projet

1.1 But du projet, porté et objectifs (Q4.1)

Le projet dans le cadre du cours INF3995 vise à concevoir, développer et démontrer une preuve de concept d'un système d'exploration multi-robot. En effet, dans l'optique d'une exploration planétaire, plusieurs robots simples et autonomes seront déployés, plutôt qu'un unique robot complexe. Ce système constitué de deux robots permet l'exploration d'une zone donnée, en communiquant entre eux avec une station de contrôle au sol, tout en transmettant les données recueillies vers une interface opérateur. Ce projet se veut une simulation réaliste d'un contrat qui pourrait être passé entre un sous-traitant et une agence spatiale dans le monde du travail. Il permettra ainsi de démontrer la faisabilité technique d'un tel système en atteignant le niveau NMS 4 (Niveau de Maturité de la Solution), en mettant l'accent sur l'autonomie des robots, leur capacité à cartographier l'environnement, et leur interaction avec un opérateur humain.

Le système à développer comprend trois composantes principales étant une station au sol, des robots physiques et une simulation Gazebo. En ce qui concerne la station au sol, c'est une application web permettant l'interaction à l'opérateur d'interagir avec le système, d'envoyer des commandes, de surveiller l'état des robots et de visualiser les données collectées. Pour ce qui est des robots physiques, il s'agit de deux robots de type AgileX Limo équipés de capteurs et finalement la simulation Gazebo est une réplique virtuelle permettant de tester les algorithmes et les fonctionnalités du système sans recourir aux robots physiques. Le projet se doit donc d'intégrer ces trois composantes de façon cohérente afin d'offrir une interface unifiée à l'opérateur.

L'objectif principal du projet est de démontrer la faisabilité d'un système d'exploration autonome composé de plusieurs robots capables de collaborer pour explorer une zone inconnue. Le système doit permettre à un opérateur de superviser les missions à distance, grâce à une interface web intuitive. Les robots devront se déplacer de façon autonome, éviter les obstacles et cartographier l'environnement en temps réel.

Biens livrables attendus:

Preliminary Design Review (PDR) - 14 février 2025

- Prototype préliminaire et présentation des premiers requis R.F.1 et R.F.2.

Critical Design Review (CDR) - 28 mars 2025

- Version intermédiaire du système.

Readiness Review (RR) - 15 avril 2025

- Version finale du système multi-robot avec tous les requis techniques.

1.2 Hypothèse et contraintes (Q3.1)

Le développement du projet jusqu'à son fonctionnement repose sur plusieurs hypothèses de base qui encadrent la conception et la mise en oeuvre du système d'exploration multi-robot:

-Les robots physiques AgileX Limo fournis seront en bon état de fonctionnement tout au long du projet.

-Les capteurs déjà installés sur les robots (IMU, lidar, etc.) seront pleinement opérationnels tout au long de la session et suffisants pour répondre aux exigences d'exploration et de cartographie.

-Le réseau WiFi mis à disposition assurera une connexion stable entre la station au sol et les robots physiques.

-Le simulateur Gazebo sera fonctionnel et permettra de reproduire fidèlement les comportements des robots physiques dans un environnement virtuel.

-L'équipe de projet aura accès aux différentes ressources pour mener à bien la conception du projet, telles que la salle de laboratoire pour les tests, les exigences techniques, etc.

-La communication au sein de l'équipe sera fluide tout au long de la session et les membres de l'équipe effectueront un travail de manière collaborative en respectant les délais et les responsabilités attribuées.

Le projet comprend plusieurs contraintes qu'elles soient techniques, humaines ou organisationnelles:

Délais fixes: Le projet est structuré autour de trois jalons incontournables : la révision préliminaire de conception (PDR) le 14 février 2025, la révision critique de conception (CDR) le 28 mars 2025, et la révision de préparation (RR) le 15 avril 2025. Ces dates imposent un rythme soutenu pour respecter les livraisons.

Charge de travail limitée : L'équipe dispose d'une charge de travail maximale de 630 heures-personnes pour l'ensemble du projet. Toute proposition dépassant ce seuil serait jugée non conforme.

Disponibilité des équipements : Les robots et le matériel fourni sont partagés entre plusieurs équipes. L'accès aux robots physiques et au local d'expérimentation est donc limité, ce qui impose une planification rigoureuse des tests sur le matériel réel.

Respect des exigences techniques : Le système doit se conformer aux exigences matérielles, logicielles, fonctionnelles et de conception spécifiées dans le document « Exigences Techniques ». Certains requis sont obligatoires et conditionnent donc l'acceptation des livrables.

Conteneurisation : Toutes les composantes logicielles, à l'exception du code embarqué sur les robots, doivent être conteneurisées avec Docker afin d'assurer la portabilité et la reproductibilité des environnements de développement et d'exécution.

Sécurité : Les robots ne peuvent être utilisés que dans les locaux désignés.

Démonstrations vidéo : Toutes les fonctionnalités du système doivent être démontrées par des vidéos claires et concises, déposées sur GitLab, pour valider les livrables.

1.3 Biens livrables du projet (Q4.1)

Preliminary Design Review (PDR) – 14 février 2025

-Documentation initiale du projet (PDF sur GitLab), présentant la planification et l'architecture préliminaire.

-Démo vidéo – Robots physiques : Réponse d'un robot à la commande "Identifier" (R.F.1).

-**Démo vidéo – Simulation Gazebo** : Exécution des commandes "Lancer la mission" et "Terminer la mission" (R.F.2).

Critical Design Review (CDR) – 28 mars 2025

-Documentation révisée du projet (PDF sur GitLab), intégrant les décisions de conception.

-Présentation technique orale (10 min).

-Démos vidéos démontrant les fonctionnalités principales (R.F.1, R.F.2, …) et la collecte de journaux.
Readiness Review (RR) – 15 avril 2025

-Documentation finale du projet (PDF sur GitLab).

-Présentation finale.

-Démos vidéos de toutes les fonctionnalités complétées.

-Document "Tests.pdf" et scripts de tests.

Suivi hebdomadaire – Chaque semaine

Rapport d'avancement sur Discord.
Mise à jour des issues et milestones sur GitLab.

