RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

---

Contribution à la Conception d'un Système de Reconstruction de Trajectoire pour Véhicules Deux-Roues Motorisés.

Ghouthi BOUKLI HACENE et Mouaadh KELLAL

Sous la direction de
Pr. Samir BOUAZIZ
Dr. Stéphane ESPIE
Dr. Rabah SADOUN

Présenté et soutenu publiquement le 18/06/2016

Composition du Jury :

Président	M. Mohamed TRABELSI	Prof.	ENP
Rapporteur	M. Rabah SADOUN	Dr.	ENP
Examinateur	M. Adel BELOUCHRANI	Prof.	ENP
Examinateur	M. Mourad ADNANE	Dr.	ENP

ENP 2016

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

---

Contribution à la Conception d'un Système de Reconstruction de Trajectoire pour Véhicules Deux-Roues Motorisés.

Ghouthi BOUKLI HACENE et Mouaadh KELLAL

Sous la direction de
Pr. Samir BOUAZIZ
Dr. Stéphane ESPIE
Dr. Rabah SADOUN

Présenté et soutenu publiquement le 18/06/2016

Composition du Jury :

Président	M. Mohamed TRABELSI	Prof.	ENP
Rapporteur	M. Rabah SADOUN	Dr.	ENP
Examinateur	M. Adel BELOUCHRANI	Prof.	ENP
Examinateur	M. Mourad ADNANE	Dr.	ENP

ENP 2016

Remerciements

Nous adressons nos remerciements aux personnes qui nous ont aidé dans la réalisation de ce projet de fin d'études.

Nous remercions en premier lieu Dr. Rabah SADOUN d'avoir accepté de nous diriger, de nous avoir fait confiance en nous intégrant dans un vrai projet de recherche, et de nous avoir conseillé et soutenu tout au long de notre travail.

Nous tenons particulièrement à remercier le professeur Samir BOUAZIZ, notre encadrant durant notre stage au DIGITEO à Paris de nous avoir orienté et aidé dans les moments difficiles.

Nous exprimons notre gratitude au Dr. Stéphane ESPIE, de nous avoir accueilli, et qui nous a prodigué de nombreux conseils. Nous retenons sa grande gentillesse et sa disponibilité.

Nos remerciements vont également à toute l'équipe du projet VIROLO++ de nous avoir reçus et d'avoir fait leur mieux pour répondre à nos questions techniques.

Puis, nous remercions les membres du jury : Professeur Mohamed TRABELSI président du Jury, Pr. Adel BELOUCHRANI et Dr. Mourad ADNANE de nous avoir fait l'honneur d'examiner notre mémoire de projet de fin d'études.

ملخص

في حين أن تقنيات سلامة المركبات رباعية العجلات في تحسن مستمر خلال العقود الأخيرة، فإن المركبات ثنائية العجلات ما زالت واحدة من أكثر وسائل التنقل خطورة. إن فقدان التحكم في المنعرجات يمثل أكثر من 50 بالمئة من أسباب حوادث المركبات الواحدة.

مشروع VIROLO++ الذي تم في إطاره تحضير هذه المذكرة، يهدف إلى التقليل من عدد حوادث الدراجات النارية عن طريق تصميم نظام للمساعدة على فهم طريقة أخذ المنعرجات.

في هذه المذكرة، نعرض هذا النظام بشكل عام، إضافة إلى مساهمتنا في تطويره.

الكلمات المفتاحية : السلامة المرورية، دمج معطيات المستشعرات، معالج رقمي للحركة، نظام إعادة تشكيل المسار.

Abstract

While four-wheeled vehicles safety technology is steadily improving throughout the last decades, PTW (Powered Two Wheeler) vehicles remain some of the least safe means of transport. The loss of control of PTW in bends accounts for over 50% of the causes of single vehicle accidents.

The VIROLO++ project, within which this thesis is presented, aims to reduce motorbikes accidents via the design of a system that will help understand bend taking practises.

In this thesis, we will present this system in general along with our contribution to its making.

Key words : Road safety, sensor data fusion, digital motion processor, path reconstruction system.

Résumé

Pendant que la technologie de sécurité des véhicules quatre-roues est en constante évolution au cours des dernières décennies, les véhicules deux roues restent l'un des moyens de transport les moins sécurisés. La perte de contrôle des véhicules deux-roues dans les virages provoque plus de 50% des accidents individuels.

Le projet VIROLO++, dans lequel s'inscrit ce mémoire, vise à réduire le nombre d'accidents des véhicules deux-roues à travers l'élaboration d'un système aidant à comprendre la façon de négocier un virage.

Dans ce mémoire, on présente ce système en général ainsi que notre contribution à sa conception.

Mots clés : Sécurité routière, fusion de données capteurs, processeur digital de mouvement, système de reconstruction de trajectoire.

Table des matières

Table des figures

Introduction

Les véhicules deux-roues sont les véhicules les plus vulnérables sur la route. Pour une même distance parcourue, le risque d'un accident fatal pour un motard est 35 fois plus important que celui d'un automobiliste. Plus de 80% des accidents moto rapportés résultent en blessures ou fatalités contrairement à 20% des accidents automobiles [1]. Cependant, le nombre des usagers des véhicules deux roues est en constante augmentation.

Les véhicules deux roues devraient alors bénéficier des dernières technologies de sécurité et de l'aide à la conduite.

Le projet VIROLO++ (ANR France), dans lequel se présente notre projet de fin d'études, s'inscrit dans ce contexte. Il vise à développer les outils et les méthodes nécessaires pour l'étude du comportement des conducteurs des véhicules deux roues lors de la prise de virage. Le projet s'intéresse particulièrement à cette manœuvre car plus de 50% des accidents moto individuels sont dus à la perte de contrôle en virage. Pour ce faire, un système embarquable sur moto est développé pour mesurer les interactions entre conducteur et véhicule.

Dans le cadre de ce projet, nous avons été chargés de la conception et l'étude de deux composantes du système : L'exploitation de la centrale de mesures inertielles et le développement d'un support de visualisation de données.

Ce rapport comporte, outre ce premier chapitre d'introduction, cinq autres chapitres dont la conclusion.

Le chapitre II introduit le modèle physique d'une moto et met l'accent sur les grandeurs utiles à mesurer. Ces grandeurs sont la base sur laquelle le choix des capteurs était fait.

Dans le chapitre III, le système développé par l'équipe VIROLO++ est présenté. Notre contribution au sein du projet est détaillée dans le chapitre IV qui traite le nœud visuel et la centrale de mesures inertielles.

Finalement, Le chapitre V expose et évalue les résultats expérimentaux.

Chapitre 1

Modèle d'un véhicule deux roues

1.1 Introduction

Instrumenter un véhicule deux roues nécessite un minimum de connaissances sur le modèle physique d'un tel système. Dans ce chapitre, le modèle physique d'un véhicule deux roues est présenté. Ce modèle comporte les grandeurs géométriques et dynamiques qui décrivent l'état et le mouvement d'une moto.

1.2 Modélisation d'un véhicule deux roues

Un modèle de moto (figure 1.1) se compose de plusieurs parties qui sont[2] :

- la masse suspendue (cadre ou châssis) reliée aux masses non suspendues (roues) par des suspensions (ressorts, amortisseurs),
- 2 masses non suspendues (roues),
- le contact roue/sol,
- des entrées de braquage issues du guidon.

Cependant la modélisation de la moto doit être faite en tenant compte de quelques spécificités qui lui sont liées. En effet, nous pouvons noter que :

- la moto reste instable à l'arrêt sans intervention extérieure,
- l'angle de roulis est beaucoup plus important dans le cas d'une moto. Ce qui peut rendre son renversement plus fréquent,
- l'angle de carrossage est plus important,
- le rôle du pilote est très important puisqu'il agit sur l'angle de guidon pour le maintien de la moto sur sa trajectoire en toute sécurité.

1.2.1 Géométrie d'une moto

On considère deux caractéristiques importantes [3] :

- Angle de chasse : L'angle que fait l'axe du guidon avec la perpendiculaire de la roue au sol, au point de contact avec la roue. Un angle de chasse grand implique plus de stabilité mais une direction plus lourde.

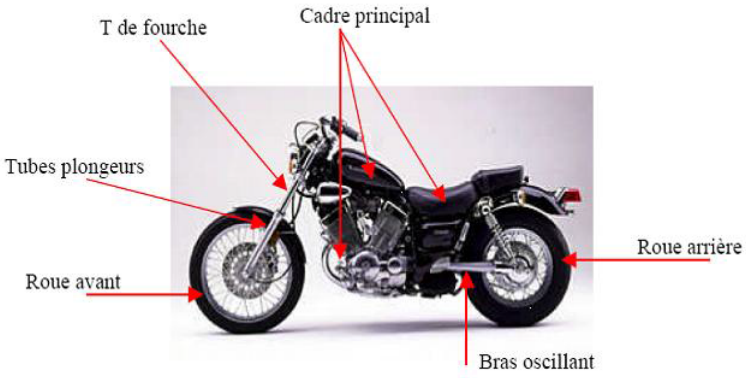


FIGURE 1.1: Schéma d'un deux roues motorisés

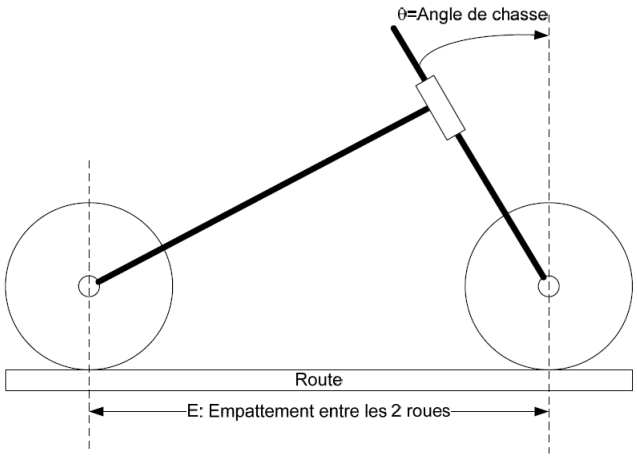


FIGURE 1.2: Géométrie d'une moto

- Empattement : Distance entre points de contact roues-sol. Un empattement plus grand donne plus de stabilité en ligne droite mais une prise de virage plus difficile (figure 1.2).

1.2.2 L'effet gyroscopique de la roue

Lorsqu'une roue en rotation (Axe B) est soumise à un couple sur un axe différent (Axe A : Rotation du guidon), un couple perpendiculaire aux deux axes (Axe C) se produit. Ceci est dit : Effet Gyroscopique. figure 1.3

1.2.3 Prise de virage en moto

Pour prendre un virage à gauche, tourner le guidon à droite va entraîner une force qui va vouloir faire tourner la roue à gauche et par voie de conséquence faire pencher la moto à gauche ; cette technique est appelée le contre-braquage. Les différents moments que subit la moto sont :

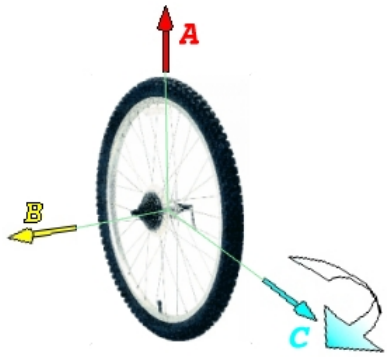


FIGURE 1.3: L'effet gyroscopique de la roue

1.2.3.1 Roulis

Le roulis est le résultat de l'effet gyroscopique sur la moto. Si le pilote veut prendre un virage, il doit tourner le guidon et donc créer un basculement (Effet gyroscopique). Le roulis dépend donc de l'angle de braquage par rapport au sol. Ce dernier est fonction de l'angle de braquage du guidon ainsi que l'angle de chasse. Seul l'angle de braquage guidon est facilement mesurable. L'angle de braquage par rapport au sol est déduit du modèle de la moto.

1.2.3.2 Tangage

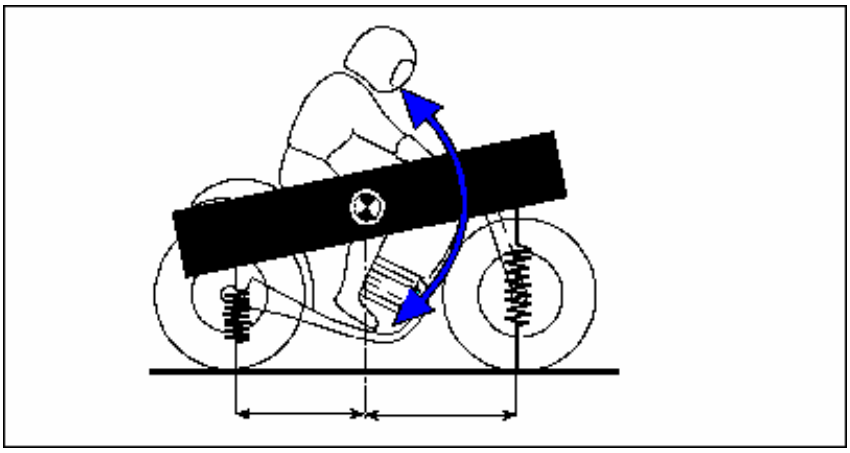


FIGURE 1.4: Angle de tangage

Lors des accélérations, le couple de la roue arrière fait lever l'avant de la moto. Le phénomène inverse se produit dans le cas de freinage (figure 1.4). Il est important alors de mesurer le freinage ainsi que les accélérations.

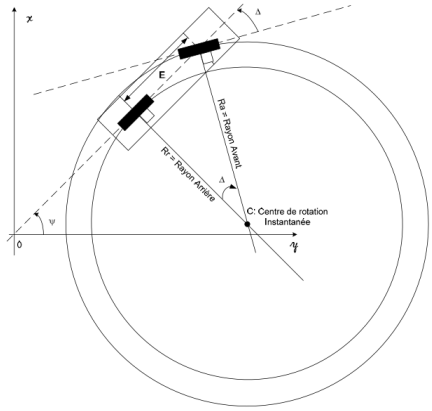


FIGURE 1.5: Angle de lacet - Vue de dessus

1.2.3.3 Lacet

Lors d'un virage à angle de braquage supposé constant, l'angle de lacet qu'effectue la moto est l'angle (noté Δ) entre les deux lignes droites portées par les roues avant et arrière. L'intersection des deux lignes définit le centre des deux cercles inscrits par les deux roues lors du virage. Ce point est dit : centre de rotation instantanée. En pratique, la moto est penchée lors d'un virage (figure 1.5). Par conséquent, le centre de rotation instantanée est modifié par l'angle de roulis.

1.2.4 Bilan des forces

Dans les conditions normales (Pas de glissade), on établit le bilan de forces agissant sur la moto :

1.2.4.1 Bilan des forces en ligne droite

Le bilan de forces en ligne droite est illustré par la figure 1.6

En réalité, la moto n'effectue jamais des lignes droites parfaites. Le motard assure en permanence sa stabilité. La moto est à 99.99% de son temps en virage.

1.2.4.2 Bilan des forces en virage

Le bilan de forces en virage est illustré par la figure 1.7

1.2.5 Récapitulatif des actions du pilote

- Une action sur l'angle de braquage au sol induite, soit par un angle de braquage au guidon, soit par une inclinaison du corps qui entraîne le braquage du guidon,
- Une action sur les freins,
- Une action sur l'accélérateur.

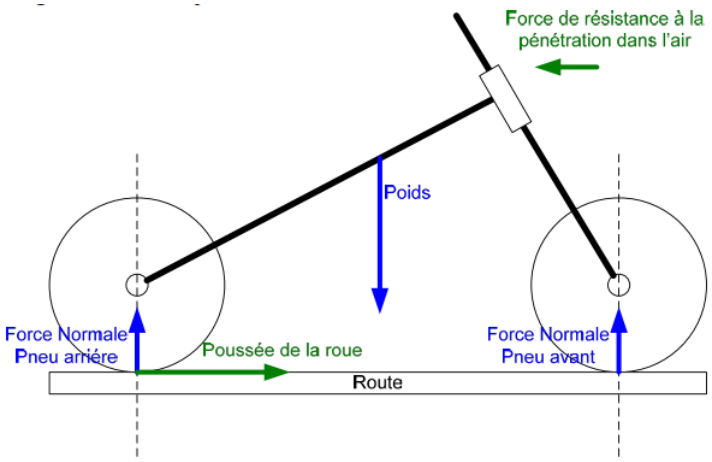


FIGURE 1.6: Bilan des forces en ligne droite

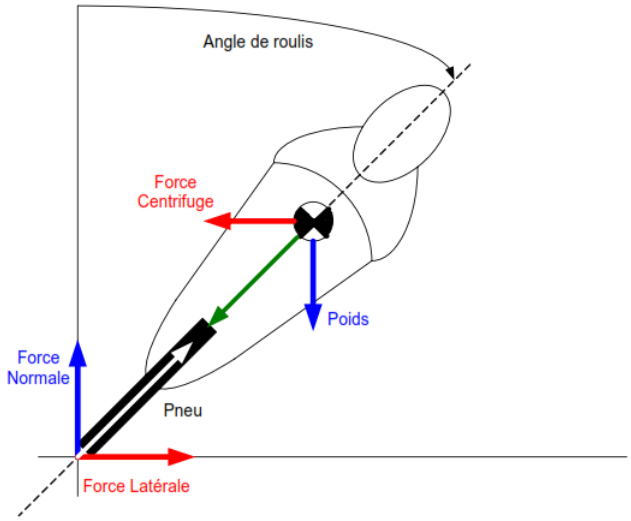


FIGURE 1.7: Bilan des forces en virage

1.3 Conclusion

Dans ce chapitre, les grandeurs définissant l'état et la dynamique d'une moto étaient discutées. Afin de pouvoir relever des mesures utiles, le système conçu par l'équipe VIROLO++ devait comporter un certain nombre de capteurs, à savoir :

- Accéléromètre pour mesurer les accélérations sur les trois axes.
- Gyroscope qui permettra la mesure des moments appliqués sur la moto.
- Système de référence d'orientation qui donnera les angles cités dans le modèle.
- Capteur d'angle guidon et capteur de freinage pour relever l'interaction du conducteur sur la moto.
- Capteur de localisation GPS qui montrera la trajectoire globale empruntée par la moto.

Les capteurs utilisés pour mesurer ces grandeurs sont explicités dans le chapitre 3.

Chapitre 2

Système de reconstruction de trajectoire pour une 2RM

2.1 Introduction

Dans ce chapitre, le système élaboré par l'équipe VIROLO++ est présenté d'une façon générale. Ce système est constitué essentiellement des capteurs spécifiés dans le chapitre précédent, inter-connectés via un bus CAN avec des nœuds d'enregistrement et de visualisation de données.
Le système en question doit respecter certains critères, parmi eux :

- Embarquabilité du système.
- La flexibilité du système : Architecture de système facilement modifiable.
- Consommation minimale de puissance.
- Fréquences de capteurs suffisamment supérieures à la dynamique de la moto.
- Données capteurs qui permettent une reconstruction de trajectoire avec une précision de 10cm.
- Un système de visualisation des données capteurs et de la trajectoire.
- Un coût réduit.

