

INF3995 – Hiver 2025 : Exploration Multi-Robot Équipe 102



**POLYTECHNIQUE
MONTRÉAL**

UNIVERSITÉ
D'INGÉNIERIE

Conception d'un système d'exploration multi-robot

Amine Zerouali - 2132908

Kevin Santiago Gratton Fournier - 2193996

Issam Haddadi - 2066062

Rafik Hachemi Boumila - 2217969

Yassine Mohamed Taha Abassi - 2127936

Mario Junior Milord - 2128026



Source : Limo ROS2 [1]



Source : Image générée par une IA

Introduction & Contexte du projet

Introduction & Contexte du projet

- Exploration planétaire : plusieurs robots simples, autonomes, et collaboratifs
- Besoin d'un système fiable, autonome, et supervisé à distance
- Projet simulant un contrat réel entre sous-traitant et agence spatiale

Objectif:

- Conception d'un **système d'exploration multi-robot complet**
- Utilisation de **deux robots LIMO** avec capteurs embarqués (lidar, IMU, caméras)
- Développement de **trois composantes intégrées** :
 - Station au sol (interface web – supervision, commandes)
 - Robots physiques (déplacement autonome, collecte de données)
 - Simulation Gazebo (tests des algorithmes)





1. Structure d'organisation

1. Structure d'organisation : Méthodologie




Approche Agile – modèle Scrum

- L'équipe fonctionne en sprints de 1,5 mois avec planification en début de cycle.
- Méthode choisie pour favoriser l'adaptation rapide, la collaboration et la responsabilisation.
- Réunions hebdomadaires : suivi d'avancement, résolution de problèmes, ajustement des tâches.

Principes guidant l'organisation

-  Priorité aux interactions humaines
-  Logiciels fonctionnels avant la documentation
-  Collaboration avec les parties prenantes
-  Adaptation continue plutôt que plan figé

Répartition des rôles

-  *Scrum Master* (Kevin Santiago G. Fournier) : facilite la communication et l'organisation.
-  *Product Owner* (Issam Haddadi) : vision du projet, suivi des requis, lien avec les encadrants.
-  *Équipe de développement* (tous) : conception, développement, tests et intégration.










Outils utilisés

- GitLab pour le suivi des tâches : *issues*, assignation par jalons (PDR, CDR, RR).
- Assure une bonne traçabilité et transparence dans le déroulement du projet.



Source : Image générée par une IA

1. Structure d'organisation : Répartition des Responsabilités

 Rôle	 Responsable	 Responsabilités principales
 Coordonnateur de projet	Amine	Supervision globale, gestion des ressources, échéances, revue de code
 Expert ROS & robotique	Kevin	Implémentation et optimisation des comportements robotiques
 Expert simulation & tests	Issam	Développement et validation des tests en simulation (Gazebo)
 Expert serveur & tests	Yassine	Implémentation backend, tests unitaires, validation fonctionnelle
 Expert base de données	Rafik	Conception des bases de données, intégration backend
 Expert frontend & UX	Mario	UI/UX, intégration frontend-backend, communication serveur



Source : Image générée par une IA

1. Structure d'organisation : Entente contractuelle

- Échéances fixes : PDR – 14 février, CDR – 28 mars, RR – 15 avril
- Charge de travail plafonnée à 630 heures-personnes
- Gestion stricte du temps par tâche
- Objectif pédagogique : Acquisition de compétences techniques, Gestion de projet
- Entente à terme adaptée aux exigences du cours INF3995



Source : Image générée par une IA

2. Vue d'ensemble de la solution développée

2. Vue d'ensemble de la solution développée : Requis optionnels choisis

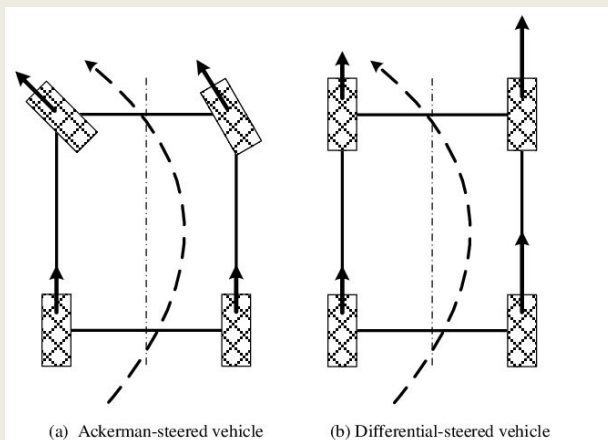
R.F.15 – Deux modes de contrôle (5 pts) : Ackermann + différentiel

R.F.17 – BDD missions (5 pts) : historique, tri, consultation Web

R.F.18 – Cartes sauvegardées (5 pts) : ouverture + inspection dans l'interface

R.F.19 – P2P robots (5 pts) : distance, icône, sans serveur

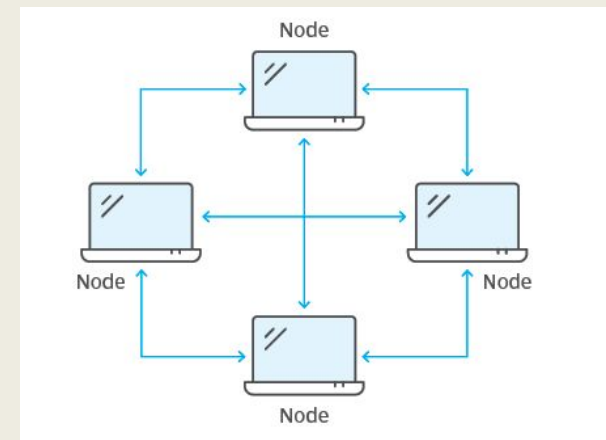
R.C.5 – Détection automatique robots (5 pts) : adaptation à 1 ou 2 robots sur gazebo



