

ÉCOLE NORMALE SUPÉRIEURE PARIS SACLAY

Equipe 1 - SOLUTION CONSTRUCTEUR

COBRA

Course de Ballons Réactifs et Autonomes

Élèves :

Léo-Paul DEWAELE, Arthur POURE
Mathias LE MAIGAT, Emilien DUFRESNES
Pierre-Louis FILOCHE, Mathieu GUERIN

Enseignants :

Bruno DENIS
Anthony JUTON
Bastien DURAND
Bruce ANGLADE

23 Mars 2024

Sommaire

1 Capteurs	4
1.1 Protocole i2c	4
1.2 La centrale inertie - BNO 055	4
1.3 Le capteur à ultrason - SRF10	5
1.3.1 Introduction	5
1.3.2 Fonctionnement général	6
2 Mécanique du système	7
2.1 Système de propulsion	7
2.2 Système de direction	8
2.3 Electronique	8
3 Modèle approché du comportement du dirigeable	10
3.1 Expérience	10
3.2 Analyse	10
4 Commande du dirigeable	12
4.1 Commande générale	12
4.2 Asservissement de l'angle yaw	12
4.3 Asservissement sur l'altitude	13
5 Optimisation : conception de la carte électronique	14
5.1 Introduction	14
5.2 Conception informatique : génération des fichiers nécessaires	14
5.3 Matériel nécessaire	15
5.4 Conception physique de la carte	16
6 Localisation absolue du dirigeable	17
6.1 Introduction	17
6.2 Vision	17
6.2.1 Calibration de la caméra	17
6.2.2 Principe de fonctionnement des apriltags	19
6.2.3 Application au dirigeable	20
6.2.4 Limite	23
6.3 Marvelmind	24
6.3.1 Principe de fonctionnement	24
6.3.2 Implémentation sur le dirigeable	25
6.3.3 Caractéristiques du système	26

Introduction

Le projet COBRA consiste en une course de ballons réactifs et autonomes. Un ballon autonome est un ballon dirigeable n'ayant pas besoin d'intervention de l'utilisateur pour se diriger. Le but du projet est donc de réaliser un comparatifs de trois solutions et de faire une course pour déterminer laquelle des trois est la plus efficace.

Ce compte-rendu portera sur la solution de l'**équipe 1** dite "solution constructeur" car celle-ci reprend la partie mécanique proposée sur le ballon à l'achat. La description de cette solution sera explicité dans la partie **2**.

Pour concevoir notre système, nous avons réparti notre travail entre l'étude et l'incorporation des capteurs ainsi que la commande et la prise en compte de la mécanique du système.

1 Capteurs

La solution constructeur nous a été livré comme commandable avec une radiocommande. Celle-ci ne possède aucun capteur utile à notre étude. Pour acquérir les grandeurs physiques nous servant à piloter notre système nous avons utilisé :

1.1 Protocole i2c

Pour utiliser les différents éléments autour de notre microcontrôleur nous avons utilisé le protocole i2c. L'avantage de ce type de communication, est de nécessiter seulement deux pins sur la raspberry.

Ce protocole est un protocole maître-esclave dont le fonctionnement repose sur une adresse d'identification de chaque capteur permettant de choisir avec lequel on souhaite communiquer.

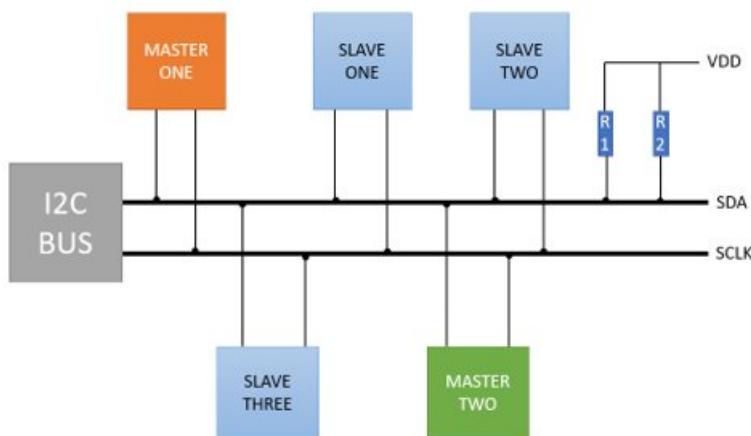


Figure 1 – Schéma explicatif du protocole i2c

Par rapport à la figure 1, nos "slave one", "slave two" et "slave three" sont les différents capteurs et le "master one" sera la raspberry pi zéro 2. Nous n'avons pas d'autre "master".

Comme vu précédemment seulement 2 pins sont utilisés pour la communication : le pin SCLK ou SCL et le pin SDA. Le premier est une horloge et permet de synchroniser les messages entre les capteurs et le maître. SDA correspond donc à la transmission du message. Une des particularités du protocole i2c vient du fait que les deux pins sont reliés à la tension VDD à l'aide de résistance PULLUP (généralement de $4.7\text{K}\Omega$) et donc le protocole fonctionne à une tension donnée. Pour la raspberry pi zéro 2, la tension de travail de l'i2c est 3.3V, ce qui est important à garder en tête.

1.2 La centrale inertie - BNO 055

Tout d'abord, pour piloter notre dirigeable, nous avons cherché à connaître son orientation dans l'espace.

Nos premières idées se tournaient vers une boussole. Ce dispositif permet d'avoir une orientation précise et relative à une donnée fiable (ou presque) : le magnétisme terrestre.

Néanmoins, le contexte de la course nous a freiné dans cette démarche. En effet, la course se passant dans l'atrium de l'ENS Paris-Saclay, cette solution est limitée par les perturbations magnétiques présentent dans cet endroit ; nous avons pu être témoins de variation de plus de 15° en se déplaçant dans celui-ci.

Nous avons donc choisi d'utiliser une centrale inertielles : Notre choix s'est tourné vers le **BNO 055**. Ce capteur nous permet d'obtenir :

- la vitesse angulaire autour de 3 axes ;
- l'accélération en translation selon 3 axes ;
- la position angulaire du capteur.

Notre but est de déterminer la position angulaire à tout instant. Pour éviter les valeurs aberrantes, nous avons utilisé plusieurs données renvoyées par le capteur pour déterminer les angles de rotation du ballon par rapport au sol. Nous sommes donc remontés aux angles d'Euler à partir du vecteur gravité retourné par le capteur et à partir des quaternions. Nous avons finalement fait la moyenne des :

- angles retournés directement par le capteur ;
- angles calculés avec le vecteur gravité ;
- angles calculés à partir des quaternions.

