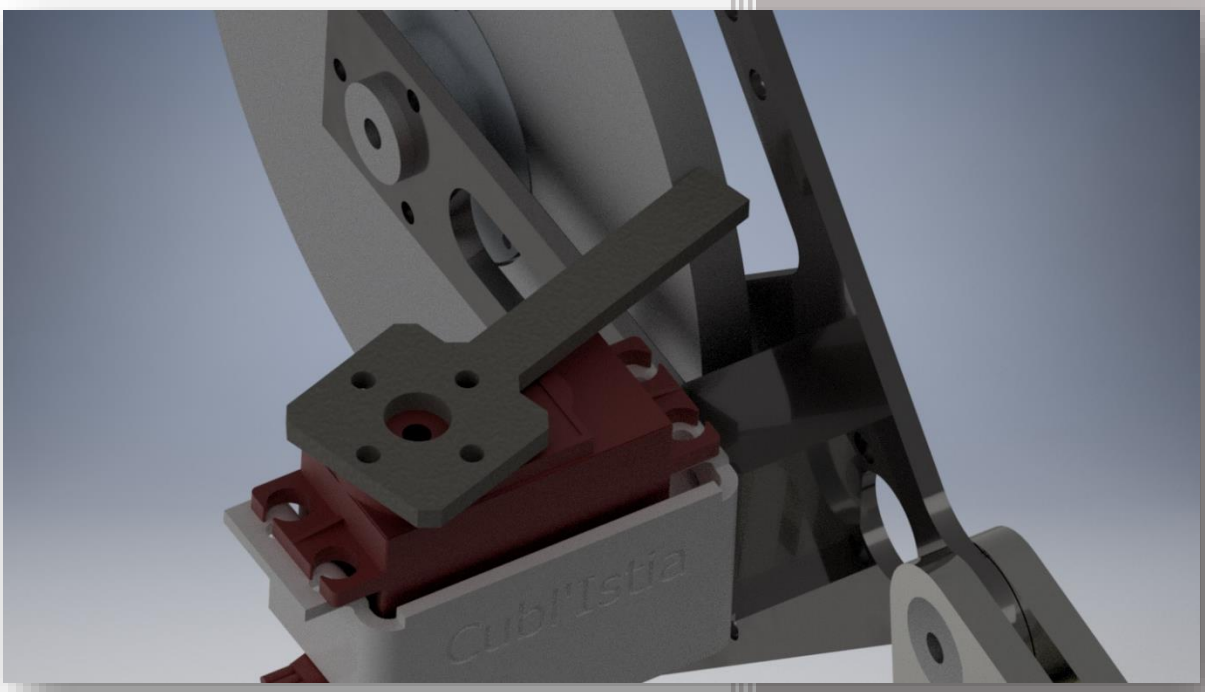
2016
2017

STABILISATION PAR EFFET GYROSCOPIQUE



Guillaume RECOLIN-BLARDON
Quentin BOCHE
Benjamin TEXIER

Mécatronique club
ISTIA (Université d'Angers)

REMERCIEMENTS

REMERCIEMENTS

Tout d'abord, nous remercions Monsieur Sébastien LAGRANGE, notre tuteur pédagogique, pour nous avoir encadrés lors de notre projet.

Nous souhaitons aussi remercier Monsieur Franck MERCIER pour son aide lors des différentes parties de notre projet. Nous avons apprécié sa disponibilité et ses conseils.

Puis, nous remercions également Monsieur Hassan BOULJROUFI, pour son aide et nous avoir prêté plusieurs fois des outils pour la réalisation des tâches.

Enfin, nous exprimons nos remerciements à tous ceux qui nous ont encouragés et aidés dans notre réflexion.

SOMMAIRE

PRESENTATION DE L'EQUIPE

QUI SOMMES-NOUS ?

Actuellement en deuxième année du cycle d'ingénieur à l'ISTIA, école d'ingénieurs de l'Université d'Angers, nous sommes en spécialité « Automatique et Génie Informatique ».

Notre équipe est composée de trois personnes venues de parcours divers :

Guillaume Recolin-Blardon :

Paces (Tours, 37)

Pass'Med, EI3 AGI (ISTIA, Université d'Angers, 49)

Quentin Boche :

BAC STI Génie électronique (Lycée Chevrollier d'Angers 49)

DUT Génie Electrique et Informatique Industrielle (IUT d'Angers 49)

Pass'Med, EI3 AGI (ISTIA, Université d'Angers, 49)

Benjamin Texier:

BAC STI Génie électronique (Lycée Leonard de Vinci, 85)

DUT Génie Industriel et Maintenance (IUT de Saint-Nazaire, 44)

Pass'Med, EI3 AGI (ISTIA, Université d'Angers, 49)

NOS ATTENTES ?

En effectuant ce projet, nous avions plusieurs attentes :

Premièrement, nous n'avons jamais eu la chance d'approfondir l'effet gyroscopique durant nos études. Ce projet était donc l'occasion de comprendre comment exploiter le principe de la conservation du moment angulaire.

Deuxièmement, nous avions déjà entendu parlé du Cubli 3D. Ce projet était pour nous l'occasion de comprendre comment le système fonctionnait et peut-être même la chance de réussir à reproduire ce système ou du moins, la version 2D.

Pour finir, la possibilité de mettre en pratique beaucoup de matières vues/étudiées durant notre parcours scolaire (microcontrôleur, modélisation et simulation, modélisation 3D, électronique, électricité, mécanique). C'était pour nous, un des projets les plus complet et varié parmi les différents projets de la liste.

PRESENTATION DU PROJET

STABILISATION GYROSCOPIQUE

DEFINITION

L'origine étymologique du mot : « qui observe la rotation », nous renseigne sur ce qu'est la gyroscopie. En effet un gyroscope est basé sur le principe de la conservation angulaire : c'est un appareil qui donne la position angulaire sur un, deux ou trois axes par rapport à un référentiel inerte.

AERONAUTIQUE

En aviation les gyroscopes servent à conserver le cap et à d'autres mesures qui permettent le guidage de l'appareil.

De même les stations spatiales sont équipées de gyroscopes afin de contrôler leur attitude lors des trajectoires autour de la terre.

Gyroscope de satellite

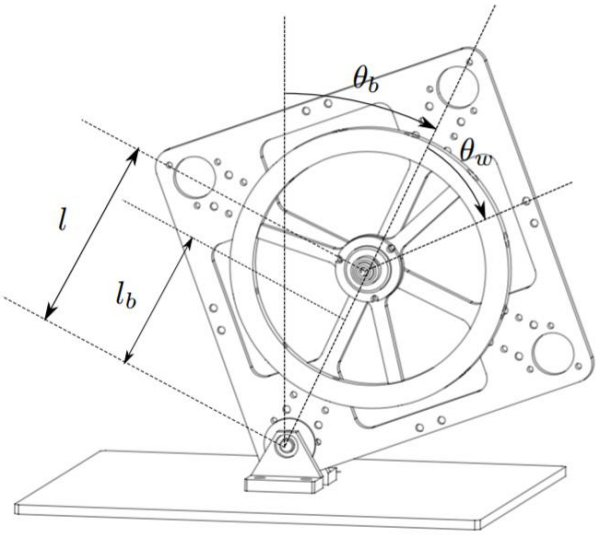
Gyroscope pour reflexe

STABILISATION AU QUOTIDIEN

La gyroscopie arrive aujourd'hui dans le quotidien avec par exemple les stabilisateurs pour smartphones et autres cameras afin d'avoir des images plus nettes. On retrouve également des gyroscopes sur les smartphones ou bien les manettes de consoles pour les jeux vidéo. Tout cela sert à connaître l'orientation de ces objets par rapport à la Terre.

CUBLI 2D

Aussi appelé « One Dimensional Cubli » en anglais, c'est la première étape avant de passer à un Cubli sur 3 axes dont nous parleront plus tard. Ce Cubli 2D constitue donc notre projet, c'est pourquoi nous allons le décrire et en parler un peu. Pour commencer voici un schéma qui illustre la maquette de départ de notre projet :



« One Dimensional Cubli »

Nous avons 3 parties sur cette maquette :

- Un socle qui permet au tout de rester à plat au sol,
- Sur ce socle est fixé une plaque de forme carrée qui peut pivoter selon l'axe θb (+/- 45°),
- Au centre de cette plaque est fixé un disque qui peut tourner selon un angle θw (360°).

En plus de ces éléments nous avons une carte moteur, une carte Arduino et une centrale inertielle. Tous ces éléments sont là pour commander le Cubli afin de le stabiliser. En effet en faisant tourner le disque à une certaine vitesse dans un sens ou dans l'autre on doit être capable grâce à l'inertie emmagasinée de faire pivoter la plaque carrée et de la faire tenir en équilibre sans qu'elle repose au sol.

