



**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. H2023-INF3995 du département GIGL.

***Conception d'un système aérien d'exploration***

Équipe No <**101**>

*De Blas David  
Deloumeau Nicolas  
Desgroseilliers Philippe  
Louesdon Théo  
Ngay Munga Exaucé-Constantin*

24 mars 2023

## Table des Matières

### Table des Matières

1. Vue d'ensemble du projet
  - 1.1 But du projet, porté et objectif (Q4.1)
  - 1.2 Hypothèse et contraintes (Q3.1)
    - 1.2.1. Hypothèses
    - 1.2.2 Contraintes à prendre en compte
      - 1.2.2.1 *Contraintes matérielles et logicielles*
      - 1.2.2.2 *Contraintes techniques*
      - 1.2.2.3 *Contrainte concernant l'équipe*
  - 1.3 Biens livrables du projet (Q4.1)
2. Organisation du projet
  - 2.1 Structure d'organisation (Q6.1)
  - 2.2 Entente contractuelle (Q11.1)
3. Description de la solution
  - 3.1 Architecture logicielle générale (Q4.5)
  - 3.2 Station au sol (Q4.5)
  - 3.3 Logiciel embarqué (Q4.5)
  - 3.4 Simulation (Q4.5)
  - 3.5 Interface utilisateur (Q4.6)
  - 3.6 Fonctionnement général (Q5.4)
4. Processus de gestion
  - 4.1 Estimations des coûts du projet (Q11.1)
  - 4.2 Planification des tâches (Q11.2)
  - 4.3 Calendrier de projet (Q11.2)
  - 4.4 Ressources humaines du projet (Q11.2)
5. Suivi de projet et contrôle
  - 5.1 Contrôle de la qualité (Q4)
  - 5.2 Gestion de risque (Q11.3)
  - 5.3 Tests (Q4.4)
  - 5.4 Gestion de configuration (Q4)
  - [5.5 Déroulement du projet \(Q2.5\)](#)
6. Résultats des tests de fonctionnement du système complet (Q2.4)
- [7. Travaux futurs et recommandations \(Q3.5\)](#)
- [8. Apprentissage continu \(Q12\)](#)
  - [8.1 De Blas David](#)
  - [8.2 Deloumeau Nicolas](#)

[8.3 Desgroseilliers Philippe](#)

[8.4 Louesdon Théo](#)

[8.5 Ngay Munga Exaucé-Constantin](#)

[9. Conclusion \(Q3.6\)](#)

10. Références (Q3.2)

## 1. Vue d'ensemble du projet

### ***1.1 But du projet, porté et objectif (Q4.1)***

Le but du projet consiste à concevoir un système composé d'un ensemble de logiciels permettant à 2 robots de cartographier une pièce d'un bâtiment. Les données récoltées par les robots doivent être envoyées à l'opérateur via une interface web où il sera possible de visualiser la représentation de la pièce. L'opérateur pourra envoyer des instructions aux robots et surveiller leur état grâce à l'interface web, lui permettant ainsi d'interagir avec eux. Concernant le choix des deux robots, nous pouvons soit opter pour un drone CogniFly (The CogniFly Project, 2023) et un robot mobile AgileX Limo (AgileX Robotics, 2022) soit pour deux robots du même type. Tout au long du projet, l'équipe s'assurera que les robots respectent un certain nombre de critères assurant la qualité au niveau du matériel et du logiciel.

Concrètement, le système est composé de trois grandes sections : une station au sol, une partie embarquée pour les deux robots et une simulation logicielle afin de planifier le comportement des robots. On doit identifier des technologies adéquates pour chaque section du système et trouver un moyen de les relier afin de garantir son bon fonctionnement.

Le travail se divisera tout au long du trimestre en 3 parties :

- Le Preliminary Design Review (PDR)
- Le Critical Design Review (CDR)
- Le Readiness Review (RR)

Pour la première étape, il s'agit de construire un prototype minimal du projet, pour la seconde, il est nécessaire de développer une structure complète du système et pour la dernière, il faudra créer un système complet et opérationnel.

## **1.2 Hypothèse et contraintes (Q3.1)**

### **1.2.1. Hypothèses**

- Nous supposons que l'environnement Gazebo (Hedges, et al., s.d.) possède une physique et des composants des robots assez proches de la réalité. Ainsi, si nous arrivons à implémenter le comportement souhaité des robots dans gazebo (Open Source Robotics Foundation, 2023), on ne devrait pas trop avoir de problème à reproduire ce comportement dans la réalité.
- Nous présumons que l'ensemble du matériel fourni ne présente pas, ou peu, de problèmes qui pourraient influencer sur le comportement des robots et sur le fonctionnement des différents capteurs.
- Nous faisons l'hypothèse que le temps de transmission des commandes entre la station au sol et les robots sera très rapide, car tous les tests se feront dans une pièce fermée et utilisée à cet effet, ce qui réduit fortement les interférences.

### **1.2.2 Contraintes à prendre en compte**

#### **1.2.2.1 Contraintes matérielles et logicielles**

- Tous problèmes matériels et logiciels (endommagement du matériel et gestion de bugs)
- Tous les composants logiciels doivent être conteneurisés avec Docker (Docker Inc., 2023)
- Le logiciel complet de la station doit être lancé avec une seule commande impliquant Docker (Docker Inc., 2023) dans un terminal Linux roulant Ubuntu 20.04 (Canonical Ltd., 2018).
- L'interface utilisateur devrait rester identique lorsque la station au sol est reliée à la simulation ou aux robots réels.
- Le mouvement des robots doit se faire au niveau du code embarqué, la station au sol ne doit pas dicter ses mouvements

### 1.2.2.2 Contraintes techniques

Identifiant	Description
R.F.1	Chaque robot doit répondre individuellement à une commande “Identifier” qui clignote une DEL ou émet un son.
R.F.2	Les robots doivent répondre aux commandes “Lancer la Mission” et “Terminer la Mission”.
R.F.3	L’interface utilisateur doit montrer le type et l’état de chaque robot.
R.F.4	Les robots doivent être en mesure de se déplacer de façon autonome.
R.F.5	Les robots doivent être en mesure d’éviter des obstacles détectés par leurs capteurs
R.F.6	Un bouton “Retour à la base” doit être disponible pour faire retourner les robots à moins de 0.5m de leur position initiale.
R.F.7	La fonction “Retour à la base” doit être automatiquement déclenchée si la batterie atteint un pourcentage de moins de 30%. Le pourcentage de la batterie doit être affiché sur l’interface et les robots ne devraient pas décoller si leur batterie est à moins de 30%.
R.F.8	La station au sol doit recueillir les données captées par les robots et produire une carte de l’environnement qui est visible sur l’interface.
R.F.9	Lors d’une mission, la position du robot doit être visible sur la carte.
R.F.10	L’interface doit être disponible comme service web et visible sur plusieurs appareils (mobile, PC, tablettes) via réseau. Au moins deux appareils doivent être en mesure de se connecter en même temps.
R.F.11	Les deux robots utilisés sont de types différents et explorent en même temps.
R.F.13	Le système doit pouvoir détecter un “crash” de drone et afficher l’état “crashed” sur l’interface utilisateur.
R.F.17	Une base de données présente sur la station au sol doit enregistrer des attributs comme la date et l’heure de la mission, le temps de vol, la distance totale parcourue, etc. L’utilisateur doit aussi être en mesure de fouiller les anciens logs des missions précédentes à partir de l’interface.

R.F.18	La carte générée durant une mission doit être sauvegardée sur la station au sol. L'interface doit aussi permettre d'afficher les cartes des missions précédentes sauvegardées dans la base de données.
R.F.19	Les robots doivent pouvoir communiquer entre eux leur distance par rapport aux points de départ, si le rover est plus loin, il doit montrer une icône sur son écran.
R.C.1	Des logs de débogage doivent être disponibles en continu pour s'assurer que le système fonctionne correctement. Ces logs doivent être sauvegardés sur la station et accessibles en tout temps.
R.C.2	Le logiciel au complet doit pouvoir être lancé à partir d'une commande Linux (docker-compose). La procédure doit être détaillée dans un README pour la simulation et le physique.
R.C.3	L'environnement virtuel de simulation doit être généré aléatoirement avec minimum 3 murs en plus des 4 murs externes.
R.C.4	L'interface utilisateur doit être facile d'utilisation, lisible, et suivre les 10 heuristiques d'interface usager (Nielsen Norman Group, 2020).
R.C.5	Le système doit s'adapter automatiquement pour marcher avec 1 ou 2 robots.
R.Q.1	Le format du code doit être standardisé à travers le projet.
R.Q.2	Des tests unitaires doivent être écrits pour chaque composante logicielle. Si aucun test ne peut être écrit, une procédure de tests doit être détaillée.