Nous avons donc finalement réussi à obtenir de manière fiable l'orientation du ballon de manière relative, c'est à dire en faisant une initialisation au démarrage

1.3 Le capteur à ultrason - SRF10

1.3.1 Introduction

Une donnée que ne pouvait pas nous fournir la centrale inertielles de manière fiable est l'altitude. Nous aurions plus intégrer les accélérations linéaires obtenues mais celles-ci étaient assez bruitées et la double intégration ne donnait pas de bon résultats. Nous avons donc décidé de placer un capteur à ultrason **SRF10** sur la nacelle, orienté vers le bas pour nous donner directement la hauteur du ballon.

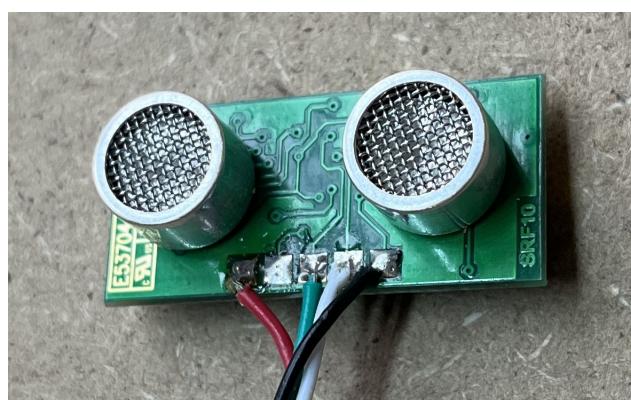


Figure 2 – Capteur ultrason - SRF10

Ce capteur communique aussi via le protocol i2c mais travaille avec une tension de 5V. il est donc obligatoire de passer par une abaisseur de tension pour ne pas griller le microcontrôleur de la raspberry. Nous avons utilisé le **PCA9617** [3] qui permet de passer d'une tension VDDB coté capteur en une tension VBBA < VDDB coté microcontrôleur.

1.3.2 Fonctionnement général

Comme son nom l'indique, ce capteur utilise la technologie ultrason pour évaluer une distance. Le capteur envoie une onde qui va se réfléchir sur une surface et revenir vers lui. Il reste donc à calculer le temps de voyage de l'onde. Un désavantage de ce capteur est donc sa sensibilité à son environnement : si l'onde se réfléchit sur d'autres surfaces, la distance mesurée ne sera pas forcément la bonne. Notre démonstration se faisant dans un amphithéâtre et le sol étant suffisamment plat, la distance calculée reflètera plutôt bien la distance réelle.

Finalement, on peut aussi modifier les gains internes au capteur [1] modifiant sa sensibilité et donc obtenir des valeurs plus précises. Cela a été nécessaire dans notre cas pour obtenir des valeurs plus précises et fiables.

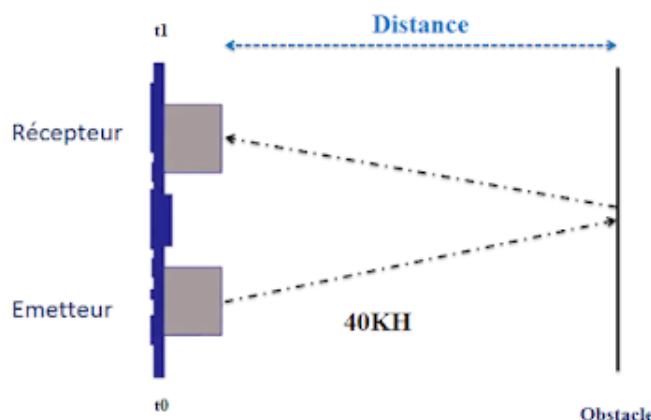


Figure 3 – Principe de fonctionnement du capteur SRF10

2 Mécanique du système

La solution constructeur, qui est celle que nous étudions est composée de trois moteurs qui permettent de contrôler le dirigeable. Deux de ces moteurs servent à la propulsion du dirigeable et le dernier sert à sa direction

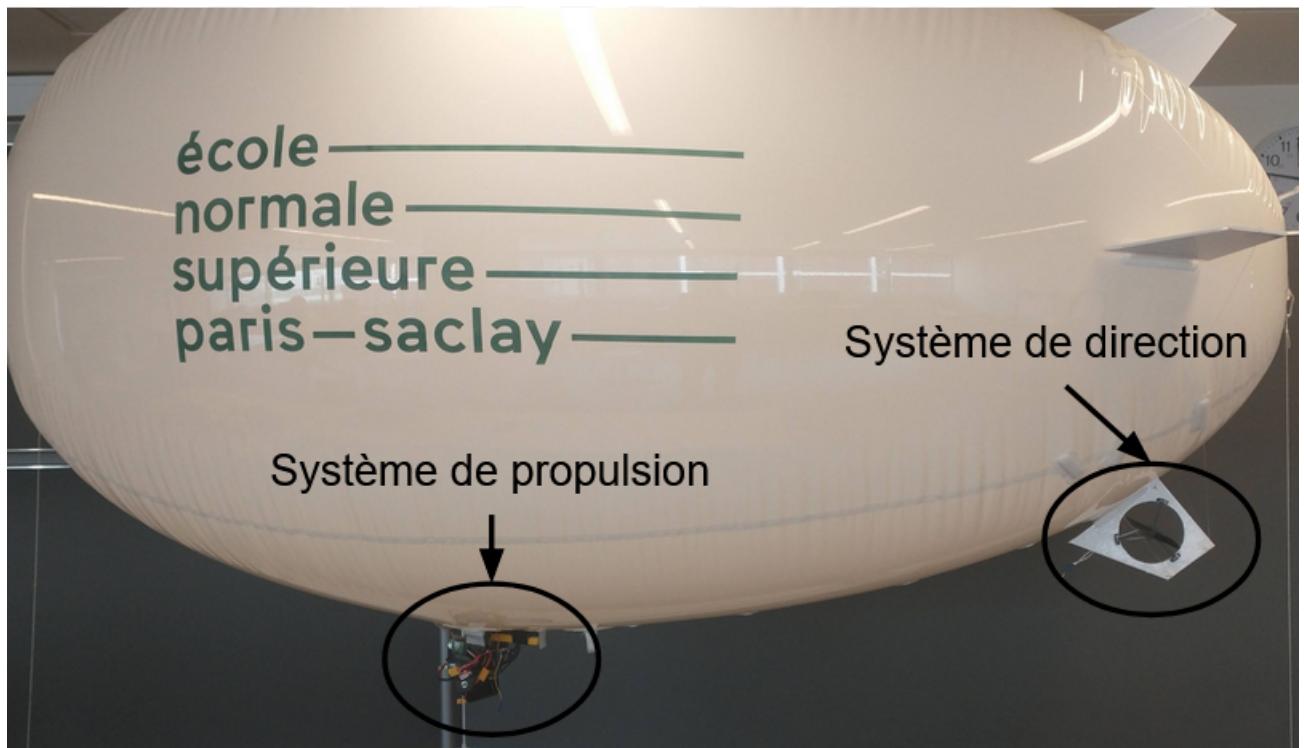


Figure 4 – Image du dirigeable

2.1 Système de propulsion

La propulsion du système est assurée par le moteur qui se trouve sous le dirigeable. Ce moteur est un moteur brushless qui ne tourne que dans un sens. Il est orientable selon l'angle de tangage grâce à un second moteur, qui est un moteur stepper et un système poulie-courroie. Ce qui permet de se servir de ce moteur pour propulser le dirigeable vers l'avant et aussi de contrôler son altitude.

