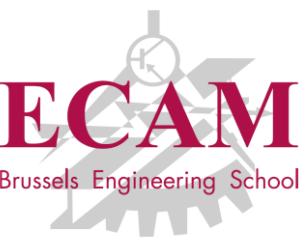


Haute Ecole LEONARD de VINCI



Institut Supérieur Industriel

Étude et conception de l'architecture de contrôle d'un drone destiné à la construction d'une structure en intérieur

Travail de fin d'études présenté par

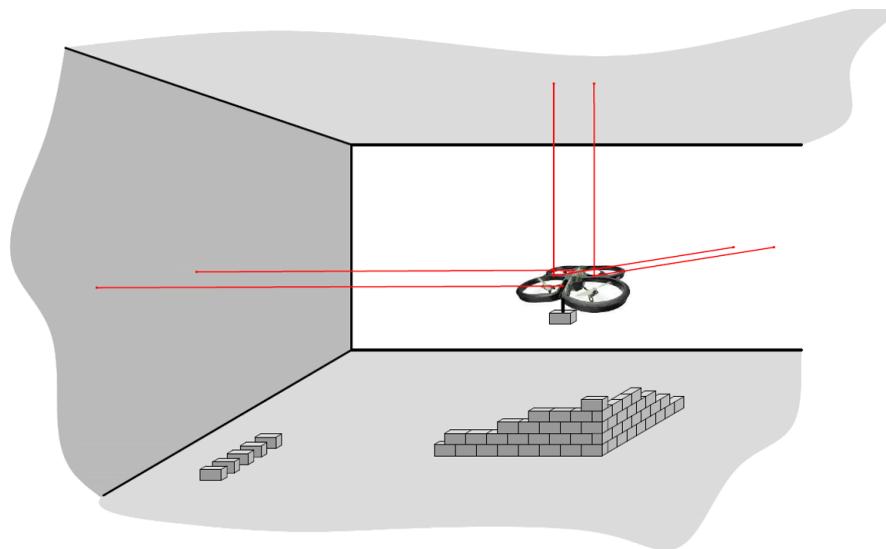
Nabil NEHRI
Alexis PAQUES

En vue de l'obtention du diplôme de
Master en Sciences de l'Ingénieur Industriel finalité Électronique

Année académique 2015 - 2016



Étude et conception de l'architecture de contrôle d'un drone destiné à la construction d'une structure en intérieur



Étudiants :

Nabil NEHRI & Alexis PAQUES

Promoteur : M. Pierre LATTEUR

Tuteur : M. Sébastien GOESENS

Superviseur : M. Cédric MARCHAND

Année académique 2015 – 2016

CAHIER DES CHARGES RELATIF au TRAVAIL DE FIN D'ETUDES de

Alexis Paques inscrit en Master en électronique (ECAM)
Nabil Nehri inscrit en Master en électronique (ECAM)

Année académique : 2015-2016

Titre provisoire : Étude et conception de l'architecture de contrôle d'un drone destiné à la construction d'une structure en intérieur.

Objectifs à atteindre :

- Réaliser un système de localisation à l'aide de lasers
- Contrôler un drone via le PixHawk en le localisant dans un espace virtuel à l'intérieur d'un bâtiment

Principales étapes :

- Réalisation d'un système d'acquisition pour les lasers ;
- Vérification et amélioration de la fiabilité des lasers ;
- Extraction de la position du drone provenant des lasers ;
- Développement des modules d'acquisition, d'algorithme, de télémétrie de commande et de tâches ;
- Réalisation d'une architecture de contrôle modulaire.

Fait en trois exemplaires à Louvain-la-Neuve, le 03/12/2015

L'Etudiant

Nom-prénom :

Alexis Paques
APaques

Signature

NEHRI NABIL
N.N.

Le Tuteur

Nom-prénom :

MARCIANIS
CEDRIC

Département/Unité

.....G.E.I.....

Signature

clachanot

Le Promoteur

Nom-prénom :

GOESSENS
SEBASTIEN

Société

...VCL.....

Signature

Gosse

Remerciements

Pierre Latteur, qui porte le projet et le suit activement ;

Cédric Marchand, pour nous avoir suivis tout au long de l'année ;

Sébastien Goessens, qui a participé activement à la mise en œuvre du projet ;

Guy Henriet de nous avoir permis d'utiliser le laboratoire LEMSC comme lieu de recherche ;

Alex Bertholet, de nous avoir fourni un avis extérieur critique et constructif ;

Samuel Laurent, d'avoir toujours été prêt nous aider ;

Antoine Bietlot, d'avoir mis de la bonne ambiance dans le laboratoire ;

Viviane Delmarcelle, de s'être occupée de nos nombreux documents administratifs ;

Bernadette Vandevenne, qui nous a évité bien des soucis administratifs ;

À nos relecteurs

Kenza Nehri, Assya Kasry, Youssra Demane ;

Vorleak Nou, Anh Sophie Noël, Manoel da Silva ;

À nos collègues de projet

Amaury Montcourrier & Adrien Naveau ;

Charles Coppieters de Gibson ;

Milan Renier ;

À nos collègues ECAMIens

Lionel Gosselin, qui nous a aidés pour l'impression 3D ;

Raphaël Van Hoolandt, qui nous a accueillis à l'étage H ;

Aux 5MEO-MIN ;

Mais aussi

À toute l'équipe du laboratoire LEMSC ;

Et surtout à nos familles.

Résumé

Galilée, la révolution industrielle, l'informatique, autant de révolutions qui ont transformé le secteur de la construction. Cela dit, autant pour la construction de la pyramide de Kheops, des temples grecs, du pont du Gard ou d'un bâtiment récent, un problème persiste : celui du facteur humain. En effet, d'une part 10% des coûts d'un chantier proviennent des erreurs humaines ou des arrêts de travail aléatoires pour cause de mauvais temps. D'autre part, la mauvaise considération de l'humain et les mauvaises conditions de travail rendent le secteur de la construction dangereux et provoquent de nombreux accidents. Suite à cela, plusieurs systèmes de constructions automatiques ont vu le jour, tels que la paveuse automatique, la plafonneuse, etcétera.

C'est un nouveau regard de cette automatisation qu'a Mr. Pierre Latteur, diplômé d'ingénieur civil en 1994 et professeur à l'UCL. En effet, pourquoi ne pas se baser sur la nature ? Pourquoi, tels des oiseaux faisant leur nid ou des abeilles leur ruche, ne serait-il pas possible d'avoir un essaim de drones qui irait construire brique à brique un bâtiment ?

Tel est le projet : construire des structures avec un drone. Pour y arriver, plusieurs parties du processus doivent être repensées, telles que les briques, la pince ou le système de positionnement de ces briques. C'est la conception de ce dernier que relate ce travail de fin d'études, au travers de trois délivrables. Un *système de contrôle autonome et modulable d'un drone* qui simplifie l'intégration de nouveaux modules, un *module de localisation à lasers fixes* et un *module d'interface de contrôle*.

Le travail tel qu'il est présenté se sépare en 3 parties : l'étude, la réalisation et les tests. Il a pour but d'être la référence du projet pour implémenter un nouveau module, en posant les jalons qui rendent le projet modulable. Il montre aussi de façon concrète les différentes étapes de conception des trois délivrables ainsi que les tests réalisés pour s'assurer de leur bon fonctionnement.

À l'issue de ce travail, le drone est capable de se positionner dans une pièce rectangulaire connue avec une erreur maximale de 5 centimètres et de se déplacer dans la pièce de façon autonome.

Mais bien sûr, comme dans tout projet, il est encore possible d'améliorer le résultat. C'est pourquoi nous avons posé le fil conducteur dirigeant les principales améliorations à réaliser afin d'obtenir un système plus robuste et plus sûr pour l'opérateur.

Table des matières

Table des matières	1
I Projet	7
1 Introduction	8
2 Contexte	9
2.1 L'équipe	9
2.2 Produit	11
2.3 Aspects juridiques	11
2.4 Drone de test et drone du projet	13
2.5 Contrôleur de vol	15
3 Cahier des charges et résultats	18
3.1 Finalités du projet	18
3.2 Cahier des charges	19
3.3 Résultats	20
II Étude	22
4 Architecture de contrôle	23
4.1 Tout sur le Pixhawk	23
4.2 Arduino	24
4.3 Companion Computer avec ROS	24
4.4 Architecture choisie	25
5 Méthodes de localisation	26
5.1 GPS	27
5.2 D-GPS	28
5.3 GPS-RTK	28
5.4 Ultrasons	29
5.5 Lasers fixes	29
5.6 Lasers tournants	30

5.7 Station totale	31
5.8 Balises UWB	32
5.9 SLAM	32
5.10 Localisation via de multiples caméras	33
5.11 Vision integration	33
5.12 Choix réalisé	34
6 Choix du matériel	35
6.1 Lasers	35
6.2 Arduino	37
6.3 Ordinateur de bord	38
7 ROS	40
7.1 Introduction	40
7.2 Principe	40
7.3 Environnement	42
7.4 Exemple	43
7.5 Conclusion	44
III Réalisation	45
8 Vue d'ensemble de la réalisation	46
9 Localisation	47
9.1 Méthode de localisation	47
9.2 Besoin de l'algorithme	48
9.3 Acquisition des lasers	49
9.4 Correction d'entrée	50
9.5 Algorithme de positionnement	53
9.6 Prédiction de l'état réel	59
9.7 Résultats du système de positionnement	61
10 Contrôle	62
10.1 Les différents Nœuds	62
10.2 Modularité	64
11 Interface	67
11.1 Problématique	67
11.2 Le besoin	67
11.3 En pratique	68
11.4 Conclusion	70

IV Tests et performances	71
12 Vue d'ensemble des tests de performance	72
13 Caractéristiques des lasers	74
13.1 Test de fiabilité	74
13.2 Essais de répétabilité	76
13.3 Test d'offset et de linéarité	76
13.4 Conclusion	77
14 Performances de l'algorithme de positionnement	78
14.1 Caractéristiques de l'état angulaire du PixHawk	78
14.2 Précision linéaire de l'algorithme de positionnement	81
14.3 Précision angulaire de l'algorithme de positionnement	83
14.4 Conclusion	84
15 Tests de contrôle	85
15.1 Précision du positionnement	85
15.2 Résilience aux perturbations	89
15.3 Réactivité de vol avec un scénario	89
15.4 Conclusion	90
V Conclusion	91
16 Perspectives	92
16.1 Fusion des senseurs	92
16.2 Amélioration de l'acquisition des lasers	95
16.3 Moyen drone	95
17 Conclusion	96
Bibliographie	97
VI Annexes	99
A PixHawk	100
A.1 Le contrôleur	100
A.2 Communication	100
A.3 Configuration	101
A.4 Modes de vol	102
A.5 Utilisation pratique	103
B Librairie d'acquisition des lasers	105

B.1	Introduction	105
B.2	Exemple concret	106
B.3	Erreurs typiques	108
C	Carte électronique	109
D	Installation et configuration du système	113
D.1	ROS	113
D.2	Linux	115
E	Utilisation du projet	116
E.1	Installation du projet	116
E.2	Démarrage du drone	118
F	Design 3D	122
F.1	Supports des lasers	122
F.2	Bras des lasers	123
G	Liste du matériel	125

Glossaire

Arduino est un microcontrôleur placé sur une carte de développement open-source et open-hardware avec des GPIO (*General purpose input/output*), des bus de communications hardware, des sorties PWM et des entrées analogiques. De plus, étant un projet communautaire, il est très documenté et beaucoup de librairies existent pour faciliter la programmation.

ARM est une architecture matérielle pour réaliser des processeurs ayant un rendement de calcul par watts maximal (*Téléphones portables ou Linux embarqué*). De plus, les processeurs ARM de nos jours comprennent aussi la RAM, la mémoire flash, des senseurs, etc. Les architectures x86 et x64 à contraria cherchent à avoir le rendement de calculs par dollars (*Ordinateurs portables ou fixes*).

Baromètre est un outil pour connaître l'altitude puisque la pression dépend de la hauteur par rapport au niveau de la mer.

CAN est un bus de communication série, bidirectionnelle, half-duplex, robuste (*transmission avec une paire différentielle et CRC*), implémentations propriétaires.

CRC (*Cyclic Redundancy Check*) est un système logiciel permettant de détecter les erreurs de transfert par ajout de combinaisons de données redondantes.

Duplex est une caractéristique d'un bus précisant la capacité aux appareils connectés ensemble de communiquer au même moment. Half-duplex précise que la communication ne peut pas être simultanée, au contraire de Full duplex qui précise que la communication est simultanée.

ESC est un contrôleur de puissance des moteurs. Il convertit la commande provenant du PixHawk en un signal triphasé pour contrôler les moteurs brushless.

Gyroscope est un instrument de mesure qui donne l'état angulaire du drone dans l'espace.

IMU (*Inertial Measurement Unit*) est composant électronique composé d'un accéléromètre, d'un gyroscope et souvent d'un magnétomètre, thermomètre et baromètre.

I2C est un bus de communication série, bidirectionnel, half-duplex, avec un maître et jusqu'à 255 esclaves (*127 pour l'Arduino*). Ce protocole est très répandu sur les capteurs en tout genre.

LiPo est un type de batterie rechargeable et légère. Cette batterie se base sur une réaction électrochimique de Lithium à l'état de polymère (*forme de gel*). La batterie se compose de plusieurs cellules chacune à une tension de 3.2V à 4.2V. NOTE : Il est important que les cellules ne descendent pas en dessous de 3.0V sous risque de destruction de la batterie.

Magnétomètre est un instrument qui donne la direction grâce au champ magnétique terrestre.

MAVLink est un protocole de communication utilisé par le PixHawk, standard, open-source, permettant de communiquer avec le PixHawk ou d'autres contrôleurs compatibles.

Mavros est une application pour ROS qui permet de communiquer facilement avec un véhicule utilisant le protocole MAVLink afin d'en prendre le contrôle.

Odroid XU4 est un ordinateur embarqué basse consommation (*architecture ARM*), qui tourne avec Linux.

PixHawk est un contrôleur de vol open-hardware et open-source utilisé pour nos tests. Il est implémenté sur un RTOS. Il est constitué d'une IMU et d'un processeur afin de stabiliser le multi rotor, gérer les communications, les procédures d'urgence et les commandes de vol.

PWM (*Pulse Wide Modulation*) est une façon de transmettre une donnée codée temporellement, selon le temps haut et temps bas du signal, afin d'être moins perturbée par les ondes électromagnétiques. Ce mode est souvent utilisé pour le contrôle de vitesse des moteurs ou de position des servomoteurs.

RaspberryPi est un ordinateur embarqué basse consommation (*architecture ARM*), qui tourne avec Linux.

ROS (*Robotic Operating System*) est une surcouche à installer sur un système Linux, qui implémente déjà beaucoup d'algorithmes de positionnements, de SLAM, de régulation ou encore de simulation. En général, la régulation est laissée à un système en temps réel tel qu'un microcontrôleur ou un RTOS car ROS n'est qu'une application.

RTOS (*Real Time Operating System*) est un système d'exploitation en temps réel. En général, ces systèmes fonctionnent avec peu de threads, souvent un seul par cœur.

ToF (*Time of flight*) est la durée d'un aller-retour d'un signal.

Première partie

Projet

CHAPITRE 1

Introduction

Galilée, la révolution industrielle, l'informatique, autant de révolutions qui ont transformé le secteur de la construction. Cela dit, autant pour la construction de la pyramide de Kheops, des temples grecs, du pont du Gard ou d'un bâtiment récent, un problème persiste : celui du facteur humain. En effet, d'une part 10% des coûts d'un chantier proviennent des erreurs humaines ou des arrêts de travail aléatoires pour cause de mauvais temps. D'autre part, la mauvaise considération de l'humain et les mauvaises conditions de travail rendent le secteur de la construction dangereux et provoquent de nombreux accidents.

C'est un nouveau regard de cette automatisation qu'a Mr. Pierre Latteur, diplômé d'ingénieur civil en 1994 et professeur à l'UCL, en effet, pourquoi ne pas se baser sur la nature ? Pourquoi, tels des oiseaux faisant leur nid ou des abeilles leur ruche, ne serait-il pas possible d'avoir un essaim de drones qui irait construire brique à brique un bâtiment ?

Tel est le projet : construire des structures avec un drone. Pour y arriver, plusieurs parties du processus doivent être repensées, telles que les briques ou la pince ou le système de positionnement de ces briques. Ce de la conception de ce dernier que relate ce travail de fin d'études, au travers de trois délivrables. Un *système de contrôle autonome et modulable d'un drone* qui simplifie l'intégration de nouveaux modules, un *module de localisation à lasers fixes* et un *module d'interface de contrôle*.

Le travail tel qu'il est présenté se sépare en 3 grandes parties : l'étude, la réalisation et les tests. Il a pour but d'être la référence du projet pour implémenter un nouveau module, en posant les jalons qui rendent le projet modulable. Il montre aussi de façon concrète les différentes étapes de conception des trois délivrables ainsi que les tests réalisés pour s'assurer de leur bon fonctionnement.

CHAPITRE 2

Contexte

Abstract

Dans ce chapitre, les différents acteurs seront mis en avant, suivis du contexte du projet, ainsi que le domaine juridique du secteur des drones.

2.1 L'équipe

L'année passée, deux étudiants de l'UCL ont réalisé une étude de faisabilité du projet. Afin de réaliser ce projet ambitieux, 4 travaux de fins d'études ont été lancés afin d'analyser et réaliser les différents aspects de ce drone. Chaque TFE est en lien avec les autres et apporte sa brique à l'édifice.

- Amaury Moncourrier et Adrien Naveau, deux étudiants en génie civil, s'occupent du développement d'une brique simple et autobloquante qui permet d'augmenter la possibilité d'imprécision de placement de la brique, en concordance avec le développement du système d'agrippement de la brique.
- Le travail de Milan Renier, aussi étudiant de GCE, est de fournir des instructions de construction *drone compatible* via des plans existants BIM¹ ou une interface simple de construction. De plus, il fera le calcul de stabilité et l'ordre dans lequel chaque brique doit être posée. Le tout, bâti à partir des briques d'Amaury et d'Adrien.

1. Building Information Modeling : est un modèle commun entre tous les acteurs, de l'architecte au maître de chantier afin de diminuer les problèmes de communication entre ceux-ci

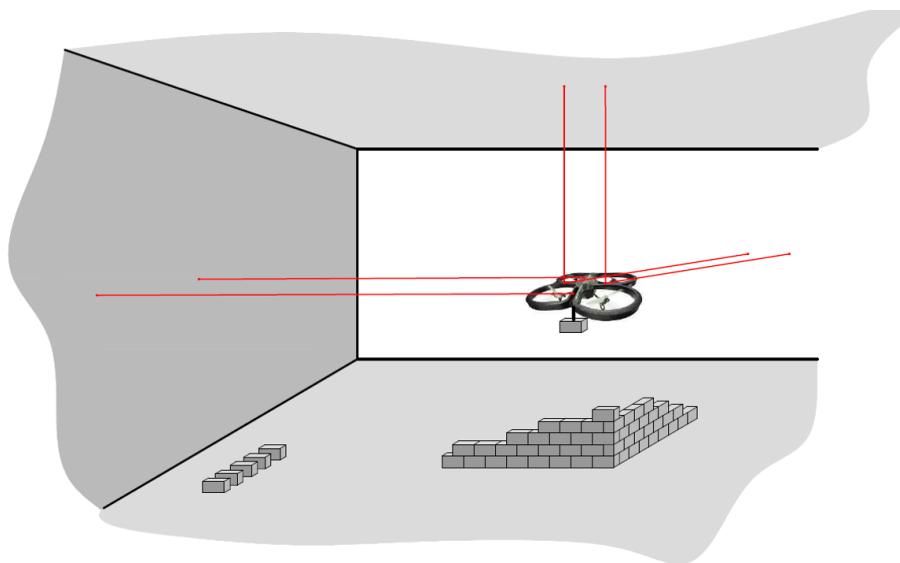


FIGURE 2.1 – Système avec 6 lasers proposé pour notre TFE

Dans le processus de construction, la forme du bloc est connue ainsi que son emplacement, mais il manque toute la partie contrôle et positionnement du drone. Cela dit, aucun système de positionnement automatique n'est encore mis en place. C'est pourquoi nous avons intégré le projet afin de localiser et positionner le drone dans un espace fermé.

Notre travail est de développer un système de localisation du drone en intérieur. Le système demandé est basé sur six lasers dirigés dans trois directions connues. Ce travail doit donc se charger du système de positionnement, mais également du contrôle du drone autonome.

2.2 Produit

De manière générale, beaucoup de solutions commerciales de positionnement existent, mais ce projet comporte différentes subtilités et particularités par rapport à un système classique qu'il faut éclaircir dès le départ. De plus, ce travail de positionnement d'un drone est la charnière entre les différents modules du produit final. Typiquement, la complexité des briques et de la pince dépend quasi exclusivement de l'imprécision maximale de positionnement du drone.

Cahier des charges exigé

- Être modulable
- Pouvoir être installé en intérieur
- Être précis de l'ordre de quelques centimètres
- Pouvoir être installé rapidement afin de réaliser des tests
- Être peu perturbé par l'environnement

Contraintes supplémentaires :

- Il faut considérer qu'il n'y a aucun signal GPS
- Il ne faut pas se fier au magnétomètre pour connaître l'orientation dans la pièce

Enfin, le projet a besoin d'un système de localisation pour la fin de cette année académique, quitte à être plus simple afin de pouvoir réaliser au plus vite les premiers tests du processus complet. Il a fallu tenir compte de toutes ces considérations lors des choix de matériaux et de la méthode de localisation.

2.3 Aspects juridiques

Avant de travailler sur une application impliquant les drones, il est évident qu'il faut s'intéresser à tous les volets juridiques qui peuvent être fort contraignants. De plus, dans le milieu des objets volants, il faut tenir compte des lois et des contraintes administratives puisque certaines demandes d'autorisation peuvent parfois prendre beaucoup de temps.

Dès le départ, la question de la faisabilité juridique de l'application a dû être posée. Il a ainsi fallu étudier la question et analyser le contexte politique pour savoir quelle serait la marge de liberté dans les tests et dans l'application finale.

Les principales raisons des blocages proviennent de la responsabilité civile et du respect de la vie privée. En effet, l'avantage principal d'un véhicule volant est d'avoir un autre point de vue, et donc prendre des photos, filmer ou tout simplement de survoler un terrain privé, une carrière ou une forêt. De plus, l'utilisateur peut perdre le contrôle du drone en plein vol à cause d'imprévus et causer des dégâts. Il doit donc être conscient de sa responsabilité au même titre qu'un conducteur automobile.

2.3.1 En Belgique

Dans un premier temps, il a fallu s'intéresser à la législation belge, car les développements et les tests se feront en Belgique.

En prenant en compte le fait que la législation évolue constamment la situation peut être synthétisée comme suit :

- Une demande d'autorisation à la DGTA devra être introduite avant chaque vol en extérieur.
- Un drone est considéré comme étant un drone de loisir en-dessous d'un kilo. Les drones de loisirs peuvent être utilisés sur des terrains privés avec une hauteur max de 10 m.
- Deux catégories ont été ajoutées pour les drones à usage professionnel : classe 1 et 2. La classe 1 correspond aux drones allant jusqu'à 5 kg et la catégorie 2 comprend les drones allant jusqu'à 150 kg. En fonction de la catégorie, la hauteur de vol varie ainsi que l'âge et le type de licence du télé-pilote.
- L'intervention d'un télé-pilote sera obligatoire sauf pour les drones à ailes fixe.
- Les activités d'épandage et de fret seront totalement interdites.
- Une assurance devra être souscrite par le propriétaire conformément à l'article 7 du règlement des assurances des transporteurs aériens.
- Une licence théorique et pratique devra être passée par le télé-pilote pour effectuer un vol.

Dans le cadre de ce projet, l'arrêté royal ne doit pas trop être inquiétant car pendant les débats, il n'a jamais été question de législation pour des drones indoor. Dès lors, ce travail visant une application d'intérieur qui pourra donc voir le jour puisque le drone pourra voler en toute légalité sans réelle demande d'autorisation.

Cela dit, la question de la législation intervient dans les tests de vols en extérieur lors du développement et interviendra si l'application vise à être utilisée pour un chantier en extérieur.



FIGURE 2.2 – Drone de test [19]

2.4 Drone de test et drone du projet

Tout au long de ce document, il sera question de parler d'un drone. Si cela n'est pas précisé, il faut considérer que c'est le drone de test.

En effet, deux drones étaient à disposition pour le travail de fin d'études. L'un était celui qui a servi pour le développement alors que l'autre, beaucoup plus imposant et encombrant n'a pas été utilisé pour le développement.

2.4.1 Drone de test

Le drone de la figure 2.2 est celui sur lequel tous les tests ont été réalisés. Il est de taille réduite, 40 centimètres de diamètre, de la marque DJI.

Travailler sur un drone moins imposant permet de faire de nombreux essais sans prendre le risque qu'une erreur occasionne des frais couteux. De plus, bien que le DJI 550 soit un hexacopter, il a été considéré comme une boîte noire, puisque de toute façon, le contrôleur de vol s'occupe de gérer les différentes configurations. Le frame du drone n'est qu'une configuration que le contrôleur prend en compte pour gérer sa régulation.



FIGURE 2.3 – Drone final du projet en pleine construction d'une colonne

Configuration :

- Nom : DJI 550
- Frame² : HEXA-X montré à la figure 2.4
- Poids avec matériel 2.4 kg
- Payload³ avec matériel : 0 kg
- Alimentation : 4S (14.6 Volts)
- Diamètre : 60 centimètres
- Hauteur sans pieds : 25 cm
- Taille des hélices : 20 cm de diamètre
- Matière : principalement plastique
- Prix : 550€

2.4.2 Le Drone final

Ce drone, pouvant porter jusqu'à 80 kg à pleine puissance⁴ est celui qui a été présenté dans les journaux ou reportages, ainsi qu'au TEDx UCLOUVAIN⁵.

C'est un drone commandé chez RC Take-OFF spécialement pour le projet de construction avec des drones. Le grand drone est un octocopter qui est dimensionné pour pouvoir soulever des éléments pouvant peser jusqu'à 40 kg, lorsqu'il a une hélice de moins. C'est pourquoi sa charge utile nominale est de 80 kg.

2. Le frame est le nombre de moteurs et leur disposition

3. Charge utile que le drone est capable de porter en plus

4. Notons qu'il est limité à 40 kg pour des raisons de sécurité

5. TEDx est un événement TED Talk indépendant, constitué de plusieurs conférences

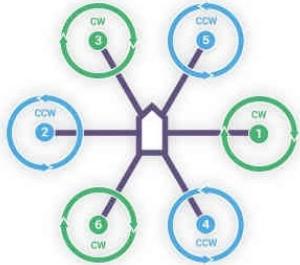


FIGURE 2.4 – Frame HEXA-X [11]

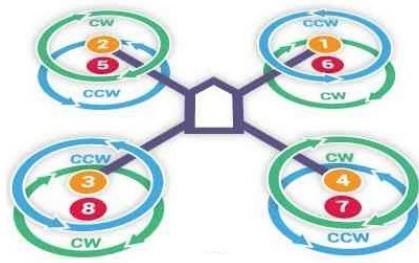


FIGURE 2.5 – Frame OCTO-QUAD [11]

Configuration :

- Nom : Grand Drone
- Frame : OCTO-QUAD⁶ montré à la figure 2.5
- Poids : 3 kg
- Payload : 40 kg
- Batterie autonomie : 2 batteries 6S, 22Ah, 30 minutes d'autonomie
- Alimentation : 12kW⁷
- Diamètre : 2 mètres
- Hauteur sans pieds : 25 cm
- Taille des hélices : 60 cm de diamètre
- Matière : principalement carbone
- Prix : 15 k€

2.5 Contrôleur de vol

Le contrôleur de vol est l'élément primordial d'un drone. En effet, c'est lui qui s'occupe de réaliser la régulation de 4 à 8 moteurs, de 25 à 400 fois par secondes, sur base d'un IMU et ce pour les 6 degrés de liberté du drone, afin de pouvoir réguler sa position.

L'IMU ou Inertial Measurement Unit est composé d'un accéléromètre, magnétomètre, un gyroscope ainsi que d'un baromètre. Ces composants permettent d'obtenir la position angulaire du drone, comme montré à la figure 2.8.

De plus, le contrôleur de vol peut proposer de faire la fusion des senseurs afin d'intégrer la position du drone. Typiquement, afin d'extrapoler la position du drone entre deux valeurs de position reçue via le GPS.

Enfin, il fournit en général un protocole de communication, que ce soit standard pour une télécommande ou via une connexion série.