2.2 Architecture hardware

Le système de datalogging comporte plusieurs nœuds connectés par un seul bus CAN. Un nœud peut être producteur de données (nœud capteur), consommateur de données (nœud enregistreur) ou nœud d'affichage de données (nœud visuel) (figure 2.1).

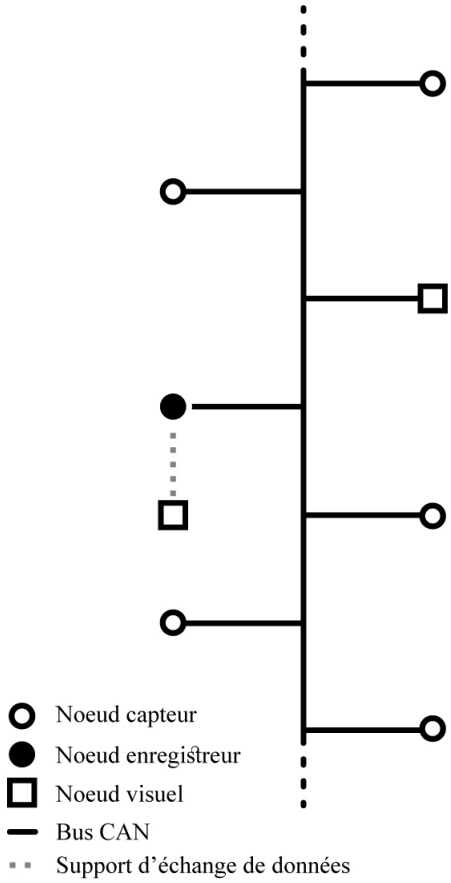


FIGURE 2.1: Architecture matérielle générale du système

Cette architecture permet d'ajouter ou retirer un ou plusieurs nœuds (capteurs ou enregistreurs) selon le besoin, sans affecter le fonctionnement des autres nœuds. Ceci offre une flexibilité de travail primordiale dans la phase de développement du système.

2.2.1 Nœud capteur

Un nœud capteur est un nœud producteur de données. Il se compose d'un microcontrôleur programmable ARM Mbed ainsi qu'un ou plusieurs capteurs et un connecteur CAN comme illustré par la figure 2.2 :

Dans le cas général, le microcontrôleur fait objet d'interface entre les capteurs et le bus CAN. Il reçoit les données provenant des capteurs via les protocoles appropriés (I2C, SPI, UART, etc...), les met dans un format spécifique (2.3.0.1) et les envoie sur le bus CAN.

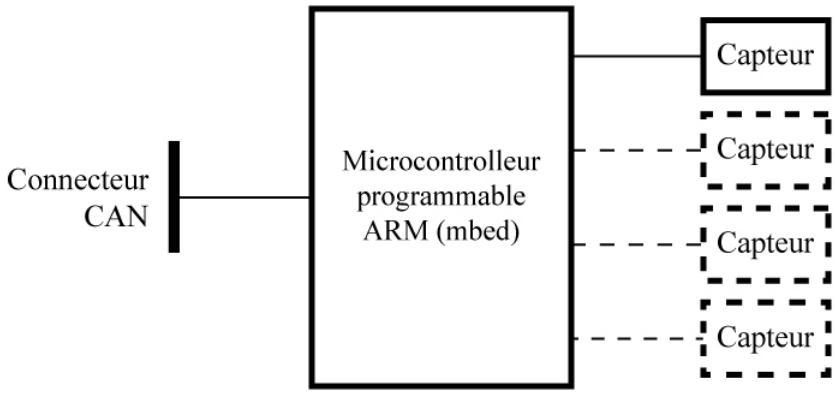


FIGURE 2.2: Schéma général d'un noeud capteur

2.2.1.1 Les cartes ARM Mbed

Les microcontrôleurs utilisés dans les nœuds capteurs sont des cartes de la plateforme ARM Mbed.
Mbed est une plateforme basée sur des microcontrôleurs ARM Cortex-M 32bits. La plateforme Mbed est développée par ARM en collaboration avec d'autres partenaires.

Applications Mbed Les applications pour la plateforme Mbed peuvent être développées en utilisant l'IDE (Integrated Development Environment) en ligne de Mbed, il s'agit d'un éditeur de texte et d'un compilateur de code en ligne gratuit dans lequel le code est écrit et compilé dans un navigateur Web.
L'environnement de développement de Mbed fournit des espaces de travail privés avec possibilité d'importer, d'exporter et de partager du code. Les applications peuvent être développées aussi avec d'autres environnements de développement tels que Keil μVision, IAR Embedded Workbench, et Eclipse.

SDK (Software Development Kit) Le SDK de Mbed fournit une plateforme Mbed logicielle qui comporte un ensemble de librairies permettant la création de micro logiciel pour microcontrôleurs. Cette plateforme logicielle permet de compiler le même code sur les différentes cibles (cartes) supportées par Mbed.

HDK (Hardware Development Kit) Le kit de développement matériel Mbed fournit des informations pour construire du matériel spécifique pour soutenir le SDK Mbed. Il est composé de l'interface firmware et schémas pouvant être utilisés pour créer

facilement des cartes de développement, modules OEM et les produits reprogrammables adaptées à la production.

Microcontrôleurs Les microcontrôleurs Mbed sont une série de modules de prototypage Mbed officiels basés sur le HDK Mbed. Ils offrent des solutions de prototypage rapides et flexibles Il s'agit d'une solution professionnelle de conception. Les microcontrôleurs Mbed sont conçus autour du core ARM Cortex –M 32 bit.
Les microcontrôleurs utilisés dans les nœuds capteurs du système sont :

- Mbed LPC 1768 (figure 2.3)
- STM32 F303 K8 (figure 2.4)

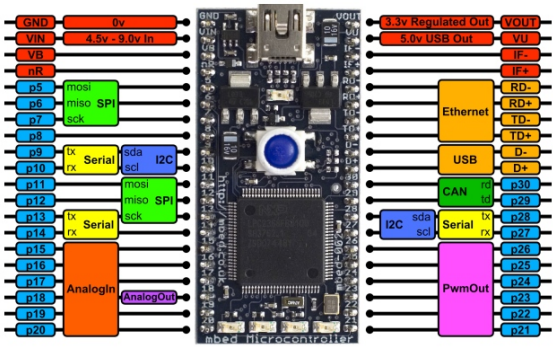


FIGURE 2.3: Carte Mbed LPC 1768

Mbed LPC 1768 Fonctionnalités :

- ARM® Cortex™-M3 Core 96MHz
- 32KB RAM, 512KB FLASH
- Ethernet, USB maître/esclave, 2xSPI, 2xI2C, 3xUART, CAN, 6xPWM, 6xADC, GPIO sur 40-pins
- Alimentation 5V USB où 4.5-9V
- Programmateur FLASH intégré « USB drag 'n'drop »
- Compilateur en ligne
- SDK haut niveau C/C++
- Espace de partage de librairies et projets

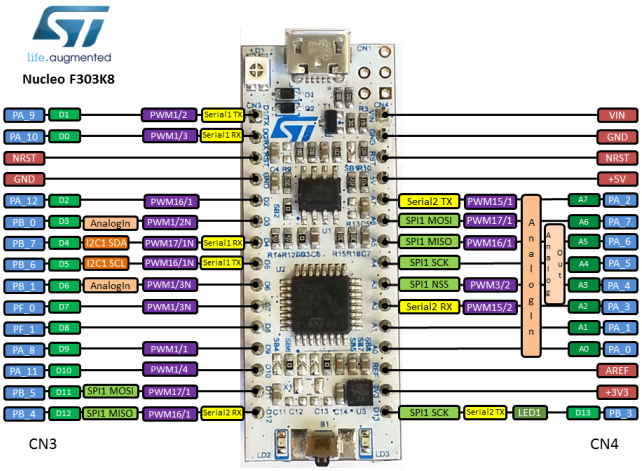


FIGURE 2.4: Carte STM32 F303K8

La carte STM32 F303K8 Fonctionnalités :

- ARM®32-bit Cortex®-M4 CPU avec FPU
- Fréquence CPU maximale de 72 MHz
- VDD de 2.0 V à 3.6 V
- 64 KB Flash
- 16 KB SRAM
- Timers
- SPI/I2S (1)
- I2C (1)
- USART (2)
- CAN (1)
- 12-bit ADC (2), 9 canaux
- 12-bit DAC (2), 3 canaux
- GPIO (25) avec possibilité d'interruption externe

2.2.1.2 Capteurs utilisés

Le système, comme conçu par l'équipe VIROLO++, comporte les capteurs suivants :

- Capteur guidon,
- capteur tour de roues,
- capteur frein,
- capteur d'inclinaison,
- capteur GPS,
- centrales de mesures inertielles.

Ce qui suit présente ces capteurs de façon plus détaillée :

Guidon Ce capteur est utilisé pour récupérer l'angle absolu du guidon par rapport au châssis. Il s'agit d'un capteur magnétique AS5047D (AMS) [4]. Le capteur comprend une

puce intégrée sensible au champ magnétique fixée sur le châssis et un aimant permanent fixé sur le guidon. La rotation du guidon entraîne un changement de la direction du champ magnétique par rapport à la puce qui traduit ce changement en un angle absolu (figure 2.5). L'angle est codé sur 14bits qui permet une résolution maximale de 2048 pas par rotation complète soit 0.176°/pas.
La puce communique avec un microcontrôleur STM32F303 via le protocole SPI et envoi

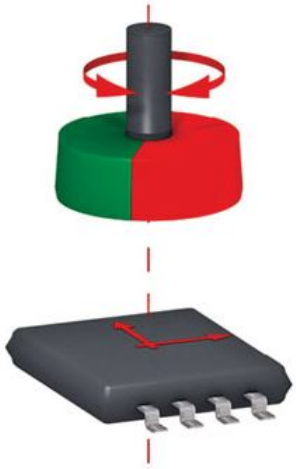


FIGURE 2.5: Capteur guidon : Capteur magnétique d'angle

la donnée angle absolu avec une fréquence d'actualisation de 4Mhz, le microcontrôleur ré-échantillonne cette donnée avec une fréquence de 100Hz pour l'envoyer sur le bus CAN.

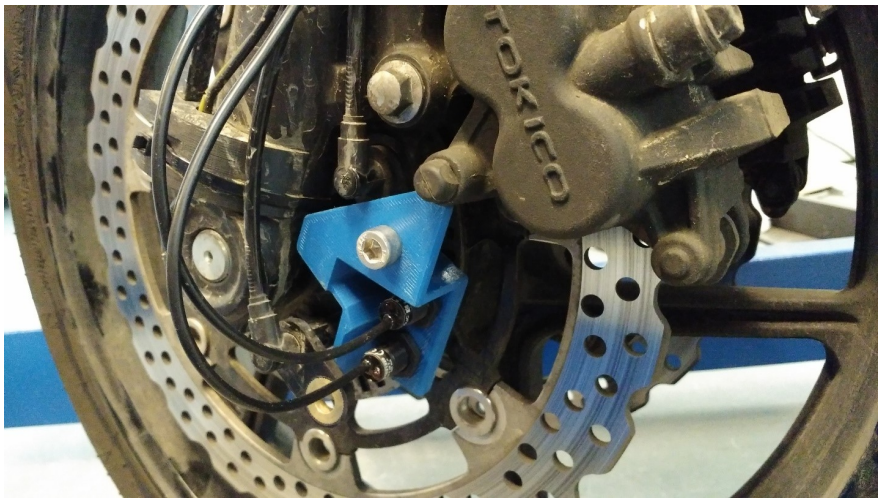


FIGURE 2.6: Les deux capteurs à effet hall sur la moto

Tour de roue Deux capteurs à effet Hall en quadrature sont fixés sur la roue avant (figure 2.6) pour mesurer la vitesse angulaire de roue et en déduire la vitesse longitu-

dinale de la moto. Il s'agit d'une sonde sensible au champ magnétique qui distingue la présence des dents métalliques sur l'anneau fixé sur la roue. Dans le cas d'une rotation de la roue, le capteur génère des impulsions dont la fréquence est reliée directement à la vitesse angulaire de la roue. En sachant le diamètre de la roue, on peut déduire la vitesse longitudinale. Les deux capteurs sont utilisés le long d'un compteur quadratique pour permettre l'identification du sens de rotation de la roue.

Frein L'action de freinage est prise en considération et non son intensité. Donc un simple contact (tout ou rien) fera office de capteur. Cette information est récupérée sur un microcontrôleur Mbed LPC 1768 à travers une entrée numérique sur 1 bit.

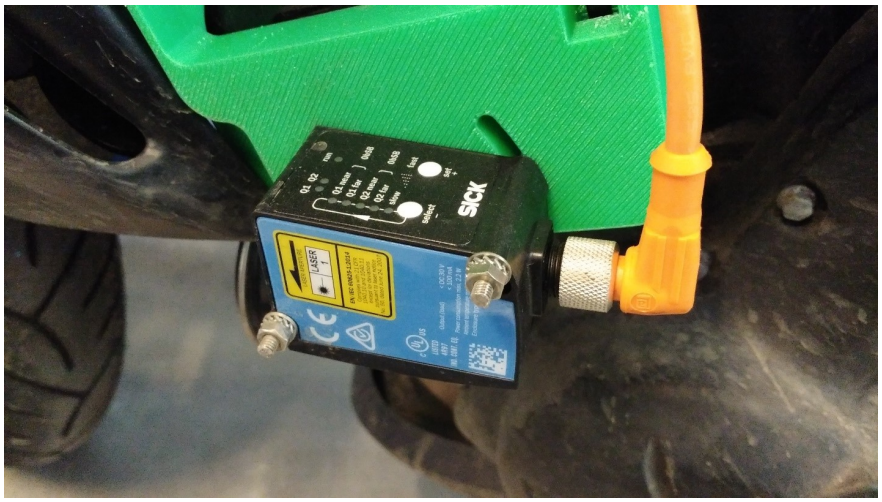


FIGURE 2.7: Un des deux capteurs laser sur la moto

Inclinaison (laser) Pour mesurer l'inclinaison de la moto, deux capteurs optiques identiques (laser) de distance sont placés sur les deux cotés (droit et gauche) de la moto (figure 2.7). L'angle d'inclinaison peut être donc calculé à partir de la différence suivant une formule empirique déduite d'une série de mesures (différence de distance / angle). La relation entre la différence de distance et l'angle est linéaire dans l'intervalle -30° à 30°, ce qui couvre la dynamique d'inclinaison d'une moto.

Le capteur de distance utilisé est un capteur SICK D35-B15551 [5] interfacé avec la carte Mbed via une pin analogique.

GPS Pour localiser la moto géographiquement, un module GPS Maestro A2200-A [6] (figure 2.8) est utilisé et qui interface avec la carte Mbed LPC 1768 via une liaison série UART.

Le module GPS travaille sur la bande L1 (1.575 MHz) et assure une précision horizontale de position de moins de 2.5 m. En plus de la position, le module permet de recevoir les données temps et vitesse dans un format NMEA.

FIGURE 2.8: Module GPS Maestro A2200-A

Le Système Mondiale de Géo-Positionnement (Global Positioning System) est un système de navigation basé sur satellite. Il fournit les informations de position et temps sur n'importe quel point de la terre à condition que 4 ou plus satellites GPS soient visibles. Les satellites envoient leurs positions (connu avec précision) ainsi que l'heure exacte indiquée par leurs horloges atomiques (figure 2.9). Le module GPS calcule sa position et vitesse à partir des informations reçues [7].

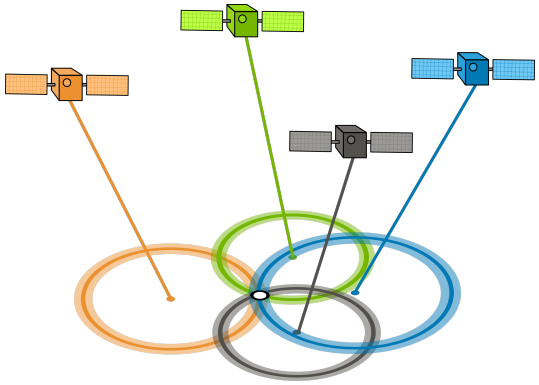


FIGURE 2.9: Principe de fonctionnement du GPS

La mesure de position en utilisant un seul module GPS subit des erreurs dû à :
- La distorsion du signal à cause des effets atmosphériques
- Les erreurs d'éphémérides
- L'effet Multi-trajets (réflexions de l'onde)
- La désynchronisation des horloges

GPS RTK Dans le but de calibrer les données du GPS Maestro, on utilise comme référence un récepteur GPS Septentrio Altus APS-3 [8] placé sur le coffre de la moto (figure 2.10b) avec un autre fixe placé sur un point connu en mode station de base (figure 2.10a). Cette configuration permet un positionnement précis au centimètre prés.

Ce capteur ne figurera pas dans le système final. Il est utilisé comme référence uniquement dans la phase de développement du système.

Le récepteur Altus APS-3 communique avec la carte Mbed LPC 1768 via une connexion série RS232.

(a) Altus APS-3 - Station de base

(b) Altus APS-3 - rover

FIGURE 2.10: Configuration GPS RTK : Rover - Station de base

RTK (Real Time Kinematic) est une technique utilisée pour améliorer la précision de positionnement GPS. Elle utilise les mesures de la phase du signal et les compare avec les mesures provenant d'une station de base dont la position est connue. Les corrections se font en temps réel via une communication radio entre l'unité mobile et la station de base.

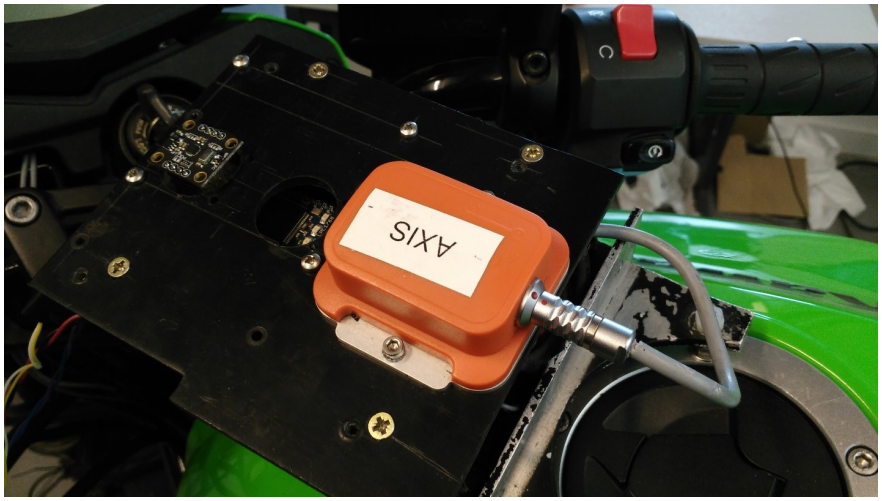


FIGURE 2.11: Centrale inertielle Xsens MTi sur la moto

Xsens (Centrale Inertielle de référence) La Xsens MTi (figure 2.11) est un système miniaturisé et léger de référence de cap et d'attitude (AHRS) à 9 degrés de liberté. Le MTi contient des accéléromètres, Gyroscopes et Magnétomètres suivant les trois axes, faisant ainsi une unité inertielle de mesure (IMU : 3.2.1). L'orientation 3D est calculée par son processeur interne implémentant un filtre de Kalman pour la fusion de données [9].

