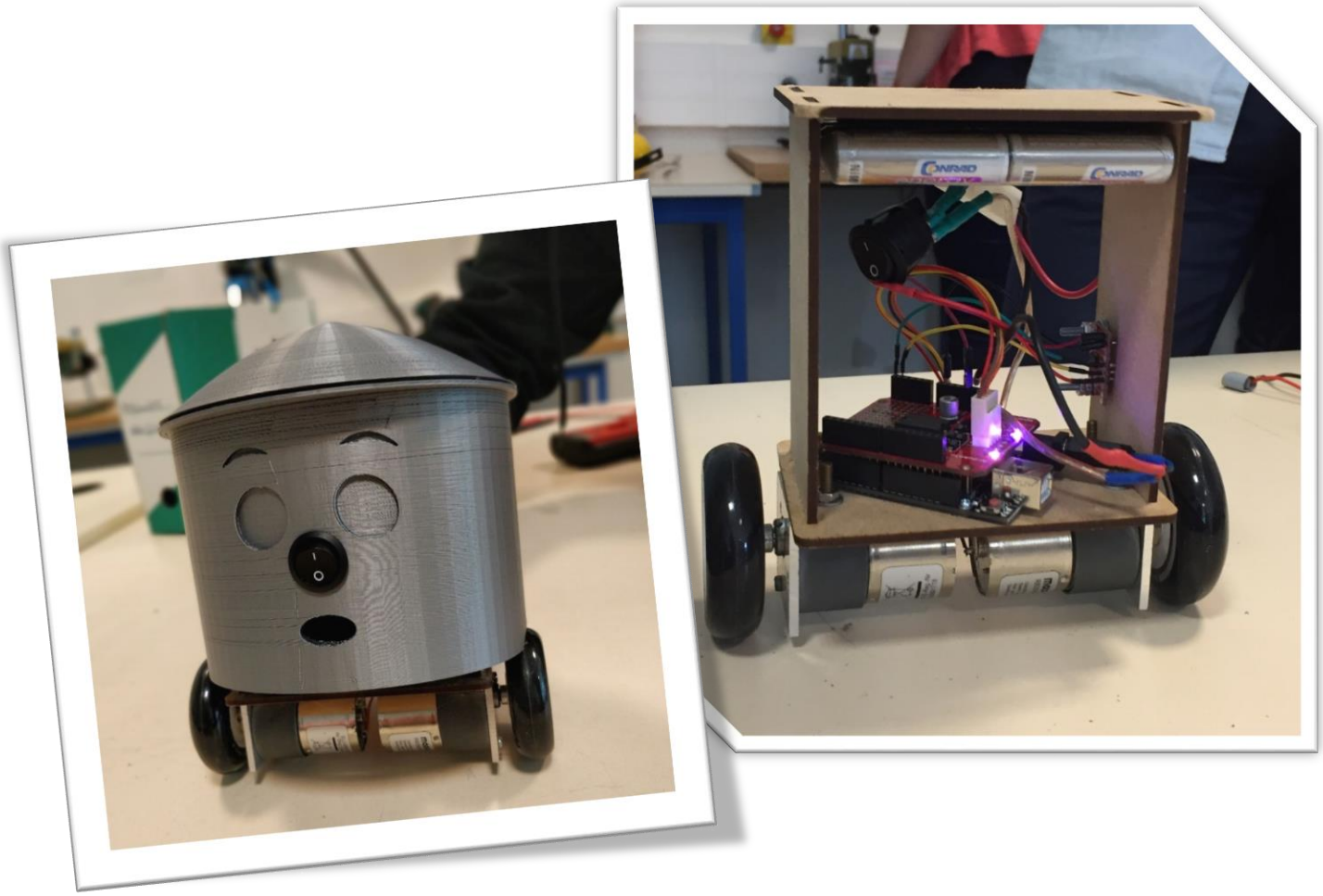
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Projet de Physique P6

STPI/P6/2018 – 022

ROBOT GYROPODE



Etudiants :

Elise ARNOLD — Alana CALSAVARA

Lucien DESSEAUX — Astou DIALLO

Clément DUVEAU — Leah MEURIE

Enseignant-responsable du projet :

Fabrice DELAMARE

Cette page est laissée intentionnellement vierge.

Date de remise du rapport : *18/06/2018*

Référence du projet : *STPI/P6/2018 – 022*

Intitulé du projet : ***Robot gyropode (il se maintient en équilibre sur deux roues comme un segway)***

Type de projet : ***expérimental***

Objectifs du projet:

- Concevoir et assembler un robot gyropode (un robot qui se maintient seul en équilibre sur deux roues parallèles) ;
- Initiation à la robotique, familiarisation avec le système de codage Arduino et des logiciels tels que Solidworks.

Mots-clefs du projet: ***robotique, autonomie, conception, programmation***

TABLE DES MATIERES

.

1. INTRODUCTION

Au cours du 4ème semestre d'étude, les élèves en STPI2 de l'INSA de Rouen sont amenés à réaliser un projet dans le cadre de l'EC de P6 : projet physique. Les objectifs de ce projet sont multiples.

Pour notre groupe, l'objectif principal était de concevoir et assembler un robot gyropode, à partir de nos connaissances, de notre réflexion, et de documentations et informations trouvées sur internet.

D'autre part, ce projet avait aussi pour but de nous permettre de nous familiariser avec le système de codage Arduino et avec différents logiciels tels que SolidWorks.

Nous avons réalisé ce projet de façon autonome. Ce fut à nous de choisir la forme et le matériau de notre robot, de réfléchir à la façon d'assembler les différents composants, et de réaliser dans son intégralité la programmation du robot.

Le principe du robot :

Le robot gyropode doit se maintenir seul en équilibre. C'est l'action des moteurs qui va permettre cela. Par exemple, si le robot penche vers l'avant, il suffit d 'accélérer les moteurs dans la même direction afin de le redresser. Au premier abord, le principe est relativement simple. Cependant, la difficulté est d'arriver à régler l'intensité de la réaction du robot en fonction de son degré d'inclinaison par rapport à la verticale.

