
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. A2022-INF3995 du département GIGL.

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

Équipe No 101


| Nom et Prénom | Signature |
|---|---|
| Paul Marie Akoffodji | P.M |
| M. Ibrahim Majid | M.I.M |
| Manel Mokrani | M.M |
| Erika Fossouo | E.F |
| Patrick Cobanovic | P.A.C |


01 Octobre 2022

# 1. Vue d'ensemble du projet

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

Le but du projet est de concevoir un ensemble de logiciels opérationnels permettant à un essaim de drones miniatures d'explorer une pièce ou un compartiment de manière autonome et de rapporter des métriques de vol à l'opérateur au moyen d'une interface web. En effet, le drone doit répondre à bon nombre de critères et de requis d'ordre matériels, logiciels et fonctionnels. Ces derniers doivent être présentés ultérieurement à l'aide de vidéos démonstratives. L'utilisateur doit donc pouvoir interagir avec une interface utilisateur afin de lancer ou donner des instructions. Ces dernières peuvent être données aux drones physiques ainsi qu'aux drones simulés.

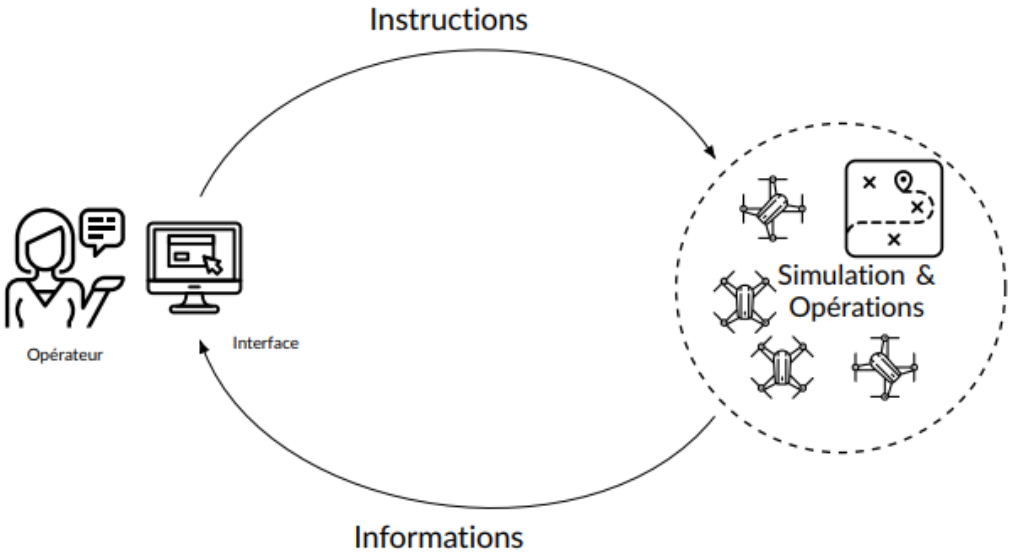


Figure 1 : Schéma de communication globale

En ce qui concerne les objectifs, la réalisation du projet nécessite la conception de trois grandes parties : la station au sol, la partie embarquée sur les drones physiques et la partie simulée avec Argos.

À long terme, le projet permettra d'offrir des options de solutions dans le cadre de l'exploration spatiale de terres inconnues, comme Mars, en se basant sur de

nouvelles approches d'exploration, comme l'utilisation d'un essaim de drones pour optimiser l'exploration de ces territoires.

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

**1.2.1-** Dans le cadre du projet, plusieurs hypothèses seront faites :

- Les drones seront utilisés à des fins d'exploration dans un cadre académique et ne seront pas utilisés à d'autres fins.
- On pourra supposer que les drones voleront dans un espace fermé sécuritaire avec la présence de murs et d'obstacles.
- On assumera qu'au moins un drone de l'essaim est toujours en communication avec la station au sol.
- En trouvant la solution pour opérer sur deux drones il serait aisé de l'implémenter sur dix drones.

**1.2.2** - En termes de contraintes

- Les drones ne doivent pas décoller avec un niveau de batterie inférieur à 30%.
- Le retour à la base doit rapprocher les drones de leur position de départ pour qu'ils soient à moins de 1 m de celle-ci.
- L'environnement simulé doit avoir un minimum de 3 murs (sans compter les 4 murs extérieurs)
- Toutes les composantes logicielles, sauf celles embarquées sur le drone physique, doivent être conteneurisées avec Docker.
- Le seul moyen de communication entre la station au sol et les drones physiques doit être une Bitcrazy Crazyradio PA connectée à la station au sol.

