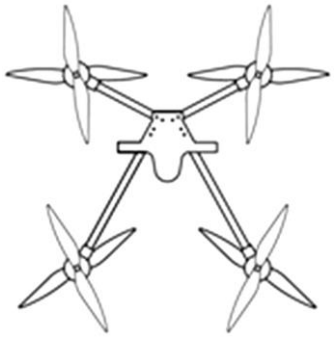
Système de guidage d'un drone à l'aide de capteurs lasers embarqués, destiné à la construction automatique d'ouvrages en maçonnerie



Ing. N. NEHRI
Ing. A. PAQUES
Ir C. MARCHAND
ECAM – Bruxelles

Prof. PhD C. MUELLER
MIT – Boston/USA

Ir S. GOESSENS
Prof Dr Ir P. LATTEUR
UCL – Louvain-la-Neuve

Ce travail s'inscrit dans le cadre d'un projet de collaboration entre l'UCL/pôle génie civil et environnemental et le MIT (Boston) visant à prouver la faisabilité de la construction automatisée d'ouvrages maçonnés à l'aide de robots volants, plus communément appelés drones. En particulier, ce travail concerne le développement d'une méthode de guidage et de positionnement d'un drone à l'aide de capteurs lasers embarqués pointant sur des écrans placés autour de la zone dans laquelle le drone doit déposer les blocs de maçonnerie.

Mots-clefs : drone, UAV, construction, maçonnerie, autonome, PixHawk, ROS, lasers, Arduino, Odroid

This work is part of a collaborative project between UCL/Civil and Environmental Engineering department and MIT (Boston) which intends to demonstrate the feasibility of automated construction of buildings using flying robots, commonly called drones. In particular, this work concerns the development of a drone guiding and positioning system using laser sensors, pointing on screens placed around the construction area.

Keywords: drone, UAV, construction, massonry, autonomous, PixHawk, ROS, lasers, Arduino, Odroid

1. Contexte

Dans le cadre d'un projet de recherche initié par Pierre Latteur en 2014 (UCL/Ecole Polytechnique/pôle génie civil et environnemental), un drone a été fabriqué sur mesure et constitue actuellement le plus gros drone multicoptère civil en Belgique (figure 1). Il est théoriquement capable de soulever des charges supérieures à 40 kg. A ce jour et pour différentes raisons liées notamment au train d'atterrissage qui nécessite un renforcement, il a été testé pour des charges allant jusqu'à 20 kg.

Figure 1 : le drone construit sur mesure pour cette recherche est capable de soulever des charges de 20 à 40 kg.

Le gros œuvre des bâtiments peut être constitué d'acier, de béton, de bois, de maçonnerie et de nombreux autres matériaux. Ce projet de recherche s'est focalisé dans un premier temps sur la construction en maçonnerie, en particulier sur des blocs en béton à géométrie adaptée pouvant être assemblés avec une certaine imprécision liée au drone qui, à ce jour, s'avère être de l'ordre de 5 cm en pilotage manuel. Les premiers résultats de cette recherche ont été présentés dans différentes conférences et ne seront pas repris dans la présente publication [1,2,3]. La figure 2 montre quelques exemples des blocs qui ont été développés.

Dans un tel processus constructif visant une mise en place automatique et préprogrammée des blocs de maçonnerie, la précision du système de guidage du drone est essentielle car de celle-ci dépend la façon dont devra être modifiée la géométrie de ces blocs.

Ce mémoire s'inscrit dans la mise au point d'un système de guidage précis, basé sur l'utilisation de trois paires de lasers dirigés respectivement vers 3 écrans (dont l'un peut-être le sol) qui entourent la zone de construction et dont les mesures permettent, par trigonométrie spatiale, de retrouver la position du drone dans l'espace (figure 3).

