INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Université de Toulouse

INSTITUT NATIONAL DES SCIENCES APPLIQUEES

TOULOUSE

Département de Génie Électrique & Informatique

PROJET DE FIN D'ETUDES

Spécialité : I

Filière : ISS

Développement embarqué

Fusion de données

Auteur :

LEON Florian

Entreprise :

SII

Référent INSA :

GUERMOUCHE Nawal

Responsable du stage :

SABORET Romain

Année 2021 - 2022

RÉSUMÉ

Ce rapport de stage présente mes réalisations au cours des six mois passés à SII dans le lab Embarqué. Ce dernier fait partie du département Recherche et Développement de l'entreprise et s'attelle au développement logiciel embarqué pour de la recherche mais également pour le compte de Continental, le principal client du lab.

J'interviens sur la partie R&D du lab. Le but de cette partie est de développer un robot capable de se déplacer à l'aide de capteurs Ultra Wideband (UWB) et d'un capteur à effet hall. Le robot doit se diriger automatiquement vers une base qui comporte un aimant et une balise UWB.

Dans le but de fiabiliser les déplacements du robot, il a été décidé de réaliser une fusion de données entre un accéléromètre et un gyroscope pour obtenir une distance de déplacement ainsi qu'un angle parcouru. Ce rapport décrit donc mes recherches et travaux dans le but de fournir le meilleur retour possible.

Cependant, les tests ont vite montré les limites de cette solution. En effet, la mesure de distance s'avère trop dépendante de son environnement et des perturbations tout comme la mesure d'angle. A ces problèmes, une solution a été trouvée pour la mesure de l'angle, par le rajout d'un magnétomètre. Ce dernier, en plus d'apporter une grande fiabilité à la mesure, permet également d'intervenir dans la dernière phase d'approche en remplaçant le capteur a effet hall. Ce faisant, il permet d'augmenter la portée du code de la dernière phase d'approche. Aucune solution viable n'a cependant été trouvée pour la mesure de distance.

REMERCIEMENTS

Je tiens tout d'abord à remercier Romain SABORET et Clément PENE pour leur accueil et la confiance qu'ils m'ont accordés au cours de ce stage. Leurs conseils et les aides qu'ils m'ont apportés ont été précieux tout au long de mon stage.

Je tiens ensuite à remercier l'ensemble de l'équipe grâce à qui ce stage s'est aussi bien déroulé. Ils ont contribué à rendre mon stage enrichissant et motivant.

Je tiens enfin à remercier les personnes qui m'ont aidé à relire et corriger ce rapport, grâce à leurs recommandations.

TABLE DES MATIÈRES

GLOSSAIRE

Crate : Librairie de code en open source

GPIO : General Purpose Input/Output

HAL : Hardware Abstraction Layer

I2C: Inter-Integrated Circuit Bus

IDE : Environnement de développement intégré

IMU : Inertial Measurement Unit ou centrale à inertie

IoT : Internet of Things ou Internet des Objets

Lab : Département de recherche

R&D : Recherche et Développement

ROS : Robot Operating System

SII : Société pour l'informatique Industrielle

UWB : Ultra Wide Band

CHAPITRE 1. INTRODUCTION

Mon projet de fin d'étude se déroule à SII dans le département Recherche et Développement et s'étale du 7 Février 2022 au 5 Août 2022 à SII Sud-Ouest à Toulouse.

Le centre de R&D de SII a été créé en 2020 et a pour but d'explorer des projets afin d'attirer de potentiel clients. Ce centre explore plusieurs thématiques allant de l'intelligence artificielle au développement embarqué en passant par le développement d'interface homme machine et les big data. La plupart de ces projets sont développés pour répondre à des besoins clients comme Airbus, Thalès ou encore Continental.

C'est dans ce contexte que j'évolue au sein du pôle embarqué. Ce dernier est séparé en deux labs, le lab vision collaborative et le lab positionnement fin dans laquelle je suis.

Les objectifs du lab de positionnement fin sont multiples. En premier lieu, le lab collabore avec Continental et Volterio sur un projet de chargement de voiture électrique autonome. Ce projet prend la forme d'un robot placé au sol et est doté d'un bras qui peut se déplacer pour se positionner sous une voiture, se connecter à sa prise et enfin la charger. Toujours avec Continental, l'équipe travaille avec des modules UWB qui est un protocole de communication sans fil à courte portée comme le wifi ou le Bluetooth mais à plus haute fréquence. Ici, l'utilisation de plusieurs modules permet de déterminer la distance entre deux ou plusieurs modules. Dans notre cas, nous avons trois modules placés en triangle sur un robot et un autre module cible fixe placé au sol. Plusieurs techniques de triangulation sont ensuite appliquées pour déterminer la distance entre le robot et le module. Ces techniques doivent nous permettre de nous localiser ou du moins de nous rapprocher avec le robot à une quinzaine de centimètres de la cible. C'est la première étape du positionnement fin. Ensuite, un aimant puissant est situé près de la prise de recharge et est utilisé dans la dernière phase de positionnement. On utilise alors deux capteurs à effet hall pour déterminer l'angle par rapport à l'aimant et ainsi se rapprocher à quelques millimètres au-dessus de l'aimant.

L'objectif de mon stage est ainsi d'intervenir en renfort de ces deux parties à l'aide d'une IMU. L'IMU pour Inertial Measurement Unit est une unité à neuf degrés de liberté composée d'un accéléromètre 3 axes, d'un gyroscope 3 axes et d'un magnétomètre 3 axes. Chaque axe représente une direction de mesure. Par exemple, l'accéléromètre va mesurer l'accélération en m/s^2 dans un repère cartésien donc selon X, Y et Z. Initialement, je disposais uniquement d'un accéléromètre et d'un gyroscope. L'objectif est donc de déterminer si la combinaison de ces deux capteurs peut fournir une mesure de distance et d'angle fiable pour apporter un retour sur le déplacement du robot. On utilise différentes techniques dont la fusion de données pour estimer la distance par le capteur ainsi que l'angle parcourue. Cette approche avec uniquement deux capteurs a montré plusieurs points faibles qui nous ont poussé à changer de capteur et à rajouter un magnétomètre pour améliorer les mesures.

Le développement des algorithmes se fait en trois phases. Tout d'abord une phase de recherche qui sert à poser les bases de calculs pour les mesures. Ensuite, une première implémentation en C++ à des fins de tests sur un ESP32. Enfin, l'implémentation finale en Rust sur Raspberry Pi en repartant de zéro car aucune librairie n'existait déjà pour ce capteur.

Ce rapport présente donc mon travail au cours de ces 6 mois de stage au sein du lab positionnement fin aussi appelé lab embarqué.

La partie suivante décrit plus en détail le cadre et les objectifs du stage. Je repositionne mon stage au sein de l'entreprise et présente les objectifs qui m'ont été fixés au début du stage puis comment ces objectifs ont évolué. Enfin, j'explique comment mon sujet de stage est lié aux enseignements de ma spécialité.

La prochaine partie présente l'état de l'art réalisé au cours du stage et qui donne les bases scientifiques de ma recherche pour ce stage. Cet état de l'art porte sur la première partie de mon stage c'est-à-dire sur un capteur six axes composés d'un accéléromètre et d'un gyroscope uniquement. Ce premier rapport permet de choisir les technologies utilisées et de les tester individuellement avant de les implémenter. Il présente également les limites de la première solution envisagée qui nous a permis par la suite de changer de stratégie.

Ensuite, je présente mes réalisations du code écrit aux résultats obtenus en passant par les difficultés rencontrées au cours des tests. J'explique également les limites des solutions obtenues et analyse leur viabilité au sein du projet. C'est alors l'occasion de remettre en perspective mon travail.

Enfin, je détaille mes activités diverses et annexe à mon projet.

CHAPITRE 2. CADRE ET OBJECTIFS DU STAGE

Ce chapitre présente le service dans lequel j'évolue c'est-à-dire le lab Embarqué et ses objectifs au sein de SII. Je décris les objectifs de mon stage ainsi que les exigences de mon stage qui ont évolué au fil de l'avancement du projet. Enfin, je montre mon matériel à disposition et présente le contexte métier et technique.

2.1 Présentation du LAB Embarqué

Le LAB R&D systèmes embarqués a pour but l'étude d'utilisabilité des technologies émergentes du point de vue de la sûreté de fonctionnement et la rationalité de leur implémentation au sein de projets industrialisables, en se focalisant principalement sur des sujets tels que :

- L'exploration de nouvelles technologies ou concepts pour la fiabilité et la sureté des logiciels appliqués aux systèmes embarqués critiques.
- Le développement de solutions de logiciels embarqués critiques plus sûres.
- La simplification du développement de systèmes complexes.

Plusieurs projets sont abordés au sein du Lab. :

1. Projet Vision Collaborative : L'étude de ce projet se concentre sur l'amélioration de la sécurité routière grâce à la mise en place d'une collaboration entre plusieurs véhicules et grâce à une approche préventive plutôt que corrective (e.g. pour l'évitement de collision). L'approche « Vision Collaborative » vise à mettre en commun les informations obtenues par les capteurs et par des fonctions intelligentes. Les solutions envisagées seront analysées en relation avec leur applicabilité aux véhicules et normes actuelles. De plus, ce projet a la particularité de faire intervenir des compétences et des ressources de plusieurs Lab. R&D de SII Sud-Ouest.
    a. Le Lab. IA (Intelligence Artificiel au service de l'ingénierie) qui a pour but de :
        - Travailler les savoir-faire IA au service de nos métiers
        - Étudier et optimiser les algorithmes d'apprentissage
    b. Le Lab. IHM (Interaction Homme-Machine) qui a pour but d'accompagner l'humain dans les transports (cockpit, habitacle, contrôle aérien) pour rendre l'usage plus sûr et accessible à tous.
2. Projet RUST : Étude du langage RUST appliqué aux systèmes embarqués (critiques et comparaisons avec d'autres langages). Cette étude nous a permis de développer des compétences solides en développement logiciel sur microcontrôleur dans ce langage (STM32, drivers). Le RUST est ainsi devenu un outil important dans nos stratégies de développement et se retrouve dans d'autres sujets d'étude.
3. Projet Positionnement fin : L'étude de ce projet est de proposer un système de guidage précis en intérieur (< 1cm) afin de pouvoir automatiser des tâches de maintenance chez les particuliers dans le domaine de l'automobile. En comparant les mécanismes existants, nous souhaitons proposer un environnement viable pour la conception d'un prototype afin de faire des essais autour de la précision des techniques de localisation, la fiabilité des algorithmes (résistance aux interférences, répétabilité) et la cyber-sécurité (empêcher les attaques du type « man in the middle »).

2.2 Objectifs du projet Positionnement Fin

L'idée de cette étude est de proposer un système de guidage précis en intérieur afin de pouvoir automatiser des tâches de maintenance dans le domaine de l'automobile. Ce système, en dehors de la problématique de la précision, porte aussi sur les notions de fiabilité et de cyber-sécurité.

En 2021, l'équipe en place a esquissé l'ébauche du projet, qui consistait à déterminer la précision des technologies IoT pour réaliser un positionnement en intérieur. Après une phase d'étude des différentes technologies existantes, le choix s'est arrêté sur deux technologies :

- L'Ultra Wideband, qui est un protocole IoT de moyenne portée, et qui semble avoir été développé dans le but d'améliorer la précision des systèmes de positionnement, avec certains aspects intéressants, notamment liés à la traversée des obstacles et à la sécurité.
- Les capteurs à effet hall, qui sont de plus en plus utilisés et permettent d'obtenir des informations liées à la position d'une source magnétique.

En combinant ces deux technologies, nous espérions obtenir un système robuste en utilisant les points forts de chacune des technologies utilisées, augmentant ainsi la précision et la fiabilité du système final.

En 2022, nous souhaitons continuer à développer ce projet en se basant sur les conclusions de l'année passée qui nous permettent d'envisager des résultats positifs sur la solution choisie.

Nous avons également choisi de placer notre application dans le monde de la robotique, en utilisant ses outils, et nous permettant d'appliquer la solution sur des cas concrets. Cela nous permet également de placer notre système dans un milieu évolutif plutôt que dans un milieu statique.

À la suite de ces études, SII a pour objectif de proposer son expérience aux acteurs automobiles afin de les aider à relever les défis technologiques liés à l'automatisation des tâches dans ce domaine.

2.3 Le stage au sein du lab

2.3.1 Objectif initial

Des prototypes de fusion de données doivent être réalisés afin d'essayer d'améliorer le système de positionnement actuel.

Pour rappel, le projet de positionnement fin comprend deux phases :

- Une phase de positionnement peu précis longue distance. Cette phase est effectuée par trilatération par UWB.

- Une phase de positionnement fin à courte distance. Cette phase est faite avec le capteur à effet Hall.

La fusion de données peut intervenir à plusieurs points permettant d'améliorer les précisions du système. Le sujet de ce stage répond donc à une problématique précise. En effet, la fiabilité des données est une des raisons principales pour la mise en place de ce retour sur la position du robot. Les algorithmes utilisés pour déterminer la position de notre cible, que ce soit par méthode grossière ou fine, prennent en compte la mesure de distances ainsi que la position où cette mesure a été faite. En introduisant un retour sur la position du robot, et intrinsèquement un retour sur la position de nos capteurs, cela permet de pouvoir mieux traiter les données afin d'être de plus en plus fiable au cours de la mesure et du déplacement du robot.

Le guidage précis du robot jusqu'à la cible est la raison pour laquelle il est souhaitable d'avoir une donnée fiable dans notre étude. Les algorithmes de traitement des données sont beaucoup plus précis avec une grande reproductibilité des mesures, permise par ce retour de position.

Dans le cadre du projet de positionnement fin, l'utilisation d'un accéléromètre et d'un gyroscope est prévue pour compléter la phase de positionnement peu précis effectuée par trilatération par UWB (Ultra Wide Band). L'objectif est d'utiliser la fusion de données pour déterminer l'angle et la distance parcourue.

L'ajout d'un accéléromètre couplé à un gyroscope permettrait (par intégration mathématiques) de déterminer la position relative du robot par rapport à son point de départ. En utilisant cette technologie en parallèle de l'UWB, de meilleures performances peuvent être possiblement obtenues.

L'accéléromètre permet de calculer le trajet effectué et le gyroscope permet de calculer les angles dans lequel s'est tourné le robot. Ensemble, une position précise de la navigation du robot peut être calculée.

Le déplacement du robot prend comme entrée une position déterminée par l'UWB. Cette position est ensuite convertie en chemin à parcourir par le robot (ex: Tourne de 67° puis avance de 145cm). Cette commande est ensuite envoyée aux moteurs. En rajoutant le gyroscope ainsi que l'accéléromètre, il est alors possible de mieux garantir que les commandes aient étaient respectées par les moteurs.

Pour pallier à ce manque, la méthode actuelle est de s'arrêter en chemin pour le robot, afin de refaire une mesure UWB et d'ainsi déterminer sa position actuelle. En utilisant le couple accéléromètre/gyroscope, on pourrait avoir une plus grande précision sur la distance parcourue par le robot, et ainsi mieux traiter le bruit de la mesure d'UWB.

2.3.2 Évolution du projet

La phase initiale du projet fait intervenir un capteur à 6 axes uniquement. Cependant, les tests effectués lors de l'état de l'art nous ont amené à changer de capteur et ainsi passer sur un capteur à 9 axes avec l'ajout d'un magnétomètre.

Dans les derniers centimètres, le capteur à effet Hall sera utilisé pour déterminer d'où provient l'aimant le plus proche. Les mouvements seront lents et de quelques centimètres mais ces capteurs peuvent nous aider à savoir si notre robot patine ou se déplace correctement.

Si le déplacement est cartographiable pendant le déplacement fin, ces données peuvent être couplées avec le capteur à effet Hall afin de déterminer bien mieux où exactement nous nous situons près de l'aimant. Il permettrait aussi de facilement revenir sur nos pas si l'aimant devient indétectable.

L'aimant va malheureusement grandement perturber notre magnétomètre. Le magnétomètre repère le champ magnétique le plus intense le parcourant et le décrit sur chaque axe. Proche de l'aimant, le champ magnétique de la Terre est plus faible que celui de l'aimant, donc une fusion de données entre gyroscope et magnétomètre n'est plus possible. Celui-ci nous sera tout de même utile afin de déterminer la position à moyenne distance de l'aimant (de 50cm à 10cm). De plus, il reste possible d'obtenir une estimation de l'angle parcouru par le robot grâce au couple accéléromètre et gyroscope. Mais même si cette estimation est moins précise que celle obtenue avec le magnétomètre elle reste utilisable pour nos besoins.

Il a donc été décidé de scinder le reste de l'étude en deux parties : l'une pour le guidage à longue distance et l'autre à courte distance (< 50 cm). Ce choix a été fait car le magnétomètre n'est utilisable pour un retour de position que quand nous sommes assez loin de l'aimant qui est notre cible. En effet, à courte distance, le champ magnétique émit par l'aimant utilisé pour le positionnement n'est plus négligeable et impacte nos mesures. Le capteur ne mesurera plus le nord magnétique mais son orientation par rapport à l'aimant. Le reste du projet, c'est-à-dire la mesure de distance, ne sera pas impacté par ces changements.

2.3.2.1 Longue distance

À grande distance, on considère que seul le champ magnétique terrestre viendra influencer les mesures faites par le magnétomètre. Le magnétomètre pourra donc être utilisé pour mesurer le Nord magnétique de la Terre. Dû au fait que le Nord magnétique est différent du Nord des cartes, une correction est faite à ce niveau pour une utilisation depuis la France, les autres régions du monde nécessitent une correction différente. Cette correction s'appelle la déclinaison magnétique et est calculable à l'aide des coordonnées approximatives sur notre position.

2.3.2.2 Courte distance

À courte distance, le magnétomètre ne sera pas utilisé pour le retour sur la position. Dans cette configuration, il n'est pas possible de corriger le gyroscope grâce au Nord magnétique de la Terre. On considère cependant que les mouvements du robot seront considérés comme « lents ». L'accéléromètre a une grande fiabilité si de faibles mouvements y sont appliqués. Il est utilisé afin de compenser le gyroscope. Cependant, le magnétomètre pourrait nous donner un angle relatif non plus par rapport au nord magnétique mais à l'aimant. Autrement dit, si le robot tourne d'une position A à une position B, on pourrait être en mesure de déterminer l'angle parcouru.

2.4 Ressources à disposition

2.4.1 Les capteurs

Initialement, le capteur dont je dispose est un GY-521 [1] qui comporte une puce MPU-6050 [2] et communique par I2C. Cette puce comporte un accéléromètre à 3 axes et un gyroscope à 3 axes pour un total de 6 degrés de libertés. La Figure 1 [1] est une photo du capteur.

