

KÉVIN REN, THÉO MASSA
GUILLAUME GARDE, HUGO HOFMANN

UV 6.1 – PROJET AU LAC DE GUERLÉDAN

DOCKING

26 février 2024



Table des matières

1	Introduction	4
1.1	Présentation du sujet	4
1.2	Matériel à disposition	4
1.2.1	Dock	4
1.2.2	Drone AUV	5
1.2.3	Rover	5
2	Conception du dock	7
2.1	Mise en place d'une base RTK	7
2.1.1	Capteurs à disposition	7
2.1.2	Installation de la base	8
2.1.3	Configuration du dock	8
2.1.4	Un mot sur les antennes	9
2.1.5	Test, difficultés et résultats	9
2.2	Mise en place du dock	9
2.3	Communication avec le reste du système	10
3	Stratégie d'approche de docking	12
3.1	Filtre de Kalman	12
3.2	Guidage par champ de potentiel artificiel	13
3.2.1	Champ de potentiel retenu	13
3.3	Algorithme	15
4	Architecture logicielle	16
4.1	ROS	16
4.1.1	Présentation générale	16
4.1.2	Limitations	16
4.2	Architecture du projet	17
5	Essais à Guerlédan	18
5.1	Première semaine	18
5.2	Deuxième semaine	19
6	Conclusion	20
6.1	Résultats	20
6.1.1	Rover	20
6.1.2	Drone	21
6.1.3	Synthèse	21
6.2	Perspectives	22

7 Annexes	23
7.1 Configuration des modules <i>u-blox</i> avec <i>U-Center</i>	23
7.1.1 Configuration de la base	23
7.1.2 Configuration du dock	24
7.1.3 Informations sur les LEDs	24
7.2 Configuration des antennes <i>XBEE S2C 2,4 GHz</i> avec <i>XCTU</i>	25

Remerciements

Nous tenons à exprimer, dans ce paragraphe, notre gratitude envers les personnes qui ont participé à la réalisation de ce projet. Tout d'abord nos parents sans lesquels rien de tout cela n'aurait été possible. Merci aux Hofmann, aux Massa, aux Ren et aux Garde de nous avoir mis au monde. Nous pensons à la base nautique de Guerlédan qui nous a accueilli, toujours chaleureusement, et nous a offert ce cadre de travail privilégié au contact de la nature et dans des conditions d'expérimentations de terrain. On pensera aux sons, toujours reposants, de tamtam et de ukulélé de nos amis saltimbanques les hydros qui nous ont accompagné tout au long de ce projet. Nous remercions dans la foulée nos professeurs de robotique de l'ENSTA Bretagne et notamment Simon Rohou qui a mené un travail de front pour la réalisation de ces semaines de travail et dont les efforts rigoureux et l'investissement pour la filière robotique de l'école porteront, à n'en pas douter, leur fruits. Nous souhaitons enfin exprimer notre reconnaissance envers Ronan, Yvan et Dominique, nos encadrants du LABSTIC et de l'UBS. Grâce à eux, nous avons pu développer nos compétences en travaillant sur un projet stimulant. Leur disponibilité et leur expertise nous ont été précieuses tout au long de ce projet. Ils ont su nous guider, toujours avec bienveillance et humour, et nous faire confiance pour développer ce projet. Nous avons pris plaisir à travailler pour et avec eux, et leur souhaitons le meilleur pour la suite. Merci enfin aux lecteurs de ce rapport qui consacreront de leur temps à la découverte de notre travail, que nous espérons à la hauteur de leurs attentes.

Chapitre 1

Introduction

1.1 Présentation du sujet

Que ce soit dans le monde de la recherche ou dans les différents domaines dans lesquels sont utilisés les drones et tout particulièrement les AUV ou UAV, l'une des grandes difficultés rencontrées est très certainement celle du docking. En effet, bien qu'il faille tôt ou tard récupérer le drone à la fin de sa mission ou en cas de problème, cela peut se révéler dans le meilleur des cas fastidieux (dans le cas où la manœuvre est manuelle par exemple) et dans le pire des cas difficile voire dangereux pour le drone (dans le cas d'une mer agitée par exemple). Dans le cas d'une mission autonome loin du bateau, on comprend qu'il peut être très intéressant de développer une solution de docking autonome, permettant au drone de revenir vers sa base tout seul et avec précision. L'objectif de ce projet Guerlédan sera donc de développer cette solution en s'intéressant au cas d'un AUV sur le lac. Les objectifs seront les suivants :

1. Le drone doit être capable de planifier et réaliser son retour vers le dock
2. Le dock doit pouvoir communiquer sa position et son attitude en temps réel et potentiellement à longue distance pour que le drone puisse effectuer son retour peu importe la position (parfois changeante) du dock
3. Le docking doit se faire avec une précision suffisante (à la dizaine de centimètres voire mieux) pour que le drone puisse correctement entrer dans la structure

1.2 Matériel à disposition

La majorité du matériel nécessaire nous a directement été fournie. En voici une liste (non exhaustive).

1.2.1 Dock

Pour ce qui est du dock, nous avions déjà les éléments nécessaires :

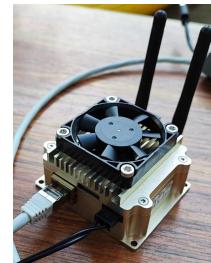
- Un récepteur GNSS *Ublox* monté sur une carte *ArduSimple*
- Une antenne radio *Xbee* pour recevoir les corrections RTK
- Une IMU *SBG Ellipse A*
- *NVIDIA Jetson Nano avec ROS*

Quant à la structure en elle-même du dock, nos encadrants ont fourni pour la deuxième semaine d'expérimentations un dock 1.1a permettant d'accueillir le drone. Puisque la mission

nécessite que le dock communique avec le drone, et ce à longue distance, nous avons également reçu deux modems *Simpulse SL2001.1b* qu'il a fallu configurer.



(a) Le dock



(b) Modem Simpulse

FIGURE 1.1 – Matériel fourni

1.2.2 Drone AUV

Pour ce qui est de l'AUV, nous avons pu travailler (seulement le temps des deux semaines sur le lac) avec le *M1800* d'*IM Solutions*.



FIGURE 1.2 – L'AUV en question, sans les capteurs supplémentaires

Le drone embarque déjà tous les capteurs nécessaires à la mission comme elle a été envisagée :

- Récepteur GNSS RTK
- IMU *SBG Ellipse D*

Sur le modèle exact qui nous a été fourni, de nombreux autres capteurs tels que des caméras et un LIDAR ont été montés ; s'ils n'ont pas été utilisés pendant la mission (dont le plus important était surtout dans un premier temps l'approche du drone vers le dock), ils pourraient en revanche se révéler utiles dans un second temps pour de prochaines expérimentations sur le projet.