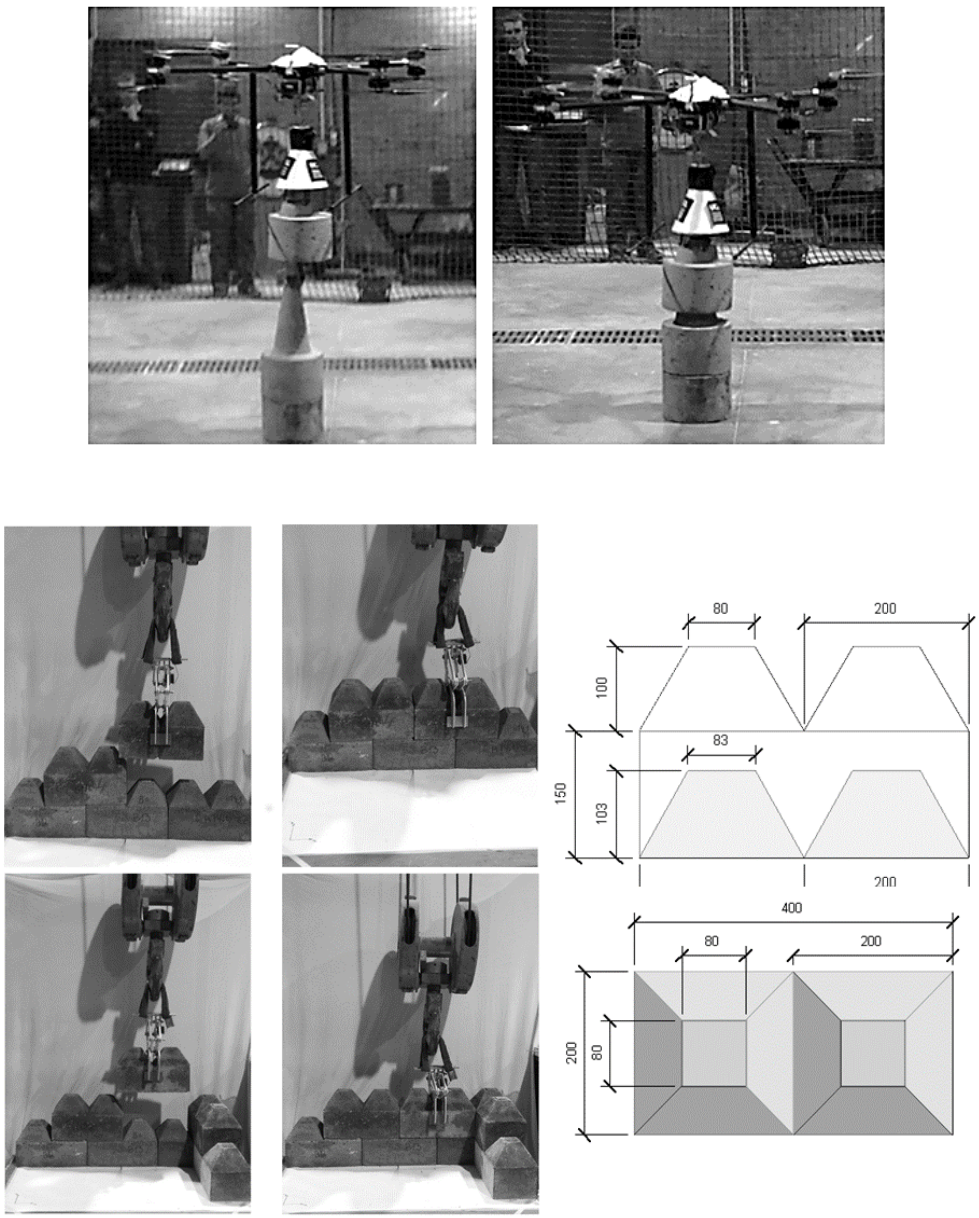


Figure 2 : quelques exemples des blocs de maçonnerie « drone compatible » développés à l'UCL dans le cadre du projet de recherche.

2. Objectif du travail

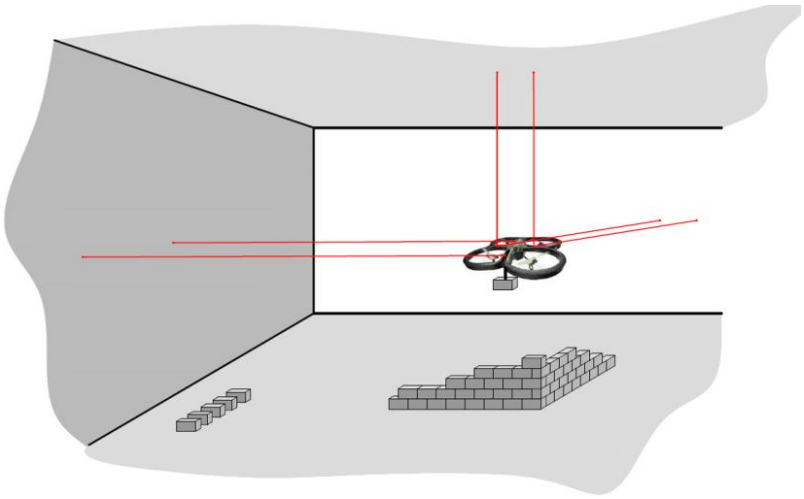


Figure 3 : méthode de localisation développée dans le cadre de ce travail de fin d'études. Les lignes représentent les faisceaux lasers

Les systèmes de localisation d'un drone se composent en général d'un IMU[1] souvent associé à un GPS ou un GPS-RTK[2]. Certains systèmes utilisent aussi d'autres capteurs comme des caméras embarquées ou des capteurs à ultrasons. Cependant, étant basé sur le GPS, ces systèmes ne fournissent une précision acceptable, bien qu'encore insuffisante, qu'à l'extérieur des bâtiments et en zone dégagée. Pour une localisation à l'intérieur d'un bâtiment, il existe des systèmes de type MOCAP, constitués d'un quadrillage de caméras (6 à 12) permettant d'obtenir une précision de l'ordre du centimètre, mais pour des prix importants avoisinant les 120k€.

Dans le cadre de ce mémoire, un système particulier, basé sur l'utilisation de capteurs lasers, a été développé. Trois paires de lasers embarqués visant des murs de référence ont été installés sur le drone. Notons que le désavantage de cette méthode est d'une part de nécessiter des écrans et, d'autre part, d'être tributaire de l'angle yaw, c'est-à-dire de la rotation du drone sur lui-même dans le plan horizontal. En effet, si l'un des lasers ne pointe plus vers son écran de référence, la valeur qu'il

[1] IMU *: Inertial Measurement Unit* permet de déterminer l'état angulaire (les angles de rotations autour des axes X, Y et Z) du drone via un magnétomètre, un accéléromètre, un gyroscope avec ou sans redondances.

[2] GPS-RTK : *Real Time Kinematics* qui permet théoriquement, sur base des mêmes signaux GPS, d'obtenir une précision de (20+1ppm) mm en ajoutant une base (= une borne) précisément localisée, plus ou moins loin du lieu de travail. Remarque : 1 ppm = 1 mm d'erreur par km d'éloignement de la base.

renverra ne sera plus pertinente. Par contre, le système est conceptuellement simple, peut s'utiliser aussi bien à l'extérieur qu'à l'intérieur, est peu coûteux, et fournit une précision meilleure que 5 cm, que des développements futurs pourront probablement porter à 1 cm. L'objectif du mémoire était :

- De maîtriser l'acquisition des données provenant des capteurs lasers ;
- De caractériser la sensibilité de ce système aux variations de température et d'angle entre le rayon et l'écran ;
- D'équiper un drone avec les capteurs lasers;
- De développer le système de contrôle et de stabilisation du drone sur base des mesures effectuées par les capteurs lasers