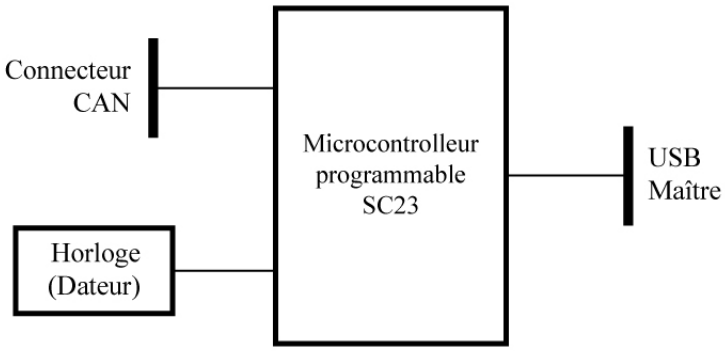
La Xsens est utilisée dans le système comme référence d'accélérations, vitesses angulaires, champ magnétique et d'orientation.

l'IMU communique avec le microcontrôleur STM32 F303F8 via une liaison UART en utilisant un MAX3232 comme pont d'adaptation.

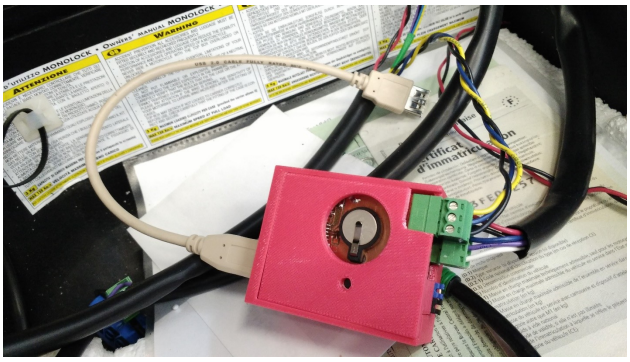
IMU (Centrale Inertielle MPU9250) L'étude du noeud 10DOF et l'utilisation de l'MPU-9250 sont détaillées dans le chapitre suivant.

2.2.2 Nœud enregistreur

Un nœud enregistreur est un nœud consommateur de données. Le système actuel ne fait appel qu'à un seul nœud enregistreur qui se compose d'un microcontrôleur programmable SC23 ainsi que d'un USB maître et un connecteur CAN comme illustré dans le schéma de la figure 2.12a.



(a) Schéma fonctionnel



(b) Sur la moto

FIGURE 2.12: Noeud enregistreur

FIGURE 2.13: Microcontrôleur IPC@CHIP SC23

2.2.2.1 Microcontrôleur IPC@CHIP SC23

SC23 est un contrôleur web intégré, développé par BECK pour les applications de communications et contrôle industriel (figure 2.13). Cette plateforme est équipée d'un système d'exploitation temps-réel multitâches avec un système fichier et une interface d'application [10].

Fonctionnalités :

- Processeur SC186-EX avec une fréquence de 96 MHz
- 8 MB DRAM
- 2 MB Flash (SC23)
- 1 x 10/100 MBit/Ethernet avec PHY
- 3 interfaces séries RS232/TTL
- 2 x CAN2.0b
- 1 x USB1.1 Maître ou Esclave
- 1 x SPI, 1 x I2C
- 17 GPIO, 3 entrées d'interruption. Timers matériels.
- Alimentation 3.3V
- IPC@CHIP® RTOS, TCP/IP, serveur web et C-API

2.2.2.2 Fonctionnement du nœud enregistreur

La fonction principale de l'enregistreur est de collecter les messages envoyés sur le Bus CAN, les mettre dans un format spécifique en fonction de l'identificateur du message et les placer dans un fichier C2U sur la clé USB. Lors de la réception d'une trame CAN, l'enregistreur écrit un nouveau bloc de données dans le fichier C2U contenant la trame CAN et le temps de réception. L'enregistreur date le fichier à l'aide d'un module horloge temps réel (Real Time Clock) muni d'une petite pile.

2.2.3 Nœud visuel

Le nœud visuel est le nœud qui intéresse le plus l'utilisateur, car il lui permet de visualiser les informations capteurs et système. Il est donc une interface homme-machine (IHM)

Le système fait appel à deux nœuds visuels :

- Indicateur : qui affiche l'état des capteurs permettant ainsi de savoir si ces derniers communiquent avec le datalogger.
- Visuel mobile : ce nœud permet après une manipulation de visualiser les données des différents capteurs, et de les comparer afin de pouvoir les interpréter et les exploiter.

2.2.4 Système d'interconnexion et protocoles de communication

Les différents noeuds du système sont inter-connectés via le bus CAN. Nous définissons dans cette partie le bus CAN et les protocoles utilisés entre capteurs et microcontrôleurs.

2.2.4.1 Bus CAN

Le bus CAN [11] (Controller Area Network) originalement développé pour l'automobile, permettant aux microcontrôleurs et aux différents dispositifs électronique de communiquer entre eux sans faire appel à un ordinateur hôte. C'est un protocole à base de message conçu pour la communication entre composants électroniques dans les automobiles. Il est aussi utilisé dans d'autres applications.

L'architecture du bus CAN Le CAN est un bus de données sérié Half-duplex (communication dans un seul sens à la fois).

Les dispositifs connectés sont appelés nœuds et chaque nœud et connecté au bus à travers une paire torsadée. Les Deux bouts du bus CAN sont rebouclés par deux résistances de 120 ohm. Ces résistances font office d'adaptation et servent à prévenir les réflexions des ondes pour éviter les interférences (figure 2.14).

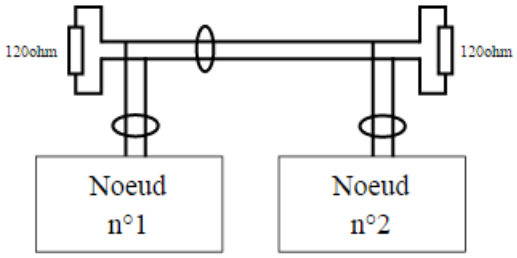


FIGURE 2.14: Schéma d'un bus CAN

La composition d'une trame CAN La trame envoie des données aux autres nœuds et elle se compose de 7 parties (figure 2.15) :

- Le début de trame ou SOF (Start Of Frame) matérialisé par 1 bit dominant,
- Le champ d'arbitrage (identificateur) composé de 12 ou 30 bits,
- Le champ de commande (ou de contrôle) composé de 6 bits,
- Le champ de données composé de 0 à 64 bits (de 0 à 8 octets),
- Le champ de CRC composé de 16 bits,
- Le champ d'acquittement composé de 2 bits,
- La fin de trame ou EOF (End of Frame) matérialisée par 7 bits récessifs.

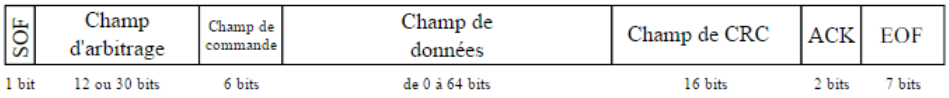


FIGURE 2.15: Format d'une trame CAN

Transmission de données Dans le cas où un nœud commence la transmission, à chaque fois qu'il transmet un bit il se met à l'écoute du bus pour voir si le bit transmit est le bit envoyé. Si ce n'est pas le cas, cela voudrait dire que le bit était écrasé par un autre bit dominant (un bit est dominant s'il est égal à 0) envoyé par un autre nœud. Le premier noeud interrompe donc sa transmission et il attend la fin de l'envoie des données afin de réessayer à nouveau.

2.2.4.2 Protocole I2C

I2C [12] (inter-integrated circuit) est un bus de communication inventé par Philips Semiconductor, utilisé pour relier des périphériques de faible vitesse à des microcontrôleurs sur une courte distance.

Topologie I2C est un bus série bidirectionnel synchrone half-duplex, qui peut connecter plusieurs périphériques qui soient maîtres ou esclave entre eux. L'échange de données se fait uniquement entre un maître et un ou plusieurs esclaves à l'initiative du maître cependant un dispositif peut changer son statut de maître à esclave et vis-versa (figure 2.16).

La connexion électrique est faite à base de deux lignes qui sont :

- SDA (Serial Data Line) : ligne de données bidirectionnelle.
- SCL (Serial Clock Line) : ligne de l'horloge de synchronisation.

Fréquence de communication Le temps de transmission est défini par rapport au mode de transmission. Les deux modes principaux sont :

- Mode standard : avec une vitesse maximale de 100 Kbits/s.
- Fast mode : avec une vitesse maximale de 400 Kbits/s.

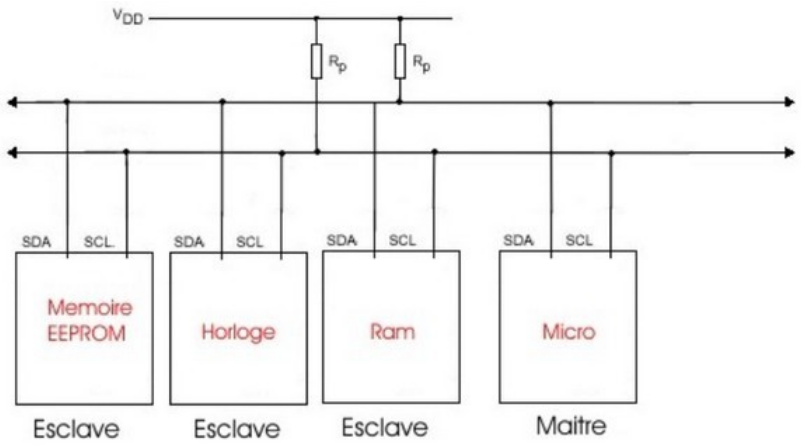


FIGURE 2.16: Schéma d'un bus I2C

2.2.4.3 Protocole SPI

SPI [13] (Serial Peripheral interface) est un bus de communication série synchrone, utilisé pour des communications à courte distance.

Contrairement à I2C, le SPI communique en full-duplex et ne comporte qu'un seul maitre, et plusieurs esclaves.

Topologie Le bus SPI fait appel à quatre signaux (figure 2.17) :

- SCLK (Serial Clock) : Horloge généré par le maitre.
- MOSI (Master output, Slave input) : généré par le maitre.
- MISO (Master input, Slave output) : généré par le maitre.
- SS (Slave Select) : actif à l'état bas, généré par le maitre pour sélectionner l'esclave à qui il souhaite lui envoyer de l'information.

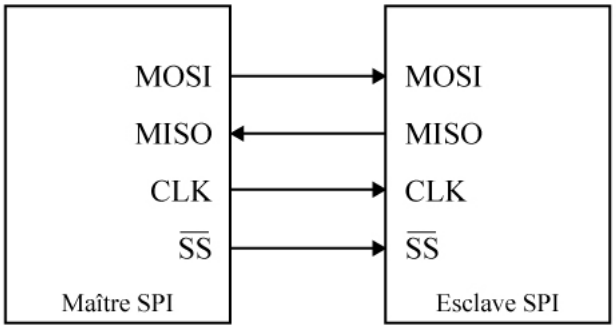


FIGURE 2.17: Configuration maître-esclave en SPI

Fréquence de communication La fréquence maximale d'un adaptateur SPI peut atteindre une valeur de 100 MHz, qui est une fréquence considérablement élevée comparée avec la fréquence du I2C. Un système communiquant via SPI est donc naturellement plus rapide. Cependant, ce protocole n'admet pas la notion d'adressage et donc il doit ajouter une ligne de selection (SS) pour chaque esclave.

2.2.4.4 UART

UART [14] (Universal Asynchronous Receiver Transmitter) est un bus de communication série full-duplex qui utilise généralement trois lignes électrique à savoir :

- Ligne de transmission TX,
- Ligne de réception RX,
- Ligne de masse commune entre les deux dispositifs GND.

UART comprend une propriété très importante qui est le baud et qui représente une unité de vitesse de transmission équivalente au bit/seconde. Le baud est normalisé par multiples et sous multiples de 9600, et donc on peut changer le baud suivant nos besoins.

2.2.5 Récapitulatif de l'architecture système

La figure 2.18 récapitule l'architecture du système hardware.

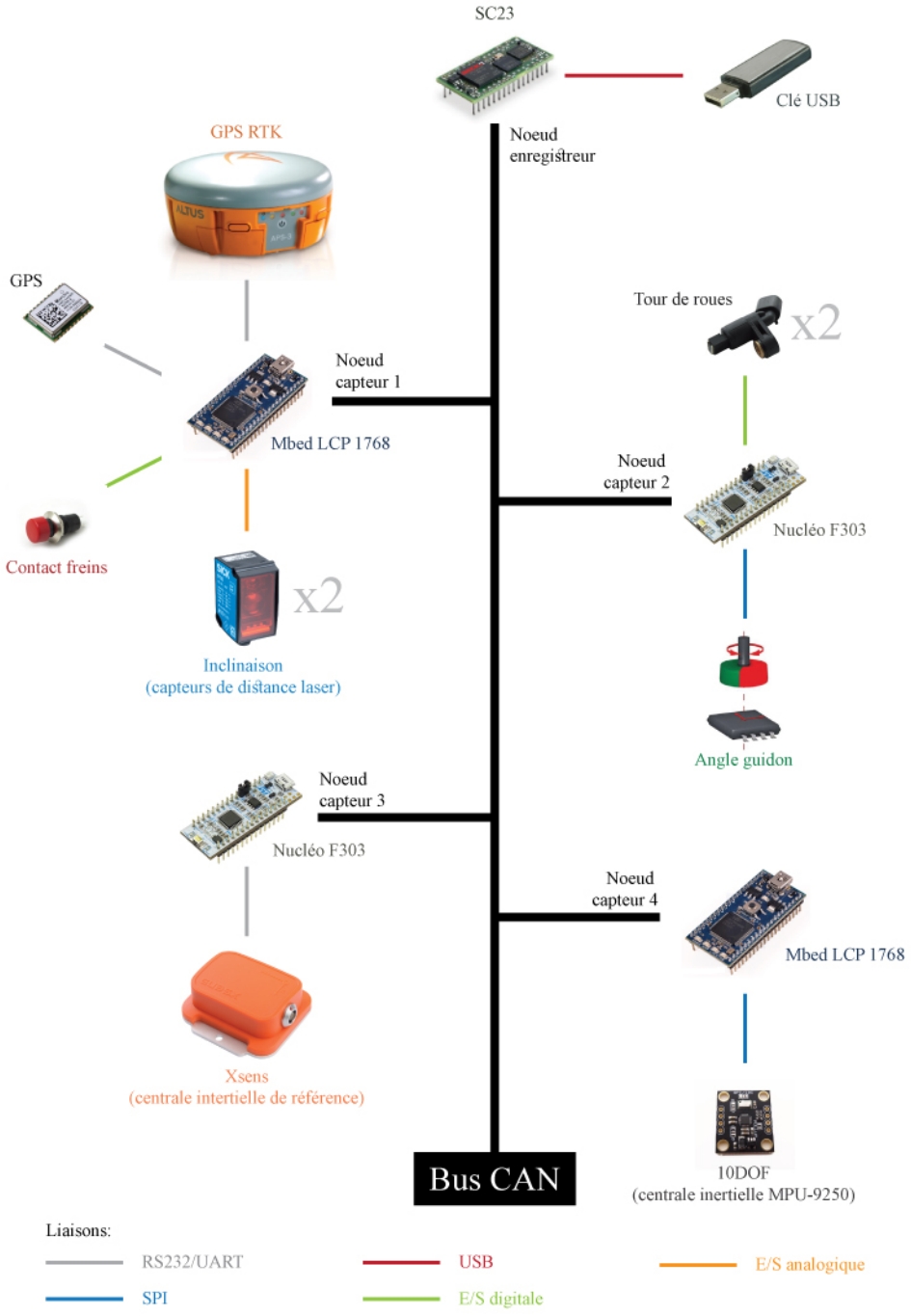


FIGURE 2.18: Récapitulatif de l'architecture système

2.3 Architecture software

Le système implémenté sur le véhicule deux roues est un système reparti du point de vue software. Ceci car chaque nœud implémente une application totalement indépendante des autres nœuds. La communication entre ces applications se fait à travers des trames de données suivant un format spécifique. La figure 2.19 résume l'architecture software du système en tenant compte du schéma dans la figure 2.18 :

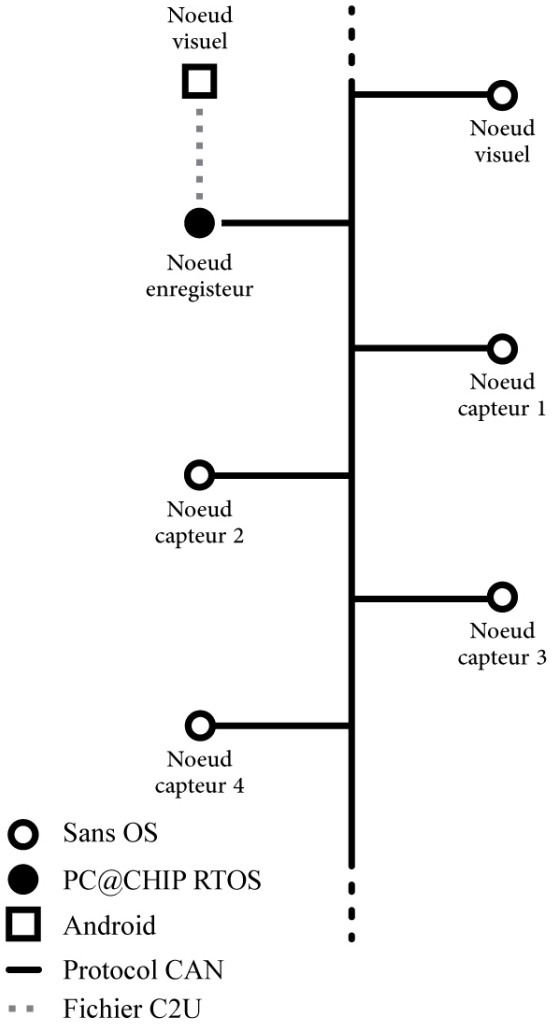


FIGURE 2.19: Architecture software générale du système

Les fichiers C2U contiennent des blocs de données ayant le format suivant :

2.3.0.1 Format des données C2U

Un message C2U donné est codé sur 16 octets selon le format de la figure 2.20.

