

# Projet Njord

## Topographie d'une zone par communication d'une équipe de drones

Aigreault Clément, Henrio Jordan, Pham Chitin

16 Mars 2015

- Environnements inaccessibles par l'être humain
- Mission d'urgence, chantier...
- Besoin d'un intermédiaire

- Technologie à un stade intéressant
  - Robotique
  - Intelligence artificielle
  - Communication sans fil
- Possibilité de "donner vie" à des machines
- Une perte matérielle est moins importante qu'une perte humaine

- Élaboration d'un réseau étoilé
- Le noyau : un serveur
- Les branches : drones volants
- Les drones récoltent des informations
- Le serveur cartographie la zone

# Introduction

## Solution

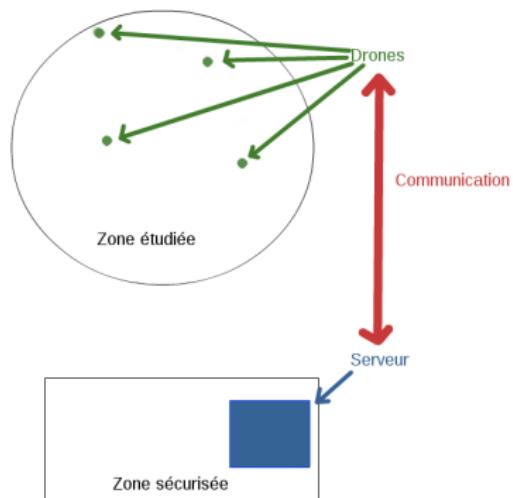


Figure : Schéma du réseau

- Drones autonomes et équipés d'un capteur ultrason
- Communication par fréquences radio
- Représentation graphique de la zone en temps réel

# Déroulement de la présentation

- 1 Serveur
- 2 Drone
- 3 Analyse
- 4 Conclusion

## 1 Serveur

- Principe
- Développement
- Démonstration

## 2 Drone

## 3 Analyse

## 4 Conclusion

- Traite les informations recoltées
- Chaque drone connaît le serveur, mais pas les autres drones
- Le serveur connaît tous les drones
- Possibilité d'envoyer des ordres

- Constitué de trois entités distinctes
  - Communication
  - Sauvegarde des données
  - Cartographie
- Concurrence
- Limiter la perte d'informations

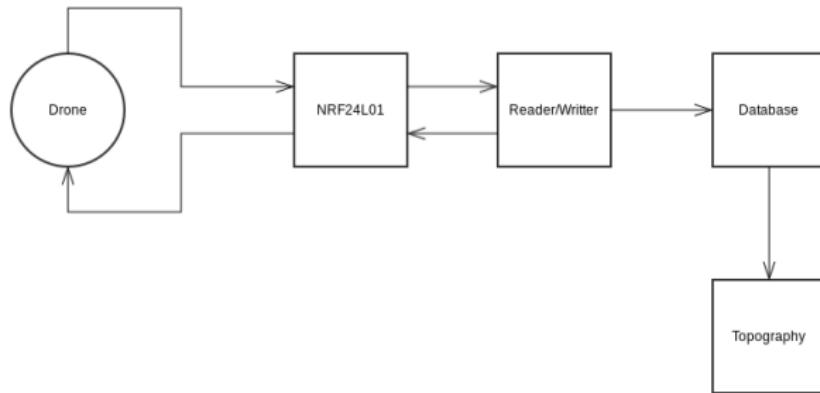


Figure : Modèle du serveur

## Tâche 1 : Communication

- Lit les messages des drones
- Écrit sur le port série de la machine
- Lit le port série
- Envoie des ordres aux drones

## Tâche 2 : Sauvegarde des données

- Lit le port série
- Insère les messages dans la base de données (BDD)
- Écrit sur le port série

## Tâche 3 : Cartographie

- Lit le contenu de la BDD
- Insère chaque entrée dans une matrice
- Dessine le contenu de la matrice
- Détermine si un ordre doit être envoyé

# Tâche 1 : Communication

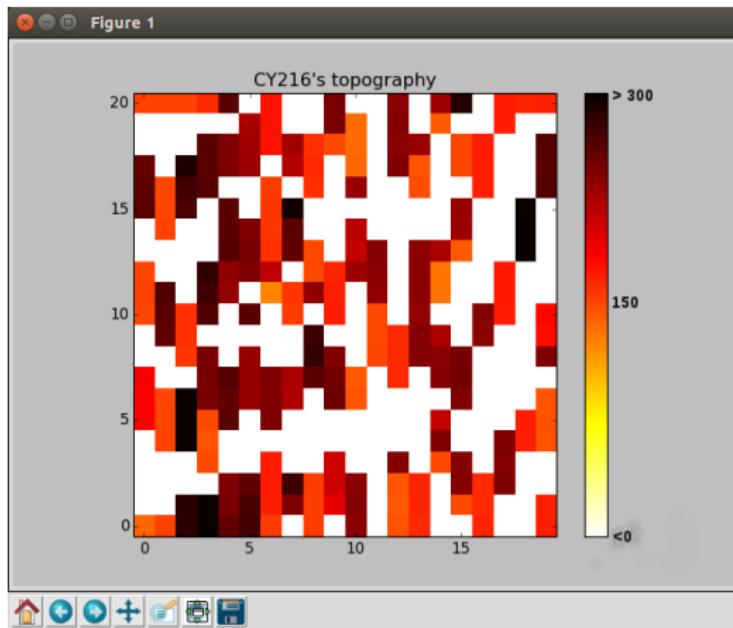
- Utilisation d'un NRF24L01 (composant)
- Implémentation en Arduino