On peut retrouver ce type de robot dans la vie de tous les jours. Par exemple, le *Segway* qu'on utilise pour se promener (voir image ci-dessous), fonctionne exactement selon ce principe. Pour accélérer, il suffit de se pencher en avant, et inversement pour ralentir. Ainsi, l'utilisateur n'a aucun risque de tomber ou de perdre l'équilibre.

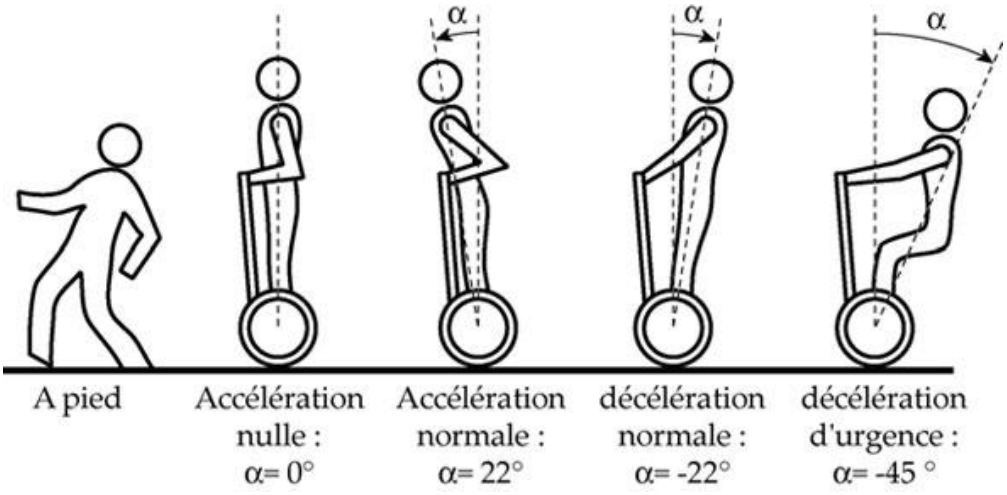


Schéma explicatif du fonctionnement d'un Segway

Le fonctionnement du robot :

Les moteurs sont contrôlés par une carte Arduino. Toutefois, afin de détecter cette perte d'équilibre, 2 instruments sont indispensables : l'accéléromètre et le gyroscope. L'accéléromètre permet de mesurer le vecteur d'accélération du robot, lié à la force de pesanteur et à toute autre force extérieure subie par le robot. Ainsi, on sait si le robot est en équilibre. Si ce n'est pas le cas, une correction est appliquée grâce aux moteurs. Quant au gyroscope, il permet de mesurer la vitesse angulaire sur les 3 axes de l'espace. Ces 2 capteurs font plusieurs centaines de mesures par seconde afin d'assurer à tout instant le maintien en équilibre, quel que soit le degré d'inclinaison.

Principe de fonctionnement du système électronique :

- Récupération des données d'angle par l'accéléromètre/gyroscope ;
- Analyse des données par la carte Arduino/le programme ;
- Activation des moteurs en conséquence ;
- Le degré d'inclinaison par rapport à la verticale détermine la puissance déployée par les moteurs pour retrouver l'équilibre, et donc la vitesse des roues nécessaire.

Liste du matériel nécessaire à la réalisation du robot :

- Un MPU-6050 contenant accéléromètre et gyroscope ;
- Une batterie 8x1.2V = 9,6V ;
- Deux moteur à courant continu 12V ;
- Deux roues de roller Ø63mm ;
- Une carte Arduino Uno R3 ;
- Un Motor Shield pour Arduino Uno ;
- Des plaques de bois ;
- Vis, écrous, colle et autre matériel de bricolage ;
- Une imprimante 3D pour l'impression de la coque.

2. METHODOLOGIE / ORGANISATION DU TRAVAIL

Dès la première séance, nous avons identifié les différentes tâches à réaliser : mécanique, électronique et programmation. Nous avons effectué des recherches, chacun de notre côté, afin d'avoir une compréhension globale du projet.

Dans un second temps, nous avons pris le temps de réfléchir tous ensemble à l'aspect esthétique de notre robot, à la disposition des différents composants et nous avons défini ses dimensions.

Ensuite, Alana et Leah ont dessiné les différentes pièces sur Solidworks afin d'avoir une idée claire du rendu final.

Une partie du groupe (Alana, Elise, Lucien, Leah) a travaillé sur l'assemblage du robot, pendant que le reste du groupe (Clément, Astou) commençait à travailler sur le développement du code.

Une fois l'assemblage du robot presque achevé, il nous restait à concevoir la coque (Elise, Astou, Leah, Alana), terminer le code et commencer les tests (Clément, Lucien).

Concernant la rédaction du rapport, nous y avons tous participé.

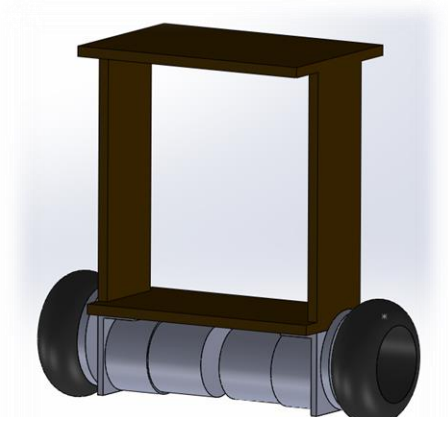
Voir en annexe le tableau détaillé de l'organisation du projet.

3. PRINCIPE DE REALISATION D'UN ROBOT GYROPODE

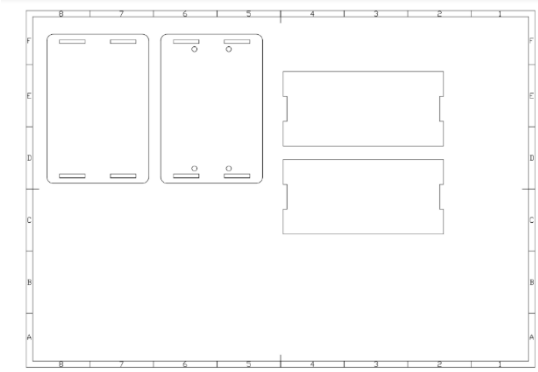
3.1. Partie Mécanique

Nous avons passé plusieurs semaines à travailler sur la partie mécanique de la conception du robot. En effet, il nous a fallu tout d'abord réfléchir à la forme que nous souhaitions donner à notre robot et la réaliser sur SolidWorks. Les roues et les moteurs étant fournis, nous avons choisi d'y ajouter un bâti en forme de « boîte », composée de 4 plaques de bois mesurant 12 cm par 6 cm pour les côtés et 8 cm pour les extrémités, sur laquelle nous avons fixé tous les composants (circuit électronique, capteurs, batterie).

