



Rapport de stage ouvrier

(4 mai – 11 juin 2015)

**CPP
La prépa des INP
de NANCY**



Tuteur : Ernest Galbrun, Ingénieur de recherche

Entreprise : ENSEM

Adresse : 2 rue de la Fôret de Haye,
54516 Vandoeuvre-lès-Nancy



Table des matières

| | |
|--|----|
| Remerciements..... | 2 |
| Résumé..... | 3 |
| Introduction..... | 4 |
| 1Présentation de l'environnement..... | 5 |
| 1.1L'ENSEM, les TP SAMI..... | 5 |
| 1.2Les outils de travail..... | 6 |
| 2Déroulement du stage..... | 9 |
| 2.1Déroulement du stage..... | 9 |
| 2.2Bilan..... | 12 |
| 3Aspects pédagogique et personnel..... | 14 |
| 3.1Aspect pédagogique..... | 14 |
| 3.2Aspect personnel..... | 14 |
| Conclusion..... | 15 |
| Annexe..... | 16 |
| 1. Fonctionnement de l'AR.Drone 2.0..... | 16 |
| 2. Communication, réception des données..... | 18 |

Remerciements

Avant tout, c'est Ernest Galbrun, mon tuteur, que j'aimerais remercier, d'abord d'avoir accepté que je fasse mon stage à ses côtés, puis de m'avoir parfois aidé et conseillé, de m'avoir apporté son expérience et ses connaissances et enfin de m'avoir éclairé sur le métier d'ingénieur en informatique.

Ensuite je voudrais remercier les élèves avec qui j'ai cohabité durant ces 6 semaines, avec qui j'ai échangé à propos de leurs projets et du mien, et qui ont affiné l'idée que je me fais de la vie en école d'ingénieur.

Enfin, je remercie les enseignants et l'équipe de filière ISA (Ingénierie des Systèmes Automatisés) pour leur accueil amical.

Résumé

Ce rapport présente le stage que j'ai effectué à l'ENSEM, durant la période du 4 mai au 11 juin 2015 avec pour maître de stage Ernest Galbrun, ingénieur de recherche en informatique au laboratoire CRAN ainsi qu'à l'ENSEM.

L'école dispose de plusieurs drones quadcoptères ainsi que d'une plateforme munie de caméras Infra-Rouge, destinée à l'étude des systèmes automatisés par les étudiants. L'année précédente, des élèves avaient pour projet de piloter ces drones avec une baguette. L'objectif de mon stage était de rajouter des fonctionnalités à leur projet dans le but de rendre le pilotage à la baguette plus facile en développant une interface graphique pour l'utilisateur, et permettre une meilleure étude du comportement du quadcoptère en recueillant les données de navigation qu'il nous envoie.

Ce projet a été réalisé dans l'environnement de développement de Matlab. Toutefois, il s'est avéré nécessaire d'utiliser une autre langage de programmation : Java. Tout au long, j'ai mené de nombreuses recherches documentaires afin d'apprendre les langages de programmation auxquels j'avais affaire et de mieux maîtriser les outils dont je disposais tels que les drones ou le logiciel Cortex.

L'avancée du projet s'est faite en plusieurs étapes. Il a d'abord fallu comprendre le fonctionnement et la configuration du drone afin d'établir une communication simple, puis améliorer cette communication en l'intégrant aux programmes précédemment créés. Cependant, la difficulté principale était de ne pas pénaliser les fonctions déjà présentes en ajoutant les nouvelles.

Je retire plusieurs bénéfices de ce stage. D'abord, un développement de mon autonomie, puis le contact avec un ingénieur en informatique ainsi qu'avec des élèves ingénieurs, l'assimilation de nombreuses connaissances importantes concernant l'informatique, le réinvestissement de compétences acquises lors des deux dernières années et enfin l'accomplissement d'une démarche que j'ai construite pour répondre à un besoin donné.

Introduction

La mission qui m'a été confiée pour mon stage est la suivante : développer une interface utilisateur permettant le pilotage d'un drone ainsi que la visualisation graphique de son état en temps réel. Ceci se faisant dans le cadre du projet SAMI (Systèmes Autonomes Multi-agents Interactifs) destiné à proposer des travaux pratiques aux élèves de l'ENSEM dans un cadre pédagogique. Ayant suivi les enseignements des thèmes Math et TI2E, ce stage est aussi l'occasion de réinvestir des connaissances en situations réelles.

Avant le stage, les seules commandes auxquelles nous avions accès étaient le décollage. Ensuite, il fallait espérer qu'il n'y ait pas de problème pendant le vol et attendre qu'il se termine. Ce n'est qu'à la fin que nous avions accès aux informations concernant le comportement du drone vis-à-vis de la baguette. Il était donc impossible d'empêcher un crash contre un mur ou de détecter un éventuel problème concernant l'appareil pendant le vol. C'est pour pallier à ce problème que l'on m'a confié cette mission.