3. Matériel utilisé

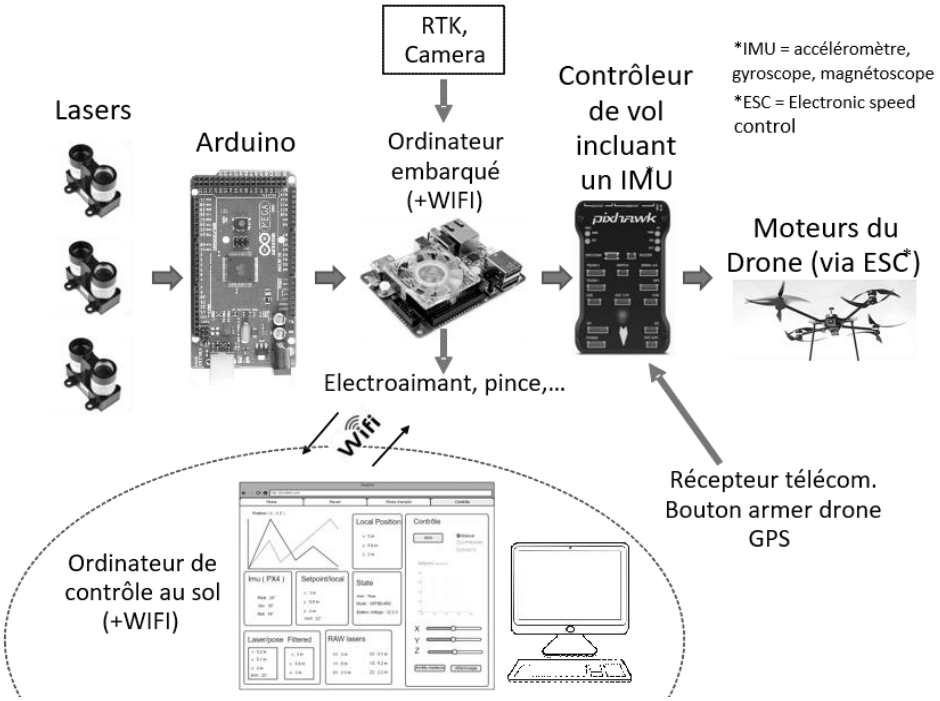


Figure 4 : schéma de principe général des connections hardware

3.1. Les capteurs lasers

Trois alternatives « grand public » de télémètres lasers existent à des prix acceptables. Teraranger, Leddartech et LidarLite sont trois concurrents, et le choix s'est porté sur le LidarLite v2 (figure 5, à gauche), car, pour un prix identique et des caractéristiques similaires, il est plus simple d'utilisation grâce à son port de communication (I^2C) permettant d'en interfacer plusieurs en série.

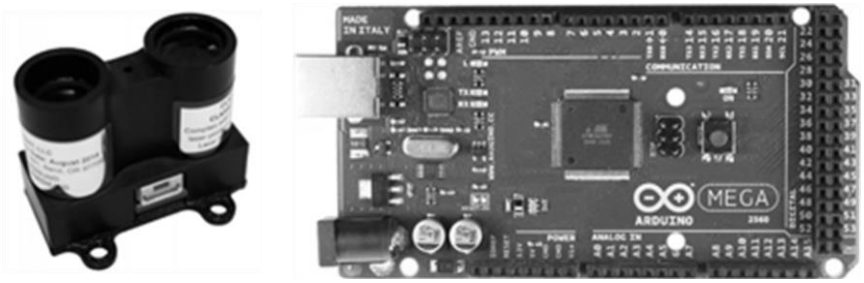


Figure 5 : à gauche, Laser LidarLite v2.
A droite, Microcontrôleur Arduino

Les caractéristiques du LidarLite v2 sont les suivantes :

- Portée : jusqu'à 50 mètres ;
- Jusqu'à 500 mesures par seconde ;
- Prix : 125$;
- Précision : 2.5 cm ;
- Communication : I²C.

3.2. Le microcontrôleur Arduino

Afin de réaliser l'acquisition des lasers, un microcontrôleur est nécessaire (figure 5, droite). Il a deux avantages face à un ordinateur « standard », qui, par définition est multitâches, et ne donne donc pas la priorité aux mesures. En effet, l'Arduino n'effectue que le contrôle des mesures des lasers qui peuvent donc s'effectuer à des intervalles de temps très précis. Le second avantage est de rendre le système modulable et de pouvoir ainsi utiliser les lasers directement sans programmation de « bas niveau ».

Le choix s'est porté pour une Arduino Mega 2560 car il possède plus de GPIO[3] (4 GPIO par laser soit 24 GPIO au minimum) et de mémoire programme afin de pouvoir utiliser plusieurs librairies sans se soucier de l'augmentation de la taille du programme.

3.3. L'ordinateur de bord

L'ordinateur est le cerveau du drone. Il permet d'organiser et connecter tous les autres modules entre eux, de gérer les communications ainsi que de réaliser des calculs comme par exemple du traitement d'images ou la gestion de l'algorithme trigonométrique de transformation des mesures lasers en position spatiale. De nombreuses solutions commerciales existent, dont les plus connues sont les RaspberryPi et les Odroid (figure 6).

[3] GPIO : General Purpose Input/Output donne accès à des entrées sorties numériques ou analogiques permettant de communiquer avec d'autres périphériques ou instruments.

Figure 6 : de gauche à droite RaspberryPi 3, Odroid XU4 et PixHawk [4]

Le choix s'est porté sur l'Odroid-XU4 de chez Hardkernel, entreprise sud-coréenne, dont les designs sont beaucoup plus performants que ceux de Raspberry-Pi. En effet, l'Odroid XU4 incorpore un processeur 4 fois plus rapide que ses concurrents (Equivalent au Galaxy Samsung S5) et comporte une mémoire eMMC allant jusqu'à 120mo/s contre une carte SD pour la RaspberryPi allant jusqu'à 20 mo/s. Le tout pour 70$, soit seulement le double d'une RaspberryPi.

3.4. Le contrôleur de vol PixHawk

Le contrôleur de vol utilisé est le PixHawk [4] (figure 6, droite). Contrairement aux nombreux autres contrôleurs de vol disponibles sur le marché, il a la particularité d'être Open-Source et Open-Hardware. Il est donc possible d'aller modifier le code soi-même. Ci-dessous, yaw correspond à l'angle de rotation du drone autour de l'axe Z.