1.2.3 Rover

Puisque le drone n'était disponible que peu de temps et uniquement lors des semaines à Guerlédan, nous avions besoin d'un remplacement pour pouvoir continuer à travailler et tester nos algorithmes à l'ENSTA. C'est pour cela que nous avons pu travailler avec l'*AION R11.3*, un rover à la structure software équivalente à celle du drone (tournant sous *ROS*, et gérant les commandes moteurs via une carte *Pixhawk*). On peut ainsi commander le rover exactement de

la même manière que l'AUV, c'est-à-dire dans notre cas lui envoyer des commandes en vitesses linéaire et angulaire.



FIGURE 1.3 – l'*AION Robotics R1*

Chapitre 2

Conception du dock

2.1 Mise en place d'une base RTK

Le positionnement GPS acquis par les capteurs à disposition confère une précision métrique, les signaux GPS étant perturbés par l'atmosphère. Or, il faut de meilleures performances pour espérer docker le drône dans un environnement confiné et restreint. L'idée était donc de mettre en place un système de positionnement plus précis (centimétrique dans l'idéal) en utilisant le dispositif du *Real Time Kinematic* (RTK). Pour cela, il faut installer une base GPS fixe de position connue précisément. Elle servira de médiateur en diffusant des corrections de positionnement aux systèmes GPS connectés. L'installation est donc en deux parties, la première, à terre, l'autre, directement sur le dock.

2.1.1 Capteurs à disposition

L'acquisition de signaux GPS se fait grâce à deux modules *ArduSimple* équipés de processeurs *u-blox ZED-F9P*.



FIGURE 2.1 – Module GPS utilisé dans le projet : *ArduSimple* + *u-blox ZED-F9P* + *Xbee*

Les deux modules communiquent grâce à leur antenne radio *Xbee S2C 2,4 GHz*. La base dispose d'une antenne GPS qui permet une meilleure acquisition des signaux. Il est possible d'utiliser une *Raspberry* pour configurer la base GPS en lui conférant une interface graphique. Un guide de configuration des capteurs est disponible en annexe.

2.1.2 Installation de la base

L'installation de la base peut se faire de deux manières différentes.

1. En utilisant uniquement le module *u-blox*. Il s'agit de la méthode la plus simple. Si la position précise de la base est connue, on configure le module en *Fixed Mode* et les corrections seront directement disponibles. Sinon on le configure en mode *Survey-In* ce qui va lancer l'acquisition de trames GPS provenants de satellites jusqu'à ce que la position soit connue avec une précision suffisante. Cela peut prendre une dizaine d'heures pour une précision centimétrique. La configuration du module se fait avec le logiciel *XCTU*(sous Windows). Dans tous les cas, il faut que la base soit alimentée en permanence pour ne pas perdre la position acquise et que l'antenne soit bien dégagée pour une meilleure réception des signaux.
2. En utilisant le module *u-blox* et une *Raspberry*. Cette solution est issue de la documentation fournie par *Centipède* pour la mise en place d'une base RTK déclarée sur leur réseau. Dans ce cas, aucun *TimeMode* n'est spécifié pour le module GPS. La *Raspberry* est flashée et configurée pour enregistrer les trames GPS reçues par le *u-blox* puis faire une sauvegarde par jour (vers 4h du matin). L'idée est ensuite de récupérer une sauvegarde de 24 heures, de l'envoyer à *IGN* qui fournit un service de triangulation. *IGN* renvoie ensuite la position précise calculée à partir des données. A partir d'ici, il n'est plus nécessaire de suivre les instructions de *Centipède*¹. Il faut en revanche passer le module *u-blox* en mode *Fixed Mode* et renseigner la position déterminée précédemment. Avec la *Raspberry*, on peut accéder aux données directement en se connectant en *ssh* ou alors en se connectant à l'interface web graphique de la carte.

Une fois l'installation terminée, la base RTK diffuse ses corrections de positionnement à l'aide de l'antenne radio *Xbee*. Le statut de la base est indiqué par les LEDs de la carte *Ardu-Simple*.

- GPS FIX : **OFF** lorsqu'il n'y a pas de fix, **une impulsion par seconde** lorsque la position est valide.
- NO RTK : **OFF** lorsque RTK fixe, **clignotant** lors de la réception de données RTCM, **ON** lorsqu'aucune correction n'a lieu.
- GPS→XBEE : **clignotant** lorsque le module GPS envoie des données à l'antenne radio.

2.1.3 Configuration du dock

La configuration du dock est plus courte. Il suffit d'alimenter le module *u-blox* et de le connecter à une antenne. La configuration précise est donnée en annexe. Une fois alimenté, le module va acquérir les signaux satellites et les corrections RTK envoyées par la base. Si jamais aucune correction n'est reçue, le *u-blox* va fournir une position GPS classique. Les LEDs fournissent les indications utiles :

- GPS FIX : **OFF** lorsqu'il n'y a pas de fix, **une impulsion par seconde** lorsque la position est valide.
- NO RTK : **OFF** lorsque RTK fixe, **clignotant** lors de la réception de données RTCM, **ON** lorsqu'aucune correction n'a lieu.
- XBEE → GPS : **clignotant** lorsque le module GPS reçoit des données de l'antenne radio.

1. Le but n'est pas de déclarer la base sur leur réseau.

2.1.4 Un mot sur les antennes

Les antennes de communication des deux modules GPS sont des *XBEE S2C 2,4 GHz*. Elles doivent être correctement paramétrées pour pouvoir échanger. Les spécifications sont données en annexe.

2.1.5 Test, difficultés et résultats

La mise en place de la base RTK n'a malheureusement pas été fructueuse. Malgré de nombreuses tentatives tout au long de ce projet, nous n'avons jamais réussi à obtenir du module de la base qu'il communique les corrections *RTK* sur le canal radio dédié. La première difficulté que nous avons rencontrée était que les modules *base* et *dock* avaient été inversés. En effet, le module *dock* était configuré pour être une base RTK et inversement. On pourrait penser que cela n'est pas problématique mais *ArduSimple* précise dans sa documentation que les modules sont vendus pré-configurés pour être soit une base soit un rover. De fait, nous avons dû inverser les modules et vérifier leur configuration. Il se trouve que les antennes n'étaient pas bien configurées non plus et qu'il nous était impossible de les régler avec *XCTU* jusqu'à ce que nous trouvions l'astuce du court-circuit (cf. annexe sur la configuration des capteurs). La bonne configuration des modules a donc pris plus de temps que prévu et de multiples tests soldés par des échecs. Nous avons également suivi un tutoriel fait par l'école pour la configuration des modules, mais celui-ci contenait une petite erreur que nous avons mis beaucoup de temps à repérer (sur la configuration des protocoles d'entrée de certains ports qui empêchait de transmettre les corrections). Une fois que nous avons trouvé la bonne configuration des modules, nous avons pu constater que la base était capable de déterminer sa position à 1 mètre près en quelques heures. Cependant, nous n'avons jamais réussi à obtenir des corrections RTK sur le canal radio dédié. Même avec un test en mode *Fixed* (où l'on renseigne la position manuellement), il nous a été impossible de faire communiquer les deux modules. Pourtant, nous arrivons bien à faire communiquer les modules manuellement en opérant directement sur les antennes qui peuvent se détecter à plusieurs dizaines de mètres. Nous pensons que cela est dû à un réglage trop exigeant du mode d'acquisition et des conditions nécessaires à l'émission des corrections RTK. Toutefois, le module GPS du dock était assez précis, même tout seul, pour que le bateau/rover puisse s'approcher et se positionner dans un rayon de 1,5 mètres autour du dock. Une piste d'amélioration serait de mettre à jour les *u-blox* avec *U-Center* pour s'assurer de partir sur de bonnes bases et puis de recommencer la configuration en tenant compte de notre expérience sur le sujet.

