Projet de Physique P6-3
STPI/P6-3/2011 – n°21

CAO ET REALISATION ELECTRONIQUE POUR PROJET BATEAU EN AVIRON DE COMPETITION

Étudiants :

Nicolas BEDOUET
Kim Quyen NGUYEN
Luc PETIT
Thibault ROSSELIN
Valentin SAUVAGE
Michel TOTAIN LYCZKO

Enseignant responsable du projet :
M. Fabrice DELAMARE

Date de remise du rapport : ***14*/06/11**

Référence du projet : ***STPI/P6-3/2011 – n°21***

Intitulé du projet : ***CAO et réalisation électronique pour projet bateau en aviron de compétition***

Type de projet : ***expérimental***

Objectifs du projet *(10 lignes maxi)* :

Le but de ce projet est de mettre en œuvre la partie électronique d'un système de stabilisation d'un aviron, c'est à dire un système qui, en fonction de l'inclinaison de l'embarcation entrainera une correction afin de maintenir cette dernière à flot. Ce système devra donc :
-déterminer l'inclinaison de l'ensemble à tout instant (nécessité d'avoir un capteur).
-appliquer rapidement une correction en fonction du signal émis par le capteur.

Étant donné les dimensions d'un aviron (pas plus d'une quarantaine de centimètres de largeur) notre projet comprendra aussi des contraintes dimensionnelles et, puisque les avantages d'un aviron sont la vitesse et la maniabilité, il devra également être le plus léger et le moins encombrant possible.

Mots-clefs du projet *(4 maxi)* : ***programmation, conception , organisation, mise en œuvre***

Notations et acronymes

PID : Proportionnel Intégrale Dérivée, système de régulation en boucle fermée d'un système électronique.

Bus : Un bus informatique est un système de communication entre les composants d'un ordinateur.

Table des matières

1 Introduction

Ce projet, se déroulant au cours de notre quatrième semestre de nos études d'ingénieur à l'INSA, est pour nous l'occasion de mettre en application nos quelques connaissances acquises lors des semestres précédents en programmation informatique mais surtout de les approfondir, n'étant pas des informaticiens nés.

En effet son principal objectif est la programmation d'un système de stabilisation d'un aviron mettant en jeu différents composants électroniques que nous allons avoir à découvrir et à maîtriser.

C'est également l'opportunité de travailler, une nouvelle fois, en équipe, dans un groupe cette fois constitué d'élèves aux parcours différents, ce qui permet de nous rapprocher au plus près de notre futur métier d'ingénieur où nous aurons à œuvrer au côtés de personnes issues de tous horizons.

Un tel système comprenant également une partie mécanique conçue par un autre groupe, partie que nous devons piloter, nous avons donc également à travailler en coopération avec un autre groupe sous la tutelle d'un autre enseignant, M. Didier VUILLAMY, puisque notre projet ne prend tout son sens que mis en commun avec le leur, la dimension organisation de ce projet n'en est donc que plus grande.

2 Méthodologie / Organisation du travail

2.1 Remarques préliminaires :

Au cours de ce projet nous avons du faire face à plusieurs problèmes que nous avons tenté, au mieux, de surmonter.

Ce projet n'étant en réalité que la partie conception électronique d'un projet de plus grande ampleur nous avons eu à composer avec les membres d'un second groupe, chargés de la partie mécanique. Nous fûmes donc, tout du long du semestre, tributaires des travaux de ce groupe, ayant à plusieurs reprises à nous enquérir de ses avancées auprès de l'enseignant responsable. En effet pour nous lancer sur une programmation précise nous dûmes attendre que le premier projet mécanique soit mis en place ne pouvant alors que nous familiariser avec les composants, puis, lorsque nous voulûmes commencer nos tests sur l'objet final, il nous fallut attendre que la partie mécanique soit disponible.

D'autres part nous dûmes également faire face au problème de la différence de langue, en effet pour notre camarade Vietnamienne la compréhension de la documentation dans une langue, pour elle, étrangère puis la retranscription en français ne fut pas chose aisée, il en a été de même quant à la compréhension du travail réalisé par les autres.

Enfin, ayant mal interprétés l'intitulé même du sujet, l'ensemble des membres du groupe s'attendant plus à un projet de CAO et à quelques assemblages électroniques de niveau 3e technologique, nous devons reconnaître que l'aspect programmation de ce projet nous effraya quelques peu étant donné qu'aucun d'entre nous ne vouait à cet discipline une quelconque passion. Mais nous fîmes en sorte de surmonter ces obstacles, trouvant néanmoins que la découverte, à travers ce projet, du mode de fonctionnement de nombre de systèmes électroniques nous intéressait fortement.

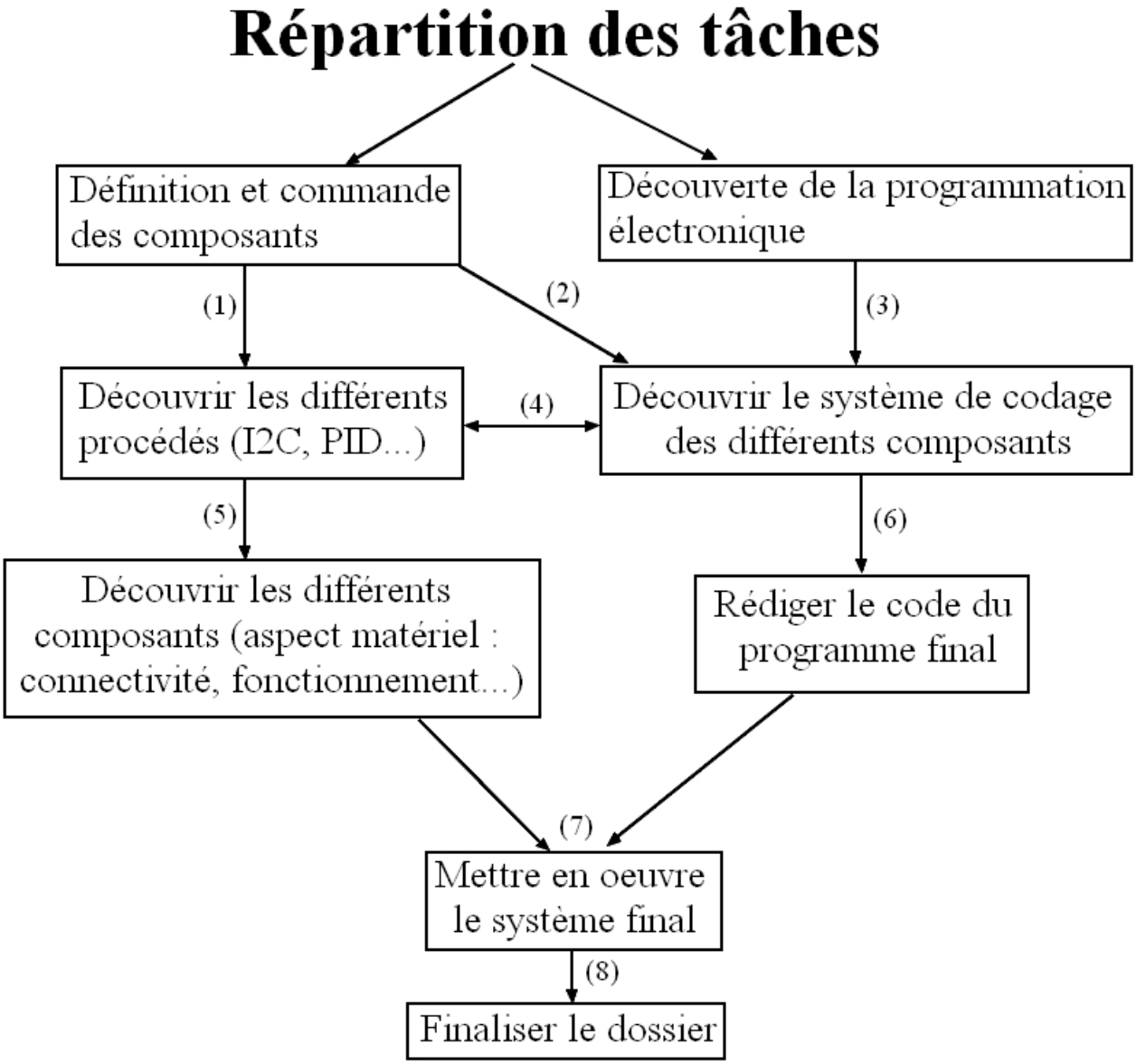
2.2 Organisation :

Dès le début de ce projet nous avons fait en sorte, même si les tâches à effectuer ne nous paraissaient pas toujours évidentes, de nous répartir les tâches au maximum. Ceci, conjugué au fait qu'il fallait commander les pièces au plus tôt afin de les recevoir rapidement et de pouvoir aussitôt les mettre en œuvre, fit que nous décidâmes tout d'abord de créer deux premiers groupes autour de deux tâches principales :

-Sous-groupe 1 : la définition des pièces nécessaires au projet : N. BEDOUET et M. TOTAIN LYCZKO puis K.Q. NGUYEN lorsqu'elle arrivera.

-Sous-groupe 2 : la découverte de la programmation des composants électroniques : L. PETIT, T. ROSSELIN, V. SAUVAGE

Pour mieux illustrer ce que nous comptions faire nous commenterons le schéma ci-après (se reporter aux commentaires suivant dont la numérotation correspond à celle indiquée le long des flèches) :



Étapes :

Note 1 : afin de rédiger la bibliographie dans les meilleures conditions chaque membre du groupe devait, à chacune de ses recherches, noter les liens utiles et leurs dates de consultation.

1 : Une fois la commande des pièces faite les membres du groupe les ayant déjà découvertes feront une brève recherche sur leurs méthodes de contrôle

2 : Avant (1) ceux étant chargés de la commande devront bien évidemment donner les références des composants afin que ceux en charge de la programmation étudiants les spécificités de code de ces composants.

3 : Une fois la méthode générale de programmation en électronique découverte, ceux en charge de cette partie devront s'atteler à la découverte de la programmation spécifique aux composants (différentes possibilités d'action...)

4 : Pour que le groupe tout entier bénéficie des avancés de chacun il faudra que chaque sous groupe échange ses informations de façon à ce qu'aucun aspect du problème ne nous échappe, afin que nous n'avancions pas avec des oeillières dans ce projet.