Nous avons utilisé au départ des chutes de bois de récupération trouvées dans la salle de robotique, mais après quelques séances, nous avons dû changer de matériau. En effet, lorsque nous avons voulu assembler les différentes plaques à l'aide de vis, le bois se cassait et n'offrait donc plus une résistance suffisante pour notre robot. A la place, nous avons utilisé des plaques de même largeur et longueur, mais plus fines, d'un bois différent, découpées grâce à une découpeuse laser et assemblées grâce à des encoches et de la colle. Le découpage a été réalisé avec l'aide de Monsieur David Arrouche.

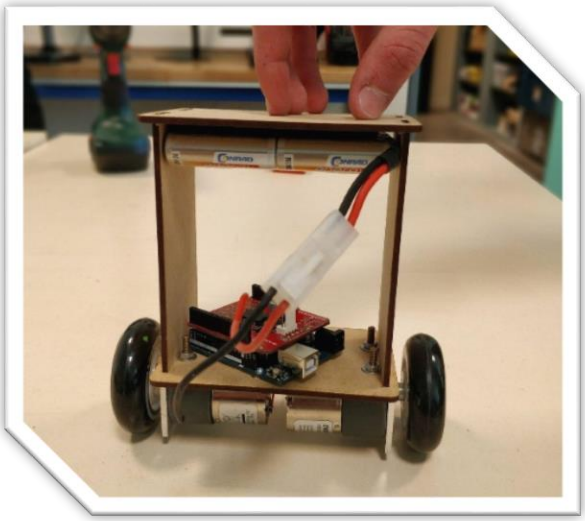


Modélisation SolidWorks de la structure du robot



Mise en plan SolidWorks des plaques à découper

Plaques en bois après découpage laser



Assemblage bâti, ensemble roues/moteurs, carte Arduino et batterie

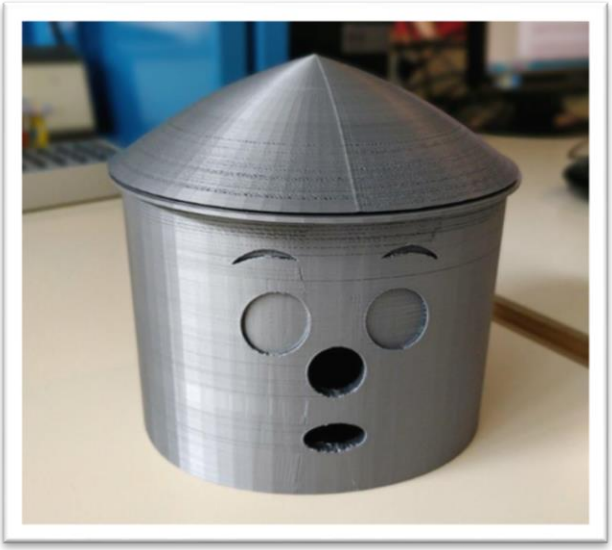
Une fois les plaques de bois assemblées, nous les avons fixées à l'ensemble moteurs/roues grâce à des vis. Il ne nous restait plus qu'à ajouter les autres composants, en choisissant différents moyens de fixation. Nous avons vissé la carte sur la plaque inférieure, fixé la batterie sur la plaque supérieure grâce à du velcro.

Nous avons choisi de placer la batterie en haut car déplacer le centre de gravité vers le haut permet de ralentir la chute du robot. C'est le principe du pendule inversé (voir bibliographie « Inverted Pendulum »). De plus, les vis des roues qui dépassent de la plaque inférieure nous auraient posé problème pour y mettre la batterie.

Nous avons placé la carte Arduino sur la plaque inférieure, même si les vis venant des moteurs ont gêné l'assemblage. Nous avons dû essayer différents placements avant de trouver celui qui convenait.

Enfin, pour masquer l'assemblage de notre robot qui n'était pas vraiment esthétique, nous avons décidé de concevoir une coque à l'aide de l'imprimante 3D *Cr10s* appartenant à Lucien. Conçue sur Solidworks, nous avons voulu lui donner une apparence amusante, avec des yeux, un nez (qui est en réalité l'interrupteur). Cette coque a été imprimée en 24 heures au total, avec une température de fusion de fil de 215 °C. Le matériau est du PLA (acide poly lactique), fabriqué avec de l'amidon de maïs.

Modélisation SolidWorks de la coque



Coque imprimée

3.2. Partie Electronique

3.2.1. Principe de fonctionnement du moteur

Marche/arrêt du moteur

Nous avons intégré un interrupteur dans l'assemblage afin de mettre le moteur en marche lorsque nous le souhaitons. Le circuit comprend une pile, un interrupteur fermé et un moteur. Voici un schéma du circuit mis en place :