2. Organisation du projet

2.1 Structure d'organisation (Q6.1)

L'équipe de projet est composée de six membres, et son fonctionnement repose sur une organisation collaborative inspirée des principes de la méthode Agile. Cette approche est privilégiée afin de favoriser la flexibilité, l'adaptation rapide aux imprévus et l'implication continue de tous les membres.
Plutôt que d'attribuer des rôles fixes et stricts à chaque personne, l'équipe fonctionne sur la base d'une répartition dynamique des tâches selon les compétences, les intérêts et la charge de travail de chacun. Les membres se répartissent les lots de travail en fonction des priorités du moment, tout en assurant un suivi régulier de l'avancement.

Chaque semaine, une **réunion d'équipe** est tenue afin de :

- Faire le point sur l'avancement des travaux.
- Identifier les obstacles rencontrés.
- Ajuster la répartition des tâches si nécessaire.

L'utilisation des issues et milestones sur GitLab permet d'assurer le suivi des tâches et des jalons du projet. Chaque fonctionnalité ou correction fait l'objet d'une issue, assignée à un ou plusieurs membres, et suivie jusqu'à sa fermeture.

Bien que chaque membre contribue à plusieurs aspects du projet, une attention particulière est portée à la coordination entre les trois grandes composantes du système : logiciel embarqué sur les robots, simulation Gazebo et interface utilisateur/station au sol. La collaboration entre ces sous-ensembles est essentielle pour garantir l'intégration cohérente du système final.

2.2 Entente contractuelle (Q11.1)

Le type d'entente contractuelle retenu est une entente à terme. Ce choix s'explique par la nature académique du projet et les contraintes précises imposées:

Échéances fixes : Trois jalons sont imposés (PDR le 14 février, CDR le 28 mars et RR le 15 avril), nécessitant une livraison des livrables à des dates déterminées, ce qui est typique d'une entente à terme.

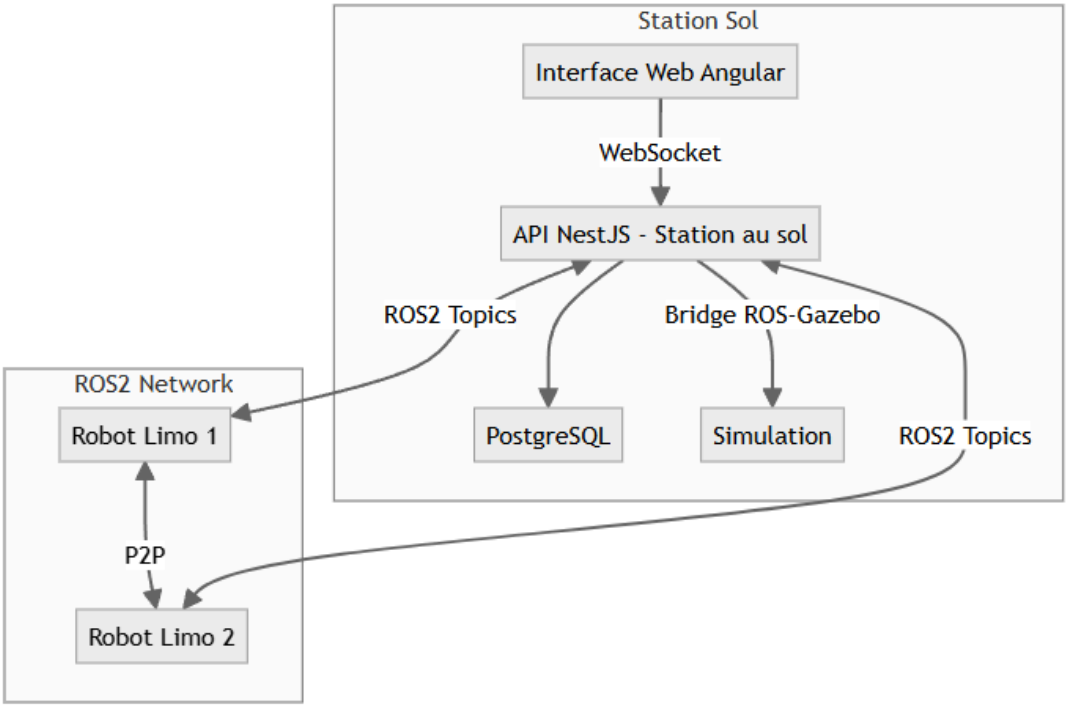
Charge de travail définie : La limite de **630 heures-personnes** impose une gestion stricte du temps consacré à chaque tâche.

Objectif pédagogique : L'objectif principal étant d'acquérir des compétences techniques et en gestion de projet, tout en respectant des échéances précises, une entente à terme est cohérente avec ce cadre.

Cette entente assure une bonne maîtrise du calendrier et de la charge de travail, en phase avec les attentes et les exigences du cours INF3995.

3. Description de la solution

3.1 Architecture logicielle générale (Q4.5)



frontend: Angular 16
backend: NestJS
database: PostgreSQL
robotique: ROS2 Humble
simulation: Gazebo Fortress

Station au sol (NestJS, PostgreSQL) : Elle est le cœur du système, centralisant les communications et les données (R.F.2, R.F.3, R.F.8, R.C.1).

