N° 22

ECOLE POLYTECHNIQUE UNIVERSITAIRE DE MONTPELLIER

Projet industriel de fin d'études
Département Electronique, Robotique, Informatique Industrielle

ANNEE 2021-2022

PROJET

Ayoub LACHEHAB

Fusion de données : la calibration des données brutes envoyées par une unité de mesure inertielle pour les robots sous-marins

ECOLE POLYTECHNIQUE UNIVERSITAIRE DE MONTPELLIER
UNIVERSITE MONTPELLIER II SCIENCES ET TECHNIQUES DU LANGUEDOC
Place Eugène Bataillon 34095 MONTPELLIER CEDEX 5
Tél. : 04 67 14 31 60 – Fax : 04 67 14 45 14
E-mail : scola@polytech.univ-montp2.fr

POLYTECH MONTPELLIER

Table des matières

Remerciements

Avant de commencer la rédaction de ce rapport, je tiens tout d'abord à remercier mes tuteurs de projet M. René Zapata et M. Éric Dubreuil qui m'ont accompagné tout au long de ce projet.

Je tiens également à remercier mon tuteur industriel M. Grégory Reuillard pour toute l'aide qu'il a pu me prodiguer au cours de ces derniers mois.

A cette occasion, la promotion MEA2022 souhaite à M. René Zapata une bonne continuation pour les projets à venir.

I. Introduction

Dans le cadre de notre formation d'ingénieur proposée par l'école Polytech Montpellier, nous sommes amenés à travailler en relation avec des entreprises via le projet de fin d'études. Ce dernier a pour but l'application des compétences théoriques acquises durant le cursus MEA, et de les mettre en pratique sous la forme d'un projet long terme. Ce travail comprend plusieurs aspects essentiels comme l'étude de faisabilité, la gestion de projet, la conception et la réalisation des prototypes.

Le projet que j'ai choisi est proposé par l'entreprise ECA ROBOTICS, une entreprise basée à Montpellier qui fait partie de la filiale française d'ECA GROUP, spécialisée dans plusieurs domaines comme la robotique (terrestre et maritime), l'aérospatial, la défense. Cette entreprise innovante développe des équipements technologiques qui répondent aux besoins des clients dans plus de 80 pays dans le monde.

Le travail réalisé est orienté dans le domaine du développement logiciel embarqué. Son but est d'étudier les données brutes envoyées par une unité de mesure inertielle utilisée dans les robots sous-marins créés.

Figure 1 Les robots sous-marins de chez ECA ROBOTICS

Les robots sous-marins embarquent différents capteurs inertiels qui permettent d'effectuer des mesures et du contrôle en mer. L'enjeu majeur est d'effectuer ces mesures de manière non intrusive, sans bruit et sans consommer d'énergie. C'est de ce postulat que j'ai démarré mes travaux de recherches. J'ai dans un premier temps étudié les données acquises par les capteurs.

Ces derniers ont connu une croissance très rapide ces dernières années notamment dans le domaine de la navigation maritime. Ils sont appelés MEMS (Micro Electro-Mechanical System). Ils utilisent des systèmes mécaniques sur des puces en silicium, technologie présente dans le domaine de la fabrication des microprocesseurs.

La combinaison de plusieurs capteurs inertiels sur une même électronique permet la fusion de leurs données. Cette dernière est basée sur l'utilisation d'un capteur pour améliorer ou enrichir les mesures de son homologue. C'est avec ce principe que se basent les différents capteurs utilisés dans ce projet. Ces derniers sont une combinaison de capteurs inertiels et magnétiques qui permettent de faire une analyse de l'objet sur lequel est placé ce dernier.

Les capteurs utilisés dans la carte électronique fournie par ECA ROBOTICS sont

- Gyromètre, qui permet de faire une mesure de la vitesse de rotation
- Magnétomètre, qui détermine la direction du champ magnétique terrestre et en conséquence de faire un repérage du nord.
- Accéléromètre qui permet de faire une mesure de l'accélération linéaire.

Durant le projet, j'ai défini et choisi une méthode de calibration pour rapprocher le plus possible les mesures acquises des données réelles, qui sera expliquée ultérieurement.

II. Lancement du projet

Tout d'abord, la première séance de projet s'est déroulée avec mes tuteurs scolaires et mon tuteur industriel. Nous avons essayé ensemble de trouver un moyen efficace qui permet de déterminer le code de calibration qui recevra les données brutes des capteurs. Ensuite, nous devions trouver une méthode pour effectuer cette calibration. Nous avons conclu, à l'issue de cette réunion, qu'il faut mettre en place un prototype de vérification permettant d'évaluer toutes les mesures effectuées par ces capteurs après la phase de calibration.

Après la réunion, j'ai pu mettre en place avec mes tuteurs l'ensemble des tâches à réaliser pour le projet.

1. Mettre place une méthode de calibration pour rendre toutes les données brutes envoyées par la carte de traitement plus adaptées à l'environnement de mesure.
2. Créer un prototype de vérification qui permet l'évaluation des mesures effectuées par les capteurs inertiels.
3. Suivre une démarche scientifique pour valider la fiabilité de ces données calibrées sur le long terme.

II.1 La démarche imaginée

II.1.1 Répartition des taches

Premièrement, j'ai commencé par étudier les différentes méthodes qui peuvent être utilisées pour calibrer des données inertielles. En effet, il existe plusieurs manières de procéder selon les différentes applications. Dans mon cas, les capteurs intégrés dans cette carte de traitement sont utilisés pour acquérir des données pendant de longues périodes en mer. C'est dans cette optique que j'ai décidé de ne pas utiliser la méthode d'intégration des données. En effet, comme les acquisitions sont longues, l'intégration génère du bruit, non bénéfique aux mesures (position à partir des données reçues via l'accéléromètre ou les angles Euler en utilisant les vitesses angulaires reçues par le gyroscope).

La solution qui a été retenue est d'appliquer un processus qui sert à combiner les données acquises par le gyromètre, l'accéléromètre et le magnétomètre. Par la suite, j'utilise la méthode de Madgwick pour le calcul du vecteur complexe de quaternions.

La deuxième phase de ce procédé est de vérifier les données en sortie de l'étalonnage. L'idée était de concevoir un outil qui corrige en temps réel, ce qui permet de regarder l'impact de chaque paramètre, ajustables dans la mesure.

De plus, la fixation des capteurs sur des moteurs pas à pas engendre un élément de solution supplémentaire. Ces derniers sont commandés en parallèles pendant le processus d'acquisition des données. Cette solution nous donne la possibilité d'envoyer différents types de commandes sur les moteurs pour mesurer la marge d'erreur. Également, ce procédé permet la validation des mesures effectuées par les capteurs.

Après avoir pris en compte les différentes remarques et les décisions de mes tuteurs, j'ai pu répartir les tâches à accomplir dans cet ordre.

- Calibrer les données acquises par le capteur du gyromètre et du magnétomètre.
- Appliquer l'algorithme de Madgwick pour calculer le quaternion complexe.
- Mettre en œuvre les résultats dans un outil graphique.
- Concevoir le prototype de vérification et commander des moteurs pas à pas.
- Mettre en place les capteurs inertiels sur le prototype.
- Evaluer les données calibrées à l'aide du prototype

II.1.2 Organisation temporelle

Afin de pouvoir suivre l'avancement du projet et finir dans le temps imparti, j'ai utilisé comme outil le diagramme de GANTT. Ce dernier permet d'agencer la répartition des tâches dans le temps et de vérifier leur durée.