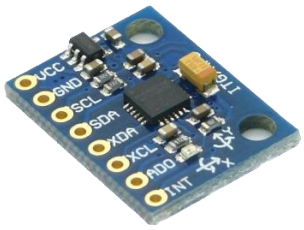


Figure 1 : GY-521 avec la puce MPU-6050 au centre

La suite du projet nous amène à utiliser une paire de capteur afin de créer une machine inertielle à neuf degrés de liberté. Le FXOS8700 est un capteur comprenant 3 accéléromètres et 3 magnétomètres soit un sur chaque axe et le FXAS21002 comprenant un gyroscope sur 3 axes lui aussi. Les deux capteurs sont présents sur une seule Breakout Board fourni par Adafruit [3]. La Figure 2 [3] ci-dessous est une photo du capteur.

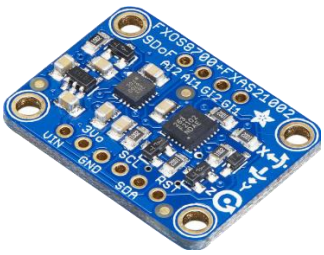


Figure 2 : Adafruit Precision NXP 9-DOF Breakout Board - FXOS8700 + FXAS21002

2.4.2 Microcontrôleurs utilisés

Tous les tests sont principalement réalisés avec un ESP32 [4] conçu par Espressif. Il est basé sur l'architecture Xtensa LX6 de Tensilica et est programmable facilement via le logiciel Arduino IDE. Il permet un accès facile et rapide au port du GPIO pour la configuration et l'obtention des données en I2C. Grâce à l'IDE d'Arduino, on a accès à des milliers de bibliothèques dont celles développées pour les capteurs que j'utilise. Cela permet un développement et des tests plus rapides car on dispose d'une base de travail. La Figure 3 [4] ci-dessous montre la carte de développement avec les ports GPIO.

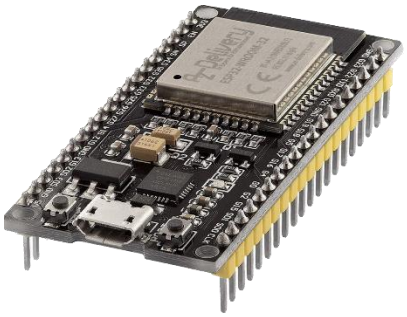


Figure 3 : ESP32 Az-Delivery

Seul l'ESP32 était censé être utilisé à la base car théoriquement il était possible de coder en Rust sur cette architecture. Cependant, malgré tous les tests effectués, il est bien possible de faire tourner du Rust mais les usages sont très limités et donc pas adaptés à nos besoins. Il a donc été décidé de passer sur une carte STM32 [5] qui est plus puissante et qui dispose d'une HAL écrite en Rust et qui permet d'utiliser plus rapidement les composants essentiels. La Figure 4 ci-dessous montre la carte utilisé.

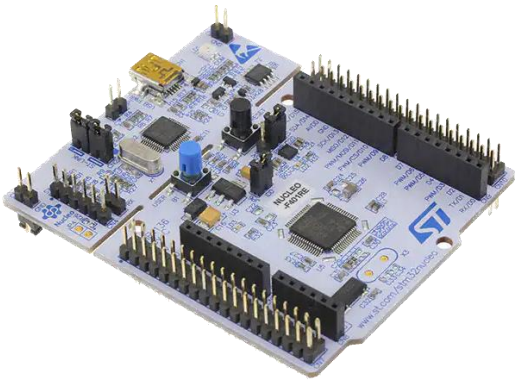


Figure 4 : Carte STM32F401RE

Cependant, et bien que cette carte fait tourner parfaitement le code demandé en Rust, des problèmes ont été rencontrés lors de l'édition des délais. En effet, il est nécessaire lors de la récupération des données sur l'IMU d'avoir un délai de stabilisation entre chaque mesure. Ce délai s'est avéré peu fiable et donc difficilement utilisable. On a donc décidé de passer sur une Raspberry Pi 4 [6], qui est un mini-ordinateur capable de faire tourner un noyau linux et qui dispose donc de librairies standard. Cette solution règle donc la plupart des problèmes rencontrés et permet ainsi de se concentrer sur l'essentiel c'est-à-dire le développement de nos drivers. La Figure 5 ci-dessous montre la carte.

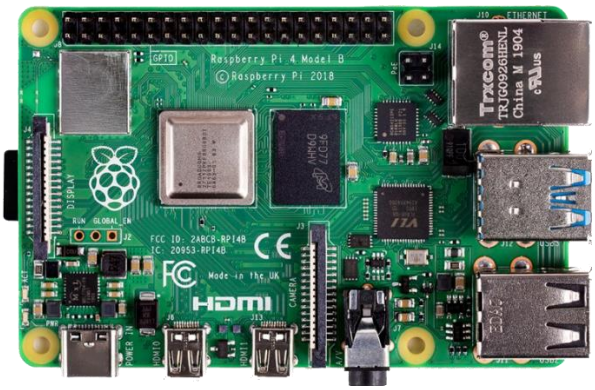


Figure 5 : Raspberry Pi 4 modèle B

En plus de disposer des librairies standard nativement apportés par Linux Ubuntu, c'est aussi la carte de développement principale utilisée par l'équipe pour les autres capteurs et la gestion du robot via ROS.

2.5 Contexte

2.5.1 L'équipe et mode de travail

Au sein du lab Embarqué, l'équipe est composée de 6 à 8 personnes. À la fin de mon stage, il y a 4 stagiaires et 2 lab owner étant les chefs de projets dont fait partie mon tuteur de stage. Il y a deux personnes sur les modules UWB, une personne sur le capteur à effet hall et moi sur l'IMU. Enfin, les deux lab owners supervisent notre travail et sont également en projet avec Continental.

Je suis, sur la majeure partie du projet, en autonomie pour le développement de mes drivers. J'ai ensuite participé à l'intégration de mon travail avec l'équipe ou une partie de l'équipe en particulier sur la partie avec le capteur à effet hall car c'était le plus abouti à la fin de mon stage.

2.5.2 Contexte technique

Ce stage fait appel à de nombreuses compétences acquises à l'INSA notamment en programmation en C et C++. Les enseignements du module Smart Devices ont été particulièrement utiles pour mon stage grâce aux cours sur les capteurs, en électronique et en programmation Arduino. Ces compétences sont donc idéales dans ce contexte de développement logiciel embarqué.

CHAPITRE 3. ÉTAT DE L'ART

Ce chapitre est dédié à mon état de l'art qui porte sur la fusion de données entre un accéléromètre et un gyroscope. L'introduction est volontairement omise car redondante avec les parties précédentes.

3.1 Définitions

3.1.1 Accéléromètre

Comme son nom l'indique un accéléromètre permet de mesurer une accélération selon un ou plusieurs axes. Son application la plus courante est la mesure de vibrations et de chocs. Il y a deux grandes catégories : Piézoélectrique et Capacitifs [7].

Un élément piézoélectrique est un type de matériau renvoyant une énergie électrique proportionnelle à l'effort appliqué. Un accéléromètre piézoélectrique est composé d'un élément piézoélectrique précontraint par une masse sismique. La vibration fait varier la pré contrainte (en compression ou cisaillement) et déforme l'élément piézoélectrique qui génère alors un signal électrique exprimé en unités pC/g ou mV/g. Ils répondent ainsi à une grande variété d'applications notamment dans le domaine des mesures basses fréquences telle que le confort vibratoire, la mesure de chocs ou encore dans l'analyse sismique géologique ou d'infrastructures.

Néanmoins ils ne permettent pas d'observer des fréquences très basses, et ils ne « passent » pas la composante continue contrairement aux accéléromètres capacitifs.

Un accéléromètre capacitif illustré sur Figure 6 [7] ci-dessous est constitué de 2 condensateurs ainsi que d'un élément mobile. La mise en mouvement de cet élément va faire varier la valeur de capacité des deux condensateurs, ce qui nous permettra d'en déterminer l'accélération. C'est ce type de capteurs que nous utiliserons dans notre projet.

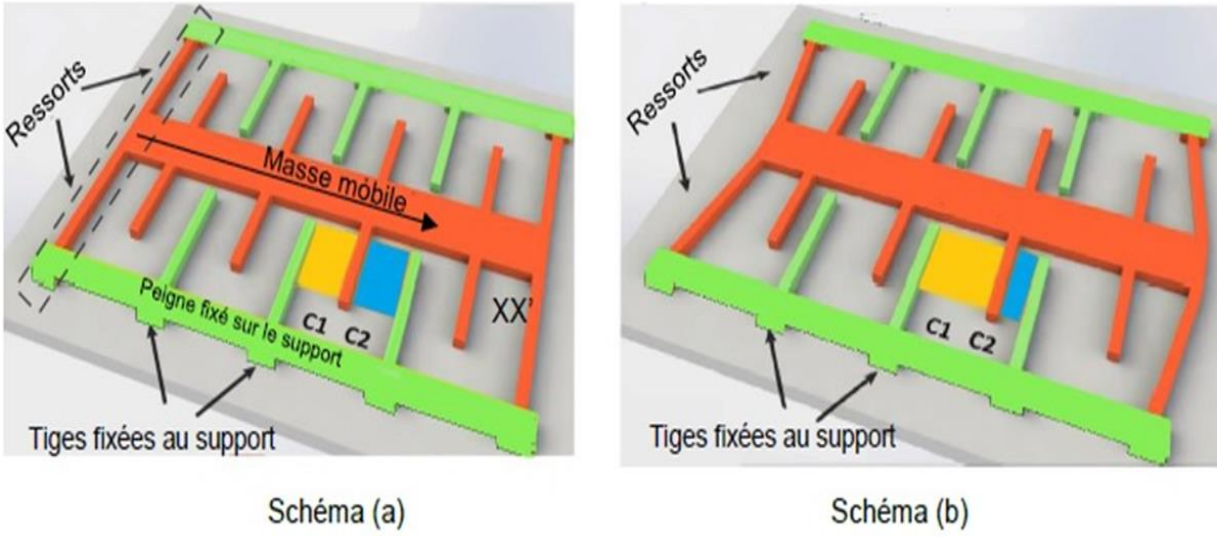


Figure 6 : Représentation d'un système MEMS

Elle est composée d'une partie mobile en forme de peigne avec des tiges en silicium se déplaçant entre des tiges fixes, écartées d'environ 1,3 micromètres. Lorsque l'accéléromètre d'un smartphone bouge, la tige mobile oscille d'un côté ou de l'autre. En mesurant la variation de capacité électrique entre les tiges, on déduit le sens et l'ampleur du mouvement.

Ces capteurs passent la composante continue, c'est à dire qu'ils peuvent également mesurer une valeur statique d'accélération (exemple : la composante de g, pour en déduire une inclinaison). Bien que leur échelle de mesure se limite de 3 à 100g, les accéléromètres capacitifs sont robustes et sont capables de supporter des surcharges allant jusqu'à 10 000 fois la force de gravité *g* tout en fournissant un signal de sortie élevé.

3.1.2 Gyroscope

Un gyroscope est un dispositif utilisant la gravité terrestre pour déterminer l'orientation. Les capteurs gyroscopiques sont des dispositifs détectant la vitesse angulaire, qui est le changement d'angle de rotation par unité de temps. La vitesse angulaire est généralement exprimée en degrés/s (degrés par seconde). Il existe trois types de gyroscopes de base : rotatifs (classique), à structure vibrante et optiques. En français, on parle plutôt de gyromètre pour mesurer une vitesse angulaire.

3.1.3 Défauts des données brutes

Indépendamment et/ou sans traitement, les données en sorties de l'accéléromètre et du gyroscope ne sont pas utilisables en l'état. En effet, la Figure 7 illustre la sortie d'un accéléromètre selon un de ses axes et complètement immobile.

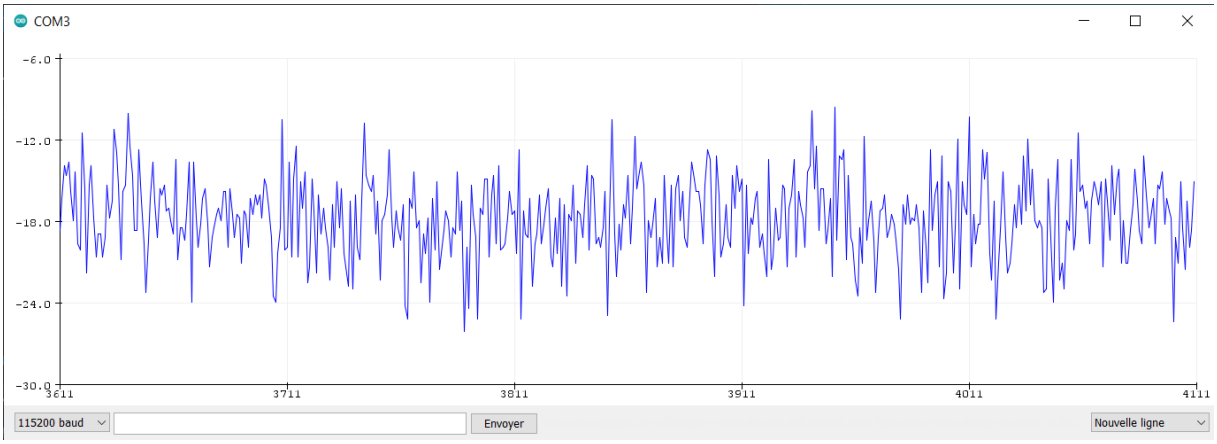


Figure 7 : Visualisation de l'accélération brute

Le signal n'est pas stable et il est alors impossible d'intégrer le signal pour obtenir la distance parcourue car cette dernière divergerait vers l'infini alors même que le capteur n'a pas bougé. En effet, l'erreur obtenue lors de l'intégration est quadratique.

Pour pouvoir utiliser ces données, il faut calibrer avec précision le capteur et appliquer des filtres pour lisser le signal tout en conservant les informations.

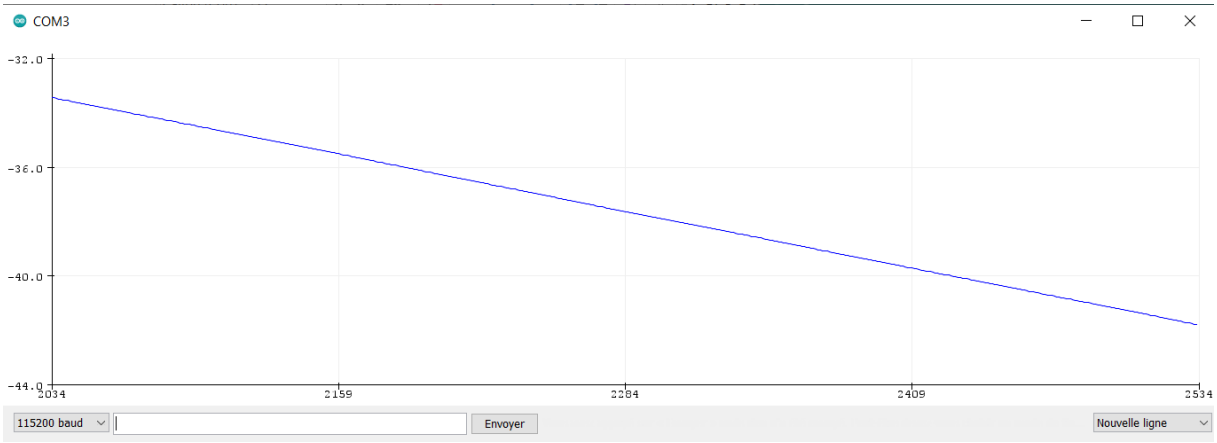


Figure 8 : Dérive du Gyroscope

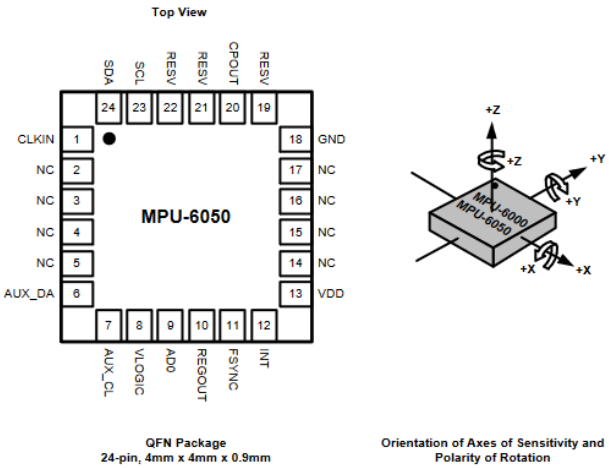
Lorsqu'on observe le comportement du gyroscope alors qu'il est posé à plat et ne bouge pas, on observe sur la Figure 8 une dérive du signal vers moins l'infini (on devrait observer une ligne droite). Lorsqu'on le bouge on observe une perturbation mais cela n'arrête pas la dérive.

Il faut donc trouver un moyen de corriger cette dérive pour rendre le signal utilisable et ainsi pourvoir l'intégrer pour trouver un angle.

3.2 Le capteur : GY-521

3.2.1 Caractéristiques

La carte GY-521 [2] est muni d'un capteur MEMS MPU-6050. Le circuit contient trois capteurs : un accéléromètre à 3 axes, un gyroscope à 3 axes et un capteur de température. La Figure 9 montre les sorties et les axes de la puce. La Figure 10 illustre le diagramme bloc de son fonctionnement.