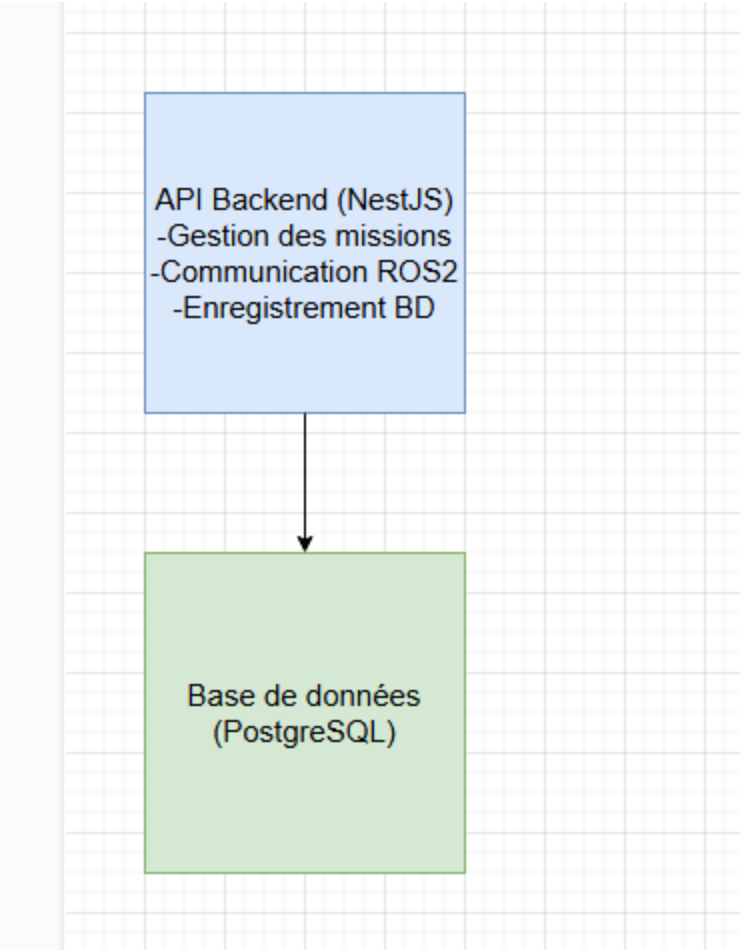
Interface utilisateur (Angular) : Application web accessible sur différents appareils pour superviser les robots (R.F.10, R.C.4).

Logiciel embarqué (ROS2 Humble) : Pilote les robots physiques, assure l'autonomie et la gestion des capteurs (R.F.4, R.F.5, R.F.9).

Simulation (Gazebo Fortress) : Permet de tester les comportements et algorithmes avant d'utiliser les robots réels (R.C.3).

Les communications s'effectuent via **Socket.IO** entre le frontend et le backend, et via **ROS2** pour les communications robot-station.

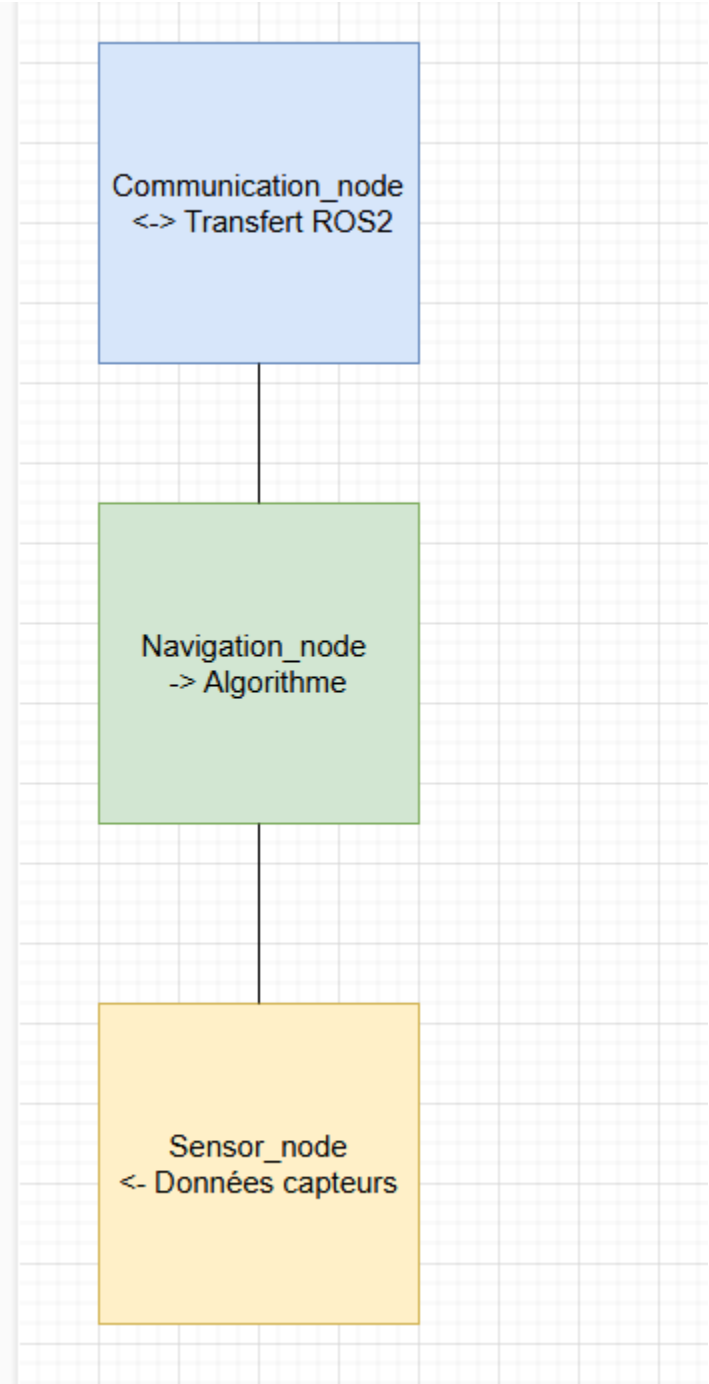
3.2 Station au sol (Q4.5)



Séparer la **gestion des données** et la **communication** permet de respecter les exigences de robustesse et de gestion en temps réel (R.F.3, R.F.8, R.C.1).

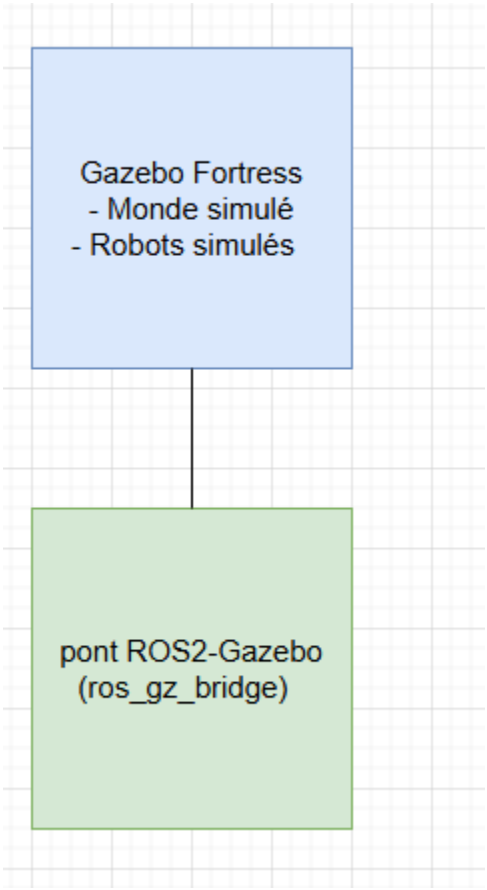
Utiliser **NestJS** assure une structure modulaire, et **PostgreSQL** garantit la persistance des données (missions, cartes, logs) (R.F.17, R.F.18).