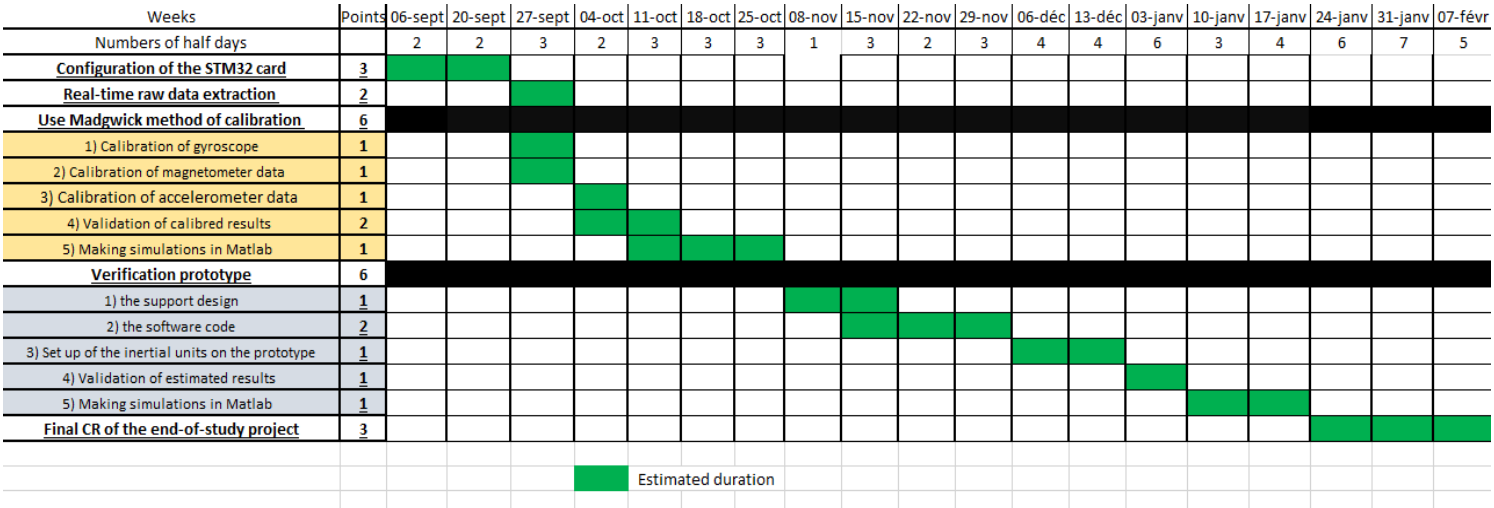


Figure 2 Le diagramme de Gantt

J'ai débuté par la configuration de la carte STM32 et tous ses périphériques essentiels comme le SPI, les timers, l'USART et la clock par exemple. Ensuite je me suis consacré à la partie d'extraction des données en temps réel, afin de pouvoir appliquer par la suite l'algorithme AHRS Madgwick. Les données en sortie de cet algorithme seront validées par le prototype.

On peut constater que ce projet s'appuie sur plusieurs compétences basées sur l'utilisation des logiciels de développement embarqués, la conception 3D des pièces ainsi la capacité à établir une démarche scientifique pour vérifier la pertinence des résultats trouvés.

III. Le travail réalisé

Tout d'abord, avant de rentrer plus dans les détails, tout le travail a été réalisé sous la plateforme de développement STM32CubeIDE. Cet IDE comporte des fonctionnalités de configuration périphérique, de génération et de compilation de code ainsi que le débogage pour les microcontrôleurs STM32.

Comme indiqué la figure ci-dessous, la carte STM32F412ZG est l'élément central du projet. J'ai réussi a réalisé l'extraction des données acquises par la centrale inertielle 9 DOF Grove via la liaison SPI. De plus, j'ai utilisé la communication USART pour récupérer les trames envoyées par la carte électronique fournie par ECA ROBOTICS. Ensuite, j'ai envoyé les commandes via la liaison SPI aux deux moteurs pas à pas dans le but d'analyser les mouvements réalisés par les centrales inertielles fixées aux arbres moteurs.

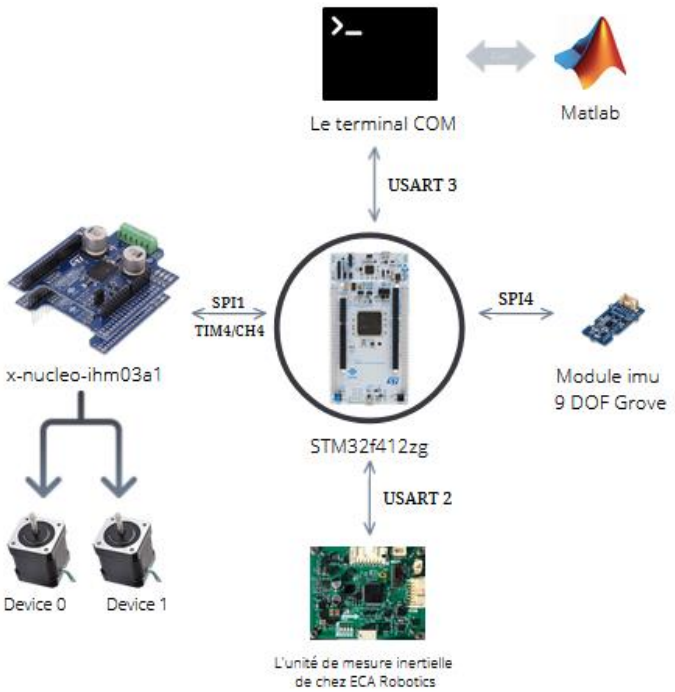


Figure 3 : Le schéma global de projet

J'ai ensuite utilisé Matlab pour tester le code réalisé avant de l'implémenter sur le STM32CubeIDE. De plus, Matlab possède l'avantage d'avoir un affichage graphique accessible et de bonne facture. J'ai donc pu faire des simulations graphiques du comportement des capteurs inertiels en temps réel lors la phase d'acquisition des données.

III.1 Mise en contexte

La carte électronique de mesure inertielle fournie par l'entreprise ECA Robotics permet d'acquérir des données brutes à partir des capteurs suivants :

- Le capteur magnétomètre : LIS2MDL
- Le capteur accéléromètre et gyroscope : ISM330DLC

Figure 4 La carte électronique d'ECA ROBOTICS

La résolution des capteurs dépend de ce dernier. Ce paramètre permet de choisir la plage mesurée par le capteur. Une résolution plus grande implique une bonne précision de mesure, une meilleure finesse dans la mesure. Voici les différents choix possibles qu'on peut effectuer pour changer la résolution des capteurs utilisés dans le projet :

- Résolution accéléromètre : ±2/4/6/8/16 g paramétrable.
- Résolution angulaire du gyromètre : ±125/±250/±500/±1000/±2000 dps.
- Résolution magnétomètre : ±50 gauss.

En utilisant ces deux capteurs, nous pouvons alors déterminer l'orientation dans notre objet dans l'espace. En effet, Il existe deux modes d'utilisation pour ces unités inertielles qui nous permettent de faire une mesure spécifique sur choix.

Le premier mode d'utilisation, appelé « IMU », se base uniquement sur les données du gyroscope et l'accéléromètre. Ces dernières sont acquises par le capteur ISM33DLC. Ce mode de mesure ne donne pas résultats précis des composantes de tangage, de roulis mais également de la mesure absolue du cap. En effet, le mode IMU ne prend pas en compte les distorsions et les interférences magnétiques.

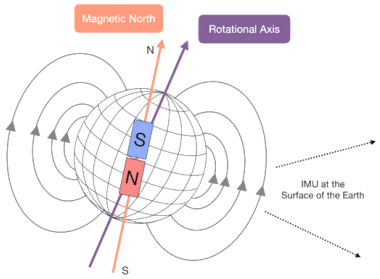


Figure 5 Le champ magnétique terrestre

Le deuxième mode de mesure, appelé « AHRS, Attitude and Heading Reference System », est basé sur l'utilisation des données du gyromètre, de l'accéléromètre et du magnétomètre. Ce mode permet de déterminer une latitude et un cap fiables par rapport au nord magnétique.

Dans mon cas, je me suis intéressé au deuxième mode de fonctionnement, grâce à ses avantages et sa précision vis-à-vis du repère absolu et l'orientation de l'objet dans ce dernier. Avant d'appliquer ce mode, j'ai d'abord commencé par la calibration des données brutes du gyromètre, de l'accéléromètre et du magnétomètre.

Ce processus de calibration est utilisé pour déterminer les différents jeux de paramètres qui permettront d'ajuster les valeurs acquises pour correspondre à une valeur connue ou à une valeur de référence. On prend par exemple le cas d'un capteur inertiel en mode statique, immobile, et l'axe z est dirigé vers le haut. Les valeurs obtenues par le gyroscope doivent être égales à zéro dans les trois axes (pas de rotations car immobile). De plus, on doit obtenir une valeur de 1g pour l'axe z et 0 pour les axes x et y de l'accéléromètre. Ces différents paramètres sont très utiles pour corriger les données acquises par les deux capteurs.