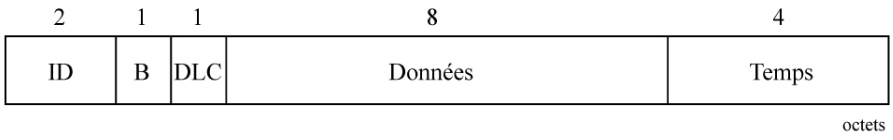


FIGURE 2.20: Format d'un bloc de données C2U

Tous les messages reçus, ou écrits dans le fichier C2U ont le même format décrit précédemment. La seule différence réside dans le format de la donnée sur les 8 octets qui change d'un capteur à un autre.

Format des données Accélération Chaque donnée est codée sur 2 octets en little Endian (2.3.0.1), représentant un entier signé qu'il faut diviser par 2 à la puissance 9 pour obtenir la valeur réelle de l'accélération. Ce format de données s'applique sur les deux capteurs 10DOF et XSENS, et donc la seule différence entre les deux est le ID qui est de 104 pour la 10DOF et 204 pour la XSENS.

GyroX 10.6 deg/s	GyroY 10.6 deg/s	GyroZ 10.6 deg/s	Bourrage
2 octets	2 octets	2 octets	2 octets

Format des données Gyroscope Chaque donnée est codée sur 2 octets en little Endian, représentant un entier signé qu'il faut diviser par 2 à la puissance 6 pour obtenir la valeur réelle du gyroscope. Ce format de données s'applique sur les deux capteurs 10DOF et XSENS, et donc la seule différence entre les deux est le ID qui est de 105 pour la 10DOF et 205 pour la XSENS.

GyroX 10.6 deg/s	GyroY 10.6 deg/s	GyroZ 10.6 deg/s	Bourrage
2 octets	2 octets	2 octets	2 octets

Format des données Magnétomètre Chaque donnée est codée sur 2 octets en little Endian, représentant un entier signé qu'il faut diviser par 2 à la puissance 8 pour obtenir la valeur réelle du magnétomètre. Ce format de données s'applique sur les deux capteurs 10DOF et XSENS, et donc la seule différence entre les deux est le ID qui est de 106 pour la 10DOF et 206 pour la XSENS.

MagX 8.8 µT	MagY 8.8 µT	MagZ 8.8 µT	Bourrage
2 octets	2 octets	2 octets	2 octets

Format des données des angles d'Euler Chaque donnée est codée sur 2 octets en little Endian, représentant un entier signé qu'il faut diviser par 2 à la puissance 7 pour obtenir la valeur réelle de l'angle. Ce format de données s'applique sur les deux capteurs 10DOF et XSENS, et donc la seule différence entre les deux est le ID qui est de 107 pour la 10DOF et 207 pour la XSENS.

EulerX 9.7 deg	EulerY 9.7 deg	EulerZ 9.7 deg	Bourrage
2 octets	2 octets	2 octets	2 octets

Format des données GPS La donnée GPS (longitude ou latitude) est codée sur 4 octets en little Endian, représentant un entier de 10 chiffres, où les trois premiers chiffres représentent le degré de la cordonnée et les Cinq autres représentent la minute. Et donc pour convertir du format degré minute au format décimal on effectue l'opération suivante :

1. On sépare les degrés du reste en divisant par 10 millions.
2. On sépare les minutes du reste en utilisant un modulo 10 millions.
3. On divise les minutes par 10 milles car c'est les deux premiers chiffres qui représentent la partie entière.
4. On divise le résultat obtenu par 60 pour convertir les minutes en degrés.
5. On somme les résultats de (1) et (4)

```
float latiF=lati/10000000+((float)(lati%10000000)/60.0f)/100000;
float longiF=longi/10000000+((float)(longi%10000000)/60.0f)/100000;
```

Longitude	Latitude
4 octets	4 octets

Définition de l'Endianness La représentation d'un nombre entier peut se faire sur plusieurs octets. L'ordre dans lequel on organise ces octets est appelé endianness.

Il existe deux formats d'organisation et qui sont big et little Endian :

- Big Endiant : prenons l'exemple d'un entier sur 32 bits 0xAB01CD02 (notation hexadécimale), ce format enregistre les octets dans l'ordre suivant : AB 01 CD 02.
- Little Endiant : reprenons le même exemple, ce format enregistre les octets dans l'ordre suivant : 02 CD 01 AB.

2.4 Conclusion

Dans ce chapitre l'architecture globale du système était abordée de manière générale. Cette architecture était élaborée par l'équipe VIROLO++ en se basant sur le modèle physique explicité dans le chapitre II, et répondant aux contraintes d'embarquabilité et d'extensibilité. Ce chapitre définit le contexte dans lequel les briques du chapitre suivant (notre contribution) sont développées.

Chapitre 3

Noeuds étudiés

3.1 Introduction

Dans le contexte du système conçu par l'équipe VIROLO++, nous avons été chargés de l'étude et le développement de deux noeuds :

- Noeud IMU : plus particulièrement l'exploitation du processeur interne de l'MPU-9250 et l'évaluation de la librairie fournie par le fabricant du capteur.
- Noeud visuel : Le dévéloppement d'une application sur plateforme mobile qui permet la visualisation des données capteurs.

Dans ce chapitre, nous détaillons l'étude et le développement de ces noeuds, les problèmes que nous avons rencontrés, et les démarches suivies pour les résoudre.

3.2 Nœud IMU

Il s'agit d'un nœud capteur faisant appel à une centrale inertielle (Inertial Measurement Unit).

3.2.1 Présentation d'une IMU

une IMU est un dispositif électronique qui mesure et donne les forces appliquées sur un corps physique, sa vitesse angulaire et le champ magnétique qui l'entoure.

une IMU détecte le taux d'accélération en utilisant un ou plusieurs accéléromètres et détecte les vitesses angulaires de rotation en utilisant un ou plusieurs gyroscopes, et parfois des magnétomètres internes afin d'assister la calibration et éliminer la dérive de l'orientation.

Certaines IMU contiennent un processeur interne DMP (digital motion processor).

Définition du DMP Le DMP est un processeur spécialisé dans le traitement des données de mouvements. Il reçoit les données à partir des accéléromètres, gyroscopes et magnétomètres et les traite. Le résultat du traitement est souvent une représentation de l'orientation du solide sur lequel est placée l'IMU.

L'objectif du DMP est de réduire la charge de calcul que subit le processeur hôte. Il réalise cela en faisant appel à des fonctions primitives et des accélérateurs matériels qui effectuent l'opération de fusion de données de façon plus efficaces.

L'IMU utilisée dans le système est une MPU-9250 pilotée par une carte Mbed LPC 1768.

3.2.2 Présentation du MPU-9250

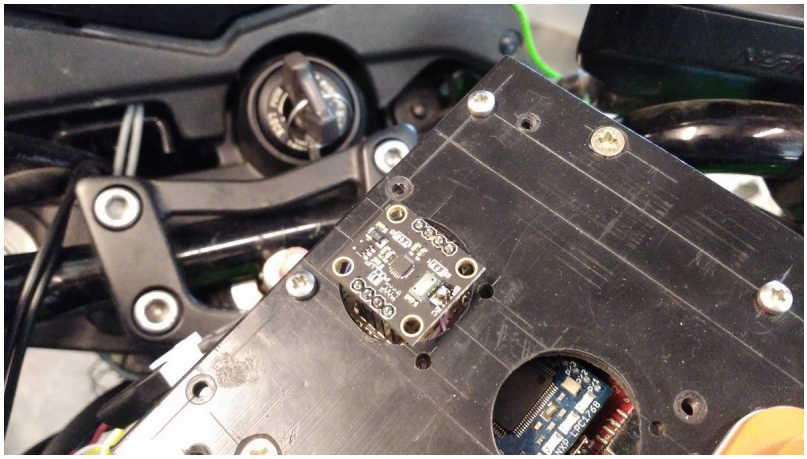


FIGURE 3.1: MPU-9250 placé sur la moto

La MPU-9250 est un module à plusieurs puces (MCM), fabriqué par Invensens et constitué de deux emplacements. Le premier emplacement abrite un accéléromètre et un gyroscope suivant les 3 axes, tandis que le deuxième emplacement contient le magnétomètre AK8963 suivant les 3 axes [15]. Par conséquent, le MPU-9250 est un dispositif de suivi de mouvement, qui combine 3 axes d'accéléromètres, de gyroscope et de magnétomètres ainsi qu'un processeur digital interne DMP (Digital Motion Processor), il contient aussi un capteur de température, faisant de lui un 10DOF (10 degrés de libertés). Le schéma interne du MPU-9250 est explicité dans la figure 3.2.

3.2.3 Les blocs clés du MPU-9250

Le module MPU-9250 comporte plusieurs blocs :

- Capteur gyroscope à 3 axes avec un convertisseur numérique analogique sur 16 bits.
- Capteur accéléromètre à 3 axes avec un convertisseur numérique analogique sur 16 bits.
- Capteur magnétomètre à 3 axes avec un convertisseur numérique analogique sur 16 bits.
- Conditionneur de signal : qui permet d'amplifier et de filtrer le signal de sortie du convertisseur analogique numérique afin de produire un voltage proportionnel

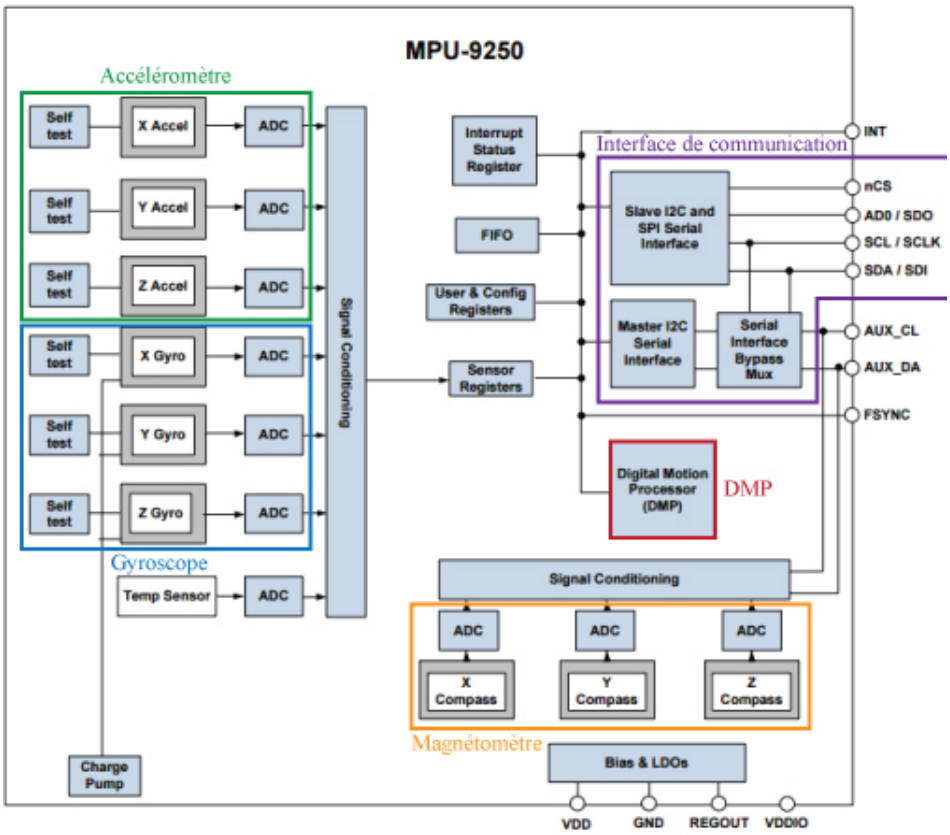


FIGURE 3.2: Schéma synoptique du MPU 9250 [15]

à la donnée en question (les étapes du processus ne sont pas accessibles car c'est une technologie propriétaire).

- Processeur digital de mouvement interne DMP : le DMP se trouve à l'intérieur du MPU-9250 qui réduit la charge de calcul qui subit le microcontrôleur. Le DMP acquiert les données des différents capteurs afin de les traiter et mettre le résultat dans la FIFO.
- Interface de communication série I2C et SPI primaire : le MPU-9250 peut communiquer avec un processeur externe en utilisant I2C ou SPI. Il joue toujours le rôle de l'esclave.
- Interface de communication série I2C auxiliaire : le MPU-9250 communique avec les autres capteurs via le bus I2C auxiliaire qui a deux modes de fonctionnement :
  - Mode Maitre : dans ce mode le MPU-9250 joue le rôle du maître par-rapport aux capteurs externes et donc le processeur externe communique avec le MPU-9250 qui lui transfert les données aux autres capteurs.
  - Mode Bypass : dans ce mode le MPU-9250 connecte le bus I2C primaire et auxiliaire permettant au processeur externe de communiquer directement avec les autres capteurs.
- Self-test : est un mécanisme qui teste la validité des données des capteurs accéléromètre et gyroscope en les comparant avec des valeurs limites acceptées.

- Horloge : permettant un fonctionnement synchrone des capteurs.
- Registres de données capteurs : c'est des registres de données contenant les dernières valeurs données par les différents capteurs. C'est des registres a lecture seule qui sont accessible via l'interface série de communication. Les données peuvent être lues à tout moment à partir de ces registres.
- FIFO : le MPU-9250 à une FIFO de 512 bytes. Le registre de configuration FIFO spécifie quel type de données on peut écrire. Il est possible de choisir d'inclure les données des différents capteurs (accéléromètre, gyroscope, magnétomètre, capteurs externes ... etc.).
- Interrupts : la fonctionnalité d'interruption est configurée. Les évènements pouvant provoqués une interruption sont :
  - Générateur d'horloge.
  - De nouvelles données disponibles au niveau du FIFO
  - Interruption générée par l'accéléromètre.
  - Le MPU-9250 ne reçoit pas d'accusé de réception d'un capteur auxiliaire.
- Sortie numérique du capteur de température
- Bias and LDO : ce bloc génère l'alimentation interne et le voltage de référence ainsi que le courant électrique nécessaire à l'MPU-9250.
- Charge Pump : Convertisseur à pompe de charge sur puce qui génère le voltage nécessaire pour le fonctionnement des oscillateurs internes.

3.2.4 Interfaces de communication entre MPU9250 et Mbed

L'échange de données peut se faire à l'aide de l'une des deux interfaces de communications série supportées par MPU-9250 à savoir I2C et SPI.

Pour choisir un des deux modes, il suffit de souder la pin de sélection du mode et de souder/dessouder la pull-up et de faire le branchement adéquat (voir figure 3.3).

On utilise l'interface SPI dans ce noeud car la vitesse de communication SPI est supérieure à celle du I2C.

La communication entre le Mbed et l'MPU-9250 se fait par lectures / écritures directes dans les registres interne de l'MPU-9250.

3.2.4.1 Initialisation de l'MPU-9250 :

Il faut d'abord initialiser la communication SPI sur Mbed en spécifiant la fréquence et le format des transferts :

```
spi.format(8,0);
spi.frequency(1000000);
```

Pour initialiser le MPU-9250, on doit :

- Réinitialiser la puce en software (écriture dans un registre)
- Choisir une source d'horloge (interne ou externe)
- Activer les capteurs accéléromètre et gyroscope
- Définir la fréquence de coupure du filtre passe-bas des deux capteurs

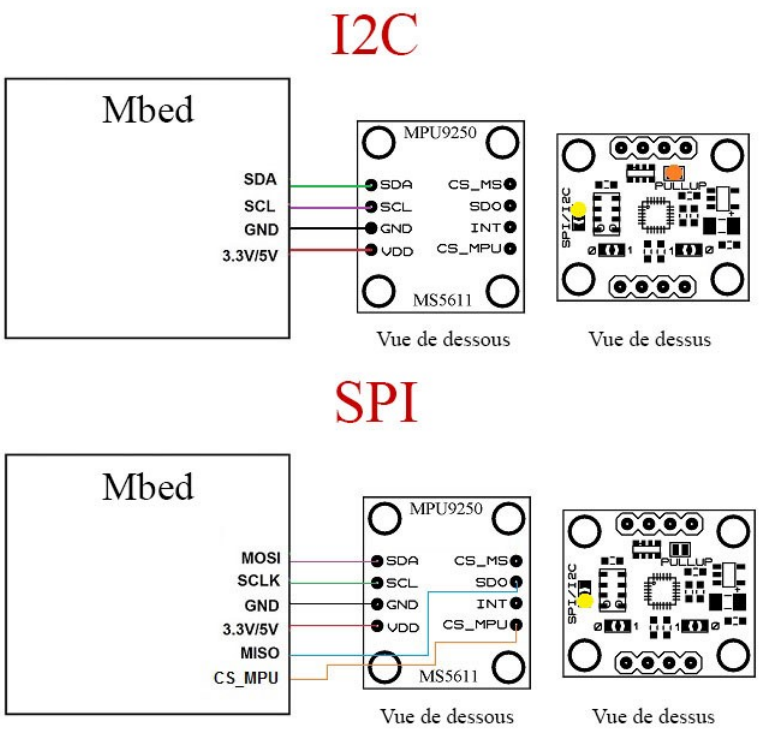


FIGURE 3.3: Interfaces de communication entre MPU9250 et Mbed [16]

- Définir la dynamique du gyroscope (de 200°/s à 2000°/s)
- Définir la dynamique de l'accéléromètre (de 2g à 16g)
- Spécifier la fréquence d'échantillonnage des capteurs.

3.2.4.2 Initialisation du magnétomètre (AK8963) :

Le magnétomètre étant un capteur auxiliaire, on doit d'abord configurer le maître I2C pour le piloter. Il faut :

- Ecrire l'adresse I2C du magnétomètre dans le registre d'adresse de l'esclave 0
- Définir le registre interne du magnétomètre sur lequel on veut effectuer le transfert
- Spécifier le type et le format de transfert (lecture/écriture, nombre d'octets)
- Mettre la donnée à écrire (dans le cas d'écriture) dans le registre de donnée de l'esclave 0

En suivant les étapes décrites, on :

- Réinitialise le magnétomètre
- Définit le mode de mesure et la sensibilité (Continu/programmé, 14 ou 16 bits)

On temporise entre chaque opération pour donner au maître I2C le temps d'exécuter les transferts configurés.

3.2.4.3 Récupération des données Accéléromètre et Gyroscope :

Les données provenant de l'accéléromètre et du gyroscope sont automatiquement écrites dans des registres de sorties à chaque période d'échantillonnage. Chaque valeur est codée sur deux registres d'un octet. Le premier contenant l'octet le plus fort et le deuxième l'octet le moins fort, qui, ensemble, représentent un entier signé qu'on doit multiplier par la sensibilité du capteur (Combien d'unités de la grandeur mesurée représente un bit) pour obtenir la valeur réelle de la grandeur concernée [17].

Exemple : Accélération sur l'axe X.

