POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

Département de génie informatique et génie logiciel

INF3995

Projet de conception d'un système informatique

Proposition répondant à l'appel d'offres
no. H2021-INF3995 du département GIGL.

Conception d'un système aérien minimal pour exploration

Équipe N°***3***

Mazigh Ouanes

Tarik Agday

Nabil Dabouz

Mohamed Yassir El Aoufir

Paul Clas

Remis le 15 février 2021

Table des matières

1. Vue d'ensemble du projet

1.1 But du projet, porté et objectif

Ce présent document agit à titre de réponse à l'appel d'offres de l'Agence Spatial: Système aérien minimal pour exploration, dans le cadre du cours INF3995. L'objectif de ce projet est de produire une preuve de concept d'exploration avec des drones équipés de capteurs. Nous chercherons à démontrer qu'il est possible d'établir une communication et de contrôler ces drones à distance en plus d'explorer un terrain inconnu. Comparativement à l'astromobile Curiosity de la Nasa, nous tenterons de prouver que plusieurs petits drones peuvent parcourir une plus grande distance en parallèle que ce type d'engins. Idéalement, ces petits drones couvriraient un terrain afin de déterminer les points d'intérêts **(3)** où l'on pourrait par la suite déployer un robot plus lent et plus robuste, mais mieux équipé pour répondre à certains besoins spécifiques.

La solution que nous proposons se divise en 5 composantes que nous détaillerons dans la section 3 de ce document. Globalement, nous observons une station au sol, 2 drones (incluant une radio de communication), une interface client et un système embarqué permettant de programmer les drones. La portée de ce projet se veut avant tout exclusive à cette réponse à l'appel d'offres de l'Agence Spatiale.

Voici donc, les éléments que nous planifions livrer à la fin de ce projet:

- Service web: permettant l'envoi d'instructions de la part de l'utilisateur aux drones. Ce service affiche aussi certaines informations récupérées par ces derniers pendant leurs missions d'exploration.
- Réseau de communication: assurant un lien entre la station au sol et les drones. Le réseau est un canal de communication de 2.4GHz.
- 2 Drones Bitcraze Crazyflie 2.1: fournis par l'agence spatiale, ils devront communiquer avec la station au sol à l'aide du réseau établi. Ils répondront aux instructions (vol, atterrissage, mise à jour système et retour à la base) indiquées par l'utilisateur.
- Station au sol: Interface web assurant une communication avec les drones à l'aide de leurs antennes du réseau établi.
- Base donnée NoSQL: stockage des données récupérées par les drones lors des missions d'explorations. Peut inclure: positions des drones, obstacles détectés, niveau de batterie, etc.
- Système embarqué: Partie logiciel fournie aux microcontrôleurs des drones assurant l'implémentation de notre code.

1.2 Hypothèse et contraintes

La solution que nous proposons englobe cinq composantes ayant chacune ces spécifications et des contraintes différentes. Voici donc les hypothèses que nous avons faites pour chacune de ses composantes. Celles-ci correspondent non seulement au point de vue technique, mais aussi aux éléments externes pouvant affecter notre système. Le système d'exploitation suggéré pour développer notre solution est Linux (Ubuntu ou Fedora) considérant sa compatibilité avec l'ensemble des composantes physiques et logiciels de notre solution.

De plus, il est attendu d'utiliser *Docker* pour l'ensemble de notre projet. En effet, cet outil a été conçu de sorte à faciliter la création et le déploiement d'application en utilisant des conteneurs **(1).** Cette contrainte nous permettra d'assurer un fonctionnement de nos sous-ensembles peu importe l'environnement informatique où ils seront utilisés.

Service web

Ce service sert de point de contrôle à l'utilisateur. Aucune contrainte n'ayant été mentionnée par l'Agence Spatiale quant à sa programmation, nous faisons l'hypothèse qu'il serait préférable d'utiliser ReactJS pour sa compatibilité avec nos besoins, mais aussi pour sa productivité et sa facilité de maintenance **(2)**. Toutefois, ce composant doit offrir les options suivantes à l'utilisateur:

- Décoller les drones / Départ de mission
- Atterrir les drones / Fin de mission
- Mettre à jour les drones
- Retour à la base / station au sol

Les drones ne sont pas autorisés à décoller lorsque leur batterie affiche un niveau de 30% et moins. De plus, nous avons une contrainte de compatibilité pour ce service web. En effet, il doit être accessible sur PC, tablette et téléphone intelligent, d'où l'utilisation de ReactJS **(2)**. De plus, ce service doit assurer une mise à jour des éléments suivants à une fréquence de 1Hz: nombre de drones, états des drones, vitesse courante des drones, niveau de batterie des drones et la carte générés durant l'exploration (*R.F.5 du document de requis*).

Réseau de communication

La station au sol utilisera une clé USB Bitcraze Crazyradio *PA* **(8,9)** afin de communiquer avec les drones. Les signaux envoyés seront récupérés par les antennes de ces derniers. La composante logicielle de ce canal de communication n'étant pas fournie, nous devrons concevoir un réseau sur la station au sol permettant de synchroniser ce canal. Nous sommes d'ailleurs contraints au fait que cette composante soit un réseau de 2.4GHz.

Drones

Les drones sont fournis par l'Agence Spatiale et aucune modification physique sur ceux-ci n'est attendue. Il est donc compris que l'utilisation des drones se résume en ces points:

- Réception et application des commandes de l'opérateur (commandes de navigation + allumage de D.E.L.)
- Envoi des informations de détection perçues par le laser intégré (mapping)
- Mise à jour logiciel
- Communication intra drone
- Exploration autonome des drones
-

Nous serons toutefois responsables de la partie embarquée qui sera programmée sur ces drones afin de répondre à la demande d'exploration du terrain. De plus, il est supposé par *l'Agence Spatiale* qu'un des drones de l'essaim est toujours en communication avec la station au sol.

Station au sol

La station au sol est représentée par un laptop ou un PC responsable d'assurer la communication entre les drones, à l'aide de l'antenne radio (USB), et l'interface client en plus, d'afficher le service web. Elle est d'ailleurs munie d'un côté serveur permettant d'intégrer et de faire les manipulations nécessaires aux données fournies par les drones afin que l'on puisse les interpréter correctement (e.g., l'optimisation des données de cartographie afin d'illustrer la pièce parcourue).

Systèmes embarqués

La partie embarquée du projet englobe la partie logicielle de notre solution intégrée aux microcontrôleurs des drones. Le langage de programmation utilisé pour cette partie sera C / C++. D'ailleurs, nous utiliserons l'environnement de simulation d'ARGoS avant de mettre la main sur les drones. Ce simulateur nous permettra aussi de faire des tests.

Base de données

Cette composante ne fait état d'aucune contrainte particulière. Toutefois, nous faisons l'hypothèse que son intégration au sein de notre projet devra répondre à un besoin de rapidité au niveau de récupération des données considérant la nécessité d'un certain niveau de temps réel. La base de données servira à contenir tous les logs et positions des drones en plus des commandes effectués par l'opérateur.

Élément externe

Des éléments externes peuvent modifier les contraintes techniques qui sont évoquées ci-dessus. Par exemple, des imprévus en cours de route qui peuvent entraîner des changements dans le projet. C'est le cas particulier de la pandémie, qui rend difficile la tenue des rencontres en présentiel. Tout se fait principalement à distance. De plus, le client pourrait apporter des changements dans les requis ou faire des ajustements de dernière minute auxquels l'équipe devra s'adapter. Ainsi, le mot clef́ à retenir est flexibilité, nous sommes après tout dans un processus de développement agile.

1.3 Biens livrables du projet

Ce projet se fera tout au long du semestre d'hiver 2021 prévu par le calendrier de Polytechnique Montréal. Nous aurons donc approximativement 4 mois pour le développement de cette preuve de concept. L'agence Spatiale exige trois livrables tout au long du projet. Ceux-ci serviront de référence pour documenter l'avancement du projet. Nous avons d'ailleurs jugé pertinent l'ajout d'un quatrième artefact, le Qualification Review (QR) dans le but de valider l'expérience utilisateur de notre produit final avant la dernière remise. Ce livrable s'inscrit dans notre démarche de s'inspirer de meilleures pratiques liées au développement agile qui émane des normes ECSS-E-HB-40-01A et ISO 90003 **(18,19).**

Preliminary Design Review PDR et Prototype

Le Preliminary Design Review ou PDR est le premier livrable que nous aurons à produire. Il est entendu de le compléter pour le 15 février 2021 et il sera accompagné d'un modeste prototype de notre système embarqué qui sera simulé sur ARGoS afin de présenter un parcours aléatoire parcouru par deux drones. Le PDR est un document présentant notre réponse à l'appel d'offres de l'Agence Spatiale. Il inclut entre autres notre plan de gestion de projet et permet à ses lecteurs d'avoir une idée globale de la planification et du développement du produit final. D'autre part, voici de manière exhaustive ce qui est attendu du prototype accompagnant le PDR:

- Simulation ARGoS avec deux drones suivant un parcours aléatoire
- Affichage du niveau de batterie des drones
- Interface web permettant l'allumage d'une lumière D.E.L. des drones

Critical Design Review CDR (8 mars 2021)

Ce deuxième livrable fait état de la conception détaillée du système incluant toutes ses fonctionnalités. Il sera remis le 8 mars 2021 (la date peut changer).

- Simulation ARGoS avec 4 drones qui volent dans un environnement avec murs générés aléatoirement
- Les drones dans la simulation utilisent des capteurs de distance (comme le ranging deck) pour éviter les obstacles et décider leur parcours
- Serveur web interfacé avec la simulation ARGoS
- Commande « Take off » et « Return to base » implémentées
- Interface du système implémenté selon le requis R.F.5
- Code embarqué sur les drones qui envoie les mesures du ranging deck selon les requis R.F.5 et R.L.6
- Un prototype de visualisation de la carte générée par les robots

Qualification Review (QR) (entre le 8 et 29 mars 2021)

Le livrable Qualification Review ou QR est le troisième livrable qui a pour objectif de tester l'utilisation de notre solution clef en main par des utilisateurs. Ce livrable consiste à réaliser cinq entrevues d'utilisateurs pour tester notre interface en réalisant différents types d'action avec le navigateur web et le simulateur ARGoS (et/ou les drones si les restrictions gouvernementales liées à la COVID19 nous le permettent):

1. Connexion au navigateur web pour opérer les drones et test de l'affichage:
    1. Pouvez-vous indiquer le nombre de drones connectés ?
    2. Pouvez-vous indiquer leur niveau de batterie ?

3. Pouvez-vous indiquer la vitesse d'un drone ?
4. Pouvez-vous indiquer les contrôles de navigation de l'essaim de drones ?
5. Naviguer vers une page de détail d'un drone.
6. Agrandissez l'affichage de la carte généré par un drone.

2. Lancement d'une mission avec des drones, dans une salle générée aléatoirement par ARGoS, puis retour automatique à la station des deux drones après que le seuil des batteries soit inférieur à 30%.
3. Lancement d'une mission avec des drones puis atterrissage d'urgence des drones.
    1. Lancement d'une mission interrompue avec des drones puis action pour le retour à la station au sol.
4. Consultation des logs des logs de la mission 3 et 3.1.

Les tests seront effectués le 29 mars avec des volontaires de l'Agence Spatiale. Les fichiers du système testé seront soumis à l'agence le 29 mars avec un guide d'utilisation. Un rapport des tests sera soumis avec le Readiness Review (RR) le 12 avril.

Readiness Review RR (12 avril 2021)

Ce dernier livrable contiendra toute la documentation développée au cours du projet, en plus des artefacts suivants :

- Un vidéo montrant le fonctionnement du système en simulation et avec 4 vrais robots. Ce vidéo comportera notamment:
    - Un guide vidéo d'utilisation de notre solution.
    - Le lancement d'une mission avec des drones et le retour automatique à la station.
    - Le lancement d'une mission avec des drones et le retour actionné manuellement du retour à la station.
- Tout script nécessaire pour lancer une simulation du système avec une seule commande sur Linux
- Le rapport des tests d'utilisation du 29 mars.

Acceptance Review AR (19 avril 2021)

Il s'agit de la présentation du projet et, si possible, sa démonstration en temps réel au professeur et étudiant du Département de génie informatique et génie logiciel.

2. Organisation du projet

2.1 Structure d'organisation

Tout au long du projet, nous adopterons une approche agile et itérative. Nous travaillerons principalement en mode décentralisé. Cette méthode de travail nous permet de mieux collaborer en équipe sachant que le projet requiert plusieurs connaissances et composantes techniques variées. Chaque membre de l'équipe se concentrera sur trois à quatre composantes principales. Néanmoins, la structure de travail décentralisé nous permettra de réajuster nos ressources si nécessaire. Nous travaillerons principalement en binôme. Ainsi, notre projet se divisera en plusieurs itérations ou sprints, de sorte qu'un produit soit livré à la fin de chacune de ces itérations.

Une itération ou sprint se compose des tâches suivantes :

- Revenir sur les imprévus du sprint passé
- Estimer le niveau d'effort nécessaire à la complétion de chacune des taches de l'itération actuelle
- Estimer la priorité de chacune des tâches de l'itération actuelle
- Répartir les différentes tâches entre les membres
- Mettre à jour le calendrier
- Réalisé au moins 3 réunions de 30 minutes par semaine
- Réaliser les tâches

Le produit final est lui-même décomposable en plusieurs tâches interagissant avec différents produits/systèmes. Chaque produit sera sous la responsabilité d'un producteur et d'au moins un développeur analyste.