5 : Une fois que le premier sous-groupe se sera brièvement informé du mode de commande des composants il sera à même de comprendre leur mode complet de fonctionnement et, pour les plus importants, de rédiger une fiche sur ceci.

6 : Après avoir découverts différents exemples de programmation et de fonctionnement des composants le deuxième sous-groupe devra commencer à élaborer le programme qui sera à même de répondre à nos attentes.

7 : Une fois les composants connectés entre-eux par le premier sous-groupe et le programme élaboré par le second il faudra mettre en commun nos acquis en les testant sur le système final que nous aura fourni le groupe responsable de la partie conception mécanique.

8 : Lorsque tout ceci aura été effectué il ne nous restera plus qu'à regrouper nos recherches dans le dossier final, et à rédiger les quelques parties qui nécessitaient que le projet soit achevé.

Remarques :

-naturellement au sein des sous-groupes chacun prendra en charge un élément pour ne pas user nos forces, ce qui n'exclut pas une collaboration lorsque des problèmes de compréhension se font sentir.

-il nous a semblé plus cohérent qu'un seul élève s'occupe du carnet de bord afin que la forme de celui-ci reste la même et que l'élève puisse suivre l'avancement du projet et mieux en reporter à ses camarades.

-au fil du temps nous avons pu observer une évolution de la composition des sous-groupes (se reporter au carnet de bord), en effet la tâche étant parfois ardue et ne nécessitant pas forcément d'avoir effectué les précédentes nous nous sommes à quelques reprises échangés les rôles et ce pour deux raisons : la première étant que lorsque l'un d'entre nous peine à comprendre certaines explications il vaut mieux qu'un autre, dont le mode de compréhension est immanquablement différent, le remplace, la seconde étant simplement qu'il ne fallait pas non plus peiner sur une tâche jusqu'à se décourager alors qu'une autre solution était possible et ne représentait qu'une perte de temps minime...

3 Travail réalisé et résultats

3.1 État de l'art

Afin d'avoir un aperçu de l'utilité et de l'avancée des recherches sur les systèmes de stabilisation électronique et les différents composants qui sont susceptibles de les composer,nous avons eu à faire, avant d'entrer dans le vif du sujet, des recherches sur l'utilisation actuelle de tels systèmes.

3.1.1 Nouveautés au niveau des batteries

A l'heure actuelle, la recherche sur les batteries est surtout dictée par une demande croissante des constructeurs automobiles pour l'autonomie de leur véhicule de type électrique ou hybride. Mais les dernières avancées dans ce domaine ne profitent pas seulement aux constructeurs automobiles, en effet, ces technologies peuvent être utilisées sur de plus petits produits, plus enclin à nous servir dans ce projet.

Ainsi, on note certaines avancées technologiques dans ce domaine ; le japonais Panasonic va commercialiser courant 2012 et 2013, deux nouveaux types de batteries lithium-ion. Elles seront dotées d'une cathode en nickel renforcé, et d'une anode en graphite pour la première (sortie prévue en 2012), en silicium pour la seconde. Portant leur densité d'énergie à 730 Wh/l pour la première, et à 800 Wh/l pour la seconde. En comparaison, une batterie lithium-ion actuelle possède une densité énergétique de seulement 620 Wh/l. Malheureusement, le poids de ces batteries est plus important que celui des batteries déjà existantes, ainsi, la deuxième batterie qui sera plus performante accuse une augmentation de 22 % au niveau de sa masse, mais un gain de près de 30 % en densité d'énergie par rapport à une batterie actuelle !

Avec de telles caractéristiques, la firme américaine et constructeur automobile Tesla a avoué être intéressé pour installer ce type de batteries sur ses futurs modèles.

Les chercheurs de nombreux pays se penchent sur ce problème de batteries apportant différentes solutions. Des américains du MIT ont accentué leurs recherches surtout sur le problème du temps de recharge de la batterie, ils ont ainsi réussi à inventer une batterie lithium-fer-phosphate ($LiFePO_4$), qui se recharge en seulement 6h (pour une recharge totale) au lieu de 8h actuellement.

D'autres chercheurs comme ceux de l'Imperial College de Londres, ont réussi à créer un nouveau matériau composite, susceptible de stocker la charge électrique, et suffisamment solide et léger pour servir de matériau pour faire office de carrosserie. Imaginez un téléphone de l'épaisseur d'une carte de crédit, un GPS qui fonctionnerait grâce à son boitier !

3.1.2 Les capteurs inertiels

Deux types de capteurs inertiels sont généralement utilisés : les accéléromètres et les gyroscopes. Les accéléromètres permettent de mesurer les

accélérations linéaires de l'objet auquel ils sont reliés selon la loi fondamentale de la dynamique. La position est estimée par une double intégration des mesures fournies. Ce type de capteur est rapide, cependant le processus d'intégration tend à accumuler les erreurs au cours du temps ce qui produit une dérive (*drift*) entre la position estimée et la position réelle. Les gyroscopes mesurent les vitesses angulaires, et fournissent l'orientation.

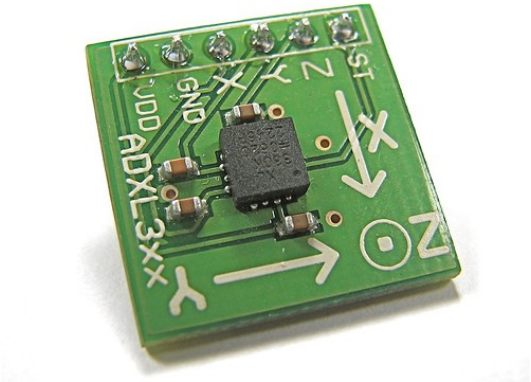


Illustration 1: Exemple d'accéléromètre trois axes moderne

Les capteurs inertiels mécaniques traditionnels demeurant encombrants. L'avènement des nanotechnologies a permis de miniaturiser ces dispositifs appelés MEMS (Micro-Electro-Mechanical Systems). La méthode consiste à intégrer des éléments mécaniques, capteurs, actionneurs et leur électronique sur un substrat commun en silicium par le biais des technologies de micro-fabrication. L'électronique est fabriquée en utilisant les méthodes classiques de production des circuits intégrés tandis que les éléments micro-mécaniques sont obtenus par le biais de processus compatibles qui vont découper et retirer des parties du silicium pour constituer les micro-mécanismes. Dans le cas des accéléromètres, ceci a permis de réduire les coûts de fabrication par 10, tout en miniaturisant et en augmentant la fiabilité de ces derniers.

Aujourd'hui, ces capteurs sont très répandus dans de nombreux objets de la vie courante. On les retrouve dans bon nombre de téléphones portables, consoles de jeux, ainsi que dans l'automobile (déclenchement de l'airbag par exemple).

3.1.3 Le cas du porte-avions Charles-De-Gaulle :

Illustration 2: Le Charles De Gaulle en mer

Ce porte-avions est équipé d'un système de stabilisation SATRAP: (Système Automatique de TRAnquillisation et de Pilotage). Ce système permet de réduire les mouvements non désirés du navire : le roulis, le lacet et l'embardée. Il coordonne, par l'intermédiaire d'un calculateur centralisé, l'action des différents organes : deux paires d'ailerons stabilisateurs, le gouvernail et un ensemble de masses mobiles pour COmpenser la GITE (Cogite). Ce système qui nous intéresse particulièrement est composé de douze lignes de wagonnets métalliques (représentant une masse totale de plus de 240 tonnes) se déplaçant dans des tunnels perpendiculaires à l'axe de la coque, le système COGITE permet au porte-avions de virer à plat et de mettre en œuvre son groupe aérien embarqué avec des conditions de navigation dégradées. Ce système de stabilisation permet au *Charles-de-Gaulle* de mettre en œuvre des avions de 20 tonnes par mer de force 5 et 6. Par comparaison, le Clemenceau et le Foch avaient été étudiés pour l'emploi d'appareils de 12 tonnes par mer force 3 à 4.

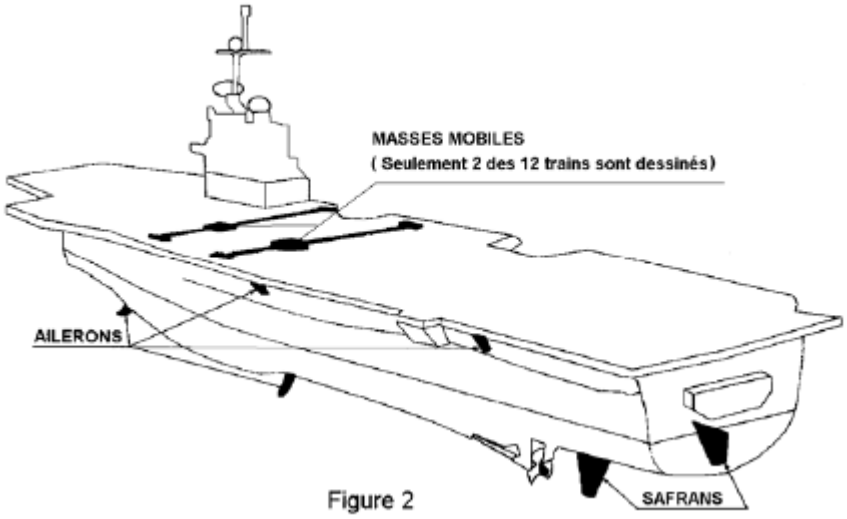


Illustration 3: Description du système de stabilisation du porte-avions

Les trains de masses ont une masse de vingt deux tonnes chacun !

Il est également à noter que la première utilisation de masses pour stabiliser un navire remonte à l'invention de la quille lestée sur les bateaux. En effet, il s'agissait d'abaisser le centre de gravité du navire pour une plus grande stabilité.

3.1.4 La technologie FLEXX Tronic

La technologie FLEXX Tronic est fondée sur la plate-forme mécatronique conçue par l'équipe Ingénierie, Recherche et développement de Bombardier. La technologie FLEXX Tronic offre des capacités uniques de direction radiale active (ARS) et de stabilisation active du bogie.

Les principaux avantages de la technologie FLEXX Tronic sont les suivants : compatibilité totale inter-réseaux diminution de l'usure des roues et des rails allongement des intervalles de maintenance diminution de la masse du véhicule, des vibrations et du bruit réduction de l'incidence sur la voie et frais d'accès moins élevés sur les réseaux visés par des frais liés à la performance.

Illustration 4: Un TGV utilisant la technologie FLEXX Tronic

Fiche technique

Le dernier développement de cette plateforme est le système FLEXX Tronic WAKO. Ce système innovant – intégré dans la suspension secondaire existante - assure la compensation du mouvement naturel de roulis des caisses.