1 Présentation de l'environnement

1.1 L'ENSEM, les TP SAMI

Le stage s'est déroulé à l'ENSEM, école d'ingénieur en électricité et en mécanique, qui propose une formation Énergie et une formation Systèmes numériques. Dans le cadre pédagogique, les élèves sont amenés à réaliser différents projets concernant l'automatique, l'informatique, l'électronique, etc. ainsi que de nombreux travaux pratiques.

Pour être menés à bien, ces TP et ces projets ont une salle qui leur est réservée : la salle SAMI (Systèmes Autonomes Multi-agents Intelligents). C'est dans cette salle que j'ai travaillé tout au long de mon stage.

Cette salle est équipée d'une part de nombreux ordinateurs de travail tournant sous Linux ou Windows 7 selon les besoins. D'autre part, la salle comprend une plateforme dédiée au pilotage de différents robots, y compris des drones. Cette plateforme est munie de plusieurs caméras Infra-Rouge.

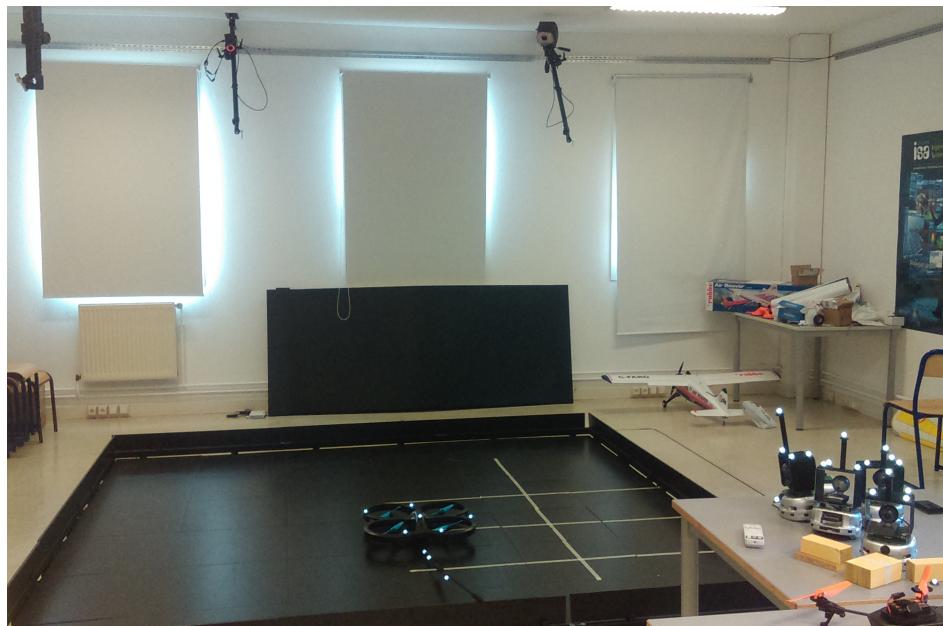


Illustration 1: La plateforme de la salle SAMI avec les caméras IR au plafond

Ces drones sont l'objet de mon projet de stage. Ce sont des AR.Drone 2.0 de la marque française Parrot, ce sont des quadcoptères munis d'une caméra frontale et d'une caméra ventrale.

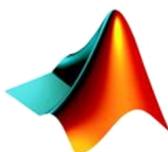


Illustration 2: AR.Drone 2.0 de Parrot

Ces quadcoptères ont été commercialisés dans un but video-ludique. À l'ENSEM ils sont utilisés pour permettre aux élèves d'étudier des systèmes automatisés. En l'occurrence il y a déjà eu plusieurs projets réalisés sur ces engins.

1.2 Les outils de travail

L'un des projets précédent sur l'AR.Drone était de pouvoir le piloter simplement avec une baguette. Pour cela, le drone et la baguette sont tous deux équipés de marqueurs qui leur sont propres et qui sont détectables par les caméras IR. Ensuite, l'un des ordinateurs est relié aux caméras. Sur cet ordinateur est installé le logiciel de « motion tracking » Cortex. Il permet de modéliser en 3D les objets détectés par les caméras. C'est lui qui sert de relai entre la plateforme et les programmes développés par les élèves. Ces programmes ont été créés grâce au logiciel de calcul numérique Matlab qui inclut un environnement de développement.



