Remerciements

Avant tout, nous remercions Dieu le tout puissant qui nous a donné la force et nous a aidé à mener à terme ce travail.

Nous remercions ensuite toutes les personnes qui ont directement et indirectement contribué à la confection de ce travail.

Nous voudrions remercier notre promoteur Monsieur DILMI Smain, pour ces conseils prodigués et son encouragement.

Nous tenons aussi à exprimer nos sincères remerciements et respects à tous les professeurs qui nous ont enseigné et qui par leurs compétences nous ont soutenu dans la poursuite de nos études.

Enfin, nous remercions nos familles, nos amis, et nos camarades pour leur soutien tout le long de notre cursus universitaire.

Résumé

Le but de ce projet est principalement de mener un système de navigation inertiel élémentaire et de tirer partie des capacités offertes par le système micro-électromécanique (MEMS). Les principaux aspects couverts dans ce projet sont la simplicité, l'intuition et l'équité, en utilisant des capteurs MEMS et un microcontrôleur programmable Arduino pour émuler le système de navigation par inertie.

Les informations fournies par le prototype réalisé en tant qu'évaluation en temps réel du mouvement et de la déviation momentanée de l'avion sont illustrées par une animation graphique sous forme d'un simulateur du vol personnalisé. Le travail accompli se révèle être un système simplifié qui permet de comprendre et de construire un système inertiel abordable, fiable et performant.

Abstract

The purpose of this project is mainly to convey an elementary inertial navigation system, and take advantage of the capabilities that the Microelectromechanical system (MEMS) affords. The main aspects cover the simplicity, intuition, and fairness in the design of the system, using MEMS sensor and a Programmable microcontroller Arduino to emulate the inertial navigation system.

The information yielded from the prototype as real time evaluation of the momentarily motion and deviation of the airplane is illustrated on a personalized flight simulator interface, which is a neat technique of showing the aircraft attitude.

The outcome of the work achieved turns to stand up as simplified system for the sake of understanding and building an affordable and reliable system.

التجريد

الغرض الأساسي من هذا المشروع هو التعبير عن نظام الملاحة بالقصور الذاتي و الاستفادة من المميزات التي يقدمها نظام الكهرو ميكانيكيات الدقيقة. الجوانب التي يغطيها المشروع شملت البساطة و البديهية في تصميم النموذج. استخدمنا في هذا المشروع تقنية الكهرو ميكانيكيات الدقيقة و متحكم دقيق قابل للبرمجة متمثل في أردوينو , و ذلك من أجل محاكاة حركة الطائرة. و لقد توافقت مكونات هذا المشروع من حيث السعر المناسب و الأداء المتوقع منها. البيانات التي تم التحصل عليها هي قياس لحظي لحركة الطائرة و تم عرضها على شكل واجهة رسومية لمحاكي طيران خاص. النتيجة الاخيرة للمشروع كانت جيدة و مرضية.

.

Sommaire

CHAPITRE III : Réalisation & Simulation

Liste des Figure

Chapitre I

Chapitre II

Chapitre III

Liste des Tableaux

Liste des Abréviations

Introduction:

GPS: Global Positioning System

MEMS: Micro-Electro-MechanicalSystem

Chapitre I

FBW : Flyby-Wire

IMA : Integrated Modular Avionics

AFDX : Avionics Full-Duplex Switched Ethernet.

Chapitre II

DR : Dead Reckoning

AHRS : Attitude Heading Reference Système.

INS : Inertial Navigation System.

VFR : Visual *Flight* Rules

IFR: Instrument *Flying Rules*

VMC: Visual Meteorological Conditions

VOR: Very High Frequency Omni-directional Radio Range

HEA: Heur Estimé d'Arriver

SINS: Système de navigation inertielle des navires

IN : Inertial Navigation.

ECI: Earth Centered Inertial.

ECEF: Earth-Centered, Earth-Fixed.

NED : North East Down.

DCM : Direction Cosine Matrix.

IMU : Inertial Measurement Unit.

Chapitre III

DOF: Degree of Liberty

MPU: Motion Processor Unit.

DMP : Digital Motion Processor.

CI : Common Interface.

ST : STMicroelectronics.

CIMOS: Center for Intelligent Manufacturing of Semiconductors.

SCL: Serial Clock.

SDA: Serial Data.

FIFO: First In First Out.

CPU: Central Proccesing Unit.

IDE: Integrated development environment.

CM: Complementary Filter.

KM: Kalman Filter.

Introduction

L'évaluation de l'attitude, ou l'orientation dans un espace à 3 dimensions est devenue au fil du temps un enjeu scientifique important. Déterminer l'attitude d'un corps rigide est l'objet de nombreuses études dans déférentes disciplines, avec des applications aussi variées que le contrôle en attitude des satellites et des aéronefs, la robotique aérienne, ou bien encore l'explosion des applications liées aux jeux d'animation.

Le but d'un système d'orientation et de positionnement est de fournir des informations précises, sur la position, sur la vitesse et l'attitude à tout instant et en tout point du globe ainsi que l'incertitude qui y est associée.

Un intérêt considérable s'est alors développé au sein de la communauté scientifique et industrielle pour les techniques de Positionnement. Lorsque l'objet à repérer se trouve dans un environnement à ciel ouvert, le système GPS est utilisé pour déterminer sa position (longitude, latitude et altitude) à tout moment. Cependant il arrive souvent que la réception GPS soit faible et de ce fait l'estimé du positionnement n'est pas fiable. Pour cela i et qui consiste à Fusionner les données de positionnement et d'estimation d'attitude sous un système de navigation inertielle.

Le développement de l'industrie aéronautique connait une course de plus en plus rapide est spécialement vers la miniaturisation mais sans laissée les origines. En effet la demande d'appareils de petites dimensions a mené les conducteurs à faire plus d'efforts pour répondre aux exigences du marché. C'est dans ce cadre que se situent les travaux présentés dans ce mémoire de fin d'étude.

Les progrès récents de la technologie MEMS (Micro-Electro-MechanicalSystem) a conduit à la conception des capteurs dédiés de taille réduite, ayant une faible consommation en énergie et faible cout tels que les accéléromètres et les gyromètres. Ces capteurs, conçus sous forme d'une triade orthogonale. Le gyromètre mesure la vitesse angulaire, l'accéléromètre mesure la gravité ainsi que l'accélération linéaire La fusion de toutes ces données permet de remonter à l'attitude du corps rigide dans l'espace à 3 dimensions. L'adaptation d'une telle technique contribue à l'augmentation de la sécurité et la fiabilité de l'aéronef.

L'objectif de ce travail est donc d'étudier les systèmes micro-électromécanique MEMS et réaliser une carte électronique sous forme d'une puce de Système micro électromécanique

avec système de navigation inertielle intégré basé sur un microcontrôleur afin d'estimer en temps réel l'attitude d'un aéronef a 3 dimensions, et aussi Sélectionnez une méthode appropriée pour simuler les mouvements (attitude) d'un avion à 3 dimensions.

Pour y aboutir, J'ai suivi le plan suivant :

Chapitre I : est consacré à la présentation du système avionique en donnant les définitions, caractéristiques, notamment l'évolution des architectures avionique embarquées et quelques fonctionnalités avionique.

Chapitre II : la présentation de la centrale inertielle et sa constitution, le principe de navigation inertielle et le principe des MEMS a été considérée dans ce chapitre pour pouvoir plonger dans la compréhension du fonctionnement du système de navigation inertielle.

Chapitre III : une présentation de l'ensemble des outils matériels et logiciels utilisés et déployés pour aboutir à la conception du projet. Et les différentes étapes suivies pour la réalisation du projet

Chapitre IV : Enfin nous avons présenté dans le dernier chapitre nos résultats concernant l'étude et la réalisation.

Conclusion et perspectives pour améliorer le projet, clôturent ce mémoire.

I.1. Introduction

Ce chapitre est une introduction aux systèmes avioniques dont l'objectif est de présenter l'ensemble des composants essentiels pour le bon fonctionnement de ce celui-ci.

I.2. Historique

Avant les années 60, la quasi-totalité des équipements embarqués dans l'avion étaient analogiques. C'est-à-dire, basés sur des techniques mécaniques par exemple.

L'évolution des besoins dans les avions implique une augmentation et une complexation des équipements, ainsi qu'un accroissement des données échangées.

Les éléments analogiques ne permettent plus de répondre aux besoins en termes de performance et de fiabilité de fonctionnement. Une évolution majeure de l'avionique est l'intégration de l'informatique et de l'électronique. Ainsi, au début des années 60, le remplacement des équipements analogiques par des équipements numériques se démocratise. Dans les années 70/80, des systèmes hydrauliques, électriques ainsi que les premiers micro-ordinateurs sont installés à bord des avions. Airbus, pour répondre aux besoins croissants en termes de performance et de fiabilité, a progressivement introduit des composants numériques dans ses aéronefs. Nous remarquons une nette progression des besoins. Ainsi, le volume logiciel est passé de quatre Méga octets pour l'A310 dans les années 70 à dix Méga octets à les années 80 pour l'A320. Aujourd'hui, il y a plusieurs centaines de Méga octets dans l'A380 et cette croissance exponentielle ne ralentit pas jusqu'à présent [3].

I.3. Description du système avionique

«Avionique» est un mot dérivé de la combinaison de l'aviation et de l'électronique.

Un système avionique signifie l'ensemble des moyens informatiques et électroniques embarqués à bord d'un avion. Plus précisément, l'avionique s'entend de tout système de l'aéronef qui dépend de l'électronique pour son fonctionnement, bien que le système puisse contenir des éléments électromécaniques. Par exemple, un système de contrôle de vol Flyby-Wire (FBW) dépend de calculateurs électroniques pour son fonctionnement efficace, mais il existe également d'autres éléments tout aussi essentiels dans le système. Il s'agit notamment des gyroscopes à taux d'état solide et des accéléromètres pour mesurer le mouvement angulaire et linéaire des aéronefs et des capteurs de données aériennes pour mesurer l'altitude, la vitesse et l'incidence. Donc l'avionique est l'ensemble des capteurs, Actionneurs, calculateurs, logiciels et bus réseaux, permettant aux systèmes de remplir les différentes fonctionnalités d'un avion. Ces fonctionnalités sont soumises à des contraintes temps réel et de criticité [4].

Le bon fonctionnement d'un système temps réel ne dépend pas seulement de la justesse des résultats, mais aussi du temps auquel ils sont fournis (au bon instant).

Il est donc soumis à deux types de contraintes : fonctionnelles qui font référence aux résultats de calcul et temporelles qui font référence aux exigences sur les dates auxquelles ces résultats doivent être délivrés. Un système avionique en temps réel et distribué, car il est composé d'un ensemble de sous-systèmes à temps réel hétérogènes repartis dans l'avion et qui doivent intercommuniquer via divers réseaux embarqués, pour réaliser ses fonctions. En avionique, les données en entrée du système sont fournies par les capteurs ou les consignes. Ces données sont traitées par les logiciels embarqués sur les calculateurs d'un ou de plusieurs sous-systèmes, puis en fonction des résultats obtenus, ces derniers agissent sur le procédé contrôlé (l'avion) au travers des actionneurs [3].

Il ressort ainsi deux dimensions dans l'avionique :

- Premièrement, une dimension système chargée de la mise en place des architectures physiques embarquées dans l'avion et de la gestion temps réel des applications qu'elles supportent.
- Deuxièmement, une dimension réseaux chargée de la mise en place des bus d'interconnexion entre les diverses architectures physiques et de la gestion temps réel des messages transmis sur ces bus.

L'industrie de l'avionique est une industrie majeure de plusieurs milliards de dollars et l'équipement avionique d'un avion militaire ou civil moderne peut représenter environ 30% du coût total de l'avion [4].

Le rôle joué par les systèmes avioniques dans un aéronef moderne pour permettre à l'équipage d'effectuer la mission de l'avion peut s'expliquer par une structure hiérarchique comprenant des couches de tâches spécifiques et des fonctions du système avionique, comme le montre la Figure I.1 Cela montre les fonctions principales, ou «centrales», qui sont principalement dans des aéronefs militaires et civils. Il faut cependant noter que certains systèmes avioniques ont été omis de ce diagramme pour plus de clarté.

En se référant à la Figure I.1, on peut voir que les principaux sous-systèmes avioniques ont été regroupés en cinq couches en fonction de leur rôle et de leur Fonction. Ceux-ci sont brièvement résumés ci-dessous afin de donner une image globale des rôles et fonctions des systèmes avionique dans un avion.

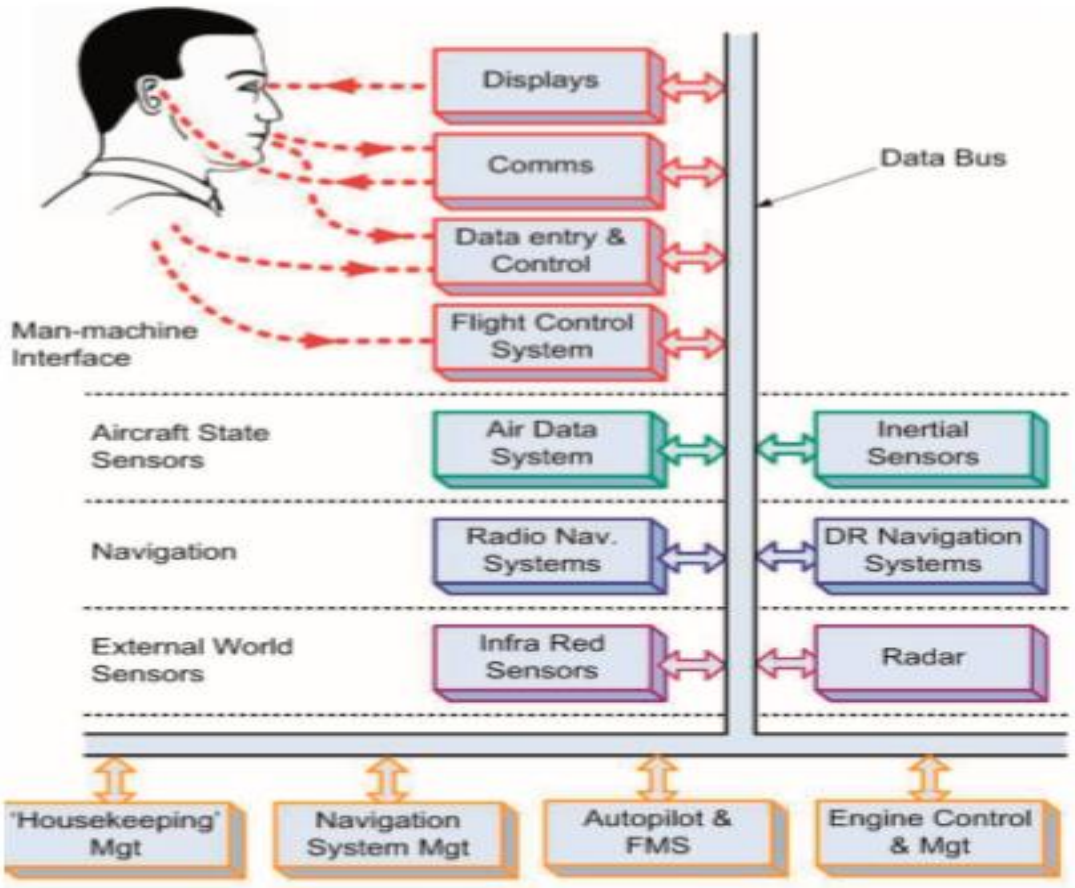


Fig I.1 : Organigramme du système avionique principal [4].

I.4. Exemples de fonctionnalités avioniques :

Un système avionique doit réaliser un ensemble de tâches pour assurer le bon fonctionnement de l'avion. Sans être exhaustif, nous nous citons dans cette section, que quelques-unes d'entre elles :

- Génération et distribution électrique et gestion du carburant.
- Planification et contrôle de la trajectoire de l'avion (suivi de plan de vol, tenue de cap, mesure de l'altitude...).
- Contrôle des commandes de vol.
- Pilotage automatique de l'avion.
- Gestion des écrans d'affichage du cockpit.
- Gestion des alarmes.
- Système antigivrage et dégivrage.
- Gestion du confort des passagers dans l'avion.

I.5. Évolution des Architectures avioniques

Depuis l'avènement de l'avionique jusqu'à nos jours, plusieurs architectures se sont succédé pour répondre aux besoins de l'évolution des systèmes avioniques. L'architecture d'un système influe directement sur la façon dont ses entités sont reliées entre elles et comment elles coopèrent [3].

Nous listons ci-dessous brièvement ces principales différentes architectures avioniques :

Architecture centralisé : c'est la toute première des architectures avioniques. Les applications embarquées dans les équipements sont gérées par un calculateur unique central. Les équipements passent par ce calculateur pour exécuter leurs fonctions.

Architecture fédérale : les équipements embarqués sont répartis dans l'avion, à proximité des capteurs et des actionneurs. Le but de cette répartition est de gagner en temps de communication entre ces équipements et les capteurs/actionneurs, et de réduire la longueur du câblage ainsi le poids.

Avionique Modulaire Intégrée (Integrated Modular Avionics IMA) : dernière génération d'architecture avionique, elle est utilisée sur les avions A380, A350 cette architecture propose une découpe du système avionique en plusieurs sous-systèmes et une structuration des divers sous-systèmes autour d'un réseau cœur (l'AFDX) de haute performance servant de relai d'intercommunication entre ces derniers.

I.6.Conclusion

Dans ce chapitre, nous avons étudié le contexte des systèmes avioniques, l'historique, une définition, quelques exemples de leurs fonctionnalités et enfin les différentes architectures qui se sont succédées.

II .1. Introduction

La Navigation est l'acte, ou la science de diriger le mouvement d'un aéronef, donc elle implique à la fois le contrôle de la trajectoire du vol et le guidage pour sa mission.

L'attitude de l'avion par rapport au plan horizontal en termes de tangage, de roulis et Lacet, est une mesure essentielle pour le contrôle et le guidage, et elle est très vitale pour le pilote afin de guider l'avion en toute sécurité dans toutes les conditions météorologiques, y compris celles où la visibilité normale de l'horizon est pauvre ou non disponibles.

L'information sur l'attitude et le cap est également essentielle pour les systèmes avioniques clés qui permettent à l'équipage d'effectuer la mission de l'avion, ces systèmes comprennent le système de pilotage automatique (par exemple, les modes Attitude & Heading Hold, Autoland, etc.) [4].

II.2.Les méthodes de navigation

Il existe deux méthodes de navigation de base, à savoir la navigation à l'estime (DR) et les systèmes de navigation pour la localisation de position

Les principaux types de systèmes de navigation DR aéroportés sont classés ci-dessous sur la base des moyens utilisés pour dériver les composantes de vitesse de l'aéronef.

Dans l'ordre d'exactitude croissante, ce sont [4] :

1. Navigation par DR basée sur les données de l'air. L'information de base utilisée comprend la vrai vitesse de l'air (à partir de l'ordinateur de données aériennes) avec la vitesse et la direction du vent (prévisions ou estimé) et le cap de l'Attitude Heading Reference Système, (AHRS).

2. Systèmes de référence Doppler / cap. Ceux-ci utilisent un capteur de vitesse radar Doppler système pour mesurer la vitesse au sol et l'angle de dérive de l'avion. Le cap de l'avion est fourni par l'AHRS.

3. Systèmes de navigation inertielle. Ceux-ci dérivent les composantes de la vitesse de l'avion intégrant les composantes horizontales de l'accélération de l'aéronef au temps. Ces

composants sont calculés à partir des sorties de très haute précision gyroscopes et accéléromètres qui mesurent le Mouvement angulaire et linéaire.

4. Systèmes de navigation inertielle Doppler. Ceux-ci combinent les sorties Doppler et INS, généralement au moyen d'un filtre de Kalman, pour augmenter la précision de la navigation DR.

Donc La navigation aérienne est accomplie par diverses méthodes, la méthode ou le système qu'un pilote utilise pour naviguer dans le système d'espace aérien actuel dépendra du type de vol.

Cela se produira (VFR ou IFR), quels systèmes de navigation sont installés sur l'aéronef et quels systèmes de navigation sont disponibles dans une certaine zone.

Donc on distingue deux régimes du vol en commençant par le VFR :

II.2.1 Navigation à vue VFR

VFR (règle de vol à vue) La navigation à vue est pratiquée depuis les origines de l'aéronautique et reste encore le moyen le plus utilisé par l'aviation légère. Le pilote connaît sa position en cherchant au sol des repères qui figurent sur sa carte. Il suit une trajectoire en se déplaçant d'un point de repère à l'autre, ou même en suivant un repère continu tel qu'une autoroute ou une rivière [4].

La navigation à vue ne nécessite aucun instrument mais elle n'est praticable que lorsque les conditions météorologiques VMC en vigueur dans la classe d'espace aérien sont réunies. Cependant, dans les espaces contrôlés, le contrôleur peut délivrer une autorisation de VFR spécial qui permet de voler hors conditions VMC.

Selon cette méthode (VFR), il existe 03 formes de navigation de base [6] :

- ✓ **Le cheminement :** Cheminer consiste à suivre les lignes naturelles caractéristiques bien visibles depuis un avion. Cette méthode peut être utilisée chaque fois qu'une partie du parcours amène à longer un repère naturel ou artificiel (rivière importante) pendant un certain temps. Il est important de choisir de bons repères, facilement visibles et reconnaissables, comme les fleuves, les autoroutes, les côtes, les voies ferrées importantes.

On appelle aussi cheminement le fait de se diriger, à vue, d'un point connu à un repère identifié, puis de celui-ci à un autre repère identifié. On peut cheminer de VOR en VOR, en utilisant les moyens de radionavigation.

- ✓ **Navigation à l'estime :** Le principe de l'estime est simple, connaissant une position de départ, il s'agit de déterminer le cap à prendre et l'Heure Estimée d'Arrivée (HEA) pour arriver sur un point caractéristique ou sur un aérodrome.
  L'estime est la technique de navigation adaptée lorsque l'on souhaite joindre deux points par le trajet le plus direct la ligne droite.

- ✓ **Navigation par erreur systématique :** La Combinaison des deux précédentes méthodes. Elle consiste à naviguer à l'estime en direction d'un repère facilement "cheminable", mais très en amont (erreur Systématique) du repère que l'on souhaite réellement atteindre, la destination par exemple. Il suffit alors de cheminer le long du premier repère, l'erreur systématique permet de connaître à coup sûr la direction à prendre à partir de ce premier repère remarquable.

II.2.2 .Navigation aux instruments IFR

IFR (règle de vol instrumentale) le pilote ne s'appuie que sur les instruments aéroportés, en raison de (météo, nuit, altitude etc..).

Selon cette méthode, il existe 03 formes de navigation de base [6] :

- ✓ **Radionavigation :** La radionavigation est un type de navigation utilisant les propriétés des ondes Radioélectriques. Elle nécessite des équipements extérieurs à l'avion, généralement des émetteurs placés au sol, et des équipements embarqués à bord de l'aéronef, des récepteurs « intelligents » à disposition du pilote.
  La radionavigation permet de définir ou confirmer une position et de réaliser une navigation sans avoir besoin de référence visuelle.

- ✓ **Navigation inertielle :** Repose sur la connaissance de la position initiale, la vitesse, et l'attitude et par la suite mesurant les taux d'attitude et accélérations. Le fonctionnement des systèmes de navigation par inertie (INS) dépend des lois de Newton de la mécanique classique. C'est la seul forme de navigation qui ne s'appuie pas sur des références externes.

II.3. Le système de Navigation Inertielle

Une centrale à inertie ou centrale inertielle est un instrument utilisé en navigation, capable d'intégrer les mouvements d'un mobile (accélération et vitesse angulaire) pour estimer son orientation (angles de roulis, de tangage et de lacet), sa vitesse linéaire et sa position. L'estimation de position est relative au point de départ ou au dernier point de recalage.

Les désignations habituelles pour une centrale inertielle sont Inertial Reference System (IRS), Inertial Navigation System (INS) et Unité de Mesure Inertielle (IMU).

La centrale à inertie n'utilise aucune information extérieure au mobile. Malgré les progrès spectaculaires des systèmes de positionnement par satellites, elle reste utilisée sur les avions de ligne dont la sécurité des vols ne peut reposer uniquement sur le GPS, insuffisamment fiable. Généralement, les véhicules militaires par exemple en sont également équipés pour pallier les brouillages du GPS susceptibles d'être rencontrés en temps de guerre.

Il est instructif d'examiner brièvement les raisons du développement de la navigation inertielle et son importance en tant que capteur d'état de l'aéronef [4].

Les attributs d'un système de navigation et de guidage idéal peuvent être résumés comme suit:

• Haute précision.

• Autonome (ne dépend pas d'autres systèmes).

• Passif (ne rayonne pas).

À la fin des années 1940, ces attributs constituaient une «liste de souhaits» et indiquaient le développement de la navigation inertielle comme seul système capable de répondre à toutes ces exigences. Il a donc été initialement développé au début des années 1950 pour la navigation et le guidage de missiles balistiques, de bombardiers stratégiques, de navires et de sous-marins (Système de navigation inertielle des navires, SINS) [4].

D'énormes programmes de recherche et de développement ont été menés dans le monde entier impliquant plusieurs milliards de dollars de dépenses pour réaliser des systèmes viables.

Les accéléromètres de précision ont dû être développés avec des incertitudes de biais de moins de 50 μg. La tâche principale d'atteindre les précisions de calcul requises devait être résolue et en fait les premiers ordinateurs numériques fonctionnant en temps réel ont été développés pour les systèmes IN.

Une fois ces problèmes résolus, l'INS peut fournir:

- ✓ Position précise dans toutes les coordonnées requises (par ex. latitude/ longitude).
- ✓ Vitesse au sol et angle de piste.
- ✓ Angles d'Euler: position, tangage et roulis à très haute précision.

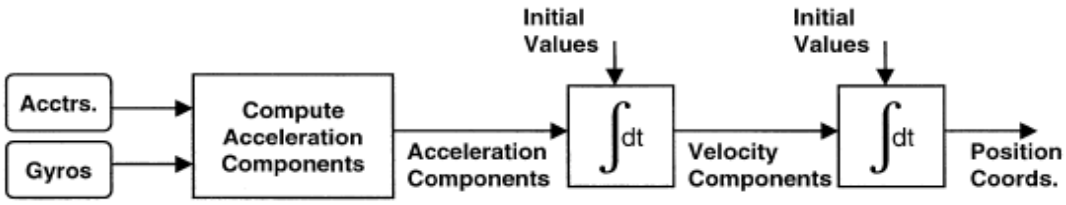


Fig.II.1 Le principe de base de la navigation inertielle [4].

Les caractéristiques autonomes d'un système de navigation inertielle plus la capacité pour fournir une attitude très précise et une référence verticale précise, a conduit à l'installation de Systèmes IN dans les avions de transport civil à longue portée à la fin des années 1960. Ils sont maintenant très largement utilisés dans tous les types d'avions civils [4].

La navigation par inertie utilise plusieurs cadres de référence, qui sont montrés ci-dessous.

II.3. 1 Repères Utilisée Dans La Navigation Inertielle

Pour la navigation, des tétraèdres sont utilisés afin de représenter un système d'axes orthogonaux. Ces axes se trouvent à l'origine du système de coordonnées. Il existe plusieurs repères différents autour desquelles les équations de navigation peuvent être développées. Les principaux sont maintenant présentés.

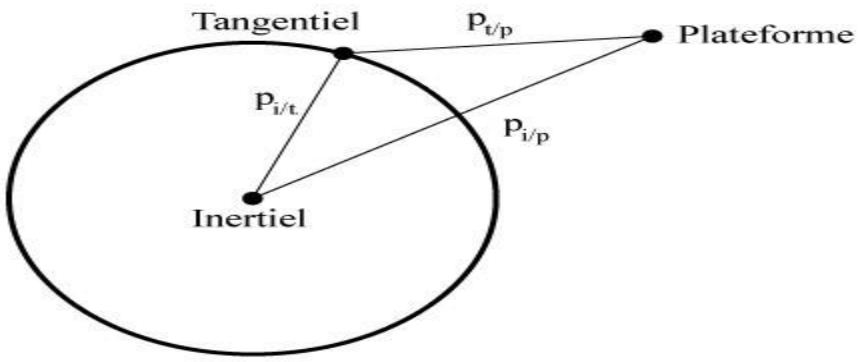


Fig.II.2 : Les positions des repères de navigation.

La Figure II.2 exprime les différentes positions des repères de navigation ainsi que leurs notations. Dans cette figure, Pt/*P* désigne la position du repère de la plateforme par rapport au repère tangentiel (NED-North East Down), P*i/t* désigne la position du repère tangentiel par rapport au repère inertiel et P*i*/*P désigne* la position du repère de la plateforme par rapport au repère inertiel. En général, on considère que le repère inertiel ne bouge pas [7].

II.3.1.1. Repère ECI (« Earth Centered Inertial ») ECI

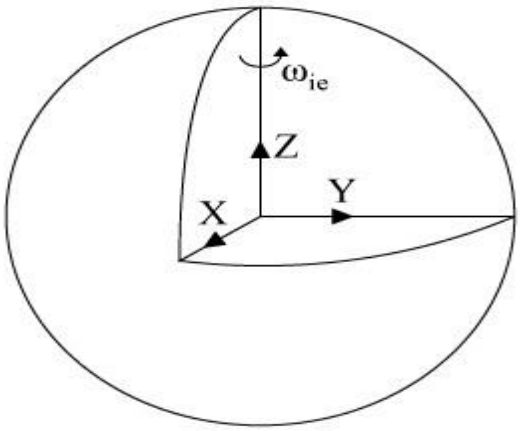


Fig II.3 : Repère ECI [10]

L'ECI est un repère inertiel. Ce repère de référence est placé au centre de la Terre et ne tourne pas avec la Terre. Comme illustré par la Figure II.3, l'axe z s'aligne avec l'axe de rotation de la Terre [7].

II.3.1.2. ECEF («Earth-Centered, Earth-Fixed»)

Le repère ECEF se retrouve au centre de la Terre à la même position que le repère ECI. Cependant, ce repère tourne avec la terre. Son axe x est aligné avec le premier méridien (Greenwich) sur le plan équatorial et son axe z est dans le même axe que celui de la rotation de la Terre.» [7]. ECEF est défini ici par l'indice e. Ce repère tourne autour de l'axe z à la même vitesse que la Terre. Cette vitesse est déterminée par l'équation suivante :

$$\omega i/e = 7.2921 * 10^{-5}\ rad/s \quad [10] \quad ; \quad \omega_{i/e}^{i} = [0 \quad 0 \quad \omega_{i/e}] \quad [10]$$

Où $\omega_{i/e}^{i}$ est la vitesse angulaire du repère « e » par rapport au repère « i » exprimé dans le repère i.

II.3. 1.3. NED (Tangentiel)

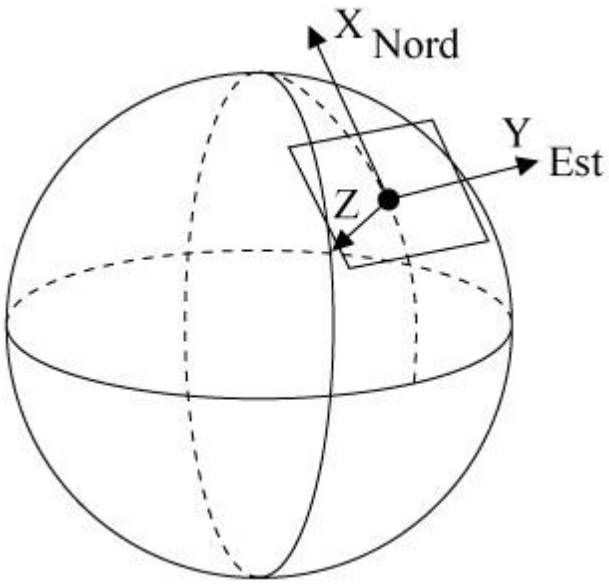


Fig II.4 : Repère tangentiel [10].

Tel qu'indiqué sur la Figure II.4, l'origine du repère tangentiel est située à la surface de la Terre autour d'un point de référence. Son axe z pointe vers le centre de la Terre alors que l'axe x pointe vers le nord géographique. Ce repère est équivalent au repère NED («North East Down»). Il tourne conjointement avec la Terre. Le repère tangentiel est défini ici par l'indice t.

II.3. 1.4. Repère mobile (Plateforme)

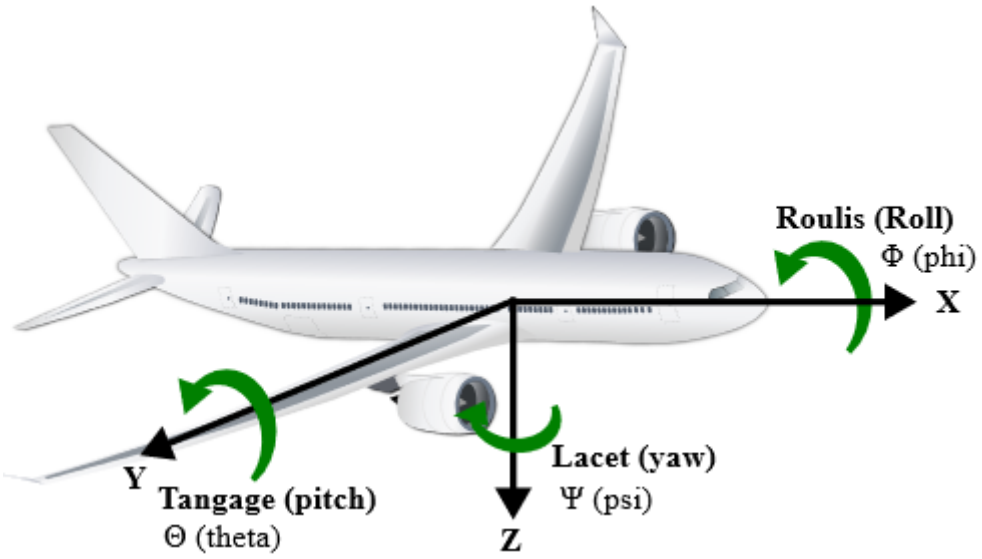


Fig II.5 : Repère mobile.