QFN Package
24-pin, 4mm x 4mm x 0.9mm

Orientation of Axes of Sensitivity and Polarity of Rotation

Figure 9 : Pin out et axes du MPU-6050

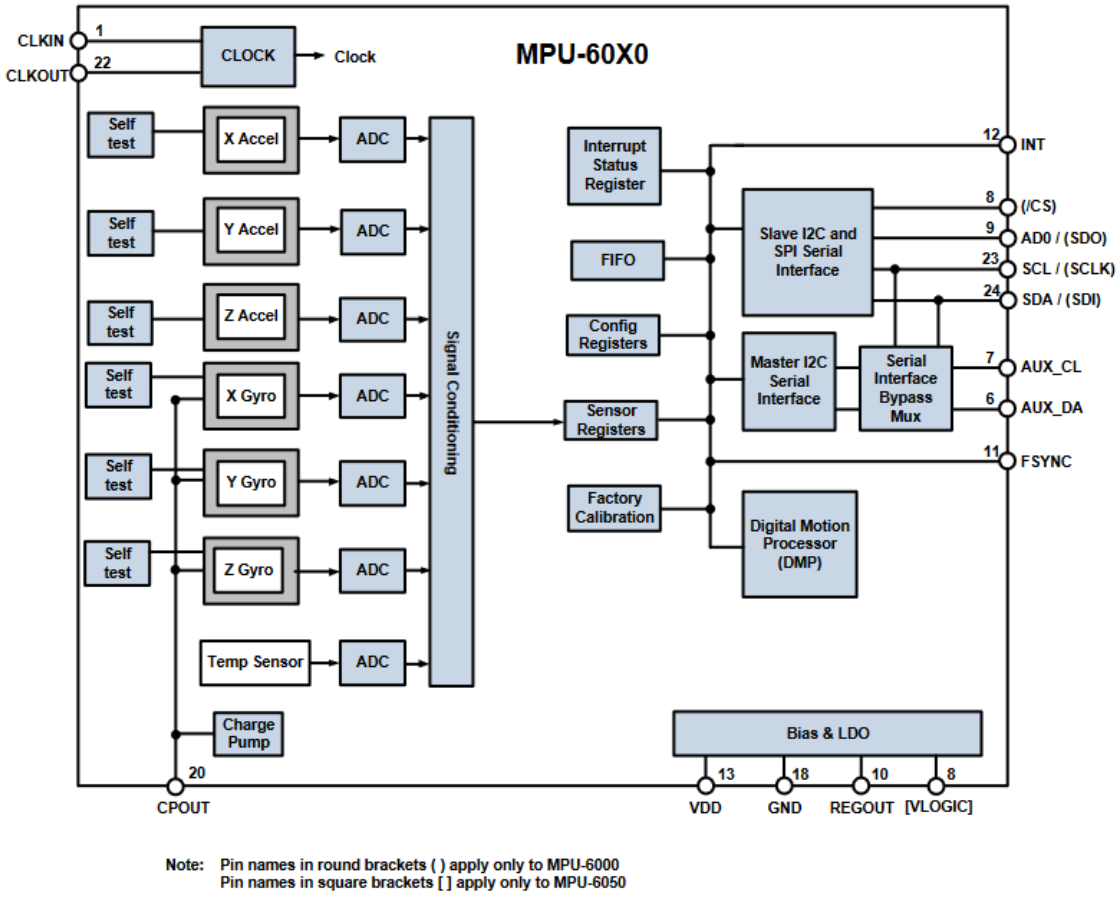


Figure 10 : Diagramme de bloc du MPU-6050

- Caractéristiques du gyroscope :
  - Le gyroscope MEMS à trois axes du MPU-60X0 comprend un large éventail de fonctionnalités:
  - Capteurs de vitesse angulaire (gyroscopes) à sortie numérique des axes X, Y et Z avec une plage pleine échelle programmable par l'utilisateur de ± 250, ± 500, ± 1000 et ± 2000 ° / s
  - Le signal de synchronisation externe connecté à la broche FSYNC prend en charge la synchronisation d'image, vidéo et GPS
  - Les ADC 16 bits intégrés permettent l'échantillonnage simultané des gyroscopes
  - Compensation de la stabilité en température intégrée dans le capteur
  - Amélioration des performances de bruit basse fréquence
  - Filtre passe-bas programmable numériquement
  - Courant de fonctionnement du gyroscope: 3,6 mA
  - Faible consommation au repos 5 µA
  - Facteur d'échelle de sensibilité calibré en usine
  - Auto-test de l'utilisateur

- Caractéristiques de l'accéléromètre :
  - L'accéléromètre MEMS à trois axes du MPU-60X0 comprend un large éventail de fonctionnalités:
  - Accéléromètre à trois axes à sortie numérique avec une plage de pleine échelle programmable de ± 2g, ± 4g, ± 8g et
  - ± 16g

- Les ADC 16 bits intégrés permettent l'échantillonnage simultané des accéléromètres sans multiplexeur
- Courant de fonctionnement normal de l'accéléromètre: 500µA
- Courant du mode accéléromètre basse puissance: 10µA à 1,25Hz, 20µA à 5Hz, 60µA à 20Hz, 110µA à 40 Hz
- Détection et signalisation d'orientation
- Détection de robinet
- Interruptions programmables par l'utilisateur
- Interruption High-G
- Auto-test de l'utilisateur

3.2.2 Fonctionnement

- Pin utilisable du capteur :
  - VCC → 3.3V ou 5V (de préférence)
  - GND → Ground
  - SCL → Communication I2C
  - SDA → Communication I2C
  - XDA → Communication I2C Auxiliaire
  - XCL → Communication I2C Auxiliaire
  - AD0 → Adresse I2C 0x68 (ou 0x69 si bit à 1)
  - INT → Bit d'interruption

- Communication avec le capteur :

Pour récupérer les données du capteur, on passe une communication I2C. Pour initier le projet, nous avons choisi d'utiliser une librairie en C++ développé par Rob Tillaart [8]. Lors du passage du C++ au Rust, nous utiliserons une autre librairie ou la développerons nous-même. Le but de la librairie est de simplifier la récupération des données d'accélérations et gyroscopiques pour ne pas s'attarder sur l'acquisition des données mais plutôt sur leur traitement.

3.3 Détermination de l'angle

3.3.1 Principales idées explorées

L'objectif est de déterminer l'angle de rotation autour de l'angle Z. Pour cela, nous avons besoin des données du gyroscope (Gz). À partir des données brutes on calcule le lacet. Habituellement, la notion de lacet est utilisée pour un avion. On parle alors d'axe de tangage (Pitch), de roulis (Roll) et de lacet (Yaw) [9] comme le montre la Figure 11. Ces trois notions nous seront utiles dans la suite du projet.

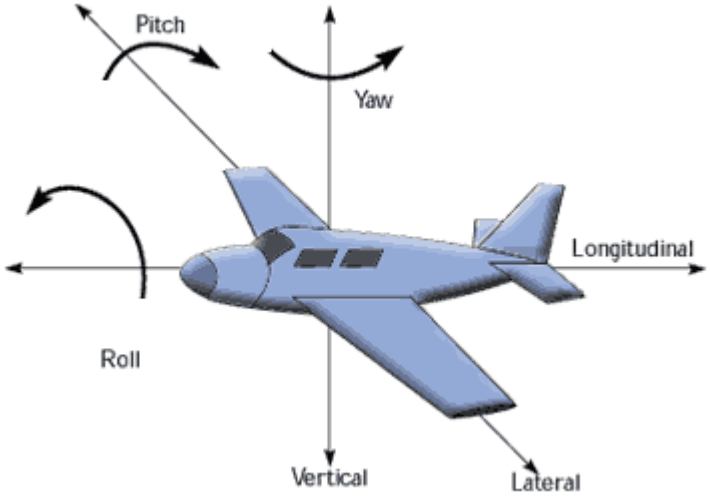


Figure 11 : Visualisation du tangage, roulis et lacet

Pour cette partie, nous voulons connaitre le lacet ce qui correspond à l'angle Z (du gyroscope) sur le capteur MPU6050. Pour rappel, les données du gyroscope ne sont pas stables au cours du temps et dérivent très rapidement. À contrario, les données de l'accéléromètre sont plus stables à court terme. On peut donc utiliser de la trigonométrie classique pour déterminer l'angle de lacet. Pour son calcul, on utilise la formule suivante :

$$\text{Yaw} = \tan^{-1}\frac{accZ}{\sqrt{accX^2 + accZ^2}}$$

Équation 1 : Calcul du lacet

Ce n'est pas le seul moyen de calculer cet angle, néanmoins, l'angle calculé dérive autant que pour le gyroscope. Il faut donc utiliser un filtre spécial pour stabiliser le signal en y combinant les données du gyroscope.

3.3.2 Filtre de Kalman

Le filtre de Kalman [10] est une méthode permettant d'estimer des paramètres d'un système évoluant dans le temps à partir de mesures bruitées. La force de ce filtre est sa capacité de prédiction des paramètres et de rectification des erreurs, non seulement des capteurs, mais aussi du modèle lui-même. Il suffit d'intégrer un terme d'imprécision sur le modèle lui-même, ce qui lui permet de donner des estimations correctes malgré les erreurs de modélisation (pour peu que les erreurs restent raisonnables).

Le fonctionnement du filtre de Kalman peut se diviser en deux étapes :

- Une première étape de prédiction de l'estimation selon le modèle du système. Pour ce faire, le filtre de Kalman reprend l'estimation précédente des paramètres et de l'erreur et prédit les nouveaux paramètres et la nouvelle erreur en fonction de la modélisation du système.

- La seconde étape va faire la mise à jour de cette prédiction grâce aux nouvelles mesures. Ces mesures (par définition bruités) vont permettre d'obtenir une estimation des paramètres et de l'erreur à partir de la prédiction faite. Si jamais le modèle comporte des erreurs, cette étape de mise à jour permettra de les rectifier.

Ce filtre est couramment utilisé dans les méthodes de corrections d'erreurs liées aux données accéléromètre/gyroscope en réalisant une fusion de données entre les deux capteurs. Nous ne détaillerons pas les principes mathématiques car ce n'est pas le but ici mais nous détaillerons les étapes de l'algorithme :

- Lire les données du capteur
- Calculer le lacet
- Fixer l'angle de départ dans le filtre de Kalman pour les calculs
- Calculer le nouvel angle en utilisant le lacet et la valeur du gyroscope sur l'angle associé (ici Z)
- Revenir à la première étape

Le fonctionnement est analogue pour le roulis et le tangage.

Cependant, à la différence du roulis et du tangage, en sortie du filtre de Kalman nous n'avons pas un angle mais une vitesse angulaire en degrés par seconde (car l'angle du gyroscope et du lacet ont été convertis en degrés par seconde). En effet, les axes de roulis et de tangage, une fois passés par le filtre de Kalman, fournissent un angle absolu (dans les faits il est relatif à la calibration du capteur) car ils dépendent en grande partie des accélérations sur X et Y qui sont stables si soumises à un angle car elles sont situées dans le plan d'inclinaison. Pour le lacet, c'est la rotation du plan formé par les axes X et Y qui est mesurée. Or, les angles sur X et Y n'étant pas modifiées lors d'une unique rotation sur autour du plan, la seule information que nous disposons est alors la vitesse de rotation. C'est ainsi, qu'en sortie du filtre de Kalman nous avons une vitesse angulaire et non pas un angle. Il faut donc intégrer le signal au cours du temps pour obtenir l'angle de déplacement qui sera relatif à la position de départ.

Ce fonctionnement soulève un problème de taille. En effet, le fait de devoir intégrer le signal au cours du temps rajoute une erreur qui se rajoute à la mesure à chaque intégration. La précision est donc grandement diminuée.

Avec cette méthode, il est donc possible d'avoir une approximation de l'angle parcouru avec le filtre de Kalman mais pas une valeur aussi précise que pour le roulis et le tangage qui eux sont suffisamment précis pour être utilisables directement. L'unique moyen pour obtenir un résultat aussi fiable serait d'utiliser un magnétomètre et de faire de la fusion de données avec le gyroscope.

De plus, la moindre perturbation sur l'angle X ou Y, modifie grandement la courbe sur l'angle Z. Ce phénomène sera illustré dans la prochaine partie.

3.3.3 Premiers résultats

La Figure 12 ci-dessous montre le lacet en fonction du temps (courbe bleue). Ici, on a tourné le capteur d'environ 45°. Pour le calcul de l'angle, seule le temps en rouge est pris en compte car on applique un léger offset à partir duquel on va commencer à mesurer. C'est fait pour éviter de voir la mesure d'angle augmenter au cours du temps même lorsque le capteur ne bouge pas.

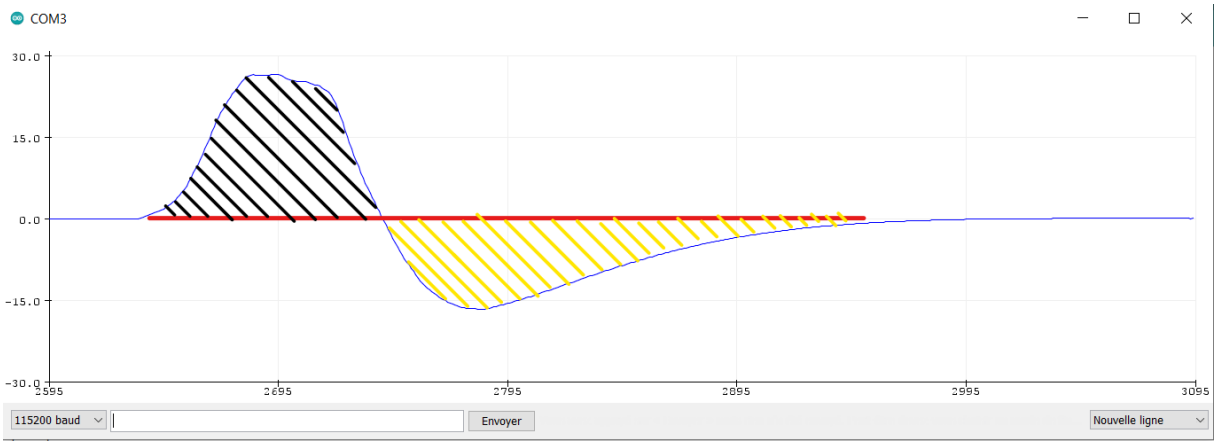


Figure 12 : Vitesse angulaire en sortie du filtre de Kalman

Pour obtenir la valeur de l'angle, on intègre la partie noire et la valeur absolue de la partie jaune. Ce calcul, nous donne dans le cas présent un angle relatif d'environ 44°. La mesure varie énormément selon la vitesse de rotation. Dans le cas d'une rotation très lente, la mesure sera sous-estimée et dans le cas d'une rotation très rapide, la mesure sera surestimée.

La Figure 13 ci-dessous montre maintenant l'influence des déplacements sur X et Y sur la vitesse angulaire sur Z. On a toujours notre vitesse de rotation en bleu et les lignes rouges servent de délimiteurs. Dans la zone 1, on bouge uniquement l'axe X, dans la zone 2 uniquement l'axe Y et enfin dans la zone 3 on bouge légèrement les deux axes pendant la même durée que pour la zone 2 avant de laisser le capteur retrouver son état de repos.

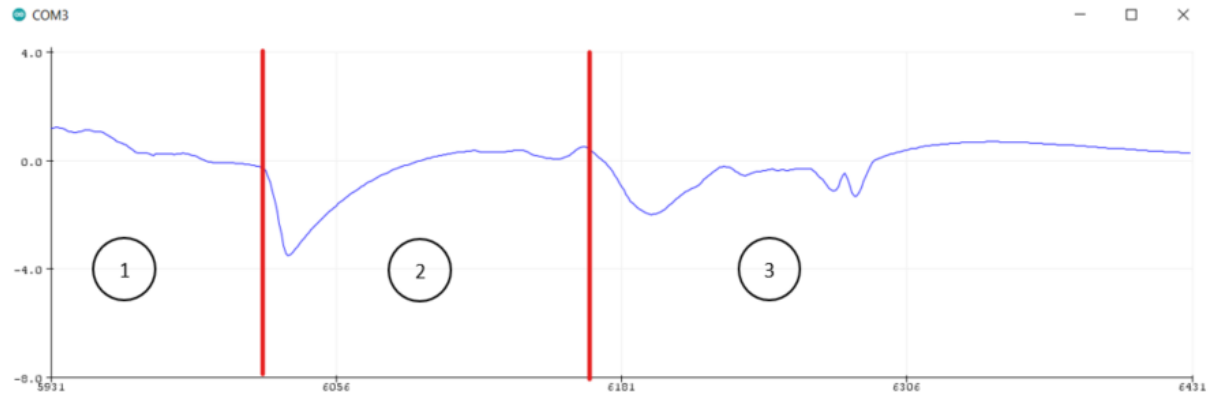


Figure 13 : Perturbations des axes X et Y sur l'axe Z

Le code utilisé permet également d'appliquer le même filtre sur les axes X et Y. Cette fois, on obtient bel et bien un angle absolu comme illustré sur la Figure 14. On part donc de 0° dans la zone 1 où on ne touche pas le capteur (il est en repos).

Dans la zone 2, on effectue une rotation de -90° et on observe donc que le signal indique directement -90° sans avoir besoin d'intégration. Dans la zone 3, le capteur est de nouveau au repos.

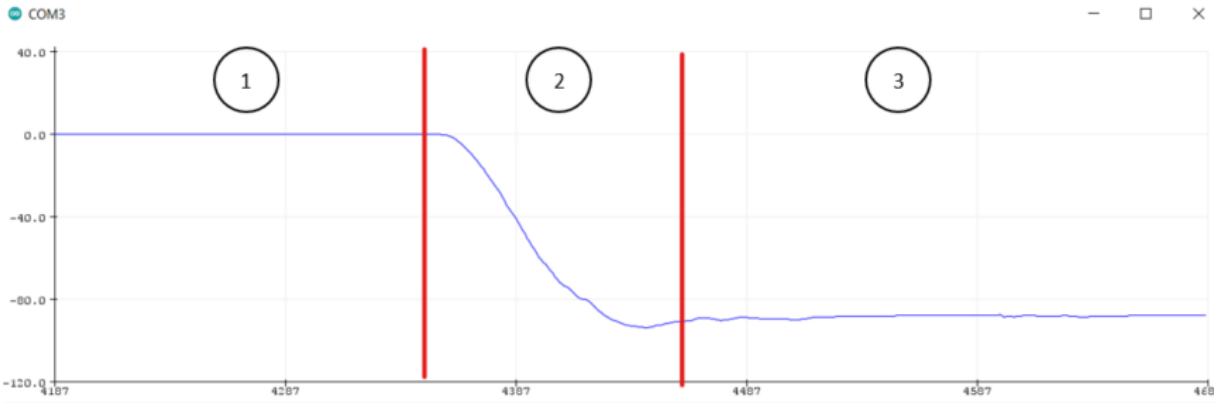


Figure 14 : Filtre de Kalman sur l'axe X

De la même manière que pour l'angle sur l'axe Z, on observe des perturbations mais uniquement sur l'axe concerné.

En conclusion, sans magnétomètre il est impossible d'obtenir une mesure précise sur l'axe Z et on obtiendra au maximum une approximation par intégration qui ne durera pas dans le temps à l'inverse des axes X et Y.

