Dernière modification: 9 février 2021

INF3995: Projet de conception d'un système informatique
Hiver2021
Réponse à l'appel d'offres

Système aérien minimal pour exploration

Proposition répondant à l'appel d'offres n°H2021-INF3995 du département GIGL

Équipe No. 203

Bal, Samba Bousso

Chritin, Mathurin

Grootenboer, Hubert

Maghni, Issam Eddine

Sangam, Eya-Tom Augustin

Table des matières

Introduction

Bla bla bla

1 Vue d'ensemble du projet

1.1 But du projet, porté et objectifs

L'objectif principal de ce projet est de créer un système informatique de gestion de drones. Le dit système permettra à son utilisateur d'explorer et cartographier un milieu arbitraire depuis une station au sol. Le système ferra fonctionner une colonie de drones, communiquant entre eux, pour fournir une cartographie du milieu exploré.

Un tel système se révélera très utile durant les explorations sur Mars, encore peu connue de tous. Nous imaginons une situation dans laquelle, un robot plus complexe, plus lent et limité dans sa capacité de mouvement, se fera diriger par une colonie de drones qui lui indiquera les endroits les plus intéressants à explorer.

1.2 Hypothèses et contraintes

1.2.1 Hypothèses sur l'utilisation du système

- **Utilisation du système** :
  Le système conçu sera utilisé uniquement à des fins d'explorations et non d'espionnage. Selon le milieu exploré, l'utilisateur du système se devra d'obtenir les autorisations nécessaires conformément au lois régissant le lieu d'utilisation.
- **Sécurité** :
  L'opérateur du système devra être âgé d'au moins 16 ans et devra se munir de lunettes de sécurité durant toute interaction avec le drone.
- **Milieu exploré** :
  Le milieu exploré devra se trouver dans des conditions météorologiques convenables. Idéalement, une pièce fermée à température ambiante.

1.2.2 Contraintes matérielles

- **Les drones** :
  L'opérateur du système devra posséder au moins deux kits Bitcraze Crazyfly 2.0 STEM Bundle. Uniquement le « Flow deck » et le « Ranger Deck » fournis dans les kits devront être installés sur les drones.
- **La station au sol** :
  L'ordinateur de la station de contrôle devra être muni d'un système d'exploitation Linux, virtualisé ou non, possédant au minimum 4 Gio de mémoire vive. Un fureteur à jour et un terminal doivent être présent sur l'ordinateur.

— **Communication avec la station au sol** :
Le seul moyen de communication entre la station au sol et les drones doit être l'antenne Bitcraze CrazyradioPA connectée à la station au sol.

1.2.3 Contraintes fonctionnelles

Le système devrait être en mesure de répondre au contraintes ci-dessous :

— **Commandes supportées** :
Les commandes sont des instructions envoyées depuis l'interface de contrôle aux drones. Trois commandes devraient pour être supportées par les drones :

— Lancer la mission :
Une fois la commande de départ reçue, les drones doivent être capables de parcourir le périmètre d'une pièce d'au maximum 100 m^2 en absence d'autres commandes de la station au sol.
Durant cette opération, les drones doivent éviter les obstacles sur leur chemin, incluant les autres drones.
L'algorithme d'exploration de l'environnement n'a pas de contraintes. Cela peut être une séquence de mouvements aléatoires. Cependant l'exploration doit se faire de façon à pouvoir générer une carte du milieu exploré.
Les drones ne doivent pas décoller avec un niveau de batterie inférieur à 30%.

— Fin de mission :
Cette commande permet d'arrêter l'exploration. Après cette commande, les drones se mettre en position stable en attendant une autre commande de la station au sol.

— Retour à la base :
Le retour à la base doit rapprocher les drones de la station à terre afin qu'ils soient à moins de 1m. Le retour à la base et l'atterrissage doit être activé automatiquement dès que le niveau de batterie devient moins de 30%. Mise à jour logicielle :
L'interface doit permettre la mise à jour du logiciel sur les drones seulement lorsqu'ils sont au sol. La mise à jour doit être implémentée comme l'envoi d'un paquet binaire en utilisant l'API de BitCraze.

