Sommaire

Introduction

Remerciements

I. Présentation du projet

1. *Les différentes phases du projet*
2. *Le temps passé et l'avancement de chaque étape du projet*

II. Cahier des charges et présentation des éléments du projet

1. *Cahier des charges*
2. *Présentation des différents éléments du projet*

III. Choix des éléments constituant le projet

1. *Moteur*
2. *Gyroscope/accéléromètre*
3. *Châssis*
4. *Batterie*
5. *Carte programmable*

IV. Programmation LabVIEW

1. *Présentation du logiciel*
2. *Programme réalisé*

V. Réalisation de la carte électronique pour la connectique

VI. Conclusion

Annexes

IUT Belfort-Montbéliard

UFC UNIVERSITÉ DE FRANCHE-COMTÉ

Introduction

Etudiants en licence professionnelle VEGA, nous sommes amenés à choisir un sujet de projet tuteuré qui nous permet de mettre en pratique certains savoirs acquis au cours de la formation. Ce projet doit aussi nous apprendre à travailler en groupe sous contrainte de temps et enfin démontrer notre capacité d'initiative, d'autonomie et de responsabilité.

C'est dans cette optique que nous avons choisi un projet parmi plusieurs sujets disponibles. Ce projet s'intitule « Pendule inversé » et il a été proposé par Mr. Gavignet.

Le projet que nous avons choisi consiste à concevoir un pendule inverse sous le logiciel LabVIEW.
Celui-ci doit garder son équilibre donc rester à la verticale (l'ensemble est sur roues).
Pour mener ce projet nous avons choisi une carte programmable intégrant un FPGA qui a pour nom « myRIO » du constructeur National instruments.

Dans ce rapport nous allons tout d'abord présenter le déroulement du projet tuteuré, ensuite nous allons exposer le cahier des charges qui nous a été prescrit.
Enfin nous allons détailler le travail effectué qui se décompose en plusieurs parties.

Remerciements

Nous tenons à remercier toutes les personnes qui nous ont aidés pour ce projet de près ou de loin.

Aussi, nous remercions notre enseignant responsable du projet Mr. Gavignet, pour l'aide qu'il nous a apporté tout au long du projet, mais aussi pour nous avoir guidés dans nos démarches.

Nous n'oublions pas Vincent, le magasinier, pour toute son aide et pour ses conseils précieux.

I. Déroulement du projet tuteuré

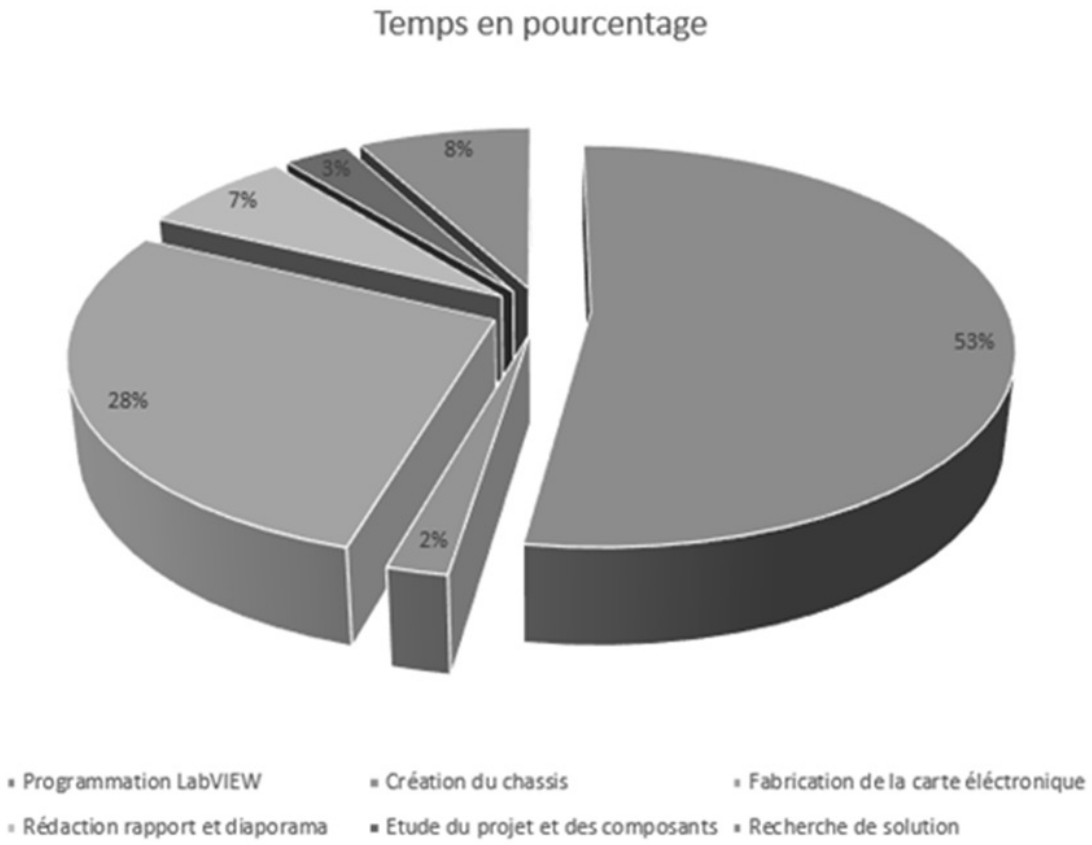
1. Les différentes phases du projet

Tout le projet s'est déroulé à l'IUT dans la salle VEGA.
Il s'est déroulé en 5 phases principales :

- **Phase de recherche capteurs**, sur laquelle nous avons travaillé les premières semaines.
- **Phase de programmation des capteurs**, cette phase consistait à faire dialoguer les capteurs choisis avec le logiciel LabVIEW.
- **Phase de programmation LabView des différents éléments,** sur laquelle nous avons passé le plus clair de notre temps car c'est le cœur du projet.
- **Phase d'élaboration de la carte électronique,** qui consistait à réaliser une carte électronique pour regrouper toutes les connectiques.
- **Phase de programmation commune,** étapes ou nous avons réuni les différentes parties (gyroscope, moteur, …) précédemment établies lors de la programmation des différents éléments.