Source : ResearchGate [1]



Source : OVH cloud [2]



Source : P2P [3]

Mario - 2128026

2. Vue d'ensemble de la solution développée : Architecture modulaire et séparée

Principe fondamental

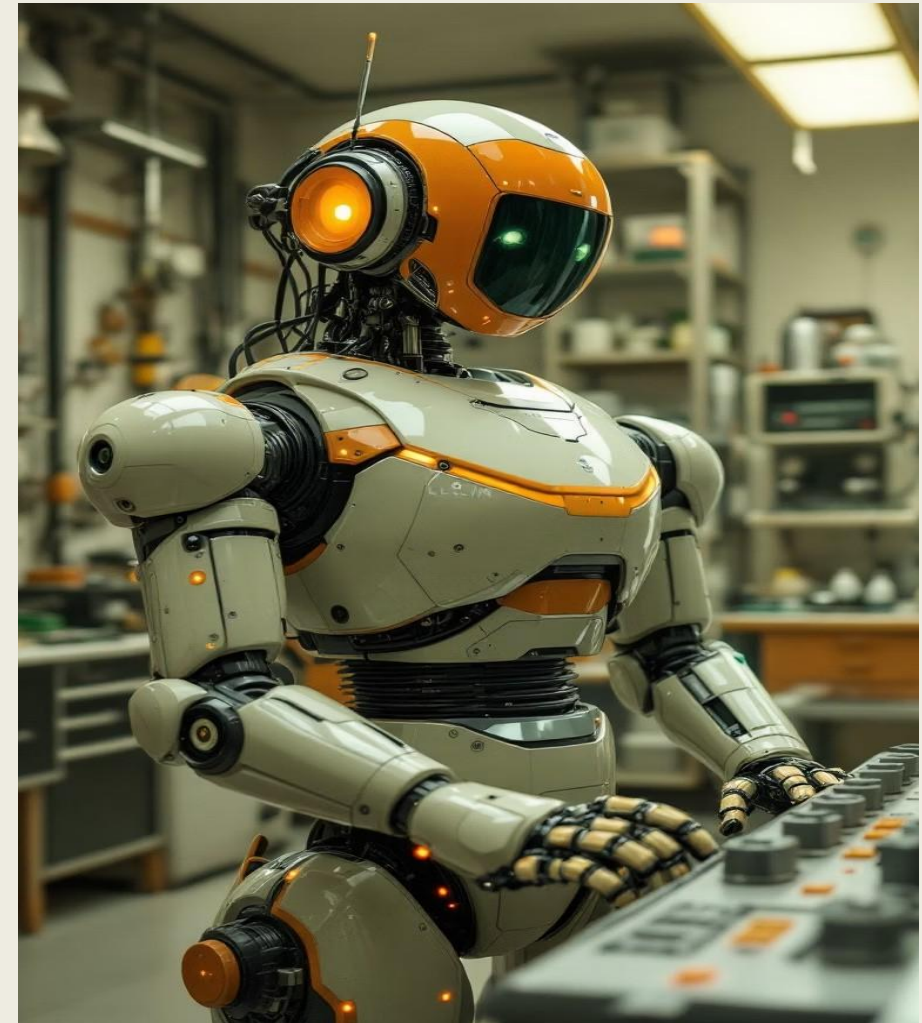
Séparation claire entre **communication** et **gestion des données**, assurant robustesse, performance temps réel et résilience.

📌 Bénéfices clés

- 📡 *R.F.3* : Affichage des états à 1 Hz, sans conflit avec la base de données
- 🗺️ *R.F.8* : Gestion fluide des cartes pendant les missions
- 🔄 *R.C.1* : Résilience aux erreurs réseau, continuité des logs

🔧 Modules principaux (NestJS)

- 🤖 *Interface Robot* : réception des messages ROS et redirection vers les services
- 🗺️ *Cartographie* : fusion et mise à jour des cartes locales et globale
- 🧭 *Navigation* : planification des trajectoires, gestion des obstacles
- 💾 *Base de données* : accès structuré (robot, mission, timestamp)
 - *R.F.17* : enregistrement des missions
 - *R.F.18* : sauvegarde persistante des cartes