Les bénéfices majeurs du système FLEXX Tronic WAKO sont : Augmentation de la vitesse : ce système permet d'accroître d'environ 15 % la vitesse dans les courbes et ainsi de réduire la durée des déplacements tout en minimisant les investissements en infrastructure. Grande fiabilité : un niveau de fiabilité très important est atteint grâce à l'utilisation des dernières technologies en matière d'électronique et d'un système 100% redondant, incluant les vérins. En cas de panne d'un des composants, il n'y aura donc pas de conséquences sur le temps de trajet ou les horaires. Complexité mécanique réduite: la réduction de la complexité mécanique est atteinte par l'organisation multifonction du système mécatronique, qui permet un design compact avec un empattement minimum, et ce même avec un système de traction intégré. La cinématique du système permet un comportement sans-faille et permet d'avoir un design de voiture large avec un confort de première classe dans un train à double-étage. Grand niveau de confort : grâce à la compensation du mouvement de roulis et le contrôle actif du confort base sur des principes de conception avancés, un très haut niveau de confort est atteint, les oscillations sont limitées et comparativement à d'autres systèmes pendulaires, les voyageurs ne souffrent pas du mal des transports.

3.2 Les différents composants et systèmes de commande

Pendant et après avoir effectué les commandes des pièces qui serviront à la conception de notre système nous avons eu à nous documenter sur celles-ci et sur leur mode de fonctionnement.

3.2.1 Le mode de communication I2C

Un bus **I2C** (pour *Inter Integrated Circuit*) est un bus série et synchrone composé de trois fils :
un signal de donnée (SDA) ;
un signal d'horloge (SCL) ;
un signal de référence (masse).
Exemple de communication I2C:

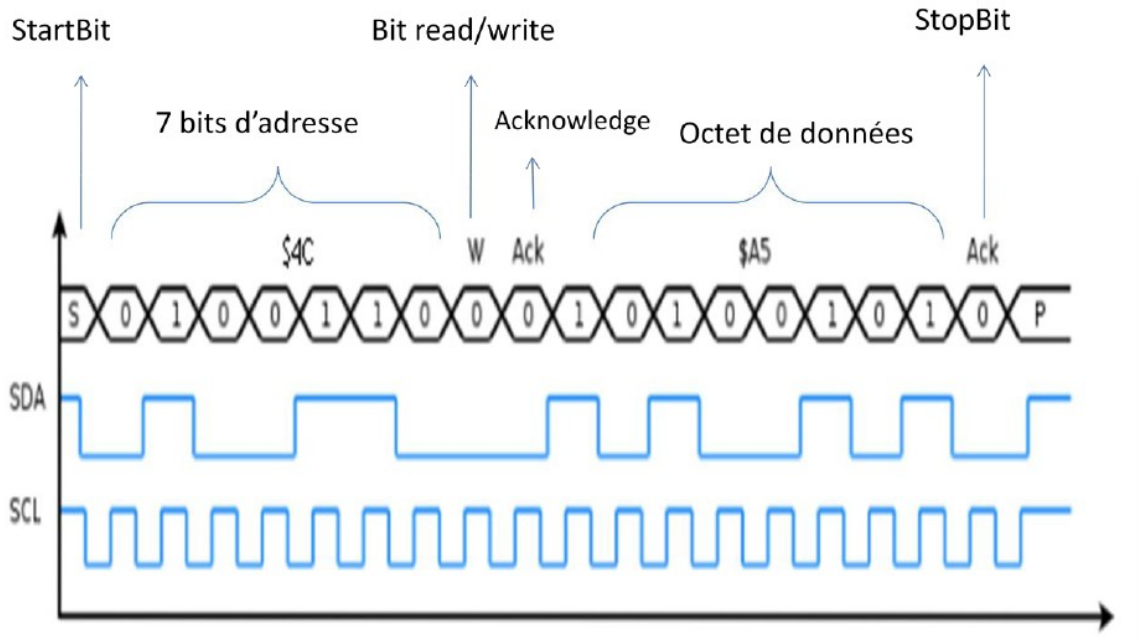


Illustration 5: Exemple de communication en I2C

La communication commence par le **StartBit** : la communication SDA passe de 1 à 0. Puis l'**adresse**, (sur 7bits) elle indique avec quel composant la carte Arduino va communiquer. Elle est suivie d'un **bit de read/write**, 0 correspondant à l'écriture et 1 à la lecture. Enfin l'**acknowledge** est une sorte d'accusé de réception renvoyé par le composant récepteur : 0 si le message est passé, 1 sinon. Vient ensuite un **octet de données** où l'action à effectuer est spécifiée de nouveau l'**acknowledge** et enfin le **StopBit** qui indique la fin séquence (la communication SDA passe de 0 à 1).

3.2.2 Le PID

Un contrôleur PID fournit une correction à partir du signal d'erreur soit la différence entre ce que l'on veut et ce que l'on mesure.

La correction résulte de la somme de 3 termes :
- Proportionnel (facteur P ou K) : plus l'erreur est grande et plus la correction est grande. K règle le facteur multiplicatif sur l'erreur.
- Intégrale (facteur I ou Ti) : la correction augmente si l'erreur reste constante. Ti règle la vitesse à laquelle est intégrée l'erreur. Plus Ti est petit, et plus on fait croître rapidement le signal de correction.
- Dérivée (facteur D ou Td et N) : la correction est temporairement importante à chaque "saut" du signal d'erreur. Td règle le temps d'application de ce petit "plus" sur la correction globale et N sa hauteur. Ce paramètre doit évidemment être utilisé avec précaution. En effet, il peut générer des sollicitations excessives de l'actionneur pouvant entraîner une fatigue prématurée : tout bruit hautes fréquences sur la mesure se traduit par des variations du même ordre sur la commande.

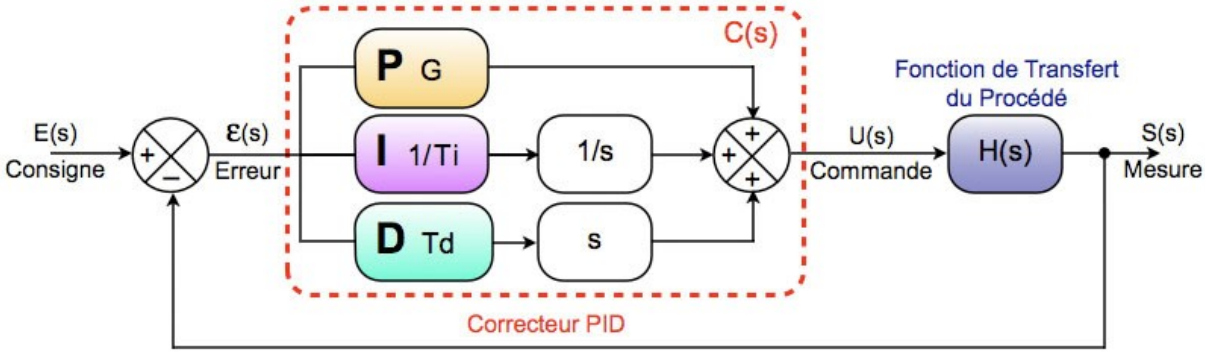


Illustration 6: Schéma du mode de fonctionnement du système de régulation PID

Note 1 : ce système étant en réalité conçu pour réaliser des corrections de haute précision une simple correction proportionnelle devrait suffire dans notre cas pour peux que nous ayons le système mécanique à disposition pour faire nos tests.

3.2.3 Les interruptions

Au cours de notre boucle (loop) il faudra faire appel (certainement toutes les 0,1s) à un petit programme qui s'enquerra de l'état de notre capteur gyroscopique (inclinaison et accélération) et devra entraîner une réaction en conséquence.

Comme le programme principal sera stoppé en cours de fonctionnement son contexte (état de fonctionnement) sera stocké dans la mémoire vive (RAM) pour être retransmis ensuite à la fin de l'exécution du petit programme.

Pour permettre un tel arrêt il faut faire appel à l'outil des interruptions.

Note 1 : Finalement la taille et les fonctions du programme s'avérant être réduites nous n'avons pas eu à faire appel à ce procédé.

3.2.4 La carte Arduino Duemilanove

Se reporter aux annexes, partie « description des composants » pour plus de détail.

La carte Arduino Duemilanove("2009") est une carte à microcontrôleur basée sur l'ATmega168 pour les premières version ou sur l'ATmega328 pour les versions actuelles.

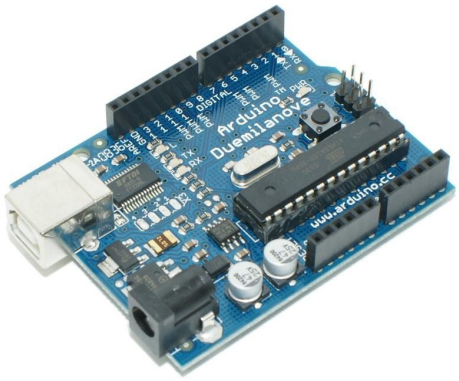


Illustration 7: La carte Arduino Duemilanove

Elle dispose :

1. de 14 broches numériques d'entrées/sorties (dont 6 peuvent être utilisées en sorties PWM (largeur d'impulsion modulée)),
2. de 6 entrées analogiques (qui peuvent également être utilisées en broches entrées/sorties numériques),
3. d'un quartz 16Mhz,
4. d'une connexion USB,
5. d'un connecteur d'alimentation jack,
6. d'un connecteur ICSP (programmation "in-circuit"),
7. et d'un bouton de réinitialisation (reset).

Elle contient tout ce qui est nécessaire pour le fonctionnement du microcontrôleur; Pour pouvoir l'utiliser, il suffit simplement de la connecter à un ordinateur à l'aide d'un câble USB (ou de l'alimenter avec un adaptateur secteur ou une pile, mais ceci n'est pas indispensable, l'alimentation étant fournie par le port USB).

Alimentation

La carte Arduino Duemilanove peut-être alimentée soit via la connexion USB (qui fournit 5V jusqu'à 500mA) ou à l'aide d'une alimentation externe. La source d'alimentation est sélectionnée automatiquement par la carte.

