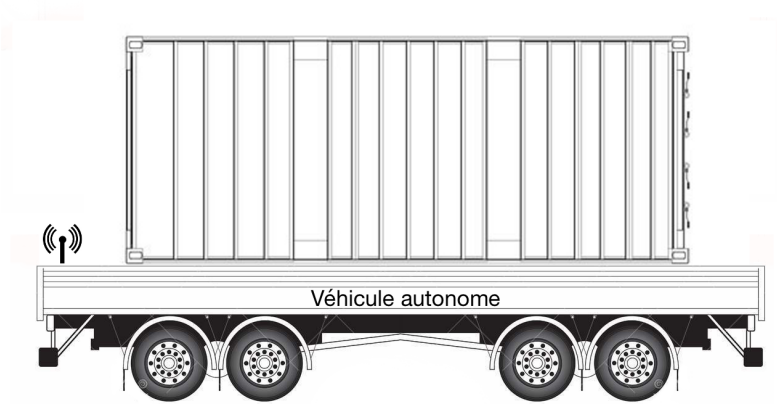
POLYTECH LILLE

Élève : Appercé Pierre Tuteur : Merzouki Rochdi

Aide à la navigation d'un véhicule autonome



Rapport : Projet de fin d'études
Date : 24 fevrier 2015

Table des matières

1 Remerciements

Je tiens à remercier toutes les personnes qui ont contribué à ce projet, pour leur aide, leurs conseils et le temps qu'ils m'ont accordé :

Tout d'abord, j'aimerai remercier Mr MERZOUKI, mon tuteur de PFE. Il a imaginé cette étude de faisabilité pour trouver une solution à un problème rencontré sur le projet InTraDE mais aussi pour répondre à mon envie de travailler sur un projet comportant un drone.

Je tiens à remercier tout particulièrement Mr COLLEN, qui grâce à son expérience m'a guidé lors de mon travail lorsque j'ai rencontré des problèmes.

Je souhaite également remercier

- Mr POLLART, pour son soutien technique et sa disponibilité pour venir assister aux vols d'essai.

- Mr FLAMAND, pour son soutien et ses précieux conseils techniques pour la réalisation et l'assemblage des différentes parties de ce projet.

Étant en monôme, il est difficile de resoudre de certaines difficultés techniques ou d'avoir un esprit critique sur son travail. J'aimerai donc remercier Valentin Vergez, élève en IMA5 d'avoir pris du temps pour discuter de mon projet et d'avoir apporté un regard neuf sur mon travail.

2 Projet de fin d'études

Le projet de fin d'études (PFE) est un projet de 5ème année d'école d'ingénieurs qui s'étale sur la période septembre-février. Il a pour objectif principal la transposition des connaissances acquises dans le milieu scolaire en compétences dans un environnement plus professionnel.
L'élève doit faire preuve d'autonomie et de responsabilité. Ce projet lui permettra d'exprimer son savoir-faire, mais il aura aussi pour but de l'encourager à être autodidacte, que ce soit pour des nouveaux composants ou des langages techniques.
Ce PFE concernera l'aide à la navigation d'un véhicule autonome le RobuTAINeR par un drone.

3 Contexte

3.1 Description du projet

Le RobuTAINeR est un véhicule poids lourd, omnidirectionnel, sur-actionné et électrique qui a été développé dans le cadre du projet InTraDE [1] pour le transport du fret à l'intérieur d'espaces confinés portuaires. Il est autonome et s'adapte à l'environnement d'exploitation. Pour assurer une autonomie de navigation précise, le véhicule dispose d'un système GPS pour son positionnement. Malheureusement, dans un environnement portuaire, le signal GPS [2] est souvent perturbé par la présence de conteneurs, de bâtiments, d' arbres,..etc. Dans le cadre de ce projet, on se propose de tester la faisabilité de corriger la navigation du véhicule autonome par un drone volant à proximité. Le drone va échanger avec le véhicule à chaque fois que ce dernier diverge de sa trajectoire, soit après une perte de l'information GPS, soit pour une autre cause.

1. Intelligent Transportation for Dynamic Environment
2. Global Positioning System

3.2 RobuTAINeR

3.2.1 Présentation / Mission

Il s'agit du premier véhicule robotisé capable de naviguer de manière autonome et sûre en se rendant d'un point à un autre grâce à un itinéraire pré-établi. En programmant ses déplacements, le robot va augmenter les cadences et aider les acteurs portuaires à gagner en productivité. Le RobuTainer s'adapte à l'environnement existant, tout en limitant les risques de dysfonctionnement grâce à ses 8 roues motorisées (soit 4 pour la traction et 4 pour la direction).
Il embarque des capteurs laser qui détectent les obstacles ou les piétons, une centrale inertielle couplée à un récepteur GPS pour se repérer dans l'espace, un émetteur et un récepteur GSM [3] reliés à l'ordinateur de bord. Le RobuTAINeR est piloté automatiquement ou manuellement par le docker. Il fait l'objet d'un suivi à distance par un serveur dédié qui comporte également des fonctions de simulation pour programmer ses missions. L'objectif étant de doper la compétitivité des ports du nord-ouest de l'Europe situés le long du littoral, de l'Irlande jusqu'au Pays-Bas. En effet c'est une région où les acteurs portuaires souffrent d'un manque d'espace, d'une congestion du trafic et de la pollution.

FIGURE 1 – Photo du RobuTAINeR en action dans un port

3. Global System for Mobile

3.2.2 Problème

Lorsque le RobuTAINeR traverse le port, il est amené à circuler entre des murs de conteneurs. La hauteur et la composition métallique de ces murs empêchent une bonne réception du signal GPS. Or ce signal GPS est capital pour que le RobuTAINeR puisse se déplacer avec précision. Actuellement, seuls des lasers permettent au RobuTAINeR de continuer sa mission. Mais au-delà d'une vingtaine de secondes, la trajectoire devient trop approximative.

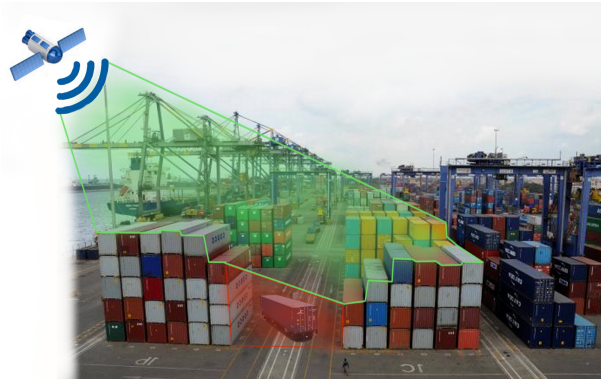


FIGURE 2 – Représentation de la perte du signal GPS

3.3 Drone

Le drone est le moyen qui a été choisi pour aider le RobuTAINeR à naviguer dans le port.

3.3.1 Mission

Le drone aura pour principale mission de relayer le signal GPS venant des satellites au RobuTAINeR. Avant de partir en mission, le drone recevra le même parcours que le RobuTAINeR. Au départ, le drone décollera et viendra se placer au dessus du RobuTAINeR. Puis, il accompagnera le RobuTAINeR tout au long de son parcours. Lorsque celui-ci commencera à perdre le signal GPS, alors le drone prendra le relai. Pour cela, il lui faudra repérer exactement la position du RobuTAINeR afin de calculer la position de celui-ci par rapport à la sienne et dans un second temps de lui envoyer ses coordonnés GPS.