Comme son nom l'indique, le repère de la plateforme est fixé à une plateforme mobile qui est un avion (voir la Figure II.5). Le repère se déplace donc avec la plateforme. L'axe x pointe vers le devant du véhicule. L'attitude d'un véhicule est souvent définie avec le roulis (rotation autour de l'axe x), le tangage (rotation autour de l'axe y) et le lacet (rotation autour de l'axe z). Ces angles peuvent aussi être désignés par une des conventions des angles d'Euler.

Le repère de la plateforme sera défini par l'indice b.

II.3.2. Représentation d'attitude

L'attitude d'un corps rigide dans l'espace peut être représentée avec différentes paramétrisations, chacune ayant ses avantages et inconvénients. En conséquence, il est nécessaire, compte tenu de l'application traitée, de choisir une représentation adéquate de l'attitude. On résume ci-dessous les principales représentations d'attitude utilisées ici [12].

- **Matrice de rotation**

  Une matrice de rotation représente l'orientation tridimensionnelle sous forme matricielle. Elle est composée de neuf paramètres représentés dans une matrice 3 par 3.

  La matrice possède jusqu'à trois degrés de liberté; c'est-à-dire que parmi ses neuf paramètres, il y En a tout au plus trois qui sont indépendants. Cette matrice s'appelle aussi DCM («Direction Cosine Matrix») [8].

$$\boldsymbol{R} = \begin{bmatrix} R_{11} & R_{12} & R13 \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

La matrice de rotation respecte les propriétés suivantes :

- Son déterminant est toujours égal à un,
- Elle est orthogonale : $\boldsymbol{R}^{-1} = \boldsymbol{R}^{T}$
- La norme de chacune de ses colonnes est unitaire.

- **Changement de repère à l'aide des matrices de rotation**

  Une matrice de rotation contient l'information sur l'orientation, elle permet donc de projeter un vecteur d'un repère à l'autre. Par exemple, la matrice $\boldsymbol{R}_b^a$ indique l'orientation du repère **b** par rapport au repère **a**, mais elle permet aussi de projeter un vecteur **v** exprimé dans le repère **b** sur le repère a de la façon suivante :

$$\boldsymbol{V^a = R_b^a V^b}$$

Cette projection peut également être inversée en inversant la matrice :

$$\boldsymbol{R_b^{aT} = R_b^{a^{-1}} = R_a^b}$$

- **Angles d'Euler**

  Les angles d'Euler sont les plus utilisés en aéronautique. Ils ont comme principaux avantages d'être faciles à lire et à comprendre et de ne comporter que trois paramètres pour décrire une orientation. Quoi qu'il existe douze conventions possibles pour ces axes, celle utilisée dans ce mémoire, qui se nomme Roulis-Tangage-Lacet, considère les axes z, y et x associés aux trois rotations suivantes:

  - Roulis (φ) représente une rotation autour de l'axe x;
  - Tangage (θ) représente une rotation autour de l'axe y;
  - Lacet (ψ) représente une rotation autour de l'axe z.

Cette convention respecte la règle de la main droite. Par exemple,

- une rotation autour de z de x vers y est positive;
- une rotation autour de y de z vers x est positive;
- une rotation autour de x de y vers z est positive.

La représentation de l'orientation par les angles d'Euler est couramment utilisée puisqu'elle possède une interprétation physique. Cependant, puisqu'il s'agit de trois rotations, les angles d'Euler sont très propices au phénomène de blocage de cadran (Gimbal lock) communément appelé singularité de représentation à des angles de tangage de ± 90°.

- **Quaternion**

  Pour s'affranchir du phénomène de blocage de cadran (Gimbal lock) on peut utiliser les quaternions unitaires (ou paramètres d'Euler) qui représentent une alternative aux angles d'Euler. Les quaternions s'appuient sur une représentation de l'orientation à l'aide d'une rotation δ autour de l'axe d'un vecteur unitaire *k*. Les quatre paramètres sont définis par une fonction des trois composantes du vecteur *k* et de la grandeur de

  La rotation δ.

$$\mathbf{q} = \begin{bmatrix} \eta \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} = \begin{bmatrix} \cos(\delta/2) \\ k_x \sin(\delta/2) \\ k_y \sin(\delta/2) \\ k_z \sin(\delta/2) \end{bmatrix}$$

Un des avantages de la représentation de l'orientation par un quaternion unitaire est son conditionnement numérique qui est particulièrement bon du fait que sa norme est toujours unitaire :

$$\eta^2 + \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon} = 1$$

- **Alignement d'attitude**

L'alignement d'attitude comporte deux étapes. Tout d'abord, la plate-forme est nivelée par l'initialisation des angles de roulis (Φ) et de tangage (θ), puis le compas gyroscopique fournit une première valeur du cap (également connu sous le nom d'angle de lacet (ψ), ou azimut).

- **Modèle de gravité**

L'attraction gravitationnelle n'est pas constante selon la position sur Terre. En effet, cette force dépend de la latitude (et du rayon de la Terre qui ne sont pas constants puisque la Terre n'est pas parfaitement ronde, mais plutôt ovale). Il faut donc avoir recours à un modèle de gravité afin d'estimer sa valeur avec plus de précision. Le modèle de gravité utilisé est défini par le WGS84 (World Geodetic System) [10] Selon la latitude sur le géoïde terrestre, il est donné par

$$\mathrm{g}(\phi) = 9.7803267714 \left(\frac{1 + 0.00193185138639\, sin^2(\phi)}{\sqrt{1 - 0.00669437999013\, sin^2(\phi)}} \right) \frac{m}{s^2}$$

II.4. Accéléromètres et gyroscopes

II.4.1 Accéléromètre

L'accéléromètre mesure directement les accélérations de la plateforme par rapport à un repère inertiel. Cependant, sur Terre, les forces gravitationnelles, centripètes et de Coriolis doivent être compensées.

- **L'effet de Coriolis** : G.G. de Coriolis a montré qu'un point matériel auquel on applique une rotation et une translation est soumis à une accélération due à la combinaison de ces deux mouvements. On peut exprimer cette accélération par le produit vectoriel de la vitesse angulaire Ω de rotation et de la vitesse de translation *v*.

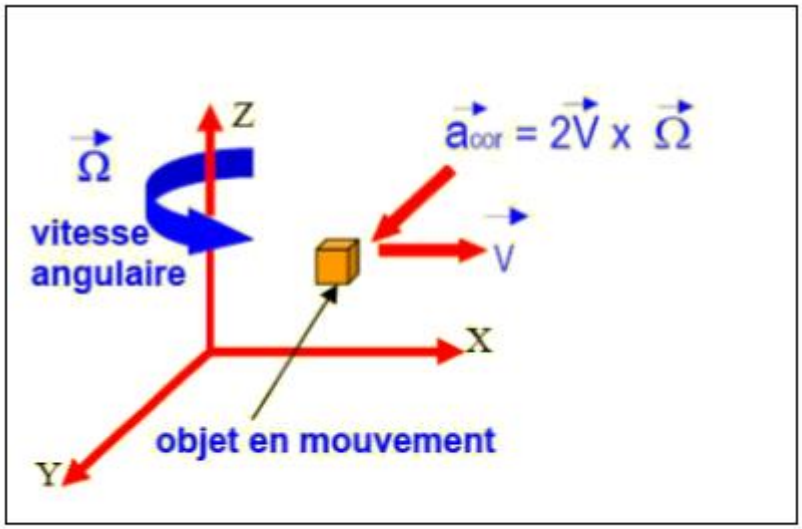


Fig II.6 : Concept de l'accélération de Coriolis.

V : la vitesse linéaire.

Le principe de base d'un accéléromètre est une masse, dont le mouvement est restreint par un ressort ou des ressorts, dans une ou plusieurs dimensions. Lorsque le capteur subit une accélération, la masse cherche à rester au repos et déforme le ou les ressorts. Cette déformation peut être mesurée pour obtenir l'accélération. L'accéléromètre mesure une accélération souvent appelée force spécifique ou accélération propre. Une force spécifique se décrit comme une accélération non gravitationnelle. Il s'agit de l'accélération par rapport à une chute libre. Cette force spécifique contient donc l'accélération de la plateforme de laquelle doit être soustraite l'accélération gravitationnelle ainsi que l'accélération centripète causée à la rotation de la Terre. Cette mesure est par rapport à un repère inertiel [8].

Cette force spécifique se décrit comme suit [10]:

$$\boldsymbol{f}^{p}_{i/p}=\boldsymbol{a}^{p}_{i/p}-\boldsymbol{g}^{p}_{i/p}-\boldsymbol{\Omega}^{t}_{i/t}\boldsymbol{\Omega}^{t}_{i/t}\boldsymbol{p}^{t}_{t/p}$$

Où $\boldsymbol{a}_{i/p}^{p}$ est l'accélération de la plateforme par rapport au repère inertiel exprimée dans le repère de la plateforme, $\boldsymbol{g}_{i/p}^{p}$est l'accélération gravitationnelle exprimée dans le repère de la plateforme (cette valeur est variable selon la position sur terre) et $\boldsymbol{\Omega}_{i/t}^{t}\boldsymbol{\Omega}_{i/t}^{t}\boldsymbol{p}_{t/p}^{t}$ est l'accélération centripète liée à la rotation de la Terre. L'accélération centripète peut se combiner avec la force de gravité pour donner le vecteur de gravité local (Figure II.7) :

$$\check{\mathrm{g}}_{i/p}^{t} = \boldsymbol{g}_{i/p}^{t} - \boldsymbol{\Omega}_{i/t}^{t}\boldsymbol{\Omega}_{i/t}^{t}\boldsymbol{p}_{t/p}^{p}$$

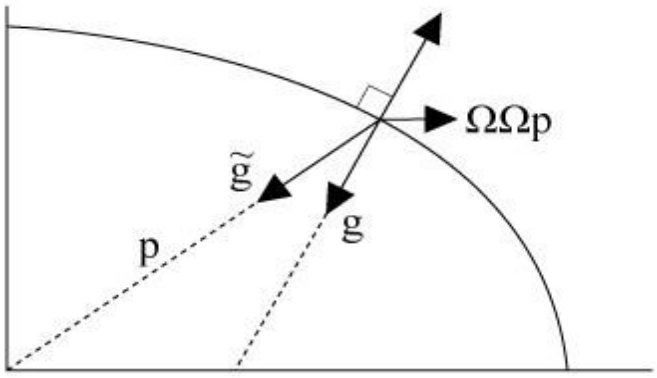


Fig II.7 : Vecteur de gravité local [10].

II.4.2 Gyroscopes

Les gyroscopes d'une centrale inertielle sont des capteurs essentiels pour la mesure de l'orientation [9]. Les termes gyromètres et gyroscopes sont souvent confondus. Ils désignent un capteur mesurant les vitesses de rotation par rapport à un repère inertiel. Puisque la navigation s'effectue sur Terre, les vitesses de rotation provenant du gyromètre doivent être compensées pour la rotation de la Terre afin de retrouver la vitesse de rotation réelle du véhicule. Le repère tangentiel (NED) est fixe sur la Terre et tourne avec cette dernière. La rotation de ce repère est donc la même que celle de la Terre. Par conséquent :

$$\boldsymbol{\omega}_{i/e}=\boldsymbol{\omega}_{i/t}$$

Ainsi, la vitesse angulaire de la plateforme par rapport au repère tangentiel est définie par :

$$\omega_{t/p}^{p} = \omega_{i/p}^{p} - R_{t}^{p}\omega_{i/e}^{t}$$

Où $\omega_{i/p}^{p}$: est la mesure provenant des gyromètres et $\omega_{i/e}^{t}$ est la vitesse angulaire de la Terre. Les gyromètres offrent une mesure de vitesse angulaire. Afin de retrouver l'orientation dans l'espace de la plateforme, il faut projeter les vitesses angulaires et les propager dans le temps pour ensuite procéder à leurs intégrations. En intégrant, l'orientation est obtenue selon la représentation choisie. Ce résultat décrit la rotation actuelle de la plateforme par rapport au repère tangentiel.

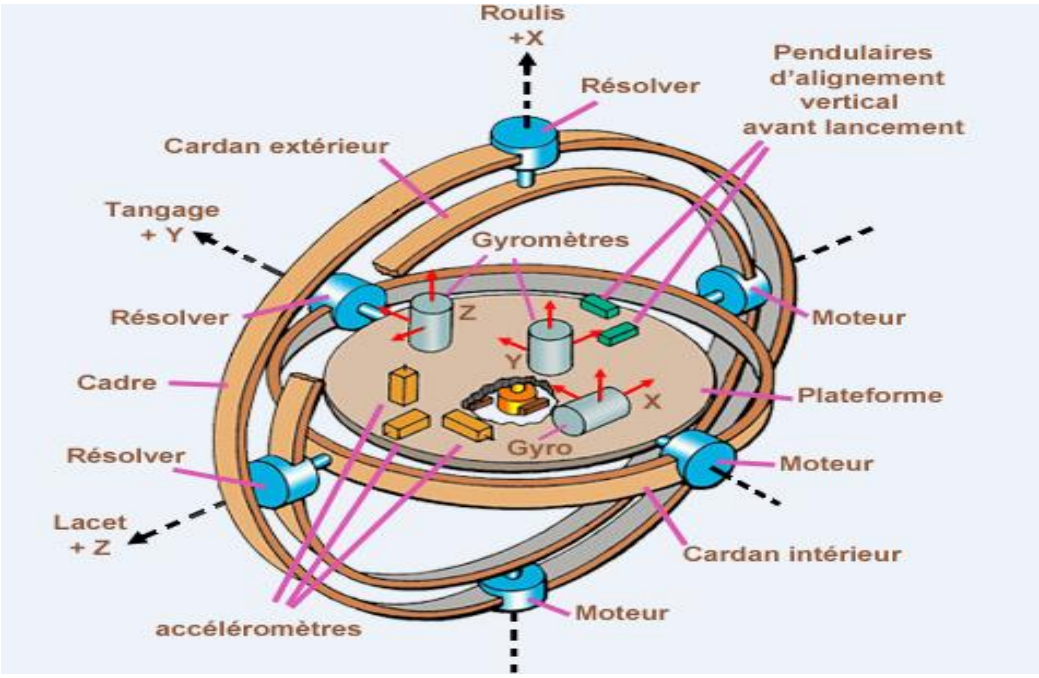


Fig II.8 : Schéma de principe d'un Gyroscope.

II.5. Les Erreurs des capteurs Inertiels

Les systèmes inertiels sont affectés par plusieurs erreurs de navigation qui peuvent se présenter en trois sources principales :

Erreurs liées aux capteurs : les erreurs des capteurs inertiels sont de deux types. Le premier type concerne les erreurs d'instrumentation dont les variables captées ne peuvent pas être égales aux vraies quantités physiques à cause des imperfections dans les capteurs. Le deuxième type d'erreur concerne l'alignement des capteurs et leur plate-forme avec leur direction désirée.

Erreurs de calcul : les équations de navigation sont typiquement implantées par un ordinateur digital, et donc des erreurs de quantifications, de saturations et des erreurs numériques (ex. Intégration) peuvent subvenir. Les effets des erreurs de calcul sont souvent modélisés comme un bruit additif dans le modèle du système.

Erreurs de l'environnement : l'environnement ne peut pas être modélisé exactement alors qu'une compensation des mesures est exigée. Les capteurs inertiels sont sujets à diverses erreurs qui deviennent plus complexes en fonction de la mauvaise qualité, Elles sont classées selon deux grandes catégories, des erreurs systématique et stochastique (ou aléatoire), dans la section suivantes on va s'intéresser aux plus importantes.

II.5.1. Erreurs systématiques

- **Biais (systématique Bias offset)**

Le biais est une valeur additionnelle qui est déterminée en appliquant une entrée nulle. Néanmoins, il arrive qu'il soit différent à chaque fois que le capteur est remis en marche. Par ailleurs, sa valeur n'est pas fixe au cours du temps. Ceci est dû aux variations de la température. Ainsi, la relation obtenue est : Mesurée = vraie + biais

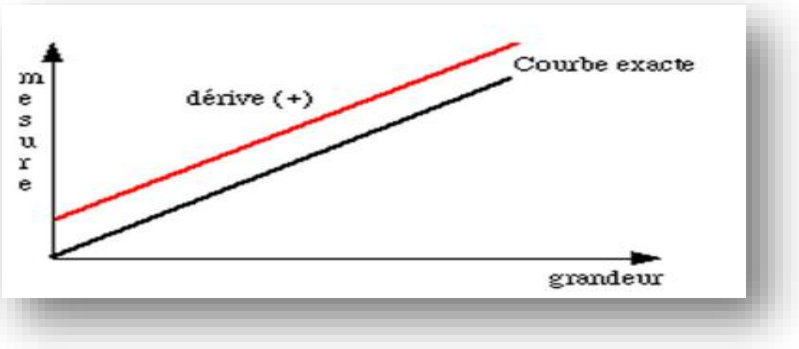


Fig II.9 : Le biais d'un capteur inertiel.

- **Facteur d'échelle**

Il s'agit de la déviation de la pente d'entrée-sortie de l'unité. L'erreur de la sortie d'un accéléromètre qui est due à une erreur du facteur d'échelle est proportionnelle à la véritable force spécifique le long de l'axe sensible, alors que l'erreur de sortie d'un gyroscope provoqué par une erreur du facteur d'échelle est proportionnelle à la vitesse angulaire réelle de l'axe sensible.

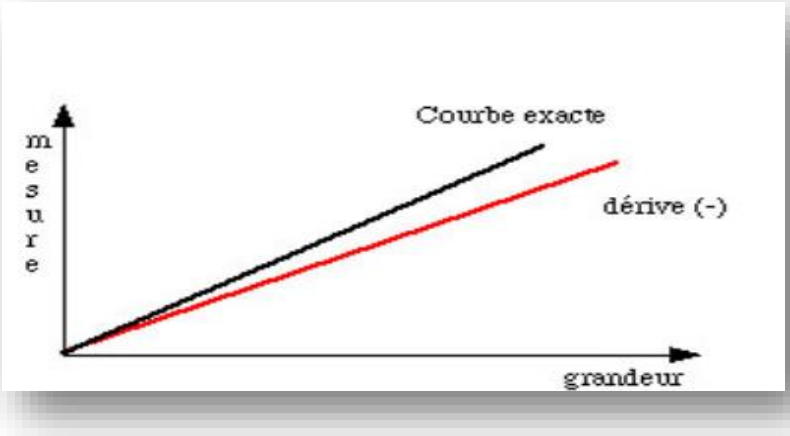


Fig II.10 : L'erreur du facteur d'échelle d'un capteur inertiel.

II.5.2. Erreurs aléatoire

Les capteurs inertiels sont exposés à un certain nombre d'erreurs aléatoires :

- Run-to-Run Bias Offset.
- Erreur de dérive.
- Instabilité du facteur d'échelle.
- Bruit blanc.
- Bruit de mesure.

II.6. Caractéristiques de Performance de Capteur Inertielles

Pour évaluer un capteur inertiel en vue d'une application particulière, de nombreuses caractéristiques doivent être prise en considération.

- **La répétabilité :** la capacité d'un capteur à fournir la même sortie pour les applications répétées à la même entrée, en supposant que tous les autres facteurs de l'environnement restent constants. Il se réfère à la variation maximale entre les mesures répétées dans les mêmes conditions sur plusieurs termes.
- **La stabilité :** il s'agit de la capacité d'un capteur pour fournir la même sortie lorsque les mesures d'une entrée sont constantes pendant une durée du temps. elle est définie pour un seul terme.
- **La dérive :** le terme dérive est souvent utilisée pour décrire le changement qui se produit dans les mesures d'un capteur quand il n'y a aucun changement de l'entrée. Il est également utilisé pour décrire le changement qui se produit quand l'entrée est à zéro.

II.7. Le principe de base de la navigation inertielle

Les systèmes de navigation inertielle sont des systèmes entièrement autonomes permettant de calculer la position, la vitesse et l'attitude d'un véhicule basé uniquement sur des mesures d'accélération linéaires et de vitesses angulaires provenant de capteurs inertiels (c.-à-d. gyroscopes et accéléromètres). Contrairement aux systèmes GPS, les systèmes INS ne sont pas des systèmes de positionnement absolu mais font plutôt parti de la grande famille des systèmes de navigation à l'estime (DR), c'est à dire qu'ils permettent uniquement de calculer le déplacement d'un mobile à partir de son état initial [13].

De manière générale, un système INS est constitué de deux principaux éléments soient une centrale inertielle (IMU – Inertial Measurement Unit) et un ordinateur de calculs.

L'IMU consiste en un assemblage d'accéléromètres et de gyroscopes disposés de manière à mesurer, respectivement l'accélération (plus précisément les forces spécifiques) et la vitesse angulaire.

La Figure II.11 illustre bien la structure typique d'un système INS à plateforme stabilisée.

D'une plateforme en mouvement selon trois axes orthogonaux. L'ordinateur de calculs quant à lui s'occupe d'exécuter les équations dynamiques du système inertiel de manière à calculer l'attitude, la vitesse et la position du mobile selon la séquence suivante :

1) le vecteur de vitesses angulaires mesuré par les gyroscopes est intégré afin de mettre à jour l'attitude du mobile.

2) la nouvelle attitude du mobile est utilisée afin de transformer la représentation du

Vecteur de forces spécifiques vers le repère de navigation.

3) les composantes de force d'attraction gravitationnelle, de force centripète et d'accélération de Coriolis sont retirées du vecteur de forces spécifiques afin d'obtenir uniquement l'accélération du mobile selon le repère de navigation.

4) l'accélération est intégrée afin d'obtenir la vitesse du mobile.

5) la vitesse est intégrée afin d'obtenir la position du mobile.

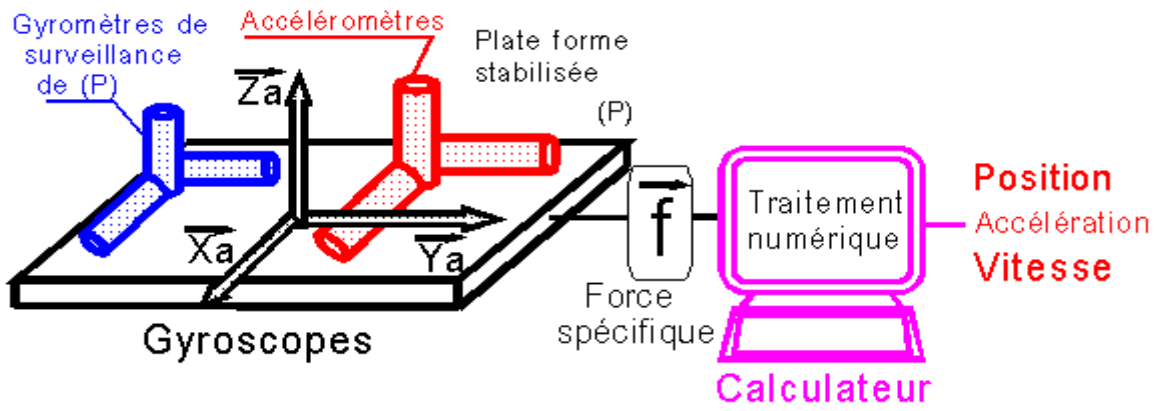


Fig II.11 : Structure du système INS a plateforme stabilisé.

Les angles de roulis (Roll), de tangage (Pitch) et de lacet (Yaw) sont évalués en fonction des équations suivantes:

$$P = arctan\left(\frac{Ay}{\sqrt{Ax^2+Az^2}}\right);$$ **P:** Angle d'Euler de rotation suivant l'axe Y (Tangage)

$$\Psi = arctan\left(\frac{\sqrt{Ax^2+Ay^2}}{Az}\right);$$ ` **Ψ :** Angle d'Euler de rotation suivant l'axe Z (Lacet)

$$\phi = arctan\left(\frac{Ax}{\sqrt{Ax^2+Ay^2}}\right);$$ **ϕ :** Angle d'Euler de rotation suivant l'axe X (Roulis)

II.8. Les principaux modules du système de navigation

- Une unité de mesure inertielle IMU contenant trois accéléromètres et trois gyroscopes.
- Une unité de prétraitement.
- Calculateur.

II.9. Implémentation physique du central inertiel

Il existe de nombreux modèles d'INS avec des performances et différentes Caractéristiques, mais ils tombent généralement dans deux catégories [14] :

- Centrale inertielle à plate-forme stabilisée (Gimballed System).
- Centrale inertielle à composants liés (Strapdown system).

II.9.1 Centrale inertielle à plate-forme stabilisée (Gimballed System IRS)

Les applications originales de la technologie INS utilisaient des techniques de plate-forme stables.

La centrale à inertie à cardan comporte :

- une plateforme stabilisée.
- trois gyroscopes.
- trois accéléromètres.

- **Principe de fonctionnement**

La plateforme est maintenue horizontale par rapport aux repères terrestres quels que soient les mouvements de l'avion. Ceci est réalisé en utilisant des cardans (cadres) qui permettent la liberté de la plate-forme dans les trois axes (lacet, roulis et tangage). Les gyroscopes montés sur la plate-forme détectent n'importe quelles rotations de la plate-forme. Ces signaux sont renvoyés à des moteurs couple qui tournent les cardans pour annuler ces rotations, ce qui maintient la plate-forme alignée avec le cadre global. Pour suivre l'orientation de l'appareil, les angles entre les cardans adjacents peuvent être lus à l'aide de capteurs d'angle.

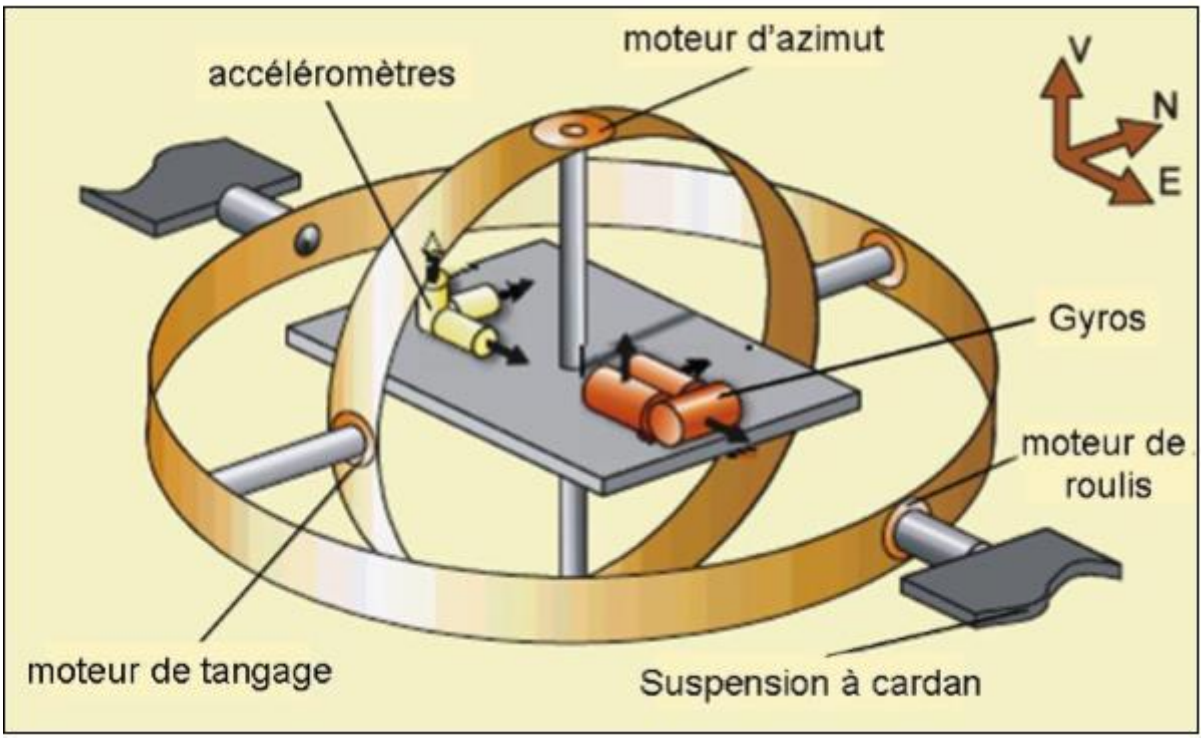


Fig II.12: Schéma de principe de Gimballed System [15].

II.9. 2 Centrale à inertie à composants liée (StrapDown INS)

Dans ce type de centrale liée ou Strapdown les accéléromètres sont fixés par rapport à la structure de l'aéronef, mais la vitesse de rotation est mesurée par trois gyromètres, le traitement du signal permet ensuite de faire les changements de positions.

- **Principe de fonctionnement**

Les trois gyromètres détectent le taux de roulis, de tangage et de lacet. Les trois accéléromètres détectent les accélérations le long de chaque axe de l'aéronef. Un calculateur les intègre et effectue tous les calculs pour satisfaire les six degrés de mouvement (orientation et accélération des axes nord / sud, est / ouest et haut / bas).

Comme pour la centrale stabilisée le calculateur connaît la valeur de la gravité terrestre en tout point et la retranche et calcule la force de Coriolis pour s'en affranchir.

- **Avantage la centrale liée**

Avec peu de pièces mobiles, elles sont plus faciles à entretenir et plus fiables au fil du temps. Elles nécessitent des gyroscopes plus précis et une plus grande puissance de calcul. Cependant, les avantages d'un coût, d'une taille, d'un poids et d'une fiabilité réduits font de ces systèmes le choix préféré des centrales pour les aéronefs.

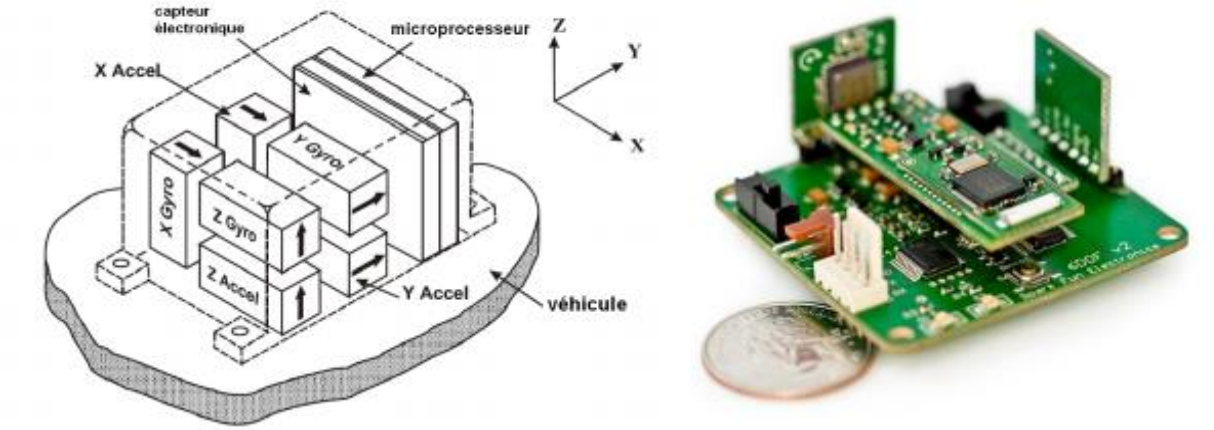


Fig.II.13: Configuration d'une central inertielle à composants lié : Un schéma simplifié (à gauche) et un montage commercial (à droite) [14].

II.10. Les systèmes micro-électromécaniques (MEMS)

Les Systèmes Micro-Electro-Mécaniques (MEMS) intègrent des éléments mécaniques, des capteurs, des actionneurs et des composants électroniques sur un substrat commun grâce à l'utilisation de la technologie de micro fabrication ou de la «micro-technologie» [5].

Les systèmes micro électromécaniques (MEMS) se réfèrent à une collection de microsecondes et d'actionneurs capables de détecter l'environnement et de réagir aux changements de cet environnement grâce à un contrôle par microcircuit.

Ils comprennent, en plus des emballages microélectroniques conventionnels, des structures d'antennes intégrant des signaux de commande dans des structures microélectromécaniques pour des fonctions de détection et d'actionnement souhaitées. Les microcomposants rendent le système plus rapide, plus fiable, moins cher et capable d'intégrer des fonctions plus complexes.

Au début des années 90, les MEMS ont vu le jour grâce au développement de procédés de fabrication de circuits intégrés dans lesquels les capteurs, les actionneurs et les fonctions de commande sont fabriqués en silicium.

Depuis lors, des progrès remarquables en matière de recherche ont été réalisés dans le domaine des MEMS dans le cadre des importantes promotions des capitaux du gouvernement et des industries.

Outre la commercialisation de certains dispositifs MEMS moins intégrés, tels que les micro-accéléromètres, la tête d'impression à jet d'encre, les micro miroirs de projection, etc., les concepts et la faisabilité de dispositifs MEMS plus complexes ont été proposés et démontrés pour des applications dans des domaines aussi variés comme micro fluidique, aérospatiale, biomédicale, analyse chimique, stockage de données de communication sans fil, affichage, optique, etc.