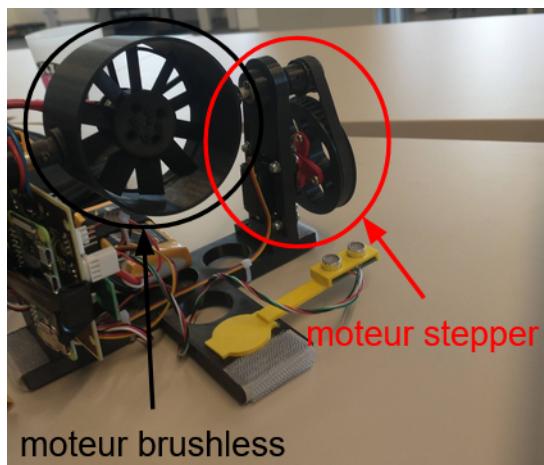


Figure 5 – Image du système de propulsion

2.2 Système de direction

Pour ce qui est de contrôler la direction dans laquelle avance le dirigeable, il y a le moteur qui se trouve à l'arrière du dirigeable. Ce moteur est un moteur à courant continu. Il possède donc une zone morte, mais tourne dans les deux sens, ce qui permet de contrôler l'angle de lacet.



Figure 6 – Image du système de direction

2.3 Electronique

Notre système a aussi besoin de capteurs et d'une batterie, qu'il faut placer sur le dirigeable. Tous ces éléments seront placés sur la nacelle où se trouve le système de propulsion. Les capteurs et moteurs sont reliés à la carte raspberry pi par un circuit imprimé. Le capteur BNO055 se trouve d'ailleurs sur ledit circuit imprimé. Mais le capteur à ultrason ne peut pas être sur ce circuit car il a besoin d'être orienté vers le bas et ne doit pas être obstrué par les éléments de la nacelle. Pour cela nous avons créé une pièce qui permet de l'accrocher à la nacelle.

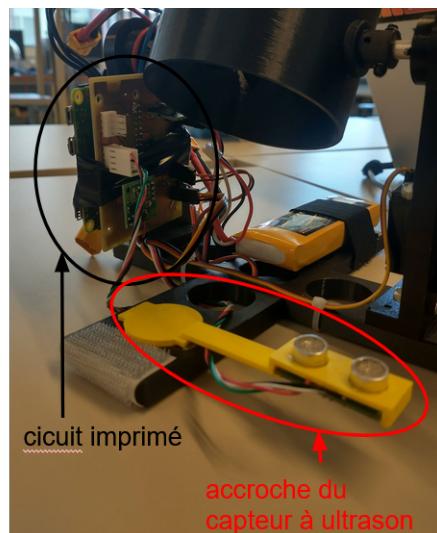


Figure 7 – Accroche de la carte électronique et du capteur ultrason.

Enfin, pour ce qui est de la batterie, elle est accrochée à la nacelle par une bande élastique. Et nous avons à notre disposition une batterie à deux cellules qui sort du 7.4V et fournit du 1800mA/h



Figure 8 – Accroche de la batterie

3 Modèle approché du comportement du dirigeable

Pour notre asservissement nous avions besoin de la vitesse de rotation du dirigeable et de son évolution au cours du temps.

3.1 Expérience

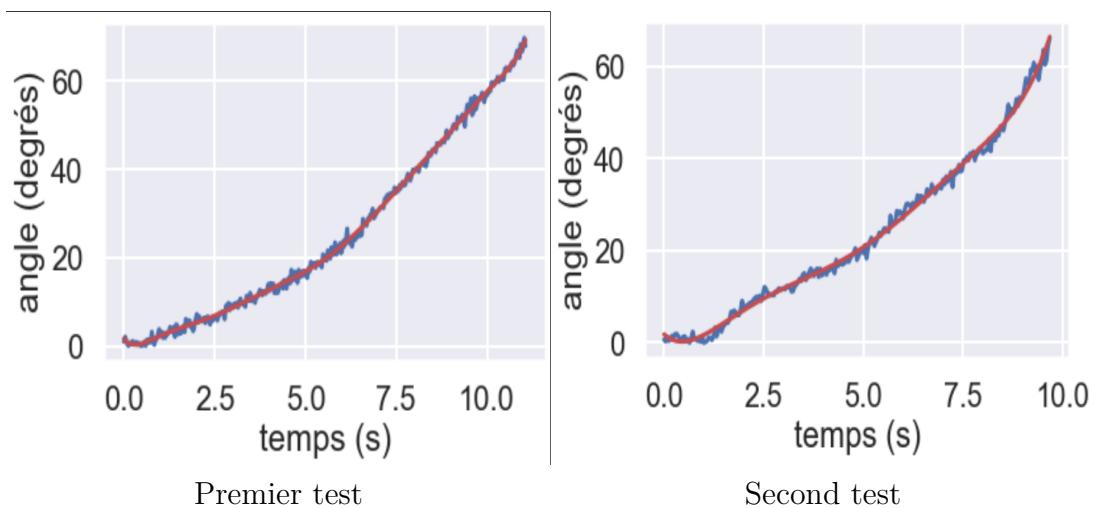
Pour obtenir notre modèle nous avons fait une expérience dans laquelle nous avons donné une consigne de rotation au moteur arrière du dirigeable et avons filmé depuis une hauteur assez importante son comportement. Nous avons répété cette expérience une seconde fois. Ce qui nous a donné deux vidéos du dirigeable.