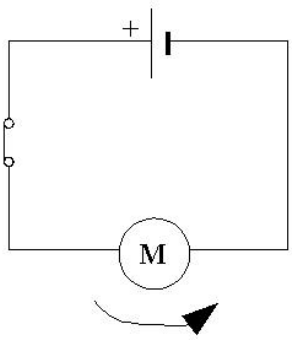
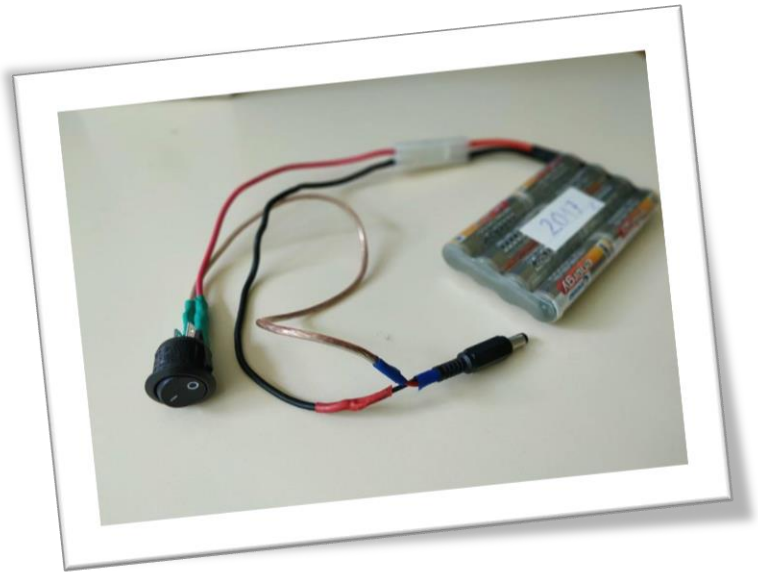


Schéma du circuit pile - interrupteur - moteur

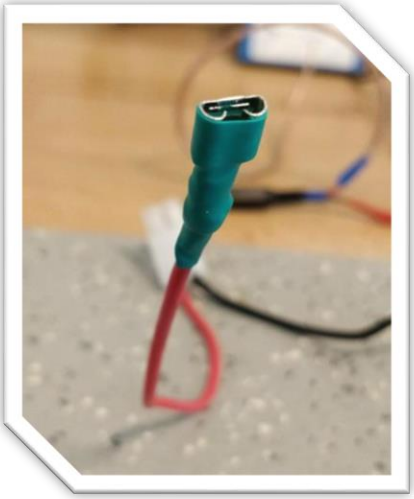


Assemblage pile - interrupteur - moteur

Nous avons donc réalisé ce circuit comme ci-contre. Sur la photo nous pouvons voir la prise qui sera branchée à la carte Arduino. La pile est connectée suivant le schéma précédent : le « + » (fil rouge) branché sur l'interrupteur et le « – » (fil noir) branché sur le moteur. Ensuite l'interrupteur est connecté au moteur par un troisième fil.

L'assemblage fils-interrupteur a été réalisé par nos soins, en utilisant des cosses femelles à sertir fast-on. Ce choix a été fait par question de praticité après conseil de Monsieur Fabrice Delamare. En effet nous avons placé l'interrupteur sur la coque et avec un assemblage à l'aide de ces cosses on peut facilement débrancher les fils de l'interrupteur, ce qui nous permet d'enlever la coque du robot et tout en laissant l'interrupteur dessus. Le tout a été solidifié à l'aide d'une gaine thermo rétractable. Nous avons testé l'interrupteur après assemblage et il fonctionnait correctement.

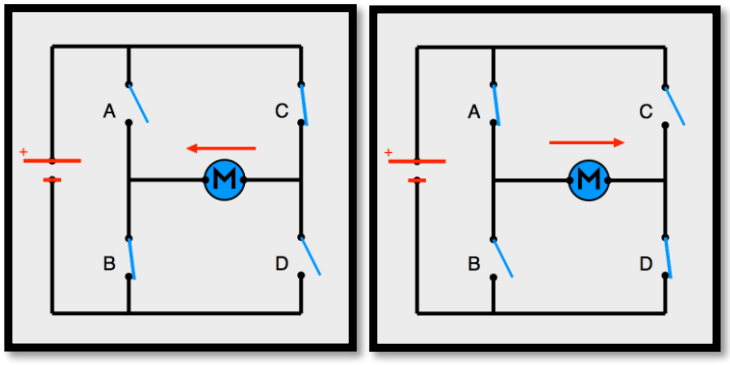
Cosse fast-on et fil avant assemblage



Assemblage solidifié

Contrôle du sens de rotation des moteurs

Notre robot doit pouvoir se déplacer en avant et en arrière, c'est pourquoi il est nécessaire de pouvoir contrôler le sens de rotation des roues, c'est-à-dire pouvoir inverser le sens du courant parcourant les moteurs. Un double pont en H (schématisé ci-ci-contre) permet cela : en activant les interrupteurs C-B ou A-D, on ferme le circuit et alimente le moteur dans un sens ou dans l'autre.



Motor shield utilisé : Ardumoto de Sparkfun

Un motor shield nous a donc été fourni, c'est-à-dire un circuit intégré dédié se connectant à la carte Arduino. Comportant 2 doubles ponts en H, il permet le contrôle de deux moteurs à courant continu. Ici des transistors pouvant être commandés par la carte Arduino remplacent les interrupteurs mécaniques, et c'est la batterie 9,6V qui alimente les moteurs.

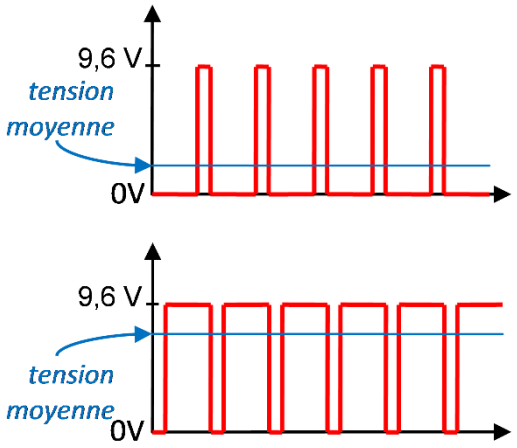
Modulation de la puissance fournie aux moteurs

Les ponts en H étant contrôlés par des interrupteurs, les moteurs sont alimentés en tout ou rien (9,6V ou 0V dans notre cas). Or afin de contrôler notre robot avec précision, il est nécessaire de pouvoir faire varier la puissance fournie aux moteurs.

On utilise alors une méthode appelée modulation de largeur d'impulsion, ou MLI (Pulse Width Modulation, ou PWM en anglais). Le principe de cette méthode est d'activer et désactiver l'alimentation plusieurs centaines de fois par secondes, afin d'obtenir une tension moyenne plus faible que la tension nominale des batteries.