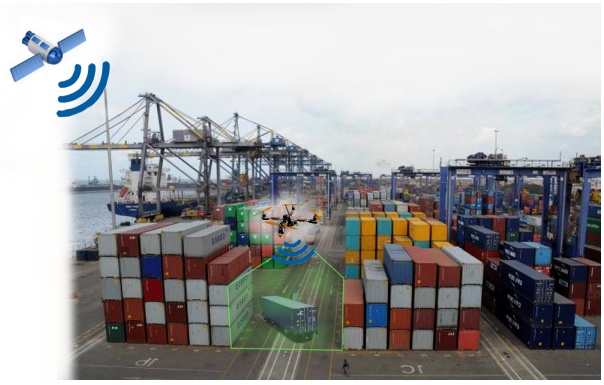


FIGURE 3 – Représentation du relai du signal GPS par le drone

4 État de l'art et positionnement

Un véhicule est dit autonome lorsque dans un environnement donné, il est capable de ce déplacer en évitant les obstacles et sans l'intervention d'une personne. Je vais dresser un rapide état de l'art sur la navigation autonome. Puis nous verrons dans une seconde partie, en quoi ce projet est innovent dans le monde de la navigation autonome.

4.1 État de l'art

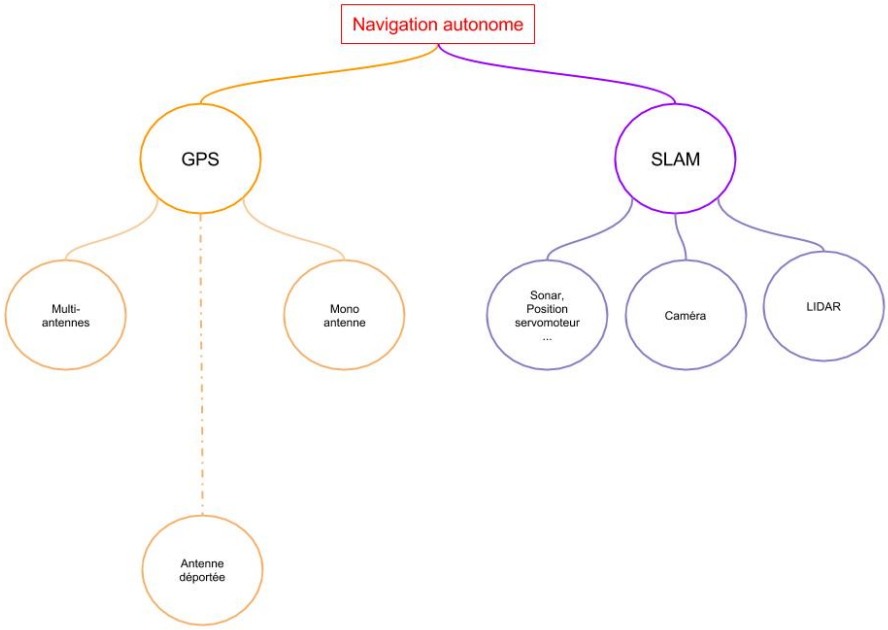


FIGURE 4 – État de l'art : La navigation autonome

Signification des différents termes utilisés :
SLAM : "Simultaneous Localization And Mapping" qui signifie Cartographie et localisation simultanées
LIDAR : "Laser Detection And Ranging", mesure de distance à base de laser

4.2 Positionnement

La partie innovante de ce projet se trouve dans le fait qu'une partie des capteurs qui servent à la navigation seront déportés sur un autre véhicule. On trouve généralement un ou plusieurs capteurs (GPS par exemple) sur le véhicule pour assurer la navigation autonome. Dans le cas présent, le RobuTAINeR qui est un véhicule autonome recevra deux signaux pour se localiser. Le premier est un signal GPS, venant directement des satellites et le second sera les coordonnées GPS transmis par un autre véhicule, ici un drone. Le but étant d'assurer au minimum la transmission d'un des signaux pour que le RobuTAINeR puisse se déplacer en toute fluidité. Ce projet à pour but de démontrer l'intérêt de déporter une partie des capteurs sur un autre véhicule.

5 Cahier des charges

5.1 Conditions d'utilisation

Pour cette étude de faisabilité, l'environnement de fonctionnement sera très simplifié. C'est-à-dire que le drone ne sera utilisé que dans de bonnes conditions météorologiques et que l'espace dans lequel il se déplace sera dépourvu de tout obstacle.
Pour les conditions atmosphériques, le vent devra être faible pour que le drone puisse facilement suivre le RobuTAINeR et le temps devra être beau (en effet le drone n'est pas prévu pour effectuer un vol par temps pluvieux). Au niveau des obstacles, le port est un environnement dangereux pour le drone. La présence de câbles aériens, de murs de conteneurs, ainsi que de grues rendent le survol à basse altitude risqué.

5.2 Définition des besoins

Le drone devra être capable de recevoir un signal GPS et de l'interpréter. Ce qui veut dire que le drone devra emporter l'équipement nécessaire pour qu'à partir du signal GPS et d'autres capteurs, il puisse en déduire sa position (coordonnées GPS , altitude, orientation). L'autonomie requise pour le drone devra être d'une trentaine de minutes.

5.3 Spécifications techniques

Afin de ne pas perturber le comportement du drone et de ne pas réduire son autonomie, l'équipement nécessaire à l'accomplissement de sa mission sera séparé de son équipement d'origine. L'appareillage devra donc avoir sa propre source d'énergie, ses propres capteurs et une gestion des données séparée. L'ensemble devra également respecter l'autonomie demandée.
La communication entre le drone et le RobuTAINeR devra être de bonne qualité et résistante au milieu portuaire (émission et réception dans un environnement sujet aux perturbations). Le drone devra effectuer un vol stabilisé au dessus du RobuTAINeR à une altitude d'environ 30 mètres.

6 Schéma global de la solution retenue

6.1 Pour le RobuTAINeR

Pour ce projet, M. Coelen et moi allons installer un ordinateur de bord dans le RobuTAINeR. Il servira à la fois à mon projet et aussi à d'autres projets en cours. Nous équiperons cet ordinateur d'un module radio et d'une antenne GPS. Bien que le RobuTAINeR dispose déjà d'un système de localisation GPS, et comme ce projet n'est qu'une étude de faisabilité, nous allons greffer un autre module GPS pour ne pas perturber le premier. Le module radio permettra la communication entre le RobuTAINeR et le drone. Nous avons choisi un module radio et non un module wifi car la portée est plus grande et la résistance aux perturbations meilleure.

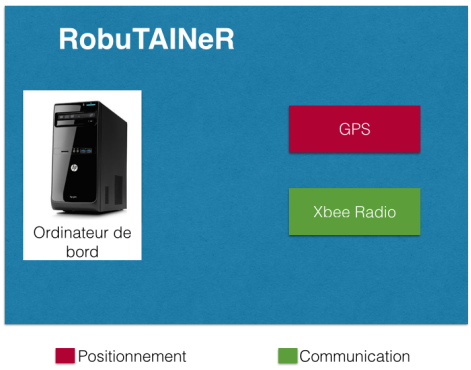


FIGURE 5 – Composants ajoutés au RobuTAINeR

6.2 Pour le drone

De son coté, le drone emportera une caméra pour pouvoir repérer le RobuTAINeR, un altimètre, un module GPS et une centrale inertielle qui lui permettront de connaitre avec exactitude sa position. Un module radio sera aussi fourni pour pouvoir échanger les données avec le RobuTAINeR. L'ensemble de ces composants sera géré par une Raspberry Pi et alimenté par une batterie [4]. Le drone qui a été retenu pour ce projet et ses caractéristiques se trouvent en annexe A.