Par rapport aux données du magnétomètre, on peut également s'appuyer sur la référence du champ magnétique terrestre pour évaluer la qualité de la mesure du capteur. Ces valeurs de référence sont connues, elles dépendent de l'orientation et de l'amplitude du champ magnétique terrestre à l'endroit où se trouve le magnétomètre.

Par la suite, je vais m'intéresser à la calibration du gyroscope, de l'accéléromètre et à celle du magnétomètre.

III.2 La calibration du gyroscope

Afin de bien calibrer les données brutes du gyroscope, j'ai positionné mon capteur en mode statique, immobile. Ensuite, j'ai fait une moyenne d'un ensemble de mesures sur les axes x, y et z. Cette moyenne doit être utilisée pour ajuster l'erreur de mesure. Pour cela il suffit juste de soustraire les moyennes calculées pour chaque axe aux mesures du gyromètre et donc d'avoir une valeur nulle pour les trois axes.

J'ai remarqué après plusieurs essais que les valeurs d'offset varient d'une journée à l'autre, c'est pour cela j'ai utilisé l'User Button de la STM32 pour refaire la calibration à chaque mesures du gyroscope.

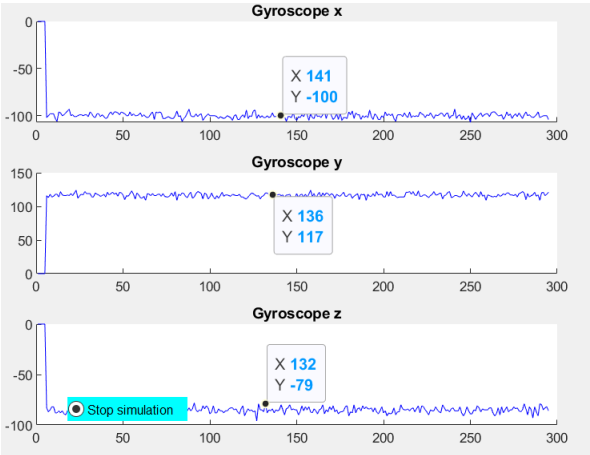


Figure 6 Les données gyromètre non calibrées

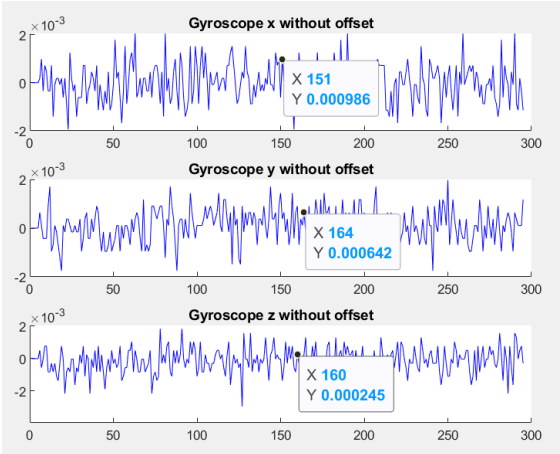


Figure 7 La mesure du gyromètre calibré en mode statique

Les données du gyroscope peuvent être calculées à partir des mesures d'accélérations angulaires autour de chaque axe en passant par une intégration. En théorie, il est possible d'estimer l'orientation dans le temps mais cela ne fonctionne pas forcément dans le cas pratique à cause l'accumulation des erreurs, phénomène appelé « drift ».

Ce phénomène est aussi présent lors du calcul des angles d'Euler à partir des vitesses angulaires. Les figures ci-dessous montrent le problème dû aux erreurs de mesure cumulées sur les trois axes et le choix de ne pas prendre l'intégration comme idée de démarrage.

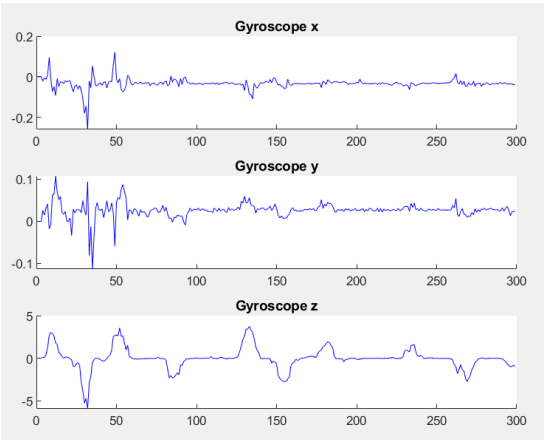


Figure 8 : Les données gyromètre

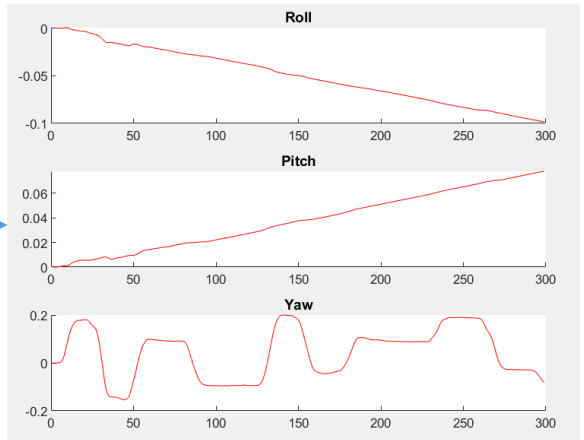


Figure 9 Les angles Euler par la méthode d'intégration

III.3 La calibration de l'accéléromètre

La calibration des données brutes d'accéléromètre sensiblement la même que pour le gyromètre. J'ai remarqué que le capteur affiche une valeur non nulle lorsqu'il est immobile, ce qui correspond à la projection du vecteur gravité sur l'axe z de l'accéléromètre. De ce fait, la méthode de calibration consiste à aligner le capteur dans l'axe de la gravité et enlever le décalage pour avoir une valeur finale de 1g dans l'axe z et la valeur de zéro dans les autres axes.

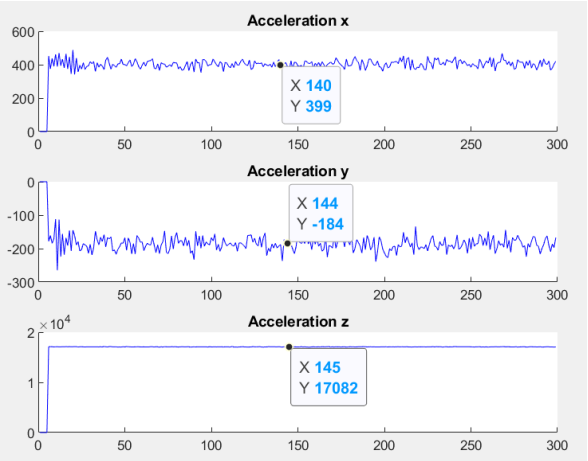


Figure 10 Les données d'accéléromètre non calibré

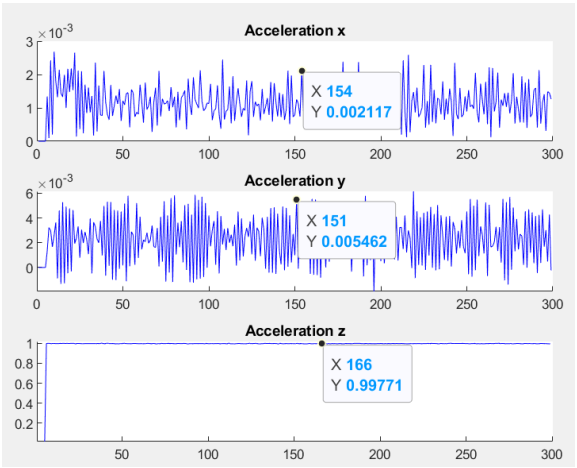


Figure 11 les données d'accéléromètre calibré en mode statique

III.4 La calibration du magnétomètre

La calibration du magnétomètre utilise les valeurs connues du champ magnétique terrestre. Ces valeurs comprennent les angles d'inclinaison et de déclinaison de ce dernier, ainsi que son amplitude. Elles dépendent de l'endroit où se trouve le capteur. L'algorithme de calibration du magnétomètre implémenté dans le logiciel est basé sur le calcul la valeur maximale et minimale d'un ensemble de mesures. En effet, selon ces deux valeurs, on peut soustraire les décalages créés sur les trois axes. Les erreurs de décalages sont dues aux phénomènes d'interférences magnétiques autour du capteur.

