



Département de génie informatique et génie logiciel

INF3995

Projet de conception d'un système informatique

Rapport final de projet

*Conception d'un système aérien minimal pour
exploration*

Équipe No 204

Émilie Vaudrin - 1960277

Alexandre Talens - 1885948

Julio Dandjinou - 1804263

Farid Elfakhry - 1875036

Alexandre Morinvil - 1897222

Avril 2021

Table des matières

1	Objectifs du projet	2
2	Description du système	2
2.1	Logiciel embarqué (Q4.5)	2
2.2	La station au sol (Q4.5)	5
2.2.1	Gestion de l'environnement	5
2.2.2	Gestion des drones	6
2.2.3	Gestion de la génération de carte	7
2.2.4	Génération de logs	8
2.3	L'interface de contrôle (Q4.6)	8
2.4	Fonctionnement général (Q5.4)	12
3	Résultats des tests de fonctionnement du système complet (Q2.4)	14
4	Déroulement du projet (Q2.5)	15
5	Travaux futurs et recommandations (Q3.5)	16
6	Apprentissage continu (Q12)	17
6.1	Farid El Fakhry	17
6.2	Alexandre Morinvil	17
6.3	Julio Dandjinou	17
6.4	Alexandre Talens	18
6.5	Émilie Vaudrin	18
7	Conclusion (Q3.6)	18

1 Objectifs du projet

L'objectif du projet était de trouver une alternative pour l'exploration planétaire. En effet, l'agence spatiale envoyait déjà des robots de grosse taille pour l'exploration, mais ceux-ci ne couvraient pas beaucoup de terrain. Une autre alternative était d'utiliser plusieurs drones de petite taille pour qu'il couvre plus de terrain en parallèle. La solution technique consiste notamment en un développement d'un programme informatique s'exécutant sur les robots fournis par l'Agence spatiale de Polytechnique. Comme exigé, le système sera composé d'un nombre arbitraire de drones qui pourront explorer de manière autonome leur environnement. Ceux-ci seront contrôlables à l'aide d'une interface utilisateur Web hébergée depuis une station au sol. Cette interface permettra alors d'afficher les informations envoyées par les drones et ainsi reconstruire une carte de l'environnement exploré. Le projet présenté a ainsi pour objectif de démontrer une preuve de concept pour accompagner l'agence dans ses futurs projets. Les détails techniques de la solution proposée seront détaillés ci-après.

2 Description du système

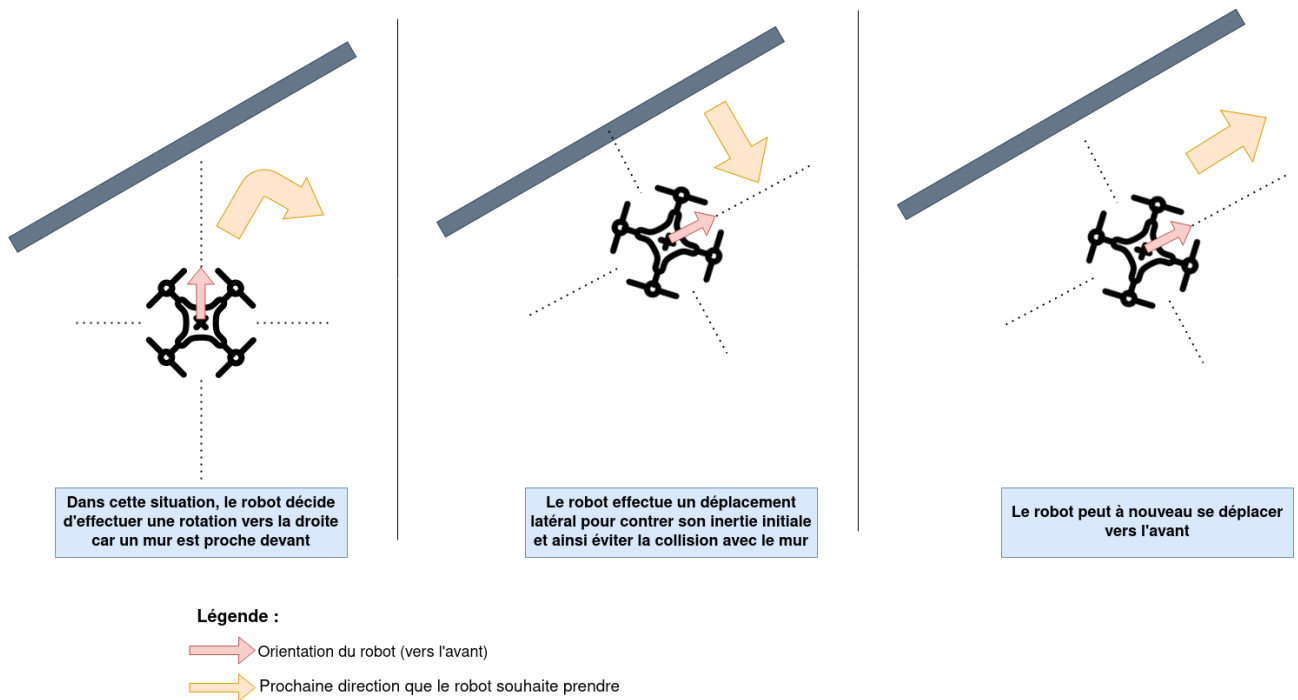
2.1 Logiciel embarqué (Q4.5)

L'architecture générale du logiciel embarqué est très similaire à l'architecture décrite dans la CDR. Nous gardons la même machine à état et la même structure de code. Ainsi nous avons séparé notre code en 3 fichiers principaux. Nous avons le fichier `alfred` qui contient notre main et donc notre machine à état. Nous possédons 2 autres fichiers que l'on pourrait qualifier de fichier ressources : `sens` et `moving`. Ces derniers contiennent la grande majorité de nos algorithmes.

Il est à noter que dorénavant les drones connaissent leur positions absolue dans l'environnement. En effet, comme expliqué plus en détail dans les parties suivante du rapport, le serveur communique aux drones leur positions initiales. Ainsi en effectuant une simple addition de cette position initiale par rapport à leur position courante (relatif à leur système de coordonnées) il leur est possible de connaître leur positions absolue.