2.1.1 Organisation équipe :

Voici quelques règles de travail en équipe que nous avions fixées.

- Diviser l'application en plusieurs produits/services différents. Les tâches respectives de ceux-ci seront réparties au sein de l'équipe
- Chaque personne est 'product owner' d'un produit sur lequel elle est aussi développeur-analyste au moins au produit dont il n'est pas product owner
- L'architecture de chaque produit est conçue/validé par toute l'équipe
- Le rôle de scrum master/chef de projet et de secrétaire n'est pas fixe (varie de sprint en sprint)

En ce qui a trait aux rôles de chaque membre de l'équipe, nous avons discuté nos forces et faiblesses, ainsi, nous avons pu déterminer des responsables pour chaque produit et service contenu dans les livrables. Nous avions déterminé 10 composantes principales dans ce projet que nous avions réparties entre nous. Nous serons tous impliqués dans le développement du code et collaborons étroitement pour bien coordonner notre progression du travail.

Composantes	Responsables:
Serveur Web	Tarik, Yassir
Interface Usager	Paul, Yassir
Pont Crazyflie/Web	Yassir, Tarik
Pont Argos/Web	Nabil, Paul
Simulation ARGoS	Nabil,Mazigh
CrazyFlie Firmware	Nabil, Mazigh
Navigation Autonome	Mazigh, Paul
Docker Compose	Mazigh, Yassir
Gestion du projet	Paul, Tarik

Tableau 1 Composantes du projet et leurs responsables

2.1.2 Organisation projet :

Durant le projet les tâches seront définies selon leur type, leurs priorités, le sous-produit auquel elles sont attachées (microservice) et l'effort nécessaire pour accomplir la tâche.

- Nous avons pour l'instant prévu trois niveaux de priorités pour nos tâches: bas, moyen et élevée. Celles-ci seront priorisées sur GitLab.
- On distingue les types de tâches suivants : Développement, Test, Bug, Refactor, Assurance Qualité, Gestion de projet, Conception, Déploiement, Planification.
- L'effort nécessaire pour accomplir une tâche sera évalué en nombre d'heures/personne nécessaires

Le respect des normes d'assurances qualité et le maintien de pratiques d'intégration continue seront obligatoires dès le début du projet. Toutes ses règles/conventions seront mises en pratique à l'aide des outils de gestion de projet et d'intégration continue fournis par GitLab.

2.2 Entente contractuelle

Afin de répondre à l'appel d'offres de manière efficace, nous avons choisi d'offrir une solution clef en main à l'Agence Spatial. La solution que nous proposerons aura donc un prix ferme et final. Ainsi, en tant que contractants du projet, avec notre solution clef en main, nous assurerons à l'Agence qu'il n'y aura pas d'autres frais fixes ou variables, autres que celui que nous aurons soumis. Grâce à notre expertise interne, nous sommes convaincus d'offrir une solution innovante et fonctionnelle à l'Agence tout en permettant à celle-ci de maximiser ses coûts d'investissement pour ce projet de type recherche et développement. Nous serons donc plus compétitifs. De plus, nous sommes convaincus de pouvoir fournir une solution de niveau de maturité technologique demandé (Niveau demandé: 4, Validation de principe reposant sur l'intégration d'applications et de concepts en vue de démontrer la viabilité) (3).

Notre expertise dans le domaine de la robotique et du génie logiciel nous permettra d'assurer à l'Agence Spatial un livrable final fonctionnel avec suivi minimal de la part de l'agence. Finalement, même dans le cas d'un éventuel changement pouvant occasionner une fluctuation des coûts de projet, nous pourrons rassurer l'Agence que celle-ci n'aura qu'un coût fixe à nous fournir lors du dernier livrable. Encore une fois, sommes convaincus que grâce à notre expertise le projet sera mené dans les délais et nous pourrons offrir une solution d'un produit démontrant une solution viable.

3. Solution proposée

Notre solution proposée aura pour objectif de répondre à 4 objectifs d'opération d'un système de navigation et d'exploration d'un essaim de drones:

- Navigation et exploration autonome d'un espace inconnu par un essaim de drones
- Simulation de mission
- Contrôle de la mission par un opérateur
- Portabilité et évolutivité du système

3.1 Architecture logicielle générale

Pour atteindre l'objectif du projet, trois architectures principales doivent être mises en œuvre: le contrôle de l'essaim via un opérateur d'une station au sol, la communication entre la radio PA/drone et drone/drone ainsi que le vol autonome. La station au sol va être implémentée avec la pile FReMP (Flask, ReactJS, MongoDB et Python) hébergée sur un serveur web compartimentalisé en utilisant Docker Compose. Docker sera également utilisé pour avoir un conteneur Flask pour gérer les requêtes Python provenant du Crazyflie à travers le CrazyFlie Radio PA, mais aussi les requêtes Python pour communiquer avec le conteneur du simulateur ARGOS. Une représentation schématique de l'architecture du système Docker est présente dans la section 3.3. **(1,16,17)**

L'API python développé par BitCraze pour le Crazyflie à tous les logiciels nécessaires au vol et au développement préinstallés. La logique de l'application va être placée sur une station de contrôle au sol qui se connecte au Crazyflie mais également, en partie, directement sur le Crazyflie. Le micrologiciel Crazyflie est écrit en C++/C et l'API officielle Crazyflie est écrite en Python. Pour ce projet, il a été décidé qu'une station de contrôle, utilisant l'API Python, devrait contrôler le Crazyflie 2.1 **(12).**

Le schéma suivant présente l'architecture globale du projet:

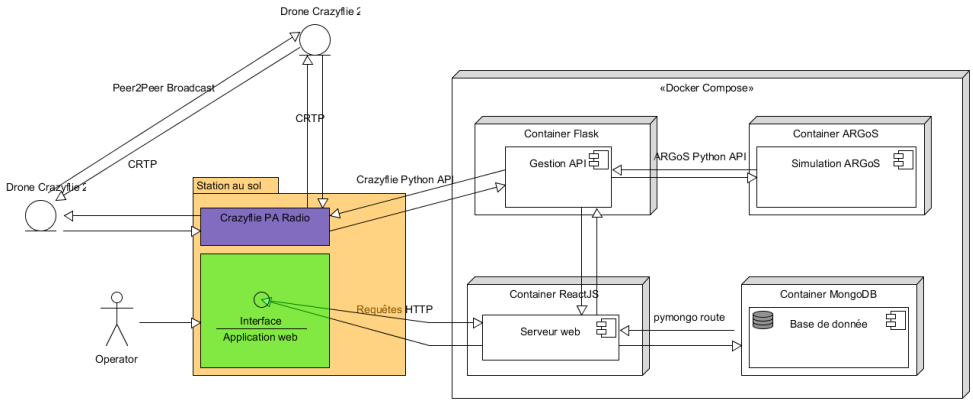

Figure 1: Architecture générale du système informatique minimal d'opération d'essaim de drones.

3.2 Architecture logicielle embarquée

Le Crazyflie 2.1 est la troisième version de quadricoptère miniature développé par Bitcraze, une compagnie suédoise. Le Crazyflie que nous utilisons dans le cadre d'INF3995 est livré en kit que la compagnie nomme "STEM Bundle". Le Crazyflie pèse 27g et mesure 92x92mm une fois assemblé. Il dispose de quatre moteurs "brushless" DC de 7 mm qui peuvent supporter un poids maximal au décollage de 42 g **(12).** Cela permet au Crazyflie 2.1 de transporter plus de matériel sous la forme de platines d'extension pouvant fournir des fonctionnalités supplémentaires, telles que des capteurs. Un deck d'extension peut être placé sur le dessus ou sous le Crazyflie. Une batterie entièrement chargée donne au quadcopter environ 6 minutes de temps de vol. La figure 2 ci-dessous explicite le lien entre chacune des composantes du systèmes embarqué et les connexions entre le client et le système embarqué. Le sigle CRTP indique le Crazyflie Real Time Protocol qui est le protocole de communication entre le drone et la radio. Un API Peer2Peer est disponible sur la plateforme du manufacturier Bitcraze afin que les drones puissent d'échanger des paquets. Le CrazyFfie API est écrit en Python et permet d'utiliser les classes et méthode du système embarqué du drone.

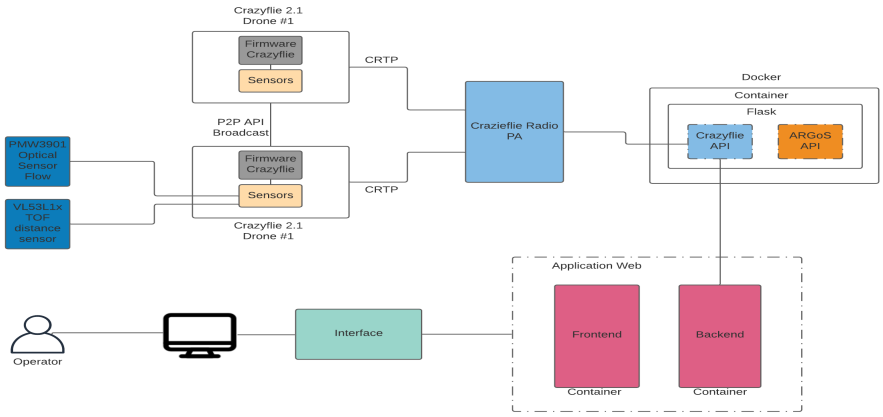

Figure 2: Architecture logicielle embarquée du CrazyFlie et ses interfaces

Le deck Flow V2 donne au Crazyflie 2.1 la capacité de comprendre quand il se déplace dans n'importe quelle direction **(7).** Le capteur ToF VL53L1x mesure la distance au sol avec une grande précision et le capteur de débit optique PMW3901 mesure les mouvements du sol. Le capteur ToF VL53L1x est le plus rapide d'après nos recherches sur le marché avec une précision allant jusqu'à 4 m et une fréquence allant jusqu'à 50 Hz. Le deck multi-ranger **(6)** donne à Crazyflie 2.1 la capacité de détecter la distance par rapport aux objets qui l'entourent avec une précision de mm,

jusqu'à 4 mètres de distance. Il contient 5 capteurs de télémétrie VL53L1X orientés vers l'avant, la gauche, l'arrière, la droite et le haut. Lorsqu'il est utilisé avec le pont Flow, le Crazyflie 2.1 peut se déplacer de manière autonome et éviter les obstacles **(4,5,7)**.

Pour nous déplacer dans les environnements inconnus, nous avons décidé d'utiliser un algorithme, précédemment utilisé par d'autres chercheurs, qui s'appelle Swarm Gradient Bud Algorithm **(4)**. Nous détaillerons l'utilisation du SGBA dans la section 3.4. La figure 3 ci-dessous montre l'intéraction entre les composantes qui permettront de contrôler le drone.

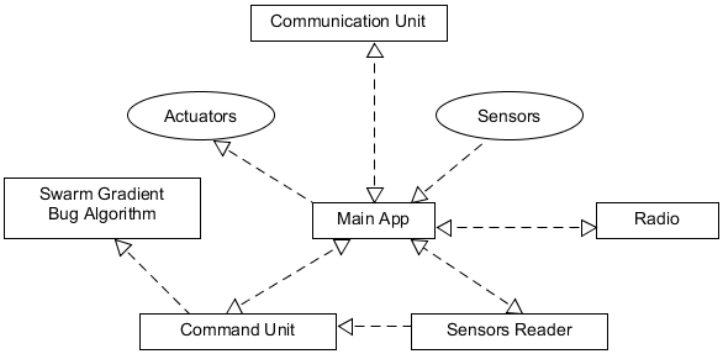

Figure 3: Architecture logicielle embarquée du CrazyFlie et ses interfaces

3.3 Architecture logicielle station au sol

La station au sol possède une interface web qui sera construite sur le modèle CRUD (Create, Read, Update, Delete) appliqué à une architecture REST (POST, GET, PUT/PATCH, DELETE). Pour construire cette application web, nous avons choisi d'utiliser la pile FReMP **(10).**

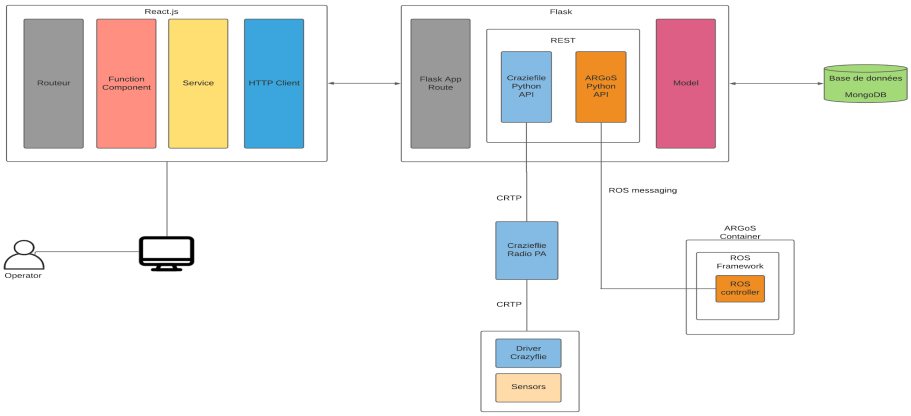

Figure 3: Architecture de la station au sol reposant sur la pile de développement FReMP (Flask, ReactJS, MongoDB, Python)

FReMP est l'acronyme pour Flask, ReactJS, MongoDB et Python. Contrairement à d'autres piles célèbres comme MEAN et MERN, cette pile utilise Python pour gérer les opérations back-end **(11).** Nous avons choisi cette pile, car autant le CrazyFlie drone et ARGoS utilisent des API python pour communiquer avec un client web. Ainsi l'utilisation de cette pile facilitera l'interface avec ces différents systèmes **(16,17).**