J'ai utilisé le toolbox **ellipsoid_fit** définie dans Matlab pour modéliser le champ magnétique mesuré sous la forme d'un ellipsoïde. Après calibration, ce dernier prend la forme d'une sphère. Autrement dit, toutes les valeurs mesurées par le magnétomètre restent centrées sur l'origine de référence liée aux trois axes.

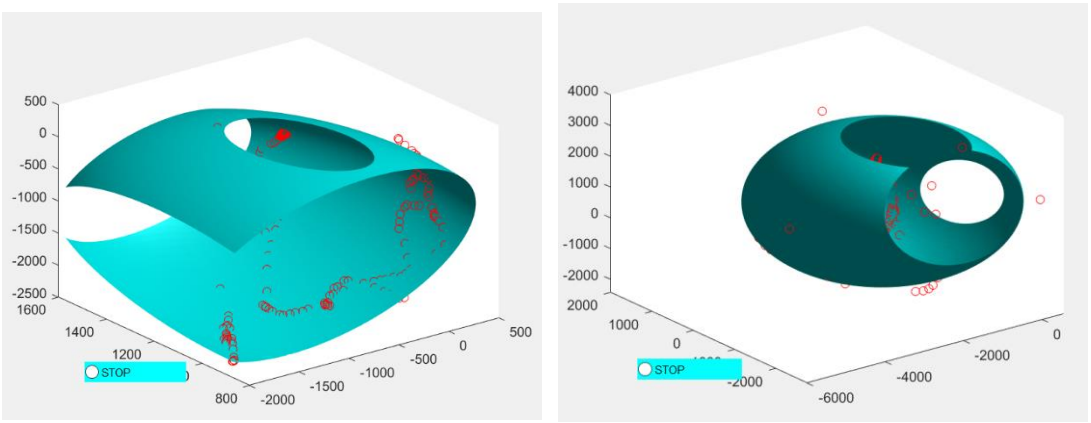


Figure 12 Les données magnétomètre modélisé en utilisant le toolbox ellipsoid_fit

Toutes les mesures et calibrations en rapport avec le magnétomètre ont été réalisées dans les locaux de Polytech Montpellier, au bâtiment 14. J'ai essayé de corriger les données brutes envoyées par le magnétomètre, en me basant sur les valeurs d'angle d'inclinaison et de déclinaison de ma position. Il faut savoir que ces valeurs varient en cas de changement position du capteur, il faut penser donc à refaire cette étape si besoin.

III.5 L'algorithme Madgwick AHRS

Après avoir calibré les données brutes des trois capteurs, Nous allons passer à l'étape suivante qui consiste à utiliser ces données pour calculer le vecteur complexe de quaternion en appliquant la méthode de Madgwick. Cette méthode consiste à calculer les quaternions qui s'écrivent sous la forme d'un nombre complexe à 4 composantes :

$$q = q_0 + q_1 * i + q_2 * j + q_3 * k = [q_0 \quad q_1 \quad q_2 \quad q_3]$$

q_0 , q_1, q_2, q_3 Réels et i, j, k coefficients imaginaires.

Les quaternions nous permettent de décrire la nature des orientations en trois dimensions, cette présentation est basée sur l'utilisation des données d'accéléromètre, de magnétomètre. De plus, l'algorithme de descente de gradient est présent pour calculer l'erreur de mesure du gyroscope en tant que la dérivée du quaternion.

J'ai utilisé la méthode AHRS mentionnée dans la partie précédente pour calculer les quaternions.

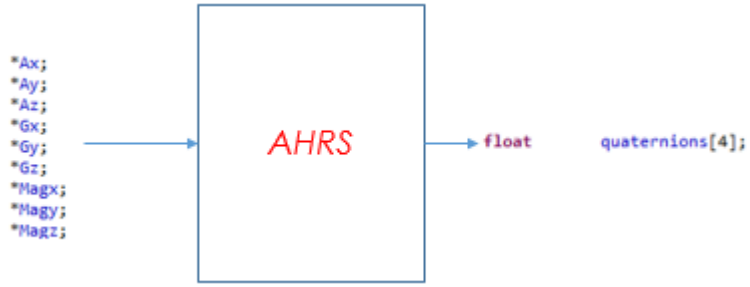


Figure 13 : Schéma bloc d'algorithme AHRS

Les figures ci-dessous visualisées sur le logiciel STMStudio, montrent les données entrées-sorties de l'algorithme AHRS. Sont disponibles dans l'annexe les explications et les différentes étapes du calcul des quaternions avec cette méthode.

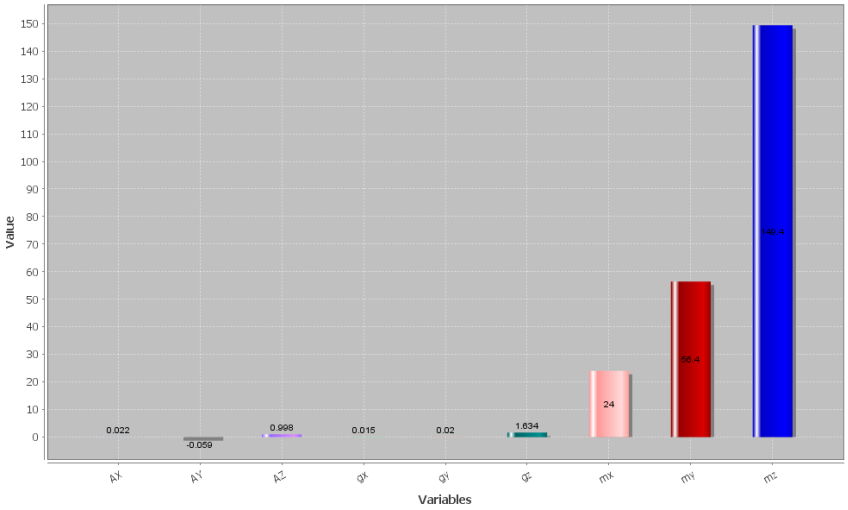


Figure 14 Les données entrées d'algorithme AHRS

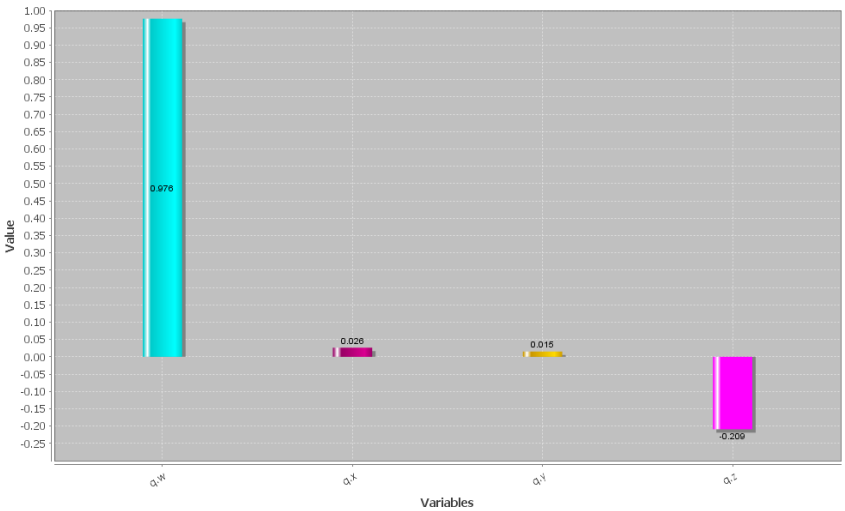


Figure 15 Les données sorties d'algorithme AHRS

Par la suite, j'ai modélisé sous la forme d'un parallélépipède les différentes variations du vecteur de quaternion pour représenter le capteur en temps réel. J'ai également montré le repère absolu lié au capteur et le repère mobile qui représente le changement du vecteur de quaternion.