3.3 Logiciel embarqué (Q4.5)



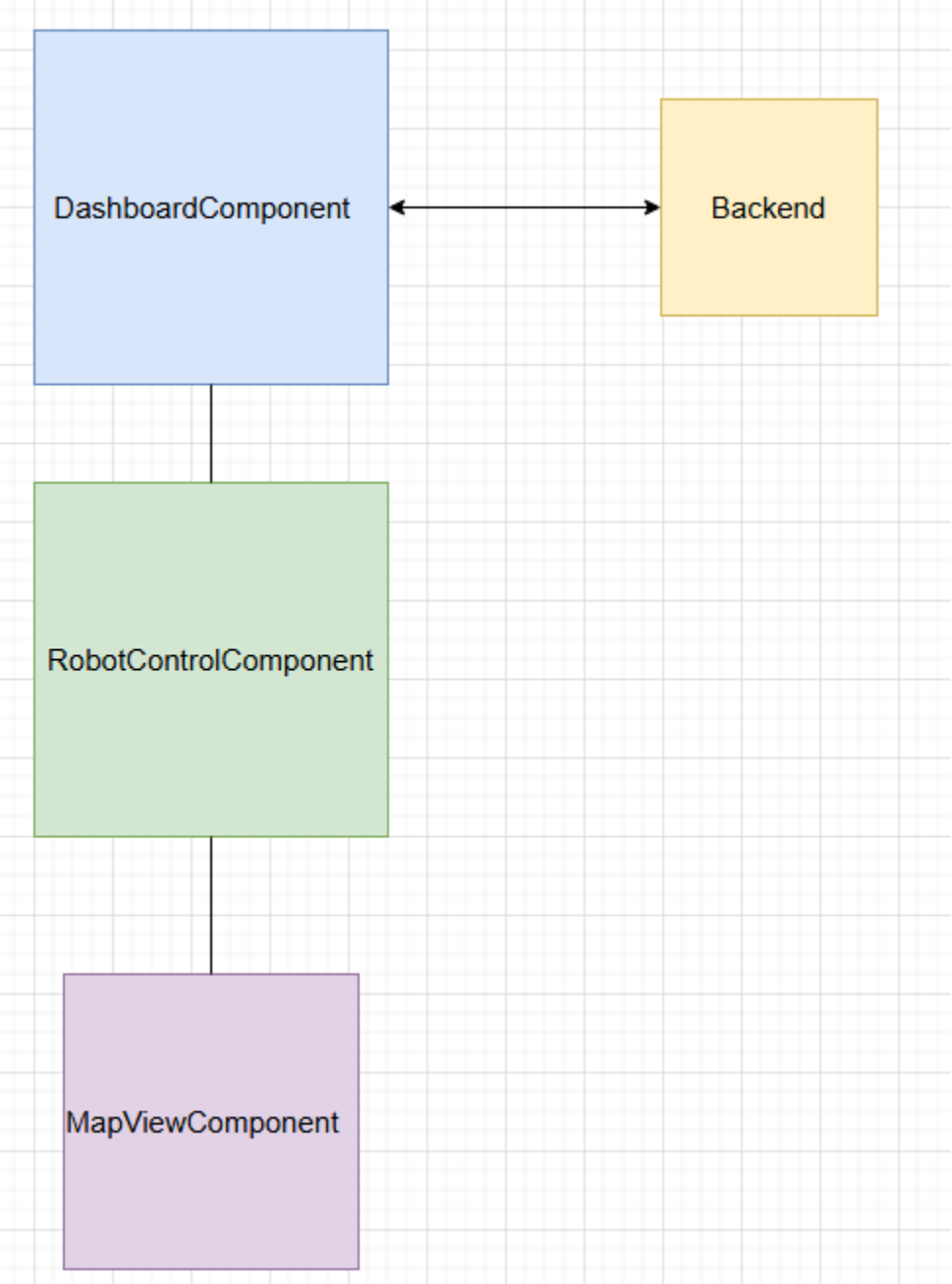
ROS2 structuré en nodes isole les responsabilités : communication, navigation, capteurs. Cela favorise la fiabilité et la maintenance (R.F.4, R.F.5, R.F.9). Permet de facilement gérer l'autonomie des robots sur le terrain et de respecter le fait que la station au sol n'envoie que des commandes de haut niveau.

3.4 Simulation (Q4.5)



Gazebo permet une évaluation réaliste des algorithmes de navigation et de collecte de données (R.F.4, R.F.5, R.C.3).
Le pont **ros_gz_bridge** permet l'interconnexion entre la simulation et le logiciel ROS2, assurant la compatibilité avec le code réel.

3.5 Interface utilisateur (Q4.6)



Séparer les composants Angular améliore la clarté et facilite la maintenance (R.F.3, R.F.8, R.C.4).
Dashboard : Supervision générale.
RobotControl : Commandes sur les robots.
MapView : Visualisation des cartes et positions en temps réel.

3.6 Fonctionnement général (Q5.4)

Pour exécuter le système :

1. ***Télécharger le dépôt GitLab** contenant le projet.*
2. ***Station au sol** : Lancer `docker-compose up` pour démarrer le backend NestJS, PostgreSQL et le serveur web.*
3. ***Interface utilisateur** : Accessible via un navigateur sur `localhost:4200`.*
4. ***Logiciel embarqué** : Démarrer ROS2 sur chaque robot avec les nodes nécessaires (`communication_node`, `navigation_node`, `sensor_node`).*
5. ***Simulation** : Lancer Gazebo avec le pont `ros_gz_bridge` et exécuter les nodes ROS2 de simulation.*
6. ***Exécution des missions** : Utiliser l'interface pour démarrer une mission, observer la carte et surveiller les robots.*
7. ***Tests** :*
    - *Tests unitaires : `npm test` (frontend), `npm run test` (backend), `pytest` (ROS2).*
    - *Tests d'intégration : Déployés via ROS2.*
    - *Tests E2E : `npx cypress open`.*

4. Processus de gestion

4.1 Estimations des coûts du projet (Q11.1)

Le coût total du projet est estimé en fonction du temps de travail alloué, des taux horaires définis pour chaque rôle au sein de l'équipe, ainsi que des coûts matériels requis pour la réalisation du système.