2. Vue d'ensemble de la solution développée : Schéma

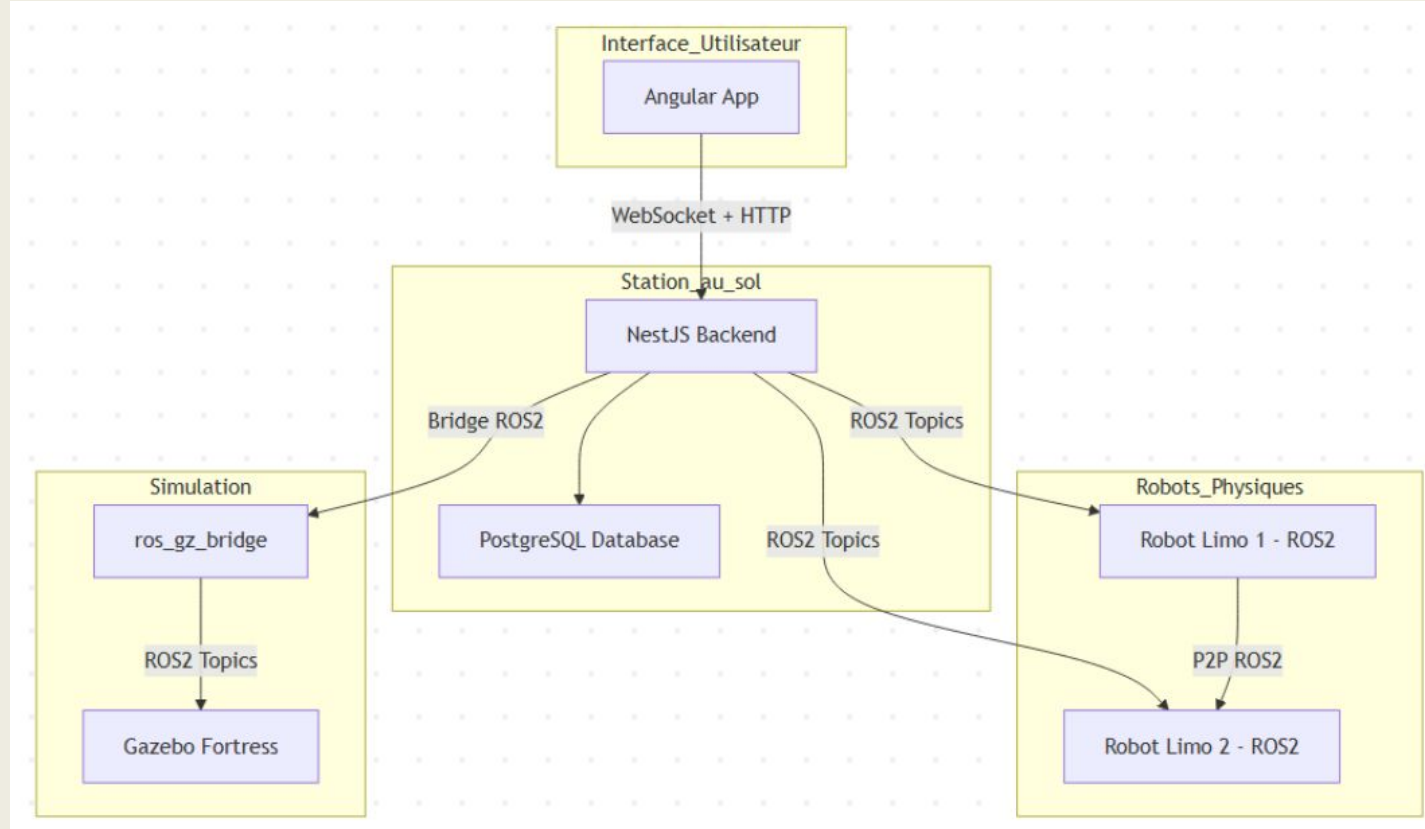
Station au sol : application Web Angular pour démarrer, surveiller et contrôler la mission

Robots embarqués : deux robots AgileX LIMO autonomes, communiquant via ROS2

Simulation : environnement Gazebo connecté à l'interface Web pour tests réalistes

Base de données PostgreSQL : stockage des cartes, journaux et historiques de missions

P2P : Communication directe P2P entre robots (sans serveur intermédiaire)



3. Conception technique

3. Conception technique : Interface Web

Rôle central

- Lancement de missions
- Supervision des robots en temps réel
- Consultation des cartes
- Réaction aux événements critiques (perte, obstacle, etc.)

Développement avec Angular

- Architecture modulaire
- Composants spécialisés
- Communication temps réel via **WebSocket + API NestJS**

Composants clés

DashboardComponent

- Vue d'ensemble (états, batterie, logs, mission)
- Rafraîchissement 1 Hz via WebSocket (*R.F.3*)

RobotControlComponent

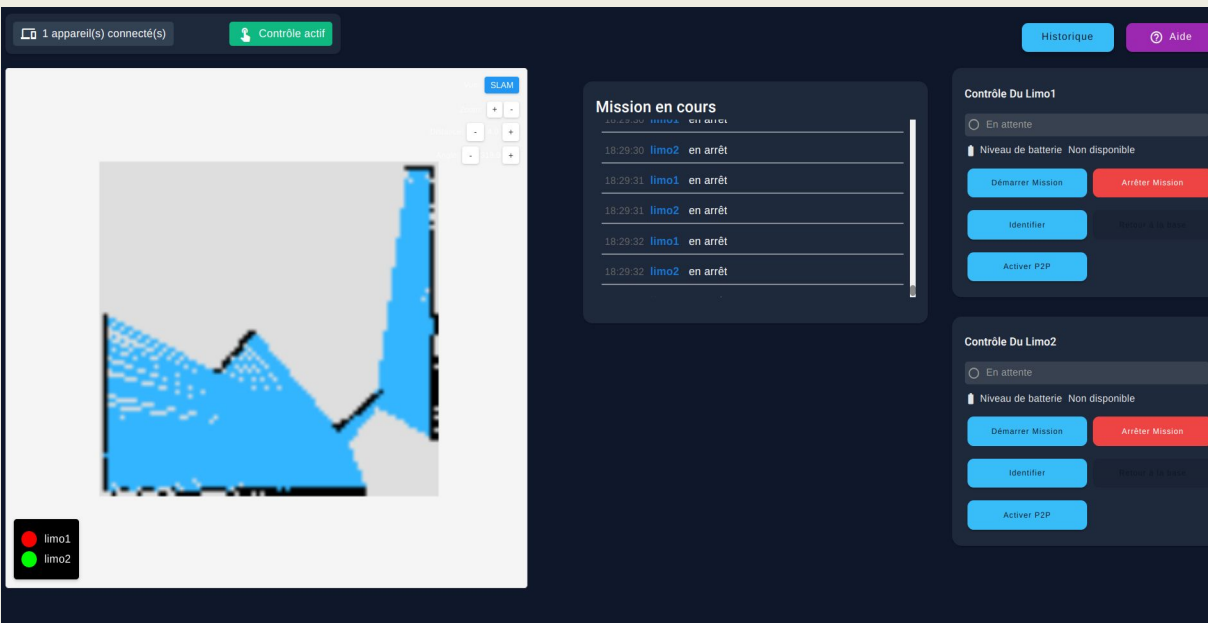
- Commandes : **start_mission**, **end_mission**, **go_base**
- Interaction en quasi temps réel avec les robots
- Suivi visuel de la mission en cours (*R.F.9*)