Le contrôleur de vol reçoit :

- Par l'odinateur embarqué, la position (x,y,z,yaw) du drone, calculée à partir des mesures lasers (figure 9) ;
- Une consigne de position spatiale (x,y,z,yaw) depuis l'ordinateur situé au sol (via le WIFI) ;
- Les mesures (gyroscope, compas, accéléromètre, baromètre) de la centrale inertielle (IMU) située dans le PixHawk ;
- En cas d'urgence, une consigne provenant de la télécommande (via un opérateur).

Le contrôleur de vol effectue :

- Le calcul de l'état angulaire du drone (roll, pitch, yaw) à partir des mesures de l'IMU ;

- La régulation de la puissance à appliquer aux moteurs pour, d'une part, assurer un vol stable et, d'autre part, répondre à la consigne donnée par l'ordinateur au sol.

4. Le système de localisation par lasers

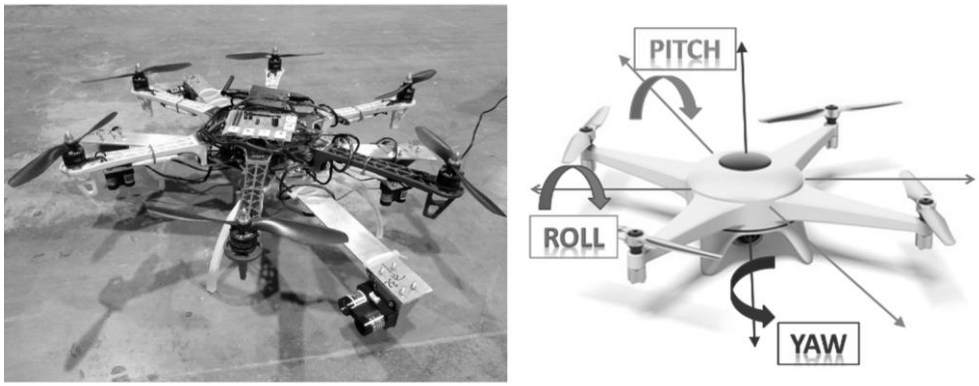


Figure 7 : à gauche, le drone équipé de ses capteurs lasers.
A droite, illustration des angles Pitch, Roll et Yaw.

Le système de localisation comprend :

- L'acquisition des lasers ;
- L'algorithme trigonométrique ;
- Le filtre de sortie.

Le système de localisation qui a été développé est basé sur des capteurs lasers de type LidarLite. Toutefois, grâce au système ROS (voir §5) installé sur l'ordinateur de bord, si un autre système de localisation est développé (RTK, MOCAP, …), ou si d'autres types de capteurs lasers étaient utilisés, il suffirait de remplacer le programme s'occupant de l'acquisition, sans que le reste du programme ne doive être adapté.

4.1. Acquisition des lasers

L'acquisition se réalise via un microcontrôleur Arduino, sur lequel sont connectées trois paires de lasers pointant chacune vers une surface de référence (l'une d'elles étant le plafond ou le sol). Le driver fourni afin de réaliser les mesures des lasers n'étant pas assez robuste, un nouveau driver a été développé par les auteurs [5] de façon à s'assurer qu'il n'y ait pas de problèmes en vol, ainsi que pour augmenter le nombre de mesures par secondes.

4.2. Algorithme trigonométrique

Les inconnues sont au nombre de 6 : les 3 coordonnées (x,y,z) du drone et ses 3 positions angulaires (roll,pitch,yaw). L'IMU fournit avec grande précision les valeurs du pitch et du roll qui sont une image de son horizontalité. Le yaw, par contre, est donné par l'IMU avec beaucoup moins de précision car le compas est peu précis en intérieur, et de plus sensible à la présence de pièces métalliques avoisinantes et aux champs magnétiques créés notamment par les moteurs ou l'électroaimant qui sert à suspendre les charges. On dispose donc d'une redondance double dans les mesures puisque les lasers fournissent 6 mesures en plus du pitch et du roll fournis par l'IMU. L'algorithme développé intègre la position des 3 paires de lasers sur le drone et est schématisé à la figure 9. Cet algorithme, basé sur une formulation trigonométrique relativement complexe, a été validé par un modèle Geogebra (figure 8).

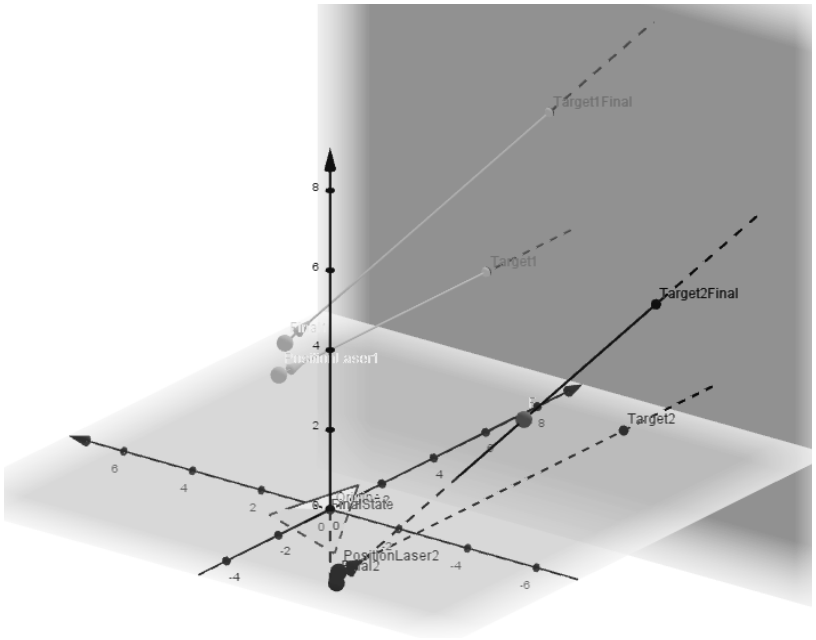
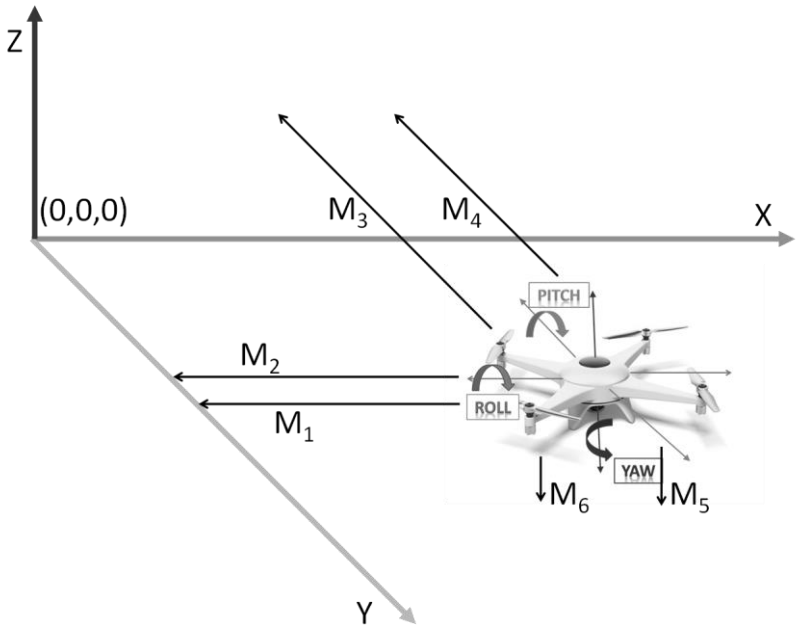