En faisant varier le rapport entre les temps haut et les temps bas, la valeur moyenne de la tension varie également.

$$tension_{moyenne} = tension_{nominale} \times \frac{temps_{hauts}}{temps_{hauts} + temps_{bas}}$$



Signal MLI (modulation de largeur d'impulsion)

En opérant cette méthode à très haute fréquence (plusieurs centaines de fois par seconde), aucun à-coup n'est ressenti, et les moteurs fonctionnent comme s'ils étaient alimentés par une tension continue.

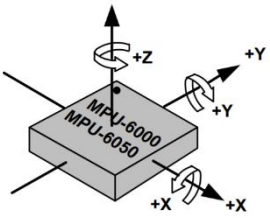
Dans notre cas, l'application de cette méthode fut très simple, puisque la carte Arduino a une fonction dédiée : *analogWrite(borne, puissance)*.

Ici, la puissance peut être comprise entre 0 et 255. À 0, les moteurs sont éteints, à 255, ils sont alimentés au maximum de la capacité de la batterie.

3.2.2. Principe de fonctionnement du capteur

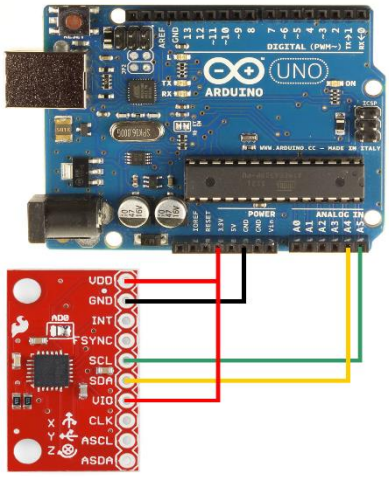
Le capteur utilisé est un MPU-6050 de Sparkfun. Il est constitué d'un thermomètre, d'un gyroscope ainsi que d'un accéléromètre.

L'accéléromètre enregistre l'accélération du capteur sur les 3 axes de l'espace, et le gyroscope les vitesses angulaires autour de ces trois axes. En combinant ces informations, on peut calculer l'angle du robot par rapport au sol, ce que nous allons expliquer dans la suite de ce rapport.



Orientation of Axes of Sensitivity and Polarity of Rotation

Connectique



Carte Arduino et capteur connectés

Le capteur est alimenté en 3,3V directement par la carte Arduino.

Le transfert des données est assuré par un bus I²C (Inter-Integrated Circuit). Les connections SDA et SCL du capteur sont donc reliées respectivement aux bornes A4 et A5 de l'Arduino.

L'alimentation de la borne VIO du capteur sert uniquement à définir l'adresse I²C du capteur (0x68[1] si on, 0x69 si off).

[1] Le préfixe 0x indique que le nombre qui suit est en hexadécimal

3.3. Partie Programmation

Pour concevoir un robot gyropode, c'est à dire, un robot capable de se maintenir seul debout sur deux roues parallèles, nous avons besoin qu'il puisse détecter son inclinaison par rapport au sol, et qu'il puisse se redresser. Pour cela, nous nous servons de deux capteurs, un accéléromètre et un gyroscope, ainsi que de deux moteurs à courant continu.

L'angle d'inclinaison est calculé par rapport à la normale au sol. S'il est nul, le robot est parfaitement droit. S'il est positif, le robot penche en avant, et s'il est négatif, il penche en arrière.

Le programme Arduino complet est fourni en annexe.

3.3.1. Calcul de l'angle du robot

N'ayant malheureusement reçu le capteur que fin mai, à cause d'un délai de livraison important, nous n'avons pu commencer les tests que très tardivement. C'est pourquoi nous avons préféré nous concentrer sur le cas d'un sol plat horizontal. Adapter le maintien en équilibre du robot sur sol quelconque aurait nécessité beaucoup plus de temps.

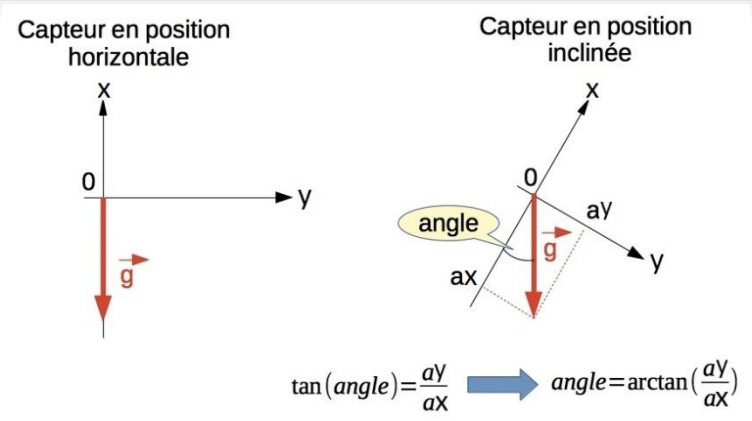
Les performances des deux capteurs sont complémentaires. L'accéléromètre permet de mesurer l'angle d'inclinaison du robot avec précision, mais uniquement lorsque le seul vecteur d'accélération est le vecteur de pesanteur. Il ne permet pas de calculer un angle fiable dès lors que le robot accélère (moteurs en marche) ou au moindre à-coup. Le gyroscope quant à lui mesure des vitesses angulaires. En les intégrant par rapport au temps, on peut calculer un angle précis, mais une valeur initiale est nécessaire. C'est là que l'accéléromètre intervient.

Les programmes Arduino sont composés de deux parties, la première est la fonction « setup » qui permet d'initialiser le robot, elle n'est lue qu'une seule fois. La seconde est la boucle « loop », qui contient l'essentiel du code, elle est lue un nombre infini de fois et c'est dans celle-ci que se situe le calcul de l'angle et la réponse des roues.