MATLAB®

Illustration 4: Logo du logiciel de calcul numérique Matlab



Illustration 3: Logo du logiciel de motion tracking Cortex

Il faut savoir que le drone possède un système embarqué de type Linux qui interprète les commandes reçues et les corrige éventuellement. Il envoie aussi de nombreuses informations concernant ses données de navigation appelées plus couramment *navdata*. La configuration du drone est très flexible car il existe beaucoup de paramètres. Toutes ces échanges d'informations se font par un réseau wifi émis par l'engin. Il suffit de se connecter à ce réseau pour communiquer avec le drone. La communication est répartie sur différents ports¹ du réseau. Chacun à un rôle défini :

- le port 5554 est celui par lequel le drone envoie les navdata,
- le port 5555 est celui par lequel le drone transmet le flux video de sa caméra,
- le port 5556 est celui par lequel on configure et commande le drone.

Ce sont les trois ports principaux permettant une utilisation correcte. Le 5556 est celui dont se sont occupés les auteurs du projet de pilotage à la baguette. L'objectif était alors de reprendre leurs programmes écrits en langage Matlab pour ajouter certaines fonctionnalités, à savoir : récupérer les navdata et les afficher en temps réel et construire une interface graphique qui entraînerait une utilisation plus simple de leurs programmes. Mon projet permettrait donc de mieux comprendre le drone et d'étudier plus précisément ses réactions face aux commandes qu'il reçoit.

¹ Un port est assimilable à un canal de communication ou à une fréquence radio.

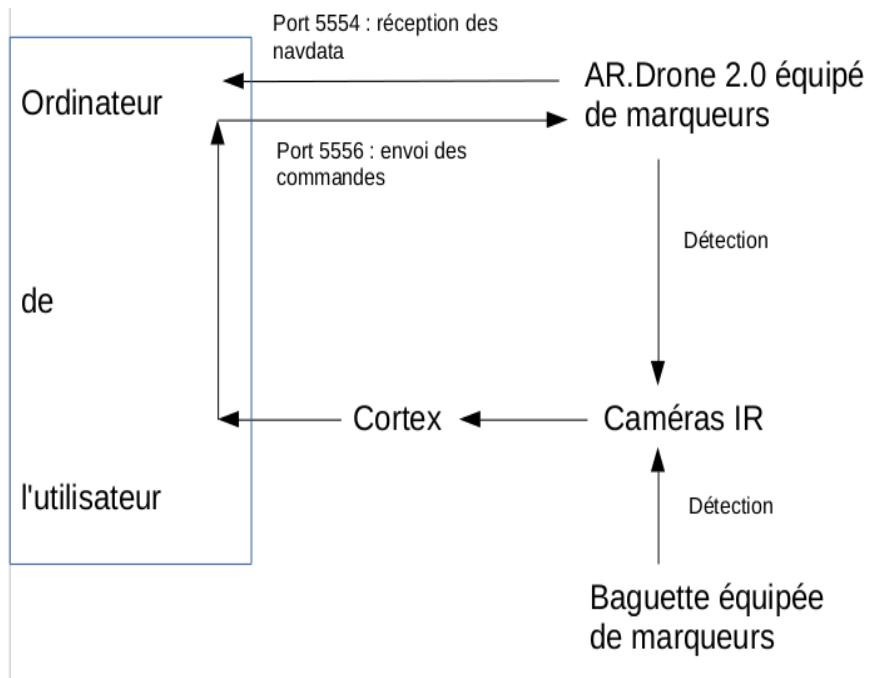


Illustration 5: Schéma de fonctionnement de la communication avec le drone



Illustration 6: AR.Drone 2.0 de Parrot avec ses marqueurs

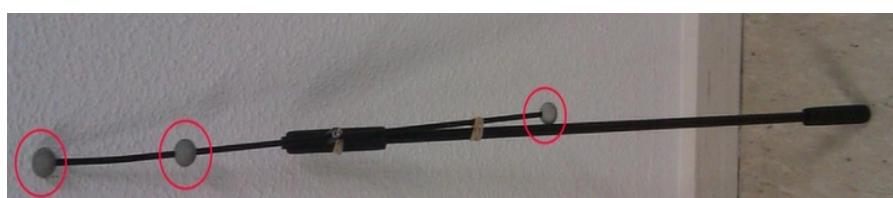


Illustration 7: La baguette modélisée par trois marqueurs alignés

2 Déroulement du stage

2.1 Déroulement du stage

Le premier jour, M. Galbrun m'a présenté la salle SAMI ainsi que le fonctionnement global des différents logiciels que j'aurai à utiliser.

Le premier problème dans mon projet était de comprendre comment communiquer avec le drone. Pour cela, l'idée était de trouver une application qui savait le faire, et d'étudier son fonctionnement au moyen du logiciel Wireshark. Il permet de surveiller un réseau est d'intercepter les données échangées entre les différents appareils. En se servant de la documentation du drone, on pouvait déchiffrer ce que l'application envoyait et ce qu'elle recevait. C'est à ce moment que les cours d'informatique de première année ce sont avérés utiles. En effet, les données que nous recevions étaient sous forme hexadécimale. Pour les interpréter en temps réel, il fallait être capable de créer un algorithme pour les convertir en nombres décimaux. À la suite de ça, et après avoir fait des recherches documentaires pour apprendre à utiliser Matlab, j'ai pu écrire un premier programme capable d'initialiser la communication et de récupérer les données, ainsi que de les afficher sous forme décimale. Cependant, ce programme était indépendant et ne permettait en aucun cas le pilotage. J'ai donc repris les programmes de pilotage des anciens élèves afin de leur intégrer la réception des données et leur affichage.