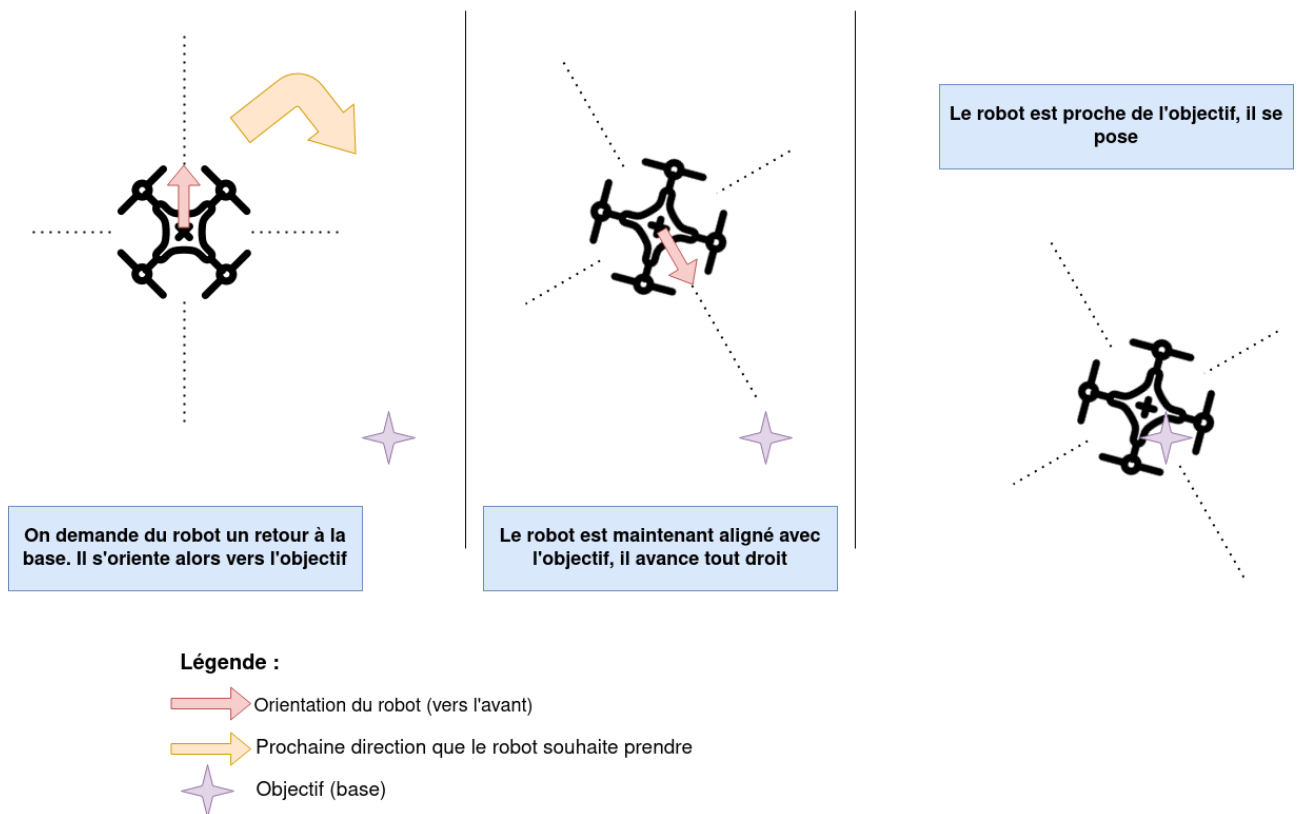
Nous avons apporté des modifications dans le fonctionnement de nos algorithmes. La gestion de la collision a été par exemple grandement modifiée afin d'avoir des résultats supérieurs. Nous allons dans cette section décrire nos trois principaux algorithmes : l'exploration, le retour à la base et l'évitement de collision. Tout d'abord concernant l'exploration, nous avons fait le choix de faire avancer le robot principalement par son avant. Ainsi lorsqu'il rencontre un mur il va pivoter jusqu'à que son capteur avant devienne libre et qu'ils

puisse avancer. Ce principe est illustré dans le schéma ci dessous.



Il est à noter que le robot a la possibilité de se mouvoir de manière latérale afin de compenser son inertie et d'éviter certains obstacles qui ne sont pas visible directement par le laser avant. Ce cas là est montré dans l'état du milieu de la figure.

Ensuite, nous avons le retour à base. Celui-ci est assez simple. L'idée est de se s'orienter vers la position de la base (0, 0, 0) puis d'avancer tout droit. Cette fois-ci en cas de collision nous essayerons d'éviter l'objet qui obstrue le chemin en se déplaçant latéralement. Le choix de la direction à prendre dans ce cas-ci est déterminé de manière optimale par notre algorithme. Ainsi par exemple si un objet se trouve devant le robot et que l'objectif à atteindre se trouve relativement à sa gauche, le robot décidera d'effectuer une translation vers la gauche. La figure ci-dessous illustre un cas simple de retour à la base.

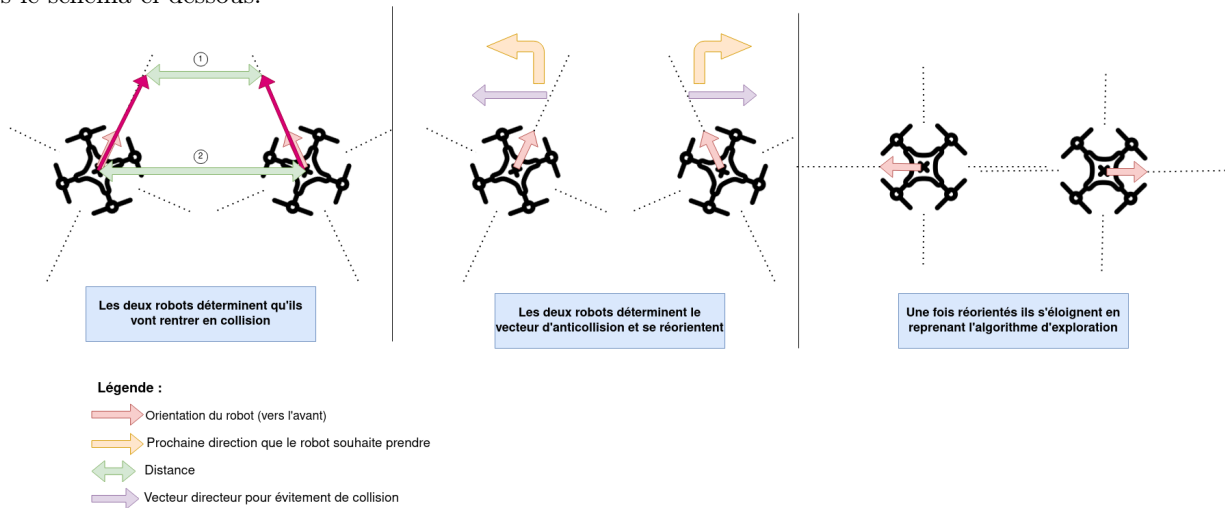


Le robot se posera alors à 2 conditions. Il faut tout d'abord qu'il se trouve proche de la base (à moins de 1m). Cette distance est calculée à l'aide de son système de coordonnées. Ensuite, il doit confirmer cette donnée avec le RSSI. Celui ci devra présenter une valeur suffisamment élevée afin qu'on puisse avoir assez de certitude que le robot soit proche de la base.

