CAHIER DES CHARGES
PROJET "DRONE AUTOPILOT"

UE 5I456 - Ingénierie dirigée par les modèles

BENTAHAR Athmane
DIALLO Ousmane 3
HDADECH Nizar
PANGANIBAN Paul

Table des matières

1 Introduction

Ce document a pour but de présenter les fonctionnalités que nous allons implémenter pour le projet “DroneAutoPilot”. Dans cette partie nous expliquerons le contexte du projet et ses objectifs. Dans la partie suivante, nous prendrons le temps d’expliquer les fonctionnalités du drone sur lesquels se basent nos solutions. Nous expliquerons en deux parties notre solution pour répondre aux objectifs du projet. Nous présenterons ensuite les étapes clefs du projet, qui incluent pour la plupart un livrable.

1.1 Contexte

Grâce à la mobilité fluide et stable des drones, de nouvelles perspectives voient le jour dans de nombreux domaines. Dans le monde du commerce par exemple, avec l’entreprise Amazon qui a pour objectif d’utiliser les capacités de vol des drones, pour livrer des colis dans des lieux reculés. Dans le domaine militaire, pour désamorcer des bombes à distance ou les utiliser comme armes d’espionnages. Mais de manière plus populaire, nous les retrouvons aussi dans les domaines artistiques, comme la photo ou le cinéma pour prendre facilement de belles images en plongée.

Notre projet d’ “*Ingénierie Dirigée par les modèles*” fait aussi partie de cette branche artistique. Il a pour but d’associer la danse et les drones. Les drones ayant des mouvements précis et fluides, il est envisageable de les incorporer dans une chorégraphie innovante mêlant la technicité des artistes et des machines. Le ciel deviendrait alors le prolongement de la scène de danse qui sera rythmé par un ballet de drones.

1.2 Objectifs

L’objectif du projet est de pouvoir implémenter de manière simple une chorégraphie qui sera exécutée par le drone. Pour créer cette chorégraphie, nous allons créer un langage spécifique au domaine (DSL [1]) de la danse. Il sera simple à manipuler, grâce à l’utilisation de termes hauts niveaux et un niveau de configuration simple et suffisant.

2 Fonctionnement de base du drone

Dans cette partie nous expliquons le fonctionnement global du drone AR Parrot 2.0.

2.1 Fonctionnement des rotors

Le drone se compose de 4 moteurs (aussi appelés rotors), chaque paire de rotors (en diagonale) tourne dans le même sens (sens des aiguilles d’une montre et l’opposé). Les manoeuvres sont exécutées en variant la vitesse des quatres rotors, ainsi le drone exécutera des rotations, déplacements et inclinaison.