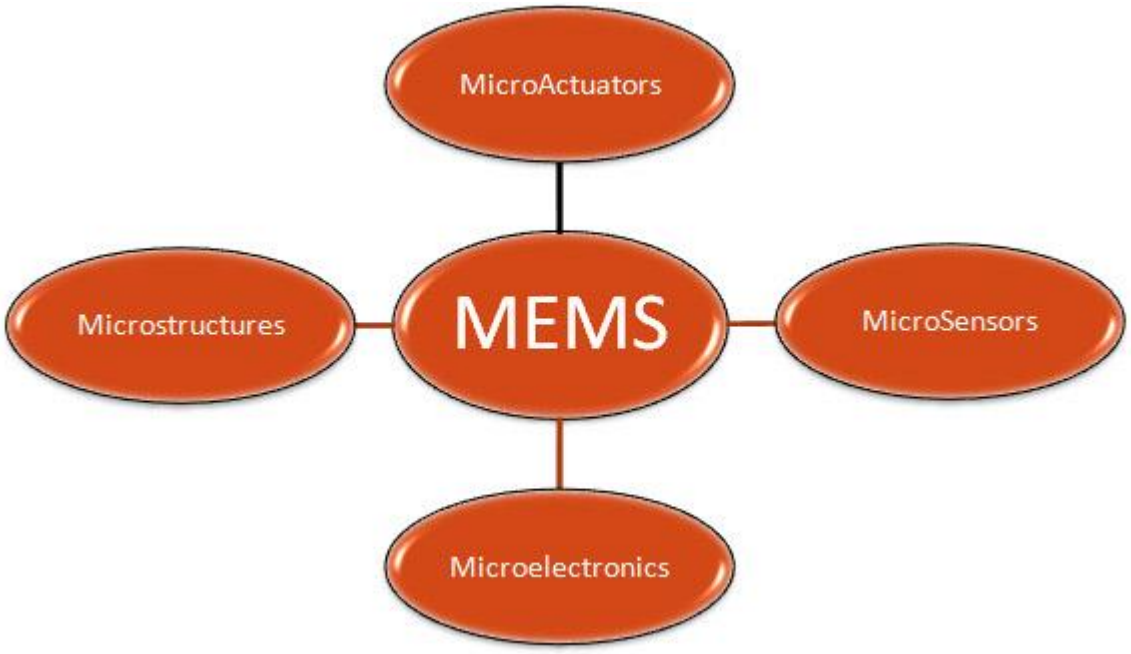


Fig II.14: Les composants des MEMS.

Les mécanismes physiques sous-jacents aux accéléromètres MEMS comprennent les accéléromètres capacitifs, piézo-résistifs, électromagnétiques, piézoélectriques, ferroélectriques, optiques et à effet tunnel.

Les types les plus performants sont basés sur la transduction capacitive grâce à la simplicité de l'élément de capteur, sa petite taille, sa faible consommation d'énergie et sa stabilité sur une large plage de température.

Le premier accéléromètre micro-usiné a été conçu en 1979 à l'université de Stanford, mais il a fallu plus de 15 ans avant que de tels dispositifs ne deviennent des produits courants pour des applications à grand volume.

Dans les années 1990, les accéléromètres MEMS ont révolutionné l'industrie des systèmes d'airbags pour automobiles, depuis lors ils ont permis des fonctionnalités et des applications uniques allant de la protection du disque dur sur les ordinateurs portables aux contrôleurs de jeux. Plus récemment, la même technologie de capteur-noyau est devenue disponible dans des dispositifs entièrement intégrés et complets convenant aux applications industrielles, les accéléromètres micro-usinés sont une technologie hautement habilitante avec un énorme potentiel commercial. Ils fournissent une puissance plus faible, une détection compacte et robuste. Plusieurs

Capteurs sont souvent combinés pour fournir une détection multi-axes et des données plus précises. Les accéléromètres sont intégrés dans de plus en plus d'appareils

Électroniques personnels, tels que les lecteurs multimédias et les appareils de jeu, et Ont également trouvé des applications en temps réel pour le contrôle et la Surveillance des systèmes militaires et aérospatiaux. L'avantage que les MEMS peuvent apporter est lié à l'intégration du système. Au lieu d'avoir une série de composants externes (capteur, inducteur…) connectés par fil ou soudés à une carte de circuit imprimé.

II.11. Conclusion

Au terme de cette étude théorique, nous avons donné une vision générale sur la centrale inertielle et sa constitution et son principe de fonctionnement. Nous avons abordé aussi le principe des Micro-Electro-Mécaniques Systèmes MEMS. Le chapitre suivant sera alors dédiée à la réalisation et la simulation.

III.1.Introduction

Dans ce chapitre on va présenter un prototype de système de navigation inertielle, les principes de base des capteurs IMU (unité de mesure de inertielle), le microcontrôleur utilisé, et un bref résumé sur l'interface entre le microcontrôleur et le meilleur capteur IMU disponible, ainsi que logiciel de simulation.

Ensuite, nous nous intéressons à la description des différentes étapes suivies pour la réalisation et simulation du projet.

III.2.Présentation de l'IMU

Une unité de mesure inertielle (IMU) est la combinaison de plusieurs capteurs inertiels tels que les accéléromètres et les gyroscopes comme le montre la figure III.1 suivante :

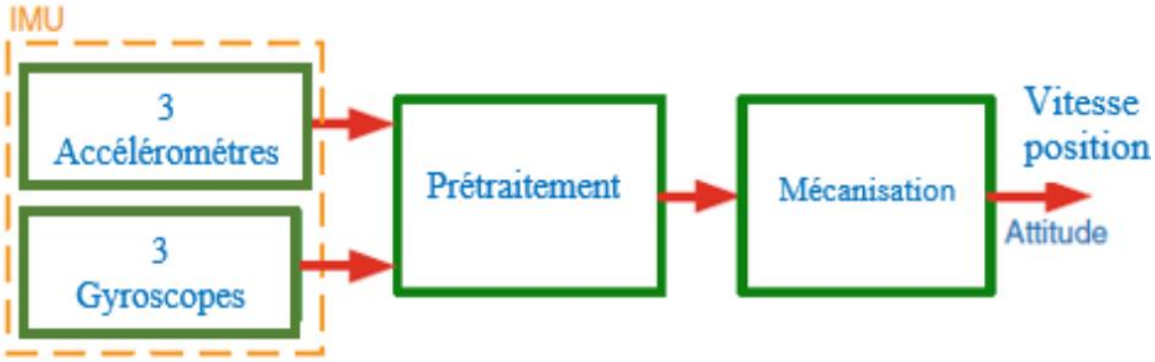


Fig.III.1 : Structure général du système de navigation inertielle.

Son principe de fonctionnement est basé sur la détection des changements de position et d'orientation. Ceux-ci peuvent être utilisés comme une entrée pour la représentation d'orientation d'un contrôle d'attitude tel que le roulis, le tangage et lacet.

Le positionnement géométrique des capteurs d'attitude sont montrée sur la figure III.2.

Ces composants d'attitude sont positionnés géométriquement pour fournir X, Y et Z mesures basées sur les coordonnées, respectivement Roulis, Tangage, Lacet.

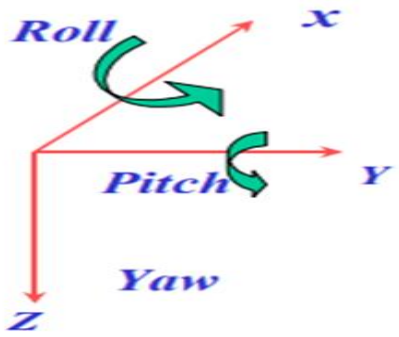


Fig.III.2 le trièdre de l'IMU.

En résumé, les progrès dans le développement des accéléromètres, des gyroscopes et des IMU dépendront vraisemblablement l'attraction du marché qui est entièrement motivée par des applications civiles. Nouvelle encapsulation, emballage et intégration des techniques seront cependant nécessaires pour répondre aux exigences des futures applications.

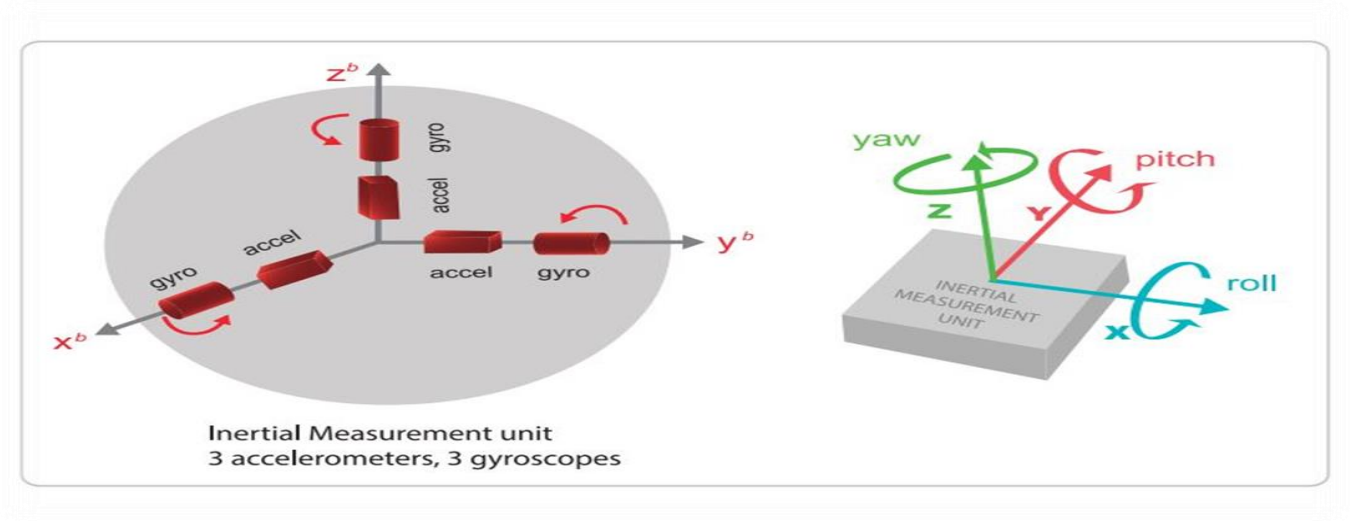


Fig.III.3 : Composants d'une unité de mesure inertielle.

Les capteurs IMU ont donc un nombre prolifique d'applications. Il est même considéré comme une composante inexorable dans les quadrotors. Certains des capteurs sur lesquels on a pu mettre la main était :

1. Accéléromètre ADXL345.
2. ITG 3200 gyroscope.
3. Carte capteur Sparkfun 6 DOF IMU.
4. MPU6050 Invensense.

On était capable de travailler avec les accéléromètres et les gyroscopes séparément, Cependant, ils ne sont pas aussi précis que lorsqu'ils sont combinés. Et parmi le lot, On a trouvé le MPU 6050 Invensense qui peut être le plus fiable et considérer comme un capteur IMU précis.

III.3. MPU (Microprocessor Unit)

Le capteur que nous allons utiliser porte la référence GY-521 architecturé autour d'un MPU 6050 et composé de deux capteurs et un processeur (voire figure III.4) :

- ✓ Un capteur accéléromètre 3 axes (x, y et z) qui mesure l'accélération.
- ✓ un capteur gyroscope 3 axes qui mesure la vitesse angulaire.
- ✓ Digital Motion Processor (DMP) Processeur de mouvement Digital capable de stocker des données et les restituer.

Le MPU-6050 est le premier dispositif de suivi de mouvement en 3D intégré au monde, conçu pour répondre aux exigences de faible consommation, de faible coût et de haute performance des smartphones, tablettes et capteurs portables. Il combine sur la même puce de silicium un gyroscope et un accéléromètre pour chaque axe et un Processeur de mouvement interne DMP "Digital Motion Processor " comme le montre la figure III.5. Cette combinaison de capteurs est souvent appelée «unité de mesure inertielle» utilisée dans les avions, les engins spatiaux et d'autres appareils.

Le DMP peut effectuer des calculs rapides directement sur la puce ou bien d'autre puces. Cela réduit la charge du microcontrôleur .

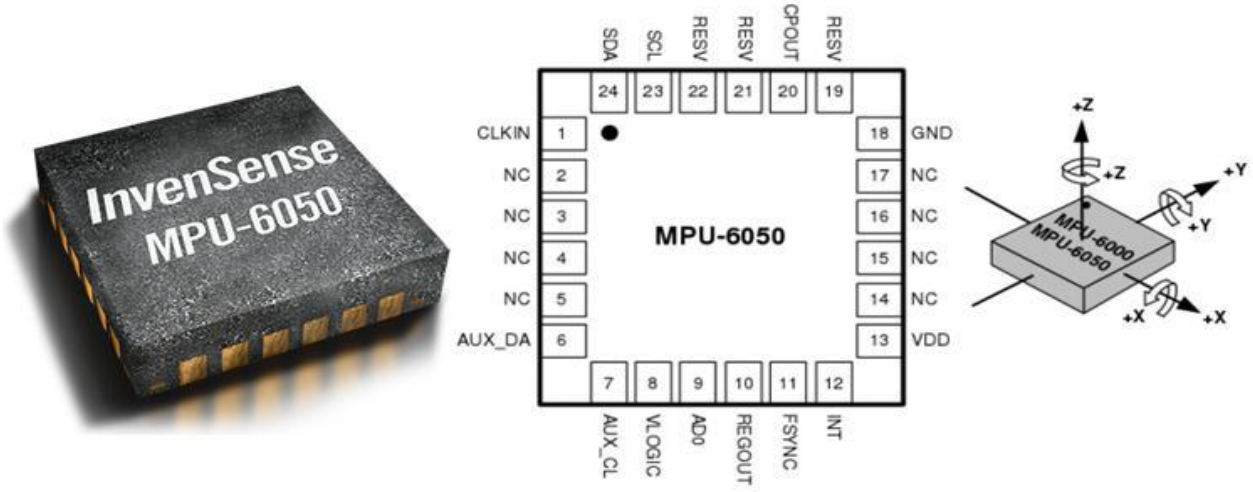


Fig.III.4 : La puce MPU6050 [31].

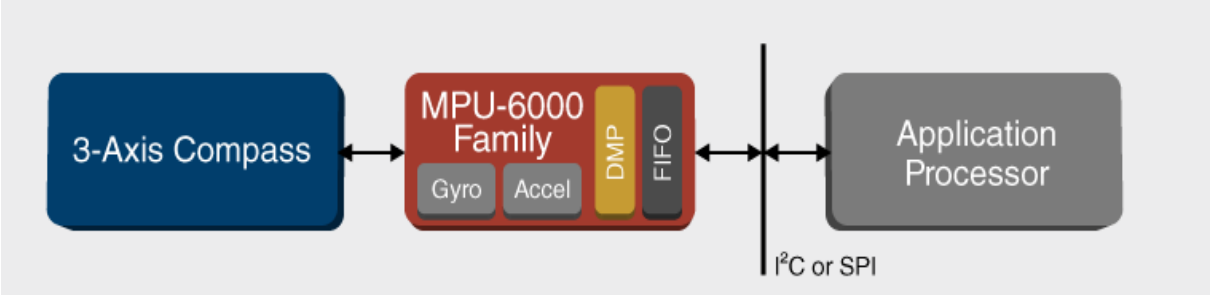


Fig.III.5 : le schéma de processus de MPU6050**.**

❖ **Les caractéristiques du GY521**

Le capteur InvenSense MPU-6050 contient un accéléromètre ADXL335 MEMS et un gyroscope L3GD20H MEMS dans une seule puce. Il est très précis, car il contient d'un convertisseur analogique-numérique 16 bits pour chaque canal et les caractéristiques suivantes :

- **Mode de communication :** protocole de communication IIC standard.
- **Convertisseur AD 16 bits intégré** : sortie de données 16 bits.
- **Gamme de gyroscopes :** +/- 250 500 1000 2000 degré / sec.
- **Gamme d'accélération :** +/- 2g, +/- 4g, +/- 8g, +/- 16g.

Les pins de la puce GY521 sont indiqués par le tableau III.1 suivant :

PINS	DESCRIPTION
VCC	Alimentation
GND	Ground
SCL	Horloge Série (Serial Clock)
SDA	Données série (Serial Data)
XCL	Horloge auxiliaire (Auxiliary Clock)
XDA	Données auxiliaires (Auxiliary Data)
INT	INTerrupt

Tableau III.1 : Description des Pin GY-521.

III.3.1 Le Gyroscopes L3GD20H

Le Gyroscopes L3GD20H est un capteur de taux angulaire triaxial à faible puissance, qui inclut un élément de détection et une interface CI (Common Interface) capable d'offrir le taux angulaire mesuré via une interface numérique (I^2C/SPI). L'élément de détection est fabriqué à l'aide d'un procédé de micro-usinage développé par ST (STMicroelectronics) pour produire des capteurs à inertie et des actionneurs sur des wafers en silicium. L'interface de CI est fabriquée à l'aide d'un procédé CMOS qui permet un haut niveau d'intégration pour concevoir un circuit dédié et ajusté correspondant au mieux aux caractéristiques de l'élément de détection. Le L3GD20H dispose d'une échelle pleine de ±245/±500/±2000 dps [31].

- **Caractéristiques**
  - ✓ Capteurs de vitesse angulaire à axe X, Y et Z à sortie numérique (gyroscopes) avec une échelle pleine grandeur programmable par l'utilisateur
  - ✓ tolérance de calibration : ±3%
  - ✓ plage de ± 250, ± 500, ± 1000 et ± 2000 ° / sec
  - ✓ Interface I2C
  - ✓ Les CAN 16 bits intégrés permettent l'échantillonnage simultané de gyroscopes.
  - ✓ Performances améliorées du bruit à basse fréquence.
  - ✓ Courant de fonctionnement du gyroscope : 3.6mA
  - ✓ courant de veille: 5µA
  - ✓ Facteur d'échelle de sensibilité étalonné en usine.
  - ✓ Auto-test utilisateur

III.3.2 ADXL335

Ce module accéléromètre 3 axes ADXL335 est de très petite taille et consomme très peu d'énergie, en faisant un composant idéal pour les applications embarquées. Il est capable de mesurer des accélérations sur une plage de ±3 g, ainsi que l'accélération statique (gravité) pour les applications de détection d'inclinaison. Il peut aussi mesurer l'accélération dynamique résultant du mouvement, des chocs ou bien des vibrations [31].

- **Caractéristiques**

  - ✓ Accéléromètre à trois axes à sortie numérique avec une plage d'échelle complète programmable de ± 2g, ± 4g, ± 8g et ± 16g

- ✓ Fonctionnement par alimentation unique : 1,8 V à 3,6 V
- ✓ Axes X, Y et Z intégré sur une puce unique
- ✓ Excellente stabilité thermique
- ✓ Résistance aux chocs jusqu'à 10 000g
- ✓ tolérance de calibration : ±3%
- ✓ interface de communication : I2C
- ✓ Détection d'orientation et signalisation
- ✓ Détection de tapotement
- ✓ Interruptions programmables par l'utilisateur
- ✓ Auto-test utilisateur

Dans la figure suivante présente le schéma électrique de la puce MPU6050.

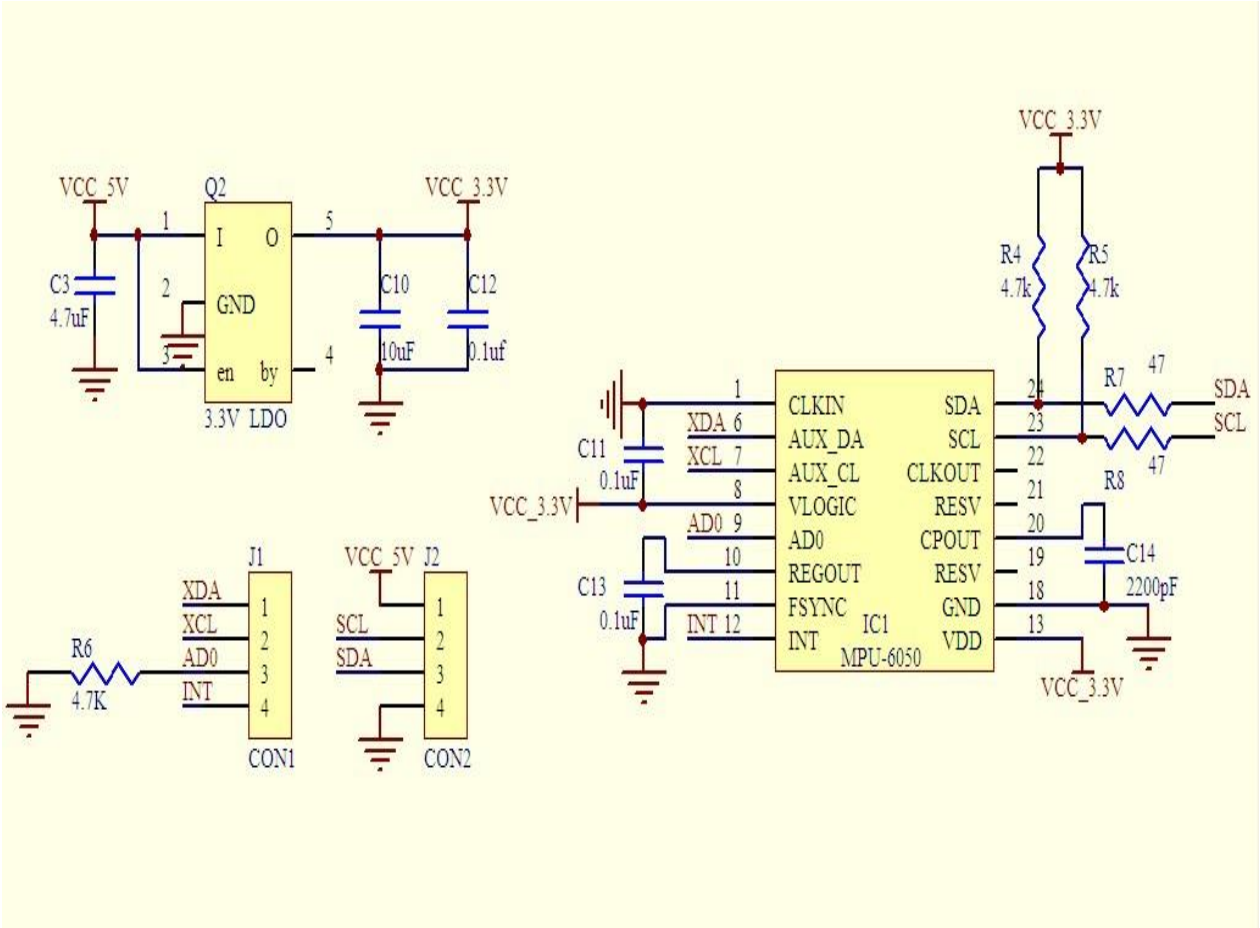


Fig.III.6 : Le schéma de la carte de dérivation GY-521 pour la puce MPU6050.

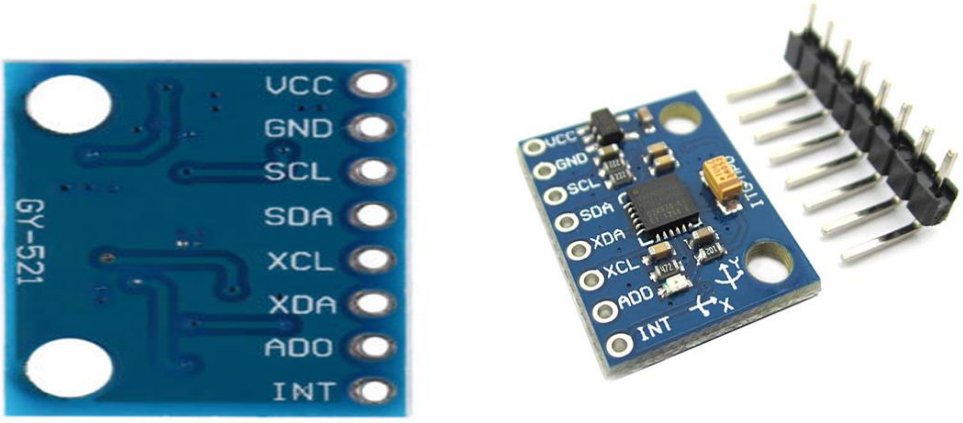


Fig.III.7 : GY-521.

III.4. Protocol de communication

Le bus de communication I2C est très populaire et largement utilisé par de nombreux appareils électroniques car il peut être facilement implémenté dans de nombreuses conceptions électroniques qui nécessitent une communication entre un maître et plusieurs périphériques esclaves ou même plusieurs périphériques maîtres. Il s'agit de l'adresse de l'appareil afin que le maître puisse choisir avec quels appareils communiquent (figure III.8) [27].

Selon la figure III.8, les deux lignes, sont appelés horloge série (SCL) et données série (SDA). La ligne SCL porte le signal d'horloge qui synchronise le transfert de données entre les périphériques sur le bus I2C et celui généré par l'appareil maître. L'autre ligne SDA transporte les données vers les périphériques.

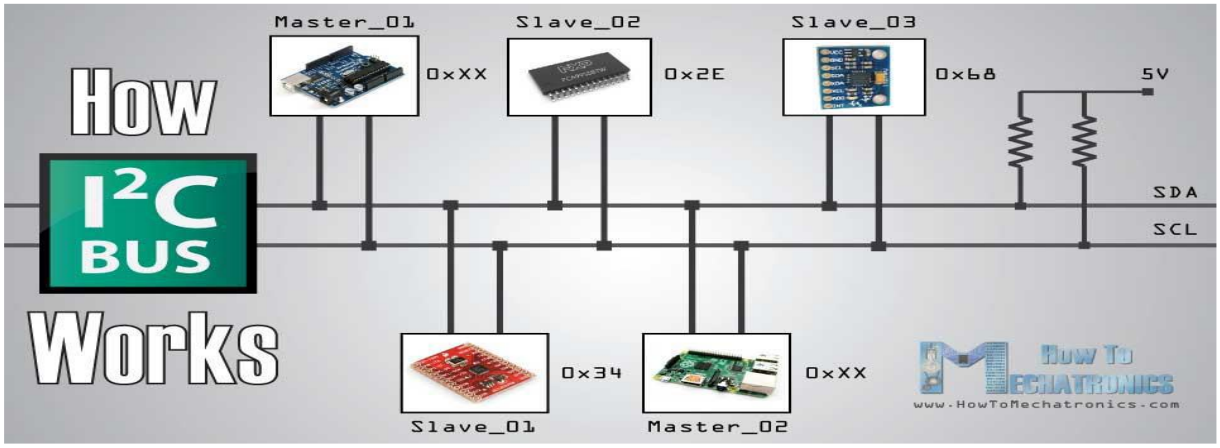


Fig.III.8 : Le principe de la communication I2C.

Les deux lignes sont "open-drain", ce qui signifie que des résistances de rappel doivent être attachées afin que les lignes soient hautes car les périphériques du bus I2C sont actifs bas.

Les valeurs couramment utilisées pour les résistances vont de 2K pour des vitesses supérieures à environ 400 kbps, à 10K pour des vitesses inférieures à environ 100 kbps.

Enfin, le MPU-6050 dispose d'un tampon FIFO, associé à un signal d'interruption intégré. Il peut être demandé de placer les données du capteur dans le tampon et la broche d'interruption indiquera au microcontrôleur, lorsque les données sont prêtes à être lues.

III.5. Présentation de microcontrôleur utilisée

Les microcontrôleurs sont devenus assez simples à mettre en œuvre, grâce en particulier à des environnements de développement comme l'Arduino et MikroC. Mais seule une compréhension en profondeur permet de tirer parti au maximum de leur potentiel dans de nombreuses applications.

Toutes les solutions à base de composants programmables ont pour but de réduire le nombre de composants sur le circuit électronique et donc fiabiliser le circuit.

Le Microcontrôleur est en concurrence avec d'autres technologies suivant les applications, le modèle UNO de la société ARDUINO est une carte électronique dont le cœur est un microcontrôleur ATMEL de référence ATMega328 comme le montre la figure III.9. Ce dernier est un microcontrôleur 8bits de la famille AVR dont la programmation peut être réalisée en langage C [17].

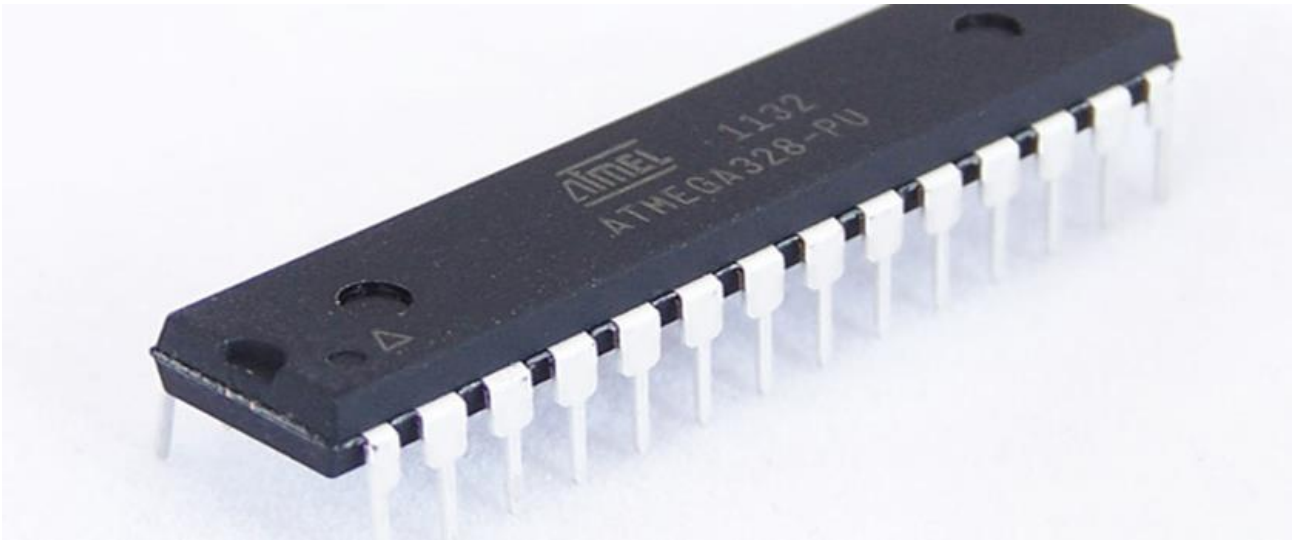


Fig.III.9 : Microcontrôleur ATMega328.

III.5.1 La carte Arduino

Les cartes Arduino font partie de la famille des microcontrôleurs, Arduino a été inventé 2005 par l'enseignent Massimo Banzi dans une école de dessin à Ivea en Italie avec l'aide de David Cuartielle ingénieur en microcontrôleurs et aussi leur étudiant Mellis qui est spécialiste dans les langages de programmation.

Une carte Arduino comme le montre la figure III.10, est une interface programmable capable de piloter des capteurs et des actionneurs afin de simuler ou créer des systèmes automatisés. Elle peut stocker un programme et le faire fonctionner. La carte reçoit des informations analogiques ou numériques sur ces entrées. Le microcontrôleur traitera ces informations et les transmettra vers les sorties numériques [16].

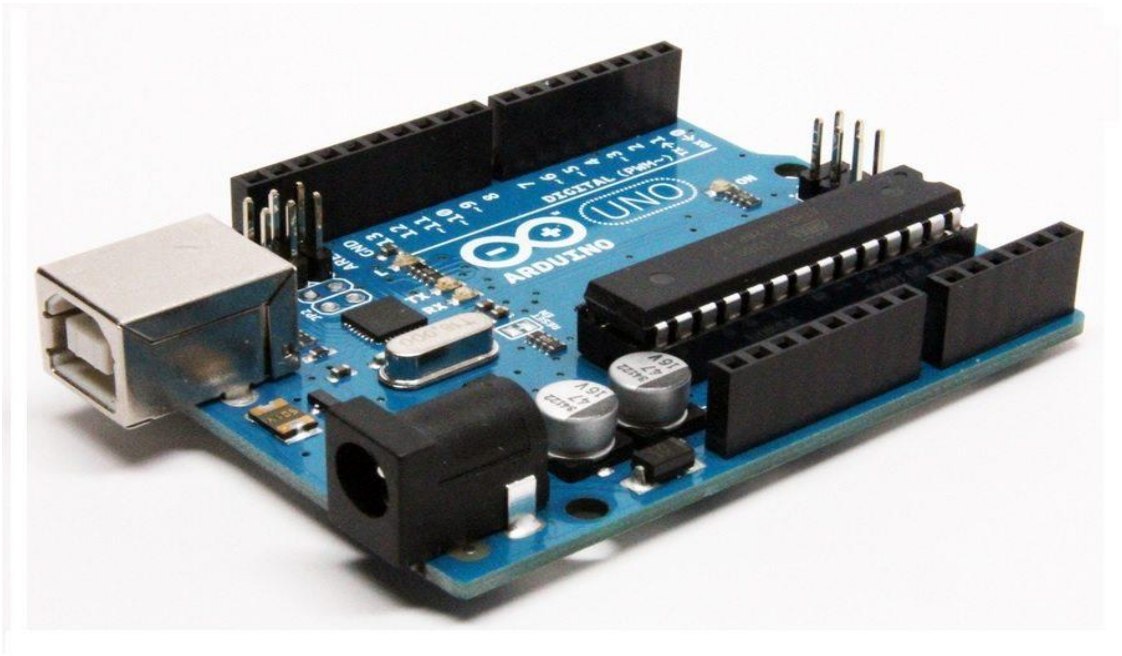


Fig.III.10 : La carte Arduino UNO.

L'intérêt principal des cartes ARDUINO est leur facilité de mise en œuvre. Arduino fournit un environnement de développement s'appuyant sur des outils open-source.

Des exemples des Shields (cartes d'extension) avec des fonctions diverses qui s'enfichent sur la carte Arduino sont classer ci-dessous :

- Relais, commande de moteurs, lecteur carte SD…
- Ethernet, WIFI, GSM, GPS…
- Afficheurs LCD, Écran TFT…

III.5.2. Choix de l'Arduino UNO

Parmi plusieurs types, nous avons choisi une carte Arduino UNO. L'intérêt principal de cette carte est de faciliter la mise en œuvre d'une telle réalisation qui sera détaillée par la suite. L'injection du programme déjà converti par l'environnement sous forme d'un code « HEX » dans la mémoire du microcontrôleur se fait d'une façon très simple par la liaison USB. La carte Arduino contient une mémoire morte de 1 kilo. Elle est dotée de 14 entrées/sorties un cristal à 16 MHz, une connexion USB et possède d'un bouton de remise à zéro et une prise jack d'alimentation. La description de carte est illustrée dans la figure ci-dessous [18].