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

Tout au long du projet, nous allons livrer des livrables spécifiques à différents moments. Voici le récapitulatif de ces dates et une brève présentation du contenu de ces livrables.

| Date | Titre du livrable | Description du livrable |
|---|---|---|
| Vendredi 30 Septembre 2022 à 23h55 | (PDR) Preliminary Design Review | - Plan du projet<br>- Présentation de l'architecture globale<br>- Prototype préliminaire |
| Vendredi 04 Novembre 2022 à 23h55 | (CDR) Critical Design Review | - Prototype avancé<br>- Interface web évoluée avec la présence des commandes complémentaires<br>- Présence de la base de données conservant les métriques de vol<br>- Cartographie des surfaces explorées |
| Vendredi 07 Décembre 2022 à 23h55 | (RR) Readiness Review | - Implémentation de toutes les fonctionnalités<br>- Plan et résultats des tests<br>- Retour sur l'apprentissage continu |

# 2. Organisation du projet

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

Pour les besoins du projet et au vu de la charge de travail nous avons décidé de faire des binômes et d'avoir des personnes en renfort pour épauler les binômes ou de travailler individuellement sur des petits modules du projet. Il en résulte l'assignation de rôles suivante:

- Majid : Coordonnateur de projet

**Responsabilité principale** : Logiciel embarqué des drones
**Responsabilités annexes**:

- ❖ Scrum Master
- ❖ Planification des rencontres et répartition des tâches

- Paul Marie : Analyste développeur

**Responsabilité principale** : Logiciel embarqué drones
**Responsabilités annexes**:

- ❖ firmware
- ❖ Interface utilisateur

- Manel : Analyste développeuse

**Responsabilité principale:** Interface utilisateur
**Responsabilité annexe**: Simulation Argos

- Patrick : Analyste développeur

**Responsabilité principale** : Simulation Argos
**Responsabilité annexe** : firmware

- Erika : Analyste développeuse

**Responsabilité principale**: Interface utilisateur
**Responsabilité annexe** : Simulation Argos

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

Pour le compte du projet, nous avons opté pour un **contrat en régie avec honoraires fixes.** L'agence a spécifié ses attentes. Ces attentes sont peu susceptibles de changer au cours du projet. De plus, elle a également fixé la charge horaire globale pour la réalisation du projet. Les honoraires ne seront versés que lorsque le travail sera achevé et ne changeront pas en raison des performances. Cela permettra donc à l'équipe de développeurs de toucher leurs

honoraires tout en n'ayant pas de fortes contraintes de rendement, mais en veillant néanmoins à livrer un produit de qualité à la fin du projet. Cela est un aspect important à notre avis dans la réalisation du projet.

# 3. Description de la solution

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

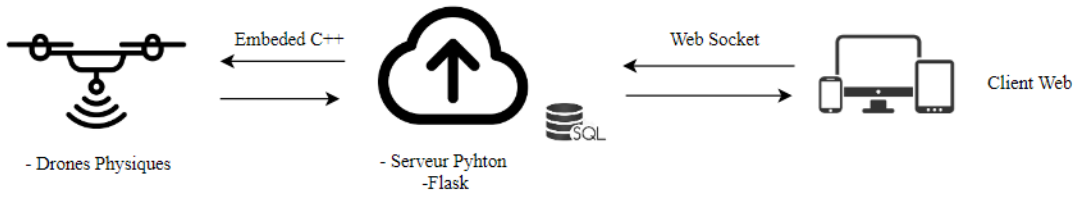


Figure 2 : Schéma Architecture logicielle générale

Ci-dessus le schéma de l'architecture globale de notre système en partant d'une vue générale. On aura a priori trois grands modules:

- **Le client web** : représenté par l'opérateur qui à travers l'interface web donne les instructions de haut niveau aux drones et choisit le type de support pour le lancement de la mission (physique vs simulation). Pour l'implémentation de cette partie, nous avons choisi d'utiliser le framework Angular associé au typescript, car ils sont familiers pour tous les membres de l'équipe.
- **Le serveur backend python** : Il aura pour rôle de faire le lien entre les actions de l'opérateur via l'interface web et le système de communication des drones physiques. De plus, le serveur recueillera les informations collectées par les drones physiques afin de les afficher dans l'interface web conformément au requis R.C.1, via la communication web socket et/ou http client. Il sera aussi en mesure de conserver les données liées à l'historique

des missions effectuées dans une base de données. Nous avons choisi d'utiliser le framework Flask pour notre serveur.

- **La station de drones de physique:** Cette station est composée de deux drones physiques, ces derniers communiquent des informations destinées au serveur web. Ils reçoivent des instructions provenant du client web également.

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

La station au sol a pour objectif de donner des directives aux drones aussi bien les physiques que ceux présents en simulation. En effet, nous avons opté pour l'instanciation de trois principaux modules pour la communication à ce niveau.

