RÉALISATION D'UN BANC D'ÉQUILIBRAGE

Laura Pérez Amigó
Sandra Terradellas Balaguer

Tuteur: Jérôme Noailles
Troyes, printemps 2014

Remerciements

On souhaite, à travers ces lignes, remercier tous ceux qui ont aidée et ont contribué à la réalisation de ce travail de recherches.

On tient tout particulièrement à remercier M. Jérôme Noailles et M. Pierre Antoine Adragna pour nous aider avec tous les problèmes et questions qui nous ont surgit pendant la réalisation du projet.

À M. Benoît Panicaud pour faire possible notre participation en ce projet.

Remercier aussi aux techniciens du Halle Industrielle M. François Weil, M. Bruno Lesage et M. Abdellah Oukkir pour nous fournir avec touts les outils nécessaires et fabriquer les pièces qu'on a demandées et de plus nous donner des conseils.

À M. Olivier Didon pour réaliser toutes les plaques imprimés qu'on a utilisé et pour nous aider à résoudre des problèmes et doutes en relation à la part électronique.

Dernière, mais non moins important, remercier M. Laurent Daniel pour nous aider avec le logiciel LabView et tout les problèmes qu'on a eu avec ce logiciel et sa programmation.

SOMAIRE

Remerciements 2
1. Introduction et objectifs 5
2. Mis en fonctionnement 5
2.1. Arduino 5
2.2. Premiers programmes 5
3. Matériaux pour réaliser le banc 10
3.1. Moteur 10
3.2. Multiwatt15 10
3.3. Système de poulies 12
3.4. Support moteur 14
3.5. Protection supérieur 15
4. Components de mesure 16
4.1. Capteur courant 16
4.2. Capteur proximité 20
4.3. LED 26
4.4. Bouton arrêt d'urgence 28
5. Banc définitif 29
5.1. Programme définitif 29
5.2. Montage définitif 31
6. Équilibrage 31
6.1. Équilibrage statique et dynamique 31
6.2. Software utilisé 32
6.2.1. LabView 32
6.2.2. PicoScope 33
6.3. Accéléromètre 34
6.3.1. Accéléromètre Brüel & Kjaer type 8325 34
6.3.2. Générateur de courant 35
6.4. Résultats 37
6.4.1. Premiers résultats 37
6.4.2. Filtrage 39
6.4.3. Résultats finals 41
7. Conclusions 43
8. Bibliographie 44

ANNEXE 46

I. Mises en plan 46

II. Code du programme définitif 50

1. Introduction et objectifs

Dans le monde de la mécanique est indispensable que tous les solides qui sont en rotation soient équilibres. Un mobile bien équilibré permet un meilleur rendement et une performance optimale parce que le déséquilibrage provoque des vibrations.

La finalité de ce projet est réaliser l'équilibrage dynamique d'un disque d'inertie, pour ça, un banc d'essai à échelle réduite doit être réalisé. Ce banc permettra de tester et roder un disque d'inertie et le régler en phase d'utilisation. Ce projet a débuté avec nous et il continuera les prochains semestres avec des autres étudiants. Ils continueront avec un banc d'essai plus grand et avec plus de puissance qui ressemble plus à la réalité de l'étude qu'on veut faire. Est pour ça que dans ce rapport on explique tous les problèmes surgis et les procédés suivis pour trouver la meilleure solution.

La première étape a consisté à faire l'alimentation du moteur et l'asservissement en vitesse. La seconde étape est réaliser l'acquisition et le traitement du signal pour corriger les défauts. Finalement, la dernière étape est tester le disque d'inertie avec différents poids de 0,5, 1 et 2 N et voir s'il est bien équilibré ou non.

2. Mis en fonctionnement

2.1. Arduino

Arduino est un circuit imprimé avec un microcontrôleur qui peut être programmé pour analyser et produire des signaux électriques grâce auquel on peut contrôler appareils électroniques. Dans ce projet il sera nécessaire cette carte pour connecter l'ordinateur avec le moteur qu'on veut contrôler pour transmettre l'information.