MapViewComponent

- Carte 2D, position et trajet des robots
- Objets détectés, obstacles
- Affichage dynamique des cartes via WebSocket (*R.F.8*)

API NestJS

- WebSocket : données temps réel (états, cartes, logs)
- HTTP : commandes vers les robots (réels ou simulés)
- Centralisation + redirection vers ROS2 et PostgreSQL



3. Conception technique : station au sol

- API REST & WebSocket pour communication avec frontend et ROS 2
- Sauvegarde des missions dans MongoDB
- API pour consultation et tri des missions
- Connexion en temps réel avec robots et simulation
- Authentification P2P entre robots via le backend
- Chargement et synchronisation des cartes historiques
- Support des logs accessibles via interface



PostgreSQL



NestJS



WebSocket

1 appareil(s) connecté(s)

Contrôle actif

Retour

Aide

Historique des Missions

Charger les missions

ID Mission	Date Début	Date Fin
15ccc7f1-4279-44e1-9e8b-f8adcb6e62ed	16/04/2025 18:29:58	16/04/2025 18:30:40
a18f6781-59fb-447d-bc1d-ac8b06cf7c1a	16/04/2025 18:16:48	16/04/2025 18:18:21
0fc34c87-1605-404d-8386-9c9036b64305	16/04/2025 18:09:22	
20fd20ed-eb37-4818-81f0-98fcd2d0af69	16/04/2025 18:02:29	
d4136a59-c44a-45f7-a761-9b32c05042d9	16/04/2025 17:41:59	16/04/2025 17:45:35
f8e89f41-b2ed-45b8-af44-f88e9d4eee0f	16/04/2025 17:27:52	
98717ce7-fadb-460e-9b11-add8c871e594	16/04/2025 17:10:10	
67370f3d-3611-4b96-8722-6e02cbffeeb1	16/04/2025 17:05:17	16/04/2025 17:07:22
5cb5c7c6-fba0-42e1-9366-48edba574234	16/04/2025 10:53:39	
6d8735bd-022c-47b1-97a5-1a911412a707	16/04/2025 10:53:32	

Yassine Abassi

- 2127936

3. Conception technique : simulation

Gazebo Fortress

- Environnement avec obstacles, sol irrégulier
- Robots Limo simulés avec capteurs (Lidar, IMU, Odom)
- Tests des exigences :
 - R.F.4 : navigation autonome
 - R.F.5 : évitement d'obstacles
 - ⚠ Situations critiques (batterie faible, blocage)

ros_gz_bridge

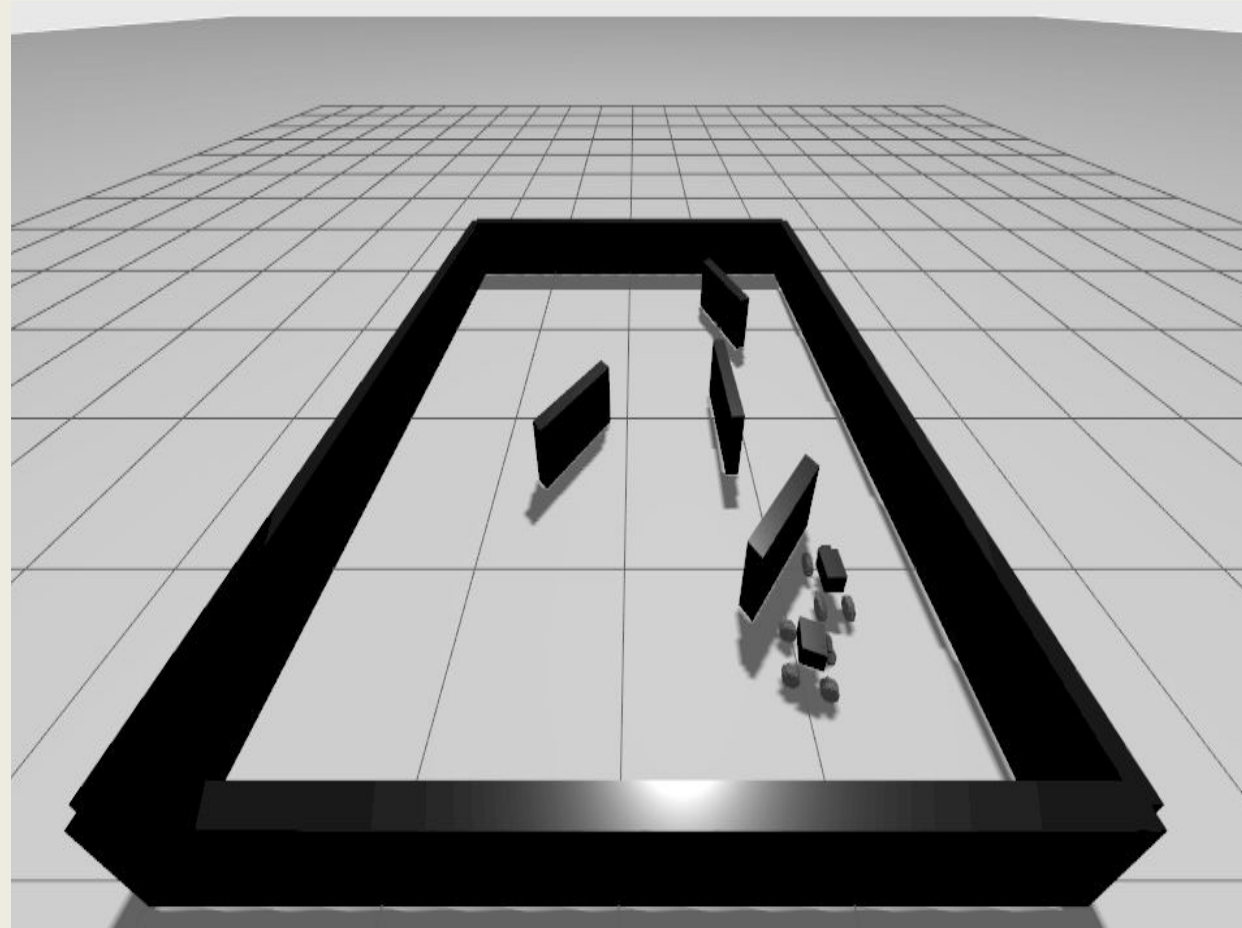
- Pont ROS2 ↔ Gazebo : `/cmd_vel`, `/scan`, `/odom`, etc.
- R.C.3 : Même code utilisé en simulation et en réel