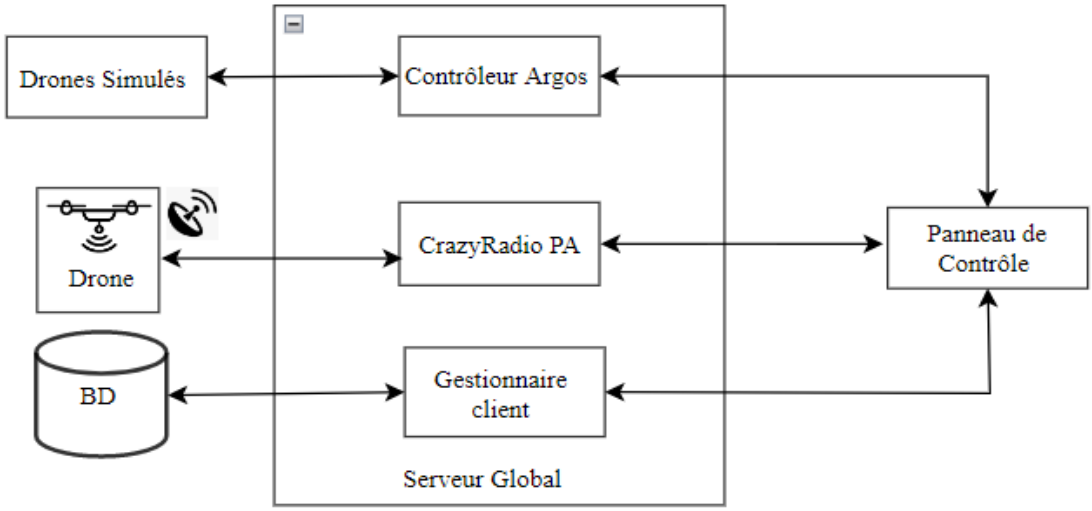


Figure 3 *:* Schéma Station au Sol

On distinguera :

- **Le Contrôleur Argos** : Ce dernier sera codé en C++ majoritairement afin de permettre aux drones de s'orienter et de se diriger dans la simulation.
- **Le CrazyRadio PA**: Ce module permettra d'établir la communication radio entre les drones et la station au sol conformément au requis R.M.4 et permettra à ce dernier d'envoyer des informations et des paquets qui seront sauvegardés dans la base de données conformément au requis R.F.8

- **Gestionnaire de Client:** Ce module permettra d'assurer la gestion des clients qui essaient de se connecter afin de suivre en direct une mission en cours. Nous avons choisi dans ce cas de restreindre l'utilisation de l'interface à un seul utilisateur conformément au requis R.F.10.
- La station au sol sera également en mesure de conserver les données liées aux missions effectuées en communiquant avec une base de données de plus les données liées aux cartes d'exploration élaborée par les drones devront être enregistrées sur la station au sol , cela répondra aux requis R.F.17 et R.F.18.

## *3.3 Logiciel embarqué (Q4.5)*

Le code embarqué ainsi que le code d'exploration du produit final sera implémenté sur les boards des drones en utilisant l'API de BitCraze. Afin de répondre aux requis R.L.1 et R.L.3. L'utilisation du «ranging deck» et du «Flow deck V2» sera primordiale dans cette partie de la structure logiciel du code embarqué, afin de permettre aux drones de détecter les objets ainsi que les obstacles de manière autonome grâce aux métriques offertes par les senseurs montés sur les drones physiques conformément aux requis R.M.3 et R.F.5.

Un système de machine à états pourrait également nous permettre à ce niveau de définir une approche d'exploration pour nos drones physiques. Une succession d'états et d'actions sera faite dépendamment de certaines situations dans lesquelles les drones se trouvent. Cela entraînera des transitions entre les états afin de permettre aux drones de varier leurs actions et d'effectuer des mouvements spécifiques dans la suite du projet tout en effectuant l'exploration.

De plus, nous allons essayer de nous baser sur le principe du "wall-following" afin de rendre les mouvements du drone autonomes conformément au requis R.F.4. Toute cette logique sera donc mise sur les drones directement .