3.3.4 Limites de la méthode

Tout mouvement dans le plan X et Y perturbera la mesure de l'angle sur Z et comme vu précédemment, le filtre de Kalman ne fournissant uniquement une vitesse angulaire pour le lacet, nous n'aurons qu'une approximation de l'angle. Ainsi, la mesure de l'angle sera d'autant plus instable que le capteur bougera. On atteindra des aberrations. Il nous faut donc un magnétomètre pour nous permettre d'avoir une mesure d'angle absolue et directe et ainsi réduire l'influence des perturbations. Dans la suite du projet, il serait intéressant de tester la fusion de données avec un magnétomètre.

3.4 Détermination de la distance parcourue

3.4.1 Principales idées explorées

Pour déterminer la distance parcourue par le capteur, on n'a pas besoin d'utiliser de fusion de données mais le signal doit être filtré pour pourvoir être utilisé. En effet, le signal doit être intégré deux fois et à chaque intégration une erreur se rajoute et vient grandement fausser la mesure. Plusieurs filtres ont donc été testés puis combinés pour observer leurs effets sur la précision de la mesure. Les résultats des méthodes seront donnés dans la partie suivante.

3.4.1.1 Moyenne glissante

Une moyenne glissante ou moyenne mobile [11] permet de lisser une série de valeurs exprimées en fonction du temps. Elle permet d'éliminer les fluctuations les moins significatives. Elle calcule la moyenne sur les X dernières valeurs.

D'abord testée comme solution à part entière, cette méthode a été utilisée par la suite pour améliorer les courbes des autres méthodes.

3.4.1.2 Filtre numérique

Le signal filtré à l'instant N est égal au signal filtré à l'instant N-1 corrigé d'une fraction K de l'écart entre la mesure à l'instant N et le signal filtré de l'instant N-1[12]. Voici un exemple de l'influence de K sur un filtre d'ordre 1.

3.4.1.3 Filtre passe-haut et passe-bas

Un filtre passe-haut permet de ne garder que les fréquences au-dessus de sa fréquence de coupure. De la même manière, un filtre passe-bas permet de ne garder que les fréquences, cette fois-ci en dessous de sa fréquence de coupure. Dans les deux cas, ces filtres permettent d'éliminer la composante continue d'un signal et/ou d'en diminuer le bruit. Dans notre cas, nous l'utiliserons pour recentrer notre signal autour de 0 et pour diminuer le bruit important de l'accéléromètre. Recentrer le signal autour de zéro nous permet de garder les variations du signal qui sont les informations essentielles pour nos calculs et d'atténuer fortement les perturbations liées à l'éventuel inclinaison du capteur. En d'autre mots, le filtre passe haut supprime ou du moins attenue les angles de roulis et de tangage. En effet, un angle sur l'un ou l'autre de ces axes se traduit par l'ajout d'une composante continue qui s'ajoute au signal existant créant ainsi un signal intégré qui diverge rapidement vers l'infini. Le filtre passe haut permet alors de limiter ces effets en supprimant cette composante. Le signal étant très bruité, une solution évidente est d'utiliser un filtre passe bas, qui à l'aide d'une fréquence de coupure particulière, permet de limiter le bruit a un niveau acceptable. L'ajout d'une moyenne glissante en sortie de ces filtres permet d'obtenir un signal stable et utilisable par le biais de buffer zone.

Formule utilisée pour le filtre passe-bas :

$$s[n+1] = s[n] + \frac{T_e}{\tau} \cdot (e[n] - s[n])$$

Avec :

- s[n+1] : Sortie du filtre passe bas au temps n + 1
- s[n] : Sortie du filtre passe bas au temps n (valeur actuelle)
- T_e : Temps d'échantillonnage
- *T :* Constante de temps
- e[n] : Entrée du filtre (nouvelle valeur)

Formule utilisée pour le filtre passe-haut :

$$s[n+1] = s[n] + (e[n+1] - e[n]) - \frac{T_e}{\tau} \cdot s[n]$$

Avec :

- s[n+1] : Sortie du filtre passe haut au temps n + 1
- s[n] : Sortie du filtre passe haut au temps n (valeur actuelle)
- T_e : Temps d'échantillonnage
- *T :* Constante de temps
- e[n] : Entrée du filtre (ancienne valeur)
- e[n+1] : Entrée du filtre au temps n + 1 (nouvelle valeur)

3.4.1.4 Filtre de Butterworth

Le filtre Butterworth est un type de filtre de traitement du signal conçu pour avoir une réponse en fréquence aussi plate que possible dans la bande passante. Il est également appelé filtre d'amplitude au maximum plat. Butterworth a montré que des approximations successives plus proches étaient obtenues avec un nombre croissant d'éléments filtrants des bonnes valeurs.

3.4.2 Solution retenue

La solution retenue est celle de la combinaison d'un filtre passe-haut d'ordre 1 puis d'un filtre passe-bas d'ordre 1 également. Le tout agit comme un filtre passe bande. L'idée est de ne garder que les fréquences comprises entre 1 et 18 Hz car après analyse du spectre, c'est ici que se concentre l'essentiel de l'information. Du fait du temps de réponse des filtres, on utilise une fréquence de coupure de 0.05 Hz et 20 Hz respectivement pour le filtre passe-haut et passe-bas. Ces valeurs seront susceptibles d'être modifiées au fur et à mesure des tests. La Figure 15 présente le schéma bloc de la solution :

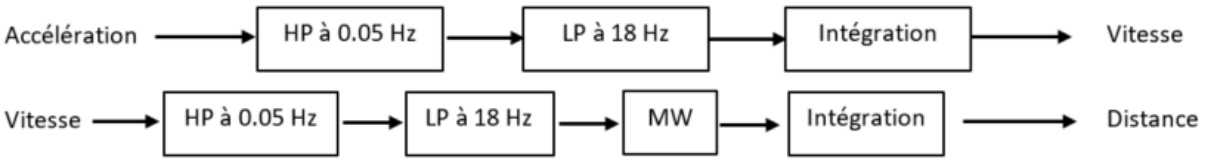


Figure 15 : Étapes de la solution finale

HP : Filtre passe-haut ; LP : Filtre passe-bas ; MW : Moyenne glissante sur X valeurs

Avant d'obtenir la distance parcourue et avant l'intégration on effectue une moyenne glissante pour lisser une dernière fois le signal et faciliter la mise en place d'offset pour le calcul. On aurait pu le faire avant l'intégration de l'accélération également mais on aurait pu perdre trop d'informations pour les étapes suivantes et cela aurait rajouté du temps de calcul.

3.4.3 Résultats des solutions testées

Pour rappel, les données d'accélérations sont très bruitées et ce peu importe l'axe d'observation. On peut également noter ci-dessous que le capteur n'est pas parfaitement droit. De plus, la librairie utilisée renvoie les valeurs en *g* et non pas en m/s². Les données de l'accélération sont donc converties en cm/s² avant de pourvoir être exploitées. L'idée est d'observer les améliorations (ou non) des différentes solutions testées sur le signal. Si le signal est suffisamment exploitable, on essaie aussi de mesurer la distance parcourue.

3.4.3.1 Moyenne Glissante

La Figure 16 et la Figure 17 montre les effets de la moyenne glissance lorsque que le capteur est en mouvement puis immobile. On retrouve en bleu l'accélération sur Y et en rouge il s'agit d'une moyenne glissante sur les 30 dernières valeurs. La moyenne glissante parvient à réduire le bruit ambient et retranscrit plus fidèlement le mouvement transmis (au milieu du signal).

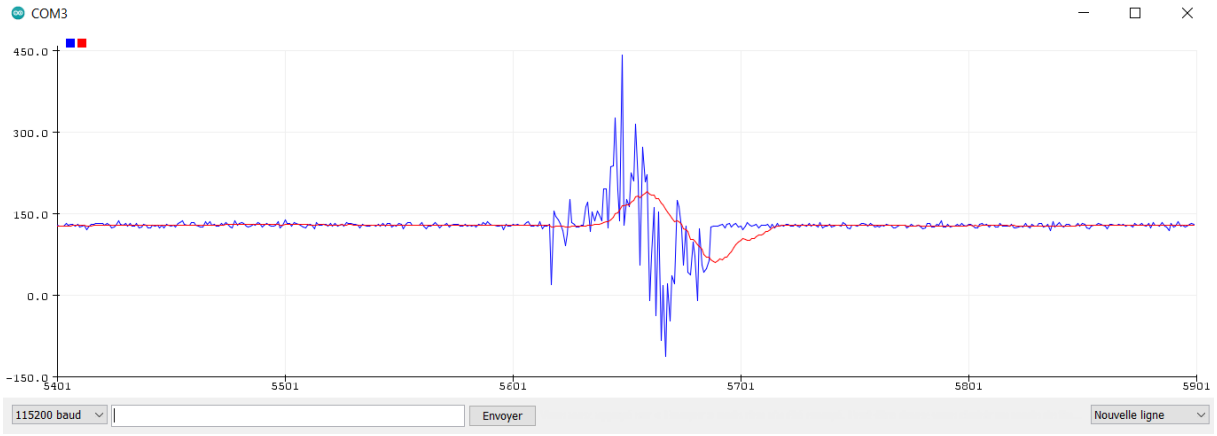


Figure 16 : Effets de la moyenne glissante lors de la détection d'un mouvement

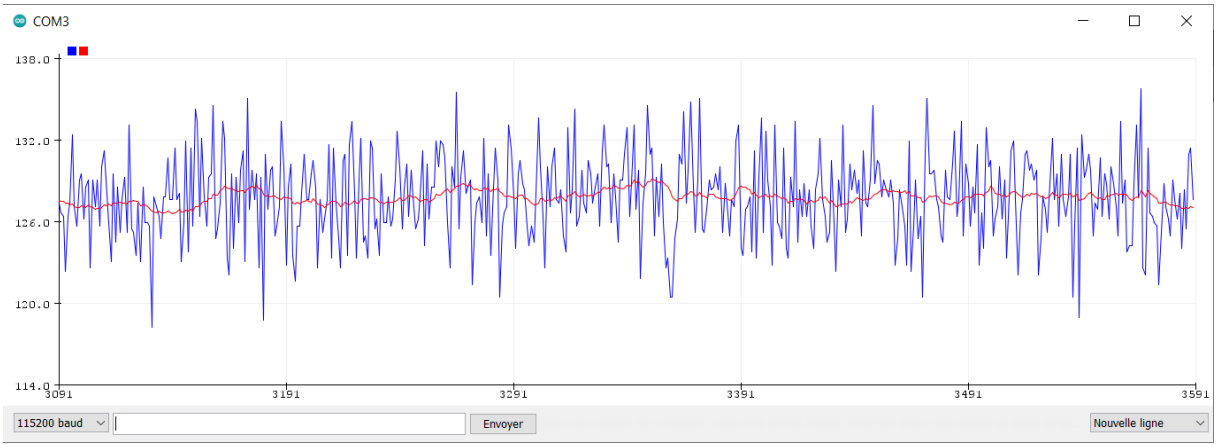


Figure 17 : Effets de la moyenne glissante lorsque le capteur est immobile

Cependant, à lui tout seul, ce n'est pas suffisant pour mesurer une distance. Mais bien qu'il y ait une perte d'informations et qu'il induit un décalage temporel, sur un signal moins bruité, il lisse parfaitement le signal facilitant ainsi son utilisation future. C'est pour cette raison que cette méthode sera beaucoup utilisée dans la suite.

3.4.3.2 Filtre numérique

Ici, on utilise un filtre d'ordre 3, avec un poids K de 0.3. En bleu, on a l'accélération, en rouge le filtre d'ordre 1, en vert d'ordre 2 et en orange d'ordre 3. Sur la Figure 18, est représenté le capteur au repos.

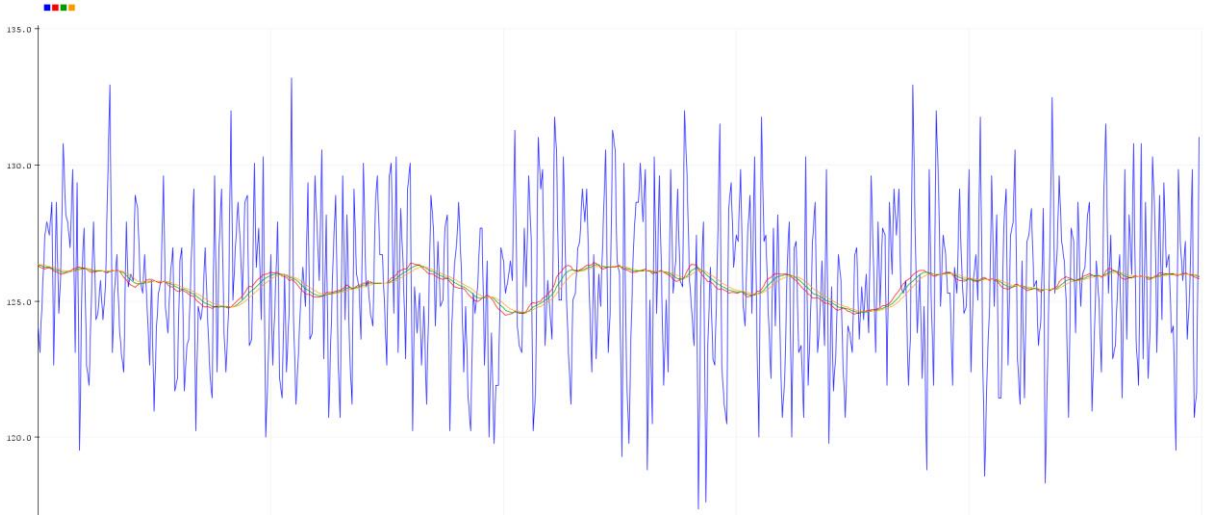


Figure 18 : Effets du filtre numérique, capteur au repos

Sur la Figure 19, on retrouve en bleu l'ordre 1, en rouge l'ordre 2 et en vert l'ordre vert. Un mouvement est appliqué au capteur.

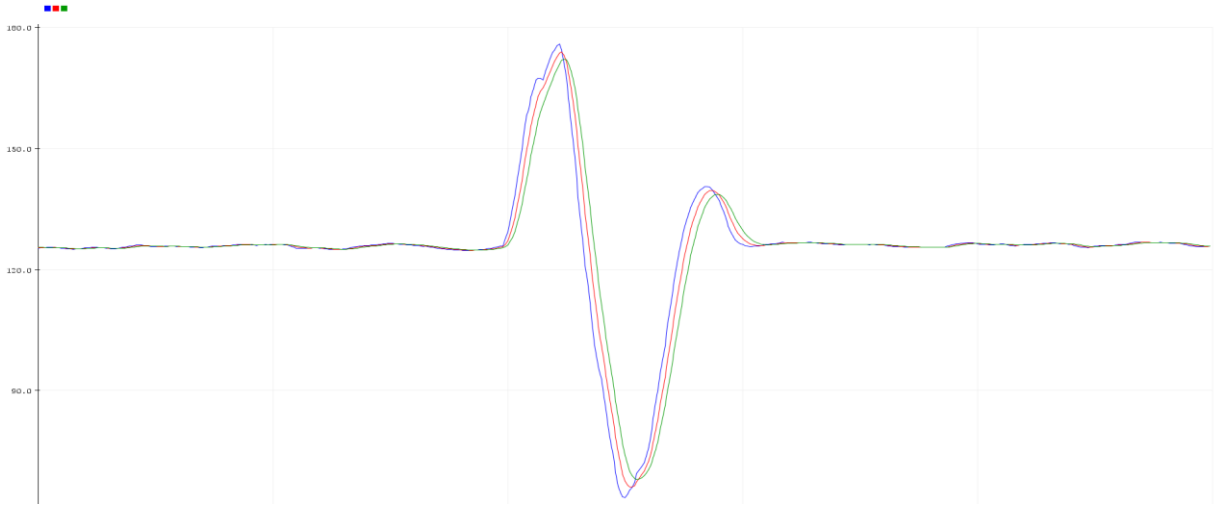


Figure 19 : Effets du filtre numérique lors d'un déplacement du capteur

Finalement, une moyenne glissante et un offset sont appliqués sur le filtre d'ordre 3 pour lisser le signal autour de 0.

Nous avons mesuré les performances de cette méthode et dans les axes positifs les résultats étaient relativement bon avec une erreur de 5 à 7 cm en moyenne sur un déplacement de 15 à 20 cm. Cependant, dans les X ou Y négatifs, la mesure atteignait les 100 à 200 cm pour un déplacement de 15 cm. Après quelques améliorations dans le code, on a pu atteindre une erreur de seulement 2 à 4 cm mais avec toujours le même problème dans le sens négatif.

De plus, on met en place un offset à l'initialisation pour recentrer le signal sur 0 et on a ce qu'on appelle une zone tampon fixé arbitrairement à 10 cm/s². Cette zone tampon génère une petite perte d'information qui se rajoute à l'erreur déterminé précédemment. Mais en dessous de cette valeur, les perturbations étant trop importantes (même quand le capteur ne bouge pas), la distance qu'on mesure augmente même quand on ne bouge pas le capteur.

3.4.3.3 Filtre passe-haut

Ici, on a d'abord mis en place un filtre passe-haut avec une fréquence de coupure à 0.05 Hz. La Figure 20 représente en bleu l'accélération, en rouge le filtre passe-haut et en vert la moyenne glissante du filtre.

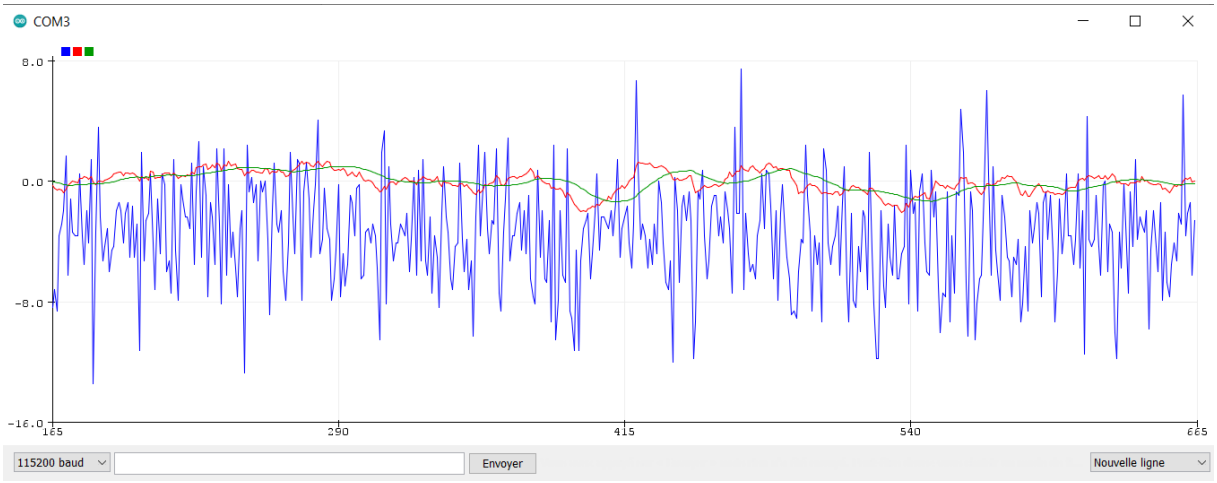


Figure 20 : Effets du filtre passe-haut, capteur au repos

On peut voir que le filtre supprime la composante continue de l'accélération et recentre donc le signal autour de 0. Beaucoup de perturbations restants visibles, on met en place une moyenne glissante.