L'alimentation externe (non-USB) peut être soit un adapteur secteur (pouvant fournir typiquement de 3V à 12V sous 500mA) ou des piles (ou des accus). L'adaptateur secteur peut être connecté en branchant une prise 2.1mm positif au centre dans le connecteur jack de la carte. Les fils en provenance d'un bloc de piles ou d'accus peuvent être insérés dans les connecteurs des broches de la carte appelées Gnd (masse ou 0V) et Vin (Tension positive en entrée) du connecteur

d'alimentation.

La carte peut fonctionner avec une alimentation externe de 6 à 20 volts. Cependant, si la carte est alimentée avec moins de 7V, la broche 5V pourrait fournir moins de 5V et la carte pourrait être instable. Si on utilise plus de 12V, le régulateur de tension de la carte pourrait chauffer et endommager la carte. Aussi, la plage idéale recommandée pour alimenter la carte Duemilanove est entre 7V et 12V.

Mémoire

L'ATmega 168 a 16 Ko de mémoire FLASH pour stocker le programme (dont 2ko sont utilisés par le bootloader); l'ATmega 328 a 32Ko de mémoire FLASH pour stocker le programme (dont 2Ko également utilisés par le bootloader). L'ATmega 168 a 1Ko de mémoire SRAM (volatile) et 512Ko d'EEPROM

Communication

La carte Arduino dispose de toute une série d'outils pour communiquer avec un ordinateur, une autre carte Arduino, ou avec d'autres microcontrôleurs.
Le logiciel Arduino inclut une fenêtre terminal série (ou moniteur série) sur l'ordinateur et qui permet d'envoyer des textes simples depuis et vers la carte Arduino. Les LEDs RX et TX sur la carte clignote lorsque les données sont transmises via le circuit intégré FTDI et la connexion USB vers l'ordinateur

Caractéristiques Mécaniques

Les longueurs et largeurs maximales de la Duemilanove sont respectivement 6.86 cm et 5.33 cm, avec le connecteur USB et le connecteur d'alimentation Jack s'étendant au-delà des dimensions de la carte. Trois trous de vis permettent à la carte d'être fixée sur une surface ou dans un boîtier.

3.2.5 Les cartes moteur MD03 et MD22

Se reporter aux annexes, partie « description des composants » pour plus de détail.

Note 1 : Nous nous intéressons également dans ce paragraphe à la carte MD22, même si elle ne servira pas dans notre projet final, puisqu'elle a servi à effectuer une série de tests préalables et que son fonctionnement est très proche de celui de la MD03.

Note 2 : Nous ne nous intéresserons qu'au mode de fonctionnement I2C de la carte. En effet elle peut également être contrôlée en Analog, c'est à dire par de simples variations de tension mais comme nous allons le voir l'I2C offre plus de possibilités dans notre cas.

Ces deux cartes fonctionnent sur le même principe de registres, c'est à dire que, lorsque l'on codera un programme où l'on voudra envoyer ou recevoir une information on précisera la nature de l'information (vitesse, température, accélération...) en donnant le numéro du registre concerné (voir tableaux ci-après). Par exemple pour commander une action relevant du registre température de la carte MD03 on écrira dans le code : i2c_write(4);

Spécificités de la carte moteur MD22 :

La différence majeure entre la carte MD22 et la MD03 est que la MD22 permet de contrôler deux moteurs en même temps (d'où les deux ports moteur supplémentaires). Naturellement comme notre système n'en nécessite qu'un, nous n'avons branché qu'un seul moteur sur cette dernière.

Illustration 8: Photographie d'une carte moteur MD22.

Les registres utilisables de la carte MD22 :

Register Adress	Name	Read/Write	Description
0	Mode	R/W	Mode of operation
1	Speed	R/W	Left motor speed (mode 0,1) or speed (mode 2,3)
2	Speed2/Turn	R/W	Right motor speed (mode 0,1) or turn (mode 2,3)
3	Acceleration	R/W	Acceleration for I2C (mode 0,1)
4	Unused	Read Only	Read as zero
5	Unused	Read Only	Read as zero
6	Unused	Read Only	Read as zero
7	Software Revision	Read Only	Software Revision Number

La bibliothèque d'adresse pour carte MD22 :

Mode	Switch 1	Switch 2	Switch 3	Switch 4
I2C Bus - address 0xB0	On	On	On	On
I2C Bus - address 0xB2	Off	On	On	On
I2C Bus - address 0xB4	On	Off	On	On
I2C Bus - address 0xB6	Off	Off	On	On
I2C Bus - address 0xB8	On	On	Off	On
I2C Bus - address 0xBA	Off	On	Off	On
I2C Bus - address 0xBC	On	Off	Off	On
I2C Bus - address 0xBE	Off	Off	Off	On

0v - 2.5v - 5v Analog	On	On	On	Off
0v - 2.5v - 5v Analog + Turn	Off	On	On	Off
RC Servo	On	Off	On	Off
RC Servo + Turn	Off	Off	On	Off

En fonction que l'on placera manuellement les 4 fiches sur On ou Off on utilisera les

Spécificités de la carte moteur MD03 :

Les registres utilisables de la carte MD03 :

Register Address	Name	Read/Write	Description
0	Command	R/W	Write 01 Forwards - 02 Reverse (00 for instant stop - Rev9 firmware only)
1	Status	Read only	Acceleration, Temperature and Current Status
2	Speed	R/W	Motor Speed 0-255 (0×00 - 0xFF)
3	Acceleration	R/W	Motor Acceleration 0-255 (0×00 - 0xFF)
4	Temperature	Read only	Module temperature
5	Motor Current	Read only	Motor Current
6	Unused	Read only	Read as zero
7	Software Revision	Read only	Software Revision Number- Currently

Illustration 9: Photographie d'une carte moteur MD03. http://www.electronika.fr/blog/?p=105

Le schéma ci-dessous représente les liaisons électroniques entre le contrôleur (ici la carte Arduino), la carte MD03 et le moteur :

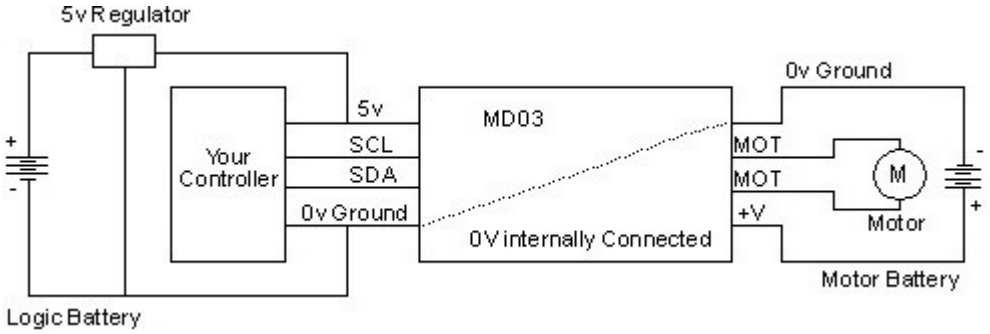


Illustration 10: Branchements de la carte moteur MD03 avec la carte arduino et le moteur. http://www.lextronic.fr/P1917-module-de-commande-de-moteur-md22.html

3.2.6 Le moteur MY1018

Note 1 : Le moteur présenté en premier est un modèle MY1016 alors que celui que nous utiliserons dans notre système de stabilisation est un MY1018, mais sa forme extérieure est la même et ses caractéristiques sont très proche.

Dessin du moteur avec dimension :

Le poids du moteur est de 1.95 kg, il est équipé d'origine d'un pignon de 11 dents, ayant une largeur de 3mm. La fixation se fait via un socle en fer percé de 4 trous filetés. Les fils d'alimentation sont équipés chacun d'un cosse plate mâle.

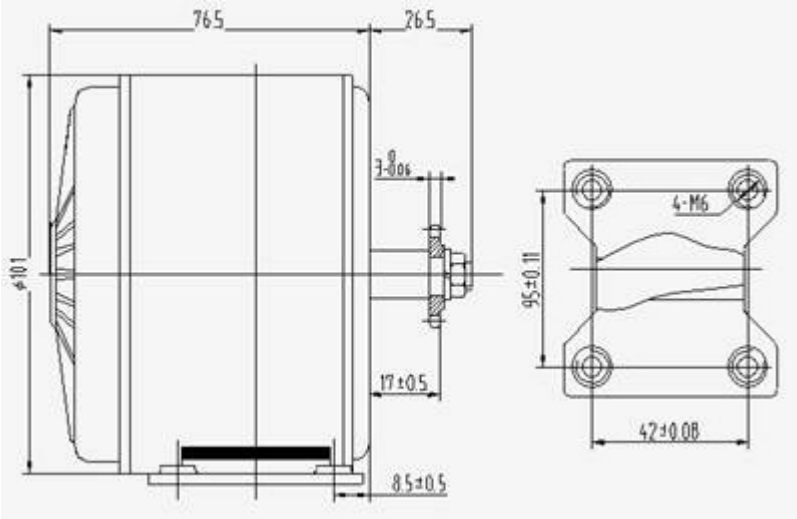


Illustration 11: Schéma du moteur avec ses dimensions

Le moteur MY1018 (MY1018 250W) présenté ci-dessous est équipé d'un réducteur (qu'il n'est pas possible de supprimer) et d'un pignon au pas de 12.7 mm, 3.3 mm d'épaisseur (pas compatible avec des chaînes de vélo classique ce dont il faudra tenir compte lors de l'assemblage mécanique de notre système de stabilisation).

Illustration 12: Photo d'un moteur MY1018

Informations générales :

Le poids du moteur est de 2.2 kg, il est équipé d'origine d'un pignon de 9 dents pas 1/2", ayant une largeur de 3.3 mm. La fixation se fait via un support qui se fixe sur le corps du moteur par trois points, elle se vend dans le commerce, mais il est plus économique de le fabriquer soit même (assez simple). Le réducteur permet de réduire la vitesse nominale du moteur qui est de 2750 tr/mn par 7.2 (7.2:1) Les fils d'alimentation sont équipés chacun de cosse plate.