6. OCTO-QUAD est un frame comportant 8 moteurs, mais disposés deux par deux, afin de pouvoir reprendre une rotation sur lui même si le multi-rotor perd une hélice.

7. Pour un ordre de grandeur, ceci équivaut à une bouilloire électrique, un micro-onde, le lave-vaisselle, le four électrique et les taques de cuisson, en même temps



FIGURE 2.6 – Contrôleur de vol - A2 [16]



FIGURE 2.7 – Contrôleur de vol - Pixhawk [24]

2.5.1 A2

Le contrôleur A2 est celui qui est installé de base sur le drone final pour le moment, pour tester ce drone, car l'A2 est considéré comme fiable. Il ne nécessite qu'une installation minimale et est destiné à une utilisation **ready-to-fly**. Il n'est configurable que pour des multi rotors et dispose de toutes les fonctionnalités principales pour effectuer 2 types de vol : Mode autonome avec GPS et les différents modes manuels tels que *Altitude Control*. Il est emballé dans une boîte métallique et n'est PAS configurable, tout est propriétaire, jusqu'au protocole du GPS. Dès lors, il est complètement impossible d'utiliser ce contrôleur de vol en vue d'une conception personnalisée hors GPS.

De plus son prix s'élève à 1300\$.

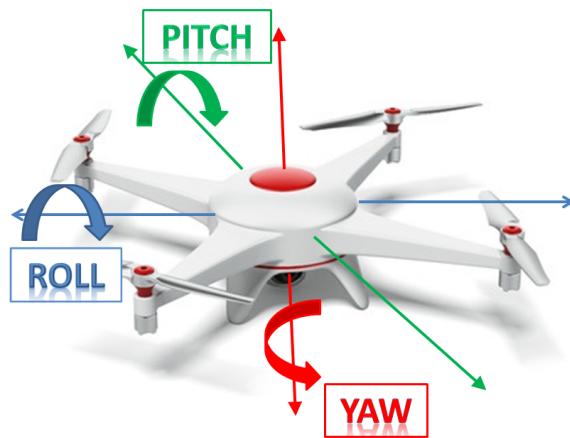


FIGURE 2.8 – État angulaire du drone récupéré de l'IMU

2.5.2 PixHawk

Le contrôleur de vol reçu pour ce projet est un PixHawk. Contrairement à l'A2, le PixHawk est sous licence libre, open source et open hardware.

Cela permet d'une part de pouvoir modifier le programme à sa guise, d'autre part, le code est disponible sur Github [15] et il est donc possible de directement discuter avec le développeur, collaborateurs et autres utilisateurs afin de régler des problèmes éventuels.

Dès lors, il est clair que ce contrôleur de vol donne tout ce qui est nécessaire pour pouvoir développer une application personnelle.

Voici quelques informations sur le PixHawk :

- Operating System : RTOS⁸
- Communication Onboard : UART/I2C/SPI/CAN, des canaux de communication série, permettant de brancher directement des capteurs optionnels sur le Pixhawk
- IMU : Composé d'un baromètre, accéléromètre et magnétomètre afin de déterminer la position angulaire du véhicule dans l'espace
- Protocoles : MavLink, u-blox (GPS)

8. RTOS : Real-Time Operating System

CHAPITRE 3

Cahier des charges et résultats

Abstract

Ce chapitre explique les finalités du projet complet, notre implication dans ce projet ainsi qu'une première vue des résultats obtenus.

3.1 Finalités du projet

Le projet de construction avec des drones de l'UCL est prévu d'être réalisé en plusieurs années et après de nombreuses recherches. Cette année fut le lancement de la réalisation d'un prototype comportant toutes les sous-parties nécessaires au projet final. Après plusieurs réunions de projet, la liste des délivrables de cette année a pu être définie clairement pour chacun des membres de l'équipe.

Dricks¹ Brique drone compatible

Pince adaptée au Dricks

Path planning Crée une suite d'instruction pour le drone

Système de localisation Détermine la position du drone

Système de contrôle Couche d'abstraction du drone

Interface de contrôle Affiche toutes les informations en temps réel du drone



FIGURE 3.1 – Prototype de briques autobloquantes

3.2 Cahier des charges

Au final, ce travail de fin d'études est constitué des trois derniers délivrables, soit le **système de localisation**, le **système de contrôle** et l'**interface de contrôle**

Dès lors, le **cahier des charges** est devenu :

- Contrôler le drone avec le contrôleur de vol PixHawk
- Tester la stabilité du PixHawk en vol normal ainsi qu'avec un moteur de moins
- Faire un état de l'art et explorer les différentes pistes pour positionner un drone dans un espace fermé
- Faire l'étude et la conception d'un système de vol autonome en intérieur afin de placer des dricks
- Développer un système de localisation pour une pièce de 10x10 mètres précis à l'ordre du centimètre

Tout en prenant en compte les **besoins spécifiques du projet** :

- Solution intérieure
- Modulable et extensible²
- Localisation relative à la pièce
- Simple, rapide, efficace

2. Afin de pouvoir être repris par les mémorants suivants

3.3 Résultats

Le résultat final du travail sur la localisation et le contrôle du drone, qui sera développé plus en détail par la suite, permet d'attribuer au drone les caractéristiques suivantes :

Module de localisation :

- Précision linéaire :
 - 5 centimètres en dynamique
 - 2 centimètres en statique (stabilisation)
- Précision angulaire :
 - 1.5° en dynamique
 - 0.5° en statique (stabilisation)
- Fréquence : 100 positions par seconde
- Système de coordonnées : ENU³
- Pièce de 20x20 mètres, rectangulaire, avec murs plats
- Prix du prototype : 700€

Système de contrôle :

- Contrôleur de vol : PixHawk
- Micro-ordinateur : Odroid XU4
- Système de tâches
- Système de communication extérieure
- Prix : 350€

Interface de contrôle :

- Affichage de la position du drone
- Affichage de la télémetrie
- Envoi d'une mission
- Contrôle manuel
- Monitoring des lasers
- Export des données de vol
- Prix : Non pertinent

3. East North Up

Le drone est donc capable à l'issu du développement de se déplacer dans un espace fermé vers une position donnée avec une précision inférieure à 5 cm. Il peut maintenir sa position ou effectuer un ensemble de tâches qui lui sont données.

Il est cependant *clair* que ce prototype n'est pas encore capable de réaliser un chantier, car le système de localisation est limité. Cela dit, il est prévu dans les années futures d'intégrer d'autres systèmes de localisations complémentaires et indépendants les uns des autres. Ainsi, dans le futur, différents systèmes pourraient être intégrés : un GPS RTK, une Caméra⁴ ou un laser tournant.

Les autres technologies ajouteront certaines capacités nécessaires sur un chantier telles que la robustesse, l'évitement d'obstacles ou l'évolution de l'environnement. Cette solution est expliquée en détail dans le chapitre 16 sur les perspectives futures.

4. À la suite du travail des étudiants de Julien Hendrickx, il sera possible d'obtenir une position locale via du SLAM et l'intégration de la position. De plus, plusieurs projets existent tels que ETH Zurich [12].

Deuxième partie

Étude

CHAPITRE 4

Architecture de contrôle

Abstract

Cette partie de l'étude commencera avec une explication des architectures de contrôle possibles afin de choisir la meilleure architecture. Elle est importante dans le sens où elle lie chaque module. Cela dit, l'architecture choisie sera reprise au chapitre 10, p62, plus en détail.

L'architecture de contrôle sert à définir comment le système fonctionnera. C'est l'architecture qui définit le rôle de chacun. Ainsi pour faire un vol autonome il est nécessaire d'avoir un ensemble de modules qui effectueront l'ensemble des tâches nécessaires pour le vol. Les différentes approches étudiées ont été :

1. **Tout sur le Pixhawk** : utilisation du contrôleur de vol en lui fournissant l'intelligence nécessaire pour faire un vol autonome.
2. **Arduino** : utilisation d'un microcontrôleur pour sous-traiter la partie commande. Le microcontrôleur se chargerait de commander le contrôleur de vol.
3. **Companion computer** : utilisation d'un ordinateur embarqué pour les mêmes raisons que l'arduino.

4.1 Tout sur le Pixhawk

La première architecture prise en compte a été la modification du contrôleur de vol afin d'y inclure le système de positionnement et les différents algorithmes. Cette architecture est rendue possible par le caractère open source de cette technologie. Néanmoins, elle est non seulement difficile à mettre en place mais en plus c'est la plus compliquée à reprendre dans un projet ultérieur.

En effet, il faut modifier le firmware¹ de base qui comprend actuellement des dizaines de milliers de lignes de code. Au moindre changement dans le firmware, il faut recompiler et recharger le firmware dans le Pixhawk.

C'est l'architecture la plus rapide à imaginer et à structurer, mais ce n'est pas la plus pratique pour réaliser un projet avec plusieurs équipes.

De plus, le développement est rendu difficile au niveau de l'intégration des tests².

4.2 Arduino

Lorsqu'il a fallu se détacher du Pixhawk et de travailler avec un organe de contrôle externe, la première solution qui fut explorée fut d'utiliser un microcontrôleur³ de type Arduino⁴

Celui-ci communique directement avec le Pixhawk. Il est en effet tout à fait possible de communiquer avec le contrôleur via le protocole de communication MavLink⁵.

Cette architecture reste assez simple, mais n'est pas assez modulable pour une application autonome. En effet, imaginons que quelqu'un doive ajouter un module, il devra réaliser son code sur la même Arduino et dans le même code, en risquant de mettre hors service le système de localisation. Avec cette architecture il est impossible de compartimenter le projet et de travailler avec un ensemble de modules indépendants. L'environnement de développement qu'offre un microcontrôleur n'est pas suffisant pour un projet aussi complexe.

Il a fallu une architecture beaucoup plus modulable, plus fiable et qui permette le monitoring par un intervenant extérieur.

4.3 Companion Computer avec ROS

Les deux premières solutions ne suffisaient pas afin d'obtenir un système qui soit pratique, modulable et simple d'utilisation.

Dès lors, l'autre solution était de rejoindre la communauté de développeurs de robots ROS. En effet, ROS⁶ est une grande communauté ayant déjà développé de nombreux outils permettant de simplifier la communication entre différents modules, ainsi que des outils [27] spécifiques à la robotique et de nombreux paquets [28] réalisés par la communauté.

-
1. Logiciel interne, fourni par le fabricant du matériel
 2. Par tests, cela sous-entend les tests unitaires utiles afin de s'assurer que tout soit fonctionnel et ne comporte pas des *bugs*
 3. Un microcontrôleur est le lien entre le monde physique et le monde du logiciel
 4. L'Arduino est un microcontrôleur ayant la particularité d'avoir une communauté très active et une grande base de données de librairies afin de faciliter sa programmation
 5. Protocole de communication utilisé avec le Pixhawk
 6. ROS : Robotic Operating System est une surcouche logiciel sur un système Linux permettant de contrôler les robots en général
-

C'est un environnement de travail très pratique dans un développement d'une application robotique.

En plus de fournir une excellente modularité de notre système en permettant de réaliser un code sous forme de modules, ROS donne aussi accès à une communauté active dont certains membres travaillent dans le même domaine. Cela donne accès à un réel support collaboratif.

Enfin, il y a eu des échanges notamment avec les équipes de TFE de Julien Hendrickx⁷.

Il en ressort que ces derniers développent désormais également sous ROS. Leurs développements seront donc compatibles avec un autre système ROS une fois fini.

4.4 Architecture choisie

Pour les raisons qui ont été décrites plus haut l'architecture finale comprendra ROS et se sépare en 3 segments :

1. **Ordinateur embarqué** : un ordinateur compagnon qui communique avec le Pixhawk, lui donne des consignes et réalise tous les calculs de haut niveau.
2. **Contrôleur de vol** : contrôle les moteurs⁸ afin d'assurer une bonne stabilité lors du vol.
3. **Ordinateur exporté** : c'est là qu'est l'interface de contrôle et qui dit au drone ce qu'il doit faire.

Les avantages principaux de la solution ROS qui est expliquée en détail au chapitre 7 :

1. **Facilité** : la programmation est aisée (python/C++) et possibilités de test unitaire
2. **Modularité** : séparation de tous les modules pour une meilleure compréhension de l'architecture globale
3. **Intégration** : l'intégration aisée de **nouveaux** modules
4. **Paquets** : de nombreux paquets fournissent des modules préprogrammés utilisables afin d'alléger notre tâche

7. Julien Hendrickx : Professeur à l'UCL et responsable d'une autre équipe de recherche travaillant sur des drones

8. Aussi aisément que cela puisse sembler, le contrôleur de vol doit réguler un engin *volant* ayant 6 degrés de liberté, et ce, plus de 100 fois par seconde

CHAPITRE 5

Méthodes de localisation

Abstract

Il sera question ici d'un état de l'art des différentes méthodes de localisation. Bien que le système à utiliser fut déjà décidé, il était important de dresser la liste des systèmes existants, car le système à installer sur le produit final au bout de 4 ans ne sera ni celui développé cette année, ni un des systèmes ci-dessous, mais une combinaison des systèmes pour augmenter la robustesse de la localisation ainsi que sa précision. La suite explique les raisons de l'utilisation de lasers fixes pour ce travail.

Les différentes méthodes de localisation classique pour un drone sont :

1. GPS
2. D-GPS
3. GPS-RTK
4. Ultrasons
5. IMU
6. Lasers fixes
7. Lasers tournants
8. Balises UWB
9. Station totale
10. SLAM
11. Localisation par multiples caméras
12. Vision integration

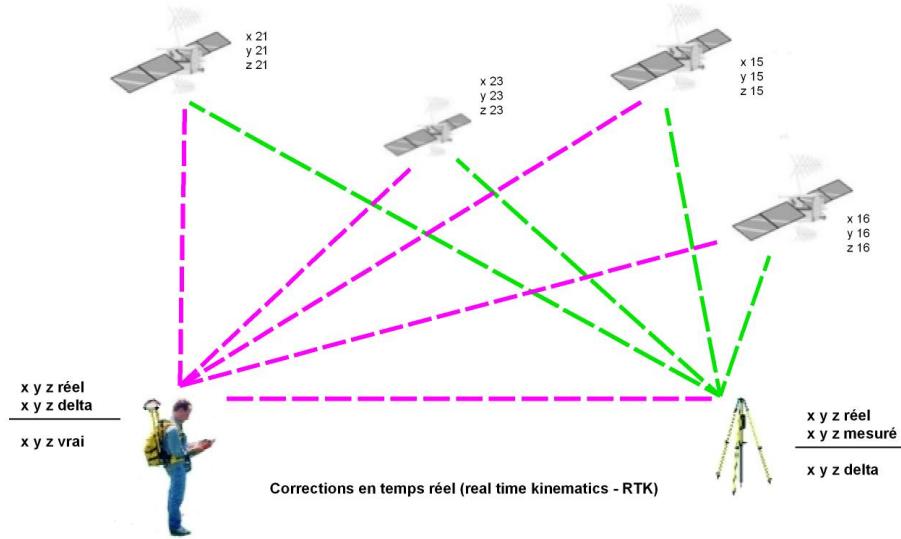


FIGURE 5.1 – Principe GPS et RTK [20]

5.1 GPS

Le GPS est une technologie basée sur le temps de trajet (ToF) d'ondes électromagnétiques provenant de satellites jusqu'au sol. Chaque satellite envoie son heure (provenant d'une horloge atomique) et sa position dans l'espace. Ainsi, le récepteur peut, avec 4 satellites¹ ou plus, déterminer sa position absolue (relative à la terre).

Avantages

- Fonctionne presque partout sur terre
- Donne une position absolue
- Un millionième de seconde d'erreur lors de la synchronisation provoque 300 mètres d'erreur

Inconvénients

- Peu précis (15m)
- Ne fonctionne pas en intérieur

1. Avec 3 satellites, il n'est pas possible de synchroniser l'horloge, or $1\mu s$ d'erreur équivaut à 300 mètres sur terre.

5.2 D-GPS

Le D-GPS (Differential GPS) prend en compte la position de GPS stationnaires, russes et européens. Cela permet d'avoir une précision de 10 à 20 cm dans les meilleures implémentations.

Avantages

- Bonne précision
- Sans abonnement

Inconvénients

- Ne fonctionne pas en intérieur
- Risques de blocage du signal GPS stationnaire
- Drift naturel et aléatoire (non corrigible) des satellites stationnaires

Utilisation

- Guidage d'un véhicule volant en extérieur
- Direction assistée d'un tracteur
- GPS d'une voiture
- Autoguidage

5.3 GPS-RTK

Le système RTK (Real Time Kinematic) se base sur le GPS pour obtenir une valeur plus précise. Mais en parallèle, une station localisée précisément reçoit le même signal GPS, qui a subi les mêmes perturbations dues à l'atmosphère. Cette station envoie donc une correction de la position en temps réel, afin que le dispositif corrige sa propre position. Il est possible d'arriver à une précision de 2.5 cm (+1cm/km de la station).

Avantages

- Très bonne précision

Inconvénients

- Abonnement (550€/ans) en Wallonie
- Risques de déblocage du signal GPS RTK
- Prix du hardware élevé

Utilisation

- Agriculture, autoguidage des tracteurs
- Guidage d'un drone en extérieur

Le principe du GPS et du GPS RTK est expliqué à la figure 5.1.



FIGURE 5.2 – Capteur à ultrasons [17]

5.4 Ultrasons

Le système ultrason (figure 5.2) est un système de mesure de distance typiquement basée sur le temps de parcours, ToF (Time of flight).

Avantages

- Capteurs peu chers
- Fonctionne en intérieur

Inconvénients

- Long temps d'acquisition
- Portée courte (2m)
- Vitesse du son modifiée par la température ou la composition de l'air

Utilisation

- Détection d'obstacles à courte portée

5.5 Lasers fixes

Le système de mesure *laser* se base souvent sur le temps de parcours (ToF), en faisant la corrélation d'un message codé envoyé et du message reçu, il est possible d'obtenir une précision au centimètre. Il faut noter que d'autres méthodes telles que l'angle du signal reçu sont aussi utilisées pour des distances plus faibles, mais elles n'ont pas été développées.

Avantages

- Temps d'acquisition faible (vis-à-vis des ultrasons)
- Portée maximale selon la méthode de mesure, de 6m, 10m, 50m jusqu'à 3km.
- Fonctionne en intérieur et en extérieur
- Peu dépendant de l'ambiance (selon la longueur d'onde du laser)

Inconvénients

- Coûts variables (125\$ grand public, plus de 3000\$ pour les professionnels)
- Danger potentiel pour les yeux selon la classe du laser
- Les lasers ne fonctionnent pas sur certaines surfaces (plexiglas ou vitres)

Utilisation

- Mesure de distance à longue portée
- Pointeurs de visée
- Station totale

5.6 Lasers tournants

Deux cas sont possibles, soit un laser fixe est fixé à un axe rotatif ou un miroir rotatif est fixé au-dessus du laser (figure 5.3). Dans tous les cas, le laser va prendre des mesures dans un plan 2D tout autour de lui. Avec une synchronisation entre le taux d'acquisition et la vitesse de rotation, il est possible d'avoir une certaine résolution angulaire. De plus, certains lasers tournants fonctionnent avec un jeu de miroirs et de multiples lasers de haute précision pour réussir à faire une carte en 3D.

Avantages

- Augmentation radicale du nombre de repères. Avec des lasers fixes, il est nécessaire d'avoir deux murs de référence, ici toute la pièce peut être balayée.
- Possibilité de ne plus considérer l'angle yaw (rotation selon Z d'un engin volant)
- Fonctionne en intérieur et en extérieur

Inconvénients

- Coûts élevés (600\$ grand public, plus de 7500\$ pour les professionnels)
- Danger potentiel pour les yeux selon la classe du laser
- Ne fonctionne pas sur certaines surfaces
- La correction selon l'attitude (position du véhicule selon pitch, yaw et roll) est compliquée à synchroniser.
- Le temps d'acquisition du laser utilisé doit être le plus faible possible.

Utilisation

- Découverte d'un lieu inconnu
- Mapping en 3D d'un silo ou d'un endroit



FIGURE 5.3 – Laser tournant RPLidar [7]

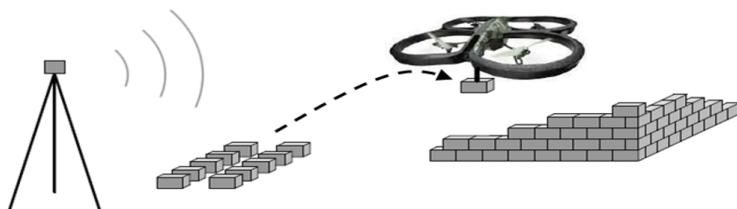


FIGURE 5.4 – Positionnement à l'aide d'une station totale

5.7 Station totale

Un traqueur suit le drone à la trace. Suivant l'angle précis avec lequel l'engin est visé, la position précise de la station et la distance mesurée jusqu'à l'engin volant, il est possible de déterminer la position exacte de celui-ci. Il est à noter qu'une station totale est beaucoup plus complète que la description ci-dessus, mais ce serait la seule option qui sera susceptible d'être utilisée.

Avantages

- Très grande précision (centimètre, voire millimètre, si la station est très bien positionnée)
- Très grande portée (2 km)
- Fonctionne en intérieur et en extérieur

Inconvénients

- Coûts très élevés (10000\$)
- Position initiale définie avec le GPS (RTK)
- Si plusieurs drones travaillent en parallèle, les stations totales risquent de viser le mauvais drone et de le perdre.

Utilisation

- Réalisation de mesures sur le terrain pour les géomètres

5.8 Balises UWB

La balise ultra larges bandes est un système de localisation en 3D, relatif à un point. Il faut placer trois balises dans une pièce, à des positions définies précisément. En fonction de la distance mesurée (ToF) entre les balises et le robot et par tri ou multilatération, il est possible de déterminer sa position. Ce système est utilisé pour améliorer la précision d'un autre système.

Avantages

- Position relative
- Un seul système au sol peut gérer plusieurs engins volants

Inconvénients

- Si le système n'est pas couplé à un autre moyen de localisation, il y a des risques de dérive.

Utilisation

- L'idée est similaire au système de positionnement GPS

5.9 SLAM

Simultaneous Localization And Mapping est un algorithme utilisé lors de l'exploration d'un lieu inconnu par un robot. Il peut être basé sur un système de caméra de profondeur, de caméras RGB, de lasers tournants ou autres. Grâce à la quantité importante d'informations sur ce qui l'entoure ainsi qu'à sa position, le robot crée une carte interne qui lui permet de se localiser dans ce nouvel espace. Pour le cas d'un multi rotor cette technique est difficile à réaliser, en effet, il est nécessaire d'avoir un système qui ne dépend pas de la façon dont le drone a exploré la pièce.

Avantages

- Position relative
- Apprentissage automatique du système
- Possibilité d'avoir un environnement évolutif

Inconvénients

- Difficile à mettre en œuvre en 3D

Utilisation

- Mapping de cavernes
- Voiture de Google autonome

5.10 Localisation via de multiples caméras

Il est possible de localiser le drone via des caméras qui filment le(s) drone(s). Ce système est presque plug 'n play, mais a été rejeté à cause du prix du système et par la difficulté de mise en place du système.

Avantages

- Fonctionne très vite une fois placé
- Fonctionne même avec plusieurs drones
- Fonctionne en espace fermé

Inconvénients

- Prend du temps à installer
- Prix : 10k-120k€

Utilisation

- Contrôle de drone en intérieur

5.11 Vision integration

C'est un système de localisation réalisé en intégrant les variations d'un flux vidéo. Le repère est local selon la position originale du drone.

Avantages

- Fonctionne très bien sous certaines conditions
- Est relativement peu cher
- Permet de développer la carte du terrain

Inconvénients

- Repère relatif au drone et non à la pièce
- Prix : maximum 1k€

Utilisation

- Contrôle de drone en intérieur

5.12 Choix réalisé

Parmi ces systèmes, plusieurs furent envisagés pour ce travail de fin d'études, les lasers fixes, les lasers tournants ainsi que les balises ultra larges bandes.

La méthode des lasers fixes fut choisie, avec le promoteur du projet, car cette solution est la solution la plus rapidement réalisable et la plus appropriée pour commencer le développement. Si il y avait uniquement le système de localisation à réaliser, alors la localisation aurait probablement été implémentée via des balises ultra larges bandes. Ce choix a été longuement discuté lors de réunions de projet et a été retenu pour le prototype.

De nombreuses autres possibilités sont imaginables mais l'idée est de concevoir un premier prototype qui soit fonctionnel et qui puisse être la base de développements futurs. Ce sujet sera traité plus tard mais la meilleure solution est de faire une fusion de différents moyens de localisations pour une meilleure fiabilité, redondance et précision. Le choix s'est donc porté sur les lasers fixes avec comme but d'avoir un système de localisation raisonnablement fiable et précis de l'ordre de 5 cm.

CHAPITRE 6

Choix du matériel

Abstract

Afin de réaliser un produit fonctionnel qui correspond au cahier des charges, le choix du matériel est important. C'est le but de ce chapitre, expliquer chacun des choix pris, mais surtout mettre en avant les avantages qui en découlent. Il sera question ici également des choix vis-à-vis des lasers, et de l'ordinateur de bord.

6.1 Lasers

Une fois que la méthode de localisation a été choisie, le cahier des charges fut établi. Le choix s'est porté sur un système de localisation à lasers fixes pour une pièce rectangulaire de 10x10 mètres, ainsi que des lasers de classe 1¹. Afin de faciliter le choix du matériel, une série de critères ont été mis en place.

6.1.1 Critères

Les critères sont triés par ordre d'importance

- Portée : plus de 10 mètres
- Taux de rafraîchissement : plus de 100Hz pour des lasers précis au centimètre, 10Hz pour des lasers précis au millimètre
- Prix : 100 - 500€
- Précision : Centimètre ou millimètre selon la fréquence
- Communication : possibilité d'interfaçage avec une Arduino ou une Odroid

1. Classe des lasers sans danger pour les yeux



FIGURE 6.1 – Laser lidar Lite V2 utilisé pour le système de localisation [21]



FIGURE 6.2 – Capteur de distance TeraRanger [22]

6.1.2 Différents modèles possibles

Il existe 3 gammes de lasers.

- Grand public : 100-200€ précis au centimètre
- Industrielle mais bon marché : 200-800€, précis au centimètre
- Industrielle et performant : 800-1500€, précis au millimètre

La gamme industrielle bon marché a donc été écartée, car elle ne fournissait rien de plus que la grand public. Etant donné que le prix de 6 lasers grand public est équivalent au prix d'un seul laser de précision, c'est le choix qui a été fait.

6.1.3 Choix modèle

Trois alternatives grand public existent : Teraranger, Ledgartech et LidarLite. C'est le LidarLite v2 (figure 6.1) qui fut retenu, car pour le même prix et les mêmes performances, il a une plus grande simplicité d'utilisation grâce au port de communication.

Caractéristiques du LidarLite v2 :

- Portée : 50 mètres
- Taux de rafraîchissement : 500Hz (250Hz pour une mesure moins bruitée)
- Prix : 125\$
- Précision : 2.5 cm
- Communication : I2C

6.1.4 Avantages

Le LidarLitev2 est un laser à vitesse d'acquisition rapide (jusqu'à 500Hz), ce qui permet de filtrer les échantillons afin d'ôter l'erreur aléatoire et d'avoir des mesures plus précises. Ainsi, il y aura deux modes, l'un moins précis lorsque le drone est loin de la position



FIGURE 6.3 – Différents modèles d'arduino [6]

voulue, et un second lorsque le drone a une dynamique faible, afin de bénéficier d'un filtrage augmentant grandement la précision des lasers.

6.2 Arduino

Une Arduino est une carte comportant un microcontrôleur. Un microcontrôleur permet de réaliser un lien entre le monde physique et informatique. Ainsi, il est possible d'envoyer des états logiques en dehors du monde informatique, de réaliser des mesures du monde réel et de communiquer avec d'autres composants.

L'Arduino est utilisée en tant qu'élément d'acquisition du système de lasers afin d'en faire abstraction par la suite. L'Arduino communique avec tous les lasers et envoie les mesures à l'ordinateur de bord.

6.2.1 Choix du modèle

Il existe de nombreux modèles (figure 6.3), et il faut donc choisir l'Arduino qui correspond au mieux à nos besoins.