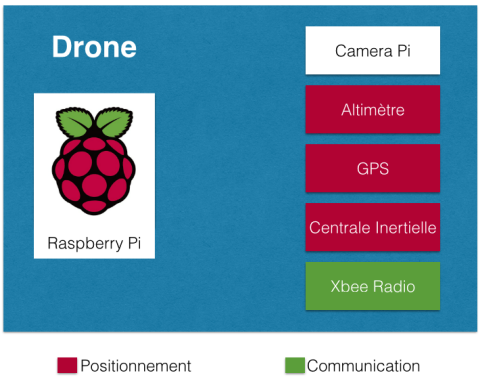


FIGURE 6 – Ensemble des composants ajoutés au drone

4. de même type que les batteries externes pour les téléphones portables

7 Réalisations

7.1 Matériel

7.1.1 Le Drone

Durant ce projet, plusieurs vols d'essais ont été réalisés. Dans un premier temps, je devais me familiariser avec les commandes de vol du drone. Apprendre à piloter correctement le drone est important pour pouvoir effectuer les tests dans de bonnes conditions et réaliser les démonstrations en toute sécurité. J'ai équipé ce drone de deux boitiers, un avec les différents composants et l'autre avec la batterie externe. Le détail de ces boitiers se trouve en partie 7.1.11 page 14. Pour plus de détails sur les caractéristiques de ce drone se référer à l'annexe A.

FIGURE 7 – Walkera QR x800

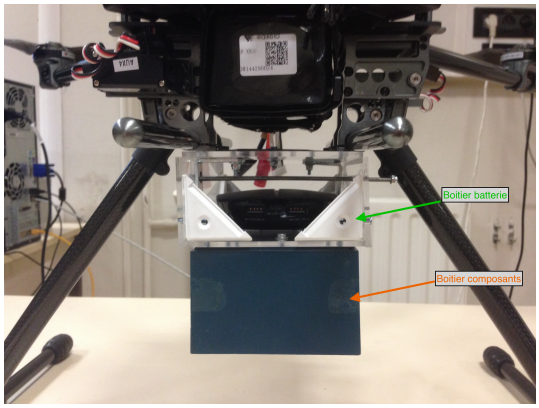


FIGURE 8 – Photo des boitiers installés sous le drone

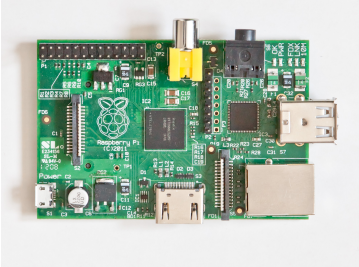
7.1.2 La télécommande du drone

La télécommande du drone n'est pas une tâche à laquelle on pense quand on fait le planning. Et pourtant, celle-ci demande du temps. En effet, les réglages et les commandes de vol sont complexes. Il faut compter un à deux jours de formation pour pouvoir utiliser correctement cette télécommande. Néanmoins, dans notre cas, l'utilisation reste basique. Effectivement, il suffit de placer le drone au-dessus du RobuTAINeR puis de le suivre en essayant de conserver le RobuTAINeR dans le champ de la caméra installée sur le drone. J'ai rédigé une documentation technique complète sur mon projet dans le but de faciliter sa reprise en main [a]. Toutes les commandes de bases et les différents conseils y sont répertoriés.

a. Attention, peu de fonctions de base de la télécommande, comme le vol stationnaire, sont activées. Si l'une de ces fonctions est activée par l'appui sur l'un des bouton de la télécommande, alors tous les moteurs s'arrêteront.

7.1.3 La Raspberry Pi

La Raspberry Pi peut être considérée comment l'ordinateur de bord. Elle gère l'ensemble des composants et permet l'exécution des programmes. D'un point de vu matériel, la Raspberry n'a pas beaucoup évolué. La connexion entre la Raspberry et les composants se fait maintenant à l'aide une nappe. Depuis début janvier, le boitier que l'on avait commandé n'est plus utilisé pour des raisons d'encombrement et de connexion. En effet la Raspberry vient désormais se loger dans le boitier composants. Il n'est pas nécessaire de venir la protéger une seconde fois avec un autre boitier. Cette solution n'est pourtant pas parfaite car la caméra se retrouve plus exposée.
Le câblage de la Raspberry avec les autres composants est disponible en annexe B.

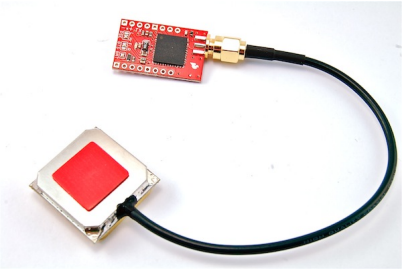


7.1.4 La caméra de la Raspberry

La caméra que nous avons installée est un module complémentaire de la Raspberry Pi, c'est-à-dire qu'il se branche directement sur la carte. Il suffit ensuite d'activer cette caméra par quelques lignes dans le terminal de la Raspberry Pi. La caméra permet de repérer une cible qui est placée sur le RobuTAINeR. Ce processus est expliqué dans la partie 7.2.3 page 17.

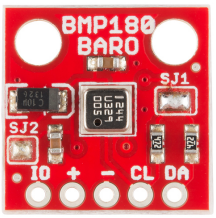
7.1.5 Le module GPS

Pour communiquer avec le module GPS, on utilise la liaison série de la Raspberry. Alors que ce module se trouve dans le boitier composants, l'antenne qui permet la réception du signal a été placée sur le dessus du drone. Cela permet d'avoir une meilleure réception.



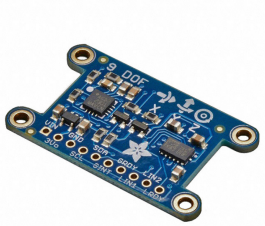
7.1.6 L'altimètre

L'altimètre est connecté à la Raspberry par l'intermédiaire d'un bus I2C. Le placement de l'altimètre me semble litigieux. Je pense que les mesures qu'il réalise sont faussées par la chaleur dégagée par les composants qui l'entourent (plus de détails sont disponibles dans la partie 8 page 25).



7.1.7 La centrale inertielle

Comme l'altimètre, les informations de la centrale inertielle sont récupérées à l'aide du bus I2C. J'ai mis la centrale inertielle de telle façon qu'elle soit positionnée sur le même axe vertical que le centre de gravité du drone. Cette position permet de simplifier les calculs qui seront faits par la suite. En effet, il n'y aura qu'un décalage vertical à prendre en compte.



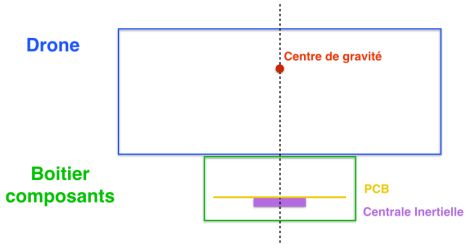


FIGURE 9 – Position de la centrale inertielle par rapport au centre de gravité du drone

7.1.8 Le module Radio Xbee

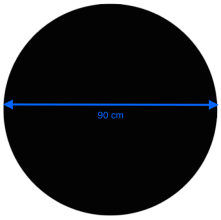
Pour communiquer avec le module radio, on doit utiliser une liaison série. Or la Raspberry ne dispose que d'un bus série. J'ai donc décidé d'utiliser l'un des ports USB de la Raspberry pour connecter ces deux éléments. La connexion se fait par l'intermédiaire d'un cordon qui permet d'adapter le signal. Le module Xbee a été place près d'une paroi du boitier afin que l'antenne de celui-ci puisse sortir à l'extérieur et ainsi avoir un bon signal d'émission.

7.1.9 La batterie externe

Elle fournie l'énergie nécessaire au fonctionnement de la Raspberry et des composants. Elle permet d'avoir une autonomie de 5h et surtout de ne pas diminuer l'autonomie du drone. Cette durée est amplement suffisante vu que l'autonomie du drone n'est que de 40min.