Caractéristiques électriques et mécaniques principales :

	Couple N.m	**Vitesse de rotation tr/min**	**Puissance fournie W**	**Tension V**	**Courant A**	**Puissance consommée W**	**Rendement %**
A vide	0.03	3150	10	25	2	50	20
Rendement maxi	-	-	-	-	-	-	-
Régime nominal	0.87	2750	250	24	12.9	311	80

Remarques et interprétations du tableau ci dessus :

Les données commerciales inscrites sur le moteur sont les suivantes : 24V, 250W et 2750 tr/min, ratio 7.2:1. Le moteur tourne à 2750 tr/mn en sortie d'arbre moteur, ce qui faire 2750/7.2 = 382 tr/min en sortie de réducteur. Le couple au régime nominal est de 0.87 N.m, il peut atteindre les 1.1 N.m au maximum, si les batteries alimentant le moteur permettent de maintenir 24V à ses bornes avec un courant supérieur à 16A.
Le couple nominal en sortie de réducteur est de 6.3 N.m, et peut atteindre 7.9 N.m au maximum. Le rendement est très correct pour ce moteur relativement bon marché, il varie entre 78% et 82%, selon le courant I consommé (donc la charge). En réalité, le courant moyen consommé sera inférieur à 10A pour la plupart des applications, ce qui permet une autonomie moyenne de plus de 30min avant d'atteindre le seuil de recharge de batteries 7Ah.

Puissance et exemples de calcul:

Le couple disponible en sortie de réducteur sera supérieur à 6.3 N.m, lorsque le moteur fournira 250W. Pour se représenter cette valeur, nous pouvons comparer cette puissance fournie avec celle d'un cycliste moyen sur son vélo. Pour une vitesse moyenne de 28 km/h sur un terrain plat (voir détail sur ce site : http://derrieny.club.fr/Puissance.html), le cycliste fournie une puissance de 183W. On peut en déduire que ce moteur n'aurait aucun mal à propulser le vélo d'un adulte de 75 kg, ce qui a déjà été réalisé par certains fabricants chinois. Concernant le couple, à titre de comparaison, le couple maximum d'un petit scooter de 50cc chinois que l'on trouve partout en grande surface est d'environ de 2.5 N.m pour une puissance fournie de 1180 W. Ce type de moteur thermique permet de déplacer le scooter avec son passager à une vitesse de 50 km/h, avec un poids total de 160 kg. Il faut relativiser la puissance, qui est bien souvent celle disponible au vilebrequin, comme sur les motos, et elle sera donc plus faible en sortie de transmission. (10% environ selon les modèle) On imagine donc facilement que notre moteur MY1018 pourra entrainer aisément notre système de stabilisation. La vitesse maximale théorique dépendra du rapport de transmission. En résumé, le MY1018 avec réducteur permettra de privilégier le couple, alors que le MY1016 (ou MY1018, MS1020) sans réducteur privilégie la vitesse maximale.

3.2.7 Le capteur gyroscopique IMU3000

Se reporter aux annexes, partie « description des composants » pour plus de détail.

Notre mission lors de ce projet est de stabiliser un aviron de compétition, pour ce faire nous pouvions faire appel à de nombreux éléments. Par exemple, une carte ARDUINO, un moteur MY1018, mais il nous manquait cependant l'élément qui nous permet de résoudre le problème posé par le projet.

Cet élément primordial de notre montage électronique est monté sur le shield qui recouvre la carte ARDUINO. L'accéléromètre Sparkfun IMU 3000 combo est ce qui va nous permettre de retrouver le bon angle pour éviter que l'aviron ne puisse chavirer. Ce modèle de chez Sparkfun est dit combo car il est, en fait, composé de deux différents éléments, un accéléromètre l'ADXL 345 et un gyroscope IMU 3000. C'est cette combinaison, celle d'un accéléromètre et d'un gyroscope qui va conditionner la réponse du moteur et qui va ainsi rétablir l'équilibre du système total.

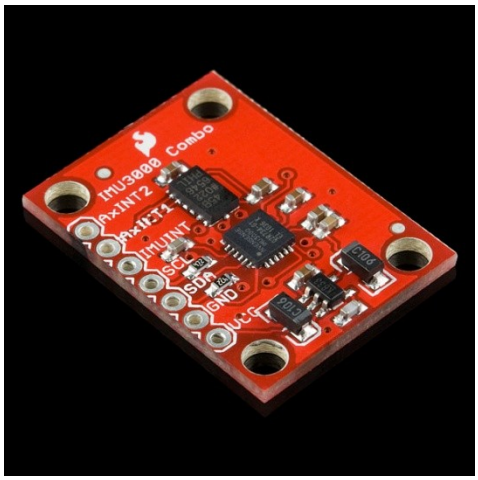


Illustration 13: Le capteur Gyroscopique IMU300

La communication avec cet élément via la carte ARDUINO se fait par I2C, via les entrées visibles sur la photo ci-dessus. Nous avons utilisé les entrées VCC, SCL, GND et SDA, qui seront reliées via le shield aux entrées de la carte ARDUINO suivantes : 3.3 V, analog 4, analog 5 et la masse GND.

L'accéléromètre ADXL 345, est un accéléromètre de haute résolution (13 bits), pouvant mesurer jusqu'à une accélération de plus ou moins 16 G selon les trois axes. Un élément important de ce composant est sa résistance aux éléments climatiques car si sa fragilité à l'eau n'intervient que très peu du fait de son placement dans un boitier étanche, il faut qu'il résiste à un grand panel de température du fait de son utilisation pour l'entrainement en aviron de compétition. Celui-ci résiste à des températures allant de -40 °C à 105 °C. Il est également à noter que ce type de composant est utilisé dans les téléphones portables, pour les jeux et est notamment capable de détecter toute sorte de chute.

Le gyroscope IMU3000 se compose comme son nom l'indique d'un gyroscope trois axes et également d'un Digital Motion Processor, qui, couplé à un accéléromètre en fait un excellent capteur 6 axes. De la même manière que l'accéléromètre, il est très résistant, et est opérant dans une gamme de température qui convient parfaitement à l'utilisation que l'on veut en faire.

3.3 La conception mécanique

Alors que nous pensions initialement qu'elle serait au centre de notre projet, la conception mécanique se révéla relativement courte à réaliser, en voilà les détails.

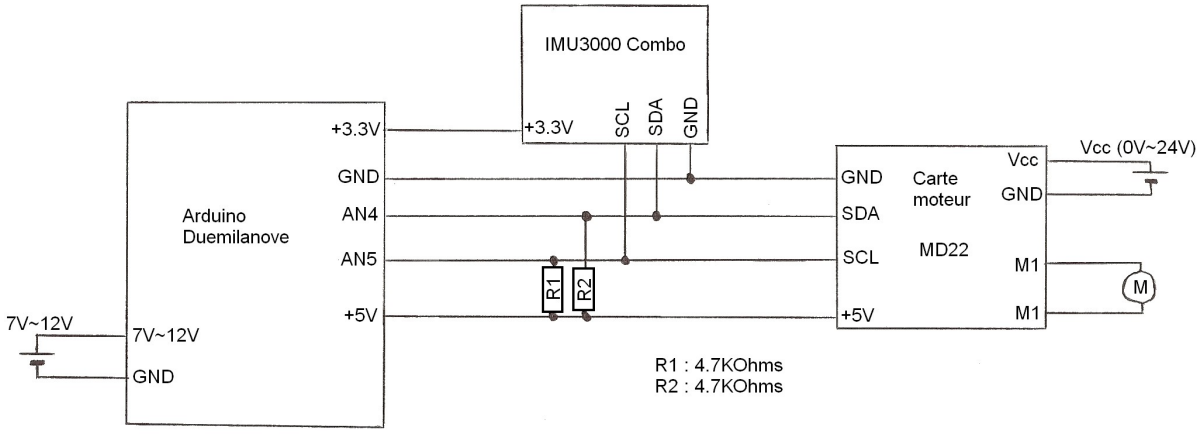


Illustration 14: Schéma de notre montage final

Comme le capteur gyroscopique/accéléromètre et la carte moteur fonctionnent grâce au bus I2C, seules les sorties analogiques 4 et 5 de la carte arduino sont utilisées. Ainsi la sortie analogique 4 est commune aux broches SDA du capteur et de la carte moteur, et la sortie analogique 5 est commune aux broches SCL. Deux résistances de pull-up ont été insérées pour maintenir à l'état haut les deux lignes SDA et SCL en l'absence de signal envoyé.

3.4 La programmation

Avant de réaliser nos programmes nous nous sommes inspirés de deux

programmes trouvés sur internet concernant un moteur commandé par une carte moteur du même type que la notre et la carte IMU3000, ainsi que le programme réalisé par les élèves ayant travaillé sur le Segway. (se reporter aux annexes relatives à la programmation).

Pour créer le programme qui allait gérer le moteur, il nous a fallu d'abord nous familiariser avec le langage Arduino. Pour cela, nous avons effectué de nombreux tests sur des programmes relativement simples afin de comprendre et de mettre en œuvre le bon fonctionnement du moteur. C'est principalement sur le net que nous avons trouvé les informations nécessaires à la réalisation du programme, que ce soit sur le site ww.arduino.cc ou encore sur différents forum qui traite de problèmes de programmations arduino.

Mais avant tout nous établîmes un petit algorithme afin de guider notre programmation :

Lecture valeur gyroscope
(Toute les 0,1s)

Tant que (angle ≠0)
Faire

Sinon
Si angle > 0

Sens moteur : droite
Lecture valeur accéléromètre (toute les 0,01 s)

Si accélération faible (a ≤Constant) alors moteur tourne avec une vitesse faible

Sinon moteur tourne vite}

Sens moteur : gauche :
Lecture valeur accéléromètre (toute les 0,01 s)

Si accélération faible (a ≤Constant) alors moteur tourne avec une vitesse faible

Sinon moteur tourne vite

Moteur tourne à gauche :
Lecture valeur accéléromètre (toute les 0,01 s)

{ si accélération faible(a ≤Constant) alors moteur tourne avec une vitesse faible
Sinon moteur tourne vite}

Sinon arrêt moteur

3.4.1 Le programme de la carte moteur MD22 :

Le premier programme nous a permis de faire communiquer entre eux la carte moteur MD22 et un petit moteur, qui sera par la suite remplacer par le moteur MY1018 plus adapté à répondre aux efforts mécaniques du problème. Pour cela il nous a fallu définir les différentes adresses des composants. Ainsi nous avons utilisé

la documentation technique de la carte MD22 (voir partie présentation des composants) et nous avons ainsi choisit l'adresse 0xB0. De plus, on a trouvé et utilisé le protocole de communication en bus I2C de la carte MD22 dans la documentation technique de la carte.