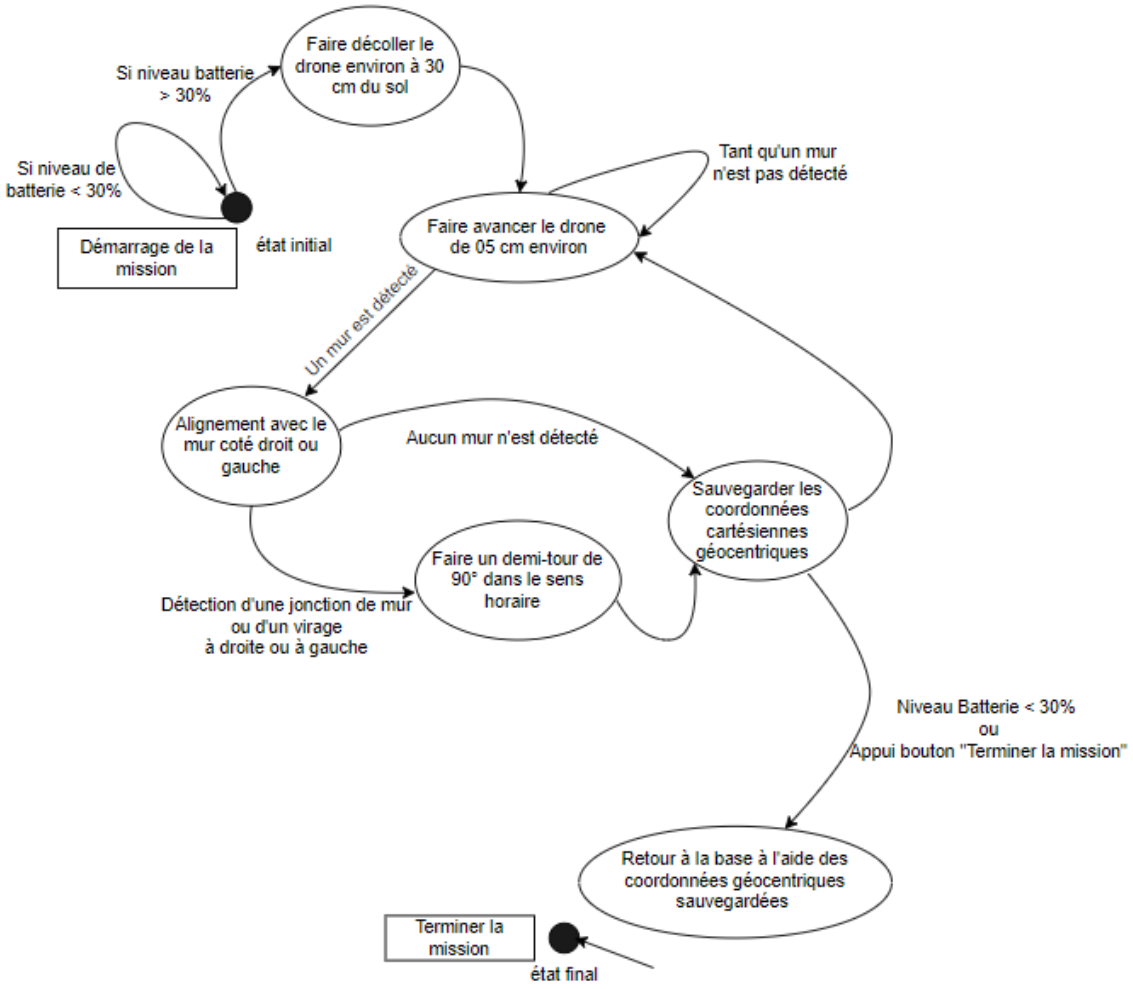


Figure 4 : Schéma Logiciel embarqué

Le diagramme ci-dessus présente une succession des états de changements du drone. Globalement, le principe est de faire décoller le drone au-dessus du sol puis de le faire avancer par la suite sur la base des données du senseur. Dès qu'il rencontre le premier mur, il s'aligne avec ce dernier. Par la suite, s'il rencontre un autre mur ou une jonction de mur, il effectue une rotation à 90° dans le sens horaire. Tant qu'il n'a pas détecté de mur, il continue d'avancer tout en sauvegardant en boucle les coordonnées géocentriques afin de mémoriser sa position. La logique de retour à la base a été représentée dépendamment du niveau de la batterie ou de l'action de l'opérateur dans l'interface utilisateur. Elle utilisera les coordonnées de position sauvegardées au cours de la mission conformément aux requis R.F.6 et R.F.7.

En conséquent, l'état "crashed" des drones pourra être détecté en analysant la valeur de la coordonnée en hauteur une fois le drone en vol sachant que l'on a décollé d'une distance d'une hauteur de dix centimètres. Donc si la valeur renvoyée par le senseur situé au-dessus du drone est inférieure à une distance de 5cm environ, on pourra en déduire que le drone est retourné. Cette approche permettra de répondre au requis R.F.13.

## *3.4 Simulation (Q4.5)*

Le rôle de la simulation est très important dans le projet. Il est primordial, car il permet de tester sur demande les fonctionnalités de la station au sol et de pouvoir tester conceptuellement les algorithmes de contrôle pour les drones avant de les déployer sur les drones physiques.