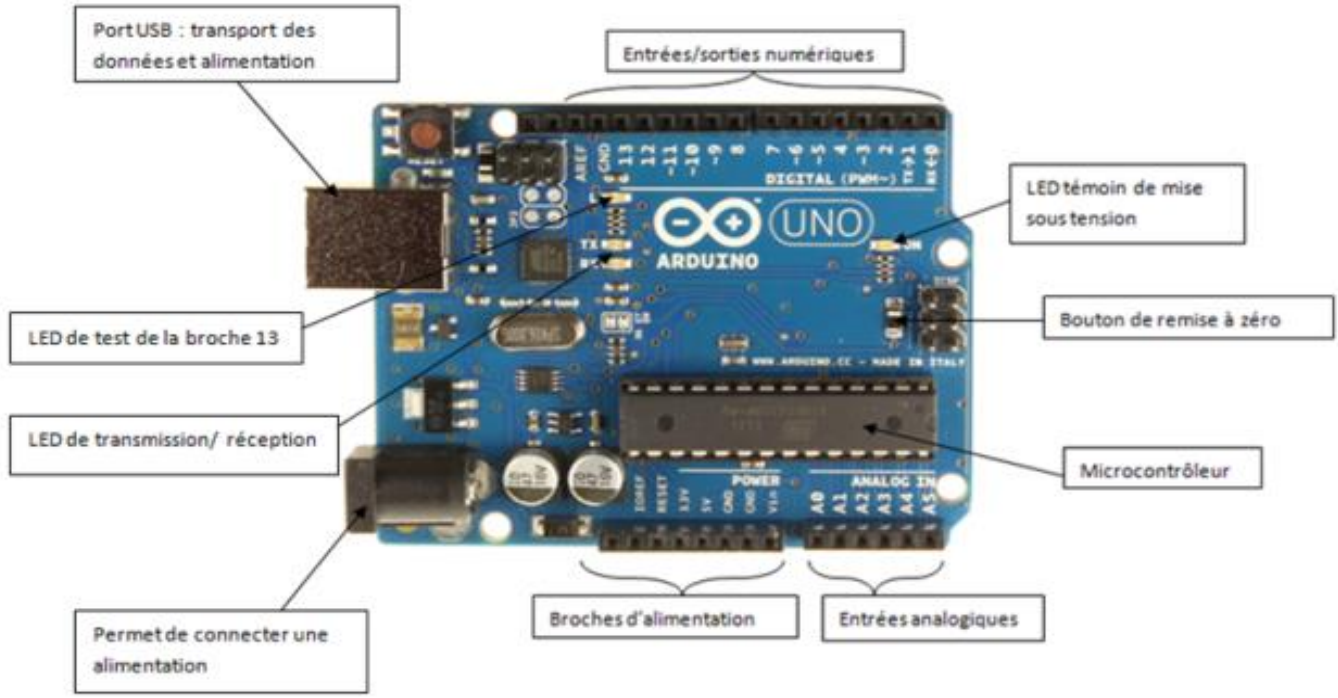


Fig.III.11 : Description de la carte Arduino UNO.

III.5.3. Constitution de la carte Arduino UNO

Un microcontrôleur est constitué par un ensemble d'éléments que chacun a une fonction bien déterminée [19]. Il est en fait constitué des mêmes éléments que sur la carte mère d'un ordinateur :

1. La mémoire

Il en possède 5 types :

•**La mémoire Flash :** C'est celle qui contiendra le programme à exécuter. Cette mémoire est effaçable et réinscriptible.

•**RAM :** c'est la mémoire dite "vive", elle va contenir les variables de notre programme. Elle est dite "volatile" car elle s'efface si on coupe l'alimentation du microcontrôleur.

•**EEPROM** : C'est le disque dur du microcontrôleur. Vous pourrez y enregistrer des informations qui ont besoin de survivre dans le temps, même si la carte doit être arrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme.

•**Les registres** : c'est un type de mémoire utilisé par le processeur.

•**La mémoire cache** : c'est une mémoire qui fait la liaison entre les registres et la RAM.

2. Le processeur

C'est le composant principal du microcontrôleur. C'est lui qui va exécuter le programme qu'on lui donnera à traiter. On le nomme souvent le CPU.

Pour que le microcontrôleur fonctionne, il lui faut une alimentation, Cette alimentation se fait en générale par du +5V. D'autres ont besoin d'une tension plus faible, du +3,3V.

En plus d'une alimentation, il a besoin d'un signal d'horloge. C’est en fait une succession de 0 et de 1 ou plutôt une succession de tension 0V et 5V. Elle permet en outre de cadencer le fonctionnement du microcontrôleur à un rythme régulier. Grâce à elle, il peut introduire la notion de temps en programmation.

L’image suivante montre d’une façon détaillée L’architecture interne d’un microcontrôleur Atmega328 AVR [19].

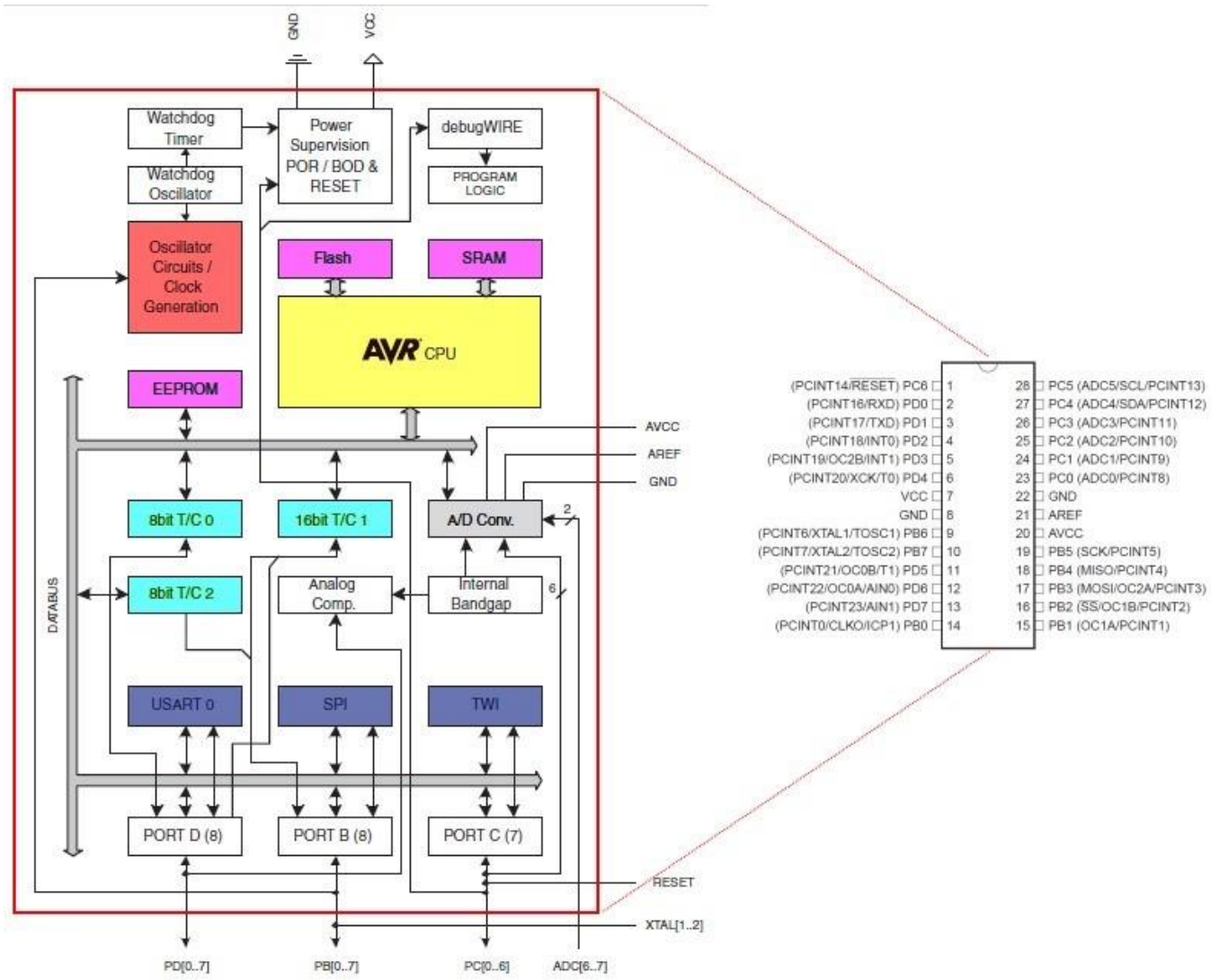


Fig.III.12 : L'architecture interne d'un microcontrôleur Atmega328 AVR.

III.5.4. Entrées & sorties de la carte

Arduino Cette carte Possède14 broches numériques (numérotée de 0 à 13) digitales (dont 6 peuvent être utilisées en tant que sortie PWM), 6 entrées analogiques. Elle peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique, en utilisant les instructions pinMode (), digitalWrite() et digitalRead() du langage Arduino.

Ces broches fonctionnent en 5V. Chaque broche peut fournir ou recevoir un maximum de 40mA d'intensité et dispose d'une résistance interne. Cette dernière s'active sur une broche en entrée à l'aide de l'instruction DigitalWrite [19].

En plus, certaines broches ont des fonctions spécialisées :

Interruptions Externes : Broches 2 et 3.Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur. Impulsion PWM (largeur d'impulsion modulée). Broches 3, 5, 6, 9, 10, et 11, Fournissent une impulsion PWM 8-bits à l'aide de l'instruction analogWrite().

SPI (Interface Série Périphérique) : le SPI est également un bus de communication destiné à dialoguer avec les circuits périphériques. À la différence de l'I2C, il est bidirectionnel. Il permet aussi un débit de communication plus important. 4 broches sont employées pour le SPI, l'horloge, les données en sortie, les données en entrée et la sélection du circuit avec lequel l'Arduino veut dialoguer.

LED : Broche 13. Il y a une LED incluse dans la carte connectée à la broche13. Lorsque la broche est au niveau HAUT, la LED est allumée, lorsque la broche est au niveau BAS, la LED est éteinte.

III.5.5. Architecture interne de Arduino UNO

La carte Arduino n'est rien d'autre qu'un microcontrôleur de la famille AVR. Associé à une interface de programmation qui permet de communiquer simplement avec l'interface de développement.

L'Architecteur interne est très propre, contient 32 registre généraux et un jeu d'instruction RISC permettent une utilisation efficace pour un compilateur C.

Block Diagram of the AVR Architecture (extraite de la datasheet de l'ATtiny45)

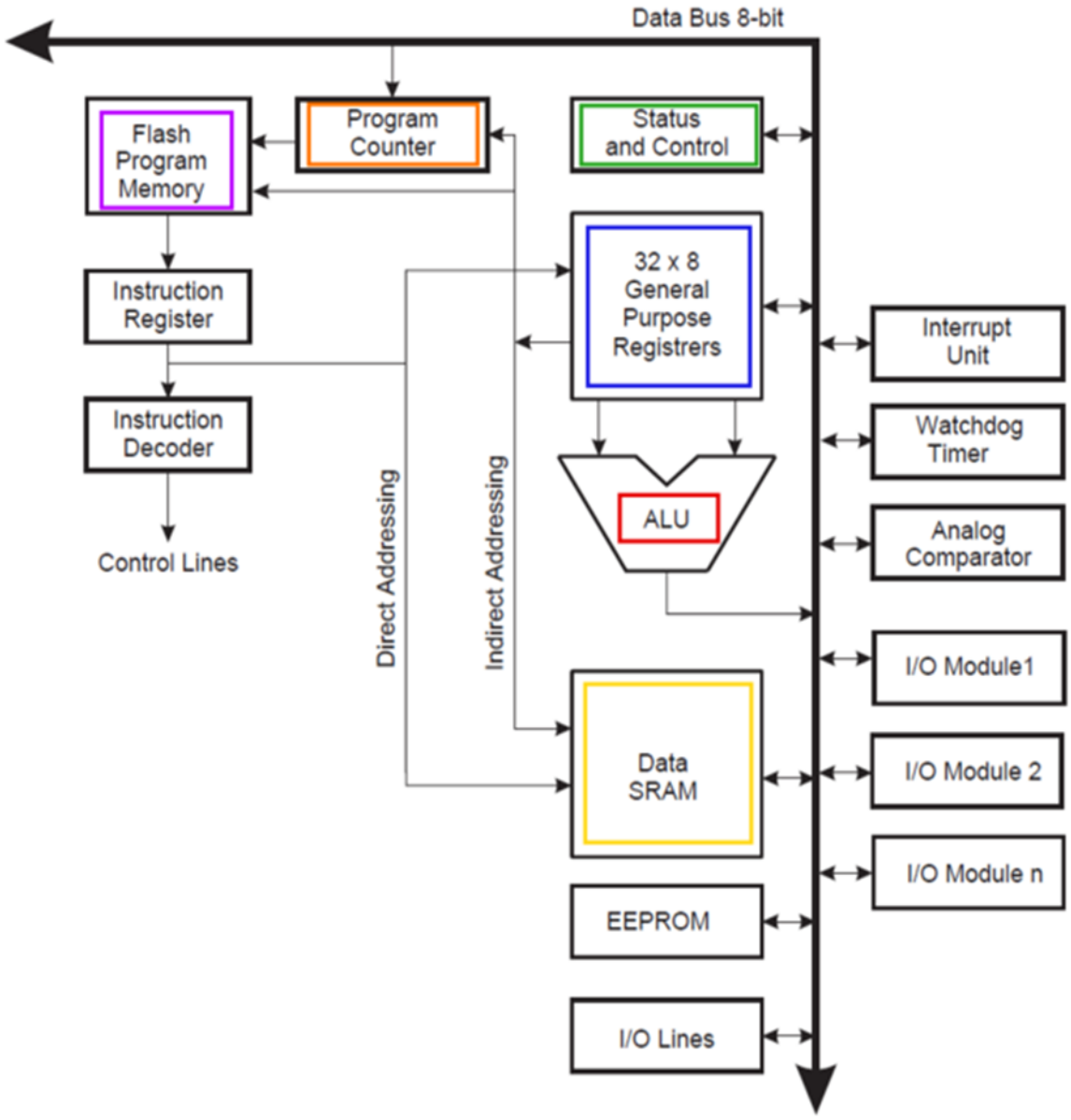


Fig.III.13 : Architecture interne de la carte Arduino UNO.

III.6. Étude de la partie logicielle

Une telle carte d'acquisition qui se base sur sa construction sur un microcontrôleur doit être dotée d'une interface de programmation comme est le cas de notre carte. L'environnement de programmation open-source peut être téléchargé gratuitement (pour Mac OS X, Windows, et Linux).

III.6.1. L'environnement de la programmation

Le logiciel de programmation de la carte Arduino sert d'éditeur de code (langage proche du C). Une fois, le programme tapé ou modifié au clavier, il sera transféré et mémorisé dans la carte à travers la liaison USB. Le câble USB alimente à la fois en énergie la carte et transporte aussi l'information, ce programme appelé IDE Arduino [19].

III.6.2. Structure générale du programme (IDE Arduino)

Comme n'importe quel langage de programmation, une interface souple et simple est exécutable sur n'importe quel système d'exploitation Arduino basé sur la programmation en C.

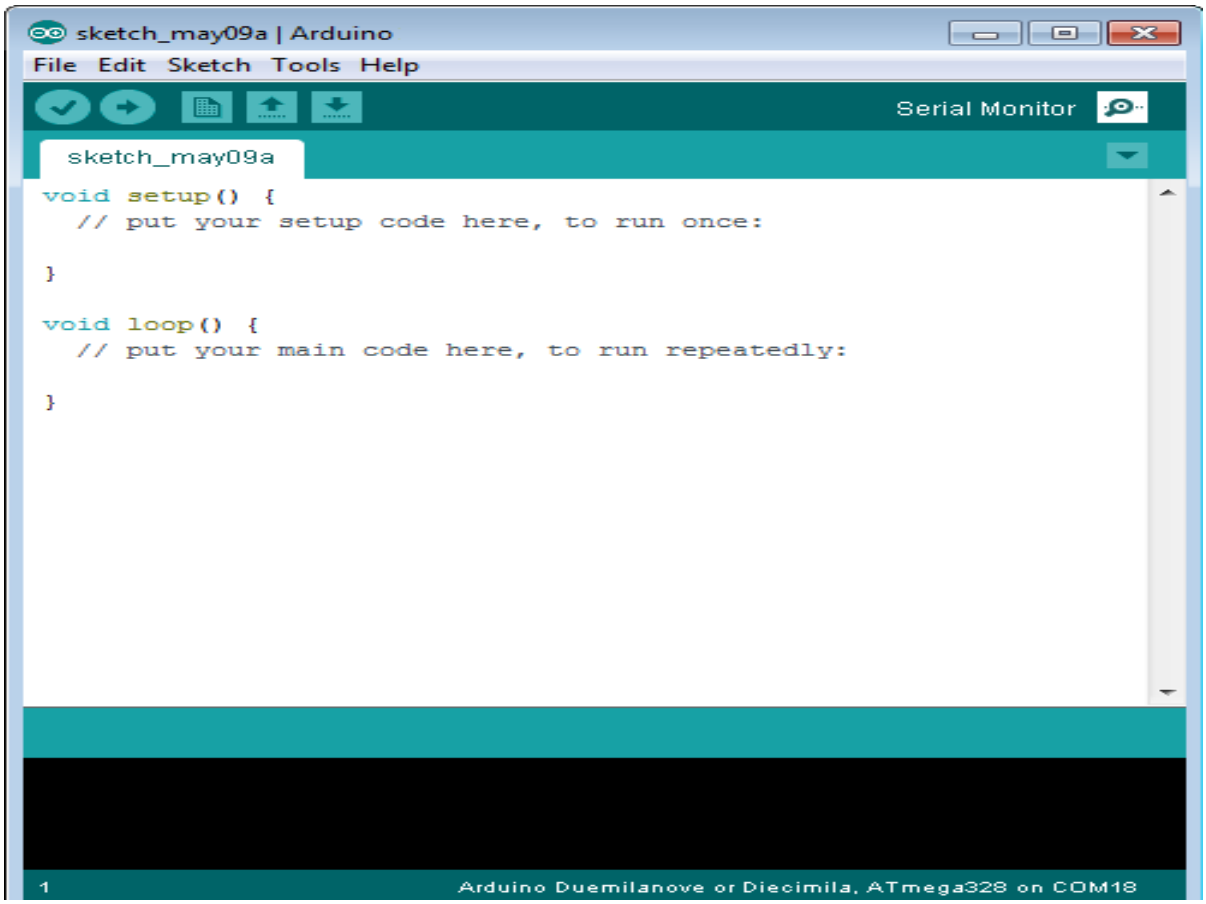


Fig.III.14 : L'IDE Arduino.

III.6. 3. Le software utilisée "Processing"

L'IDE de Processing est une excellente source de création des graphiques, est un langage de programmation et une interface de développement intégré (IDE) écrite en Java.

Le Processing relie les concepts de logiciel aux principes de la forme visuelle, du mouvement et de l'interaction. Il intègre un langage de programmation, un environnement de Développement et une méthodologie d'enseignement dans un système unique. Le Processing a été créé pour enseigner les fondamentaux de la programmation informatique dans un contexte visuel, être utilisé comme outil de production pour des contextes spécifiques. Les étudiants, les artistes, les professionnels du design et les chercheurs l'utilisent pour l'apprentissage, le prototypage et la production [20].

Fig.III.15 : Logo du Processing.

Les programmes réalisés avec Processing peuvent donc fonctionner sur toute machine possédant un dispositif virtuelle Java ainsi que dans les navigateurs Web équipé du plugin Java.

Les bibliothèques de Processing étendent facilement la capacité d'envoyer / recevoir des données dans divers formats et importer et exporter des formats de fichiers 2D et 3D [20].

III.6. 3.1 Les bases de l'interface de Processing

L'interface

L'interface d'utilisation de Processing est composée de deux fenêtres distinctes : La fenêtre principale dans laquelle vous allez créer votre projet et la fenêtre de visualisation dans laquelle vos créations (dessins, animations, vidéos) apparaissent.

On trouve plus précisément les éléments suivants dans l'interface [21] :

1. Barre d'actions.

2. Barre d'onglets.

3. Zone d'édition (pour y saisir votre programme).

4. Console, qui comprend un onglet les messages affichés par programme un onglet pour les erreurs. Cette console indique aussi si des mises à jour (« updates») sont disponibles pour les librairies et les modes.

5. Fenêtre de visualisation (espace de dessin).

6. Liste déroulante pour les modes.

7. Bouton pour activer le mode debug (pas à pas).

La figure III.16 suivante indique bien l'emplacement de ces éléments.

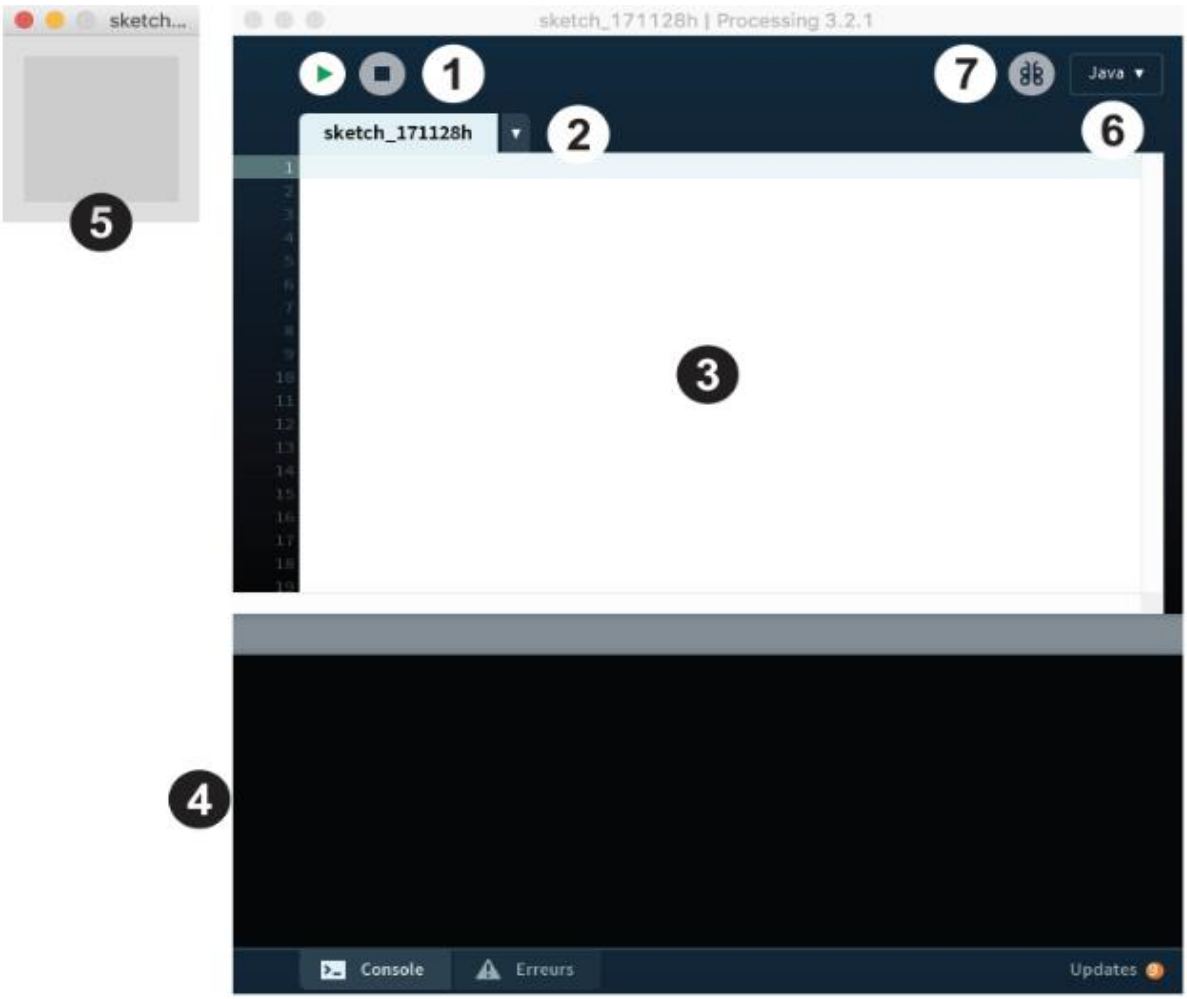


Fig III.16 : l'interface de Processing.

Barre d'actions

 Bouton "Run" : exécute votre sketch (votre programme).

 Bouton "Stop" : arrête l'exécution de votre sketch.

Processing permet de travailler dans plusieurs modes, un mode permettant de programmer dans un environnement spécifique à chaque plateforme visée (ex : application pour tablette Android, ...). Ces modes peuvent être gérés depuis une interface spécifique le « Contribution Manager ».

Vous pouvez changer ce mode à tout moment depuis l'interface, en ayant au préalable sauvegardé votre sketch.

Le dossier de travail

C'est le dossier dans lequel seront enregistrés les sketchs et les librairies (des modules externes proposant des fonctionnalités supplémentaires). Par défaut ce dossier se nomme Processing et se trouve dans Documents (sous Mac) ou Mes Documents (sous Windows). Sous GNU/Linux, il est dans votre dossier personnel sous le nom de sketchbook. Pour modifier ce dossier, allez dans le menu Processing > Préférences. Dans la boîte de dialogue qui apparaît, il suffit de cliquer sur Naviguer pour choisir le dossier qui convient [21].

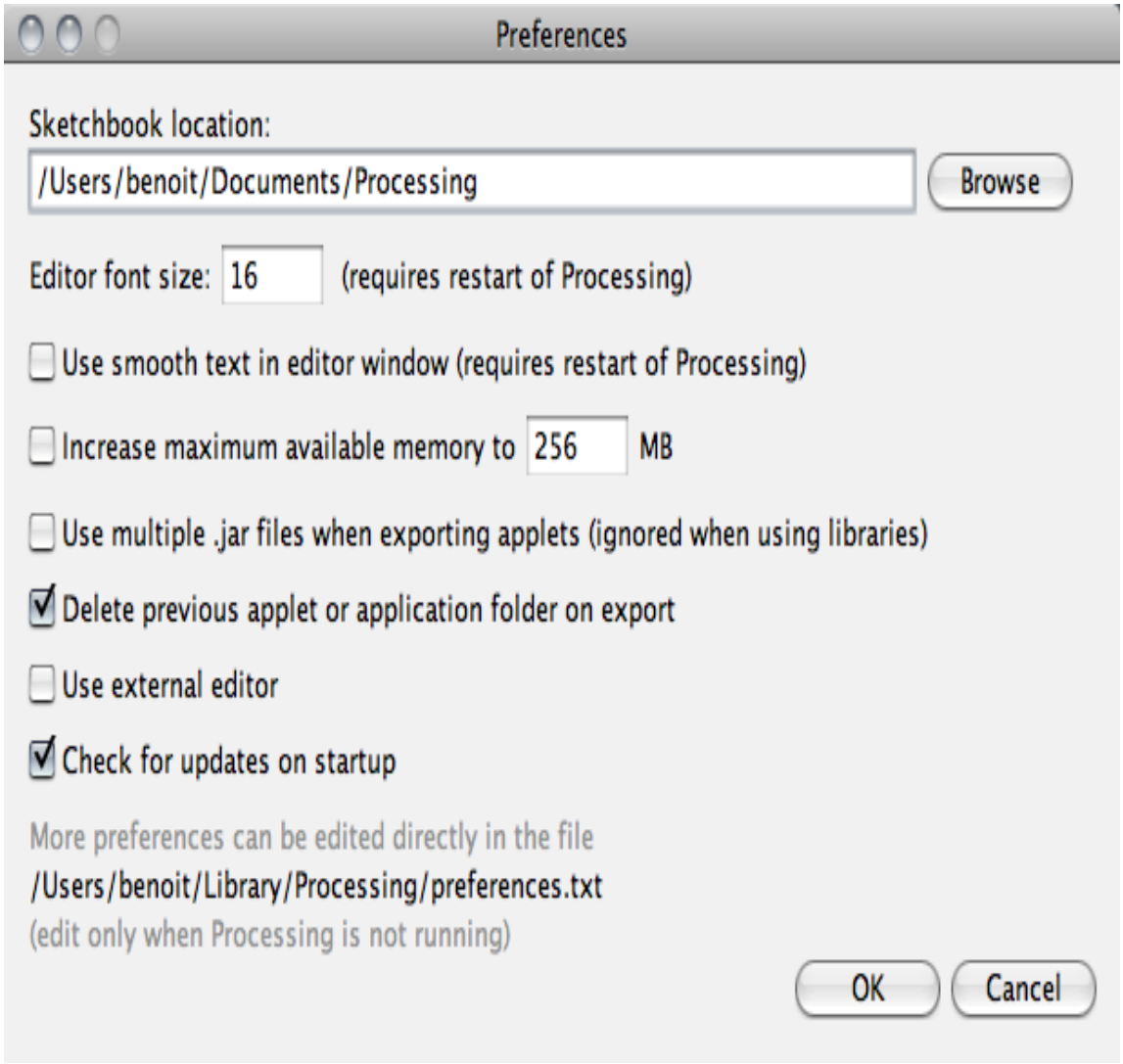


Fig III.17 : La fenêtre « Préférence » de Processing.

III.7. Communication entre la carte Arduino & logiciel Processing

L'IDE de Processing est similaire à Arduino en termes de structure. Il a des fonctions de configuration et des fonctions de dessin comme un Arduino IDE a une fonction de configuration et de boucle. L'IDE de traitement peut communiquer avec l'IDE Arduino via une communication en série. De cette façon, nous pouvons envoyer des données de l'Arduino à l'IDE de Processing et aussi de l'IDE de Processing à l'Arduino. Le déroulement de la communication entre le logiciel Arduino IDE et le Processing, se fait à travers une liaison série half duplex, les données sont téléversés vers la carte Arduino où se passe le traitement ensuite les résultats de traitement seront transférées vers le logiciel Processing sous la même liaison, voir figure III.18.

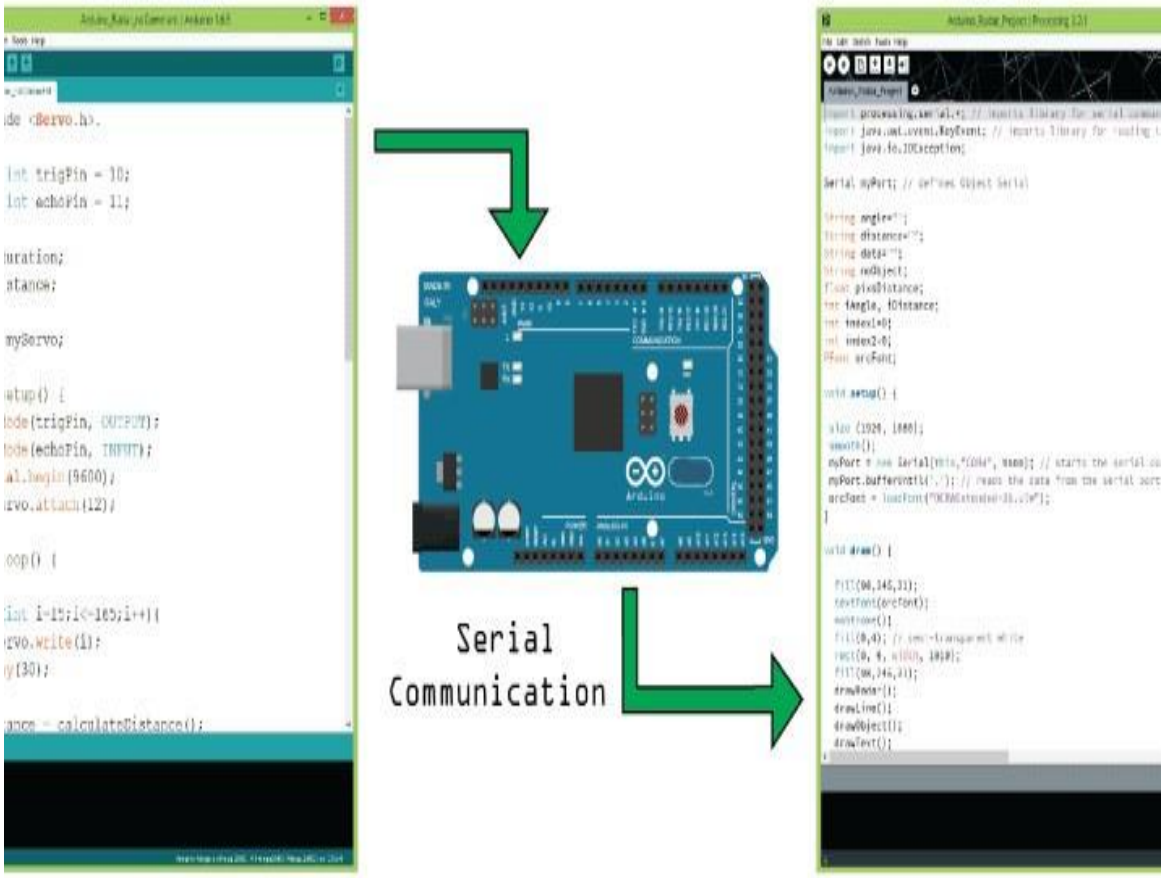


Fig.III.18 : Communication entre Arduino & Processing.

III.8. Réalisation du prototype

Après avoir donné une description théorique sur le matériel utilisé, le module Arduino et son environnement de développement, le capteur IMU choisit et le software Processing adopter. On va procéder maintenant à L'application expérimentale, pour cette raison, plusieurs étapes ont été nécessaires afin de réaliser une telle combinaison.

III.8.1. Principe du prototype

A partir de notre central inertiel on va lire en temps réel les donnes de l'accélération et la vitesse angulaire ensuite les traiter sous la norme Tangage Roulis Lacet afin de déduire la position. Ensuite on visualise toutes les données sous forme d'animation par une interface graphique d'un simulateur du vol personnalisé.

Le principe du logiciel Processing ici consiste à lire en temps réel toutes les données transmises sur la liaison série pour les présenter sous forme d'animation (deux programmes à exécuter) qui va nous permettre de visualiser ces valeurs sous forme d'un Horizon artificiel et un compas comme dans un avion réel, mais également sous la norme Tangage Roulis Lacet.