Enfin, notre dernier algorithme concerne l'évitement de collision. Comme énoncé plus haut celui ci a été modifié par rapport à la CDR. Précédemment, nous faisions varier l'altitude du robot lorsqu'il allait rentrer en collision avec un autre drone. Il y avait 2 défauts à cette pratique. Le premier est qu'il existait une possibilité où le robot qui passe au dessus perturbe le vol du drone se trouvant en dessous. La deuxième est que dans le cadre d'un nombre arbitraire de drone cela aurait pu mener à des collisions si de nombreux drones étaient proches (l'altitude à se partager deviendrait alors limitée).

Pour résoudre ces problèmes nous avons adopté une approche d'évitement basé sur leur positions absolue. Ainsi les drones se communiquent en tout temps leur positions absolues. Si deux drones détectent une collision alors ils activent l'état d'évitement de collision. Pour déterminer si les deux drones vont rentrer en collision on compare leur distance courante (1) avec la distance de leur projection (2). La projection de la position du drone est calculée par l'addition de leur vecteur vitesse et de leur position. Les deux distances sont indiquées

dans le schéma ci dessous.



Si la différence de ces distances est négative, cela veut dire qu'ils sont en train de se rapprocher. On active alors l'anti-collision. On calcule maintenant le vecteur d'évitement. Celui-ci est dirigé du vecteur projeté de l'autre robot vers le vecteur projeté du robot courant. En faisant ceci on s'assure que les deux robots partent dans une direction opposée. On fait alors pivoter le robot dans la direction d'évitement puis on fait avancer le robot. Lorsque la distance entre les deux drones devient suffisamment élevée, on repasse dans l'algorithme de base d'exploration.

2.2 La station au sol (Q4.5)

La station au sol constitue l'unité de contrôle centrale des interactions entre la flotte de drones ainsi que l'interface utilisateur. Celle-ci permet de gérer quatre aspects centraux de notre application :

- La gestion de l'environnement
- La gestion de la flotte de drones
- La gestion de la cartographie
- La génération de logs

Une vue générale de la solution apportée pour la gestion de ces trois aspects ainsi que les changements et compromis apportés avec la présente solution sont élaborés dans la section suite.

2.2.1 Gestion de l'environnement

La logiciel développé permet la gestion d'une flotte ayant un nombre arbitraire de drones (bien que le système ne soit pas optimisé pour la gestion de grandes flottes). De plus, le logiciel permet le passage à

un environnement de simulation et un environnement de drones matériels réels. Avant le début de toute mission d'exploration, l'utilisateur fournit le nombre de drones qu'il souhaite utiliser dans sa mission ainsi que le mode de fonctionnement, simulation ou réel, qu'il souhaite utiliser.

Afin de démarrer une mission d'exploration en simulation, une image Docker ayant accès à l'interface graphique de la machine hôte. Dans la situation proposée, la simulation est roulée à l'intérieur d'un conteneur docker déployée par x11docker. Le conteneur de la simulation (simulation-argos) roule en arrière plan tant et aussi longtemps que l'application est déployée. Le serveur utilise un script bash qui envoie une commande au conteneur afin de lui indiquer de supprimer la dernière simulation active et d'en démarrer une nouvelle. Lorsque la simulation est déclenchée par le serveur, les drones de la simulation chercheront à se connecter directement à une interface dédiée sur le serveur par le moyen d'un socket TCP.

Afin de démarrer une mission d'exploration en utilisant des drones réels, l'adresse de communication Crazyradio PA de chaque drone est associée à une interface sur le serveur. L'interface, dans le cas d'un environnement réel gère l'envoi et la réception de paquets via le module de télécommunication Bitcraze.

Dans le cas du passage d'un environnement de simulation à un environnement réel, ou vis-versa, l'ensemble des interfaces de communication (Bitcraze ou socket) sont fermées et ensuite supprimées afin de laisser place à de nouvelles interfaces liées à de nouveaux drones.

2.2.2 Gestion des drones

La gestion de la flotte de drones est également une des responsabilités principales de l'application. Cette responsabilité se fait en deux temps : d'une part, la réception d'information concernant l'état des drones provenant de la flotte ainsi que l'envoi de commandes de contrôle à la flotte de drones.

La gestion de la réception de l'état de la flotte de drone a été implémentée exactement tel qu'elle avait été planifiée lors des itérations précédentes. C'est-à-dire que chaque interface de communication avec les drones (une interface par drone, que les drones soient réels ou en simulation) reçoivent constamment et asynchronement un ensemble d'information concernant les drones. Ainsi, à chaque réception, une liste dite "singleton" des drones interfacés se met à jour. Ensuite à une fréquence de 2 Hz, la station au sol diffuse l'ensemble de la liste des drones à l'ensemble des clients connectés.

Pour ce qui est de l'envoi de commandes, le serveur est implémenté sur une approche orientée événement où, à la réception de certaines requêtes provenant du client, le serveur transfère un paquet aux drones spécifiés dans la requête afin d'indiquer au drone de changer d'état. L'envoi de commande peut être spécifique à un

drone individuel où diffusé à l'ensemble des drones.

2.2.3 Gestion de la génération de carte

Le fonctionnement de la génération de la carte est implémenté en adoptant une approche naïve génération utilisant une approche vectorielle consistant à additionner la position initiale d'un drone, son déplacement le vecteur des observations de ses capteurs afin de déterminer la position d'un point dans l'environnement.

Cette approche est susceptible de diminuer en précision en raison de l'incertitude sur la position initiale, la position estimée ainsi que l'orientation.

Pour la disposition des anciennes cartes, voici le diagramme de classe de cette fonctionnalité.