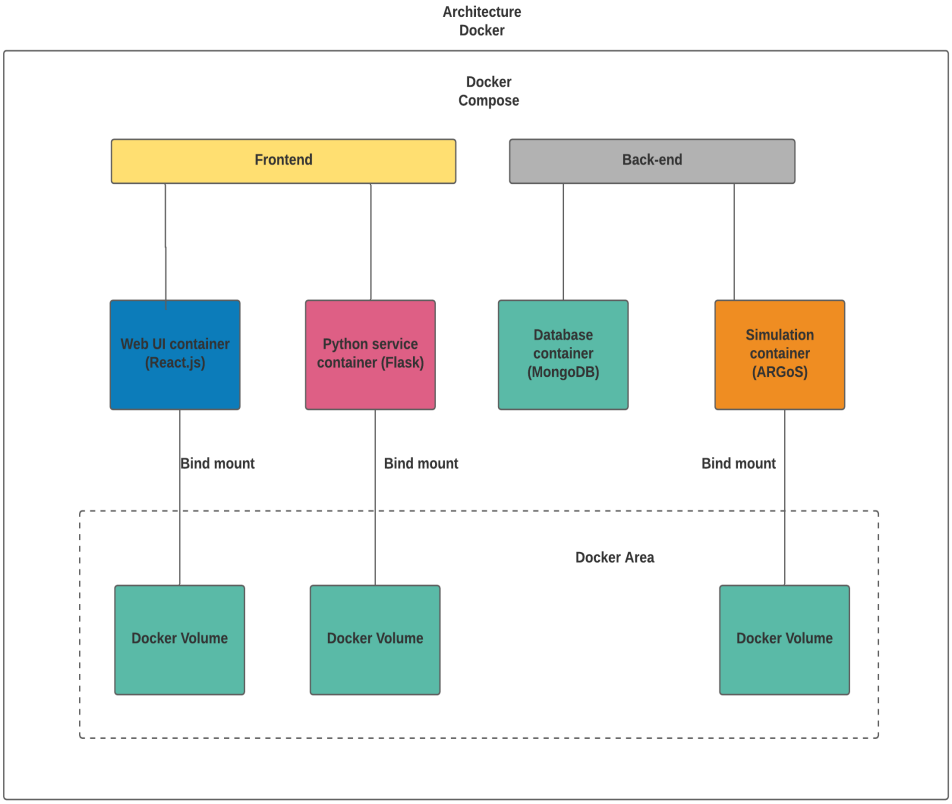

Figure 4: Architecture de Docker Compose avec les containers principaux de notre système informatique

Le client léger construit avec la pile FReMP s'occupera de gérer et distribuer les différentes requêtes aux différents conteneurs Docker qui contiennent le serveur central de l'application web créée avec ReactJS **(2,11)** et la gestion des services Python (Flask), la simulation ARGoS et la base de données MongoDB. Le serveur web sera dans un Docker et les conteneurs seront gérés par Docker Compose version 1.28.2 pour pouvoir facilement soit connecter le client avec le simulateur ARGoS ou le drone à travers l'antenne radio Crazyflie PA **(1,8,11,16,17).** Nous utiliserons MongoDB, une base de données NoSQL afin de stocker les données des logs du Crazyflie 2.1.**(12,13)**

Pour faciliter la portabilité de notre solution clef en main, nous déplacerons nos différentes composantes logicielles sur Docker en les utilisant selon le schéma Figure 4 **(1)**.

La station au sol communique avec le Crazyflie à l'aide la Crazyflie Radio PA. En utilisant le protocle CRTP de la radio, en conjoncture avec l'API Python du Crazyflie, l'opérateur peut envoyer des commandes à l'essaim de drones.

3.4 Architecture de la navigation autonome

Pour réaliser la navigation automne du drone, nous comptons utiliser en premier temps un algorithme développé par K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen et G. C. H. E. de Croon dans leur étude "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment" **(4)** . L'algorithme qu'ils ont créé, le "Swarm Gradient Bug Algorithm". Comme son nom l'indique, la méthode est inspirée des «algorithmes d'insecte», qui sont à l'origine de simples algorithmes de résolution de labyrinthe. Le concept de base est que la navigation et l'exploration ne s'effectuent pas en étant planifiées sur une carte globale avec des obstacles connus, mais en réagissant aux obstacles lorsqu'ils se trouvent à portée des capteurs télémétriques. Il s'agit d'un algorithme plus

sophistiqué que le simple suivi de mur. Cette façon de gérer les obstacles se traduit par une navigation hautement efficace en termes de calcul.

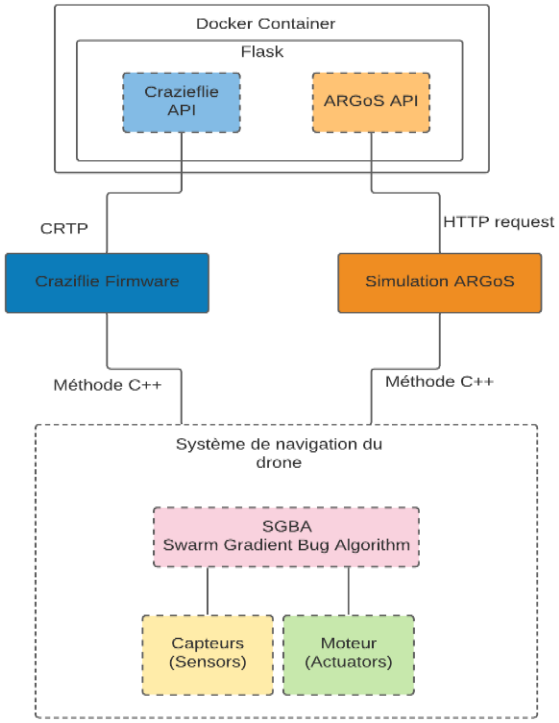

Figure 5: Schéma du fonctionnement de la navigation autonome du drone avec le Crazyflie et dans la simulation ARGoS

Le schéma ci-dessus nous aide à comprendre l'utilisation du SGBA autant par le simulateur ARGoS que le Crazyflie 2.1 **(11,16,17)**. L'utilisation d'une solution de navigation unique à travers notre système est importante pour pouvoir tester et implémenter une solution minimale de navigation opérationnelle.
Nous fournissons ci-dessous, une description sommaire de l'algorithme SGBA adapté à notre projet que nous allons utiliser dans notre solution clef en main pour l'Agence Spatiale:

Un essaim de Crazyflie 2.1 quitte la station de base au sol pour explorer un environnement. Chaque drone a une direction préférentielle initialement différente vers laquelle il essaiera d'aller. Lorsque les drones rencontrent des obstacles, ils suivent les contours de ceux-ci. Ce processus est appelé « suivi de mur» dans la littérature sur les algorithmes d'insectes et correspond à l'algorithme mis en place sur les robots du projet intégrateur de première année. Lorsqu'une direction préférée d'un robot est à nouveau libre d'obstacles, il continue de suivre sa direction préférée. Dès que la batterie du robot atteint 30% (d'après les requis de l'agence), il commence à naviguer vers la station au sol. Pour revenir à l'emplacement d'origine, les robots utilisent un mélange d'odométrie et un gradient, exprimé comme l'intensité de la force du signal (RSSI) reçu par la Crazyflie Radio PA observable vers la station de base **(8).**

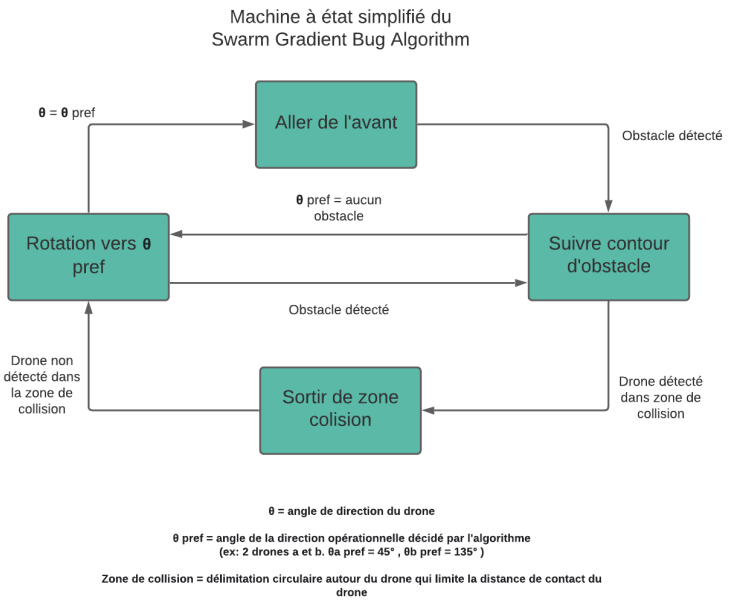

Figure 6: Machine à état de l'algorithme Swarm Gradient Bug Algorithm ***(4,5)***

Un seul robot pourrait utiliser le SGBA seul pour naviguer. Cependant, il n'a qu'une capacité de batterie limitée et ne pourra donc pas explorer tout l'environnement. Pour cette raison, il est plus avantageux d'utiliser un essaim de robots. Toutefois, le recours à un essaim de drones implique la nécessité d'avoir une communication drone à drone. Pour réaliser cette communication drone à drone, nous nous baserons sur l'API Peer2PeerBroadcast de Bitcraze. Nous n'aurons alors qu'à synchroniser avec un "clock interne" la communication par paquets entre drones/drones et drones/station au sol.

4. Processus de gestion

4.1 Estimations des coûts du projet

En ce qui a trait au coût du projet, l'estimation du coût est essentiellement liée à l'expertise de la main-d'œuvre. En effet, le coût du projet est directement en lien avec les salaires du personnel qui travaillera sur celui-ci. Par la nature technique du projet, le personnel est principalement constitué de développeurs-analystes ainsi que d'un responsable coordinateur du projet. L'équipement nécessaire à la réalisation du projet est minime puisque le personnel dispose déjà de leurs ordinateurs de travail. Ils pourront installer l'environnement de simulation nécessaire pour effectuer la preuve de concept. Des tests seront effectués sur des drones dont l'équipe devra se procurer, mais le coût de ces drones est très minime comparativement à la main-d'œuvre des employés. De plus, nous conserverons les drones programmables pour d'autres utilisations. Le tableau qui suit présente une estimation des coûts du projet.

Type de personnel	***Nombre de personnel***	***taux horaire ($/h)***	***Nombre d'heures estimé de travail***	***Coût total***
Développeur- analyste	4	130	145	70200$
Coordonnateur de projet	1	145	90	13050$
			Coût total du Projet	83250$

Tableau 2: Estimation des coûts du projet

4.2 Planification des tâches

La planification des tâches étant une activité qui demande une certaine projection dans le futur, elle est nécessairement empruntée à certaines incertitudes et d'un flou conceptuel. Voici, aux meilleures de nos connaissances, notre planification des tâches pour le projet INF3995 cette session. Le pourcentage d'allocation représente le poids relatif d'importance de chacune des tâches. Il sera transféré d'une échelle sur 100% en échelle de 1 à 10 sur Gitlab (exemple : 20% =2/10 du poids sur Gitlab)

4.2.1. Tâches du prelimanary design review (PDR)

		Sprint		1		
				Début	Fin	
Livrables		Heures PDR	Allocation	1er février	15 février	Responsables
PDR	Tâches	92				
PDR1	Simulation ARGoS avec deux drones qui suivent un parcours quelconque	26	28.26%			Nabil & Mazigh
PDR1.1	Création de la VM pour faire rouler ARGoS	12	46.15%		x	Nabil & Mazigh
PDR1.2	Simulation ARGoS avec un (1) drone qui suit un parcours quelconque	8	30.77%		x	Nabil & Mazigh
PDR1.3	Simulatio ARGoS avec deux (2) drones qui suivent un parcours quelconque	6	23.08%		x	Nabil & Mazigh
PDR2	Serveur web sur la station au sol avec une page qui montre le niveau de batterie des drones et avec un bouton qui allume ou éteint leurs D.E.L.	36	39.13%			Yassir & Tarik
PDR2.1	Création d'un serveur web	8	22.22%		x	Yassir & Tarik
PDR2.2	Création d'une application web CRUD avec le stack FReMP	10	27.78%		x	Yassir & Tarik
PDR2.3	Connexion entre le drone et l'application web	10	27.78%		x	Yassir & Tarik
PDR2.4	Création du bouton pour allumer les D.E.L. du drone	4	11.11%		x	Yassir & Tarik
PDR2.5	Affichage du niveau de batterie du drone	4	11.11%		x	Yassir & Tarik
PDR3	Documentation de gestion de projet (gabarit)	30	32.61%			Paul
PDR3.1	Création des schémas d'architecture	10	33.33%		x	Paul
PDR3.2	Insertion des fonctionnalités dans Gitlab	2	6.67%		x	Paul
PDR3.3	Complétion du gabarit PDR	18	60.00%		x	Paul

Tableau 3: Liste des tâches prévues pour le prelimanary design review(PDR)

4.2.2. Tâches du critical design review (CDR)

		Sprint		2		3		
				Début	Fin	Début	Fin	
Livrables	Tâches	Heures CDR	Allocation	15 février	26 février	26 février	8 mars	Responsables
CDR		184	100%					
CDR1	Simulation ARGoS avec 4 drones qui volent dans un environnement avec murs générés aléatoirement	24	12.00%				x	Nabil/Paul