2.2 Mise en place du dock

L'ensemble de l'électronique embarquée du dock tient dans une boîte étanche qui laisse passer le câble de l'IMU. En effet, pour pouvoir s'assurer du bon fonctionnement cette dernière sa calibration a été réalisée grâce au logiciel *SBG Center* tout d'abord en la plaçant à l'intérieur de la boîte avec le reste de l'électronique. Or en observant les valeurs données et en essayant la calibration il apparaissait que les appareils aux alentours produisaient des perturbations magnétiques bien trop importantes pour que le cap mesuré soit cohérent. C'est pour cela qu'il a été décidé de plutôt placer l'IMU en dehors de la boîte pour pouvoir qu'elle fonctionne correctement (cf 2.2a).



(a) Le bloc entier à monter sur la structure (droite : IMU)



(b) Intérieur du dock avec toute l'électronique

FIGURE 2.2 – Mise en place du dock

2.3 Communication avec le reste du système

Pour rappel, le dock doit pouvoir envoyer sa position et son attitude au drone pour que ce dernier puisse planifier son retour, et ce potentiellement à longue distance. C'est pour cela que dans le cadre de ce projet nous utilisons des modems *Simpulse*. S'il est possible de communiquer entre noeuds ROS, ce n'est pas l'approche qui sera finalement utilisée : outre les difficultés dues à ROS (qui n'existent pas en ROS2), on préférera aussi un protocole plus simple et plus universel pour que la communication s'adapte facilement sur d'autres OS ou appareils (par exemple, si le dock et le drone ont des versions différentes de ROS, la communication directe n'est pas possible). La communication se fait donc en UDP (quoi qu'on aurait pu choisir le TCP qui permet une garantie sur la bonne réception, cela n'était pas absolument nécessaire) ; on veut donc envoyer des trames à intervalle régulier contenant les informations nécessaires que sont la position GPS et l'attitude du dock. On peut donc définir un format simple permettant de véhiculer ces informations :

`${latitude}, ${longitude}; ${roll}, ${pitch}, ${yaw}`

On précise que l'utilisation des angles d'Euler, faciles à interpréter lors du débogage, est justifiée par le fait que dans notre cas on ne risque pas le *Gimbal lock* vus les angles relativement faibles sur le pitch et le roll.

La communication se fait donc assez facilement de cette manière, après avoir bien configuré les adresses du dock et du drone.

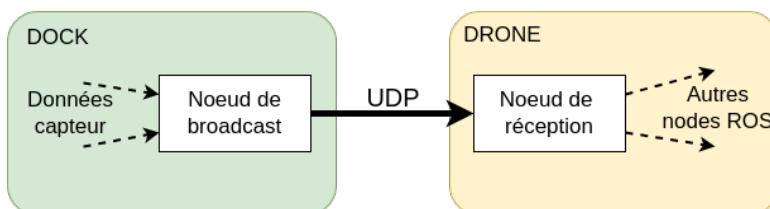


FIGURE 2.3 – La communication dock → drone

La trame est ensuite interprétée et convertie en messages ROS publiés sur des topics et utilisés par le reste du système.

Chapitre 3

Stratégie d'approche de docking

3.1 Filtre de Kalman

Le modèle cinématique choisi pour le robot est le suivant :

$$\begin{aligned} \dot{\mathbf{x}} &= B\mathbf{u} \\ \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \end{pmatrix} &= \begin{pmatrix} \cos(\varphi) \cos(\psi) & 0 \\ \cos(\varphi) \sin(\psi) & 0 \\ -\sin(\varphi) & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \end{aligned} \quad (3.1)$$

où (x, y, z) représente les coordonnées du robot dans le repère ENU, φ représente l'assiette et ψ le cap.

Ce modèle est issu du modèle de Dubins mais où contrairement à ce dernier, on suppose que le véhicule est commandé en vitesse. Cette décision a été prise parce que le rover et le bateau sont commandés à l'aide d'un message de type cmd_vel, qui spécifie à la fois la vitesse linéaire et la vitesse angulaire.

Le modèle cinématique étant choisi, le modèle d'observation peut être défini. À l'heure actuelle, seules les données de position GPS (x, y) et de cap ψ sont intégrées par le filtre.

$$\begin{aligned} \mathbf{y} &= C\mathbf{x} \\ \begin{pmatrix} x_{GPS} \\ y_{GPS} \\ \psi_{IMU} \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ \psi \end{pmatrix} \end{aligned} \quad (3.2)$$

Les matrices de covariance ont été déterminées à posteriori et sont évidemment à ajuster plus tard. Notons Q la matrice de covariance du bruit d'évolution et R la matrice de covariance du bruit de mesure, nous avons pris :

$$\left\{ \begin{array}{l} Q = \begin{pmatrix} .5 & 0 & 0 \\ 0 & .5 & 0 \\ 0 & 0 & .5 \end{pmatrix} \\ R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & .17 \end{pmatrix} \end{array} \right.$$

Le filtre de Kalman s'appliquant sur un système discret, il faut discréteriser (3.1). Ainsi au premier ordre nous avons

$$\begin{aligned}\mathbf{x}(t+dt) &= \mathbf{x}(t) + \int_t^{t+dt} \dot{\mathbf{x}}(\tau) d\tau \\ &= \mathbf{x}(t) + \int_t^{t+dt} B\mathbf{u}(\tau) d\tau \\ &= \mathbf{x}(t) + dtB\mathbf{u}(t)\end{aligned}$$

La relation (3.2) peut quant à elle être directement utilisée. Il faut également discréteriser les matrices de covariances. On montre que

$$\begin{cases} \Gamma_{\alpha_k} = dt\Gamma_\alpha \\ \Gamma_{\beta_k} = \frac{1}{dt}\Gamma_\beta \end{cases}$$

3.2 Guidage par champ de potentiel artificiel

Le guidage par champ de potentiel artificiel est un principe largement utilisé dans le domaine de la robotique mobile pour la navigation autonome des robots. L'idée fondamentale est de créer un champ de potentiel artificiel dans l'environnement dans lequel se déplace le robot, de telle sorte que le robot soit attiré vers sa destination et repoussé des obstacles.