*Tableau 1 : Requis obligatoires (beige) et optionnels (mauve) qui seront implémentés au courant de la durée du projet*

### **1.2.2.3 Contrainte concernant l'équipe**

- Toutes activités reliées aux autres cours (intra, travaux pratiques, etc.) risquent de retarder certaines tâches initialement prévues.
- Retard accumulé à la suite d'incompréhensions vis-à-vis d'une tâche.
- Fatigue accumulée selon les différentes périodes de la session.
- Un membre de l'équipe a quitté le projet, nous ne sommes plus que 5, cela risque d'avoir un impact sur le nombre de requis que nous serons capables de compléter et/ou sur la qualité des requis.

### 1.3 Biens livrables du projet (Q4.1)

Au fur et à mesure du projet, nous remettrons 3 livrables

Livable	Date	Description
Preliminary Design Review (PDR)	10 février 2023 à 23H55	- Prototype minimal du projet
Critical Design Review (CDR)	24 mars 2023 à 23H55	- Prototype avancé
Readiness Review (RR)	21 avril 2023 à 23H55	- Système complet et opérationnel

*Tableau 2 : Livrables attendus au long de la session*

Pour chacune des remises, des artefacts relatifs à chaque livrable seront remis aux dates issues du tableau ci-dessus. Ces artefacts seront organisés sous forme de dossier dans GitLab (GitLab B.V., 2023). Un rapport technique du projet, qui sera de plus en plus détaillé au fil des livrables, sera présent dans le répertoire principal et le code source sera divisé en sous-répertoires contenant les fichiers de conteneurisation Docker (Docker Inc., 2023) avec leurs fichiers de configuration pour lancer chaque module du système séparément. Enfin, des README fourniront les instructions pour exécuter les commandes dans l'invité de commande.

## 2. Organisation du projet

### 2.1 Structure d'organisation (Q6.1)

Nous nous sommes entendus sur une méthode de travail décentralisée pour le développement du projet (Collin, Desmarais, & Gendreau, 2023). Ce type de structure nous permet d'identifier nos forces et nos faiblesses individuelles pour assigner les tâches aux membres les plus spécialisés ou à ceux qui y sont le plus intéressés. En travaillant individuellement ou au plus avec un coéquipier, nous pouvons nous assurer que le projet se déroule sans trop de problèmes, si nous respectons bien les échéanciers prévus au calendrier.



Pour soutenir cette structure, nous avons décidé en équipe de déléguer le rôle de coordinateur de projet à Nicolas avec les cinq autres membres agissant comme analyste développeur. Le rôle de coordinateur est là pour s'assurer que la charge de travail sera distribuée équitablement entre les 6 membres de l'équipe en fonction de chacune de nos forces et faiblesses et les développeurs analystes, dont Nicolas fait aussi partie, sont là pour construire le projet. Nicolas sera au centre du réseau décentralisé, mais nous sommes aussi en mesure de communiquer entre nous pour résoudre toute défaillance qui pourrait survenir.

L'utilisation de rencontres hebdomadaires est un autre aspect que nous avons mis en place pour assurer le développement continu du projet et Philippe est celui qui s'occupera de les planifier et de dresser les ordres du jour. Ces réunions seront non seulement pour faire un bilan de la semaine précédente, mais aussi pour dresser une liste des tâches que nous planifions accomplir dans la semaine qui suit. Nous allons aussi utiliser ce temps pour soulever toutes inquiétudes ou problématiques que nous avons pour voir si nous pouvons en venir à une solution pour le long terme.

Pour ce qui est des choix à faire quant aux technologies utilisées, nous allons procéder d'une manière à chaque membre l'équipe ait une voix. Nous ne voulons pas mettre un coéquipier en situation d'apprentissage aigu juste parce qu'une ou deux personnes sont familières avec une technologie spécifique. Cependant, quand vient l'implémentation d'une fonctionnalité ou d'un requis, nous allons la laisser à la discrétion du membre qui l'effectue, car nous nous faisons assez confiance pour nous laisser choisir parmi les options possibles et d'implémenter la meilleure.

## **2.2 Entente contractuelle (Q11.1)**

En raison de la nature du contexte de ce projet, nous optons ici pour un contrat de type prix ferme (Collin, Desmarais, & Gendreau, 2023). En effet, nous avons une liste détaillée des requis techniques et physiques pour le projet. Ce qui nous donne une assez bonne certitude quant à l'estimation du temps et des

coûts nécessaires pour la réalisation du projet. De plus, le temps est limité à 4 mois et il est nécessaire d'avoir une remise complète.

Les avantages de ce type de contrat sont que le promoteur n'a pas nécessairement besoin de suivre le développement du projet avec un œil attentif et qu'il sait à quoi s'attendre quant au coût avant même de commencer sa conception. Le tout est défini en avance avec le contractant. En revanche, ce type de contrat nécessite une expertise prédéterminée de la part des contractants, car le cahier des charges doit être mis au clair dès le début, ce qui est le cas dans notre contexte.

### 3. Description de la solution

#### 3.1 Architecture logicielle générale (Q4.5)

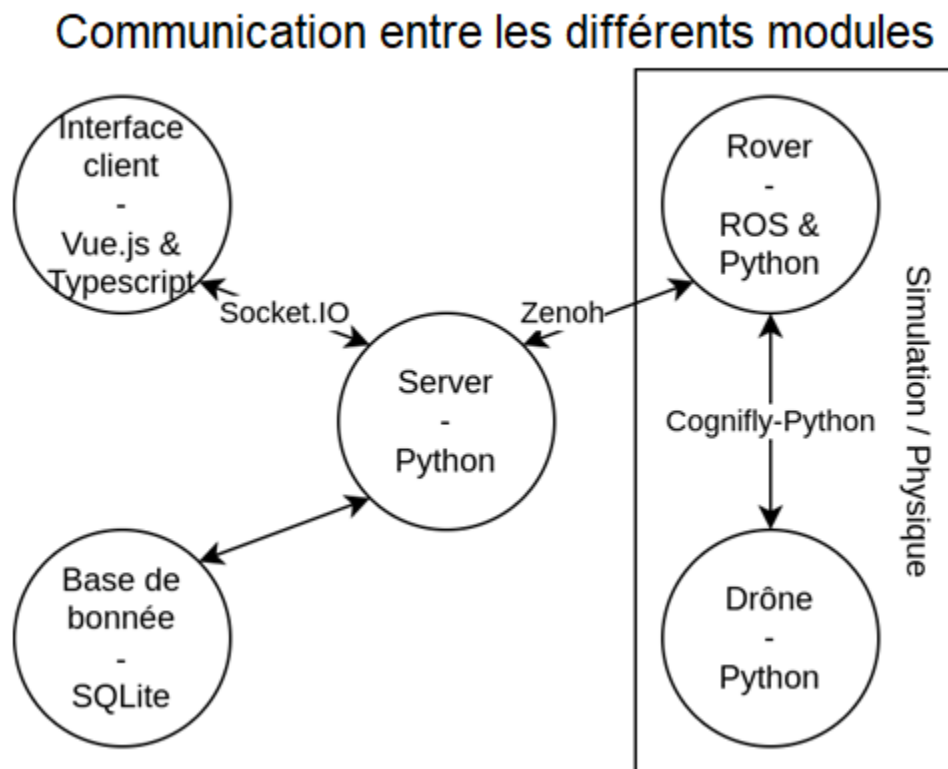


Figure 1 : Diagramme de l'architecture logicielle générale

L'utilisation de Zenoh (Eclipse Zenoh, 2023) pour la communication entre la station au sol et le(s) robot(s) permet aux utilisateurs de se connecter facilement aux robots et de les commander en utilisant des commandes simples.

Les robots peuvent également publier des données de capteurs et d'autres informations, qui peuvent être reçues par la station en s'abonnant à ceux-ci. De plus, l'utilisation de Zenoh (Eclipse Zenoh, 2023) permet une communication directe et efficace entre les robots sans avoir besoin de la station au sol (en P2P). Cela permet une latence faible et une communication efficace entre les robots. Zenoh (Eclipse Zenoh, 2023) offre donc une solution avantageuse au niveau de la facilité et de l'efficacité de la communication. Ce système de communication nous permettra de faire les requis R.F.1, R.F.2, R.F.3, R.F.6, R.F.8, R.F.9, R.F.10 et R.C.1. Tous ces requis utilisent principalement la communication entre la station au sol et les robots, qui sera faite avec Zenoh (Eclipse Zenoh, 2023).