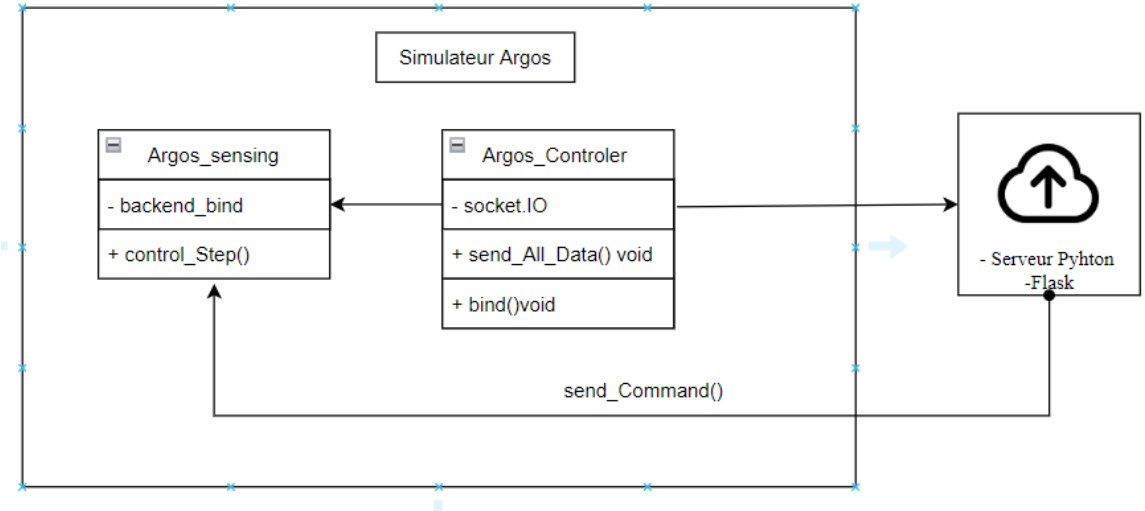


Figure 5 : Schéma Simulation

Idéalement, la simulation et les drones physiques devraient partager certains appels de fonctions et comportements afin de favoriser la réutilisabilité du code. Dans cet ordre d'idée, la classe Argos sensing conservera la logique liée aux drones simulés, puis par le biais du contrôleur Argos elle nous permettra d'envoyer des données relatives aux missions comme des métriques de vol des drones ainsi que leurs états via socket.IO qui est une librairie pour faire de la communication entre le client et le serveur par le biais d'événements. Cette approche nous

permettra de répondre au requis R.F.3. Le serveur backend en python permettra d'envoyer les commandes de haut niveau aux drones présents dans la simulation.

## *3.5 Interface utilisateur (Q4.6)*

Pour l'architecture relative à l'interface utilisateur, nous avons opté pour l'utilisation du framework Angular assez bien documenté et facile à prendre en main. Il sera utilisé afin de produire l'interface web pour le client de notre application. Nous avons opté pour ce framework, car comme évoqué précédemment, la plupart des membres de l'équipe sont assez familiers avec ce dernier, car ils l'ont utilisé tout au long du projet intégrateur 2. L'interface comportera les boutons correspondants aux différentes commandes de haut niveau à envoyer aux drones notamment ''Lancer la Mission'', "Identifier", "Retour à la base", "Terminer Mission", dépendamment de l'option choisie dans l'interface pour les drones physiques ou les drones simulés. De même, dans cette interface on aurait possiblement accès à l'état des drones déployés. Cela permettra de compléter les requis R.F.2, R.F.3, R.F.6, R.F.8 et R.F.13. On pourrait aussi voir la liste des missions effectuées dans une liste déroulante et avoir les détails liés à chaque mission. De plus, on pourrait avoir la possibilité de trier les missions selon les attributs collectés à savoir la date, l'heure, la distance parcourue, etc...

Cela permettrait de répondre aux requis R.F.17 et R.F.18. Les logs comprenant toutes les informations de débogage nécessaires au développement seront également disponibles en tout temps via une fenêtre dynamique dans l'interface utilisateur afin de répondre au requis R.C.1.