En dehors de ces commandes, le robot devrait toujours se mettre dans un état cohérent si un crash survient. La figure 1.2.3 montre un apercu des différents états du robot suivant les commandes reçues.

fréquence minimale de 1Hz :

1. Nombre des drones
2. État des drones (en attente, en mission, écrasé)
3. Vitesse courante des drones
4. Niveau de batterie des drones
5. Carte générée durant l'exploration
6. Position des drones
7. Éditeur pour le code du drone

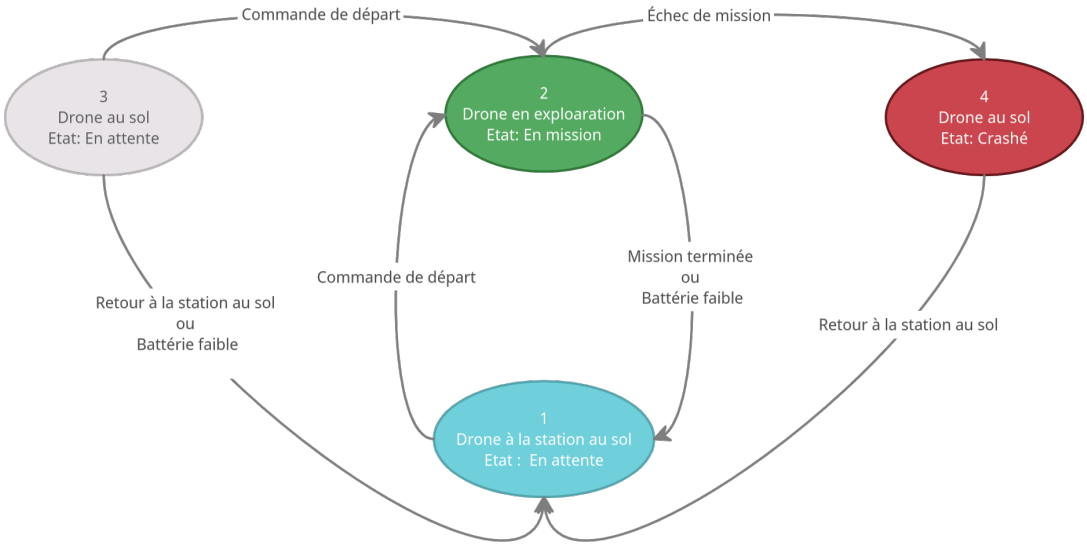

Figure 1 – Diagramme montrant les différents états et interactions possibles avec un drone

- ***// TODO : Comprendre ceci*** :
  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 [OPTIONNEL] en résolvant un problème d'optimisation permettant d'inférer la carte la plus plausible selon les données recueillies(Maximum Likelihood Estimation).

1.2.4 Contraintes de temps

La charge requise en termes d'heures pour la livraison du projet est de 630 heures-personnes. Plus de détails sont disponibles à la section 4.3.

Un version finale du projet sera livrée pour le lundi 19 avril 2021 en avant-midi.

1.3 Biens livrables du projet

Énumérer les artéfacts qui devront être créés durant le projet avec leurs dates prévues de publication.

3 briques : dashboard, serveur maître + base de données, firmware

- **Prototype minimal** : afin de démontrer au client la maîtrise des technologies utilisées (*date prévue 2021-02-15*) :
  - Interface web minimale avec boutons « Take Off » et « Land », ainsi que le niveau de batterie, la position et la vitesse du drone en action dans le simulateur Argos 3
  - Serveur maître basique faisant l'intermédiaire entre le simulateur et l'interface web : relais des messages de mise à jour temps réel des drones vers le client web, et transfert des commandes du client web vers les drones, le tout à l'aide d'une socket TCP. Pas encore de concept de base de donnée.