Sur la Figure 21 on peut voir que lorsqu'on applique un mouvement, on voit une petite perte d'informations qu'on corrige à l'aide d'un facteur qu'on détermine avec des tests. Avec cette méthode, on observe une erreur de l'ordre de 4 à 5 cm sur une distance de déplacement de 15 cm.

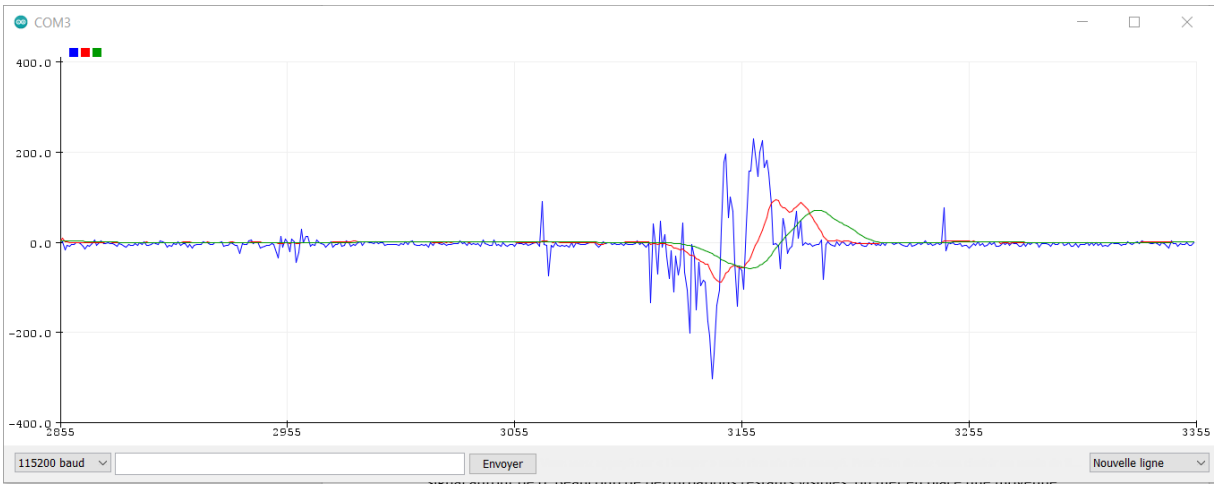


Figure 21 : Effets du filtres passe-haut lors du déplacement du capteur

3.4.3.4 Filtre passe-bas

La Figure 22 illustre les effets du filtre passe-bas avec en bleu l'accélération et en rouge le filtre passe-bas avec une fréquence de coupure fixé à 20 Hz.

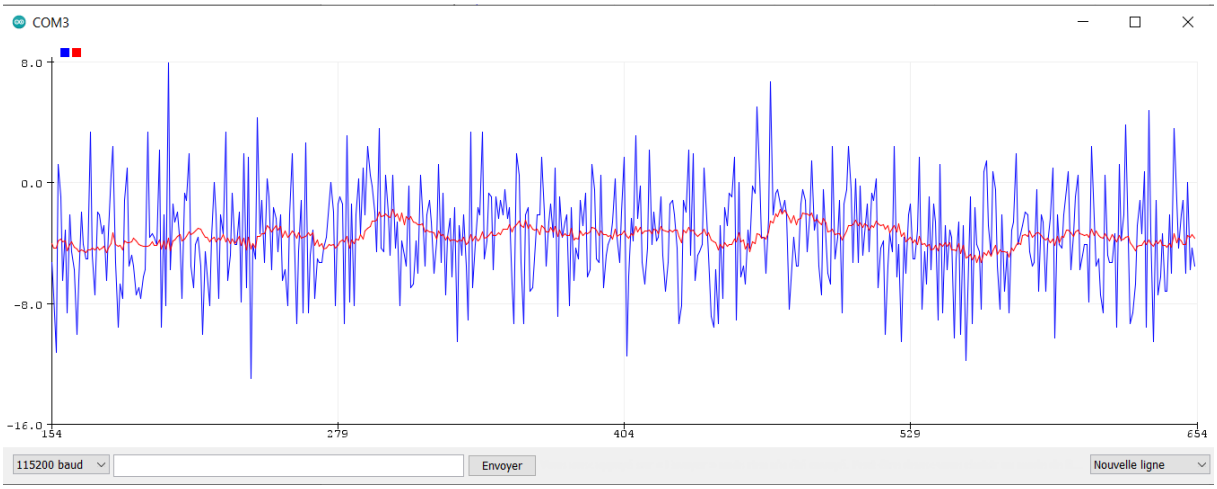


Figure 22 : Effets du filtre passe-bas, capteur au repos

On peut noter que le filtre passe-bas ne recentre pas le signal et reste bruité. Comme on peut le voir sur la Figure 23 ci-dessous, le filtre retranscrit bien un mouvement donné au capteur. Cependant, les tests n'ont pas été poussés plus loin avec le filtre passe-bas uniquement, le filtre passe-haut étant plus intéressant.

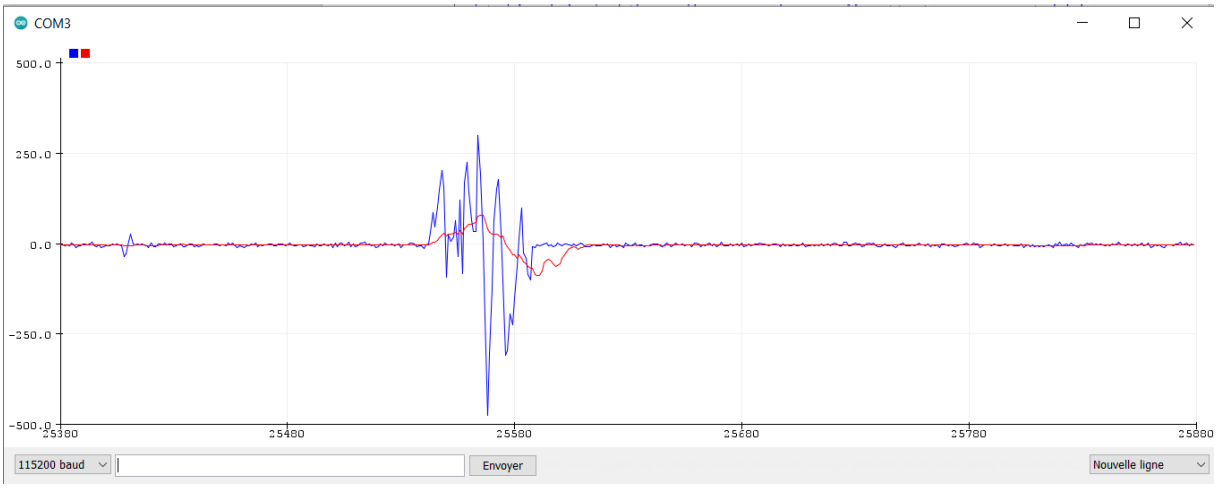


Figure 23 : Effets du filtre passe-haut lors du déplacement du capteur

3.4.3.5 Filtre de Butterworth

Dans le cas du filtre de Butterworth, bien d'attrayant sur le papier, sa mise en place révèle de nombreux problèmes. On a toujours besoin d'un offset pour le center sur 0 et surtout, comme illustré sur la Figure 24, il ne retranscrit pas assez fidèlement le mouvement qui a été transmis.

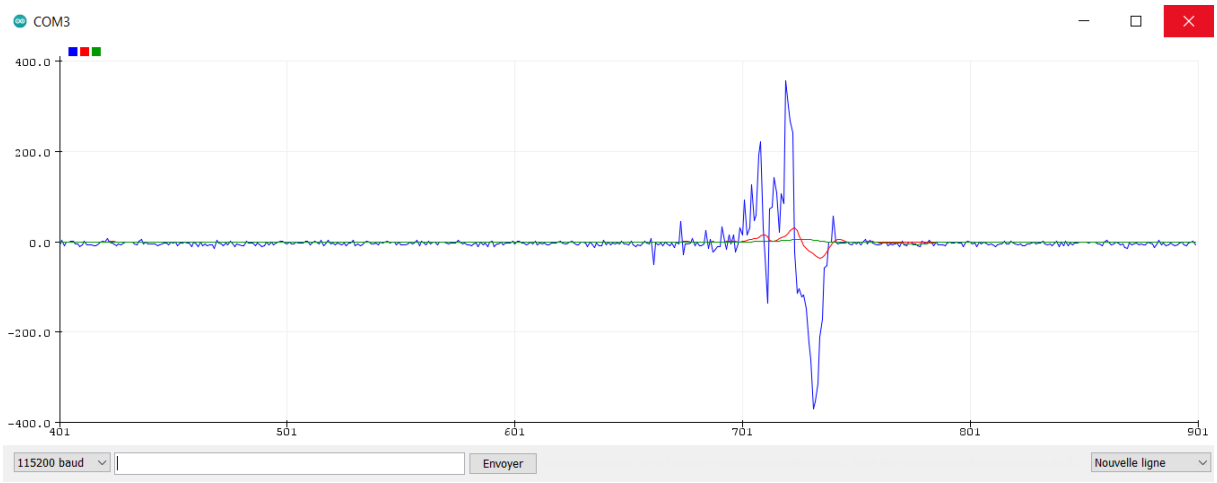


Figure 24 : Effets du filtre de Butterworth lors du déplacement du capteur

Pour toutes ces raisons, aucun test supplémentaire n'a été réalisé avec filtre. De plus, sa mise en place est plus complexe que les autres filtres.

3.4.3.6 Solution retenue

Pour rappel la solution retenue est la combinaison du filtre passe-haut, du filtre passe-bas et de la moyenne glissante comme décrite dans la partie précédente.

Les courbes d'accélération sont les mêmes que pour les parties précédentes et ne seront pas affichées de nouveau. De plus, les nombreuses étapes intermédiaires gardent toutes les informations essentielles jusqu'au tout dernier moment afin de minimiser les erreurs. Il n'y a donc pas beaucoup d'intérêt à toutes les afficher ici.

La seule courbe intéressante ici, est celle de la vitesse juste avant l'intégration finale visible sur la Figure 25. Il faut noter que l'échelle en cm/s n'est pas la bonne car un facteur multiplicatif est appliqué pour faciliter la mise en place d'une zone tampon plus précise.

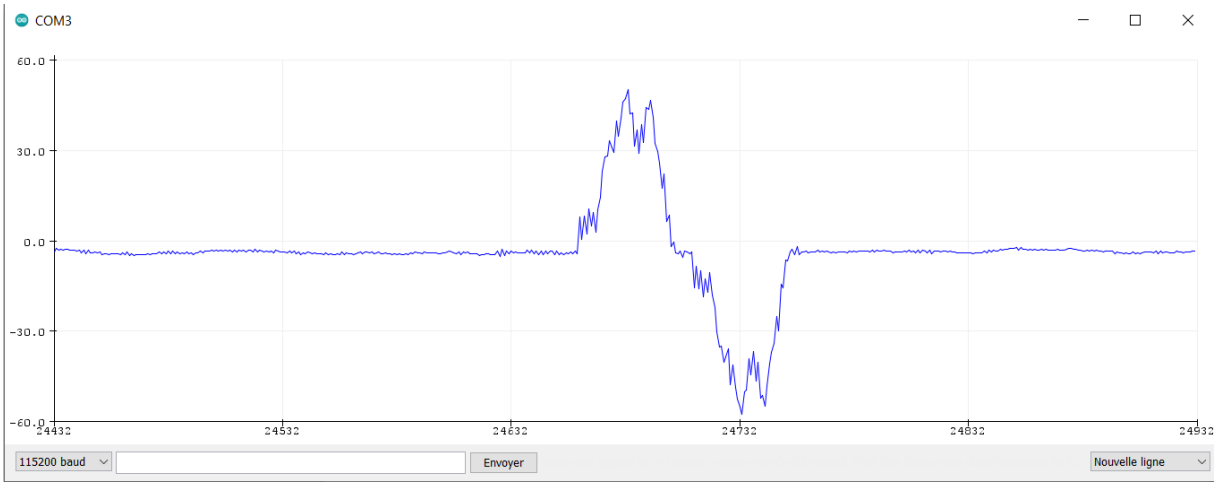


Figure 25 : Visualisation de la vitesse lors du déplacement du capteur

Lors des premières mesures de tests, on a pu atteindre une erreur moyenne de 1 à 3 cm sur une distance de déplacement de 15 cm.

3.4.4 Limites de la solution

Le capteur à une certaine sensibilité que l'on peut régler mais il faut trouver un juste milieu. En effet, si le capteur bouge trop doucement, la distance sera sous-estimée. A l'inverse s'il bouge trop rapidement, elle sera surestimée. C'est le cas pour la solution que nous avons retenu tout comme celles que nous avons testés.

De plus, toute inclinaison du capteur faussera logiquement les résultats. Cela est dû au fait que les tests et les algorithmes ont été écrits pour une surface plane pour le moment. Mais ce comportement est « normal » car l'inclinaison du capteur induit une accélération plus ou moins constante sur ses axes et donc un décalage du zéro. Cependant, l'utilisation du filtre passe-haut corrige en partie ce problème. Seulement en partie, car il y a un temps de stabilisation proportionnel à l'angle du capteur. C'est-à-dire que plus l'angle du capteur augmente rapidement plus le filtre mettra du temps pour se stabiliser autour de 0 à nouveau.

Ensuite, le bruit du signal à tendance à augmenter avec le temps rendant la mesure de moins en moins précise. Cependant, une simple remise à zéro du capteur suffit pour le rendre à nouveau opérationnel.

Avec un accéléromètre grand public comme le GY-521, il sera difficile d'arriver à une meilleure précision. Néanmoins, c'est la seule option viable tant par sa taille très réduite que par son coût très faible également.

Finalement, comme son nom l'indique un accéléromètre mesure l'accélération qui est elle-même dérivé de la vitesse. Par conséquent, peu importe la méthode utilisée, la mesure de distance ne sera fiable que dans un cours intervalle puisque lorsqu'on aura atteint une vitesse constante avec le robot, l'accélération sera nulle et les intégrations successives donneront donc une distance parcourue de 0 cm.

3.5 Banc de test

Le but du banc de test que j'ai créé est simple et est de maintenir le capteur en place tout en bloquant les axes inutiles.

L'accéléromètre étant un capteur très sensible, il est nécessaire de le garder parfaitement plat et immobile notamment au début pour la calibration et la mise en place des offsets. Il faut donc un support pour accueillir le capteur et le maintenir en place.

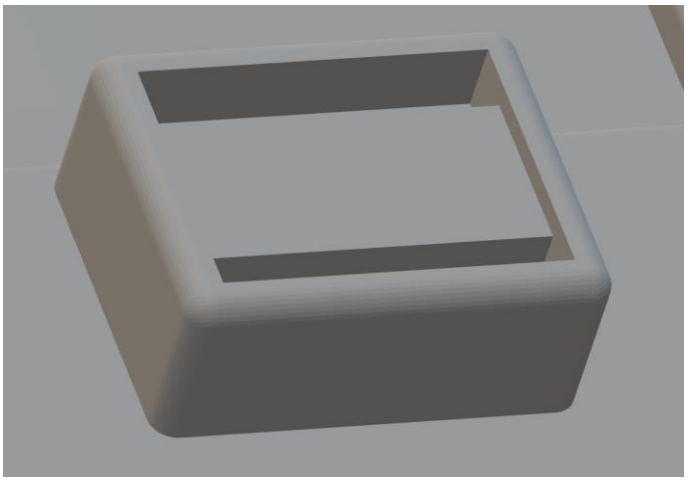


Figure 26 : Emplacement du capteur

Ainsi, la forme illustrée par la Figure 26 ci-dessus, conçue en 3D assure cette tâche et sera placé sur les bancs de tests finaux.

L'ensemble des fichiers (au nombre de 4 pour le moment) a été conçu à l'aide du logiciel de modélisation en 3D Fusion 360 d'Autodesk et sera imprimé avec une imprimante 3D. Ces pièces sont néanmoins susceptibles d'évoluer rapidement (notamment dans les dimensions) selon les besoins mais le principe de fonctionnement restera le même.

Au total, il y a donc 2 pièces pour la mesure de l'angle et 2 pièces pour la mesure de la distance. Dans les deux cas il y a une pièce mobile et une pièce fixe qui sert de référentiel.

Pour la mesure de l'angle, on a donc sur la Figure 27 une base ronde graduée tous les 45° et qui sera fixée à une table avec du scotch par exemple.

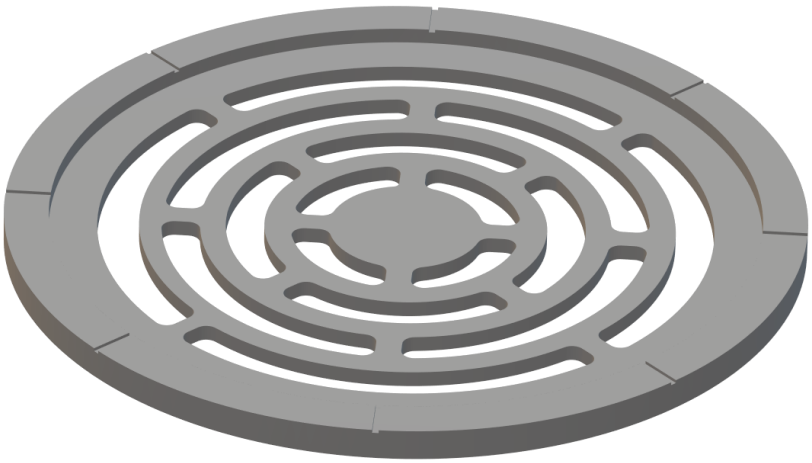


Figure 27 : Base pour le banc de test de l'angle

La pièce comporte des trous simplement par soucis d'économie de plastique et pour réduire le temps d'impression de la pièce. À l'intérieur de cette pièce, on place la pièce présente sur la Figure 28 ci-dessous.