2. *Le temps passé et l'avancement de chaque phase*

Ci-dessous nous avons un graphique nous montrant le temps passé sur notre projet. Nous pouvons voir qu'il y a deux tâches principales qui sont la programmation et la création de la carte électronique, viennent s'ajouter à cela quatre autres tâches qui sont la création du châssis, l'étude des composants et du projet, et pour finir il y a la recherche de solutions aux problèmes rencontrés.



II. Cahier des charges et présentation des éléments du projet

1. Cahier des charges

Notre objectif pour ce projet était donc de concevoir un pendule inversé à partir d'un système temps réel de type myRIO.

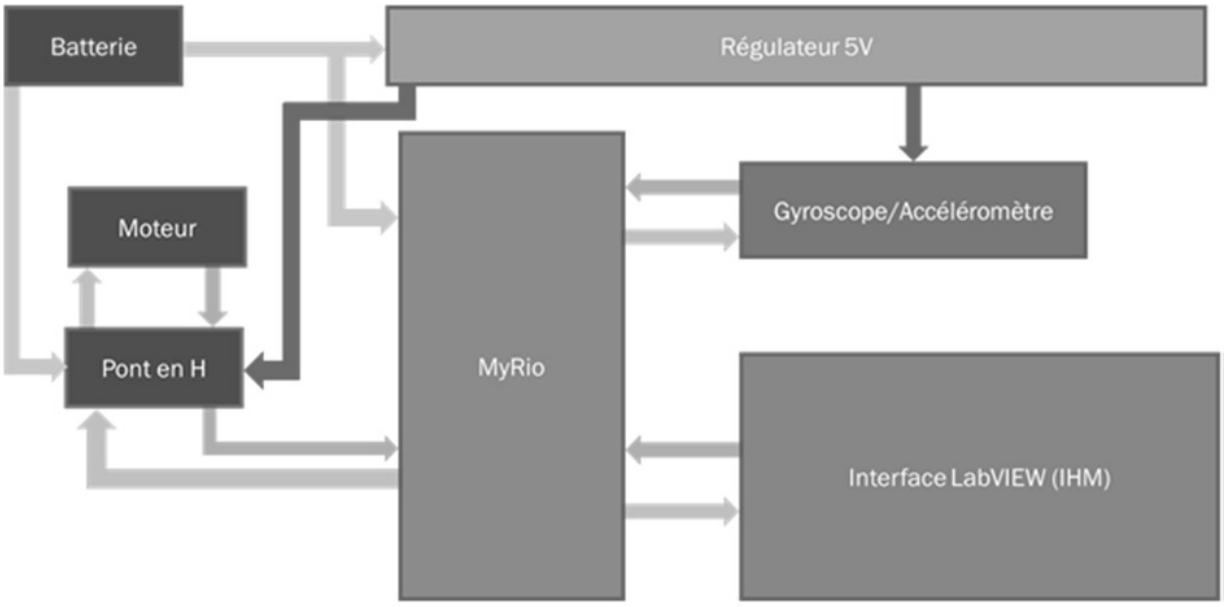
Le système final doit donc être capable de tenir en équilibre de façon autonome et de se déplacer en avant et en arrière.

Pour cela, nous avions à réaliser les tâches suivantes :

- Se documenter et choisir le matériel à utiliser (moteurs, gyroscope, batterie…).
- Effectuer des tests pour prendre en main le système myRIO avec LabVIEW.
- Etablir un diagramme de Gantt du projet.
- Faire un programme gérant la commande des moteurs (Sens de rotation, PWM…).
- Faire un programme permettant de décoder les données du gyroscope et récupérer les angles voulus.
- Réaliser la carte électronique permettant de rassembler tous nos composants sur une carte et diminuer le nombre de fils.
- Assembler les 2 parties et les faire fonctionner ensemble.
- Asservir le système grâce à un correcteur PID.
- Effectuer des tests de bon fonctionnement du système.

2. *Présentation des différents éléments du projet*

Ci-dessous, nous avons le schéma synoptique ou sont détaillées toutes les interactions entre les différents composants du projet que ce soit pour la communication avec le myRIO ou pour leurs alimentations.



Nous avons les composants suivants :

- Le myRIO qui est l'unité qui contient le programme avec son microprocesseur et son FPGA.
- L'interface LabVIEW qui permet de commander le pendule et faire le programme.
- Le gyroscope et accéléromètre qui récupère les angles, la vitesse.
- Pont en H qui gère les moteurs.
- Les moteurs qui s'activent pour permettre au pendule inverse de tenir debout et de pouvoir se déplacer.
- La batterie qui permet l'alimentation en 12V de plusieurs éléments.
- Le régulateur 5V qui alimente plusieurs composants en 5V.

III. Choix des éléments constituant le projet

1. Moteur

Pour le moteur comme pour l'ensemble du projet nous nous sommes inspirés d'un projet déjà réalisé sur internet. Cependant, nous avons analysé très précisément les caractéristiques des éléments que nous avons sélectionnés.