Caractéristiques possibles :

- Tension de fonctionnement : 3v3, 5v
- Fréquence : 16MHz, 86MHz
- Nombre d'entrées sorties : 16, 20, 54
- Mémoire flash (contenant le programme) : 32 Ko, 256 Ko, 512 Ko
- Mémoire RAM : 2 Ko, 8 Ko, 96 Ko

Avec une bonne connaissance des différentes Arduino, il est simple de choisir le modèle qui correspond aux besoins du projet. Dans ce cas-ci, le programme pèse 60 Ko, il n'y a donc eu le choix qu'entre l'Arduino Due et l'Arduino Mega. La première est plus rapide (86MHz,

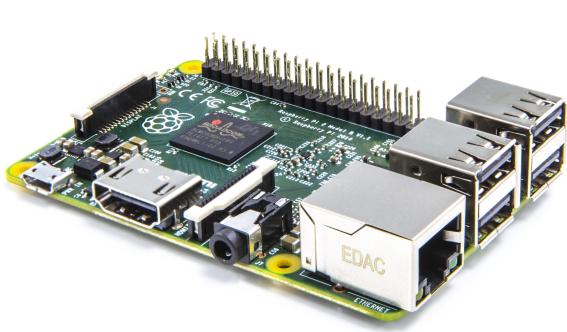


FIGURE 6.4 – Raspberry Pi [25]



FIGURE 6.5 – Odroid [23]

32 bits) que la seconde (16MHz, 8 bits) mais l’Arduino Due comporte un problème de communication pour le port de communication à utiliser (I2C). Le choix s’est donc porté sur l’Arduino Mega.

6.3 Ordinateur de bord

Un micro-ordinateur se diffère d’un ordinateur classique par une architecture de processeur basse consommation (ARM) et par sa taille proche d’une carte de banque. En effet, un tel ordinateur a la taille d’une carte de banque sur lequel tourne un système d’exploitation Linux.

À nouveau, de nombreuses cartes de développement existent, dont les plus connues sont les RaspberryPi (figure 6.4) et les Odroid (figure 6.5).

6.3.1 RaspberryPi

RaspberryPi est une compagnie sans but lucratif. C’est pourquoi les RaspberryPi sont à des prix démocratiques. De plus, une grande communauté est derrière chacune des versions des RaspberryPi. Il faut compter 55\$ pour une RaspberryPi2 avec une carte WiFi et carte SD.

6.3.2 Odroid

Hardkernel est une compagnie sud-coréenne avec but lucratif, mais dont les designs sont beaucoup plus performants que ceux de RaspberryPi. Il faut compter 110\$ pour une Odroid-XU4 avec WiFi et mémoire eMMC (figure 6.6).

6.3.3 Choix

Le choix s’est porté sur l’Odroid-XU4 car c’était celle qui offrait les caractéristiques les plus adéquates pour ce projet. De plus, les performances par dollars d’une Odroid- XU4



FIGURE 6.6 – Carte EMMC (rouge) et l'adaptateur eMMC vers MicroSD (bleu) [18]

sont doublés par rapport à la RaspberryPi 2. En effet, la mémoire eMMC (110 Mo/s) est beaucoup plus rapide que la carte SD (10 Mo/s) de la Pi2 et le processeur de l'Odroid-XU4 est 4 fois plus performante. Pour des performances plus intéressantes l'odroid était donc le choix le plus judicieux.

6.3.4 Choix du système d'exploitation

Le système d'exploitation est Ubuntu. En effet, le logiciel qui est utilisé fonctionne sans problème sur Ubuntu, dont ROS (Robotic Operating System) qui sera abordé plus tard au chapitre 7.

CHAPITRE 7

ROS

Abstract

Bien que cela ne soit pas une véritable étude de ROS, Robotic Operating System, il peut être nécessaire d'expliquer les bases. Ce chapitre est surtout utile pour toutes les personnes voulant en apprendre plus sur cette surcouche.

7.1 Introduction

Avant la création du système d'exploitation ROS, chaque application robotique devait développer son propre environnement de programmation. Cela impliquait que certaines fonctionnalités étaient sans cesse redéveloppées et c'est pourquoi une idée de créer un système qui regroupe les fonctionnalités basiques de robotique est née. Comme dit précédemment, ROS, Robotic Operating system est une surcouche logicielle open source sur Linux qui représente un répertoire d'applications. ROS peut être représenté comme un ensemble d'outils destinés aux applications robotiques.

7.2 Principe

Afin d'éviter toute confusion et de rendre la compréhension plus aisée, ce chapitre se penche sur la terminologie utilisée. Dans une application robotique de nombreux exécutables fonctionnent en parallèle et s'échangent des informations. Dans ce projet, de nombreux acteurs et développeurs réalisent ou réaliseront des applications. C'est pourquoi il a fallu trouver une architecture permettant une implémentation aisée de modules indépendants.

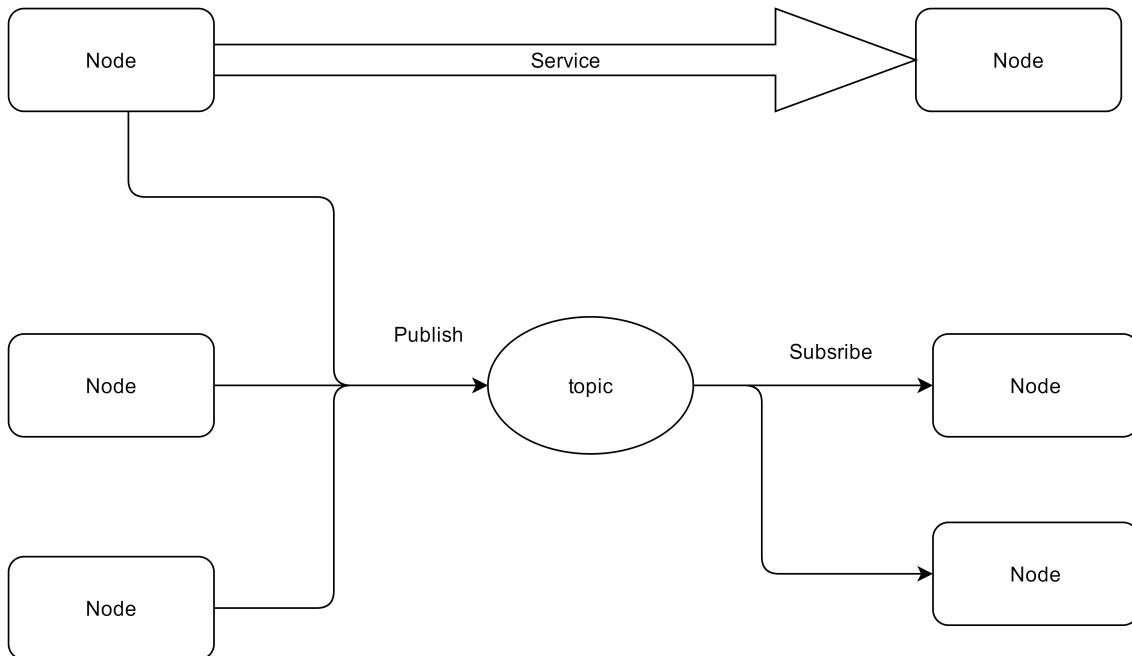


FIGURE 7.1 – Fonctionnement de ROS

Terminologie :

- **Roscore** : Application principale qui gère toutes les nodes du système. Cette application doit être lancée au démarrage pour commencer à exécuter les fonctions propres à ROS.
- **Node** : Une application qui a été lancée, qui a un nom, et qui est capable de communiquer avec d'autres nodes via des topics. Un nœud peut être par exemple une information venant d'un système de mesure ou un système de traitement de données.
- **Topics** : Le nom du sujet de discussion. Tous les messages envoyés sur un même topic doivent être le même type de message. Un node ou plusieurs node peuvent publier vers un topic ou recevoir des informations de celui-ci.
- **Suscriber** : Une node inscrite à la réception d'un message sur le topic
- **Publisher** : Une node inscrite à la publication d'un message sur le topic et qui peut donc publier un nouveau message
- **Services** : Communication Many to One, une node peut fournir un service. Un service est un type de communication synchrone qui requiert une réponse de la part du programme qui le fournit.
- **Workspace** : Espace de travail dans lequel les paquets sont placés en cours de développement.
- **Source** : Charger les variables d'environnement de ROS ou d'un workspace

Ainsi comme présenté à la figure 7.1, un nœud publie (publisher) sur un topic et de l'autre côté, un ou plusieurs nœuds s'y souscrivent (subscriber). Sur les topics transitent des messages prédéfinis (pouvant être complexe). Dès que le ou les publishers publient une information, tous les subscribers recevront le message et pourront le traiter.

Un message en tant que tel peut être très simple (un nombre) comme très compliqué (l'état complet du drone).

7.3 Environnement

L'environnement ROS donne accès à un nombre incalculable de librairies et de paquets. Le nombre de librairies utilisées n'est pas très grand, mais elles ont été des outils très pratiques.

Paquets utilisés :

1. **TF** : Librairie de transformations entre différents types de données
2. **Mavros** : Hardware abstraction layer du PixHawk. Cette librairie permet de communiquer avec le contrôleur de vol pour récupérer des informations ou bien pour commander le drone.
3. **ROSSerial** : Communication série avec l'Arduino. Cette librairie permet d'envoyer ou de recevoir des messages via les topic avec une connexion USB.
4. **Rosbridge** : Communication JSON avec un élément non-ROS. Il permet de créer un canal de communication entre une interface WEB et le drone via un Web Socket.

Il faut utiliser un workspace, un espace de travail dans lequel tout le développement est réalisé. C'est dans ce workspace que les tests sont effectués et qu'il faut ajouter des messages personnalisés ou des scripts de lancement de paquet.

Un message sert à communiquer avec d'autres nœuds, programmes, via un type défini. Par exemple, afin de partager l'état angulaire du drone, le message utilisé contient le Pitch, Yaw et Roll encodées en float32¹.

1. Format de donnée d'un nombre à virgule flottante sous 32 bits

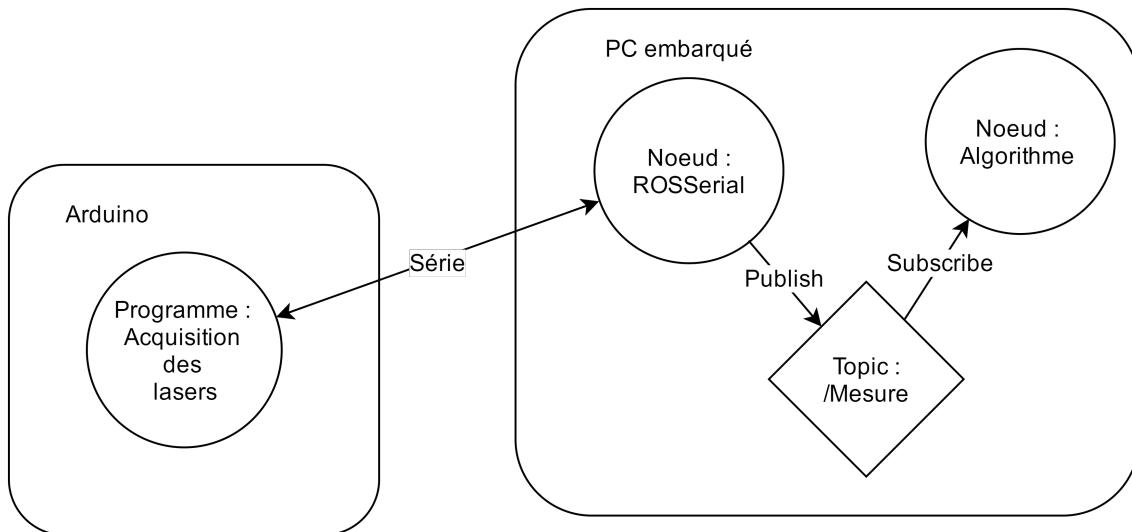


FIGURE 7.2 – Exemple d’architecture sous ROS

7.4 Exemple

La figure 7.2 illustre bien un exemple d’utilisation de ROS. Un programme sur un micro-contrôleur effectue 100 mesures par secondes. Ces mesures doivent être transmises au système embarqué.

Un programme d’acquisition tourne sur le micro-contrôleur et réalise les mesures de distance lasers.

Le nœud ROSSerial crée un canal de communication avec le micro-contrôleur. Une fois récupérées, les informations sont publiées sur le topic de mesures */Mesure*.

L’information est désormais disponible pour tous les modules qui le désirent. Ici, le nœud d’algorithmique récupère les informations de distances et applique l’algorithme de localisation.

7.5 Conclusion

En découvrant cet environnement, il a été évident qu'il fallait créer un système avec cet outil. ROS offre une facilité d'intégration qui n'est pas présente dans les autres solutions envisagées.

Avantages :

1. **Communication** : l'échange d'informations entre les différents modules est facilité par l'environnement de topics.
2. **Modularité** : si plusieurs personnes réalisent des modules pour une même application, les développements peuvent être complètement indépendants. L'intégration se fera de façon transparente car la communication est déjà réalisée.
3. **Communautaire** : le choix de ROS s'est fait aussi pour son aspect communautaire qui permet l'échange entre les différents développeurs. L'installation ainsi que les bases à connaître pour se lancer sont bien documentées.
4. **Mavros** : une surcouche logicielle à MAVLink² est déjà présente sous ROS pour faciliter l'utilisation de contrôleurs de vols tels que le PixHawk.
5. **Multi langages** : différents langages (Python, C++, C, etc..) sont disponibles afin de mieux correspondre aux besoins spécifiques de chaque application.

2. Protocole de communication utilisé pour communiquer avec le contrôleur de vol.

Troisième partie

Réalisation

CHAPITRE 8

Vue d'ensemble de la réalisation

Après avoir réalisé l'étude du système et choisi le matériel à utiliser, le développement peut commencer. Les grandes étapes de cette partie ont été séparées en modules séparés pour plus de clarté. Ainsi, les trois grands chapitres abordés sont les suivants.

1. **Localisation** dans lequel toutes les étapes du développement du système de positionnement avec les lasers est expliquée.
2. **Contrôle** qui explique l'architecture du système de contrôle en mettant en avant les différents modules implémentés.
3. **Interface** montrant ce qui a été réalisé en matière d'interface graphique afin de simplifier l'utilisation et la supervision du système tout en offrant un exemple concret au groupe qui devra reprendre cette partie.

CHAPITRE 9

Localisation

Abstract

Le premier chapitre de la réalisation traite du système de localisation basé sur des lasers fixes. Il explique la méthode utilisée afin d'obtenir la meilleure position de la part des lasers, malgré les 2.5 centimètres d'erreur pour chacune des mesures, en approfondissant l'algorithme réalisé ainsi que les différents filtrages.

9.1 Méthode de localisation

Trois paires de lasers fixes sur le drone sont utilisées afin de le localiser. Les lasers visent des murs de référence par paire. En 2D, c'est très intuitif. En corrigeant la mesure laser selon l'angle au mur en réalisant $distance_x = measure_x * \cos(\psi)$. En portant ce résultat en 3D, la mesure peut être corrigée par le biais de la mesure de l'orientation du drone réalisée par le contrôleur de vol.

Il est important de noter que cette méthode est très dépendante de l'angle yaw¹. C'est-à-dire que si un des lasers ne pointe plus vers son mur de référence, la valeur qu'il renverra ne sera pas pertinente. Or étant en intérieur, l'intensité et la direction du champ magnétique sont atténusés et peuvent varier selon la position du drone dans la pièce. Les deux autres angles, roll et pitch, sont fiables, comme montrés dans les tests, à la figure 14.3. L'angle yaw est donc calculé selon les lasers.

Le but de cette partie est de déterminer la position du drone ainsi que de limiter ou compenser les différentes sources d'erreurs.

1. yaw est la rotation autour de l'axe Z. Il est parfois appelé heading et ψ pour les équations.

Erreurs systématiques

- Offset des lasers ($10cm$)
- Linéarité des lasers (99.5%)
- Position des lasers sur le drone ($0.3cm$)
- Orientation des lasers sur le drone (1°)

Erreurs aléatoires

- Imprécision de la mesure des lasers ($2.5cm$)
- Imprécision de la vitesse et accélération du drone (retard de l'information)
- Imprécision de l'orientation du drone (retard de l'information)
- Imprécision de la planéité des murs² ($0.5cm$)
- Perturbations extérieures

Afin de simplifier le problème, la suite sera séparée en plusieurs parties, acquisition des mesures, filtrage d'entrée, détermination de l'angle ψ , détermination de la position, filtrage de sorties et résultats.

9.2 Besoin de l'algorithme

Les données en entrée et en sortie de l'algorithme de positionnement sont :

Entrées

- Pièce rectangulaire de 10×10 mètres
- L'origine du repère est à l'intersection des trois murs de référence.
- Position fixe des lasers sur le drone
- Orientation fixe des lasers sur le drone
- Mesures des lasers précises à $2.5cm$
- L'orientation du contrôleur de vol (PixHawk)

Sorties

- Position en X, Y et Z
- Angle yaw

2. La planéité est une erreur systématique, si le drone est stationnaire, mais aléatoire si le drone se déplace

9.3 Acquisition des lasers

Le système se base sur l'utilisation de 6 lasers, mais afin de les avoir à moindre prix³, ce sont des lasers grand public. La différence majeure est qu'ils ne sont pas calibrés, la première chose à faire a donc été de les calibrer, comme expliqué au chapitre 13 sur les tests des lasers.

Cela dit, un second problème est intervenu : la librairie fournie permettant de lire les valeurs des lasers était inutilisable dans un contexte où le système de localisation doit être robuste. En effet, si la lecture ne réussissait pas, alors le programme réessayait, jusqu'à 9999 fois, dès lors, si un laser ne répondait plus, alors le programme restait bloqué et aucune autre valeur n'était lue. De plus, les mesures sont prises de façon synchrone. C'est à dire que le programme attendait la fin de la mesure du laser avant de lancer mesure suivante.

Afin d'avoir une grande robustesse, le développement d'une librairie adaptée était donc inévitable. Cette librairie n'utilise pas toutes les possibilités des lasers, mais ayant les caractéristiques **nécessaires** afin d'avoir un système de mesure robuste. La première étape est de déterminer les besoins réels pour la partie acquisition.

Caractéristiques voulues

- Acquisition asynchrone
- Redémarrage et reconfiguration des lasers automatiques si trop d'erreurs de lectures surviennent
- Ajout ou suppression aisée de lasers afin de garder le côté modulable et réutilisable

Toutes ces problématiques ont été réglées par l'implémentation d'une machine à états (fig :9.1) pour chacun des lasers. Dès lors, chaque laser peut, indépendamment des autres, se remettre à zéro, faire une acquisition, se configurer, etc. Ainsi, la fréquence d'acquisition a été optimisée et le système a été rendu plus robuste.⁴

9.3.1 Carte de connexion

Une fois que les tests de communications avec les lasers étaient concluants, il a été nécessaire d'augmenter la robustesse de la connexion. Pour ce faire, une carte électronique pouvant se connecter directement sur le microcontrôleur (Arduino) a été réalisé.

3. Pour rappel, le prix des lasers est de 125\$ contre 400\$ pour les mêmes performances en gamme professionnelle

4. La librairie Arduino qui a été développée pour le projet est disponible sur Github [26].

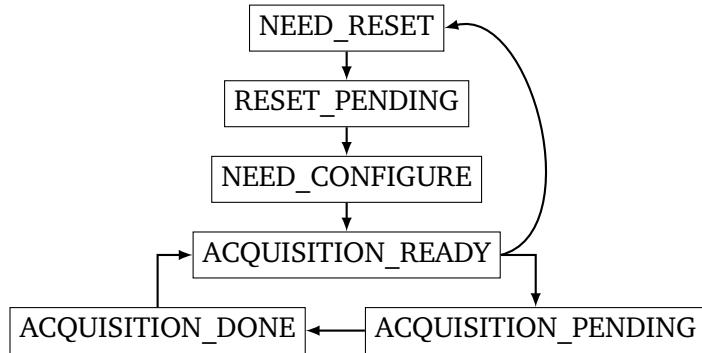


FIGURE 9.1 – Machine à états implémentée pour chaque laser

La carte :

- Connecte 6 lasers simultanément à l'Arduino
- Inclus les circuits spécifiques de chaque laser
- Inclus deux level shifters bidirectionnels, afin de convertir le bus de donnée de 5 volts à 3.3 volts.

Pour des informations supplémentaires, voir Annexes.

9.4 Correction d'entrée

Les lasers ont été testés pour déterminer leurs performances. Pour plus d'informations sur la démarche de réalisation des tests, voir partie *Tests et performances* au chapitre 13. Dès lors, les caractéristiques des lasers sont connues et trois paramètres importants pour chacun des lasers ont été déterminés.

Le premier concerne la **linéarité**, ils sont tous très linéaires, de l'ordre de 0.999. Dans un premier temps la linéarité ne sera pas corrigée.

Le second est le **offset** de chaque laser, la moyenne de la différence de la mesure et de la valeur réelle. Il est différent pour chacun des lasers et est de l'ordre de 10 centimètres. C'est donc la première correction à réaliser. (9.1)

$$(9.1) \quad offset = mean(measure_{laser} - measure_{real})$$

Le troisième paramètre est la **répartition gaussienne** autour de la moyenne. Il existe une corrélation entre les mesures, ce qui implique qu'il est donc théoriquement possible d'améliorer la mesure en réalisant une moyenne. C'est la seconde correction qu'il faut réaliser.

Le système de mesure est exprimé comme suit :

$$\text{mesure} = \text{measure}_{\text{real}}) + \text{offset} + \text{error}_{\text{random}}$$

9.4.1 Correction du offset

L'offset étant mesuré, l'équation de correction n'est pas compliquée. Il suffit, en entrée du système de soustraire l'offset à la mesure des lasers.

$$\text{Output} = \text{measure} - \text{offset}$$

9.4.2 Filtrage d'entrée

Un test de contrôle avec des données brutes non filtrées a été réalisé, mais cela, rend le drone instable et il oscille. C'est pourquoi il a été jugé nécessaire de filtrer les mesures avant de les traiter.

9.4.2.1 Filtre à réponse impulsionale finie (FIR)

Tandis que la moyenne est stable, les mesures varient beaucoup⁵ autour de la moyenne. Le filtrage d'entrée est donc un filtre de lissage, appliqué à tous les lasers qui sert à diminuer l'erreur aléatoire qui est de distribution normale.

La première idée était un filtre passe-bas à réponse impulsionale finie (FIR), mais un filtre de 10 échantillons par exemple aurait typiquement induit 50ms⁶ de retard. Avec un modèle du système à 6 degrés de liberté, la réactivité est très importante et chaque échantillon doit être à la valeur correcte à tout instant. Avec ce filtrage, le contrôleur de vol serait dans la situation :

5. Les variations sont de l'ordre de 2 centimètres

6. Sachant que le retour d'information des lasers ne se fait que 100 fois par seconde, 10 échantillons reviennent à 50 millisecondes de retard

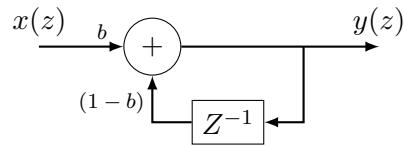


FIGURE 9.2 – Schéma d'un filtre IIR à un pôle

9.4.2.2 Filtre à réponse impulsionnelle infinie (IIR)

Le choix le plus pertinent est donc un filtre à réponse impulsionnelle infinie (IIR) à un pôle. En effet, c'est un filtre très simple, qui ne prend en compte que la valeur de sortie précédente et la valeur actuelle, ce qui permet de ne pas incorporer trop de retard sur l'information tout en ayant des performances supérieures au filtre FIR. Comme le décrit la figure 9.2, le seul paramètre à régler est b , qui détermine l'importance donnée à la nouvelle donnée par rapport à la sortie précédente. Afin de déterminer le facteur, un site web de référence de design de filtre [10] permet de comparer plusieurs valeurs afin de déterminer un paramètre b limitant les oscillations aléatoires sans trop diminuer les variations.

Le paramètre b a donc été défini afin d'avoir un signal proche de la bonne valeur lorsque le drone est en vol stationnaire, et peu retardé⁷. De plus, un drone peut fonctionner dans deux modes différents qui nécessitent des réglages différents. Ces modes sont le vol **stationnaire** à faible dynamique et le vol **mobile** à grande dynamique. Dès lors, il a fallu implémenter un changement dynamique du pôle selon qu'il doive rester sur place ou si il souhaite se déplacer. Cela assure, au prix d'un signal plus bruité, que la sortie soit optimisée à la mission du drone.

7. Le retard en pratique avec le filtrage est au maximum de 20 ms lorsque le drone se déplace à 40 centimètres par seconde

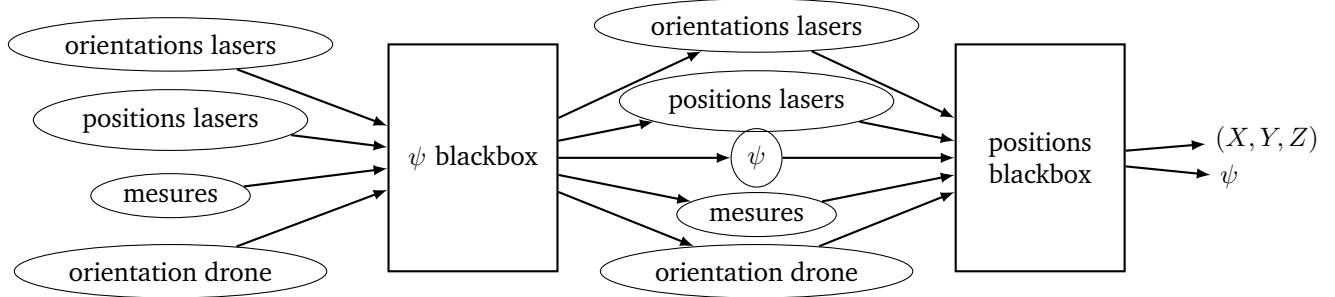


FIGURE 9.3 – Imbriquement des algorithmes de positionnement

9.5 Algorithme de positionnement

Comme cela a été précisé précédemment, l'algorithme de positionnement 9.3 est séparé en deux parties distinctes :

- La première sert à déterminer l'angle yaw
- La seconde sert à corriger les mesures par l'orientation du drone

9.5.1 Détermination de l'orientation du drone

En deux dimensions, une équation avec un arc-tangente suffit pour avoir la distance par rapport au mur de référence. En trois dimensions par contre, les angles roll et pitch influent aussi sur la mesure, ce qui signifie que si le drone n'est pas stable, alors ces deux angles doivent être pris en compte.

Mettre la problématique en équation n'est pas aisée, il faut penser au problème d'une manière différente. Le **repère** pour les calculs doit être changé. Il a été défini comme un repère à orientation statique, qui est au centre du drone et bouge avec le drone. De plus, à la place de visualiser une mesure qui est corrigée, désormais c'est la **mesure d'une distance du drone à un point du mur** qui est calculée. Dès lors, la composante X de ce point est la distance au mur X si le laser vise le mur X.

En pratique, il faut déterminer **le point visé par le laser en fonction de ψ , l'angle yaw**. Pour commencer, le laser a une position et une orientation fixée sur le drone. La première étape est d'appliquer la rotation selon les axes x et y, selon les angles roll et pitch. Ainsi, le résultat donne la position presque réelle sur le drone. En effet, la rotation selon l'angle yaw reste encore à être déterminée. De là, travailler en paramétrique permet de déterminer l'équation finale.