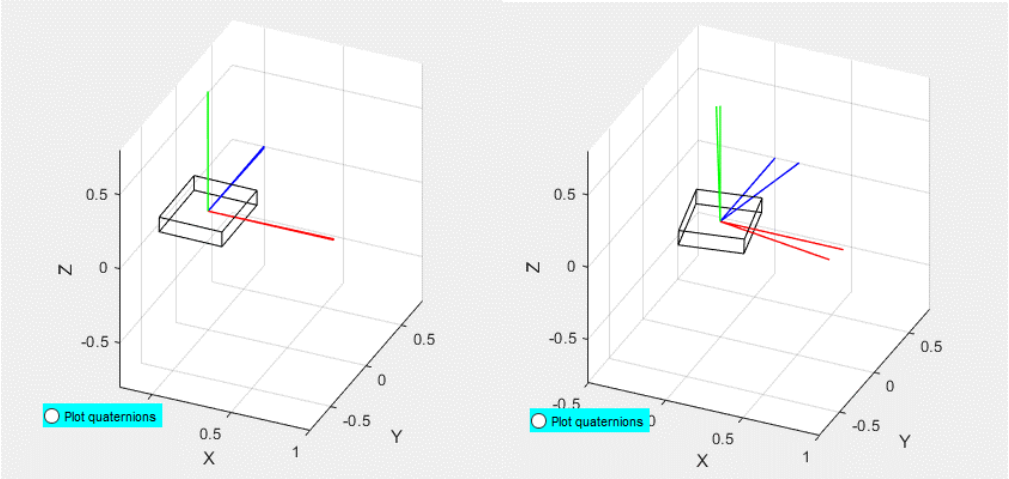


Figure 16 les orientations de capteur affichées sur Matlab

En utilisant le prototype de vérification, on peut donc effectuer des rotations précises et vérifier si le vecteur quaternion calculé par l'algorithme AHRS correspond effectivement au mouvement affiché sur Matlab.

III.6 Le prototype de vérification

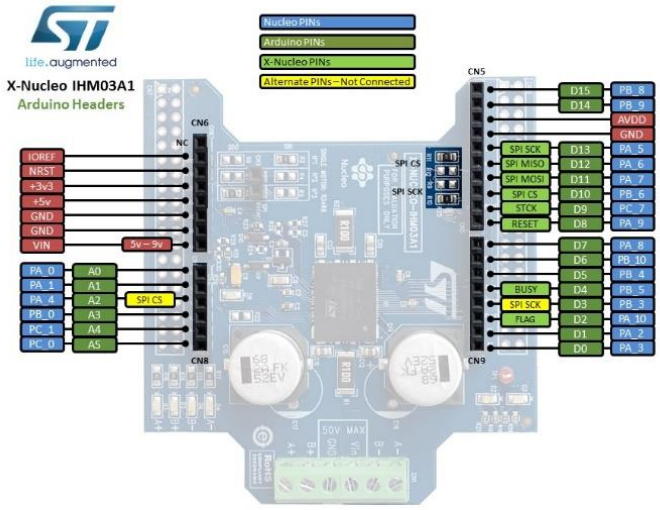


Figure 17 le driver moteur X-NUCLEO-IHM03A1

La figure ci-dessus montre le driver moteur de chez STMicroelectronics, cette carte est utilisée pour contrôler le moteur pas à pas. La communication avec le microprocesseur powerstep01 se fait par la liaison SPI comme indiqué précédemment. Dans le prototype, nous avions prévu de mettre d'implanter deux moteurs pour comparer les données acquises par les deux centrales inertielles fixées sur les arbres moteurs. Ce microprocesseur possède un avantage. En effet, ce dernier permet de gérer et modifier plusieurs paramètres comme le nombre de pas effectué par les moteurs, la vitesse, l'accélération, la décélération et bien d'autres encore. Ces paramètres sont importants pour la fiabilité du processus de vérification.

J'ai mis en place deux drivers moteur pour avoir le choix de contrôler d'une manière séparé ou simultané les deux moteurs pas à pas.

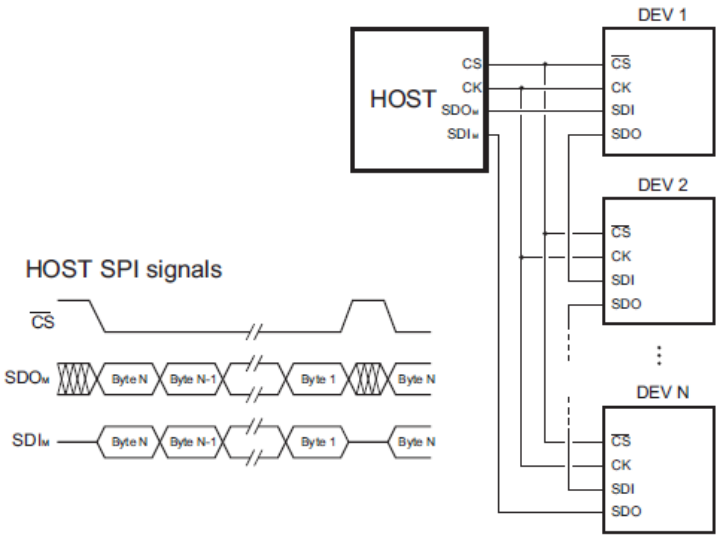


Figure 18 La communication SPI avec plusieurs modules

Cette carte permet de contrôler jusqu'à 3 moteurs, grâce à la communication SPI. Durant cette phase du projet, j'ai utilisé le registre Status du microprocesseur pour lire l'état actuel du driver moteur. Ce registre est la solution aux nombreux problèmes que j'ai rencontrés dans cette étape.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 15
STALL_A	STALL_B	OCD	TH_STATUS		UVLO_ADC	UVLO	STCK_MOD
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMD_ERROR	MOT_STATUS		DIR	SW_EVN	SW_F	BUSY	HiZ

Figure 19 Le registre Status de Powerstep01

La trame envoyée par le driver contient plusieurs éléments essentiels :

- Lorsque le bit OCD est à 0, il indique un événement de détection de surintensité.
- Lorsque le bit UVLO_ADC est à 0, il indique un événement de sous-tension ADC.
- Lorsque le bit UVLO est à 0, il définit un verrouillage de sous-tension ou des événements de réinitialisation.
- Le bit BUSY est à 0 lorsque la vitesse du moteur est constante.
- Les bits MOT_STATUS indiquent l'état de moteur pas à pas.

MOT_STATUS		Motor status
0	0	Stopped
0	1	Acceleration
1	0	Deceleration
1	1	Constant speed

- Le bit DIR indique le sens de rotation de moteur s'il y a une commande envoyée.

DIR	Motor direction
1	Forward
0	Reverse

Au vu du contexte du projet, j'ai mis en place dans le logiciel quatre fonctions qui permettent de changer le nombre de pas, la vitesse, l'accélération et la décélération du moteur.

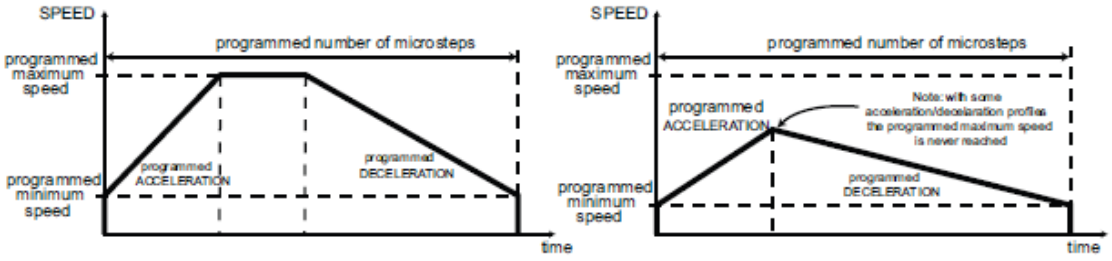


Figure 20 : Les commandes de mouvement pour les moteurs pas à pas

Je me suis, par la suite, intéressé à la variation du nombre de pas des moteurs et regarder en sortie la direction du vecteur quaternion. Puisque ce vecteur ne peut pas être interprété, j'ai traduit ce vecteur sous la forme des angles Euler. Les équations utilisées dans le code logiciel qui permettent de calculer les angles Euler sont :

$$Pitch = \tan^{-1}\frac{2 * (q_0 * q_1 + q_2 * q_3)}{{q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2}$$

$$Roll = \sin^{-1} 2 * (q_1 * q_3 + q_0 * q_2)$$

$$Yaw = \tan^{-1}\frac{2 * (q_1 * q_2 + q_0 * q_3)}{{q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2}$$

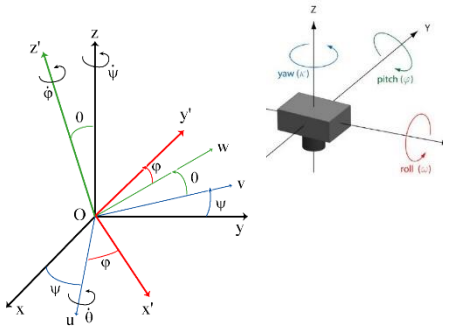


Figure 21 les angles d'Euler

La constante définie dans le logiciel RAD_TO_DEG de 57.2957795 permet de convertir les angles Euler de radian en degré.