Figure 8 : validation de l'algorithme par Geogebra

L'algorithme permet d'obtenir une précision angulaire pour le yaw qui dépend de la distance relative entre les deux lasers M_1 et M_2. Dans le cadre de ce travail, les paires de lasers ont été espacées de 40 cm afin d'obtenir une précision angulaire sur le yaw suffisante (< 5°), malgré l'imprécision des lasers estimée à 25 mm (voir §7.1).



M_1, M_2 : distances mesurées avec lasers selon l'axe x
M_3, M_4 : distances mesurées avec lasers selon l'axe y
M_5, M_6 : distances mesurées avec lasers selon l'axe z (vertical)
$roll_{IMU}$, $pitch_{IMU}$: donnés avec précision par IMU (Pixhawk)
4 inconnues à déterminer : yaw, x,y,z
2 redondances au total

↓

Equations trigonométriques disponibles :

$M_1=f(x,roll_{IMU},pitch_{IMU},yaw_x)$
$M_2=f(x,roll_{IMU},pitch_{IMU},yaw_x)$
→ Calcul de yaw_x (et x)

$M_3=f(y,roll_{IMU},pitch_{IMU},yaw_y)$
$M_4=f(y,roll_{IMU},pitch_{IMU},yaw_y)$
→ Calcul de yaw_y (et y)

$yaw_{moy}=(yaw_x + yaw_y)/2$

↓

Equations trigonométriques disponibles :

$M_1=f(x_1,roll_{IMU}pitch_{IMU},yaw_{moy})$
$M_2=f(x_2,roll_{IMU},pitch_{IMU},yaw_{moy})$
→ $x=(x_1 + x_2)/2$

$M_3=f(y_1,roll_{IMU},pitch_{IMU},yaw_{moy})$
$M_4=f(y_2,roll_{IMU},pitch_{IMU},yaw_{moy}$
→ $y=(y_1 + y_2)/2$

$M_5=f(z_1,roll_{IMU},pitch_{IMU},yaw_{moy})$
$M_6=f(z_2,roll_{IMU},pitch_{IMU},yaw_{moy})$
→ $z=(z_1 + z_2)/2$

Figure 9 : algorithme de calcul de la position (x,y,z,yaw) à partir des 6 mesures lasers et des valeurs de (roll,pitch) fournies par l'IMU.

4.3. Filtre de sortie

Il est nécessaire de disposer d'une dizaine de positions par seconde pour que le drone puisse adapter la puissance des moteurs correctement. Par ailleurs chaque laser fournit 100 mesures par seconde, avec pour certaines une erreur dont l'amplitude peut atteindre 25 mm, même en l'absence de mouvement du drone. Il est donc nécessaire de filtrer ces mesures pour obtenir une valeur plus stable et correcte. Un premier filtre de type IIR (Infinite Impulse Response) est appliqué directement sur les mesures de chaque capteur. Ce filtre consiste à fournir une valeur basée sur les précédentes mesures, ce qui lisse les mesures. Ensuite, une pondération de trois filtres est utilisée ([6], figure 10). Le premier filtre réalise une prédiction de la position par l'intégration de l'accélération et de la vitesse du drone. Le deuxième est une moyenne sur 10 mesures et le troisième est un passthrough[4]. La pondération est calculée en fonction du type de vol et de l'erreur de mesure des lasers (constante). Si le vol correspond à de petits déplacements (phase de stabilisation du drone) le deuxième gain sera augmenté. Pour un vol correspondant à de grands déplacements (dans l'espace de vol), on préfèrera augmenter le gain du 1er filtre.

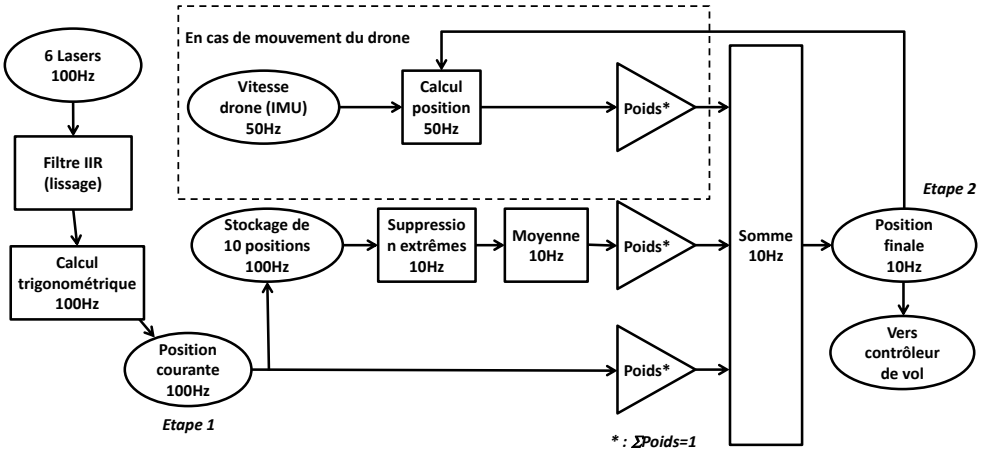


Figure 10 : filtre de sortie

5. Système ROS installé sur l'ordinateur embarqué

Dans ce système, chaque exécutable appelé nœud (node), peut communiquer avec tous les autres nœuds à travers un « topic » qui, en quelque sorte, représente le lieu d'échange entre les informations (figure 11). Un topic porte un nom explicite, par exemple le nœud d'acquisition des lasers publie au topic « /lasers/raw » les me-