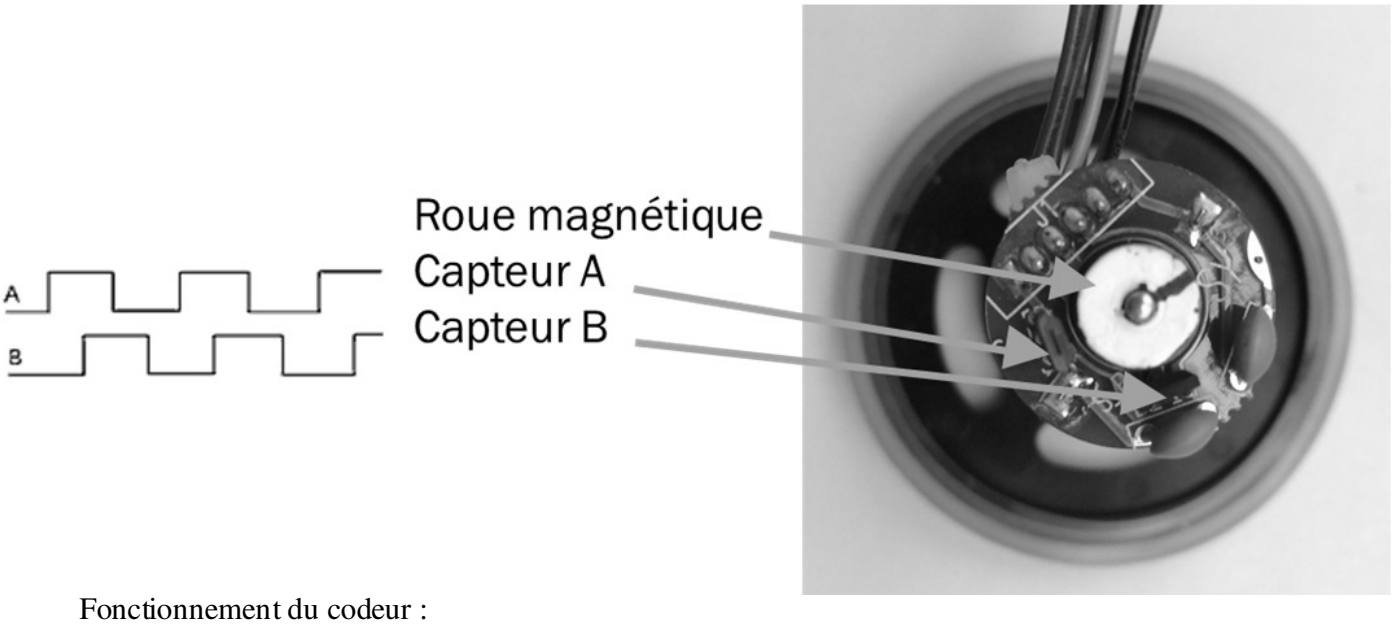
Nous avons donc étudié ce moteur :

C'est un moteur DC conçu pour des applications de robotique du fabricant Digilent.

Les caractéristiques sont les suivantes:

- ✓ Moteur DC 12v.
- ✓ Boite de vitesse (réduction nombre de tour augmentation du couple) rapport 1 :19.225.
- ✓ 89 tours par minute à la sortie de la boite de vitesse.
- ✓ Equipé d'un codeur absolu a deux capteur (A/B)
- ✓ Consommation hors charge 200 mA
- ✓ Consommation en charge 620 mA
- ✓ Couple radial de 8N
- ✓ Couple axial de 6N

Nous avons choisi ce moteur car il est compact, possède un couple assez fort pour notre utilisation, fonctionne à une tension continue de 12V et a une faible consommation ce qui permet une alimentation par batterie. Pour conclure, un des gros avantages de ce moteur est qu'il possède un codeur absolu directement équipé à l'arrière ce qui permet d'obtenir la vitesse mais aussi le sens de rotation de celui-ci.

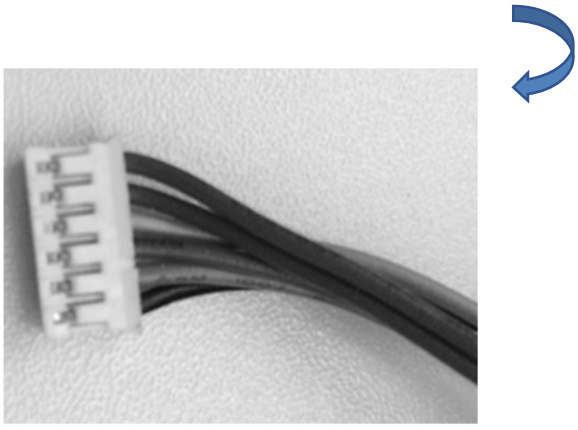


Fonctionnement du codeur :

Ce codeur permet une résolution de l'ordre de 1.57°.

En sortie de ce moteur nous avons un connecteur 6 broches comportant l'alimentation et le retour d'information des capteurs.

Grâce à son connecteur, ce moteur peut directement venir se connecter sur le pont en H que nous avons choisi.



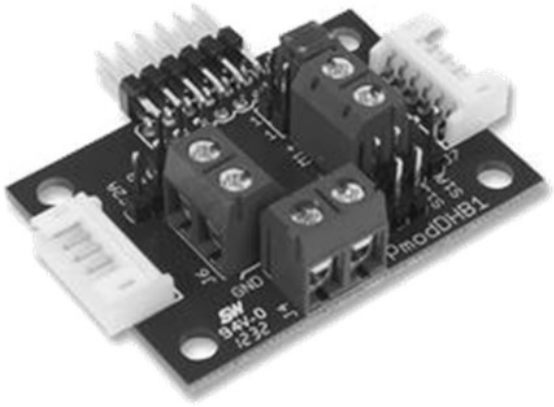
- Fils violet : Capteur B
- Fils bleu : Capteur A
- Fils vert : Capteur GND
- Fils marron : Capteur Vcc
- Fils rouge : +Moteur
- Fils noir : -Moteur

Pour plus d'informations sur le moteur, la documentation se trouve en Annexe 1.

Pour commander le moteur nous avons besoin d'un pont en H, qui va nous permettre de faire passer plus de courant, mais aussi de commander les moteurs, ON/OFF mais aussi Avant/Arrière. Nous avons choisi ce pont car il est spécialement conçu pour les moteurs que nous utilisons.