CDR1.1	Capacité à faire interagir 4 drones qui volent avec un parcours quelconque	10	41.67%		x			Nabil
CDR1.2	Création d'un environnement avec des murs générés aléatoirement	14	58.33%				x	Paul
CDR2	Les drones dans la simulation utilisent des capteurs de distance (comme le ranging deck) pour éviter les obstacles et décider leur parcours	20	10.00%		x			Nabil/Mazigh/Paul
CDR2.1	Les drones dans la simulation détectent les murs	10	50.00%		x			Mazigh/Nabil
CDR2.2	Les drones dans la simulation détectent les autres drones	10	50.00%		x			Nabil/Paul
CDR3	Serveur web interfacé avec la simulation ARGoS	25	12.50%		x			Nabil/Paul
CDR3.0.1	Construction du pont entre ARGoS et le navigateur web à travers Flask	14	56.00%		x			Mazigh
CDR3.0.2	Configuration du pont entre ARGoS et Docker	5	20.00%		x			Mazigh
CDR3.0.3	Configuration de Docker compose	4	16.00%		x			Mazigh
CDR3.0.4	Mise à jour du readme pour expliquer les démarches	2	14.29%		x			Mazih/Paul
CDR3.1	Commandes « Take off » et « Return to base » implémentées	20	10.00%		x			Yassir/Tarik
CDR3.1.1	Implémentation de l'API Python, l'interface web et le ROS du Crazyflie	4	20.00%		x			Yassir/Tarik
CDR3.1.2	Implémentation du "Take off"	8	40.00%		x			Yassir
CDR3.1.3	Implémentation du "Return to base"	8	40.00%		x			Tarik
CDR3.2	Interface du système implémenté selon le requis R.F.5	28	14.00%				x	Yassir/Tarik
CDR3.2.1	L'essaim de drones doit répondre au minimum aux commandes suivantes, disponibles sur l'interface utilisateur : — Take-off/Start mission — Land/End mission — Software update — Return to base	12	42.86%				x	Yassir/Tarik
CDR3.2.2	Conception de l'interface web, tablette et mobile	6	21.43%		x			Paul
CDR3.2.3	Ajout du statut du serveur et des drones	4	14.29%				x	Tarik
CDR3.2.4	Ajout liste des drones connectés	4	14.29%				x	Yassir
CDR3.2.5	Création d'une page de détail par drone connecté	2	7.14%				x	Paul
CDR4	Code embarqué sur les drones qui envoie les mesures du ranging deck selon les requis R.F.5 et R.L.6	37	18.50%				x	Yassir/Tarik/Mazigh
CDR4.1	L'interface graphique affiche la vitesse des drones	4	10.81%		x			Yassir/Mazigh
CDR4.2	L'interface graphique comporte un éditeur de code	9	24.32%		x			Yassir/Tarik
CDR4.3	L'interface graphique comporte une carte dessinée des drones	10	27.03%		x			MazighTarik
CDR4.4	L'interface graphique affiche la position des drones	6	16.22%		x			Yassir/Tarik
CDR4.5	Programmation de l'algorithme "Swarm Gradient Bug Algorithm" (SGBA)	8	21.62%				x	Mazigh/Nabil
CDR5	Un prototype de visualisation de la carte générée par les robots	30	15.00%				x	Tarik/Paul
CDR5.1	Affichage des points de contact détectés par les senseurs des Crazyflie	8	26.67%		x			Tarik/Paul
CDR5.2	Plotting de ces points avec la libraire Matplotlib	14	46.67%				x	Yassir/Tarik
CDR5.3	Affichage de carte généré sur le navigateur web	8	26.67%				x	Yassir/Paul

Tableau 4: Liste des tâches prévues pour le critical design review (CDR)

4.2.3. Tâches du qualification review (QR)

		Sprint		4		5		
				Début	Fin	Début	Fin	
Livrables	Tâches	Heures CDR	Allocation	8 mars	19 mars	19 mars	29 mars	Responsables
QR		110	100%					
QR1	Organisation des user test	14	12.73%				x	Paul/Mazigh
QR1.1	Préparation des outils de tests (Mobile, Tablette, Laptop), de deux caméras pour filmer en détail les gestes de l'utilisateur et l'environnement de test ainsi qu'un microphone externe.	2	14.29%				x	Paul
QR1.2	Location d'une salle, envoi des invitations, tâches administratives	1	7.14%		x			Mazigh
QR1.3	Protocole des tests	11	78.57%				x	Paul/Mazigh
QR2	L'application web répond au requis RL4, RL5, RF4 et RF5	36	32.73%				x	Tarik/Yassir
QR2.1	L'interface de contrôle est disponible comme service web et visualisable sur plusieurs appareils (PC, tablette, téléphone...) via réseau	9	0.25				x	Yassir/Paul
QR2.2	L'essaim de drones répond aux commandes suivantes sur l'interface utilisateur : — Take-off/Start mission — Land/End mission — Software update — Return to base	4	11.11%		x			Tarik/Yassir
QR2.3	L'interface peut faire la mise à jour du logiciel sur les drones seulement lorsqu'ils sont au sol.	7	19.44%		x			Tarik
QR2.4	L'interface graphique affiche lorsque le drone est au sol ou en vol	6	16.67%		x			Yassir
QR2.5	L'interface graphique affiche l'état de la mise à jour	4	11.11%		x			Tarik
QR2.6	L'interface utilisateur doit montrer les informations suivantes, mises à jour avec une fréquence minimale de 1 Hz : 1. Nombre des drones 2. État des drones (standby, in-mission, crashed) 3. Vitesse courante des drones 4. Niveau de batterie des drones 5. Carte générée durant l'exploration 6.	6	16.67%				x	Tarik/Yassir
QR3	L'opérateur du système peut vérifier que le système de collecte de données opère correctement et que les données elles-mêmes sont manipulées correctement (logs).	26	23.64%					Nabil/Mazigh
QR3.1	Le retour à la base et l'atterrissage est actif automatiquement dès que le niveau de batterie devient moins de 30%. Les drones ne décollent pas avec un niveau de batterie inférieur à 30%.	8	30.77%		x			Nabil/Mazigh
QR3.2	La distance entre les drones et la station au sol (pour le retour à la base) doit être estimée à travers la puissance de signal (RSSI) reçu par le Crazyradio PA	8	30.77%				x	Mazigh
QR3.3	Les drones envoient les mesures du « ranging deck » durant l'exploration à la station au sol avec une fréquence minimale de 1 Hz et celle-ci est affichée sur la page détail du drone	5	19.23%				x	Mazigh
QR3.4	La page détail du drone indique si le drone se rapproche de la station lorsque le signal devient inférieur à 1Hz	5	19.23%				x	Nabil/Mazigh
QR4	La simulation ARGoS implémente le SGBA	34	30.91%					Nabil/Paul
QR4.1	Programmation de reconnaissance d'angle d'attaque préférée pour X drones	10	29.41%		x			Nabil
QR4.2	Programmation de l'état de suivi d'obstacle	7	20.59%		x			Paul
QR4.3	Programmation de l'état de recul face à un drone détecté	7	20.59%				x	Nabil

QR4.4	Programmation de l'état de déplacement en suivant l'angle d'attaque préférée	6	17.65%				x	Nabil
QR4.5	Test et peaufinage avec des simulations dans des salles quelconques et 2 à une dizaine de drones.	4	11.76%				x	Nabil/Paul

Tableau 5: Liste des tâches prévues pour la qualification review (QR)

4.2.4. Tâches du readiness review (RR)

Étant donné la nature agile de notre développement logiciel, nous nous laissons une flexibilité de 50% sur l'assignation des heures pour les sprints 5 et 6 de la RR afin de prendre en compte les différents obstacles et ralentissements que nous aurions vécus entre le sprint 1,2,3 et 4. Le décalage de la remise du PDR du 8 février au 15 février est un bon exemple qui justifie notre approche de laisser 50% des heures à assigner pour les sprints 5 et 6. Nous ré-évaluerons leur assignation à la remise du CDR le 8 mars.

		Sprint		5		6		
				Début	Fin	Début	Fin	
Livrables	Tâches	Heures CDR	Allocation	19 mars	29 mars	29 mars	12 avril	Responsables
RR		184	100%					
RR1	Revues d'assurance qualité des livrables	20	10.87%				x	Mazigh/Nabil/Tarik
RR2	L'application web est au niveau de maturité 4	22	11.96%		x			Tarik/Yassir
RR2.1	Affichage de la carte des drones sur l'application client	8	36.36%		x			Tarik/Yassir
RR2.2	L'intégration des mesures des différents drones est faite par la station au sol en supposant que les positions et orientations initiales des drones sont connues ou en résolvant un problème d'optimisation permettant d'inférer la carte la plus plausible selon les données recueillies (Maximum Likelihood Estimation).	8	36.36%				x	Tarik/Yassir
RR2.3	Éditeur de codes permet de modifier certaines fonctions de navigation du drone	4	18.18%		x			Tarik/Yassir
RR2.4	Écriture du guide d'utilisateur	2	9.09%		x			Paul
RR3	Développement du P2P Crazyflie API	20	10.87%				x	Nabil/Mazigh
RR3.1	Échange de paquets entre drones	12	60.00%		x			Nabil
RR3.2	Communication pour détection entre drones établie	8	40.00%				x	Nabil/Mazigh
RR4	Programmation du SBGA sur le micrologiciel du drone	20	10.87%				x	Paul/Nabil
	Transfert du SGBA pour le simulateur ARGoS avec le micrologiciel Crazyflie 2.1	12	60.00%		x			Nabil
RR4.1	Résolution du problème de boucle perpétuel	8	40.00%				x	Paul
RR5	Création du vidéo pour la remise	10						Tarik/Mazigh
RRX.X	À assigner le 8 mars 2021	92	50.00%				x	À déterminer

Tableau 6: Liste des tâches prévues pour le readiness review

4.2.5. Tâches du acceptance review (AR)

		Sprint		7		
				Début	Fin	
Livrables	Tâches	Heures PDR	Allocation	12 avril	19 avril	Responsables
AR		12	100%			
AR1	Présentation keynote	4	33.33%		x	Paul
AR2	Création de la structure	2	16.67%		x	Paul
AR3	Pratique de la présentation	4	33.33%		x	Tous
AR4	Post-mortem final	2	16.67%		x	Tous

Tableau 7: Liste des tâches prévues pour le acceptance review (AR)

4.3 Calendrier de projet

Notre projet sera divisé en 5 jalons importants. Ces jalons sont inspirés des normes ECSS-E-HB-40-01A et ISO 90003 **(18,19)** en processus de développement agile appliqué au développement de système informatique :

Jalons	Descriptions	Date de remise
PDR	Preliminary Design Review	15 février 2021
CDR	Critical Design Review	8 mars 2021
QR	Qualification Review	29 mars 2021
RR	Readiness Review	12 avril 2021
AR	Acceptance Reviw	19 avril 2021

Tableau 8: Liste des jalons importants du projet

Ces jalons seront découpés en 7 sprints. Ces sprints seront accompagnés de rencontre Scrum hebdomadaire pour assurer un suivi des livrables et de l'assurance qualité. Les tâches de chacun de ces sprints sont détaillées dans la section 4.2 de ce rapport.

Sprints	Début	Fin
Sprint 1	12h45, 1er février	12h45, 15 février
Sprint 2	12h45, 15 février	13h45, 26 février
Sprint 3	13h45, 26 février	12h45, 8 mars
Sprint 4	12h45, 8 mars	13h45, 19 mars
Sprint 5	13h45, 19 mars	12h45, 29 mars
Sprint 6	12h45, 29 mars	12h45, 12 avril

Sprint 7	12h45, 12 avril	12h45, 19 avril

Tableau 9: Liste des sprints qui découpent les livrables et le projet

Voici une vue globale du calendrier du projet avec les différents livrables, les sprints correspondants et leur date de complétion. Nous montrons également le poids de chacun des livrables en pourcentage d'allocation des 630 heures.

		Sprint	1		2		3		4		5		6		7	
			Début	Fin	Début	Fin	Début	Fin	Début	Fin	Début	Fin	Début	Fin	Début	Fin
Livra-bles	Répar-tition	Alloca-tion	1er février	15 février	15 février	26 février	26 février	8 mars	8 mars	19 mars	19 mars	29 mars	29 mars	12 avril	12 avril	19 avril
PDR	92	14.60%		PDR												
CDR	184	29.21%						CDR								
QR	110	17.46%										QR				
RR	184	29.21%												RR		
AR	12	1.90%														AR
Rencontres	48	7.62%														

Tableau 10: Calendrier sommaire des jalons et des sprints

Nous avons planifié 8 heures de rencontre pour les Sprints 1 à 6 ce qui équivaut à 48h. En tout, nous prévoyons 90 heures allouées à la gestion d'équipe et du projet. Ces heures incluent la rédaction et planification des tâches au Sprint 1 pour le PDR qui totalise 30 heures ainsi que 12 heures au Sprint final pour la préparation de la présentation aux départements et le post-mortem du projet.

4.4 Ressources humaines du projet

L'exécution de ce projet reposera sur deux types de rôles principalement, un coordinateur (Scrum master) ainsi que des développeurs-analystes. Notre équipe sera composée d'un coordinateur de projet ainsi que de quatre développeurs-analystes.