Pour le programmer il faudra utiliser le logiciel Arduino qui est open source. Le programme sera écrit dans un document et après on pourra l'envoyer à la carte. Une fois envoyée on pourra voir les résultats et envoyer des commandes avec le moniteur sérié. [1][2]

Il a des différents entrées et sorties (analogiques et digitales) pour connectes les composants nécessaires du circuit qu'on veut programmer. On peut trouver aussi l'entrée du port USB avec lequel sera possible la connexion à l'ordinateur.

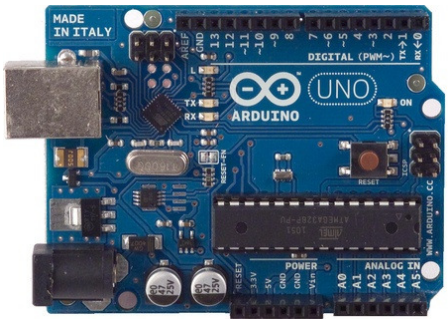


Image 1: Carte Arduino UNO [1]

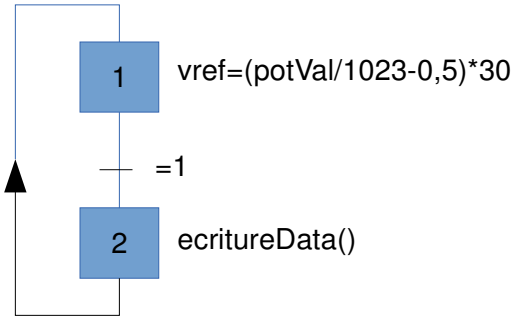
2.2. Premiers programmes

Pour faire fonctionner le moteur comme on veule c'est nécessaire créer un programme que permettra donner des instructions. Ce programme est programmé en C++ et sa structure général est composé par trois parties : la partie déclaration des variables, la partie initialisation et configuration des

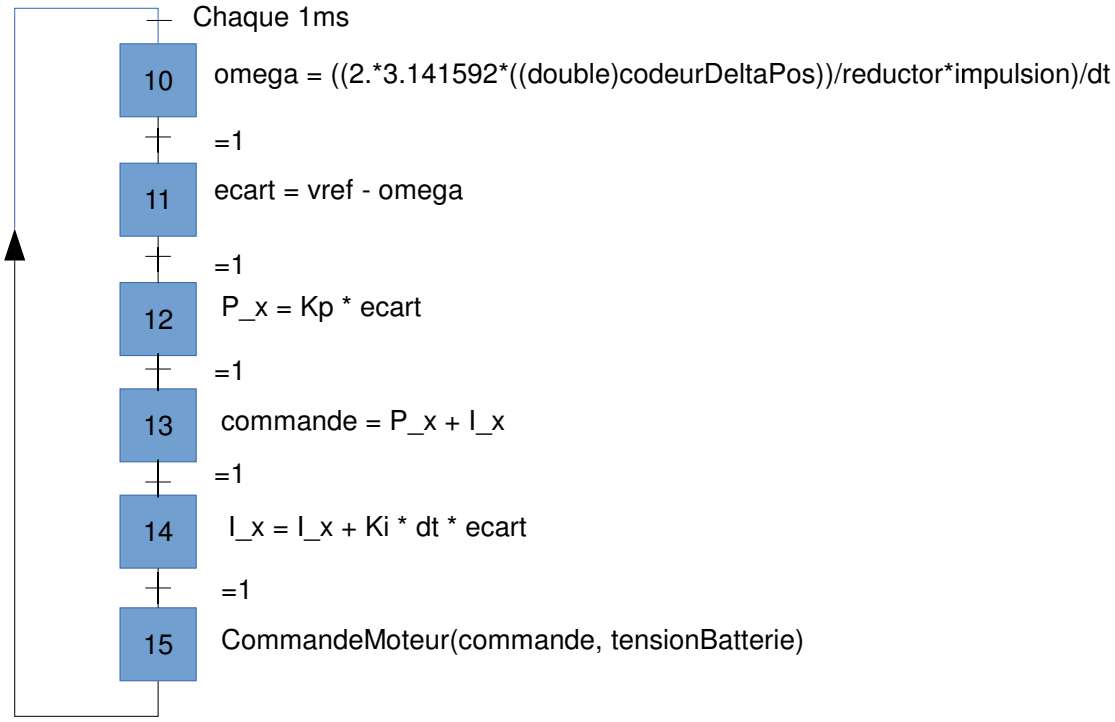
entrées/sorties (la fonction *setup*) et la partie principale qui s'exécute en boucle (la fonction *loop*). Une fois le programme est crée on doit le télécharger dans la carte Arduino pour envoyer les instructions au moteur. [2]