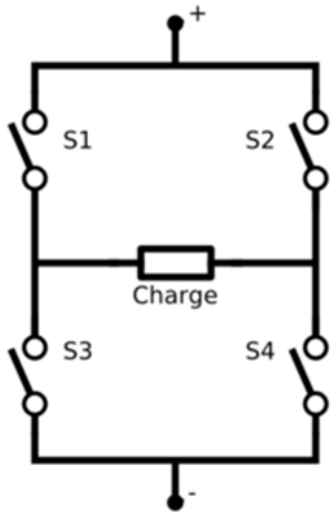
Les caractéristiques :

- ✓ Double pont en H
- ✓ 1.5A RMS
- ✓ Protection contre les surintensités
- ✓ Compatible 3.3V et 5V (pour la partie commande)
- ✓ 2 connecteurs 6PIN pour les deux moteurs Digilent vus précédemment
- ✓ Tension de sortie moteur recommandé 10.8V max.



Header J1					
Pin	Signal	Description	Pin	Signal	Description
1	EN1	Motor 1 Enable	7	EN2	Motor 2 Enable
2	DIR1	Motor 1 Direction	8	DIR2	Motor 2 Direction
3	S1A	Motor 1 Sensor A Feedback	9	S2A	Motor 2 Sensor A Feedback
4	S1B	Motor 1 Sensor B Feedback	10	S2B	Motor 2 Sensor B Feedback
5	GND	Power Supply Ground	11	GND	Power Supply Ground
6	VCC	Power Supply (3.3V/5V)	12	VCC	Power Supply (3.3V/5V)

Brochage du connecteur permettant la commande du pont en H.



Principe du pont en H :

En fonction des commutations des interrupteurs S1, S2, S3, S4, le moteur va tourner dans le sens horaire si on ferme S1 et S4 et inversement, si on ferme S2 et S3 le moteur va tourner dans le sens antihoraire.

Pour plus d'informations sur le pont en H la documentation se trouve en Annexe 2.

2. *Gyroscope/accéléromètre*

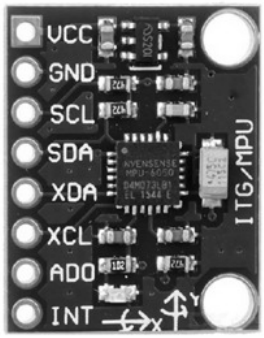
Pour ce qui est du gyroscope permettant de récupérer l'angle d'inclinaison de notre système, nous avons choisi le MPU6050.
Celui-ci permet d'assurer la fonction d'accéléromètre et de gyroscope dont on se sert pour notre application. Il communique en I2C.
I2C est un bus série synchrone bidirectionnel half-duplex, où plusieurs équipements, maîtres ou esclaves, peuvent être connectés au bus.
Les échanges ont toujours lieu entre un seul maître et un (ou tous les) esclave(s), toujours à l'initiative du maître (jamais de maître à maître ou d'esclave à esclave). Cependant, rien n'empêche un composant de passer du statut de maître à esclave et réciproquement.

A noter qu'il dispose également d'un capteur de température que nous avons utilisé pour vérifier le bon fonctionnement et faire des tests.

Nous avons choisi ce gyroscope car il était utilisé dans un projet déjà réalisé que nous avons vu sur internet et qu'il correspondait bien aux besoins que nous avions pour le projet.

Voici quelques caractéristiques utiles du MPU6050 dans notre projet :

- Tension d'alimentation : 2.375V-3.46V
- Communication I2c sur les pins SCL (Serial Clock Line) et SDA (Serial Data Line)
- Précision jusqu'à ± 2000 °/s pour le gyroscope
- Précision jusqu'à ± 16g pour l'accéléromètre



DIO15 / I2C.SDA	34	33	+3.3 V
DIO14 / I2C.SCL	32	31	DIO10 / PWM2
DGND	30	29	DIO9 / PWM1
DGND	28	27	DIO8 / PWM0
DIO13	26	25	DIO7 / SPI.MOSI
DGND	24	23	DIO6 / SPI.MISO
DIO12 / ENC.B	22	21	DIO5 / SPI.CLK
DGND	20	19	DIO4
DIO11 / ENC.A	18	17	DIO3
DGND	16	15	DIO2
UART.TX	14	13	DIO1
DGND	12	11	DIO0
UART.RX	10	9	AI3
DGND	8	7	AI2
AGND	6	5	AI1
AO1	4	3	AI0
AO0	2	1	+5V

Pour ce qui est du câblage, nous avons branché le MPU6050 directement sur des pins du myRIO. Nous avons donc utilisé la pin 33 qui délivre une tension de 3.3V qui alimente le gyroscope mais également la pin 30 pour la masse ainsi que les pins 34 et 32 pour la liaison I2C (34 pour SDA et 32 pour SCL).

Ensuite, pour récupérer et exploiter les données du gyroscope, il a fallu créer un soft en LabVIEW permettant de décoder les informations brutes qui sont reçues.

Le soft correspondant au décodage du MPU6050 est disponible en annexe 3 (face avant et diagramme).
Pour réaliser ce soft, il a fallu tout d'abord se renseigner et trouver les registres dont nous aurions besoin.

Voici les différents registres que nous avons utilisé (le tableau complet est disponible en annexe 4)

- Adresse du module : 0x68
- Configuration générale : 0x6B
- Configuration gyroscope : 0x1B
- Données accéléromètre : 0x3B à 0x40 (2 en X, 2 en Y et 2 en Z)
- Données gyroscope : 0x43 à 0x48 (2 en X, 2 en Y et 2 en Z)

L'adresse du module permet l'identification du module par le MyRIO.

La configuration générale permet à l'utilisateur de configurer le mode d'alimentation et la source d'horloge. Il fournit également un bit pour réinitialiser l'ensemble du dispositif, et un bit pour désactiver le capteur de température.

La configuration gyroscope est utilisée pour déclencher l'autotest du gyroscope et configurer la précision de celui-ci.