7.1.10 La cible

Étant donné la qualité des photos prise par la caméra, j'ai convenu avec Mr MERZOUKI qu'il est préférable de faire les premiers tests de reconnaissance de cibles à une altitude inférieure à celle que nous avions défini dans le cahier des charges. Nous avons donc convenu que les essais soient réalisés à une altitude de 10 à 15m.
Les premiers essais ont été effectués avec des cercles noirs d'une quinzaine de centimètres de diamètre imprimés sur une feuille blanche. Une cible de cette taille ne pourrait pas être détectée à une hauteur de 10m mais ces expériences avaient pour but de paramétrer correctement les différentes fonctions qui permettent la détection de cercles.
Une fois le programme de traitement d'images réalisé, j'ai peint une nouvelle cible, à placer sur la plateforme du RobuTAINeR. Cette cible est de taille conséquente, 90 cm de diamètre. J'ai choisi d'employer une cible de couleur noire et de forme circulaire car il me semble que c'est parmi les formes les moins présente dans un environnement portuaire.



7.1.11 Les boitiers

Un premier boitier fixé sous le drone accueille l'ensemble des composants (module GPS, radio Xbee, altimètre...). La figure suivante montre la répartition des éléments à l'intérieur de ce boitier.

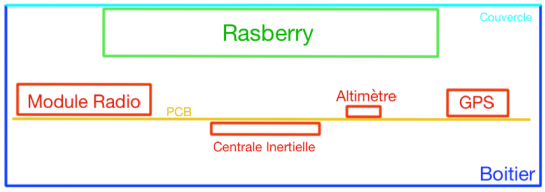


FIGURE 10 – Schéma de la répartition des composants dans le boitier

Le second boitier permet à la fois de stocker la batterie externe et de fixer le premier boitier sous le drone. Je l'ai fabriqué au Fablab de Polytech. Après avoir fait les schémas des différentes parties, j'ai utilisé la découpeuse laser pour tailler les pièces dans une plaque de plexiglass, puis je les ai assemblées.

J'ai conscience que l'utilisation de deux boitiers n'est pas la meilleure solution. Effectivement, le poids n'est pas optimisé car ce montage demande plus de matériaux. Toutefois, j'apporte une solution alternative dans la partie 8 page 25

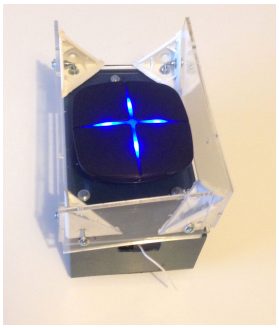


FIGURE 11 – Boitiers vue de dessus

FIGURE 12 – Boitiers vue de dessous

7.2 Logiciel

De septembre à début décembre, je me suis servi de la Raspberry pour tester les composants du projet (communication série ; bus I2C...). Le fait d'avoir travaillé sur des arduinos [5] m'a bien aidé. Les premiers programmes, écrits en Python, étaient donc relativement simples. Ils permettaient de configurer le bus I2C, la liaison série et de prendre une photo ou video avec la caméra.

Dans la deuxième partie de ce PFE, j'ai développé des programmes plus complexes, qui permettent de récupérer les informations issues des capteurs, de les exploiter et de les sauvegarder. La figure 1 est un algorithme simplifié, qui facilite la compréhension des différentes programmes réalisés pour communiquer au RobuTAINeR ses coordonnées. C'est-à-dire de la prise d'une photo jusqu'à l'envoi des données de position. L'ensemble de ces programmes a été codé en Python.

7.2.1 Algorithme

```
Initialisation des modules (GPS / Caméra / Altimètre / Xbee);
while True do
    Programme : Traitement image;
    if Cercle détecté then
        Programme GPS;
        if Signal GPS then
            Programme Changement Base;
            Programme Communication;
        else
            Afficher "Pas de signal GPS"
        end
    end
end
```

Algorithm 1: Algorithme explicatif simplifié

Dans les parties suivantes, je vais détailler chaque programme afin de préciser les actions et opérations réalisées.

5. Cartes électroniques similaires

7.2.2 Programme GPS

Le signal GPS est mis à jour toutes les secondes. Sa précision est de l'ordre de 5m. Ce module n'est pas très précis et sera remplacé par un GPS RTK dont la précision est de l'ordre du centimètre. Lors des premier tests du module GPS, je n'ai pas utilisé de code Python pour visualiser le signal. Une commande dans le terminal de la Raspberry suffit à afficher les informations fournies par le module. Lorsqu'on exécute les deux commandes ci-dessous, on obtient les informations suivantes (Fig : 14).

— sudo gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock
— cgps -s

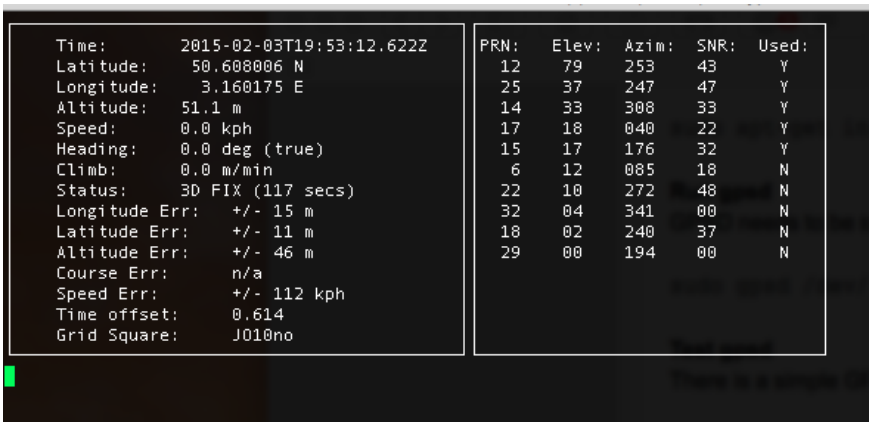


FIGURE 13 – Exemple de retour GPS

Cet affichage est très complet : on y trouve l'heure, la latitude, la longitude, l'altitude. Ce sont des données qui nous seront utiles pour la suite. Parmi les informations que je n'ai pas exploitées, il y a la vitesse, mais aussi sur le coté droit de l'affichage, les différents satellites vus par le module GPS.

Dans la seconde moitié de ce PFE (de janvier à février), j'ai écrit un code permettant de récupérer les informations nécessaires et de les stocker dans un fichier texte afin de permettre une analyse hors ligne après un vol (un exemple de données de vol est disponible en annexe C).

Initialisation du module GPS;
while *True* **do**
  if *Signal GPS* **then**
    Ouvrir le fichier "FlyingData.txt" Enregistrer les données : Latitude, Longitude, Altitude, Heure;
    Fermer le fichier "FlyingData.txt"
  end
end

Algorithm 2: Algorithme explicatif simplifié

7.2.3 Programme de traitement d'images

Le traitement d'images s'est révélé plus difficile que prévu, essentiellement pour des raisons techniques qui ont considérablement ralenti mon travail. En effet, je n'ai pas réussi à utiliser le flux vidéo de la caméra pour effectuer le traitement d'images. J'ai donc décidé de prendre une photo et d'appliquer ensuite le traitement d'images. Ce processus est répété le plus rapidement possible.
Cependant, ce travail n'est pas sans intérêt. Seule la partie d'initialisation de la caméra est à changer dans le cas ou l'on trouve une solution technique ou si l'on change de caméra. En effet, toutes les opérations effectuées sur l'image sont les mêmes.