## Tâche 2 : Sauvegarde des données

- Implémentation en Python
- BDD implémentée à l'aide de Redis
  - Utilisation simple (fonctionnement, Python)
  - Système robuste comme SQL inutile

## Tâche 3 : Cartographie

- Implémentation en Python
- Utilisation de Numpy et Matplotlib
- Matrice souvent redimensionnée



**Figure :** Exemple de rendu de topographie

## Démonstration

1 Serveur

2 Drone

- Étude préliminaire
- Composants
- Développement et aquisition des pièces manquantes
- Résultat final

3 Analyse

4 Conclusion

- Réalisation du cahier des charges
- État de l'art
- Plusieurs type de drones (grand/petit, rapide/lent, agile/prise de vues...)
- Étape non négligeable d'un projet

- Coup de coeur pour le Crazyflie de BitCraze
- Pièces détachées, code source disponible, libre
- Un peu cher et ne correspond pas parfaitement à notre application

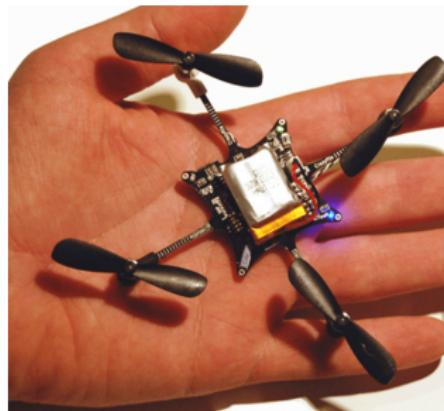


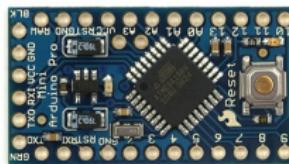
Figure : Photo du Crazyflie de BitCraze

- Devis
- Premières estimations
  - Poids
  - Puissance requise
  - Consommation en courant
  - Autonomie
- Plus lourd que le Crazyflie mais moins cher

# Arduino

- Cours d'Arduino en début d'année
- Micro-contrôleurs très en vogue
- Similaire au C/C++
- Bon marché et existe en petite taille

# Arduino



**Figure :** Photo de l'Arduino Pro Mini

# Gyroscope/Accéléromètre

- Besoin de stabiliser le drone
- MPU-6050 très répandu, petit, bon marché et embarque un accéléromètre
- Besoin de déterminer la position du drone dans l'espace
  - GPS : Trop peu précis pour une salle
  - Triangulation : Trop coûteux, peu adapté à la problématique
  - Mesure de l'accélération
- Solution non exploitable

# Gyroscope/Accéléromètre

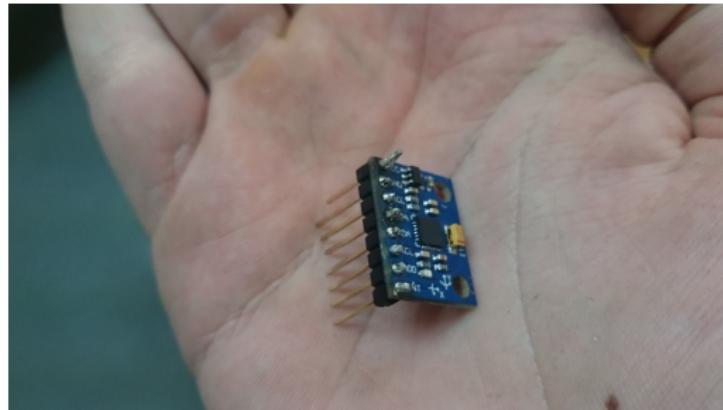


Figure : Photo du MPU-6050

# Communication sans fil

- Nombreux composants sur le marché
  - Modules XBee : Très célèbre mais trop coûteux ( $\sim 30 \text{ €}$ ) et trop gros
  - Fréquence radio : Célèbre, bon marché ( $\sim 0,80 \text{ €}$ ) et petit
- NRF24L01 suffisant au sein d'une même pièce