La décision de prendre Python (Python Software Foundation, 2023) et Vue.js (VueJs, 2023) comme langages de programmation est réellement basée sur la préférence des membres de l'équipe. De plus, Vue.js (VueJs, 2023) ayant un cadre léger et Python (Python Software Foundation, 2023) étant un langage interprété, facile à utiliser permet d'uniformiser la compréhension de l'équipe sur ce qui a été fait par d'autres membres.

L'utilisation de Typescript (Microsoft, 2023) plutôt que JavaScript (ECMA International, 2022) est encore une fois, pour faciliter la compréhension du code pour les autres membres de l'équipe. Cela obligera également les membres de l'équipe à écrire explicitement les types de variables et les types de retours de fonction.

SQLite (SQLite Consortium, 2023) est encore une fois une préférence de l'équipe, car nous avons appris le langage SQL (W3Schools, 2023) et il sera assez performant et léger pour ce qui est demandé dans le cadre de ce projet. Cette base de données nous permettra de compléter les requis R.F.17, R.F.18 et R.C.1.

Quant à ROS (ROS, 2023), nous avons pris la décision de le favoriser à ROS2 (Open Source Robotics Foundation, 2023) pour sa documentation très précise et abondante. De plus, l'environnement de simulation et les robots physiques sont déjà installés de façon à héberger ROS (ROS, 2023).

### 3.2 Station au sol (Q4.5)

La station au sol sera composée d'un contrôleur accessible sur le web, de services, de robots et d'une base de données. Ces modules communiquent ensemble par l'entremise de Zenoh (Eclipse Zenoh, 2023).

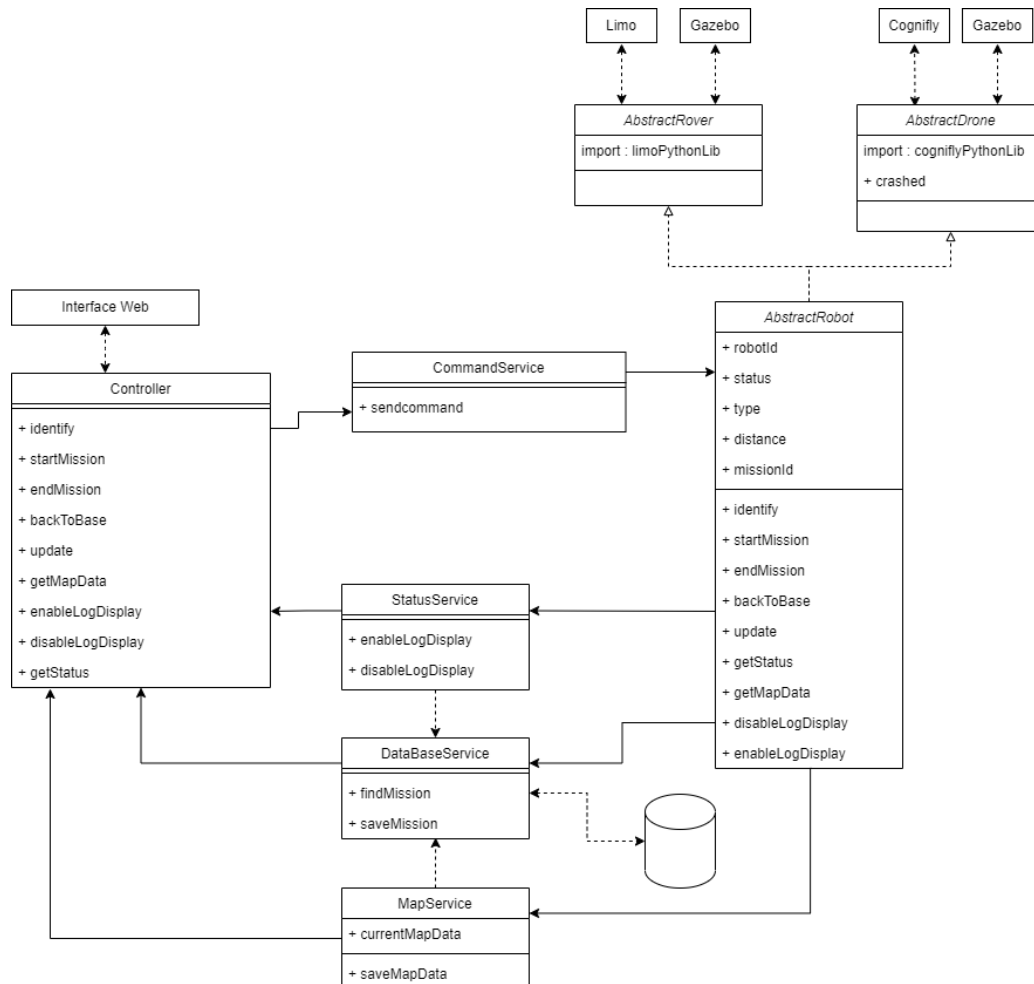


Figure 2 : Diagramme de classe de la station au sol

Le contrôleur accessible sur une interface web permet à l'utilisateur d'envoyer des commandes de base aux robots. La première commande "Identify" sera nécessaire afin d'identifier les robots, tel qu'il est demandé dans le requis R.F.1. Il y a aussi les commandes `startMission` et `endMission` qui seront utilisées pour lancer et mettre fin à une mission, tel qu'il a été énoncé dans le

requis R.F.2. Il y aura une commande backToBase pour ramener le robot à son point de départ, tel qu'il a été demandé par le requis R.F.6. La commande testSystem sera nécessaire afin de tester le système tel qu'il est indiqué par le requis R.C.1. Puis, la fonction getMapData sera utilisée afin de récupérer les informations de la carte graphique qui doit être générée.

Les services doivent s'occuper de toute la partie logique du système. Il y aura donc un mapService pour la gestion des cartes géographiques qui sont générées par les robots, un DataBaseService pour la gestion de la base de données et un CommandService pour gérer les commandes envoyées par le contrôleur.

Finalement, les robots auront beaucoup de méthodes identiques indépendamment de leur type et quelques méthodes qui varient selon leur type. Par exemple, l'état "crashed" ne doit être détecté que sur le CogniFly (The CogniFly Project, 2023).

### 3.3 Logiciel embarqué (Q4.5)

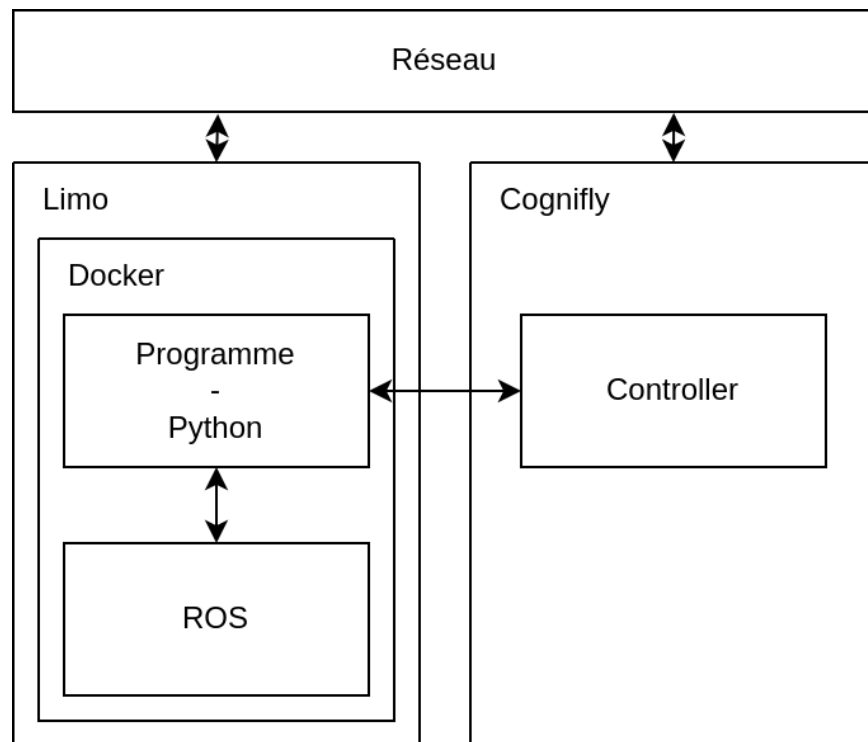


Figure 3 : Diagramme du logiciel embarqué

Le logiciel embarqué de notre système sera basé sur ROS (Open Source Robotics Foundation, 2014) pour faciliter l'interaction avec le robot et simplifier les processus de développement. Nous avons choisi d'utiliser Python (Python Software Foundation, 2023) comme langage de programmation pour le logiciel embarqué, car il dispose de nombreuses implémentations et bibliothèques disponibles pour nous aider dans la réalisation de notre projet. La simplicité de Python (Python Software Foundation, 2023) permet également un développement plus rapide et facilite la maintenance et l'évolution du système. L'utilisation de ROS (ROS, 2023) nous permettra de compléter les requis la plupart des requis en relation avec les robots comme les requis R.F.4, R.F.5, R.F.6, R.F.7, R.F.8 et R.F.9.

