

# Révision de l'état de préparation

«Readiness Review»

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

**INF3995 Projet de conception d'un syst. Informatique**

Équipe: 103

Nabil Dabouz, 1925256

Mazigh Ouanes, 1721035

Paul Clas, 1846912

Mohamed Yassir El Aoufir 1885972

Tarik Agday, 1851603

16 avril 2021

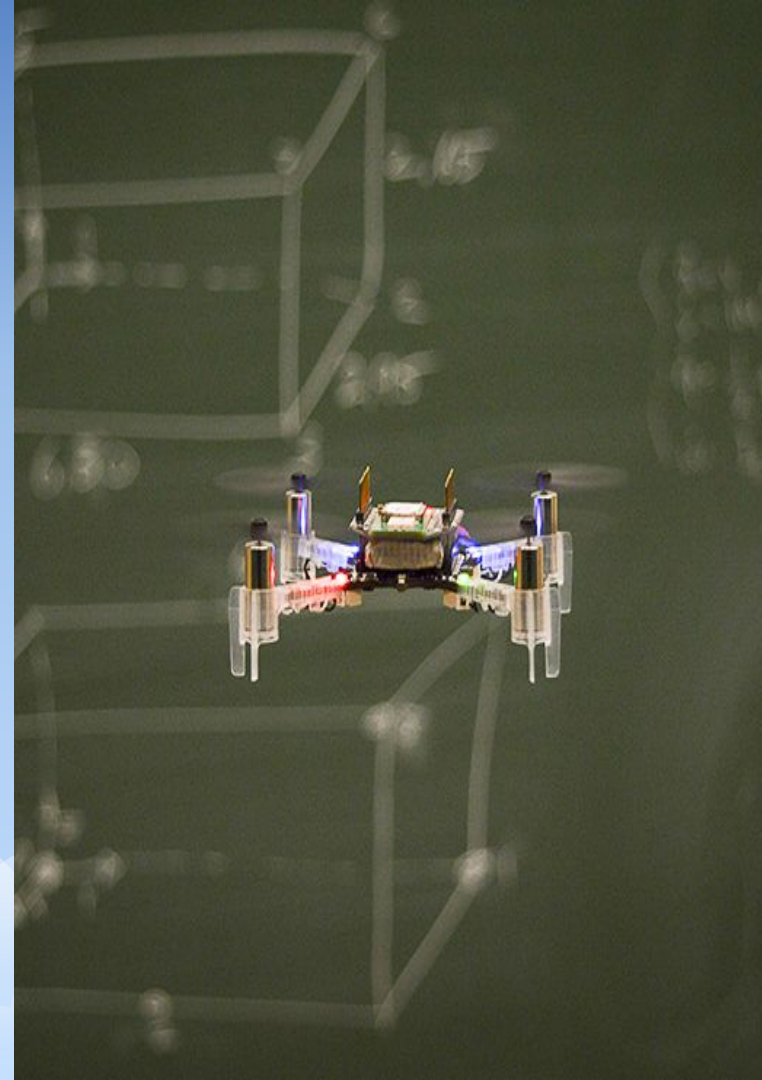


**POLYTECHNIQUE  
MONTRÉAL**

UNIVERSITÉ  
D'INGÉNIERIE

# Plan de la présentation

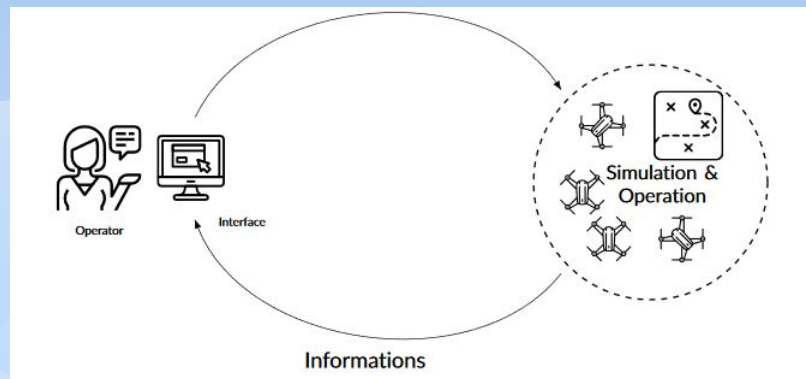
- Introduction.....3
- Équipe et gestion du projet.....4
- Architecture globale du système & Docker.....9
- Architecture du serveur.....11
- Environnement de simulation ARGoS.....14
- Crazyflie.....20
- Application web client & carte.....29
- Conclusion.....36
- Démonstration.....37
- Questions et réponses.....39
- Références.....40



# Introduction

L'objectif de notre projet est de répondre à l'appel d'offre de l'Agence Spatiale qui est de réaliser un système composé de plusieurs logiciels permettant à un essaim de drones miniatures, de type *Crazyflie 2.0*, d'explorer un environnement de taille moyenne tout en cartographiant l'environnement exploré sur une application web.

De plus, notre application web devrait permettre à son opérateur d'envoyer des commandes de contrôle bien définies aux drones.



# Partie 1

## Équipe de gestion de projet



# Organisation de l'équipe

- Utilisation de la méthodologie Agile. 10 Sprints.
- Utilisation de l'outil de gestion de versionnement GitLab.
- 4 rencontres hebdomadaires
- Plusieurs composantes de projet assignées à un (ou des) responsable(s)
- Utilisation de Discord. Communication accrue. Travail d'équipe.

**Tableau 1: Répartition des composantes du projet par responsable.s**

Composantes	Responsables:
Serveur Web	Yassir, Tarik
Interface Opérateur Client	Paul, Yassir
Pont Crazyflie/Web	Yassir, Nabil
Pont ARGoS/Web	Yassir, Nabil
Simulation ARGoS	Nabil, Paul
CrazyFlie Firmware	Yassir, Mazigh
Carte Crazyflie	Tarik, Paul
Navigation Autonome	Tarik, Paul, Nabil
Docker Compose	Yassir, Mazigh
Gestion du projet	Paul, Mazigh
Assurance Qualité	Mazigh, Tarik

# Heures et budgets

Afin de répondre à l'appel d'offre de manière efficace, nous avons choisi d'offrir une solution clef en main à l'Agence Spatiale. La solution que nous proposerons aura donc un prix ferme et final.

Tableau 2: Répartition budgétaire par rapport aux heures et au type de rôle de la main-d'œuvre

Rôle	Nombre d'heures (hr)	Taux horaire (\$/hr)	Budget (\$)	<b>CDR et PDR</b> (hr)	<b>RR</b> (hr) *incluant le <b>UXT</b>
Coordinateur de projet	90	145	13050	54	36
Développeur Analyste	540	130	70200	246	294
Total	<b>630</b>		<b>83250</b>		

## Légende:

- **PDR**: Révision de la conception du produit, «Product Design Review»
- **RR**: Révision de l'état de préparation, «Readiness Review»
- **CDR**: Révision de la conception du code, «Code Design Review»
- **UXT**: Test d'expérience utilisateur, «User eXperience Test»

# Calendrier et remise des livrables

Tableau 3: Échéancier global et répartition des heures des différents livrables du projet

	Sprint		1		2		3		4 et 5		6 et 7		8 et 9		10	
			Début	Fin	Début	Fin	Début	Fin	Début	Fin	Début	Fin	Début	Fin	Début	Fin
Livrables & Rencontres	Répartition des 630 heures (hr)	Allocation (%)	1 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								
UXT	110	17.46%										QR				
RR	184	29.21%														RR
PO	12	1.90%														PO
Rencontres	48	7.62%														

### Légende:

- **PDR:** Révision de la conception du produit, «Product Design Review»
- **CDR:** Révision de la conception du code, «Code Design Review»
- **UXT:** Test d'expérience utilisateur, «User eXperience Test»
- **RR:** Révision de l'état de préparation, «Readiness Review»
- **PO:** Aperçu de projet, «Project Overview»

# Assurance qualité

- Généralisation du format de codage. Respect des conventions de codage (indentation, style de commentaires, boucles ...)
- Respect des normes de taille et de complexité des classes.
- Organisation hiérarchique des différents composants logiciels.
- Couverture accrue par des tests unitaires du client et du serveur. Test de régression.
- Tests de fonctionnalité de ARGoS et du code embarqué du drone.