4.4.1. Compétences des développeurs-analystes

Voici une brève liste d'exemple des compétences du développeur-analyste:

Habiletés techniques	Habiletés interpersonnelles
• Connaissance de la conteneurisation (docker) • Maîtrise des langages de programmation Python et C • Connaissance du protocole HTTP • Compréhension des différents types d'architecture logicielle et de systèmes embarqués	• Rigueur • Autonome • Esprit d'équipe • Proactif

Tableau 11: Compétences des développeurs-analystes

4.4.2. Compétences du gestionnaire de projet

Voici une brève liste d'exemple des compétences du coordinateur de projet:

Habiletés techniques et de gestion	Habiletés interpersonnelles
• Maîtrise des outils de gestion et versionnement informatique • Capacité de décisionnelle et d'analyse technique • Gestion du temps et de priorité des tâches	• Sens de l'organisation • Excellent communicateur • Esprit d'équipe • Leadership

Tableau 12: Compétences du gestionnaire du projet

4.4.3. Compétences du personnel

À travers ce projet, le coordinateur de projet assumera principalement son rôle, mais également le rôle de développeur-analyste et vice-versa. Chaque membre du personnel est également responsable du bon déroulement de celui-ci. Le tableau ci-dessous présente les compétences du personnel de l'équipe.

	C-P	Développeurs-Analystes			
	Paul	Mazigh	Nabil	Yassir	Tarik
Programmation C / C++	x	x	x	x	x
Python	x	x		x	x
Méthodologie Agile		x	x	x	x
Développement d'une application web	x		x	x	x
Développement d'un serveur web	x	x	x	x	x
Connaissance des conteneurs Docker		x		x	
Connaissance du système d'exploitation Linux et Windows	x	x	x	x	x

Connaissance outils de gestion de versionnement logiciel	x	x	x	x	x
Tests, assurance qualité			x	x	
Intégration continue			x	x	
Développement système embarqué et micrologiciel/middleware		x	x		
Base de données	x	x	x	x	x
API	x	x	x	x	x

Tableau 13: Compétences du personnel

5. Suivi de projet et contrôle

5.1 Contrôle de la qualité

Le contrôle de la qualité de notre solution se fera tout au long du projet (toutes les phases) en s'appuyant sur une liste de vérification, des lignes directrices de développement et de mécanismes de programmation en binôme ou de révision en groupe.

- La liste de vérification établie dès le début du projet est constituée de l'ensemble des exigences pour chaque composante attendue de la solution. Elle sera utilisée conjointement avec notre calendrier et la planification Gantt pour s'assurer de leur conformité avant la remise de chaque livrable.

- Les lignes directrices définies dès le début permettront de s'assurer du respect des recommandations ou bonnes pratiques en matière de développement logiciel. Quelques lignes directrices sont les suivantes: i) Les noms des variables et fichiers doivent être en CamelCase. ii) Les constantes sont en lettres majuscules. iii) La longueur d'une ligne doit être un maximum de 140 caractères.

- Les mécanismes de révision permettront de s'assurer de la qualité de notre solution. Le premier mécanisme qui prend en compte le niveau de compétence technique des membres de notre équipe reposera sur la notion de programmation en binôme à mettre en œuvre autant que possible. Elle peut paraître parfois lente, mais elle s'avère toutefois efficace dans des contextes similaires au nôtre. Le second sera celui de la révision collective un ou deux jours avant la date de remise de chaque livrable. Elle consistera à passer en revue le livrable (fonctionnalités attendues, assurance qualité) pour s'assurer de la qualité de ce dernier. Le troisième mécanisme est celui de l'entraide au sein de l'équipe. Le principe étant de solliciter le plus tôt possible l'aide de tout membre de l'équipe en cas de difficulté.

Nous avons également la liste de condition intangible du projet qui nous permettra d'évaluer à chaque remise la qualité de notre projet:

Code	Tâches
INTG	Intangible for Continuous Development
INTG.RM1	Le prototype doit être implémenté avec deux drones Bitcraze Crazyflie 2.1 avec le « STEM Ranging bundle » complètement installé, fournis par l'Agence
INTG.RM2	Le seul moyen de communication entre la station au sol et les drones doit être une seule Bitcrazy Crazyradio PA connectée à la station au sol, fournie par l'Agence
INTG.RM3	Les drones doivent avoir seulement le « ranging deck » et le « optical flow deck » installées pour opérer
INTG.RM4	La station au sol doit être un laptop ou PC avec une Crazyradio PA connectée par port USB

INTG.RC1	Le système doit suivre un processus de conception qui valide l'approche en simulation avec le simulateur ARGoS avant l'expérimentation sur le matériel.
INTG.RC2	Chaque composant du logiciel doit avoir un test unitaire correspondant.
INTG.RC3	Le système de développement logiciel doit avoir un ou plusieurs tests de régression qui valident chaque addition ou retrait de code
INTG.RC6	La simulation de la mise à jour du logiciel avec ARGoS n'est pas nécessaire
INTG.RL1	Les drones doivent être programmés en utilisant l'API de BitCraze. Cependant, il est possible d'utiliser une machine virtuelle basé sur l'API à bord du drone (p.ex. la machine virtuelle de Buzz, BVM).
INTG.RL2	Les drones doivent communiquer en utilisant l'API pour la communication P2P de BitCraze. L'utilisation d'un serveur comme relais des messages entre les drones est interdite.
INTG.RQ1	Le format (indentation, style de commentaires, boucles...) du code doit être le même pour tous les fichiers.
INTG.RQ2	Les classes fonctions et lignes de code doivent avoir des longueurs et des complexités raisonnables.
INTG.RQ3	Les noms des variables, fonctions, classes, etc. doivent être clairs et respecter les conventions de nommage.
INTG.RQ4	L'organisation (hiérarchie) des fichiers de code doit être irréprochable.

Tableau 14: Liste des critères intangibles du projet

5.2 Gestion de risque

Il est évident que durant le cheminement du projet, notre équipe rencontrera plusieurs risques. En effet, ceux-ci serviront à tester l'habileté de notre équipe à faire face à des situations imprévues et stressantes. Afin de ne pas être pris par l'élément de surprise, nous avons pris la peine d'énumérer des risques que nous pourrions probablement rencontrer lors de la réalisation du projet et les solutions à ceux-ci.

- Tout d'abord, il y a les risques concernant les retards de remise lors d'une échéance. En effet, en négligeant l'importance de la communication au sein de notre équipe, il est possible d'ignorer sans le vouloir les difficultés à progresser de l'un de nos coéquipiers. Afin de remédier à cette situation, nous comptons faire des réunions Scrum hebdomadaires afin de s'assurer de la progression uniforme de tous les membres de groupe. Durant ces rencontres chaque membre de l'équipe devra expliquer ce qu'il a réalisé depuis la dernière rencontre Scrum et sur quoi il compte travailler à partir d'aujourd'hui. Ce risque est d'une très grande importance, car il pourrait rendre le client mécontent et augmenter les coûts du projet. Le Scrum master aura donc un grand rôle à jouer afin qu'on ne rencontre pas de retard.

- Ensuite, il est possible qu'un membre de l'équipe réalise une fonctionnalité qui ne correspond pas du tout aux attentes du client. En effet, les directives et requêtes du document "exigence technique" peuvent parfois être ambiguës ou mal comprises et mener donc à des résultats différents de ce qui était demandé. Afin de remédier à cette situation, notre équipe va s'assurer de clarifier toutes les interrogations avec le client en cas d'incompréhension d'une exigence. Par la suite, afin de nous assurer que tout le monde ait bien compris ses tâches, nous ferons un tour de table où tout le monde expliquera en détail ce qu'il doit réaliser. Ce risque est d'une très grande importance, car elle pourrait amener d'autres problèmes comme un retard sur l'échéance, l'insatisfaction du client ou une dispute au sein du groupe.

- Par ailleurs, il faudrait prendre en compte la possibilité que l'un des membres de l'équipe décide d'abandonner le projet ou de s'absenter pour plusieurs jours sans donner de nouvelle. Afin d'éviter ce risque, nous nous sommes d'abord assurés que tous les membres de l'équipe avaient l'intention de rester jusqu'à la fin du projet. Ensuite, nous nous sommes partagé nos contacts téléĥonique et Discord afin de pouvoir rester en communication en tout temps en cas de besoin. Donc, si une personne à une situation d'urgence et qu'il ne pourra pas être présent lors d'un événement important (réunion Scrum), il a le devoir de le faire savoir

au reste de son équipe. Ce risque n'est pas vraiment contrôlable, mais grâce à une bonne communication on peut parvenir à réduire ces dégâts. Celui-ci est d'une importance capitale, car la disparition d'un membre de l'équipe peut ralentir tout le processus du projet et créer une grande situation de stress au sein de l'équipe.

5.3 Tests

Nous comptons mettre en place 4 types de tests pour nous assurer que le développement de notre système informatique respecte la liste des conditions intangibles de notre projet en plus de mettre en action les principes fondamentaux du développement agile tels que vus dans les normes ECSS-E-HB-40-01A et ISO 90003 **(18,19).**

- **Test unitaire**: Un test unitaire sera effectué sur nos unités de code autonome, comme la classe CBatteryDefaultSensor() d'ARGoS **(16,17)** ou la méthode CCrazyflie::sendsSetpoints() du CrazyFlie Radio **(8)**, et doit être effectué chaque fois qu'une unité a été mise en œuvre ou que la mise à jour d'une unité est terminée.

- **Test d'intégration:** Le test d'intégration vise à tester l'interaction entre plusieurs unités. Ce type de test sera effectué chaque fois qu'une nouvelle forme de communication a été établie entre nos conteneurs Docker, notre serveur web et la gestion des API du Crazyflie et ARGoS**(1,16,17).** Cela signifie qu'il est exécuté chaque fois qu'une unité récemment écrite est intégrée dans le reste du système ou chaque fois qu'une unité qui interagit avec d'autres systèmes a été mise à jour (et a réussi ses tests unitaires). Un exemple d'utilisation des tests d'intégration est l'utilisation du package pytest comme framework de base pour nos tests sur le conteneur Flask.

- **Test de régression:** Des tests de régression sont effectués chaque fois qu'un issue sera incorporée dans notre système, afin de vérifier qu'aucun nouveau bogue n'a été introduit. Cela signifie qu'il est exécuté après tous les correctifs, mises à niveau, corrections de bogues. Les tests de régression peuvent être considérés comme un cas particulier de test unitaire combiné et de test d'intégration. Nous réalisons notre interface usager grâce à ReactJS. Nous allons utiliser deux outils de test de régression pour travailler dans React.js, soit Jest et Enzyme. Voici trois stratégies que nous utiliserons pour tester nos composants ReactJS **(2) :**
    - Nous pouvons vérifier qu'une fonction particulière dans les *props* a été appelée quand un événement est distribué.
    - Nous pouvons également obtenir le résultat de la fonction *render* en fonction de l'état actuel du composant et le faire correspondre à une disposition prédéfinie.
    - Nous pouvons même vérifier si le nombre d'enfants du composant correspond à une quantité attendue.

- **Test d'acceptation:** Les tests d'acceptation sont effectués chaque fois qu'il est pertinent de vérifier qu'un sous-système (éventuellement l'ensemble du système) répond à toutes ses spécifications. Cela signifie qu'il est principalement exécuté avant de terminer un nouveau livrable ou d'annoncer l'achèvement d'une tâche plus importante. Les tests d'acceptation seront notre dernière vérification pour voir si nous avons vraiment atteint nos objectifs. Nous effectuerons un grand test d'acceptation le 29 mars, il s'agit de notre Qualification review. Ce test permettra de vérifier aurpès des utilisateurs externes la fonctionnalité et facilité d'utilisation de notre solution clef en main. Un autre exemple de test d'acceptation est l'utilisation de branche protégée avec Gitlab. Lors des pull et merge request, un développeur aura comme impératif de valider la complétion de l'issue et la liste des critères intangibles du projet.

5.4 Gestion de configuration

Pour le système de contrôle de versions, nous allons utiliser Gitlab. Ce logiciel nous permettra de décentraliser le travail afin que tout le monde puisse avoir une copie complète du projet. De plus, ce système nous permet de gérer les différentes branches des applications et de pouvoir revenir sur des versions antérieures ou créer de nouvelles

branches, sans affecter le projet présent. Lors de la publication d'une nouvelle version, au moins une autre personne de l'équipe devra accepter le merge request afin de vérifier que le code compile et ne brise pas d'autres fonctionnalités déjà complétées. Ainsi, tout le code source utilisé pour le projet sera stocké sur le GitLab de l'école qui n'est pas un répertoire public. Pour organiser le tout, chaque partie du projet aura sa propre branche : le serveur web, ARGoS, la gestion des API avec Flask, le micrologiciel du Crazfylie, etc. De plus, nous aurons une couche supplémentaire de division en branche avec une branche par sprint.

Comme mentionné à la section précédente, des tests unitaires seront utilisés pour s'assurer que les fonctionnalités ne brisent pas en cours de route. Toutes les données recueillies(logs) des drones seront stockées sur MongoDB et seront disponibles pour le serveur. Pour la documentation relative au code source et la conception, un fichier Readme.md sera créé un maintenu tout au long du développement. L'équipe s'assurera de mettre à jour la documentation au moins à chaque livrable afin de bien décrire les fonctionnalités et l'utilisation des logiciels. Des tags seront émis sur le master pour indiquer la remise des différents livrables (PDR, CDR, QR, RR, AR).