Soit \mathbf{v} le vecteur vitesse du robot et M sa position. Formellement, le champ de potentiel peut être modélisé par une fonction $U : \mathbb{R}^2 \rightarrow \mathbb{R}$. Le robot devra ensuite être contrôlé de sorte que :

$$\mathbf{v}(M) = -\vec{\nabla}U(M)$$

3.2.1 Champ de potentiel retenu

Idée générale

L'idée est d'adapter le champ de potentiel suivant la position M du robot. Il y a deux cas possibles :

- Le robot se trouve dans le demi-plan devant le dock (cas 1)
- Le robot se trouve dans le demi-plan derrière le dock (cas 2)

Dans le premier cas le robot va aller sur la droite passant par le dock et dirigé dans le même sens que le dock. Dans le second cas le robot va d'abord aller dans le demi plan en face du robot en suivant un champ de vecteur uniforme tout en faisant attention à pas rentrer dans le dock - ce dernier devenant un point répulsif -. Puis en étant assez éloigné du demi plan en face du dock, le robot se retrouvera dans le premier cas et agira en conséquence.

Mise en équation

Soit M la position du robot et M' la position du dock. En s'inspirant des champs de potentiels physique comme le champ électrostatique, le champ de potentiel engendré par un point répulsif peut être défini ainsi :

$$U_1(M) = \frac{1}{\|MM'\|^\alpha}$$

où $\alpha \in \mathbb{R}$ est un coefficient déterminant la répulsivité du point.

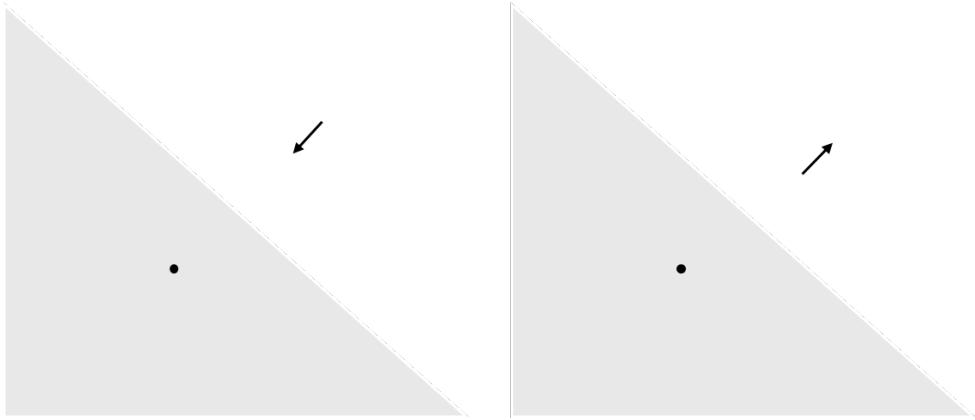


FIGURE 3.1 – La flèche indique la position et l'orientation du dock et le point représente la position du robot (a) À gauche le robot se trouve dans le demi-plan en face du robot (b) À droite le robot se trouve dans le demi-plan derrière le robot

Le champ de potentiel d'une ligne attractive peut quant à lui être défini par :

$$U_2(M) = \overrightarrow{M'M}^T \cdot \vec{n} \cdot \vec{n}^T \overrightarrow{M'M}$$

Ce champ permet d'attirer le véhicule sur la ligne. Le véhicule arrive perpendiculairement à cette dernière et s'arrête ce qui signifie que le robot n'est pas bien orienté.

Enfin le champ de potentiel issu d'un champ de vecteur vitesse uniforme \vec{v}_d peut être défini par :

$$U_3(M) = -\vec{v}_d^T \overrightarrow{OM}$$

Les champs de vitesses correspondant à chacun des champs de potentiels sont alors :

$$\begin{aligned}\vec{v}_1 &= \frac{\overrightarrow{M'M}}{\|\overrightarrow{MM'}\|^{\alpha+2}} \\ \vec{v}_2 &= -2\vec{n} \cdot \vec{n}^T \cdot \overrightarrow{M'M} \\ \vec{v}_3 &= \vec{v}_d\end{aligned}$$

Dans le premier cas, il faut une ligne attractive pour positionner le robot sur la ligne. Cependant, s'il n'y avait que ce champ, le véhicule arriverait perpendiculairement à la ligne et s'arrêterait. Il faut donc un champ uniforme pour l'orienter dans la direction voulue et pour qu'il avance vers le dock. Le champ de potentiel total est alors égal à une somme des deux champs pondérée par des coefficients $\lambda_{1,i}$:

$$U_{\text{tot}}^1(M) = \sum_{i=2}^3 \lambda_{1,i} U_i(M)$$

Dans le second cas, on a besoin d'un champ uniforme et d'un champ généré par un point répulsif qui correspond au dock :

$$U_{\text{tot}}^2(M) = \lambda_{2,1} U_1(M) + \lambda_{2,2} U_3(M)$$

En différenciant les potentiels, on obtient les champs de vitesses voulus.