The logo for flask-unittest, featuring the text "flask-unittest" in white lowercase letters on a dark blue rectangular background.

Fig 1.1 : Test unitaire serveur

The header for the Jest + Enzyme logo, with the text "Jest + Enzyme" in white on a dark grey background.  
The Jest logo, a white icon of a person with a lightbulb above their head, on a blue background.  
The Enzyme logo, a white icon of a stylized 'A' shape, on a green background.

Fig 1.2 : Test unitaire client



# Partie 2

## Architecture globale du système && Docker





docker

Fig2.3: Docker

### Fig 2.1: Architecture globale et Docker

# Partie 3

## Architecture du serveur



# Architecture du serveur

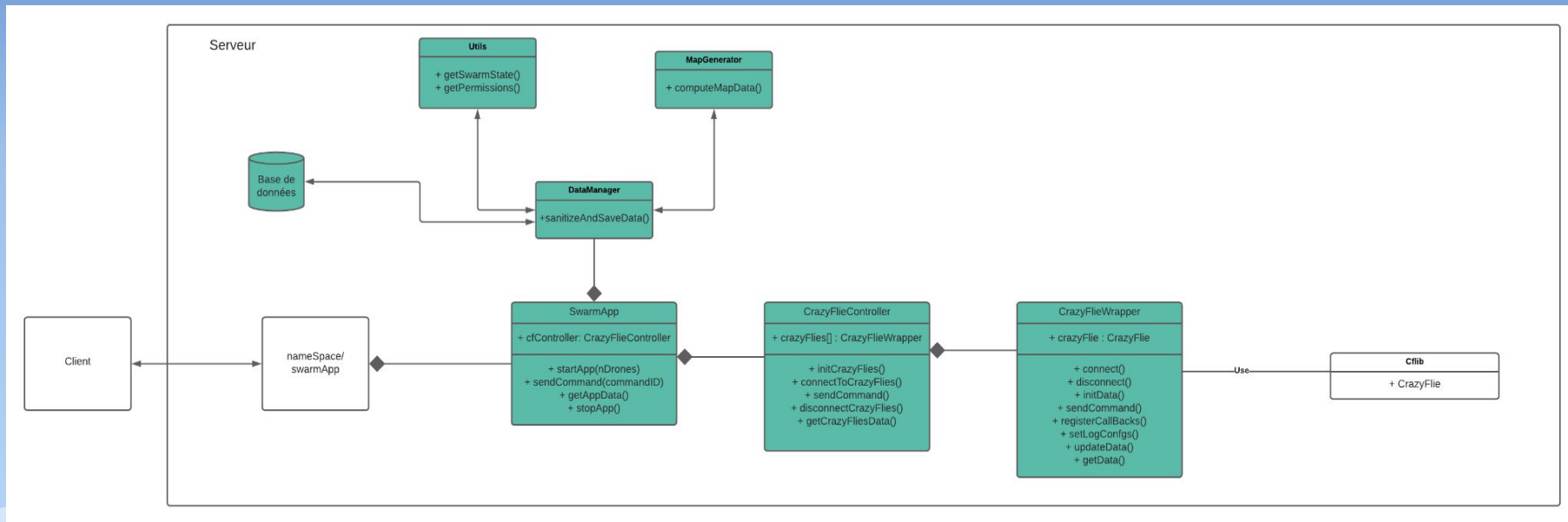


Fig 3.1: Architecture du serveur

- Explication de la logique et de l'architecture du serveur en prenant pour exemple «SwarmApp»
- Logique très similaire pour «SimulationApp»
- «DataManager» est utilisé par «SimulationApp» et «SwarmApp» de la même façon
- Serveur codé en Python 3.0

# Gestion des permissions

- Les permissions sont gérées au niveau du serveur
- Les permissions sont établies en fonction de l'état de l'essaim de drones

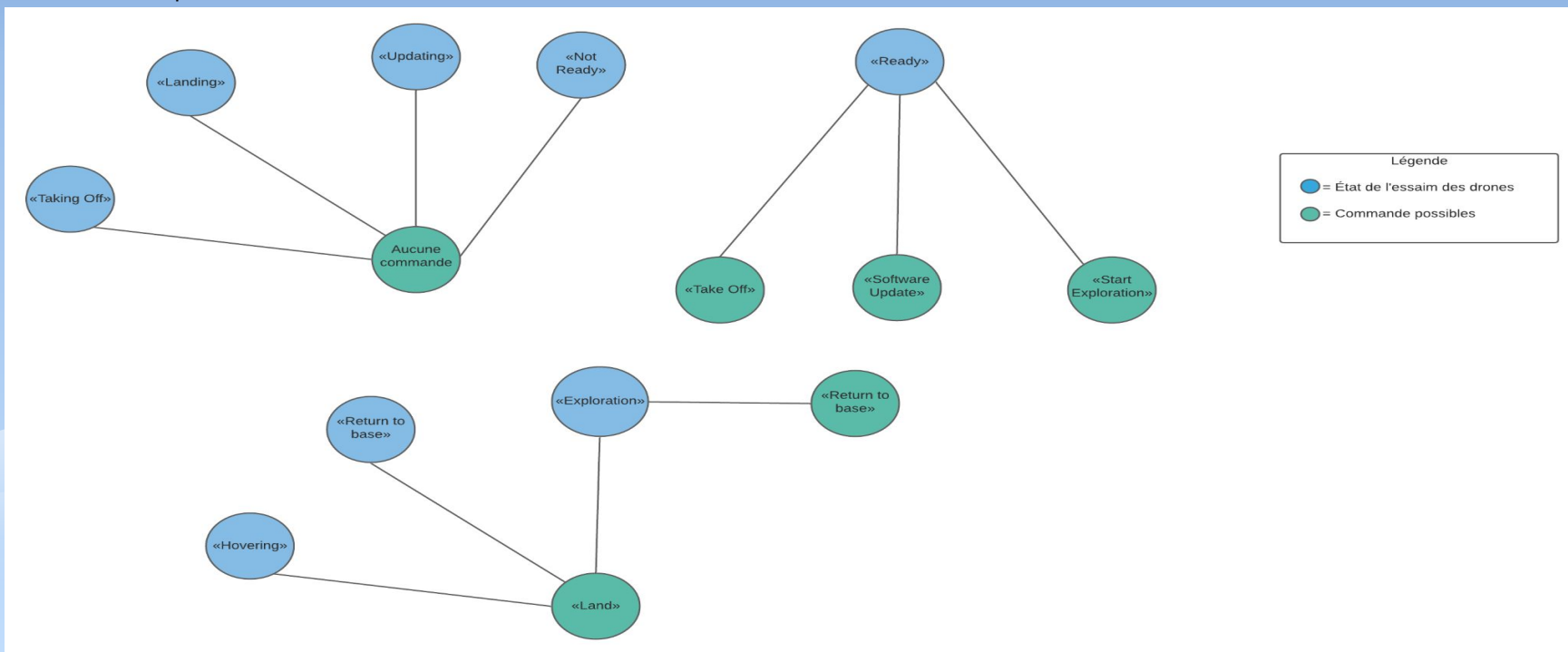


Fig 3.2: Gestion des permissions

# Partie 4

## Environnement de simulation ARGoS



# Environnement de simulation ARGoS

«ARGoS est un simulateur basé sur la physique et conçu pour simuler des essaims de robots à grande échelle. Les résultats de référence montrent qu'ARGoS peut effectuer une simulation physique précise impliquant des milliers de robots en une fraction de temps réel.» (MISTLab, 2020)