6. Références

1. What is Docker? | Opensource.com. (n.d.).(22 janvier 2021), https://opensource.com/resources/what-docker

2. DA14, Top 10 advantages of using React.js (1 février 2021), https://da-14.com/blog/its-high-time-reactjs-ten-reasons-give-it-try

3. Sokolsky, J. J., & Jockel, J. T. (2019). Canada and the Future of Strategic Defense. Perspectives on Strategic Defense, 185-197. doi:10.4324/9780429301506-17

4. K. McGuire, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment" (DataverseNL, 2019); https://hdl.handle.net/10411/YVP873. 43. Bitcraze AB, Crazyflie 2.0 (2019); www.bitcraze.io/crazyflie-2/.

5. McGuire, K. (2019). Indoor swarm exploration with Pocket Drones. https://doi.org/10.4233/uuid:48ed7edc934e-4dfc-b35c-fe04d55caee1

6. Bitcraze AB, Multi-ranger deck (2019); www.bitcraze.io/multi-ranger-deck/.

7. Bitcraze AB, Flow deck v2 (2019); www.bitcraze.io/flow-deck-v2/.

8. Bitcraze AB, Crazyradio PA (2019); www.bitcraze.io/crazyradio-pa/.

9. Noronha, Justin, "Development of a swarm control platform for educational and research applications" (2016). Graduate Theses and Dissertations. 15783. https://lib.dr.iastate.edu/etd/15783

10. Maulloo, H. (2020, July 06). The FReMP Stack - Building a full stack web application. Retrieved from https://medium.com/@akhilmaulloo/the-fremp-stack-building-a-full-stack-web-application-91308e505250

11. Crazyflie 2.1, Bitcraze. Available at:https://store.bitcraze.io/products/crazyflie-2-1

12. Bitcraze webpage, Bitcraze. Available at:https://www.bitcraze.io/

13. The bitcraze virtual machine, Github. Available at: https://github.com/bitcraze/bitcraze-vm/releases/

14. AutonomousSequence.py example script, github. Available at: https://github.com/bitcraze/crazyflie-lib-python/blob/master/examples/ autonomousSequence.py

15. Motion commander class, github. https://github.com/bitcraze/crazyflie-lib-python/blob/master/cflib/positioning/ motion commander.py

16. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G. D., Ducatelle, F., Stirling, T., Gutiérrez, A., Gambardella, L. M., & Dorigo, M. (2011). ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS 2011) (pp. 5027–5034). Los Alamitos: IEEE Comput. Soc.

17. Pinciroli, C. (n.d.). ARGoS website. Retrieved from https://www.argos-sim.info/dev_manual.php

18. IEEE Guide--Adoption of ISO/IEC 90003:2004 Software Engineering--Guidelines for the Application of ISO 9001:2000 to Computer Software. (n.d.). doi:10.1109/ieeestd.2008.4690902

19. Ahmad, E., Raza, B., Feldt, R., & Nordebäck, T. (2010). ECSS standard compliant agile software development. *Proceedings of the 2010 National Software Engineering Conference on - NSEC 10*. doi:10.1145/1890810.1890816

20. Flask Documentation Release 1.1x. (n.d.). Retrieved from https://flask.palletsprojects.com/en/1.1.x/

ANNEXE

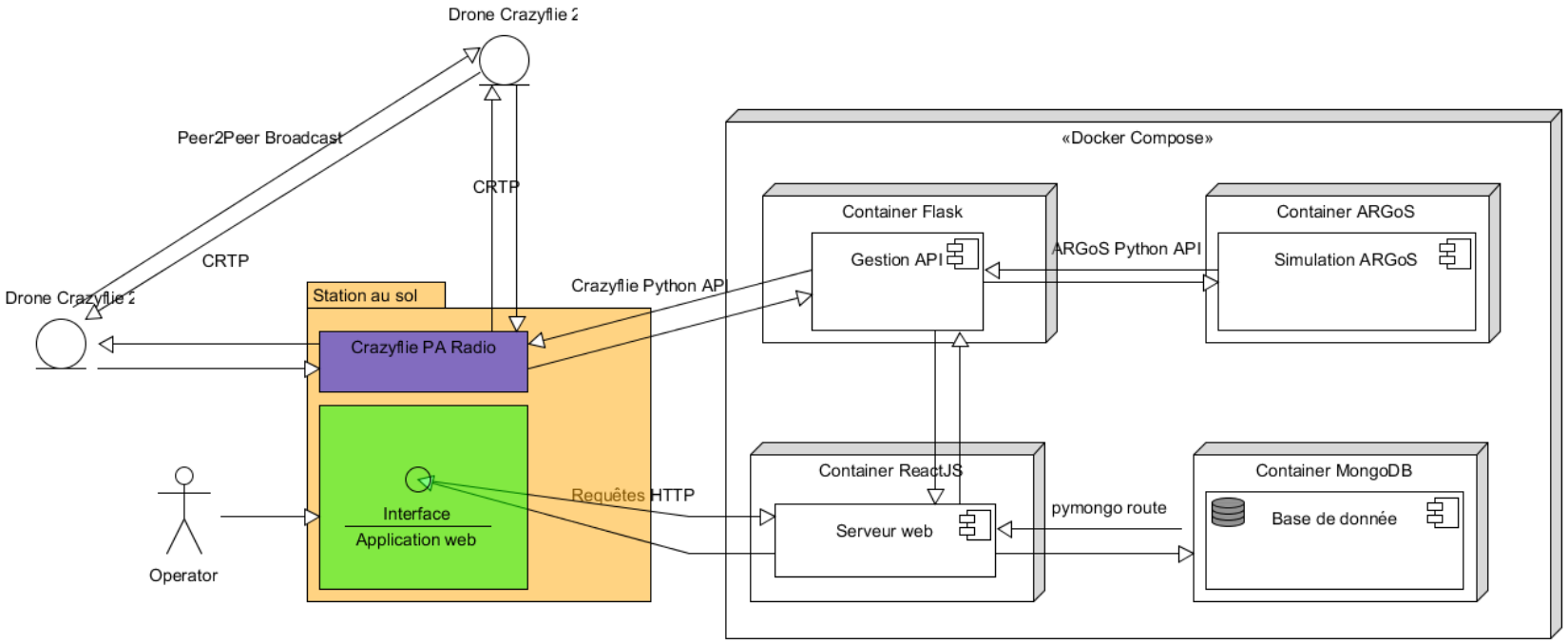

Figure 7: Architecture générale du système informatique minimal d'opération d'essaim de drones.