Ce Cubli à une dimension a pour but de vérifier la faisabilité d'un Cubli en 3D dont nous allons parler juste après. Il nous permet de développer les algorithmes de contrôle pour ce Cubli 3D.

La maquette de départ qui nous a été fournie au début du projet comprenait tous les éléments cités plus haut. Cependant nous en reparlerons plus en détail par la suite mais nous avons dû rajouter un élément constitué d'un servomoteur. Ce servomoteur avec fixé dessus une sorte de clé nous a servi pour freiner plus rapidement le disque au moment de le faire tourner dans l'autre sens.

CUBLI 3D

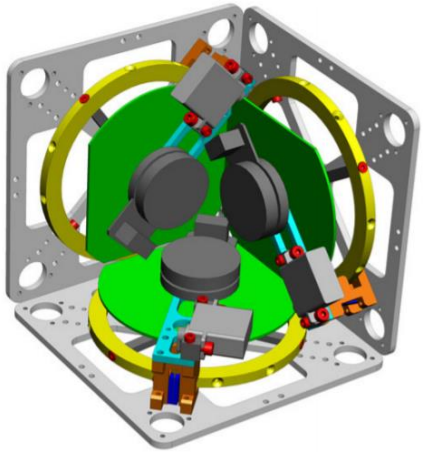
Le Cubli 3D est donc la suite logique du Cubli 2D car il reprend le principe de stabilisation gyroscopique mais cette fois-ci pour stabiliser un cube entier et non une face d'un cube. C'est pour cela que nous allons en parler brièvement.

Cubli 3D en équilibre

Le but de ce Cubli est de rester en équilibre sur un des coins comme sur la photo ci-dessus. Certains modèles sont également programmés pour se déplacer.

Ce modèle est donc constitué de 3 « Cubli 2D » en quelques sortes. Ce qui lui permet de se mettre en équilibre selon 3 axes et donc sur un sommet.

Ce Cubli a été conçu par l'école polytechnique de Zurich et a été inspiré des systèmes qui permettent aux navettes spatiales et satellites d'ajuster leur attitude.



Cubli 3D vue de l'intérieur

TRAVAIL REALISE

LOGICIELS ET TECHNOLOGIES UTILISES

Nous avons utilisé beaucoup de logiciels pour la réalisation de notre projet. Les logiciels peuvent être décomposés en plusieurs catégories :

LES LOGICIELS DE MODELISATION 3D

Nous avons utilisé 3 logiciels de modélisation et visualisation 3D pour notre projet :

- SolidWorks : Déjà installé sur les PC de l'ISTIA.
- Autodesk Inventor : Pour un meilleur rendu 3D. Plus complet que SolidWorks.
- 3D Builder : Pour modifier l'unité de mesures des pièces sous format STL afin d'éviter les conflits entre les différents logiciels et outils de réalisation 3D (imprimante 3D, Charly Robot).

PROGRAMMATION DES DIFFERENTES CARTES

Nous avons utilisé deux logiciels pour la programmation des cartes.

- Atmel Studio : pour la programmation de la carte moteur.
- Arduino : pour celle de notre carte Arduino et son extension atMega2560.

CREATION DE LA CARTE

Nous avons réalisé une carte pour convertir la tension de 5 à 3 volts. Pour cela, nous avons utilisé le logiciel : EAGLE (CadSoft).

PARTAGE DES DONNEES

Pour le partage du code, des images, des fichiers 3D, nous avons utilisé

- Atlassian Bitbucket : Partage du code en développement. Code non Open Source.
- Github : Partage du code final. Pour le code Open Source.
- Git & GitKraken (GIT UI) pour le versionning.
- Microsoft OneDrive : pour les fichiers image et traitement de texte.

GESTION DE PROJET

Pour le suivi de projet, nous avons utilisé :

- Excel : pour le Gantt et Scrum.
- Atlassian Trello : pour créer et gérer nos tâches.

GESTION DE PROJET

Concernant la gestion de projet nous nous sommes tout de suite organisés et cela a été assez simple de s'accorder sur la façon de travailler car nous avions déjà eu des projets en commun. Même si une équipe de 3 personnes peut s'auto-gérer c'est plus simple avec un chef de projet, c'est pourquoi Benjamin s'est occupé de leader le groupe sur certains points.

Avant même de commencer le projet nous avons créé une discussion sur Facebook afin de pouvoir communiquer entre nous et nous échanger rapidement les infos. Cela paraît anodin mais c'est beaucoup plus rapide que par mail quand on veut juste échanger de petites idées rapidement.

Nous avons également réalisé un Gantt afin de planifier les tâches sur les 10 jours de projet que nous avions d'assignés.

Une fois ce travail réalisé nous avons mis en place un drive avec « OneDrive » afin que chacun ait accès à tous les éléments du projet puis pour la partie programmation Arduino et site web nous avons mis en place un Git.

Concernant le phasage du projet, le Gantt est en annexe mais nous allons rapidement détailler les grandes parties de notre projet :

- **Mise en place de la méthode de travail** dont nous avons parlé juste avant,
- **Etude et intégration de la centrale inertielle à la maquette** : cette partie a pris beaucoup de temps et n'a finalement pas aboutie mais nous a permis de beaucoup apprendre,
- **Communication entre la carte Arduino et le moteur** : partie important du projet qui a été rapidement réalisée pour passer à la suite,
- **Ajout d'un frein (servomoteur)** car nous nous sommes rendu compte assez vite que le moteur de la maquette ne suffirait pas à faire basculer le cadre en métal et le disque,
- **Réalisation du poster, du rapport et du site web**.

MODELISATION 3D

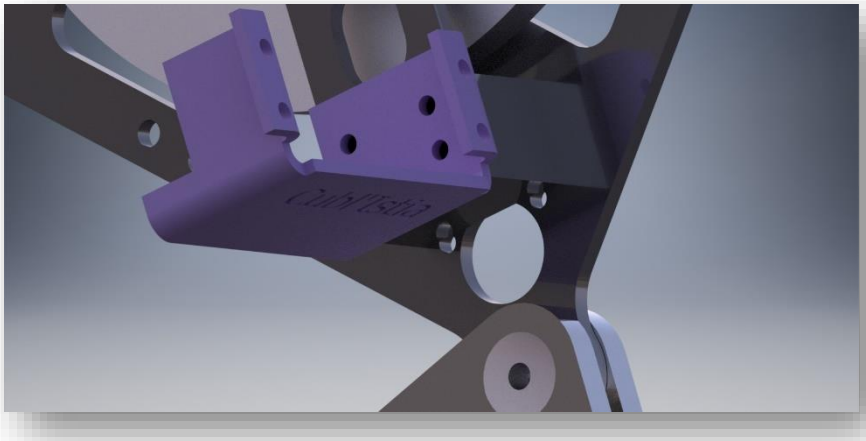
CREATION D'UN FREIN

Le freinage moteur n'étant pas suffisamment puissant pour permettre au système de s'élever, nous avons eu l'idée de rajouter un frein supplémentaire pour augmenter la force.
Pour la modélisation du frein, nous nous sommes basé sur l'assemblage que nous a fourni M. MERCIER.

POSITIONNEMENT DU SERVOMOTEUR

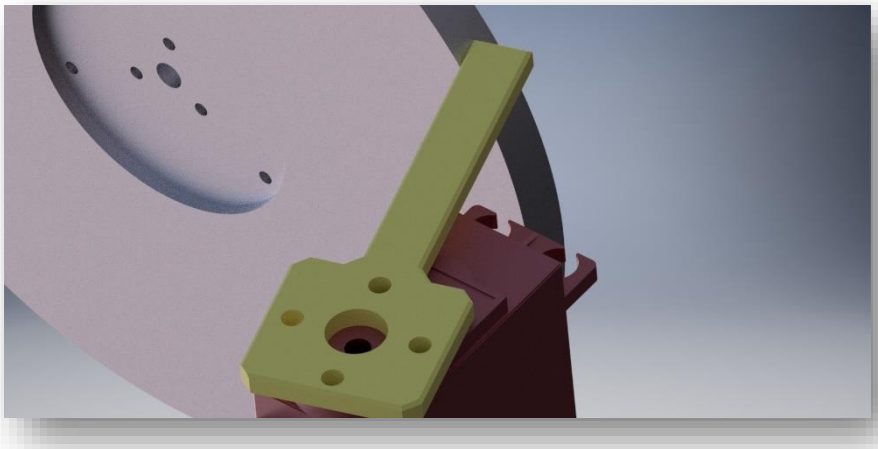
Nous avons passé beaucoup de temps à trouver un emplacement pour le servomoteur. En effet, celui-ci devait être au centre pour ne pas jouer sur le centre de gravité et suffisamment proche du disque pour pouvoir le toucher facilement. Nous avons pensé que le porte moteur était une bonne position.
Nous avons aussi essayé de réaliser un boitier solide pour supporter les chocs et les vibrations.