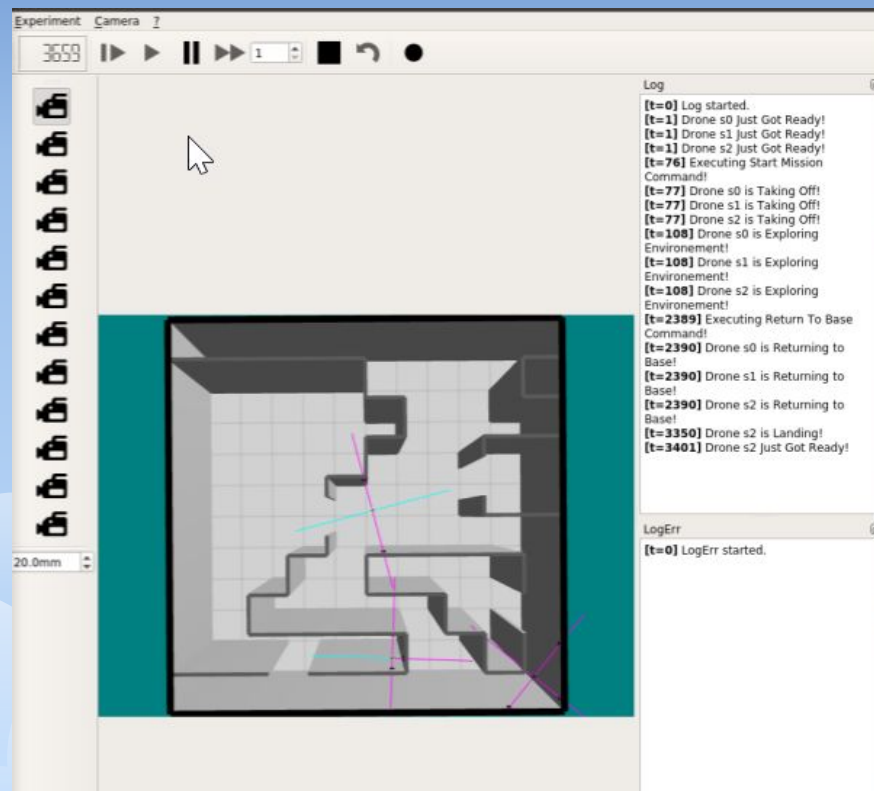


Fig 4.1: Simulation ARGoS

# Architecture du conteneur ARGoS

- Deux moyen de communication entre ARGoS et le serveur
- REST API pour lancer et arrêter l'application
- SocketIO pour l'échange des données et commande
- Le script «launch.sh» lance le générateur de murs et initialise le nombre de drones
- La communication entre le serveur et les drones se fait à travers «CrazyflieLoopFunction»

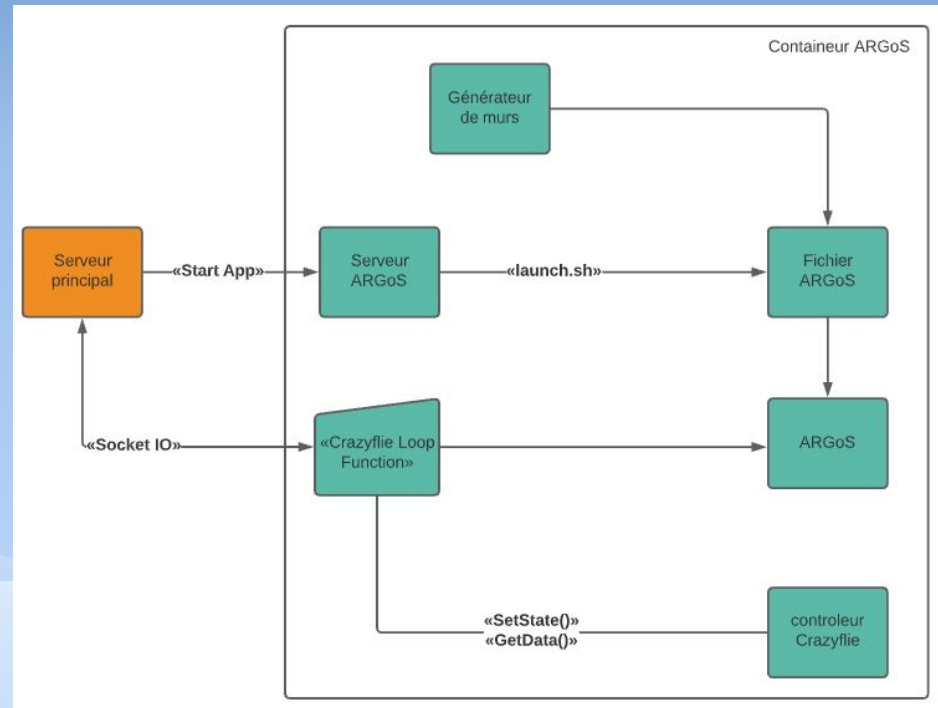


Fig 4.2: Architecture Conteneur ARGoS



# Architecture du Controlleur Crazyflie

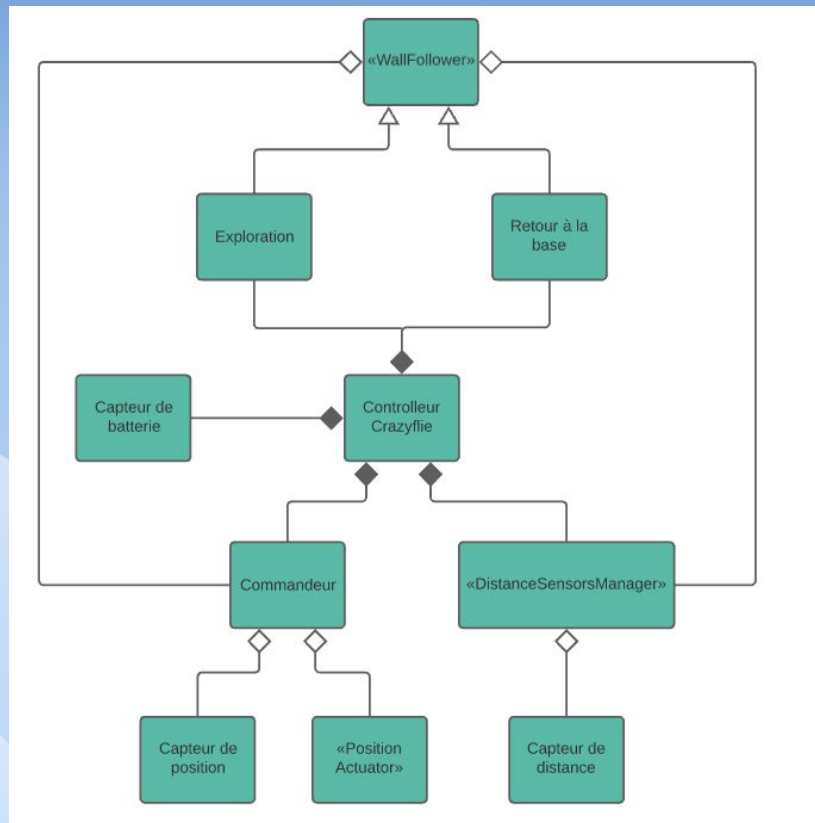


Fig 4.3: Architecture «Crazyflie Controller»