[4] Un passthrough fait passer l'entrée à la sortie sans y appliquer de traitement.

sures brutes des lasers (=nœud « publisher »). Le nœud de l'algorithme trigonométrique, quant à lui, souscrit au topic « /lasers/raw » et reçoit les mesures des lasers dès qu'elles sont disponibles (= nœud « subscriber »).

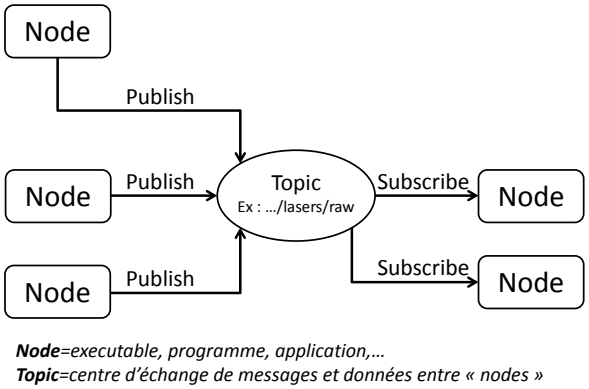


Figure 11 : principe des communications ROS

De plus, le système ROS fournit une multitude d'outils de robotique implémentant de nombreux algorithmes tels que du SLAM, de la Vision odometry, des transformations de systèmes de coordonnées et bien d'autres, qui sont préprogrammés [7]. Dans le cadre de ce travail, un nœud permettant la connexion entre des capteurs lasers et le calcul de la position du drone a été développé et mis à disposition de la communauté ROS.

Toutes ces librairies et drivers accélèrent considérablement le développement software. C'est pourquoi ROS a été choisi dans le cadre de ce mémoire comme plateforme de développement. ROS est installé à la fois sur l'ordinateur embarqué et sur l'ordinateur au sol.

La figure 12 schématise l'architecture software du système ROS embarqué, telle qu'elle a été développée dans le cadre de ce mémoire.

Les définitions suivantes sont utilisées dans cette publication:

- Tâche : une action à réaliser par le drone comme un trajet, une gestion de la pince de manutention, un décollage, un atterrissage, un vol stationaire, etc ;
- Mission : liste de tâches successives à réaliser.

La partie contrôle du drone est une suite de programmes (noeuds) utilisés afin de gérer la mission du drone. L'architecture choisie se compose de 5 nœuds : acquisition, algorithme trigonométrique, télémétrie, tâche et commande (figure 12).

Nœud d'acquisition : réalise l'acquisition des lasers et publie les valeurs sur le topic /Mesures.

Nœud d'algorithme trigonométrique : calcule la position (x,y,z,yaw) du drone à partir des lasers et des mesures fournies par l'IMU et la publie sur le topic /Position.

Nœud de Télémétrie : publie toutes les informations du drone vers une interface utilisateur web et envoie les instructions de l'utilisateur (=la mission) au nœud de tâches via le topic /Mission.

Nœud de tâches : gère la mission « ajouter ou retirer des tâches », vérifie si la tâche courante est réalisée et si oui, envoie la tâche suivante au nœud de commande.

Nœud de commande : reçoit la position du drone ainsi que la tâche à réaliser et communique ces informations au PixHawk.

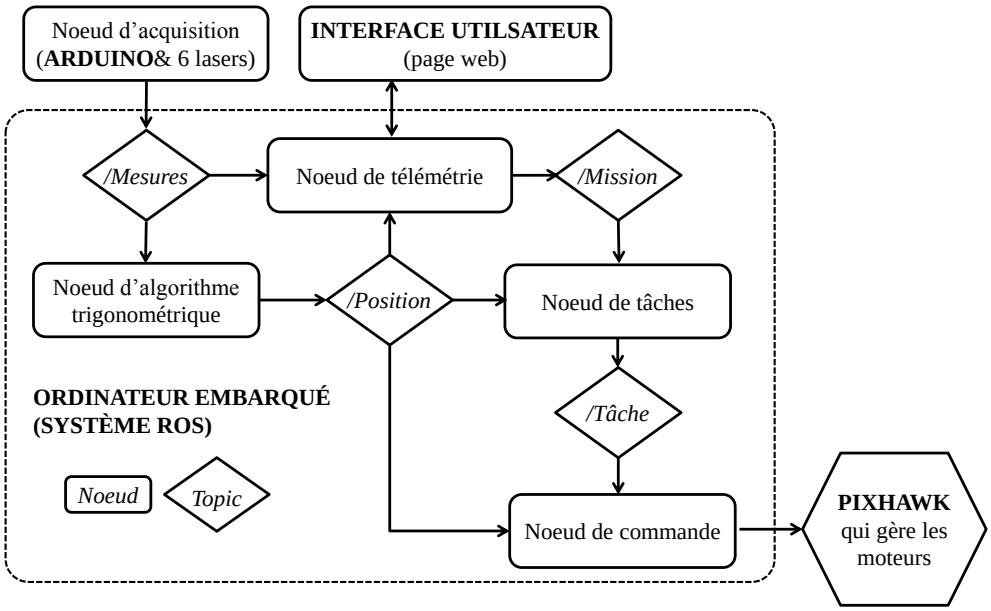


Figure 12 : architecture software globale

6. Interface

L'interface graphique est réalisée en HTML, Javascript et CSS (technologie web). L'avantage de ces langages de programmation est de rendre accessible l'interface à partir de n'importe quelle plateforme (ordinateur, smartphone, tablette) sans la moindre installation, et ceci quel que soit le système d'exploitation. La communica-

tion entre l'interface et le nœud de télémétrie est réalisée via une librairie ROS qui envoie les informations via un websocket[5] avec des messages JSON[6].