Ceci m'a conduit à un deuxième problème : la réception, la conversion et l'affichage des données ralentissaient beaucoup l'envoi des commandes. L'envoi moins régulier des commandes dégradait la précision du pilotage parce que ça rendait le drone plus instable. En effet, si le drone est en retard, il sera plus loin du point auquel il doit se rendre, la commande pour l'y envoyer sera alors plus forte et le drone, victime de son inertie, ira plus loin que son objectif. Pour ordre de grandeur, sans la réception des données, le programme est capable d'envoyer 300 commandes par seconde. Une telle fréquence permet des calculs plus précis pour répondre aux besoins du pilotage, ainsi qu'une bonne réactivité du drone aux mouvements de la baguette. En revanche, si l'on inclut la réception et l'affichage des navdata, la fréquence tombe à 10 par secondes et entraîne l'instabilité du drone. Pour résoudre ce problème, il faudrait que l'envoi des

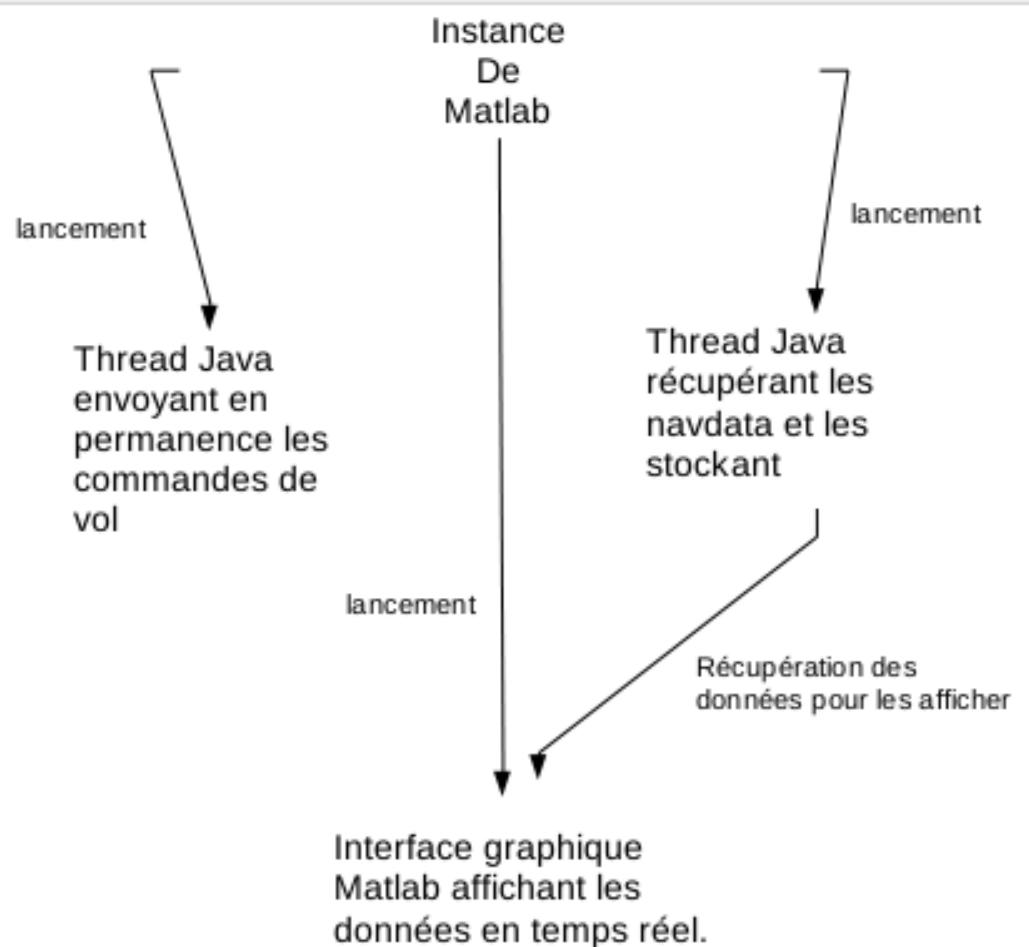
commandes et la réceptions des données soient deux processus différents, s'exécutant en parallèle. Or, Matlab ne supporte pas le *multi-threading*², c'est-à-dire qu'il ne peut exécuter qu'une seule chose à la fois. En revanche, l'environnement Matlab et le langage de programmation Java sont très liés et on peut faire facilement la liaison entre des programmes Matlab et des programmes Java. De plus, Java est un outil puissant en matière de multi-threading. J'ai donc créé un *thread* en Java, qui permettait la réception et la conversion des données, mais pas l'affichage. En effet l'affichage ne peut être fait que par Matlab puisque c'est l'environnement dans lequel s'exécute le programme et dans lequel est affichée l'interface graphique. C'est ainsi que je me suis rendu compte que l'affichage était en fait la seule chose qui était si lente à s'exécuter. La solution du problème est de faire tourner aussi l'envoi des commandes dans un programme Java s'exécutant en parallèle. Ainsi la seule chose que Matlab ferait serait d'actualiser l'affichage des données reçues afin de les voir en temps réel, sans pénaliser ni l'envoi régulier des commandes, ni la réception des données.