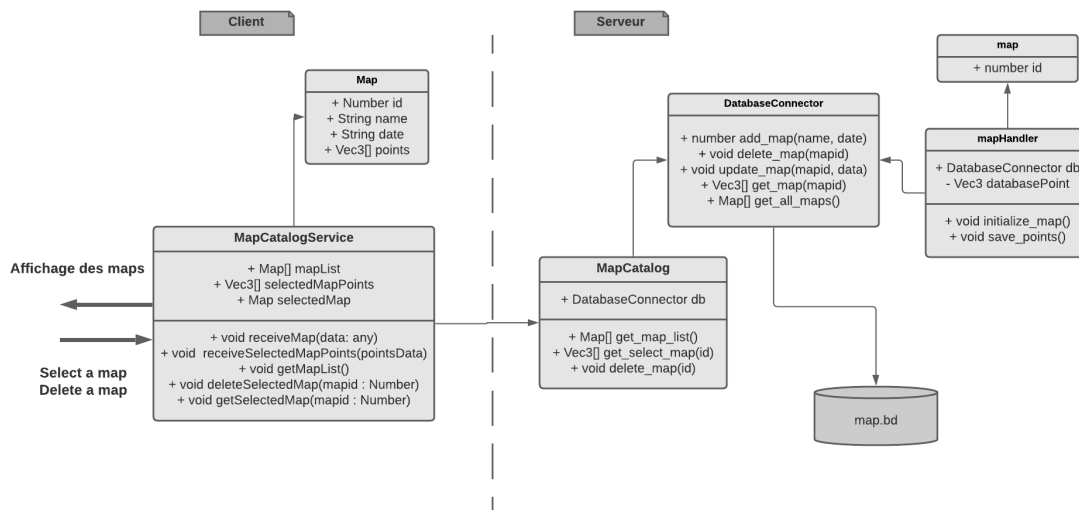


FIGURE 1 – Diagramme de classe de map catalog

Dans notre base de donnée, nous avons créer deux tables. La première est une liste de map avec un id unique, un nom et sa date de création. La deuxième table est une liste de points (x,y,z) avec un id de map qui lui est associée.

Lorsque nous créons une nouvelle map, nous appelons la fonction `initialize_map()`. Celle-ci appelle la fonction `add_map(name, date)` qui crée une nouvelle map et qui renvoie le id unique de la map. Avec ce id, on peut maintenant créer la map. Le `mapHandler` a une liste de points `databasePoint` qui enregistre les points dans la base de donnée lorsqu'il en a 20 nouveau avec la fonction `save_points()`.

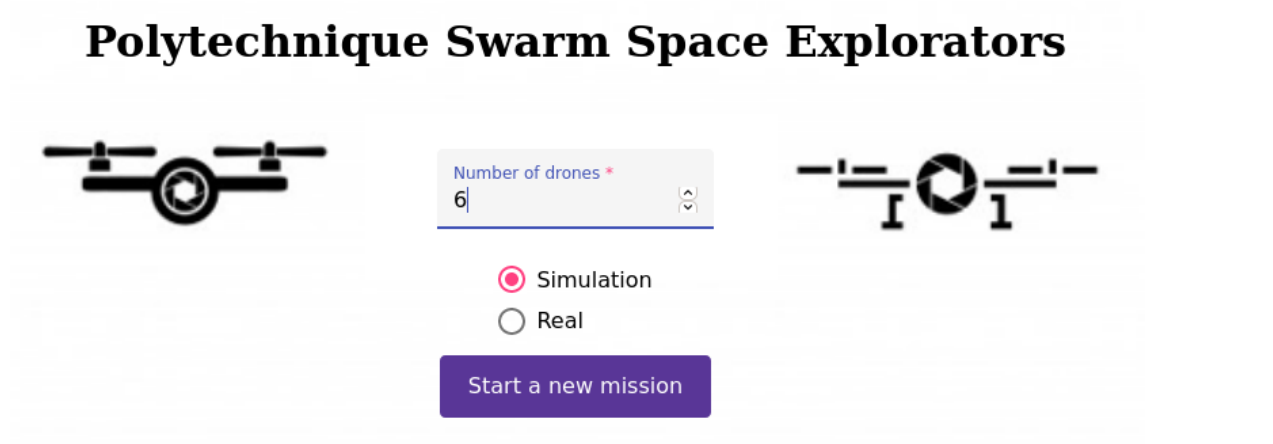
Pour communiquer du côté client, nous avons créer une classe qui s'appelle `MapCatalog`. Celle-ci permet d'accéder à la liste des cartes, à obtenir la liste des points d'une seule carte ou de supprimer une carte sur la base de donnée.

2.2.4 Génération de logs

Enfin, la génération de logs se produit tout au long de l'utilisation du serveur. Les logs générés sont répertoriés dans un dossier nommé "log" se retrouvant à la ra

2.3 L'interface de contrôle (Q4.6)

Voici sur la page initiale, l'initiation d'une mission.



The screenshot shows a web interface titled "Polytechnique Swarm Space Explorators". It features two drone icons: a top-down view on the left and a side profile on the right. In the center, there is a form with a label "Number of drones *" and a text input field containing the number "6". Below this, there are two radio buttons: "Simulation" (which is selected) and "Real". At the bottom of the form is a purple button labeled "Start a new mission".

FIGURE 2 – Initialisation d'une mission

Comme on peut voir, nous pouvons initialiser le nombre de drone que nous voulons. Par la suite, il est important de mentionner si c'est une nouvelle simulation ou une vraie mission. Dans le cas où ce n'est pas mentionné, l'option d'une vraie mission va être sélectionnée. Lorsque l'option de la vraie mission est enclenché, une modale va apparaître et va vous demander de rentrer la position de tes drones avec son adresse. Ici, on peut rentrer le nombre de drone qu'on veut, mais nous l'avons testé seulement avec deux drones. Lorsque l'option simulation est enclenché, notre client envoie une commande à notre serveur qui lance un script pour initialiser la simulation.

Par la suite, nous allons regarder la page de contrôle des drones.