Robots simulés

- Exécutent les mêmes nœuds ROS2 que les robots physiques
- Interaction avec l'environnement et la station via ROS2 topics

Station au sol (NestJS)

- Commandes et retours en temps réel
- Interface Angular fonctionnelle en simulation comme en réel



3. Conception technique : Robots LIMO

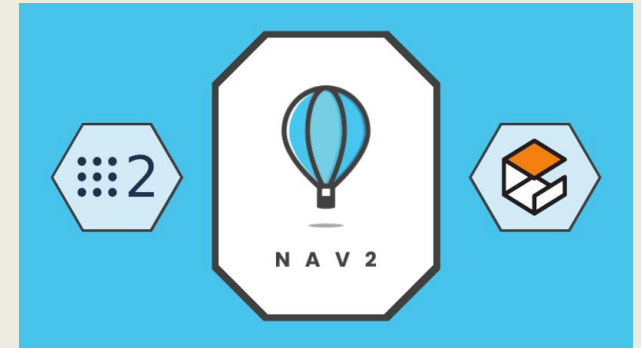
- Plateformes robotiques fournies avec ROS2 préinstallé
- Capteurs embarqués : Lidar, Odom, Imu
- Communication en WiFi 2.4/5 GHz avec la station
- Navigation autonome avec évitement d'obstacles
- Retour à la base manuel ou automatique (batterie < 30%)



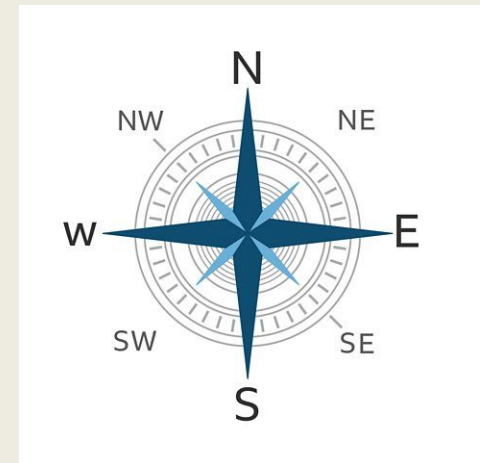
Source : Limo ROS2 [4]

3. Conception technique : Technologies et algorithmes utilisés

1. **ROS2** : fondation logicielle pour tous les composants
2. **Nav2** : planification de trajectoires et évitement des obstacles
3. **Slam Toolbox** : création des cartes 2D en temps réel
4. **Algorithme d'exploration personnalisé** :
 - Le robot évalue les 8 directions autour de lui
 - Sélectionne la meilleure cellule selon coût et l'occupation
 - Choix différencié entre les deux LIMO pour explorer des zones distinctes
 - Système léger, efficace et réactif



Source : ROS2 Nav2 [5]



Source : Points of the compass [6]

3. Conception technique : Logiciel Embarqué ROS2

Architecture modulaire ROS2

- Chaque fonctionnalité est un **nœud indépendant**
- Communication via **topics ROS2**
- + Robustesse, fiabilité, maintenance facilitée
- Respect des requis : *R.F.4, R.F.5, R.F.9*

Avantages de la structure en nœuds

- Répartition claire des responsabilités
- Réutilisabilité dans d'autres projets
- Résilience : un nœud peut échouer sans bloquer tout le système

Principaux nœuds embarqués

• Nœud Communication

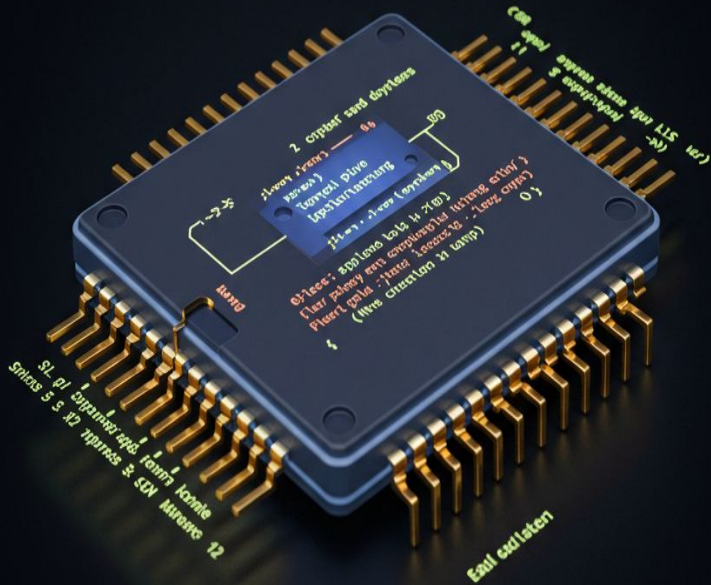
- Publie état (position, batterie, statut)
- Reçoit les commandes de mission
- Gère la connexion WebSocket avec ROSBridge

• Nœud map merge

- Fusionne dynamiquement toutes les cartes disponibles en une seule carte globale partagée

• Nœud Navigation

- Planifie les trajectoires
- Évite les obstacles en temps réel
- Met à jour l'état de progression de la mission



3. Conception technique : Stratégies de tests

🎯 Objectif global

Garantir la fiabilité, la performance et l'interopérabilité entre toutes les composantes du système (client, serveur, robots, simulation).