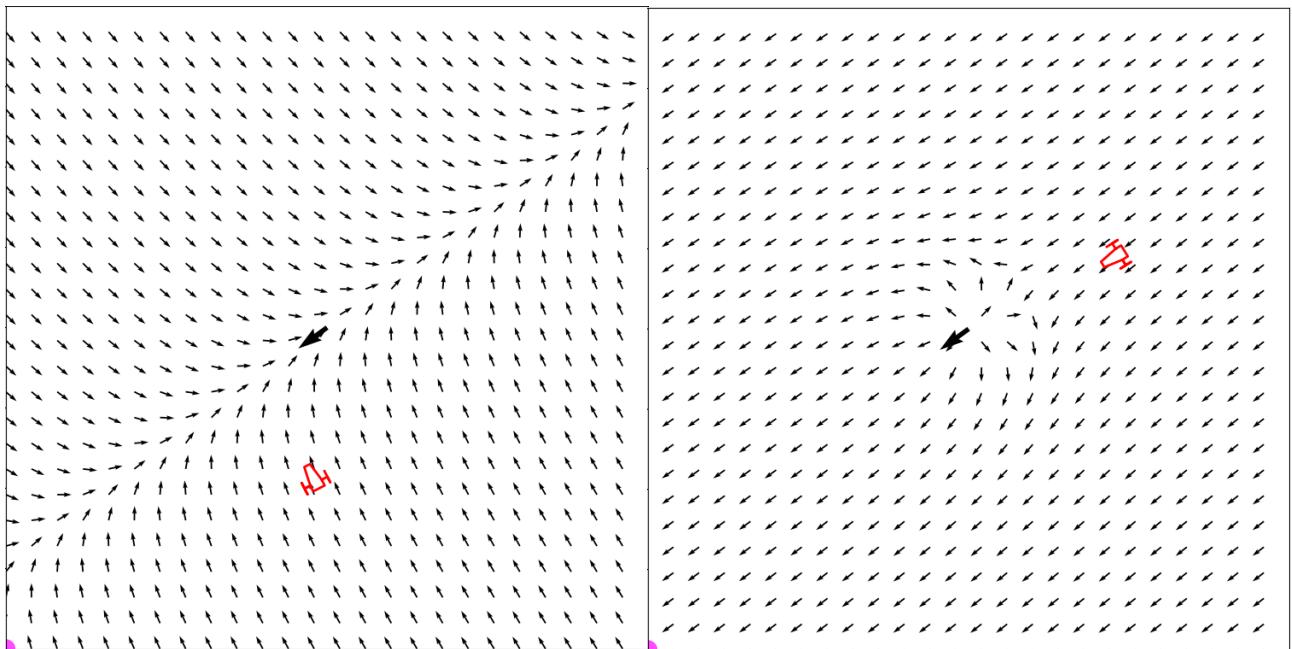


FIGURE 3.2 – (a) Champ de vecteurs dans le premier cas. (b) Champ de vecteurs dans le second cas.

3.3 Algorithme

Chapitre 4

Architecture logicielle

4.1 ROS

Afin d'avoir un système efficace quand plusieurs parties doivent communiquer entre eux, il est essentiel d'assurer une architecture logicielle claire et performante. Plusieurs choix s'offraient à nous à ce sujet, étant équipés sur la plupart des éléments de ce système de carte NVIDIA Jetson, des cartes permettant moulte options à ce sujet. En l'occurrence, nous avons opté pour une architecture ROS, middleware très utilisé dans notre domaine qui est la robotique.

4.1.1 Présentation générale

ROS (Robot Operating System) est un framework open source spécialement conçu pour le développement et la gestion de systèmes robotiques. Son architecture modulaire et distribuée offre aux développeurs une base flexible et évolutive pour la création de systèmes robotiques sophistiqués.

Suivant un modèle de messagerie publish-subscribe, ROS facilite la communication transparente entre les différentes composantes d'un système robotique, favorisant l'échange de données et de commandes. Prenant en charge plusieurs langages de programmation, ce framework propose une gamme complète d'outils et de bibliothèques qui simplifient le processus de développement.

L'infrastructure de communication dans ROS repose sur deux concepts fondamentaux : les nœuds et les topics. Les nœuds sont des modules logiciels individuels, chacun réalisant une tâche spécifique au sein du système robotique. Les nœuds, agissant comme les éléments fondamentaux d'une application ROS, interagissent en échangeant des messages. D'autre part, les topics servent de canaux de communication utilisés par les nœuds pour partager des messages au sein du framework ROS. Suivant le modèle de communication publish-subscribe, les nœuds générant des données publient des messages sur un topic spécifique, tandis que les nœuds intéressés par ces données souscrivent au topic correspondant pour recevoir les messages.

4.1.2 Limitations

Contrairement à son successeur ROS2, ROS connaît des limitations quand il s'agit de le mettre en place sur un système composé de plusieurs parties. En effet, il est plus difficile de faire transiter les données entre chaque machine sur ROS que ROS2. Alors que sur ROS2, être sur le même réseau suffit pour partager les données entre les machines, ce n'est pas le cas sur ROS, ce qui complique la gestion de l'architecture logicielle.

Malheureusement, les limitations logicielles ne nous permettaient pas de passer sur ROS2, étant équipés de cartes NVIDIA Jetson flashées avec un équivalent de Ubuntu18, qui ne supporte pas ROS2.

De plus, tout est codé en *python 2.7*, ce qui est également limitant dans nos capacités.

4.2 Architecture du projet

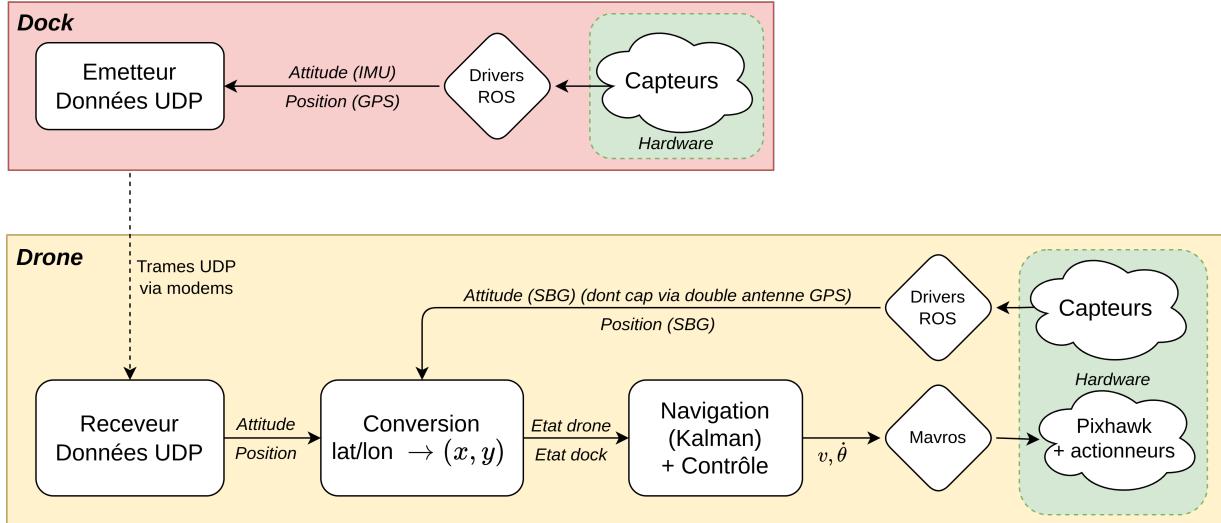


FIGURE 4.1 – Architecture logicielle globale du système de docking

Ici, on peut constater qu'un noeud est dédié à la conversion de coordonnées GPS (latitude,longitude en degrés) vers un système de coordonnées x,y (en mètres) qui correspond au repère ENU (East-North-Up). Cette conversion est effectué grâce à une projection appelée Lambert93 et la librairie *Pyproj*. Ce repère est plus avantageux pour nos algorithmes de contrôle, étant donné qu'il colle plus au contexte utilisé lors de la mise en place et des tests des algorithmes.