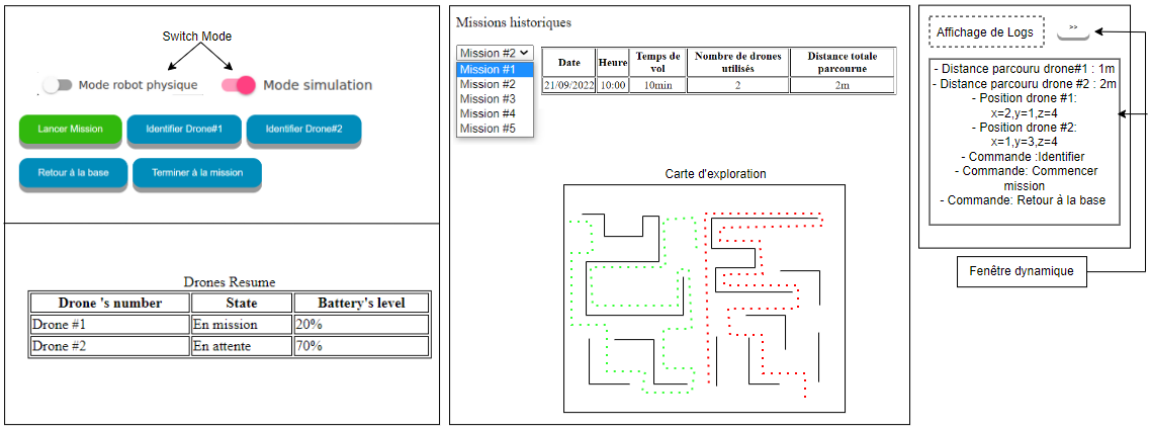


Figure 6 : Schéma interface Utilisateur

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

Afin de lancer notre système, idéalement nous aurions voulu la commande ‘’**sudo docker up -d** ’’ permettra de rouler directement tout le projet. Le client web sera accessible à travers le port 4200 et le serveur quant à lui sera accessible sur le port 5000. L'utilisation d'un laptop équipé d'Ubuntu serait aussi recommandée afin de pouvoir tester et rouler le projet sans problème. Cependant nous avons pour la remise pu faire fonctionner les dockerfiles de nos modules de façon individuelles. Dans le client il faudra **ng serve**, puis dans le serveur faire **flask run.** Pour la simulation il faut commence à rouler l'image docker avec le docker file qui se trouve dans le dossier **./simulation** la commande pour ça c'est **docker build -t simulation:test .** Puis enfin rouler le container avec la commande sur Ubuntu:

```
docker run -it \
    --env="DISPLAY" \
    --env="QT_X11_NO_MITSHM=1" \
    --volume="/tmp/.X11-unix:/tmp/.X11-unix:rw" \
    --publish 8080:8080\
    --publish 8081:8081\
    --publish 8082:8082\

    simulation:test \
```

# 4. Processus de gestion

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

En supposant que chaque membre du groupe passe en moyenne six heures lors des laboratoires et cinq heures en moyenne hors des séances de laboratoire sur le projet on pourrait aboutir à l'estimation des coûts ci-dessous sur une période de trois mois soit douze semaines :

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

| Membre d'équipe | Salaire hebdomadaire | Salaire total |
|---|---|---|
| 145$/h | 145$/h * 11h = 1595$ | 19140$ |
| 130$/h | 130$/h * 11h = 1430$ | 17 160$ |
| Salaire Finale Équipe : 19140$ + (17160$ * 4) = 87 780$ | | |

## *4.2 Planification des tâches (Q11.2)*

L'avancement des tâches va se faire avec l'outil de gestion de projet Gitlab. Chaque requis choisi sera documenté sous forme de "issues" et aura un poids synonyme du niveau de difficulté possible. Grâce à la plateforme, on sera en mesure d'assigner les issues puis de fixer des dates d'échéances. Le projet aura trois principaux milestones assimilables à des sprints correspondant aux échéances de remises des trois livrables. Le premier "milestone" (sprint 1) servira à répondre aux requis du PDR et certaines fonctionnalités de base en autres. Les prochains milestones (Sprint 2 et Sprint 3) seront entre autres alloués à la réalisation des requis restants pour la remise des deux derniers livrables : CDR et RR. La répartition des jalons reste arbitraire et pourrait changer éventuellement en fonction de l'avancement du projet et des contraintes qui se présenteront à nous.

| Tâches | Jalon | Temps alloué | Personne responsable |
|---|---|---|---|
| R.F.1 | Sprint 1 | 04 semaines | Majid /Paul Marie |
| R.F.2 | | 04 semaines | Manel/Erika |
| R.F.3 | | 04 semaines | Patrick |
| R.F.4 | | 04 semaines | Majid/Paul Marie |
| R.C.2 | | 04 semaines | Toute l'équipe |
| R.C.3 | | 04 semaines | Toute l'équipe |
| R.F.5 | Sprint 2 | À venir | À déterminer |
| R.F.6 | | À venir | À déterminer |
| R.F.7 | | À venir | À déterminer |
| R.F.8 | | À venir | À déterminer |
| R.C.1 | | À venir | À déterminer |
| R.F.13 | | À venir | À déterminer |
| R.F.9 | | À venir | À déterminer |
| R.F.17 | | À venir | À déterminer |
| R.F.18 | Sprint 3 | À venir | À déterminer |
| R.C.5 | | À venir | À déterminer |
| R.F.10 | | À venir | À déterminer |

| | | | |
|---|---|---|---|
| R.F.11 | | À venir | À déterminer |
| R.F.12 | | À venir | À déterminer |
| R.F.20 | | À venir | À déterminer |
| R.C.4 | | À venir | À déterminer |

## *4.3 Calendrier de projet (Q11.2)*

Nous pourrons présenter le calendrier du projet à travers le diagramme de Gantt. Ce diagramme nous permettra d'avoir en premier un aperçu global du projet d'une part, il renseigne également sur l'état d'avancement des tâches, permet d'avoir un aperçu de l'allocation des ressources pour chaque tâche, et d'avoir une meilleure approche de gestion de projet. Dans le diagramme ci-dessous, les losanges bleus font référence aux dates cibles de remises du PDR, CDR et RR. Les jalons en vert, bleu et rouge représentent les tranches de temps allouées à la réalisation des requis pour le compte des sprints 1, 2 et 3.