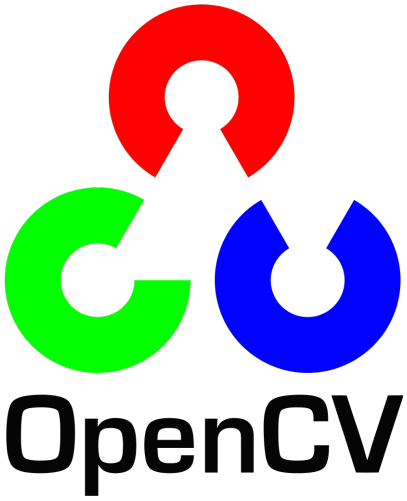


FIGURE 14 – Exemple de retour GPS

Pour effectuer les diverses manipulations sur l'image, j'ai choisi d'utiliser OpenCV, dans un code en Python. OpenCV [6] est une bibliothèque proposant un ensemble de plus de 2500 algorithmes de vision par ordinateur. Ils sont accessibles au travers d'API [7] pour les langages C, C++, et Python. Initialement OpenCV a été développé par des chercheurs de la société Intel. C'est la bibliothèque de référence pour la vision par ordinateur, aussi bien dans le monde de la recherche que celui de l'industrie.
J'ai donc utilisé les différentes fonctions proposées par cette bibliothèque afin de réaliser le traitement d'images.

6. Open Source Computer Vision
7. Interface de programmation

Pour faciliter la compréhension du traitement d'images, je vais détailler les différents étapes de ce processus à l'aide d'un exemple. Ce test a été réalisé en salle C002, avec la cible qui a été présentée dans la partie 7.1.10 en page 13. La présence de nombreuses parties sombres dans cette pièce me semble être un bon exemple de ce que l'on peut rencontrer dans un environnement extérieur.

FIGURE 15 – Niveau de gris

FIGURE 16 – Egalisation de l'histogramme

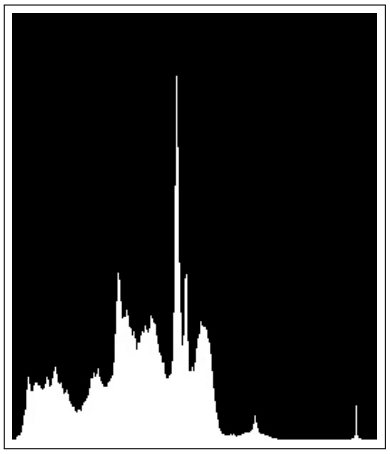


FIGURE 17 – Histogramme avant égalisation

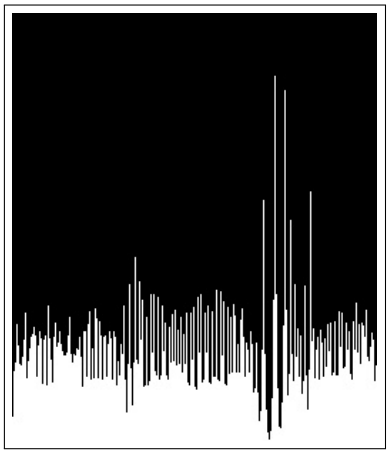


FIGURE 18 – Histogramme après égalisation

Dans un premier temps, on prend une photos avec la caméra. La photo réalisée est une photo couleur (codée en RGB). La manipulation et le traitement d'une image RGB sont assez couteux en temps de calcul. J'ai donc choisi de convertir cette image en niveaux de gris (Fig : 17). Cette opération permet de réduire le temps de calcul et par conséquent d'augmenter le nombre d'images par seconde que l'on peut traiter.

Une fois cette image convertie, on égalise l'histogramme [8]. L'égalisation de l'histogramme permet de mieux répartir les intensités sur l'ensemble de la plage de valeurs possibles. Ici, l'image en niveau de gris a des valeurs allant de 0 pour le noir à 255 pour le blanc. La figure 18 montre le résultat après égalisation. En comparant cette image à la figure 17, on se rend compte que les contrastes ont bien été augmentés. On distingue alors mieux le cercle noir sur fond blanc.

8. notion vu en cours de traitement d'images en IMA4

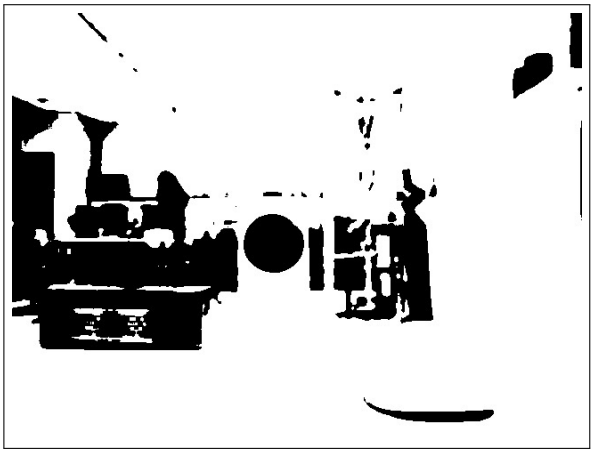


FIGURE 19 – Seuillage 1

FIGURE 20 – Seuillage 2

Le seuillage est une méthode simple de segmentation d'image [9]. À partir d'une image niveau de gris, le seuillage permet de créer une image binaire à l'aide d'une valeur seuil ajustable. Par exemple, dans la figure 19, on fixe un seuil à 50 : toutes les couleurs comprises entre 0 et 50 prennent la valeur 0 et toutes les couleurs de 51 à 255 prennent la valeur 1. La figure 20 est l'inverse de celle-ci. Ces deux images vont servir de masque par la suite. Elles permettront de conserver seulement les parties de l'image qui nous intéressent (les parties noires).

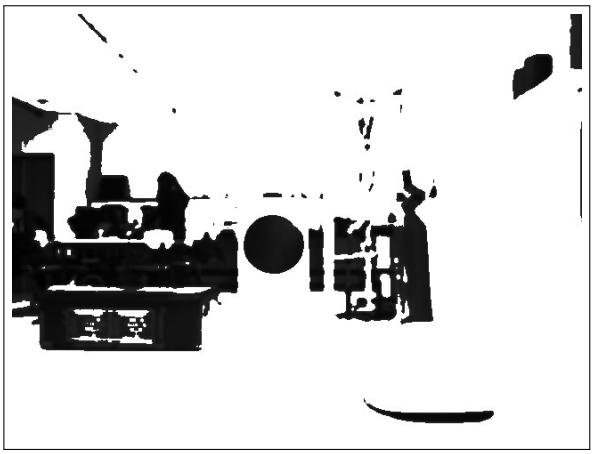


FIGURE 21 – Couleurs sombres

FIGURE 22 – Filtre de Gauss

La figure 21 est obtenue en utilisant une combinaison de l'image niveaux de gris, dont l'histogramme est égalisé, et des deux seuils. Cette opération permet de convertir toutes les couleurs claires dont la valeur est supérieure à 50 en 255. On obtient donc une image niveaux de gris avec uniquement des couleurs comprises entre 0 et 50 ou alors une valeur 255.
Par la suite, on applique un filtre de Gauss (Fig : 22) qui permet de flouter l'image. Le gradient [10] est ainsi moins fort et la détection de cercle qui est appliquée ensuite sera meilleure.

9. notion vu en cours de traitement d'images en IMA4
10. notion vu en cours de traitement d'images en IMA4

FIGURE 23 – Résultat final

Cette dernière image (Fig : **??**) n'est pas présente dans le code du traitement d'images qui sera utilisé. Cependant, elle permet d'afficher le résultat du traitement et d'avoir un retour facilement exploitable lors du réglage des différents paramètres des fonctions. Sur cette image, on voit le cercle detecté affiché en vert et le centre de celui-ci en bleu. Ce programme permet donc de détecté et d'obtenir les propriétés des cercles dans une image. J'ai préféré sauvegarder ces données dans un fichier qui sera disponible après le vol. En effet, comme ce programme n'a pas encore été testé en conditions réelles, cela me permetra de garder une trace pour d'exploiter les données hors-ligne et pour vérifier la robustesse de ce programme.