Soit P et V la position et le vecteur d'orientation du laser qui ont subi la rotation selon

roll et pitch.

$$(9.2) \quad V = \begin{bmatrix} Vx \\ Vy \\ Vz \end{bmatrix}$$

$$(9.3) \quad P = \begin{bmatrix} Px \\ Py \\ Pz \end{bmatrix}$$

Maintenant il faut réaliser la rotation R de P et V selon l'axe Z, soit ψ l'angle yaw.

$$(9.4) \quad R = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$(9.5) \quad P * R = PR = \begin{bmatrix} -\sin(\psi) * Py + \cos(\psi) * Px \\ \cos(\psi) * Py + \sin(\psi) * Px \\ Pz \end{bmatrix} = \begin{bmatrix} PRx \\ PRy \\ PRz \end{bmatrix}$$

$$(9.6) \quad V * R = VR = \begin{bmatrix} -\sin(\psi) * Vy + \cos(\psi) * Vx \\ \cos(\psi) * Vy + \sin(\psi) * Vx \\ Vz \end{bmatrix} = \begin{bmatrix} VRx \\ VRy \\ VRz \end{bmatrix}$$

Soit M la mesure du laser et K , le facteur par lequel multiplier l'orientation du laser afin de savoir la cible qu'il touche. En pratique, le dénominateur est la longueur du vecteur orientation et est calculé au début, car les rotations ne changent pas sa longueur.

$$(9.7) \quad K = \sqrt{\frac{M^2}{VRx^2 + VRy^2 + VRz^2}}$$

Désormais, le point ciblé C par extrapolation peut être déterminé par calcul.

$$(9.8) \quad C = K * VR - PR = \begin{bmatrix} K * VRx - PRx \\ K * VRy - PRy \\ K * VRz - PRz \end{bmatrix} = \begin{bmatrix} Cx \\ Cy \\ Cz \end{bmatrix}$$

Menant que le point ciblé sur le mur sont connus, $C(\psi)$ peut être retrouvé.

$$(9.9) \quad C(\psi) = \begin{bmatrix} K * VRx - PRx \\ K * VRy - PRy \\ K * VRz - PRz \end{bmatrix} = \begin{bmatrix} -K * \sin(\psi) * Vy + \cos(\psi) * Vx + \sin(\psi) * Py - \cos(\psi) * Px \\ K * \cos(\psi) * Vy + \sin(\psi) * Vx - \cos(\psi) * Py - \sin(\psi) * Px \\ K * Vz - Pz \end{bmatrix}$$

Sachant que le point $C(\psi)$ et la normale du mur $(1, 0, 0)$ sont connu, dans le cas où le laser vise le mur X :

$$(9.10) \quad mur \equiv ax + by + cz + d = 0$$

Avec une normale valant $(1, 0, 0)$ et passant par l'origine $(0, 0, 0)$

$$(9.11) \quad mur \equiv x = 0$$

Sachant que le point $C(\psi) \in mur$

$$(9.12) \quad mur \equiv (-K * \sin(\psi) * Vy + \cos(\psi) * Vx + \sin(\psi) * Py - \cos(\psi) * Px) * x = 0$$

Il faut réaliser cela également pour le second laser afin d'obtenir deux équations en fonction de ψ .

$$(9.13)$$

$$mur_1 \equiv (-K_1 * \sin(\psi) * Vy_1 + \cos(\psi) * Vx_1 + \sin(\psi) * Py_1 - \cos(\psi) * Px_1) * x = 0$$

$$(9.14)$$

$$mur_2 \equiv (-K_2 * \sin(\psi) * Vy_2 + \cos(\psi) * Vx_2 + \sin(\psi) * Py_2 - \cos(\psi) * Px_2) * x = 0$$

Cela se définit par deux équations à deux inconnues, et sachant que le même mur est visé par les deux laser, ces deux équations peuvent être égalisées.

$$(9.15) \quad -K_2 * \sin(\psi) * Vy_2 + \cos(\psi) * Vx_2 + \sin(\psi) * Py_2 - \cos(\psi) * Px_2 = \\ -K_1 * \sin(\psi) * Vy_1 + \cos(\psi) * Vx_1 + \sin(\psi) * Py_1 - \cos(\psi) * Px_1$$

En utilisant XMaxima [29], cette équation peut être simplifiée afin d'obtenir le résultat final, l'équation pour trouver l'angle ψ .

$$(9.16) \quad \frac{\sin(\psi)}{\cos(\psi)} = \frac{-K_2 * VRx_2 + K_1 * VRx_1 + PRx_2 - PRx_1}{-K_2 * VRy_2 + K_1 * VRy_1 + PRy_2 - PRy_1}$$

$$(9.17) \quad \psi = \text{atan}\left(\frac{-K_2 * VRx_2 + K_1 * VRx_1 + PRx_2 - PRx_1}{-K_2 * VRy_2 + K_1 * VRy_1 + PRy_2 - PRy_1}\right)$$

L'imprécision théorique maximale à laquelle s'attendre avec une orientation parfaite des lasers est :

$$(9.18) \quad \delta\psi = \text{atan} \frac{\delta M}{\text{distance}_{\text{lasers}} - \delta \text{distance}_{\text{lasers}}}$$

$$(9.19) \quad = \text{atan} \frac{\delta M}{\sqrt{(Px_1 - Px_2)^2 + (Py_1 - Py_2)^2} - \delta \text{distance}_{\text{lasers}}}$$

Sachant qu'en pratique, la distance inter lasers est de 49cm et avec des lasers avec un $\sigma = 1.5\text{cm}$, **sans filtre** l'imprécision théorique maximale est de 1.8° . En pratique, avec le filtre, la précision est presque doublée.

$$(9.20) \quad \delta\psi = \text{atan} \frac{1.5}{49 - 1} = 1.8^\circ$$

Vis-à-vis de la figure 9.4, P est le point rouge, V est l'orientation rouge, M est la longueur du vecteur rouge et C est, l'intersection du vecteur rouge et du plan représentant le mur. De façon similaire, l'angle ψ peut être déterminé selon l'axe Y .

9.5.2 Détermination de la position du drone

Maintenant que l'angle ψ , la rotation autour de l'axe Z a été déterminée, la position du drone peut être également déterminée. La méthode est en tout point similaire à la détermination de l'angle ψ . En effet, la seule différence est qu'il faut commencer par appliquer aux vecteurs P et V la rotation complète, comprenant le roll, pitch et yaw (ψ). Cela se fait grâce la librairie de transformations TF vue à la section 7.3, via des quaternions, afin d'éviter tout problème de convention.⁸

Les équations de départ sont les mêmes équations pour un laser dont la position est P et le vecteur orientation est V une fois la rotation effectuée selon roll, pitch et yaw.

$$(9.21) \quad V = \begin{bmatrix} Vx \\ Vy \\ Vz \end{bmatrix}$$

⁸. Pour plus d'informations sur les rotations dans l'espace, le site de référence Euclidian Space [13] est disponible.

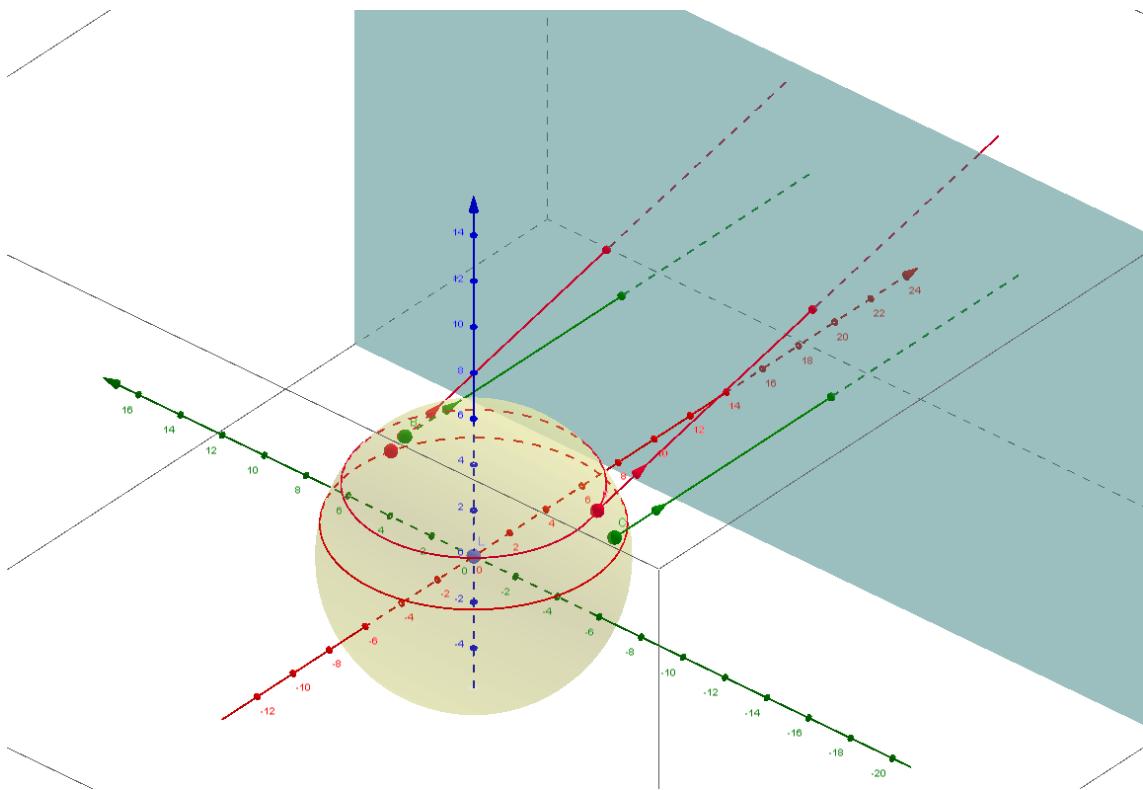


FIGURE 9.4 – Détermination du yaw en 3D, réalisé avec Geogebra [14]

$$(9.22) \quad P = \begin{bmatrix} Px \\ Py \\ Pz \end{bmatrix}$$

L'extrapolation de longueur M , la mesure des lasers, peut à nouveau être réalisée.

$$(9.23) \quad K = \sqrt{\frac{M^2}{Vx^2 + Vy^2 + Vz^2}}$$

Tout comme précédemment, la cible est déterminée, mais pour ce cas-ci, C n'est plus dépendant de psi.

$$(9.24) \quad C = K * V - P = \begin{bmatrix} K * Vx - Px \\ K * Vy - Py \\ K * Vz - Pz \end{bmatrix}$$

De la même façon, $C \in mur$, et la normale et un point du mur sont connus.

$$(9.25) \quad mur \equiv ax + by + cz + d = 0$$

Avec une normale valant $(1, 0, 0)$

$$(9.26) \quad mur \equiv a * x = 0$$

Sachant que le point $C(\psi) \in mur$

$$(9.27) \quad mur \equiv x = K * Vx - Px$$

Et ainsi de suite pour les différents murs.

$$(9.28) \quad mur_x \equiv x = K * Vx - Px$$

$$(9.29) \quad mur_y \equiv y = K * Vy - Py$$

$$(9.30) \quad mur_z \equiv z = K * Vz - Pz$$

Enfin, il faut reprendre le repère original, centré au centre des 3 murs de référence, afin de trouver la position finale D .

$$(9.31) \quad D = \begin{bmatrix} K * Vx - Px \\ K * Vy - Py \\ K * Vz - Pz \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Le résultat de cette équation permet d'obtenir les coordonnées en X, Y et Z du drone.

9.6 Prédiction de l'état réel

À ce stade, les 6 distances aux murs de références et deux mesures de l'angle yaw⁹ sont connues. Il est possible maintenant de faire la fusion de toutes les données afin de déterminer la position finale du drone.

Pour ce faire, le filtre qui sera utilisé est l'alliance entre le filtre de Kalman grandement simplifié¹⁰ et une moyenne glissante. En effet, comme expliqué au point 9.4.2 un drone peut fonctionner dans deux modes différents qui nécessitent des réglages différents. Ces modes sont le vol **stationnaire** à faible dynamique et le vol **mobile** à grande dynamique. Afin de passer d'un mode à l'autre, il suffit de changer les gains de sorties du filtre représenté à la figure 9.5.

La première branche est la branche de Kalman. Il s'agit d'une prédiction (eq : 9.32) provenant du feedback de la position précédente et de la vitesse du drone prélevée du contrôleur de vol. Par ailleurs, la vitesse du contrôleur de vol est déjà prise en compte ainsi que la position qui est envoyée par l'intégration de son accélération. En réalisant la simplification, de ne pas devoir prendre en compte l'accélération ainsi que la vitesse, cela rend la prédiction plus aisée. Le gain de la branche est appelé **Gain de Kalman**.

$$(9.32) \quad P = X_{n-1} + v_n * \delta t$$

La seconde branche est la branche de la moyenne glissante. Les 10 dernières valeurs sont stockées, soit les 100 dernières millisecondes. La plus grande et la plus petite valeur sont ensuite supprimées. Le gain de cette branche est défini comme étant le **Gain de moyenne**. Enfin, la moyenne des N valeurs restantes est calculée avant d'être additionnés aux autres branches. Avec $k = 0$ pour la valeur minimale et maximale et $k = 1$ pour les autres, l'équation est :

$$(9.33) \quad M = \frac{\sum_{n=-9}^0 x_n * k}{8}$$

La dernière branche est simplement la sortie de l'algorithme de positionnement. Le gain s'appelle le **Gain direct**. Puisque deux mesures sont disponibles, la nouvelle valeur est la moyenne des deux. En effet, le taux de confiance est identique pour les deux. Il est tout de même possible de nuancer la fusion des deux lasers si cela est nécessaire. En effet, ces lasers grands publics ont des précisions différentes. Ainsi, si des gains différents doivent être appliqués, il suffit de mettre en avant la confiance, statistiquement, envers ces lasers.

9. En pratique, une seule mesure de l'angle yaw est calculé car le poids sur le drone était limité.

10. L'idée de la prédiction et du gain de Kalman a été conservé, mais en simplifiant grandement la mise à jour du gain de Kalman.

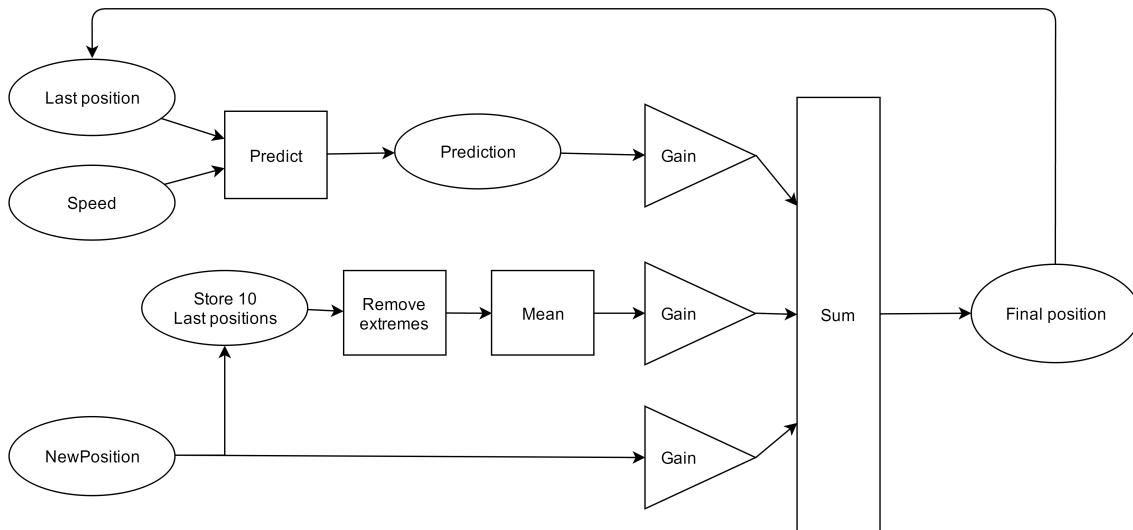


FIGURE 9.5 – Dataflow du filtre de prédiction

Soit $M_A(x)$ la mesure du laser A, G_A son gain et $M_B(x)$ la mesure du laser B, G_B son gain, selon x , la distance.

$$(9.34) \quad M_A(x) = x + \delta A$$

$$(9.35) \quad M_B(x) = x + \delta B$$

Avec δA et δB les incertitudes sur A et B

$$(9.36) \quad G_A = \frac{\delta B^2}{\delta A^2 + \delta B^2}$$

$$(9.37) \quad G_B = \frac{\delta A^2}{\delta A^2 + \delta B^2} = 1 - G_A$$

Afin d'utiliser le système, il suffit de changer les 3 gains (Kalman, moyenne et direct) pour qu'ils correspondent au mieux aux besoins du système, tout en s'assurant que la somme soit égale à 1 pour rester dans un cas de filtre stable. La méthode utilisée et conseillée est inspirée du filtre de Kalman, aux équations 9.37.

1. **Vol stationnaire** : Si le but du drone est d'avoir une dynamique faible ou nulle, afin de se **positionner** précisément, alors il faut définir un gain de Kalman de 0.2, gain de moyenne de 0.7 et gain direct de 0.1.
2. **Vol mobile** : En mouvement, quand le but est de **localiser** le drone de manière fiable et sans délai, il faut préférer le gain de Kalman de 0.7, gain de moyenne de 0 et gain direct de 0.3.

9.7 Résultats du système de positionnement

La plus grosse partie du travail fut la détermination du système complet de localisation et des équations, ce qui a demandé beaucoup de réflexion et de design.

Qualités

- Fonctionne en intérieur
- Référence relative à la pièce
- Rapide à configurer
- Indépendant des perturbations électromagnétiques, de pressions, de luminosité
- Indépendant du revêtement des murs
- Indépendant de l'état de surface des murs

Défauts

- Ne fonctionne pas en extérieur¹¹
- Nécessite des murs de référence
- Doit être calibré et configuré
- Très dépendant du yaw pour le bon fonctionnement

Précision statique

- Écart-type original des lasers $\sigma \approx 1.50\text{cm}$
- Écart-type yaw $\sigma \approx 0.45^\circ$
- Écart-type position $\sigma \approx 0.82\text{cm}$

Précision en vol

- Écart-type original des lasers $\sigma \approx 1.50\text{cm}$
- Écart-type yaw $\sigma \approx 0.65^\circ$
- Écart-type position $\sigma \approx 2.9\text{cm}$

Il est important de noter que ces résultats concernent la précision de la **localisation**. C'est en effet la précision maximale que le drone pourra atteindre pour se déplacer dans la pièce. Toutes les informations sur les performances du système de localisation sont dans la partie Tests et performances au chapitre 14.

L'algorithme de positionnement a été testée avec GeoGebra. Les équations ont été implémentées et lorsque les angles Yaw, Pitch et Roll étaient modifiés les valeurs de position restaient inchangées. Il a donc été prouvé théoriquement que les équations sont bien fonctionnelles et fournissent des valeurs de positions correctes.

11. Car la présence de murs de références est un prérequis.

CHAPITRE 10

Contrôle

Abstract

Ce chapitre a pour but d'expliquer la philosophie derrière le système de contrôle implanté. Il montre comment mettre en œuvre différents nœuds afin d'obtenir une architecture robuste et modulable pour s'interfacer avec d'autres travaux réalisés en parallèle tels que la pince et le *Path Planning*.

La partie contrôle du drone a pour fonction de déterminer la position du drone et contrôler le drone afin qu'il atteigne une position particulière. L'architecture choisie se sépare en 5 nœuds principaux.

1. Module d'acquisition
2. Algorithme de positionnement
3. Télémétrie
4. Commande
5. Tâches

10.1 Les différents Nœuds

Les différents nœuds réalisés seront expliqués dans cette partie, ainsi que leur utilité et les liens entre les modules.

Nœud d'acquisition Ce nœud se charge de récupérer les mesures des différents lasers installés sur le drone. Pour rappel, le module de mesure est constitué de 6 lasers connectés à une Arduino qui communique avec l'ordinateur embarqué via le port série. Le nœud d'acquisition doit donc interagir avec le système de mesure et transmettre les données de manière synchrone dans un topic (`/lasers/raw`). En pratique, ce nœud est générique et c'est l'Arduino, lors de l'initialisation de la communication série qui détermine le topic sur lequel publier.

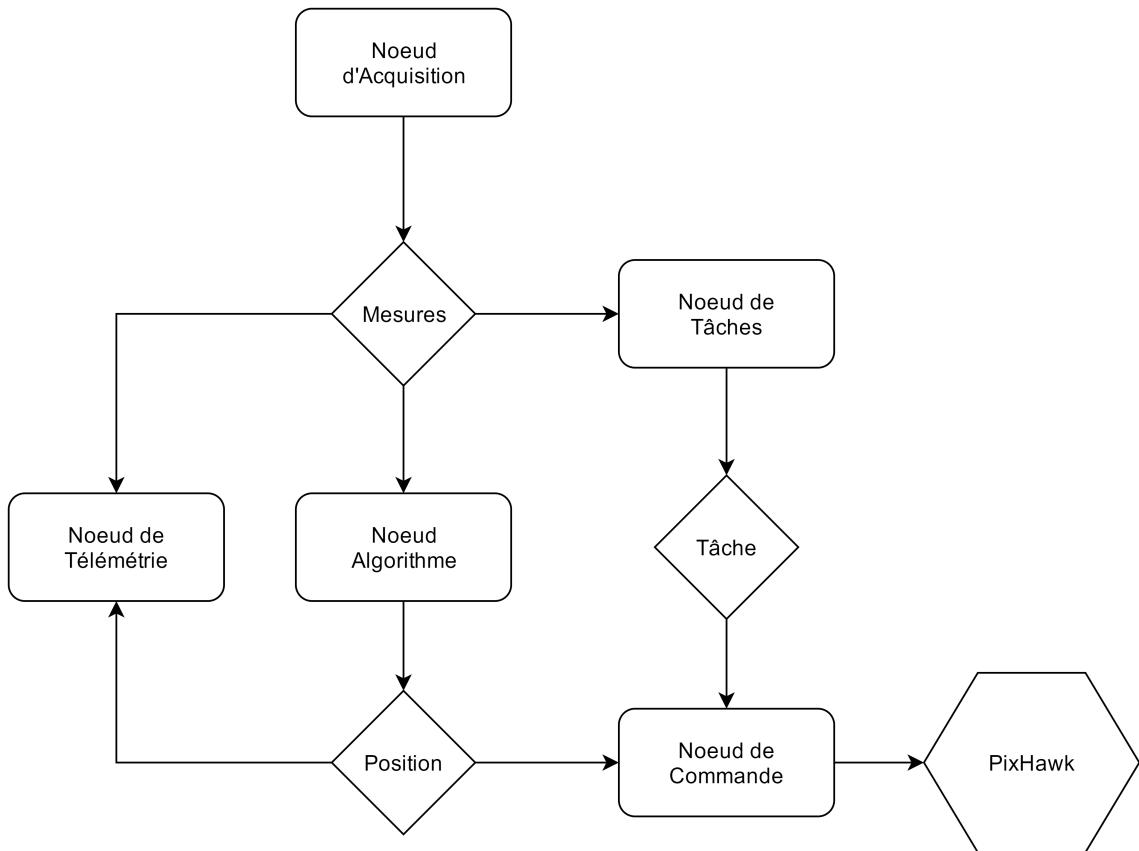


FIGURE 10.1 – Schéma de principe de l'architecture de contrôle du drone

Noeud d'algorithmique Ce noeud implémente les différents filtrages ainsi que l'algorithme de positionnement du chapitre 9. Il lit les informations du topic de mesures (/lasers/raw), calcule la position du drone dans l'espace et publie le résultat sur le topic de positions. (/lasers/filtered).

Noeud de commande Ce noeud s'occupe de commander le drone. Cela signifie qu'il reçoit les positions du noeud d'algorithmique, reçoit les informations du noeud de tâches et contrôle le drone en lui envoyant les bonnes données au bon moment.

Noeud de tâches Ce noeud a plusieurs tâches pré-encodées qui, mises ensemble, permettent de réaliser des scénarios et missions. Ces données seront du type :

1. Décolle
2. Va à la position (X, Y, Z)
3. Prends la brique en (X, Y, Z)
4. Place la brique en (X, Y, Z)
5. Rentre à ton point de départ

Nœud de Télémétrie Ce nœud se charge de récupérer toutes les informations du drone afin de pouvoir les analyser ou les visualiser. Par exemple : l'état angulaire, l'état de la batterie, sa position, les mesures des lasers... Ces données sont ensuite publiées sur un topic que l'interface pourra écouter, et ces données sont aussi stockées afin d'améliorer les filtres et algorithmes avec des données réelles.

10.2 Modularité

ROS est souvent vu, à raison, comme étant un outil modulable. En effet, comme cela est présenté dans la figure 10.1, chaque module est bel et bien indépendant. Le lien se fait via les données transmises sur les différents topics. Or, la modularité est une caractéristique principale dans un travail, car elle permet le développement de groupe.

En effet, afin de réaliser un module ou remplacer un module, il suffit de se souscrire et de publier aux bons topics. Cette simplicité de communication qu'apporte ROS permet de changer les modules sans devoir modifier le moindre code. Par exemple, la première interface réalisée fût en ligne de commande. Lorsque l'interface en ligne de commande a été changée en faveur d'une interface graphique, rien d'autre n'a dû être adapté. Il a juste fallu connecter l'interface pour qu'elle soit fonctionnelle.

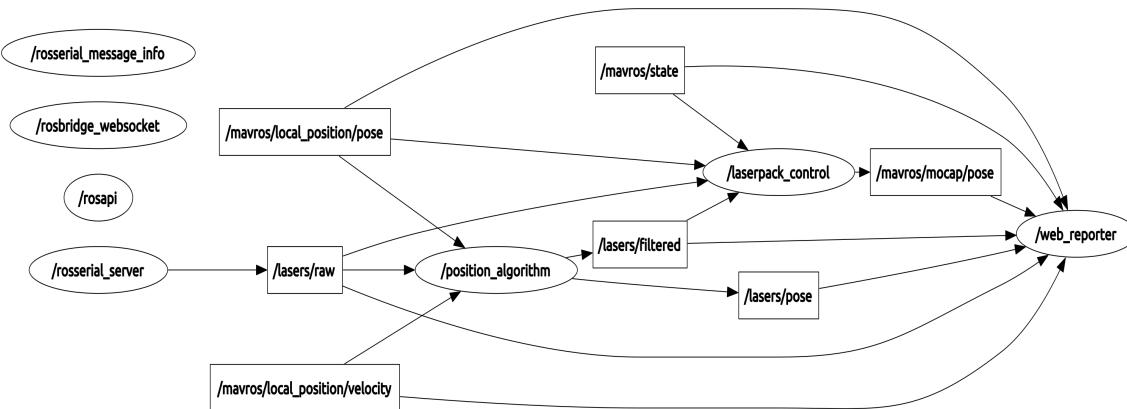


FIGURE 10.2 – Architecture ROS du système complet

10.2.1 Vue d'ensemble

La figure 10.2 permet de mieux voir et cerner les interactions entre les différents modules. Chaque noeud a une tâche spécifique et peut être très facilement remplacé par un éventuel nouveau noeud, qui publierait sous les mêmes topics.

Pour rappel il existe **5 noeuds principaux** :

- **Rosserial_server** : Noeud d'acquisition qui établit la connexion série entre le microcontrôleur et l'ordinateur de bord
- **Position_algorithm** : Noeud d'algorithme qui exécute l'algorithme de positionnement
- **Laserpack_control** : Noeud de commande qui s'occupe de la communication au drone
- **Web_reporter** : Noeud de télématétrie qui se charge de récupérer toutes les mesures du drone
- **Rosbridge_websocket** : Noeud de tâches qui provient de l'interface Web via un noeud de communication en websocket

Les **topics**¹ principaux sont :

- /lasers/raw : Mesures brutes des lasers
- /lasers/filtered : Position du drone après filtrage
- /lasers/pose : Position brute du drone
- /mavros/local_position/pose : Position récupérée du drone, intégrée à l'accéléromètre et au gyroscope interne
- /mavros/state : État du drone
- /mavros/mocap/pose : Positions que l'on envoie au drone
- /mavros/local_positon/velocity : Vitesse perçue par le drone

1. Les noms peuvent paraître complexes mais la nomenclature est arbitraire et permet d'avoir une structure pour les topics. Ainsi sur le topic /lasers/raw transitent les mesures brutes des lasers

CHAPITRE 11

Interface

Abstract

Il est important que l'interface du drone puisse être accessible au plus grand nombre. C'est pourquoi l'interface de contrôle a été réécrite sous la forme d'une page web. Celle-ci est fonctionnelle, et donne un exemple concret de comment réaliser la communication avec le drone depuis l'extérieur. Cela permet donc à une personne extérieure de juger de la qualité du travail.

11.1 Problématique

Le drone était contrôlé via une interface en ligne de commande. Mais exécuter l'ensemble des scripts en invite de commande Linux prend du temps et il faut connaître chacun des raccourcis *hardcodés*. Plus le temps avançait et plus l'architecture devenait complexe et le nombre de raccourcis augmentait sans cesse. De plus, en ligne de commande, il n'est pas possible de pouvoir facilement traquer les différentes valeurs. Il est donc difficile de s'y retrouver pour un utilisateur non habitué et c'est donc pour améliorer l'expérience utilisateur que l'interface graphique a été développée.