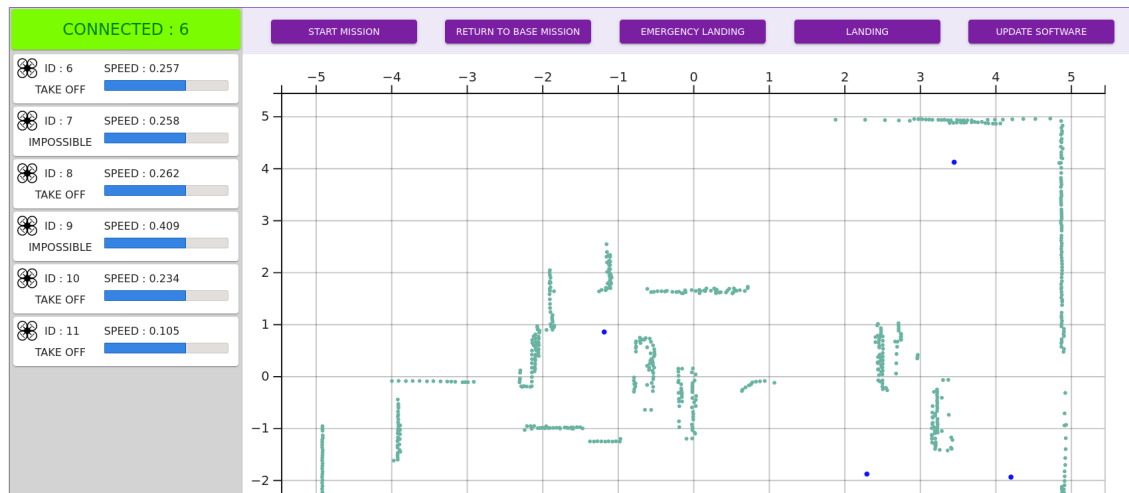


FIGURE 3 – Interface des drones et de la carte

Dans notre liste de drone à gauche, nous pouvons voir que nous avons créer une nouvelle mission avec six drones différents. La mission est débuté et les drones donnent des points qui sont générés par leur capteur. En effet, chaque nouveau paquet envoyé par les drones sont traité et par la suite, les nouveaux points sont mis dans une queue qui se gère par FIFO. Avant de les envoyer au client, à on prend les points un par un et on les arrondis. Ceci permet de voir s'il existe déjà le même point. Si oui, il va être rejeté et ne va donc jamais être envoyé au client.

Par la suite, si nous regardons les boutons en haut de la carte, on peut voir qu'il s'agit d'action sur la mission en cours. Nous avons le bouton START MISSION qui commence une nouvelle mission avec tous les drones. On fait cela en envoyant un socket au serveur avec un id de -2. Ceci va être effectué pour tous ces boutons. Ensuite, nous avons RETURN TO BASE MISSION qui permet au drone d'effectuer leur algorithme de retour à la base et donc de retour à la station au sol. Cette fonctionnalité est enclenché lorsque la batterie est à moins de 30%. Après, nous avons EMERGENCY LANDING qui arrête le moteur des drones et qui les fait atterrir brusquement. Pour LANDING, les drones vont atterrir doucement où ils se trouvent. Puis, UPDATE SOFTWARE permet d'ouvrir une modale où on peut changer le code et mettre à jour nos drones.

Voici la modale de notre mise à jour.



FIGURE 4 – Mise à jour du système

Le composant est affiché dans une mat-dialog quand on click sur le bouton "update software" du UI.

Il y a 2 façons de mettre à jour le code : en éditant le code existant (par défaut) ou bien en téléversant des fichiers (en utilisant le mat-toggle dédié à ça). Le composant, à travers son service, demande au serveur les fichiers du code embarqué. Le contenu de ces derniers est affiché dans un text area. Après modification du text area, le fichier modifié est mis à jour sur le client. L'envoi se fait quand le bouton update est enclenché.

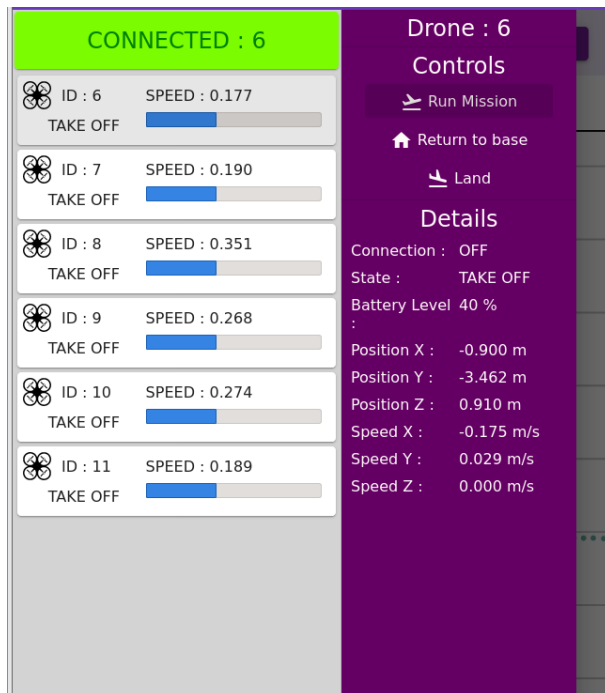


FIGURE 5 – Interface d'un drone sélectionné

Dans la page de contrôle, nous pouvons sélectionner un drone dans la liste des drones et le contrôler individuellement. En effet, une fenêtre bascule lorsque nous cliquons sur un des drones. Cette fenêtre contient des informations sur le drone comme sa position, sa vitesse, son niveau de batterie, son état et sa connexion. Nous avons quelques boutons d'action qui permettent de le contrôler.

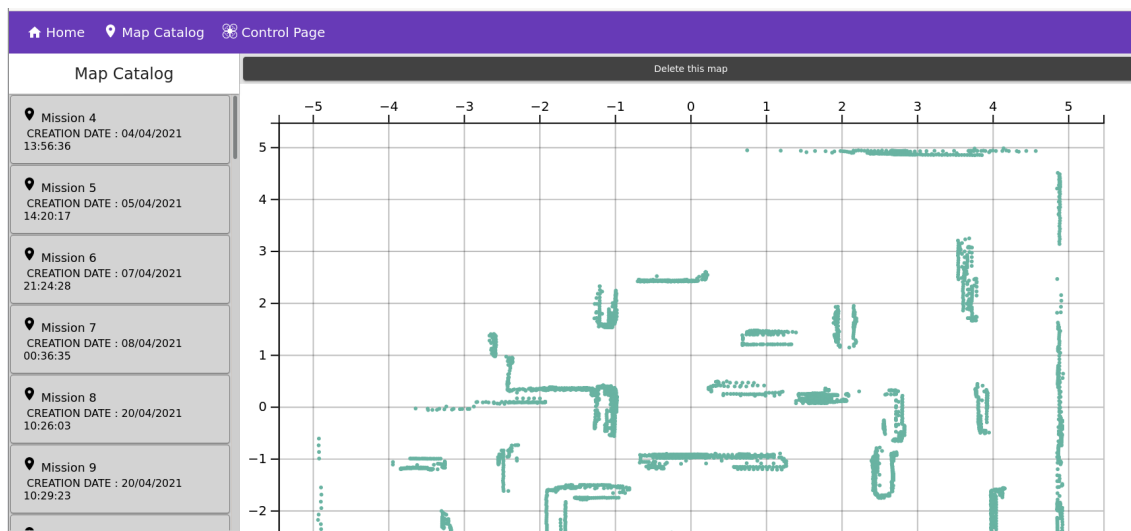


FIGURE 6 – Interface du Map Catalog

Par la suite, dans le map catalog, nous avons la liste des maps disponibles dans notre base de donnée. Ceux-ci

ont chacun un id unique et une date de leur création. Lorsque nous sélectionnons une map, ces points apparaissent dans la map à côté. Avec le bouton delete, il est possible de supprimer la map de la base de donnée.