1. Domain Specific language

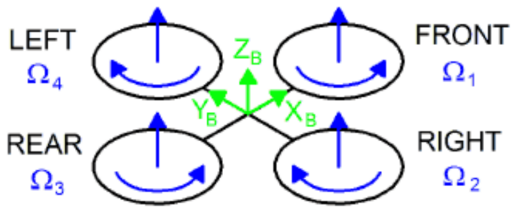

FIGURE 1 – Schéma drone (image provenant de *la Developer Guide SDK 2.0)*

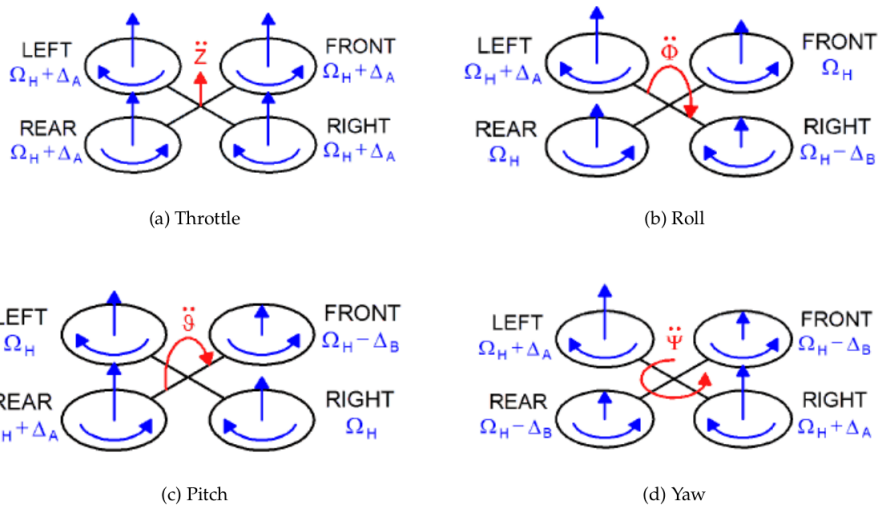

FIGURE 2 – Mouvement du drone (image provenant de *la Developer Guide SDK 2.0)*

2.2 Capteurs

L'aéronef contient plusieurs capteurs lui permettant de réguler sa configuration selon son environnement.

— 6 DOF (six degrees of freedom) : permet de calculer les degrés de rotation, d'inclinaison et de déplacement. Ce capteur est basé sur la technologie MEMS (Microelectromechanical), qui est une technologie décrivant des dispositifs microscopiques composés de pièces mobiles.
— Un télémètre ultrason permettant de détecter l'altitude du drone (n'est plus fiable à plus de 6 m).
— Une Caméra en dessous du drone, permettant de calculer la vitesse du drone.
— Un magnétomètre à trois axes, utilisé pour le pilotage en mode absolu c'est à dire que le drone réagit aux commandes de l'utilisateur selon la position de ce dernier. Cela ajoute aussi un capteur de pression permettant de calculer l'altitude du drone.

2.3 Navdata

Durant toute sa durée de fonctionnement, le drone (serveur) envoie au client une structure contenant la configuration sur une fonction particulière du drone. Cela permet à l'utilisateur de faire un choix sur la commande qu'il doit lui envoyer selon le statut de l'aéronef.

— Navdata de vol : état du drone (à terre, en vol, plane. . . .), voltage de la batterie, inclinaison, inclinaison en largeur, rotation sur lui-même, l'altitude, la vélocité sur les 3 axes (x,y,z), détection de pattern, etc. . . .
— Navdata des valeurs de capteurs en brute : accéléromètre, gyroscope...
— Navdata de pression : température, pression...
— Navdata magnetometer
— Navdata video
— Navdata vitesse du vent
— ...

3 Le langage de pilotage de drone

3.1 Description

Notre solution sera un langage avec une syntaxe textuelle. Ce langage sera généré à partir d'un modèle qui respectera un méta-modèle. Une extension de fichier spécifique au langage sera utilisée. Un système d'auto-complétion sera activé pour faciliter l'écriture d'une instance du langage.

3.2 Écriture

L'écriture de notre chorégraphie se fera en deux étapes distinctes : la configuration du drone et l'enchaînement de mouvements que le drone exécutera.

La partie configuration sera l'en-tête du fichier, elle contiendra les limites que nous fixons au drone (altitude maximale, vitesse verticale maximale, vitesse de rotation maximale, etc...). Ces contraintes permettent d'encadrer le fonctionnement du drone dans un espace sûr et maîtrisé.

L'enchaînement de mouvements du drone est la partie chorégraphie à proprement parlé. La chorégraphie sera représentée par une suite d'actions séparées par un délimiteur. La liste des actions disponibles et leurs paramètres sont listés dans la partie "commandes à implémenter". Une chorégraphie commencera forcément par un décollage et finira par un atterrissage. Avant chaque décollage, le drone fera automatiquement des vérifications sur sa position et son niveau d'inclinaison.

3.3 Déploiement

Le drone sera pilotable à partir d'un ordinateur/serveur ayant une connexion réseau Wi-Fi. Des fichiers et documents aideront pour son déploiement.

4 Commandes à implémenter

Dans cette partie nous listerons l'ensemble des commandes possibles pour composer une chorégraphie . Chaque commande sera de la forme nom_commande(param1 : val1, ...).

4.1 Décollage

Le décollage est la première action faite pour une chorégraphie. Il sera possible d'effectuer plusieurs décollages si la chorégraphie implique plusieurs atterrissages. Avant chaque décollage, le drone fera automatiquement des vérifications sur sa position et son niveau d'inclinaison. Le drone décollera jusqu'à une hauteur par défaut.