C'est l'objet du troisième problème que j'ai rencontré : pour réussir à créer un programme Java envoyant des commandes, il fallait que Java fasse appel aux fonctions de pilotage qui avaient été développées en Matlab. Après quelques recherches internet, j'ai trouvé une solution qui permet d'exécuter des programmes Matlab depuis Java, nommée *matlabcontrol*. C'est un *package*, c'est-à-dire un ensemble de fonctions que l'on peut appeler dans un programme Java après les avoir importées. Ce que j'ai réussi avec succès. Je m'attendais alors à voir mon projet aboutir comme je l'espérais. Cependant je me suis rendu compte que problème n'était pas résolu en voyant que l'envoi des commandes était toujours aussi lent. Mais cette fois, c'était due à l'appel des fonctions Matlab depuis Java, ça ralentissait énormément Java, c'était inattendu. La solution la plus simple était alors de traduire tous ces programmes que les élèves de l'ENSEM avaient écrit en Matlab, en Java. Étant débutant dans ces deux langages, la tâche n'a pas été simple et à nécessité encore quelques recherches concernant ces deux langages. Finalement, cela s'est avéré être la bonne solution et la fin des problèmes. En effet, la fréquence était à présent de 500 par secondes, et le pilotage était précis, le drone était stable.

À présent, le code Matlab lance les threads Java. Ainsi, Matlab ne s'occupe que

² Thread se traduit fil d'exécution, le multi-threading est donc la capacité à créer plusieurs fils d'exécution, permettant alors de faire tourner plusieurs processus en parallèle.

de l'actualisation en temps réel des données de navigation du drone, tandis que les threads s'occupent d'une part de la réception, de la conversion des données et de leur stockage pour permettre à Matlab de les récupérer et d'autre part de l'envoi régulier des commandes.