# Suiveur de murs (Classe Abstraite)

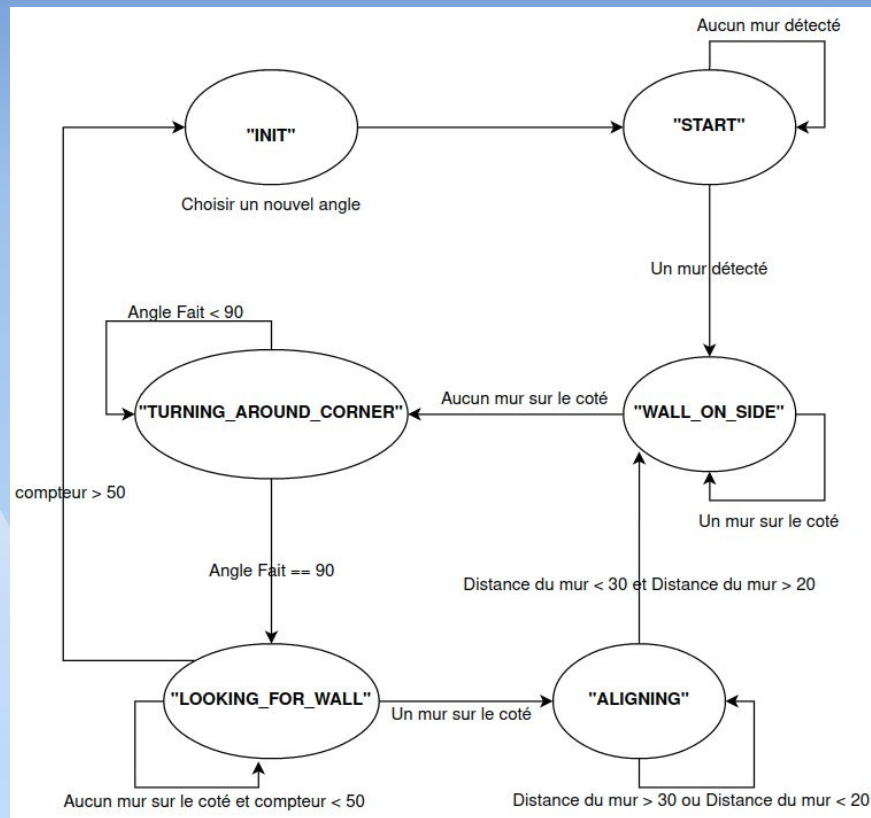


Fig 4.5: Machine à états du Suiveur de murs

# Générateur de murs (En C++)

1. Initialisation de la grille et les agents
2. Chaque agent décide de prendre une direction quelconque pendant N étapes pour construire les corridors
3. Vérifier que les corridors sont connectés
4. Traduction des corridors en murs verticaux et horizontaux
5. Ajouter des portes dans les murs
6. Importer dans le fichier «.argos»

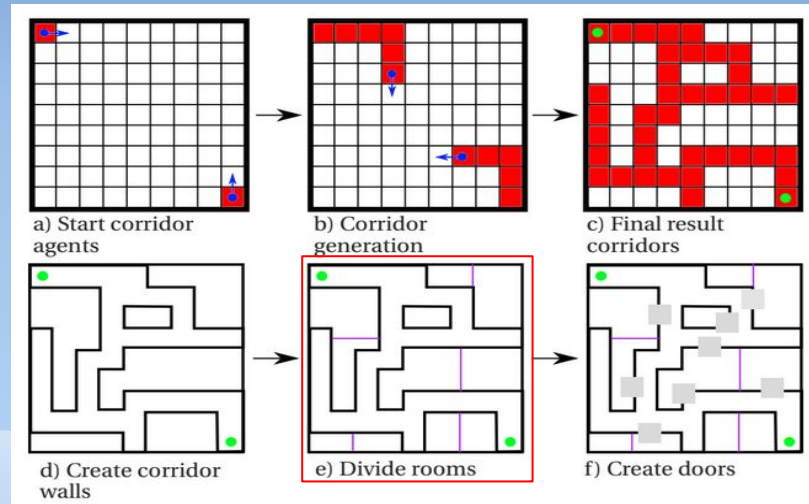


Fig 4.4: Algorithme de génération de murs

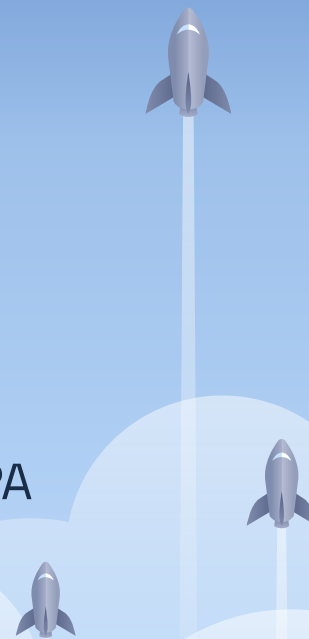
# Partie 5

## Crazyflie



# Présentation du système embarqué

1. Architecture et composantes
2. Test et intégration
3. Navigation autonome et exploration
4. Communication entre les drones et la radio PA
5. Apprentissages et améliorations futures