Syntaxe : *Decollage()*

4.2 Atterrissage

Il est possible d'effectuer plusieurs atterrissages lors d'une chorégraphie. Cependant une chorégraphie doit forcément terminer par un atterrissage.

Syntaxe : *Atterissage()*

4.3 Déplacement simple

Le déplacement simple du drone consiste à aller vers la gauche ou la droite, vers l'avant ou l'arrière, vers le haut ou le bas. La plupart des déplacements possèdent comme paramètre : la distance qui sera parcourue, l'angle d'inclinaison et le temps pendant lequel le drone doit effectuer cette action.

Syntaxe :

— *Gauche(distance : réel, inclinaison : entier, temps : réel)*
— *Droite(distance : réel, inclinaison : entier, temps : réel)*
— *Avant(distance : réel, inclinaison : entier, temps : réel)*
— *Arriere(distance : réel, inclinaison : entier, temps : réel)*
— *Haut(distance : réel, temps : réel)*
— *Bas(distance : réel, temps : réel)*

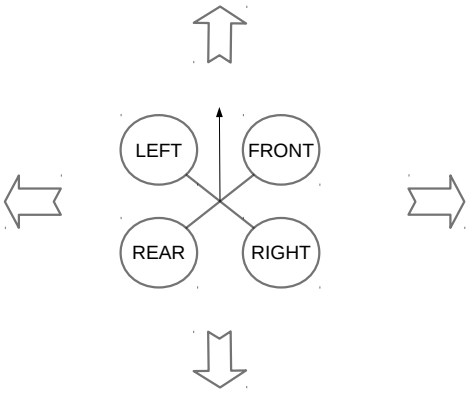

FIGURE 3 – Déplacements simples

4.4 Rotation

Le drone doit pouvoir effectuer une rotation sur lui même à l'aide des paramètres *angle* de rotation et *temps* d'exécution.

Syntaxe : *Rotation(angle : entier, temps : réel)*

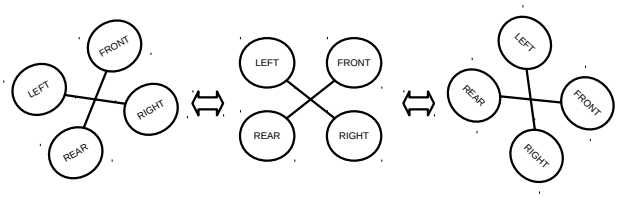

FIGURE 4 – Rotation

4.5 Patterns prédéfinis

Nous essayerons d'implémenter quelques patterns que l'on va définir : cercle, carré horizontal, carré vertical, spirale carrée, flip.

4.5.1 Cercle

Le cercle sera construit à l'aide du paramètre *rayon* et un *temps* pendant lequel il sera exécuté par notre drone.

Format : *Cercle(rayon : entier, temps : réel)*

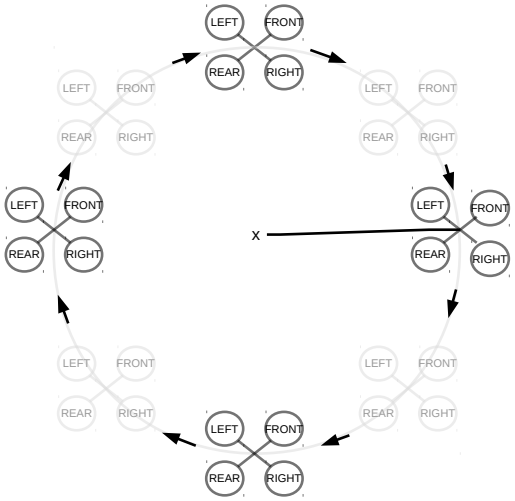

FIGURE 5 – Cercle

4.5.2 Carré horizontal

Le drone effectuera un mouvement de carré horizontal en un temps t en fonction de la *longueur* (en mètre) d'un coté.