En ce qui concerne la communication entre le rover, le drone et le serveur, nous utiliserons le protocole Zenoh (Eclipse Zenoh, 2023) pour simplifier les échanges de données. Cela permettra d'assurer une communication efficace et fiable entre les différents éléments du système, tout en simplifiant les processus de développement et de maintenance.

### 3.4 Simulation (Q4.5)

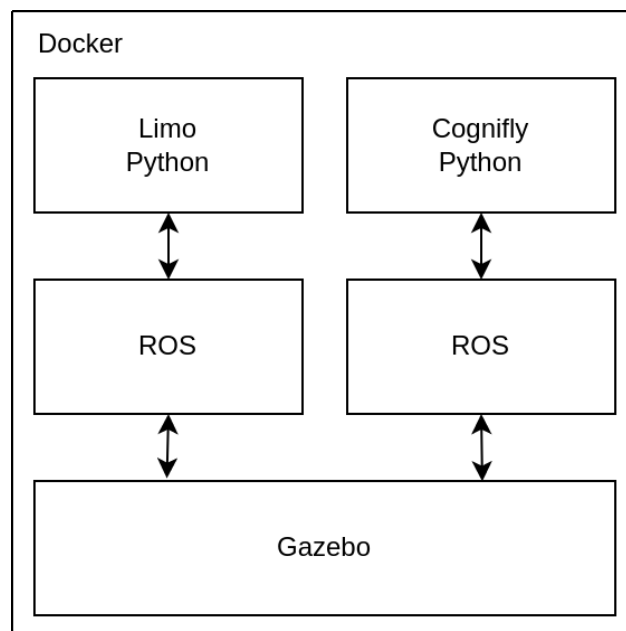


Figure 4 : Diagramme de la simulation

Pour améliorer la portabilité de notre programme, nous utiliserons un environnement Docker (Docker Inc., 2023). Cela nous permettra de garantir la compatibilité de notre programme sur différents ordinateurs de l'équipe, en évitant les problèmes de dépendances matérielles et logicielles. En utilisant un simulateur Gazebo (Open Source Robotics Foundation, 2023) pour les simulations, nous pourrions également garantir un environnement physique réaliste, permettant ainsi au programme Python (Python Software Foundation, 2023) de ne pas faire la différence entre le robot physique et sa simulation. Cela permettra de faciliter les tests et les déploiements sur le robot réel.

### 3.5 Interface utilisateur (Q4.6)

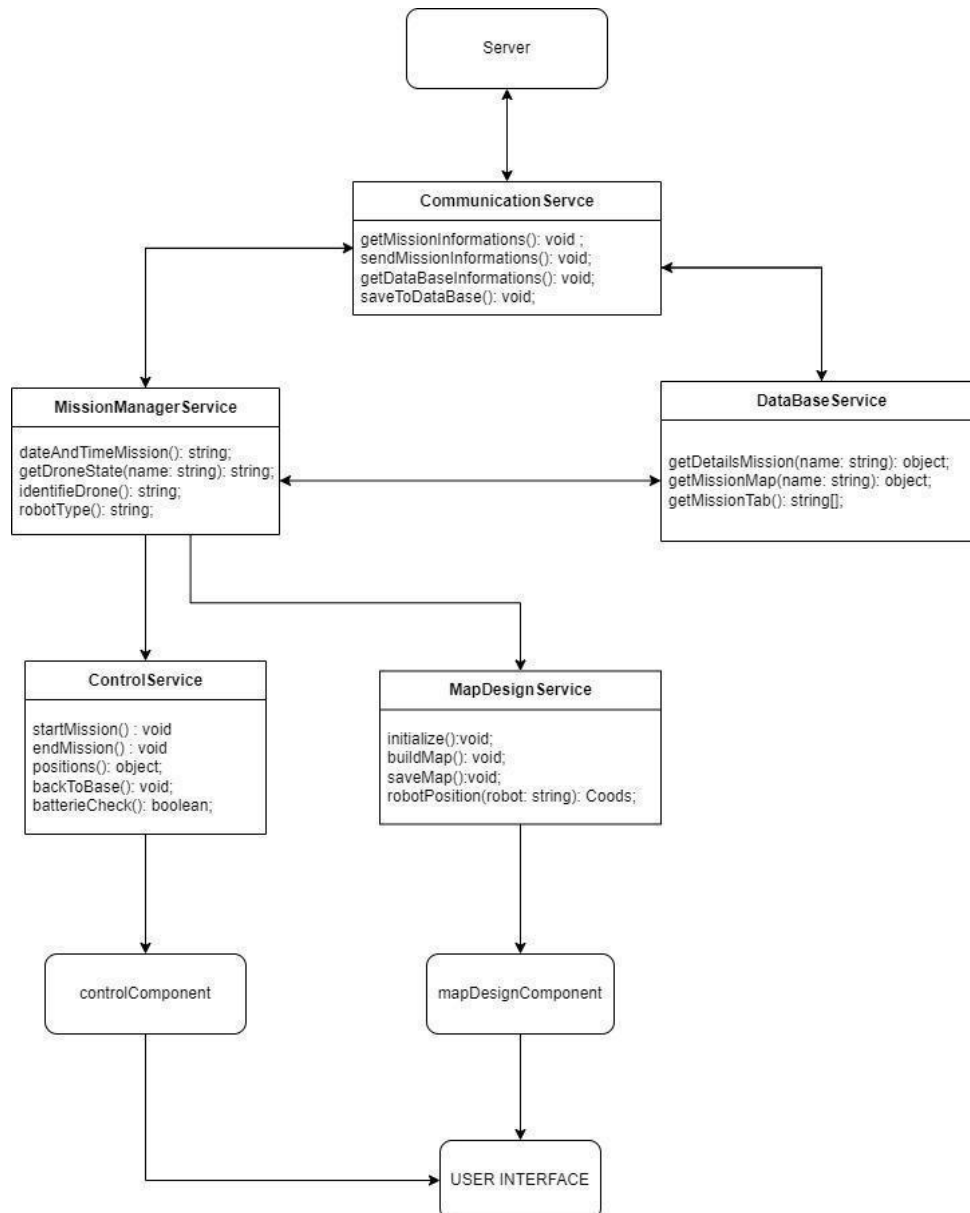


Figure 5 : Diagramme de l'interface utilisateur

L'interface utilisateur est un environnement web développé à l'aide du framework Vue.js (VueJs, 2023). La construction de cette structure est inspirée de celle que nous avons pour utiliser le framework Angular (Google, 2023) lors du deuxième projet intégrateur en sigle LOG2990. Nous utiliserons des services qui exécutent une logique, et des composants pour la gestion de l'affichage qui permettent l'interaction avec l'utilisateur. HTML (WHATWG, 2023), CSS (CSS Snapshot 2023, 2023), fichiers de test et fichiers TypeScript (Microsoft, 2023),



tels sont les fichiers de code qui peuvent être trouvés dans notre interface utilisateur.

CommunicationService permet la communication de l'interface client avec notre serveur. La communication se fera via Socket.io (Socket.IO, 2023), ce qui permettra le transfert des informations nécessaires aux services MissionManagerService et DataBaseService pour le bon fonctionnement de l'interface.

Le service MissionManagerService est utilisé pour gérer les opérations de la mission. Celui-ci contient des sous-entités telles que MapDesign pour la carte et Control pour contrôler les robots effectuant les tâches. Ensemble, ils répondent aux exigences suivantes : R.L.3, R.F.1, R.F.2, R.F.3, R.F.6, R.F.7, R.F.8, R.F.9, R.F.11, R.F.13.

Le service DataBaseService fournit des informations provenant de la base de données, importantes pour les opérations de l'interface utilisateur. Le service MissionManager y récupère les données nécessaires à son fonctionnement et à celui de ses sous-systèmes. Cela répond aux exigences de R.F.17 et R.F.18.