Le programme ordonne simplement au moteur des actions simples comme tourner dans un sens, dans l'autre ou encore arrêt…

```
#include <SoftwareSerial.h>              // appel des bibliothèques utilisées
#include <Wire.h>

#define md22Address 0x58                 // Adresse de la carte 0xB0 (en hexadécimal
B0=58)
#define softReg 0x07                     // Byte for reading software register
#define motorleft 0x01                   // Byte de l'instruction moteur tourne à  gauche
#define motorright 0x02                  // Byte de l'instruction moteur tourne à droite
#define accelReg 0x03                    // Byte de l'instruction d'accélération

void setup(){

  Wire.begin();
  delay(100);                            // Attendre que tout s'allume
   getSoftware();                        // Fonction qui renvoit les  de fonctionnement à l'écran
  setMode();                             // Fonction qui passe en mode 2 et définit l'accélération
}

void loop()                              // début de la boucle d'instructions

{
  Wire.beginTransmission(md22Address);   //  transmition à l'adresse 0xB0
  Wire.send(motorleft);                  // transmission  à l'instrucion moteur  tourne a
gauche
  Wire.send(255);                        // vitesse maximum
  Wire.endTransmission();                // fin de la transmission

  delay(6000);                           // durée de l'action: 6s.
```

On demande au moteur de s'arrêter pendant 3s.

```
  Wire.beginTransmission(md22Address);
  Wire.send(motorleft);
  Wire.send(0);
  Wire.endTransmission();
  delay(3000);
```

On demande au moteur de tourner vers la droite en plein régime pendant 3s.

```
   Wire.beginTransmission(md22Address);
  Wire.send(motorright);
  Wire.send(255);
  Wire.endTransmission();
  delay(3000);

  Wire.beginTransmission(md22Address);
  Wire.send(motorleft);
  Wire.send(128);                        // instruction qui ordonne au moteur de s'arrêter
  Wire.endTransmission();
  delay(3000);
```

```
}
void getSoftware(){                          // Reads abd displays the software version of MD22
  Wire.beginTransmission(md22Address);              // Calles software register
  Wire.send(softReg);
  Wire.endTransmission();

  Wire.requestFrom(md22Address, 1);               // Requests one byte
  while(Wire.available() < 1);                  // Wait for it to arrive
  int software = Wire.receive();                 // Get byte
}
void setMode(){
  Wire.beginTransmission(md22Address);              // Set a value of 255 to the acceleration register
  Wire.send(accelReg);
  Wire.send(0xFF);
  Wire.endTransmission();
}
```

3.4.2 Le programme gyroscope :

Nous avons par la suite effectué des tests sur le capteur gyroscopique IMU 3000 qui nous a permis de comprendre comment filtrer et extraire les valeurs utiles données par le capteur. Nous nous sommes ainsi servi de ce programme dans le programme principal pour utiliser les valeurs enregistrées par le capteur IMU 3000.

3.4.3 Le programme principal :

Pour réaliser le programme principal, nous nous sommes servi de nos compétences que l'on venait d'acquérir grâce à la conception des deux programmes ci-dessus et l'on s'est également inspiré du programme commandant le segway (réalisé par des membres du groupe de robotique de l'année 2009-2010).

```
#include <Wire.h>                                    // appel des bibliothèques utilisées
pour l'I2C

On définit les adresses des composants
#define GYRO 0x68        // gyro I2C address
#define REG_GYRO_X 0x1D   // IMU-3000 Register address for GYRO_XOUT_H
#define ACCEL 0x53       // Accel I2c Address
#define ADXL345_POWER_CTL 0x2D
#define md22Address 0x58                    // address of md 22 (0xB0)

char buffer[12];   // Array to store ADC values              // création d'un tableau pour stocker des
valeurs
int gyro_x;
On définit les variables
int gyro_y;
int gyro_z;
int accel_x;
int accel_y;
int accel_z;
int i;

void setup()
{
   analogReference(DEFAULT);
   Wire.begin();                                  // initialisation du protocole I2C comme esclave
   writeTo(GYRO, 0x16, 0x0B);                     //réglage des paramètres du gyroscope
   writeTo(GYRO, 0x18, 0x32);                      //réglage de l'adresse pour l'accélération
```

```
   writeTo(GYRO, 0x14, ACCEL);                    //réglage de l'adresse esclave de l'i2c : accel
   writeTo(GYRO, 0x3D, 0x08);                     //réglage du power control pour mesurer les valeurs
   writeTo(ACCEL, ADXL345_POWER_CTL, 8);
   writeTo(GYRO, 0x3D, 0x28);

 Serial.begin(9600);
  delay (100);
}
```

On crée la sous classe write to qui va permettre de lire les valeurs enregistrées par le capteur gyroscopique IMU3000

```
void writeTo(int device, byte address, byte val)

 {
  Wire.beginTransmission(device); // start transmission to device
  Wire.send(address);            // send register address
  Wire.send(val);                // send value to write
  Wire.endTransmission();        // end transmission
}

void loop()                                          // début de la boucle d'instructions
{

float kG=0.98,kA=0.02,angle,dt=0.01;                 // Constante définie expérimentalement
int valMotor;

for(int i=0;i<4;i++)

          {

readALL(&gyro_x, &gyro_y, &gyro_z, &accel_x, &accel_y, &accel_z);      // appel de la sous-classe readALL
angle=kG*(angle+gyro_y*dt)+kA*accel_x;                                 // définitions de la valeur de l'angle
valMotor=(int)240*abs((0.20*gyro_x/512+0.80*accel_x/512));     // définitions de la valeur de vitesse du moteur


                    if (angle>0)
                              {
```

On applique les instructions de l'algorithme : le moteur tourne dans un sens ou dans l'autre en fonction de la valeur de l'angle retournée par le capteur. On détermine la valeur de vitesse du moteur grâce à la variable valmotor définie ci-dessus

```
                              commandeMotor(1,valMotor);

                              }
                    else
                              {
                              commandeMotor(2,valMotor);

                              }
             }

            delay(10);                  // délai de la boucle  (10ms)

}
```

========== Fonction de lecture des valeurs Accéléromètre et gyromètre==========

```
void readALL(int *gyro_x, int *gyro_y, int *gyro_z, int *accel_x, int *accel_y, int *accel_z)

    //lecture des valeurs de l'accélération sur les axes X, Y, Z et de l'angle sur les axes X, Y et Z


  {

   Wire.beginTransmission(GYRO);                        // Réglage de l'adresse pour l'axe X du
gyroscope
   Wire.send(REG_GYRO_X);
   Wire.endTransmission();

   Wire.beginTransmission(GYRO);
   Wire.requestFrom(GYRO,12);
   i = 0;
   while(Wire.available())
   {
      buffer[i] = Wire.receive();
      i++;
   }
   Wire.endTransmission();

   //Combine bytes into integers
   // Gyro format is MSB first
   *gyro_x = buffer[0] << 8 | buffer[1];
   *gyro_y = buffer[2] << 8 | buffer[3];
   *gyro_z = buffer[4] << 8 | buffer[5];
   // Accel is LSB first. Also because of orientation of chips
   // accel y output is in same orientation as gyro x
   // and accel x is gyro -y
   *accel_y = buffer[7] << 8 | buffer[6];
   *accel_x = buffer[9] << 8 | buffer[8];
   *accel_z = buffer[11] << 8 | buffer[10];

delay(200);          // délai de la boucle  (200ms)

}


 ============================ Fonctions de commande moteurs
========================

void commandeMotor (int direccion,int commande)
{
        Wire.beginTransmission(0xB0);
        Wire.send(0x00);
        Wire.send(direccion);                           //va définir la direction(1=gauche, 2=droite)
        Wire.send(0x02);
        Wire.send(commande);                                    //  doit être une valeur entre 1 et 243
(commande right)
        Wire.endTransmission();                                 //fin de la transmission
}


 ================================ Fonctions de lecture des valeurs Moteurs
====================
void lectureMotor(int *infoMotor)
{
        Wire.beginTransmission(0xB0);
        Wire.send(0);                                   //  on envoie 0 pour lui demander des valeurs
        Wire.endTransmission();                                 //fin de la transmission
        Wire.requestFrom(0x01, 3);                      //on demande à l'appareil 0x01 de nous
envoyer 3 bytes
        *infoMotor = Wire.receive();
```

```
}
```

3.4.4 Conclusion :

Lors de la programmation du programme principal de nombreux problèmes de compilation sont survenus, principalement au moment où l'on a fusionné le programme des lectures de valeurs du capteur gyroscopique avec le programme de commande moteur. On a ainsi dû adapter les différentes variables pour rendre compatibles les deux programmes.

4 Conclusions et perspectives

4.1 Conclusion sur le travail réalisé et perspectives

Difficile, c'est le mot qui définit au mieux notre projet, d'une part dans la compréhension des moyens de parvenir à sa réussite et d'autre part dans sa réalisation. Le défi majeur de ce projet fut tout d'abord de comprendre tous les enjeux et de mettre en pratique toutes les techniques utilisées dans l'univers de la robotique.

Ce projet, nous a permis de découvrir une nouvelle utilisation de la programmation. Il nous a fallu énormément de temps pour en partie apprivoiser le langage Arduino, et comprendre, globalement, comment l'utiliser. Cependant, il est nécessaire de rappeler que nous nous sommes énormément inspirés de tous les programmes Arduino qui nous ont été mis à disposition, que ce soit par notre professeur (notamment avec le programme du Segway) ou sur internet.

Ce projet est réellement un travail d'équipe, en effet, si nous travaillions, dans notre groupe, à 6, nous étions totalement tributaires du travail, d'un autre groupe ce qui a rajouté une pression supplémentaire, sachant que tout notre projet n'était pas sous notre entier contrôle.

Pour finir il est certain qu'il nous reste des étapes que nous n'avons pu réaliser du fait du temps et des moyens qui nous étaient impartis. En effet l'interaction entre le capteur et le moteur semblent poser des problèmes de cohérence et nous n'avons pu faire aucun tests grandeur nature du fait que la partie mécanique n'était pas conçue, ce que pourront faire de futurs STPI-2, passionnés d'informatique, en reprenant nos travaux là où ils en sont restés. Mais cela reste toujours un plaisir de travailler en groupe, et c'est très enrichissant de se rapprocher un peu plus du monde du travail de cette manière.

4.2 Conclusions sur l'apport personnel de cet E.C. Projet

4.2.1 Conclusion de N. BEDOUET

Ce projet de P6_3 m'a permis de découvrir la robotique qui était, pour ma part, un domaine scientifique totalement inconnu. J'ai ainsi pris en compte le fait que ce dernier mêlait tout autant l'aspect théorique (algorithmes, programmation) que l'aspect pratique (soudures des composants, câblages…). Cet aspect plus « manuel » était, selon moi, le plus intéressant mais malheureusement il fut celui qui prit le moins de temps lors de ce projet.