Format : *CarreHorizontal(longueur : entier, temps : réel)*

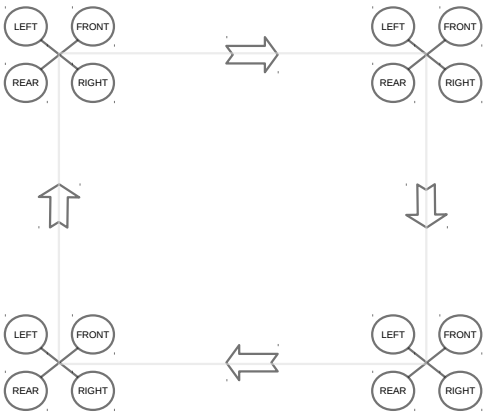

FIGURE 6 – Carré horizontal

4.5.3 Carré vertical

Le drone effectuera un mouvement de carré vertical en un temps t en fonction de la longueur (en mètre) d'un coté.

Format : *CarreVertical(longueur : entier, temps : réel)*

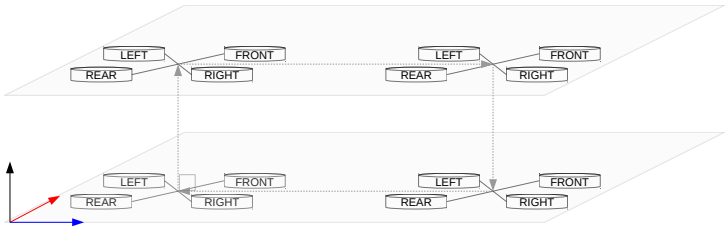

FIGURE 7 – Carré vertical

4.5.4 Spirale carré

Le drone pourra effectuer un mouvement de spiral carrée en fonction de la longueur des côtés du carré et de la hauteur à parcourir pendant un temps défini.

Format : *SpiralCarree(longueur : entier, hauteur : entier, temps : réel)*

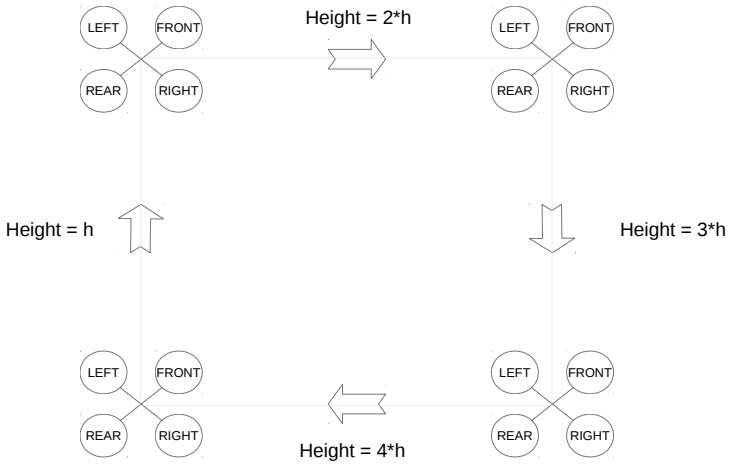

FIGURE 8 – Spirale carré

4.5.5 Flip

Le drone doit pouvoir faire un flip, dans la direction voulue.

Format : *Flip(direction : Gauche|Droite|Avant|Arriere)*

4.6 Combinaison de déplacements

Il est aussi intéressant de combiner certaines commandes pour effectuer des actions complexes (courbe, spirale montante, spirale descendante,etc)

4.6.1 Courbe

Pour le mouvement de courbe, on fournira une direction et une distance à parcourir pendant le tracé.

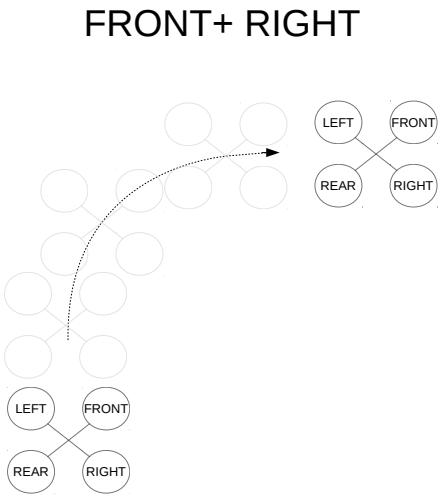

FIGURE 9 – Courbe

4.6.2 Spirale montante

La spirale montante de notre drone sera construite en fonction du rayon, de la hauteur à parcourir et d'un temps d'exécution.