2.4 Fonctionnement général (Q5.4)

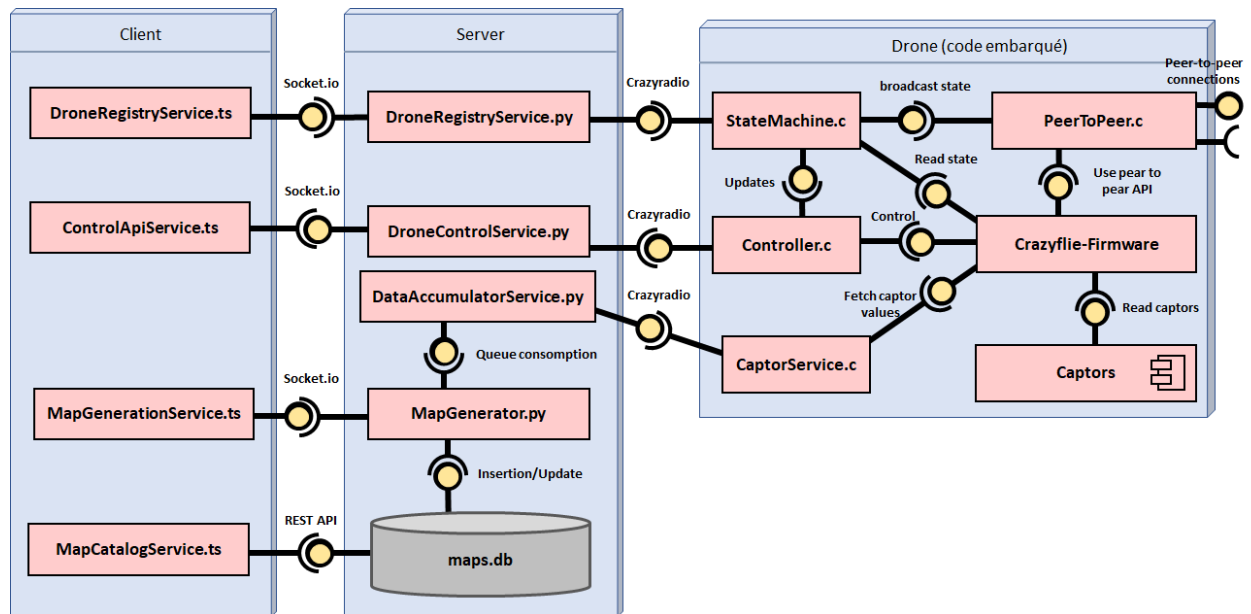


FIGURE 7 – Schéma générique du diagramme de composants de la solution proposée

Pour lancer docker, il faut lancer le script start-alfred.sh à la base du projet.

Lorsque notre docker est lancé il est important de positionner les robots avec soin. La qualité de la carte ainsi que la gestion des collisions en dépendent. Pour cela il faut comprendre quel système d'axe est utilisé dans les drones afin pouvoir inscrire les bonnes valeurs. Il sera décrit dans l'exemple qui suit.

Afin de démarrer les drones, l'utilisateur doit se rendre sur l'interface utilisateur. Il inscrit le nombre de drone qu'il veut faire voler, choisi le mode "Real" et clique sur "Control Map". Une modale s'ouvre alors pour inscrire les positions initiales des robots. Voici un exemple qui permettra de mieux comprendre les coordonnées qu'il faut inscrire. Dans cet exemple, nous avons deux drones (qui possèdent les adresses E7E7E7E7E7 et E7E7E7E7E5). Dans la réalité, ils sont positionnés comme suit.

E7E7E7E7E5

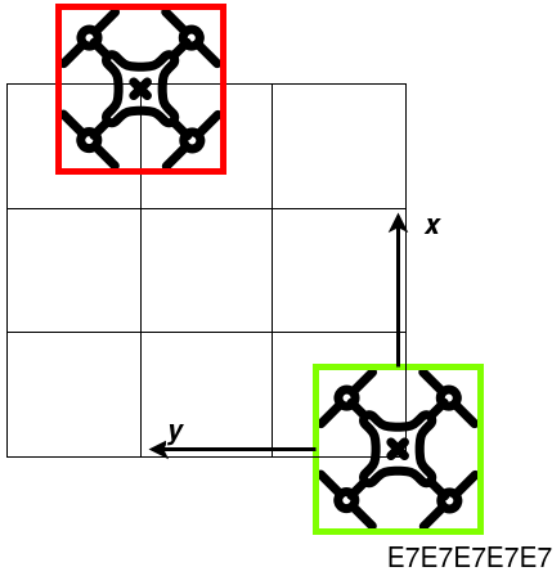


FIGURE 8 – Positionnement des drones

Please enter relative positions (in m)
All drones should have same orientation

Please enter relative positions (in m)
All drones should have same orientation

1 Drone 1

Address *
E7E7E7E7E7

Position in x *
0

Position in y *
0

Validate

Next

2 Drone 2

Drone 1

2 Drone 2

Address *
E7E7E7E7E5

Position in x *
3

Position in y *
2

Validate

Next

FIGURE 9 – Position initiale drone 2

Chaque case correspond à 1m. Si nous prenons comme référence le drone vert avec les coordonnées (0,0) nous devront alors entrer les valeurs suivantes dans l'interface web.

Attention le point (0,0) est considéré dans notre application comme la position de la base. Ainsi le drone vert devra être situé très proche de la base afin que les 2 drones puissent effectuer le retour vers la base. Aussi, comme indiqué dans la page web il est primordial que les drones soient orientés dans la même direction.

3 Résultats des tests de fonctionnement du système complet (Q2.4)

Notre système a été testé module par module afin de vérifier son bon fonctionnement.

Sur le client, nous avons utilisé les frameworks Jasmine et Karma qui sont des outils déjà intégrés aux projets angular. Avec ces outils intégrés, il a été facile de mettre en place un environ de tests adéquat pour le développement des différents composants de notre client. Ce client est essentiellement composé de classes, de composants et de services. Les services qui contiennent la logique des opérations effectuées sur l'interface utilisateur sont systématiquement testés. Pour lancer les tests du client, il faut utiliser la commande ***npm run test*** dans le dossier ***client***. Avec la figure suivante, nous pouvons voir les résultats de nos tests qui prouvent le bon fonctionnement du client.