Voici le modèle final à son emplacement :



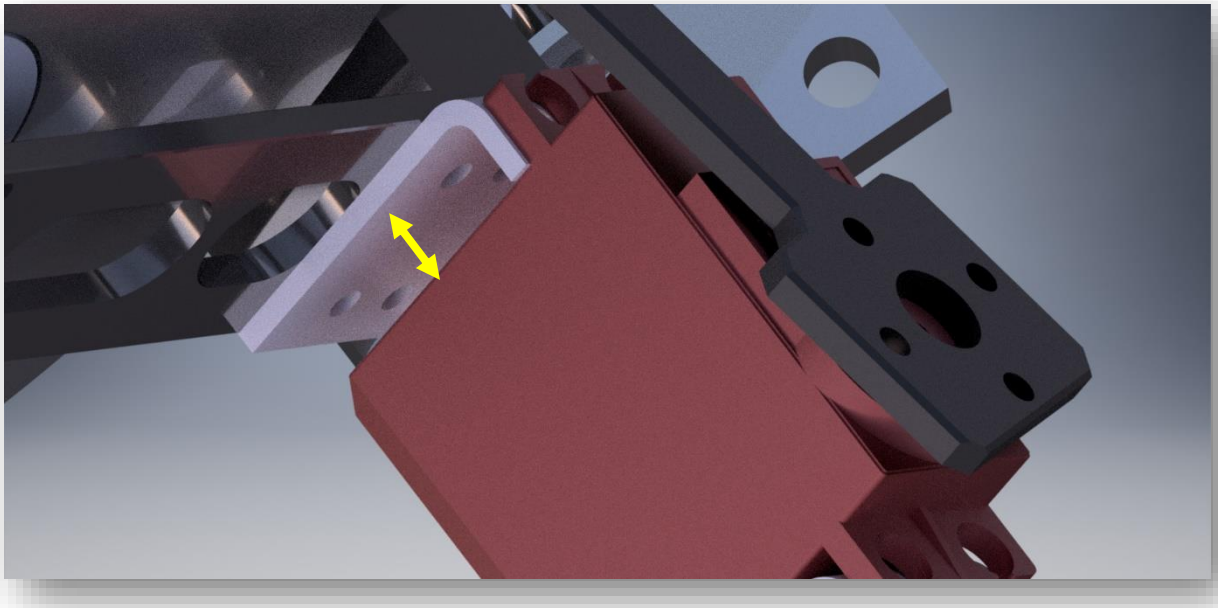
BLOCAGE DU DISQUE

Pour le blocage du disque, nous avons décidé de réaliser un bras en métal afin de frotter contre le disque. L'inconvénient de ce système est que seul une petite partie du bras est en contact avec la roue. Celui-ci sera usinée à partir d'une plaque Sandwich Alu brossé de 3mm d'épaisseur (Alupanel-Dipond). Cette matière a l'avantage d'être très résistent aux chocs et se scie, se plie, de perce, se colle et s'usine très bien.



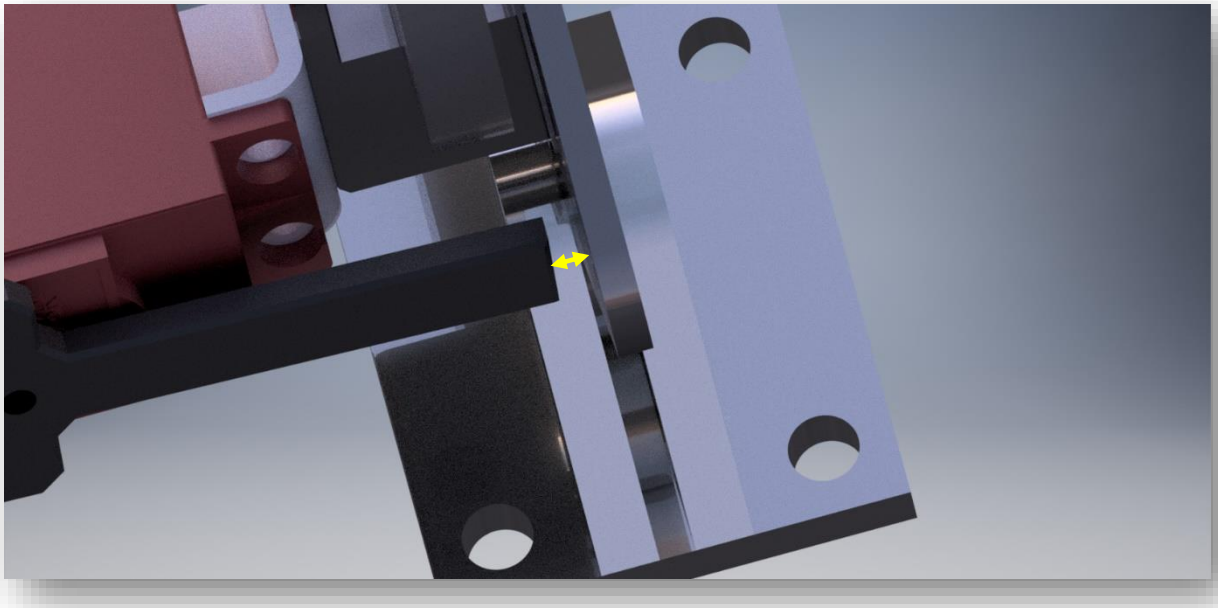
ASSEMBLAGE

Voici ci-dessous une visualisation du montage souhaité :



Plus de visualisation en annexe

Comme on peut le voir, nous avons laissé un peu de place entre le servomoteur et le boitier de fixation pour laisser passer la tête des vis de maintien (7mm d'après nos plans).



Comme on peut le voir sur le modèle ci-dessous, il n'y a pas beaucoup d'espace entre le bras et le châssis (2.3mm). Nous savons ainsi que la précision lors du montage sera très importante et qu'il faudra essayer de limiter au maximum la rotation du bras en cas de jeu.

PROGRAMMATION ARDUINO

INITIALISATION DU BUS CAN

```
// Test BUS CAN
Serial.begin(115200);

while (CAN_OK != CAN.begin(CAN_500KBPS))
{
        Serial.println("CAN BUS Shield init fail");
        Serial.println(" Init CAN BUS Shield again");
        delay(100);
}
Serial.println("CAN BUS Shield init ok!");
```

ENVOI DES INFORMATIONS VERS LA CARTE MOTEUR

La carte moteur permet de faire tourner la roue. Nous communiquons que 4 fois avec cette carte :

INITIALISATION

Pour l'initialisation, nous envoyons à la carte moteur le type de moteur que l'on souhaite sur l'id d'initialisation : 0x04.

```
byte _init[1] = {0x01};   // Attribut
CAN.sendMsgBuf(0x04,0, 1, _init);
```

AVANCER ET CHANGEMENT DE SENS

Pour avancer, nous envoyons un message de 3 bytes (1 byte pour le sens et la vitesse est codé sur deux bytes). Ici, pour les deux exemples, seul le sens de rotation est inversé.

```
byte _vitesse[3] = {0x00, 0x04, 0x20}; // Attribut
CAN.sendMsgBuf(0x03,0, 3, _ vitesse);


byte _inverse[3] = {0x01, 0x04, 0x20}; // Attribut
CAN.sendMsgBuf(0x03,0, 3, _ inverse);
```

ARRET / STOP

Pour dire au moteur que l'on souhaite s'arrêter, nous mettons simplement la vitesse à zéro.

```
byte _stop[3] = {0x00, 0x00, 0x00}; // Attribut
CAN.sendMsgBuf(0x03,0, 3, _stop);
```

ENVOI DES INFORMATIONS VERS LE SERVOMOTEUR

La communication avec le servomoteur est utile simplement pour le frein.