Les données accéléromètre sont récupérées en brut (valeur de 0 à 255) de même pour les données gyroscope.

Une fois les données acquises, il faut effectuer des manipulations pour les convertir et obtenir des valeurs avec des unités souhaitées.

- Accélération : division des données par 16384 pour obtenir une précision de ± 2g
- Gyroscope : division des données par 131 pour obtenir une précision de ± 250 °/s

Ainsi l'équation finale pour obtenir les angles en degrés est la suivante :

Angle en x = 0,98*(angle précédant+Gx*0,001/131) + 0,02*(angle en degrés)
Angle en y = 0,98*(angle précédant+Gy*0,001/131) + 0,02*(angle en degrés)
Angle en z = 0,98*(angle précédant+Gz*0,001/131) + 0,02*(angle en degrés)

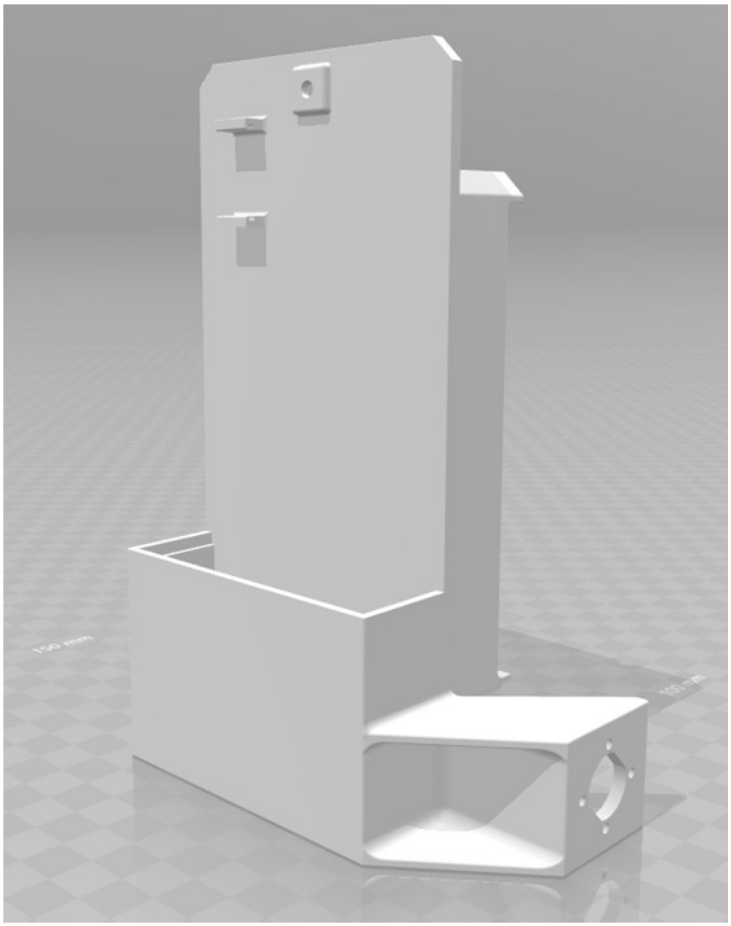
Les coefficients 0.98 et 0.02 sont utilisés afin d'empêcher les angles de dériver.
Il s'agit d'un filtre de Kalman.
Gx, Gy et Gz correspondent aux données du gyroscope pour chaque axe
L'angle en degrés correspondant à la deuxième partie de l'équation est calculé de la façon suivante :

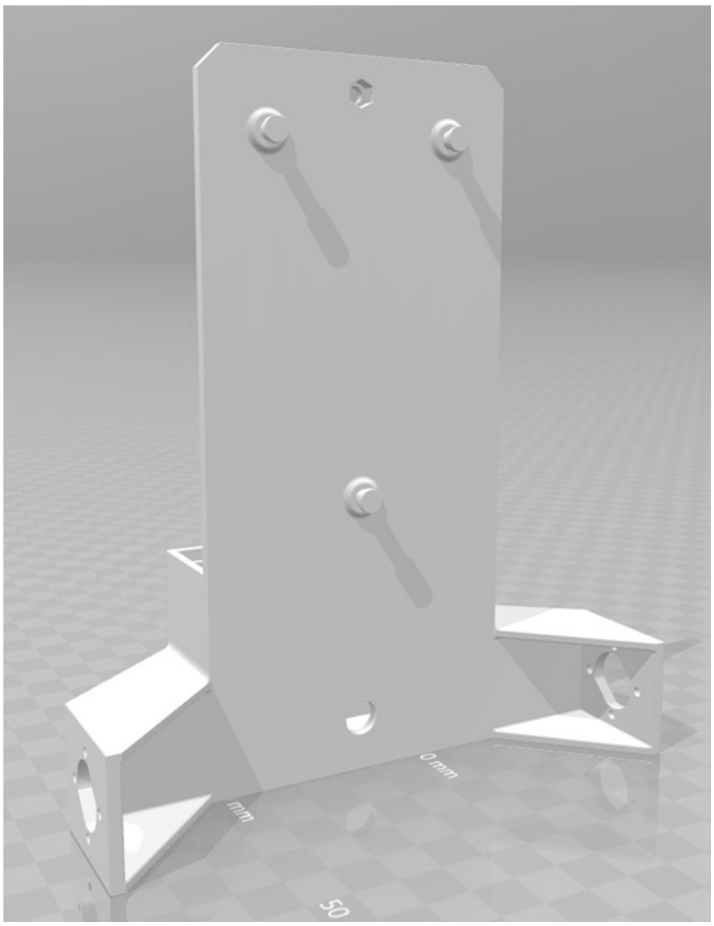
- Arctan (az/ay) * 180/pi en x
- Arctan (ax/az) * 180/pi en y
- Arctan (ax/ay) * 180/pi en z

Ainsi, l'équation finale permet d'avoir une valeur stable qui ne dérive pas dans le temps.