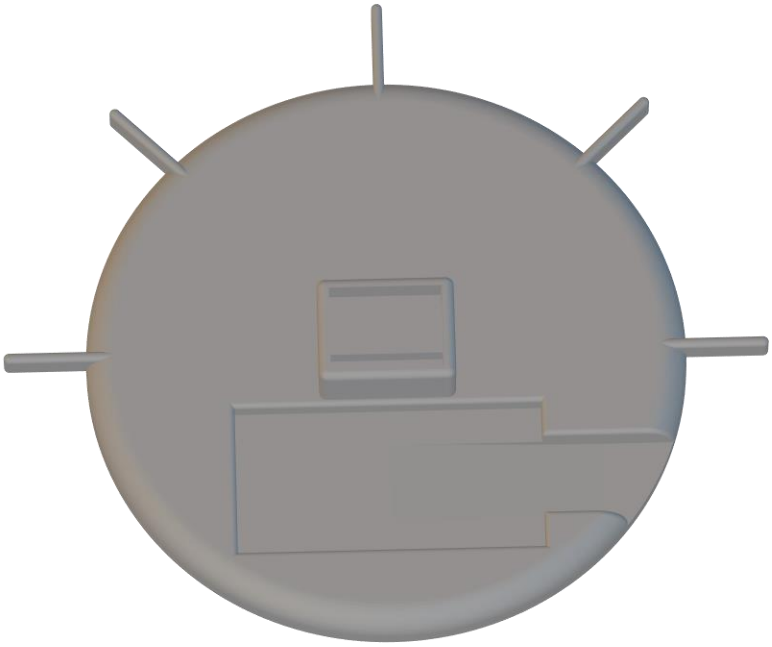


Figure 28 : Band de test de l'angle

Cette pièce sera libre uniquement en rotation autour de l'axe Z pour éviter les déplacements involontaires de la main lorsque l'on bouge la pièce. On y retrouve un socle pour le capteur au centre même de la pièce. Un espace est prévu sur le bas pour accueillir un ESP32 et son câble d'alimentation. Enfin, des « mini-bras » sont placés sur la partie haute et sont espacés de 45°. Le but est de faciliter le mouvement de la pièce et d'augmenter la pertinence du banc de test car ils permettent de se déplacer d'un angle fixe.

De la même manière que pour la mesure de l'angle, pour la distance on a une base, rectangulaire cette fois-ci et graduée tous les centimètres. La pièce illustrée par la Figure 29 ci-dessous est néanmoins qu'une partie de la base. L'idée est d'en imprimer plusieurs, de les mettre bout à bout fixés sur une table et ainsi atteindre une distance d'au moins un mètre.

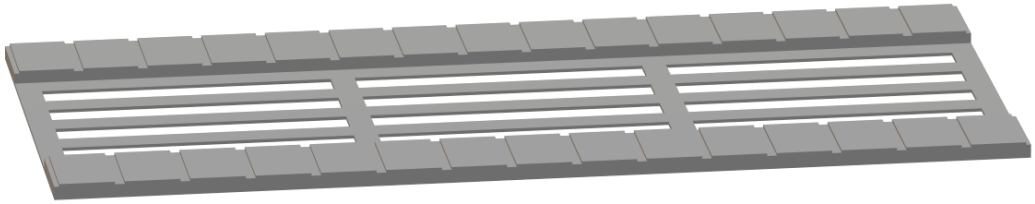


Figure 29 : Base banc de test pour la distance

La pièce fait donc 15 cm de long et est conçue pour que les graduations puissent être continues en ajoutant des pièces bout à bout. Tout comme la base pour la mesure de l'angle, les trous sont là pour réduire la quantité de plastique et diminuer le temps d'impression.

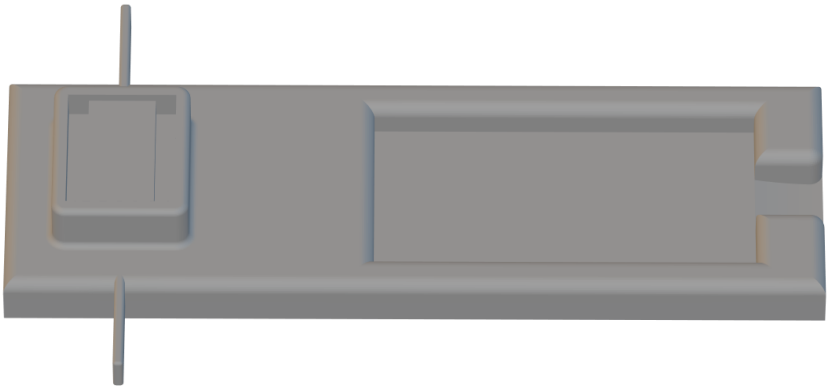


Figure 30 : Banc de test pour la distance

La pièce sur la Figure 30 accueille donc les mêmes composants que pour la mesure de l'angle mais placés différemment. Les « mini-bras » pointent vers le milieu du capteur et permettront de connaitre la distance de déplacement à l'aide des marques sur la base.

3.6 Conclusion et perspectives

Assurer un suivi de distance et de mesure d'angle n'est pas une tâche facile avec un accéléromètre grand public et ne disposant que de 6 degrés de liberté. En effet, c'est un capteur donc les sorties sont relativement bruitées et dont les signaux sont donc difficilement exploitables en l'état. Cependant, grâce à la fusion de données entre les données de l'accéléromètre et du gyroscope, il est possible d'obtenir des mesures d'angles précises presque au degré près pour le roulis et le tangage à l'aide du filtre de Kalman. En ce qui concerne le lacet, il est possible, en utilisant la même méthode, d'approximer l'angle de rotation mais contrairement au roulis et au tangage, il dépendra fortement de la vitesse de rotation qu'on applique au capteur. Pour contrer ces effets, il faut intégrer un magnétomètre à la fusion de données. De cette manière, on obtiendra un angle très précis et les perturbations extérieures auront un impact limité.

En revanche, la détermination de la distance s'est avéré plus compliqué que prévu. En effet, bien que très fiable, le filtre de Kalman ne permet pas de terminer une position et donc une distance. Il a donc fallu passer par d'autres méthodes n'incluant pas la fusion de données car les données du gyroscope ne sont pas pertinentes pour la mesure de distance.

Plusieurs filtres ont donc été testés pour rendre utilisable les données de l'accéléromètre. Cependant, même si on a montré que c'était possible, l'erreur reste grande et le système est fortement sensible aux perturbations extérieures. Il n'existe pas à ce jour de moyens beaucoup plus fiables de mesure de la distance dans le budget du projet. Un tel appareil couterait plusieurs milliers d'euros et surtout serait beaucoup trop gros pour être embarqué. Le type de capteur que nous disposons actuellement est donc parfaitement adapté pour cette partie du projet. De plus, la mesure de distance ne peut être effectuée que sur de courtes périodes du fait de la nature même de ce type de capteur. D'autres solutions seront donc à explorer pour cette problématique.

Une des idées à la base était d'utiliser le filtre de Kalman pour compenser le décalage de la variante continue de l'accélération mais le filtre passe-haut supprimant cette composante continue rend cette solution inutile au final.

Plus de tests devront être menés pour tenter d'améliorer la précision et de réduire l'impact des perturbations extérieures. Certains problèmes ont d'ores et déjà été réglé mais la fiabilité de la solution mise en place sur le long terme est mise en question.

CHAPITRE 4. ÉVOLUTION DU PROJET

Cette partie présente plus en détail les évolutions du projet telles que décrites au début de ce rapport et présente les étapes essentielles ainsi que les résultats finaux obtenus. Peu de changement ont été apportés à la partie mesure de distance mais nous reviendrons sur ses limites.

Pour rappel, on est passé sur un nouveau capteur qui dispose maintenant d'un magnétomètre. La partie suivante décrit une partie essentielle lors de l'utilisation d'un magnétomètre qui est sa calibration.

4.1 Calibration du magnétomètre

La calibration du magnétomètre est une étape essentielle afin d'obtenir une donnée fiable en sortie d'algorithme. On passe alors par trois étapes distinctes, le « Hard Iron », le « Soft Iron » et le « tilt compensation » ou la compensation d'inclinaison [13].

La distorsion du Hard Iron est produite par des matériaux qui présentent un champ additif constant au champ magnétique terrestre, générant ainsi une valeur additive constante à la sortie de chacun des axes du magnétomètre. Une distorsion en fer dur peut être identifiée visiblement par un décalage de l'origine du cercle idéal par rapport à (0, 0) comme illustré sur la Figure 31 ci-dessous.

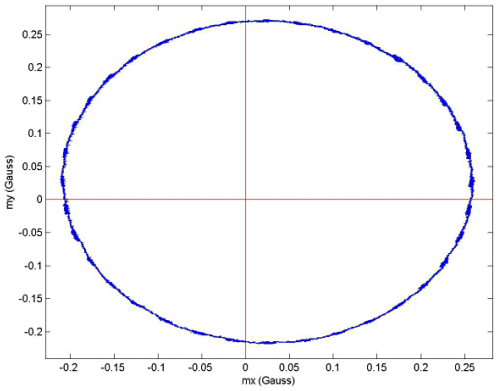


Figure 31 : Impact du Hard Iron

Si aucun effet de distorsion n'est présent, la rotation d'un magnétomètre sur un minimum de 360° et le tracé des données résultantes en fonction de l'axe des y par rapport à l'axe des x donneront un cercle centré sur (0, 0) ce qui n'est pas le cas ici. Pour corriger cette distorsion, il suffit de calculer et d'appliquer les offsets sur X et Y. Il faut obligatoirement appliquer la compensation d'inclinaison avant de déterminer les offsets. Pour calculer cet offset, il faut prendre la moyenne entre la valeur maximum et minimum de la valeur du champ au cours d'un ou plusieurs tours complets du capteur.

Contrairement à la distorsion du Hard Iron, où le champ magnétique s'ajoute au champ terrestre, la distorsion du Soft Iron est le résultat d'un matériau qui influence, ou déforme, un champ magnétique, mais qui ne génère pas nécessairement un champ magnétique lui-même, et n'est donc pas additif. Le fer et le nickel, par exemple, génèrent une distorsion du Soft Iron.

Alors que la distorsion du fer dur est constante quelle que soit l'orientation, la distorsion produite par les matériaux en fer doux dépend de l'orientation du matériau par rapport au capteur et au champ magnétique. Ainsi, la distorsion du fer doux ne peut pas être compensée par une simple constante, une procédure plus complexe est nécessaire.

Comme le montre la Figure 32 ci-dessous, une déformation en Soft Iron se manifeste généralement par une perturbation du cercle idéal en une ellipse.

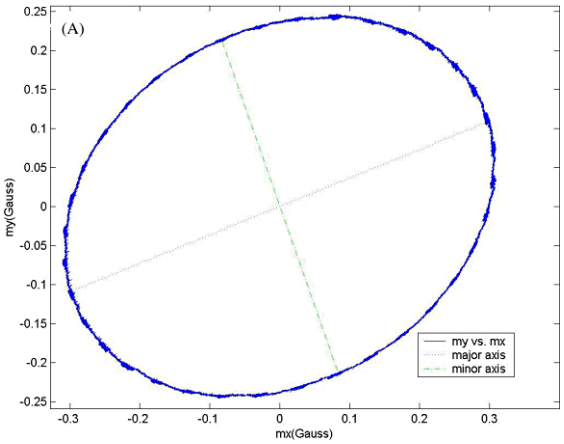


Figure 32 : Impact du Soft Iron

Dans notre cas, c'est-à-dire un capteur fixé sur une surface plane, seul le Hard Iron peut être déterminé facilement sur X et Y mais pas sur Z. En effet, il faudrait pouvoir effectuer des rotations stables autour de l'axe X ou Y ce qui n'est pas faisable dans notre cas. Le problème est le même pour le Soft Iron. Dans les deux cas, il faut effectuer des rotations du capteur dans les trois dimensions. Le concepteur de nos capteurs, Adafruit, nous fournit un outil qui nous permet de calculer ces distorsions. Les points rouges sont nos points de mesures dans l'espace. En vert, on retrouve le vecteur pour le Hard Iron et en bleu la matrice pour le soft Iron. La Figure 33 ci-dessous montre les résultats obtenus.

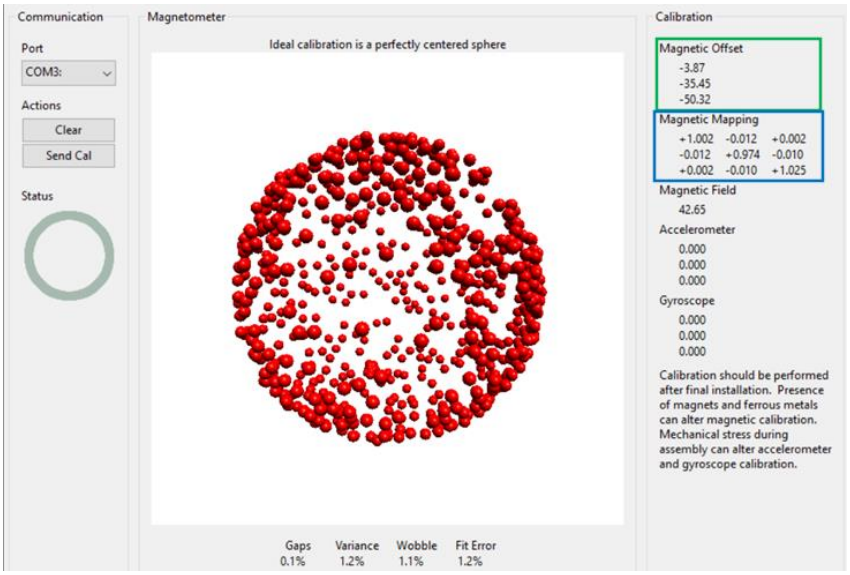


Figure 33 : Outil de calibration d'Adafruit

On applique alors l'Équation 2 ci-dessous pour terminer la calibration du magnétomètre :

$$\begin{bmatrix} m_{c_x} \\ m_{c_y} \\ m_{c_z} \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \tilde{m}_x - b_{H_0} \\ \tilde{m}_y - b_{H_1} \\ \tilde{m}_z - b_{H_2} \end{bmatrix}$$

Équation 2 : Formule de calibration du magnétomètre

La matrice *C* correspond à la matrice bleue dans la figure ci-dessus et les coefficients *b* au vecteur vert.

Enfin, nous devons appliquer la compensation d'inclinaison. C'est un algorithme qui prend en entrée le roulis (roll) et le tangage (pitch) pour compenser toute inclinaison du capteur. Sans cette étape, on peut se retrouver avec un angle de sortie pour le lacet (yaw ou heading) qui varie de 100°. En l'appliquant et sous réserve d'avoir une bonne calibration, on peut descendre à une erreur inferieure à 2°. L'algorithme se matérialialise de la façon suivante sur la Figure 34.

```
1 double heading(double roll, double pitch) {
2   magX = magRX * cos(pitch) + magRY * sin(roll) * sin(pitch) - magRZ * cos(roll) * sin(pitch);
3   magY = -magRY * cos(roll) + magRZ * sin(roll);
4   return atan2(-magY, magX) * RAD_TO_DEG;
5 }
```

Figure 34 : Code pour déterminer l'angle absolu

4.2 Résultats avec 9 degrés de liberté

Les tests ici ont été réalisés à l'aide des capteurs FXOS8700 et FXAS21002C décris plus haut dans le document. Il faut noter que l'ensemble des 9 axes des capteurs sont nécessaires pour obtenir une donnée fiable.

Les résultats de cette partie ont été obtenus après calibration du magnétomètre et la compensation d'inclinaison. La Figure 35 ci-dessous montre le capteur au repos. On peut voir que la variation d'angle reste relativement bien contenue.

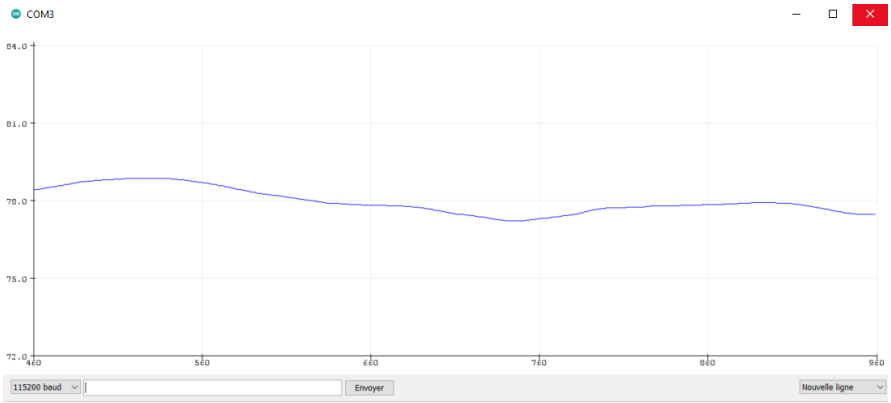


Figure 35 : Sortie de filtre de Kalman après traitement pour l'angle de lacet

On peut observer son évolution au cours du temps et calculer l'écart à la moyenne. Sur un échantillon de près de 1400 mesures, on observe sur la Figure 36 une moyenne à 78.90° et un écart relatif par rapport à cette moyenne de 0.34 %.

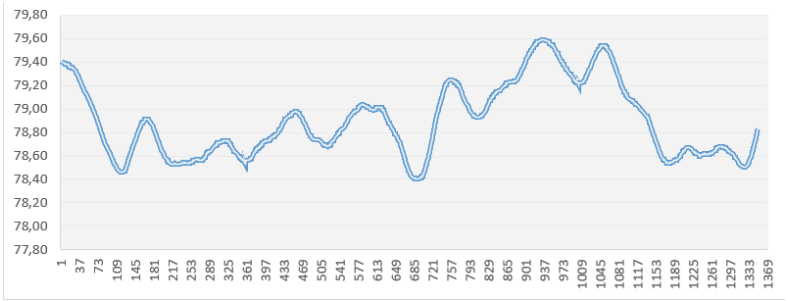


Figure 36 : Sortie du filtre, capteur au repos

Les résultats en sortie du filtre de Kalman sont moins nets que pour les axes X et Y mais restent cependant fiable pour nos mesures. La Figure 37 montre la réponse de notre filtre pour un ordre d'angle aléatoire.