On lit 2 registres à partir de l"adresse 0x3B

(ACCEL-XOUT-H et ACCEL-XOUT-L)

```
uint8_t data_bits[2];
ReadRegs(MPUREG_ACCEL_XOUT_H,data_bits,2);
```

On récupère l'entier signé représenté par les deux octets :

```
short  accelX;
accelX=((short)data_bits[0]<<8)|data_bits[1];
```

On multiplie l'entier obtenu par la sensibilité pour avoir la valeur réelle :

```
float accelXval;
accelXval=accelX*accelSens;
```

On prend, comme exemple, une dynamique de 2g (où g, l'accélération gravitationnelle est 9.81 m^2/s), la sensibilité se calcule comme suit :

- On sait que la valeur maximale de l'accélération est 2g
- La valeur maximale que peut représenter un short est 32768.

1 bit représente alors (2/32768) g. La sensibilité est :

```
accelSens=2.0f/32768;
```

3.2.4.4 Récupération des données Magnétomètre :

On configure le maître I2C pour une opération de lecture en mettant l'adresse dans le registre d'adresse de l'esclave I2C 0 et spécifiant l'adresse de début de lecture (AK8963-HXL) et le nombre d'octets à lire (6 octets).

```
WriteReg(MPUREG_I2C_SLV0_ADDR,AK8963_I2C_ADDR|READ_FLAG);
WriteReg(MPUREG_I2C_SLV0_REG, AK8963_HXL);
WriteReg(MPUREG_I2C_SLV0_CTRL, 0x86);
```

On attend que le maître I2C exécute l'opération de lecture puis on lit les registres de l'MPU qui reçoivent automatiquement les données lises à partir d'un capteur externe (comme le magnétomètre).

```
wait(0.001);
ReadRegs(MPUREG_EXT_SENS_DATA_00,response,6);
```

Les 6 octets contiennent 3 entiers signés (chacun codé sur 2 octets en Big Endian) représentant les valeurs du champ magnétique sur les 3 axes à une constante près. Cette constante est la sensibilité et se calcule suivant la formule [17] :

$$\frac{(ASA-128)\times 0.5}{128}+1$$

Où ASA est la valeur d'ajustement de sensibilité écrite dans des registres internes de l'AK8963 lors de sa production. ASA doit être récupérée avant la première lecture des données magnétiques.

3.2.5 Fusion de données capteurs (Les angles d'Euler)

3.2.5.1 Définition de la fusion de données capteurs

la fusion de données est un processus d'intégration de données obtenu généralement de sources différentes. Les capteurs d'où proviennent les données peuvent ne pas être identiques. Les données fusionnées ont une meilleure précision que celles issues des capteurs de façon indépendante.

On s'intéresse, dans notre cas, à tirer les angles d'Euler à partir des données accéléromètres, gyroscopes et magnétomètre provenant de la centrale inertielle (IMU).

Les angles d'Euler Les angles d'Euler sont un ensemble de 3 angles (roll, pitch, yaw) décrivant l'orientation d'un solide par rapport à un repère en trois dimensions (figure 3.4).

Les trois angles nous renseignent sur l'état de la moto, notamment son inclinaison et sa direction.

Pour obtenir les angles d'Euler, deux méthodes s'offrent à nous :

3.2.5.2 Utilisation d'un algorithme de fusion exécuté sur le microcontrôleur hôte Mbed (sans DMP) :

Généralement, un algorithme de fusion de données inertielles (accéléromètre, gyroscope et magnétomètre) donne en sortie la direction du solide dans la forme de quaternions. Les angles d'Euler peuvent être donc calculés à partir des quaternions (figure 3.4).

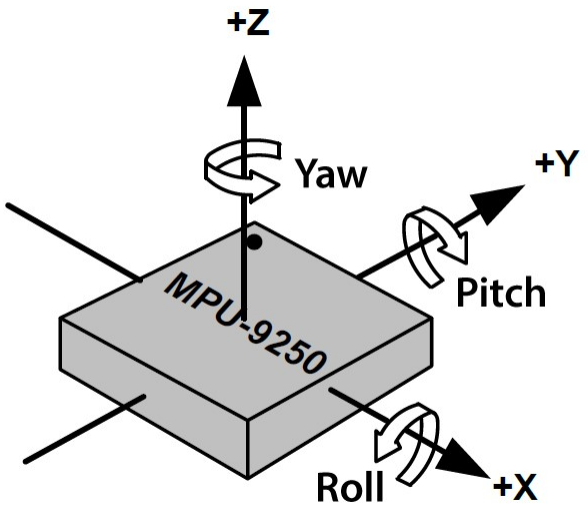


FIGURE 3.4: Representation des angles d'Euler

Définition des quaternions Les quaternions sont une représentation mathématique vectorielle à 4 dimensions présentant l'orientation et la rotation d'un solide par rapport à un repère à trois dimensions. C'est plus facile de les composer à partir des données inertielles par rapport aux autres représentations.

Il existe plusieurs algorithmes de fusion de données permettant le calcul des quaternions. L'objectif est d'utiliser un algorithme existant, performant et non-volumineux. Le candidat remplissant ces critères est l'algorithme de Madgwick.

Algorithme de Madgwick L'algorithme de Madgwick est un filtre élaboré pour supporter des systèmes portables de suivi de mouvements, et donc des systèmes à capacité de calcul limitée. Cet algorithme est applicable aux centrales inertielles comportant des capteurs d'accélération, de vitesse angulaire (gyroscope) et de champ magnétique. L'algorithme utilise une représentation en quaternions se basant sur une méthode du gradient descendant optimisée. L'algorithme permet une précision comparable à celles des algorithmes d'orientation basés sur le filtre de Kalman tout en nécessitant moins de ressources, en termes de capacité de calcul et de fréquence d'échantillonnage de données [18].

Orientation estimée à partir des gyroscopes Les gyroscopes mesurent les vitesses angulaires autour des axes x, y et z. L'orientation de l'IMU par rapport a la terre peut etre calculée par une integration numerique de la dérivée des quaternions qui est égale aux vitesses angulaires.

On note l'orientation estimée à partir des gyroscope par q_g.

Orientation estimée à partir des accéléromètres et des magnétomètres Les accéléromètres mesurent l'amplitude et la direction du champ gravitationnel et les

accélérations linéaires, dues au mouvement de l'IMU. De même, les magnétomètres mesure l'amplitude et la direction du champ magnétique terrestre ainsi que les fluctuations et distorsions magnétiques. Dans le contexte du filtre d'orientation, on suppose, pour un premier temps, que les accéléromètres ne mesurent que le champs gravitationnel de la terre et que les magnétomètres ne mesurent que le champ magnétique terrestre.

Si la direction du champ terrestre est connue dans un repère fixe, la mesure de la direction du champ par rapport au repère de l'IMU permet la déduction de l'orientation du repère de l'IMU par rapport au repère fixe. Cependant, l'orientation déduite n'est pas une solution unique. Toute orientation autour d'un axe parallèle à la direction du champ peut être une solution.

Une représentation quaternions nécessite qu'une solution unique soit trouvée.

Pour ce faire, les deux directions des champs gravitationnel et magnétique terrestres sont utilisées pour calculer l'orientation de l'IMU par rapport à la terre. La combinaison des deux directions permet de déduire une solution d'orientation unique.

on note l'orientation estimée à partir des accéléromètres et magnétomètres par q_a.

L'algorithme de fusion : Une estimation d'orientation de l'IMU est obtenue à travers la fusion des deux orientations calculées auparavant. La fusion est décrite par la formule suivante où q_e est l'orientation estimée [18] :

$$q_e = \gamma q_g + (1 - \gamma) q_a, (0 < \gamma < 1)$$

γ représente le coefficient de pondération qui est proportionnel au taux de changement d'orientation (voir le récapitulatif de l'algorithme figure 3.5).

Pour implémenter l'algorithme de Madgwick dans le programme, on inclut la fonction C correspondante (Disponible en ligne en Open Source) et on lui donne en entrée les 9 données capteurs (Accélération, vitesse angulaire et champ magnétique sur les 3 axes).

```
mpu9250.MadgwickQuaternionUpdate(ax, ay, az,
gx*PI/180.0f, gy*PI/180.0f, gz*PI/180.0f,  my,  mx, mz);
```

Les vitesses angulaires étant données en degrés/seconde doivent être converties en radians/seconde.

La fonction MagdwickQuaternionUpdate affecte les valeurs des quaternions calculées à une variable globale sous forme de tableau q[4].

Pour une interprétation plus simple de l'orientation, on convertit les quaternions en angles d'Euler en utilisant des relations suivantes :

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}(2(q_0 q_1 + q_2 q_3), 1 - 2(q_1^2 + q_2^2)) \\ \arcsin(2(q_0 q_2 - q_3 q_1)) \\ \text{atan2}(2(q_0 q_3 + q_1 q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix}$$

Où ϕ est le roll, θ le pitch et ψ le yaw.

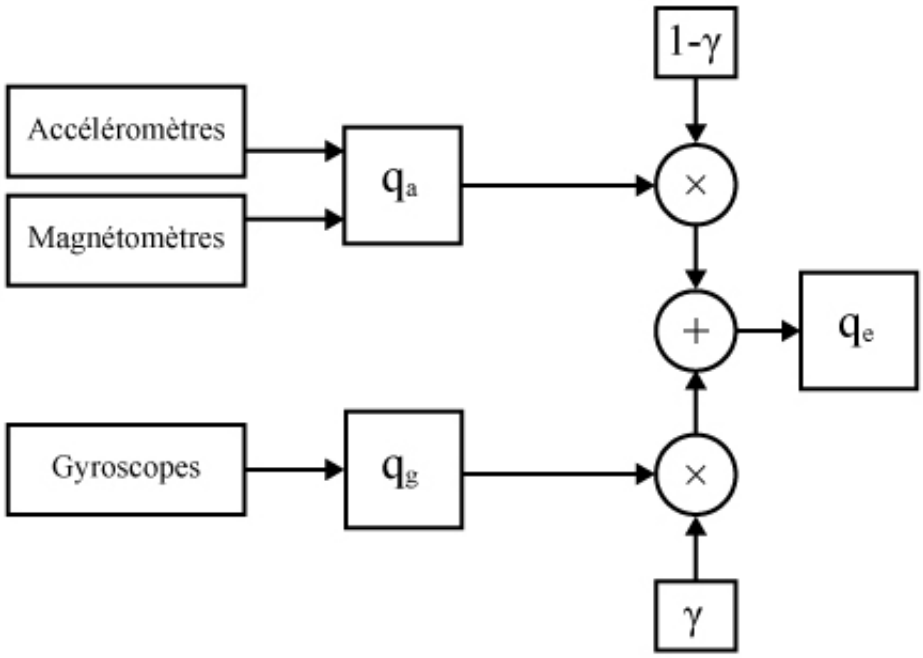


FIGURE 3.5: Récapitulatif de l'algorithme de Madgwick

On convertit les angles obtenus en degrés et on corrige la déclinaison magnétique terrestre (l'angle entre le nord géographique et le nord magnétique).

```
pitch *= 180.0f / PI;
yaw   *= 180.0f / PI;
yaw   -= 0.33f; //Déclinaison magnétique à Paris
roll  *= 180.0f / PI;
```

L'utilisation des algorithmes de fusion est appuyée par une calibration faite par l'équipe VIROLO++ et qui vise à comparer les valeurs obtenues avec celles de l'approche utilisant le processeur interne. Les résultats de cette approche sont exposés dans le chapitre suivant.

3.2.5.3 Utilisation du processeur interne de MPU-9250 (DMP) :

Une autre alternative pour faire la fusion de données est d'utiliser le DMP (Digital Motion Processor), présent sur la centrale inertielle MPU-9250. Le DMP utilise pour la fusion de données, les accélérations et les vitesses angulaires sur les 3 axes et génère les quaternions en interne sans faire appel aux ressources du microcontrôleur. Ils sont donc des quaternions à 6 degrés de libertés. Pour obtenir les quaternions à 9 degrés de libertés, Invensens fournit une fonction pour la fusion à 9 degrés de libertés qui prend en entrée les quaternion 6 DoF (Degrees of Freedom) et les données du capteur magnétique sur les 3 axes.

Les angles d'Euler sont obtenus de la même manière à partir des quaternions 9 DoF.

Problématique d'exploitation du DMP : Notre travail sur l'IMU avait comme objectif principal la mise en route du DMP sur une carte Mbed car ces cartes faisaient déja partie du système et sont facilement embarquables (de petite taille). Cette approche est très peu documentée.

La seule solution disponible pour l'exploitation du DMP est La librairie Motion Driver fournie par Invensens, le fabricant de l'MPU-9250. Cependant, aucune explication détaillée n'est disponible pour cette librairie. Donc, on a dû explorer d'autres pistes pour comprendre son fonctionnement et son utilisation.

Remarque : La fonction de fusion 9 DoF fait partie de la librairie "Motion Driver 6.12". Cette librairie est précompilée (Code source inaccessible) et développée par Invensens (Fabricant de l'MPU-9250).

Les algorithmes de fusions utilisés dans le DMP et la fonction de fusion 9 DoF sont propriétaires à Invensens et donc non-accessibles.

Mise en route du DMP : Invensens fourni avec la bibliothèque un exemple de code utilisant le DMP avec la carte STM32F404V, qui est une carte ST non-supportée par la plateforme Mbed.

La communication entre la carte et l'MPU-9250, dans cet exemple se fait en I2C.

FonctionnementFonctionnement du code : L'exécution du code peut être résumée dans les points suivants :

- Initialisation de la carte (STM23F404) et activation des interruptions
- Initialisation de l'MPU-9250 :
  - Réinitialisation de la puce
  - Configurer la dynamique du gyroscope à 2000 degrés/seconde
  - Configurer la dynamique de l'accéléromètre à 2g (g=9.81 m^2/s)
  - Configurer le filtre passe bas à 42 Hz
  - Réinitialiser la FIFO
  - Configurer le magnétomètre :
    - Activer le mode bypass
    - Chercher l'adresse de l'AK8963
    - Récupérer les ajustements de sensibilité
    - Activer le mode maître I2C (désactiver le mode bypass)
    - Configurer l'esclave I2C 0 pour lire à partir des registres de données
    - Configurer l'esclave I2C 1 pour écrire dans les registres de configurations
    - Définir la fréquence d'activation des actions des deux esclaves (fréquence d'échantillonnage)
- Activer les quaternions 6 DoF
- Activer la fusion de données 9 DoF

- Activer la calibration automatique du magnétomètre (Elle sera appliquée dès qu'un nombre suffisant de données magnétiques est recueilli)
- Activer la calibration du gyroscope (Quand aucun mouvement n'est détecté)
- Réveiller tous les capteurs
- Définir la fréquence d'échantillonnage de la FIFO à 100Hz
- Vérifier toutes les configurations (Fréquence d'échantillonnage, dynamiques des capteurs)
- Charger le Firmware sur le DMP (Programmer le processeur)
- Activer le DMP
- Exécuter dans une boucle infinie les instructions suivantes à chaque interruption générée par MPU-9250 :
  - Lire la FIFO
  - Mettre à jour la structure contenant les données capteurs
  - Appliquer les calibrations sur les données capteurs et générer les quaternion 9 DoF à partir des quaternion 6 DoF et des données magnétiques.
  - Formater les angles d'Euler et les envoyer sur le bus CAN
  - Formater les accélérations et les envoyer sur le bus CAN
  - Formater les vitesses angulaires et les envoyer sur le bus CAN
  - Formater les données magnétiques et les envoyer sur le bus CAN

Pour porter ce code sur une carte Mbed, on a dû passer par deux étapes :

1. Migration vers la carte LPC 1768 en I2C :

   Pour migrer vers la carte Mbed il fallait :
   - Inclure la bibliothèque "mbed.h"
   - Réassigner les pins I2C
   - Redéfinir les fonctions i2c write et i2c read en utilisant les fonction writeBytes et readBytes qui font appel à i2c.write et i2c.read

```
#define i2c_write   writeBytes
#define i2c_read    readBytes
```

   Remarque : Les fonctions i2c.read, i2c.write attendent que l'adresse de l'esclave I2C soit codée sur les 7 MSB (Most Significant Bits) de l'octet passé. On doit donc décaler l'octet contenant l'adresse une fois vers la gauche.
   - Utiliser le mode polling à la place du mode interruptible (Attacher la lecture des nouvelles données à une interruption temporelle (Ticker))
   - Supprimer les fichiers et les fonctions inutiles (Les fichiers et fonctions spécifiques à STM32F404)
2. Migration vers le mode SPI sur la carte LPC 1768 :

   Les modifications effectuées :
   - Assigner les pins SPI et configurer la vitesse et le format de communication
   - Réécrire les fonctions de lectures et écritures dans les registres en SPI

Ecriture en SPI : Pour écrire dans un registre MPU-9250 à travers le protocole SPI, il faut d'abord sélectionner la puce MPU-9250 en forçant la pin CS-MPU à 0 (Chip Select).

Pour avoir accès au registre adressé par "WriteAddr", on envoie la valeur de "WriteAddr" sur la pin MOSI en utilisant la fonction spi : :write faisant partie de la librairie <mbed.h>. "WriteAddr" est codée sur 7 bits, le 8ème bit (le plus fort) est égale à 0 qui signifie que la transaction est une écriture :

```
spi.write(WriteAddr);
```

On envoie la donnée à écrire sur le pin MISO en utilisant la même fonction :

```
for(i=0; i<Bytes; i++)
     spi.write(WriteData[i]);
```

On désélectionne l'MPU-9250 (CS à 1) et on temporise pendant 50 microsecondes pour permettre aux signaux de se stabiliser.

Lecture en SPI : La lecture en SPI se fait de la même manière que l'écriture sauf que le 8ème bit est mis à 1 pour dire que la transaction est une opération de lecture.

```
spi.write(ReadAddr|0x80);
```

La donnée du registre adressé est la valeur retournée par l'écriture d'un octet fictif (0x00 par exemple) "Bytes" fois, où "Bytes" est le nombre d'octets à lire à partir de l'adresse spécifiée.

```
for(i=0; i<Bytes; i++)
    ReadBuf[i] = spi.write(0x00);
```

On désélectionne l'MPU-9250 (CS à 1) et on temporise pendant 50 microsecondes.

Remarque : Dans le cas d'une écriture ou lecture SPI, il suffit de spécifier l'adresse du début uniquement car les transferts SPI consécutifs subissent une incrémentation automatique d'adresse.

- Réécrire les configurations du magnétomètre (faites en mode bypass) en mode maître I2C. Pour faire cela, il faut passer par le maître I2C pour configurer le magnétomètre et récupérer les ajustements magnétiques.

3.3 Nœud Visuel

Pour visualiser les données capteurs sauvegardées par l'enregistreur, nous avons développé une application de visualisation répondant aux critères suivants :

3.3.1 Cahier de charge

- Application portable.
- Multiplateformes (En tenant compte du temps de développement).
- Lire les fichiers C2U enregistrés sur la clé USB.
- Afficher la trajectoire sur une carte.
- Afficher les données accéléromètres, gyroscopes, magnétomètre et Euler de la 10DOF et la XSens pour une comparaison qualitative.