Les trois processus tournent en parallèle

Illustration 8: Fonctionnement du programme final

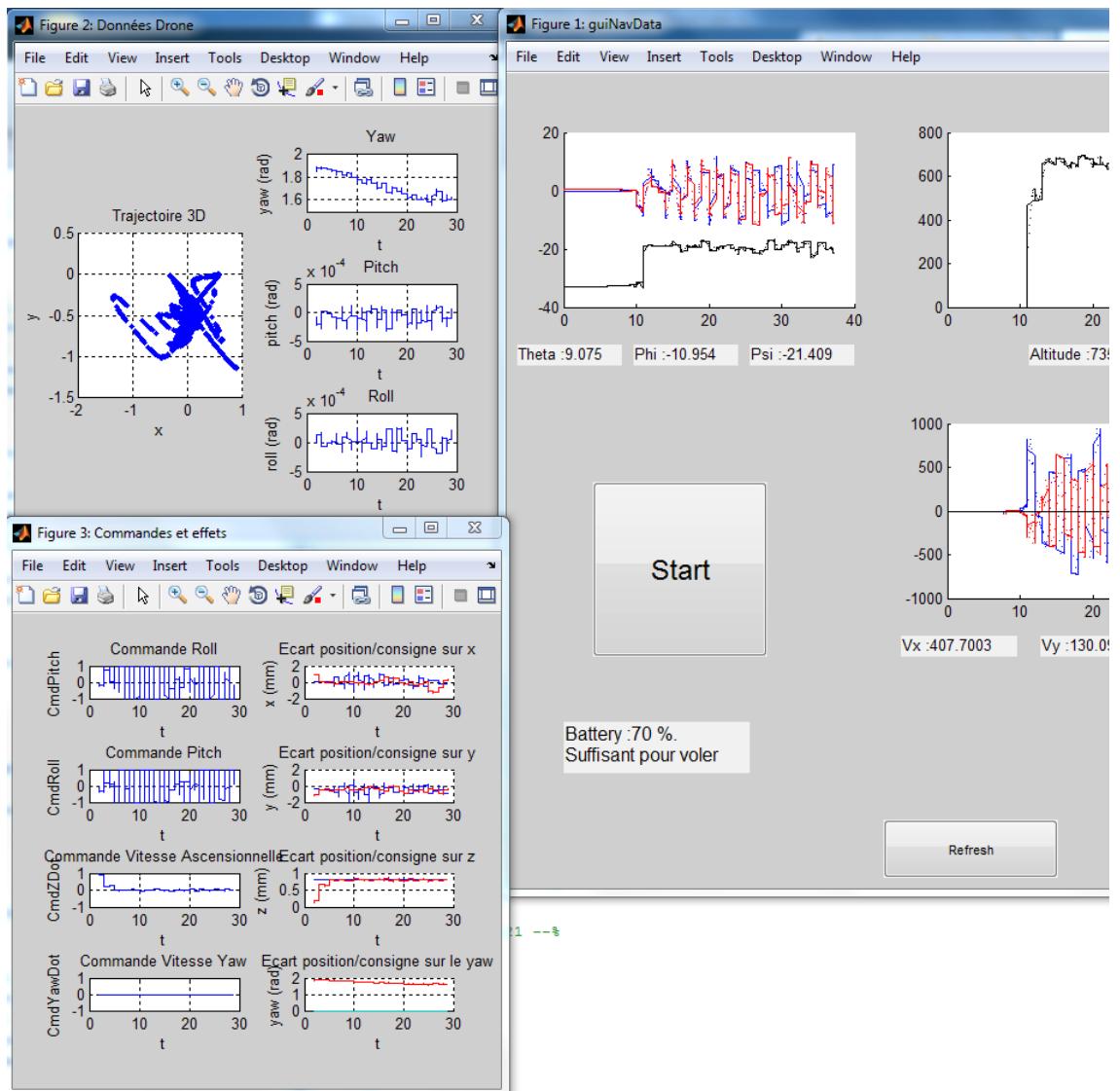


Illustration 9: Rendu visuel du programme

2.2 Bilan

Après six semaines de stage, le rendu final correspond plutôt bien à la mission qui décrive dans la fiche de liaison. En effet, les fonctions d'interfaçage permettant la communication entre le drone et l'utilisateur sont faites, ainsi qu'une interface graphique facilitant l'utilisation du programme. Aujourd'hui, le pilotage est accessible à toute personne n'ayant que peu de connaissances en informatique ou en automatique.

En ce qui concerne le développement des programmes, il a été assez lent à cause du temps de recherche nécessaire à apprendre les langages de programmation, le fonctionnement du drone ainsi que d'autres connaissances diverses sur les programmes déjà faits et l'environnement de travail. De plus, les problèmes rencontrés étaient inattendus. Avec un prolongement du stage, des améliorations auraient pu être possibles telles que l'ajout des images de la caméra du drone en direct, ou l'ajout de nombreuses autres données sur la navigation du drone afin de mieux comprendre son état et d'améliorer ainsi son utilisation. Cependant, certaines de ces améliorations sont facilement réalisables pour des personnes qui travailleraient sur le drone prochainement et seraient compatibles avec les programmes déjà faits.

3 Aspects pédagogique et personnel

3.1 Aspect pédagogique

Ce stage m'a donné l'occasion de mettre en œuvre une réelle démarche pour répondre à une demande. C'est une composante essentielle du métier d'ingénieur. Dans le même temps j'ai assimilé beaucoup de connaissances, notamment sur la programmation et les réseaux. Cela m'a permis de trouver un aspect pratique à certaines matières étudiées durant ces deux dernières années. Ce stage m'a aussi contraint à utiliser l'anglais, car toutes les documentations que je lisais étaient écrites en anglais. Puis les résultats des recherches internet sont beaucoup plus nombreux lorsque la requête est faite en anglais et c'est parfois ce qui permet de trouver la bonne solution. De plus, j'ai pu apprendre deux nouveaux langages de programmation qui sont parmi les plus répandus dans l'industrie (Java) et le domaine scientifique (Matlab). Plus particulièrement, en apprenant le langage Java, j'ai assimilé un concept très important en informatique, la programmation orientée objet. Me destinant à évoluer dans ce domaine là, j'ai la conviction que ce stage me sera bénéfique pour la suite de mes études ainsi que pour ma future carrière professionnelle.

3.2 Aspect personnel

D'un point de vue personnel, j'ai pu développer plusieurs compétences, notamment l'autonomie. En effet, je travaillais seul sur le projet, j'étais responsable de trouver et de mettre en œuvre les solutions aux problèmes, d'apprendre et de comprendre les informations dont j'avais besoin pour aboutir. Ensuite, en travaillant dans cette salle de TP, je suis entré en contact avec de nombreux élèves de l'ENSEM et en particulier certains anciens élèves du CPP. Nous avons pu échanger autour de leurs études et des miennes, afin que je me fasse une meilleure idée de ce qu'est la vie d'un élève ingénieur et des compétences requises en école. D'autre part, en discutant avec mon maître de stage j'ai pu obtenir des informations supplémentaires sur le métier d'ingénieur en informatique.