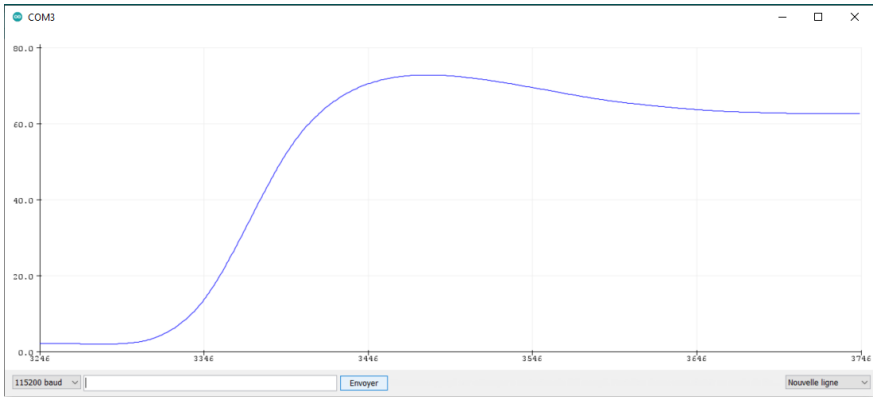
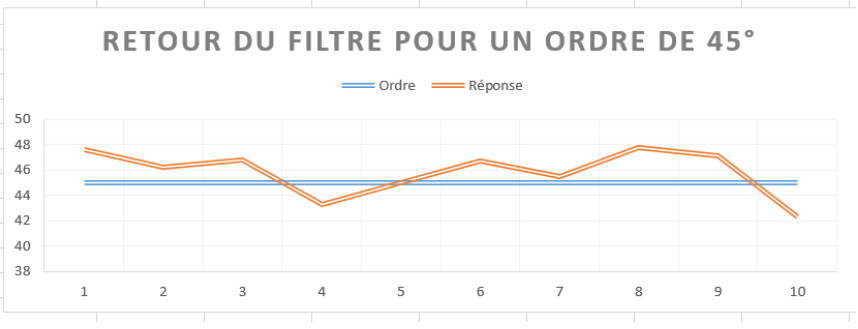


Figure 37 : Réponse du filtre à un ordre aléatoire

On note qu'une fois le mouvement terminé le signal se stabilise très rapidement. Nous avons également effectué une série de tests sur de petits échantillons pour des ordres donnés de 45 puis 90° afin de calculer nos erreurs de mesures.

Ordre	Réponse	Erreur
45	47,6	5,78%
45	46,2	2,67%
45	46,8	4,00%
45	43,3	3,78%
45	45	0,00%
45	46,7	3,78%
45	45,5	1,11%
45	47,8	6,22%
45	47,1	4,67%
45	42,3	6,00%
Erreur moyenne :		**3,80%**



Ordre	Réponse	Erreur
90	91,2	1,33%
90	93,5	3,89%
90	91	1,11%
90	87,3	3,00%
90	92	2,22%
90	92,4	2,67%
90	89,2	0,89%
90	89,9	0,11%
90	95,1	5,67%
90	92	2,22%
Erreur moyenne :		**2,31%**

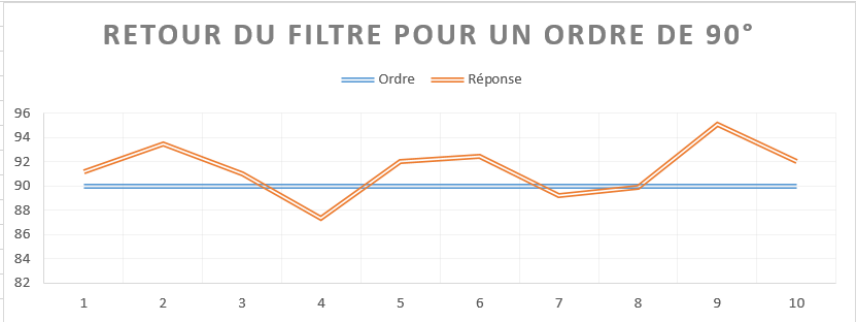


Figure 38 : Retour du filtre pour un ordre de 45 puis 90°

On observe alors sur la Figure 38 ci-dessus une erreur moyenne absolue de respectivement 3.8 et 2.8%. Il faut noter que ces erreurs peuvent éventuellement être diminuées en respectant un temps de stabilisation du signal un peu plus long. Cette fois, la vitesse de rotation n'impacte plus les mesures comme c'était le cas avec 6 degrés de liberté.

En conclusion, le passage de 6 à 9 degrés de liberté sur notre capteur a été payant car il nous a permis d'obtenir un angle de lacet absolu et indépendant des perturbations extérieures telles que l'inclinaison ou la vitesse de rotation.

4.3 Utilisation de ROS

Le travail de conception d'un système robotique entier nécessite une grande quantité de compétences dans des domaines bien distincts. Nous avons donc cherché à trouver une solution nous permettant de gagner du temps sur le développement du prototype fonctionnel.

Pour cela, nous nous sommes donc tournés vers la conception d'un système robotique en utilisant ROS (Robot Operating System), nous permettant de définir une architecture plus facilement et de réutiliser des briques robotiques déjà existantes pour gérer les paramètres sur lesquels nous ne prévoyons pas de faire de la recherche (lois physiques de déplacement du robot, gestion du déplacement suivant une commande, télé opération du prototype, etc..).

L'environnement ROS nous a également permis de faciliter l'entrée de certains profils sur le projet car l'ensemble du projet n'est plus dépendant d'un seul langage informatique. En effet, l'environnement ROS nous a permis de pouvoir utiliser plusieurs langages pour développer des blocs fonctionnels différents. Son fonctionnement est analogue à celui de MQTT, c'est-à-dire qu'il y a des topics dans lesquels on peut souscrire et/ou publier une information. La Figure 39 [14] montre l'architecture finale du projet.

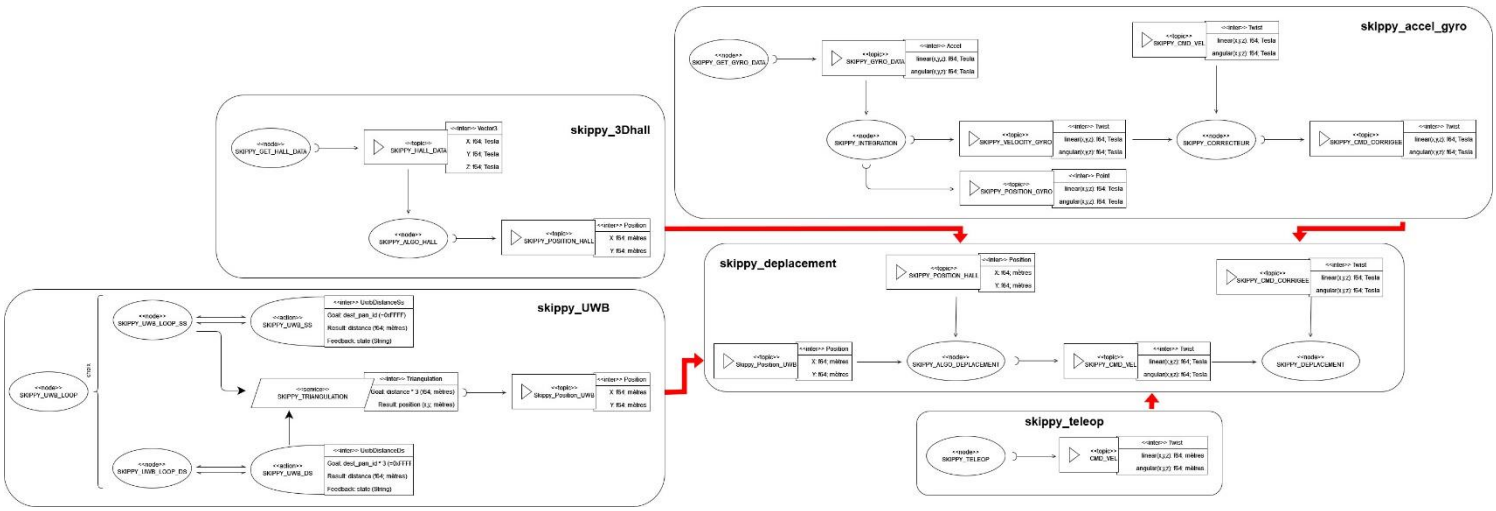


Figure 39 : Architecture finale du projet

Concrètement, il me suffit de publier mes données calculées sur des topics prédéterminés pour qu'ils soient automatiquement disponibles pour tous les nodes qui ont souscrit aux topics. Chaque node à l'avantage de pouvoir être codé dans beaucoup de langages différents.

4.4 Code

Au cours du stage, les principaux tests ont été effectués en C++ avec l'IDE Arduino. L'intégration finale ainsi que le développement du driver pour mon capteur ont été réalisés en Rust. C'était un langage que je ne connaissais que de nom avant de commencer mon stage et que j'ai donc appris au tout début. Le but était de pouvoir développer dans le même langage que l'équipe. Il m'a également permis de pouvoir aider et participer plus facilement au projet.

L'ensemble du code écrit pendant le stage est disponible sur mon GitHub (florianleon) [15] et la documentation des deux librairies sera en annexe. Le code disponible sur GitHub ne sera disponible publiquement uniquement jusqu'au jour de ma soutenance. Le code des librairies sera quant à lui disponible sur datafusion_imu [16] et adafruit_nxp [17] sur crates.io.

CHAPITRE 5. AUTRES RÉALISATIONS

5.1 Impression 3D

Accompagné d'une autre personne de l'équipe, j'ai apporté mon expertise dans l'impression 3D. En effet, c'est une compétence que j'ai acquise sur mon temps libre depuis plus d'un an maintenant et qui a été utile durant ce projet. SII dispose d'une imprimante 3D dans les locaux qui est accessible aux employés.

Elle nous permet de prototyper rapidement en créant divers objets pour intégrer nos capteurs. En témoigne les pièces que j'ai dessiné puis imprimé pour pouvoir faire des mesures fiables ou du moins éliminer le plus possible les perturbations extérieures.

De plus, l'imprimante 3D de SII a été mal entretenue depuis le début du Covid, il a donc été nécessaire de la remettre en état. Ainsi, elle pourra servir sans souci particulier pour les futurs prototypes du Lab Embarqué. Elle est notamment utilisée pour intégrer tous les capteurs du projet sur le même robot. On peut voir sur la Figure 40, la base violette qui intègre les capteurs du projet.

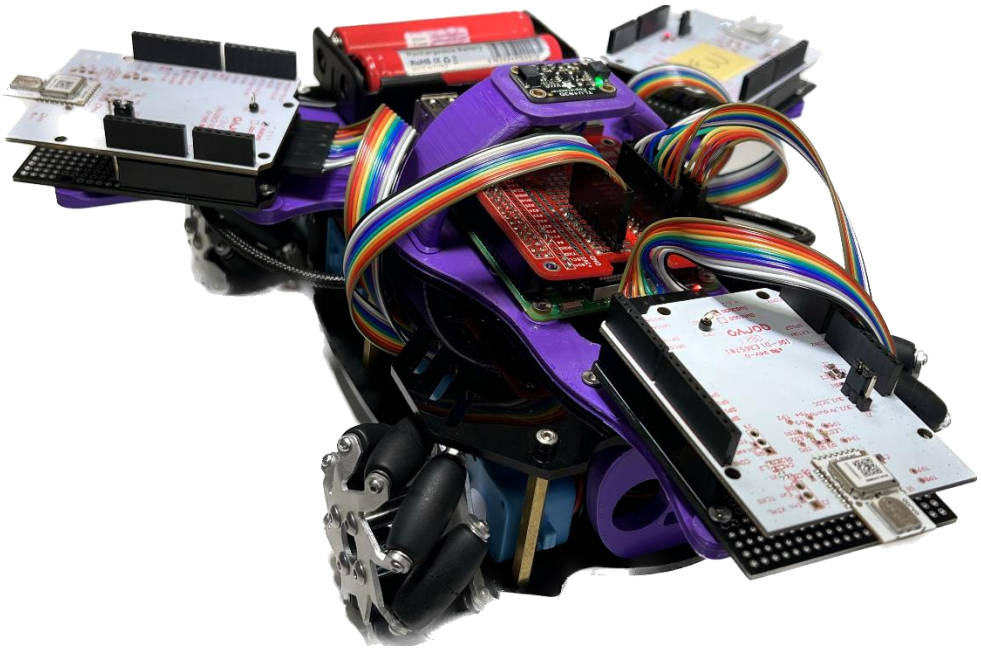


Figure 40 : Skippy, le robot du lab Embarqué

5.2 Aides diverses

En complément de mes taches, j'ai apporté mon aide, en particulier sur la fin de mon stage, sur la partie support de mon code. En effet, une fois mes parties terminées, il a fallu les intégrer aux autres parties en commençant par le capteur à effet hall. L'idée est de déterminer si nos deux capteurs peuvent cohabiter sur le même robot. Pour rappel, le magnétomètre est censé capter l'aimant à une plus grande distance que le capteur à effet hall. Les premiers tests ont confirmé cette hypothèse. Cependant, l'aimant trop puissant, a abimé de façon permanente le capteur lors des tests de détections reportant ainsi son intégration finale. Le capteur n'était alors plus capable de

détecter le champ terrestre correctement et les données d'angles calculées étaient donc erronées. Bizarrement, les valeurs d'angles étaient quand même correctes lorsque le champ terrestre devenait très négligeable devant la puissance de l'aimant. Pour cette partie, nous nous sommes donc limités à l'écriture et aux tests du code qui intègre les deux capteurs dans l'architecture ROS.

Enfin, j'ai contribué à l'écriture du dossier Crédit Impôt Recherche ou dossier CIR. Ce dernier est un élément essentiel à la survie d'un lab de R&D. En effet, c'est grâce à ce dossier que l'État décide ou non d'accorder un financement à un projet. Il comporte les choix techniques et technologiques du projet ainsi que les différents verrous adressés qui justifient la mise en place d'un tel projet.

CHAPITRE 6. CONCLUSIONS ET PERSPECTIVES

En conclusion, ce stage a été un réel atout autant sur le plan personnel que professionnel. Tout d'abord sur le plan personnel, j'ai fait la rencontre d'une équipe formidable avec laquelle j'ai pris plaisir à travailler chaque jour. Cette même équipe m'a permis d'atteindre mes objectifs et de participer activement à ce projet.

Mes objectifs ont donc été atteints bien qu'ils soient à nuancer partiellement. En effet, le capteur dont je disposais au début du projet n'était pas suffisant pour remplir l'ensemble des objectifs. Cependant, j'ai pu être en mesure de faire les tests et recherches nécessaires pour rédiger un état de l'art et ainsi poser les bases de ma partie du projet. Ensuite, si certaines parties n'ont pas évolué à l'image du calcul de la distance, d'autres ont été complétement imaginés et revu en cours de projet comme l'intégration de mon capteur qui s'est fait finalement via ROS.

Avec le nouveau capteur, la mesure d'angle est stable et surtout absolue. On ne dépend plus d'un mouvement mais simplement d'une position pour calculer l'angle. La mesure peut donc être réalisée à n'importe quel instant. La décision de passer sur une capteur 9 axes a donc été un choix déterminant pour la réussite de cet objectif. Malheureusement, la panne du capteur dans les derniers jours de mon stage ont mis fin prématurément aux tests.

La mesure de la distance est le point à nuancer sur les objectifs. En effet, bien que certains tests se soient révélés concluant, l'environnement extérieur est un facteur de perturbation trop important. Peu importe les filtres mis en place ou l'approche utilisée et quand bien même les courbes sont exploitables, la mesure a tendance à diverger si des zones tampons ne sont pas mises en place. De plus, la mesure de distance dépend fortement de la vitesse du capteur. Enfin, lorsque la vitesse devient constante, donc sur des distances relativement grandes (supérieure à 50cm), la mesure n'est tout simplement plus possible. Pour toutes ces raisons, ma conclusion sur la mesure de distance est qu'elle reste peu exploitable en l'état sur des capteurs grands publics.

Grâce à ce stage j'ai pu redécouvrir encore plus le monde du développement embarqué dont j'ai pu avoir un aperçu lors de mon projet innovant. C'est aussi pour cette raison que j'ai décidé de continuer dans ce domaine à la suite de mon stage. J'ai également pu apprendre un nouveau langage de programmation, le Rust et d'améliorer mes compétences en C++.

Toutes ces compétences nouvellement acquises ou améliorées me permettent maintenant d'aborder le début de ma future carrière avec plus de confiance et dans un domaine qui me plait.

CHAPITRE 7. ANNEXES

7.1 Driver Adafruit NXP

Adafruit NXP sensor driver

This is a driver for the Adafruit Precision NXP 9-DOF Breakout Board - FXOS8700 + FXAS21002C. It is based on the libraries provided by Adafruit in C++ for the FXAS21002C and the FXOS8700.

`embedded_hal` based driver with i2c access to the Adafruit chip. It provides a basic interface to get raw and scaled data from the sensors. All 3 sensors are separated in their own structure, respectively `Accelerometer`, `Magnetometer` and `Gyroscope`.

Calibration

You might need to calibrate and provide offset for the accelerometer and the gyroscope -> check `accel_gyro_calibration.rs` example code or the function `calibration()`. For the magnetometer, you might need to compute hard and soft iron correction. If you want to learn more about those correction you can go here. We won't provide any tool for those corrections. However, you can follow this tutorial provided by Adafruit.

More information

This crate allows for a full customization except for Fifo and hardware interruptions implementation. This is not plan in the near future.

To use this driver you must provide a concrete `embedded_hal` implementation both for Delay and I2C.

You must be able to compute a delta time. e.g. the execution time of each loop. Most examples uses a Raspberry Pi for practicals reasons. NB: The more accurate the delta time, the more accurate the measurement.

A better ROS2 example might be provided in the future.