III.8.2. Déroulement du projet

Notre projet de réalisation est basé sur deux parties essentielles :

La première, est la conception Hardware. Et la deuxième est la simulation des donnes via l'interface graphique sous forme d'instruments du vol.

Pour ce faire, on doit passer par :

- La configuration du module MPU6050 avec notre microntrolleur Arduino.
- Visualisation des résultats retournés sur l'écran d'ordinateur à l'aide de logiciel Processing.

III.8.3. Schéma synoptique général

Le schéma synoptique général du projet est indiqué par la figure III.19

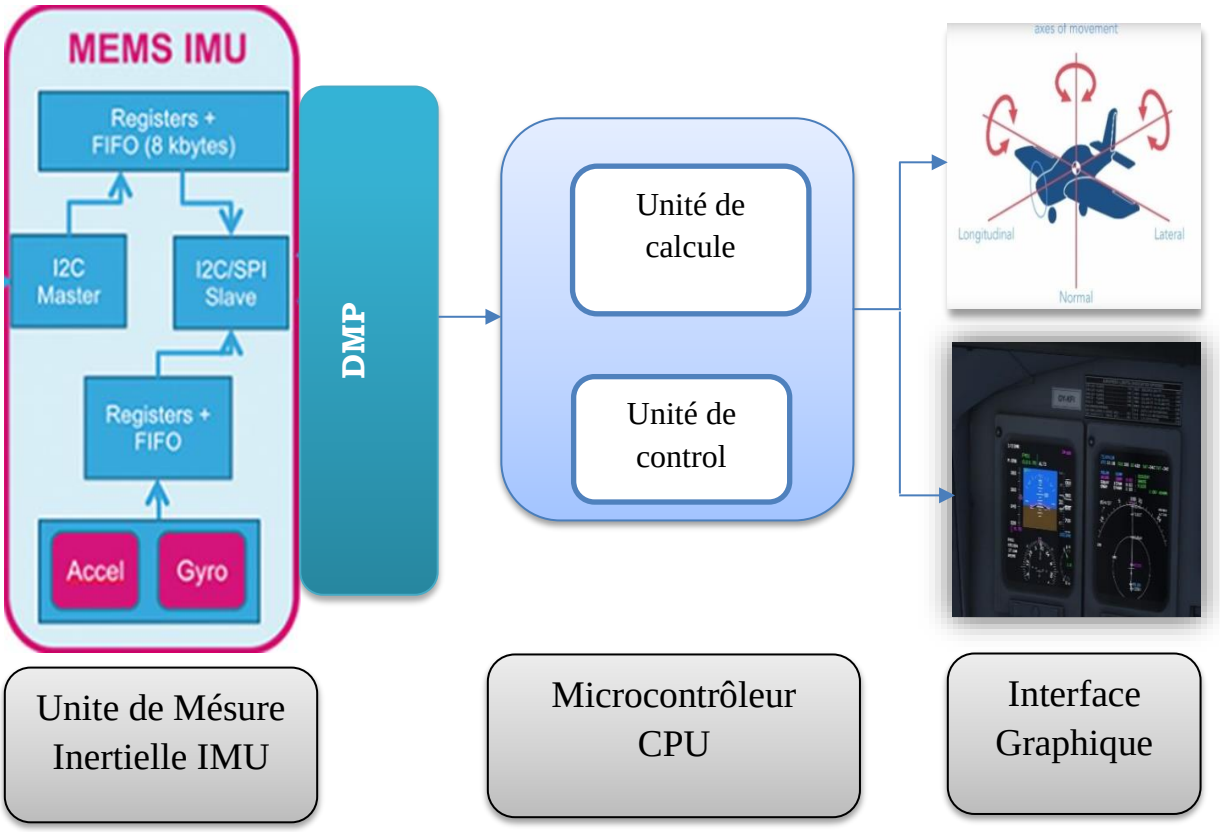
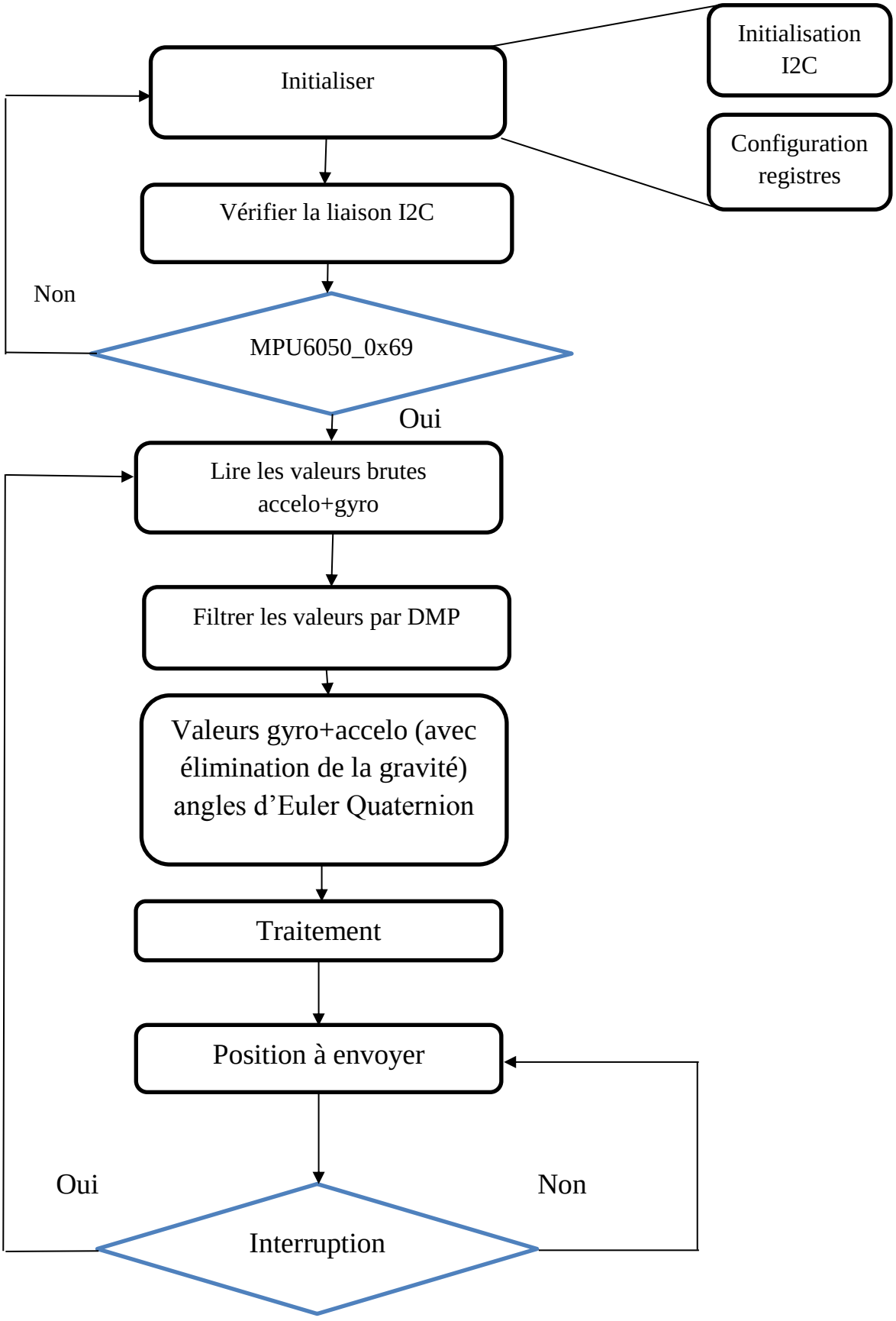


Fig.III.19 : Schéma synoptique général.

L'organigramme suivant présente les étapes logicielles suivies afin d'exploiter le composant (IMU), en commençant par initialiser le bus I2C et configurer les registres du module MPU6050, après on s'assure que la liaison est établie en lisant la valeur de registre MPU6050. En effet ce registre contient par défaut la valeur 0x69 d'après le Datasheet [31] du module MPU6050, ensuite le processeur DMP traite les valeurs brutes de l'accéléromètre et le gyroscope pour en sortir les angles d'Euler, le quaternion et les valeurs de l'accéléromètre et le gyroscope en éliminant la gravité.

Algorithme principale de la détection de position



III.8.5. Etapes de Réalisation de projet

- **Composants utilisés**

Pour notre réalisation, on a assemblé les différents composants suivants :

Une Carte Arduino UNO

- ✓ Une mini maquette de prototypage (BreadBord)
- ✓ une carte GY-521 _MPU6050
- ✓ Certains câbles de démarrage Male-Femal
- ✓ Fer à souder + soudure
- ✓ Une petite maquette d'avion

Alimentation du circuit

L'ensemble des dispositifs Arduino, le capteur MPU6050, exigent une alimentation stabilisée de (+5V), pour notre travail de réalisation, nous avons alimenté notre montage à travers le port USB de l'ordinateur.

III.8.5.1. Test du fonctionnement de la carte Arduino

Initialement, on teste le fonctionnement de la carte Arduino, en connectant cette dernière avec le port USB de PC. Si la LED power s'allume la carte est bonne. Ensuite, on clique sur le bouton Reset de la carte, pour supprimer tout ancien programme et de la réinitialiser.

III.8.5.2. Le fonctionnent du Module MPU6050

Le gyroscope - accéléromètre MPU-6050 comporte 6 axes. Sa puce MEMS est très précise avec une conversion analogique-digitale sur 16 bits simultanée sur chaque canal, et une interface I2C (400 kHz). Le capteur contient un registre FIFO de 1024 octets que le microcontrôleur Arduino peut lire, étant prévenu par un signal d'interruption. Le module fonctionne en esclave sur le bus I2C vis à vis de l'Arduino (pins SDA, SLC).

III.8.5.3. Câblage

Pour le schéma de câblage et le profil de connexion, on connecte les broches des composants suivant le tableau III.2.

Les différentes connexions et numéro des pins de circuit sont présentés par le tableau suivant :

Pin GY-521	Pin Arduino UNO
VCC	+5V
SCL	A5
SDA	A4
D2	INT
GND	GND

Tableau III.2. : Broche et connexion du notre circuit électronique.

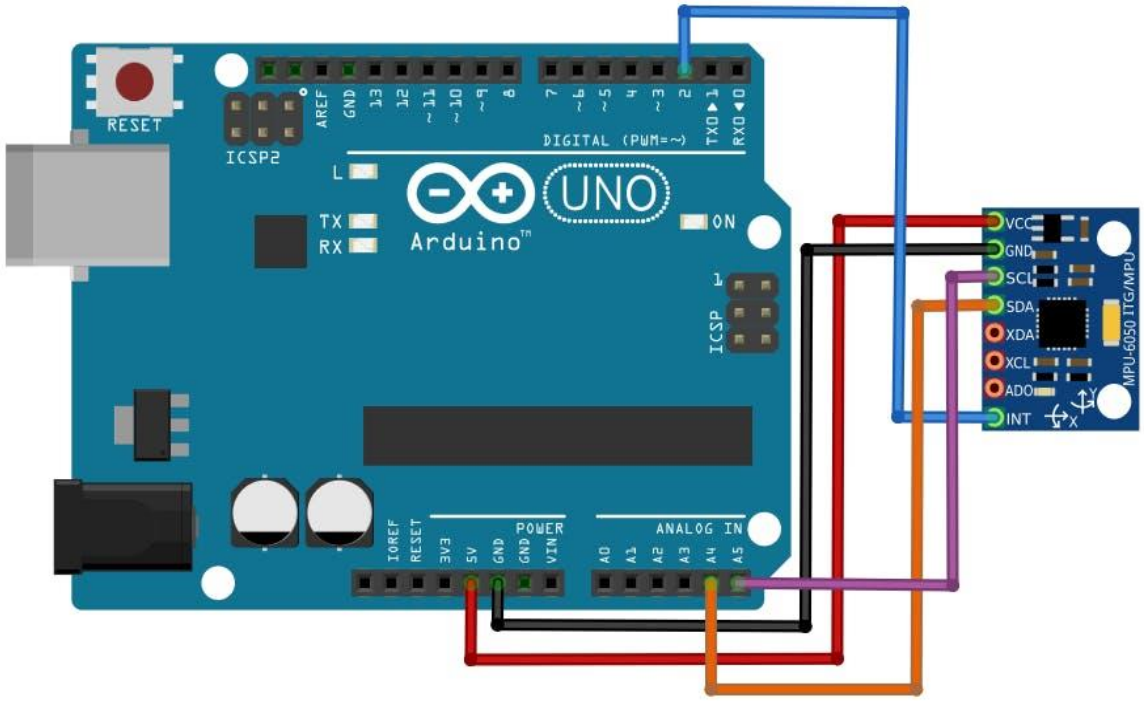


Fig.III.20 : Schéma de Circuit des connexions de la réalisation.

III.8.6. Démarche de programmation

Tout d'abord, afin de tester le MPU 6050, on doit installer deux bibliothèques indispensables pour le fonctionnement de notre projet.

- **Bibliothèque MPU 6050 pour Arduino**

Pour interagir le module MPU6050 avec Arduino, tout d'abord, il faut télécharger la librairie mpu6050 disponible sur internet, décompresser et copier les deux fichiers nommés «MPU6050. LIB et MPU6050.IDX » et les placer dans le dossier des bibliothèques de notre logiciel Arduino [29].

- **Bibliothèque de I2C pour Arduino**

Pour assurer la communication entre l'Arduino et le module IMU mpu6050 on doit aussi télécharger la bibliothèque I2C nommé "I2Cdev.zip" disponible sur internet, qui présente trois fichiers nommés « I2C.IDX, I2CTEP.LIB, I2CTEP.HEX » Maintenant, on les place dans la bibliothèque de logiciel Arduino [29].

Alors maintenant, dans le dossier "librairies" d'Arduino, nous avons deux nouvelles entités Fig.III.21.

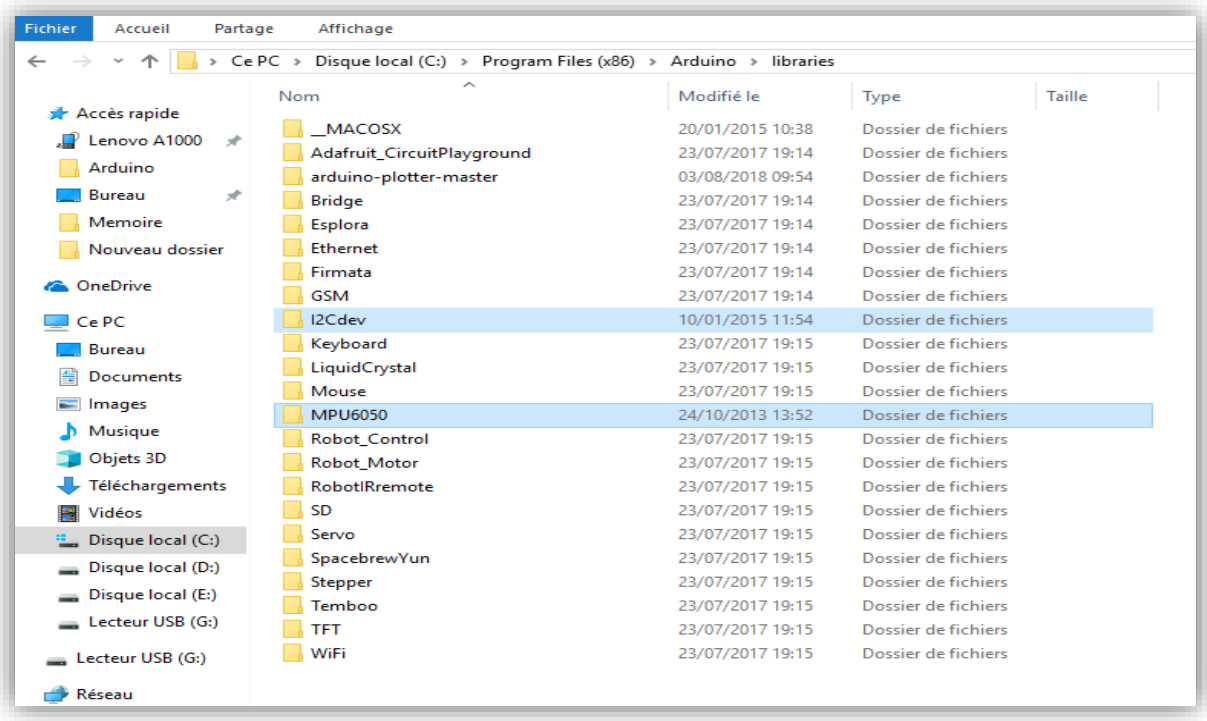


Fig.III.21 : Dossier " librairies " de Arduino.

Ensuite on clique sur l'IDE Arduino pour voir si ces nouvelles bibliothèques sont visibles (Fig.III.22).

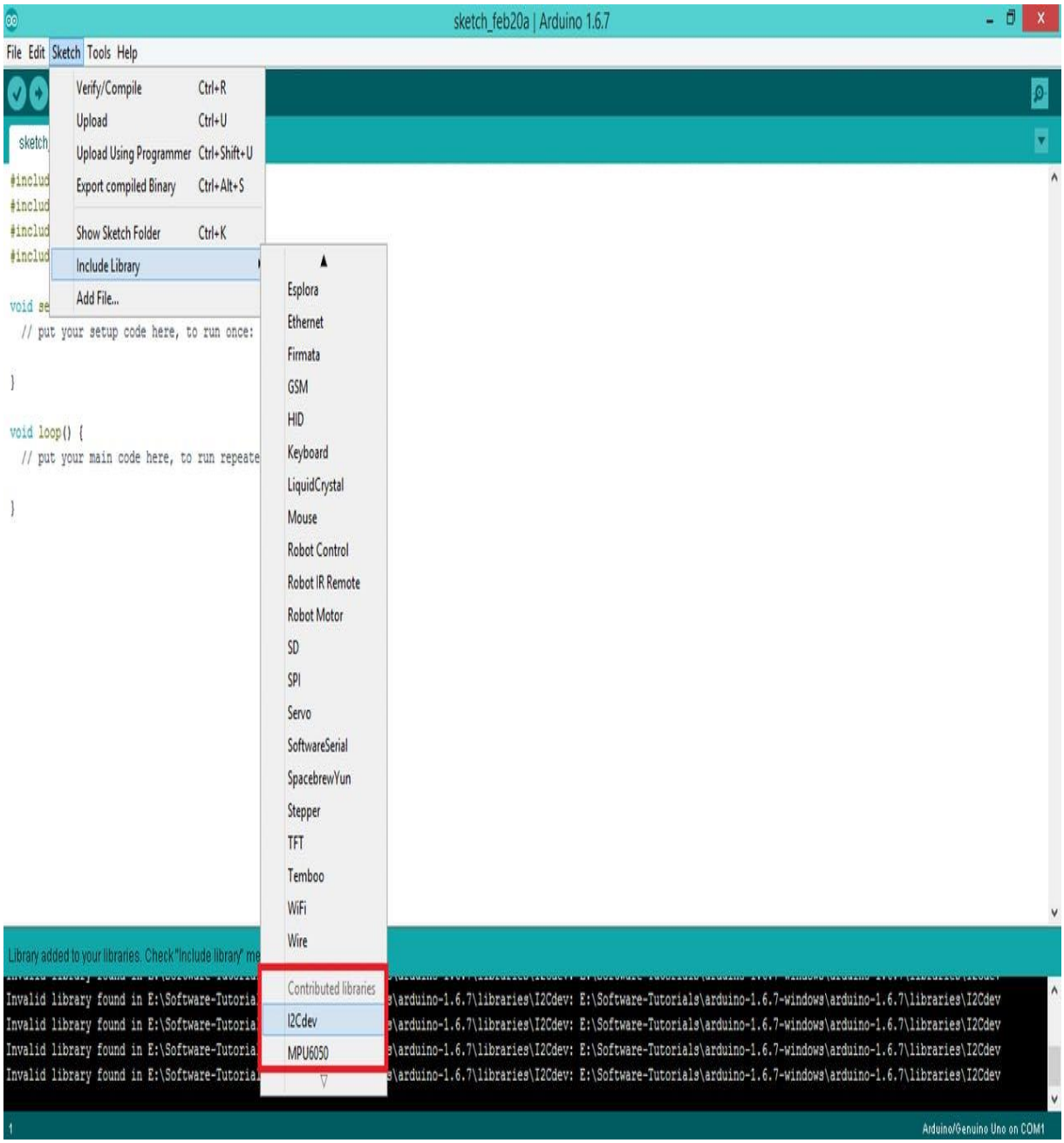


Fig.III.22 : les bibliothèques I2C et MPU6050 sous l'IDE Arduino.

Avant d'inclure ces bibliothèques dans notre sketch, on doit rétablir le code de configuration de MPU6050.

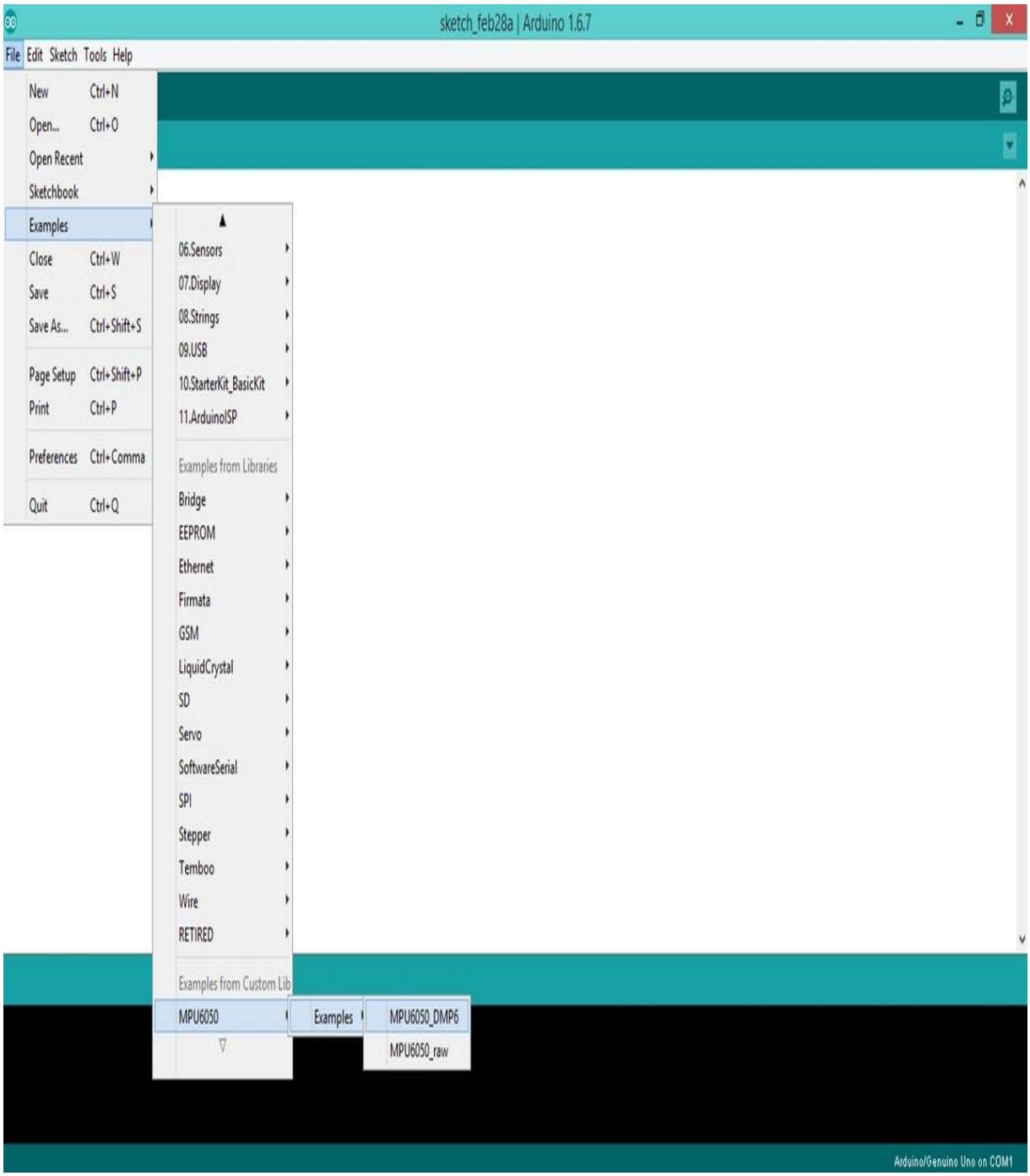


Fig.III.23 : La fenêtre de l'IDE Arduino pour rétablir le code de configuration.

Ensuite, on inclue les bibliothèques "I2Cdev" et "MPU6050" dans ce sketch.

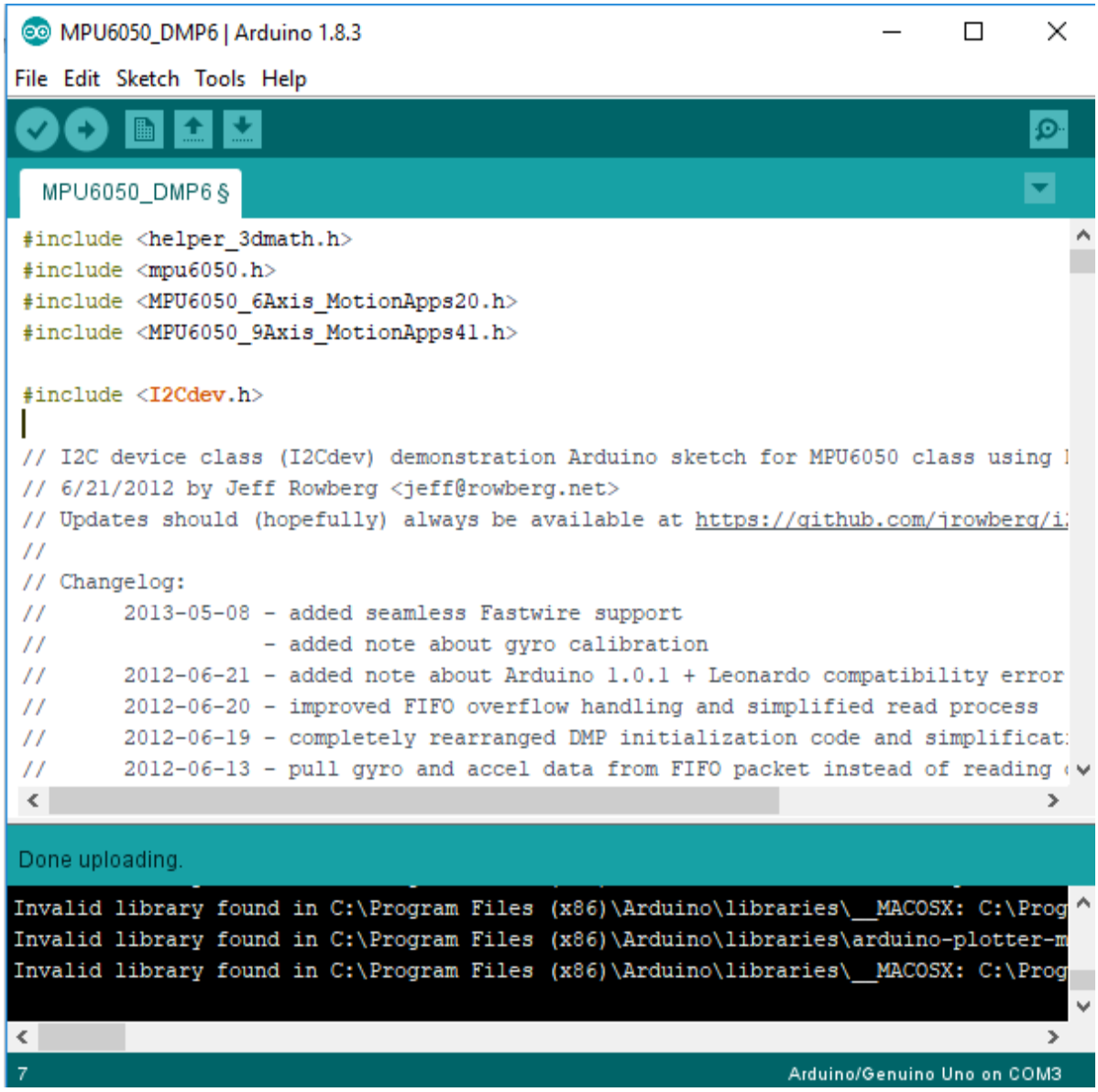


Fig.III.24 : l'inclusion des bibliothèques « I2Cdev » et « MPU6050 ».

Après avoir suivi toutes les étapes, on compile le sketch.

Fig.III.25 : La compilation du skectch.

Apres avoir compilé le skectch on aura cette fenêtre qui signifie que notre capteur MPU6050 est prête à être manipuler.

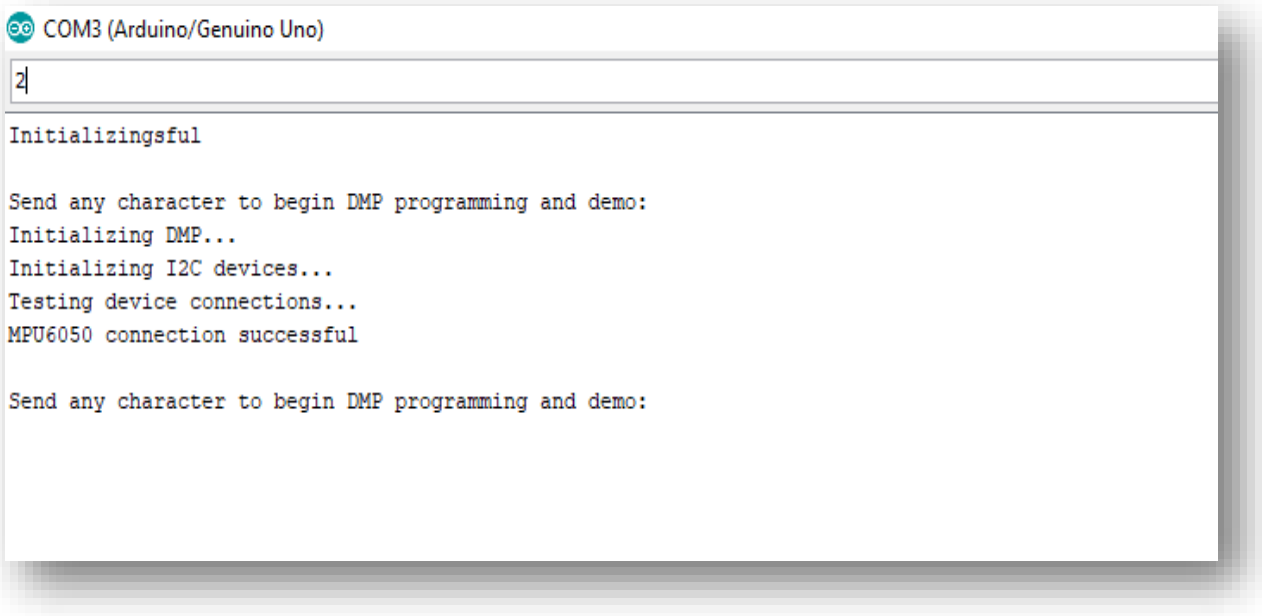


Fig.III.26 : le test de fonctionnement de l'MPU6050.

On voit une ligne qui indique “Send any character to begin DMP programming and demo”

Alors maintenant que nous avons configuré le matériel et logiciel, il est temps de compiler le programme sous Arduino IDE, et le téléverser vers la carte pour l’exécution via le câble USB.

Après la compilation du code on aura les valeurs des degrés tridimensionnels qui sont Phi, Thêta, Psi suivant les axes Roulis (**R**oll), Tangage (**P**itch), Lacet (**Y**aw), sur MPU6050 en utilisant le moniteur série de l’IDE Arduino.

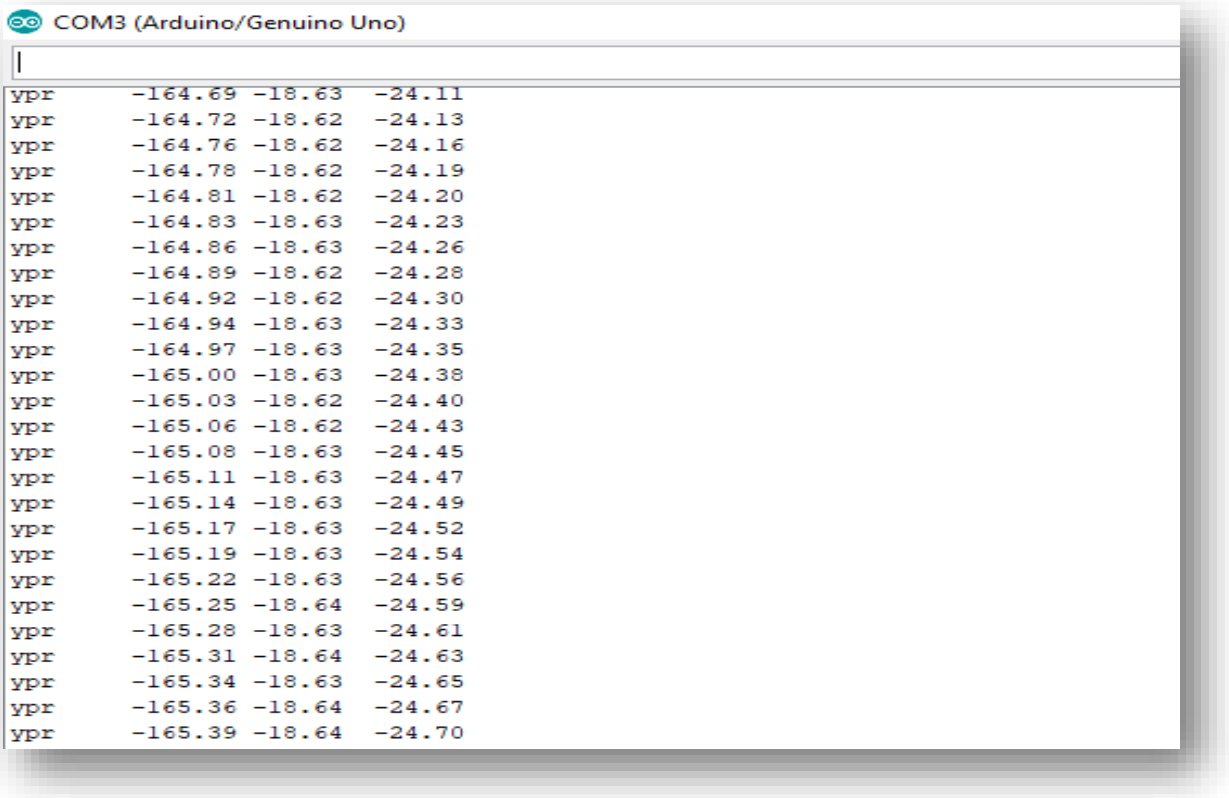


Fig.III.27 : Valeurs retournées par le module MPU6050.