### **3.6      *Fonctionnement général (Q5.4)***

Pour lancer notre système, il faut lancer la partie liée à la station de contrôle (client et serveur) en se plaçant dans le root du repository Gitlab (GitLab B.V., 2023) et en exécutant la commande ***docker compose up --build groundstation***. Le serveur et le client communiquent sur le port 8000, et le tout est lancé en mode production, ce qui signifie que le serveur sert les fichiers nécessaires au client. La station au sol devra être équipée de Docker (Docker Inc., 2023).

Pour faire fonctionner la simulation des robots, il faut suivre les étapes précisées dans le README du dossier sim. Ces étapes permettent de lancer une simulation avec deux robots et d'assigner un script d'exécution différent à chacun.

## 4. Processus de gestion

### 4.1 Estimations des coûts du projet (Q11.1)

En considérant que chaque membre du groupe consacre en moyenne six heures lors des séances de laboratoire et trois heures et demie en dehors de ces séances pour travailler sur le projet, nous pourrions arriver à une estimation des coûts pour une période de onze semaines de travail en utilisant les taux horaires suivant:

- Développeur-analyste : 130\$/h
- Coordonnateur de projet : 145\$/h

Nous obtenons:

Membre	Salaire hebdomadaire	Total
Coordonnateur de projet	145\$/h * 9.5h = 1377.5\$	15 152.5\$
Développeur-analyste	130\$/h * 9.5h = 1235\$	13 585\$

*Tableau 3 : Prix total du coût de la main-d'oeuvre*

Pour une équipe composée d'un coordinateur de projet et de cinq développeurs analyste sur une période de onze semaines soit un total de 627 heures de travail, nous obtenons un prix total de l'équipe de 15 152.5\$ + 13 585\$ \* 5 = 83 077.5\$

On ajoute 4300 dollars de coût de matériel, dont 1000 dollars pour la tour de contrôle, 400 dollars pour le CogniFly (The CogniFly Project, 2023) et 2900 dollars pour le LIMO (Agilex Robotics, 2022) (en sachant que la société nous fournit les 6 Laptops et le local), et nous obtenons un total de 87 377.5\$.

## 4.2 Planification des tâches (Q11.2)

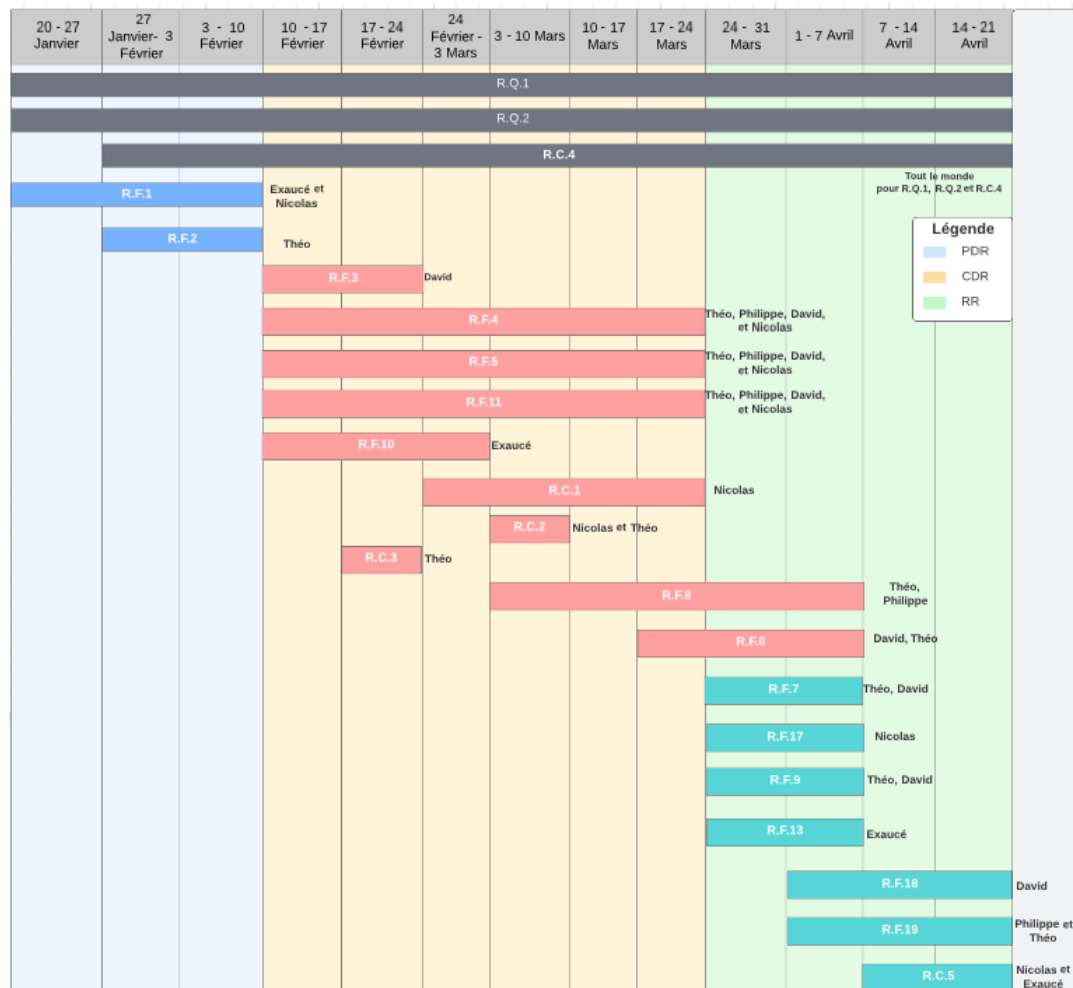


Figure 6 : Diagramme de Gantt de la planification des tâches pour le projet

## 4.3 Calendrier de projet (Q11.2)

Trois principaux livrables seront à remettre au cours de la session, correspondant aux trois principales versions de notre projet. Le calendrier ci-dessous donne les dates de remises et un aperçu de ce qui doit être complété pour chaque livrable.

<i>Calendrier du projet</i>		
Livrable	Date	Aperçu
Preliminary Design Review (PDR)	Vendredi 10 février 2023	- Plan de projet complet - Démo vidéo des 2 premiers requis
Critical Design Review (CDR)	Vendredi 17 mars 2023	- Plan de projet plus détaillé et révisé - Démo vidéo de 7 requis - Présentation technique des concepts du produit
Readiness Review (RR)	Vendredi 21 avril 2023	- Plan de projet final - Démo vidéo de tous les requis - Écriture des instructions liées à la compilation

*Tableau 4 : Description des livrables à remettre tout le long de la session*

Le premier livrable décrit la documentation initiale du projet. Il montre également le fonctionnement des 2 premiers requis par l'intermédiaire de démonstration vidéo.

Le deuxième livrable fournit une documentation plus détaillée et révisée du premier livrable. Il montre aussi les changements apportés depuis le premier livrable par l'intermédiaire d'une présentation technique. Aussi cette fois-ci 7 démonstrations vidéo sont requises pour illustrer la bonne fonctionnalité des requis suivants : R.F.1, R.F.2, R.F.3, R.F.4, R.F.5, R.F.10 et R.C.1.

Enfin le dernier livrable présente la documentation finale du produit. Tous les requis doivent avoir une démonstration vidéo représentant leur bon fonctionnement. Tous les tests doivent être finalisés et des instructions nécessaires à la compilation et au lancement du produit doivent être fournies.

## **4.4 Ressources humaines du projet (Q11.2)**

Les ressources humaines du projet sont constituées de 1 coordinateur de projet et 5 développeurs-analystes. Nous avons aussi accès à l'équipe académique du cours INF3995 qui est constitué du Chargé de Cours Giovanni Beltrame et des deux chargés de laboratoires Guillaume Ricard et Antoine Robillard.

Les qualifications techniques et habiletés personnelles attendues du coordonnateur de projet sont de:

- Maîtriser des outils de gestion de projet comme GitLab (GitLab B.V., 2023)
- Pouvoir prendre des décisions rapidement, en ayant consulté les membres de l'équipe
- Déléguer les tâches de manière stratégique en fonction des compétences et des expériences de chacun des membres.
- Avoir une bonne capacité d'organisation, de planification et de leadership.

Du côté des développeurs-analystes, les qualifications techniques et habiletés personnelles attendues sont de:

- Maîtriser les technologies nécessaires pour effectuer le projet (Langages de programmation, écriture de tests, base de données, Docker (Docker Inc., 2023), Linux (Canonical Ltd., 2018))
- Maîtriser la programmation de systèmes embarqués
- Être ponctuel aux réunions comme aux remises
- Savoir travailler en méthode agile

Afin d'optimiser l'avancée de notre projet, le coordonnateur de projet devra probablement exécuter des tâches similaires à celles d'un développeur-analyste pour assurer un bon déroulement de celui-ci.