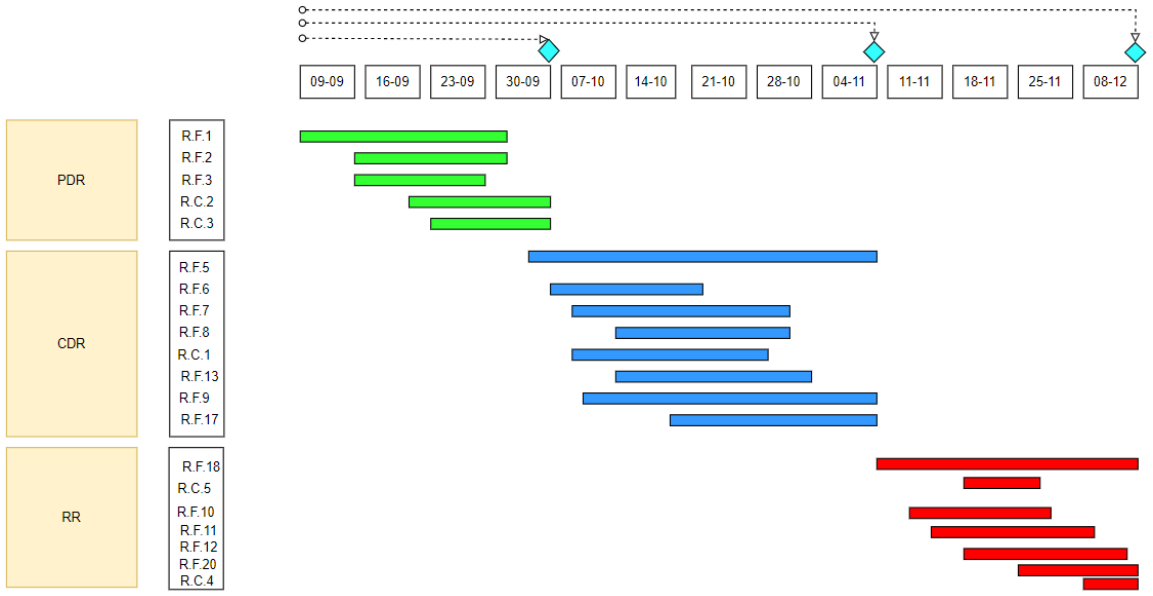

Figure 7 : Schéma Diagramme de Gantt

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

L'équipe est composée de cinq membres possédants chacune des compétences techniques et relationnelles acquises à Polytechnique et aussi grâce à leur expérience en stage. On comptera au sein de l'équipe :

**Paul Marie Akoffodji**
Paul Marie en est à sa troisième année de baccalauréat en génie informatique. Il a de l'expérience avec C/C++. Il a aussi de l'expérience en réalisation d'artéfact de conception logiciel grâce à son stage effectué en équipe de projet avec Giro.

**Ibrahim M. Djido Abdoul Majid**
Majid en est également à sa troisième année en génie informatique, il a de l'expérience en programmation orientée objet C/C++. Il a également travaillé avec Polystar où il s'est familiarisé avec les systèmes de contrôles embarqués notamment.

**Manel Mokrani**
Manel en est également à sa troisième année en génie informatique, elle a de l'expérience avec C/C++. De plus elle a de l'expérience avec la programmation graphique cela pourrait être un atout exploitable au cours du projet.

**Erika Fossouo**
Erika en est également en troisième année de génie informatique est a de l'expérience en C/C++. De plus, elle a des connaissances avec le langage python appliqué au backend.

**Patrick Cobanovic**
Patrick est en quatrième année de génie informatique, il a de l'expérience en C/C++ et Angular. De plus, il a également des connaissances en réseaux informatiques plus spécifiquement en architectures technologiques de transmissions.

# 5. Suivi de projet et contrôle

## *5.1 Contrôle de la qualité (Q4)*

Tout au long du projet, nos livrables seront soumis à un processus de révision continu aussi bien au niveau du code que sur les produits livrables. Pour le code python nous allons utiliser la norme de style "PEP 8 – Style Guide for Python Code" afin de nous assurer que nos noms de classes, de fonctions , de variables soient conformes aux standards de programmation. Pour le code typescript dans le client et pour le Framework Angular, nous allons opter pour la norme ECMAScript afin de rencontrer les standards de programmation. Cela nous permettra de répondre au requis R.Q.1. Pour le code embarqué, nous allons adopter Doxygen comme convention pour le formatage et la présentation. Chaque module du client et du serveur sera vérifié et possédera des tests unitaires précis décrivant le protocole de validation de ce dernier. Quant à l'interface web, il devra répondre aux dix règles heuristiques de l'interface usager web. Enfin, tous les livrables seront revus en groupe avant soumission afin de garantir la soumission d'un travail élaboré.