Conformément aux exigences du projet, la charge de travail maximale est de 630 heures-personnes. Avec une équipe composée de six membres, cela correspond à 105 heures de travail par personne.

Notre équipe est constituée de cinq développeurs-analystes et d'un coordinateur de projet. Le tableau suivant présente les coûts estimés en fonction des taux horaires et du temps alloué pour chaque rôle :

Rôle	Taux horaire ($CAN/h)	Temps alloué (h)	Coûts ($CAN)
Coordonnateur	145	105	15 225
Développeurs	130	525	68 250
Total estimé		**630**	**83 475**

Tableau 1 : Coût des ressources humaines

L'Agence Spatiale Canadienne (ASC) exige l'utilisation d'au moins deux robots **AgileX Limo Pro** pour le développement et la validation du système multi-robot. Le coût unitaire est estimé à 3 200 USD, soit 4 542,81 $CAN, sur la base d'un taux de change de 1,42 $CAN pour 1 USD en date du 13 février 2025.

Matériel	Coût unitaire ($CAN)	Quantité	Coût total ($CAN)
Robot AgileX Limo	4 542,81	2	9 085,62
Total estimé			**9 085,62**

Tableau 2 : Coûts matériels

Le tableau suivant résume les coûts totaux du projet :

Ressource	Coût ($CAN)
Ressources humaines	83 475
Matériel	9 085,62
Total estimé	**92 560,62**

Coûts Totaux

Le coût total estimé pour la réalisation du projet est donc de 92 560,62 $CAN. Cette estimation inclut le temps de travail nécessaire pour la planification, le développement, les tests, la validation et la documentation du système multi-robot, ainsi que les coûts matériels requis.

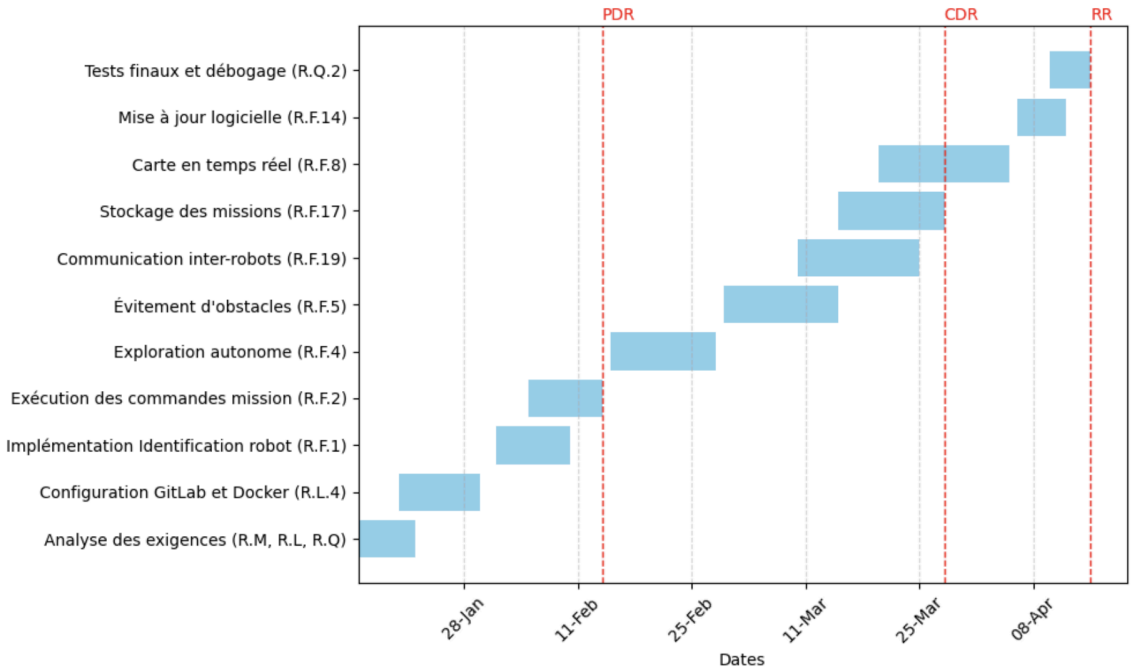
4.2 Planification des tâches (Q11.2)

La planification des tâches pour ce projet suit une approche méthodique et structurée, divisée en trois principaux jalons : PDR, CDR, RR.
L'organisation du travail suit une approche collaborative où chaque membre de l'équipe prend en charge un ensemble de requis fonctionnels et techniques.

Diagramme indiquant l'allocation du temps pour chaque tâche :

Tâches principales	Janvier	Février	Mars	Avril
Analyse des exigences (R.M, R.L, R.Q)	■■■■■■■			
Configuration GitLab & Docker (R.L.4)	■■■■■■■■			
Implémentation Identification robot (R.F.1)		■■■■■■		
Exécution des commandes mission (R.F.2)		■■■■■■■■		
Exploration autonome (R.F.4)		■■■■■■■■■■		
Évitement d'obstacles (R.F.5)			■■■■■■■	
Communication inter-robots (R.F.19)			■■■■■■■■■	
Stockage des missions (R.F.17)			■■■■■■■■	
Carte en temps réel (R.F.8)			■■■■■■■■	■■■■
Mise à jour logicielle des robots (R.F.14)				■■■■■■
Tests finaux et débogage (R.Q.2)				■■■■■■■■