7.2.4 Le programme altimètre

L'altimètre nous permet de connaitre l'altitude à laquelle le drone se trouve. Sa précision est supérieur à celle du GPS. Cette donnée viendra compléter les informations fournies par le module GPS. Un programme permet de retourner la température, la pression, l'altitude et de les afficher sur l'écran. Mais celle-ci ne correspond pas à l'altitude réelle du drone. Il faut effectivmeent initialiser l'altitude avant que le drone ne décole et se servir de cette donnée comme référence. On a donc une altitude relative et non absolue.

7.2.5 Le programme centrale inertielle

L'entreprise qui fabrique la centrale inertielle a mis en ligne un code qui permet d'exploiter directement les accélérations sur les 3 axes. Mais le code est écrit en language Arduino et ne peut pas être directement executé sur la Raspberry. Je suis capable d'écrire et de lire dans certains registes mais je ne me suis pas attardé sur cette partie par manque de temps.

7.2.6 Le programme de changement de base

Dans cette partie, je vais détailler les calculs qui permettent de passer des coordonnées du centre du cercle données par le programme de traitement d'images aux coordonnées envoyées au RobuTAINeR.
Pour avoir une idée globale du processus, voici un algorithme simplifié, on peut ainsi identifier les étapes successives qui permettent d'arriver à la solution finale.

Hypothéses :

1. Le drone est orienté dans le même sens que le RobuTAINeR ;
2. La centre de la cible est sur l'axe $\vec{x_R}$ à une distance T_{xr} du centre du repère R ;
3. La centre du repère caméra est sur l'axe $\vec{z_D}$ à une distance T_{zc} du centre du repère D ;
4. Les distances DCRx et DCRy peuvent être obtenues avec une fonction d'OpenCV.

Data: Coordonnées du centre du cercle détecté
while *Cercle détecté* **do**
    Déterminer les coordonnées du centre du RobuTAINeR dans le repère de la caméra;
    Déterminer les distances $D_{CR}x$ et $D_{CR}y$;
    Calculer les distances $D_{\perp CR}x$ et $D_{\perp CR}y$;
    Obtenir les coordonnées GPS du drone;
    Déterminer l'orientation du drone par rapport à la base Origine;
    Exprimer les coordonnées GPS du drone dans le repère local (base Origine) Calculer les coordonnées du RobuTAINeR;
end

Algorithm 3: Calcul des coordonnées du RobuTAINeR

En plus de cet algorithme, je vais m'appuyer sur les deux schémas suivants. Ils aideront à interpréter les calculs qui seront énumérés par la suite.

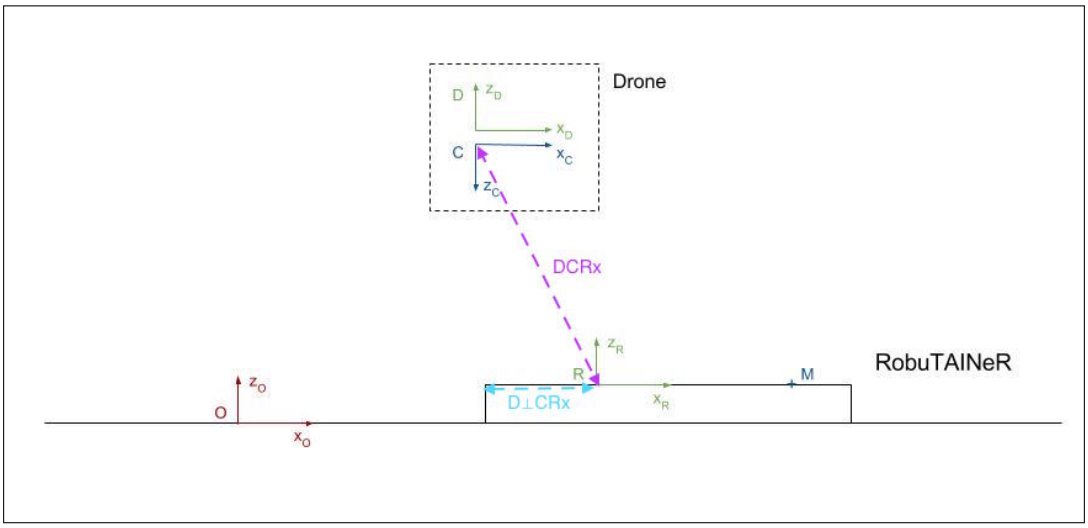


FIGURE 24 – Représentation schématique de profil des bases (Origine, Drone, Caméra et RobuTAINeR)

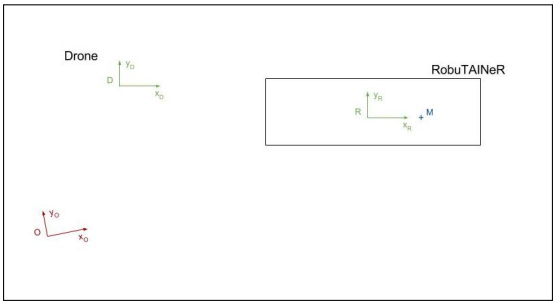


FIGURE 25 – Représentation schématique de dessus des bases (Origine, Drone, Caméra et RobuTAINeR)

Comme on peut le voir dans l'algorithme, ce programme a besoin des coordonnées du centre de la cible (le point M de la figure 24) pour pouvoir fonctionner. Ces coordonnées sont exprimés dans la base de la caméra $(C\,;\,\vec{x_C}\,;\,\vec{y_C})$.

L'étape suivante consiste à faire une translation du point M vers R (Fig : 24). On exprime ainsi le centre du RobuTAINeR dans le repère de la caméra $(C\,;\,\vec{x_C}\,;\,\vec{y_C})$. On utilise le calcul matriciel pour effectuer cette translation. Il suffit de multiplier les coordonnées du point M par la matrice de translation ${}^{M}_{R}T$

$$ {}^{M}_{R}T = \begin{pmatrix} 1 & 0 & T_{xr} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} $$

On cherche maintenant la distance qui sépare le point C du point R. Pour cela, on va utiliser des fonctions d'OpenCV. En effet, à partir de l'image prise et des coordonnées calculées de R, on peut déterminer les distances DCRx et DCRy après calibration de la caméra. Par manque de temps, je n'ai pas pu implémenter cette fonction.

Si on regarde la figure 24, la hauteur du repère R dans la base Origine sera notée Hr. L'altitude du drone est connue est sera notée AltiD. Comme on connait la valeur de DCRx et de DCRy, on est capable de calculer les distances $D_{\perp CR}x$ et $D_{\perp CR}y$.

$$D_{\perp CR}x = \sqrt{DCRx^2 - (AltiD - Hr)^2}$$
$$D_{\perp CR}y = \sqrt{DCRy^2 - (AltiD - Hr)^2}$$

On récupère les coordonnées GPS du drone.

On détermine l'orientation du drone par rapport à la base d'Origine (O ; $\vec{x_0}$; $\vec{y_0}$). La centrale inertielle nous permet de connaitre l'angle $\theta(t)$ entre la base d'Origine et la base du drone.

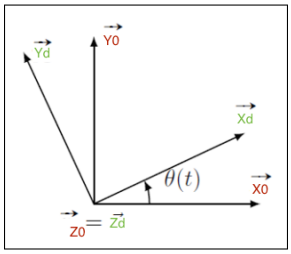


FIGURE 26 – Représention de l'angle $\theta(t)$

Pour exprimer le point R dans le repère du drone, il suffit de passer de la base (C ; $\vec{x_C}$; $\vec{y_C}$) à la base (D ; $\vec{x_D}$; $\vec{y_D}$. Pour cela, il faut utiliser une matrice de passage composée d'une matrice rotation suivant l'axe $\vec{y_C}$ (Rotation de $\theta 1 = 180$) et d'une matrice de translation suivant l'axe $\vec{z_C}$ de T_{zc}. On obtient ainsi la matrice de passage suivante ${}^{C}_{D}T$

$${}^{C}_{D}T = \begin{pmatrix} \cos\theta 1 & 0 & \sin\theta 1 & T_{xr} \\ 0 & 1 & 0 & 0 \\ -\sin\theta 1 & 0 & \cos\theta 1 s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

7.2.7 Le programme de communication

Le programme de communication permet de transmettre à la fois les coordonnées GPS calculées précédemment mais également de préciser l'heure à laquelle ces données ont été envoyées. Cette dernière information est très précieuse. C'est grâce à celle-ci que l'on pourra connaitre le retard entre l'envoi et la réception de la donnée. En pratique, le temps de transmission est négligeable devant la vitesse de déplacement du RobuTAINeR.