11.2 Le besoin

L'interface graphique doit pouvoir remplir les conditions suivantes :

- Interface web compatible sur différents supports utilisés
- Interface responsive, compatible avec les écrans de toute taille
- Récupération de toutes les informations qui sont utiles lors d'un vol afin de comprendre ce qu'il se passe
- Afficher des graphiques en fonction du temps du système d'acquisition
- Contrôle du drone par l'envoi de setpoint en X, Y et Z
- Background simple afin d'être comprise par la personne qui voudrait l'améliorer

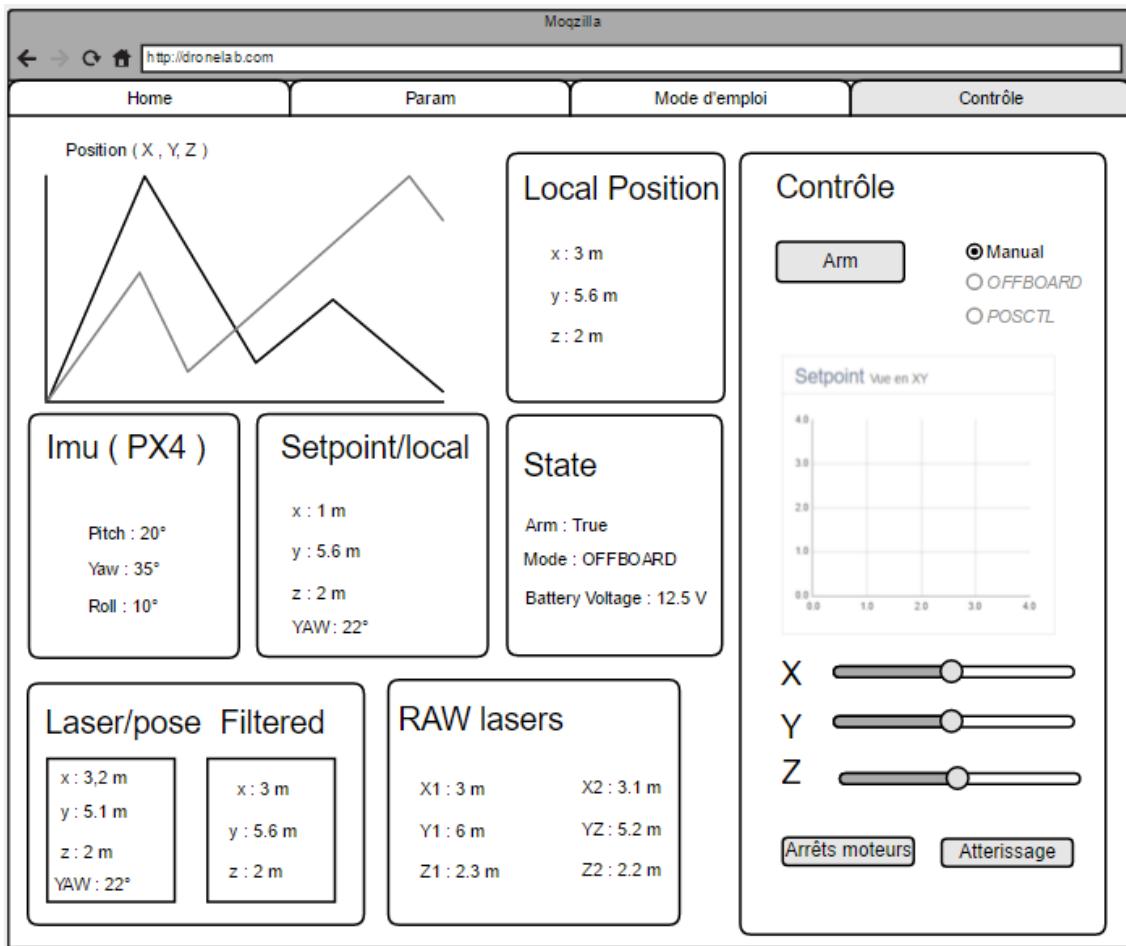


FIGURE 11.1 – Maquette de l'interface graphique

11.3 En pratique

La première maquette réalisée était telle que présentée à la figure 11.1. L'interface graphique est désormais réalisée grâce aux technologies WEB, afin que tout le monde puisse y accéder facilement depuis son ordinateur.

La communication est réalisée via une librairie faisant le pont entre le système ROS et le système extérieur, RosBridge. Cette librairie permet d'avoir un lien vers les machines sans ROS et de pouvoir tout de même communiquer via des JSON¹. Les informations reçues par l'interface de contrôle proviennent du nœud de télématrice.

Au point de vue sécurité, rien n'est encore implémenté, mais il serait judicieux d'implémenter une couche de connexion avec ROSBridge. En effet, cette partie ne fait pas partie du travail de cette année, mais sera primordiale pour la suite.

La version avancée de l'interface web est présentée à la figure 11.2. Sur la page, l'utilisateur peut avoir un retour direct sur toutes les valeurs de télématrice et peut avoir un

1. Javascript Object Notation

CHAPITRE 11. INTERFACE

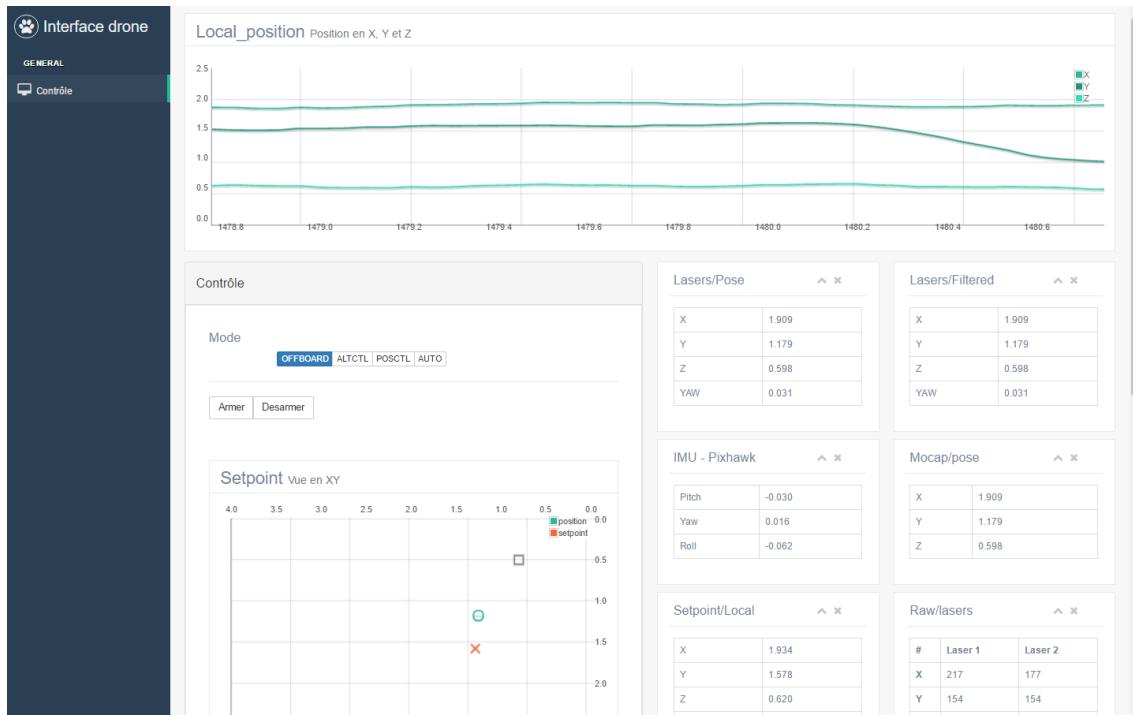


FIGURE 11.2 – Interface graphique

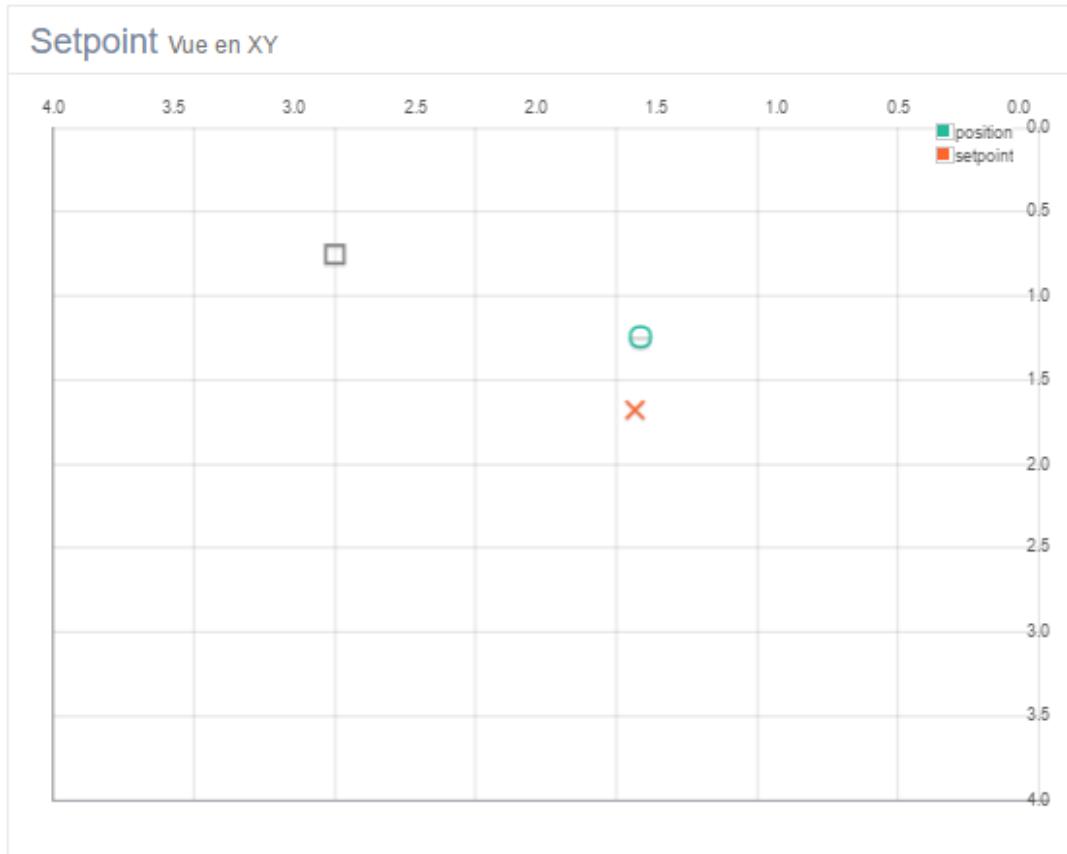


FIGURE 11.3 – Partie contrôle

graphique en temps réel. Le graphique permet de voir en un coup d'œil quand un laser a eu une valeur complètement erronée ou simplement voir la dynamique du drone.

Dans la partie contrôle de la figure 11.3 l'utilisateur peut directement donner des instructions au drone. Le cercle vert correspond à la position vue par le drone, la croix rouge correspond au setpoint et le carré gris est le pointeur de la souris.

Avec cette fenêtre, le comportement du drone et son objectif sont directement visibles. Si la croix est sur le cercle, cela signifie que le drone est arrivé à son objectif (en XY car la hauteur n'est pas reprise en 2D). L'utilisateur peut cliquer n'importe où sur le graphique pour changer l'emplacement de la croix qui correspond au setpoint. Ainsi grâce à cette partie il peut directement contrôler le drone dans l'espace et voir comment le drone réagit aux consignes.

11.4 Conclusion

La partie interface n'a pas été une grosse partie du développement et c'est pourquoi elle a besoin encore de certaines améliorations et de l'ajout de certaines fonctionnalités. Cette interface n'était pas dans le cahier des charges, mais elle a été faite pour faciliter les tests et rendre les démonstrations plus conviviales.

Cette interface est donc une base de travail pour les futurs développeurs. Ainsi s'il a besoin d'ajouter des graphes, il lui suffira de créer l'espace pour le nouveau graphique en sachant que la communication et la récupération d'informations sont déjà fonctionnelles.

Quatrième partie

Tests et performances

CHAPITRE 12

Vue d'ensemble des tests de performance

Une série de tests ont été réalisés pour tester le système actuel et définir les caractéristiques statistiques des différents modules. Ces tests permettent de quantifier les performances des lasers, de la localisation et du positionnement du drone présent à la figure 12.1. Certains tests statiques ont été réalisés pour les lasers et la localisation, et d'autres tests dynamiques ont été réalisés pour le positionnement du drone.

Les tests statiques correspondent aux tests qui ont été réalisés à l'arrêt sans intégrer la dynamique du drone alors que les tests dynamiques ont été faits en contrôlant le drone. Le but est donc de déterminer la précision des lasers, l'intolérance de pose et la précision lors d'un vol. Les premiers tests sont statiques, afin de diminuer les inconnues et permettre de tester chaque module indépendamment. Les tests suivants ont été réalisés en dynamique afin de pouvoir déterminer les performances en vol du drone.

Tests statiques

1. Caractéristiques des lasers
2. Caractéristiques de l'état angulaire du PixHawk
3. Précision linéaire de l'algorithme de positionnement
4. Précision angulaire de l'algorithme de positionnement

Tests dynamiques

1. Précision du positionnement
2. Résilience aux perturbations
3. Réactivité de vol avec un scénario

Pour chacun des tests, un module d'acquisition stocke 100 fois par seconde l'entièreté des informations utiles du drone, grâce au nœud de télémétrie.

Les **mesures** principales sont :

- Local_position : Position perçue par le drone
- Raw : Mesure brute des lasers, sans traitement
- Setpoint : La consigne (x, y, z) envoyée au drone

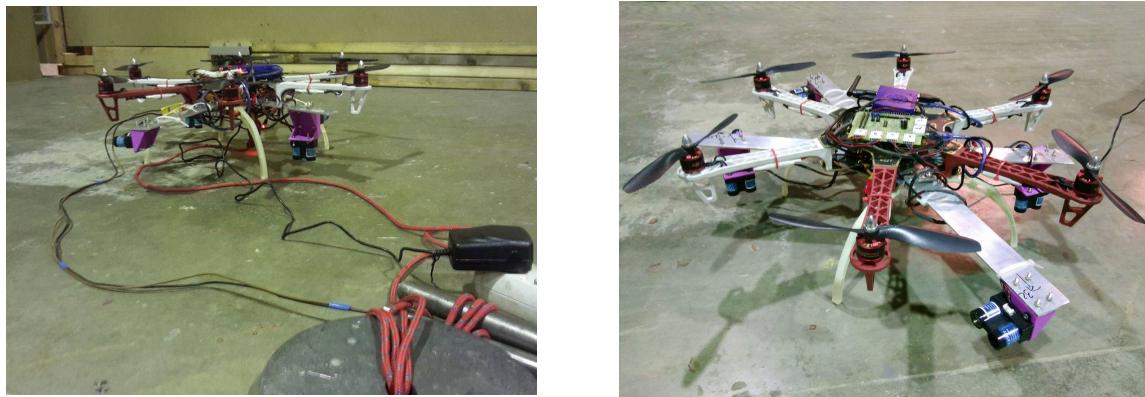


FIGURE 12.1 – Drone utilisé lors des tests

13

CHAPITRE

Caractéristiques des lasers

Abstract

Étant classé dans la catégorie tout public, il était nécessaire de vérifier les paramètres métrologiques de chacun des lasers, c'est-à-dire la fidélité, la justesse et l'exactitude des lasers, ainsi que la répétabilité¹ dans le temps. Ce chapitre tend à prouver que les lasers ont une très bonne répétabilité et sont fidèles, mais nécessitent une correction afin de les rendre juste.

13.1 Test de fiabilité

Les lasers (figure 13.1) étaient au départ inconnus et n'avaient pour caractéristique que celle donnée par le fabricant, une précision de $+/- 2.5 \text{ cm}$.

Pour tester les lasers, 20 mesures pour chacun des lasers ont été prises sur une course allant de 1m à 8m. En pratique cette plage correspond à la zone de travail des lasers puisque le drone ne devrait pas survoler des zones trop proches du mur de référence.

Pour les tests, les mesures des lasers doivent pouvoir être comparées avec une mesure de référence. Notre pseudo étalon dans notre cas correspond à un télémètre laser précis de l'ordre du mm. Pour ne pas introduire d'erreurs supplémentaires, un banc de test présent sur l'image 13.3 a été construit et placé sur un rail (image 13.2) pour avoir une avance guidée. Le banc de test permet de fixer les 4 lasers ainsi que le télémètre et de faire l'acquisition des différentes mesures qui sont stockées dans un fichier Excel.

1. La répétabilité mesure la capacité d'un système à reproduire une tâche.



FIGURE 13.1 – Prototype utilisé pour tester les lasers sans le rail



FIGURE 13.2 – Rail utilisé lors de nos tests

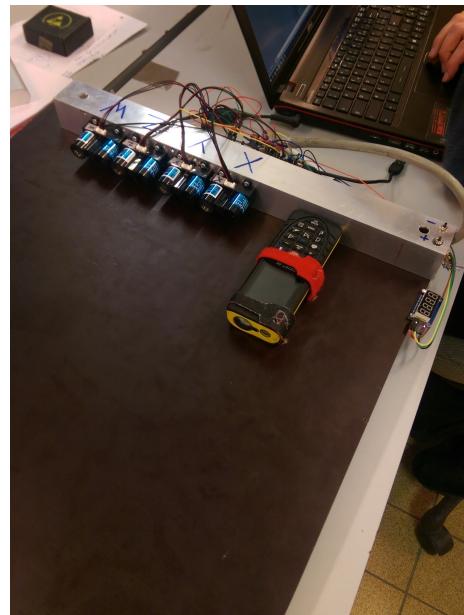


FIGURE 13.3 – Banc de test

Réelle	Essai 1	Essai 2	Essai 3	Maximum	Minimum	Moyenne	Répétabilité
2000	2069	2068	2046	2069	2046	2061	1.12
2500	2533	2537	2542	2542	2533	2537	0.35
3000	3048	3048	3048	3048	3048	3048	0
3500	3546	3557	3549	3557	3546	3551	0.31
4000	4033	4037	4036	4037	4033	4035	0.1
4500	4555	4559	4556	4559	4555	4557	0.09
5000	5034	5047	5039	5047	5034	5040	0.26
5500	5541	5555	5545	5555	5541	5547	0.25
6000	6068	6068	6072	6072	6068	6069	0.07
6500	6560	6560	6559	6560	6559	6560	0.02
7000	7066	7063	7072	7072	7063	7067	0.13
7500	7747	7564	7559	7747	7559	7623	0.09
8000	8057	8060	8047	8060	8047	8055	0.16

TABLE 13.1 – Répétabilité d'un des lasers, toutes les mesures sont en millimètres

13.2 Essais de répétabilité

Trois essais ont successivement été réalisés afin de déterminer la répétabilité des lasers. De plus, chaque mesure affichée correspond à une moyenne sur dix mesures consécutives. Les mesures pour un des lasers se retrouvent dans le tableau 13.1 :

- **Mesure réelle** : mesure de référence prise avec le télémètre laser
- **Essai** : moyenne de 10 valeurs prise avec un laser. Ces mesures ont été répétées 3 fois dans des tests différents.
- **Répétabilité** = $((Valeur_{MAX} - Valeur_{MIN}) / Moyenne) * 100$

Cet essai qui permet de calculer la répétabilité de la mesure détermine que les lasers sont bien réguliers. La mesure prise par le laser sur des instants différents donnera toujours une mesure correcte et plus ou moins semblable. Cette observation permet de fiabiliser les lasers dans le temps et assure que les mesures soient régulières. Cela montre aussi que les lasers ne sont pas parfaits, car une bonne répétabilité est représentée par un facteur de répétabilité compris entre 0 et 0.1.

13.3 Test d'offset et de linéarité

Les lasers sont réguliers, mais présentent un offset constant qu'il faut corriger pour chacun des lasers. Cet offset est propre au laser et s'ajoute à la mesure. Pour exemple, le laser X est toujours décalé de 19 cm par rapport à la vraie mesure. Il est dû aux caractéristiques physiques des lasers. Sur des lasers professionnels, cet offset est corrigé d'usine.

Pour cet essai, 60 mesures ont été effectuées entre 1 et 8 mètres pour chacun des lasers. Sur ces 60 mesures, seulement 95% ont été gardée afin d'éliminer les valeurs trop entachées d'erreurs.

Avec les 95% de valeurs restantes, 3 droites furent tracées afin de déterminer les indices de corrélation. Ces indices sont :

	X	Y	Z
R	9.999	0.998	0.999

TABLE 13.2 – Linéarité des mesures lasers pour 3 lasers

Cet indice donne une information sur la corrélation entre le nuage de point et la droite. Ainsi, avoir un indice de corrélation de 1 indique que la droite mesurée coïncide parfaitement avec la droite calculée. Ces tests permettent de conclure que les lasers peuvent être considérés comme linéaires. Ainsi pour déterminer l'offset il suffira de prendre deux points et de les comparer avec le télémètre laser.

13.4 Conclusion

Suite au test de fiabilité, il fut nécessaire de corriger deux caractéristiques, l'offset et l'architecture d'acquisition.

Lors des tests, il est arrivé qu'un laser donne des valeurs complètement fausses. Il faut donc pouvoir identifier l'erreur et empêcher qu'elle se reproduise. C'est l'une des raisons pour laquelle la librairie d'acquisition a été développée. De plus, afin de ne pas rester bloqué sur un laser défectueux, et pour augmenter la fréquence d'acquisition, la librairie se devait de réaliser une acquisition asynchrone. Finalement, l'acquisition asynchrone se fait à 100Hz pour 6 lasers et si un laser est défectueux, il sera réinitialisé automatiquement.

Les résultats montrent que les lasers sont fidèles, et linéaires selon la distance, mais il faut noter la présence d'un offset différent sur chacun des lasers allant de 2 à 20 centimètres. Cette erreur systématique peut simplement être corrigée en prenant quelques points qui seront comparés avec une mesure de référence. Ensuite l'offset observé pourra être pris en compte dans la partie software.

CHAPITRE 14

Performances de l'algorithme de positionnement

Abstract

Il est important de pouvoir quantifier la qualité de la localisation du drone. En effet, la complexité du duo brique & pince dépend principalement de l'imprécision de pose du système. Cette partie sert donc à mettre en avant les performances du système de localisation avec des lasers mais aussi à en analyser les points faibles.

14.1 Caractéristiques de l'état angulaire du PixHawk

Les valeurs d'angle de Pitch, Yaw et Roll n'agissent pas toutes de la même manière sur le système. Il faut donc déterminer la fiabilité de chacune de ces valeurs dans le temps afin de savoir lesquelles garder et lesquelles mesurer.

Les questions :

1. Ces mesures sont-elles fiables ?
2. Ces mesures sont-elles constantes dans le temps ?

La méthode a été de réaliser deux tests où les 3 valeurs d'angles à chaque instant était stockés pendant 10 minutes. Le premier sans bouger ni perturber la centrale inertie et le second en venant perturber la centrale inertie avec un outil métallisé.

14.1.1 Test angulaire statique sans perturbation

Durant ce test, 6000 mesures ont été réalisées, réparties sur 10 minutes, dont le résultat est visible à la figure 14.1.

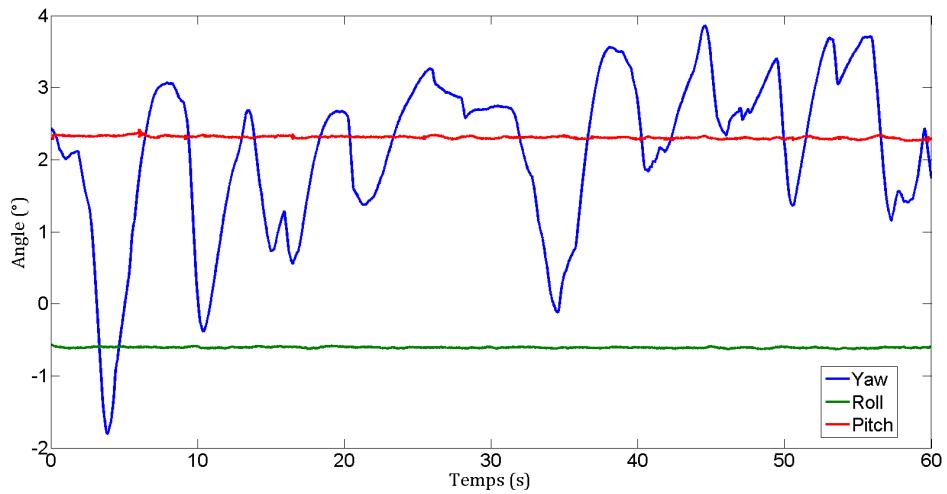


FIGURE 14.1 – Évolution des angles Pitch, Yaw et Roll pour une position fixe, obtenus du PixHawk

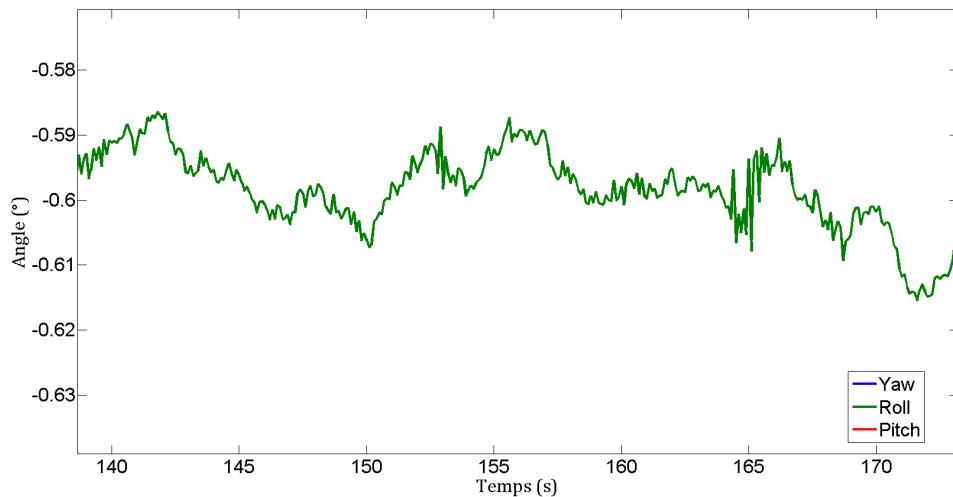


FIGURE 14.2 – Zoom sur l'angle Roll

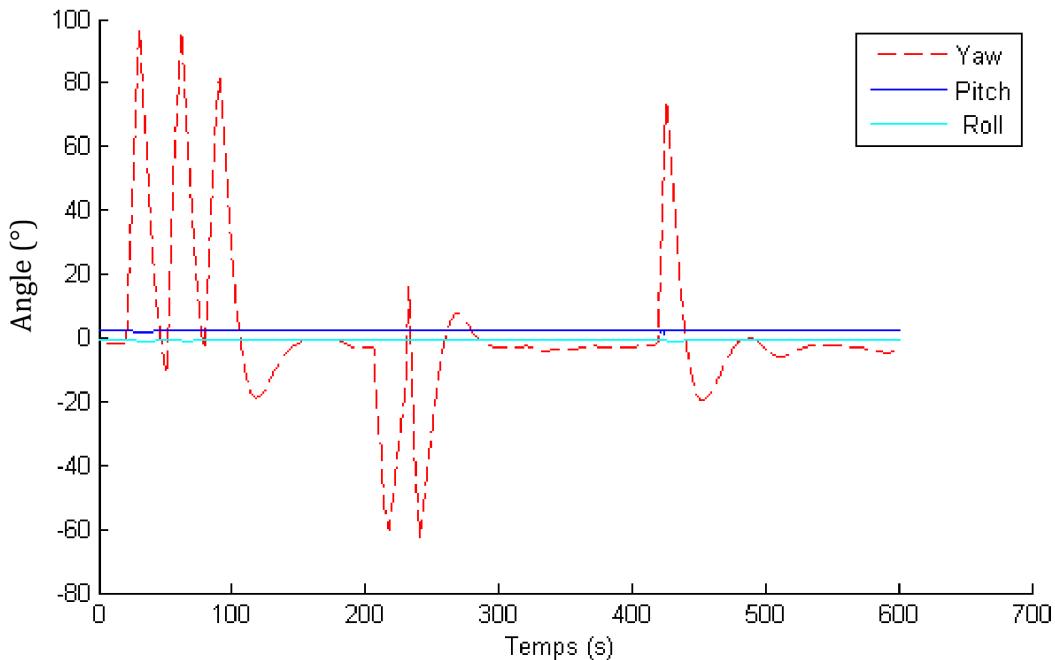


FIGURE 14.3 – Évolution des angles Pitch, Yaw et Roll provenant du PixHawk, avec des perturbations par un matériau métallique

Les 3 angles ont été orientés à un angle proche de 0 afin de mieux constater les oscillations et pouvoir tracer les angles côté à côté. La figure 14.1 met en avant la stabilité des angles Pitch et Roll qui ne varient que de l'ordre du $\frac{1}{10}$ de degré et les oscillations de l'angle Yaw de 5°crête à crête.

En sachant que le Yaw est mesuré notamment grâce à un magnétomètre, et que le système est en intérieur¹ dans un environnement grillagé, il était de rigueur de réaliser un test de perturbation du PixHawk de manière ponctuelle.

Le test consiste à rapprocher un objet métallique proche du PixHawk, à plusieurs reprises, et avec un angle d'attaque différent, sans toutefois le toucher.