Ensuite, j'ai envoyé des commandes de rotations simples dans un moteur afin vérifier la véracité des données calibrées. Les rotations effectuées par le moteur peuvent être interprétées par des changements d'angle de lacet. Une seule rotation complète du moteur correspond à 25600 pas. Cette valeur est variable selon le paramétrage effectué dans la partie initialisation des moteurs. Dans mon cas, j'ai configuré les moteurs en 128 microsteps. La commande qui doit être envoyée au moteur pour le tourner avec un angle α s'écrit de la forme suivante :

$\frac{\alpha}{1.8} . 128 = X$, où X correspond au nombre de microsteps à envoyer.

La figure ci-dessous montre les courbes théoriques et réelles des différents angles de lacet, envoyés au moteur et mesurés par le capteur fixé sur le moteur. J'ai remarqué que les valeurs deviennent de plus en plus fiables quand l'algorithme tourne plus longtemps. L'algorithme implémenté dans le logiciel est basé sur l'utilisation de matrices d'optimisation. Elles permettent d'adapter progressivement les nouvelles valeurs mesurées aux anciennes. J'ai remarqué que l'écart varie entre 2 et 9 degrés au début du lancement du logiciel et ce dernier se stabilise entre 1 et 3 degrés si on laisse tourner le logiciel tourné plus longtemps.

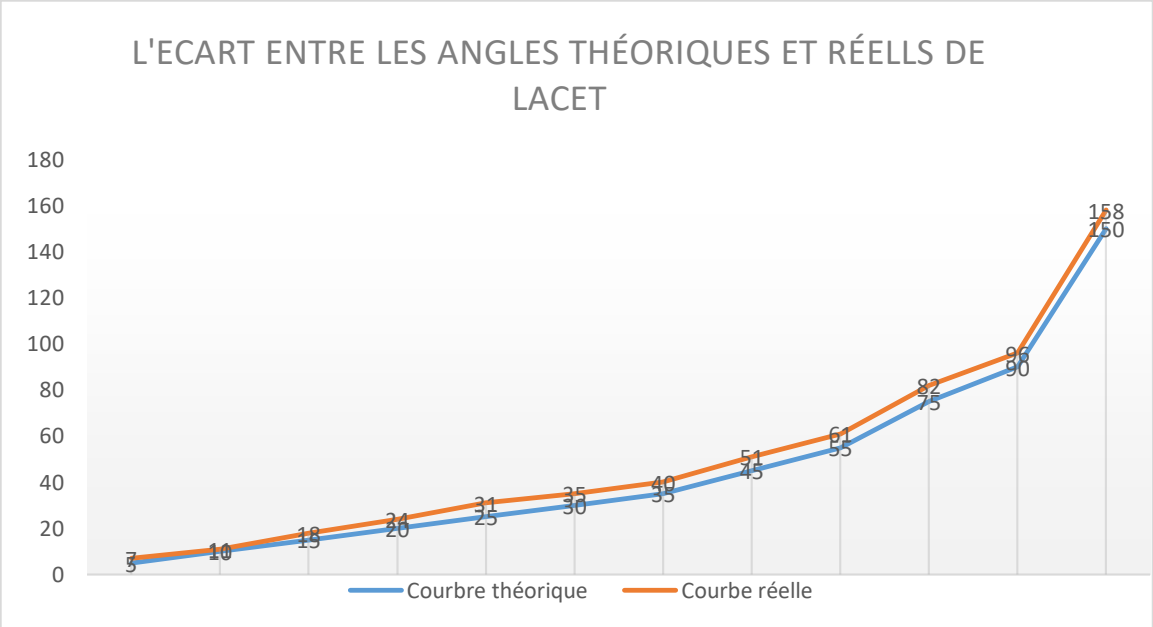


Figure 22 L'écart entre les mesures théoriques et réelles de Lacet

L'erreur de mesure qu'on peut remarquer est due à plusieurs raisons, par exemple à la calibration qui n'est pas optimisée ou aux coefficients de résolution que j'ai appliquée dans le code logiciel, qui ne sont pas suffisamment précis. On peut dire après plusieurs essais sur ce capteur que, les coefficients de calibration changent d'une journée à l'autre c'est pour cela nous avons créés trois fonctions pour calibrer le gyromètre, le magnétomètre et l'accéléromètre pour trouver des coefficients optimaux. Cependant, au vu de l'erreur infime, on peut dire que ce procédé est fonctionnel.

La figure ci-dessous montre les quatre coordonnées du vecteur quaternion calculé durant une mode de fonctionnement normal.

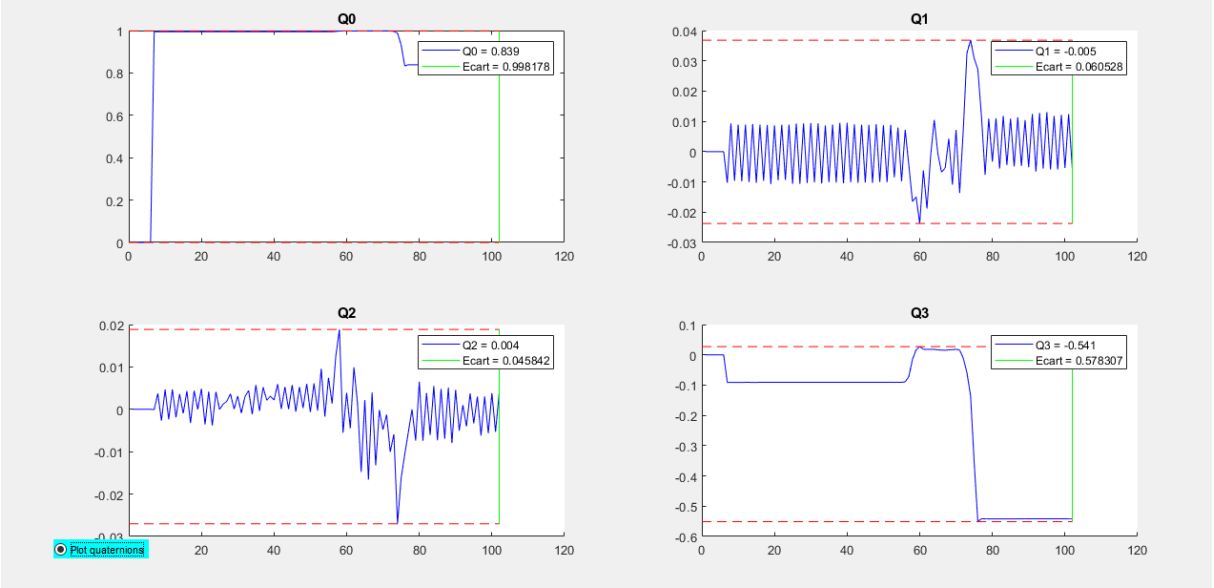


Figure 23 Les coordonnés du vecteur quaternion

Comme on peut regarder dans la figure ci-dessous, les angles d'Euler sont calculés à partir les coordonnés de quaternions.