8 Problèmes rencontrés et solutions

Dans cette partie, je vais détailler les problèmes que j'ai rencontré, probléme qui perssistent toujours. J'ai essayé pour chacun proposé une solution ou d'ébaucher des pistes. Mais, je n'ai pas eu les moyens de les mettre en oeuvre avant la fin de ce PFE.

8.1 Cible sur le RobuTAINeR

Pour le moment, le RobuTAINeR est localisé par l'intermédiaire d'une cible posée sur la plateforme. Évidemment cette solution ne peut pas être retenue car la cible serait cachée par le conteneur. Il faut donc trouver un autre moyen de repérer le RobuTAINeR.

8.2 Caméra

La caméra embarquée sur le done ne permet pas d'avoir une qualité d'images suffisante pour ce que l'on veut réaliser. J'ai été confronté à de nombreux soucis logiciels. Une camera USB avec meuilleure qualité d'image permettrai de faire du traitement vidéo plus précis. Il faudra cependant être vigilant car toutes les caméras ne sont pas compatibles avec la Raspberry pour faire du traitement vidéo.

8.3 Carte électronique

La carte électronique développée est fonctionnelle. Cependant, il faudrait vérifier la compatibilité électromagnétique entre les composants, afin qu'il n'y ait pas ou peu d'interférences entre eux. Comme je l'ai mentionné antérieurement, la place de l'altimètre est à revoir. En effet, il se trouve dans le même boitier que des composants qui chauffent (exemple la Raspberry). Comme il utilise la température pour connaitre l'altitude, sa mesure s'en trouve donc faussée.

8.4 LED de contrôle

Lors des derniers essais, j'ai réalisé qu'il n'y avait aucun moyen simple et rapide de vérifier l'état des composants. Une solution efficace serait de mettre en place un système de LED commandé par la Raspberry sur le coté du boitier. Ces LEDs indiqueraient l'état des différents composants. Si on utilise une LED RGB par capteur, on pourrait convenir du code couleur suivant :

— VERT : Module fonctionnant correctement ;
— ORANGE : Module en cours d'initialisation ;
— ROUGE : Module hors service.

8.5 Boitiers

Les boitiers réalisés sont "artisanaux". La carte n'étant pas définitive, je n'ai pas commandé de boitiers pour l'ensemble des composants. J'ai reçu un premier boitier qui m'a permis de loger l'ensemble des composants puis j'ai fabriqué le second boitier pour la batterie externe. Évidemment, le poids de l'ensemble n'est pas optimal. La solution qui pourrait être apportée serait de réaliser, à l'aide d'une imprimante 3D, un boitier pouvant accueillir l'ensemble des composants en respectant les contraintes fixées ci-dessus.

8.6 Programme général

À l'heure actuelle, j'ai regroupé l'ensemble des programme réalisés dans un seul et même code. Mr Conrard m'a fait remarqué que l'on pourrait garder un code par composant et les synchroniser à l'aide d'un autre programme en Python. Cette solution serait particulièrement intéressante car on se rapprocherait de la structure d'un système temps réel. On pourrait fixer des priorités aux différents programmes et ainsi hiérarchiser l'échantillonnage les données par ordre d'importance.

8.7 Traitement d'images

Le traitement d'images actuel permet de repérer les cercles noirs dans une image. Cependant, ce traitement n'est pas assez robuste pour être déployé. En effet, celui-ci trouve régulièrement de faux cercles dans l'image. Une solution pertinente serait de complexifier la cible à détecter. Mais plus la cible devient compliquée, plus les calculs nécessaires au traitement d'images deviennent complexes. OpenCV permet de reconnaitre ce genre de forme complexe (voir Feature2D dans la documentation de OpenCV), même quand celle-ci est partiellement cachée.
Voici un exemple de cible que l'on pourrait mettre en place :

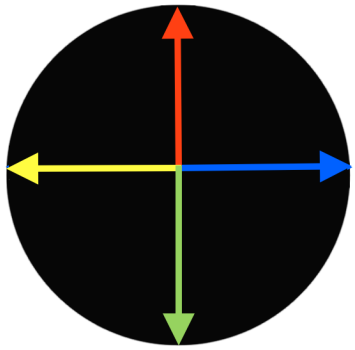


FIGURE 27 – Cible possible pour une détection avancée

9 Conclusion

9.1 Objectifs fixés et objectifs atteints

Le premier pojnt clé de ce projet était la communication entre le drone et le RobuTAINeR, le but était d'avoir une communication fluide entre les deux véhicules. J'ai réalisé plusieurs séries de tests pour valider ce point. Si le RobuTAINeR se déplace à une vitesse de 5km/h, il est possible de mettre à jour sa position 6 à 7 fois par seconde, soit tous les 20 cm. Comme le GPS du drone n'est mis à jour que toutes les secondes, on peut conclure que ce n'est pas la partie communication qui posera problème.

Le second objectif était la réception et le stockage des données GPS. Les données GPS reçues sont stockées pour une éventuelle analyse hors ligne. Cet objectif est donc parfaitement atteint

Pour la détection du RobuTAINeR ,bien que le traitement d'images ne se fasse pas sur un flux vidéo, il a été mis en place et permet de détecter la cible qui est posée sur celui-ci. Cependant, les exigences du cahier des charges ont été revues à la baisse. En effet, l'altitude de vol du drone imposée est de 30m. Nous avons réduit cette altitude de moitié pour que la qualité de la photo puisse être suffisante pour détecter la cible. Cependant, j'ai constaté que le traitement d'image réalisé n'est pas suffisant. Il est nécessaire de développer un processus plus efficace et plus robuste.

Deux autres points clé sont également bien avancés mais n'ont pas abouti. Ce sont l'altimètre et la conversion des coordonnées GPS en coordonnées locales. Une grande partie du travail a été tout de même réalisée.

9.2 Avis personnel

Pour ma part, je trouve que cette étude de faisabilité est concluante. En effet, malgré quelques dysfonctionnements, cette étude montre que le drone offre une vue d'ensemble beaucoup plus large que celle du RobuTAINeR et que ceci est un atout important. Je trouve que le drone apporte un réel plus au système existant. Je suis assez satisfait du travail accompli en particulier pour la gestion de mon projet car j'ai respecté le planning que je mettai fixé. Travailler seul a des avantages au niveau de l'organisation, mais il est évident que le travail en groupe ou au moins en binôme apporte beaucoup lors de la survenue d'un problème.

9.3 Perspectives futures

Dans le domaine portuaire :
Les drones ont beaucoup d'atout dans un environnement portuaire : leur déplacement ne gène ni les personnes ni les véhicules qui travaillent au sol. Leur vitesse de réaction et de déplacement en font un atout incontestable. Bien sûr, il s'agit pas attribuer un drone par véhicule autonome, l'autonomie des drones ne le permet pas. Mais la mise en place d'un réseau de drones serait certainement une très bonne chose. Ils permettraient de fluidifier grandement le déplacement des véhicules autonomes comme le RobuTAINeR lorsque celui-ci perd le signal GPS.