# Architecture

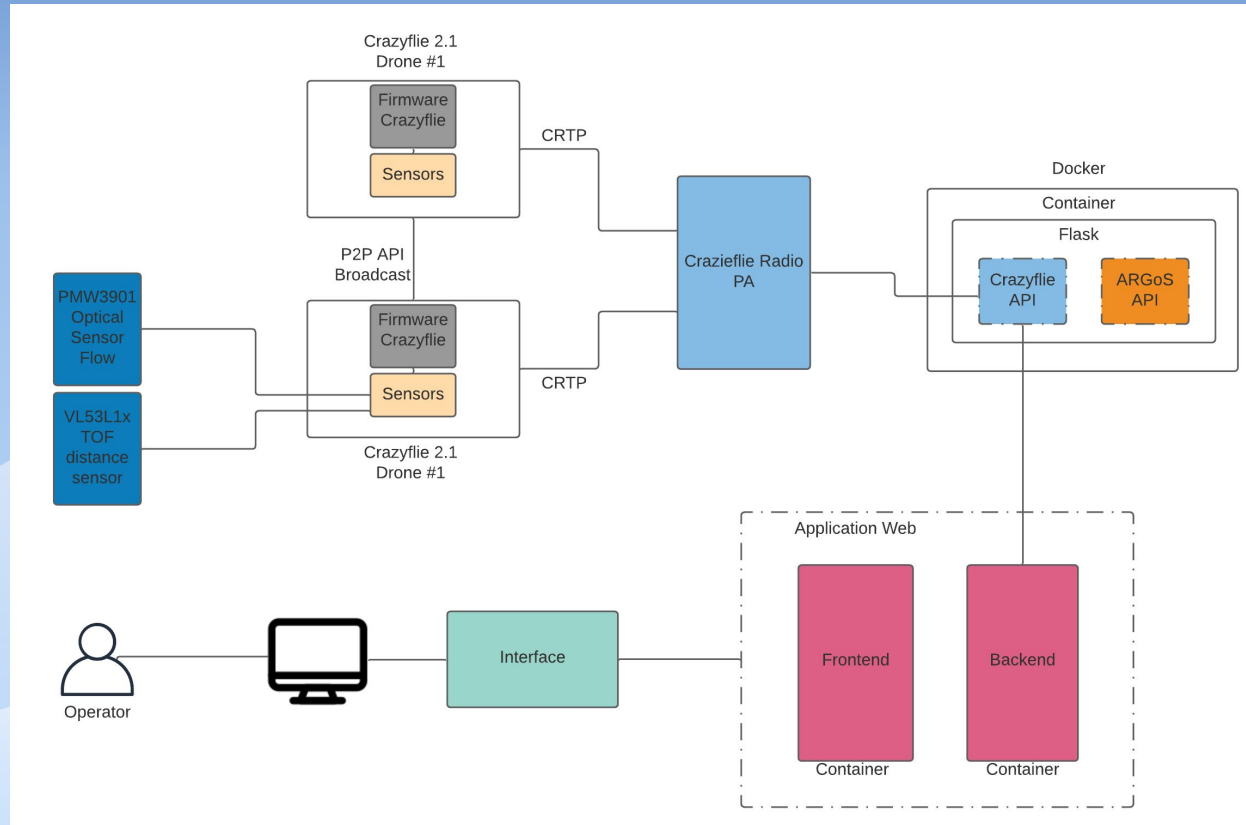


Fig 5.1: Architecture du système embarqué

# Composantes du Crazyflie

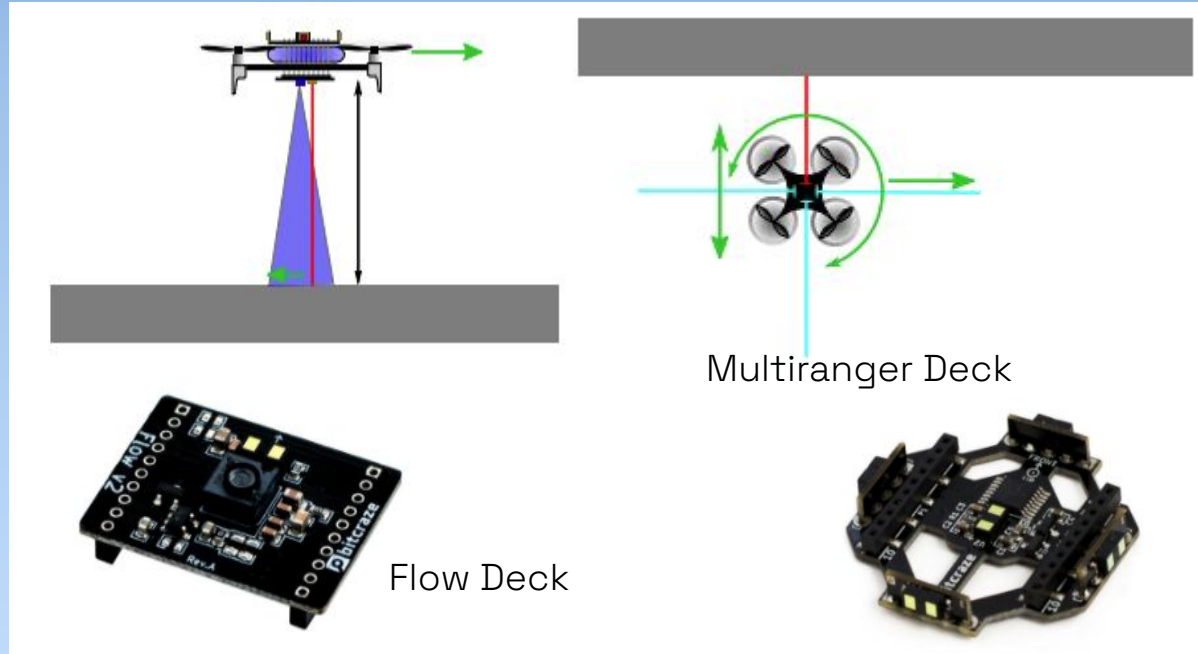


Fig 5.2: Composantes de navigation du Crazyflie

# Tests et intégration

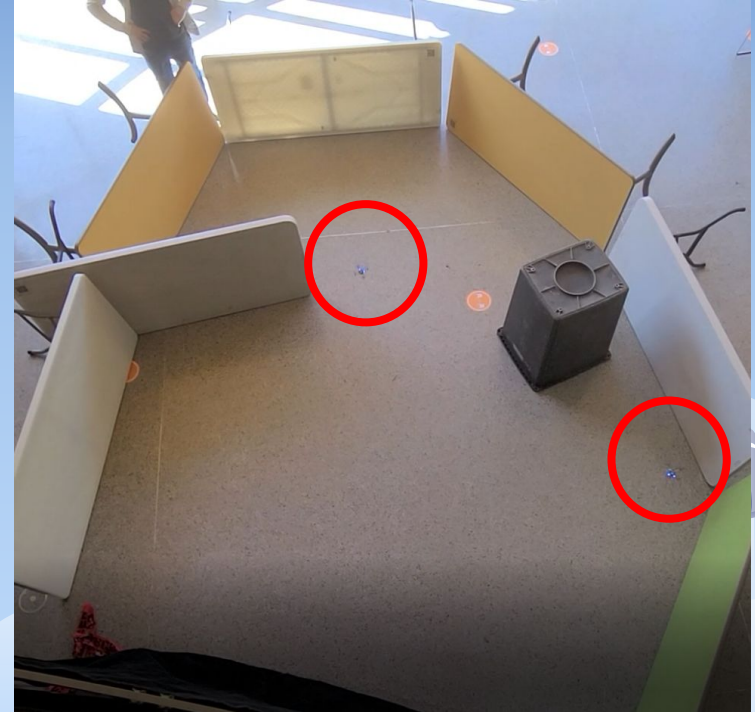
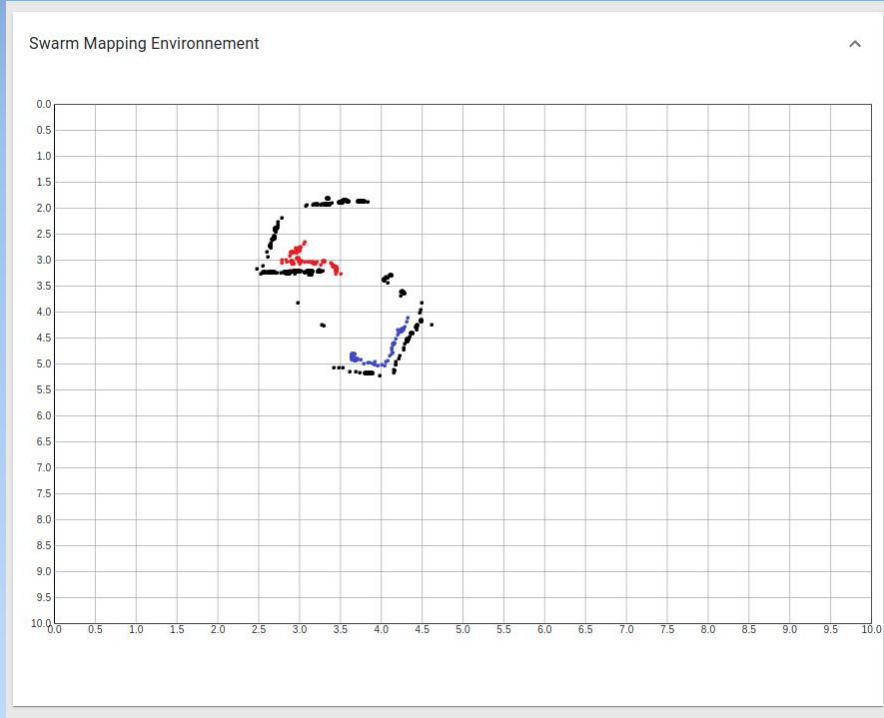


Fig 5.3: Visualisation de la carte générée lors de nos tests



# Navigation et exploration

« SGBA maximise la couverture d'espace en faisant voyager les robots dans différentes directions loin du point de départ. Les robots naviguent dans l'environnement et gèrent les obstacles statiques à la volée au moyen d'une odométrie visuelle et de comportements de suivi des murs. De plus, ils communiquent entre eux pour éviter les collisions et maximiser l'efficacité de la recherche. » (McGuire, 2019)