3. *Châssis*

Nous avons choisi ce châssis qui nous a été proposé sur le site WEB de Texas instruments.
Nous avons vérifié les mesures pour être sûr qu'elles correspondent à notre utilisation.
Voici la vue 3D du châssis que nous avons réalisé.





4. Batterie

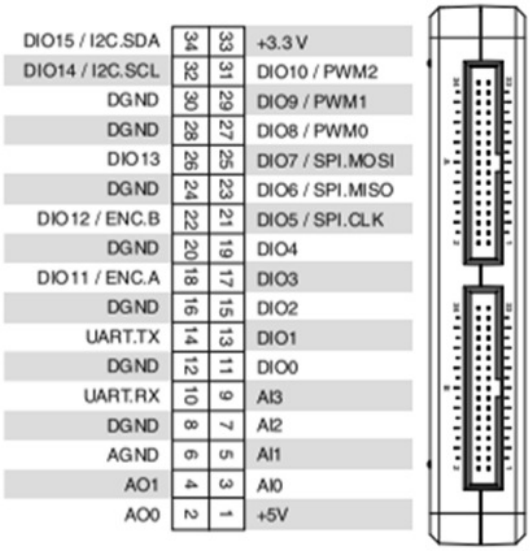
Les caractéristiques de la batterie utilisée sont les suivantes :

- Batterie TopFuel ECO-X 25C 1300mAh
- Capacité : 1300 mAh
- Nombres de cellules : 3
- Tension : 11,1 V
- Poids : 110 g
- Technologie : LiPo

Cette batterie alimente le myRio et le pont en H, et par la suite elle alimentera les moteurs et rendra le système autonome au niveau de l'alimentation. Elle a été choisie car elle a déjà été utilisée pour le même projet par d'autres personnes car leur utilisation au niveau des moteurs et des composants était quasiment identique. De plus, les caractéristiques techniques correspondent bien au projet que ce soit au niveau de son poids, de sa capacité ou de sa tension.

5. *MyRIO 1900*

Cet outil pour l'étude de systèmes embarqués, inclut des E/S qu'on retrouve sur les deux côtés du boîtier via les connecteurs MXP et MSP. Il est doté d'entrées analogiques, de sorties analogiques, de lignes d'E/S numériques, de LED, d'un bouton poussoir, et intègre un accéléromètre, un FPGA Xilinx et un processeur double cœur ARM Cortex-A9. Certains modèles incluent le support Wifi. L'outil myRIO est programmable en LabVIEW ou en C.
Les détails et les caractéristiques techniques sont disponibles en annexe 5.



DIO15 / I2C.SDA	34	33	+3.3 V
DIO14 / I2C.SCL	32	31	DIO10 / PWM2
DGND	30	29	DIO9 / PWM1
DGND	28	27	DIO8 / PWM0
DIO13	26	25	DIO7 / SPI.MOSI
DGND	24	23	DIO6 / SPI.MISO
DIO12 / ENC.B	22	21	DIO5 / SPI.CLK
DGND	20	19	DIO4
DIO11 / ENC.A	18	17	DIO3
DGND	16	15	DIO2
UART.TX	14	13	DIO1
DGND	12	11	DIO0
UART.RX	10	9	AI3
DGND	8	7	AI2
AGND	6	5	AI1
AO1	4	3	AI0
AO0	2	1	+5V

Pour notre projet, voici les éléments que nous avons utilisés :

- Des E/S numérique pour la PWM (PIN 25, 27,29)
- Des E/S numériques pour les encodeurs (PORT C)
- Alimentation du gyroscope (PIN 33)
- Communication I2C pour le gyroscope (PIN 32,34)

IV. Programmation LabVIEW

1. Présentation du logiciel

Le nom du logiciel LabVIEW signifie " Laboratory Virtual Instrument Enginerring Workbench". Ce langage de programmation graphique est un environnement de programmation à caractère universel particulièrement bien adapté à la mesure, au test, à l'instrumentation et à l'automatisation. Il a été développé par la société National Instrument à partir de 1983. LabVIEW est un des premiers langages de programmation graphique destiné au développement d'applications d'instrumentation. Couplé à des cartes d'entrées/sorties, il permet de gérer des flux d'informations numériques ou analogiques et de créer ou de simuler des instruments de mesures (oscilloscope, générateur de signaux, multimètre, etc.).

Quel type de programmation utilise-t-on pour LabVIEW ?

Le langage propre de LabVIEW est une programmation graphique en langage « G ». Le langage G se base sur le principe du flot de données, auquel ont été rajoutées des structures de programmation afin d'obtenir un langage de programmation complet.