Le tableau ci-dessous répertorie les compétences et les expériences de chaque membre de l'équipe, qu'elles aient été acquises au cours de leur parcours professionnel ou personnel:

Qualifications	David	Philippe	Exaucé	Nicolas	Théo
Maîtrise du langage Python	X		X	X	X
Maîtrise de Vue.js et Typescript		X	X	X	
Maîtrise de programmation de système embarqué		X	X		
Écriture de bons tests	X	X		X	X
Maîtrise de Docker				X	
Utilisation d'une base de données SQLite	X		X	X	X
Connaissance du système d'exploitation Linux	X		X	X	X
Maîtrise des fonctionnalités de Gitlab	X	X	X	X	X
Développement d'une application Web	X	X	X	X	X
Développement d'un serveur Web en Python				X	X
Connaissance du protocole de communication Zenoh					
Connaissance des méthodes agiles	X	X		X	X

Tableau 5 : Expériences des membres de l'équipe

## 5. Suivi de projet et contrôle

### 5.1 Contrôle de la qualité (Q4)

Au cours de notre projet, nous veillerons à soumettre à tous nos livrables un processus de révision régulier, que ce soit pour les aspects techniques ou pour les produits finaux. Pour assurer la qualité du code, nous adopterons les

normes de style les plus couramment utilisées pour garantir la qualité de notre code. Ainsi, nous utiliserons PEP 8 pour le code Python (Python Software Foundation, 2023) et Prettier (Prettier, 2023) ainsi que ESLint (OpenJS Foundation, 2023) pour le code TypeScript (Microsoft, 2023) et Vue.js (You, Get Started, 2023). Cela nous permet de maintenir des standards élevés de programmation pour les noms de classes, de fonctions et de variables. De plus, tout ajout au code principal (merge) devra être revu par au minimum deux personnes de manière exhaustive pour s'assurer de la bonne qualité du code et du respect des conventions. Ceci répondra au requis R.Q.1.

### **5.2 Gestion de risque (Q11.3)**

Au cours de la réalisation de notre projet, il est possible que des situations ou des problèmes techniques, matériels, ou de gestion surviennent, mettant ainsi en péril la réussite de celui-ci. Pour préparer ces éventualités, nous avons établi un tableau qui récapitule les différents types de problèmes potentiels, leurs niveaux d'importance (sur une échelle de 1 à 5) et des solutions envisageables. Cela nous permettra de réagir rapidement et efficacement en cas de besoin pour minimiser les retards et les coûts supplémentaires.

Risque	Niveau d'importance	Solutions envisageables
Manque de communication sur l'avancement des tâches	4	<ul style="list-style-type: none"> <li>- Repensé la solution de gestion de tâche</li> <li>- Augmenter la fréquence des réunions</li> </ul>
Panne d'équipement du drone ou du Rover	3	<ul style="list-style-type: none"> <li>- Manipuler le drone avec plus de délicatesse</li> <li>- Utiliser le simulateur Gazebo tant que les robots sont en panne</li> </ul>
Mauvais choix d'architecture ou de patrons de conceptions	3	<ul style="list-style-type: none"> <li>- Choisir des patrons qui ne sont pas des anti-patrons</li> <li>- Faire du code réutilisable et facilement modifiable</li> </ul>
Mauvaise gestion du temps	3	<ul style="list-style-type: none"> <li>- Prendre le temps de bien estimer la durée d'implémentation d'une</li> </ul>

		fonctionnalité - Avertir des avancements de chaque tâche pour qu'un retard puisse être prévisible
Capacité du serveur trop petite pour la quantité de requêtes	2	- Optimiser le nombre de requêtes à ce qui est strictement nécessaire - Instancié un second serveur pour prendre en charge des requêtes en plus
Mauvaise compréhension de la fonctionnalité à implémenter	3	- Parler avec les membres de l'équipe pour s'assurer d'une compréhension commune d'une tâche - Poser des questions de précision au client si une tâche est ambiguë

*Tableau 6 : Niveau d'importance des risques envisageables*

### 5.3 Tests (Q4.4)

Afin de garantir la qualité de chaque sous-système, nous effectuerons des tests unitaires et des tests d'intégration pour chacun d'entre eux. Ces tests nous permettront de vérifier que chaque partie du système fonctionne correctement et de détecter les erreurs potentielles avant la mise en production.

Nous utiliserons donc:

- Pour la partie client de l'application web, nous utiliserons les librairies de tests incluses dans Vue.js (Fu, et al., 2023) pour faire tous nos tests unitaires et pour viser une couverture de code et de branches de 100%.
- Pour le serveur et le code embarqué en Python (Python Software Foundation, 2023), nous nous servirons de la librairie de test unittest (Python Software Foundation, 2023) pour rédiger des tests unitaires pour chaque fonction du code.
- Pour les tests matériels tels que les déplacements d'un robot ou les itinéraires qu'il emprunte, nous procéderons d'abord à des tests dans le simulateur Gazebo (Open Source Robotics Foundation, 2014) pour garantir un fonctionnement optimal de la logique. Ensuite, nous validerons ces fonctionnalités sur les robots réels, dans l'espace dédié à cet effet.



## **5.4 Gestion de configuration (Q4)**

En ce qui concerne la gestion de configuration, nous stockerons notre code sur notre dépôt GitLab (GitLab B.V., 2023), sur la branche principale (main). Cette gestion de configuration nous permet de gérer les différentes versions des fichiers et de pouvoir facilement retourner à des versions antérieures ou créer de nouvelles versions, sans impacter le projet en cours.

Notre projet sera séparé en quatre parties, un dossier pour la station au sol composé d'un client et d'un serveur, un dossier pour le code du CogniFly (The CogniFly Project, 2023) un dossier pour le Limo (Agilex Robotics, 2022) et un dernier dossier pour la simulation Gazebo (Agilex Robotics, 2021). Chaque dossier comprend un Dockerfile (Docker Inc., 2023) pouvant lancer les parties séparément. Toute la documentation sera effectuée en français, mais le code source sera écrit en anglais.

Pour les tests, comme mentionnés dans la partie 5.3, nous effectuons des tests unitaires sur toutes les fonctions possibles, et pour les autres tests, nous les testerons de manière matérielle, dans le simulateur Gazebo (Open Source Robotics Foundation, 2023) puis avec les robots réels.

Toutes les données relatives à une exploration seront stockées sur SQLite (SQLite Consortium, 2023), elles seront disponibles sur le client par le biais du serveur web qui communiquera avec la base de données.

## **5.5 Déroulement du projet (Q2.5)**

Comme décrit dans ce document, nous avons choisi d'utiliser un robot au sol et un drone pour répondre à l'appel d'offres. La simulation du rover (Agilex Robotics, 2022) nous a permis de progresser dans la réalisation des exigences de R.F.4 et R.F.5, c'est-à-dire d'être capable d'explorer ainsi que d'éviter les obstacles. Cependant, nous avons rencontré des ralentissements dans le processus pour faire fonctionner physiquement les robots. Nous avons dû changer de cartographe, vers gmapping (Gerkey, 2020) puisque celui choisi au départ ne nous a pas donné les résultats escomptés.

Concernant le drone, nous n'avions eu accès à la simulation qu'à la fin de la troisième semaine après avoir commencé le CDR. Contrairement au rover, nous n'avions jamais simulé un drone avant le CDR. Il nous a donc fallu un certain temps pour installer les données nécessaires pour faire fonctionner celui-ci. Après plusieurs tentatives, nous n'étions toujours pas capables de faire fonctionner la simulation. Enfin nous avons changé les paramètres de la simulation utilisée pour résoudre ce problème.

Tester physiquement le fonctionnement du drone nous a posé de nombreux problèmes. Ces problèmes étant liés principalement au matériel, nous avons eu à tester plusieurs drones qui n'ont pas pu mener nos tests à un résultat concret. Nous avons finalement eu accès à un drone fonctionnel, mais qui ne possède pas de caméra comme c'est un Raspberry Pi Zéro (Raspberry Pi, s.d.) et que la connectique pour les caméras à notre disposition ne sont pas les mêmes.

En ce qui concerne l'équipe, nous avons dû gérer le fait qu'un membre de notre équipe (Anne Jolette) a dû arrêter le projet pour des raisons de santé. Nous avons donc dû nous répartir les tâches dont elle devait s'occuper pour le CDR et le RR.