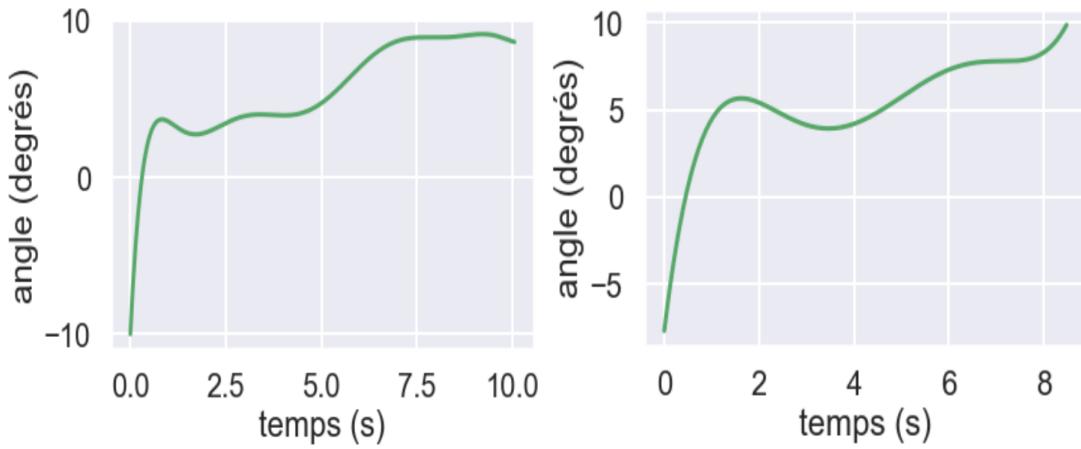
Figure 9 – Image de l’expérience

3.2 Analyse

Une fois l’expérience réalisée nous avons analysé les images obtenues sur le logiciel Tracker pour obtenir l’angle du dirigeable en fonction du temps, ce qui nous a donné pour les deux vidéos :



Nous avons ensuite fait un fitting des courbes par des polynomes, pour pouvoir obtenir des dérivées des positions angulaires assez précises :



Premier test

Second test

On observe donc que notre système physique a un retard de 1 seconde et que la vitesse de rotation n'est pas vraiment constante mais reste toujours inférieure à 10 degrés par seconde pour une consigne constante sur la vitesse de rotation du moteur arrière.

4 Commande du dirigeable

Pour commander le dirigeable nous avons opté pour un système de commandes par états :

4.1 Commande générale

Nous commençons par initialiser le repère du dirigeable avec un angle initial par rapport au sol de 90° et une altitude initiale. Nous utilisons ensuite trois états pour commander le dirigeable :

- L'état d'avancée : on commande au système de propulsion d'avancer. Et on utilise deux asservissements, un sur l'altitude et un sur l'angle yaw du dirigeable par rapport au sol
- L'état arrêt : on inverse la propulsion du dirigeable tout en conservant les asservissements sur l'angle et sur l'altitude.
- L'état de demi-tour : on arrête la propulsion du dirigeable et on impose une consigne en échelon sur le moteur de direction jusqu'à ce que l'on atteigne un angle de 160° par rapport au repère du sol. Ce qui correspond à une rotation de 180° du ballon

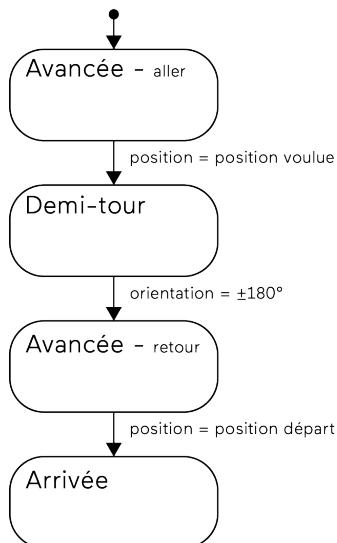


Figure 10 – Statechart du control du ballon

4.2 Asservissement de l'angle yaw

Pour asservir l'angle yaw on utilise le système de direction, qui est notre moyen d'action sur cet angle, et la centrale inertie BNO055 qui est notre moyen de mesure de l'angle et de sa dérivée. Nous utilisons un correcteur PD sur notre système pour compenser le retard et la très grande inertie de notre système.

4.3 Asservissement sur l'altitude

Pour asservir l'altitude on utilise le système de propulsion, et plus particulièrement le moteur stepper qui est notre moyen d'action sur l'inclinaison du moteur brushless et donc sur l'effort vertical appliqué au dirigeable. Et comme moyen de mesure nous utilisons le capteur ultrason SRF10.

5 Optimisation : conception de la carte électronique

Dans cette partie sera explicité les étapes de conception de la carte électronique

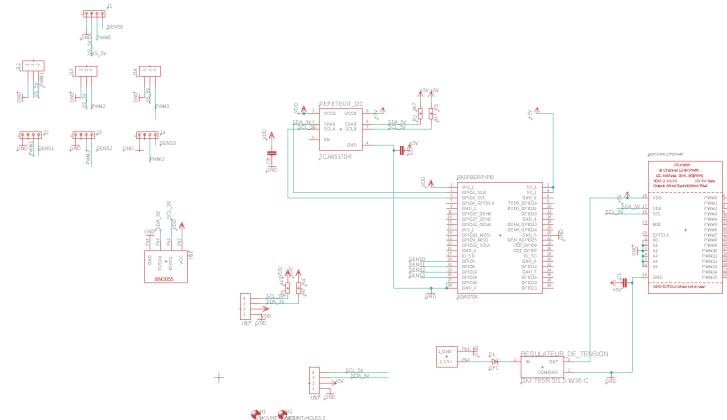


Figure 11 – Schéma eagle de la carte électronique

5.1 Introduction

Dans une optique d'optimisation maximale, on se propose de réduire le poids global du dirigeable tout en simplifiant la prise en main électronique des dispositifs des moteurs et capteurs, ceci revient à concevoir la carte électronique qui permet de relier tout ce qui commande le dirigeable d'une façon à avoir le poids le plus faible possible.

5.2 Conception informatique : génération des fichiers nécessaires

Pour produire la carte physiquement, il nous faut au moins définir informatiquement quoi imprimer afin de suivre le protocole (qui sera défini plus tard) de conception de la carte réelle.

On se sert du logiciel **Eagle**. Ce logiciel nous permettra de réaliser un modèle de carte vérifiant ces critères :

- entrée alimentation : prise en compte du système d'alimentation qui sera relié aux pistes de toute la carte
 - solutions adaptatives pour les différents groupes du défi : la carte pourra être utilisée pour tous les groupes, qu'ils utilisent un des cas parmi la solution constructeur, un moteur à deux hélices, etc...
 - réalisation des pistes pour rendre la carte fonctionnelle
 - sélectionner les composants électroniques (BNO055, PCA, etc...) que les groupes utilisent
- Toutes ces démarches sont à réaliser sur le logiciel Eagle.