J'ai également pu voir les réelles applications de la programmation (langage C++) sur des mécanismes électroniques quelque peu complexes et ainsi utiliser «le langage Arduino ».

Ce travail d'équipe fut une expérience très enrichissante au niveau humain car nous avons dû apprendre à travailler avec des personnes que l'on ne connaissait pas auparavant, et ainsi s'adapter aux méthodes de travail de chacun pour un produit optimal.

Pour conclure, je perçois les projets comme des aperçus de notre futur travail d'ingénieur et je pense que c'est une bonne chose de nous proposer des travaux

sous forme de projet à l'INSA.

4.2.2 Conclusion de K.Q. NGUYEN

Ce n'est pas le premier projet du 2e année mais ce projet de physique nous a permis de développer nos connaissances scientifiques et notre aptitude à travailler en équipe plus pratiquement. De plus, chaque étudiant a pu trouver un domaine dans lequel ses compétences lui permettaient de s'épanouir, informatique, mathématique et physique.
En fait, ce projet n'est pas compatible avec le domaine qui m'intéresse. Avant ce projet, je ne connaissais pas beaucoup la fonction ainsi que la constitution des cartes mémoires et les moteurs, grâce à lui j'ai beaucoup appris plusieurs informations utiles et intéressantes. J'ai eu peur face à ces difficultés mais après les efforts et le fait que l'assemblage marche j'ai été rassurée. Bien sur pour les étudiants étrangers, ce projet devient plus difficile lorsque l'on communique et partage le travail. Mais heureusement finalement cela semble bien. Enfin, merci pour l'enseignement du professeur et l'aide des autres surtout Michel qui ont été accueillants et m'ont expliqué beaucoup de fois malgré les difficultés de langue.

4.2.3 Conclusion de L. PETIT

CAO et réalisation électronique pour projet bateau en aviron de compétition. C'était là l'énoncé de notre sujet. Mais il s'est avéré que le projet reposait surtout sur la partie électronique, ce qui n'était pas pour me déplaire au début, mais ça s'est très vite montré compliqué.

Mais voilà, voir les robots fonctionner, faire ce qu'on veut qu'ils fassent, c'est pas aussi simple que ça en a l'air. C'est sans doute le point que je retiendrai longtemps de se projet.

Le projet physique m'a permis de découvrir comment fonctionnait les composants électroniques d'un ensemble robotisé, mais malheureusement le peu de connaissance que j'avais au début du semestre ne m'a pas permis de maîtriser totalement la programmation, qui rappelons le est en C++/bus I2C. Il m'a cependant permis de m'y intéresser et de vouloir en apprendre plus sur le sujet.

Ce projet m'a ainsi permis de voir différemment la robotique, qui avant d'être un ensemble et une suite de petits éléments qui doivent être interconnectés ensemble à l'aide de la programmation informatique. J'avais vraiment envie de mener à bien ce sujet, en tant que chef de projet, mais on s'est tous rendu compte très vite que la partie n'étais pas gagnée d'avance.

Je regrette simplement de ne pas avoir pu faire plus de CAO, et seulement de la programmation, qui n'est vraiment pas mon point fort.

4.2.4 Conclusion de T. ROSSELIN

La robotique c'est compliqué ! C'est le point principal que je retiendrai à propos de ce cours. Ce cours nous a permis de toucher du doigt le monde de la robotique, et j'ai trouvé cela très complexe. En effet, cette discipline de la mécanique s'apparente plus pour moi à de la programmation, domaine de l'informatique que je n'apprécie guère. La difficulté majeure de notre projet fût de rechercher des informations sur internet et de les comprendre, afin de les adapter à notre projet. Si les informations trouvées étaient nombreuses elles nous étaient souvent inaccessibles du fait de notre petit niveau de programmeur.

Mais le but de ce cours était, je pense de nous faire découvrir l'électronique ; les bus I2C, les capteurs gyroscopiques, les commandes de moteur électrique ou

encore les cartes Arduino ont désormais moins de secrets, tout cela dans un travail de groupe. J'ai d'ailleurs réellement apprécié le travail de groupe qui était assez hétérogène, chacun apportant son savoir et ses connaissances.

Pour finir, je suis simplement un peu déçu du résultat et car le coté conception assistée par ordinateur, qui était en tête du titre de ce projet est inexistant en réalité.

4.2.5 Conclusion de V.SAUVAGE

Avant ce projet, j'avais une très bonne image de la robotique et je portais un intérêt certain pour ce domaine, bien que ce ne fût pour moi qu'une vague représentation puisque je n'étais jamais entré dans le vif du sujet. Après ce projet, cette image s'est trouvée transformée et contrastée. En effet, la robotique, ce n'est pas une mince affaire. J'ai été déçu que la majeure partie du projet concerne la programmation, alors qu'il s'intitule « CAO et réalisation électronique ».

Comme nous étions absolument novices sur le sujet, nous avons donc du reprendre toute les bases de la robotique depuis zéro, et chercher ce que voulaient dire des mots barbares comme « I2C », « PID » ou encore « Arduino ».

Je retiendrais de ce projet l'organisation et le travail d'équipe que nous avons implémenté, ainsi que l'approche du monde de la robotique qu'il nous a apporté.

Malgré tout les moyens mis en place, il est regrettable que nous n'ayons pu terminer le projet, et faire des essais grandeur nature. Le projet reste tout de même une expérience enrichissante et indispensable pour nous former à notre futur métier d'ingénieur.

4.2.6 Conclusion de M. TOTAIN LYCZKO

Ce projet fut, par de nombreux aspects, une source d'enseignement.

Tout d'abord sur le plan de la culture scientifique il m'a, indéniablement, permis de découvrir un aspect de l'électronique qui m'était, jusque là, tout à fait étranger, à savoir la programmation et le mode de commande des composants. En effet, sans ce projet, moi qui me destinait à une carrière aux antipodes de l'électronique serais resté sans les moindres bases dans ce domaine.

De plus il nous a fait également progresser sur l'aspect organisationnel, en effet, nous qui étions tous de groupes différents et ne nous connaissions pas forcément, nous avons eu rapidement à nous répartir les tâches en fonctions des préférences et affinités de chacun pour que tous travaillent avec le plus de motivation possible et en autonomie à chaque fois que cela était possible, comme nous aurons très certainement à faire tout au long de notre carrière, quelle qu'elle puisse être.

Je ne peux que déplorer le fait que nous n'ayons pu mener le projet à son terme mais les progrès que nous avons pu faire dans le domaine de l'électronique, puisque nous partions tous d'un niveau proche de zéro, ne le rendent pas vain pour autant.

5 Bibliographie

Ressources en ligne :

Pages web :

-Description du module MD03.*Electronika.fr* [consulté le 15/3/2011]. Disponible à l'adresse : http://www.electronika.fr/blog/?p=105

-Module de commande de moteur "MD22".*Lextronic.fr*[consulté le 22/03/2011].Disponible à l'adresse : http://www.lextronic.fr/P1917-module-de-commande-de-moteur-md22.html

-Arduino : gestion des temporisations. *Laboelectronique.be* [consulté le 10/05/11]. Disponible à l'adresse : http://www.laboelectronique.be/ardtimer.html

-Interrupts. *Arduino.cc* [consulté le 10/05/11]. Disponible à l'adresse : http://arduino.cc/fr/Main/Interrupts

-Moteur 24V DC 250W. *Csmoto.fr* [consulté le 24/05/11] Disponible à l'adresse : http://pocketquadelectrique.csmoto.fr/informations/datasheet.php

-La carte Arduino Duemilanove. *Arduino.cc* [consulté le 12/05/11] Disponible à l'adresse : http://arduino.cc/fr/Main/MaterielDuemilanove

-IMU Fusion Board - ADXL345 & IMU3000. Sparkfun.com [consulté le 11/06/11]. Disponible à l'adresse : http://www.sparkfun.com/products/10252

Documents :

-MD22 - Dual 24Volt 5Amp H Bridge Motor Drive. *Robot-electronics.co.uk* [consulté le 15/03/2011]. Disponible à l'adresse : http://www.robot-electronics.co.uk/htm/md22tech.htm

-PDF : Carte de commande moteur MD03. *Selectronic.fr* [consulté le 5/04/2011]. Disponible à l'adresse : http://www.selectronic.fr/upload/produit/fichetechnique/06611.pdf

-PDF : PID Controller and PWM Generator for Motor Driver Based on PIC or ATMEL µController. *Alpen-Adria Universität.* [consulté le 17/05/11] Disponible à l'adresse : http://www.uni-klu.ac.at/tewi/downloads/2009_Master_proposal_motor_control.pdf

-PDF : IMU-3000 Motion Processing Unit Product Specification Rev 1.1 [consulté le 11/06/11]. Disponible sur : http://www.sparkfun.com/datasheets/Sensors/IMU/ps-imu-3000a-00-01.1.pdf

6 Annexes

6.1 La description des composants

Les documentations représentant plus d'une centaine de pages, par souci économique et écologique nous nous contenterons ici de donner les liens renvoyant à la documentation détaillée des composant.