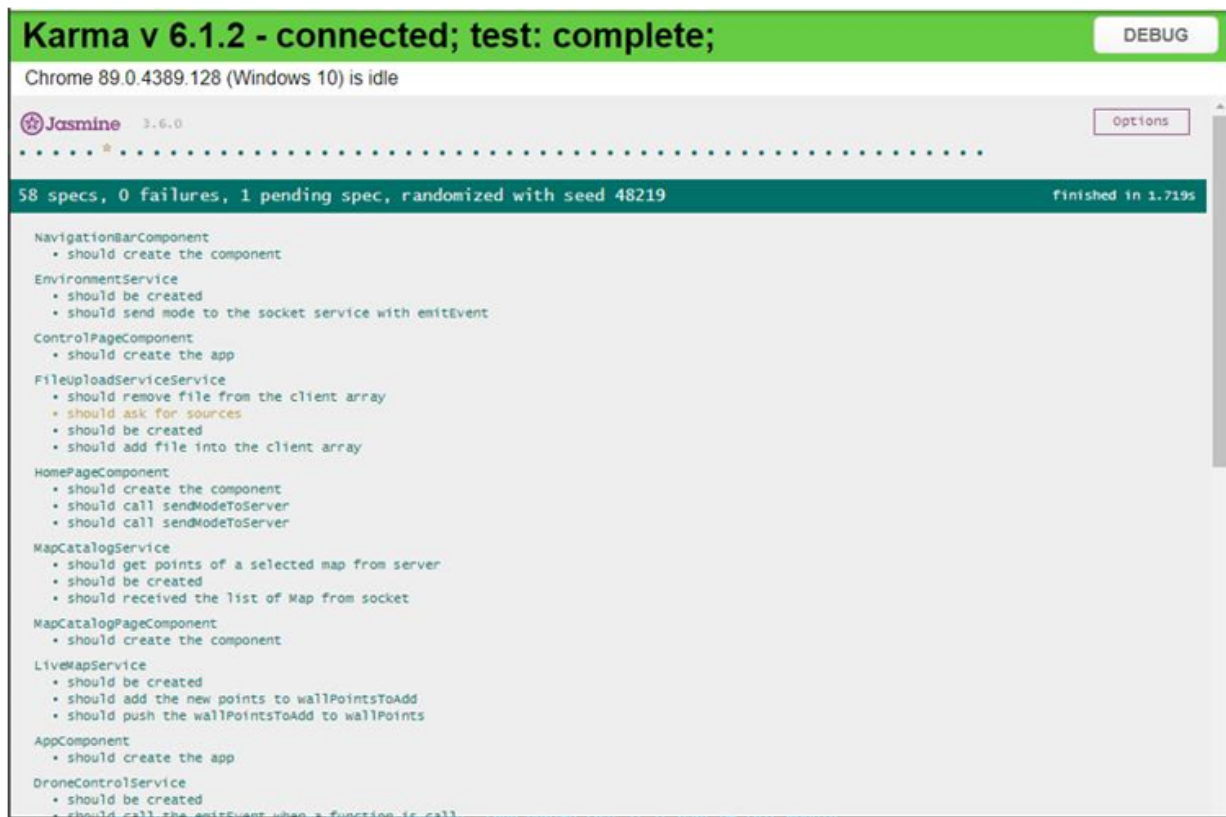
Pour les tests du serveur, nous avons préféré l'outil unittest de python. C'était plus simple de l'intégrer à notre projet parce qu'il fait partie de la bibliothèque standard de python et il est facile d'utilisation. Nous utilisons la commande ***python -m unittest discover test*** pour lancer tous les tests du serveur dans le dossier ***test*** du serveur. Le nombre de tests effectués sur le serveur est de 43 et les résultats de ces tests sont visibles dans la figure suivante.

Pour le système embarqué et la simulation argos, les tests ont été fait avec l'outil Googletest et son intégration à notre projet a été relativement facile. Pour lancer les tests, il faut d'abord lancer le conteneur docker et ensuite y accéder. À partir du dossier racine (root) du conteneur, on va dans le dossier simulation/build et lancer la commande ***./launch_all_tests.sh***. Nous avons implémenter 30 tests pour vérifier le comportement de nos fonctions avec le système embarqué. la figure suivante donne un exemple de nos résultats de test.

Dans notre projet, nous avons mis l'accent sur les tests unitaires en particulier pour les fonctionnalités qui contiennent de la logique. Ainsi, nous avons pu éviter de faire de la regression à chaque fois que nous faisons de nouveaux ajouts de codes à notre projet mais et surtout nous avons pu vérifier le bon fonctionnement de nos algorithmes et ensuite les améliorer. C'est le cas spécifiquement des algorithmes d'exploration, de retour à la base et d'évitement de collision qui contiennent de nombreuses logiques mathématiques et qui, avec les tests ont pu être améliorés.

Nous n'avons pas malheureusement testés toutes les classes comme nous l'aurions voulu.

FIGURE 10 – Résultat des tests effectués sur le client



4 Déroulement du projet (Q2.5)

Du côté organisation, nous avons respecté nos engagements du départ. Ceux-ci étaient : un scrum le mercredi soir, une rencontre formelle le vendredi soir et une rencontre le dimanche pour organiser la semaine suivante. Nous n'avons pas enlevé une rencontre jusqu'à la remise. Lorsqu'il était important de se voir plus souvent comme prêt de la date d'une remise, nous essayons de trouver une heure où le monde était disponible pour ajouter une rencontre. Cependant, dans le contrat, nous avons indiqué que nous allons devancer les dates des remises de deux jours pour s'assurer en cas d'imprévu que tout soit bien terminé. Pour la deuxième remise, CDR, nous n'avons pas respecté cet engagement ce qui a fait que notre vieille date de remise a été très mouvementée. Nous avons appris de cette erreur et pour la remise finale, nous nous sommes pris de l'avance en se concentrant particulièrement sur les tâches qui étaient obligatoires.

Nous avons essayé de respecter le temps mis dans notre appel d'offre, mais nous avons vite été obligés d'aller utiliser nos heures tampons pour certaines fonctionnalités. Il est dur de voir si nous avons dépassé le

FIGURE 11 – Résultat des tests effectués sur le client

```
(server-VwYpr4EB) julio@julio-VirtualBox:~/Documents/Projet3/alfred/server/src$ python -m unittest discover test
.....S.....
-----
Ran 43 tests in 17.654s

OK (skipped=1)
```

FIGURE 12 – Résultat des tests effectués sur le client

```
[ OK ] sensorstest.ReturningSide_one_free_side_test_back_right (0 ms)
[-----] 22 tests from sensorstest (14 ms total)

[-----] Global test environment tear-down
[=====] 22 tests from 1 test suite ran. (18 ms total)
[ PASSED ] 22 tests.
root@julio-VirtualBox:~/simulation/build#
```

plafond de 630h parce que plusieurs heures informelles n'ont pas été enregistrées dans notre git lab.