💻 1. Tests Client (Angular)

- ✅ *Unitaires* : composants/services (.spec.ts, Jasmine)
- 👁 *UI* : affichage, navigation
- 🔗 *Intégration* : échanges frontend ↔ backend

💻 2. Tests Serveur (NestJS)

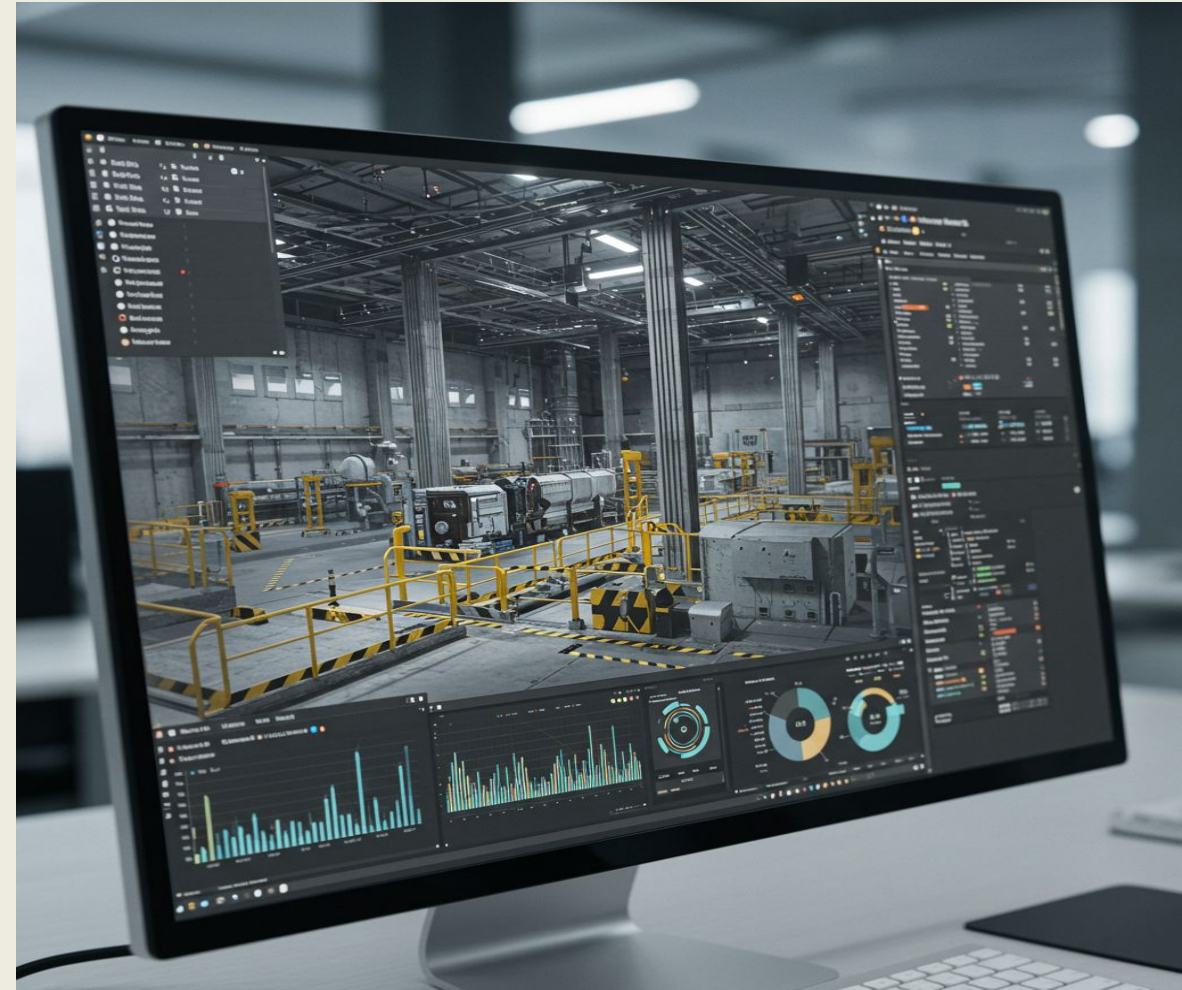
- ✅ *Unitaires* : services (Jest)
- 🌐 *API* : endpoints REST/WebSocket (Postman, Supertest)
- 📊 *Performance* : temps de réponse sous charge

🤖 3. Tests Robots (ROS2)

- 🧭 *Navigation* : respect des trajectoires
- ⚠ *Obstacles* : évitement via capteurs
- 🔄 *Communication* : messages ROS2 ↔ station au sol

🔧 4. Tests Simulation (Gazebo)

- 🗺 *Exploration* : déplacement autonome
- 🧱 *Cartographie* : cohérence avec l'environnement réel