6.1.1 La carte Arduinno Duemilanove

http://arduino.cc/fr/Main/MaterielDuemilanove

6.1.2 La carte moteurMD22

http://www.robot-electronics.co.uk/htm/md22tech.htm

6.1.3 La carte moteur MD03

http://www.selectronic.fr/upload/produit/fichetechnique/06611.pdf

6.1.4 Le capteur IMU3000

http://www.sparkfun.com/datasheets/Sensors/IMU/ps-imu-3000a-00-01.1.pdf

6.2 Les programmes nous ayant inspiré

6.2.1 Programme modifiant les valeurs d'accélération et de vitesse

```
#include <SoftwareSerial.h>
#include <Wire.h>


#define md22Address 0x58                                    // address of md 22
(all mode switches on)
#define softReg 0x07                                        // Byte for reading
software register
#define motorleft 0x01                                          // Byte for first
motor
#define motorright 0x02                                       // Byte to set
vitesse
#define accelReg 0x03                                      // Byte to set
```

```
acelleration



void setup(){

  Wire.begin();
  delay(100);                                          // Wait for
everything to be powered up
   getSoftware();                                       // Function that
gets and prints software revision to screen
  setMode();                                           // Function that sets
mode to 2 and sets acceleration
}

void loop(){



  Wire.beginTransmission(md22Address);                 // Set first motor to
a speed of 255
  Wire.send(motorleft);
  Wire.send(255);
  Wire.endTransmission();

  delay(6000);


  Wire.beginTransmission(md22Address);                 // Set first motor to
speed 0
  Wire.send(motorleft);
  Wire.send(0);
  Wire.endTransmission();
  delay(3000);

   Wire.beginTransmission(md22Address);                 // Set first motor
to speed 0
  Wire.send(motorright);
  Wire.send(255);
  Wire.endTransmission();
  delay(3000);

  Wire.beginTransmission(md22Address);                 // Set first motor to
speed 0
  Wire.send(motorleft);
  Wire.send(0);
  Wire.endTransmission();
  delay(3000);


}
void getSoftware(){                                    // Reads abd displays
the software version of MD22
  Wire.beginTransmission(md22Address);                 // Calles software
register
  Wire.send(softReg);
  Wire.endTransmission();

  Wire.requestFrom(md22Address, 1);                    // Requests one byte
  while(Wire.available() < 1);                         // Wait for it to
arrive
  int software = Wire.receive();                       // Get byte
}
void setMode(){
  Wire.beginTransmission(md22Address);                 // Set a value of 255
```

```
to the acceleration register
  Wire.send(accelReg);
  Wire.send(0xFF);
  Wire.endTransmission();
}
```

6.2.2 Programme renvoyant les valeurs relevées par la carte IMU3000 :

```
// IMU Fusion Board - ADXL345 & IMU3000
// Example Arduino Sketch to read the Gyro and Accelerometer Data



#define GYRO 0x68         // gyro I2C address
#define REG_GYRO_X 0x1D   // IMU-3000 Register address for GYRO_XOUT_H
#define ACCEL 0x53        // Accel I2c Address
#define ADXL345_POWER_CTL 0x2D

char buffer[12];   // Array to store ADC values
int gyro_x;
int gyro_y;
int gyro_z;
int accel_x;
int accel_y;
int accel_z;
int i;
#include <Wire.h>

void setup()
{
    Serial.begin(9600);

    Wire.begin();
    // Set Gyro settings
    // Sample Rate 1kHz, Filter Bandwidth 42Hz, Gyro Range 500 d/s
    writeTo(GYRO, 0x16, 0x0B);
    //set accel register data address
    writeTo(GYRO, 0x18, 0x32);
    // set accel i2c slave address
    writeTo(GYRO, 0x14, ACCEL);

    // Set passthrough mode to Accel so we can turn it on
    writeTo(GYRO, 0x3D, 0x08);
    // set accel power control to 'measure'
    writeTo(ACCEL, ADXL345_POWER_CTL, 8);
    //cancel pass through to accel, gyro will now read accel for us
    writeTo(GYRO, 0x3D, 0x28);

}

// Write a value to address register on device
void writeTo(int device, byte address, byte val) {
  Wire.beginTransmission(device); // start transmission to device
  Wire.send(address);             // send register address
  Wire.send(val);                 // send value to write
  Wire.endTransmission();         // end transmission
}

void loop()
{
    // Read the Gyro X, Y and Z and Accel X, Y and Z all through the gyro
```

```
    // First set the register start address for X on Gyro
    Wire.beginTransmission(GYRO);
    Wire.send(REG_GYRO_X); //Register Address GYRO_XOUT_H
    Wire.endTransmission();

    // New read the 12 data bytes
    Wire.beginTransmission(GYRO);
    Wire.requestFrom(GYRO,12); // Read 12 bytes
    i = 0;
    while(Wire.available())
    {
        buffer[i] = Wire.receive();
        i++;
    }
    Wire.endTransmission();

    //Combine bytes into integers
    // Gyro format is MSB first
    gyro_x = buffer[0] << 8 | buffer[1];
    gyro_y = buffer[2] << 8 | buffer[3];
    gyro_z = buffer[4] << 8 | buffer[5];
    // Accel is LSB first. Also because of orientation of chips
    // accel y output is in same orientation as gyro x
    // and accel x is gyro -y
    accel_y = buffer[7] << 8 | buffer[6];
    accel_x = buffer[9] << 8 | buffer[8];
    accel_z = buffer[11] << 8 | buffer[10];

    // Print out what we have
    Serial.print(gyro_x);  // echo the number received to screen
    Serial.print(",");
    Serial.print(gyro_y);  // echo the number received to screen
    Serial.print(",");
    Serial.print(gyro_z);  // echo the number received to screen
    Serial.print(",");
    Serial.print(accel_x);  // echo the number received to screen
    Serial.print(",");
    Serial.print(accel_y);  // echo the number received to screen
    Serial.print(",");
    Serial.print(accel_z);  // echo the number received to screen

    Serial.println("");     // prints carriage return
    delay(200);             // wait for a second
}
```

6.2.3 Le programme du segway en I2C (projet de l'année dernière)

```
#include <Wire.h>   // include Wire library for I2C

const int acceXRead=0;
const int acceZRead=1;
const int gyroXRead=2;
const int gyroYRead=3;
const int potentioRead=4;
void setup()
{
  analogReference(DEFAULT);
  Wire.begin();// inicialisation du protocole I2C comme esclave
  Serial.begin(9600); // initialize serial communications
  delay (100);
}
```

```
void loop()
{

float degAccX,degAccZ,degGyrX,degGyrY;
float tableValeurs[10], intergralValue;
float kG=0.98,kA=0.02,angle,dt=0.01;
int valMotor;
float potentio;
potentio=readPotentiometre();
for(int i=0;i<4;i++)
        {
          readAccelerometre(&degAccX, &degAccZ);
          readGyroscope(&degGyrX, &degGyrY);
          angle=kG*(angle+degGyrX*dt)+kA*degAccX;
          valMotor=(int)240*abs((0.20*degGyrX/512+0.80*degAccX/512));
                if (angle>0)
                        {
                        // on ne prend pas en compte la direction car elle
est donnee par le signe de la valeur de la fonction potentiometre
                                commandeMotorLeft(1,valMotor+potentio/valMo
tor);
                                commandeMotorRight(2,valMotor-
potentio/valMotor);
                        }
                else
                        {
                        commandeMotorLeft(2,valMotor+potentio/valMotor);
                        commandeMotorRight(1,valMotor-potentio/valMotor);
                        }
          }
          delay(10);
}

// ================================ Fonction de lecture des valeurs
Acceleromètre=========================================
void readAccelerometre(float *degAccX, float *degAccZ)
{
float x_acc;
float z_acc;
float x_accsum=0;
float z_accsum=0;
float acc_offset=357;
int valAccX=0;
int valAccZ=0;
        for(int i=0;i<20;i++)
            {
            valAccX=analogRead(acceXRead); //lecture de la valeur d'entrée
sur la broche A0
            valAccZ=analogRead(acceZRead); //lecture de la valeur d'entrée
sur la broche A1
            x_accsum=x_accsum+valAccX;
            z_accsum=z_accsum+valAccZ;
            }
        x_acc=(x_accsum/20) - acc_offset; //valeur moyenne de x_accsum qui
est la somme des 20 valeurs envoyés par l'accéléro, le '-512' permet de
ramener la plage de valeur [0;1023] à [-512;+512]
        z_acc=(z_accsum/20) - acc_offset; //valeur moyenne de z_accsum qui
est la somme des 20 valeurs envoyés par l'accéléro
        *degAccX=-x_acc/1,2857; //Convertion de la valeur analogique (-
250,250) en valeur angulaire plus facilement manipulable, La valeur -0.77 a
été déterminé expérimentalement
        *degAccZ=z_acc/-0.7777; // Pas sur de la nécessité de passer en
valeur angulaire en z?
}
// ================================ Fonction de lecture des valeurs
```

```
Gyroscope=========================================
void readGyroscope(float *degGyrX, float *degGyrY)
{
float valGyrX;
float gyr_offset=512; // valeur de reference a definir
float valGyrY;
        valGyrX=analogRead(gyroXRead);
        valGyrY=analogRead(gyroYRead);
        *degGyrX=(valGyrX-gyr_offset)/0.6206;
        *degGyrY=(valGyrY-gyr_offset)/0.6206;
}
// ================================ Lecture du potentiomètre
float readPotentiometre()
// attention a bien envoyer une valeur centree a zero en position
d'equilibre
{
        float potentio;
        potentio=analogRead(potentioRead)-736; // 736 est la valeur lu pour
le manche en position verticale
        return potentio;
}
// ================================== Fonctions de commande moteurs
================================

void commandeMotorLeft(int direccion,int commandeLeft)
{
        Wire.beginTransmission(0xB0);      // part address is 0x01 or
0101111b   =======verifier l'adresse======
        Wire.send(0x00);
        Wire.send(direccion);
        Wire.send(0x02);
        Wire.send(commandeLeft); //  doit être une valeur entre 1 et 243
(commande right
        Wire.endTransmission(); //fin de la transmission
}
void commandeMotorRight(int direccion,int commandeRight)
{
        Wire.beginTransmission(0xB2);      // part address is 0x02 or
0101111b   =======verifier l'adresse======
        Wire.send(0);
        Wire.send(direccion);
        Wire.send(2);
        Wire.send(commandeRight); //  doit être une valeur entre 1 et 243
(commande right
        Wire.endTransmission(); //fin de la transmission

}
// ================================ Fonctions de lecture des valeurs
Moteurs ========================================
void lectureMotorLeft(int *infoMotorLeft)
{
        Wire.beginTransmission(0xB0);      // part address is 0x01 or
0101111b   =======verifier l'adresse======  )
        Wire.send(0); //  on envoie 0 pour lui demander des valeur
        Wire.endTransmission(); //fin de la transmission
        Wire.requestFrom(0x01, 3); //on demande à l'appareil 0x01 de nous
envoyer 3 bytes
        *infoMotorLeft = Wire.receive();
}

void lectureMotorRight(int *infoMotorRight)
{
        Wire.beginTransmission(0xB2);      // part address is 0x01 or
0101111b   =======verifier l'adresse======
        Wire.send(0); //  on envoie 0 pour lui demander des valeur
```

```
        Wire.endTransmission(); //fin de la transmission
        Wire.requestFrom(0x01, 3); //on demande à l'appareil 0x01 de nous
envoyer 3 bytes
        *infoMotorRight = Wire.receive();
}
```