INITIALISATION

Nous initialisons le servomoteur en indiquant simplement le port de commande et nous lui indiquons sa position initiale :

```
Servo myservo; // Attribut

myservo.attach(9);
myservo.writeMicroseconds(1500);
```

FAIRE TOURNER LE SERVOMOTEUR

```
FreinServo(1125) ;

void FreinServo(int pos)
{
        myservo.writeMicroseconds(pos); // position de freinage
        delay(200);
        myservo.writeMicroseconds(1500); // position initiale
        delay(200);
}
```

RECUPERATION DES INFORMATIONS DE LA CARTE GYROSCOPIQUE

Cette partie est la plus délicate. Nous n'avons pour le moment, pas réussi à l'effectuer. Voici cependant le code que nous pensons bon. Pour cela, nous pensons utiliser la bibliothèque « Wire ».

INITIALISATION

On teste la connexion I2C :

RECUPERER LES DONNEES

Pour la récupération des données, nous pensions utiliser le code donné sur le site suivant :

https://knowledge.parcours-performance.com/ajouter-gyroscope-a-robot-arduino/

Qui utilise lui-aussi cette même bibliothèque afin de récupérer les valeurs d'un capteur gyroscopique pour une carte Arduino

ELECTRONIQUE

MATERIEL

Pour ce projet, nous avions à disposition une centrale inertielle (INEMO-M1) et une carte Arduino atMEGA2560. Pour commander et utiliser le capteur. Le but de ce capteur est de nous fournir des informations d'accélération et gyroscopique sur les 3 axes.

Figure 1: INEMO-M1

Pour communiquer avec cette centrale, il nous a été soumis utiliser le canal de communication I²C. I²C est un bus informatique qui permet une communication facile entre un microcontrôleur (carte Arduino) et un circuit électronique (centrale inertielle). I²C est un bus série synchrone bidirectionnel half-duplex, où plusieurs équipements, maîtres où esclaves, peuvent être connectés au bus. La connexion à ce bus ce bus est réalisée par l'intermédiaire de deux files :

-SDA (Serial Data Line) qui représente la ligne de donnée

-SCL (Serial Data Line) qui représente l'horloge de synchronisation bidirectionnelle

Les deux bus sont tirés au Vcc via des résistances de pull-up. Le bus I²C possède 7 bits d'adressage ce qui permet de connecter 128 périphériques et 1 bit R/W pour indiquer l'émetteur et le récepteur.

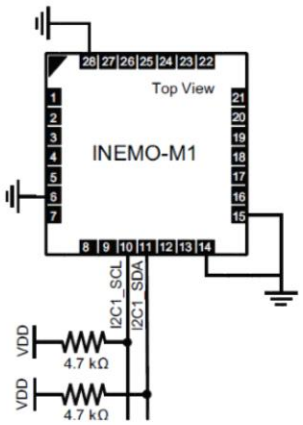
La vitesse typique de transmission du bus est de 100kbit/s. Cependant, on peut utiliser le bus en mode « fast-mode ». Dans ce cas le bus peut aller jusqu'à une vitesse de transmission de 400kbit/s.

CABLAGE

La centrale inertielle est capable de communiquer via I²C. Grace aux pin 10 & 11 respectivement I2C1_SCl & I2C1_SDA. Nous avons utilisé ces deux branches.

Nous avons connecté le GND à la pin 14, en vérifiant toutefois, grâce à un ohmmètre que tous les GND du capteur était un reliées. Enfin, nous avons choisi de mettre le VCC à la pin 1. Le capteur est alimenté entre 2.4V et 3.6V.

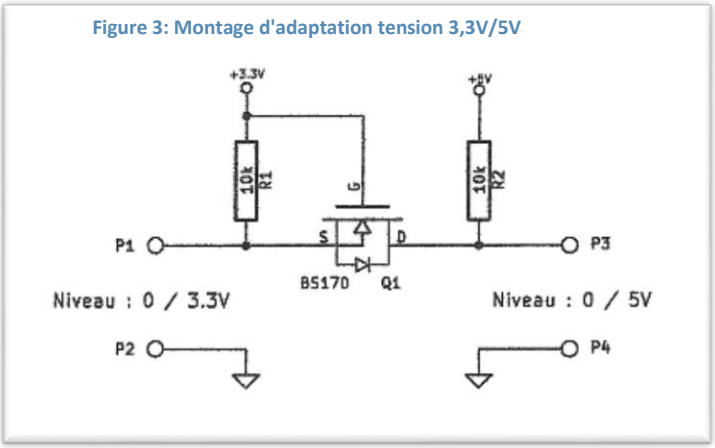
Figure 2 : INEMO-M1 configuration standard



La carte Arduino possède également des broches capables de communiquer en I²C. Il y a deux pins dédiés à cet effet se sont les pin 21 (SCL) & 22 (SDA). Ces entrées/sorties sont contenues dans le port D à l'adresse 0 et 1.

Le carte Arduino et le central inertielle ne fonctionnent pas sous la tension d'alimentation. Pour cela, nous avons dû mettre un dispositif entre les deux périphériques comme ci-dessous.

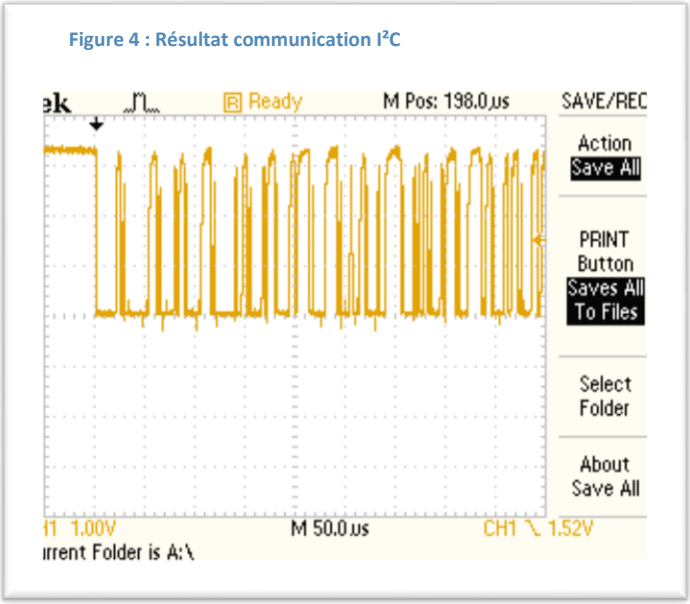
Figure 3: Montage d'adaptation tension 3,3V/5V



Le montage présenté est composé de deux résistances de pull-up et d'un transistor FET à canal N. Le montage a été doublé pour chacun des deux canaux (SCL & SDA). Nous avons réalisé ces montages sur une platine d'essai.

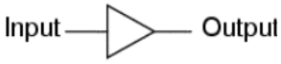
Nous développerons dans le chapitre suivant le programme de test pour la communication I²C. Le résultat de ce test fût plutôt médiocre et le test programme test ne reconnaissait pas la centrale. En voici, le résultat recueillit à l'oscilloscope.

Figure 4 : Résultat communication I²C



On peut voir que les créneaux ne sont pas très nets ce qui doit poser des problèmes de communication. Mr Hassan nous a suggéré de mettre deux portes inverseurs en série afin de jouer le rôle d'hystérésis. Ces portes sont appelées « buffer gate ».

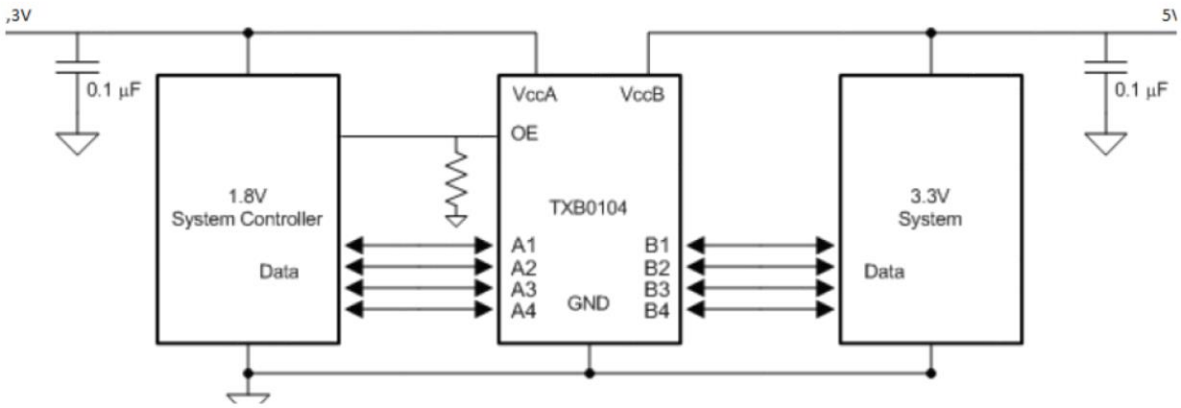
Figure 5 : Buffer gate



Input	Output
0	0
1	1

Nous n'avons pas opté pour cette solution. En effets, Mr Mercier nous a proposé de réaliser une carte en prenant un convertisseur 3,3V/5V.