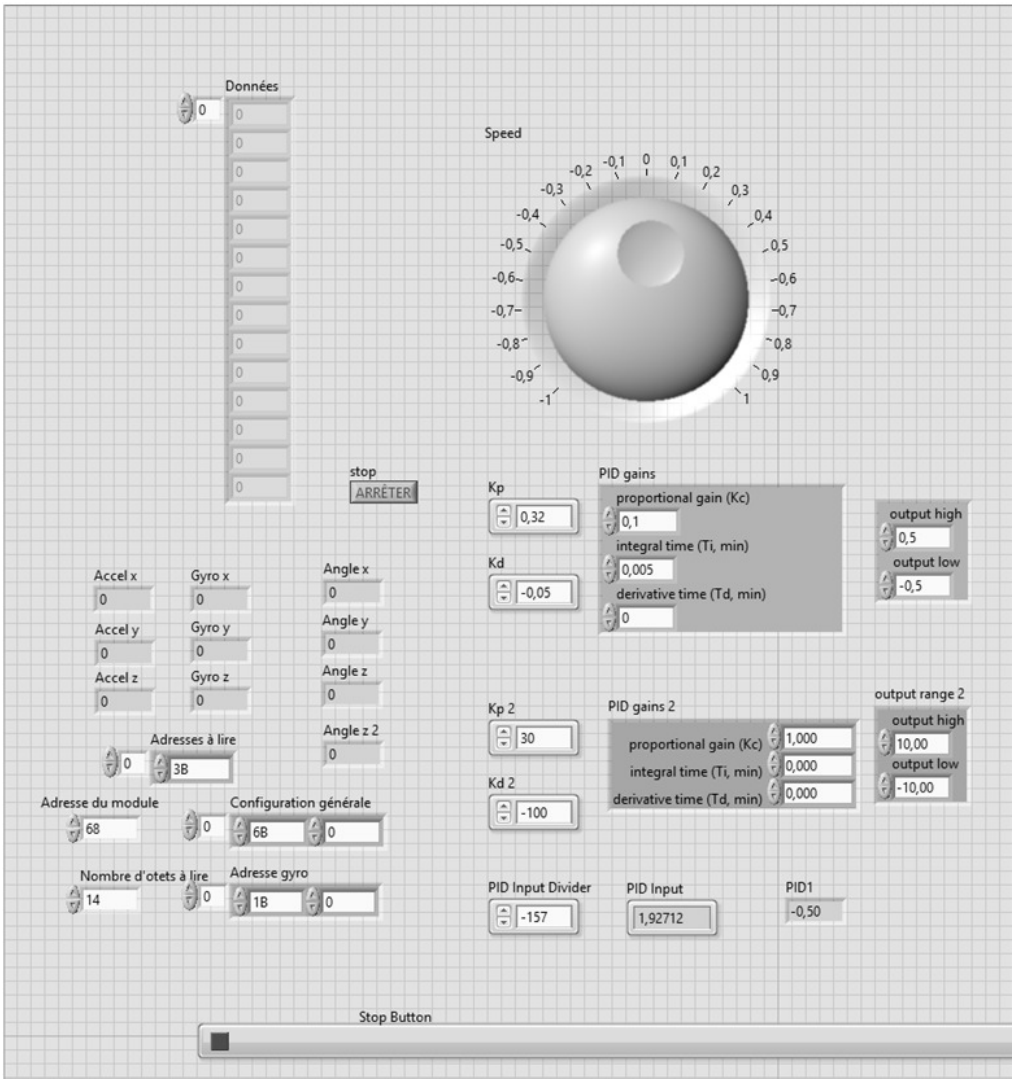
Un langage textuel suit des règles d'exécution séquentielle déterminée par la position des instructions dans le fichier source, tandis qu'un programme graphique s'exécute selon le principe de flux de donnée.

Les programmes LabVIEW comportent des Instruments Virtuels ou Virtual Instruments (VI). On parle d'instruments virtuels car leur apparence et leur fonctionnement sont semblables à ceux d'instruments réels, tels que les oscilloscopes et les multimètres.

Les VI se comportent de 3 éléments principaux :

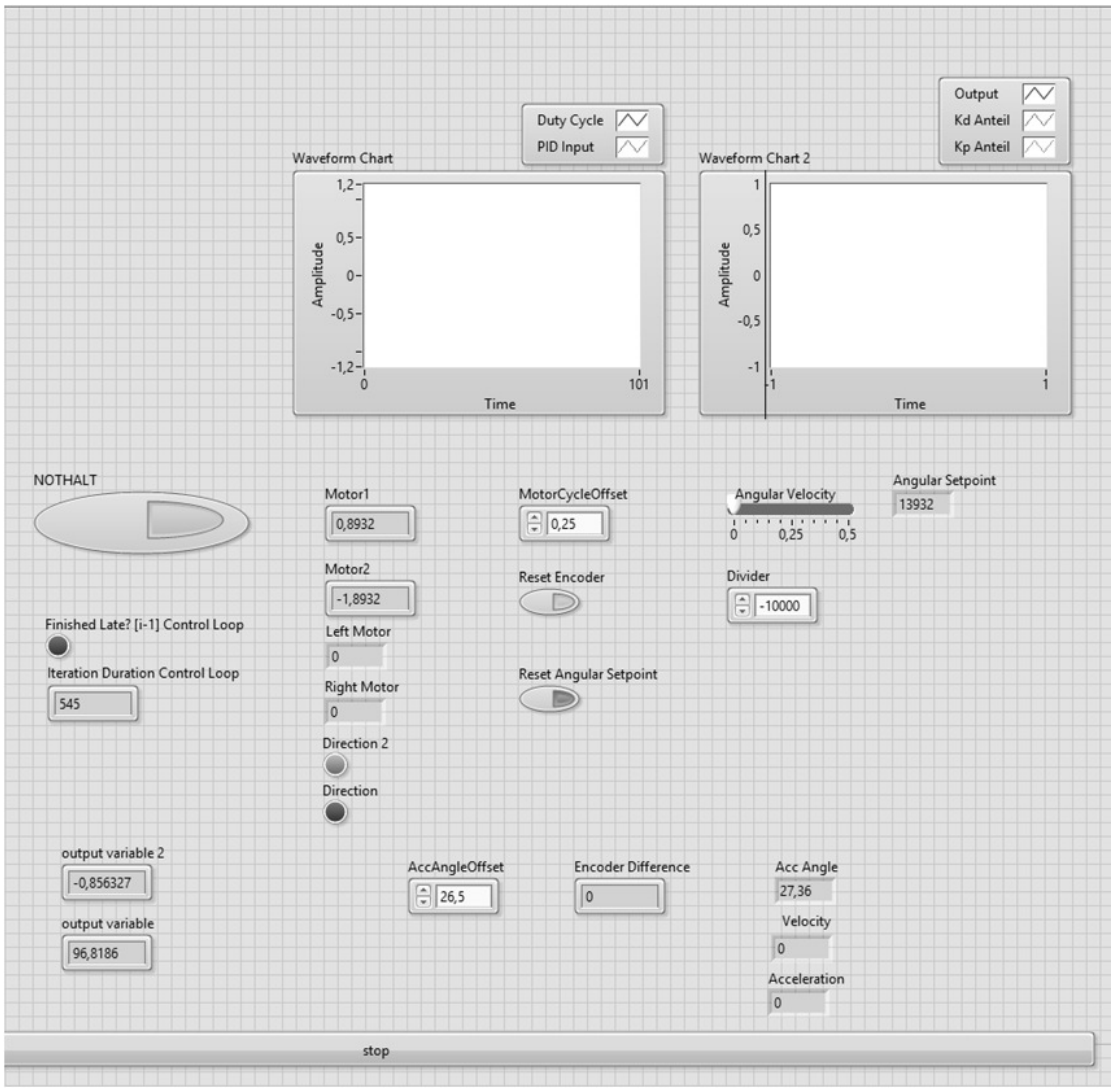
- Face avant du VI
- D'un diagramme
- D'une icône et d'un « connecteur ».