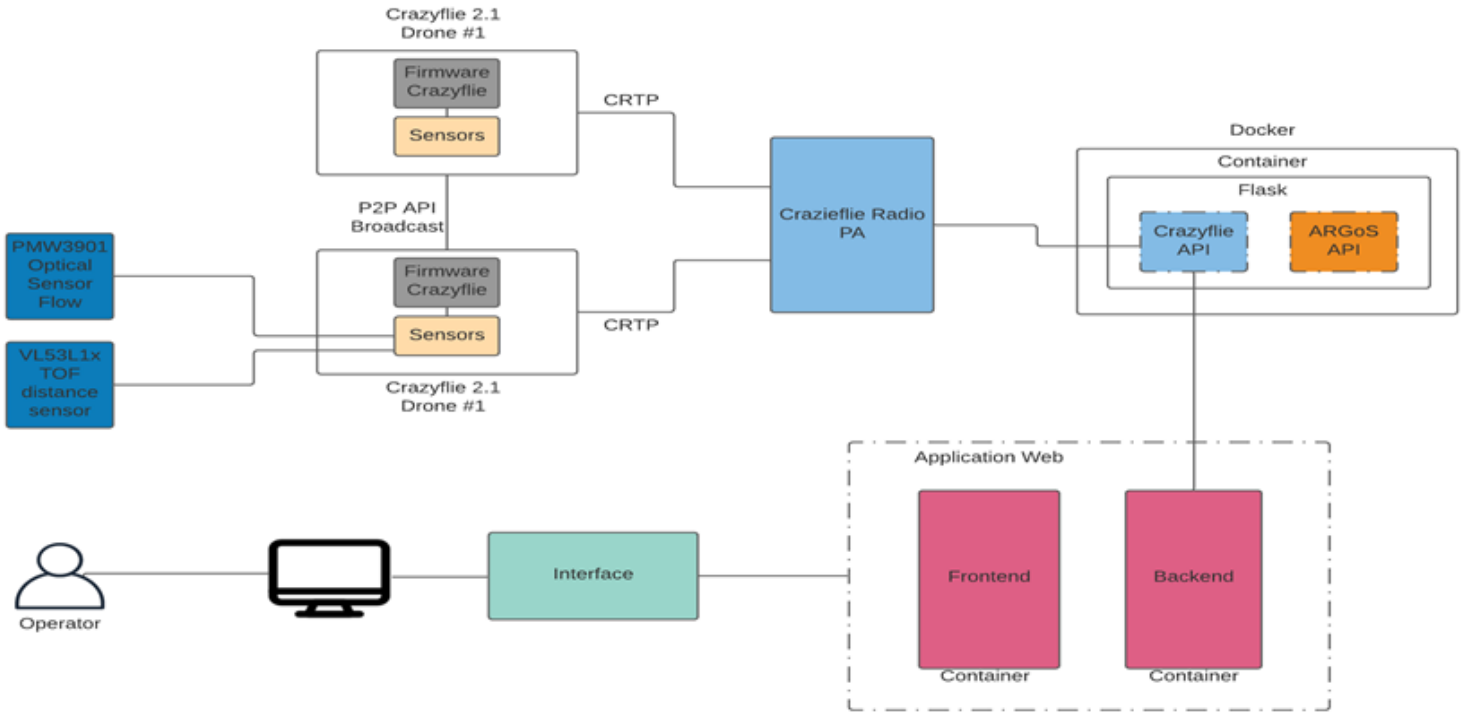

Figure 8: Architecture logicielle embarqué du CrazyFlie et ses interfaces

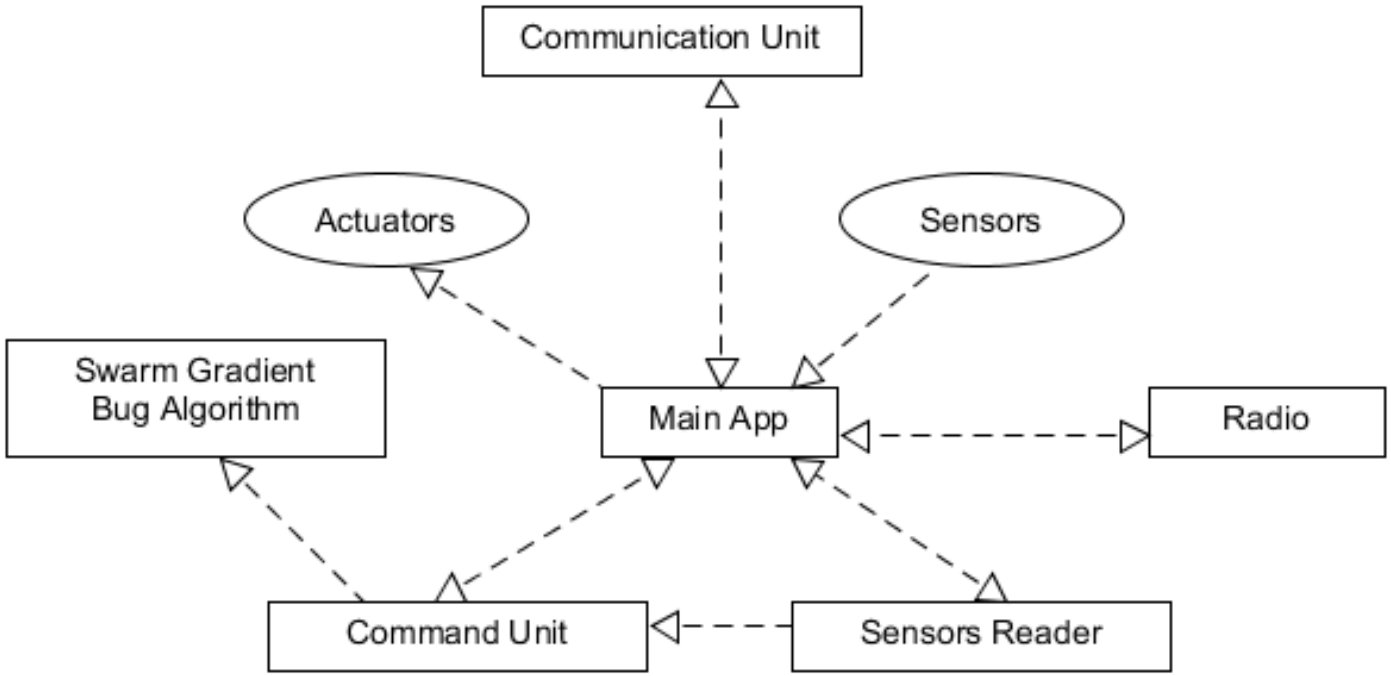

Figure 3: Architecture logicielle embarqué du CrazyFlie et ses interfaces

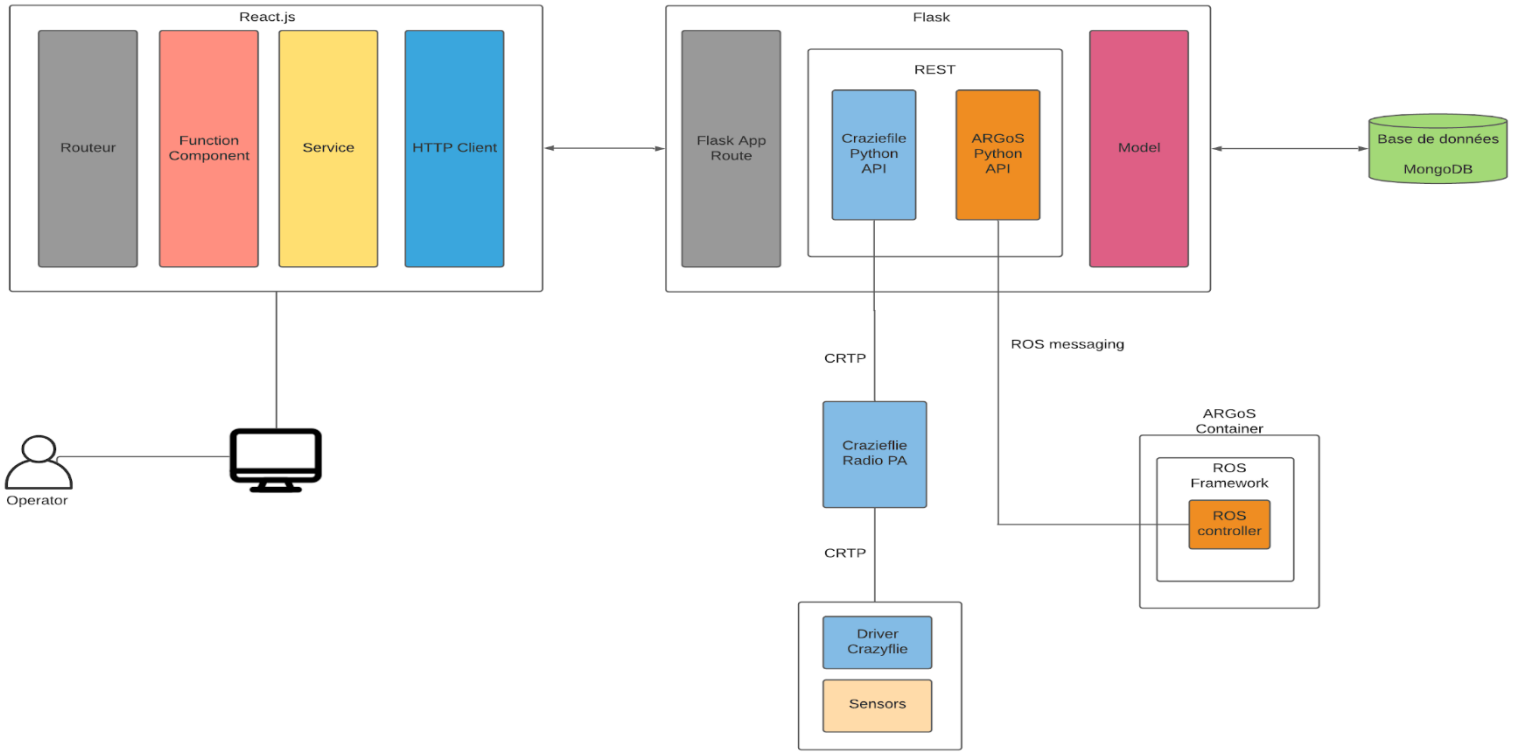

Figure 9: Architecture de la station au sol reposant sur la pile de développement FReMP (Flask, ReactJS, MongoDB, Python)

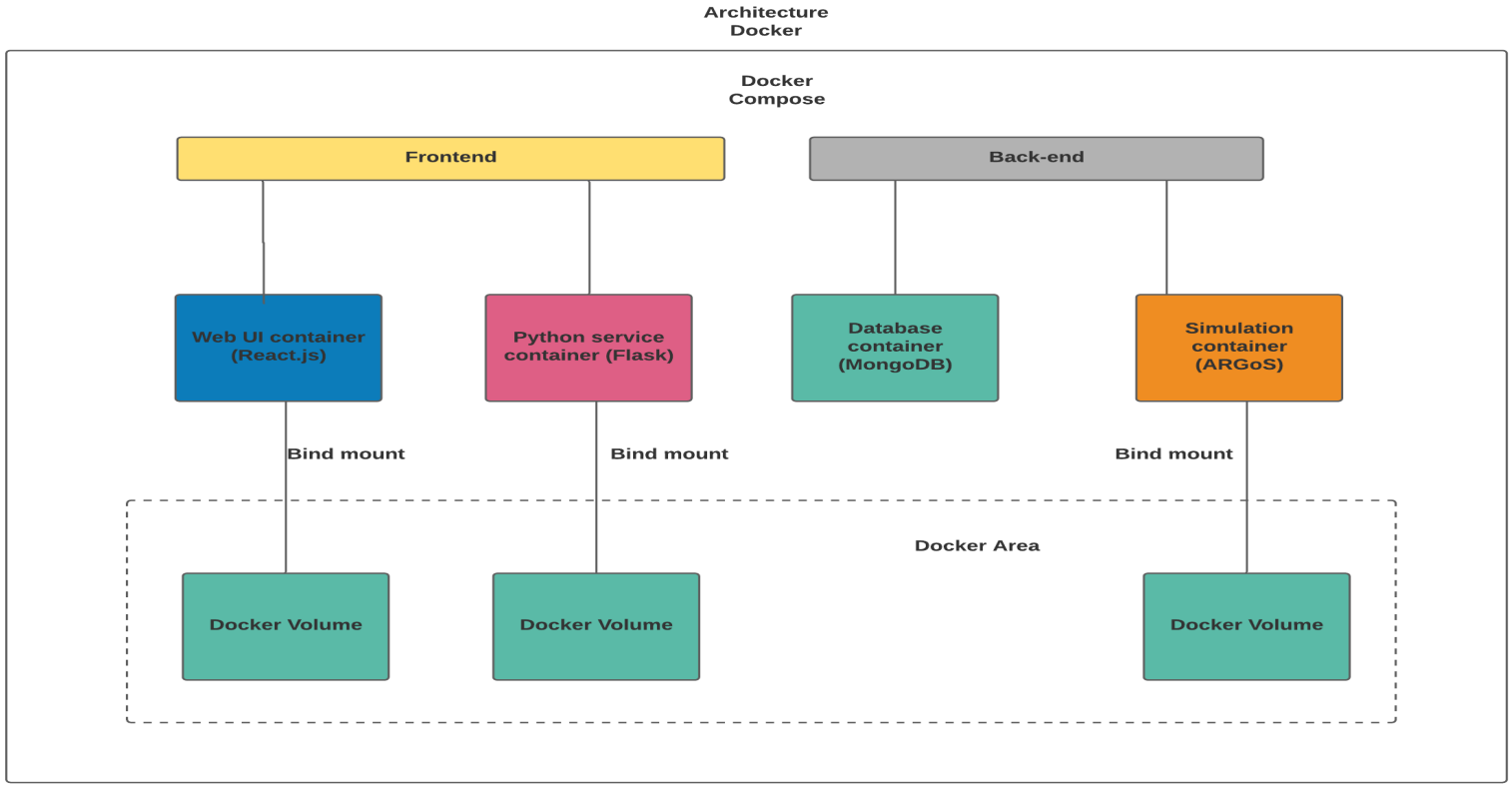

Figure 10: Architecture de Docker Compose avec les containers principaux de notre système informatique

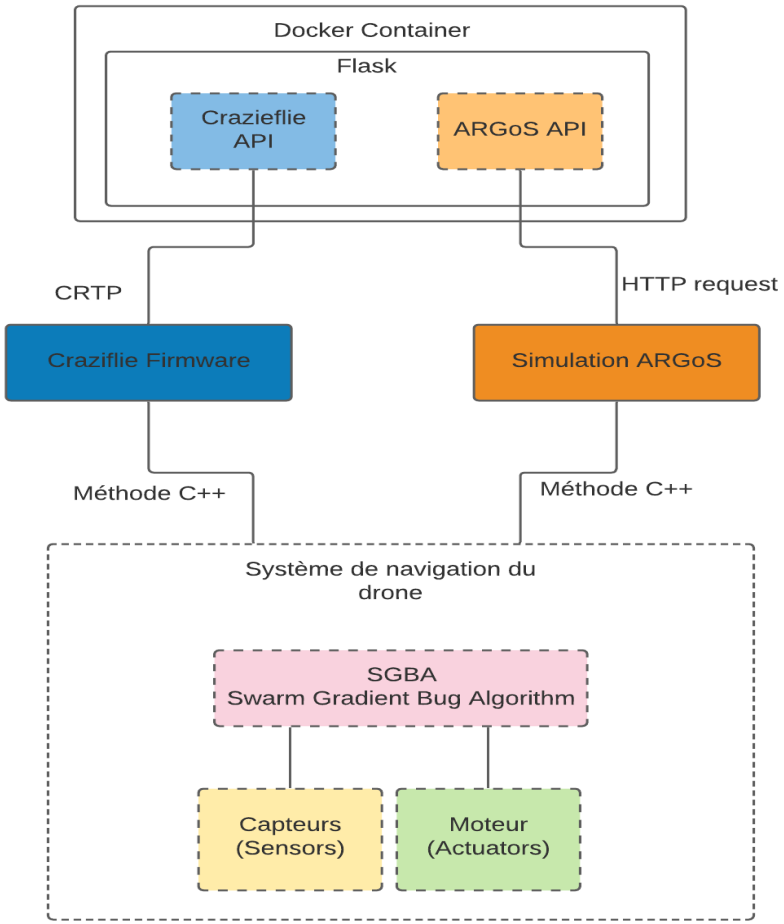

Figure 11: Schéma du fonctionnement de la navigation autonome du drone avec le Crazyflie et dans la simulation ARGoS

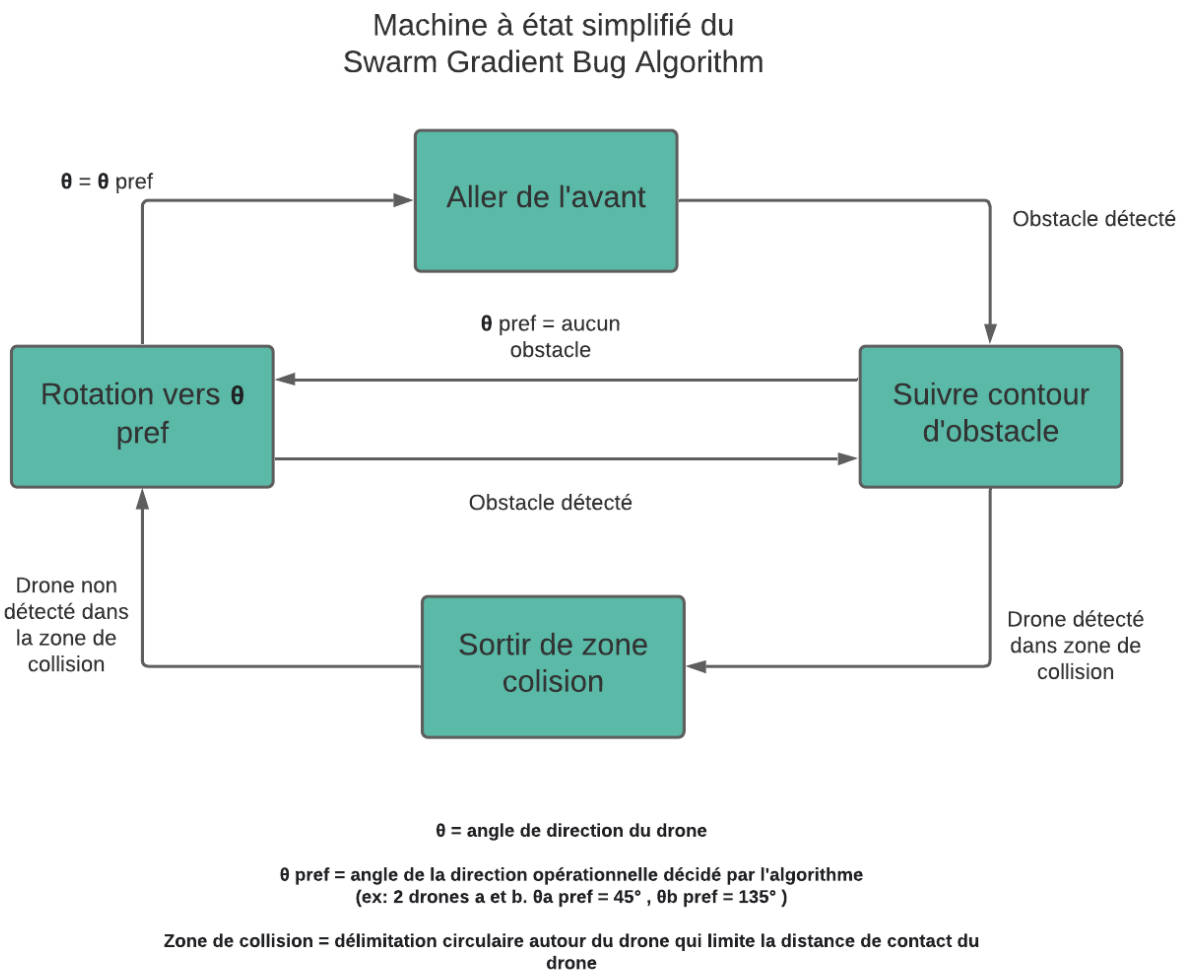

Figure 12: Machine à état de l'algorithme Swarm Gradient Bug Algorithm **(4,5)**

		Sprint		1		
				Début	Fin	
Livra-bles		Heures PDR	Allocation	1er février	15 février	Re-sponsables
PDR	Tâches	92				
PDR1	Simulation ARGoS avec deux drones qui suivent un parcours quel-conque	26	28.26%			Nabil & Mazigh
PDR1.1	Création de la VM pour faire rouler ARGoS	12	46.15%		x	Nabil & Mazigh
PDR1.2	Simulation ARGoS avec un (1) drone qui suit un parcours quelconque	8	30.77%		x	Nabil & Mazigh
PDR1.3	Simulatio ARGoS avec deux (2) drones qui suivent un parcours quel-conque	6	23.08%		x	Nabil & Mazigh
PDR2	Serveur web sur la station au sol avec une page qui montre le niveau de batterie des drones et avec un bouton qui allume ou éteint leurs D.E.L.	36	39.13%			Yassir & Tarik
PDR2.1	Création d'un serveur web	8	22.22%		x	Yassir & Tarik
PDR2.2	Création d'une application web CRUD avec le stack FReMP	10	27.78%		x	Yassir & Tarik
PDR2.3	Connexion entre le drone et l'application web	10	27.78%		x	Yassir & Tarik
PDR2.4	Création du bouton pour allumer les D.E.L. du drone	4	11.11%		x	Yassir & Tarik
PDR2.5	Affichage du niveau de batterie du drone	4	11.11%		x	Yassir & Tarik
PDR3	Documentation de gestion de projet (gabarit)	30	32.61%			Paul
PDR3.1	Création des schémas d'architecture	10	33.33%		x	Paul
PDR3.2	Insertion des fonctionnalités dans Gitlab	2	6.67%		x	Paul
PDR3.3	Complétion du gabarit PDR	18	60.00%		x	Paul