Vue d'ensemble de l'horaire avec les 3 principaux jalons : PDR, CDR, RR

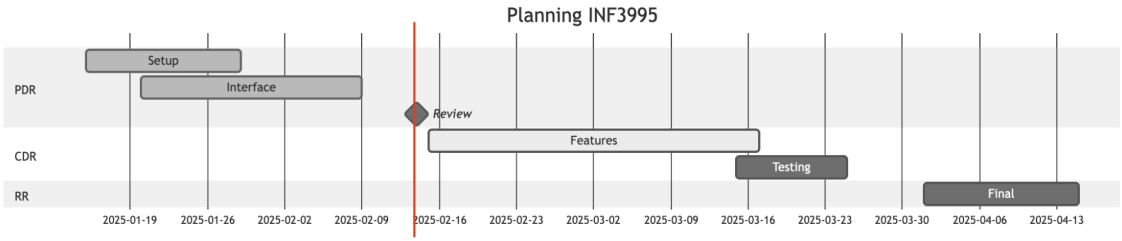


Division des tâches par rapports aux requis

Membre	Rôle	Responsabilités principales
Amine	Coordonnateur du projet	Gestion des ressources, suivi GitLab, validation des livrables
Kevin	Expert robotique & ROS	Implémentation des robots physiques *(R.F.1, R.F.2, R.F.4, R.F.5, R.F.19)*
Issam	Expert simulation & tests	Simulation Gazebo, tests d'exploration *(R.F.4, R.F.5, R.F.8)*
Yassine	Expert serveur & API	Gestion des services backend, communications *(R.F.2, R.F.19, R.F.14)*
Rafik	Expert base de données	Stockage et gestion des données *(R.F.17, R.F.18)*
Mario	Expert Frontend & UX	Interface utilisateur, affichage des cartes et données *(R.F.3, R.F.8, R.F.10, R.F.16)*

4.3 Calendrier de projet (Q11.2)

Tableau de différentes phases du projet



4.4 Ressources humaines du projet (Q11.2)

Notre équipe est composée de six membres aux compétences variées et complémentaires, assurant une couverture optimale de tous les aspects du projet. Elle inclut cinq programmeurs analystes et un coordonnateur de projet.

Les programmeurs analystes sont responsables de la conception, du développement, des tests et de la documentation du système. Ils assurent notamment l'intégration du frontend, l'optimisation des communications avec le serveur et la programmation des robots pour l'exécution des différentes missions.

Le coordonnateur de projet joue un rôle central dans la gestion des ressources, le suivi des tâches et le respect des échéances et des normes d'équipe. Plutôt qu'un modèle strictement centralisé, notre organisation repose sur un fonctionnement décentralisé favorisant la collaboration et l'autonomie.

Le suivi des tâches et des contributions est assuré via GitLab, garantissant une gestion efficace et transparente. Le tableau ci-dessous présente la répartition des rôles, les qualifications et les principales responsabilités de chaque membre de l'équipe.

Rôle	Membre	Qualifications principales	Responsabilités principales
Coordonnateur de projet	Amine	Gestion de projet, suivi GitLab, revue de code	Supervision générale, gestion des ressources et des échéances, validation des contributions
Expert ROS et robotique	Kevin	Développement ROS, programmation robotique	Implémentation et optimisation des comportements des robots
Expert simulation et tests	Issam	Simulation Gazebo, tests en simulation	Développement et validation des tests en simulation
Expert serveur et tests	Yassine	Développement backend, tests unitaires	Implémentation et maintenance du serveur, validation des fonctionnalités
Expert serveur et base de données	Rafik	Bases de données, architecture serveur	Conception et gestion des bases de données, intégration backend
Expert frontend et UX	Mario	Développement web, UX/UI, communication serveur	Conception et optimisation de l'interface utilisateur, intégration frontend-backend

Il est à noter que tous les membres participent au développement sous ROS, et les responsabilités peuvent évoluer selon les besoins du projet.

5. Suivi de projet et contrôle

5.1 Contrôle de la qualité (Q4)

La qualité des biens livrables est essentielle pour garantir la fiabilité et la robustesse du projet. Notre équipe met en place un processus rigoureux de révision et de validation à plusieurs niveaux afin d'assurer la conformité des livrables aux exigences du projet.

1. Révision des livrables clés : PDR, CDR et RR

Chaque phase du projet (Preliminary Design Review, Critical Design Review et Readiness Review) est soumise à une révision approfondie selon les critères suivants :

- **Vérification de la complétude** : Chaque livrable est évalué en fonction des requis attendus (fonctionnels, matériels, logiciels, qualité, etc.).
- **Relecture par l'équipe** : Tous les membres du groupe doivent passer en revue les documents et apporter des corrections avant la soumission.
- **Feedback et corrections** : Des itérations sont effectuées en fonction des retours reçus, afin d'améliorer la clarté et la précision du contenu.

2. Utilisation des Merge Requests et d'une norme de commits

Afin d'assurer un code de qualité et une gestion efficace du développement logiciel, nous mettons en place les pratiques suivantes :

- **Merge Requests (MR)** : Toute modification du code doit passer par une MR, qui sera revue par au moins un autre membre de l'équipe avant d'être fusionnée dans la branche principale.
- **Norme de commits** : Les commits doivent suivre un format standardisé (ex. *feat: ajout de la détection d'obstacles* ou *fix: correction d'un bug d'affichage*), permettant une meilleure traçabilité et compréhension de l'évolution du projet.
- **Revue de code systématique** : Chaque MR doit être accompagnée d'une description claire des changements et de leur impact, et être validée par au moins un autre membre.

3. Tests et validation continue

- **Tests unitaires** : Chaque composant logiciel doit être testé individuellement pour assurer son bon fonctionnement.
- **Tests d'intégration** : Des tests sont effectués régulièrement pour s'assurer de la compatibilité entre les différentes composantes du système (interface utilisateur, robots, simulation).
- **Démonstrations vidéo** : Chaque fonctionnalité clé est validée par une démonstration filmée, permettant de vérifier son bon fonctionnement et de fournir une preuve de conformité aux exigences.

Grâce à ces pratiques, nous nous assurons que les livrables sont de qualité et conformes aux attentes du projet.

5.2 Gestion de risque (Q11.3)

Tout projet comporte des risques, et le nôtre ne fait pas exception. L'important est d'anticiper ces risques et de s'adapter rapidement si notre plan initial ne se déroule pas comme prévu.

Voici les principaux risques identifiés pour ce projet, accompagnés de leur importance estimée, ainsi que des solutions de remplacement et des stratégies d'atténuation mises en place par notre équipe.