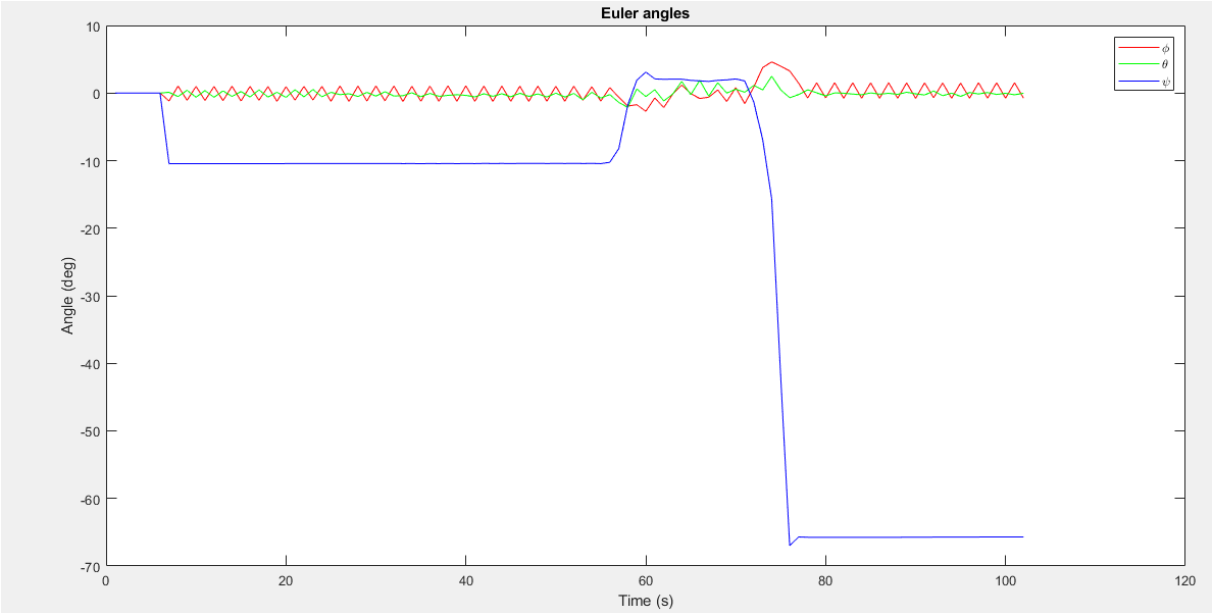


Figure 24 Les angles d' Euler calculés sur Matlab

Par la suite, l'étape suivante consiste à appliquer un filtre de Kalman sur les angles d'Euler pour prendre en compte l'écart entre la mesure réelle et la mesure théorique et donc améliorer les prochaines mesures.

III.7 Le filtre de Kalman

J'ai mentionné en annexe les différentes étapes pour appliquer ce filtre. Comme je l'avais expliqué, ce dernier permet de prendre en compte le bruit de mesure défini dans la fiche technique du capteur. Le capteur MPU9050 contient une erreur de 3.0462e-06, cette valeur doit être prise en considération lors du calcul des angles d'Euler.

L'utilisation de ce filtre est basée sur deux étapes essentielles :

- L'étape de prédiction qui utilise l'état précède pour produire une estimation de l'état courant.
- L'étape de mise à jour qui utilise les observations de l'état actuelle pour corriger le vecteur d'état de perdition et par suite d'obtenir une estimation plus précise.

L'algorithme de ce filtre est très lourd pour l'implémenter dans le logiciel de projet, j'ai seulement implémenté ce dernier sur Matlab pour évaluer son effet par des simulations graphique.

La figure ci-dessous montre l'effet du filtre de Kalman sur le calcul de l'angle de Lacet.

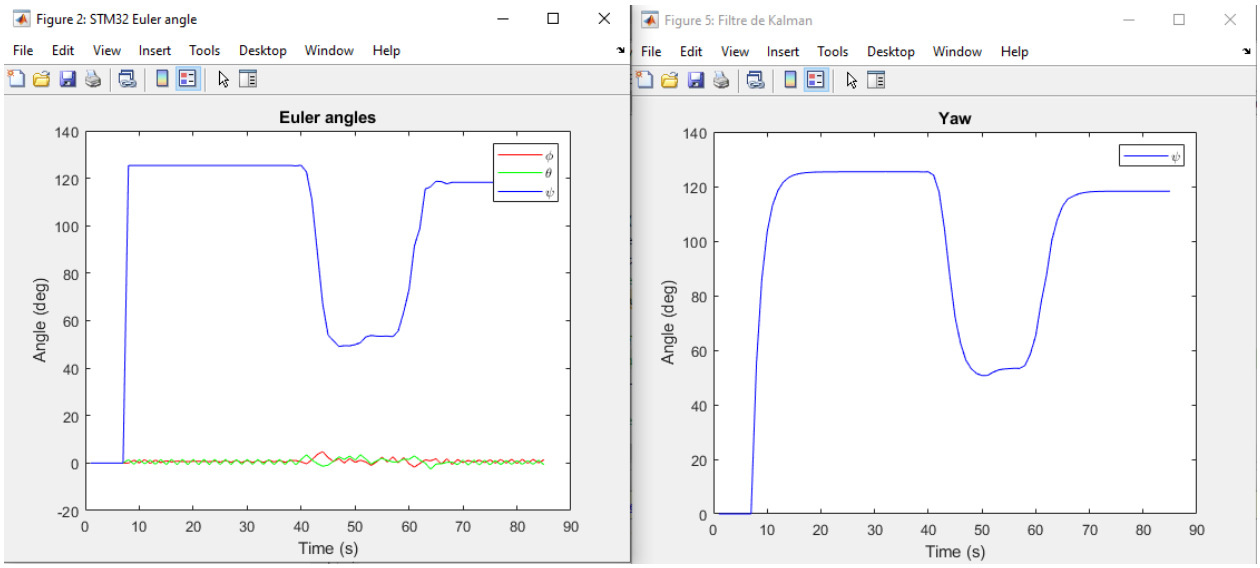


Figure 25 L'effet du filtre de Kalman sur le calcul d'angle de Lacet

J'ai constaté que le filtre de Kalman n'a pas une grande valeur ajoutée quand les mesures sont propres, cependant la courbe se rapproche de la mesure réelle grâce à la méthode récursive basée uniquement sur l'estimation de l'état actuel en fonction de l'état précédent et de la mesure actuelle.

IV. Conclusion

L'objectif de ce projet était de suivre une démarche scientifique permettant d'analyser les données brutes envoyées par une carte de traitement pour les robots sous-marins. J'ai réussi à concevoir un prototype fonctionnel pour vérifier les données acquises par deux centrales inertielles en même temps et donc de comparer si les mesures sont identiques des deux côtés.

L'un des enjeux majeurs de ce projet était la répartition du temps disponible en fonction des tâches à réaliser. C'est dans cette optique que j'ai débuté mon projet par une étude de faisabilité afin d'arriver ensuite à l'utilisation d'un diagramme de Gantt permettant de répartir toutes les tâches. Ce projet pourrait être proposé pour un futur stage ou une éventuelle reprise en projet de fin d'études.

L'une des améliorations potentielles du projet réside dans la mise en place dans le logiciel des mécanismes d'exploitation en temps réels, FreeRTOS, afin de gérer le flux de données envoyées par les deux capteurs fixés sur les moteurs pas à pas. D'autre part, suite à un problème technique de la carte de traitement fournie par l'entreprise partenaire, je n'ai pas eu l'occasion de comparer les données calibrées de cette dernière et celles de l'IMU 9 DOF Grove.

J'ai remarqué aux travers de ce projet que le domaine de l'embarqué est un sujet majeur dans le monde de l'entreprise de nos jours. Cela demande beaucoup de rigueur tout au long de la phase d'étude et la phase d'application. En effet pendant les six mois que j'ai passé à travailler sur ce sujet, j'ai acquis de nouvelles compétences dans le domaine de la programmation embarquée. De plus, j'ai acquis de l'expérience dans la gestion et l'organisation de projet. J'ai également développé plus de compétences dans le travail en autonomie.

De plus, j'ai gagné des compétences et de l'expérience technique, notamment sur l'approche et le nettoyage de données brutes issues de centrales inertielles. Il est pertinent aussi de préciser que j'ai utilisé mes expériences de projets antérieurs dans ce projet (programmation, compréhension et appréhension STM).

V. Abstract

In this project, the goal was to validate data received from the submarine's electronic board. My contribution to the project was to design a prototype that allowed the validation of the calibrated data after acquisition. Thus, I used an inertial measurement unit, or IMU, in order to determine the orientation of the submarine.

These IMUs included a gyrometer, a magnetometer and an accelerometer, of which the data received could then be used to compute a quaternion vector and get a general sense of the submarine's current orientation.

This project is useful as the program used to validate the data is modular and could be used on the same IMUs in another system.