Figure 6 : Schéma montage du convertisseur



Cette solution pouvant résoudre les problèmes électriques comme les faux contactes. Nous avons réalisé une carte électronique sur Eagle. Nous avons rajouter des résistances de pull-up au schéma électrique ci-dessus afin de répondre aux exigences de protocole I^2C.

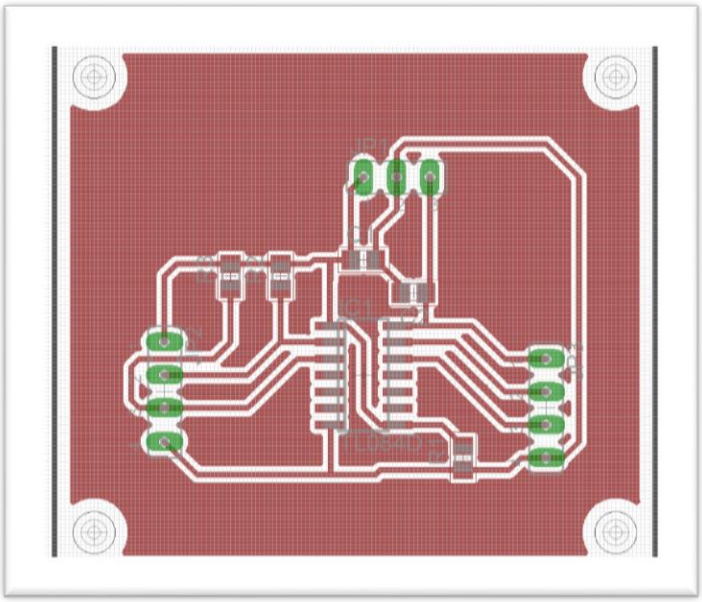
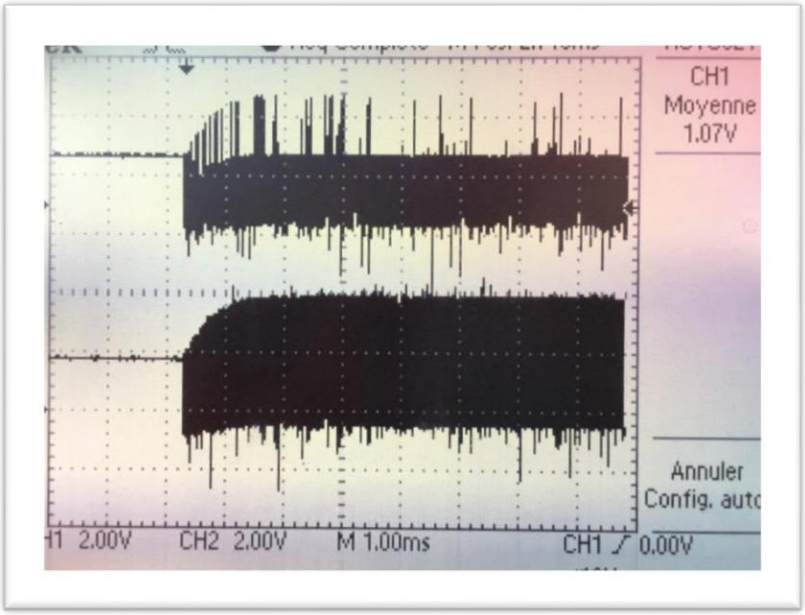


Figure 7 : Carte du convertisseur de tension

Les résultats obtenus à la suite de cette carte n'ont pas été ceux escompté. En effet, les tensions semblent varier au-dessus des tensions d'une manière étrange. Nous n'avons pas trouvé le temps de résoudre ce problème.

Figure 8 : Résultat de la carte convertisseur

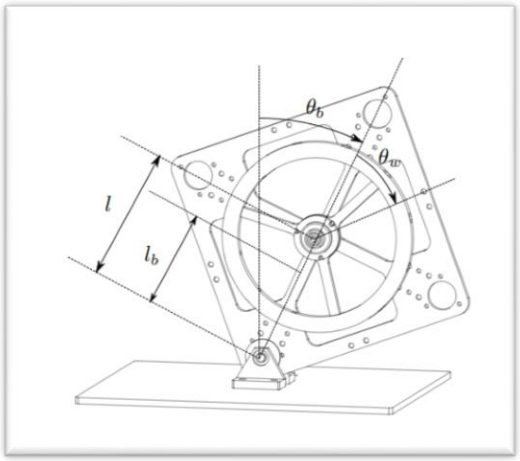


PROGRAMME DE TEST UTILISE

Les deux programmes utilisés sont les programmes fournis par Arduino pour tester la communication sur le bus i²C. Il suffit de configurer les ports souhaités pour le bus soit dans notre cas les pin 0 & 1 du port D, de mettre le fast-mode à 0. Ils existent deux programmes fournis « I2CScanSoftI2C » & « I2CScanSoftWire », ses deux programmes ont pas la même structure il initialise la communication et ensuite énumère tous les adresses du bus.

MODELISATION ET SIMULATION

Le modèle suivi est celui de l'étude intitulé « The Cubli : A Cube that can Jump Up and Balance » par Mohanarajah Gajamohan, Michael Merz, Igor Thommen et Raffaello D'Andre. Nous allons expliquer l'étude qui nous a servi de base.



Les éléments importants du cubli à deux dimensions sont *l* qui est la longueur entre l'axe du moteur et le pivot au sol, I_b est le moment d'inertie du corps de cubli, I_w est le moment d'inertie de la roue, C_w est le coefficient dynamique de friction de la roue, C_b est le coefficient dynamique de friction du corps de cubli.

La conservation du moment angulaire durant l'impact est décrite par l'équation ci-dessous.

$$I_w\omega_w = (I_w + I_b + m_w l^2)\omega_b$$

L'énergie dégagée est décrite par l'équation ci-dessous :

$$1/2\ (I_w + I_b + m_w l^2)\omega^2_b = (m_b l_b + m_w l)g(1 - 1/\sqrt{2}).$$

Enfin, l'étude utilise une équation de type $\dot{x}(t) = Ax(t) + Bu(t)$ afin de décrire le système dans sa globalité. X étant composé de (θ_b, $\dot{\theta}_b$, $\dot{\theta}_w$) décrivant respectivement l'angle par rapport à la verticale du corps de cubli, la vitesse du corps de cubli et la vitesse de la roue.

$$A := \begin{bmatrix} 0 & 1 & 0 \\ \frac{(m_b l_b + m_w l)g}{I_b + m_w l^2} & -\frac{C_b}{I_b + m_w l^2} & \frac{C_w}{I_b + m_w l^2} \\ -\frac{(m_b l_b + m_w l)g}{I_b + m_w l^2} & \frac{C_b}{I_b + m_w l^2} & -\frac{C_w((I_b + I_w + m_w l^2)}{I_w(I_b + m_w l^2)} \end{bmatrix},$$

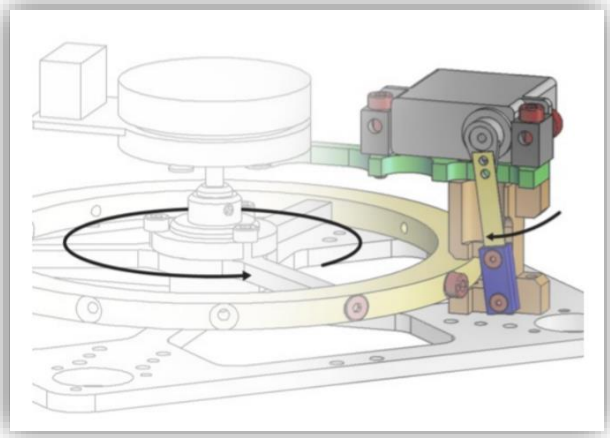
$$\text{and} \quad B := \begin{bmatrix} 0 \\ -\frac{K_m}{I_b + m_w l^2} \\ \frac{K_m(I_b + I_w + m_w l^2)}{I_w(I_b + m_w l^2)} \end{bmatrix}.$$

CONCLUSION

NOS DIFFICULTES

a) Frein :

La première difficulté que nous avons rencontrée a été lorsque nous avons réussi à faire fonctionner le moteur car nous nous sommes rendu compte que la partie à faire tenir en équilibre était trop lourde pour simplement basculer de l'autre côté. Jusque-là nous pensions qu'en faisant changer le moteur de sens, l'inertie serait suffisante pour que le tout bascule. Seulement cela n'a pas fonctionné donc nous avons dû pensé à une autre solution. Plutôt que de modifier la maquette la solution la plus simple a été de rajouter un système de freinage. En nous inspirant d'un projet similaire nous avons élaboré un système avec un servomoteur qui actionne une tige qui elle-même en frottant sur la roue en mouvement permet de freiner rapidement la roue pour qu'ensuite elle se mette à tourner dans l'autre sens. Il a donc fallu fixer le servomoteur, fabriquer la tige et programmer le tout.