Il y a un risque de précision insuffisante des robots. Nos robots pourraient ne pas être assez précis et renvoyer des données inexactes sur leur perception de l'environnement. Leur mouvement de parcours pourrait également manquer de précision en raison des imprévus du monde réel, comme l'usure du matériel ou la variation du niveau de batterie. Un comportement fonctionnel en simulation pourrait ne pas être exactement reproductible sur les robots physiques. Pour remédier à ce risque, nous effectuons de nombreux tests physiques afin de mieux comprendre les paramètres influençant la précision. Grâce à ces informations, nous optimisons les algorithmes en simulation et ajustons efficacement le code embarqué en prévoyant des calibrations adaptées.

Un autre risque important à considérer est la limitation du réseau de communication. Déjà avertis par nos chargés de laboratoire, nous savons que la connexion WiFi entre la station au sol et les robots pourrait ne pas être en mesure de supporter la quantité de données que nous aimerions transmettre, particulièrement lors de la construction de la carte générée par les capteurs. Ce problème pourrait entraîner une perte de données ou un ralentissement du système. Pour atténuer ce risque, nous devons d'abord tester la capacité maximale du réseau afin d'identifier les limites de la bande passante. Ensuite, nous mettrons en place des mécanismes de reconnexion automatique ainsi que des comportements par défaut en cas de coupure de communication. Une autre solution complémentaire est de minimiser la quantité de données transmises en optimisant leur format et en priorisant les informations essentielles, ce qui réduira la charge sur le réseau.

Le respect des délais représente également un risque majeur. Le développement de certains requis techniques pourrait prendre plus de temps que prévu, ce qui compromettrait l'échéancier global et risquerait d'entraîner une livraison bâclée des livrables. Pour éviter un tel scénario, nous avons mis en place un suivi rigoureux des tâches avec des jalons intermédiaires qui nous permettent d'évaluer l'avancement du projet et d'ajuster le planning en conséquence. Nous nous assurons également de prévoir une marge de manœuvre pour les imprévus et d'adopter une gestion agile du projet avec des revues régulières pour identifier les dépendances critiques dès le début et ajuster nos priorités en conséquence.

Nous devons aussi prendre en compte le risque d'intégration difficile entre les différentes composantes du système. La communication entre l'interface web, les robots et la simulation Gazebo est un enjeu crucial, et ces éléments, développés indépendamment, doivent rester compatibles tout au long du projet. Une mauvaise intégration pourrait entraîner des dysfonctionnements et ralentir notre progression. Pour prévenir ce problème, nous avons défini des interfaces de communication claires dès le départ et nous effectuons des tests d'intégration réguliers au lieu d'attendre la fin du développement. Cela nous permet de

détecter rapidement les éventuels problèmes de compatibilité et de les résoudre avant qu'ils ne deviennent bloquants.

Il existe également un risque de bris matériel. Les robots ou certains capteurs pourraient être endommagés au cours des tests, ce qui compromettrait nos essais physiques et ralentirait notre avancement. Pour limiter ce risque, nous manipulons les équipements avec précaution en suivant les recommandations du fabricant. En cas de panne, nous avons aussi prévu un plan de repli où la simulation Gazebo pourra être utilisée temporairement afin de poursuivre le développement du projet sans interruption.

Un autre risque à prendre en compte est l'abandon ou l'indisponibilité prolongée d'un membre de l'équipe. Une maladie, une surcharge de travail ou un engagement imprévu pourrait empêcher un membre de contribuer au projet pendant une période critique. Pour éviter que cela n'affecte notre progression, nous nous assurons de bien répartir les tâches et d'éviter qu'une seule personne ne soit responsable d'un aspect essentiel du projet. Nous maintenons également une documentation détaillée de nos avancées afin qu'un autre membre puisse reprendre facilement une tâche en cas d'absence prolongée.

Le risque de conflits au sein de l'équipe ne doit pas être négligé non plus. Même si nous avons une bonne entente et une communication efficace, des désaccords pourraient survenir quant aux décisions techniques ou organisationnelles, ce qui pourrait ralentir le développement et nuire à la motivation générale. Pour prévenir cela, nous privilégions une communication ouverte et transparente, où chaque membre peut exprimer ses idées et préoccupations. Les décisions importantes seront prises collectivement en s'appuyant sur des critères objectifs. En cas de désaccord persistant, nous mettrons en place un processus de médiation interne afin d'éviter que le conflit n'affecte le projet.

Enfin, nous devons anticiper les difficultés techniques imprévues. Certains algorithmes, comme ceux d'exploration autonome ou d'évitement d'obstacles, pourraient s'avérer plus complexes à implémenter que prévu, ce qui pourrait ralentir notre progression. Pour gérer ce risque, nous avons identifié dès le début du projet les tâches les plus complexes et nous les avons traitées en priorité. Nous nous assurons également de toujours avoir des solutions alternatives plus simples si un problème persiste. En cas de besoin, nous pourrons consulter nos chargés de laboratoire ou explorer des solutions déjà documentées pour gagner du temps.

En appliquant ces stratégies, notre équipe sera en mesure d'anticiper et de gérer efficacement les risques afin de mener à bien le projet.

5.3 Tests (Q4.4)

Les tests effectués dans ce projet visent à garantir la fiabilité, la performance et l'interopérabilité entre les différentes composantes du système. Ils couvrent à la fois le **matériel** (robots) et le **logiciel** (client, serveur, simulation).

1. Tests du Client (Frontend – Angular)

Objectif : Vérifier l'affichage, l'interaction et la communication avec le backend.

- **Tests unitaires** : Vérification des composants et services (*.spec.ts avec Jasmine/Karma).
    - Ex. : Test du composant de contrôle des robots (control-panel.component.spec.ts).
- **Tests d'interface utilisateur** : Validation de l'affichage et de la navigation dans l'interface.
- **Tests d'intégration** : Simulation des requêtes entre le frontend et le backend.

2. Tests du Serveur (Backend – NestJS)

Objectif : Vérifier le bon fonctionnement des API REST et WebSockets.