L'interface se veut simple mais complète. En plus de l'affichage brut des mesures pertinentes (état de la batterie, position du drone, consigne, roll, pitch, yaw, …), l'utilisateur a accès à plusieurs graphiques comme celui du contrôle en temps réel (figure 13). Celui-ci fournit la position du drone (le rond), la consigne (la croix, représentant l'endroit où le drone doit se placer) et la position de la souris (un carré). Si l'utilisateur clique sur le graphique, le drone reçoit comme consigne de se déplacer à cet endroit. Les autres graphes permettent de contrôler l'altitude ainsi que d'observer la variation de la position (x,y,z) en fonction du temps.

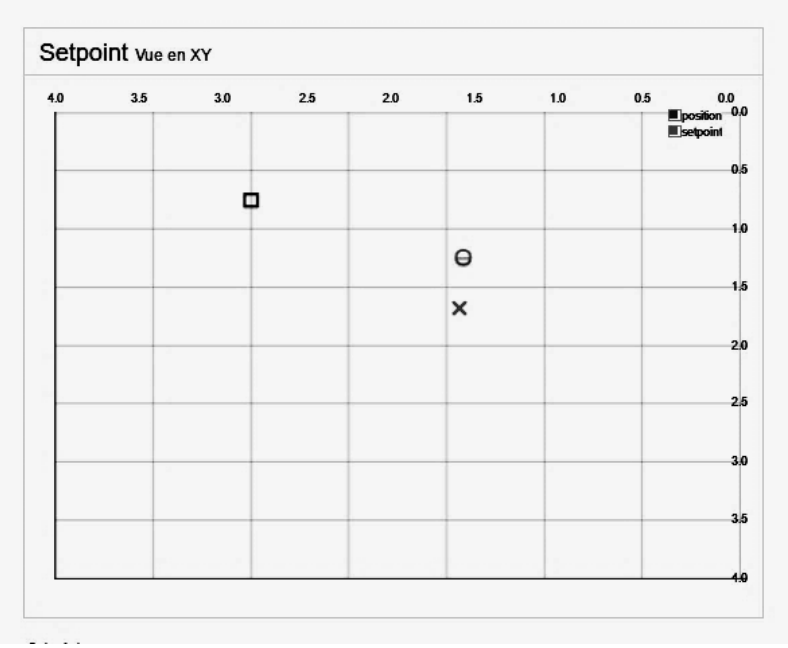


Figure 13 : contrôle de position en temps réel via l'interface graphique

7. Performances du système développé

7.1. Caractéristiques des lasers

Les mesures fournies par les lasers ont été comparées avec celles d'un étalon (un télémètre laser précis au millimètre). Pratiquement, 4 capteurs lasers ont été placés à côté avec l'étalon, en face d'un écran mobile, et une mesure (résultant d'une

[5] Websocket est un protocole standard du web visant à créer un canal de communication entre deux applications (dans ce cas-ci, une interface et un serveur)
[6] JSON est un format de données utilisé pour représenter des objets en Javascript

moyenne sur 100 mesures prises pendant 1 seconde) sur chaque capteur a été effectuée pour différentes distances allant de 50 cm à 10 m. L'amplitude de l'écart entre la valeur étalon et la moins bonne des mesures des 4 capteurs ne dépasse jamais 25 mm. Chaque capteur présente un offset constant qui lui est propre et que l'on intègre dans le traitement de la mesure.

Par ailleurs, le filtrage des mesures tel que représenté en figure 10 permet de réduire cette amplitude de 25 mm à 10 mm, comme discuté en 7.2.

7.2. Performances du système de localisation (étape 1, figure 10)

Le drone a été placé immobile sur le sol, avec un angle yaw nul (lasers perpendiculaires aux 2 écrans verticaux) pendant 90 secondes à des distances respectives de chaque écran de 186.2 cm (axe x) et 182.2 cm (axe y). Ces distances ont été mesurées à l'aide de l'étalon. 90 secondes correspondent à 9000 mesures brutes de distance (100 mesures par seconde) donnant donc lieu à 9000 positions calculées (voir figure 10, étape 1). Il s'agit de déterminer la qualité du système de localisation après calcul trigonométrique. Les résultats sont similaires pour chacun des axes, dès lors seuls les détails pour l'axe x et l'angle yaw seront considérés.

Il s'agit ici des résultats obtenus à l'étape 1 (figure 10), avant le filtrage. Les chiffres ci-dessous correspondent au jeu de 9000 valeurs :

- Moyenne de l'erreur de la position en x calculée : $| moy(x) - 186{,}2 | = 0.6\ cm$
- Écart type de la position en x calculée : $\sigma = 0.82\ cm$
- Moyenne de l'erreur de l'angle yaw calculé : $| moy(yaw) - 0 | = 0.3°$
- Écart type de l'angle yaw calculé*:* $\sigma = 1{,}0°$
- Ecart maximum par rapport à la valeur de 186,2 cm : 2.93*cm*

7.3. Performance du contrôle de position (étape 2, figure 10)

On considère ici que la consigne est constante, c'est-à-dire un vol stationnaire. Contrairement à l'étape 1 pour laquelle un étalon adapté était disponible, aucun système « étalon 3D » n'était disponible au moment du mémoire afin d'avoir une référence exacte de position du drone. D'où l'importance d'avoir caractérisé la précision du système de localisation afin de connaître l'origine des imprécisions. Notons que le drone était trop fortement chargé par rapport à la capacité de ses moteurs à cause du matériel de développement embarqué, rendant le travail du contrôleur de vol plus difficile, ce qui a été une source d'imprécision dans les résultats annoncés ci-dessous. Il est probable qu'une meilleure précision aurait été obtenue avec un drone de plus grosse capacité portante.