3.3.2 Choix de l'outil de développement

Le critère multiplateforme nous induit à opter pour un outil de développement multiplateforme et rapide. RAD Studio était un premier choix grâce au fait qu'il offre une période d'essai contrairement aux autres outils. De plus, Il est l'un des rares outils qui génèrent des applications natives pour chaque environnement.

3.3.2.1 RAD Studio

RAD studio [19] est un IDE propriétaire Embarcadero, qui permet le développement rapide d'application, grâce à sa caractéristique de multi-ciblage. Il permet de développer en haut niveau puis cibler les différentes plateformes à savoir Windows, Linux, IOS et Android, et donc développer une seule fois l'application désirée qui sera par la suite multiplateforme. Basé essentiellement sur le C++ Builder et Delphi, RAD studio permet aux développeurs de pouvoir choisir l'un des deux langages de programmation et de construire son projet en faisant appel aux différentes bibliothèques existantes dans le langage et d'utiliser d'autre bibliothèques incluses dans l'IDE.

L'application développée sur RAD Studio : En premier lieu, l'application développée devait lire les fichiers binaires de format C2U et afficher la trajectoire empruntée par la moto sur une carte.

Pour ce faire, le développement de l'application s'est divisé en deux parties :

Partie fonctionnelle Dans cette partie, une fonction décrite en C++ permet de lire le fichier C2U, d'extraire les données GPS et de les dans un tableau de points.

Interface graphique GUI L'outil graphique fondamental utilisé dans l'application est le TMapView, un outil permettant d'afficher et d'interagir avec la carte en utilisant différentes API pour différentes plateformes. Dans le cas d'Android, l'API utilisée est Google Maps.

La figure 3.6 donne un aperçu du résultat d'exécution de l'application.

FIGURE 3.6: Aperçu de l'application développée sur RAD Studio

3.3.2.2 Limitations et problèmes rencontrés durant le développement :

Après avoir validé une des fonctionnalités de base, on s'est aperçu que le module TMapView n'est pris en charge que par les deux plateformes Android et iOS, ce qui vient à éliminer la portabilité sur Windows.

Une autre fonctionnalité essentielle à intégrer est l'explorateur de fichiers qui sert à afficher, renommer, supprimer et sélectionner les fichiers pour visualiser leur contenu (les données des différents capteurs). L'outil Graphique permettant une telle opération est TOpenDialog. Cependant, cet outil n'est pris en charge que par Windows, alors que pour les autres plateformes un développement de son propre module est nécessaire.

Conclusion : RAD studio peut être un outil intéressant pour le développement de certains types d'applications, notamment celles qui font appel à des algorithmes de traitement et gestion de données, et à des communications web avec un serveur, sans pour autant se baser sur une GUI assez-compliquée.

Le fait que l'outil comporte des modules (essentiels à notre application) non-portables sur les différentes plateformes et une documentation limitée avec une communauté res-

treinte, nous a induit à reconsidérer le choix de l'outil qui dans ces circonstances ralentira le développement sans apporter la caractéristique multiplateforme. Le besoin d'avoir un prototype fonctionnel dans un délai réduit nous a poussé à migrer vers une autre solution même si cela élimine le critère multiplateforme. Cette solution devra être un moyen de développement rapide, tout en offrant les outils et les bibliothèques nécessaires pour implémenter les fonctionnalités décrites dans le cahier de charges.

La communauté Android active et étendue (problèmes et solutions), la documentation détaillée de la plateforme et le fait d'être déjà familiarisé avec, nous a induit à opter pour le développement de notre premier prototype d'application sous Android.

3.3.2.3 Présentation d'Android :

Android [20] est un système d'exploitation édité par Google pour appareils embarqués et/ou mobiles, comme les smartphones ou les tablettes. On le retrouve aussi dans certains GPS, ordinateurs de bord de voitures, dans des télévisions, autoradios, et même des montres.

Android Studio

Android Studio est l'environnement de développement officiel d'applications Android. Il propose, entre autres, des outils pour gérer le développement et visualiser l'application résultante sur des écrans de tailles différentes.

Le développement d'application Android se fait en deux parties. Une partie fonctionnelle qui se base sur le langage de programmation Java.

La deuxième partie gère les resources de l'applications (éléments graphiques, permissions, thème, style, ...). Elle se base sur le langage de balisage XML.

3.3.3 Architecture de l'application développée

L'application comporte trois parties essentielles qui sont :

1. Gestion de fichiers,

2. lecture de données,
3. visualisation de données.

Un récapitulatif de la l'architecture de l'application est illustré par la figure 3.7.

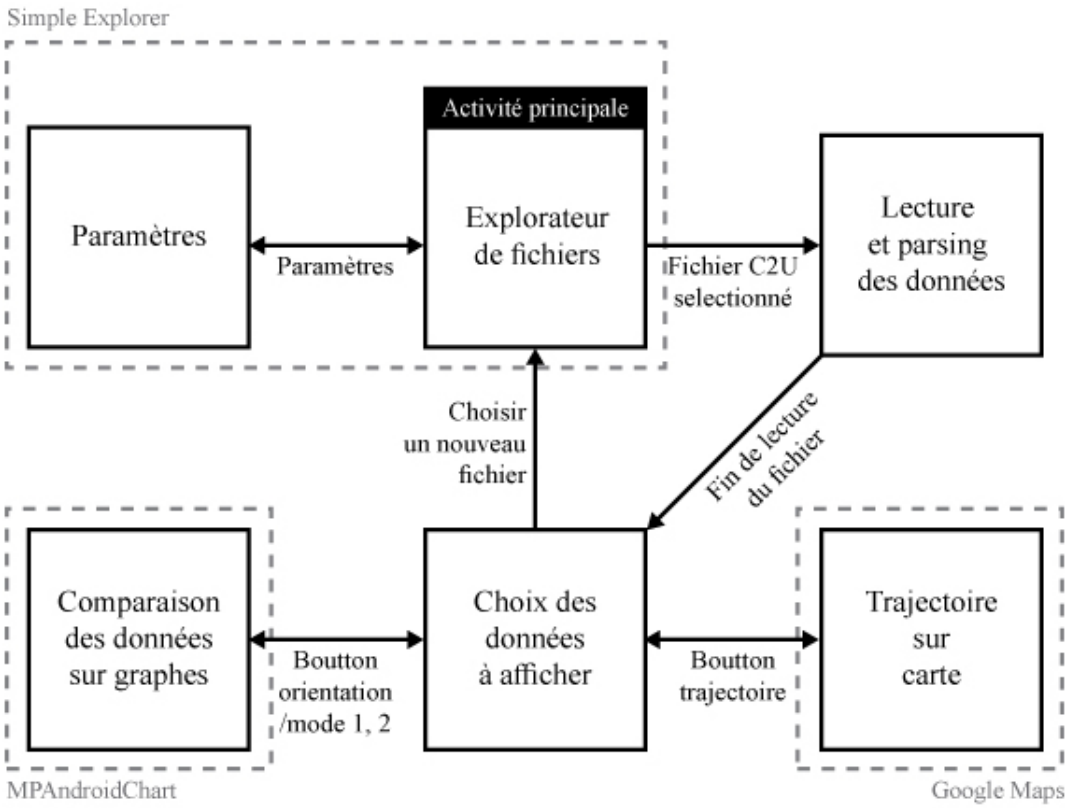


FIGURE 3.7: Architecture de l'application

3.3.3.1 Gestion de fichiers :

La première partie est la gestion de fichier, à savoir lire, modifier, renommer ou supprimer un fichier. L'utilisateur peut choisir un fichier stocké dans la mémoire interne, ou utiliser un câble USB OTG (On The Go) pour lire un fichier d'une clé USB après l'avoir enregistré sur cette dernière via la datalogger. Cette deuxième possibilité est pratique dans le cas où on peut directement visualiser les données après avoir effectué une manipulation et donc avoir une idée sur le format des données, et leurs cohérences.

Pour faire une telle gestion de fichiers, deux permissions au niveau de AndroidManifest sont nécessaires :

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Cela permet la lecture et l'écriture dans la mémoire.

Cette partie fait appel à un programme de base "Simple Explorer" [21] qui est un simple explorer de fichiers (open source) et qui fut par la suite modifié, adapté et intégré à l'application. Ayant un usage spécifique, on a fait en sorte de filtrer tous les fichiers sauf ceux de format C2U.(figure 3.8)

```
if (file.exists() && file.canRead()) {
    String[] list = file.list();

    // add files/folder to ArrayList depending on hidden status
    for (String aList : list) {
        File mFile= new File(path + "/" + aList);
        if(aList.endsWith(".c2u") || aList.endsWith(".C2U") || mFile.isDirectory())
        {
        if (!showhidden) {
            if (aList.charAt(0) != '.')
                mDirContent.add(path + "/" + aList);
        } else  {
            mDirContent.add(path + "/" + aList);
        }
        }
```

FIGURE 3.8: Filtrage de fichier (SimpleUtils.java).

Il existe, dans le projet de base, une fonction au niveau de la classe AbstractBrowserFragment.java sous forme de listner qui récupère l'évènement clique sur la ListView (liste contenant les différents fichiers et dossiers) et continuer la navigation si on clique sur un dossier alors qu'un clique sur un fichier affiche le répertoire absolu de ce dernier.

Une modification fut apportée à cette partie du code, où une variable globale de type chaîne de caractère est déclarée à qui on affecte le répertoire absolu du fichier au lieu de l'afficher et transmettre ainsi cette chaîne de caractère a la partie 2 de l'application.(figure 3.9)

```
mListView.setOnItemClickListener((parent, view, position, id) → {
        final File file = new File((mListView.getAdapter()
                .getItem(position)).toString());

        if (file.isDirectory()) {
            navigateTo(file.getAbsolutePath());

            // go to the top of the ListView
            mListView.setSelection(0);
        } else {


            MainActivity.path_str=file.getAbsolutePath().toString();
            Intent i=new Intent(getActivity(), MainActivity.class);
            startActivity(i);

        }
});
```

FIGURE 3.9: le programme permettant la navigation

Le résultat de la première partie est illustré par la figure 3.10.

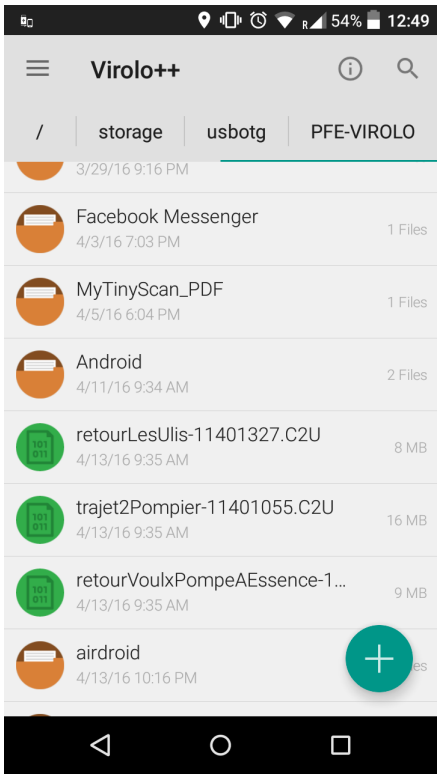


FIGURE 3.10: Interface graphique de la gestion de fichiers

3.3.3.2 Lecture de données

Dans cette partie, on fait appel à une seule classe 'CanMessageReader.java'. Cette classe reçoit le répertoire absolu et ouvre le fichier correspondant.

La classe contient des ArrayLists de type Entry (introduites et utilisées dans la troisième partie) qui a leur tours contiendront les données des différents capteurs, et deux fonctions permettant le passage d'un nombre binaire a un nombre entier. Ces deux fonctions sont les suivantes :

```
short binaryToInt16(byte[] tab,int i)
{
    return (short)((tab[i]& 0xFF) | (tab[i+1]<<8));
}
int binaryToInt32(byte[] tab, int i)
{
    return (int)(tab[i]&0xFF) | (int)(tab[i+1]&0xFF)<<8 | (int)(tab[i+2]&0xFF)<<16 |
}          (int)(tab[i+3]&0xFF)<<24;
```

La premier fonction prend un nombre binaire codé sur 16 bits et donne son équivalent en short, raison pour laquelle on voit que le deuxième nombre et décalé de 8 (qui représentera les bits les plus significatifs), et il y'a un ou logique entre les deux nombre binaire ce qui revient à mettre en 16 bits deux nombre de 8 bits, suivit d'un cast pour changer de type et pouvoir faire les différentes opérations de lectures (qui vont être explicitées dans ce qui suit).

Remarque : Dans la première fonction on remarque l'opération tab[i] & 0xFF qui peut sembler inutile. Cependant sans cette dernière l'application ne fonctionnerait pas correctement à coup sûr. Effectivement en Java une variable de type byte est considérée comme étant un nombre négatif codé en complément à 2, et donc le casté en short va ajouter des 1 sur les bits qu'on ne prend pas en considération alors qu'une constante est considérée comme étant positive.

Si on prend le cas d'une variable de type byte a=0xFE et qu'on caste en short le résultat donne 0xFFFE tandis que le caste de la constante 0xFF donne 0x00FF et donc l'opération ET logique permette de masquer les bits qui sont pas pris en considération vu que le premier byte représente les bits les moins signifiants.

La deuxième fonction suit exactement le même principe sauf que cette dernière transforme un nombre binaire codé en 32 bits en un entier (int).

Le fichier C2U est un fichier binaire contenant des trames CAN de 16 bytes, sa lecture se fait en suivant les étapes suivantes :

- Lire les deux premiers bytes qui correspondent à l'id de la trame en utilisant la fonction 'binaryToInt16'
- Grace à un switch case, on trouve quelle donnée contient la trame CAN
  Exemple de lecture de données (trame contenant les angles d'Euler fournit par la 10DOF) :
  - Grace au switch case on cherche un id valant 107, puis connaissant le format de la trame on utilise les fonctions explicitées avant pour obtenir du fichier binaire les valeurs en short puis on les devise par le coefficient correspondant (qui est explicité dans la partie datalogger), pour enfin remplir des ArrayLists qu'on transmet a la troisième partie de l'application.

```
this.eulerx10Dof.add(new Entry(intToFloat(binaryToInt16(canMsg,4),
        this.i_decimales_7),this.eulerx10Dof.size()));
this.eulery10Dof.add(new Entry(intToFloat(binaryToInt16(canMsg,6),
        this.i_decimales_7),this.eulerx10Dof.size()));
this.eulerz10Dof.add(new Entry(intToFloat(binaryToInt16(canMsg,8),
        this.i_decimales_7),this.eulerx10Dof.size()));
```

- vérifier si on a atteint la fin du fichier sinon refaire la même opération.

Après un certain nombre de testes on a remarqué que la lecture du fichier prenait beaucoup de temps (le temps de lecture d'un fichier de 20 MO est d'une minute et demi en moyenne), et qu'au niveau de la visualisation des données, on obtenait un nombre très important de points, provoquant ainsi une lenteur d'interaction avec la partie visuelle.

Les fréquences des capteurs étant suffisamment élevées, un échantillonnage lors de la lecture n'affecte pas la visualisation des données, diminue le temps de lecture d'un fichier et permet une interaction fluide avec le visuel.

Les capteurs n'ayant pas la même fréquence, une synchronisation entre les données à comparer est nécessaire pour aligner les graphes et unifier le marquage temporel. La synchronisation est faite en respectant l'algorithme explicité dans la figure 3.11 et le code correspondant illustré dans la figure 3.12.

3.3.3.3 Visualisation

Arriver enfin à la visualisation des données, on fait appel à la bibliothèque MPAndroidChart, afin de pouvoir dessiner les graphes issues de la XSens et la 10DOF et les comparés. On commence par mettre en place la charte sur laquelle on va dessigner les graphe au niveau du fichier XML.

```
<com.github.mikephil.charting.charts.LineChart
    android:id="@+id/chartGyrox"
    android:layout_width="match_parent"
    android:layout_height="300dp" />
```

On récupère cette charte grâce à une variable de type LineChaert afin d'importer les données récupérées au niveau de la deuxième partie.

On utilise ensuit une variable de type LineDataset qui a un constructeur qui admet deux argument, le premier est un arrayList de type Entry et le deuxième est un String qui va faire objet de label du graphe désigné. On crée par la suite un ArrayList de type LineDataset dans lequel on met tous les données qu'on souhaite visualiser sur la même charte, et au final on ajoute cet ArrayList à la charte.

```
LineDataSet mDataSet1 = new LineDataSet(data1,Chart1Name), mDataSet2 = new LineDataSet(data2,Chart2Name);
ArrayList<ILineDataSet> mILineDataSet = new ArrayList<>();
mILineDataSet.add(mDataSet1);
mILineDataSet.add(mDataSet2);
mDataSet1.setColor(Color.BLUE);
mDataSet1.setDrawCircles(false);

mDataSet2.setColor(Color.RED);
mDataSet2.setDrawCircles(false);

LineData mLineData = new LineData(label,mILineDataSet);

;
// add data
mChart.setData(mLineData);
mChart.setBackgroundColor(Color.WHITE);
```

Le résultat donné est illustré dans la figure 3.13

Les données GPS sont aussi visualisées, cependant le principe de visualisation demeure diffèrent. Dans ce cas on plot les données GPS directement sur une carte en utilisant l'api de Google Maps sous Android, et donc au niveau de la deuxième étape on crée une variable polyline a la place des ArrayLists dans laquelle on met tous les points GPS, puis on les intègre a une Map dans la troisième étape, faisant ainsi une reconstruction du trajet de la moto (figure 3.14).

3.4 Conlusion

L'utilisation de la carte STM32F407V, sur laquelle existait un exemple de code utilisant le DMP, était une étape clé dans l'exploitation du DMP sur la carte Mbed. Cela nous a permis de comprendre les étapes nécessaires pour l'activation du DMP et la récupérations des données fusionnées.

La première version de l'application est destinée exclusivement au debuggage, comparant ainsi les données des différents capteurs de façon qualitative, permettant de vérifier le bon fonctionnement des capteurs, et donne une idée sur la trajectoire effectuée durant la manipulation. Cette application offre un alternatif portable pour un debuggage immédiat sur place.

Dans le chapitre suivant, les résultats de notre travail sont évalués à l'aide des scénarios expérimentaux.