En raison de ces différents problèmes rencontrés, nous n'avons pu respecter le calendrier établi pour la planification des tâches prévues lors du CDR. Cependant, l'équipe a été suffisamment agile pour s'adapter ainsi que prendre l'initiative de trouver des solutions rapidement et efficacement. Nous nous sommes réparti à nouveau les tâches lors de la fin du CDR pour prendre en compte les différents problèmes rencontrés et nous avons réussi à suivre cette nouvelle planification, ce qui montre également de l'évolution dans notre estimation de durée et de complexité des tâches.

Durant la dernière phase du projet (entre la fin du CDR et la remise du RR), tout s'est donc passé comme prévu, nous avons travaillé de manière agile et avons suivi les temps impartis pour chaque tâche.

## **6. Résultats des tests de fonctionnement du système complet**

### **(Q2.4)**

Le processus suivant peut être observé: en premier lieu, les drones physiques répondent à la commande d'identification. Que ce soit les drones physiques ou ceux de la simulation (Agilex Robotics, 2021), ils peuvent lancer et terminer leur mission, qui consiste à explorer leur environnement tout en étant capables d'éviter les obstacles. De plus, l'état des drones peut être affiché sur une interface graphique adaptable à divers appareils tels que des ordinateurs, des tablettes ou des téléphones mobiles, pouvant être utilisée simultanément sur plusieurs appareils. Des logs de débogage sont également affichés dans un onglet distinct, contenant les lectures des senseurs et les commandes envoyées par la station au sol. Ensuite, une carte a été générée par le Limo (Agilex Robotics, 2022) mais aussi par le Cognifly (The CogniFly Project, 2023), contenant la position des deux robots en temps réel. Les deux robots peuvent faire un retour à la base grâce à un bouton présent dans l'interface utilisateur, mais aussi lorsqu'ils possèdent moins de 30% de batterie. La batterie est calculée en pourcentage en partant des voltages restant de manière linéaire et elle est affichée pour chaque robot dans l'interface utilisateur. De plus, la génération d'un monde avec 3 murs générés aléatoirement est faite et la simulation comme la station au sol peuvent être lancées avec une seule commande. De multiples tests unitaires ont été effectués, tant côté client que serveur ou pour les fichiers python utilisés par les robots. Nous avons aussi fait attention à garder un format de code standardisé. Une méthode P2P est également utilisée pour connaître le robot situé le plus loin de son point de départ, pour que ce dernier émette un son ou allume une DEL. Cette fonctionnalité est activable et désactivable par le biais d'un bouton "P2P" dans l'interface utilisateur. Ensuite, notre système peut fonctionner avec un seul ou deux robots lorsqu'on est dans la simulation. Finalement, lorsque le drone est retourné, le système détecte le crash du Cognifly (The CogniFly Project, 2023) et l'affiche dans l'interface utilisateur dans l'emplacement de l'état de ce dernier.

Par ailleurs, actuellement, certains aspects du projet ne fonctionnent pas encore pleinement. Pour des raisons matérielles, le seul CogniFly (The CogniFly Project, 2023) fonctionnel que nous pouvons utiliser ne possède pas de caméra, il n'est donc pas possible pour nous de générer une carte de l'environnement avec ce dernier dans des conditions réelles. Notre système est aussi dépendant du Limo (Agilex Robotics, 2022) dans le cas des robots réels car la librairie du CogniFly (The CogniFly Project, 2023) est appelé à travers ce dernier comme nous avons eu du mal à la faire fonctionner directement sur le CogniFly.

## **7. Travaux futurs et recommandations (Q3.5)**

La partie principale restant à compléter dans le système est d'ajouter une caméra fonctionnelle au CogniFly (The CogniFly Project, 2023) afin d'implémenter la génération d'une carte en temps réel pour le CogniFly, comme nous l'avons fait dans la simulation. Une autre amélioration possible du système serait d'utiliser directement l'API CogniFly sur le CogniFly lui-même, plutôt que de passer par le Limo (Agilex Robotics, 2022), ce qui peut prendre du temps pour se connecter au CogniFly, rendant ainsi chaque robot plus autonome dans ses déplacements et se rapprochant davantage d'un véritable drone autonome. Cela permettrait également un mouvement complètement autonome du drone en utilisant un outil pour générer des positions de destination plutôt que des positions fixes.

Une recommandation pour améliorer le système existant serait de créer une carte en 3D plutôt qu'en 2D, ce qui améliorerait l'expérience utilisateur mais nécessiterait également un matériel différent de celui d'une caméra 2D pour le CogniFly, comme par exemple un Lidar similaire à celui du Limo. Une autre recommandation serait de fournir, en plus de la carte générée en temps réel, des images provenant des caméras des robots pour permettre une surveillance de l'environnement des robots par les humains, ce qui pourrait révéler des détails imperceptibles lors de la création de la carte.

## 8. Apprentissage continu (Q12)

### 8.1 *De Blas David*

Concernant mes lacunes, je pense en avoir identifié plusieurs. Tout d'abord, j'ai remarqué en début de projet avoir pris du temps à m'adapter à l'utilisation de nouveaux outils comme Zenoh ou Docker. D'autre part, en essayant de mettre en relation une communication entre Zenoh et Python, j'ai remarqué que je n'avais pas une compréhension approfondie de la programmation asynchrone en Python.

Les méthodes que j'ai employées pour remédier à ces lacunes ont été, tout d'abord, de visionner des vidéos explicatives sur le fonctionnement de certains concepts et outils. Ensuite, j'ai expérimenté et cherché des exemples en ligne où des personnes rencontraient des problèmes similaires en m'inspirant de leurs solutions. Finalement, j'ai pris le temps de lire attentivement la documentation des outils concernés.

J'aurais pu comprendre plus rapidement comment tout cela fonctionnait en allant directement lire la documentation officielle de chaque outil car cela m'aurait donné une compréhension plus claire et complète des outils et concepts dès le départ. Cela m'aurait aussi évité de perdre du temps dans la recherche de problèmes similaires mais pas forcément adaptés à ma situation.

### 8.2 *Deloumeau Nicolas*

D'abord, les aspects qui m'étaient moins familiers étaient le langage Python (Python Software Foundation, 2023) ainsi que le protocole Zenoh (Eclipse Foundation, 2023). Je n'avais que peu d'expérience en Python, et aucune dans la programmation asynchrone ainsi que la synchronisation dans le langage.

Pour y remédier, j'ai principalement lu la documentation ainsi que certains guides. Je suis aussi allé explorer le code source, par exemple pour comprendre l'API du Cognifly (The CogniFly Project, 2023) en profondeur. Pour me familiariser avec la programmation asynchrone, j'ai créé quelques programmes

personnellement pour expérimenter certaines fonctionnalités. Le protocole Zenoh était plus simple que d'autres protocoles que j'avais utilisé dans le passé, donc la documentation était largement suffisante pour l'utiliser.

### **8.3 Desgroseilliers Philippe**

Même si je suis content avec le résultat final du projet, je suis très conscient de mes lacunes que j'ai éprouvées au cours de la session. Tout d'abord, j'ai vite réalisé que je manquais de connaissances en relation avec l'utilisation de Docker (Docker Inc., 2023). Bien que Docker soit un des outils les plus populaires dans le développement de logiciels, il peut être difficile à maîtriser pour une personne qui n'a pas d'expérience pratique avec ce système. J'ai vite commencé à lire la documentation qui est disponible sur le site officiel pour répondre à mes questions immédiates que j'avais en travaillant dessus. Je me trouve maintenant, à la fin de la session, avec des connaissances assez minimales, mais assez suffisantes pour être satisfait de ma contribution au projet. En rétrospective, j'aurais pu travailler plus sur les connaissances générales de Docker et non seulement sur les aspects précis qui nous concernaient. Ceci aurait peut-être pu porter fruit pour créer des images Docker plus efficacement.

Une deuxième lacune que j'ai vite identifiée serait le manque de proactivité pour parler de mes problèmes en relation avec le code avec mes collègues. Plus souvent que j'aimerais l'avouer, je me suis trouvé en retard vis-à-vis l'échéancier établi ce qui a retardé le projet à quelque reprise. Une des manières que j'ai pu résoudre ce problème dans ce cadre était de me forcer à parler en détail de mon progrès hebdomadaire et pourquoi il n'était possiblement pas où il était supposé l'être. Un autre remède que j'aurais pu utiliser pour aider ce problème serait de me forcer à détailler mon avancement hebdomadaire plus en détail que je le faisais lors des réunions. Je me trouvais parfois timide vis-à-vis mon avancement ce qui laissait mes coéquipiers dans le noir. Parler plus ouvertement de mes problèmes et essayer de négocier une solution avec

un autre cerveau favorise les relations inter-coéquipiers et rend l'équipe plus performante.