  - Version basique du micrologiciel : récupération des données de la batterie et de la vitesse, encodage sous forme `json` et envoie des données vers une socket TCP
  - Simulateur Argos faisant bouger deux drones
- **Prototype intermédiaire** : pour démontrer au client l'avancement du projet (*date prévue 20201-03-08*) :
  - Interface web avancée, avec boutons « TakeOff » et « Return to base » implémentés, et visualisation minimale des données captées par le drone
  - Simulateur avec 4 drones évoluant dans un environnement généré aléatoirement en évitant les obstacles
  - Serveur maître et micrologiciel avancés pour répondre aux besoins de la démonstration décrite par les points ci-dessus
- **Démonstration finale**, mettant en jeu :
  - L'interface web finale avec les boutons « TakeOff » et « Return to base » comme seules commandes pour lancer et terminer la mission, et affichage de toutes les informations des drones en temps réel
  - Le serveur maître final pour interfacer les commandes de l'interface web et traiter les données reçues des drones
  - Le tout sous la forme d'une vidéo pour montrer le comportement des drones dans un environnement qu'il ne connaît pas

2 Organisation du projet

2.1 Structure d'organisation

Décrire la structure d'organisation de l'équipe de projet et les différents rôles des membres. L'équipe est composée de 5 membres. Les 5 membres participeront au développement de la solution à tous les niveaux (client, serveur et micrologiciel), et seront en tout temps au courant de l'avancement des différents artéfacts du projet. Un membre de l'équipe, M. Samba Bal, assumera le rôle du gestionnaire de projet pour organiser le développement. GitLab sera utilisé pour gérer les tâches à effectuer, les assigner aux différents membres et comptabiliser le temps passé sur chacune d'elles.

2.2 Entente contractuelle

Décrire le type d'entente contractuelle proposée pour projet et les raisons de ce choix

Un contrat de livraison clef en main serait adéquat pour ce projet. En effet, le contracteur a une liste des requis complete et suffisamment précise pour ne pas avoir a la modifier grandement au cours du projet. Grace à cela, on peut prévoir le temps qu'il faudra pour réaliser le projet et ainsi prévoir un coup fixe pour la réalisation du produit.

3 Solution proposée

3.1 Architecture logicielle générale

Un diagramme qui résume l'architecture. Un texte qui décrit et justifie les choix. Inspiration : « L'ingénieur a parfois un peu peur de réaliser des choses parce que les moyens sont maintenant considérablement sophistiqués. On oublie que seulement prendre un papier et un crayon, décrire les choses, faire une esquisse, cela peut être aussi valable qu'un dessin d'ordinateur.

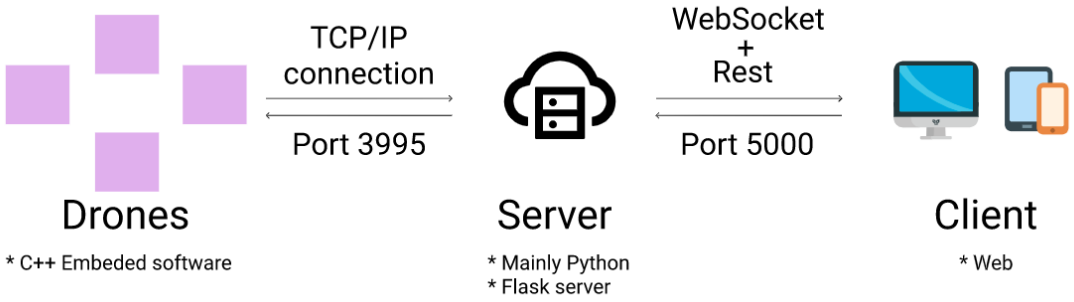

Figure 2 – Architecture globale de la solution

La solution que nous avons retenue fais état de 3 entités :

- Un client web : Il s'agit d'une application web utilisé par un opérateur. Il permet ainsi les interactions avec le logiciel des drones
- Un serveur maître : Il s'agit d'un serveur centrale qui joue le role d'intermédiaire entre le client et les drones. D'une part il transmettra les commandes du client aux drones. D'autre part, il remettra au client les résultats et informations venant des differents drones.
- Les drones : Un ou plusieurs drones qui communiquent entre eux mais également avec le serveur centrale (couple client-serveur). Ils effectuent les activités d'explorations.