De plus, le lien entre le contrôle et les commandes actionneurs n'est pas géré directement par nous, mais par l'intermédiaire MAVLink via mavros qui sert de pont et d'interpréteur entre les commandes en vitesse du véhicule et les commandes moteurs.

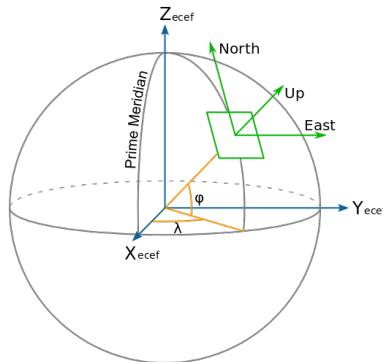


FIGURE 4.2 – Repère ENU

Chapitre 5

Essais à Guerlédan

5.1 Première semaine

La première semaine d'expérimentations au lac de Guerlédan, du 9 au 13 octobre 2023, fut davantage une semaine de découverte du projet et permit une première approche à la résolution de notre problème. Durant cette semaine, nous avons rencontré nos encadrants et pris en main le matériel que nous avions à notre disposition, c'est-à-dire principalement les composants de ce qui sera plus tard notre dock, que nous avons conçu durant cette semaine. Ainsi, à l'aide des composants décrits précédemment, nous avons été capable de créer la boîte qui contient les capteurs nécessaires au bon fonctionnement du dock.

D'autre part, nous travaillons en parallèle sur la mise en place de la correction RTK en configurant les différentes cartes et en prenant des mesures nécessaires au calcul de la correction. Nous avons pour cela commencé à suivre un tutoriel écrit par Centipède pour la mise en place d'une base RTK sur leur réseau. Nous avons réussi à obtenir des logs de données GPS que nous avons envoyé à l'IGN pour obtenir une position précise de notre base. Malheureusement, cela a pris plus de temps que prévu et nous n'avons pas pu exploiter ces données.

Enfin, une première version simplifiée de l'architecture logicielle a commencé à être implémentée.

La principale problématique de cette semaine, qui est la même que celle rencontrée durant la période séparant cette semaine de la seconde, est l'impossibilité d'expérimenter directement sur l'AUV qui est le drone principal de notre sujet. En effet, l'avancée de notre projet ne nous permettait pas la sécurité de pouvoir tester nos algorithmes de manière sereine sur le drone et nous avons également rencontré quelques difficultés avec l'utilisation de l'interface MAVROS, qui fait le lien entre notre architecture ROS et MAVLink. Le principal problème résidait dans le fait qu'il semblait que MAVLink avait en mémoire des waypoints GPS que le drone tentait de rejoindre en priorité avant de prendre en compte nos commandes. Cela empêchait toute exploitation du drone en mode automatique, ce qui est nécessaire au bon fonctionnement de nos algorithmes.

Heureusement ce problème avait tout le temps d'être réglé, étant donné que nous n'utiliserions pas le drone d'ici à la seconde semaine d'expérimentation à Guerlédan, en février 2024. De plus, nous avons décidé de concentrer nos efforts sur la mise en place et la conversion de nos algorithmes pour la mise en place sur le Rover qui nous sert de remplacement au drone et qui nous sert de moyen d'expérimentations à l'école.

5.2 Deuxième semaine

La seconde semaine d'expérimentations, qui eut lieu du 5 au 9 février 2024, fut bien davantage une semaine d'expérimentation que la première. En effet, forts du travail effectué entre les deux semaines, nous avons été capables de mettre en oeuvre et tester nos algorithmes sur le drone principal.

La première étape préliminaire à cela fut de convertir nos codes, conçus pour fonctionner sur le rover, pour qu'ils puissent se mettre correctement en oeuvre sur le drone, qui, on le rappelle, n'est pas équipé des mêmes capteurs et architecture logicielle des drivers. Il y a également des fonctionnalités supplémentaires sur le drone qui permettent plus de précision sur l'acquisition de l'état du drone, notamment l'obtention du *True Heading*, sur lequel nous reviendrons plus tard, qui permet l'obtention du cap avec une plus grande fiabilité qu'en utilisant uniquement la centrale inertielle.

Une des principales problématiques rencontrées durant la semaine réside dans la difficulté à bien calibrer les centrales inertielles. En effet, les centrales inertielles SBG nécessitent une étape de calibration afin qu'elles compensent le champ magnétique disturbé pour calculer son attitude. Malgré de nombreux essais dans de nombreuses configurations, nous n'avons jamais réussi à bien configurer la centrale pour que la sortie de son filtre de Kalman concernant le cap ne diverge pas, surtout pour le dock. Si le dock reste trop longtemps immobile, sans perturbations, son cap se met à diverger rapidement. Ce problème reste un point d'amélioration essentiel car il est une des raisons principales à l'absence d'expérimentations avec le vrai dock.

Cependant, nous avons développé des alternatives permettant tout de même de mettre en oeuvre nos algorithmes en contournant le problème. Lorsque nous lancons le système, nous avons le choix entre utiliser le vrai dock ou un dock simulé. Le dock simulé repose soit sur des logs d'essais précédents soit sur une position et orientation désirée précisées au lancement. En faisant cela, nous avons été capable de tester et prouver la robustesse de nos algorithmes, même sans présence du vrai dock.

L'obtention de corrections RTK a posé quelques problèmes. Nous avons pris beaucoup de recul sur le sens des tâches effectuées précédemment à cette fin et nous avons réalisé que beaucoup de réglages avaient été faits sans vraiment les comprendre et surtout qui compliquaient la tâche voire n'étaient même pas utiles (l'utilisation de la Raspberry par exemple). Nous avons donc repris proprement la configuration en résolvant de nouveaux problèmes un à un mais des problèmes de surtensions liés à l'alimentation de Canopée ont fait sauter plusieurs fois le réseau électrique. Ceci a interrompu deux jours de suite l'acquisition des données GPS nécessaires pour obtenir la précision souhaitée. Nous avons donc perdu beaucoup de temps sur une semaine comme cela ce qui ne nous a pas permis de tester notre installation avec le bateau. Nous avons quand même pu travailler correctement mais forcément avec une précision moindre que celle que nous aurions pu obtenir avec une correction RTK.