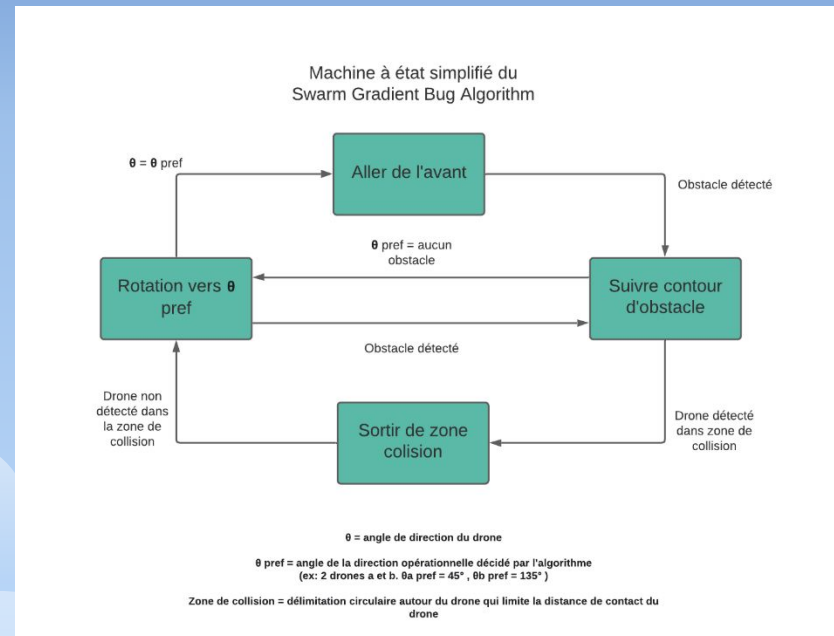
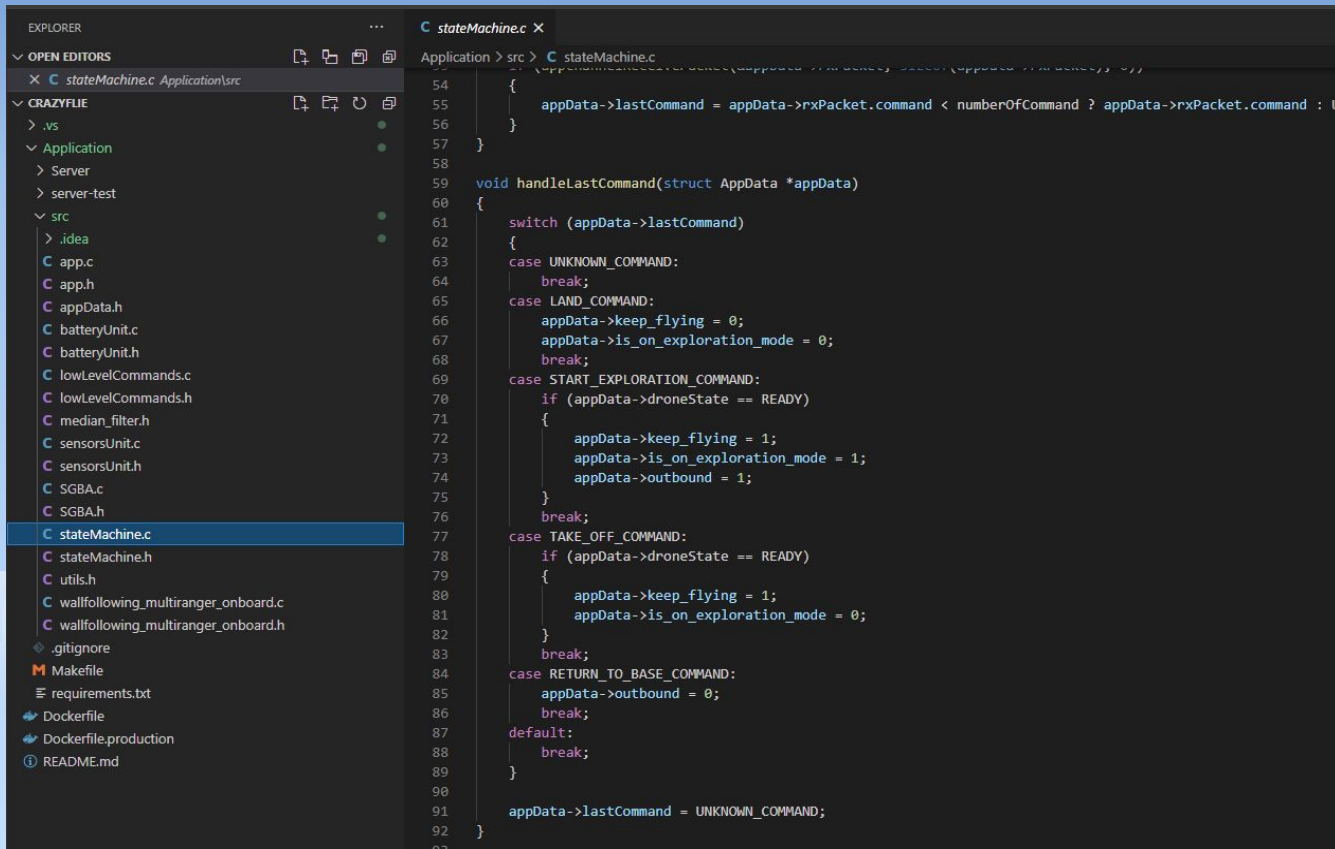


Fig 5.4: Algorithme d'exploration SGBA

# Code embarqué



```
54 {
55     appData->lastCommand = appData->rxPacket.command < numberOfCommand ? appData->rxPacket.command : UNKNOWN_COMMAND;
56 }
57
58
59 void handleLastCommand(struct AppData *appData)
60 {
61     switch (appData->lastCommand)
62     {
63     case UNKNOWN_COMMAND:
64         break;
65     case LAND_COMMAND:
66         appData->keep_flying = 0;
67         appData->is_on_exploration_mode = 0;
68         break;
69     case START_EXPLORATION_COMMAND:
70         if (appData->droneState == READY)
71         {
72             appData->keep_flying = 1;
73             appData->is_on_exploration_mode = 1;
74             appData->outbound = 1;
75         }
76         break;
77     case TAKE_OFF_COMMAND:
78         if (appData->droneState == READY)
79         {
80             appData->keep_flying = 1;
81             appData->is_on_exploration_mode = 0;
82         }
83         break;
84     case RETURN_TO_BASE_COMMAND:
85         appData->outbound = 0;
86         break;
87     default:
88         break;
89     }
90
91     appData->lastCommand = UNKNOWN_COMMAND;
92 }
93
```