Après avoir configuré notre capteur, on passe à l’étape de calibration comme suit :

- **Calibration du MPU-6050**

  La calibration est une étape indispensable pour avoir de un bon résultat de calcul, il sert à fixer les références (0, 0,0) du module. Pour calibrer notre module, nous avons appliqué l’algorithme suivant [28] :

- Mettre le module sur une plate horizontale.
- Mettre tous les offsets à 0

("SetXGyroOffset/setYGyro.ffset/setZGyroOffset/setZAccelOffset" =0).

- Passé le programme à Arduino et nous ouvrons le moniteur en série pour voir les valeurs retournées.
- Laisser le module opérant pour quelques minutes (5-10) donc la température est stabilisée
- Vérifier les valeurs dans le moniteur en série et on note-les.
- Mettre à jours nos offsets et on passe le programme à nouveau à la carte.
- Répéter ce processus jusqu'à ce que notre programme retourne à zéro pour chaque gyros, à zéro pour l'accéléromètre suivant X et Y et +16384 suivant Z.

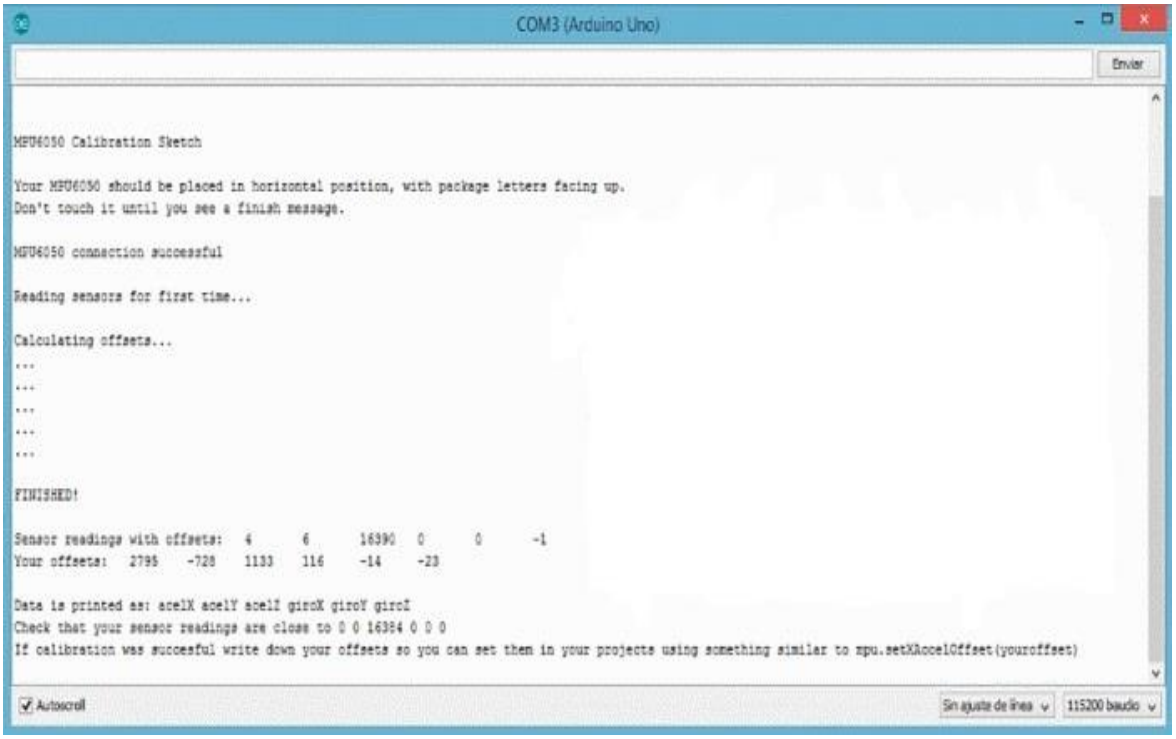


Fig.III.28 : Résultat de la calibration.

Pour la figure ci-dessus, les décalages d'étalonnage sont atteints pour les 6 degrés de liberté ; les trois premiers décalages étaient pour l'accéléromètre (X, Y, Z) et les trois derniers pour le gyroscope.

L'échelle de sortie pour n'importe quel réglage est [-32768, +32767] pour chacun des six axes. Le paramètre par défaut dans la classe I2Cdevlib est +/- 2g pour l'accélération et +/- 250 deg / sec pour le gyroscope. Si l'appareil est parfaitement horizontal et ne bouge pas, on aura [28] :

• Les axes X / Y doivent être 0

• L'axe Z accel doit indiquer 1g, soit +16384 pour une sensibilité de 2g

• Les axes gyroscopiques X / Y / Z doivent indiquer 0.

Ensuite, ces décalages résultants ont été pris et remplacés dans le code MPU6050, comme indiqué ci-dessous pour obtenir un résultat et une simulation précis. Ainsi, l'étalonnage est compensé par le code :

```
// supply your own gyro offsets here, scaled for min sensitivity
mpu.setXGyroOffset(116);
mpu.setYGyroOffset(-14);
mpu.setZGyroOffset(-23);
mpu.setXAccelOffset(2795);
mpu.setYAccelOffset(-728);
mpu.setZAccelOffset(1133); // 1688 factory default for my test chip
```

Fig.III.29 : Les résultats de calibration remplacée dans le code.

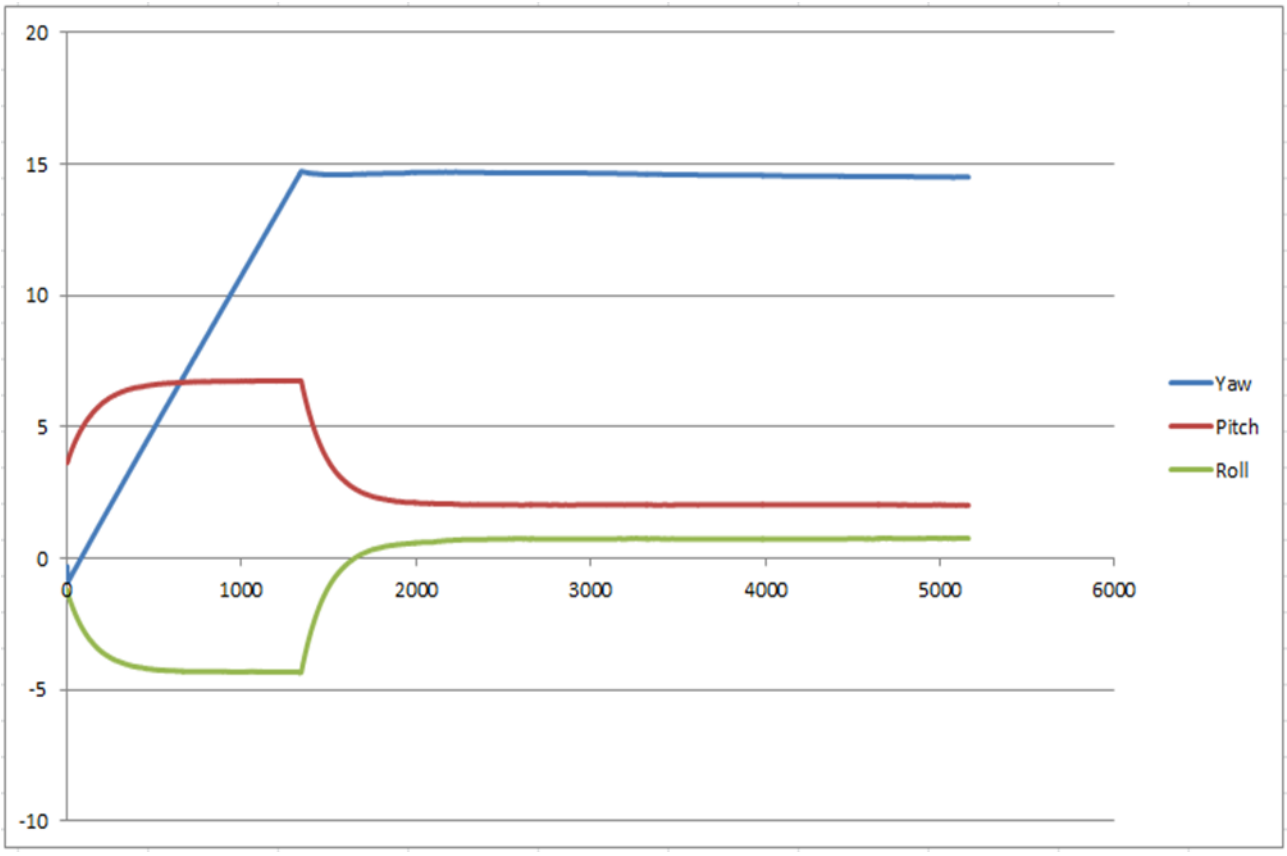


Fig.III.30 : les mesures avant calibration.

Pendant 10 à 20 secondes, on constate que la valeur du lacet augmente, puis, d'un coup, elle chute et fini par se stabiliser. Les roulis et tangage conserve leur valeur initiale pendant la même durée puis se stabilisent.

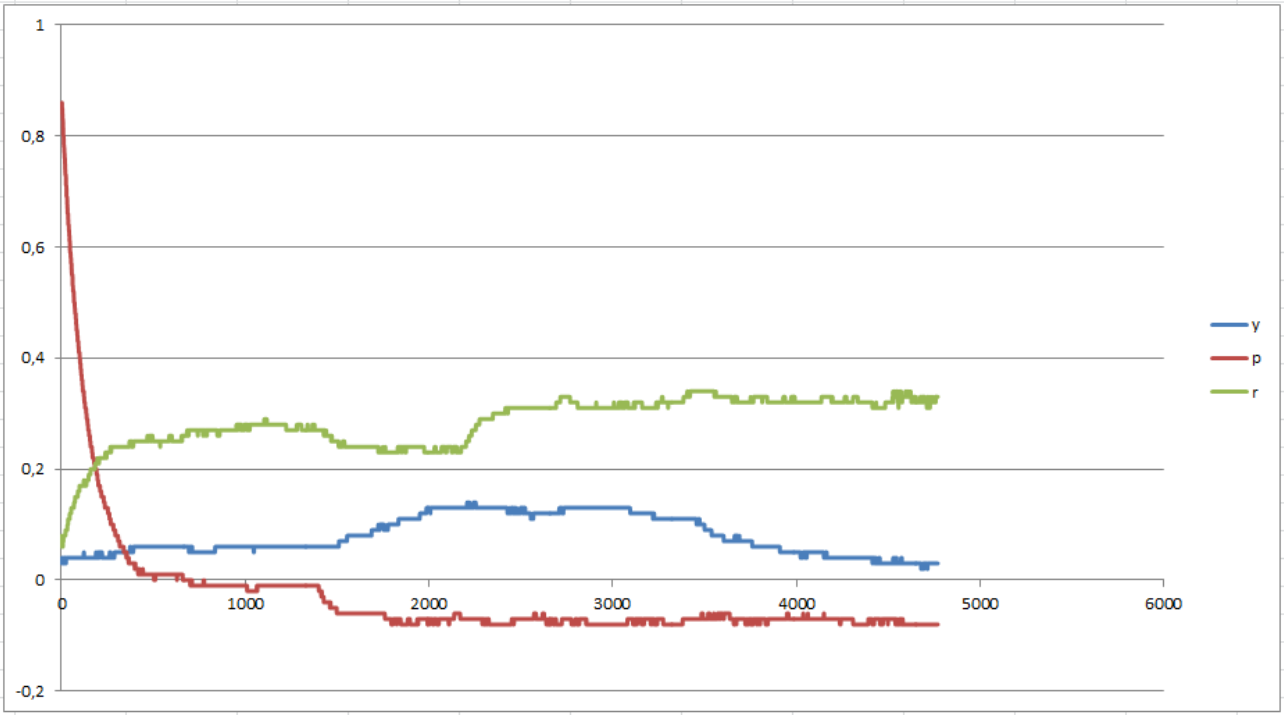


Fig.III.31 : la stabilité des mesures après calibration.

Apres avoir calibré notre capteur on a réussi à obtenir des valeurs proches de 0 et plus stable.

III.8.7. Interface graphique sous forme d'instruments de vol artificiels

Afin de faciliter la présentation des paramètres d'attitude aux utilisateurs une interface graphique qui affiche les données en temps réel a été ajoutée au prototype.

L'IDE Arduino ne fera qu'acquérir les données, pour afficher l'animation graphique nous aurons besoin de logiciel Processing.

Le Processing est assez similaire à L'IDE d'Arduino à l'exception de quelques fonctions spécialisées. Donc, on verra une influence, similitude dans l'IDE Processing.

Figure III.32 fera nos déclarations plus claires.

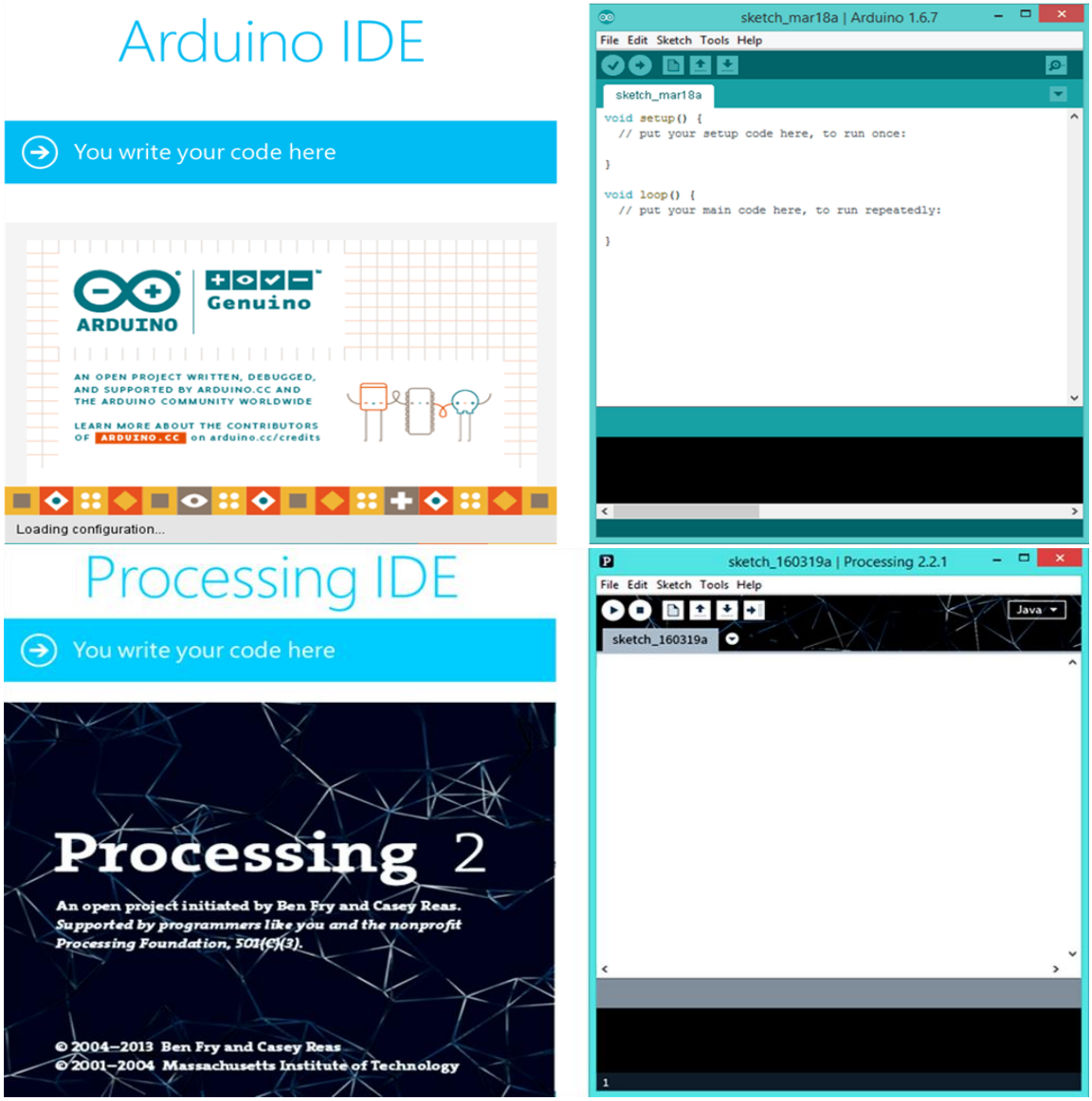


Fig.III.32 : la similitude de l'IDE Arduino et l'IDE Processing.

Donc nous voyons une similitude visuelle étonnante dans ces deux IDE.

- **L'installation du Processing :** L'installation proprement dite de Processing est assez simple et dépend de la plate-forme utilisée. Sous Windows par exemple. En cliquant sur l'icône Windows dans la page de téléchargement du site de Processing, vous allez télécharger une archive processing-xxxwindows.zip. Une fois le téléchargement achevé, décompressez l'archive et placez le dossier Processing extrait de l'archive dans le dossier C:\Program Files\. Allez ensuite dans le dossier C:\Program Files\Processing et exécutez le fichier processing.exe en cliquant dessus.

Maintenant que nous avons installé l'IDE de Processing, On doit télécharger une bibliothèque spéciale nommée "Toxi" pour des objectives du traitement.

- **Bibliothèque Toxi pour Processing IDE**

Trouver un fichier "toxiclibs-complete-0020" est très trépidant sur les sites web. Après avoir trouvé le fichier de la bibliothèque "Toxi" on extrait le dossier "toxiclibs-complete-0020" et On le Copier-coller dans le dossier "librairies" de Processing cette fois ci, au lieu d'Arduino.

Toxi : Est une collection de bibliothèque open source indépendante pour les tâches de conception informatique avec Java & Processing développé par Karsten "toxi" Schmidt (à ce jour). Les Cours sont délibérément maintenus assez génériques afin de Maximiser la Réutilisation dans différents contextes allant de la conception générative, de l'animation, de la conception d'interactions / d'interfaces, de la visualisation de données à l'architecture et à la fabrication numérique.

Après avoir configuré les composants selon le schéma de connexion conçu au début de la phase de réalisation, et avec l'utilisation du Processing on va procédée à l'étape de la conception de l'interface graphique.

Conception de l'interface graphique

- ✓ **Horizon artificiel**

On va décrire quelques étapes de base pour la création de l'interface :

Apres avoir lancé le Processing, on doit d'abord configurer la dimension de notre cadre de dessin (elle dépend de la taille de l'écran de notre ordinateur) et cela en utilisant la méthode **size** (,). La taille de notre fenêtre de dessin est 1400 pixels par 700 pixels, donc on doit indiquer **size** (1400,700)

Ensuite on doit régler la couleur du fond de notre fenêtre on doit appeler à la méthode **background,** background peut être écrit de deux façons si on écrit background(0); on obtiendra un fond noire, et cela contrôle uniquement les intensités allant de noir au blanc en passant par les différents niveaux gris possibles (entre 0 et 255 il y a 254 différents niveaux de gris, pour être exacte) (le gris est la couleur original du fond), sinon on peut ecrire **background**(0,255,0) trois paramètres au lieu d'un paramètre cette configuration donnera 100% d'intensité pour la couleur verte, et 0% d'intensité pour le reste.

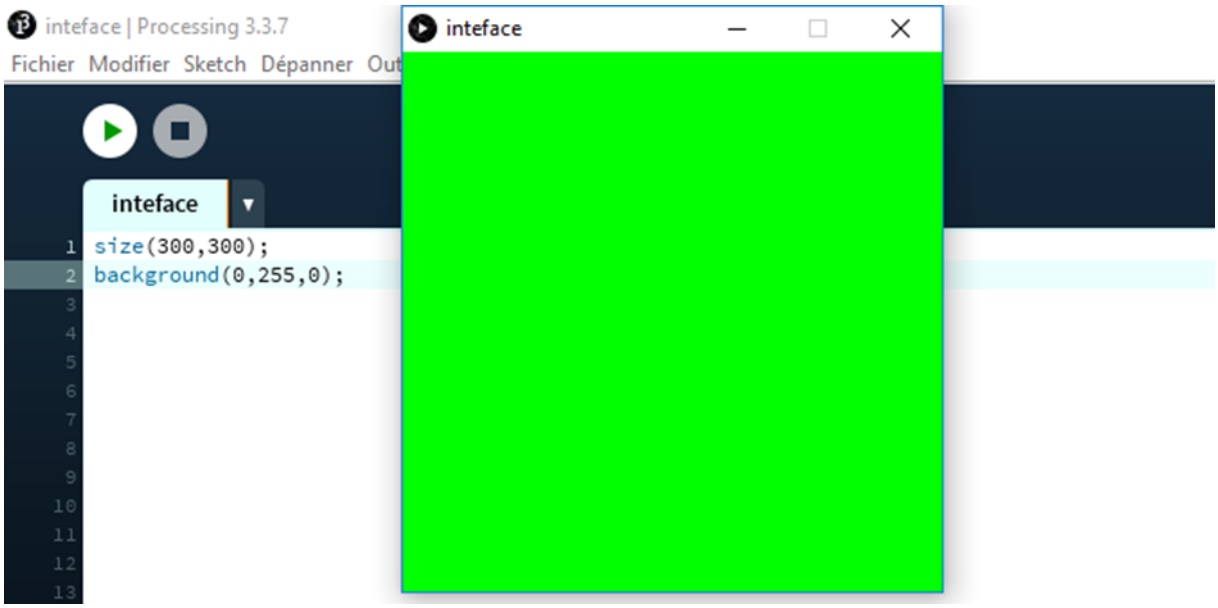


Fig.III.33 : Exemple de la configuration de la couleur de la fenêtre.

On peut même mélanger les couleurs (utiliser plusieurs couleur en même temps) par exemple **background** (0, 255,255) indique la couleur bleu vert.

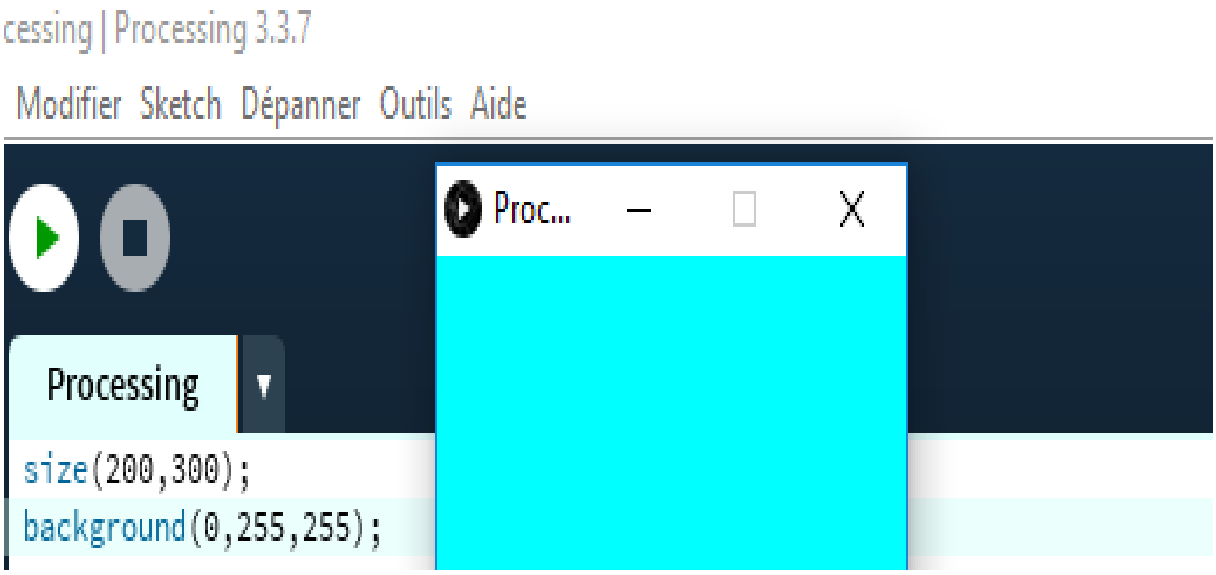


Fig.III.34 : exemple descriptif de la fusion des couleurs du fond.

Maintenant on va procéder la conception du l'horizon artificiel, Et cela en utilisant deux méthodes au premier lieu la méthode rect (0) et la méthode **fill** (), Dans notre cas **rect** (0, -100, 900, 1000); indique qu'on va dessiner à partir du point 0,-100 de notre cadre, et à partir

de cet point (x)0, (y)-100 on dessine un rectangle de 900 pixels de largeur et 1000 pixels de hauteur.

Processing comme beaucoup d'environnement de programmation fait une distinction entre le trait qui tourne autour d'une forme, et la couleur qui compose son intérieur. C'est pour cela on doit ajouter la méthode **noStroke** () et **fill** () pour avoir un rectangle rempli par la couleur bleu et sans bordures **noStroke** ().

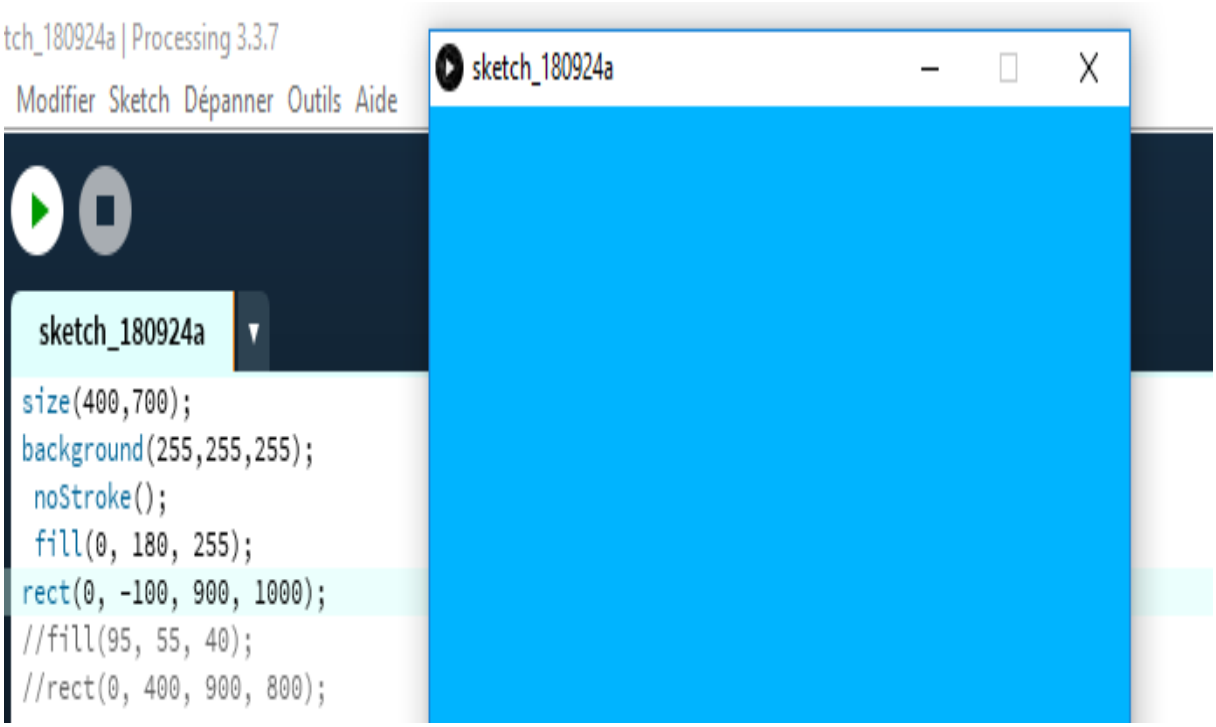


Fig.III.35 : la configuration du couleur du fond en bleu.

Ensuit pour faire diviser la surface de la fenêtre en deux on doit ajouter un autre rectangle et jouer sur le dimensionnement de notre cadre en utilisant la méthode **rect** (0, 400, 900,800) ; ensuite on le rempli par la couleur marron.

On aura ensuite les deux étages de l'horizon artificiel le bleu et le marron comme indiquée sur la figure III.36 suivante.

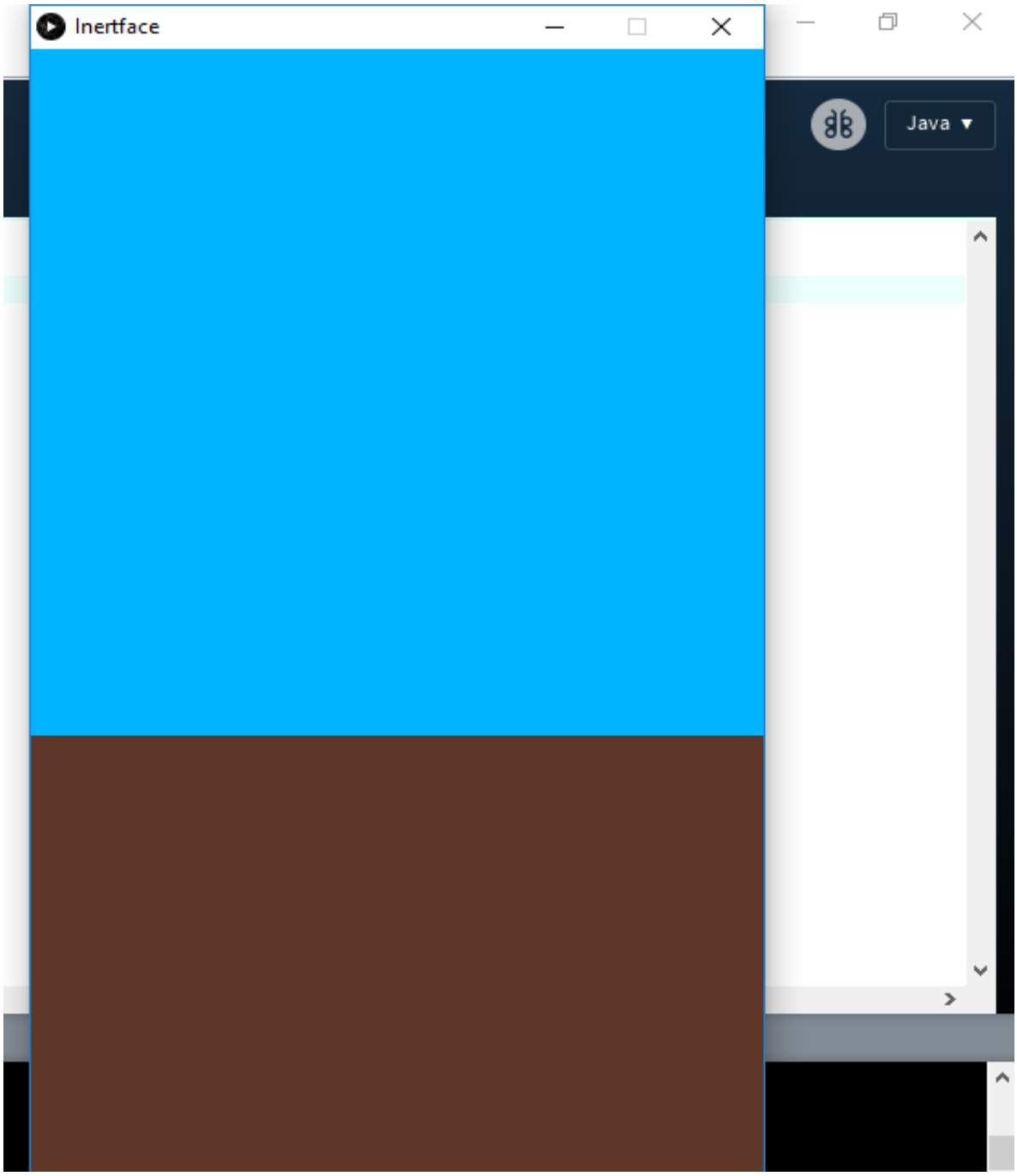


Fig.III.36 : la configuration de la forme et les couleurs de l'horizon artificiel.

Voilà donc maintenant on a les deux couleurs de l'horizon artificiel. On a donc un rectangle à deux étages Bleu qui indiquent le ciel et le marron qui indique la terre.

Maintenant on va ajouter les lignes (les trais), sous forme d'un cercle. On peut voir sur l'écran de l'horizon artificiel qu'il Ya des grandes lignes, puis des petites lignes et ainsi de suite il suffit de placer les lignes dans le cercle et de les espacer tous du même angle.

Ceci est le code pour placer les petites fines lignes et les grandes lignes sous forme d'un cercle pour la graduation de l'horizon artificielle.

```
  float angleStep = HALF_PI / 30; //L'angle que déplace le programme à chaque fois (HALF_PI / 30 intervalles de radians)
  boolean longStroke = false; //La ligne dessinée est longue.
  translate(width/2, height/2);
  for (float angle = 0; angle < TWO_PI; angle += angleStep) { // Parcourt tous les angles
    rotate(angleStep);
    if (longStroke) {
      line(0, -200, 0, -180);
    } else {
      line(0, -190, 0, -180);
    }
    longStroke = !longStroke; //Définit "longStroke" à l'opposé de ce qu'il est maintenant, donc le suivant aura la longueur opposée
  }
}
```

Fig.III.37 : le code de la graduation du l'horizon artificielle.

```
183 void Maquette()
184 {
185   fill(0);
186   strokeWeight(1);
187   stroke(0, 255, 0);
188   triangle(-20, 0, 20, 0, 0, 25);
189   rect(110, 0, 140, 20);
190   rect(-110, 0, 140, 20);
191 }
```

Fig III.38 : le code de la maquette d'avion du l'horizon artificielle.