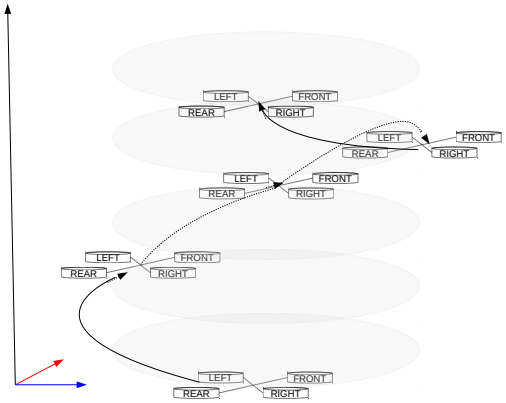

FIGURE 10 – Spirale montante

4.6.3 Spirale descendante

De même que la spirale montante, notre drone pourra faire une spirale descendante.

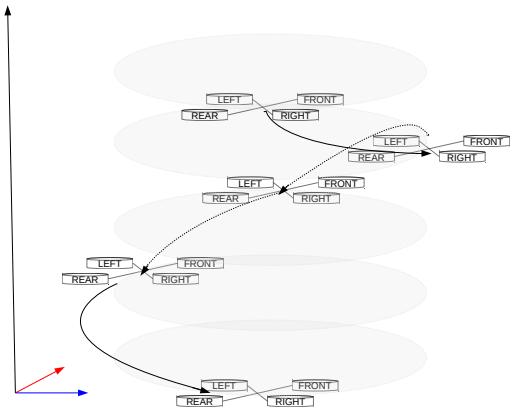

FIGURE 11 – Spirale descendante

4.7 Gestion des LED

Il sera aussi possible de modifier les couleurs des LED (selon les couleurs acceptées par le drone).

5 Étapes du projet

Nous mentionnerons dans cette partie l'ensemble des étapes clefs du projet qui seront synonyme de date livrable à fournir.

5.1 Tranche 0 - 17/10/16

Comme tout projet informatique, la première pierre sur laquelle on se base est le cahier des charges. On y retrouve l'analyse des besoins et les modalités de réalisation où elles sont détaillées. Le cahier des charges sera notre référence pendant toute la durée d'élaboration de l'application et même le jour de la livraison.

5.2 Tranche 1 - 27/10/16

— Méta-modèle et langage : La construction du DSL (Domain Specific Language) dédié à la chorégraphie, pour un scénario de pilotage du drone va se baser sur l'API Parrot du drone avec un niveau plus haut qui est l'utilisation du langage naturel. L'utilisation des verbes issus du langage commun (Avancer, Tourner, Pivoter... etc.) nous permettra de faire faire au drone des figures avec des appels aux fonctions de base décrites dans la section.

— Construire un éditeur : La phase de modélisation nous permettra de générer un éditeur spécifique au DSL basé sur le "vocabulaire" extrait du langage naturel utilisé.

— Produire l'AST : Une fois le modèle établi et l'éditeur du langage généré, un AST (Abstract Syntax Tree) sera créé afin de vérifier le bon fonctionnement et la cohérence des manoeuvres envoyées au drone.

5.3 Tranche 2 - 15/12/16

— Génération de code (langage cible choisi) : L'interface liant le DSL au code source sera écrite en langage C, sous Unix, en utilisant la librairie fournie par Parrot.

— Liaison de l'exécutif avec le SDK Parrot (et l'OS) : La traduction de l'ensemble des instructions décrites avec le DSL vers des combinaisons de fonctions de l'API du drone Parrot.

— Compilation du code généré : En utilisant de la compilation croisée du code généré, sur une machine ordinaire, à l'aide d'un compilateur dédié, cela nous permettra d'obtenir du binaire exécutable par le drone.

5.4 Tranche 3 - 03/01/17

— Identifier la configuration requise : La liste des besoins physiques et logiciels pour la réalisation du déploiement du projet sera enrichie tout au long de celui-ci.

— Élaborer un système de déploiement "à partir de rien" : La mise en place du système visant à exécuter une chorégraphie (installation du plugin du DSL, installation de la librairie Parrot, connexion du (des) drone(s), . . .) sera automatisé par un script lancé par l'utilisateur.

6 Références

1. S. Piskorski and N. Brulez and P. Eline and F. D'Haeyer, *AR.Drone Developer Guide*, SDK 2.0, May 21, 2012