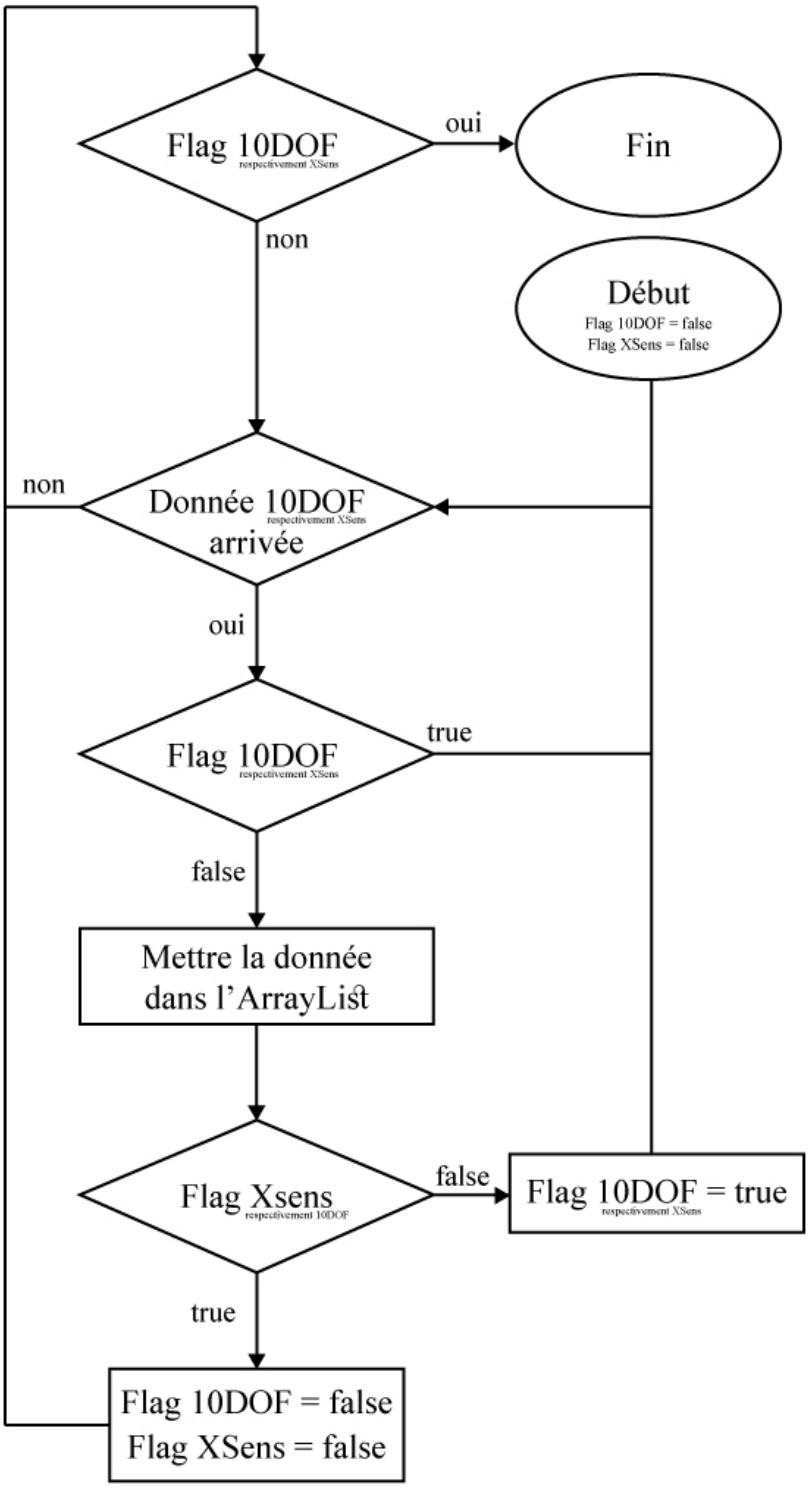


FIGURE 3.11: Algorithme de synchronisation

```
case (can_imu_10_dof_moto +7):
{
    euler10Dof++;
    if(euler10Dof%sample_rate!=0)break;
    if(euler10Dofdone)break;
    this.Dof10Mod3=true;
    this.eulerx10Dof.add(new Entry(intToFloat(binaryToInt16(canMsg,4),
            this.i_decimales_7),this.eulerx10Dof.size()));
    this.eulery10Dof.add(new Entry(intToFloat(binaryToInt16(canMsg,6),
            this.i_decimales_7),this.eulerx10Dof.size()));
    this.eulerz10Dof.add(new Entry(intToFloat(binaryToInt16(canMsg,8),
            this.i_decimales_7),this.eulerx10Dof.size()));

    if(eulerXsensdone)
    {
        eulerXsensdone=false;
        euler10Dofdone=false;
    }
    else euler10Dofdone=true;

    break;
}
```

FIGURE 3.12: le bout de code correspondant a l'échantillonnage et à la synchronisation

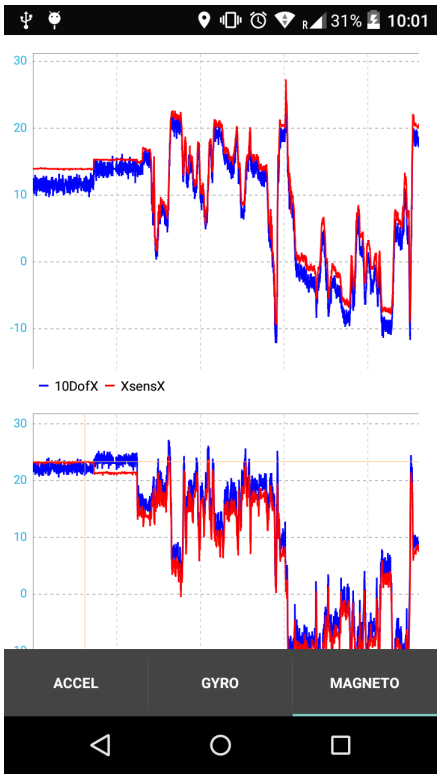


FIGURE 3.13: Interface graphique de la partie visualisation : graphes

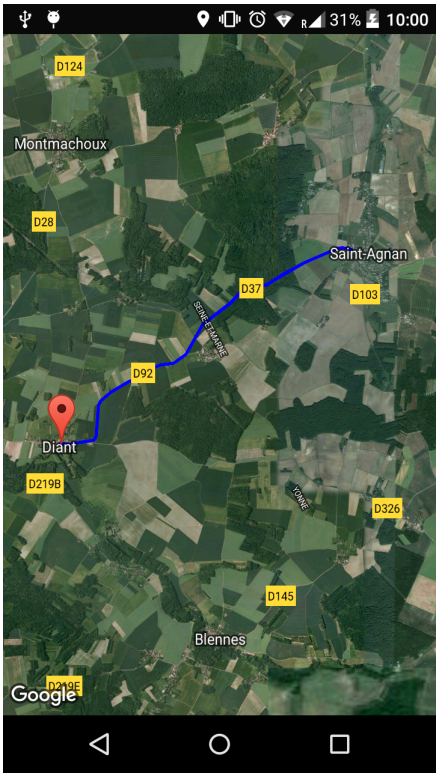


FIGURE 3.14: Interface graphique de la partie visualisation : trajectoire

Chapitre 4

Résultats expérimentaux

4.1 Introduction

Dans ce chapitre, on évalue les données recueillies à partir de la 10DOF (IMU) et on les compare avec les données de référence issues de la Xsens. Pour ce faire, on définit des scénarios expérimentaux permettant d'obtenir ces données.

Le projet étant encore en phase de validation de choix de capteurs, on s'intéresse au premier lieu à une comparaison qualitative de données qu'on effectue à l'aide de l'application Android (VIROLO++) précédemment présentée.

De même, une évaluation de précision des cartes Google Maps et Géoportail est abordée.

4.2 Scénarios expérimentaux

4.2.1 Scénario sur table

Pour évaluer qualitativement les angles d'Euler provenant de l'MPU-9250, une manipulation est développée. Cette manipulation consiste en une application qui reçoit les angles d'Euler du Mbed et oriente un cube suivant les angles reçus on faisant appel à la librairie OpenGL. Cela offre une visualisation intuitive en temps-réel des angles de l'orientation du capteur.

Ce scénario a été élaboré avant le début du stage au DIGITEO qui permettait de tester la fusion de données sans avoir accès à la moto.

Afin de développer un tel projet sous Android il faut faire appel à des bibliothèques permettant d'ouvrir et d'accéder à l'USB afin d'envoyer et de recevoir des données via celui-ci.

4.2.1.1 Librairie utilisée

La librairie Android utilisée est physicaloid [22]
Cette librairie utilise essentiellement quatre méthodes dédiées à la communication série via USB :

- openDevice() et closeDevice() : permettent l'ouverture et la fermeture de la communication.
- read() et write() qui permettent respectivement la lecture et l'écriture de données.

La méthode write() permet d'envoyer des données à partir de la plateforme Android.

L'utilisation d'un listener couplé à OnRead() permet à la plateforme Android une lecture dès qu'une donnée est présente dans le buffer.

Les paramètres de la liaison peuvent être fixés avec la méthode setBaudrate() ou avec la méthode setConfig().

4.2.1.2 Code Java :

L'application Android utilise la librairie OpenGL pour la visualisation de l'orientation.

- Présentation de l'OpenGL :
  OpenGL (Open Graphics Library) [23] est une librairie qui contient un nombre de fonctions de calcul graphique 2D et 3D. Disponible sur plusieurs plateformes (Android y compris), elle est utilisée pour des applications de jeux vidéo, CAO ou modélisation. Elle permet à un programme de déclarer la géométrie d'un objet sous forme de points, vecteurs ou polygones, et calcule par la suite les projections nécessaires pour afficher l'image sur écran.
- Description de l'application :
  On commence par décrire l'objet géométriquement avec la classe cube.java en définissant les faces constituant notre cube. (code illustré dans la figure 4.1

```
private float[] vertices = {  // Vertices of the 6 faces
   // FRONT
   -1.0f, -1.0f,  1.0f,  // 0. left-bottom-front
    1.0f, -1.0f,  1.0f,  // 1. right-bottom-front
   -1.0f,  1.0f,  1.0f,  // 2. left-top-front
    1.0f,  1.0f,  1.0f,  // 3. right-top-front
   // BACK
    1.0f, -1.0f, -1.0f,  // 6. right-bottom-back
   -1.0f, -1.0f, -1.0f,  // 4. left-bottom-back
    1.0f,  1.0f, -1.0f,  // 7. right-top-back
   -1.0f,  1.0f, -1.0f,  // 5. left-top-back
   // LEFT
   -1.0f, -1.0f, -1.0f,  // 4. left-bottom-back
   -1.0f, -1.0f,  1.0f,  // 0. left-bottom-front
   -1.0f,  1.0f, -1.0f,  // 5. left-top-back
   -1.0f,  1.0f,  1.0f,  // 2. left-top-front
   // RIGHT
    1.0f, -1.0f,  1.0f,  // 1. right-bottom-front
    1.0f, -1.0f, -1.0f,  // 6. right-bottom-back
    1.0f,  1.0f,  1.0f,  // 3. right-top-front
    1.0f,  1.0f, -1.0f,  // 7. right-top-back
   // TOP
   -1.0f,  1.0f,  1.0f,  // 2. left-top-front
    1.0f,  1.0f,  1.0f,  // 3. right-top-front
   -1.0f,  1.0f, -1.0f,  // 5. left-top-back
    1.0f,  1.0f, -1.0f,  // 7. right-top-back
   // BOTTOM
   -1.0f, -1.0f, -1.0f,  // 4. left-bottom-back
    1.0f, -1.0f, -1.0f,  // 6. right-bottom-back
   -1.0f, -1.0f,  1.0f,  // 0. left-bottom-front
    1.0f, -1.0f,  1.0f   // 1. right-bottom-front
};
```

FIGURE 4.1: Code java correspondant a la déclaration du cube

Vient ensuite la création de l'objet. Pour ce faire on doit d'abords définir les caractéristiques du cube à savoir les dimensions, la couleur et possibilité de rotation en utilisant une variable de type GL10 (appelée gl)

Au niveau de la réception de données, on récupère les angles d'Euler envoyés par le Mbed et les applique sur l'objet en suivant l'ordre suivant : Rotation autour de Z puis autour de Y et finalement autour de X (yaw, pitch, roll) en utilisant les instructions suivantes :

```
gl.glRotatef(AndroidUSBSerialMonitorLite.angley, 0.0f, 0.0f, 1.0f);
gl.glRotatef(AndroidUSBSerialMonitorLite.anglep, 0.0f, 1.0f, 0.0f);
gl.glRotatef(AndroidUSBSerialMonitorLite.angler,1.0f, 0.0f, 0.0f);
```

Finalement on dessine notre cube (figure 4.2) en utilisant l'instruction suivante :

```
cube.draw(gl);
```

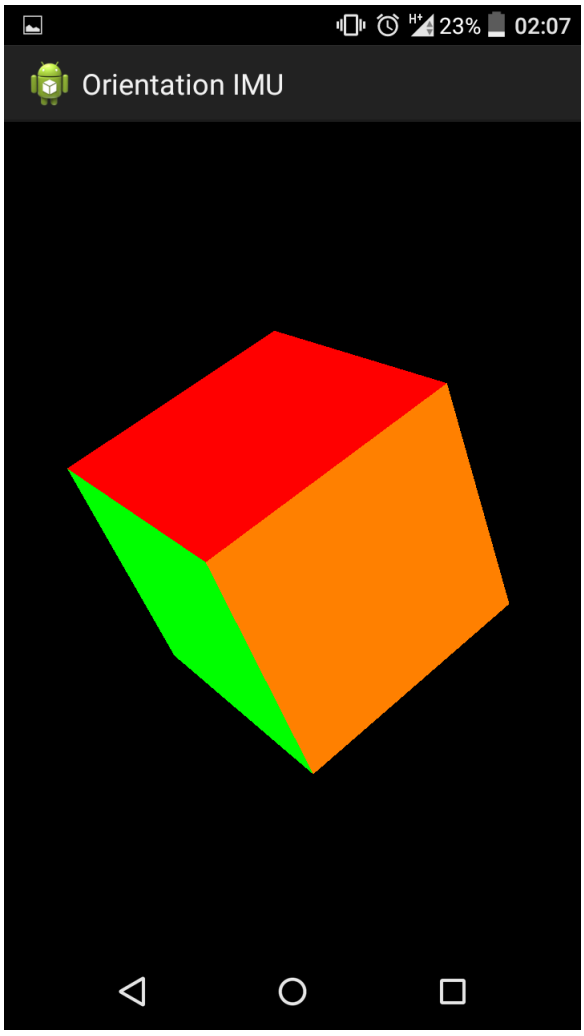


FIGURE 4.2: Aperçu de l'application Orientation IMU

4.2.2 Scénario sur Moto

Le système étant déjà installé sur la moto, il suffit d'effectuer un trajet prédéfini durant lequel la moto subit des changements considérables d'angles et de vitesses.

On récupère la clé USB branchée sur l'enregistreur, obtenant ainsi les données de l'expérience qu'on va pouvoir visualiser, les comparer et les interpréter à l'aide de l'application Android.

4.3 Evaluation des résultats

4.3.1 Données inertielles

Les données inertielle (Accélérations, vitesses angulaires et champ magnétique) obtenues par la 10DOF ont les mêmes allures que celles de la Xsens qui représente la

référence. On constate quelques différences qui sont dues au bruit au niveau de la 10DOF. Elles sont illustrées par les figures 4.3, 4.4 et 4.5.