Usage

The example below uses rppal crates and runs on a Raspberry Pi.

```
use rppal::hal::Delay;
use rppal::i2c::I2c;

use embedded_hal::blocking::delay::*;
use nalgebra::{Vector3, Matrix3};

use adafruit::*;

fn main() -> -> Result<(), SensorError<rppal::i2c::Error>> {

    // Init a delay used in certain functions and between each loop.
    let mut delay = Delay::new();

    // Setup the raspberry's I2C interface to create the sensor.
    let i2c = I2c::new().unwrap();
```

```
    // Create an Adafruit object.
    let mut sensor = AdafruitNXP::new(0x8700A, 0x8700B, 0x0021002C, i2c);

    // Check if the sensor is ready to go.
    let ready = sensor.begin()?;
    if !ready {
        std::eprintln!("Sensor not detected, check your wiring!");
        std::process::exit(1);
    }

    // If you don't want to use the default configuration.
    sensor.set_accel_range(config::AccelMagRange::Range2g)?;
    sensor.set_gyro_range(config::GyroRange::Range500dps)?;
    sensor.set_accelmag_output_data_rate(config::AccelMagODR::ODR200HZ)?;
    sensor.set_gyro_output_data_rate(config::GyroODR::ODR200HZ)?;

    // If you need more precise results you might consider adding offsets. (Values are
exemples)
    sensor.accel_sensor.set_offset(-0.0526, -0.5145, 0.1439);
    sensor.gyro_sensor.set_offset(0.0046, 0.0014, 0.0003);
    let hard_iron = Vector3::new(-3.87, -35.45, -50.32);
    let soft_iron = Matrix3::new(1.002, -0.012, 0.002,
                                 -0.012, 0.974, -0.010,
                                 0.002, -0.010, 1.025);
    sensor.mag_sensor.set_hard_soft_iron(hard_iron, soft_iron);

    loop {

        sensor.read_data();

        let acc_x = sensor.accel_sensor.get_scaled_x();
        let acc_y = sensor.accel_sensor.get_scaled_y();
        let acc_z = sensor.accel_sensor.get_scaled_z();

        let gyro_x = sensor.gyro_sensor.get_scaled_x();
        let gyro_y = sensor.gyro_sensor.get_scaled_y();
        let gyro_z = sensor.gyro_sensor.get_scaled_z();

        let mag_x = sensor.mag_sensor.get_scaled_x();
        let mag_y = sensor.mag_sensor.get_scaled_y();
        let mag_z = sensor.mag_sensor.get_scaled_z();

        // Print data
        std::println!("##############################################");
        std::println!("Accel X: {}", acc_x);
        std::print!(" Accel Y: {}", acc_y);
        std::print!(" Accel Z: {}", acc_z);
        std::println!("----------------------------------------------");
        std::println!("Gyro X: {}", gyro_x);
        std::print!(" Gyro Y: {}", gyro_y);
        std::print!(" Gyro Z: {}", gyro_z);

        std::println!("----------------------------------------------");
        std::print!("Mag X: {}", mag_x);
        std::print!(" Mag Y: {}", mag_y);
        std::print!(" Mag Z: {}", mag_z);

        delay.delay_ms(5_u8);
    }
}
```

7.2 Librairie fusion de données

Datafusion library for 6 and 9 degrees of freedom sensors.

You have the choice in this library to apply the filters on two types of sensors:

- A 6 degrees of freedom sensor with a 3-axis accelerometer and 3-axis gyroscope -> Mode `Dof6`
- A 9 degrees of freedom sensor with a 3-axis accelerometer, 3-axis gyroscope and 3-axis magnetometer -> Mode `Dof9`

> ***For now, the sensor needs to be perfectly flat to work. Also, using angle and distance measurement at the same time is not recommended with a 6Dof sensor.***
> *More specifically, for the distance it doesn't really matter as the high pass filter will automatically delete the angle offset after a few second.*
> *However, for the angle (except with a magnetometer) the sensor has to be flat or at least stay within the same inclination on the X and Y-axis after initialization.*

In the example provided below, we use our own driver for an Adafruit sensor. So of course, you can use any other driver you want.

Angle data

When the sensor is flat and whatever the mode used, X and Y angles are absolute roll and pitch values in degrees. A Kalman filter is applied between acceleration and angular speed to produce a reliable output. While using a Dof6 sensor, e.g without a magnetometer, the Kalman filter output on the the Z or Yaw-axis is an angular speed in degrees per second. Which is then integrated to produce the absolute angle in degrees. But this method is not really reliable as it heavily rely on which speed the sensor is moving. However, when using a Dof9 sensor, the Kalman filter output on the Z or Yaw-axis is an absolute angle in degrees. This is the preferred method. As long as there is no magnetic interference, you we will be able a obtain a heading angle down to a 2 to 3 degree accuracy. In the case of magnetic interferences, if it is much greater than the earth magnetic field, then the output won't be the magnetic north but a relative angle to the magnetic interference.

Distance data

For the distance data, the algorithm is the same whatever the mode used as it only uses acceleration data on X and Y. It is a succession of high pass filters, low pass filters and integrations. It works perfectly fine on a short distance, e.g < 5cm, and is impressively accurate in this range. However, over this range the result highly rely on the speed. This means that, if the speed is too low, the distance will be under estimated. Same problem if the speed is too high, the distance will be over estimated. This problem is currently being worked on and the lib will be updated if a workaround is found.

You have the ability to disable the distance measurement by calling the `disable_distance` function.

Please note that distances measurements are purely experimental but it's a place to start as the accuracy is not optimal.

Usage

The example below uses rppal crates and runs on a Raspberry Pi.

Please note that this example and library have been developed in parallel with a driver for a Dof9 sensor from Adafruit.

```
use std::time::Instant;

use rppal::hal::Delay;
use rppal::i2c::I2c;

use embedded_hal::blocking::delay::*;

use adafruit::*;
use datafusion::{self as _, Fusion};

fn main() -> Result<(), SensorError<rppal::i2c::Error>> {

    // Init a delay used in certain functions and between each loop.
    let mut delay = Delay::new();

    // Setup the raspberry's I2C interface to create the sensor.
    let i2c = I2c::new().unwrap();

    // Create an Adafruit object
    let mut sensor = AdafruitNXP::new(0x8700A, 0x8700B, 0x0021002C, i2c);

    // Check if the sensor is ready to go
    let ready = sensor.begin()?;
    if !ready {
        std::eprintln!("Sensor not detected, check your wiring!");
        std::process::exit(1);
    }

    sensor.set_accel_range(config::AccelMagRange::Range8g)?;
    sensor.set_gyro_range(config::GyroRange::Range500dps)?;
    // etc...

    // Initialize the sensor
    sensor.read_data()?;

    let acc_x = sensor.accel_sensor.get_scaled_x();
    let acc_y = sensor.accel_sensor.get_scaled_y();
    let acc_z = sensor.accel_sensor.get_scaled_z();

    let gyro_x = sensor.gyro_sensor.get_scaled_x();
    let gyro_y = sensor.gyro_sensor.get_scaled_y();
    let gyro_z = sensor.gyro_sensor.get_scaled_z();
```

```
    let mag_rx = sensor.mag_sensor.get_scaled_x();
    let mag_ry = sensor.mag_sensor.get_scaled_y();
    let mag_rz = sensor.mag_sensor.get_scaled_z();

    // Create a datafusion object
    let mut fusion = Fusion::new(0.05, 20., 50);
    fusion.set_mode(datafusion::Mode::Dof9);

    // Set data to the fusion object
    fusion.set_data_dof9(acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z, mag_rx, mag_ry,
mag_rz);

    // Initialize the datafusion object
    fusion.init();

    // Set magnetic declination --> 1.39951° in Toulouse, France
    fusion.set_declination(1.39951);

    // Setting up the delta time
    let mut time = Instant::now();

    loop {

        // Calculate delta time in seconds
        let dt = time.elapsed().as_micros() as f32 / 1_000_000.;
        time = Instant::now();

        // Update old values for the next loop
        fusion.set_old_values(acc_x, acc_y);

        sensor.read_data()?;

        let acc_x = sensor.accel_sensor.get_scaled_x();
        let acc_y = sensor.accel_sensor.get_scaled_y();
        let acc_z = sensor.accel_sensor.get_scaled_z();

        let gyro_x = sensor.gyro_sensor.get_scaled_x();
        let gyro_y = sensor.gyro_sensor.get_scaled_y();
        let gyro_z = sensor.gyro_sensor.get_scaled_z();

        let mag_rx = sensor.mag_sensor.get_scaled_x();
        let mag_ry = sensor.mag_sensor.get_scaled_y();
        let mag_rz = sensor.mag_sensor.get_scaled_z();

        // Set data to the fusion object
        fusion.set_data_dof9(acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z, mag_rx,
mag_ry, mag_rz);

        // Perform a step of the algorithm
        fusion.step(dt);

        // Collect outputs
```

```
        let angle_x = fusion.get_x_angle();
        let angle_y = fusion.get_y_angle();
        let angle_z = fusion.get_heading();
        let distance = fusion.get_final_distance();

        // Print data
        std::println!("Angle X: {} °", angle_x);
        std::println!("Angle Y: {} °", angle_y);
        std::println!("Angle Z: {} °", angle_z);
        std::println!("Total distance traveled: {} cm", distance);

        delay.delay_ms(5_u8);
    }
}
```

7.3 Présentation de l'entreprise

Toutes les informations de cette partie sont extraites du site internet du groupe SII et de l'intranet de l'entreprise.

7.3.1 Historique du groupe

7.3.1.1 Les débuts de l'histoire SII

Bernard Huvé, ingénieur de formation, se lance pour créer SII (Société pour l'Informatique Industrielle).

Spécialisé en informatique industrielle, Bernard recrute rapidement ses premiers collaborateurs pour honorer des contrats forfaitaires pour de grands groupes comme Philips, le Commissariat à l'Énergie Atomique ou encore Bull.

7.3.1.2 1980 - Premiers déploiements en France

Cette décennie est caractérisée par la création de la première agence SII à Nice (1984) à l'occasion d'un important contrat avec le laboratoire d'IBM ; puis le déploiement du modèle sur l'Île-de-France avec Cergy Pontoise (1987) pour le client Sagem et enfin le site de Vélizy (1989) pour Matra, Renault et Sextant.

L'effectif de la société dépasse alors les 100 personnes.

7.3.1.3 1990 - Structuration et développement soutenu sur le territoire

Les années 90 sont marquées par une démarche qualité volontariste autour de la formalisation des procédures, méthodes, et best practices. En 1992, SII obtient la certification ISO 9001 sur l'ensemble de ses activités.

En parallèle, la stratégie de développement se poursuit sur toute la France avec la création de multiples agences (Rennes, Aix-en-Provence, Nantes, Toulouse, Strasbourg puis Lille).

Enfin, en 1999, le groupe s'introduit en bourse pour accroître la notoriété de SII. Bernard Huvé, fondateur et Président Directeur Général, en est avec sa famille l'actionnaire majoritaire.

7.3.1.4 2000 - Cap sur l'international et évolution de la Gouvernance

Le Groupe poursuit son maillage français, et engage son développement international avec l'ouverture d'une filiale en Pologne (2006) suivi par la République Tchèque, la Belgique, le Maroc, ainsi que l'acquisition du groupe Coris en Suisse et de Concatel en Espagne.

Pour accompagner la croissance du Groupe qui dépasse alors les 1500 personnes, une nouvelle équipe de direction est constituée en 2007. Bernard Huvé prend la tête du conseil de surveillance et confie la présidence du Directoire à Éric Matteucci.

7.3.1.5 2010 - Acquisitions et poursuite du développement à l'international

La stratégie d'expansion au-delà de la France se confirme et s'accélère avec des acquisitions en Allemagne, Belgique et Colombie, la création de nouvelles implantations aux Pays-Bas, en Inde ainsi qu'au Canada et en Angleterre.

Le Groupe réalise également des opérations de croissance externe en France pour soutenir son développement avec, notamment, le rachat de Feel Europe en 2016 visant à renforcer le positionnement dans le domaine bancaire.

En 2019, SII souffle sa 40ème bougie et annonce le renforcement du « top management » du Groupe et la création d'une Direction du Développement, pour favoriser l'accélération du développement de la société.

7.3.1.6 2020 - Un futur ambitieux

Capitalisant sur une solidité acquise en plus de 40 ans (via une présence internationale dans 18 pays, et un chiffre d'affaires annuel supérieur à 676M€), et porté par un contexte de transformation profond des entreprises, le Groupe s'inscrit dans une logique de développement ambitieuse pour les prochaines années. Pour y parvenir, ils basent leur stratégie sur 4 piliers :

- Poursuivre l'accompagnement des clients sur les plans technologique et géographique.
- Investir dans la méthodologie et l'innovation pour créer de l'expertise.
- Amplifier les actions et initiatives conduisant à l'épanouissement professionnel des salariés.
- Inscrire durablement SII dans son environnement sociétal par une politique sociale responsable.

7.3.2 Le Groupe SII

Figure 41 : Le Groupe SII en chiffres

Sous la responsabilité d'Eric MATTEUCCI en France, le groupe SII s'organise en agences pour répondre à un besoin de « proximité ».

9 agences existent en France à Aix, Lille, Lyon, Nantes, Paris, Rennes, Strasbourg, Sophia, Toulouse. Le siège de SII est situé 8 rue des pirogues de Bercy, Paris 12e.

Figure 42 : Agences SII en France

L'agence constitue l'unité de base de l'organisation du groupe, elle offre une meilleure écoute et un service de proximité aux clients locaux. C'est le « Local Professional Services ». En charge de sa démarche commerciale et de sa gestion des compétences sur sa zone géographique, elle procure les avantages d'une société à taille humaine : simple, lisible et motivante pour ses collaborateurs.

Cette organisation décentralisée est source de transparence et d'efficacité. Elle permet :

- Souplesse et réactivité
- Responsabilisation des équipes
- Prise en compte des réalités locales

7.3.3 Culture d'entreprise et valeurs

La culture d'entreprise du Groupe SII se caractérise essentiellement par l'engagement, la responsabilisation, la transparence, la confiance, la qualité et le professionnalisme.

7.3.3.1 Engagement et responsabilisation

Chaque collaborateur dispose des moyens et des responsabilités de sa fonction. Les remontées d'information et les suggestions commerciales ou opérationnelles sont valorisées et encouragées. Cette culture de l'engagement sous-entend l'acceptation du droit à l'essai, du droit à l'erreur.

7.3.3.2 Transparence

Les informations concernant le fonctionnement de la société sont transmises aux salariés, aux actionnaires et aux clients de manière sincère, rapide, exhaustive et compréhensible. Les résultats, les tendances, les objectifs, le fonctionnement des agences ne sont pas considérés comme des « secrets industriels ».

7.3.3.3 Confiance

Chez SII, la confiance est donnée a priori, c'est le socle relationnel interne. Cette manière de fonctionner à une contrepartie ; l'exigence permanente de qualité et d'adhésion du collaborateur au projet et à la culture de l'entreprise.

7.3.3.4 Qualité et professionnalisme

Ce sont les valeurs historiques de la société créée par des ingénieurs issus du monde technique. Si la société s'est enrichie au fil du temps d'une dimension commerciale, ces valeurs originelles restent prépondérantes.

D'autres valeurs font partie intégrante de la culture d'entreprise de la société : l'humilité, la prudence, la délégation, le respect de l'autre, l'éthique, …

Ces valeurs sont affichées dans les bureaux, présentées en entretiens de recrutement lors de l'intégration et sont rappelées lors de certaines réunions d'agences.

BIBLIOGRAPHIE

[1] « GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module ». [Online]. Disponible sur: https://www.hotmcu.com/gy521-mpu6050-3axis-acceleration-gyroscope-6dof-module-p-83.html. [Consulté le: 28 juillet 2022]

[2] « MPU-6050 - Module MEMS, Série MotionTracking, Gyroscope/Accéléromètre 3 axes, ±16g, 2,375 V à 3,46 V, QFN-24 ». [Online]. Disponible sur: https://fr.farnell.com/invensense/mpu-6050/gyroscope-accelero-6-axes-i2c/dp/1864742. [Consulté le: 1 mars 2022]

[3] A. Industries, « Adafruit Precision NXP 9-DOF Breakout Board ». [Online]. Disponible sur: https://www.adafruit.com/product/3463. [Consulté le: 28 juillet 2022]

[4] « ESP32 NodeMCU Module WLAN WiFi Development Board avec CP2102 (modèle successeur de ESP8266) compatible avec Arduino », *AZ-Delivery*. [Online]. Disponible sur: https://www.az-delivery.de/fr/products/esp32-developmentboard. [Consulté le: 28 juillet 2022]

[5] « STM32F401RE - STM32 Dynamic Efficiency MCU, Arm Cortex-M4 core with DSP and FPU, up to 512 Kbytes of Flash memory, 84 MHz CPU, Art Accelerator - STMicroelectronics ». [Online]. Disponible sur: https://www.st.com/en/microcontrollers-microprocessors/stm32f401re.html. [Consulté le: 29 juillet 2022]

[6] R. P. Ltd, « Buy a Raspberry Pi 4 Model B », *Raspberry Pi*. [Online]. Disponible sur: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. [Consulté le: 29 juillet 2022]

[7] « Accéléromètre | Capteur d'accélération ». [Online]. Disponible sur: https://www.sensel-measurement.fr/fr/14-accelerometre. [Consulté le: 28 février 2022]

[8] R. Tillaart, « GY521 ». 18 décembre 2021 [Online]. Disponible sur: https://github.com/RobTillaart/GY521. [Consulté le: 1 mars 2022]

[9] « Attitude ». [Online]. Disponible sur: https://novatel.com/solutions/attitude. [Consulté le: 7 mars 2022]

[10] « Le filtre de Kalman : intérêts et limites » Sciences et Techniques ». [Online]. Disponible sur: http://www.ferdinandpiette.com/blog/2011/04/le-filtre-de-kalman-interets-et-limites/. [Consulté le: 7 mars 2022]

[11] « Définition - Moyenne mobile | Insee ». [Online]. Disponible sur: https://www.insee.fr/fr/metadonnees/definition/c2091. [Consulté le: 7 mars 2022]

[12] « Filtrage numérique d'un signal : application à Arduino », *Splac*, 31 août 2020. [Online]. Disponible sur: https://splac.fr/filtrage-numerique-dun-signal-application-a-arduino/. [Consulté le: 7 mars 2022]

[13] « Compensating for Tilt, Hard-Iron, and Soft-Iron Effects | Fierce Electronics ». [Online]. Disponible sur: https://www.fierceelectronics.com/components/compensating-for-tilt-hard-iron-and-soft-iron-effects. [Consulté le: 1 août 2022]

[14] « Architecture ROS - SII Team », *Google Docs*. [Online]. Disponible sur: https://drive.google.com/file/d/1rdr71fel9zrHVMBo07xaHMUJ6nhjiN6i/view?usp=sharing&usp=embed_facebook. [Consulté le: 1 août 2022]

[15] F. Leon, « DataFusion ». 1 août 2022 [Online]. Disponible sur: https://github.com/florianleon/StageSII. [Consulté le: 1 août 2022]

[16] F. Leon, « datafusion_imu - crates.io: Rust Package Registry ». [Online]. Disponible sur: https://crates.io/crates/datafusion_imu. [Consulté le: 1 août 2022]

[17] F. Leon, « adafruit_nxp - crates.io: Rust Package Registry ». [Online]. Disponible sur: https://crates.io/crates/adafruit_nxp. [Consulté le: 1 août 2022]

LISTE DES ILLUSTRATIONS