- **Tests unitaires** : Validation des services (*.service.spec.ts avec Jest).
    - Ex. : Vérification du robot.service.ts pour l'envoi des commandes aux robots.
- **Tests API** : Vérification des endpoints avec Postman ou Supertest (*.controller.spec.ts).
- **Tests de performance** : Mesure des temps de réponse sous charge.

3. Tests des Robots (Système embarqué – ROS2)

Objectif : Assurer le bon fonctionnement des robots physiques.

- **Tests de navigation** : Vérification du respect des trajectoires définies.
- **Tests de détection d'obstacles** : Validation de l'évitement en fonction des capteurs.
- **Tests de communication** : Vérification de l'échange de messages entre les robots et la station au sol.

4. Tests de la Simulation (Gazebo)

Objectif : Vérifier la cohérence entre la simulation et le monde réel.

- **Tests d'exploration** : Vérification de l'algorithme de déplacement autonome.

- **Tests de cartographie** : Comparaison des cartes générées avec l'environnement réel.

5.4 Gestion de configuration (Q4)

Système de Contrôle de Version

L'équipe utilise **GitLab** avec une gestion rigoureuse du code source et de la documentation.

- **Branches principales :**
  - master : Code stable, protégé contre les modifications directes.
  - develop : Intégration continue des nouvelles fonctionnalités avec tests automatisés.
- **Branches spécifiques :**
  - feature/ : Nouvelles fonctionnalités (ex. feature/robot-identification).
  - hotfix/ : Corrections urgentes fusionnées vers master et develop.
- **Commits :** Messages clairs en français, commits atomiques, référence aux issues GitLab.

Organisation du Code Source

Le projet suit une architecture modulaire :

- **Client (/client)** : Frontend Angular (composants, services, modèles).
- **Serveur (/server)** : Backend NestJS (API REST, WebSockets, logique métier).
- **Robot (/robot)** : Système ROS2 (contrôle, communication, navigation).

Organisation des Tests

- **Unitaires** : Tests pour chaque composant (Jasmine/Karma pour Angular, Jest pour NestJS).
- **Intégration** : Tests API et WebSockets simulant des scénarios réels.
- **Robotiques** : Vérification des capteurs et de la navigation autonome.
- **CI/CD** : Tests automatisés et déploiement continu via GitLab CI/CD.

Gestion des Données

- **Base de données PostgreSQL** : Enregistrement des missions, positions des robots et métriques de performance.
- **Fichiers de configuration** : Paramètres des robots et de la simulation centralisés en YAML.
- **Logs et métriques** : Historique détaillé des missions et suivi des performances.

Documentation

- **Techniques** : ARCHITECTURE.md, diagrammes, spécifications détaillées.
- **API** : Documentation intégrée aux fichiers source.
- **Installation** : Instructions de build et déploiement (README.md).
- **Utilisation** : Guides pour interface utilisateur et dépannage.

Cette approche assure une séparation claire des responsabilités et une traçabilité efficace du projet.

6. Références (Q3.2)

Agilex. "limo-pro." https://global.agilex.ai/products/limo-pro (accédé le 13 février 2025).

ANNEXES

[Inclure toute documentation supplémentaire utilisable par le lecteur. Ajouter ou référencer toute norme technique de projet ou plans applicables au projet.]

Regroupement des Requis

1. Requis Généraux

- **R.G.1** : Le système doit permettre l'exploration autonome d'une pièce par une équipe de robots.
- **R.G.2** : Une station au sol doit permettre la supervision et l'interaction avec les robots.

2. Requis Matériels

- **R.M.1** : Utilisation obligatoire des robots AgileX Limo.
- **R.M.2** : Communication unique via le réseau WiFi fourni par l'Agence.
- **R.M.3** : Seuls les capteurs fournis par l'Agence sont autorisés.
- **R.M.4** : La station au sol doit être un laptop ou un PC.

3. Requis Logiciels

- **R.L.1** : OS Ubuntu requis pour les robots, machine virtuelle autorisée.
- **R.L.2** : Interface utilisateur identique pour simulation et robots physiques.
- **R.L.3** : Commandes de haut niveau seulement, pas de contrôle direct des robots.
- **R.L.4** : Conteneurisation Docker obligatoire pour toutes les composantes logicielles, sauf pour les robots physiques.

4. Requis Fonctionnels

- **R.F.1** : Identification individuelle des robots via l'interface utilisateur.
- **R.F.2** : Commandes "Lancer la mission" et "Terminer la mission" disponibles.
- **R.F.3** : Affichage de l'état des robots à une fréquence d'au moins 1 Hz.
- **R.F.4** : Exploration autonome de l'environnement.
- **R.F.5** : Évitement des obstacles détectés.
- **R.F.6** : Retour à la base avec précision de 0,3 m.
- **R.F.7** : Retour automatique à la base en cas de batterie faible (<30%).
- **R.F.8** : Génération d'une carte de l'environnement en temps réel.
- **R.F.9** : Affichage continu de la position des robots.
- **R.F.10** : Interface utilisateur accessible sur PC, tablette et téléphone.
- **R.F.11** : Carte en 3D et en couleur.
- **R.F.12** : Spécification de la position initiale des robots.

- **R.F.13** : Détection et évitement des élévations négatives.
- **R.F.14** : Mise à jour du logiciel de contrôle des robots via l'interface utilisateur.
- **R.F.15** : Support de deux modes de contrôle des roues.
- **R.F.16** : Éditeur de code intégré pour modifier le comportement des robots.
- **R.F.17** : Base de données enregistrant les missions.
- **R.F.18** : Sauvegarde et inspection des cartes générées.
- **R.F.19** : Communication P2P entre robots pour transmettre la distance à la base.
- **R.F.20** : Implémentation d'une zone de sécurité (geo-fence).

5. Requis de Conception

- **R.C.1** : Disponibilité des logs de débogage en continu.
- **R.C.2** : Déploiement du logiciel avec une seule commande.
- **R.C.3** : Génération aléatoire de l'environnement virtuel dans Gazebo.
- **R.C.4** : Interface utilisateur respectant les heuristiques de Nielsen.
- **R.C.5** : Compatibilité avec un ou deux robots sans configuration manuelle.

6. Requis de Qualité

- **R.Q.1** : Code standardisé suivant des conventions reconnues.
- **R.Q.2** : Tests unitaires ou procédures de test pour chaque fonctionnalité.