Exemple de frein dont nous nous sommes inspirés

b) Centrale inertielle :

Les principaux de problèmes ont été électrique. En effet, entre les faux contacts, les mauvaises soudures, etc… Les résultats obtenus deviennent très aléatoires. Il nous est arrivé de reconnaitre la centrale inertielle à 10 ou 20 adresses en même temps et l'instant d'après ne pas le détecter. Le choix de faire une carte électronique semblait être la bonne solution pour résoudre ces problèmes. Toutefois, il semblerait qu'il n'y est pas que ce problème. La réalisation d'une carte électronique est une activité chronophage qui nous a coûté un temps précieux dont on a manqué pour dépanner cette même carte en projet.

TEMPS PASSE SUR LES DIFFERENTES PARTIES

Nous allons maintenant parler de la **répartition temporelle** des tâches de ce projet.

Pour commencer la **découverte du projet**, des outils et la mise en place de la **méthode de travail** nous a pris un certain temps. Nous avons mis 1 journée de projet le temps également de tout installer pour pouvoir travailler dans de bonnes conditions par la suite.

La **partie mécanique** concerne principalement le frein qui a été rajouté par la suite. Benjamin et Guillaume se sont occupé de cette partie qui a consisté à réfléchir sur la méthode de freinage puis à l'implémenter sur la maquette. Toute cette partie nous a pris environ 3 séances.

Ensuite la **partie électronique** a pris du temps à cause de certaines complications. Cette partie concerne uniquement la soudure de la centrale inertielle. C'est Quentin qui s'est chargé de tout ce qui touche à la centrale et ça lui a pris 4 séances pour la partir électronique, le temps également de comprendre le fonctionnement de la centrale.

La **partie programmation** concerne le moteur, le frein et la centrale. Pour commencer la programmation Arduino du moteur a été assez rapide en soi mais il nous a fallu du temps avant de réussir à communiquer avec, nous avons donc mis un peu plus d'une séance. Pour ce qui est du frein le temps de le calibrer avec le moteur la programmation nous a pris une journée de plus. Concernant la programmation de la centrale cette partie n'a pas eu lieu car nous n'avons pas réussi à l'implémenter à la maquette.

Le **livrable** n'a pas été réalisé pendant les séances de projet mais durant nos temps libres sur une période d'une semaine environ par nous 3.

Le design du **site web** a été réalisé par Benjamin en quelques jours en dehors des heures de projet puis nous avons ensuite rajouté le contenu en s'aidant du livrable.

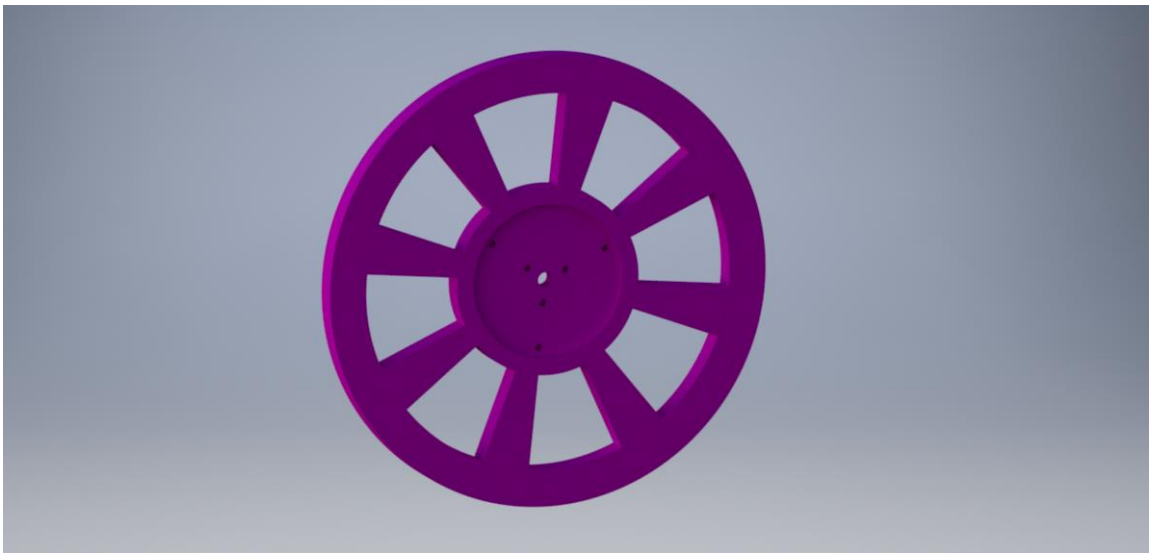
Le **poster** a été réalisé par Guillaume en quelques heures également en dehors des heures de projet.

CE QU'IL RESTE A FAIRE

Suite aux différentes pertes de temps sur les différentes parties, nous n'avons pas eu la chance de finir le projet. Voici ce que nous aurions voulu faire :

MECANIQUE

Nous n'avons pas eu l'occasion de tester le frein pour le moment, mais, nous pensons qu'il faudrait ré-usiner la roue pour réduire son poids. Sa taille resterait la même mais avec des trous. Le visuel serait ainsi le suivant :



Nous pensons qu'il serait même plus intéressant de réduire l'épaisseur des « rayons » afin de minimiser son poids au milieu et augmenter le poids externe ce qui apporterait l'inertie nécessaire pour le décollage de la roue.

RECUPERATION DES INFORMATIONS

Malgré le temps passé sur la récupération des données en I²C du capteur nous n'avons toujours pas réussi à récupérer des informations de la part de celle-ci. Si le projet est continué dans le futur, nous pensons qu'il pourrait être intéressant d'utiliser le capteur gyroscopique « gy-521 », équipé d'un MPU-6050 qui ne coûte vraiment pas cher et qui est équipé d'un capteur gyro ainsi que d'un accéléromètre (ce qui serait suffisant pour notre cas). De plus, ce capteur semble bien plus simple à utiliser et de nombreux tutoriaux sont disponible sur le WEB. De plus, nous aurions sûrement essayé de tester le capteur sur un autre canaux de communication dont il est équipé pour tenter d'établir une communication plus fiable et plus rapide.

PROGRAMMATION

Sans les informations I²C, nous n'avons pas programmer le sens de rotation du moteur en fonction des différentes données pour permettre cette stabilisation.

CE QUE L'ON A RETENU DU PROJET

QUENTIN BOCHE

Le projet était un projet ambitieux avec une partie mécanique importante. Cette partie a été pour moi une grande source de problème mais aussi une source de réflexion. Je pense que tout l'intérêt du projet était de résoudre cette partie.

Nous avons beaucoup appris sur la gestion de projet afin de réaliser un maximum d'essai pour tenter d'atteindre les objectifs du projet.

BENJAMIN TEXIER

J'ai beaucoup apprécié de travailler sur ce sujet. En effet, ce prototype a déjà été réalisé dans des écoles mais avec des composants différents.

La mise en œuvre de ce projet m'a permis de comprendre des différentes phases de l'organisation d'un projet. Ce projet était très complet (mécanique, automatique, modélisation, programmation Arduino …), il était donc important, voire essentiel de se répartir le travail au maximum.

Suite à différents problèmes, nous avons recherché/imaginé des solutions, parfois mauvaise, mais cela nous a fait beaucoup apprendre sur la création d'un prototype. Même si nous n'avons pas eu le temps de remplir nos objectifs (stabilisation du cube), nous avons beaucoup appris sur le sujet de la stabilisation gyroscopique.

GUILLAUME RECOLIN-BLARDON

Une des parties importantes de ce projet a été la recherche car le projet Cubli a déjà été réalisé par d'autres écoles et instituts de recherches. Cependant notre maquette de départ était différente donc nous n'avons pas eu les mêmes enjeux mais nous avons dû passer un certain temps à étudier les projets déjà réalisés pour s'en inspirer. Ce travail de recherche est essentiel dans tout projet et nous a donc vraiment appris des choses.

Concernant l'électronique également, personnellement je n'ai pas eu beaucoup d'occasion pour la pratiquer jusque-là donc ce projet a été une réelle opportunité d'en apprendre plus et de m'exercer dans ce domaine.