Initialisation du robot

A l'allumage, seul l'accéléromètre est utilisé afin de déterminer l'angle initial, ensuite le gyroscope permet de calculer l'angle à chaque instant.

```
//Initialisation de l'angle avec les données de l'accéléromètre
//57.296 = 180 / pi   rad -> deg
alpha = -57.296*atan((float) acc_y/acc_x)-CG;
```



On calibre également le gyroscope sur une période de 500 exécutions. En effet celui-ci donne des valeurs non nulles même au repos. C'est pourquoi on en fait la moyenne afin de la soustraire à chaque mesure. En somme, on « fait le zéro ». Le but est d'obtenir une valeur correcte et précise.

```
void init_gyro(){
  //calibration du gyroscope
  for (int i = 0; i < 500 ; i ++){      //Exécution du code 500 fois
    lecture_mpu_6050();
    gyro_z_cal += gyro_z;
    delay(5);
  }
  gyro_z_cal /= 500;                    //Moyenne sur les 500 mesures
}
```

Calcul de l'angle toutes les 4ms

Une fois le capteur calibré, on peut calculer l'angle à chaque exécution de la boucle « loop » (toutes les 4ms). Pour que l'Arduino puisse faire ce calcul, le code comprend la récupération des données brutes des capteurs, puis leur transformation en une valeur de l'angle d'inclinaison du robot.

Tout d'abord, l'accéléromètre nous donne des accélérations, donc il faut les transformer en utilisant un peu de trigonométrie, comme expliqué précédemment. Ensuite le gyroscope nous donne des vitesses angulaires, donc il faut les intégrer par rapport au temps afin d'obtenir une valeur d'angle. Ici, l'intégration est discrétisée :

$$nouvel\ angle = ancien\ angle + vitesse\ angulaire * temps\ écoulé\ (4ms)$$

Comme expliqué précédemment, la valeur de l'angle calculée avec l'accéléromètre devient fausse au moindre à-coup. Cependant, celui-ci permet de calculer l'angle d'inclinaison du robot à tout instant. La valeur calculée avec le gyroscope en revanche dépend de toutes les valeurs précédentes : les erreurs s'additionnent donc à chaque boucle, et l'angle calculé dérive de plus en plus de l'angle réel. C'est pourquoi on compose l'angle calculé avec les données des deux capteurs afin de minimiser l'erreur. Suite aux tests réalisés, nous avons retenu le calcul suivant :

$$angle_{final} = 0{,}98 \times angle_{gyroscope} + 0{,}02 \times angle_{accéléromètre}$$

```
void loop() {
//CALCUL DE L'ANGLE

  //Lecture des données brutes du capteur
  lecture_mpu_6050();

  //Prise en compte de la calibration du gyroscope
  gyro_z -= gyro_z_cal;

  //Calcul angle gyroscope
  //0.00014 = 0.004 * cste
  //nouvel angle = ancien + vitesse angulaire * temps écoulé
  alpha += gyro_z * 0.00014;

  //Calcul angle accéléromètre
  //57.296 = 180 / pi    rad -> deg
  alpha_acc = -57.296*atan((float) acc_y/acc_x)-CG;

  //Correction de la dérive du gyroscope en utilisant l'accéléromètre
  alpha = alpha * 0.98 + alpha_acc * 0.02;
```

3.3.2. Réaction des moteurs

Une fois que le robot a calculé son inclinaison, il doit se redresser en utilisant ses deux roues motorisées. La deuxième partie du code a donc ce rôle, réagir en fonction de l'angle calculé.

Si le robot penche en avant, la réponse sera de faire tourner les roues vers l'avant, et s'il penche en arrière dans l'autre sens. La puissance fournie par le moteur dépend de l'angle d'inclinaison du robot, afin d'avoir un mouvement plus maîtrisé et plus adapté à l'inclinaison.

Pour que le robot puisse se stabiliser à la verticale, il est important que les roues tournent moins vites quand l'angle est petit. Pour cela nous avons déterminé expérimentalement une fonction permettant de demander au moteur une puissance dépendant de l'angle d'inclinaison du robot.

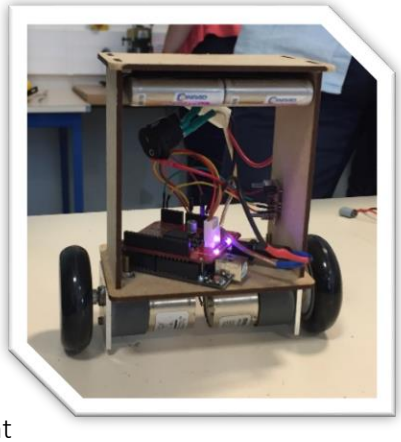
Si celui-ci est supérieur à 4°, le moteur enverra une puissance maximale (255), et si l'angle est inférieur à 4°, la puissance sera une fonction affine dépendant de la valeur absolue de l'angle :

$$Puissance = 95 + 45 \times |angle|$$

Le rôle de la constante (95) est de s'assurer que la puissance est suffisante pour faire tourner les roues. En effet, en dessous d'une puissance minimum, le poids du robot empêche les moteurs de tourner, et donc le robot de se redresser.

```
//REPONSE MOTEUR

  //Changement du sens des roues selon le signe de l'angle
  if(alpha > 0){
    digitalWrite(Dir_D, HIGH);
    digitalWrite(Dir_G, LOW);
  }else{
    digitalWrite(Dir_D, LOW);
    digitalWrite(Dir_G, HIGH);
  }

  //calcul de la puissance moteur en fonction de l'angle
  puissance_mot = 95+40*abs(alpha);
  if(abs(alpha)>4)puissance_mot = 255;

  analogWrite(PWM_G, puissance_mot);
  analogWrite(PWM_D, puissance_mot);

  //Attendre 4ms pour passer à la boucle suivante (250Hz)
  while(millis() - chrono < 4);
  chrono = millis();
}
```

4. CONCLUSIONS ET PERSPECTIVES



Conclusions sur le travail réalisé

Nous avons tous trouvé ce projet passionnant. Il est vrai qu'au début, cela nous semblait compliqué de concevoir un robot dans sa totalité du fait du peu d'expérience pratique et technique que nous avions. Cela ne nous a en rien empêché de porter beaucoup d'intérêt à ce projet. Avec le recul, nous pouvons dire que c'est grâce à notre motivation que nous avons réussi à le mener à bien, même s'il nous semblait pourtant compliqué au départ. Nous avions vraiment envie de voir l'aboutissement de notre travail, et surtout d'en être fiers. Si on fait le bilan, celui- ci est très positif. Nous sommes très satisfaits du résultat, autant sur le plan électronique, mécanique que de la programmation. Toutes les exigences du cahier des charges ont été respectées. Notre robot est solide, et nous avons utilisé une grande majorité de matériaux de récupération pour le construire.