Les perturbations sont :

1. 3 perturbations avant 100 secondes
2. 2 perturbations effectuées en amenant l'objet métallique avec un angle d'attaque opposé après 200 secondes
3. Une perturbation après 400 secondes

La figure 14.3 montre bien l'instabilité de l'angle Yaw. Avec des oscillations de 5°sans perturbations, jusqu'à 100°en rapprochant un objet métallique. Il est donc plus intéressant de prendre la mesure de l'angle Yaw via les lasers montés sur le drone.

1. Les champs magnétiques en sont atténus

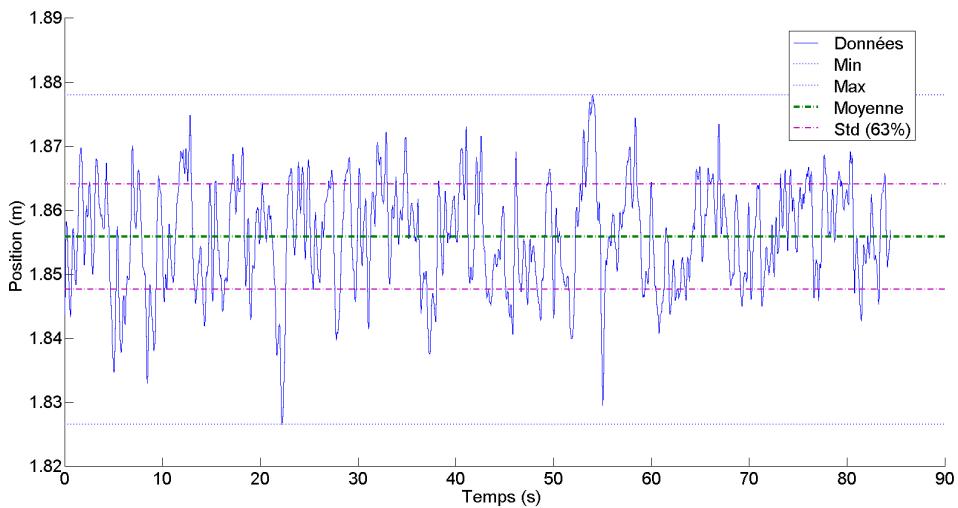


FIGURE 14.4 – Position en x en fonction de temps pour un essai statique

14.2 Précision linéaire de l'algorithme de positionnement

Le drone a été placé au sol pendant 90 secondes sans mouvements tout en prenant 100 mesures par seconde et avec une distance réelle mesurée précise² de :

- $x = 1.862m$
- $y = 1.822m$

Ce test permet de déterminer la précision de l'algorithme de positionnement sans les perturbations dues au contrôle. Soit de déterminer *quantitativement* et *statistiquement* la qualité du système de localisation laser après traitement³ des mesures. Les résultats sont similaires pour chacun des axes, dès lors seuls les détails pour l'axe X seront mis en avant.

Soit l'axe X comme référence, à la figure 14.4, à partir des 9000 points résultants de l'acquisition, les **résultats statistiques** sont :

- Moyenne : $mean(x) = \mu = 185.6cm$
- Écart type : $\sigma = 0.82cm$
- Imprécision maximale : $|x - \mu| = 2.93cm$
- Erreur systématique de calibration⁴ : $E_{sys} = \mu - x_{real} = 0.6cm$

L'histogramme de la figure 14.5 montre que la position en x suit une distribution gaussienne. En pointillé vert cela correspond à la moyenne et en mauve ; l'écart type, soit 68% des valeurs. Avoir un résultat suivant une distribution gaussienne permet de prouver l'utilité de l'application d'un filtre. De plus, le système est quasi parfaitement calibré en X puisque l'erreur systématique de calibration est très faible.

2. Position réelle mesurée grâce à un appareil précis au mm près

3. Le traitement est expliqué au chapitre 9, Localisation

4. L'erreur systématique de calibration est un facteur permettant de quantifier la qualité de la configuration des lasers.

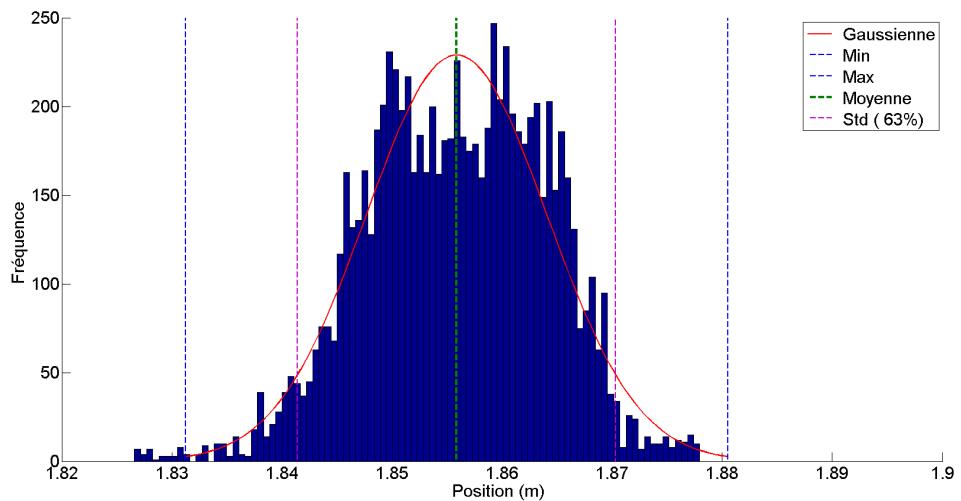


FIGURE 14.5 – Histogramme de la position X pour un essai statique, superposé de la distribution normale

En prenant :

$$P(95\%) = 2 * \sigma = 1.64 \text{ cm}$$

L'algorithme de positionnement a 95% de chance d'avoir un écart inférieur à 1.64 cm rapport à la moyenne. Dès lors, l'imprécision de pose du système de contrôle sera à priori⁵ supérieure.

Cela montre que lorsque le système est bien calibré, l'algorithme de positionnement est fort précis car autrement la moyenne ne serait pas aussi proche de la valeur réelle.

5. Il est possible, mais peu probable que l'inertie du drone ajoute un filtre physique diminuant les perturbations. Ce n'est pas le cas car il est surchargé.

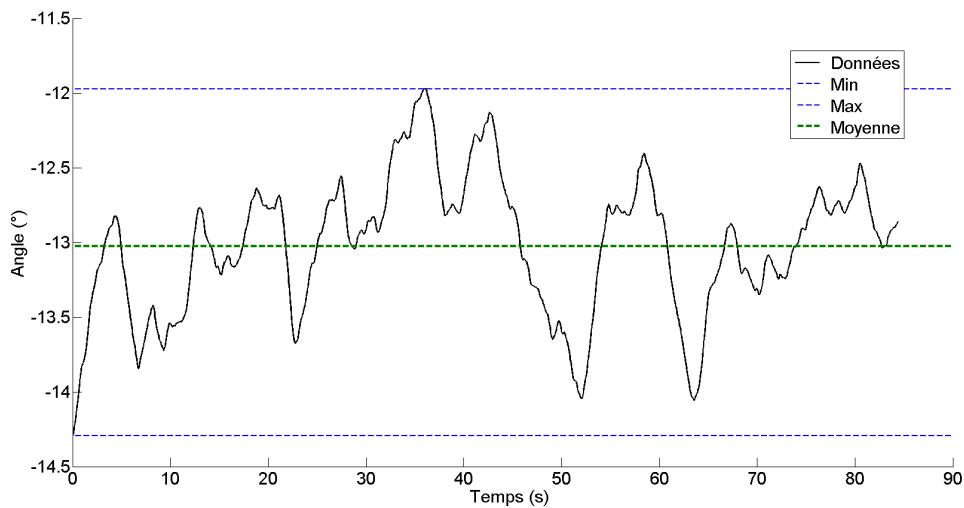


FIGURE 14.6 – Évolution de l'angle yaw mesuré avec les lasers en fonction du temps

14.3 Précision angulaire de l'algorithme de positionnement

L'angle yaw mesuré par le PixHawk étant instable, l'algorithme de positionnement se charge de le mesurer via deux mesures lasers. Le test consiste à prendre 9000 mesures de l'angle yaw en 90 secondes, en laissant le drone au sol. L'idée est de déterminer à quel point la valeur du yaw en sortie de l'algorithme de positionnement est fiable.

La figure 14.6 montre que l'angle yaw oscille jusqu'à 1.3° autour de la moyenne. Les mesures pratiques sont donc meilleures que les calculs théoriques qui prévoient 1.8° à l'équation (9.20) de l'algorithme de positionnement. En effet, la valeur théorique a été réalisée était sans compter le filtrage des lasers.

Précision

- Moyenne : $mean(x) = \mu = 13.0^\circ$
- Écart type : $\sigma = 0.5^\circ$
- Imprécision maximale : $|x - \mu| = 1.27^\circ$

14.4 Conclusion

L'algorithme de positionnement correspond aux performances attendues et espérées. Il permet d'obtenir une précision angulaire supérieure que le magnétomètre et fournit une position suffisamment précise pour le positionnement du drone. Pour rappel, la position du drone doit être précise aux 5 centimètres pour que les Dricks, *les briques drone compatible*, soient efficaces.

Précision

- En XYZ : $\sigma = 0.82\text{cm}$
- En Yaw : $\sigma = 1.5^\circ$
avec le magnétomètre
- En Yaw : $\sigma = 0.5^\circ$
avec les lasers

Fiabilité angulaire

- Les angles Pitch et Roll sont fiables et ne sont que peu perturbés
- L'angle Yaw oscille dans le temps et doit être mesuré avec les lasers

15

CHAPITRE

Tests de contrôle

Abstract

La précision du positionnement du drone est primordiale pour déterminer la complexité des briques autobloquantes. Ce chapitre a donc pour but de montrer quantitativement la précision de pose du système actuel.

15.1 Précision du positionnement

Réaliser un vol stationnaire permet de déterminer avec quelle précision le drone serait capable de prendre un bloc. Durant 120 secondes, 100 mesures par secondes ont été prises, durant lesquelles le nœud de contrôle envoie une position à maintenir. Ce test a été réalisé avec le petit drone de test qui était en surcharge de poids¹. Lors d'un vol, il est à 85% de sa puissance et n'a donc que peu de marge pour une régulation fine, mais le vol était suffisamment stationnaire pour pouvoir déterminer l'imprécision de positionnement en vol en conditions non optimales. À nouveau, les résultats sont semblables pour chacun des axes, et seul le raisonnement pour l'axe X est montré.

Position à maintenir

- $X = 1.934 \text{ m}$
- $Y = 1.577 \text{ m}$

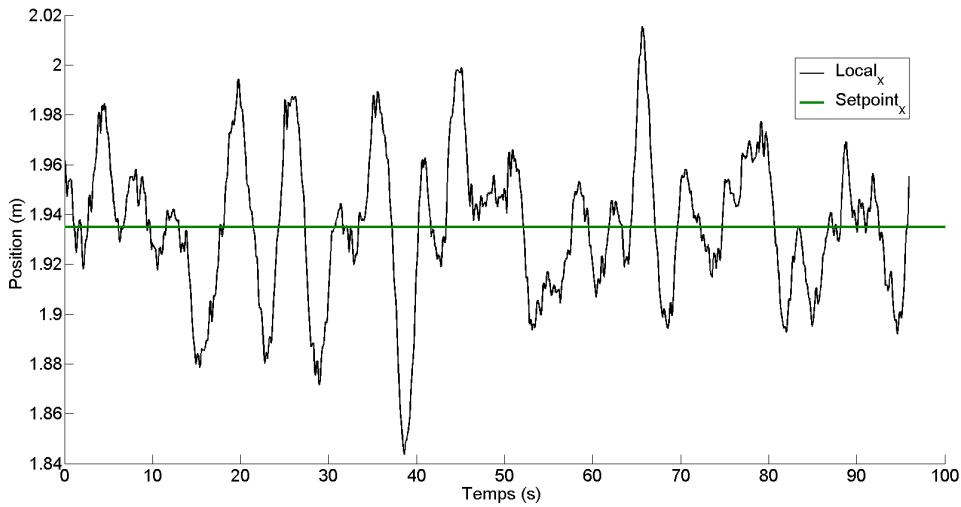


FIGURE 15.1 – Évolution de la position en X en fonction du temps pour un vol stationnaire

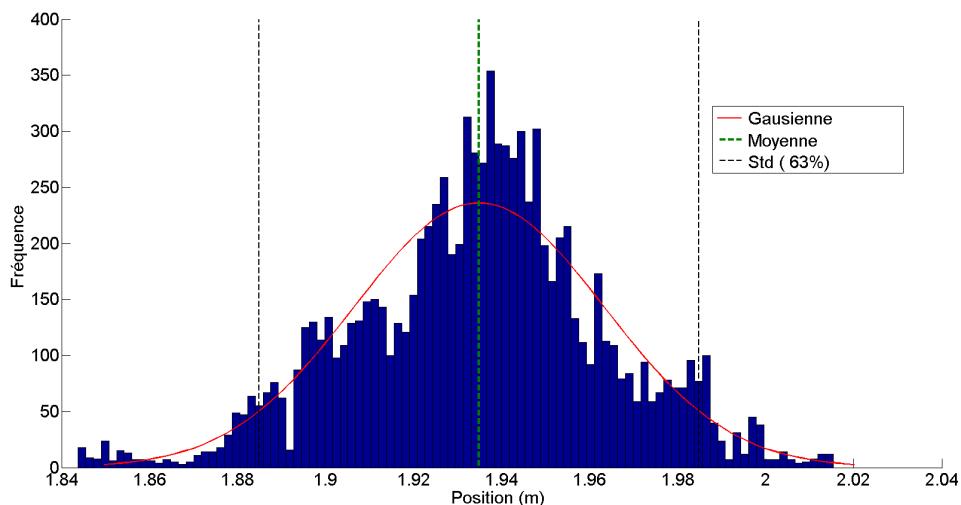


FIGURE 15.2 – Histogramme de la position en X pour un vol stationnaire

15.1.1 Position linéaire

Le test, sur base de 12000 mesures tracées sur la figure, 15.1, montre que

- Moyenne : $mean(x) = \mu = 1.934m$
- Écart type : $\sigma_x = 2.8cm$
- Écart maximum $max(abs(x - setpoint_x)) = 8.1cm$
- L'intolérance de positionnement : $2\sigma_x + |mean(x)-setpoint_x| = 5.8cm$

La première observation se fait sur l'histogramme 15.2, qui a une répartition proche d'une gaussienne. Il est donc pertinent de parler d'écart types.

1. Charge nominale : 1.7kg, charge actuelle : 2.2kg

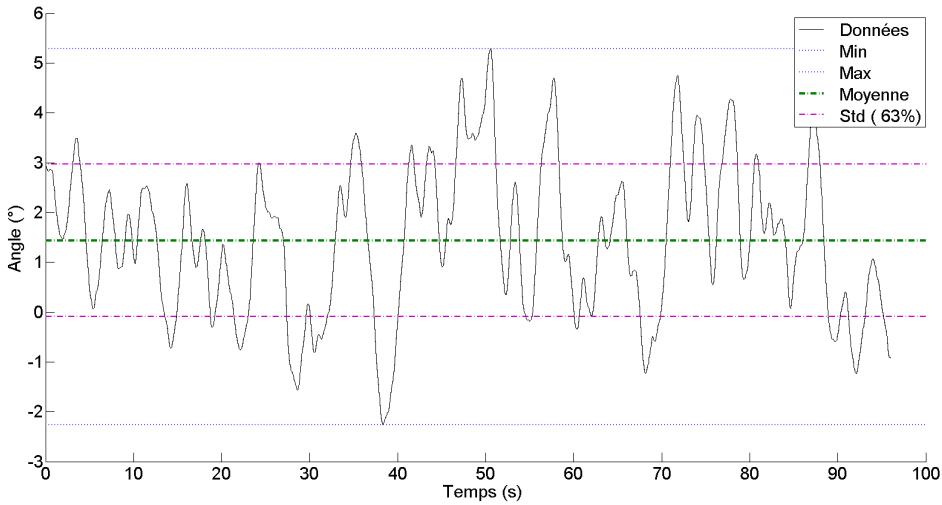


FIGURE 15.3 – Évolution de l'angle YAW au cours du temps pour un vol stationnaire

La seconde observation est la moyenne égale au setpoint pour x^2 . Le PixHawk régule donc bien sa position XYZ en moyenne.

La troisième observation est l'écart type de $\sigma = 2.8\text{cm}$. Cela signifie que 95% des positions ne dépasseront pas les 5.6cm . C'est peu et en même temps beaucoup, car cela signifie aussi que 5% du temps, le drone sera hors de ces 5 centimètres.

15.1.2 Position angulaire

- Moyenne : $mean(yaw) = \mu = 1.5^\circ$
- Écart type : $\sigma_{yaw} = 2.1^\circ$
- Écart maximum $max(abs(yaw - setpoint)) = 5.3^\circ$
- L'intolérance de positionnement : $2\sigma_{yaw} + |mean(yaw)-setpoint_{yaw}| = 5.7^\circ$

La figure 15.3 montre la bonne stabilité angulaire obtenue, mais autour d'une valeur de 1.5° plutôt que de 0° . Cette différence peut être gênante dans le sens où la mesure est tout de même corrigée et l'angle yaw ne varie pas assez que pour arriver aux valeurs critiques.

2. La différence entre la moyenne et le setpoint est de 0.3 cm pour y et d'un centimètre pour z

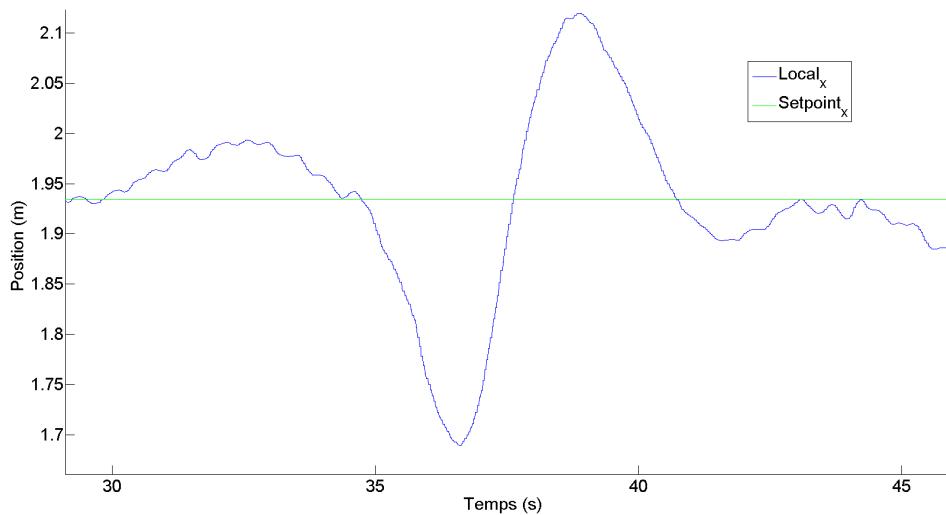


FIGURE 15.4 – Évolution de la position X en fonction du temps, ainsi que la consigne, lors d'un vol perturbé manuellement



FIGURE 15.5 – Perturbation manuelle du drone par un opérateur

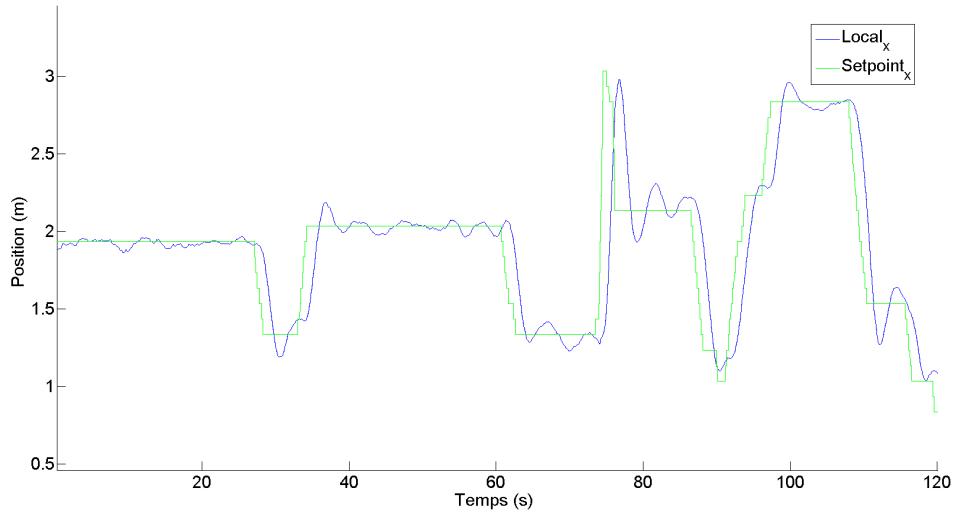


FIGURE 15.6 – Évolution de la position X et du setpoint en X au cours du temps

15.2 Résilience aux perturbations

Comme lors du vol stationnaire décrit plus haut, il a été demandé au drone de maintenir sa position, ensuite un déplacement à la main du drone a été ajouté comme montré à la figure 15.5. Cela a permis de vérifier la robustesse du drone aux perturbations (ex : le vent, etc.).

Dans la figure 15.4 le drone essaye de rester à 1.94m du mur mais après 35 secondes, l'opérateur l'a avancé de 20 cm vers le mur. L'overshoot³ lors d'un vol guidé est beaucoup plus important pour une perturbation. Il est donc primordial d'améliorer les paramètres PID du PixHawk pour qu'il soit plus résilient aux perturbations.

15.3 Réactivité de vol avec un scénario

Le test suivant fut un scénario. Il permet de prendre en compte la dynamique du drone. La figure 15.6 montre que le drone est très réactif et suit la position qui lui est donnée. Le temps de réaction peut être amélioré, mais lors de déplacement d'un point à un autre, il n'est pas nécessaire d'avoir un meilleur temps de réponse. L'overshoot observé est limité, mais peut être fortement réduit grâce au paramétrage des paramètres PID du PixHawk.

3. Dépassemement de la consigne

15.4 Conclusion

Ces différents tests ont permis de quantifier les performances du module de contrôle. Il est important de préciser que le drone est surchargé et déséquilibré. Dès lors, le PixHawk a du mal à réguler la position sans avoir de saturation sur la commande des moteurs. En cas de saturation de la commande, le drone va perdre un peu d'altitude et en essayant de reprendre de l'altitude, perd en précision sur le plan XY. Un problème similaire peut être rencontré avec un drone trop puissant. En effet, au plus un moteur est puissant au moins l'on a un contrôle fin de sa commande⁴.

Synthèse

- Écart type linéaire dynamique : $\sigma_{xyz} = 2.8\text{cm}$
- Écart type angulaire dynamique : $\sigma_{yaw} = 2.1^\circ$
- Précision de pose : 5.70 cm
- Overshoot limité en vol guidé
- Overshoot trop important en cas de perturbation
- Paramètres PID du PixHawk à modifier

4. En effet, quelque soit la puissance d'un moteur, le signal de commande reste le même.

Cinquième partie

Conclusion

CHAPITRE 16

Perspectives

Abstract

Actuellement, il est possible de contrôler un drone de manière autonome avec le système de localisation crée dans le cadre de ce travail. Mais ce n'est pas encore suffisant pour pouvoir réaliser le projet final de construction avec des blocs de 40kg. L'objectif de ce chapitre est de proposer des pistes d'amélioration pour la suite du projet.

Afin de mettre en place un système de construction avec des drones, il est important de prendre en compte les conséquences que pourrait avoir un dysfonctionnement. En effet, dans le cadre du travail de recherche, un drone de 60 cm de diamètre avec des hélices de 20 centimètres de diamètre a été utilisé. Avec ce drone, nous risquons au pire des coupures si nous touchons les hélices. S'il perd un mur de référence, ce n'est pas si grave, nous pouvons éteindre les moteurs. Dans le pire des cas si nous avions eu un accident, nous aurions racheté un moteur, une hélice ou un laser.

Or, un drone constructeur en situation réelle doit être capable de soulever jusqu'à 40 kg de briques, il doit donc faire au moins 2 mètres de diamètre et avoir des hélices de 60 centimètres de diamètre en bois et carbone. Dès lors, les conséquences d'un dysfonctionnement peuvent être létales.

C'est pour ces raisons qu'il faut continuer de développer le contrôle du drone. Voici des perspectives possibles à développer afin de rendre le système réalisable en entreprise.

16.1 Fusion des senseurs

Le problème principal provient du système de localisation. Dans le cahier des charges, il nous a été demandé de réaliser le système de localisation le plus simple, via des lasers fixes. C'est un système viable à des fins de développement, pour par exemple tester les briques autobloquantes, ou la communication de tâches sachant qu'il est interdit de tester le système en extérieur avec un RTK pour des raisons légales.

Mais au final, sur un chantier, n'est-il pas utile de pouvoir sortir du bâtiment ? Construire en se basant sur des murs signifie que l'on devrait construire un mur¹ plus grand que le bâtiment à construire. En pratique, ça serait une grande perte de temps. C'est pourquoi nous estimons qu'il est nécessaire de fusionner différents types de senseurs tout comme ce que l'on réalise avec notre filtre de prédiction en sortie de l'algorithme de positionnement 9.5. Pour cela, la librairie ROS ETHZASL-MSF (TIME DELAY COMPENSATED SINGLE AND MULTI SENSOR FUSION FRAMEWORK BASED ON AN EKF) existe et simplifiera la tâche.

Voici les différents modules que nous conseillons de développer ou d'utiliser

16.1.1 GPS ou RTK

Comme vu précédemment, nous pouvons utiliser le GPS-RTK. Il est possible d'en trouver pour seulement 1k€ avec le projet Piksie [2] par exemple. Le GPS-RTK, étant précis à quelques centimètres s'il est bien calibré, permettrait de réaliser des vols en extérieur autour du chantier, avec une très bonne précision, à la limitation près que l'on doit avoir un lien quasi direct avec la base. En effet, en agriculture de précision, il est commun de perdre sa connexion RTK lorsqu'un arbre est sur le chemin du signal. Mais un GPS ne suffirait-il pas ? Le GPS permet d'avoir une position peu précise, mais de se localiser sur la carte du monde suffisamment bien. Dès lors, l'idée serait d'utiliser le GPS afin d'arriver au chantier ou au point d'utilisation finale et de laisser la suite au système de positionnement local.

16.1.2 Caméra

La méthode Vision Integration vue au point 5.11 a été développée entre autres par ETH Zurich [12], on compte plusieurs techniques telles que SVO [5], ROVIO [4], PTAM [3] ou les travaux de Mr. Julien Hendrickx de l'UCL. Ces systèmes fonctionnent très bien sous certaines conditions. Elle nécessite une carte de développement supplémentaire (une Odroid XU4 ou une Intel NUC suffirait) afin de réaliser le traitement vidéo. En effet, il est nécessaire de déporter le traitement vidéo afin d'éviter de surcharger le cerveau du drone. Il est possible de réaliser une carte de l'environnement avec la caméra ainsi que de faire de l'évitement d'obstacle.

16.1.3 Laser tournant

En remplaçant à la caméra, nous pouvons aussi utiliser un laser tournant. Un laser tournant est plus facile à utiliser qu'une caméra lors du traitement, car nous avons une

1. Par construire un mur, nous entendons : construire ou placer un mur, voire placer des panneaux de type cinéma

mesure de distance et non une simple image. Un laser tournant permet de ne plus être limité à des murs de référence. Et tout comme avec une caméra, il est possible de réaliser une carte de l'environnement grâce à ce système ainsi que de faire de l'évitement d'obstacle.

16.1.4 Apprentissage de l'environnement

Dans tous les cas, il serait utile et presque nécessaire que le drone puisse réaliser une carte de son environnement. Avec sa carte, qu'il peut partager avec d'autres drones sur le même terrain, il peut chercher à prendre le bon chemin directement plutôt que de devoir constamment faire de l'évitement d'obstacle. Cet apprentissage permettrait de séparer la localisation du positionnement. Ainsi le drone pourrait arriver dans sa zone de travail avec son module de localisation et ajuster sa position avec le module de positionnement plus précis.

16.1.5 Système de localisation minimal

À notre sens, le système minimal serait d'avoir un GPS qui donne la localisation globale du drone, un système de caméra ou de laser tournant pour obtenir la position locale du drone et enfin des lasers fixes pour un positionnement précis, tel que montré à la figure 16.1. Par minimal, cela signifie que le drone serait capable d'évoluer dans n'importe quel environnement non hostile et de l'apprendre.