Je ne regrette donc rien de ce projet qui a été une très bonne expérience.

DEFINITIONS

ROUE DE REACTION

*« Une **roue de réaction** est un type de volant d'inertie utilisé dans les engins spatiaux pour modifier leur moment angulaire sans consommer de carburant, seulement de l'électricité. »* Wikipédia, *Roue de réaction*

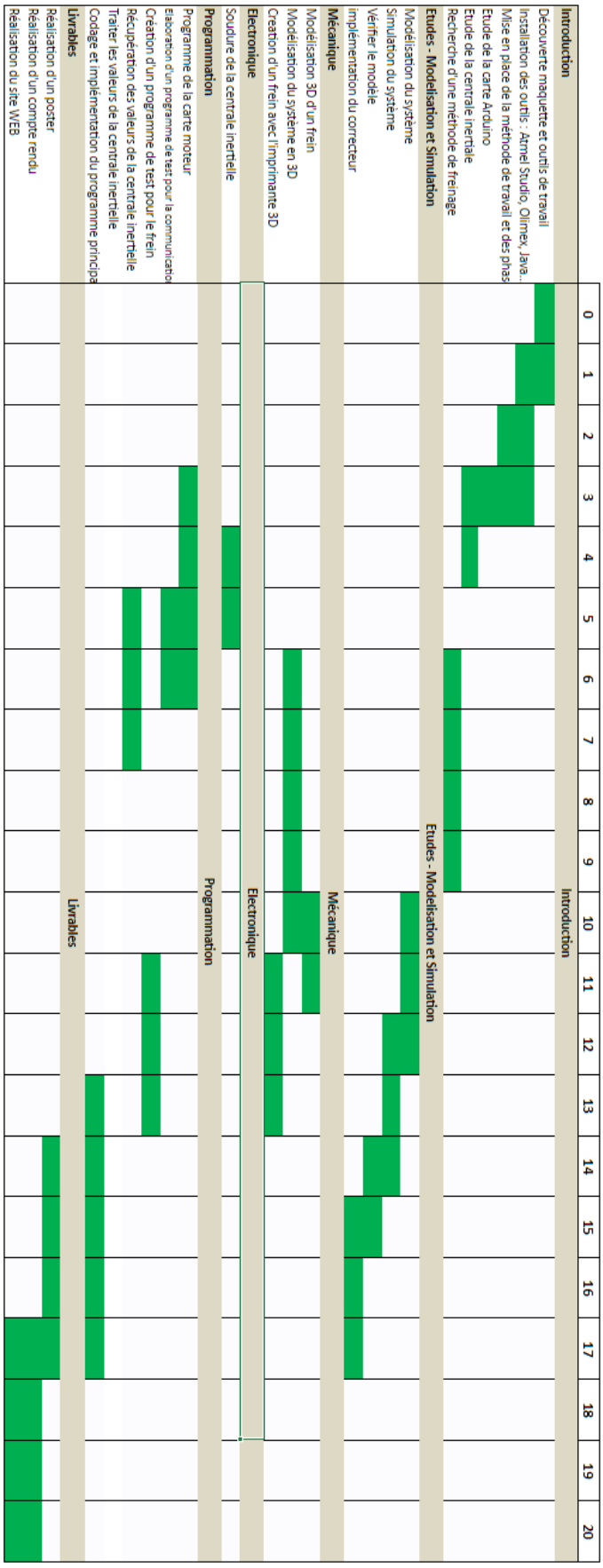
CENTRALE INERTIELLE

*« Une **centrale à inertie** ou **centrale inertielle** est un appareil de navigation de précision comportant des gyroscopes, des capteurs d'accélération et de vitesse angulaire et calculant en temps réel à partir de ces mesures l'évolution du vecteur vitesse et de la position du véhicule à bord duquel il est installé, ainsi que de son attitude (roulis, tangage, cap). Les centrales à inertie sont installées à bord de navires, d'aéronefs, de missiles et de véhicules spatiaux »* techno-science.net

ANNEXES

DIAGRAMME DE GANTT

Etape	Tâches	Personnes	Début	Durée	échéance (En demi journée)	Realisation	Retard
1	**Introduction**	**Tout le monde**					
1,1	Découverte maquette et outils de travail	Tout le monde	0	1	1	1	0
1,2	Installation des outils : Atmel Studio, Olimex, Java...	Tout le monde	1	2	3	3	0
1,3	Mise en place de la méthode de travail et des phases du projet	Tout le monde	2	1	3	3	0
1,4	Etude de la carte Arduino	Tout le monde	3	0	3	3	0
1,5	Etude de la centrale inertiale	Quentin	3	1	4	4	0
1,6	Recherche d'une méthode de freinage	Benjamin et Guillaume	6	3	9	12	3
2	**Etudes - Modelisation et Simulation**	**Quentin**					
2,1	Modélisation du système	Quentin	10	2	12		
2,2	Simulation du système	Quentin	12	2	14		
2,3	Vérifier le modèle	Quentin	14	1	15		
2,4	Implémentation du correcteur	Quentin	15	2	17		
3	**Mécanique**	**Benjamin et Guillaume**					
3,1	Modélisation 3D d'un frein	Benjamin et Guillaume	10	1	11	12	1
3,2	Modélisation du système en 3D	Benjamin et Guillaume	6	4	10	13	3
3,3	Création d'un frein avec l'imprimante 3D	Guillaume	11	2	13	14	1
4	**Electronique**	**Quentin**					
4,1	Soudure de la centrale inertielle	Quentin	4	1	5	18	13
5	**Programmation**	**Tout le monde**					
5,1	Programme de la carte moteur	Guillaume	3	3	6	4	-2
5,2	Elaboration d'un programme de test pour la communication entre moteur et l'Arduino	Benjamin et Guillaume	5	1	6	6	0
5,4	Création d'un programme de test pour le frein	Benjamin	11	2	13	15	2
5,5	Récupération des valeurs de la centrale inertielle	Quentin	5	2	7		
5,6	Traiter les valeurs de la centrale inertielle	Quentin	7	3	10		
5,7	Codage et implémentation du programme principal	Benjamin	13	4	17		
6	**Livrables**	**Tout le monde**					
6,3	Réalisation d'un poster	Guillaume	14	3	17	22	5
6,1	Réalisation d'un compte rendu	Quentin et Guillaume	17	3	20	22	2
6,2	Réalisation du site WEB	Benjamin	17	3	20	22	2


Introduction
Découverte maquette et outils de travail
Installation des outils : Atmel Studio, Olimex, Java...
Mise en place de la méthode de travail et des phas
Etude de la carte Arduino
Etude de la centrale inertiale
Recherche d'une méthode de freinage
Etudes - Modelisation et Simulation
Modélisation du système
Simulation du système
Vérifier le modèle
implémentation du correcteur
Mécanique
Modélisation 3D d'un frein
Modélisation du système en 3D
Creation d'un frein avec l'imprimante 3D
Electronique
Soudure de la centrale inertielle
Programmation
Programme de la carte moteur
Elaboration d'un programme de test pour la communicatio
Création d'un programme de test pour le frein
Récupération des valeurs de la centrale inertielle
Traiter les valeurs de la centrale inertielle
Codage et implémentation du programme principa
Livrables
Réalisation d'un poster
Réalisation d'un compte rendu
Réalisation du site WEB
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

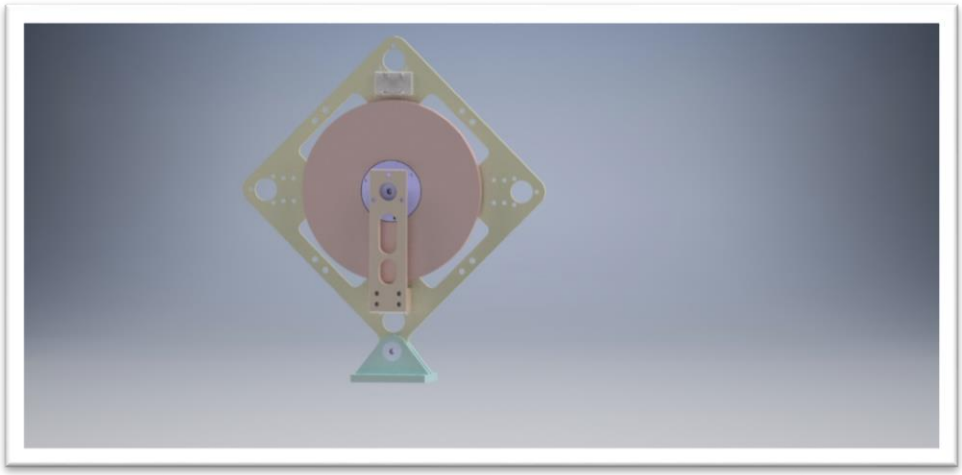
CODE ARDUINO DU PROGRAMME DE TEST – SANS CAPTEUR