Les requis nous demandent deux modules, à savoir, une station au sol qui est un ordinateur disposant d'une interface web et d'une partie embarqué pour les drones. Nous avons par la suite subdivisé la station au sol en deux éléments qui sont le client et le serveur.

3.2 Architecture logicielle embarqué

Un diagramme qui résume l'architecture. Un texte qui décrit et justifie les choix.

3.3 Architecture logicielle station au sol

Quelques blocs des principaux modules ou classes seulement. Des diagrammes sont nécessaires. Un texte qui décrit et justifie les choix.

4 Processus de gestion

4.1 Estimations des coûts du projet

Les divers frais rattachés au projet peuvent nous aider à estimer le coût global du projet.

La principale dépense n'est nulle autre que les ressources humaines. Avec quatre développeurs-analystes et un coordonnateur de projet à temps partiel, il est nécessaire d'avoir une estimation de temps pour en déduire le coût juste.

Pour le bien de la cause, nous disposons de 11 heures de travail par semaine et par développeur et 12 heures par semaine pour le coordonnateur. Sachant que le projet s'échelonne sur 11 semaines, nous obtenons un total de 484 heures pour les développeurs et 132 heures pour le coordonnateur.

Tâches	**Coordonateur [h]**	**Développeurs [h]**	**Total**
Description des exigences du système	10	20	30
Analyse des APIs (Bitcraze)	15	40	55
Analyse du simulateur Argos3		10	10
Analyse des services	10	35	45
Analyse de la base de données		10	10
Implémentation des services	25	110	135
Implémentation des interfaces	10	50	60
Tests des interfaces	8	10	18
Tests des services	8	10	18
Déploiement (Docker)	5	4	9
Tests finaux	6	10	16
Total [h]	97	309	**406h**
Total [$]	14'065	64'960	**79'025$**

Table 1 – Estimation des coûts du projet. L'équipe est constituée d'un coordonateur-développeur et de 4 développeurs-analystes.

En considérant un salaire respectif avoisinant 130$/h et 145$/h, le coût humain s'élève à priori à 79'025$. Nous disposons de deux drones équipés pour un total 800$. On alloue un 400$ additionnel qui couvre un possible bris de matériel. Le seul logiciel nécessaire est Argos3 et celui-ci ne nécessite aucune license d'utilisation.

L'estimation à priori s'élèvera donc à 80'225$.

4.2 Planification des tâches

4.2.1 *Preliminary Design Review*

Date de remise	15 février 2021	
Serveur WebSocket en Python	2 heures	Hubert Grootenboer
Construction de l'interface utilisateur avec Angular	4 heures	Eya-Tom A. Sangam
Client WebSocket dans le fureteur	3 heures	Eya-Tom A. Sangam
Serveur TCP en Python	3 heures	Hubert Grootenboer
Client TCP en C++ pour les drones	3 heures	Issam E. Maghni
Interprétation des commandes de décollage et d'atterrissage dans Argos3	2 heure	Mathurin Chritin
Ajout de containers Docker pour les trois modules	4 heures	Samba Bal

Table 2 – Planification du PDR

4.2.2 *Critical Design Review*

Date de remise	xx 2021	
Génération aléatoire de l'environnement sur Argos3	3 heures	Mathurin Chritin
Ajout de « Return to base » dans l'interface client	1 heure	Augustin Sangam
Lecture des capteurs et envoie de données	3 heures	Samba Bal
Communication inter-drones sur ARGoS	4 heures	Issam E. Maghni
Visualisation de la carte généré à partir de données brutes	3 heures	IDK

Table 3 – Planification du CDR

4.2.3 *Readiness Review*

Date de remise	xx 2021	
Documentation l'architecture et l'utilisation du système global	6 heures	Eya-Tom A. Sangam
Enregistrement du vidéo décrivant le fonctionnement de la simulation	2 heures	Issam E. Maghni
Enregistrement du vidéo décrivant le fonctionnement du système sur drone	2 heures	Samba Bal
Création d'un script permettant de lancer une simulation sur GNU+Linux	3 heures	Hubert Grootenboer
Monter la vidéo		Mathurin Chritin

Table 4 – Planification du RR

4.3 Calendrier de projet

Insérer un tableau qui indique les dates cibles de terminaison des phases importantes, des dates de version et autres jalons. Un résumé seulement.