Conclusion

Pour terminer ce rapport, j'aimerais ajouter que j'ai beaucoup apprécié ce stage, tant par le fait d'avoir appris beaucoup de choses aux côtés d'un ingénieur, que par le fait d'avoir été mis à l'épreuve quant à mon autonomie, ma capacité à m'adapter et à construire une démarche. C'est d'autant plus une grande satisfaction que d'avoir aboutit dans ce projet. Je pense en retirer un bénéfice pédagogique ainsi que personnel. De plus, il me semble avoir créer des programmes qui seront facilement réutilisables afin de les modifier ou de leur ajouter des fonctionnalités. J'espère qu'ils seront utiles aux prochains élèves qui feront un projet sur le quadcoptère AR.Drone 2.0 ainsi qu'à d'autres personnes qui souhaiteraient utiliser ce drone.

Annexe

Pour une compréhension plus approfondie de ce rapport et une vision plus technique du projet réalisé, je présenterai dans cette partie le fonctionnement détaillé du drone et de la communication qu'il impose, ainsi que la manière de déchiffrer ses données de navigation.

1. Fonctionnement de l'AR.Drone 2.0

Comme dit précédemment, ce drone est un quadcoptère. Par conséquent, comme tout objet aéronautique, il possède trois angles qui caractérisent son orientation : le roll, le pitch et le yaw. Le roll est son angle d'inclinaison droite/gauche, tandis que le pitch est l'inclinaison avant/arrière et le yaw est son orientation :

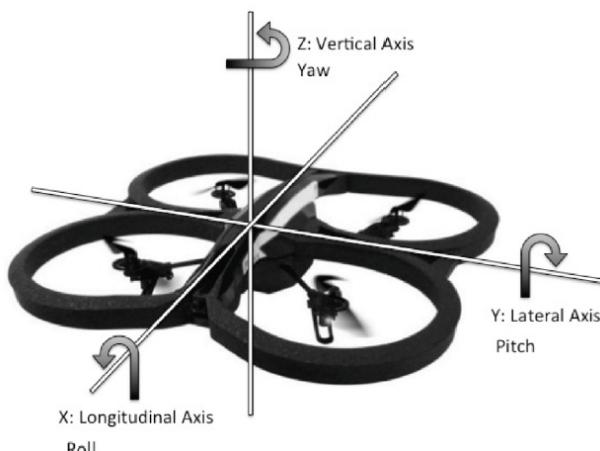


Illustration 10: Angles caractéristique en aéronautique

Pour se déplacer, ce qui revient à faire varier ses angles, le drone joue sur la vitesse de chacune de ses quatre moteurs, correspondant aux quatre hélices. Par exemple, pour avancer, il agira de la manière suivante :

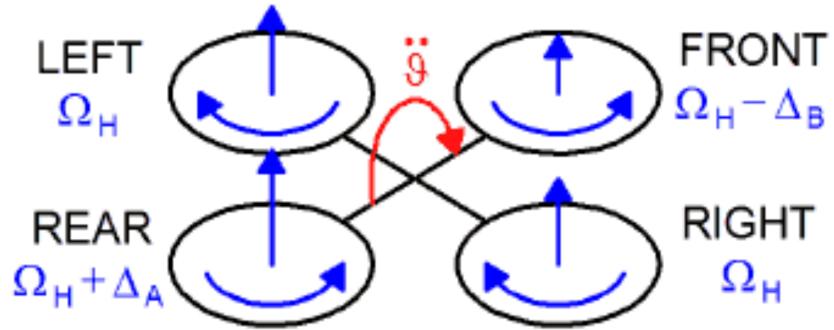


Illustration 11: Le moteur avant tourne moins vite pour que le drone s'incline et avance

Pour donner des instructions au drone depuis un mobile ou un ordinateur, il suffit de lui envoyer un message par le réseau en respectant une syntaxe bien précise. Il existe plusieurs commandes différentes nommées commandes AT. La syntaxe est la suivante :

AT*nomdelacommande=numérodesequence,paramètresdelacommande. Par exemple, pour simplement dire au drone de ne pas bouger, on peut lui envoyer :

"AT*PCMD=1,1,0,0,0\r"

Ici, PCMD correspond à « progressive command ». Le numéro de la séquence doit être incémenté à chaque nouvel envoi d'une commande, on commence donc par « 1 ». Le second « 1 » est le premier paramètre, il signifie qu'on autorise les commandes progressives. Ensuite viennent les angles et les vitesses dans l'ordre suivant :

roll, pitch, vitesse verticale, vitesse angulaire

Ces paramètres correspondent à des pourcentages du maximum. Par exemple un roll de 0,5 fera pencher le drone sur sa droite avec une inclinaison correspondant à la moitié de l'inclinaison maximale qu'il peut atteindre. Ils doivent être envoyés au drone sous la forme d'un entier sur 32 bits. Par exemple, 0,8 devient 1061997773. La conversion est la suivante :

D'après le format IEEE-754 :

0.8 devient 0 01111110 1001100110011001101 en binaire.