3. Conception technique : Contrôle de la qualité




1. Révision des livrables (PDR, CDR, RR)

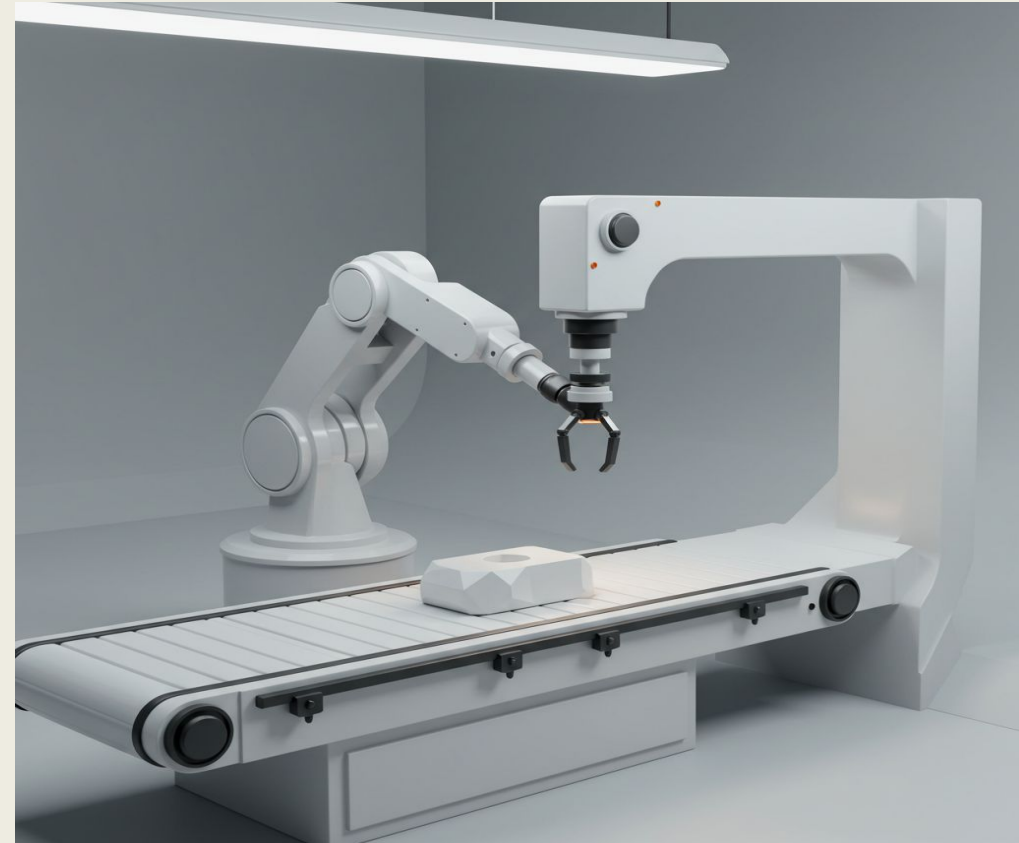
- Vérification de la complétude (fonctionnelle, matérielle, logicielle)
- Relecture collective avant soumission
- Itérations selon les retours pour améliorer la clarté

2. Qualité du code

-  *Merge Requests* obligatoires, validées par relecture
-  *Commits normalisés* (ex: **feat:**, **fix:**) pour la traçabilité
-  *Guide de style commun* :
 - Angular/NestJS : Prettier
 - Python/ROS2 : Black, pylint
 - Conventions de nommage et structure modulaire
-  *Bonnes pratiques* : modularité, documentation, gestion des erreurs

3. Validation technique

-  Tests unitaires et d'intégration réguliers
-  Démonstrations vidéo des fonctionnalités clés
-  Suivi continu de la qualité tout au long du projet





Technical Compliance Checklist

3. Conception technique : Requis Atteints

Fonctionnalités principales	R.F.1, R.F.2, R.F.3, R.F.5 à R.F.10
Données et contraintes	R.F.17, R.F.18, R.F.19
Contraintes techniques	R.C.1 à R.C.5
Qualité	R.Q.1, R.Q.2
Partiellement instables	R.F.4, R.F.15



3. Conception technique : Résultats Concrets

<0,3m

Précision

Retour à la base avec
exactitude

100%

Compatibilité

UI testée sur Windows, Android,
iOS

Exploration

Déplacement

Les robots explorent l'environnement de façon
autonome et génèrent une carte en temps réel.

4. Défis techniques et solutions apportées

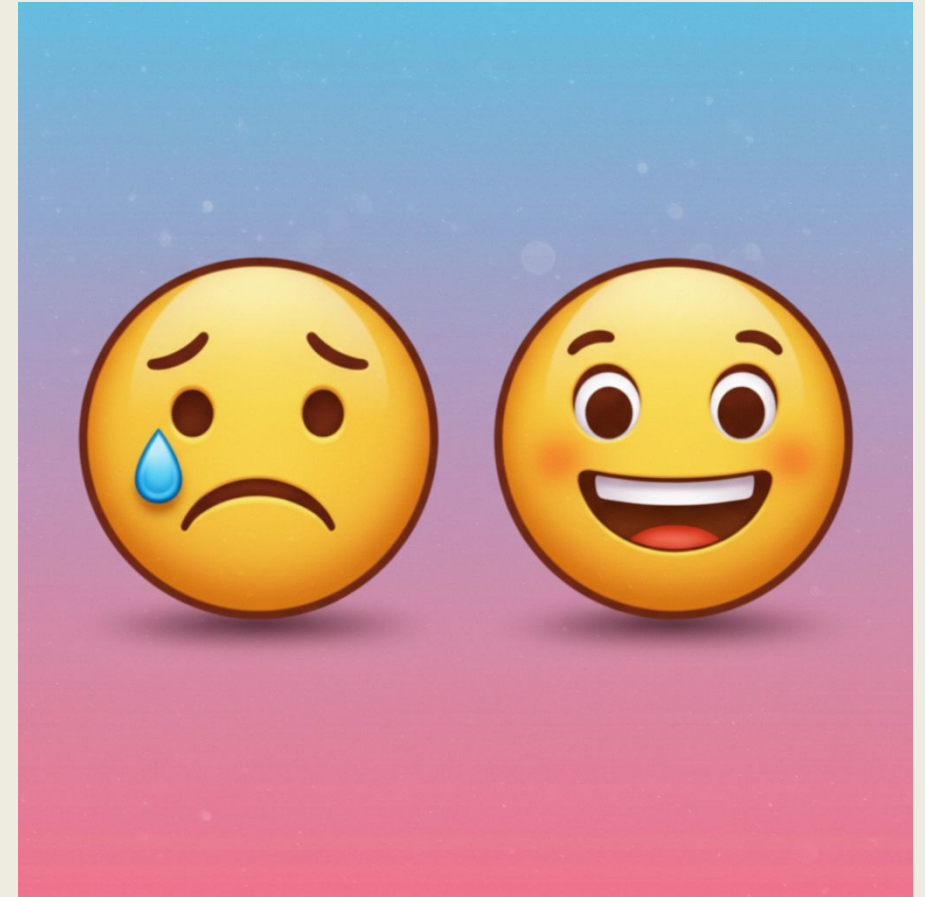
4. Défis techniques et solutions apportées