Fig 5.4: Machine à état des commandes de navigation

# Communication



Fig 5.5 : Communication USB  
Radio PA



Fig 5.6: « Peer2Peer API » entre  
les drones Crazyflie

# Apprentissages et améliorations futures



Fig 5.6 : Nabil et Yassir travaillant sur le projet

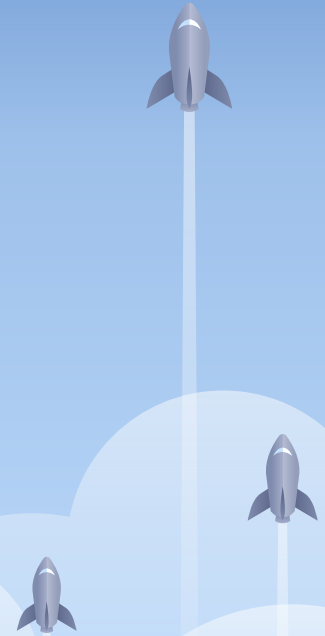
# Partie 6

## Application web client & carte



# Présentation du client

1. Technologies
2. Architecture web
3. Architecture client
4. Interface client
5. Carte



# Technologies utilisées



Fig 6.1 : React

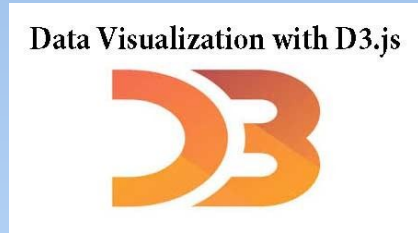
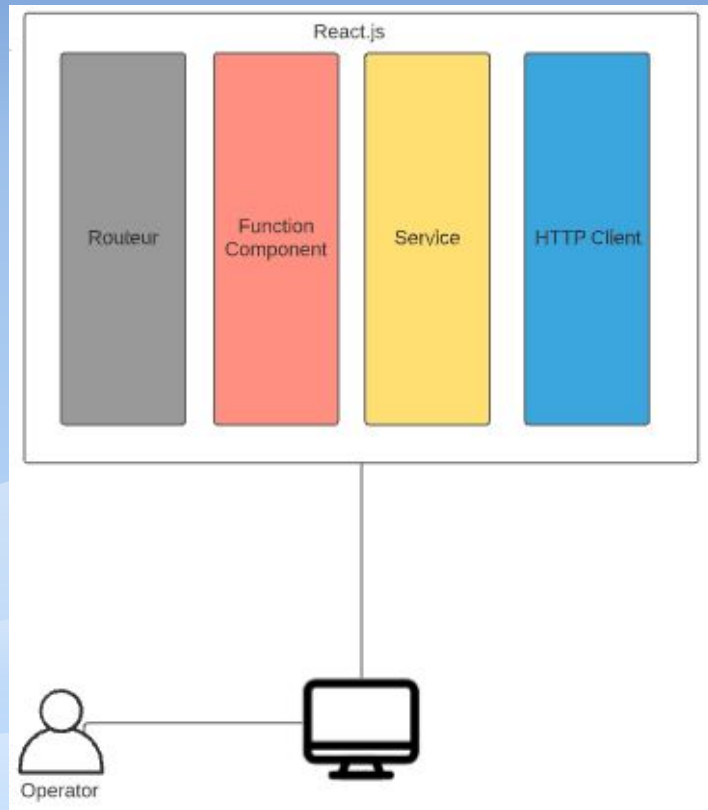


Fig 6.2 : D3

# Application web

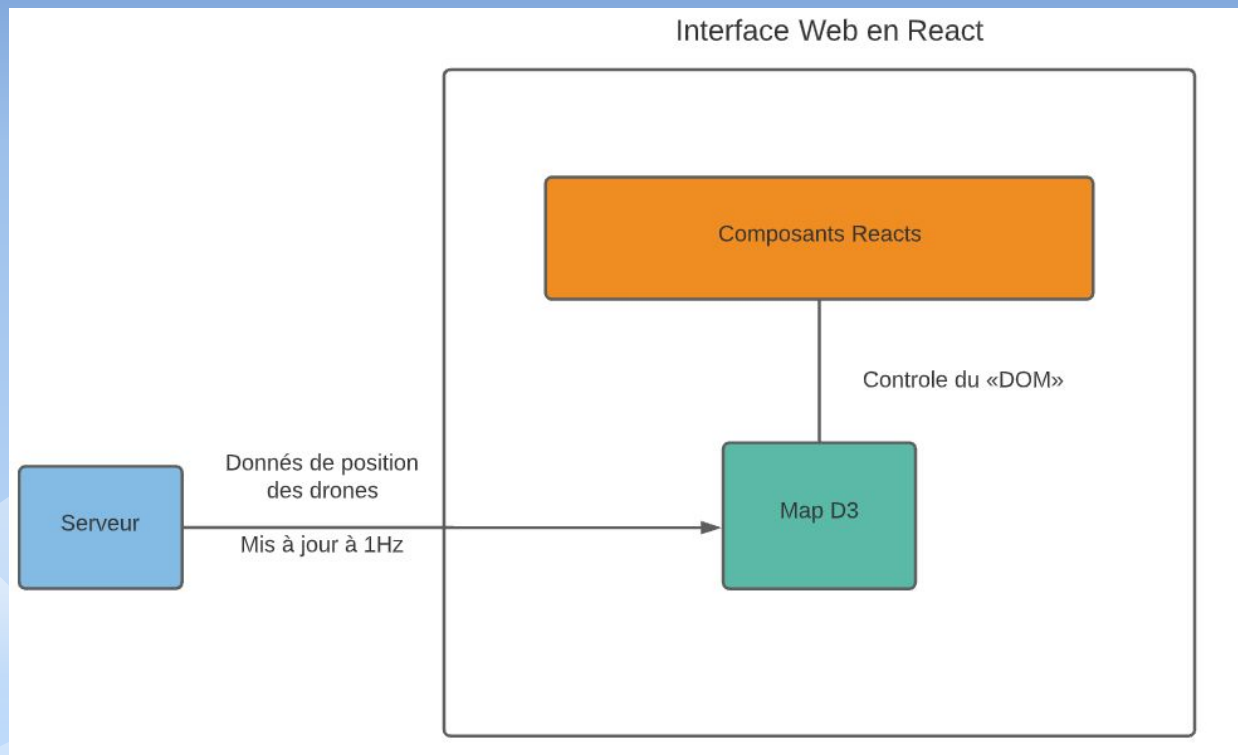


- Qu'est-ce que React?
- Les composants Reacts
- L'interface client

Fig 6.4 : Application web



# Architecture client



Description de  
l'architecture client

Intégration de la carte  
dans React

Réception des données  
des drones à chaque  
seconde

Fig 6.3 : Architecture client

# Interface client

Minimal Drone Swarm Navigation System

TutorialUser Manual

START MISSION

RETURN TO BASE

LAND

TAKE-OFF

SOFTWARE UPDATE

RESET APP

SCAN AGAIN

Select Number of Drones: 4

Connexion

Status

Server Connected

Crazyfile Swarm Application Not Ready

Swarm Status

NOT READY

Number of drones 4

Swarm Mapping Environnement

Drone List

Drone ID	Drone Status	Battery Level (%)	Speed
s1	READY	65	0.025885434734058983
s2	READY	70	0.026991300028066913
s3	DISCONNECTED		0
s4	DISCONNECTED		0

Swarm Console Log

```
{\"drones\":{\"s1\":  
  {\"positionX\":1.4221073389053345,\"positionY\":-0.15217438340187073,\"velocityX\":0.02582821436226368,\"velocityY\":0.0017201959853991866,\"state\":\"1\",\"batteryLevel\":65,\"frontX\":67.41778516997843,\"frontY\":-0.907493929824823,\"backX\":-214.56374738097023,\"backY\":2.3197804958005452,\"leftX\":9.47884916741687,\"leftY\":703.8017224813777,\"rightX\":-190.09517344187935,\"rightY\":195.799  
  {\"positionX\":1.5591517686843872,\"positionY\":0.07411293685436249,\"velocityX\":0.011967938393354416,\"velocityY\":-0.024192947894334793,\"state\":\"1\",\"batteryLevel\":70,\"frontX\":1877.9719907567364,\"frontY\":47.01937553915497,\"backX\":-257.3598281449573,\"backY\":-6.403683021587669,\"leftX\":-20.000153699180846,\"leftY\":861.8044630741484,\"rightX\":-76.69450410704242,\"rightY\":74.507;  
  {\"positionX\":null,\"positionY\":null,\"velocityX\":null,\"velocityY\":null,\"state\":\"8\",\"batteryLevel\":null,\"frontX\":1000,\"frontY\":1000,\"backX\":1000,\"backY\":1000,\"leftX\":1000,\"leftY\":1000,\"rightX\":1000,\"rightY\":1000;  
  {\"state\":\"0\",\"drones\":\"4\",\"permissions\":\"0\"}
```