5.3 Matériel nécessaire

en prenant en compte tous les critères listés ci dessus, et les besoins des différentes méthodes mises en oeuvre pour notre solution de pilotage du dirigeable, nous arrivons à cette liste de composants :

Carte : composants
BNO55
connecteurs détrompeurs : 2
OKI 78 SR
connecteurs 4 broches (mâle-mâle): 4
connecteurs 3 broches (mâle-mâle) : 3
connecteur 40 broches
condensateurs CMS : 3
Resistances CMS :4
PCA9685
I2C
Diode CMS
connecteur batterie (gaine thermo)

Figure 12 – composants nécessaires pour une seule carte

Ensuite, la problématique qui se pose est d'agencer tous les composants sur le moins de matière possible (faire la carte électronique la plus compacte) c'est un jeu de piste qu'il faut résoudre, ce qui nous a donné ceci ;

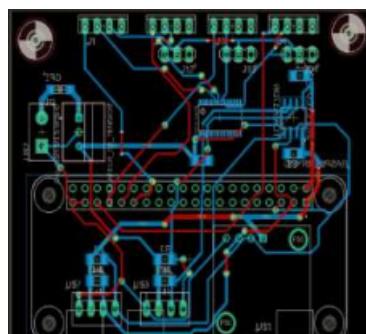


Figure 13 – Agencement des composants sur la carte

Ci dessus nous avons le schéma à imprimer de la carte, tous les pins (entrées) sont en vert, les cables situés sur la face du dessus sont en bleu et ceux du dessous sont en rouge.

Cette partie de réalisation de l'optimisation du dirigeable a de loin été, pour l'électro-nique, la plus compelexe à réaliser. En effet les exigences imposées par les groupes étaient diverses, et il fallait trouver une solution qui satisfasse à la fois tout le monde, mais la plus optimisée possible. Une fois les fichiers .brd et .sch réalisée, il suffit de suivre les étapes décrites dans la prochaine partie, plus simple mais plus délicate car plus technique à réaliser :

5.4 Conception physique de la carte

la carte se construit en plusieurs étapes, au début, nous allons expliciter comment transformer le bloc de cuivre couvert d'une surface photosensible en carte électronique pour notre dirigeable :

- Création du "pattern" de la carte : on se sert du calque imprimé de la carte pour couvrir les surfaces qui vont conduire : en passant la carte placée dans le calque, les zones non couvert par le calque n'auront plus la surface photosensible : ce seront **les pistes conductrices**.

- Ensuite, on passe l'ensemble de la carte dans un dissolvant : le cuivre devenu apparent est retiré grâce à cette solution.

Le procédé de fabrication des matériaux de la carte est à présent terminé.

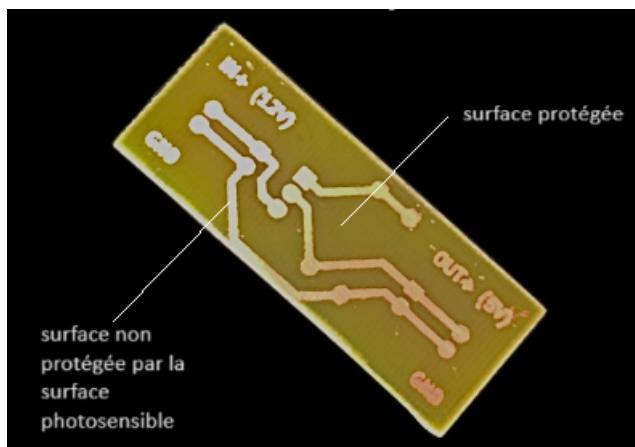


Figure 14 – Rendu de la carte post-traitement chimique

Le reste de la conception mécanique et électrique reste à faire :

- Procédé(s) mécanique(s) : perçage des trous pour réaliser les vias, les interstices pour connecter des câbles, ou des vis. Soudage pour réaliser les vias, et pour fixer des connecteurs sur la carte.

- Procédés électriques : Test post traitement pour vérifier le bon fonctionnement de la carte, test en suivant les pistes tout en s'assurant que le **bon** signal y es présent.



Figure 15 – Carte réelle

6 Localisation absolue du dirigeable

6.1 Introduction

Le dirigeable autonome nécessite un système de localisation absolue pour pouvoir être autonome et se déplacer sans intervention de l'utilisateur. L'objectif de cette partie est d'évaluer et de comparer des techniques de localisation en intérieur en vue de l'implémentation sur le dirigeable. Nombreuses solutions ont été testées, aux vues des équipements disponibles et des résultats obtenus les deux retenues sont la vision et la radio. Cependant ces deux solutions nécessitent un équipement de la zone d'utilisation (Tags , Balises).

6.2 Vision

La solution nécessite deux prérequis : la disposition de différents tags dont la position sera connue, ainsi qu'une caméra embarquée pour les détecter.

6.2.1 Calibration de la caméra

Afin que la détection se fasse correctement, on doit dans un premier temps calibrer la caméra afin de définir ses caractéristiques intrinsèques.

Pour cela, on utilise un damier dont les caractéristiques sont connues et on utilise la bibliothèque Python OpenCV.



Figure 16 – Exemple de damier utilisé

On utilise des photos du damier avec différents angles et distances, puis on applique le programme. Ce dernier va rechercher les coins internes dans un premier temps puis calculer les paramètres de la caméra.