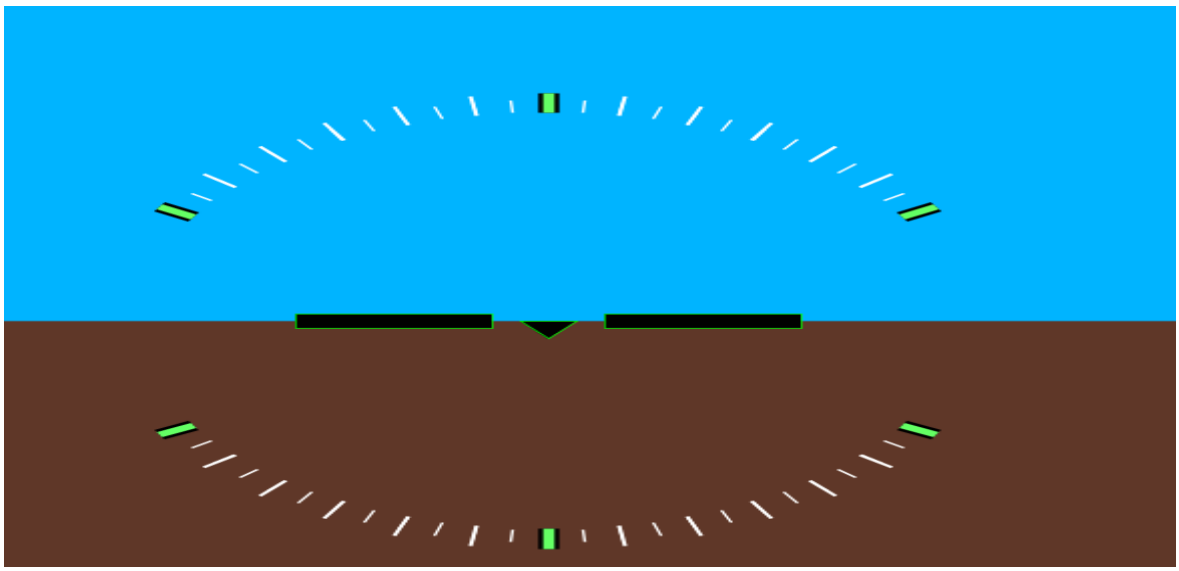


Fig.III.39 : l'interface graphique de l'horizon artificielle graduée.

On peut rendre le rectangle qui désigne l'horizon artificiel sous forme d'un cercle aussi, cela peut être effectuée en utilisant la méthode **arc** () au lieu de **rect** (). Au premier lieu on doit dessiner le demi-cercle bleu qui désigne le ciel, et cela en utilisant cette séquence du code sous l'IDE Processing.

Horizon | Processing 3.3.7

Fichier Modifier Sketch Dépanner Outils Aide

Horizon

```
1  size(700,700);
2  background(255,255,255);
3  noStroke();
4  fill(0, 180, 255);
5  arc(200, 200, 400, 400, -PI,0);
6  //fill(95, 55, 40);
7  //arc(200, 200, 400, 400,0,radians(180));
8  //rect(0, -50, 900,500);
9  //fill(95, 55, 40);
10 //rect(0, 400, 900, 1000);
11 //rotate(-PI-PI/6);
12 //rotate(PI+PI/6);
13 //rotate(-PI/6);
14 rotate(PI/6);
15 //translate(700/2, 700/2);
16
17   //stroke(200);
```

Fig.III.40 : la séquence du code pour le demi-cercle de la forme de l'horizon artificielle graduée.

Voici notre premier demi-cercle bleu dans la figure

Fig.III.41 : le dessin de demi-cercle de l'horizon artificiel sous Processing.

Ensuite pour compléter la forme du cercle on ajoute un deuxième demi-cercle marron qui désigne la terre. La figure ci-dessous montre la séquence du code utilisée.

Horizon | Processing 3.3.7

Fichier Modifier Sketch Dépanner Outils Aide

Horizon

```
1  void setup() { size(1400,700);
2  background(255,255,255);
3  noStroke();
4  //fill(0, 180, 255);
5  //arc(200, 200, 400, 400, -PI,0);
6  fill(95, 55, 40);
7
8  arc(200, 200, 400, 400,0,radians(180));
9  //rect(0, -50, 900,500);
10 //fill(95, 55, 40);
11 //rect(0, 400, 900, 1000);
12 //rotate(-PI-PI/6);
13 //rotate(PI+PI/6);
14 //rotate(-PI/6);
15 rotate(PI/6);
16 translate(700/2, 700/2);
17
18
```

Fig.III.42 : le code de la création de demi-cercle marron sous Processing.

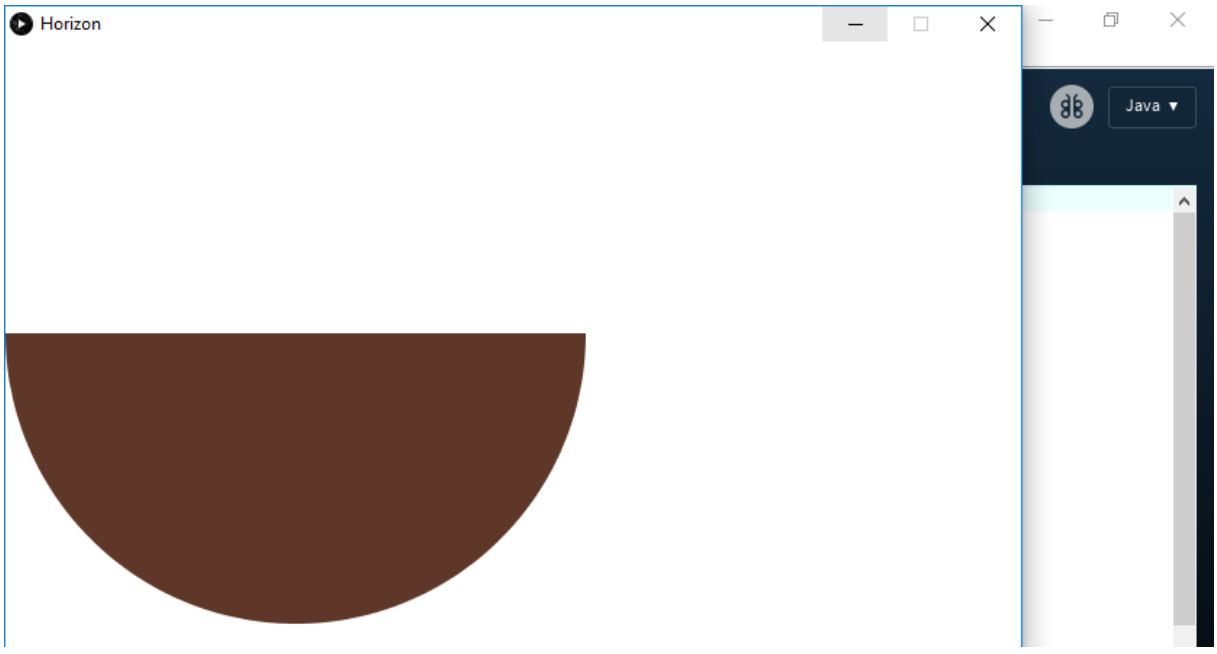


Fig.III.43 : le dessin de demi-cercle marron de l'horizon artificiel sous Processing.

Voilà maintenant nous avons obtenu le cercle complet montrant la combinaison de l'horizon artificiel

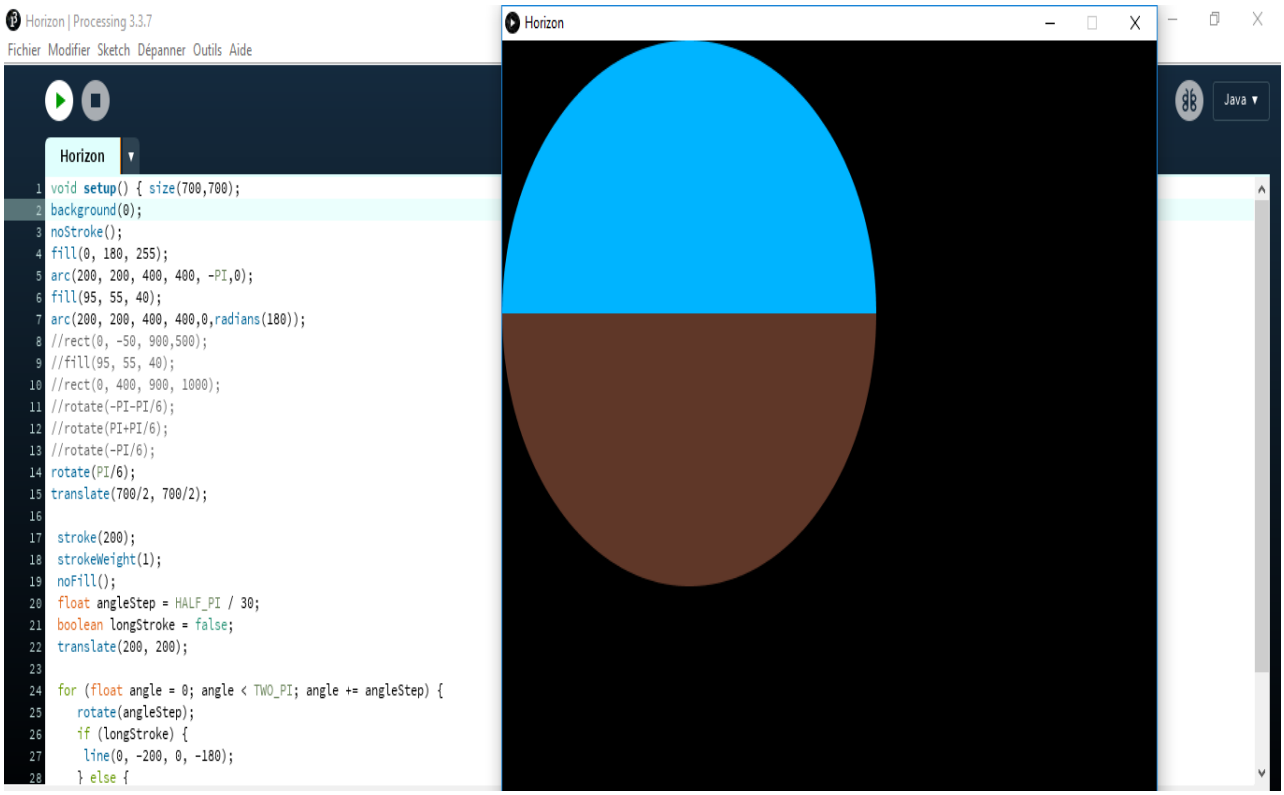


Fig.III.44 : La forme principale de l'horizon artificiel en utilisant la méthode arc () sous Processing.

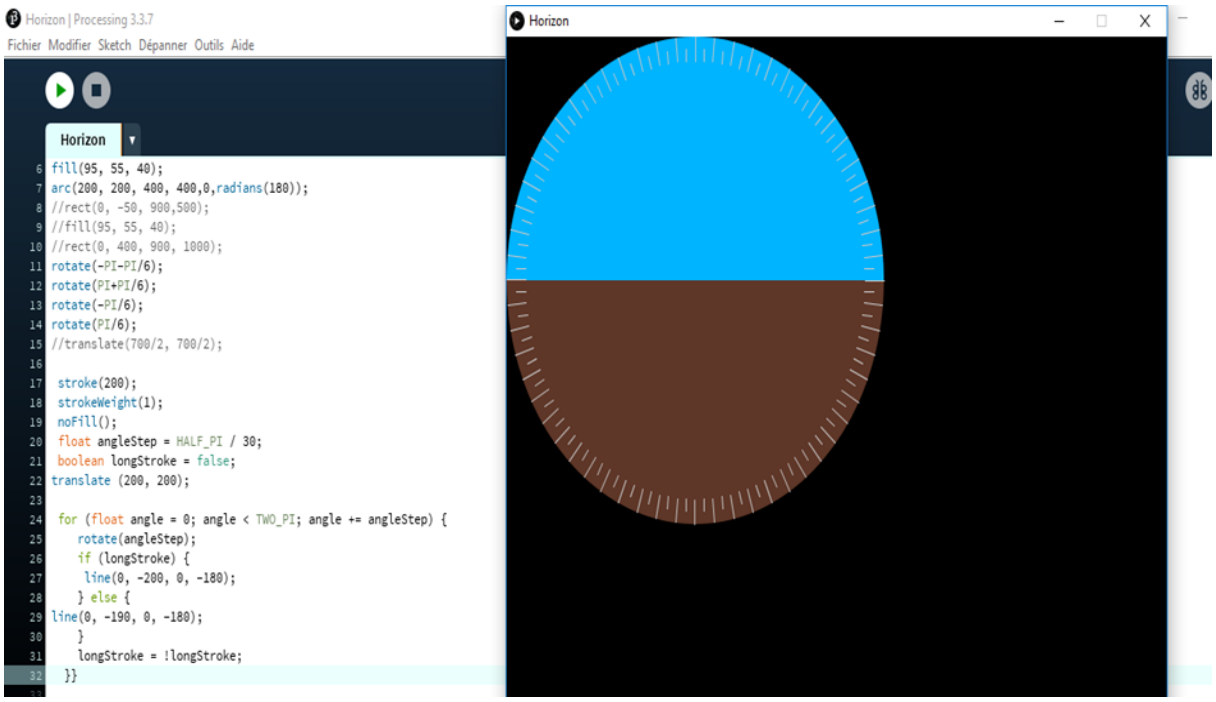


Fig.III.45 : La forme principale de l'horizon artificiel graduée.

Pour la phase de la graduation et l'ajout de la maquette d'avion on utilise les mêmes codes décrit précédemment (la conception de l'interface l'horizon artificiel en rectangle).

Le compas

Maintenant on passe à la conception du compas. En utilisant cette séquence de programme on peut avoir un cercle en utilisant plusieurs méthodes comme ellipse () et fill () pour remplir les cercle et les lignes aussi.

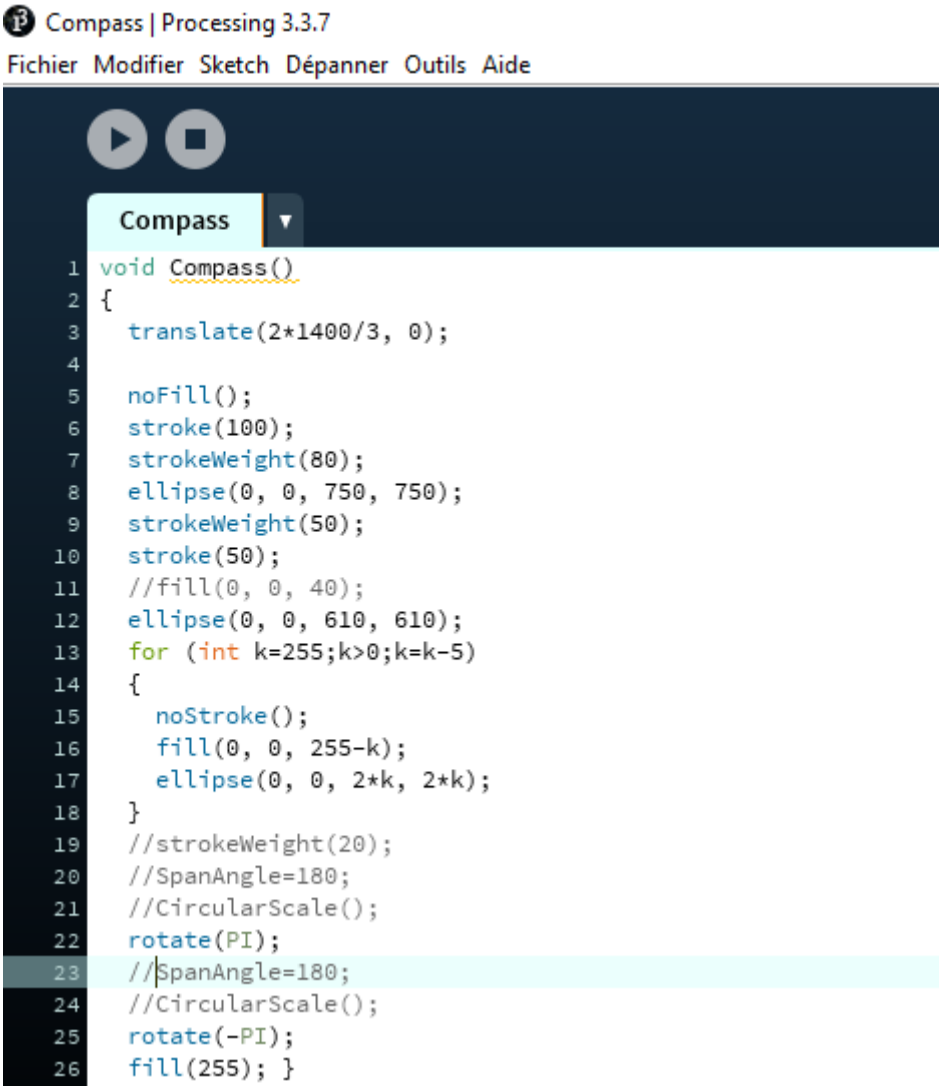


Fig.III.46 : La séquence de programme pour la création de l'interface principal du compas.

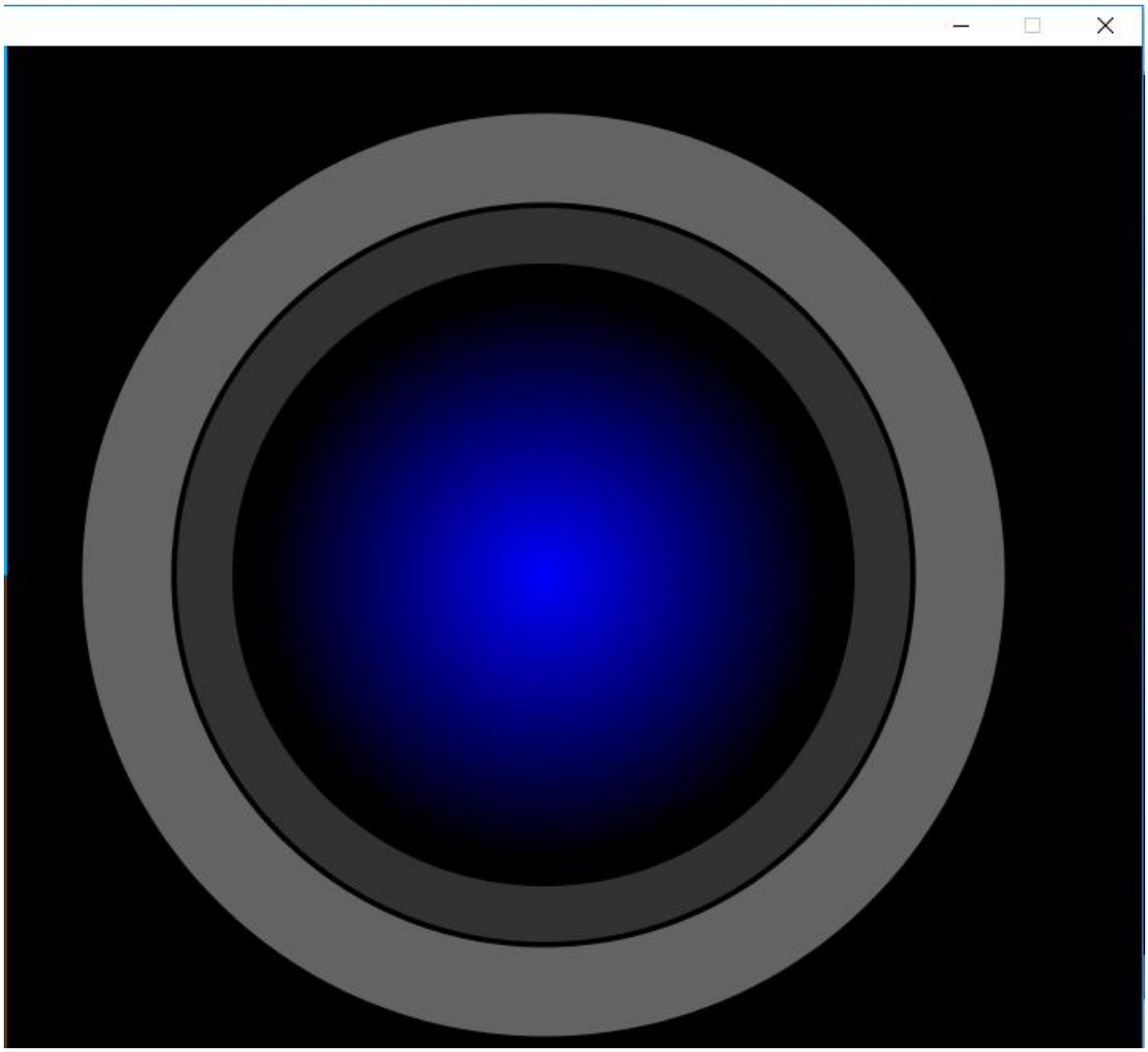


Fig.III.47 : La Forme principal du Compas.

Voici donc le format de notre compas sans aiguille et sans texte.

Maintenant on va ajouter ce groupe de mots sur le compas : « W », « E », « N », « S », « NE », « NW », « SE », « SW » et « COMPAS ».

La séquence du programme qui permet d'ajouter un texte est montrée sur la figure III.48 suivante :

Compas | Processing 3.3.7

Fichier Modifier Sketch Dépanner Outils Aide

Compas

```
160   textSize(30);
161   text("COMPASS", 0, -130, 500, 80);
162   rotate(PI/4);
163   textSize(40);
164  text("NW", -370, 0, 100, 50);
165   text("SE", 365, 0, 100, 50);
166   text("NE", 0, -355, 100, 50);
167   text("SW", 0, 365, 100, 50);
168   rotate(-PI/4);
169   CompassPointer();
170 }
171 void CompassPointer()
172 {
173   rotate(PI+radians(Azimuth));
174   stroke(0);
175   strokeWeight(4);
176   fill(100, 255, 100);
177  triangle(-20, -210, 20, -210, 0, 270);
178  triangle(-15, 210, 15, 210, 0, 270);
179   ellipse(0, 0, 45, 45);
180   fill(0, 0, 50);
181   noStroke();
182   ellipse(0, 0, 10, 10);
183   triangle(-20, -213, 20, -213, 0, -190);
184   triangle(-15, -215, 15, -215, 0, -200);
185   rotate(-PI-radians(Azimuth));
186 }
```

Fig.III.48 : le code du texte et le pointeur de compas.

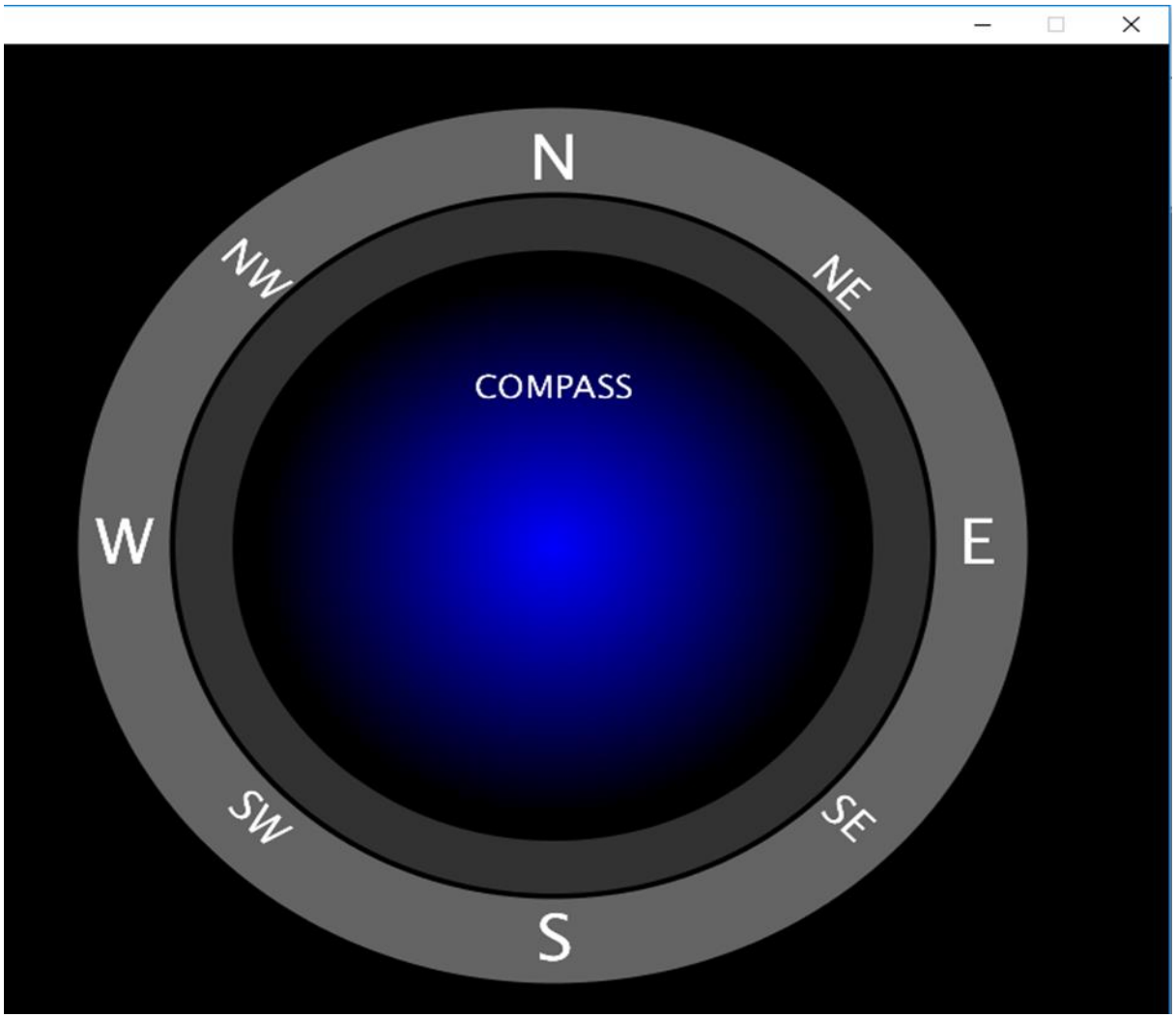


Fig.III.49 : L'interface de compas avec textes.

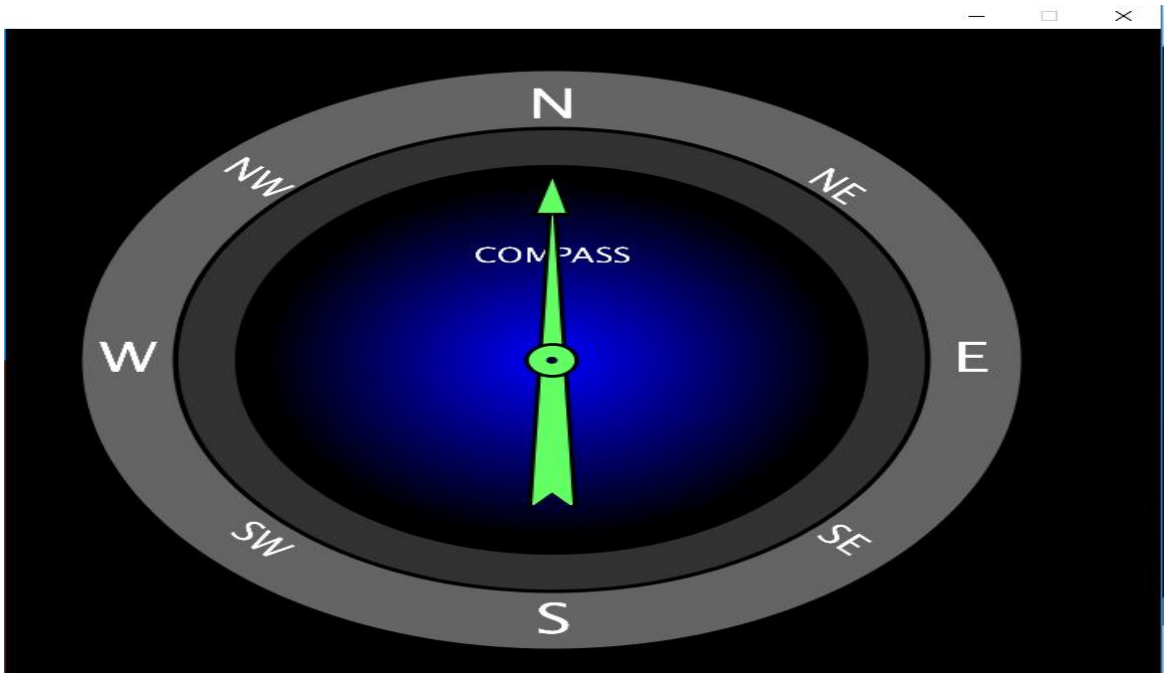


Fig.III.50 : L'interface de compas avec textes et l'aiguille.

Maintenant il nous reste qu'à ajouter le mot « Azimuth » ci-dessus et notre compas est prêt.

Voici la séquence du code utilisée :

```
108 void Azimuth()
109 {
110   fill(50);
111   noStroke();
112   rect(20, 470, 440, 50);
113   int Azimuth1=round(Azimuth);
114   textAlign(CORNER);
115   textSize(35);
116   fill(255);
117  text("Azimuth:  "+Azimuth1+" Deg", 80, 477, 500, 60);
118   textSize(40);
119  fill(25,25,150);
120
121 }
122
```

Fig.III.51 : La séquence du code pour l'ajout de mot « Azimuth ».

Voici donc notre compas à la fin.

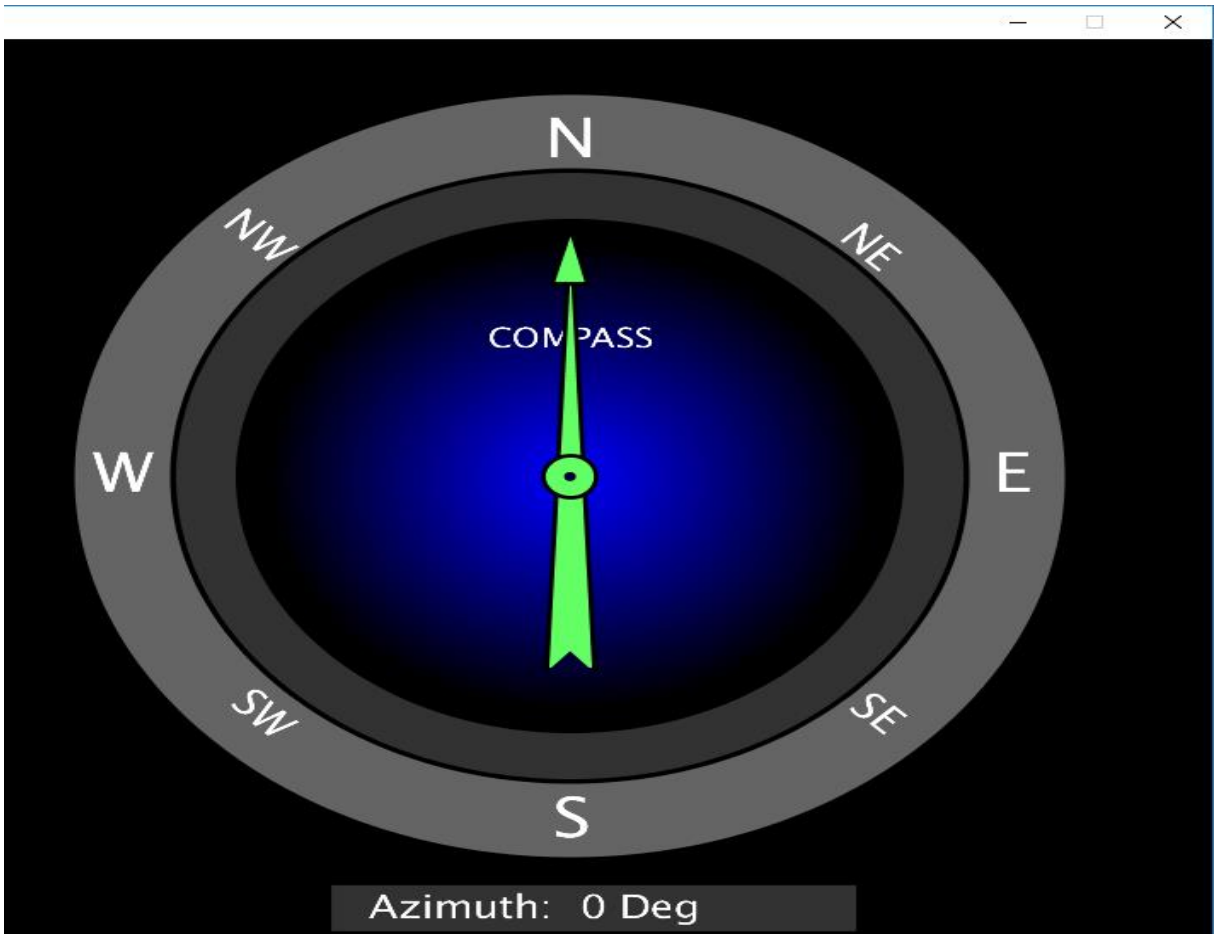


Fig.III.52 : l'interface graphique du compas.

Voici l'interface finale des deux instruments, on ajoutant un texte spécial « SIMULATEUR IAES BLIDA ».