Chapitre 6

Conclusion

6.1 Résultats

6.1.1 Rover

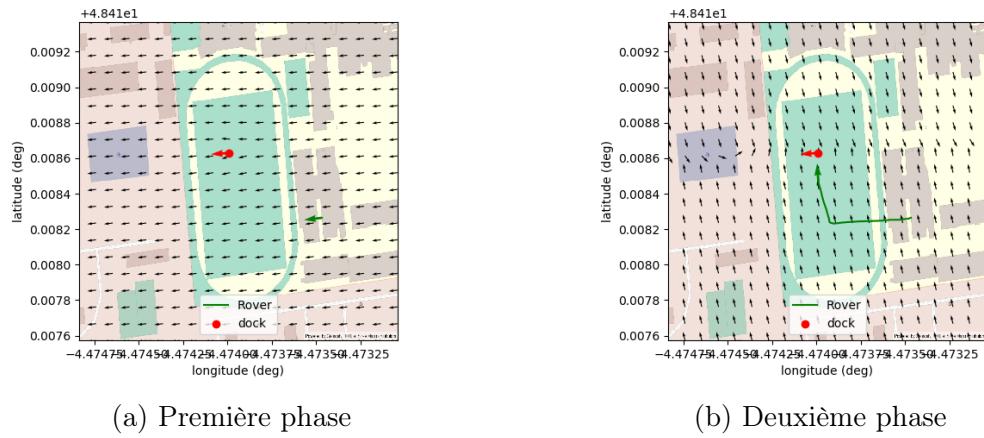


FIGURE 6.1 – Essai de docking du rover sur le stade de l'ENSTA

6.1.2 Drone

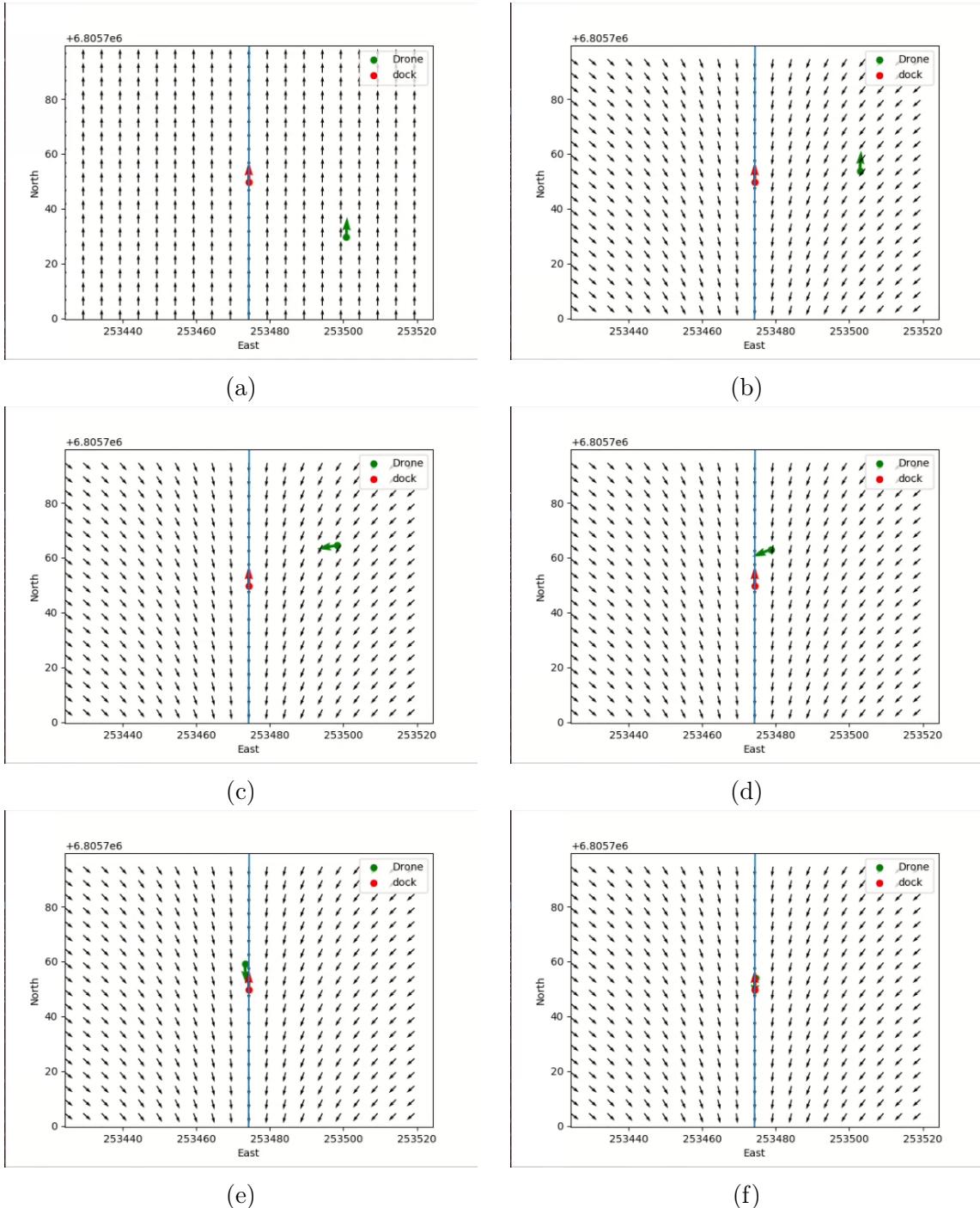


FIGURE 6.2 – Essai de docking (avec dock virtuel mais drone réel) sur le lac de Guerlédan avec du courant

6.1.3 Synthèse

Le projet se finit sur une note plutôt positive : en effet tous les tests finalement réalisés semblent démontrer que le système de docking fonctionne, et notamment le système de guidage permettant l'approche, que ce soit sur le rover ou sur le drone ; l'ombre qui reste au tableau est celle du manque d'expérimentations avec toute la structure du dock réel (qui pour rappel

n'a pas pu être mis à l'eau le dernier jour à cause de la météo capricieuse). En effet, si le drone semblait effectuer sa mission sans problème, on peut en revanche s'attendre à ce que les choses soient plus compliquées pour ce qui est de se docker précisément dans la structure, surtout sans RTK pleinement fonctionnel.

6.2 Perspectives

Ce projet contient encore beaucoup d'aspects très intéressants à aborder. Un des premiers aspects à approfondir est de faire plus de tests avec le dock.

De plus, il peut être intéressant d'enrichir la gamme de capteurs utilisés notamment pour l'étape du docking en lui-même qui nécessite de la précision. Une perspective d'évolution serait d'utiliser par exemple un LIDAR ou une caméra pour se docker bien plus précisément qu'avec uniquement le GPS et la centrale inertuelle.

Surtout, il serait bon de pouvoir finir la mise en place de la correction RTK pour obtenir une précision centimétrique et exploité le plein potentiel de notre travail. Maintenant que nous avons une bonne idée des principaux obstacles à surmonter, il serait sûrement plus facile d'avancer en ce sens.