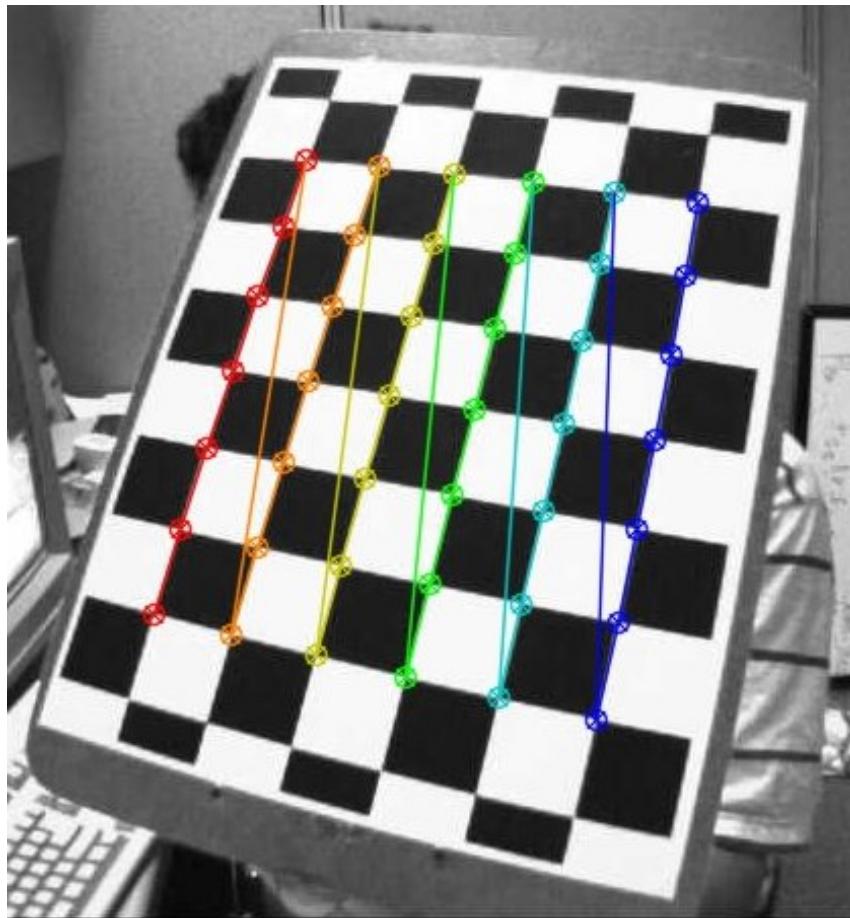


Figure 17 – Image après avoir trouvé les coins internes

ainsi on obtient les paramètres sous forme de matrice

$$\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

les termes f correspondent à la longueur focal et les termes c correspondent au centre optique. On obtient également les coefficients de distortion mais ces derniers sont négligeables et seront considérés égaux à 0.

6.2.2 Principe de fonctionnement des apriltags

Pour fonctionner, la caméra doit voir l'AprilTag dans sa totalité. Chaque AprilTag a un ID propre (il existe environ 500 différents). Il existe également différents types d'AprilTags.

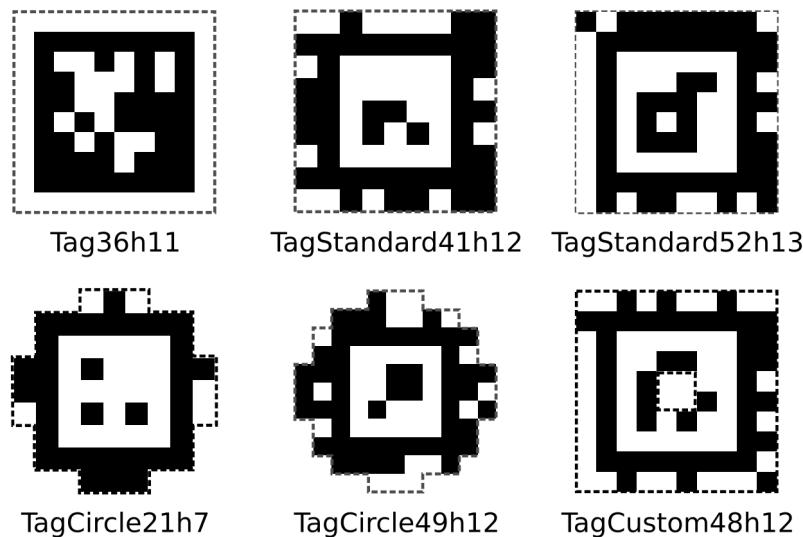


Figure 18 – Different type d’apriltags

Pour notre part nous utiliserons les types *Tag36h11* ou 11 designe l’ID du tag.

Pour détecter le tag, le programme applique une série de filtre à l’image capturé.

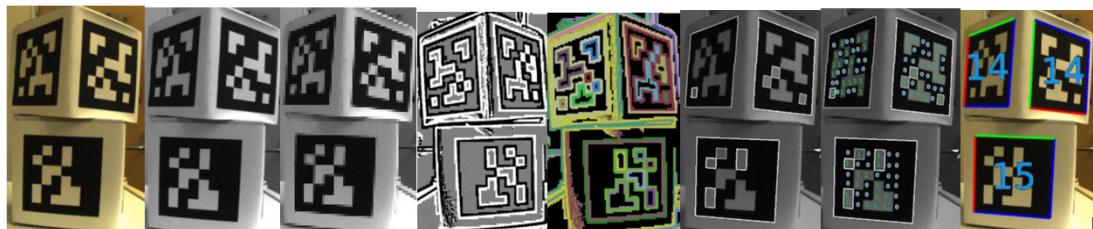


Figure 19 – Serie de filtre appliqué

On obtient de nombreuses données, celle qui nous intéresse sont la matrice de rotation (pos_R) et le vecteur translation (pos_T) du tag.

6.2.3 Application au dirigeable

Avec les données que nous récupérons grâce à la bibliothèque dt-apriltags, nous récupérons la position (X,Y,Z) du dirigeable ainsi que potentiellement son orientation (cf. limites).

Ci-dessous on peut retrouver le code servant à la localisation :

```
import cv2
from picamera2 import Picamera2
from dt_apriltags import Detector
import numpy as np

class Localisation():
```

```
def __init__(self, ListePoints):
    self.picam2 = Picamera2()
    HEIGH, WIDTH=1232, 1640
    self.picam2.configure(self.picam2.
        create_preview_configuration({'size':(WIDTH,HEIGH)}))
    self.picam2.start()

    self.at_detector = Detector(families="tag36h11", nthreads=1,
        quad_sigma=0.0, refine_edges=1, decode_sharpening=0.25, debug
        =0)

    self.fx=1242.8
    self.fy=1242.6
    self.cx=810.4
    self.cy=664.04
    self.dist=np.array([0.007343,1.236,-0.001877,0.0068044]) #  
        coefficients de distorsion de la camera

#self.camera_matrice=np.array([[self.fx,0,self.cx],[0,self.fy
    ,self.cy],[0,0,1]]) #matrice de la camera

    self.Liste_points_3D = ListePoints

    self.matrice=np.array([-1,0,0],[0,1,0],[0,0,1]) #  
        d finition d'une matrice pour la sym trie sur un axe

def capture(self):
    img=cv2.cvtColor(self.picam2.capture_array(),cv2.
        COLOR_BGR2GRAY) #prise d'une photo
    tag=self.at_detector.detect(img, estimate_tag_pose=True,
        camera_params=[self.fx,self.fy,self.cx,self.cy], tag_size
        =0.173) #tag_size=0.052 pour les petits
    return tag

def angle(self,R):
    sy = np.sqrt(R[0, 0] * R[0, 0] + R[1, 0] * R[1, 0])
    singular = sy < 1e-6

    if not singular:
        x = np.arctan2(R[2, 1], R[2, 2])
        y = np.arctan2(-R[2, 0], sy)
        z = np.arctan2(R[1, 0], R[0, 0])
    else:
        x = np.arctan2(-R[1, 2], R[1, 1])
        y = np.arctan2(-R[2, 0], sy)
        z = 0
```

```
    return np.degrees(np.array([x, y, z]))\n\n\ndef mesure(self):\n    tag=self.capture()\n    Positions=[]\n    PositionMoyenne=np.array([0,0,0],dtype='float64')\n\n        for i in tag:\n            pose=np.dot(np.transpose(i.pose_R),i.pose_t)\n            Positions.append(np.dot(self.matrice,np.transpose(pose)\n                [0])+np.array(self.Liste_points_3D[i.tag_id])) #\n                formule pour trouver la position      partir du resultat\n                apriltag pour le i- me  tag\n        for e in Positions:\n            PositionMoyenne+=e\n\n    n=len(Positions)\n    if n!=0:\n        PositionMoyenne=PositionMoyenne/n\n        PositionMoyenne[2]+=0.05 #pr sence d'un offset sur l'axe\n            z\n\n            return (PositionMoyenne,n)\n    else:\n        return None
```