#### **8.4 Louesdon Théo**

Durant le projet, j'ai identifié des lacunes dans mes savoirs et savoir-faire en ce qui concerne la prédiction de la durée d'une tâche et le système de communication Zenoh (Eclipse Zenoh, 2023). J'ai réalisé que je sous-estimais souvent le temps nécessaire pour accomplir certaines tâches et je n'avais jamais utilisé Zenoh, j'ai donc dû apprendre à l'utiliser rapidement pour le PDR en premier lieu.

Pour remédier à la difficulté de planification de tâches, j'ai tiré des leçons de mes erreurs lors de la phase de Preliminary Design Review (PDR) et j'ai cherché à améliorer ma méthodologie pendant la phase de Critical Design Review (CDR). J'ai notamment effectué davantage de recherches sur les composantes nécessaires à réaliser avant de procéder à la prédiction de la durée d'une tâche. Pour la communication Zenoh, j'ai continué à faire des recherches et j'ai surtout essayé de mettre en place ce que j'avais compris pour ensuite l'améliorer si nécessaire.

En rétrospective, j'aurais pu améliorer l'aspect de planification de tâches en consacrant plus de temps à la planification et à l'estimation réaliste des durées de tâches dès le début du projet. Cela m'aurait permis d'être plus efficace dans la gestion de mon temps tout au long du projet. Pour Zenoh, j'aurais pu passer plus de temps dès le début à apprendre ce mode de communication pour avoir moins d'essai de mon code à faire avant que ça fonctionne comme attendu.

#### **8.5 Ngay Munga Exaucé-Constantin**

Je me suis lancé dans ce projet avec une grande motivation pour apprendre et améliorer mes connaissances. Bien que je possède déjà des compétences en langage Python, en développement d'applications Web, en

bases de données, en programmation système embarquée et en utilisation de Git, j'ai abordé ce projet avec l'objectif d'élargir mes compétences.

J'ai pu apprendre les concepts de Docker, les installations nécessaires et comment l'utiliser dans notre projet. J'ai également acquis la compréhension du protocole de communication avec Zenoh et comment l'utiliser. Car en effet, je n'avais jamais travaillé avec Docker et Zenoh auparavant, j'ai vu cela comme un défi et je me suis investi pleinement dans leur compréhension.

J'ai rencontré un moment de ralentissement dans mon apprentissage, principalement pendant la phase de Critical Design Review (CDR). Cependant, j'ai redoublé d'efforts pour ne pas ralentir mon équipe. Heureusement, mes collègues ont été d'une grande aide afin de m'aider à apprendre m'aider à apprendre et à améliorer ma contribution au projet.

Grâce à ce projet, j'ai pu améliorer ma manière de réaliser des tests unitaires en TypeScript, ainsi qu'apprendre à en faire avec le langage Python. La communication et la collaboration ont été des éléments clés de la gestion de notre projet, en permettant la compréhension commune des tâches à réaliser et minimiser les retards. Ce projet m'a donc permis de développer mes compétences et de progresser tout en contribuant à l'effort collectif de l'équipe.

## **9. Conclusion (Q3.6)**

En conclusion, la démarche de conception a progressivement évolué par rapport à ce qui avait été initialement envisagé. Toutefois, cette divergence était prévisible compte tenu de l'envergure du projet en question, ainsi que de l'utilisation de nouveaux robots, ce qui a nécessité d'importants ajustements tant au niveau de la simulation que du déploiement réel. Malgré cela, nous avons réussi à livrer un système fonctionnel dont nous sommes fiers. Ainsi, nous avons dû affronter divers obstacles, qu'ils soient d'ordre logiciel ou matériel, et dans la grande majorité des cas, nous avons réussi à les surmonter grâce à notre travail d'équipe et à notre volonté de solliciter de l'aide lorsque nécessaire. En définitive,



grâce à une vision claire qui s'est affinée à chaque étape de remise (PDR et CDR), nous avons été en mesure de livrer un produit qui répond aux exigences de l'appel d'offres, avec un fonctionnement fiable et robuste.

En ce qui concerne nos hypothèses initiales, toutes ont été vérifiées pour le Limo. En effet, la simulation était très proche de la réalité, avec des communications rapides et un matériel fiable, sans aucun problème. En revanche, pour le Cognifly, deux des trois hypothèses n'ont pas été vérifiées, car le modèle dans la simulation différait considérablement du Cognifly réel, tant au niveau du matériel robuste que du logiciel souvent peu fiable dans la plupart des Cognifly mis à disposition. Les communications restaient cependant rapides, que ce soit par le biais de l'API du Cognifly ou de Zenoh.

## 10. Références (Q3.2)

- Agilex Robotics. (2022). *UGV\_Gazebo\_Sim*. GitHub: [https://github.com/agilexrobotics/ugv\\_gazebo\\_sim/tree/master/limo](https://github.com/agilexrobotics/ugv_gazebo_sim/tree/master/limo)
- Collin, J., Desmarais, M. C., & Gendreau, O. (2023). *INF3995 - Projet de conception d'un syst. informatique*. Moodle: <https://moodle.polymtl.ca/course/view.php?id=1703>
- Docker Inc. (2023). *Home*. Docker Telepresence: <https://www.docker.com/>
- Eclipse Foundation. (2023). *Home*. Zenoh: <https://zenoh.io/>
- Eclipse Foundation. (2023). *Reference Manual - Abstractions*. Zenoh: <https://zenoh.io/docs/manual/abstractions/>
- Eclipse Zenoh. (2023). *Zenoh*. GitHub: <https://github.com/eclipse-zenoh>
- Fu, A., Vladimir, Patak, Bagher, M., Demchuk, I., Sánchez, J., . . . Perkkiö, A. (2023). *Get Started*. Vitest: <https://vitest.dev/>
- Gerkey, B. (02 Octobre 2020). *gmapping*. ROS Wiki: <http://wiki.ros.org/gmapping>
- Hedges, R., Støy, K., Bers, J., Douglas, J. K., Jinsuck, K., Martignoni, A. I., . . . Dandavate, G. (s.d.). *gazebo*. ROS Wiki: <http://wiki.ros.org/gazebo>
- Meeussen, W. (02 Mars 2021). *robot\_pose\_ekf*. ROS Wiki: [http://wiki.ros.org/robot\\_pose\\_ekf](http://wiki.ros.org/robot_pose_ekf)
- Open Source Robotics Foundation. (2014). *Gazebo Tutorials*. Gazebo Sim: <https://classic.gazebosim.org/tutorials>
- Open Source Robotics Foundation. (2014). *ROS overview*. Gazebo Sim: [https://classic.gazebosim.org/tutorials?tut=ros\\_overview](https://classic.gazebosim.org/tutorials?tut=ros_overview)
- Open Source Robotics Foundation. (2023, Mars 14). *Gazebo Classic*. GitHub: <https://github.com/gazebosim/gazebo-classic>
- Open Source Robotics Foundation. (2023). *Get Started*. Gazebo Sim: <https://gazebosim.org/home>
- Python Software Foundation. (s.d.). *Python*. GitHub: <https://github.com/python>
- Python Software Foundation. (s.d.). *Python*. Home: <https://www.python.org/>
- Python Software Foundation. (s.d.). *Unit testing framework*. Python Documentation: <https://docs.python.org/3/library/unittest.html>

Raspberry Pi. (s.d.). Raspberry Pi Zero.  
<https://www.raspberrypi.com/products/raspberry-pi-zero/>

Ricard, G. (2023). *Simple Quad Gazebo*. GitHub:  
[https://github.com/willRicard/simple\\_quad\\_gazebo/tree/noetic-devel/src/simple\\_quad](https://github.com/willRicard/simple_quad_gazebo/tree/noetic-devel/src/simple_quad)

ROS - Robot Operating System. (s.d.). *ROS Core Stacks*. GitHub:  
<https://github.com/ros>

ROS. (s.d.). *Home*. ROS - Robot Operating System: <https://www.ros.org/>

The CogniFly Project. (21 Février 2023). *Cognifly Python*. GitHub:  
<https://github.com/thecognifly/cognifly-python>

VueJs. (s.d.). *Vue*. GitHub: <https://github.com/vuejs/vue>

You, E. (s.d.). *Get Started*. Vue.js - The Progressive Javascript Framework:  
<https://vuejs.org/>

You, E. (s.d.). *Introduction*. Vue.js - The Progressive Javascript Framework:  
<https://vuejs.org/guide/introduction.html>