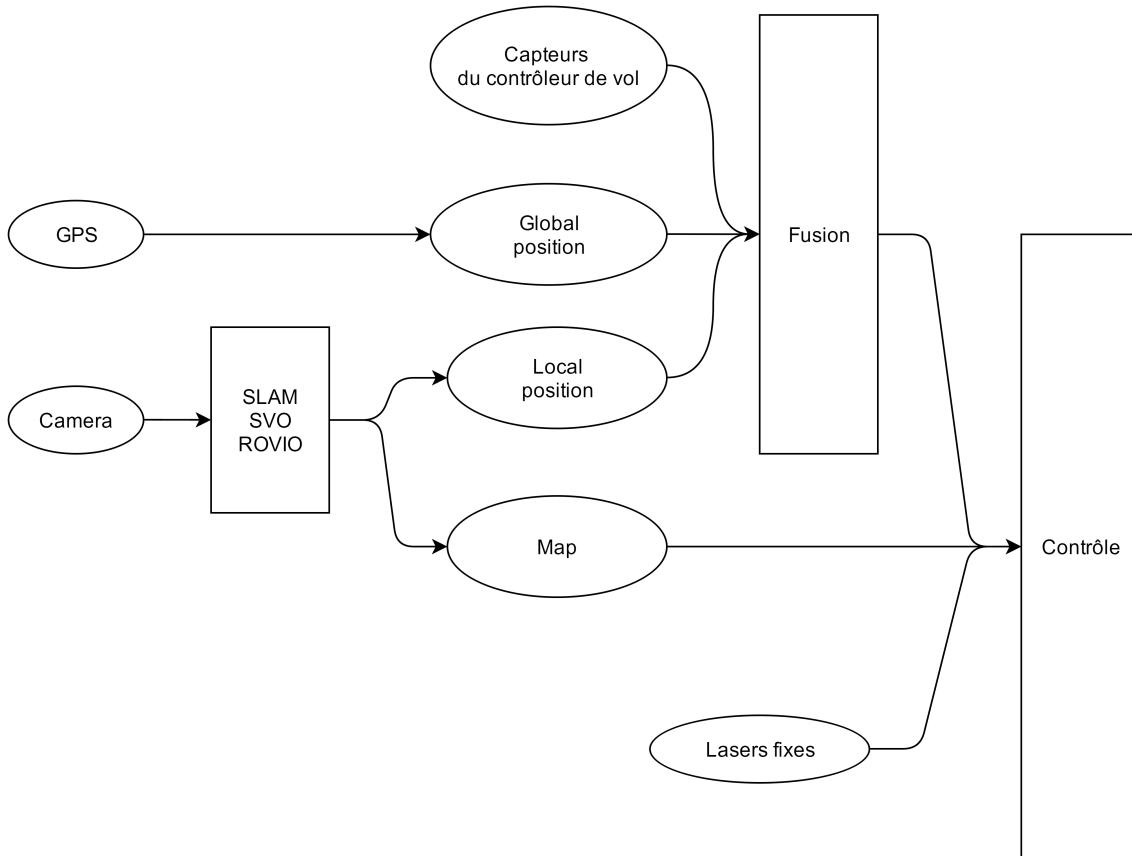


FIGURE 16.1 – Système de localisation minimal

16.2 Amélioration de l'acquisition des lasers

Notre librairie d'acquisition n'est pas parfaite. En effet, certains capacités des lasers sont ignorées car non utilisées. Nous pourrions par exemple réaliser une lecture de distance puis de vitesse successivement et de fournir cette information à l'algorithme de positionnement.

16.3 Moyen drone

Notre drone de test fut relativement suffisant pour nos développements, mais à plusieurs moments, nous furent obligés de diminuer le poids du drone en retirant par exemple les 200 grammes de la corde de sécurité ou les 250g du support des lasers en Y. Actuellement, il n'est plus possible d'ajouter quoi que ce soit sur le drone sans qu'il n'oscille par manque de liberté pour réaliser sa régulation. Il est prévu pour 1.7kg au total et nous en sommes à 2.3kg.

C'est pourquoi pour la suite du projet, nous conseillons d'investir dans un moyen drone pouvant soulever **5kg** afin de ne plus ressentir la limitation de poids.

CHAPITRE 17

Conclusion

À l'issue de cette année, nous rendons trois délivrables. Le premier est un *système de contrôle autonome et modulable d'un drone*. Ce système a pour but de simplifier l'intégration de nouveaux modules au drone, afin d'inclure le système de pince par exemple. Le second est un *module de localisation à lasers fixes*, capable de déterminer la position du drone à deux centimètres près. Le troisième est un *module d'interface de contrôle fonctionnel* montrant comment réaliser un vol, sans avoir à modifier la moindre ligne de code des modules.

Avec l'ensemble des modules implémentés, le drone de test est d'ores et déjà capable de se *positionner* dans une pièce rectangulaire connue avec une erreur maximale de 5 centimètres et de se déplacer dans la pièce de façon autonome, via l'interface de contrôle, ce qui correspond au cahier des charges. Ce résultat permet de commencer des tests grandeur nature des autres projets dès l'année prochaine.

Ce travail a aussi pour but d'être la référence du projet pour implémenter son propre module dans le drone. En effet, nous avons posé les jalons en matière de logiciels afin de rendre le projet le plus modulable possible. Cela réduit considérablement la partie d'étude d'architecture des projets suivants, qui peut être imposante alors qu'elle n'ajoute que peu de valeur au module.

Bien que le produit délivré soit fonctionnel, comme dans tout projet, il est encore possible d'améliorer le résultat. C'est pourquoi nous avons posé le fil conducteur dirigeant les principales améliorations à réaliser dans le chapitre 16 afin d'obtenir un meilleur produit. Ces perspectives concernent principalement la robustesse du système de localisation du drone, afin d'améliorer la sécurité de l'opérateur.

Bibliographie

- [1] Installation de ROS sous ubuntu. <http://wiki.ros.org/indigo/Installation/Ubuntu>.
- [2] Piksie, rtk for only 1k euro. <https://www.swiftnav.com/piksi.html>.
- [3] Ptam : Parallel tracking and mapping. https://github.com/ethz-asl/ethzasl_ptam.
- [4] Rovio : Robust visual inertial odometry. <https://github.com/ethz-asl/rovio>.
- [5] Svo : Semi-direct visual odometry. <https://github.com/uzh-rpg/rpg-svo>.
- [6] Arduino. <http://projects.xoff.org/category/projects/pip-boy/>, September 2011.
- [7] Image rplidar. <http://letsmakerobots.com/node/41122/>, Mai 2014.
- [8] Mavlink for dummies. http://dev.ardupilot.com/wp-content/uploads/sites/6/2015/05/MAVLINK_FOR_DUMMIESPart1_v.1.1.pdf, Juin 2015.
- [9] Bidirectional mosfet voltage filter. <http://fr.hobbytronics.co.uk/mosfet-voltage-level-converter>, Janvier 2016.
- [10] Design fir and iir filters - from tomroelandts.com. <https://fiiir.com/>, février 2016.
- [11] Différents frames. <http://ardupilot.org/copter/docs/connect-escs-and-motors.html>, Avril 2016.
- [12] Eth zurich. <https://pixhawk.ethz.ch/overview>, janvier 2016.
- [13] Euclidianspace. <http://www.euclideanspace.com/physics/kinematics/orientation>, janvier 2016.
- [14] Geogebra application. <https://www.geogebra.org/>, janvier 2016.

- [15] Github pixhawk firmware. <http://github.com/PX4/Firmware>, février 2016.
- [16] Image a2. [http://www.droneshop.com/dji-a2-pro-/,](http://www.droneshop.com/dji-a2-pro-/) Fervier 2016.
- [17] Image capteur à ultrasons. [https://www.parallax.com/sites/default/files/styles/full-size-product/public/28015a.png?itok=35IBmfW8/,](https://www.parallax.com/sites/default/files/styles/full-size-product/public/28015a.png?itok=35IBmfW8/) Fervier 2016.
- [18] Image carte emmc. [http://www.hardkernel.com/main/products/prdt_info.php?g_code=G138750148750,](http://www.hardkernel.com/main/products/prdt_info.php?g_code=G138750148750) Mai 2016.
- [19] Image drone f550. [https://i.ytimg.com/vi/XMCmV_ZGYgY/maxresdefault.jpg/,](https://i.ytimg.com/vi/XMCmV_ZGYgY/maxresdefault.jpg/) Fervier 2016.
- [20] Image gps, d-gps et gps-rtk. [http://tpe-arpentage.e-monsite.com/pages/dossier/iii-1-erreur-et-1-amelioration-de-la-precision.html/,](http://tpe-arpentage.e-monsite.com/pages/dossier/iii-1-erreur-et-1-amelioration-de-la-precision.html/) Fervier 2016.
- [21] Image laser lidar lite v2. [http://pulsedlight3d.com/products/lidar-lite-v2-blue-label.html/,](http://pulsedlight3d.com/products/lidar-lite-v2-blue-label.html/) Mai 2016.
- [22] Image laser terra. [http://www.robotshop.com/eu/fr/capteur-distance-tof-teraranger-one-type-a-spider.html/,](http://www.robotshop.com/eu/fr/capteur-distance-tof-teraranger-one-type-a-spider.html/) Mai 2016.
- [23] Image odroid. [http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127,](http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127) Mai 2016.
- [24] Image pixhawk. [https://3drobotics.zendesk.com/hc/en-us/articles/207358096-3DR-Pixhawk/,](https://3drobotics.zendesk.com/hc/en-us/articles/207358096-3DR-Pixhawk/) Fervier 2016.
- [25] Image raspberry pi. [https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/,](https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/) Mai 2016.
- [26] Librairie d'acquisition des lasers lidarlitev2. [https://github.com/AlexisTM/LIDAREnhanced,](https://github.com/AlexisTM/LIDAREnhanced) avril 2016.
- [27] List of ros apis. [http://wiki.ros.org/APIs,](http://wiki.ros.org/APIs) janvier 2016.
- [28] List of ros packages. [http://www.ros.org/browse/list.php,](http://www.ros.org/browse/list.php) janvier 2016.
- [29] Xmaxima application. [http://maxima.sourceforge.net/,](http://maxima.sourceforge.net/) janvier 2016.
- [30] Alexis Paques & Nabil Nehri. Localisation et contrôle d'un drone via des lasers fixes. [https://github.com/AlexisTM/Indoor_Position_lasers,](https://github.com/AlexisTM/Indoor_Position_lasers) développé en 2016.

Sixième partie

Annexes

CHAPITRE A

PixHawk

Abstract

Ce chapitre se concentre sur les détails pratiques de l'utilisation du PixHawk et de la télécommande avec le PixHawk.

A.1 Le contrôleur

Le PixHawk est un contrôleur de vol open-source et open-hardware. Il sert à contrôler chaque moteur du drone indépendamment afin de stabiliser le drone. Le PixHawk n'est pas limité à un drone, mais il excelle par rapport à ses concurrents open-source dans le domaine. Le PixHawk est supporté et vendu par *3DR Robotics*¹.

- Il est configuré dans un environnement RTOS, Real-Time Operating System
- Ports UART/I2C/SPI/CAN : Canaux de communication série, permettant de brancher directement des capteurs optionnels sur le PixHawk
- IMU : Composé d'un baromètre, accéléromètre et magnétomètre afin de déterminer la position du véhicule dans l'espace

A.2 Communication

Afin de communiquer avec le PixHawk, un protocole série adapté a été développé, il s'agit de MavLink pour Micro Aerial Vehicle Link. Ce protocole fonctionne sous forme de paquets de données qui sont envoyées de façon série. [8]

Un paquet MAVLink contient 17+ octets :

Header (6 bytes) Le destinataire, l'expéditeur, la séquence, la longueur du message

1. Un des plus grands fabricants de drones commerciaux et récréatifs aux États-Unis



FIGURE A.1 – Le PixHawk

Payload/Data (0 à 255 bytes) La charge utile, les données

Checksum (2 bytes) Code CRC pour vérifier l'intégrité du message

En pratique, il sera rare d'utiliser MavLink, car Mavros s'occupe de l'abstraction du drone.

A.3 Configuration

qGroundControl est le logiciel qui permet de communiquer (via MAVLink) avec le PixHawk afin de réaliser les réglages de base. Il permet entre autres de configurer, calibrer et mettre à jour le PixHawk. Pour ce qu'il en est des paramètres, il est possible (script `laserpack/bin/INA9_OPTIONS.py`) de les envoyer via Mavros.

- Branchez le PixHawk en USB
- Allumez qGroundControl
- En haut à droite, cliquez sur connexion puis sur PixHawk
- Dans l'aperçu principal, vous pouvez voir quelles sont les étapes qui requièrent votre attention, toutes les étapes doivent être passées en revue en cas de première utilisation.
- Lors du premier lancement, choisissez le frame du multi-copter, cela permet de faire une régulation correcte des moteurs
- Ensuite, calibrez les capteurs dans l'onglet *sensors*, puis la télécommande dans l'onglet *radio*.

A.4 Modes de vol

Les différents modes de vols permettent plus ou moins de facilités et niveaux de contrôle du véhicule.

Stabilized (Main) Le mode Stabilisé est le mode principal, donc le mode dans lequel doit être le véhicule pour être allumé ou éteint. Tous les contrôles sont donnés à l'utilisateur. Il a le contrôle du throttle, et de chacun des angles (pitch, yaw et roll), nommés attitude, du véhicule.

Altitude Altitude est similaire au mode Stabilisé à la différence que le PixHawk gérera l'altitude lui même. Ainsi, le contrôle de la puissance des moteurs est non utilisable. Montez en mode Stabilized jusqu'à l'altitude voulue puis passez au mode Altitude pour maintenir l'altitude tout en contrôlant le contrôle des angles.

Position Ceci est le mode le plus simple à utiliser. Chaque joystick est un contrôle en position, throttle commande l'altitude, au centre il reste stable, yaw fait tourner le véhicule sur lui même, pitch fait avancer le véhicule et roll le fait aller de gauche à droite.

Accro Le multirotor prend en mémoire son attitude actuelle et cherche à y retourner. Si vous tournez le joystick, une fois que vous le relâchez, le véhicule reviendra à son attitude originale.

Offboard Le contrôle des moteurs est donné à une carte extérieure qui communique en MAVLink.

Loiter Mode de maintien de position, ne fonctionne qu'avec le GPS.

Auto Le mode auto effectue une mission, en suivant des checkpoints puis revient au point initial. Ne fonctionne qu'avec le GPS.

Return Le mode return est la procédure d'urgence en cas de perte de communication, il va à une certaine altitude, retourne au point de démarrage et après avoir attendu un certain temps, descends. Ne fonctionne qu'en extérieur et avec un GPS.

A.5 Utilisation pratique

Démarrer le véhicule

1. Vérifiez que tout est branché
2. Vérifiez la connexion entre la télécommande et le PixHawk
3. Vérifiez l'état de la connexion entre qGroundControl ou le logiciel de contrôle/télé-métrie avec le PixHawk
4. Vérifiez l'état de la batterie (> 3V par cellule, proche de 4.2V)
5. Vérifiez à deux fois les headers de contrôle des drivers moteurs (ESC), dans le bon ordre
6. Restez appuyé sur le switch avec la LED, s'il clignote une fois, il n'est pas armé, s'il clignote deux fois, il est armé
7. Mettez le mode de vol Stabilisé
8. Placez votre joystick de throttle/yaw (joystick de gauche) en bas à droite
9. Mettez le mode de vol voulu et vous êtes partis

Éteindre le véhicule

1. Atterrir
2. Mettez le mode de vol Stabilisé
3. Placez votre joystick de throttle/yaw (joystick de droite) en bas à gauche

Configuration de la télécommande Le mode de configuration permet de configurer des profils, ou changer de profil. Pour chaque profil, il est possible d'associer les boutons de la télécommande aux différents channels de sortie (GEAR, AUX1, AUX2, AUX3). Les deux joysticks comptent déjà pour quatre canaux et sont transmis dans tous les cas. De plus, vous pouvez modifier les fonds d'échelles de chaque bouton, utile pour des servomoteurs par exemple.

- Mettez tous les switchs de la télécommande à l'état 0
- Mettez le throttle (puissance des moteurs, joystick de gauche) au minimum
- Démarrer la télécommande en appuyant sur le bouton de sélection rotatif
- Mettez tous les switchs à l'état 0
- Mettez le throttle (puissance des moteurs, joystick de gauche) au minimum
- Démarrer la télécommande avec le bouton central
- Lancez qGroundControl et vérifiez tous les contrôles et switchs configurés

Lier la télécommande au récepteur

- Branchez une antenne DSMX sur le PixHawk au port DSM prévu à cet effet
- Branchez une antenne DSMX sur le récepteur AR-9050
- Branchez le câble BIND au récepteur AR-9050, dans le port prévu à cet effet
- Alimentez le récepteur AR-9050 via le header RC-IN du PixHawk
- Mettez tous les switchs de la télécommande à l'état 0
- Mettez le throttle (puissance des moteurs, joystick de gauche) au minimum
- Démarrer qGroundControl, après s'être connecté au PixHawk, dans la section radio, cliquez sur Bind, la LED de l'antenne devrait clignoter
- Restez appuyé sur le bouton Trainer lors de l'allumage de la télécommande
- La procédure se lance automatiquement et est terminée lorsque vous voyez typiquement DSMX 22 sur votre télécommande et que la LED de toutes les antennes soit allumée

Calibration de la télécommande Après avoir démarré et lié votre télécommande, il faut la calibrer dans qGroundControl.

- Lancez votre télécommande
- Lancez qGroundControl
- Connectez-vous au PixHawk (USB), dans l'onglet radio, cliquez sur Calibrate
- Calibrez les joysticks, utilisez tous les switchs et vous pouvez passer la configuration des flaps et de AUX1 qui ne sont pas utilisés dans le cas d'un multi-rotor

CHAPITRE B

Librairie d'acquisition des lasers

B.1 Introduction

La librairie LidarLiteEnhanced [26] disponible sur Github est la réécriture complète de la librairie de base afin d'en améliorer les performances.

Améliorations vis-à-vis de la librairie de base

- Acquisition **asynchrone**
- Architecture **non bloquante**
- Reset **automatique** des lasers
- **Machine à états** pour chaque laser
- **Vitesse** d'acquisition optimisée
- Comprenant le bootloader Arduino, le poids de la librairie est de **7200 bytes**

Limitations

- Limitation à l'utilisation de 8 lasers, cette limite peut être augmentée
- Pas de callback avec une nouvelle mesure, on doit directement lire l'espace de stockage des mesures
- Pas de lecture de la vitesse

Le but était de rendre l'utilisation la plus simple possible, soit pour l'utiliser, il suffit de :

- Inclure les librairies
- Instancier les objets
- Ajouter des lasers
- Démarrer le bus I2C
- D'exécuter la machine à états

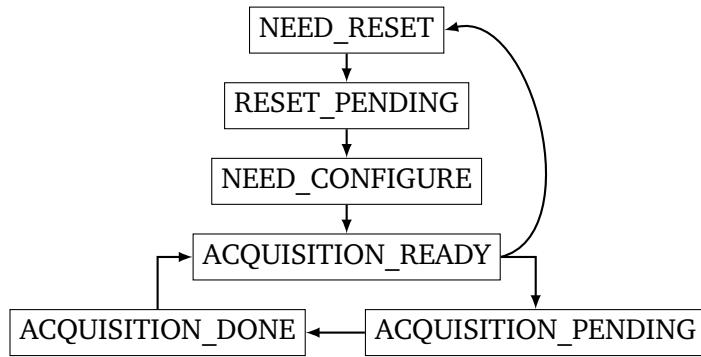


FIGURE B.1 – Machine à états implémentée pour chaque laser

Les états possibles des lasers sont comme suit, avec entre parenthèses les bits de statut équivalents. Elles s'agencent comme à la figure B.1

- NEED_RESET : Le programme vient d'être lancé où il y a eu trop d'erreurs, ce laser a besoin d'être réinitialisé. (0011 xxxx)
- RESET_PENDING : Nous devons attendre 15ms après avoir réinitialisé le laser, si les 15ms sont passées, alors on passe à l'état NEED_CONFIGURE (0101 xxxx)
- NEED_CONFIGURE : Il faut envoyer la configuration au laser (1001 xxxx)
- ACQUISITION_READY : Le laser est prêt à lancer une acquisition (0010 xxxx)
- ACQUISITION_PENDING : Le laser réalise une acquisition (0100 xxxx)
- ACQUISITION_DONE : L'acquisition est réalisée (1000 xxxx)

B.2 Exemple concret

La première chose à réaliser est d'inclure les librairies et de définir les paramètres des lasers

```

#include "LidarObject.h"
#include "LidarController.h"
#include "I2CFunctions.h"

#include <Wire.h>

/** Defines : CONFIGURATION ***/
// Defines laser ready data
#define Z1_LASER_PIN 11
#define Z2_LASER_PIN 8
#define Z3_LASER_PIN 5
#define Z4_LASER_PIN 2
#define Z5_LASER_PIN 16
#define Z6_LASER_PIN 19
// Defines power enable lines of laser
#define Z1_LASER_EN 12
#define Z2_LASER_EN 9
#define Z3_LASER_EN 6
#define Z4_LASER_EN 3
#define Z5_LASER_EN 15
#define Z6_LASER_EN 18
    
```

```
//Define address of lasers
//Thoses are written during initialisation
// default address : 0x62
#define Z1_LASER_AD 0x6E
#define Z2_LASER_AD 0x66
#define Z3_LASER_AD 0x68
#define Z4_LASER_AD 0x6A
#define Z5_LASER_AD 0x6C
#define Z6_LASER_AD 0x64
```

Ensuite, on instancie les objets LidarController et LidarObject

```
static LidarController Controller;
static LidarObject LZ1;
static LidarObject LZ2;
static LidarObject LZ3;
static LidarObject LZ4;
static LidarObject LZ5;
static LidarObject LZ6;
```

Une fois instanciés, on ajoute les lasers au Controller et on démarre le bus I2C

```
// Initialisation of the lidars objects
LZ1.begin(Z1_LASER_EN, Z1_LASER_PIN, Z1_LASER_AD, 2, 'x');
LZ2.begin(Z2_LASER_EN, Z2_LASER_PIN, Z2_LASER_AD, 2, 'X');
LZ3.begin(Z3_LASER_EN, Z3_LASER_PIN, Z3_LASER_AD, 2, 'y');
LZ4.begin(Z4_LASER_EN, Z4_LASER_PIN, Z4_LASER_AD, 2, 'Z');
LZ5.begin(Z5_LASER_EN, Z5_LASER_PIN, Z5_LASER_AD, 2, 'y');
LZ6.begin(Z6_LASER_EN, Z6_LASER_PIN, Z6_LASER_AD, 2, 'Z');

// Initialisation of the controller
Controller.begin(WIRE400K);
delay(10);
Controller.add(&LZ1, 0);
Controller.add(&LZ2, 1);
Controller.add(&LZ3, 2);
Controller.add(&LZ4, 3);
Controller.add(&LZ5, 4);
Controller.add(&LZ6, 5);
```

Ensuite, dans la boucle principale, nous exécutons le contrôleur, le plus souvent possible.

```
Controller.spinOnce();
```

Enfin, nous avons accès aux distances mesurées via distances qui est une liste d'entiers sur 16 bits et statuses qui est une liste d'integers non signés représentant l'état des lasers

```
for(byte i = 0; i < 6; i++){
    Serial.print(i);
    Serial.print(" - ");
    Serial.print(Controller.distances[i]);
    Serial.print(" - ");
    Serial.println(Controller.statuses[i]);
}
```

LidarController implémente une machine à état qui exécute la prochaine action pour les lasers à chaque fois que l'on exécute la fonction spinOnce().

B.3 Erreurs typiques

- Ne pas oublier les résistances de pull-up
- Est-ce que la masse est interconnectée entre les deux périphériques ?
- Est-ce que la tension du bus est acceptée par les deux parties ? (3v3 ou 5v)

C CHAPITRE

Carte électronique

La carte électronique est un shield¹ Arduino. Elle permet de connecter 6 lasers à une seule Arduino, via des connecteurs MOLEX-6.

La carte comporte

- Circuit des lasers
- Conversion des tensions de bus de données
- Bouton de redémarrage (RESET)

Circuit des lasers Le circuit des lasers est composé de deux circuits permettant de contrôler les lasers de différentes façons. Afin de ne pas perdre une capacité d'un laser, les deux méthodes des figures C.1 et C.2 ont été incorporées en une.

Level shifter Le niveau de tension recommandé pour le bus I2C des lasers est de 3.3V. étant donné que l'Arduino choisie fonctionne en 5V, il faut convertir ces niveaux de tension. Pour cela, il faut utiliser un *bidirectional level shifter*.

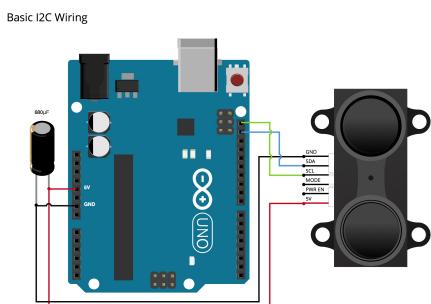


FIGURE C.1 – Méthode de connexion via l'I2C

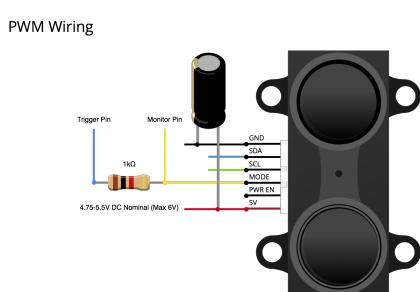


FIGURE C.2 – Connexion via la longueur d'un signal

1. Carte électronique se plaçant au-dessus d'une carte mère

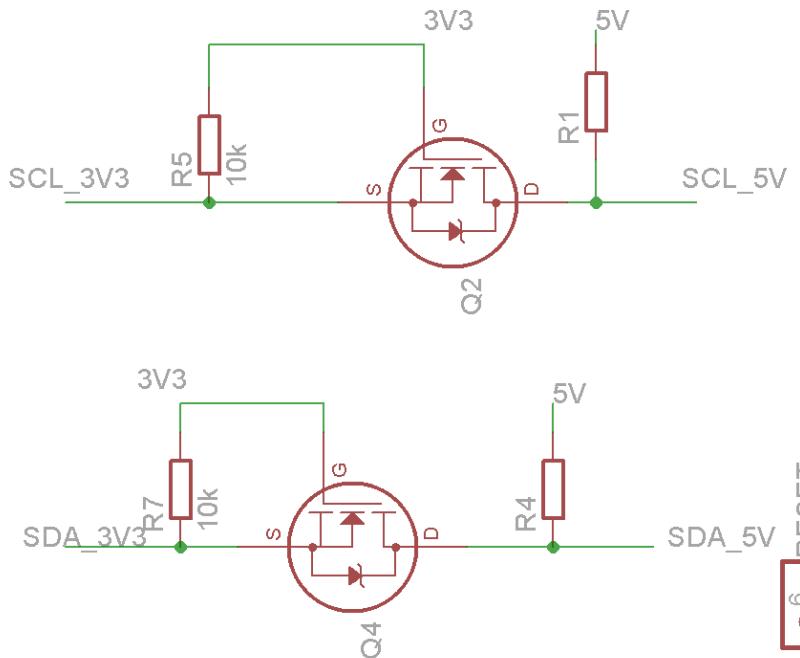


FIGURE C.3 – Level shifter bidirectionnel 3.3V, 5V

Le niveau de tension, selon la norme du bus I2C, doit être à l'état haut tout le temps sauf pour une donnée à l'état bas. Dès lors, si un des deux côtés de la ligne est à l'état bas, il faut mettre l'autre bout à l'état bas. Cela peut être réalisé soit avec deux transistors bipolaires, soit grâce à un transistor MOSFET 2N7000 comme montré à la figure C.3. Cette méthode est expliquée en long et en large sur internet [9].

La figure C.4 permet de connaître la configuration des jumpers SDA et SCL.

Les différentes pins correspondent à :

- 2 : SDA/SCL 3V3
- 3 : SDA/SCL 5V
- 4 : SDA/SCL

Pour fonctionner avec un niveau de tension de 5V à l'I2C (SDA/SCL), il faut placer le jumper sur les pins 3 et 4 dans les deux cas. La pin 4 est reliée au bus côté laser, la pin 3 au bus 5V et la pin 2 au bus 3.3V.

Bouton redémarrage L'Arduino est recouverte du shield, le bouton de reset n'est donc plus accessible, et il a donc fallu le transmettre au shield comme montré à la figure C.5.

Connections La figure C.1 permet de connaître les différentes entrées/sorties des lasers sur l'Arduino et la configuration qu'il faut utiliser dans le code d'acquisition des lasers.