Étape	**Date prévue**
Découverte de l'appel d'offre	2021-01-18
Réponse à l'appel d'offre	2021-02-15
Maîtrise du simulateur	xxxx-xx-xx
PDR	*2021-02-15*
Implémentation du	xxxx-xx-xx
Implémentation des interfaces	xxxx-xx-xx
Finalisation du serveur	xxxx-xx-xx
Version « alpha »	xxxx-xx-xx
CDR	*2021-03-08*
Base de donnée implémentée	xxxx-xx-xx
Version « beta »	xxxx-xx-xx
Finalisation du client	xxxx-xx-xx
Finalisation du firmware	xxxx-xx-xx
Version 1.0	xxxx-xx-xx
RR	*2021-04-19*

4.4 Ressources humaines du projet

Indiquer le nombre et le type de ressources humaines nécessaires, incluant les qualifications spéciales ou l'expérience des membres de l'équipe.

Dans le but de réaliser ce projet nous avons mobilisé 5 ressources. Nous avons un spécialiste du développement d'interface client web principalement Angular. Ce dernier travaille de pair avec un développeur Backend expérimenté. Deplus l'équipe dispose de deux développeurs de systême em-

barqué en C++ avec une expérience non négligeable avec les drones. Enfin il y a un spécialiste DevOps chargé de la gestion du projet.

5 Suivi de projet et contrôle

Compte tenu des 3 séances hebdomadaires programmées par l'agence spatiale de Polytechnique pour nous rencontrer, des rencontres entre 3 à 5 fois par semaine entre tous les membres de l'équipe sont prévues pour rendre compte de l'avancement de la solution. Des revues de code sont également programmées deux fois par mois, durant lesquelles un des membres de l'équipe devra sélectionner un fichier de code source et le soumettre aux autres membres pour qu'ils l'inspecte et en fassent ensemble une critique constructive. Cela bénéficiera tout autant à la qualité du code final qu'à la cohésion entre les membres de l'équipe.

5.1 Contrôle de la qualité

Tous les biens livrables doivent être soumis à un processus de révision. Une révision est requise afin de s'assurer, au moyen de lignes directrices et de listes de vérification, de la qualité de chaque bien livrable. À chaque livrable, une revue du code sera effectuée et l'ensemble des composants de notre application sera inspecté.

5.1.1 PDR

- Le simulateur est fonctionnel et se lance correctement : en appuyant sur play, les deux drones sont à l'arrêt, posés au sol et émettent des « pulse » vers le serveur maître
- Une fois le serveur maître lancé, il ne requiert plus aucune intervention humaine
- L'interface web est fonctionnelle et ergonomique. Elle présente bien le statut des deux drones en vol dans le simulateur et permet à l'opérateur de cliquer sur un bouton faisant décoller le drone, et sur un autre bouton permettant de le faire atterrir.
- Le bouton « TakeOff » a pour effet de faire décoller le drone dans le simulateur
- Le bouton « Land » a pour effet de faire atterrir le drone dans le simulateur
- Lorsque le drone est en vol, ses informations sont actualisées en temps réel dans l'interface web : niveau de batterie en pourcentage, position `x`, `y` et `z` et vitesse [km/h]
- Il est possible de faire voler plusieurs drones à la fois et de les contrôler ensemble depuis l'interface web
- Lorsqu'on relance plusieurs fois le simulateur, les drones de l'interface web sont actualisés en conséquence (les drones qui ne sont plus utilisés sont retirés, et les nouveaux drones de la nouvelle instance du simulateur sont ajoutés à leur place)

5.1.2 CDR

- L'environnement du simulateur est généré aléatoirement : des obstacles sont placés différemment à chaque lancement du simulateur