Dans les autres domaines :
Les projets robotiques pourraient bénéficier de ce genre d'aide. Lorsqu'un véhicule autonome doit se déplacer, sa prise de décision est limitée par la vision de ces capteurs. S'il se trouve dans un environnement étroit ou très perturbé (par exemple le traffic dans une ville), celui-ci devra constamment s'arrêter. De plus la probabilité pour qu'il prene le chemin optimal s'amenuise si le nombre d'obstacles augmente. Une vue aérienne est dans la majorité des cas bien meilleure qu'une vue au sol. Le champ de vision étant plus étendu, les décisions de direction pourront alors être prises avec plus de données et par conséquent elles seront plus performantes.

A Drone retenu pour ce projet

FIGURE 28 – Walkera QR x800

Type : 800 size Quadcopter
Main control board : FCS800
Servo : WK-4005
Main rotor dia. : 1200mm
Main rotor length : 400mm
Brushless motor : WK-WS-48-001
Brushless ESC : WK-WST-60A-6
Battery : 6S 22.2V 6000 10000mAh
Flight time : 30 60mins (depends on loading weight)
Flying weight : 3900g (with 6S 10000mAh 10C Walkera battery)
Dimension : 620x620x460mm
Aluminum case dimension : 530x235x560mm

B Schéma de câblage de la Raspberry et des composants

J'ai représenté sur ce schéma uniquement les composants sur lesquels j'ai travaillé (caméra, altimètre, GPS)

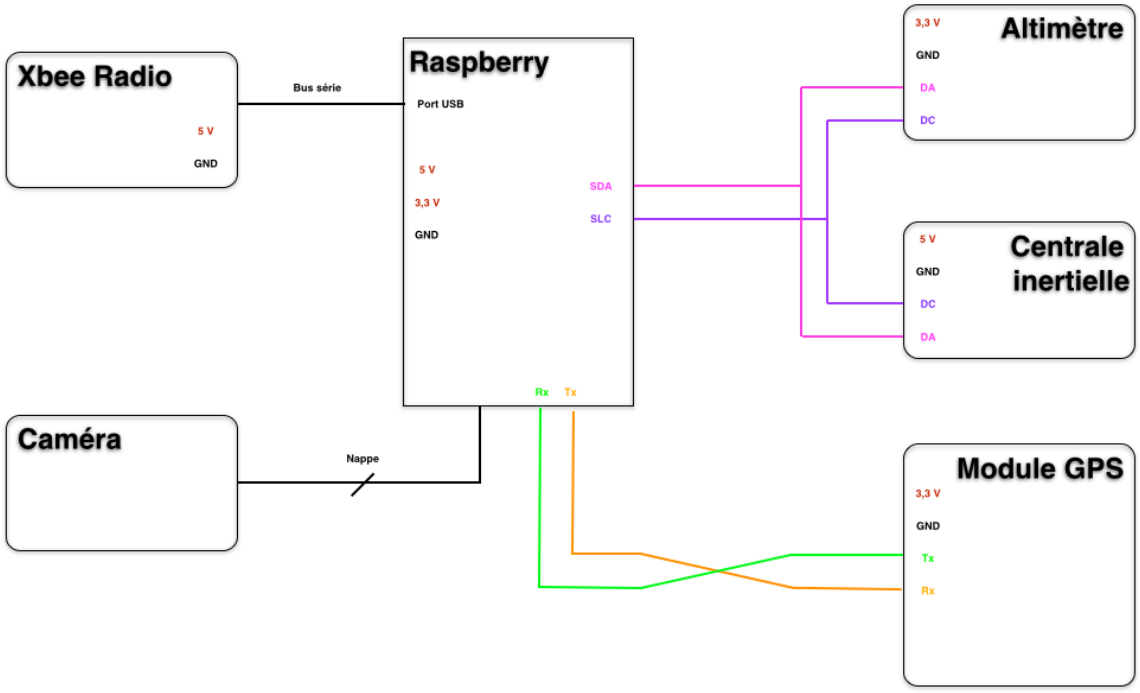


FIGURE 29 – Schéma de câblage

Signification des broches :
3,3 V : Alimentation fournie par la Raspberry
5 V : Alimentation fournie par la Raspberry
GND : Masse électrique commune à tous les composants
SDA / DA : signal de données pour la communication I2C
SLC / DC : signal d'horloge pour la communication I2C
Tx : Broche de transmission pour la communication série
Rx : Broche de réception pour la communication série

C Extrait du fichier FlyingData.txt

FlyingData.txt — Modifié

Latitude	Longitude	Time UTC	Altitude
50.607495	3.137603333	2015-02-11T17:35:36.536Z	46.3
50.607448333	3.137545	2015-02-11T17:35:37.536Z	43.0
50.607448333	3.137545	2015-02-11T17:35:38.536Z	43.0
50.607448333	3.137545	2015-02-11T17:35:39.536Z	43.0
50.607448333	3.137545	2015-02-11T17:35:40.536Z	43.0
50.60745	3.137538333	2015-02-11T17:35:41.536Z	43.8
50.607453333	3.137538333	2015-02-11T17:35:42.536Z	43.9
50.607453333	3.137541667	2015-02-11T17:35:43.536Z	43.9
50.607456667	3.137546667	2015-02-11T17:35:44.536Z	44.0
50.607456667	3.137546667	2015-02-11T17:35:45.536Z	44.0
50.607456667	3.137546667	2015-02-11T17:35:46.536Z	44.0
50.607456667	3.137546667	2015-02-11T17:35:47.536Z	44.0
50.607456667	3.137546667	2015-02-11T17:35:48.536Z	44.0
50.60746	3.137528333	2015-02-11T17:35:49.536Z	44.7
50.607465	3.13753	2015-02-11T17:35:50.536Z	44.8
50.607466667	3.13753	2015-02-11T17:35:51.536Z	44.9
50.607476667	3.137488333	2015-02-11T17:35:52.536Z	44.9
50.60748	3.137443333	2015-02-11T17:35:53.536Z	44.8
50.60747	3.137401667	2015-02-11T17:35:54.536Z	44.6
50.60745	3.137373333	2015-02-11T17:35:55.536Z	44.3
50.607425	3.137355	2015-02-11T17:35:56.536Z	43.9
50.607406667	3.13735	2015-02-11T17:35:57.536Z	43.2
50.607378333	3.137333333	2015-02-11T17:35:58.536Z	42.3
50.60735	3.137316667	2015-02-11T17:35:59.536Z	41.4
50.607318333	3.137301667	2015-02-11T17:36:00.536Z	40.7
50.607288333	3.137296667	2015-02-11T17:36:01.536Z	39.6
50.607263333	3.137313333	2015-02-11T17:36:02.536Z	38.1
50.607243333	3.137371667	2015-02-11T17:36:03.536Z	36.9
50.607226667	3.137445	2015-02-11T17:36:04.536Z	35.8
50.607208333	3.137523333	2015-02-11T17:36:05.536Z	34.9
50.60719	3.137608333	2015-02-11T17:36:06.536Z	34.2
50.607168333	3.137695	2015-02-11T17:36:07.536Z	33.7
50.607146667	3.13778	2015-02-11T17:36:08.536Z	33.3
50.607126667	3.137868333	2015-02-11T17:36:09.536Z	33.0
50.607106667	3.137953333	2015-02-11T17:36:10.536Z	32.9
50.607085	3.138031667	2015-02-11T17:36:11.536Z	32.8
50.607066667	3.138113333	2015-02-11T17:36:12.536Z	32.9
50.607048333	3.138186667	2015-02-11T17:36:13.536Z	32.9
50.60703	3.138263333	2015-02-11T17:36:14.536Z	33.1
50.607011667	3.138335	2015-02-11T17:36:15.536Z	33.2
50.606975	3.138458333	2015-02-11T17:36:17.536Z	33.4

FIGURE 30 – Donées GPS obtenues sur le parking de polytech'Lille