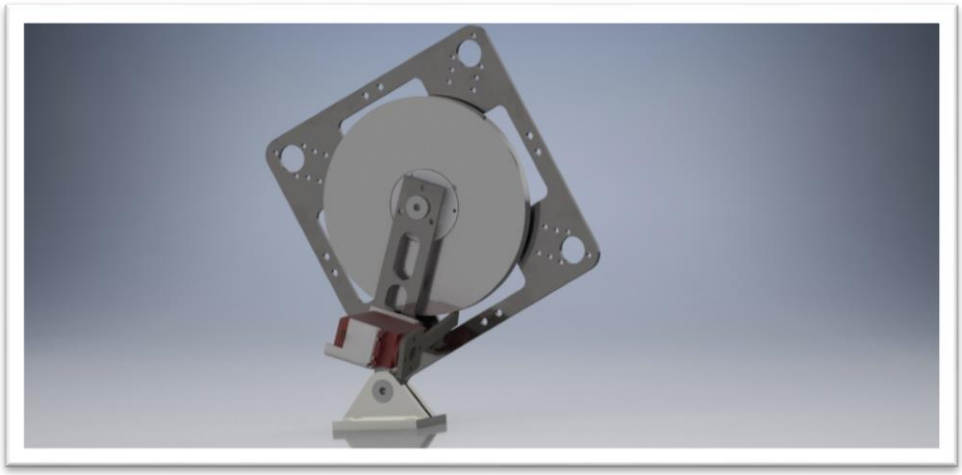
```
#include <SPI.h>
#include "mcp_can.h"
#include <Keyboard.h>
#include <Servo.h>
#include "Keyboard.h"
// Attributs
Servo myservo;  // create servo object to control a servo

const int tailletableau = 8;
int gyro[8];

//pour le moteur
byte _vitesse[3] = {0, 0x06, 0x20};
byte _stop[3] = {0x00, 0x00, 0x00};
byte _inversetest[3] = {0x01, 0x04, 0x20};
const byte _init[1] = {0x01};

// the cs pin of the version after v1.1 is default to D9
// v0.9b and v1.0 is default D10
const int SPI_CS_PIN = 9;
MCP_CAN CAN(SPI_CS_PIN) ;  // Set CS pin


char inChar = ' ';
char extChar = ' ';
int etat=0;
bool first = false;


void setup() {

  // Test BUS CAN
  Serial.begin(115200);

  while (CAN_OK != CAN.begin(CAN_500KBPS))
  {
    Serial.println("CAN BUS Shield init fail");
    Serial.println(" Init CAN BUS Shield again");
    delay(100);
  }
  Serial.println("CAN BUS Shield init ok!");

  myservo.writeMicroseconds(1500);//sert a initialiser la position du servomoteur

     // Servo
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  initialisation();  //appel de la focntion qui init le moteur

  //inertial(gyro, tailletableau);
  //affichetableau(gyro, tailletableau);
}

void loop()
  first = false;
  if(Serial.available()>0)
  {
   extChar=inChar;
   inChar=Serial.read();
   if(extChar != inChar)first = true;
```

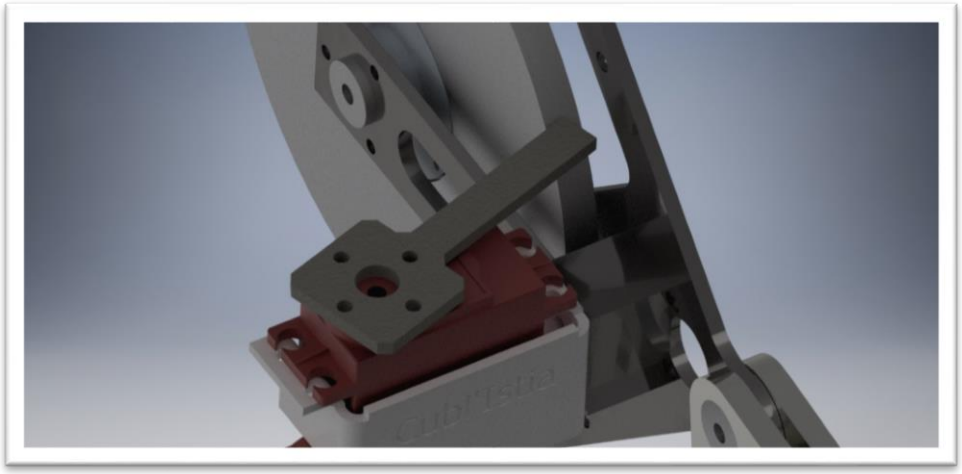
```
  }
  switch(inChar)
  {
   case 's':
    if(first)
    {
     arret();
     inChar='c';
     _vitesse[0]=1;
     avance();
    }
   break;
   case 'r':
    if(first)
    {
     _vitesse[0]=0;
     avance();
    }
   break;
   case 'c':
    if(first)
    {
     _vitesse[0]=1;
     avance();
    }
    delay(500);
   break;
   case 'k':
    if(first) arret;
   break
  }
}

/*
  *  Programme qui actionne le servo pour freiner
  *  @arg1 : position du servo pour freiner la roue
  *  return void;
*/
void FreinServo(int pos)
{

  myservo.writeMicroseconds(pos);
  delay(200);
  myservo.writeMicroseconds(1500);
  delay(200);
}

/*
  *  Programme qui fait tourner la roue
  *  return void;
*/
void avance()
{
  Serial.println("Changement de vitesse");
  CAN.sendMsgBuf(0x03,0, 3, _vitesse);
  Serial.println("Fin changement de vitesse");
}

/*
  *  Fonction pour faire freiner la roue
  *  utilise la fonction FreinServo pour freiner plus vite
  *  return void;
  */
```

```
void arret()
{

  //Arret Moteur
  Serial.println("___stop___");
  CAN.sendMsgBuf(0x03,0, 3, _stop);
  Serial.println("Sending Message");
  //delay(10);
  FreinServo(1150);

}

  /*
  *  Programme pour recupérer les valeurs de la centrale
  *  @arg1 : pointeur vers le tableau gyro
  *  @arg2 : la taille du tableau
  *  return void;
  */
void inertial(int *tableau, int tailleTableau)
{
  //.....
}

  /*
  *  Programme pour initialiser la carte moteur
  *  return void;
  */
void initialisation()
{
  Serial.println("___initialisation___");
  CAN.sendMsgBuf(0x04,0, 1, _init);
  Serial.println("Sending Message");
}

  /*
  *  Fonction pour afficher un tableau
  *  @arg1 : pointeur vers le tableau à afficher
  *  @arg2 : la taille du tableau
  *  return void;
  */
void affichetableau(int *tableau, int tailleTableau)
{
  int i;
  for (i = 0 ; i < tailleTableau ; i++)
  {
    Serial.println("%d\n", tableau[i]);
  }
}
```

IMAGES DU PROJET (3D)







RÉSUMÉ

Projet EI4 – Stabilisation par effet gyroscopique

Ce projet est basé sur l'invention réalisée par l'« Institute for dynamic systems and control » de Zurich (Suisse), nommée Cubli 2D ou One Dimensional Cubli.

Cette invention est un cube à « 1 » dimension composé d'une roue de réaction qui est capable de se stabiliser grâce à l'effet gyroscopique. Le prototype est aussi composé d'une centrale à inertie permettant de récupérer la position et l'accélération du cube.

Cet effet est principalement utilisé pour ajuster l'attitude des navettes spatiales ou des satellites dans l'espace.

Ce rapport explique les différentes étapes réalisées afin de concevoir le début de ce prototype.

Mots-clés : Cubli, Prototype, Centrale à inertie, Arduino, Mécanique, Modélisation, Roue de réaction.

ABSTRACT

EI4 project – Stabilization by gyroscopic effect

This project is based on the invention carried out by the "Institute for dynamic systems and control" by Zurich (Swiss), named 2D Cubli or One Dimensional Cubli.

This invention is a one dimension cube composed of a reaction wheel which can become stable thanks to gyroscopic effect. The prototype is also composed of an inertial measurement unit allowing to recover position and acceleration of the cube.

This effect is mainly used to adjust the attitude of the space shuttles or of satellites in the space.

This report explains the different stages accomplished to conceive the beginning of this prototype.

Keywords: Cubli, prototype, inertial unit measurement, Arduino, mechanics, modeling, reaction wheel