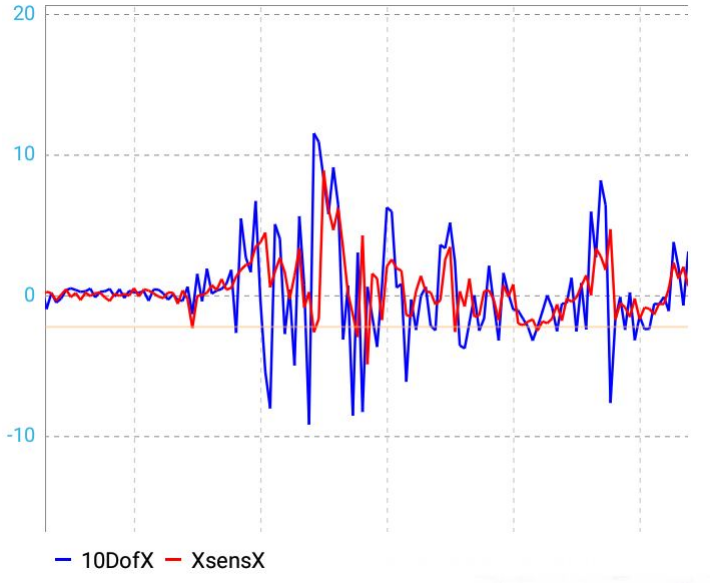


FIGURE 4.3: Accélération selon l'axe X

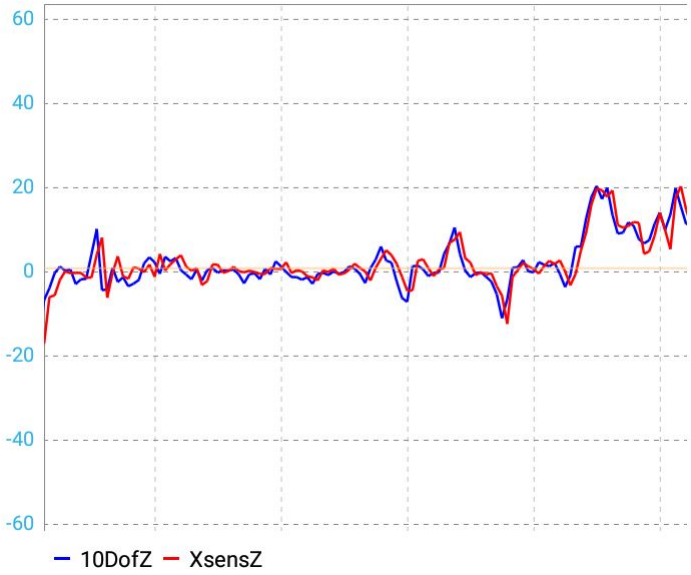


FIGURE 4.4: Vitesse angulaire suivant l'axe Z

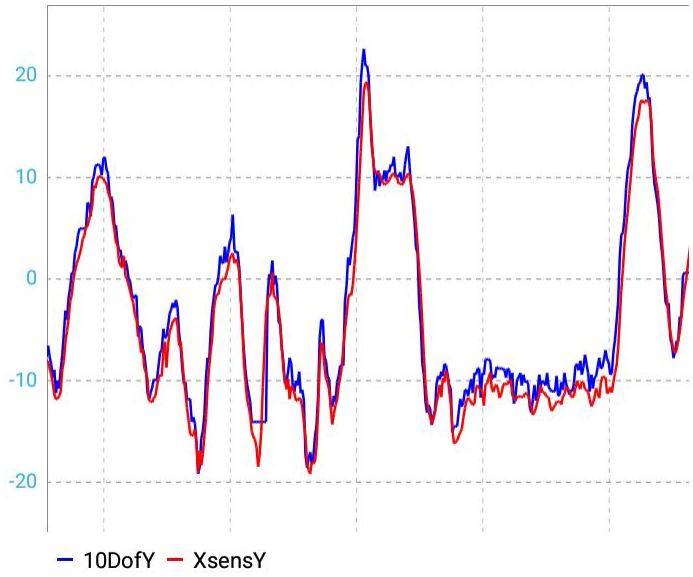


FIGURE 4.5: Champ magnétique selon l'axe Y

4.3.2 Angles d'Euler

4.3.2.1 Avec DMP :

FIGURE 4.6: Angle de roulis avec DMP

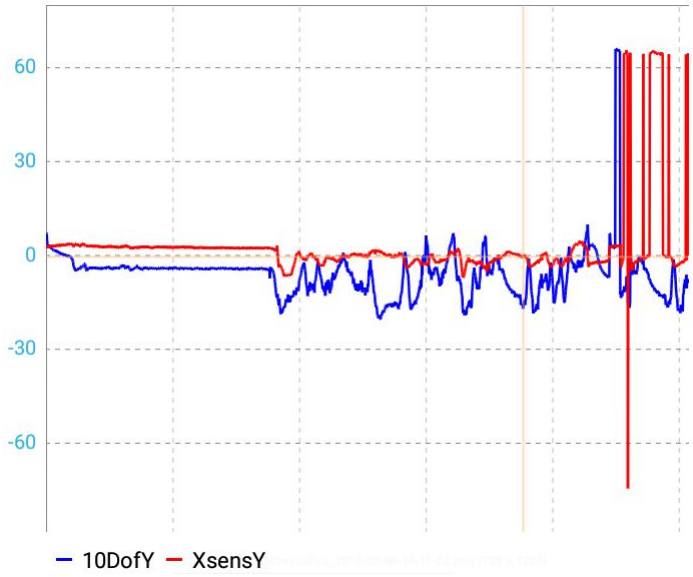


FIGURE 4.7: Angle de tangage avec DMP

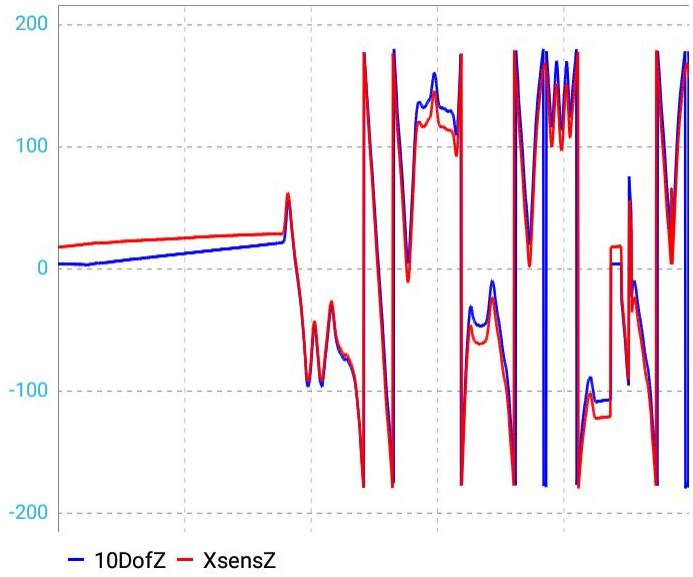


FIGURE 4.8: Angle de lacet avec DMP

Le tracé de l'angle de roulis (figure 4.6) calculé par le DMP met en avant un caractère passe-haut : Il suit bien l'allure de celui de la Xsens dans le cas des variations rapides d'angle mais un retour vers zéro ce produit dès que la valeur d'angle se stabilise, dont on connaît pas l'origine car on n'a pas accès à la bibliothèque du DMP.

L'angle de tangage (figure 4.7) calculé par le DMP suit globalement celui calculé par la Xsens mais reste considérablement bruité.

Par contre, L'angle de lacet (figure 4.8) de la 10DOF concorde avec l'angle donnée par la Xsens à une erreur près.

4.3.2.2 Sans DMP :

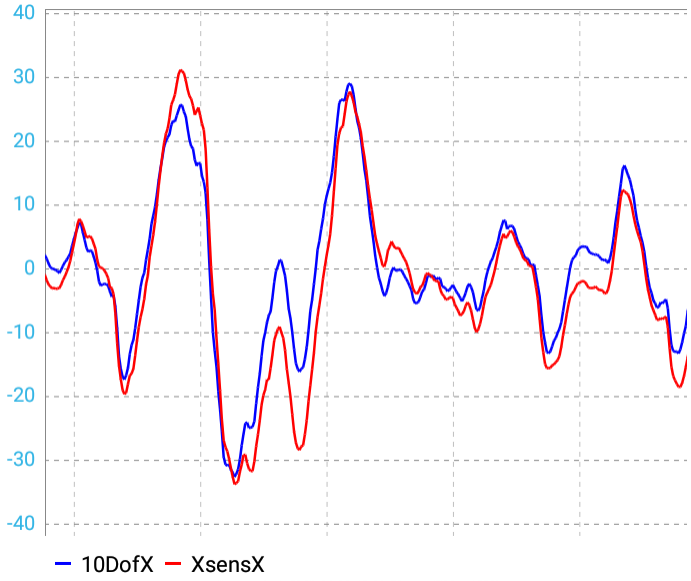


FIGURE 4.9: Angle de roulis sans DMP

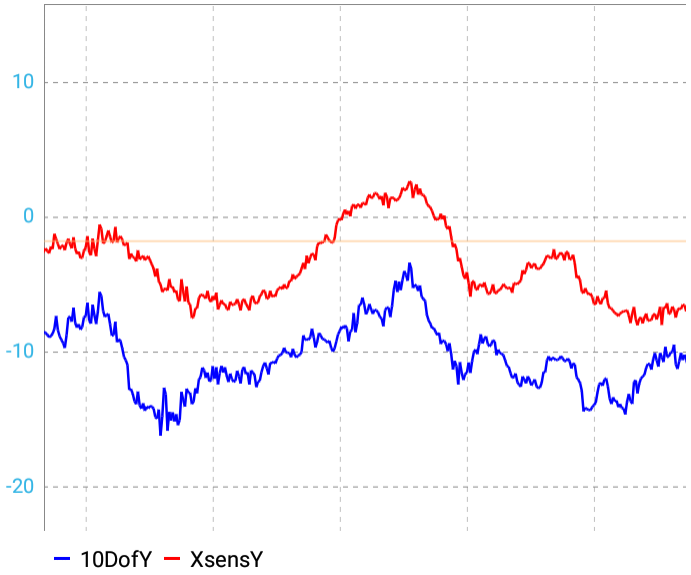


FIGURE 4.10: Angle de tangage sans DMP

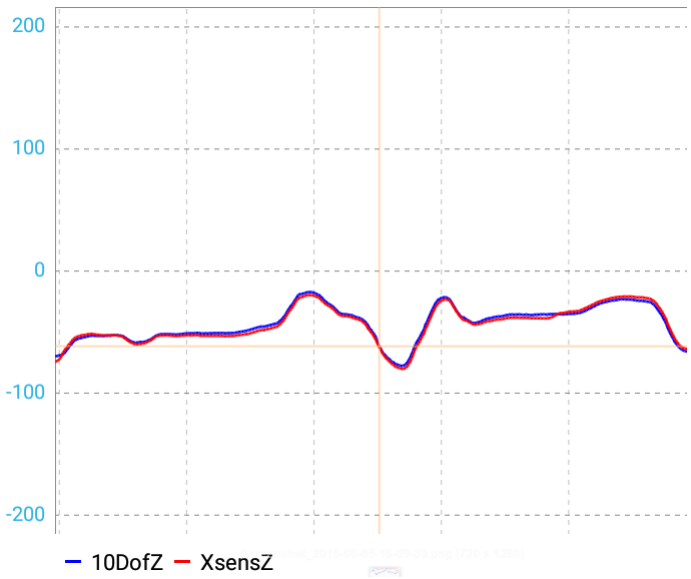


FIGURE 4.11: Angle de lacet sans DMP

L'algorithme de fusion de donnée de Madgwick donne des résultats plus satisfaisants que celles du DMP.

Pour l'angle de roulis (figure 4.9), On remarque que Le phénomène du retour à zéro n'est plus. L'écart entre l'angle de la 10DOF et l'angle de référence est néanmoins variable.

L'angle de tangage (figure 4.10) est décalé par rapport à la référence d'une valeur quasi-constante.

Les deux angles de lacet (figure 4.11) sont bien corrélés.

4.4 Évaluation de plateformes cartographiques

Il existe plusieurs solutions pour tracer des données GPS sur une carte géographique qu'elle soit aérienne ou satellitaire.

Pour notre application, deux plateformes ont été considérées, l'une utilisant des images satellitaire (Google Maps) tandis que l'autre fait appel à des images aériennes (Géoportail).

4.4.0.1 Présentation de Google Maps :

Google Maps [24] est un service gratuit de cartographie en ligne, utilisant des images satellitaire, développé par Google. Il permet de zoomer à partir de l'échelle d'un pays à l'échelle d'une rue, consulter certains détails présent dans la rue, tracer un itinéraire et connaitre sa position sur la carte. Il admet trois types de vues qui sont :

- Satellitaire qui affiche une image réelle d'un endroit spécifique.
- Classique en plan avec les noms des rues, quartiers, villes ... etc.
- Hybride qui fusionne les deux vues précédentes en une seule.

4.4.0.2 Présentation de Géoportail :

Géoportail [25] est un service web public crée par IGN (Institut national de l'information géographique et forestière), qui permet d'utiliser des services de recherche et de visualisation de données géographiques et géolocalisées.

C'est un outil ressemblant à Google Maps, excepté que celui-ci utilise des images prises par des avions et non par satellites, et qu'il couvre uniquement le territoire français et non pas tous le globe.

4.4.1 Comparaison entre les cartes Google Maps et Géoportail

On prend 4 coordonnées du GPS RTK qu'on place au niveau de Google Maps et Géoportail et on compare la position obtenue avec la position réelle ou le GPS RTK était posé (figure 4.12 et figure 4.13). Les résultats sont détaillés dans le tableau 4.1

FIGURE 4.12: Screen-shot de Google Maps

point			Google Maps				IGN (Géoportail)			
	Coordonnées RTK		Coordonnées Imagerie		Erreur		Coordonnées Imagerie		Erreur	
	Lat	Long	Lat	Long	Lat	Long	Lat	Long	Lat	Long
1	48.7128	2.1687	48.7128	2.1687	8E-06	1.1E-05	48.7128	2.16873	5.42E-07	6.44E-06
2	48.7088	2.1782	48.7088	2.1782	8E-06	4E-06	48.7088	2.1781	5.98E-07	8.97E-06
3	48.7154	2.1748	48.7154	2.1748	1E-05	1.1E-05	48.7154	2.1748	3.618E-06	8.7E-07
4	48.7090	2.1620	48.7090	2.1620	5E-06	5.7E-05	48.7090	2.16202	2.655E-06	4.69E-06
				Err. Moy.	7.75E-06	2.075E-05			1.853E-06	5.243E-06
				mètres	0.8598	1.5242			0.2056	0.3851

TABLE 4.1: Tableau de comparaison Google Maps Geoportail

L'erreur moyenne en mètre est moins importante dans le cas de Géoportail (0.20m en latitude et 0.39m en longitude) que celle de Google Maps (0.86m en latitude et 1.52m en longitude). Donc on est amené à affirmer que les cartes Geoportail sont plus précises.

FIGURE 4.13: Screen-shot de Geoportail

4.5 Conclusion

Dans ce chapitre, on a constaté que les données inertielles de la 10DOF sont suffisamment cohérentes par rapport aux de données de référence. Cependant, les angles d'Euler dépendent du choix de l'approche d'implémentation (Algorithme sur microcontrôleur ou DMP). Les résultats nous amènent à dire que l'approche DMP reste un choix inadéquat car, en plus des données incorrectes, le code implémentant cette approche est volumineux. Concernant la visualisation de la trajectoire, la plateforme Géoportail s'impose grâce à sa précision qui est considérablement meilleure que celle de Google Maps. Cependant, on a opté pour Google Maps comme première solution car elle dispose contrairement a Géoportail d'un API facilement intégrable sur Android. Cela nous a permis de développer rapidement une solution de visualisation de trajectoire.

Conclusion et perspectives

Notre projet de fin d'étude a évolué suivant plusieurs phases qu'on explicite dans le diagramme de Gantt suivant. figure 4.14 :

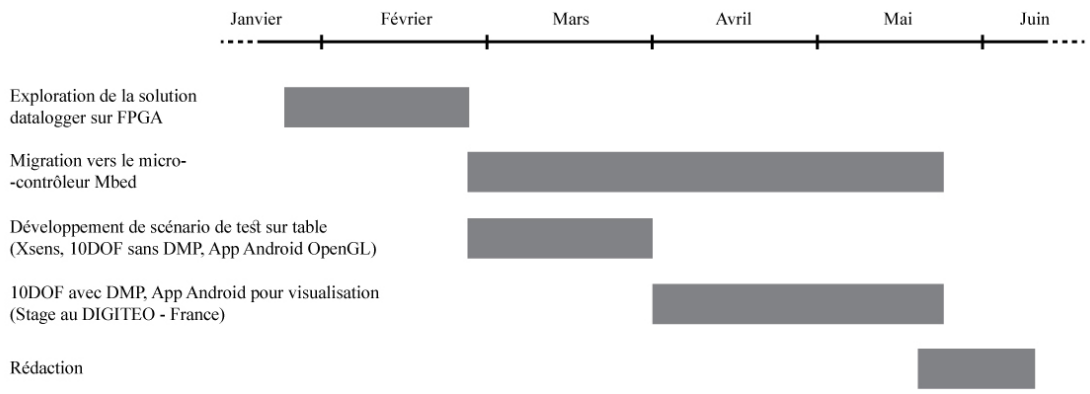


FIGURE 4.14: Diagramm de Gantt : Déroulement du PFE

Le stage au sein du laboratoire DIGITEO nous a permis de résoudre plus rapidement les problèmes rencontrés (dans l'étude et le développement des deux noeuds IMU et Visuel) et de valider notre avancement à l'aide des outils techniques disponibles et des expérimentations sur moto.

Notre travail sur l'IMU a répondu principalement à la question d'utilisation du DMP qui présentait une approche de fusion de données non explorée auparavant. Cette approche s'est avérée inadéquate car elle nécessitait plus de ressources et donnait des résultats non satisfaisants par rapport à l'utilisation des algorithmes de fusion sur microcontrôleur Mbed.

Le fait que la libraire exploitant le DMP est inaccessible (Code source non disponible) nous empêche d'aller plus loin dans l'utilisation du DMP.

Pour répondre à ce problème, l'implémentation et l'évaluation d'autres algorithmes de fusion de données (tel que le filtre de Kalman) et l'exploration des méthodes de calibration de capteurs présentent une perspective.

Pour le noeud visuel, les perspectives sont de :

- Mettre dans les paramètres, l'option de modifier ou d'ajouter un type de données (Identifiant CAN, format des données) à lire à partir des fichiers C2U.
- Avoir une charte personnalisable, où on peut ajouter des graphes selon notre besoin pour faire des comparaisons.

- Tracer la trajectoire sur des cartes IGN avec Geoportail (à la place de Google Maps).
- Afficher sur la carte les informations correspondantes à un point du trajet effectué (vitesse, angle de roulis, accélération, direction, ... etc.) lors d'un clique sur le point.
- Intégrer un outils de comparaison quantitative de données et de calcul d'erreurs.

Bibliographie

[1] NHTSA's National Center for Statistics et Analysis. *NHTSA : Motorcycles Traffic Safety Fact Sheet*. 2007. URL : http://www-nrd.nhtsa.dot.gov/Pubs/810990.PDF.

[2] Sarra SMAIAH. *Système De Reconstruction De Trajectoires Temps Réel Pour Véhicules Deux-roues Motorisées*. Rapp. tech. Ecole Nationale Polytechnique - Université de Paris Sud, 2016.

[3] Bruno LARNAUDIE. "Codesign, architecture fonctionnelle de fusion et architecture capteurs pour l'identification de situations accidentogènes. Application à la sécurisation de véhicules deux-roues." Thèse de doct. Université Paris-Sud XI, 2006.

[4] AMS. *CD4007UB*. https://ams.com/eng/content/download/595083/1609657/file/AS5047D_Datasheet_EN_v4.pdf. 2016.

[5] SICK. *Dx35 Mid Range Distance Sensors*. https://www.mysick.com/saqqara/im0049845.pdf. 2013.

[6] Maestro. *A2200-A*. http://update.maestro-wireless.com/GNSS/A2200-A/Maestro_A2200_GPS_Module_Product_Brief_v14-2.pdf.

[7] F. Duquenne et al. *GPS localisation et navigation par satellites*. Hermes, 2005.

[8] Septentrio. *Altus APS3G*. http://www.septentrio.com/products/gnss-receivers/rover-base-receivers/smart-antennas/aps3g.

[9] Xsens. *MTi : Miniature MEMS based AHRS*. https://www.xsens.com/products/mti/.

[10] BECK. *SC23 Embedded controller*. http://www.beck-ipc.com/en/products/sc2x/sc23.asp.

[11] *Road vehicles - Controller area network (CAN)*. 2003.

[12] Jean-Marc Irazabal et Steve Blozis. *I2C MANUAL*. Philips Semiconductors. 2003.

[13] Inc. Motorola. *SPI Block Guide V03.06*. 2000.

[14] Bertrand PETIT. *Architectures des Réseaux*. Ellipses, 2003.

[15] InvenSense Inc. *MPU-9250 Product Specification Revision 1.0*. https://www.invensense.com/wp-content/uploads/2015/02/InvenSense-Motion-Sensor-Universal-EV-User-Guide3.pdf. 2014.

[16] DROTEK. *IMU 10DOF - MPU9250 + MS5611*. http://www.drotek.com/shop/fr/home/466-imu-10dof-mpu9250-ms5611.html.

[17] InvenSense INC. *MPU-9250 Register Map and Descriptions*. https://www.invensense.com/wp-content/uploads/2015/02/MPU-9250-Register-Map.pdf. 2013.

[18] Sebastian O.H. MADGWICK. *An efficient orientation filter for inertial and inertial/magnetic sensor arrays*. http://www.x-io.co.uk/res/doc/madgwick_internal_report.pdf. 2010.

[19] "Embarcadero Introduces Starter Editions of C++Builder and Delphi Rapid Application Development Environments". In : *SD Times (BZ Media LLC)* (2011).

[20] Nazim BENBOURAHLA. *Android 5, les fondamentaux du développement d'application Java*. ENI, 2015.

[21] DANIEL.F. *Simple Explorer*. https://github.com/DF1E/SimpleExplorer. 2013.

[22] KSKSUE. *Physicaloid Library*. https://github.com/ksksue/PhysicaloidLibrary.

[23] *OpenGL Overview*. https://www.opengl.org/about/.

[24] *What is the Google Maps*. https://developers.google.com/maps/.

[25] *Présentation du Géoportail 3.0*. http://www.ign.fr/institut/actualites/presentation-geoportail-30. 2013.

[26] NXP SEMICONDUCTORS. *LPC1769/68/67/66/65/64/63*. http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf. 2015.

[27] *Android Studio - The Official IDE for Android*. https://developer.android.com/studio/index.html.