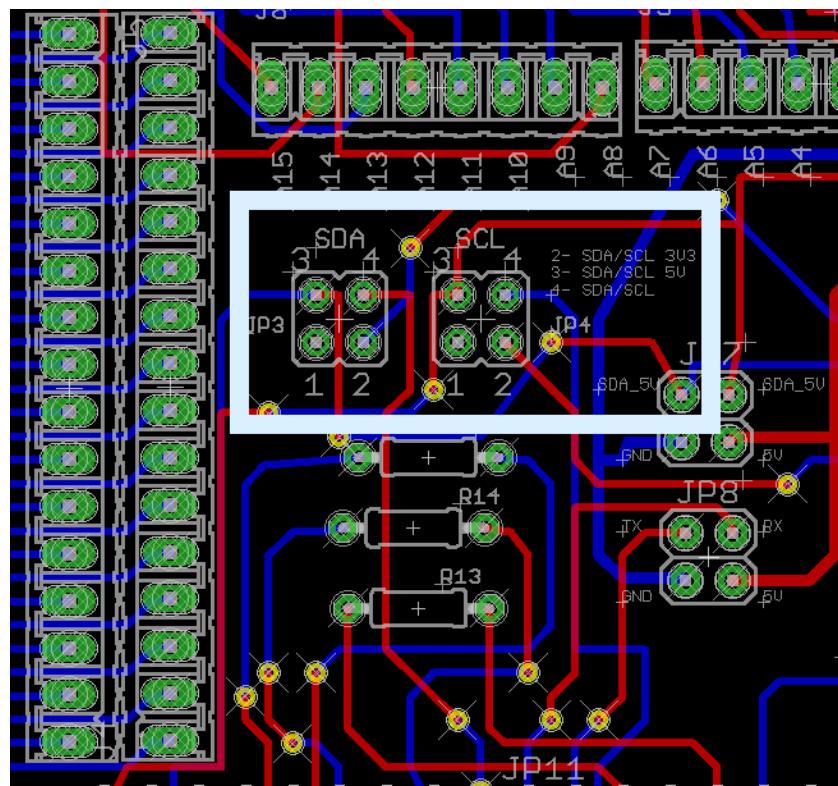


FIGURE C.4 – Jumper pour changer la tension SDA/SCL

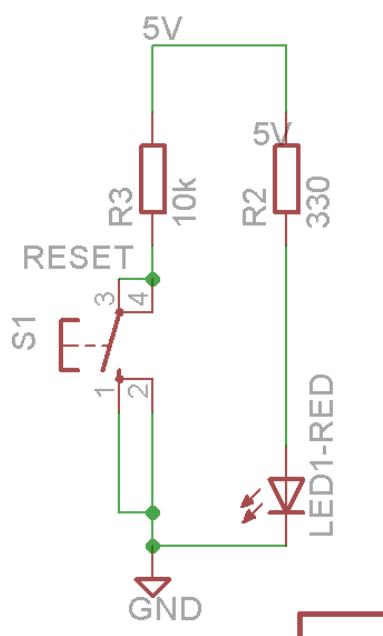


FIGURE C.5 – Schématique du reset du shield

Laser	Mode (monitor pin)	Pwr_en	Trigger pin
1	D13	D12	D11
2	D10	D9	D8
3	D7	D6	D5
4	D4	D3	D2
5	D14	D15	D16
6	D17	D18	D19

TABLE C.1 – Récapitulatif du pin mode

CHAPITRE D

Installation et configuration du système

Nous expliquerons ici comment installer ROS et configurer Linux, en détail.

D.1 ROS

Sans information, il est possible de perdre beaucoup de temps dans l'installation de ROS, ce qui est dommage sachant que tout est disponible sur leur site web [1]. C'est pourquoi, voici les étapes afin d'installer ROS.

D.1.1 Installez Ubuntu

Pour installer Ubuntu (préférez Kubuntu), il suffit de créer une clé USB Bootable via LINUX LIVE USB CREATOR avec **Kubuntu 14.04-04**

D.1.2 Redémarrez

D.1.3 Ajoutez les sources de packages ROS

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" <--> /etc/apt/sources.list.d/ros-latest.list'  
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net --recv-key 0xB01FA116
```

D.1.4 Installation via Aptitude

```
sudo apt-get update
sudo apt-get install ros-indigo-desktop-full ros-indigo-mavros ros-indigo-mavros-←
    extras python-rosinstall ros-indigo-rosserial ros-indigo-rosserial-arduino ←
    python-catkin-tools
```

D.1.5 Initialiser ROSDEP

```
sudo rosdep init
rosdep update
```

D.1.6 Initiez les variables d'environnement

```
echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc # à n'excuter qu'une fois
source ~/.bashrc
```

D.1.7 Trouver des paquets ROS

Afin de trouver des paquets ROS, il suffit de chercher dans la liste provenant d'APT.

```
apt-cache search ros-indigo
```

D.2 Linux

D.2.1 Droits d'utilisation

Afin de pouvoir utiliser les ports série, il faut ajouter son compte à la liste d'utilisateurs autorisés.

```
usermod -a -G tty ros-user  
usermod -a -G dialout ros-user
```

D.2.2 USB Symlink

Il est intéressant de créer un lien symbolique entre le périphérique et le port où l'on veut qu'elle soit. En effet, le Pixhawk et l'Arduino, par défaut, ont le même nom, soit **/dev/ttyACM0** ou **/dev/ttyACM1** selon que le périphérique soit branché en premier ou second. Afin d'éviter le problème, il suffit d'ajouter une règle UDEV selon l'ID du périphérique. Pour se faire, il faut ajouter ces quelques lignes dans un nouveau fichier du dossier : **/etc/udev/rules.d/99-ttyNames.custom.rules**

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="2341", ATTRS{idProduct}=="0042", SYMLINK+="↳  
    ttyLasers"  
SUBSYSTEM=="tty", ATTRS{idVendor}=="26ac", ATTRS{idProduct}=="0011", SYMLINK+="↳  
    ttyPixHawk"
```

Si l'on change d'Arduino, il faut donc changer le vendeur et le produit. Pour ce faire, branchez l'Arduino, et utilisez la commande **lsusb** afin d'obtenir les ID. Un exemple de sortie de **lsusb** peut être :

```
Bus 001 Device 008: ID vvvv:pppp Arduino SA Mega 2560 R3 (CDC ACM)
```

Dans ce cas-ci, **vvvv** est l'**idVendor** et **pppp** l'**idProduct**. De plus, le symlink doit avoir le nom : **ttyCustomName**, afin de savoir que c'est une connexion série.

CHAPITRE E

Utilisation du projet

Abstract

Nous allons ici expliquer en détail comment installer et utiliser notre projet.

E.1 Installation du projet

E.1.1 Installation des messages

Créez un workspace Pour plus d'informations, allez voir la documentation de catkin.

```
cd  
mkdir workspace  
cd workspace  
mkdir src  
catkin init
```

Ajoutez le projet dans le workspace Seul le dossier laserpack de Github [30] doit être copié dans le dossier src du workspace.

```
cd  
git clone https://github.com/AlexisTM/Indoor_Position_lasers  
cp -r Indoor_Position_lasers/laserpack workspace/src/laserpack  
cd workspace  
catkin build
```

Ajoutez un lien permanent vers le workspace Notez bien que username est votre nom d'utilisateur, cette commande ajoute la ligne à la fin du fichier bashrc.

```
echo 'source /home/username/workspace/devel/setup.bash' >> ~/.bashrc  
source ~/.bashrc
```

E.1.2 Installation de l'Arduino

Connexions Il suffit de connecter le shield à l'Arduino, connecter les lasers et brancher le câble USB de l'Arduino au PC.

Génération des librairies Arduino Sous un PC sous ROS, **après** avoir installé les messages, exécutez ROSSerial pour générer les librairies. Pour des raisons de facilité, le dossier ros_lib a été copié dans le dossier Arduino de Github.

```
cd  
rosrun rosserial_arduino make_libraries.py .
```

Compilez Afin de compiler le script, copiez le dossier ros_lib dans le dossier des librairies de votre IDE Arduino

```
# Sous Windows C:\users\username\Documents\Arduino\libraries\ros_lib  
# Sous Linux /home/<username>/Sketchbook/libraries/ros_lib
```

Sur Github, dans le dossier Arduino plusieurs codes existent.

```
# SixLasers : Code de test pour Windows  
# SixLasersROS : Code avec éintegration pour ROS  
# ZLasers : Code pour deux lasers sous ROS  
# ros_lib : Librairies éégnres par ROSSerial, pour plus de éfilit
```

Lorsque vous compilez avec l'IDE d'Arduino, faites attention de bien sélectionner le bon port et la bonne Arduino (MEGA 2560). Si vous vous trompez, l'IDE ne saura juste pas envoyer le programme, rien de grave.

E.1.3 Liste des messages

Tous les messages sont sur Github dans le dossier laserpack/msg. Les autres messages tels que PoseStamped ou Twist sont des messages standards de ROS.

Distance.msg Un array contenant les distances mesurées par les lasers et un second avec les status de chacun des lasers

Req_reset.msg Message à envoyer à l'Arduino afin de redémarrer un laser

Battery.msg Donne l'information de la batterie du drone, utilisé par le message Report

RPY.msg État Roll Pitch Yaw en degrés, utilisé par le message RPYPose

RPYPose.msg Donne la position et l'état angulaire du drone, utilisé par Report

Task.msg Représente une tâche à effectuer

Mission.msg Une liste de tâches, à utiliser pour effectuer une mission

Report.msg Renvoie toutes les informations du drone à la page web.

Reports.msg Renvoie toutes les informations de plusieurs drones, s'il y en avaient plusieurs.

E.2 Démarrage du drone

E.2.1 Connexion

Afin de pouvoir contrôler le drone, il vous faut tout d'abord une connexion au drone. Assurez-vous bien qu'il soit connecté au même réseau WIFI que vous. Pour ce faire, connectez un clavier, un écran (HDMI) et une souris, ainsi qu'une **clé WiFi**.

```
# user : odroid  
# password : odroid
```

WiFi Il ne sera pas possible de vous connecter à l'Odroid si vous êtes sur un réseau avec des restrictions tels que Eduroam ou les réseaux UCL. Vous devez donc faire un réseau WiFi. Cela est aisément réalisable avec un GSM Android par exemple. Sinon un clé WiFi avec un driver adapté sous Windows devrait faire l'affaire.

Trouver l'IP Si vous êtes sur l'Odroid, il vous suffit d'exécuter **ifconfig** pour trouver l'IP. Si vous savez qu'elle est connectée au réseau WiFi mais que vous ne connaissez pas son IP, vous pouvez la trouver avec **nmap**, sous Windows ou Linux, avec 192.168.1.1, votre IP.

```
nmap -T4 -F 192.168.1.1/24
```

Se connecter à l'Odroid Nous vous conseillons d'utiliser WinSCP pour vous connecter via SFTP (FTP via SSH) et putty pour vous connecter en SSH. Configurez WinSCP en ajoutant en client SSH putty, et vous aurez la possibilité d'ouvrir, sous WinSCP, une fenêtre putty connectée à l'Odroid avec le raccourci **CTRL+P**.

Naviguer sur l'Odroid Afin d'avoir plus simple que d'avoir 5 consoles ouvertes, utilisez tmux. C'est un programme permettant d'ajouter plusieurs consoles ensemble. Toutes les informations sont sur internet. (tmux cheatsheet)

E.2.2 Lancement des scripts

Nœud d'acquisition et mavros Ce script lance le serveur d'écoute des lasers rosserial, ainsi que la couche d'abstraction du PixHawk, Mavros.

```
cd  
cd Indoor_Position_lasers/laserpack  
roslaunch launch/serial.launch
```

Nœud d'algorithme Ce code calcule le yaw ainsi que la position du drone.

```
cd  
cd Indoor_Position_lasers/laserpack/bin  
python algorithm_class.py
```

Nœud de télémetrie Ce code publie sur le topic web/export toutes les informations du drone.

```
cd  
cd Indoor_Position_lasers/laserpack/bin  
python web_export.py
```

Nœud de pont vers la page web Rosbridge sert de pont entre ROS et tout autre système en envoyant un JSON contenant les paquets demandés.

```
roslaunch rosbridge_server rosbridge_websocket.launch
```

Nœud de contrôle manuel Ce code gère le contrôle de tout le drone via des setpoints. Le contrôle est hardcodé via un getch afin d'avoir un contrôle simpliste.

```
cd  
cd Indoor_Position_lasers/laserpack/bin  
python control_thread.py
```

Contrôles du nœud

```
# i = monte de 10 cm  
# k = descends de 10 cm  
# t = diminue ta position en X de 10 cm (rapproche-toi du mur x)  
# g = augmente ta position en X de 10 cm éloigne-toi du mur x)  
# h = diminue ta position en Y de 10 cm (rapproche-toi du mur y)  
# f = diminue ta position en Y de 10 cm éloigne-toi du mur y)
```

Nœud de tâches Le nœud de tâches est un nœud alternatif à celui de contrôle manuel, mais qui n'est pas terminé¹. Le système de tâche est très simple, mais il faut encore faire un nœud recevant les tâches. En attendant, le nœud test.tasks.py est un exemple d'utilisation du système de tâches.

1. Ce code devait être réalisé par un autre TFE, mais il n'a pris le temps de nous voir que mi-avril.

```
cd  
cd Indoor_Position_lasers/laserpack/bin  
python test.tasks.py
```

E.2.3 Télémétrie

Sur votre PC qui est connecté au même réseau que l’Odroid, lancez la page index.html en provenance du dossier Interface de Github. Si rien ne bouge, c’est simplement que l’IP n’est pas correcte, dès lors, ouvrez le script interface.js et changez l’IP par celle de l’ODROID dans l’objet Configurations.

E.2.4 Liste des scripts et leur utilité

Les fichiers listés ici proviennent de Github, du dossier laserpack/bin.

INAV_OPTIONS.py Script d’exemple de script de changement de variables pour le Pix-Hawk.

Kalman.py Classe avec tous les filtres y sont implémentés, ainsi qu’un container pour pouvoir instancier tous les filtres en une seule variable et appliquer les filtres à un groupe de valeurs. Le nom du fichier provient du filtre qu’il était prévu d’implémenter, que l’on a implémenté, mais qui convergeait vers un modèle beaucoup plus simple.

Kalman_test.py Script d’exemple de script de test pour les filtres implémentés dans Kalman.py.

algorithm_class.py Script réalisant l’algorithme, récupérant le yaw et déterminant la position du drone.

algorithm_functions.py Functions utilisées pour l’algorithme

control_tasks.py Script d’essai de contrôle du drone via les tâches, qui elles utilisent Mavros

control_thread.py Script de contrôle du drone directement via la surcouche Mavros

export_data_csv.py Script d’exportation les données du drone en un CSV nommé result.csv. Appuyez sur q pour arrêter l’acquisition, et l’acquisition dure maximum 120 secondes (12000 valeurs)

getch.py Librairie fournissant une interface minimale avec un contrôle via le clavier

lasers.py Classe de configuration des lasers, dont leur position et orientation sur le drone, ainsi que quelques fonctions de simplification.

tasks.py Classe reprenant toutes les tâches, la représentation d'un drone pour les tâches ainsi que le contrôleur des tâches.

test.tasks.py Script d'exemple d'utilisation de tasks.py pour contrôler le drone

transformations.py Fonctions de transformations, généré par la librairie TF, pour pouvoir développer sous Windows, devrait être remplacée par la librairie TF.

web_export.py Script concaténant toutes les données provenant du drone et les envoyant à l'interface web

CHAPITRE F

Design 3D

Abstract

Ici seront montrés les éléments 3D qu'il a fallu concevoir.

Pour fixer les lasers il a fallu dessiner différents éléments de support pour les fixer sur le drone. Les lasers en X doivent être un maximum écarté pour pouvoir avoir une bonne précision du YAW. Sur le petit drone, il était assez complexe de trouver un endroit où le laser n'est pas encombré par les hélices ou un des bras.

F.1 Supports des lasers

Un support était existant pour fixer les lasers en Open-Source et un autre qui a été dessiné pour pouvoir fixer deux lasers (Un pour l'axe des X et l'autre pour l'axe Z)

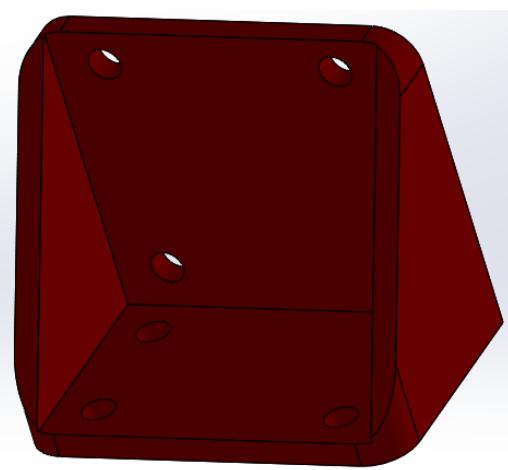


FIGURE F.1 – Support pour fixer un laser

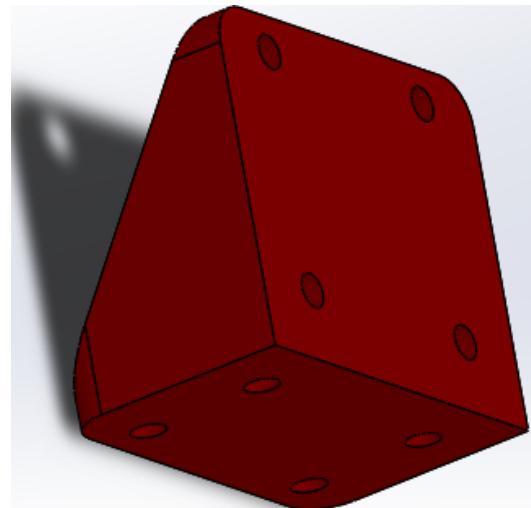


FIGURE F.2 – Support pour fixer un laser

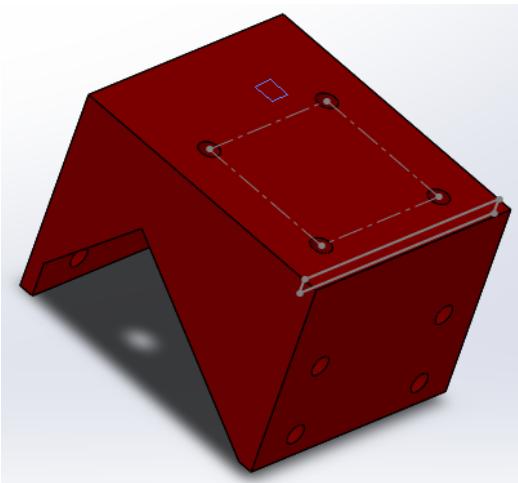


FIGURE F.3 – Support pour fixer deux lasers

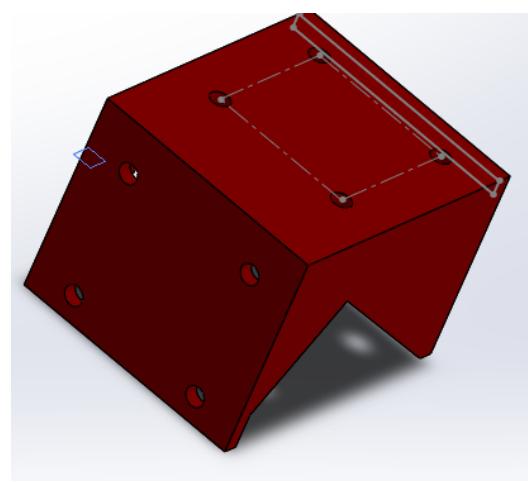


FIGURE F.4 – Support pour fixer deux lasers

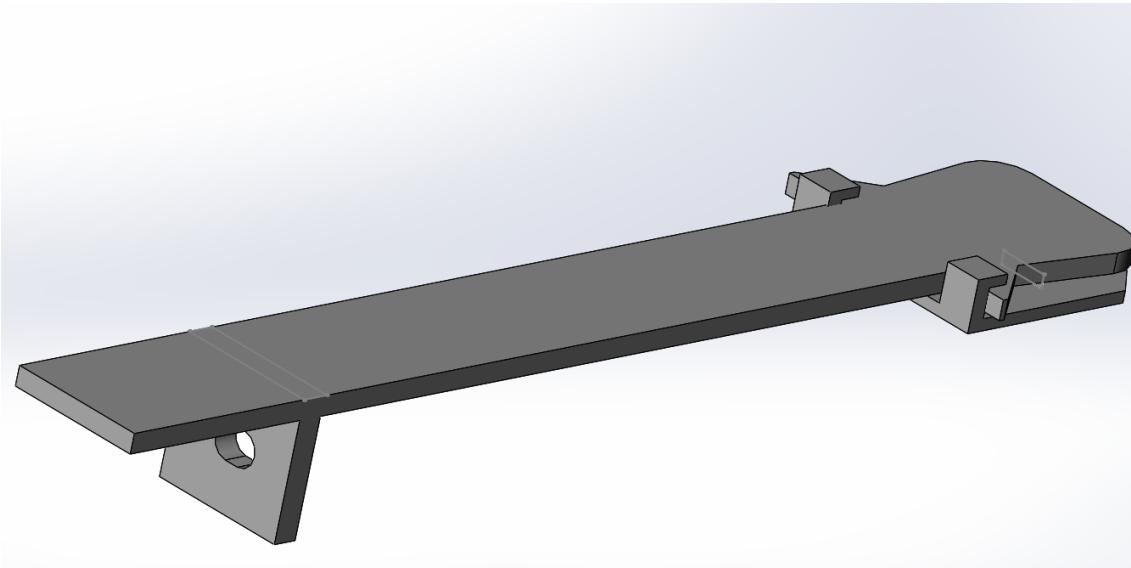


FIGURE F.5 – Assemblage avec les deux éléments

F.2 Bras des lasers

Le système est composé de deux éléments qui viennent se serrer de part et d'autre de la plaque inférieur du drone.

On a :

- Élément inférieur à fixer la plaque qui maintient les lasers sur le drone
- Élément supérieur qui sert à éloigner les lasers

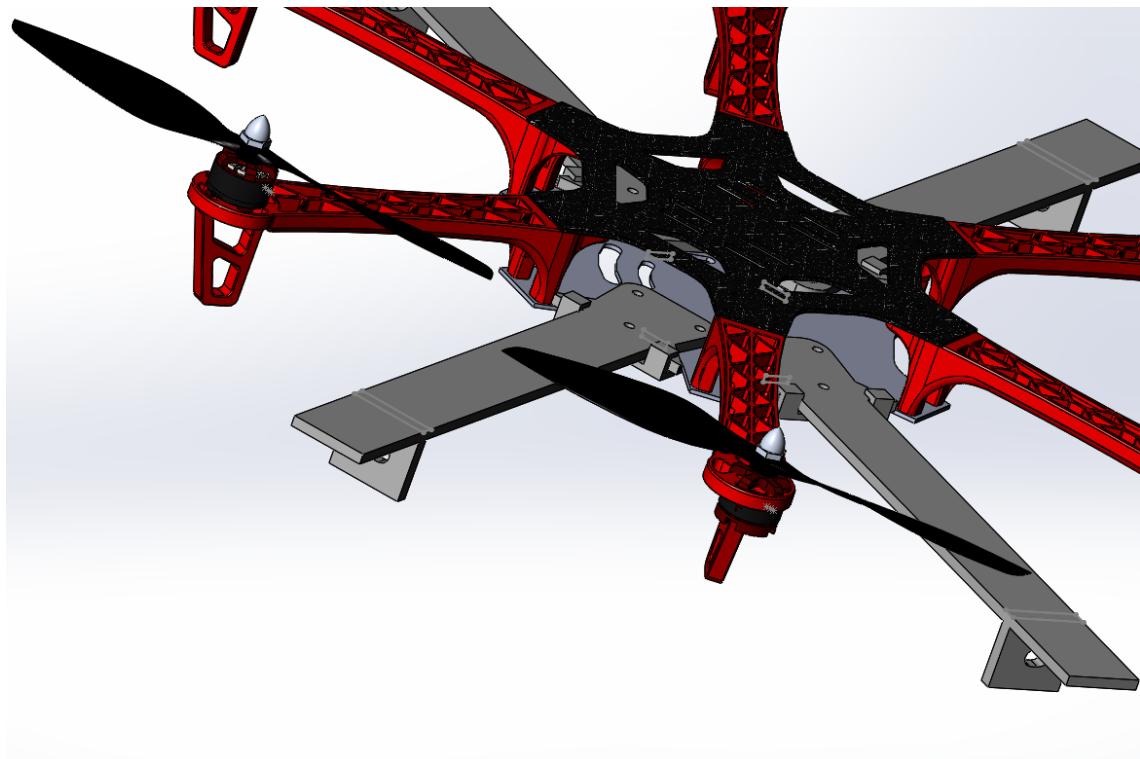


FIGURE F.6 – Installation du design sur le drone

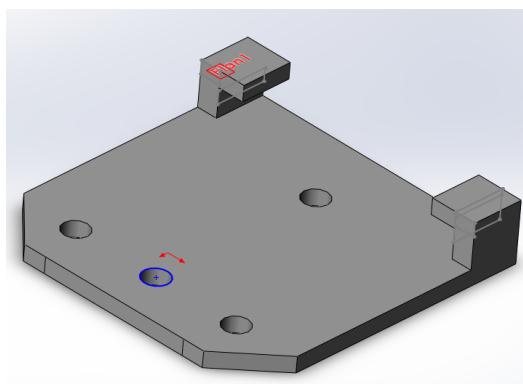


FIGURE F.7 – Élément inférieur

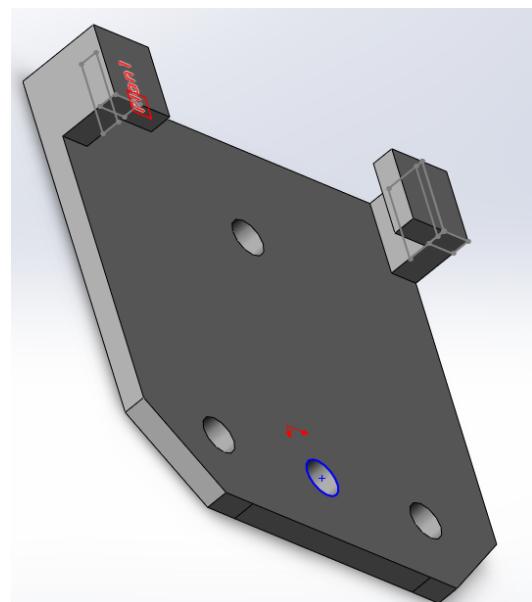


FIGURE F.8 – Élément inférieur

CHAPITRE G

Liste du matériel

Pour les développements, un ensemble d'achats ont été effectués. La liste qui suit reprend les achats nécessaires pour le petit drone de test, le système de localisation ainsi que l'installation des deux murs de références dans la DroneZone. Tous les prix sont avec TVA (toutes taxes comprises).

CHAPITRE G. LISTE DU MATÉRIEL

Description	Quantité	Prix(€)	Sous total(€)
<i>Matériel drone</i>			
Arduino Mega	2	14.00	28.00
Pixhawk	1	183.00	183.00
Carte de développement Odroid XU4	2	94.95	189.90
Frame drone F550	1	42.00	42.00
Driver moteur 30A opto	6	9.00	54.00
Moteurs brushless dji2212/920KV	6	20.00	120.00
Laser Lidar Lite V2	6	104.00	624.00
Chargeur de batterie	1	269.00	269.00
Batterie Turnigy 2200mAh 3S35-70C	1	17.50	17.28
Boitier odroid	2	7.95	15.90
Module Wifi Type 4	3	19.95	59.85
Module Wifi Type 3	1	10.95	10.95
Câble HDMI	2	3.95	7.90
Caméra USB	1	19.95	19.95
Module bluetooth USB	1	9.95	9.95
Cloudshell Odroid	1	49.95	49.95
Ecran Multitouch Odroid	1	79.95	79.95
Odroid XU4 Shifter shield	2	20.95	41.90
Breadboard	1	2.95	2.95
Carte mémoire eMMC de 32GB	2	55.95	111.90
Carte électronique	1	27.00	27.00
Module GPS NEO-7 uBox 3DR	1	99.00	99.00
Impression 3D	1	32.50	32.50
<i>Matériel Dronezone</i>			
VELCRO 20mmx2.5m	1	7.58	7.58
MDF 6 mm (122 x 244 cm)	10	11.58	115.80
Douglas 55 x 63 mm de 3.65 m	10	6.11	61.10
Paumelle 80x80 mm	8	2.85	22.80
Roulette pivot D100mm	3	17.00	51.00
		Total TVAC	2322.60