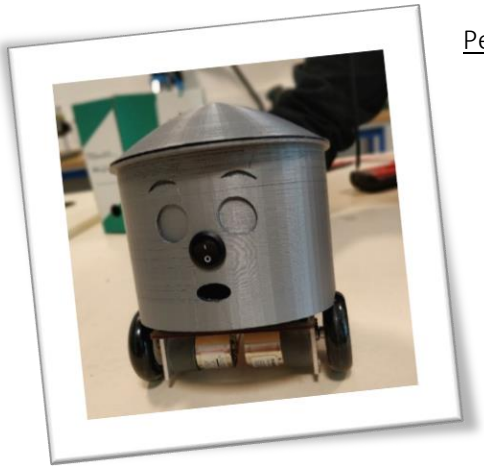
Conclusions sur l'apport personnel de cet E.C. projet

Pour conclure, ce projet nous a beaucoup apporté, tant sur le plan humain que technique.

Premièrement, d'un point de vue personnel, ce projet nous a donné l'opportunité de travailler en groupe, sur un sujet que nous avons choisi, et ce dans une durée limitée. Par ailleurs, il est agréable de constater qu'à chaque nouveau projet, nous sommes de plus en plus à l'aise quant à la façon d'organiser notre travail de groupe.

Pour ce projet, nous avons veillé à répartir le travail en respectant les préférences, les facilités et les affinités de chacun. Le tout, toujours dans le but de produire le travail le plus efficacement possible, et ce, pour fournir le meilleur résultat possible. Nous devons admettre que cela est plus facile à dire qu'à faire, mais nous avons toujours donné le meilleur de nous-même, et c'est peut-être ce qu'il y a de plus important.

Ensuite, d'un point de vue technique et scientifique, cela nous a permis de découvrir le monde de la robotique, le tout dans une ambiance toujours studieuse et conviviale. Il nous aura aussi permis d'affiner nos choix d'orientation pour l'année prochaine.



Perspectives pour la poursuite de ce projet

Nous avons rempli les objectifs qui nous étaient donnés au début de notre projet, pourtant celui-ci pourrait être poursuivi. En effet, si notre robot tient seul en équilibre, il ne peut pas être dirigé. Cela pourrait être une façon de poursuivre ce projet. Une autre perspective intéressante serait de refaire un robot gyropode de plus grande dimension. Mais quoi qu'il en soit, nous sommes très satisfaits de notre robot et d'avoir atteint les objectifs fixés.

5. BIBLIOGRAPHIE

5.1. Liens internet

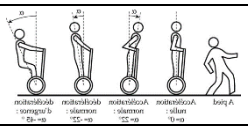
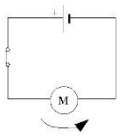
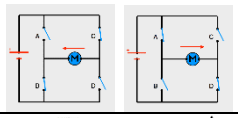
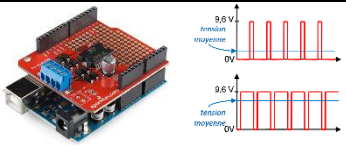
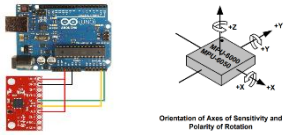
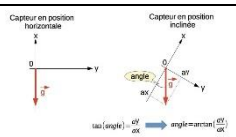
En Français :

- **Open Classrooms.** « Programmez vos premiers montages avec Arduino ». https://openclassrooms.com/courses/programmez-vos-premiers-montages-avec-arduino (Valide à la date du 16/06/2018).
- **Ornicom.** « Le gyropode segway ™: comment ca marche? ». https://www.ornicom.fr/le-gyropode-segway-comment-ca-marche/ (Valide à la date du 16/06/2018).
- **Wikiversity.** « Travail pratique : Utilisation d'un Accéléromètre MPU6050 ». https://fr.wikiversity.org/wiki/Micro_contr%C3%B4leurs_AVR/Travail_pratique/Utilisation_d'un_Acc%C3%A9l%C3%A9rom%C3%A8tre_MPU6050 (Valide à la date du 16/06/2018).

En Anglais :

- **Brokking.net.** « Your Arduino Balancing Robot (YABR) ». http://www.brokking.net/yabr_main.html (Valide à la date du 17/06/2018).
- **InvenSense Inc.** « MPU-6000 and MPU-6050 Product Specification Revision 3.4 ». http://43zrtwysvxb2gf29r5o0athu.wpengine.netdna-cdn.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf (Valide à la date du 17/06/2018).
- **Maker Pro.** « How to Build an Arduino Self-Balancing Robot ». https://maker.pro/arduino/projects/build-arduino-self-balancing-robot (Valide à la date du 17/06/2018).
- **Wikipedia.** « Inverted pendulum ». https://en.wikipedia.org/wiki/Inverted_pendulum#Essentials_of_stabilization (Valide à la date du 17/06/2018).

5.2. Crédit images

Page 5		https://www.ornicom.fr/le-gyropode-segway-comment-ca-marche/
Page 9		http://lewebpedagogique.com/vallejo/cours-de-5ieme/chapitre-2-le-sens-du-courant-electrique/
Page 10		https://openclassrooms.com/courses/programmez-vos-premiers-montages-avec-arduino/le-moteur-a-courant-continu-partie-2-le-pont-en-h-et-les-circuits-integres
Page 11		https://learn.sparkfun.com/tutorials/ardumoto-shield-hookup-guide http://arduino.blaisepascal.fr/index.php/2015/07/29/conversion-numeriqueanalogique-pwm/
Page 12		Photos officielles Sparkfun et Arduino http://43zrtwysvxb2gf29r5o0athu.wpengine.netdna-cdn.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf
Page 13		http://gilles.thebault.free.fr/spip.php?article32