- Il y a 4 drones dans le simulateur qui volent de manière autonome
- Au niveau du panneau de contrôle sur la station au sol, les fonctionnalités « Take Off » et « Return to Base » sont implémentées :
  - Take Off : le drone décolle, et se met à parcourir la zone de manière autonome sans toucher d'obstacle, et sans intervention de la part de l'opérateur
  - Return To Base : le drone retourne tout seul à l'endroit où il a décollé, et ne requiert pas non plus d'intervention de la part de l'opérateur
- Les drones envoient les données du *ranging deck* au minimum une fois par seconde.
- L'interface utilisateur présente un prototype de visualisation de la pièce, sous la forme d'un graphe qui affiche des points représentant des mesures de distance faites par les drones.
- Lorsqu'un drone « crash » ou se crash, la mission en cours est annulée

5.1.3 RR

- L'application est complète et respecte chacun des requis fonctionnels, matériels, logiciels et de conception spécifiés dans le document des requis.
- L'application peut se lancer avec une simple ligne de commande sur Linux avec l'aide de Docker.

5.2 Gestion de risque

Par exemple : Lister les principaux risques de ce projet et estimer leur importance. Donner quelques solutions de remplacement possibles et la façon dont l'équipe entend gérer les changements en cours de projet.

Le présent document présente un plan organisé des délais et timings à respecter pour le bon déroulement du projet. Le principal risque que nous avons identifié est le non respect de ce planning. Il s'agit ici d'un risque à plusieurs échelles : le non respect d'une tâche assignée une semaine donnée peut être reportée à la semaine suivante sans grandes conséquences. Si ce report déclenche une réaction en chaîne, nous pourrions risquer de ne pas terminer le projet en temps et en heure, ce qui s'avérerait fatal pour le projet d'exploration sur Mars, impliquant des pertes d'argent conséquentes.

Il est donc important pour nous de bien respecter le planning. Pour cela 2 solutions s'offrent à nous : augmenter le nombre d'heure que nous passons sur l'artéfact qui pose problème, ou (dans un cas plus urgent) rayer un certain nombre de fonctionnalités non essentielles prévues dans notre planning.

Un second risque, moins probable, serait que l'un des membres de l'équipe se blesse lors d'une manipulation avec un drone et ne puisse de ce fait plus continuer à travailler sur le projet. Conformément aux consignes de sécurité fournies avec les drones, nous nous devons donc de porter en tout temps une paire de lunettes protectrice pour éviter tout problème qui pourrait nous pénaliser par la suite.

5.3 Tests

Identifier et préciser quelques tests pour chaque sous-système, tant pour le matériel que le logiciel. Il devrait y avoir un lien entre ces tests et les tâches décrites plus haut.

5.4 Gestion de configuration

Par exemple : Donner quelques renseignements sur le système de contrôle de version, l'organisation du code source, des tests et les fichiers de données ainsi que la documentation relative au code source et à la documentation de conception. La séparation et l'intégration entre les fichiers de description du logiciel

Le code source de la solution sera organisé dans un dépôt GitLab, comprenant 3 sous-*repos* :

- `drone`, comprenant tout le code C++ du micrologiciel embarqué ainsi que les fichiers de configuration du simulateur
- `server`, comprenant tout le code Python du serveur maître qui s'occupera d'interfacer les drones, la base de donnée et l'interface web et d'effectuer un traitement sur les données reçues
- `dashboard`, comprenant l'application Angular qui s'occupera de fournir une interface web à l'opérateur pour le contrôle et l'affichage des données des drones

La documentation du code sera faite avec Doxygen, en générant ainsi 3 PDFs, un par projet, et sera stocké dans les entrepôts associés. Quant à la documentation de la conception, ainsi que la documentation de la solution en générale, elles se trouveront toutes les deux à la racine du dépôt GitLab principal.

Conclusion

Bla bla bla

Références avec `printbibliography`.

ANNEXES

Inclure toute documentation supplémentaire utilisable par le lecteur. Ajouter ou référencer toute norme technique de projet ou plans applicables au projet.