Défis rencontrés

- Accès restreint aux robots physiques
- Réseau WiFi instable et congestionné
- Complexité de l'intégration ROS2 avec le backend Web
- Algorithme d'exploration difficile à stabiliser

Solutions apportées

- Simulation Gazebo intensive pour valider sans robot
- Accès anticipé aux laboratoires pour profiter des créneaux calmes
- Utilisation de hotspots personnels pour stabiliser le WiFi
- Séparation claire des nœuds ROS pour simplifier le débogage



Source : Image générée par une IA

5. Conclusion et perspectives d'amélioration

5. Conclusion et perspectives d'amélioration : Travaux futurs et recommandations

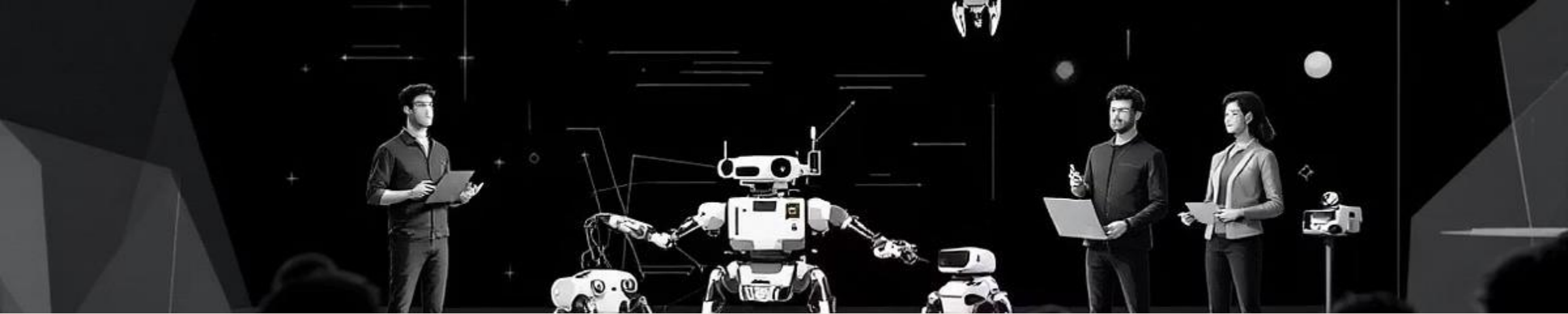
✓ **Toutes les fonctionnalités obligatoires ont été livrées**

🔧 Des améliorations sont proposées pour renforcer la robustesse, l'UX et la maintenabilité :

1. 🌀 **Stabilité de l'exploration (R.F.4)**
→ Remplacer le *random walk* par une couverture planifiée ou un système de priorités.
2. 🖥️ **Expérience utilisateur**
→ Améliorer l'interface : zoom, filtres, meilleure lisibilité des cartes et journaux.
3. 🛑 **Support en environnement non contrôlé**
→ Ajouter arrêt d'urgence, zones interdites, reconnection automatique après défaillance.
4. 📖 **Documentation technique continue**
→ Rédiger un guide clair d'installation, configuration et extension pour les futurs utilisateurs.
5. 🤖 **Extension multi-robot avancée**
→ Planification centralisée, répartition des rôles, missions complexes à grande échelle.

Source : Image générée par une IA





Source : Image générée par une IA

5. Conclusion et perspectives d'amélioration : Récapitulatif

Système robuste

Architecture modulaire,
conteneurisée et éprouvée dans
des conditions réelles.

Vision respectée

Projet ancré dans la réalité
industrielle, répondant aux
objectifs initiaux.

Ressources disponibles

Tous les livrables sur GitLab.
Démonstration possible sur
demande.

Démonstrations techniques

The screenshot shows a web application titled "ClientApp" running on "localhost:4200". The interface is divided into several sections:

- Top Bar:** Shows "1 appareil(s) connecté(s)" and a green button labeled "Contrôle actif".
- SLAM Map:** A large central area displaying a 2D occupancy grid map. A legend in the bottom-left corner identifies "limo1" with a green dot and "limo2" with a red dot. Navigation controls (SLAM, zoom in/out, reset) are on the right.
- Mission en cours:** A log of mission events.

Timestamp	Robot	Status
18:19:26	limo2	en arrêt
18:19:27	limo1	en arrêt
18:19:27	limo2	en arrêt
18:19:28	limo1	en arrêt
18:19:28	limo2	en arrêt
- Contrôle Du Limo2:** A control panel for robot limo2.
 - Status: ☐ En attente
 - Battery: Niveau de batterie Non disponible
 - Buttons: Démarrer Mission, Arrêter Mission, Identifier, Activer P2P
- Contrôle Du Limo1:** A control panel for robot limo1, identical in layout to the limo2 panel.

The bottom of the image shows a Windows taskbar with various icons and a system clock indicating "Apr 15 18:19".

Sources

- [1] https://www.researchgate.net/figure/Two-kinds-of-steering-methods_fig1_261192572
- [2] <https://www.linkedin.com/pulse/why-use-mongodb-when-muhammad-rizo-abdunazarov/>
- [3] <https://www.techtarget.com/searchnetworking/definition/peer-to-peer>
- [4] <https://indrorobotics.ca/new-limo-pro-ros2-models-bring-advanced-abilities-to-rd/>
- [5] <https://roboticsbackend.com/ros2-nav2-tutorial/>
- [6] <https://www.bbc.co.uk/bitesize/articles/z9q3f82>