0 veut dire que 0.8 est positif, 01111110 est l'exposant, et

10011001100110011001101 est la mantisse. On retrouve 0.8 par la formule : $-1^{\text{signe}} * 2^{\text{exposant}} * \text{mantisse}$.

Ensuite on convertit simplement le nombre binaire en nombre entier pour obtenir 1061997773.

Les deux caractères "\r" servent à délimiter la fin de la ligne de commande.

C'est dans ce format que sont envoyées toutes les commandes au drone.

De plus, le numéro de séquence permet de savoir si une commande est toujours d'actualité. Par exemple, si le drone reçoit une commande avec le numéro de séquence 100 mais que la commande qui suit contient le numéro 1, elle sera considérée obsolète et il ne l'exécutera pas.

Il existe quelques autres commandes permettant le decollage ou l'atterrissement, un arrêt d'urgence, la configuration des paramètres ou des animations avec les quatres LED que possède le drone.

2. Communication, réception des données

Tout d'abord, pour initialiser la communication, il est essentiel d'envoyer un message sur le port 5554, contenant uniquement le chiffre 1 codé sur 32 bits.

Ensuite, pour entretenir la communication et recevoir les navdata, il y a deux commandes essentielles :

"AT*COMWDG\r" et "AT*CONFIG=1,"general :navdata_demo","TRUE"\r"

La première est appelée *watchdog* et sert à la synchronisation avec le drone. Si elle n'est pas envoyée régulièrement (plus de 1 fois toutes les 2 secondes) le drone arrête la communication avec l'utilisateur. Ainsi il ne répond plus aux commandes et n'envoie plus les navdata. C'est pour cela que l'utilisation d'un thread est pratique. Sans thread, il serait plus compliqué de s'assurer que le watchdog est envoyé dans le délai imparti. La deuxième commande est une configuration, elle active l'envoi régulier des navdata par le drone.

Les navdata sont reçues sous la forme d'un paquet de 500 octets, tel qu'on peut l'observer avec Wireshark :

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 24 | 01 | a6 | b2 | df | 90 | 03 | b7 | 2c | 2d | ac | 08 | 00 | 45 | 00 |
| 02 | 10 | 00 | 00 | 40 | 00 | 40 | 11 | b5 | 89 | c0 | a8 | 01 | 01 | c0 | a8 |
| 01 | 02 | 15 | b2 | f6 | 6c | 01 | fc | 4e | e2 | 88 | 77 | 66 | 55 | d0 | 04 |
| 80 | 0f | d9 | 02 | 00 | 00 | 01 | 00 | 00 | 00 | 00 | 00 | 94 | 00 | 00 | 00 |
| 02 | 00 | 52 | 00 | 00 | 00 | 00 | 40 | 8c | c4 | 00 | 80 | 26 | c4 | 00 | c0 |
| 90 | c4 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 03 | 00 | 00 | 00 | 0b | e6 | |
| 7f | 3f | 17 | 6e | a7 | 3c | 36 | 76 | 9e | bc | f5 | 8a | a5 | bc | df | ed |
| 7f | 3f | 6f | 1e | 45 | 3c | ac | 6e | a0 | 3c | ef | a2 | 3e | bc | fe | ee |
| 7f | 3f | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 6f | c3 | 10 | 00 |
| 48 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ff | ff | 08 | 00 | 3d | 1e | 00 | 00 |

Illustration 12: Navdata sous forme hexadécimale, vu par Wireshark

Dans ce message reçu, les informations commencent à partir de 88776655 qui est l'identifiant de l'appareil. Ensuite, suivent des données diverses. Par exemple, 52 correspond à la batterie, c'est égal à 82 % en décimal. Puis C4 8C 40 00 est l'angle theta du drone, c'est-à-dire le roll. Ici il est égal à -1122 millidegrés dans le format IEEE-754. Et c'est de cette manière que l'on peut déchiffrer une à une les informations de navigation pour ensuite les afficher de manière lisible. En ce qui concerne le programme que j'ai réalisé, il n'affichait que les angles de roll, pitch et yaw, les vitesses angulaires et verticales, l'altitude et la batterie. L'affichage est fait de manière numérique et graphique en traçant une courbe d'évolution qui dépend du temps. Le code a été finalisé afin de permettre une personnalisation des besoins quant aux données reçues par le drone.