La procédure a été la suivante :

- Placement du drone par terre, et, à l'aide du télémètre laser ayant servi d'étalon aux §7.1 et §7.2 pour mesurer la distance entre le point de

référence du drone et chacun des 2 écrans verticaux. Dans le cas présent, ces deux distances étaient de x= 193.4 cm et y=157.7 cm;
- Décollage du drone avec comme consigne de garder la même position (x,y) ;
- Acquisition des mesures pendant 100 secondes (10 Hz en sortie du filtre, donc 1000 mesures finales). Ces mesures sont illustrées à la figure 14;
- Atterrisage.

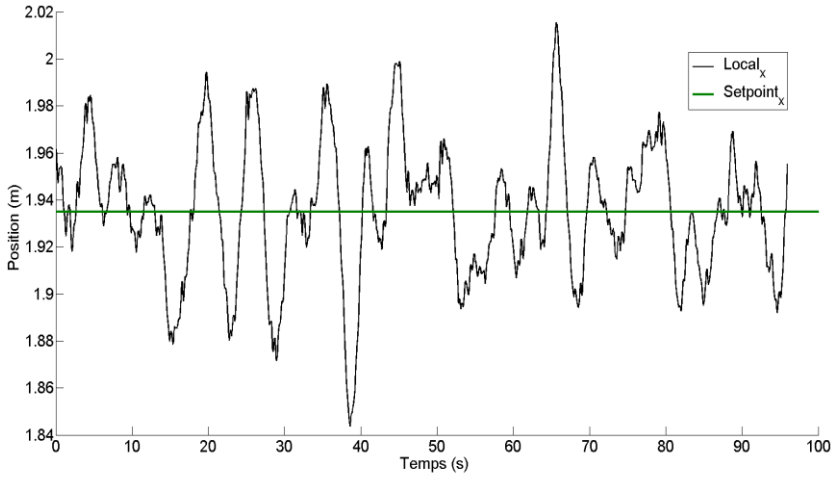


Figure 14 : Évolution de la position en X en fonction du temps pour un vol stationnaire. La ligne horizontale est le setpoint, la consigne.

Les mesures effectuées sur 100 secondes, visibles sur la figure 14, permettent de calculer les données suivantes :

- Moyenne de l'erreur de la position en x calculée : $| moy(x) - 193{,}4 | = 0.2\ cm$
- Écart type de la position en x calculée : $\sigma = 2.8\ cm$
- Moyenne de l'erreur de l'angle yaw calculé : $| moy(yaw) - 0 | = 1.5°$
- Écart type de l'angle yaw calculé: $\sigma = 2{,}1°$
- Ecart maximum en X par rapport à la valeur de 193,4 cm : 8.1*cm*
- Erreurs en x et en yaw maximales furent respectivement de 5,6 cm et 5,7°

Les valeurs données ci-dessus montrent que, bien qu'entachée d'une certaine imprécision, la moyenne de l'erreur de la position en x calculée est très bonne (0,2 cm). La régulation est donc assez performante.

Toutefois, comme on peut l'observer à la figure 14, bien que le PixHawk régule bien sa position XYZ en moyenne, sa position instantanée oscille fortement, ce qui est dû au surpoids du drone qui réduit grandement son temps de réaction.

8. Conclusion

Le travail réalisé a nécessité de nombreux développements software et hardware, dans un environnement parfois totalement inconnu comme celui des drones. Le résultat final est appréciable puisque le système développé est fonctionnel et va déboucher sur des développements complémentaires dans l'équipe de l'UCL active sur ce projet de construction avec les drones. La précision et les limitations du système ont été mis en évidence : imprécision dues aux capteurs lasers eux-mêmes, combinée à l'imprécision provenant de la régulation du vol. A ce sujet, une amélioration des résultats, déjà appréciables tels quels, devrait être obtenue en utilisant un drone avec plus de puissance qui permettrait une meilleure régulation avec les charges embarquées importantes.

Comme autres pistes d'amélioration, il semble opportun de combiner aux lasers plusieurs technologies comme le GPS (RTK), une caméra [8], ou d'autres systèmes plus particuliers comme une station totale de géomètre, probablement plus précise.

9. Références

[1] PIERRE LATTEUR, SEBASTIEN GOESSENS, MILAN RENIERS, ZHAO MA, CAITLIN MUELLER, *« Masonry Construction with drones »*, International Association for Shell and Spatial Structures, Tokyo, 26-30 septembre 2016.

[2] GOESSENS S., MUELLER C., LATTEUR P., « Vers une robonumérisation de la construction », RUGC, Liège, 25-26-27 mai 2016.

[3] PIERRE LATTEUR, SÉBASTIEN GOESSENS, JEAN-SÉBASTIEN BRETON, JUSTIN LEPLAT, ZHAO MA, CAITLIN MUELLER, *"Drone-Based Additive Manufacturing of Architectural Structures"*, International Association for Shell and Spatial Structures, Amsterdam, 17-20 août 2015

[4] LORENZ MEIER, PETRI TANSKANEN, FRIEDRICH FRAUNDORFER AND MARC POLLEFEYS, *« PIXHAWK: A System for Autonomous Flight using Onboard Computer Vision »*
Adresse URL : http://cvg.ethz.ch/publications/pdf/pub_73260.pdf.

[5] ALEXIS PAQUES, (consulté en décembre 2015) : *LidarEnchanced*
Adresse URL : https://github.com/AlexisTM/LIDAREnhanced.

[6] S LYNEN AND M ACHTELIK AND S WEISS AND M CHLI AND R SIEGWART, *« A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation »*, Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS), 2013

[7] ROS APIS, (consulté en janvier 2016) : *List of ROS APIs*
Adresse URL : http://wiki.ros.org/APIs.

[8] FORSTER, CHRISTIAN AND PIZZOLI, MATIA AND SCARAMUZZA, DAVIDE, (2014), *SVO: Fast Semi-Direct Monocular Visual Odometry*
Adresse URL: https://github.com/uzh-rpg/rpg_svo.