2. *Programme réalisé*



Nous avons ci-dessus la 1ère partie du programme final de notre projet.
Celle-ci comporte les registres à configurer pour le gyroscope, les données reçues ainsi que les valeurs des angles en degrés.

Elle comporte également les différents paramètres du correcteur PID qui permet d'asservir le système.



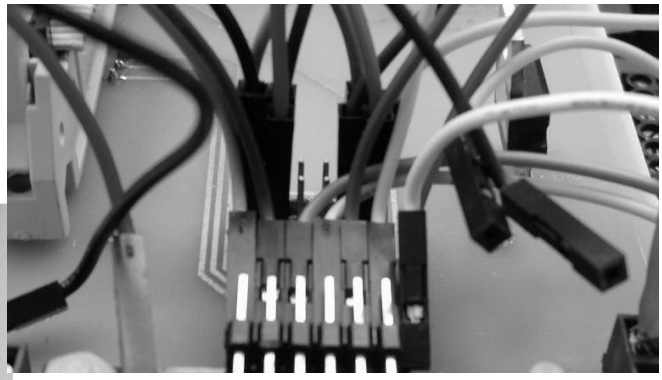
Cette 2ème partie du programme contient toutes les informations et les réglages relatifs aux moteurs et encodeurs.

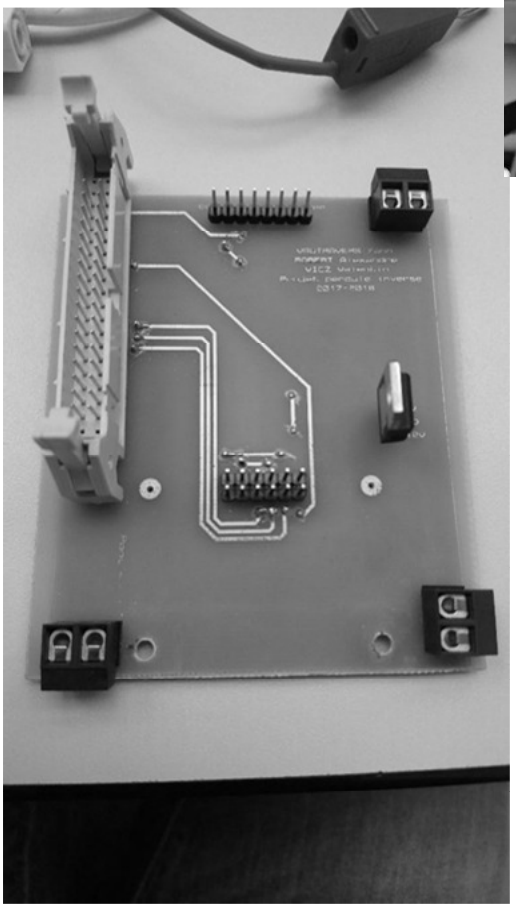
Il y a également des graphes de données correspondants au correcteur PID.

IV. Réalisation de la carte électronique pour la connectique

Voici la première carte test que nous avons réalisé pour notre projet. Elle a été pensée pour être câblée facilement ce qui permet de brancher dans l'ordre que l'on veut pour tester nos programmes.

Exemple de câblage de test





Un modèle final était en cours de conception mais suite à des problèmes de connexions, celle-ci n'a pas eu le temps d'être terminée. Par rapport à la carte de test, plusieurs modifications ont été apportées dont les connecteurs pont en H, un nouveau connecteur pour le port C et trois nouveaux condensateurs (Voir Annexe 6).

Pour ces cartes plusieurs composants sous ISIS et packages sous ARES ont été réalisés car les composants utilisés n'apparaissaient pas dans les librairies et n'avaient pas d'empreintes (Voir Annexe 7).

V. Conclusion

Le projet que nous avons choisi est le développement d'un pendule inverse sous le logiciel LabVIEW. Lors de ce projet nous devions choisir un gyroscope/accéléromètre mais aussi deux moteurs et leur pont en H , ensuite faire l'acquisition de différents signaux ainsi que la génération de tension qui permettent de donner les consignes de commande moteur et la mise en place d'un correcteur.

Lors de ce projet nous avons choisi un moteur avec codeur absolu, dont nous avons réussi à faire l'acquisition et la génération. Pour optimiser l'ergonomie, nous avons réalisé une carte électronique permettant de regrouper toutes les connectiques de notre projet.

Nous avons rencontré des difficultés lors de ce projet notamment dans la programmation LabView mais notre tuteur a su nous mettre dans la bonne voie à plusieurs reprises. Nous avons aussi rencontré quelques difficultés que nous n'avons pas su totalement résoudre notamment la mise en place du correcteur PID et donc le maintien en équilibre du pendule.

Ce projet est intéressant car il touche à beaucoup de domaines : électronique, électronique de puissance, programmation, capteurs. Et il nous a permis de mettre nos connaissances en pratique et de mettre en avant notre esprit d'initiative. Il nous a aussi permis de travailler en équipe et d'être plus autonomes.