Ce programme sera implémenté sur une carte Raspberry Pi Zero avec la caméra dont on a les caractéristiques.



Figure 20 – Raspberry pi zero avec caméra

6.2.4 Limite

Des tests de précision poussés n'ayant pas encore été effectués, on estime l'erreur à quelques centimètres. De plus, aucun essai n'a encore été fait sur la fonction qui renvoie l'angle et donc l'orientation du dirigeable. C'est la prochaine étape avant la mise en pratique.

6.3 Marvelmind

Dans le domaine des véhicules autonomes, il est crucial de pouvoir se localiser avec précision, que ce soit en extérieur ou en intérieur, où les signaux GPS traditionnels ne fonctionnent pas bien. Pour répondre à ce besoin, Marvelmind, une entreprise spécialisée dans les systèmes de positionnement intérieur (IPS), a développé une solution innovante. Cette technologie de "GPS d'intérieur" utilise des balises à ultrasons et des ondes radio hautes fréquences pour fournir une localisation extrêmement précise, avec une marge d'erreur annoncée de 2 cm. Notre solution repose sur cette technologie commerciale avancée.

6.3.1 Principe de fonctionnement

La solution se base sur un réseau de balises fixes placées au préalable dans le lieu où l'on souhaite suivre l'objet mobile. Comme mentionné précédemment, l'utilisation de ce système nécessite que l'on équipe le dirigeable d'une balise, qui sera qualifiée dans la suite de balise mobile. Toutes les balises communiquent entre elles et avec le modem via



Figure 21 – Kit de balises et modem MarvelMind

des ondes radios dans une bande de fréquence libre (915/868 MHz). Le modem sert de points de relais avec l'ordinateur sur lequel tourne le logiciel Dashboard pour configurer les balises et suivre l'évolutions des distances et la positions de la balise mobile.

En utilisant l'algorithme de trilateration, la position d'une balise mobile est calculée en se basant sur le délai de propagation des impulsions ultrasons (Time-Of-Flight ou TOF) entre les balises fixes et la balise mobile.

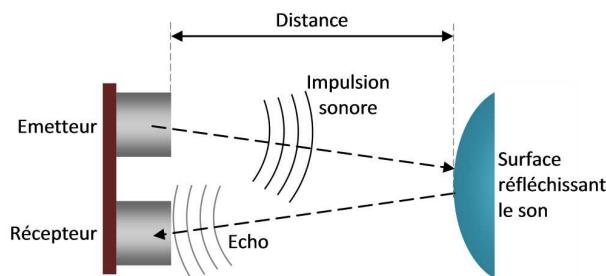


Figure 22 – Principe physique de TOF - Emetteur Récepteur ultrasons

Pour cela toutes les balises sont dotées d'une partie réceptrice séparée avec un microphone à large faisceau et des filtres DSP(technique utilisée pour traiter les signaux

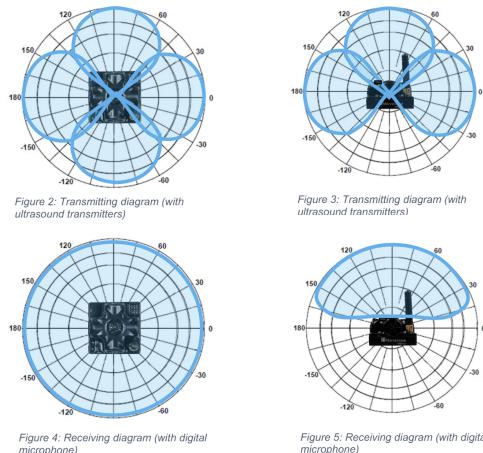


Figure 23 – Diagramme émissions/réceptions des balises

audio en temps réel en via une conversion analogique numérique puis utilisant des algorithmes numérique) précis, ce qui le rend plus sensible, plus résistant au bruit externe et plus facile à configurer. Réception de fréquences ultrasonores variées : Il peut recevoir des fréquences ultrasonores dans les bandes suivantes : 19kHz, 22kHz, 25kHz, 28kHz, 31kHz, 34kHz, 37kHz, et 45kHz. Le filtre peut être sélectionné simplement via Dashboard.

Au vu des différents diagrammes, deux choix s'offrent à nous pour que la balise mobile soit au maximum visible par les balises fixes. On peut soit placer l'ensemble des balises fixes au ras du sol et la balise mobile sous le ballon. La deuxième option consiste en le placement des balises fixes en hauteur (supérieur à l'altitude max du ballon) et la balise mobile sur le dessus du ballon. C'est cette deuxième option qui est retenue car il y a plus d'interférences au niveau du sol (personnes qui passent, mobilier, etc.).

Le système peut automatiquement créer une carte des balises fixes car elles peuvent se situer les unes par rapport aux autres. Cependant cette configuration n'est pas la plus précise, nous avons donc fait les mesures à la main et créé la carte dans le logiciel en fixant manuellement les coordonnées des balises fixes.

6.3.2 Implémentation sur le dirigeable

Notre but est de construire un dirigeable autonome. Pour cela nous avons décidé d'embarquer tous les systèmes de traitement de l'information dans le ballon. (Une autre alternative aurait été d'envoyer les informations sur un ordinateur externe au système pour qu'il traite les données et les renvoie au ballon.) Il nous faut donc récupérer toutes les informations sur l'ordinateur de bord (RaspberryPi 0). Pour cela on ne peut plus utiliser le logiciel Dashboard qui est trop gourmand en ressources. A la place nous avons développé un code python qui permet d'extraire les données directement depuis la balise mobile. Ce code extrait les coordonnées (x,y,z) de la balise mobile par rapport à la carte



Figure 24 – Portion d'atrium que nous pouvons couvrir.