VI. Bibliographie

Protocole_HY1600180_eCompass, ACR, 04/02/2020, ECA ROBOTICS

An efficient orientation filter for inertial and intertial/magnetic sensor arrays, Sebastian O.H. Madgwick, 30/04/2020

High power stepper motor driver expansion board based on powerSTEP01 for STM32, STMicroelectronics, 06/2015

System-in-package intergrating microstepping controller and 10 A power MOFSETs, STMicroelectronics, 01/2020

Nucleo-XXXXZX Nucleo-XXXXZX-P Nucleo-XXXXZX-Q Data brief, STMicroelectronics, 2021

MPU-9150 Product Specification Revision 4.3, InvenSense Inc.,18/09/2013

MPU-9150 Register Map and Descriptions Rveision 4.2, InvenSense Inc., 18/09/2013

Angle d'Euler, Wikipédia

Champ magnétique terrestre, Wikipédia

Filtre de Kalman, Wikipédia

VII. Annexes

VII.1 L'algorithme Madgwick AHRS

Vous trouverez ci-dessous les différents détails qui expliquent le principe de l'algorithme AHRS « Madgwick ».

Nous prônons le vecteur S qui représente les vitesses angulaires définies par :

$$S = [\,0\ g_x\ g_y\ g_z]$$

La dérivée du quaternion définie par la variation du changement d'orientation entre deux instants successifs.

$$\dot{q}_t = \frac{1}{2} * q_{t-1} \times S$$

$$q_t = q_{t-1} + \frac{1}{2} * (q_{t-1} \times S) * dt$$

$\times$ Ce symbole est désigné le produit de quaternion entre deux vecteurs, nous prenons l'exemple suivant :

$$Q_1 = [a_1\ b_1\ c_1\ d_1] \quad Q_2 = [a_2\ b_2\ c_2\ d_2] \quad Q_1 \times Q_2 = \begin{bmatrix} a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\ a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2 \\ a_1b_2 - b_1d_2 + c_1a_2 + d_1b_2 \\ a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2 \end{bmatrix}$$

En appliquant ce résultat sur l'équation :

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \begin{bmatrix} -q_1g_x - q_2g_y - q_3g_z \\ +q_0g_x + q_2g_z - q_3g_y \\ +q_0g_x - q_1g_z + q_3g_x \\ +q_0g_z + q_1g_y - q_2g_x \end{bmatrix}$$

Par suite, on passe par une intégration de variation du quaternion :

$$\begin{bmatrix} q_{t,0} \\ q_{t,1} \\ q_{t,2} \\ q_{t,3} \end{bmatrix} = \begin{bmatrix} q_{t-1,0} \\ q_{t-1,1} \\ q_{t-1,2} \\ q_{t-1,3} \end{bmatrix} + \frac{1}{2} * \begin{bmatrix} -q_1g_x - q_2g_y - q_3g_z \\ +q_0g_x + q_2g_z - q_3g_y \\ +q_0g_x - q_1g_z + q_3g_x \\ +q_0g_z + q_1g_y - q_2g_x \end{bmatrix} * dt$$

La normalisation du vecteur quaternion :

La normalisation est donnée par la formule suivante

$$q_{norm} = \frac{q}{|q|}\ avec\ |q| = \sqrt{{q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2}$$

$$f(q, a^S) = \begin{bmatrix} 2(q_1 q_3 - q_0 q_2) - a_x \\ 2(q_0 q_1 + q_2 q_z) - a_y \\ 2\left(\frac{1}{2} - {q_1}^2 - {q_2}^2\right) - a_z \end{bmatrix} \quad \text{et} \quad J(q) = \begin{bmatrix} -q_2 & 2q_3 & -2q_0 & 2q_1 \\ 2q_1 & 2q_0 & 2q_3 & 2q_2 \\ 0 & -4q_1 & -4q_2 & 0 \end{bmatrix}$$

La fonction gradient qui décrit les mesures du capteur d'accéléromètre à l'instant t est définie par :

$$\nabla f = J^T(q_{t-1)} * f(q, a^S)$$

Le résultat fourni ci-dessus nous permet uniquement d'avoir une estimation sur les orientations de roulis et de tangage. Nous avons passé ensuite à déterminer la fonction gradient du capteur de magnétomètre pour estimer l'orientation du cap d'objet.

$$f(q, d^E, s^S) = q^*.d^E.q - s^S$$

$$= \begin{bmatrix} 2d_x\left(\frac{1}{2} - {q_2}^2{q_3}^2\right) + 2d_y(q_0 q_3 + q_1 q_2) + 2d_z(q_1 q_3 - q_0 q_2) - s_x \\ 2d_x(q_1 q_2 - q_0 q_3) + 2d_y(\frac{1}{2} - {q_1}^2{q_z}^2) + 2dz(q_0 q_1 + q_2 q_3) - s_y \\ 2d_x(q_0 q_2 + q_1 q_3) + 2d_y(q_2 q_3 - q_0 q_1) + 2dz(\frac{1}{2} - {q_1}^2{q_2}^2) - s_z \end{bmatrix}$$

La solution qui vérifie q est trouvée par la résoudre de l'équation suivante : $\min f(q, d^E, s^S)$

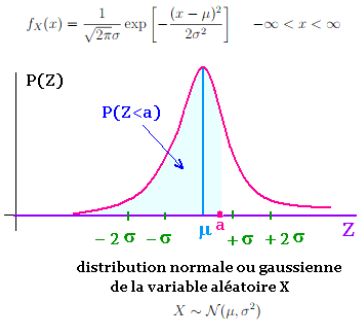
L'approche suggérée de cet estimateur consiste à utiliser l'algorithme de descente de gradient pour calculer la solution. Cet algorithme d'optimisation permet de trouver le minimum de n'importe quelle fonction convexe en convergeant progressivement vers celui-ci.

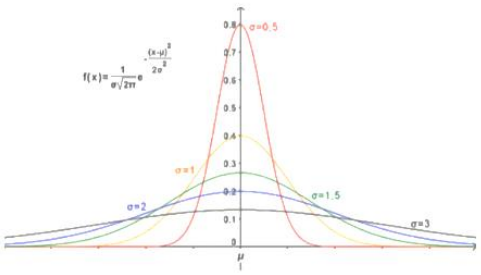
VII.2 Le filtre de Kalman

Le filtre de Kalman est un algorithme de traitement de données récursif qui estime l'état d'un objet à l'aide de mesure imprécise.

Cette méthode se repose sur deux étapes principales :

- L'étape de prédiction : permet de prédire le dernier état estimé par l'objet « la centrale inertielle ».
- L'étape de correction : permet d'utiliser les mesures bruitées pour corriger l'état prédit.





Les instruments sont bruitées ce qui veut dire que l'information mesurée doit être infectée par le bruit et donc on n'aura jamais la bonne mesure mais sur un moyen on peut avoir la bonne mesure. Ce filtre de Kalman va nous permettre d'évaluer la prochaine mesure en fonction des mesures qui ont été effectuées.

La condition : le bruit doit être gaussien puisque la probabilité calculée droit traitée en fonction du bruit inclue dans le signal mesurée.

Les sources d'erreur de mesure pouvant être due à cause de l'électronique de la carte ou due à l'algorithme de mesure des données brutes.

L'équation d'état :

$$E_t = A.E_{t-1} + q_{t-1}$$

E : le vecteur d'état, A : la matrice de transition, q le bruit gaussien (la matrice de covariance).

$$angle_t = angle_{t-1} + W_t.dt$$

$$E_t = \begin{bmatrix} Angle_t \\ W_t \end{bmatrix} \qquad A = \begin{matrix} 1 & -dt \\ 0 & 1 \end{matrix}$$

Le vecteur de mesure : $Z_t = H.E_t + r_t$

Z : le vecteur contenant les mesures

H : la matrice d'observation

r : le bruit gaussien (la matrice de covariance).

$$H = 1 \quad 0$$

Prédiction de l'état :

$$\hat{E}_{t+1} = A.\hat{E}_t$$

Estimation de la covariance de l'erreur :

$$P^{-}{}_{t+1} = A_t.P_t.A^T{}_t + Q_t$$

La correction :

Le gain de Kalman : $K_{t+1} = P_t.H^T{}_t.(H_t.P_t.H^T{}_t.R_t)^{-1}$

La correction/ innovation :

$$\hat{E}_{t+1} = \hat{E}_t.K.\left(Z_t - H^T.\hat{E}_t\right)$$

$$P_t = (Id - K_t.H_t).P^{-}{}_t$$