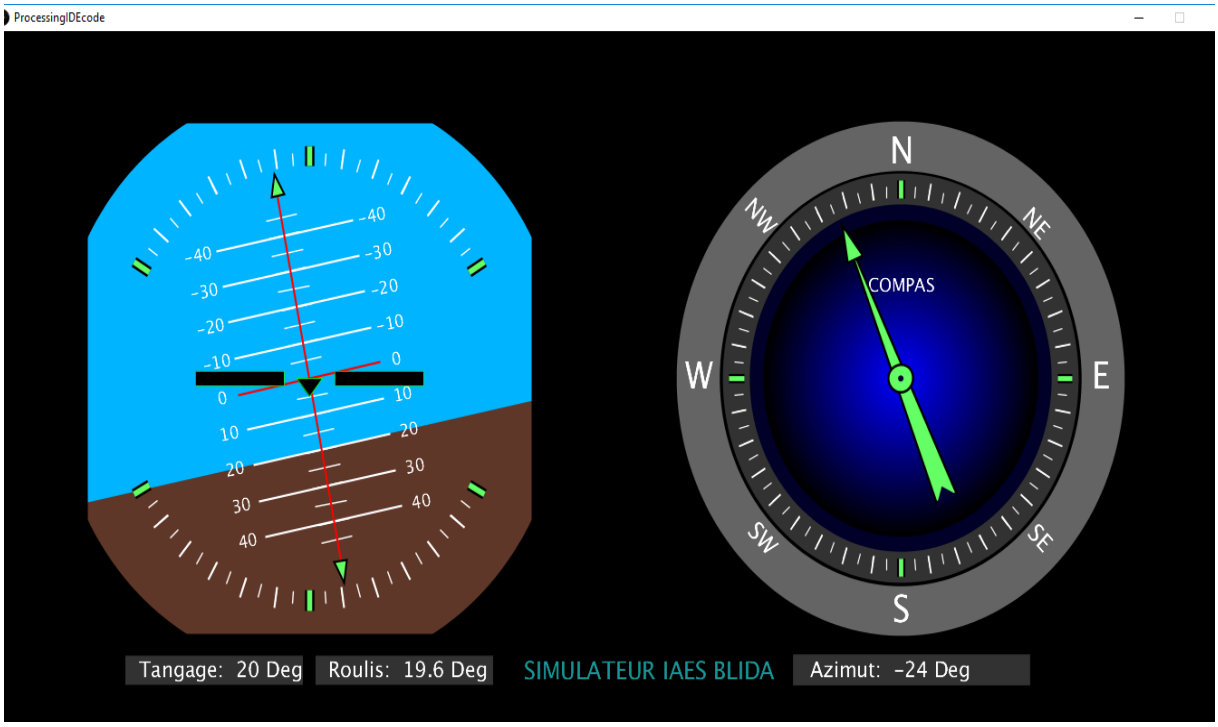


Fig.III.53 : l'interface graphique pour l'affichage des données nécessaires d'attitude en temps réel.

Apres avoir envoyé les données à partir le bus série d'Arduino nous configurons Proccesing pour communiquer avec le même bus série, et on verra l'animation des deux instruments du vol l'horizon artificiel et le compas. On doit attendre environ 7 secondes pour que le system s'étalonne et les valeurs du MPU6050 se stabilisent. Après quoi, on verra le modèle de simulateur du vol s'interagisse en conséquence avec le capteur MPU6050, l'animation effectue facilement des mouvements et des rotations de n'importe quel angle.

L'affichage de l'angle du tangage, l'angle de roulis, et la direction est présenté dans l'interface tels qu'il est dans un avion réel.

La Figure III.53 illustre bien l'interface graphique du simulateur du vol personnalisée sous forme de :

- Horizon artificiel.
- Compas.

III.9. Indication de l'horizon artificiel pour différentes positions de l'avion :

Notre modèle de simulateur du vol s'interagisse en conséquence avec le capteur MPU6050,

Et voici quelques indications de simulateur pour différentes position de l'avion :

- ✓ ***Assiette cabré :***

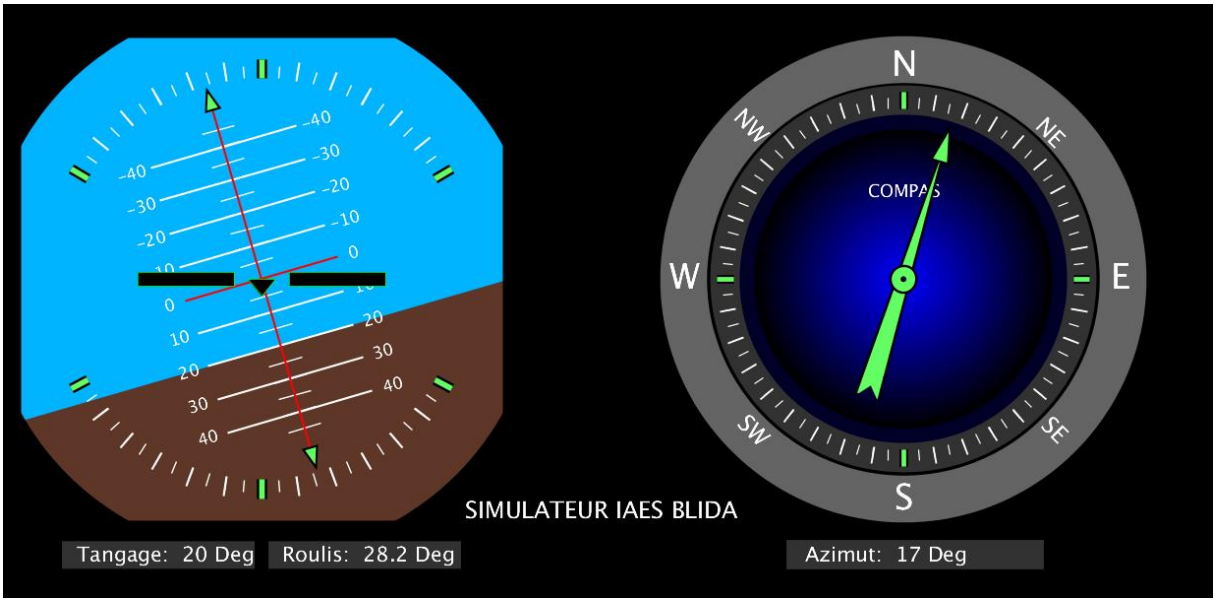


Fig.III.54 : Assiette cabré a 20 degré virage à droite.

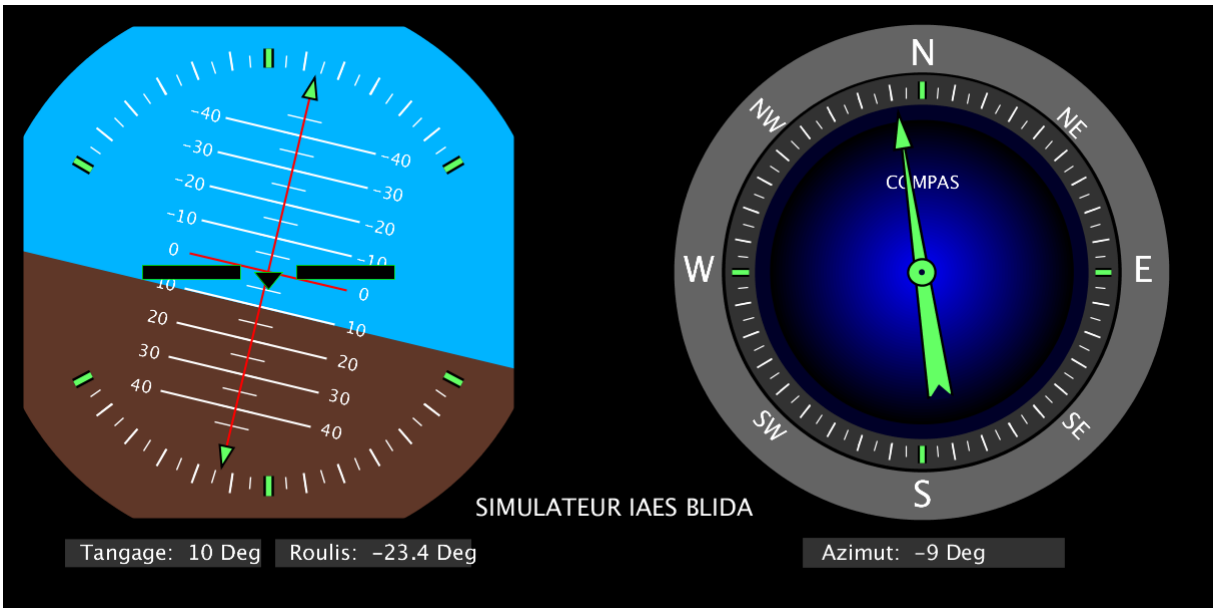


Fig.III.55 : Assiette cabré a 10 degré virage à gauche.

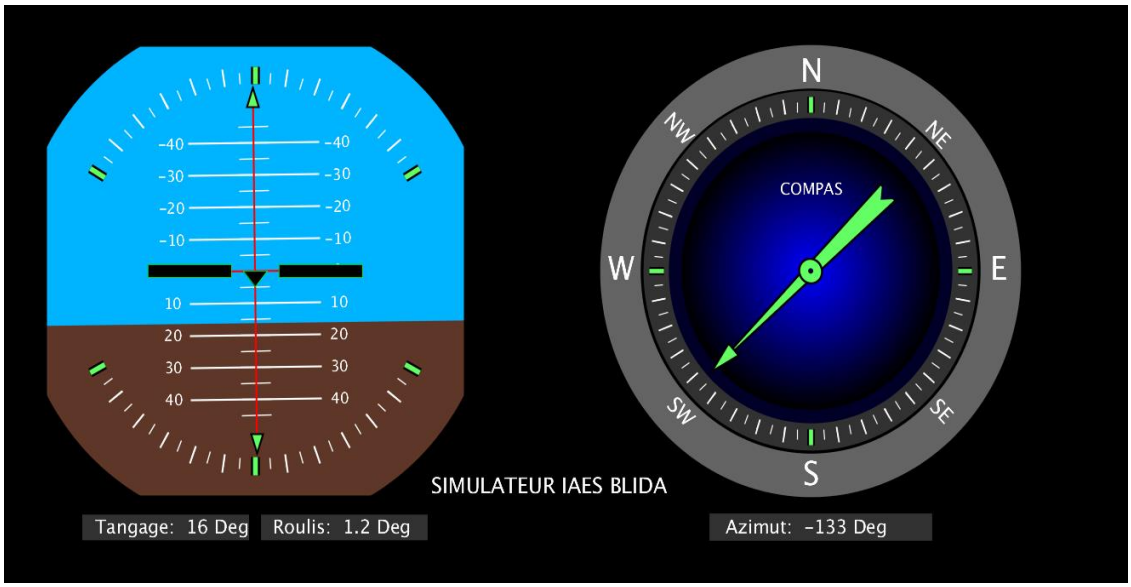


Fig.III.56 : Assiette en cabré à 16 degré vol rectiligne.

- ✓ ***Vol en palier***

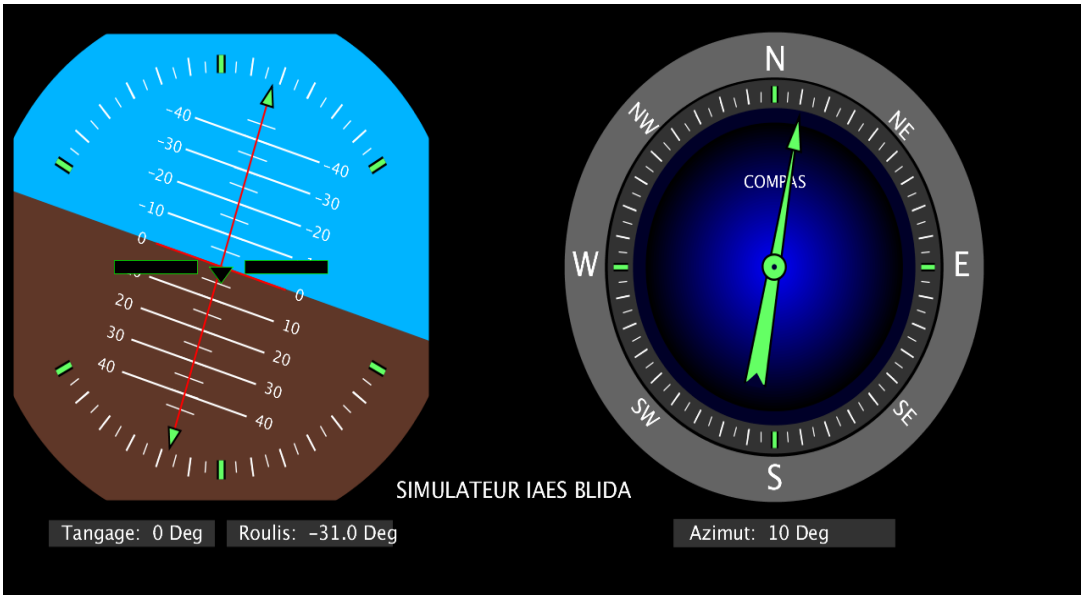


Fig.III.57 : Virage à gauche.

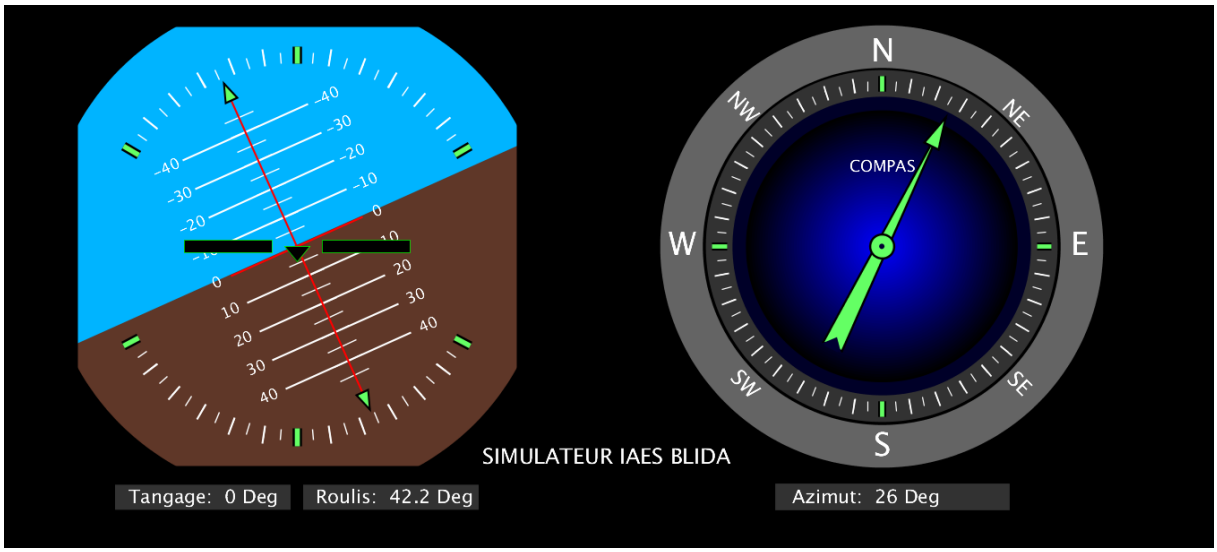


Fig.III.58 : Virage à droite.

✓ Vol piqué

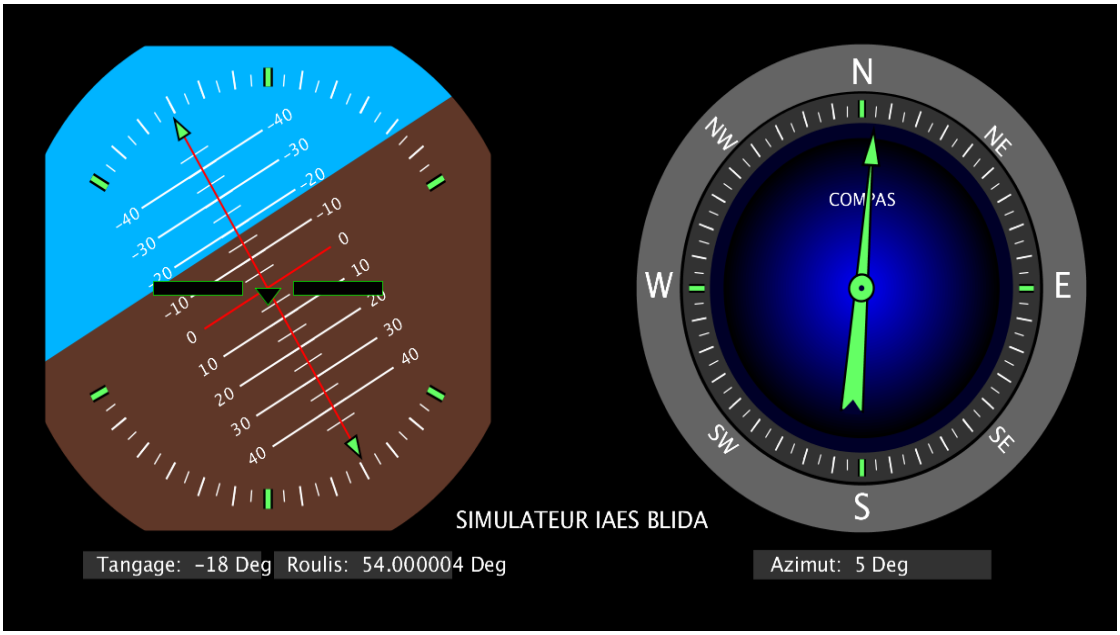


Fig.III.59 : Virage à droite piqué.

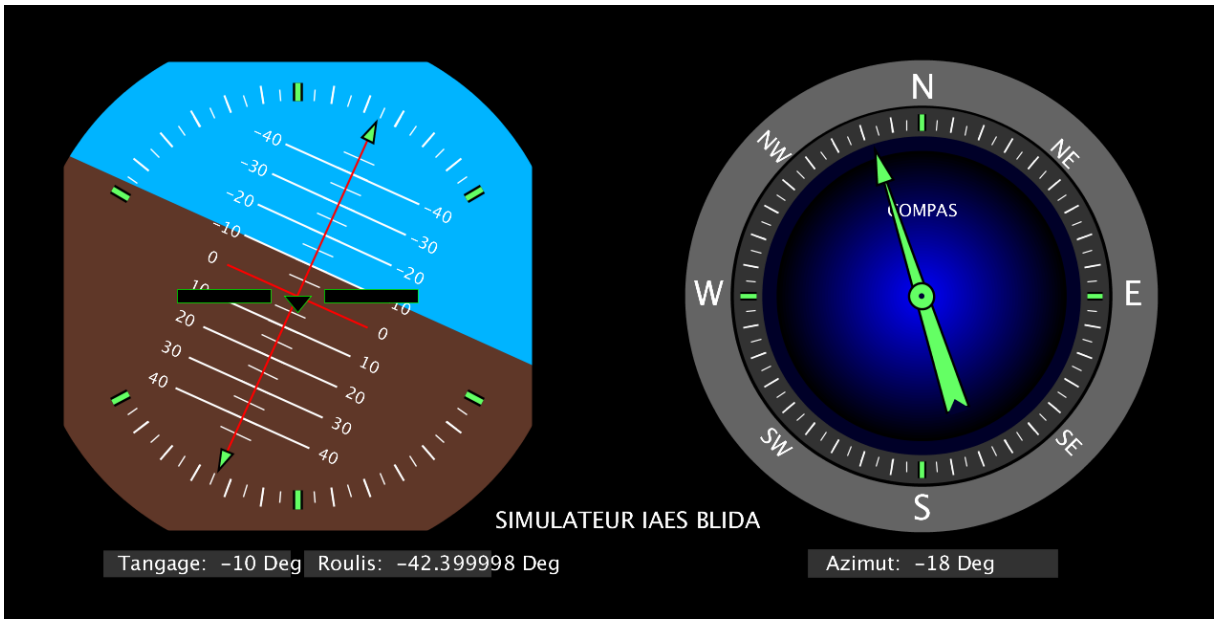


Fig.III.60 : Virage à gauche piqué.

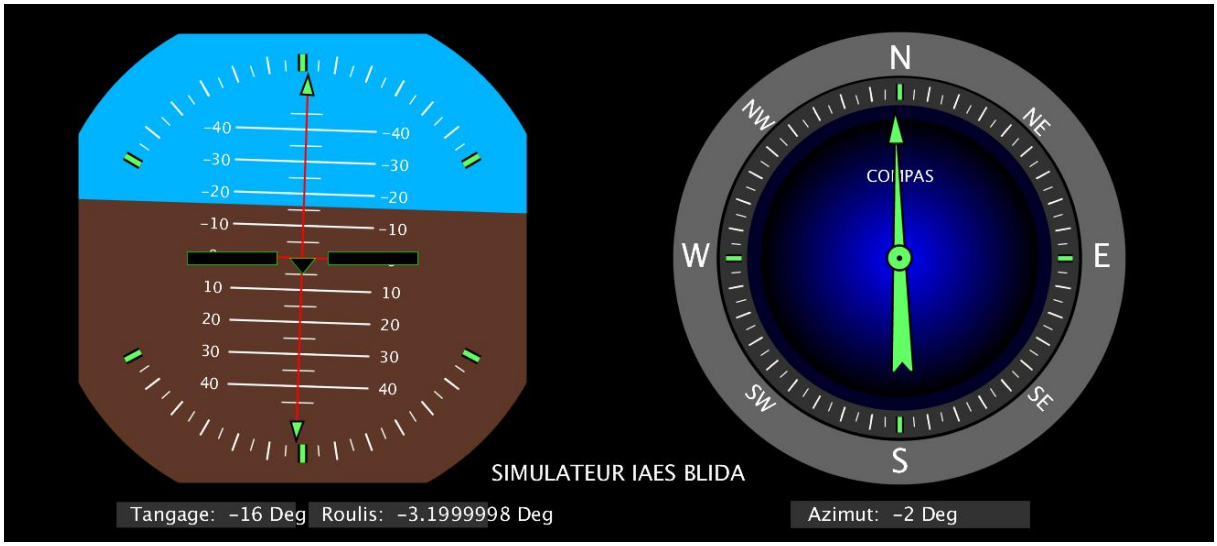


Fig.III.61 : Vol rectiligne piqué.

III.10. Conclusion

Dans ce chapitre, nous avons abordé le choix de la technologie (MEMS), les protocoles de communication, et le choix du microcontrôleur adéquat.

On a aussi donné lieu à la présentation de différentes étapes de réalisation pratique du prototype présentant notre système de Central inertielle.

Nous avons aussi projeté la lumière sur la partie logicielle utilisée pour la réalisation et la simulation dans notre projet.

Puisque les MEMS accéléromètre et gyroscopes, présentent de nombreuses erreurs de mesure, on a passé par l'étape d'étannolage du capteur.

Ensuite on a expliqué la phase de la création de l'interface graphique du simulateur du vol personnalisée en détails.

Conclusion général

La réalisation de ce projet nous a donné la chance de découvrir le monde passionnant de l'avionique embarqués, ce domaine qui nous a permis de toucher au monde réel, et d'aller de la théorie vers la pratique.

L'objectif principal de ce travail était la réalisation et la conception d'un prototype de navigation inertielle autonome afin d'estimer en temps réel l'attitude de l'aéronef.

Un capteur IMU, un microcontrôleur Arduino, le logiciel IDE de l'Arduino et le logiciel de simulation « Processing » étaient suffisants pour représenter un système de navigation simple et intuitif et d'effectuer des tests de simulation via une interface d'animation.

Afin d'évaluer le fonctionnement de notre prototype un modèle de calibrage pratiquement efficace est généralement requis a été conçus pour notre capteur IMU.

De plus, pour rendre notre système exploitable, une animation graphique a été ajouter au prototype et cela en combinaison les deux logicielle Arduino IDE et la plateforme Processing. Cette présentation graphique a été conçue sous forme d'un system AHRS (Attitude Heading Reference Système) pour un simulateur de vol personnalisé, avec les fameux instruments l'horizon artificielle et le compas.

D'après les résultats de simulation obtenus et malgré les incertitudes du calcul, les imperfections des capteurs, et les erreurs produites par le protocole de communication qui font dégradées la précision de notre système, on peut dire que notre réalisation a assuré un bon fonctionnement et a reproduit le comportement réel en attitude de la centrale inertielle.

Il est clair que de telles réalisations constituent une contribution modeste qui peut aller vers de nombreuses améliorations grâce à l'évolution technologique rapide et croissante des systèmes micro-électromécanique MEMS et les logiciels de simulation.

Perspectives :

Nous recommandons pour la suite de notre travail, la réalisation des autres interfaces du cockpit comme le PFD, EFIS, ECAM….etc. toute en améliorant notre mesure par l'intégration des filtres complémentaires

Références Bibliographiques

[1] *Estimation d'attitude et diagnostic d'une centrale d'attitude par des outils ensemblistes : NGUYEN Hoang Van, THÈSE Pour obtenir le grade de DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE, le 24 mars 2011.*

[2] *Innovation et réalisation d'une solution de géolocalisation combinée à la télécommunication : CHABBAR Houria, Mémoire de Projet de fin d'étude Préparé par Pour l'obtention du diplôme Ingénieur d'Etat en SYSTEMES ELECTRONIQUES & TELECOMMUNICATIONS, 2014.*

[3] *Georges Arnaud KEMAYO : Évaluation et validation des systèmes distribués avioniques, Thèse de doctorat, 23 septembre 2014.*

[4] *R.P.G. Collinson: Introduction to Avionics Systems, Navigation Systems, Springer Dordrecht Heidelberg London, New York 2011.*

[5] *The MEMS Revolution: Laura Castoldi, SEMI Networking Day, l'editeur life. Augmented, Milano, Italie, 20 September 2012.*

[6] *Cours théorique JAR/FCL PPL (A) : Module Navigation, Edition 4.2,10/06/2010.*

[7] *Pier-Marc COMTOIS-RIVET: Système de navigation INERTIEL par filtrage de KALMAN indirecte pour un sous-marin d'inspection robotisée, Mémoire présentée à L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE comme exigence partielle à l'obtention de LA MAÎTRISE EN GÉNIE ÉLECTRIQUE, MONTRÉAL CANADA, le 23 MARS 2012.*

[8] *Matrices Cours et Problèmes Seri Schaum: FRANK AYRES JR, Dickinson College, Group Mc Graw.Hill, Toronto, 1984.*

[9] *Savage, P. 1998. « Strapdown Inertial Navigation Integration Algorithm Design ». Journal of guidance,control, and dynamics January–February 1998, p. 19-27.*

[10] *Farrell, J. 2008. Aided navigation: GPS with high rate sensors. McGraw-Hill, 530.*

[11] *Addendum to NIMA TR 8350.2: Implementation of the world Geoditic System 1984(WGS 84), Reference Frame G1150.*

[12] *Titterton, David H., et John L. Weston. 2004. Strapdown Inertial Navigation Technology (2nd Edition). Institution of Engineering and Technology, p 558.*

[13] *R.P.G. Collinson: Inertial Navigation, p 313, Springer Dordrecht Heidelberg London, New York 2011.*

[14] *Contribution à la cartographie Mobile Développement et caractérisation d'un système basé sur un scanner Laser terrestre : Majd ALSHAWA, Thèse Docteur de l'Université de Strasbourg,2010.*

[15] *King.A.D: Inertial navigation forty years of Evolution,GEC review ISSN 02267-9337,1998.*

[16] *Arduino : Premiers pas en informatique embarquée : Simon Landrault (Eskimon) Hippolyte Weisslinger (olyte), Édition du 19 juin 2014.*

[17] *ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH*

DATASHEET

[18] *C. Tavernier, « Arduino applications avancées ». Version Dunod.*

[Invenses,] Invenses technologies, MPU-6000 and MPU-6050 Product Specification, 2013, [en ligne, Disponible sur : http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf , 13 juin 2014.

[19] *ERIK Bartmann . « Le grand liver d'Arduino » . 2eme Edition , date de parution*

02/01/2014.

[20] *Reas and Ben Fry. Processing: A Programming Handbook for Visual Designers and Artists (Second Edition) The MIT Press. publié December 2014.*

[21] *Processing.web : License GPLv2, 29/11/2017.*

[22] *Fusion de données inertielles pour l'estimation de l'attitude sous contrainte énergétique d'un corps rigide accéléré : THÈSE Pour obtenir le grade de DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES, 2006.*

[23] *S. Waldherr, R. Romero and S. Thrun:"A gesture based interface for human-robot interaction", In Autonomous Robots in Springer, vol. 9, Issue 2, pp. 151-173, 2000.*

[24] *Jay Esfandyari**:** received his Master's and PhD in EE from the U. of Technology, Vienna, and is MEMS product marketing manager at STMicroelectronics, 750 Canyon Dr., Coppell, TX 75019 USA; ph.: 972-971-4969;*

[25] *Solutions for MEMS sensor fusion By Jay Esfandyari, Roberto De Nuccio, Gang Xu, STMicroelectronics, Coppell, TX USA,2014*

[26] *Commande d'un robot mobile(Mobrob) sur deux roues : Mémoire en vue de lobtention du diplôme de Magister En Automatique, Université Aboubekr BELKAID TLEMCEN Faculté de Technologie, 2013.*

[27] *www.HowToMecatronics.com How I2C bus Works.*

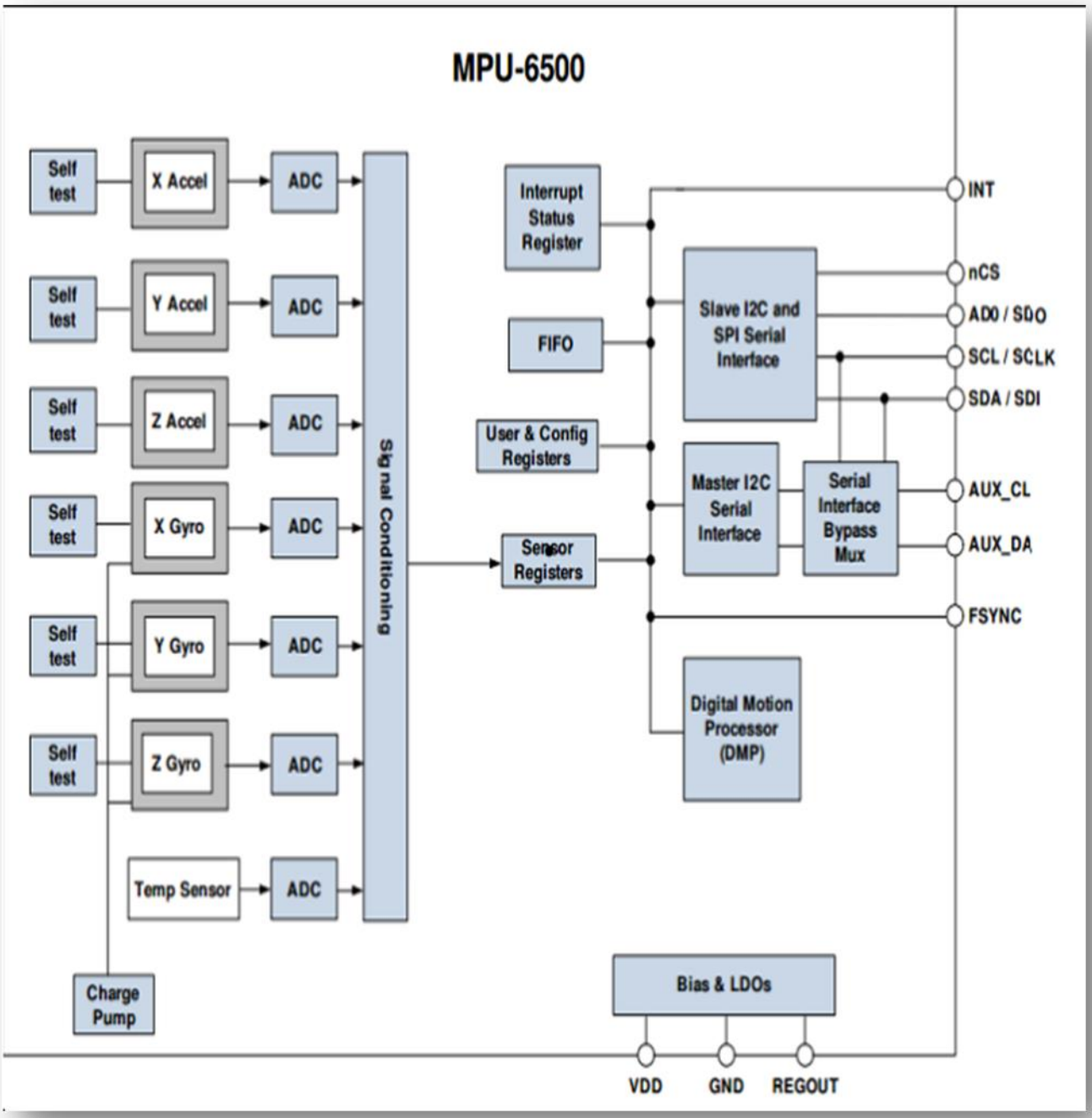
[28] *[I2CdevLib] How to decide Gyro and accelerometer offset, 2013, [en ligne], date de dernière mise à jour : 14decembre 2013, Disponible sur : http://www.i2cdevlib.com /forums/topic/91-how-to-decide-gyro-and-accelerometer-offsett/[calibration], 12 juin 2014.*

[29] *[Embeddeb Adventures, 2013] Embedded Adventures, WRL-3000 V2, 2013 ,[en ligne], date de dernière mise à jour :27juin 2013, Disponible sur : http://www.embeddedadventures.com/datasheets/WRL-3000_hw_v2_doc_v1.pdf ,13juin 2014.*

[30] *Librairies for Arduino, 2014, [en ligne], Disponible sur : http://playground.arduino.cc /Main/LibraryList#, 9 juin 2014.*

[31] *[Invense, 2013] Invenses technologies, MPU-6000 and MPU-6050 Product Specification, 2013, [en ligne], Disponible sur : http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf, 13 juin 2014.*

ANNEXA : L'architecture interne de la puce MPU6050



ANNEX B : Diagramme des Pins Arduino UNO