Tableau 15: Liste des tâches prévues pour le prelimanary design review(PDR)

		Sprint		2		3		
				Début	Fin	Début	Fin	
Livrables	Tâches	Heures CDR	Allocation	15 février	26 février	26 février	8 mars	Responsables
CDR		184	100%					
CDR1	Simulation ARGoS avec 4 drones qui volent dans un environnement avec murs générés aléatoirement	24	12.00%				x	Nabil/Paul
CDR1.1	Capacité à faire interagir 4 drones qui volent avec un parcours quelconque	10	41.67%		x			Nabil
CDR1.2	Création d'un environnement avec des murs générés aléatoirement	14	58.33%				x	Paul
CDR2	Les drones dans la simulation utilisent des capteurs de distance (comme le ranging deck) pour éviter les obstacles et décider leur parcours	20	10.00%		x			Nabil/Mazigh/Paul
CDR2.1	Les drones dans la simulation détectent les murs	10	50.00%		x			Mazigh/Nabil
CDR2.2	Les drones dans la simulation détectent les autres drones	10	50.00%		x			Nabil/Paul
CDR3	Serveur web interfacé avec la simulation ARGoS	25	12.50%		x			Nabil/Paul
CDR3.0.1	Construction du pont entre ARGoS et le navigateur web à travers Flask	14	56.00%		x			Mazigh
CDR3.0.2	Configuration du pont entre ARGoS et Docker	5	20.00%		x			Mazigh
CDR3.0.3	Configuration de Docker compose	4	16.00%		x			Mazigh
CDR3.0.4	Mise à jour du readme pour expliquer les démarches	2	14.29%		x			Mazih/Paul
CDR3.1	Commandes « Take off » et « Return to base » implémentées	20	10.00%		x			Yassir/Tarik
CDR3.1.1	Implémentation de l'API Python, l'interface web et le ROS du Crazyflie	4	20.00%		x			Yassir/Tarik
CDR3.1.2	Implémentation du "Take off"	8	40.00%		x			Yassir
CDR3.1.3	Implémentation du "Return to base"	8	40.00%		x			Tarik

CDR3.2	Interface du système implémenté selon le requis R.F.5	28	14.00%				x	Yassir/Tarik
CDR3.2.1	L'essaim de drones doit répondre au minimum aux commandes suivantes, disponibles sur l'interface utilisateur : — Take-off/Start mission — Land/End mission — Software update — Return to base	12	42.86%				x	Yassir/Tarik
CDR3.2.2	Conception de l'interface web, tablette et mobile	6	21.43%		x			Paul
CDR3.2.3	Ajout du statut du serveur et des drones	4	14.29%				x	Tarik
CDR3.2.4	Ajout liste des drones connectés	4	14.29%				x	Yassir
CDR3.2.5	Création d'une page de détail par drone connecté	2	7.14%				x	Paul
CDR4	Code embarqué sur les drones qui envoie les mesures du ranging deck selon les requis R.F.5 et R.L.6	37	18.50%				x	Yas-sir/Tarik/Mazigh
CDR4.1	L'interface graphique affiche la vitesse des drones	4	10.81%		x			Yassir/Mazigh
CDR4.2	L'interface graphique comporte un éditeur de code	9	24.32%		x			Yassir/Tarik
CDR4.3	L'interface graphique comporte une carte dessinée des drones	10	27.03%		x			MazighTarik
CDR4.4	L'interface graphique affiche la position des drones	6	16.22%		x			Yassir/Tarik
CDR4.5	Programmation de l'algorithme "Swarm Gradient Bug Algorithm" (SGBA)	8	21.62%				x	Mazigh/Nabil
CDR5	Un prototype de visualisation de la carte générée par les robots	30	15.00%				x	Tarik/Paul
CDR5.1	Affichage des points de contact détectés par les senseurs des Crazyflie	8	26.67%		x			Tarik/Paul
CDR5.2	Plotting de ces points avec la libraire Matplotlib	14	46.67%				x	Yassir/Tarik
CDR5.3	Affichage de carte généré sur le navigateur web	8	26.67%				x	Yassir/Paul

Tableau 16: Liste des tâches prévues pour le critical design review (CDR)

		Sprint		4		5		
				Début	Fin	Début	Fin	
Livrables	Tâches	Heures CDR	Allocation	8 mars	19 mars	19 mars	29 mars	Responsables
QR		110	100%					
QR1	Organisation des user test	14	12.73%				x	Paul/Mazigh
QR1.1	Préparation des outils de tests (Mobile, Tablette, Laptop), de deux caméras pour filmer en détail les gestes de l'utilisateur et l'environnement de test ainsi qu'un microphone externe.	2	14.29%				x	Paul
QR1.2	Location d'une salle, envoi des invitations, tâches administratives	1	7.14%		x			Mazigh
QR1.3	Protocole des tests	11	78.57%				x	Paul/Mazigh
QR2	L'application web répond au requis RL4, RL5, RF4 et RF5	36	32.73%				x	Tarik/Yassir
QR2.1	L'interface de contrôle est disponible comme service web et visualisable sur plusieurs appareils (PC, tablette, téléphone...) via réseau	9	0.25				x	Yassir/Paul
QR2.2	L'essaim de drones répond aux commandes suivantes sur l'interface utilisateur : — Take-off/Start mission — Land/End mission — Software update — Return to base	4	11.11%		x			Tarik/Yassir
QR2.3	L'interface peut faire la mise à jour du logiciel sur les drones seulement lorsqu'ils sont au sol.	7	19.44%		x			Tarik
QR2.4	L'interface graphique affiche lorsque le drone est au sol ou en vol	6	16.67%		x			Yassir
QR2.5	L'interface graphique affiche l'état de la mise à jour	4	11.11%		x			Tarik
QR2.6	L'interface utilisateur doit montrer les informations suivantes, mises à jour avec une fréquence minimale de 1 Hz : 1. Nombre des drones 2. État des drones (standby,	6	16.67%				x	Tarik/Yassir

	in-mission, crashed) 3. Vitesse courante des drones 4. Niveau de batterie des drones 5. Carte générée durant l'exploration 6.							
QR3	L'opérateur du système peut vérifier que le système de collecte de données opère correctement et que les données elles-mêmes sont manipulées correctement (logs).	26	23.64%					Nabil/Mazigh
QR3.1	Le retour à la base et l'atterrissage est actif automatiquement dès que le niveau de batterie devient moins de 30%. Les drones ne décollent pas avec un niveau de batterie inférieur à 30%.	8	30.77%		x			Nabil/Mazigh
QR3.2	La distance entre les drones et la station au sol (pour le retour à la base) doit être estimée à travers la puissance de signal (RSSI) reçu par le Crazyradio PA	8	30.77%				x	Mazigh
QR3.3	Les drones envoient les mesures du « ranging deck » durant l'exploration à la station au sol avec une fréquence minimale de 1 Hz et celle-ci est affichée sur la page détail du drone	5	19.23%				x	Mazigh
QR3.4	La page détail du drone indique si le drone se rapproche de de la station lorsque le signal devient inférieur à 1Hz	5	19.23%				x	Nabil/Mazigh
QR4	La simulation ARGoS implémente le SGBA	34	30.91%					Nabil/Paul
QR4.1	Programmation de reconnaissance d'angle d'attaque préférée pour X drones	10	29.41%		x			Nabil
QR4.2	Programmation de l'état de suivi d'obstacle	7	20.59%		x			Paul
QR4.3	Programmation de l'état de recul face à un drone détecté	7	20.59%				x	Nabil
QR4.4	Programmation de l'état de déplacement en suivant l'angle d'attaque préférée	6	17.65%				x	Nabil
QR4.5	Test et peaufinage avec des simulations dans des salles quelconques et 2 à une dizaine de drones.	4	11.76%				x	Nabil/Paul

Tableau 17: Liste des tâches prévues pour le qualification review (QR)

		Sprint		5		6		
				Début	Fin	Début	Fin	
Livrables	Tâches	Heures CDR	Allocation	19 mars	29 mars	29 mars	12 avril	Responsables
RR		184	100%					
RR1	Revues d'assurance qualité des livrables	20	10.87%				x	Mazigh/Nabil/Tarik
RR2	L'application web est au niveau de maturité 4	22	11.96%		x			Tarik/Yassir
RR2.1	Affichage de la carte des drones sur l'application client	8	36.36%		x			Tarik/Yassir
RR2.2	L'intégration des mesures des différents drones est faite par la station au sol en supposant que les positions et orientations initiales des drones sont connues ou en résolvant un problème d'optimisation permettant d'inférer la carte la plus plausible selon les données recueillies (Maximum Likelihood Estimation).	8	36.36%				x	Tarik/Yassir
RR2.3	Éditeur de codes permet de modifier certaines fonctions de navigation du drone	4	18.18%		x			Tarik/Yassir
RR2.4	Écriture du guide d'utilisateur	2	9.09%		x			Paul
RR3	Développement du P2P Crazyflie API	20	10.87%				x	Nabil/Mazigh
RR3.1	Échange de paquets entre drones	12	60.00%		x			Nabil
RR3.2	Communication pour détection entre drones établie	8	40.00%				x	Nabil/Mazigh
RR4	Programmation du SBGA sur le micrologiciel du drone	20	10.87%				x	Paul/Nabil
	Transfert du SGBA pour le simulateur ARGoS avec le micrologiciel Crazyflie 2.1	12	60.00%		x			Nabil
RR4.1	Résolution du problème de boucle perpétuel	8	40.00%				x	Paul
RR5	Création du vidéo pour la remise	10						Tarik/Mazigh
RRX.X	À assigner le 8 mars 2021	92	50.00%				x	À déterminer

Tableau 18: Liste des tâches prévues pour le readiness review

		Sprint		7		
				Début	Fin	
Livrables	Tâches	Heures PDR	Allocation	12 avril	19 avril	Responsables
AR		12	100%			
AR1	Présentation keynote	4	33.33%		x	Paul
AR2	Création de la structure	2	16.67%		x	Paul
AR3	Pratique de la présentation	4	33.33%		x	Tous
AR4	Post-mortem final	2	16.67%		x	Tous

Tableau 19: Liste des tâches prévues pour le acceptance review (AR)

Code	Tâches
INTG	Intangible for Continuous Development
INTG.RM1	Le prototype doit être implémenté avec deux drones Bitcraze Crazyflie 2.1 avec le « STEM Ranging bundle » complètement installés, fourni par l'Agence
INTG.RM2	Le seul moyen de communication entre la station au sol et les drones doit être une seule Bitcrazy Crazyradio PA connectée à la station au sol, fournie par l'Agence
INTG.RM3	Les drones doivent avoir seulement le « ranging deck » et le « optical flow deck » installés pour opérer
INTG.RM4	La station au sol doit être un laptop ou PC avec une Crazyradio PA connectée par port USB
INTG.RC1	Le système doit suivre un processus de conception qui valide l'approche en simulation avec le simulateur ARGoS avant l'expérimentation sur le matériel.
INTG.RC2	Chaque composant du logiciel doit avoir un test unitaire correspondant.
INTG.RC3	Le système de développement logiciel doit avoir un ou plusieurs tests de régression qui valident chaque addition ou retrait de code
INTG.RC6	La simulation de la mise à jour du logiciel avec ARGoS n'est pas nécessaire
INTG.RL1	Les drones doivent être programmés en utilisant l'API de BitCraze. Cependant, il est possible d'utiliser une machine virtuelle basée sur l'API à bord du drone (p.ex. la machine virtuelle de Buzz, BVM).
INTG.RL2	Les drones doivent communiquer en utilisant l'API pour la communication P2P de BitCraze. L'utilisation d'un serveur comme relais des messages entre les drones est interdite.
INTG.RQ1	Le format (indentation, style de commentaires, boucles...) du code doit être le même pour tous les fichiers.
INTG.RQ2	Les classes fonctions et lignes de code doivent avoir des longueurs et des complexités raisonnables.
INTG.RQ3	Les noms des variables, fonctions, classes, etc. doivent être clairs et respecter les conventions de nommage.
INTG.RQ4	L'organisation (hiérarchie) des fichiers de code doit être irréprochable.

Tableau 20: Liste des critères intangibles du projet