établie au préalable sur le logiciel dashboard (sur ordinateur en utilisation normale).

6.3.3 Caractéristiques du système

La première question qui s'est posée pour évaluer les caractéristiques du système était le positionnement des balises fixes dans la salle.

La deuxième géométrie est celle qui nous donnée la meilleure précision. Cela peut s'expliquer par le fait qu'elle couvre une plus grande zone où la balise mobile est vue par plusieurs balises fixes.

On a obtient en embarqué sur le ballon une précision statique de 5cm en (x,y) avec un taux de rafraîchissement des données de 20Hz.

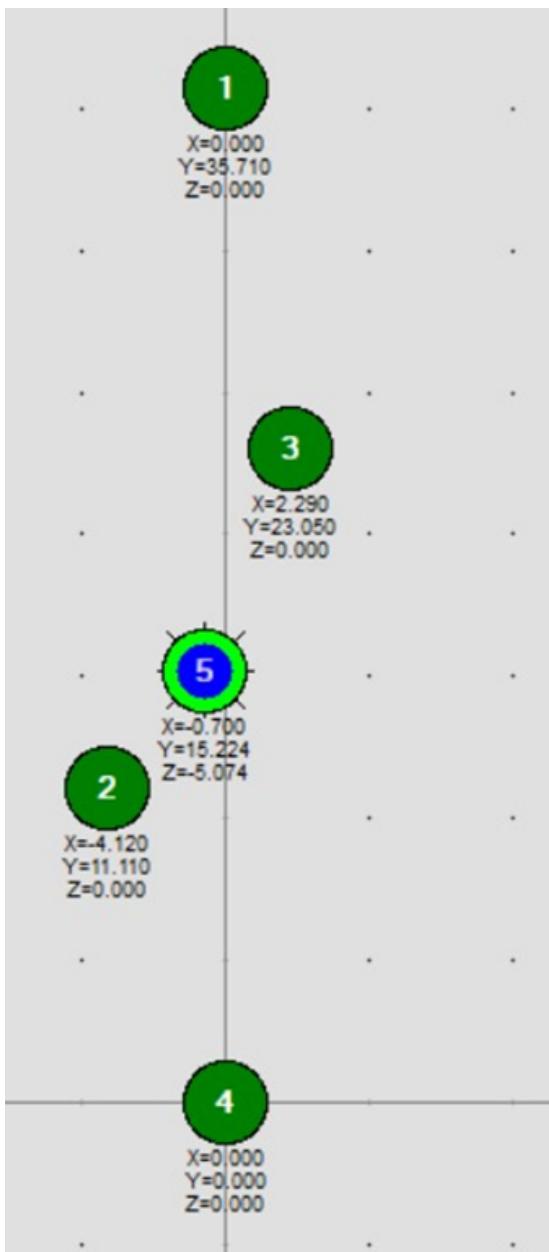


Figure 25 – Première géométrie

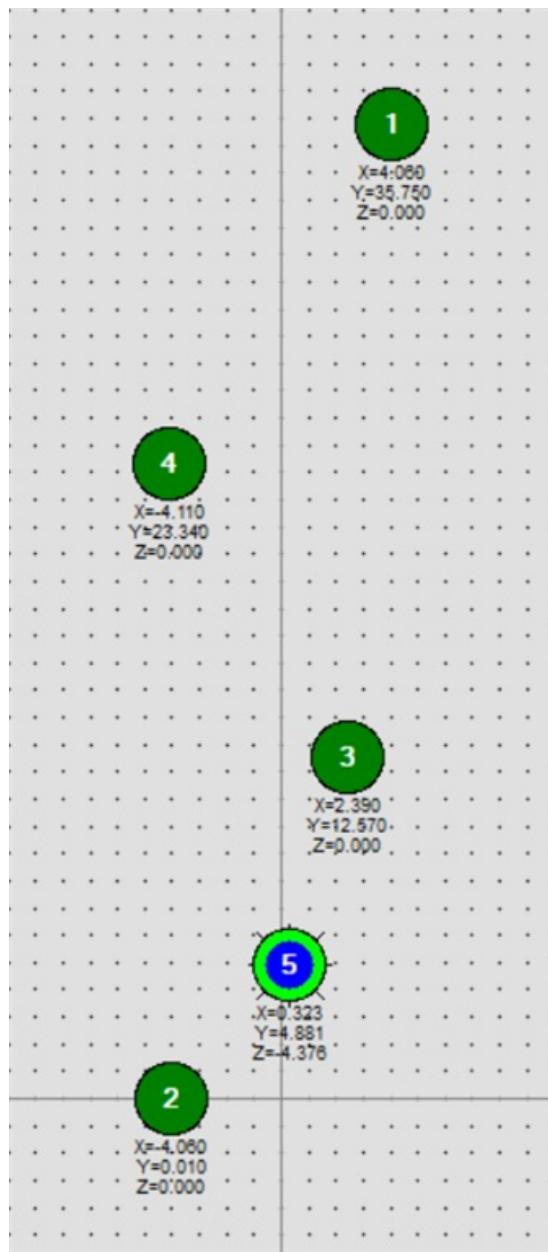


Figure 26 – Deuxième géométrie

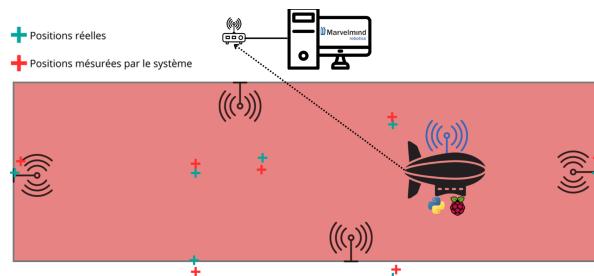


Figure 27 – Test sur la précision des balises avec la 2e géométrie

Références

- [1] *Capteur SRF10.* URL : <https://www.robot-electronics.co.uk/htm/srf10tech.htm>.
- [2] *DesignReuse.* URL : <https://www.design-reuse.com/articles/54776/i2c-interface-timing-specifications-and-constraints.html>.
- [3] DOCUMENTATION. *PCA9517, level translating I²C-bus repeater.* 2007.
- [4] *principe TOF.* URL : <https://arduino.blaisepascal.fr/capteur-de-distance-a-ultrasons/>.