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

*Au* cours du projet, plusieurs situations ou problèmes d'ordre technique, matériel, ou de gestion peuvent arriver compromettant ainsi la réalisation de ce dernier dans de bonnes conditions. Le tableau ci-dessous résume certains problèmes que nous pourrions rencontrer ainsi que quelques approches de solutions :

| Problèmes | Niveau d'impact | Approches de Solutions |
|---|---|---|
| - Le manque de familiarité des membres de l'équipe avec les nouvelles technologies (Agros, Docker,Crazyflie | 4 | - Procéder à un briefing en groupe des problèmes de logiciels rencontrés<br>- Se familiariser avec la |

| | | |
|---|---|---|
| Client) selon les versions de système d'exploitation | | documentation de ces nouvelles technologies et lien pertinents |
| - Manque de Cohésion d'équipe et de communication sur l'avancement des tâches entre les membres | 4 | - Instaurer un climat de coopération dans l'équipe<br>- Promouvoir la saine émulation des membres de l'équipe |
| - Bris matériels au niveau des drones physiques lors des tests en volière | 3 | - Veiller à utiliser le drone physique uniquement à l'intérieur de la volière<br>- Manipuler le drone avec tact et précision |
| -Bris physiques sur les opérateurs lors des tests avec les drones en volière | 3 | - Utiliser l'équipement de sécurité fourni par l'entreprise contractante (lunettes de sécurité,etc) |
| -Mauvais choix d'architecture ou de patron de conception logiciel pour le projet rendant difficile l'implémentation des fonctionnalités spécifiques | 3 | - Choisir les bons patrons de conception, éviter les antipatrons<br>- Faciliter la réutilisabilité du code |

| | | |
|---|---|---|
| -Mauvaise gestion du temps alloué à certaines tâches et retard dans les remises de fonctionnalités | 4 | - Ne pas sous-estimer la complexité d'une fonctionnalité<br>- Analyser les contours et besoins que l'implémentation d'une fonctionnalité nécessite<br>- Donner des rétroactions périodiques sur l'avancement des tâches |

## *5.3 Tests (Q4.4)*

Évoquer les tests à divers niveaux et tests sur drones physiques et simulés

Pour chaque sous-système, des tests unitaires et des tests d'intégration seront réalisés.

- Pour le web-app client, on pourrait opter pour l'utilisation de Jasmine pour faire les tests unitaires sur nos components et sur nos services .
- Pour le serveur backend python, on opterait pour l'utilisation de la librairie de test intégrée "Python Unitest" afin de tester aisément les modules importants du serveur backend python.
- Nos requêtes via HTTP seront testées également grâce à l'application PostMan.
- Pour le matériel, on pourrait procéder à une vérification visuelle des drones physiques hebdomadairement, afin de voir si toutes les pièces sont en

norme afin de s'assurer qu'il n'y a pas de pièces défectueuses susceptibles de compromettre le vol de ces derniers.

- On pourrait tester également le fonctionnement des boutons 'Identifier Drone', 'Lancer Mission', 'Terminer mission', 'Retour à la base' et 'Mise à jour' dans l'interface afin de s'assurer que nos drones sont identifiables.
- On pourrait également tester le retour à la base des drones pour un niveau de batterie en dessous de 30%.
- On pourrait tester l'envoi des données en continu de nos drones concernant leurs états ainsi que quelques métriques de vol depuis la station au sol aussi bien en simulation que sur les drones physiques.

## *5.4 Gestion de configuration (Q4)*

Pour ce qui est de la gestion de configuration, notre code sera disponible sur notre repos GitLab sur la branche master. Nous comptons séparer le projet en trois gros modules: un dossier pour le web client, un dossier pour la simulation avec Argos et un dernier dossier pour le firmware embarqué pour nos drones physiques. Chaque dossier sera conteneurisé individuellement et contiendra un fichier .git nous permettant de mettre à jour les dossiers individuellement si nécessaire. L'en-tête des fichiers sera documenté et référencé de sorte à expliquer en surface la responsabilité du fichier. Des commentaires seront mis dans le code également afin de renseigner sur certaines fonctions et fichiers. Des fichiers de README.md seront également ajoutés à la racine du projet afin de donner des informations sur la compilation des fichiers et ainsi que les commandes Docker pour rouler les conteneurs.

# 6. Références (Q3.2)


- Maze Escape. *Maze Ezcape with Wall-Following Algorithm*. (Février, 2021). Tiré de : *https://andrewyong7338.medium.com/maze-escape-with-wall-following-algorithm-170c35b88e00*
- Bitcraze. *Getting started with the Crazyflie 2.X.* (Septembre, 2022). Tiré de : *https://www.bitcraze.io/documentation/tutorials/getting-started-with-crazyflie-2-x/*
- Python. *PEP Index.*(Septembre, 2022). Tiré de : *https://peps.python.org/pep-0008/*

- Blog.Net. L'essentiel de la syntaxe Typescript.(Septembre, 2022). Tiré de : *https://cdiese.fr/syntaxe-typescript-en-10-min/*

- Doxygen. *Generate documentation from source code*. (Septembre, 2022) .Tiré de : *https://doxygen.nl/*

# ANNEXES