Enfin, il pourrait également être envisageable d'intégrer des algorithmes d'évitement d'obstacles à l'aide des nouveaux capteurs utilisés. L'ajout de cette fonctionnalité serait de plus compatible avec la façon dont on procède, étant donné que l'on travaille avec des champs de vecteurs pour le contrôle, ce qui est avantageux pour l'évitement d'obstacles.

Chapitre 7

Annexes

7.1 Configuration des modules *u-blox* avec *U-Center*

Pour configurer les modules GNSS *u-blox*, il faut utiliser le logiciel *U-Center* sous **Windows**. La carte doit être connectée par son port série **Power+GPS** à l'ordinateur.

7.1.1 Configuration de la base

Il faut d'abord se connecter au bon port *COM* et choisir le bon *baudrate*, normalement 9600. Pour ouvrir la fenêtre de configuration, cliquer sur *Configuration View*. Il faut ensuite renseigner les différents réglages suivants en appuyant bien sur **Send** à chaque modification :

- Onglet *Msg* : cocher la case *USB-On* et *UART1-On* pour les messages suivants :
 - F5-05 RTCM3.3 1005
 - F5-05 RTCM3.3 1077
 - F5-05 RTCM3.3 1087
 - F5-05 RTCM3.3 1127
 - F5-05 RTCM3.3 1230
 - 01-13 NAV-HPOSECEF
 - 01-02 NAV-POSLLH
 - 02-15 RXM-RAWX
 - 02-13 RXM-SFRBX
 - F5-4A RTCM3.3 1074
 - F5-4D RTCM3.3 1077
 - F5-54 RTCM3.3 1084
 - F5-57 RTCM3.3 1087
 - F5-5E RTCM3.3 1094
 - F5-61 RTCM3.3 1097
 - F5-7C RTCM3.3 1124
 - F5-7F RTCM3.3 1127
- Onglet *NMEA* : cocher *High precision mode* pour le réglage *CFG-NMEA(DATA2)*
- Onglet *TIMEMODE3* : choisir le mode *Survey-In* pour avoir un temps d'acquisition de la position et une précision minimale ; ou le mode *Fixed Mode* si la position précise est déjà connue. En mode d'acquisition longue, il faut prévoir plusieurs heures avant d'avoir une précision décimétrique.
- Onglet *PRT* :
 - *Target* : 1-UART1, *Protocol in* : none, *Protocol out* : 0+1+5-UBX+NMEA+RTCM3,

Baudrate : 115200, Databits : 0, Stopbits : 1, Parity : None, Bit Order : LSB First

- *Target : 3-USB, Protocol in : 0+1+5-UBX+NMEA+RTCM3, Protocol out : 0+1+5-UBX+NMEA+RTCM3*
- Onglet *CFG* : sélectionner toute la rubrique *Devices*, s'assurer que *Save current configuration* est coché, cliquer sur *Send*.

7.1.2 Configuration du dock

Il faut d'abord se connecter au bon port *COM* et choisir le bon *baudrate*, normalement 9600. Pour ouvrir la fenêtre de configuration, cliquer sur *Configuration View*. Il faut ensuite renseigner les différents réglages suivants en appuyant bien sur **Send** à chaque modification :

- La configuration des messages du rover est optionnelle. Onglet *Msg* : cocher la case *USB-On* pour les messages :
 - *F5-05 RTCM3.3 1005*
 - *F1-00 PUBX00*
 - *F1-03 PUBX03*
 - *F1-04 PUBX04*

Puis cocher aussi la case *UART1-On* pour les messages :

- *01-13 NAV-HPOSECEF*
- *01-02 NAV-POSLLH*
- *F0-0A NMEA GxDTM*
- *F0-00 NMEA GxGGA*
- *F0-01 NMEA GxGLL*
- *F0-02 NMEA GxGSA*
- *F0-03 NMEA GxGSV*
- Onglet *NMEA* : cocher *High precision mode* pour le réglage *CFG-NMEA(DATA2)*.
- Onglet *TIMEMODE3* : mode *Disabled*.
- Onglet *PRT* :
 - *Target : 1-UART1, Protocol in : none, Protocol out : 0+1+5-UBX+NMEA+RTCM3.*
 - *Target : 3-USB, Protocol in : 0+1+5-UBX+NMEA+RTCM3, Protocol out : 0+1+5-UBX+NMEA+RTCM3.*
- Onglet *CFG* : sélectionner toute la rubrique *Devices*, s'assurer que *Save current configuration* est coché, cliquer sur *Send*.

7.1.3 Informations sur les LEDs

La carte comprend 7 voyants d'état, qui indiquent que :

- ALIMENTATION : la carte simpleRTK2B est alimentée.
- GPS FIX : la configuration par défaut d'u-blox pour la broche TIMEPULSE est utilisée : OFF lorsqu'il n'y a pas de fix, 1 impulsion par seconde lorsque la position est valide. Cette LED a une couleur spéciale VERT.
- NO RTK : la configuration par défaut d'u-blox pour la broche RTK_STAT est utilisée : OFF lorsque RTK fixe, clignotant lors de la réception de données RTCM, ON lorsqu'aucune correction n'est effectuée. Cette LED a une couleur spéciale ROUGE.
- XBEE>GPS : Le XBEE radio reçoit des données par voie hertzienne et les envoie au ZED-F9P.
- GPS>XBEE : Le ZED-F9P produit des données que le XBEE radio reçoit et envoie par voie hertzienne.
- 5V IN/OUT : vous indiquera s'il y a une tension sur cette broche.
- IOREF : vous indiquera si la broche IOREF est activée.

7.2 Configuration des antennes *XBEE S2C 2,4 GHz* avec *XCTU*

La configuration des antennes *XBEE S2C 2,4 GHz* se fait avec le logiciel *XCTU* sous **Windows**. Il faut d'abord connecter le module par son port série *Power+Xbee* à l'ordinateur. En cas de problème avec le logiciel (impossibilité de détecter le module ou de lire ses paramètres), il faut court-circuiter les pins *RESET N* et *GND* du module pendant la configuration. Cela devrait résoudre le problème. On clique sur *Add a radio module* puis on choisit le bon port *COM* et le bon *baudrate*, normalement 115200. On clique ensuite sur *Finish* pour valider. Une fois que le module est ajouté, on clique sur les paramètres pour le configurer. Pour les deux antennes, on doit choisir le même *PAN ID* et le même *Channel* pour qu'elles puissent communiquer. Il faut également faire correspondre les *Destination Address High* et *Destination Address Low* pour que les antennes puissent communiquer entre elles. L'idée est que le *DH* de l'un soit le *SH* de l'autre, et que le *DL* de l'un soit le *SL* de l'autre, et inversement. On règle aussi *RP* à 5. La seule chose qui va différer est le *CE* qui doit être *Enabled* pour la base et *Disabled* pour le dock.

Bibliographie