- Survol des parties
- Explications des fonctionnalités

Fig 6.5 : Interface client

# Carte

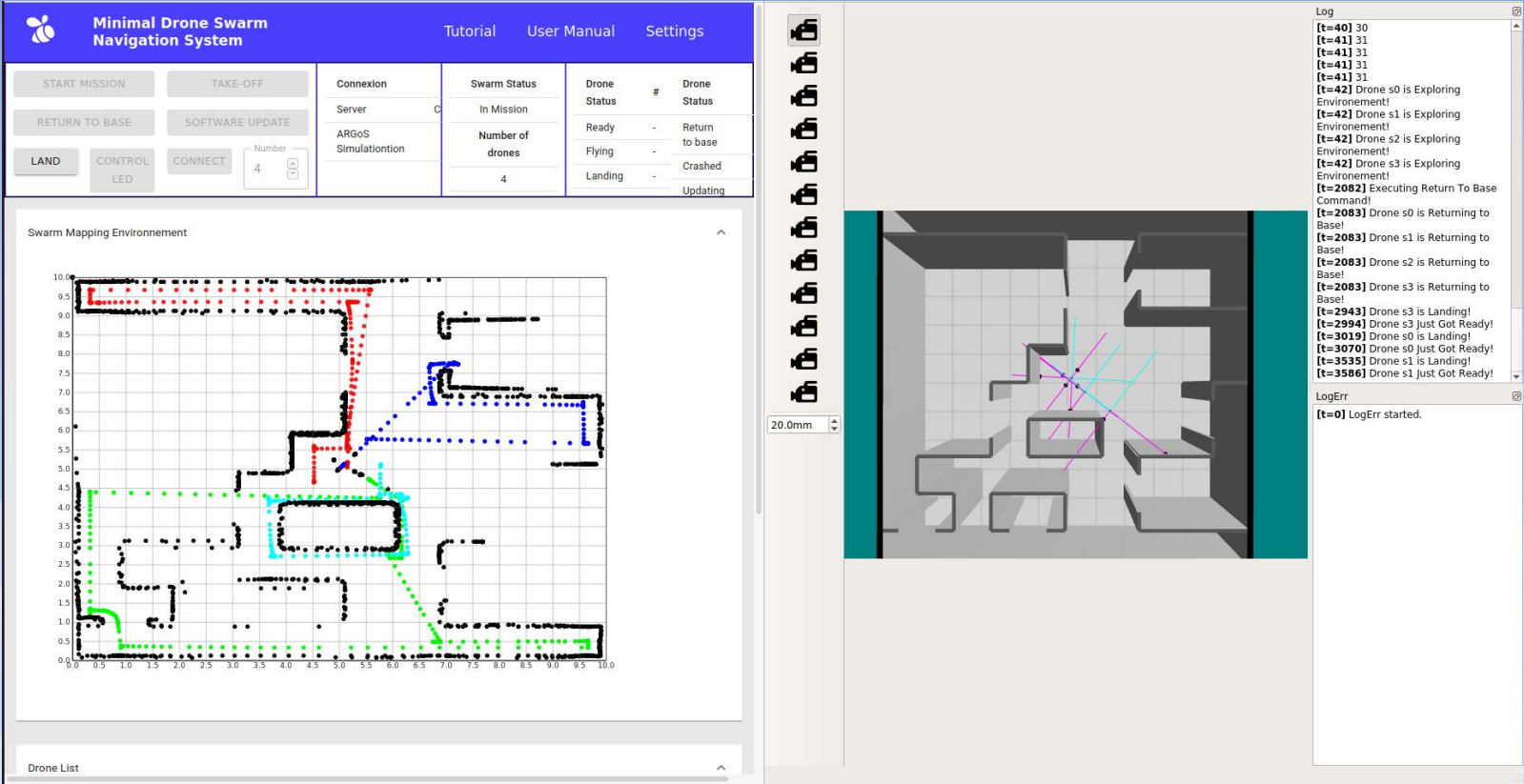


Fig 6.6 : Carte D3 et carte ARGoS

# Conclusion

Ce projet nous a permis de nous dépasser techniquement, ainsi que de manière organisationnelle et communicationnelle.

Nous sommes chanceux d'avoir eu l'opportunité de travailler sur le projet de **conception d'un système aérien minimal pour exploration** en tant que contractant.

Composantes du projet:

Équipe et gestion de projet  
Architecture global du système  
Serveur et Docker  
ARGoS  
Crazyflie  
Application web client et la carte

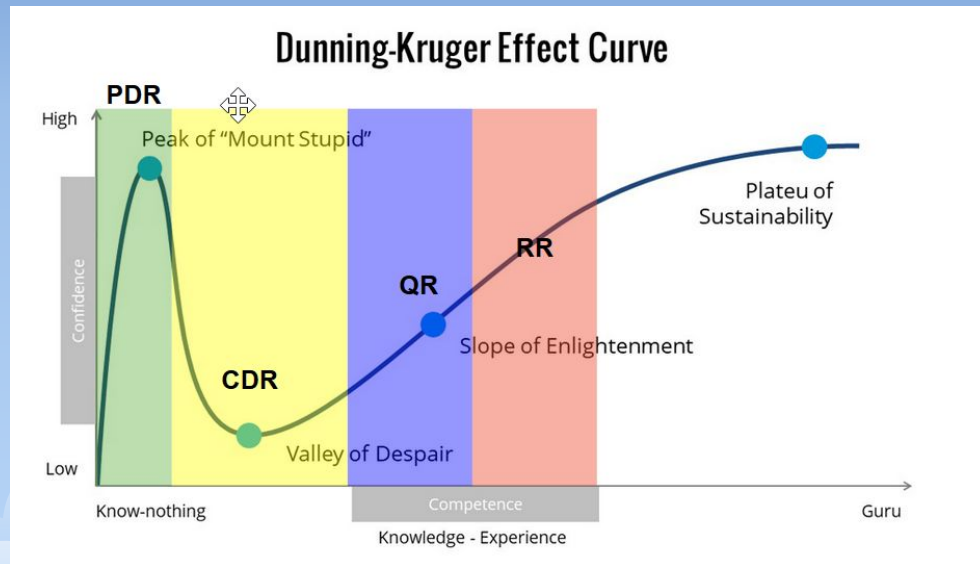


Fig 7 : Courbe d'apprentissage continue du projet de Dunning-Kruger.

# Partie 7

## Démonstration ARGoS et Crazyflie





# Merci

« L'ingénieur, c'est celui qui a la tête dans  
les nuages et les pieds sur terre. »  
– Bernard Lamarre, Po 52

# Questions & Réponses



# Références

MISTLab, ARGoS README (2019) Tiré de: <https://github.com/MISTLab/argos3>

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

Flask Documentation Release 1.1x. (n.d.). Tiré de <https://flask.palletsprojects.com/en/1.1.x/>

Jest et Enzyme test, tiré de <https://jestjs.io/docs/getting-started>

Flask, tiré de <https://flask.palletsprojects.com/en/1.1.x/>

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

ARGoS Simulator, tiré de [https://www.argos-sim.info/user\\_manual.php](https://www.argos-sim.info/user_manual.php)

Bitcraze webpage, Bitcraze. Tiré de [at:https://www.bitcraze.io/](https://www.bitcraze.io/)

React webpage, React tiré de <https://fr.reactjs.org/docs/getting-started.html>

D3 webpage, tiré de <https://d3js.org/>

Indoor Swarm Exploration with Pocket Drones, tiré de  
<https://www.semanticscholar.org/paper/Indoor-swarm-exploration-with-Pocket-Drones-McGuire/0de3345606b171e4b2bbe8f86f9b599ac37a3364>