# Communication sans fil

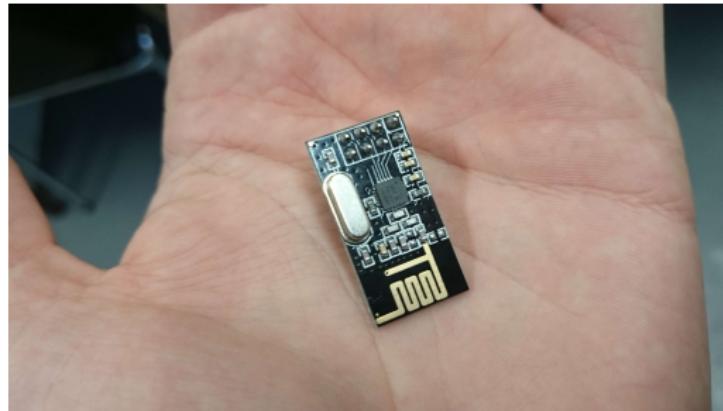


Figure : Photo du NRF24L01

# ESC

- Electric Speed Controller(ESC)
- Coûteux ( $\sim 15 \text{ €/ moteur}$ )
- Lourd ( $\sim 25\text{g} / \text{ESC}$ )
- Frabication possible (transistor)

## Moteur, batterie, total

- Même moteurs et batterie que le Crazyflie
- ~ 25 € contre 120 €
- 35g contre 19g (estimation)

- Développement de bibliothèques composant avant assemblage
  - Moteur
  - Communicateur radio : Surcouche de RadioHead
  - Gyroscope : Inspiration de Jeff Rowberg

- Encore deux pièces manquantes
  - Circuit imprimé
  - Fixations moteurs
- Pas de matériel à l'EISTI, besoin d'aide extérieur
- Visite au fablab de Gennevilliers

# Circuit imprimé

- Dessin du circuit à l'aide du logiciel Fritzing
- Support de la part de l'ENSEA

# Circuit imprimé

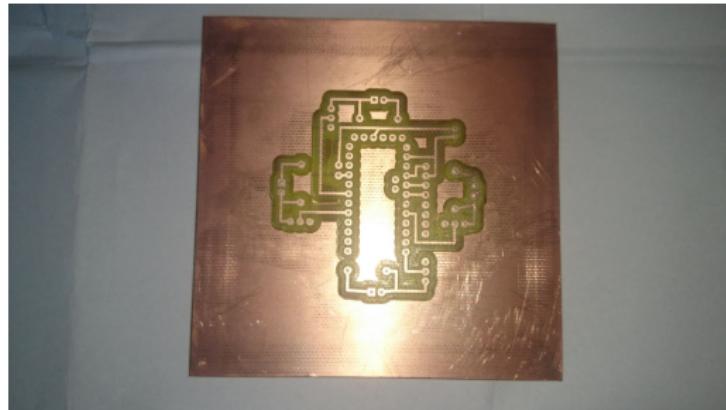


Figure : Photo du circuit

# Fixations moteur

- Modèle des pièces disponible sur GitHub
- Demande auprès d'une entreprise, mais pièces trop minutieuses
- Demande auprès de Polytech Instrumentation lors de Robafis

# Fixations moteur



Figure : Photo des fixations moteur

# Après assemblage

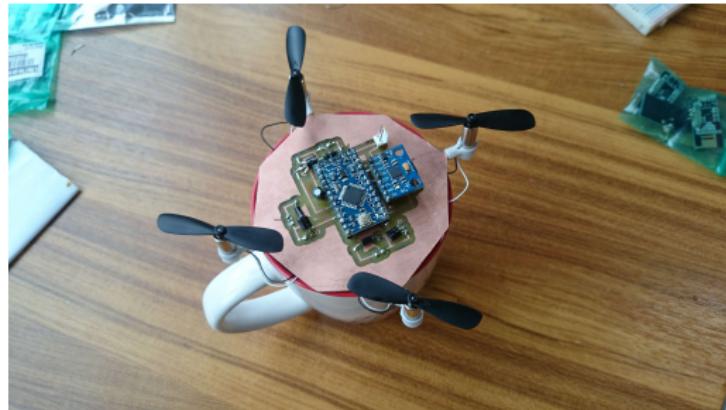


Figure : Photo du drone après assemblage

- Premier réflexe : Vérifier le décollage
- Plusieurs problèmes
  - Contrôle moteur depuis l'Arduino : Non suffisant, capacité maximum des moteurs non atteinte
  - By-pass de l'Arduino : Peut-être suffisant, un moteur défectueux

## Démonstration

## 1 Serveur

## 2 Drone

## 3 Analyse

- Conception
- Entités externes
- Expérience

## 4 Conclusion

Serveur  
Drone  
**Analyse**  
Conclusion

Conception  
Entités externes  
Expérience



Serveur  
Drone  
**Analyse**  
Conclusion

Conception  
Entités externes  
Expérience

- 1 Serveur
- 2 Drone
- 3 Analyse
- 4 Conclusion