6. ANNEXES

6.1. Tableau : organisation du projet

Répartition des tâches	Date de la séance : / Responsables	6/02	13/02	20/02	20/03	20/03	27/03	3/04	10/04
Partie programmation	Clément Duveau et Lucien Desseaux	Lancement	Familiarisation avec le langage de programmation	Compréhen-Sion du Modèle mathématique	→	→	Codage de la détection de L'angle avec Le sol	→	→
Partie électronique	Lucien Desseaux, Alana Calsavara et Clément Duveau	Lancement	Commande du matériel	→	→	→	→	→	Etablissemnt de la liaison entre la pile l'interrupteur et le moteur
Partie mécanique	Alana Calsavara, Lucien Desseaux, Leah Meurie, Astou Diallo et Elise Arnold	Lancement	Essai de réalisation de la coque en sciant des chutes de bois → échec car trop fragile	Conception Sur Solidworks de la structure du robot	→	Impression de la structure à l'aide d'une imprimante laser	Assemblage de la structure et collage	Création de la coque du robot sur Solidworks	→

Répartition des tâches	Dates de la séance : / Responsables	17/04	15/05	22/05	29/05	5/06	12/06
Partie programmation	Clément Duveau et Lucien Desseaux	Codage de la réponse des roues	Vérification du code	TESTS	TESTS	TESTS FINAUX/ Finitions	Préparation de l'oral
Partie électronique	Clément Duveau, Alana Calsavara et Lucien Desseaux	Soudures au niveau du moteur	Soudure au niveau de la carte arduino	Transfert du programme sur la carte arduino	-	-	Préparation de l'oral
Partie mécanique	Alana Calsavara, Lucien Desseaux,Leah Meurie, Astou Diallo et Elise Arnold	-	-	Assemblage du robot avec tous ses composants	Impression de la coque avec une imprimante 3D	Assemblage final du robot avec la coque	Préparation de l'oral
Rédaction du rapport	Tous	Rédaction du rapport	→	→	→	Relecture du rapport	Préparation de l'oral

6.2. Programme Arduino complet

```
//Bibliothèque I2C
#include <Wire.h>

//capteur et angle
int gyro_x, gyro_y, gyro_z;
long acc_x, acc_y, acc_z;
int temperature;
long gyro_x_cal, gyro_y_cal, gyro_z_cal;
float alpha, alpha_acc;
//horloge
long chrono;
//moteurs
int puissance_mot;
int PWM_D=11;
int Dir_D=13;
int PWM_G=3;
int Dir_G=12;
//centre de gravité du robot
//(constante expérimentale)
float CG=-0.5;

void setup() {
  //Init moteurs
  pinMode(PWM_D,OUTPUT);
  pinMode(Dir_D,OUTPUT);
  pinMode(PWM_G,OUTPUT);
  pinMode(Dir_G,OUTPUT);

  //Init capteur
  Wire.begin();
  init_mpu_6050();
  init_gyro();

  //Initialisation de l'angle
  //  avec les données de l'accéléromètre
  //57.296 = 180 / pi   rad -> deg
  alpha = -57.296*atan((float) acc_y/acc_x)-CG;

  //Initialisation de l'horloge
  chrono = millis();
}

void loop() {
//CALCUL DE L'ANGLE

  //Lecture des données brutes du capteur
  lecture_mpu_6050();

  //Prise en compte de la calibration du gyroscope
  gyro_z -= gyro_z_cal;

  //Calcul angle gyroscope
  //0.00014 = 0.004 * cste
  //nouvel angle
  //  = ancien + vitesse angulaire * temps écoulé
  alpha += gyro_z * 0.00014;

  //Calcul angle accéléromètre
  //57.296 = 180 / pi   rad -> deg
  alpha_acc = -57.296*atan((float) acc_y/acc_x)-CG;

  //Correction de la dérive du gyroscope
  //  en utilisant l'accéléromètre
  alpha = alpha * 0.98 + alpha_acc * 0.02;

//REPONSE MOTEUR

  //Changement du sens des roues selon le signe de l'angle
  if(alpha > 0){
    digitalWrite(Dir_D, HIGH);
    digitalWrite(Dir_G, LOW);
  }else{
    digitalWrite(Dir_D, LOW);
    digitalWrite(Dir_G, HIGH);
  }
```

```
  //calcul de la puissance moteur en fonction de l'angle
  puissance_mot = 79+22*abs(alpha);
  if(abs(alpha)>8)puissance_mot = 255;

  analogWrite(PWM_G, puissance_mot);
  analogWrite(PWM_D, puissance_mot);

  //Attendre 4ms pour passer à la boucle suivante (250Hz)
  while(millis() - chrono < 4);
  chrono = millis();
}


void lecture_mpu_6050(){
  //Lecture des données brutes du capteur

  Wire.beginTransmission(0x68);
  //Envoi du signal de démarrage
  Wire.write(0x3B);
  Wire.endTransmission(false);
  //Besoin de 14 octets du capteur
  Wire.requestFrom(0x68,14);
  //Attente de la réception
  while(Wire.available() < 14);
  acc_x = Wire.read()<<8|Wire.read();
  acc_y = Wire.read()<<8|Wire.read();
  acc_z = Wire.read()<<8|Wire.read();
  temperature = Wire.read()<<8|Wire.read();
  gyro_x = Wire.read()<<8|Wire.read();
  gyro_y = Wire.read()<<8|Wire.read();
  gyro_z = Wire.read()<<8|Wire.read();
}


void init_mpu_6050(){
  //Activation du capteur

  Wire.beginTransmission(0x68);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();
  //Configuration de l'accéléromètre (+/-8g max)
  Wire.beginTransmission(0x68);
  Wire.write(0x1C);
  Wire.write(0x10);
  Wire.endTransmission();
  //Configuration du gyroscope (500 deg/s max)
  Wire.beginTransmission(0x68);
  Wire.write(0x1B);
  Wire.write(0x08);
  Wire.endTransmission();
}


void init_gyro(){
  //calibration du gyroscope

  //Exécution du code 500 fois
  for (int i = 0; i < 500 ; i ++){
    lecture_mpu_6050();
    gyro_z_cal += gyro_z;
    delay(5);
  }
  //Moyenne sur les 500 mesures
  gyro_z_cal /= 500;
}
```