Pour implémenter notre projet, nous nous sommes vraiment fier sur l'appel d'offre pour l'architecture. Nous avons mis beaucoup de détail dans l'appel d'offre donc il était facile pour nous de se fier à celle-ci et de la prendre comme base pour notre projet. Tous les diagrammes étaient très précis et il nous ont beaucoup aidé.

Pour l'échéancier, nous l'avons presque suivi à la lettre. Nous avons ajouté quelques tâches supplémentaire imprévues, mais en général nous avons suivi le calendrier pour compléter nos tâches. Les tâches s'accumulaient quelque fois, mais à chaque semaine, nous ajoutions nos tâches de la semaine du calendrier dans le git lab.

5 Travaux futurs et recommandations (Q3.5)

Le projet est fonctionnelle mais peut être amélioré sur plusieurs points :

La carte générée utilise actuellement un calcul vectoriel simple qui ne prend pas en compte les erreurs de la précision du matériel. Une cartographie par résolution de problème d'optimisation serait une amélioration possible. Un deuxième point à améliorer serait la position des drones. Actuellement calculer en utilisant les valeurs données par l'API des drones, la position est vraiment approximative. Un algorithme de SLAM de cartographie et localisation simultanée serait une bonne addition à notre travail.

Nous recommandons des ajouts de fonctionnalité et de sécurité, par exemple :

Une fonctionnalité dont nous avons discuté en équipe serait de pouvoir demander aux drones de se déplacer à

un point spécifique sur la carte. Cette fonctionnalité serait une valeur ajoutée au projet permettant l'ample contrôle des drones au besoin. Notre architecture permet cela, les étapes à suivre sont triviales :

- 1- Trouver les coordonnées du "mouseclick" sur la map.
- 2- Envoyer ces coordonnées au drone sélectionné.
- 3- Sur le drone : calculer le vecteur directeur nécessaire et le suivre jusqu'à atteindre le point.

6 Apprentissage continu (Q12)

6.1 Farid El Fakhry

Durant le déroulement du projet, j'ai dû faire face à plusieurs obstacles. L'obstacle le plus important était l'intégration en continu d'un projet complexe. Notre équipe est formée de 5 membres, qui travaillent souvent sur des parties différentes, ces parties sont souvent utilisées l'une dans l'autre. Nous avons pu gérer partiellement, issues...etc sur gitlab. Avec notre organisation, une énorme quantité de temps a été allouée sur la mise en commun du code. Ceci cause d'autres problèmes comme la gestion du temps, durant le temps où on communique nous n'avancions pas dans le développement et cela crée du retard. Cela aurait pu être amélioré en utilisant plus d'outils d'intégration comme Jenkins et en suivant une approche de réunion plus simple.

6.2 Alexandre Morinvil

Cette expérience m'a permis de développer mes aptitudes de gestion du travail en équipe dans une équipe constituée de membres ayant des aptitudes techniques et relationnelles différentes. Je considère que cette expérience de projet en situation de COVID où l'ensemble de la charge de travail a été réalisé à distance aura surtout été bénéfique en ce qui a trait à aptitudes transversales. Je pense que j'aurais dû me montrer plus présent pour aider davantage mon équipe. Il sera préférable pour mes prochains projets intégrateurs de ne pas prendre autant de cours et de responsabilités que j'ai pris durant cette session afin de pouvoir mieux me consacrer à mon projet.

Pour ce qui est de mes apprentissages techniques, je considère que ce projet a été une belle opportunité de développer mes aptitudes en développement web et en développement en python.

6.3 Julio Dandjinou

Notre projet nous a permis d'apprendre de nouvelles technologies comme le langage python, l'utilisation de la simulation argos et l'implémentation des comportements des drones dans argos et dans le bitcraze. Cependant, j'ai eu des difficultés parfois à comprendre certaines choses dans les codes avec le bitcraze et pour pallier à

cela, j'en discute avec les collègues. Il y a aussi le fait que je n'ai pas pu vite partir le projet à cause de mis en place du matériel qui est essentiellement un environnement linux. Au cours du projet, nous nous sommes entraides sur les problèmes que nous avons rencontrés et la bonne communication entre tous membres a pu aidé. L'organis

6.4 Alexandre Talens

Il est était très intéressant pour moi de travailler sur ce projet. Il m'a permis notamment de mettre en évidence certaines lacune technique et d'organisation. En effet, notamment au début du projet, il a été difficile pour moi de travailler en Python. Je suis habitué à travailler en Java notamment lors de mes stages. Les concepts sont alors pour certains très différents. L'aide des mes coéquipiers a été alors très précieuse. Aussi, en terme d'organisation j'ai appris à mieux communiquer avec mes coéquipier pour comprendre leur problèmes et les aider. Cependant, les communications n'était toujours parfaite. Des incompréhensions sont survenues sur les taches qui devaient être réalisées. C'est un point sur lequel j'ai besoin de m'améliorer. Je pense qu'une utilisation encore plus rigoureuse de Gitlab pourrait aider à améliorer ce point.

6.5 Émilie Vaudrin

Initialement, je n'avais jamais travaillé avec python et avec des interfaces comme docker et bitcraze. J'ai eu de la difficultés au départ à m'adapter avec ceux-ci. Pour python, je me suis mis à écrire le serveur et lorsque j'avais des hésitations j'allais voir mes coéquipiers ou je chechais sur les internets. Je me suis facilement adapté avec ce langage, car il était plutôt simple. Puisque je n'avais pas beaucoup d'expérience avec les drones et ARGoS, je me suis mis sur le côté serveur et client ce qui a fait que je n'ai jamais vraiment pu toucher à ces sections. Lorsqu'il y avait des problèmes dans ARGoS, je ne pouvais pas souvent aider. J'ai essayé de regarder le code poussé régulièrement pour me garder à jour. Dans le futur, il serait bon de ne pas travailler séparément et d'expliquer nos parties pour garder tout le monde a jour.

7 Conclusion (Q3.6)

En bref, Notre solution a pu prouver le concept d'essaim de drones explateurs pour l'exploration spatiales tel que requis. Nous avons déjà repartie le travail en plusieurs taches et sous tache durant notre reponse a l'appelle d'offre. Nous avons attribue un temps et une importance a ces dernieres ense basant sur notre architecture originale. Notre architecture generale a tres peu variée. Des mineurs changement ont été apporte a notre solution et a notre approche. En generale nos hypothese de conception etais correcte bien qu'il existe de la place pour des ameliorations et bien sur pour des travaux futurs. Nous esperons pouvoir collaboré encore plus avec l'agence spatiale de polytechnique pour livrer plus de solutions de qualite !