La première version du programme était jusqu'à contrôler le moteur et calculer les paramètres nécessaires ainsi que modifier le sens de rotation grâce à les interruptions. Mais avec ce programme on ne peut pas modifier la vitesse ou le sens de rotation, tout est fait par le programme. Comme on veule pouvoir réguler ces paramètres, c'était nécessaire le modifier. La deuxième version était ajouter un potentiomètre qui régulait la vitesse et changeait le sens de rotation et un interrupteur marche/arrête en mettent une commande dans le loop. Ce programme est expliqué ci-dessous :

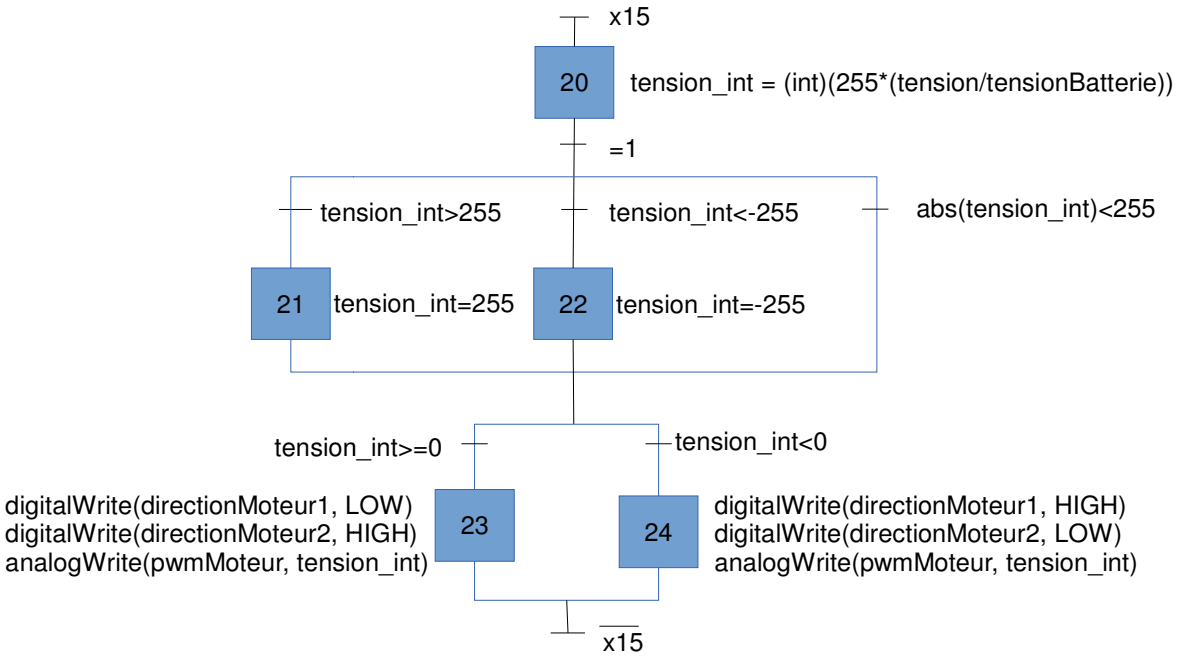
loop()



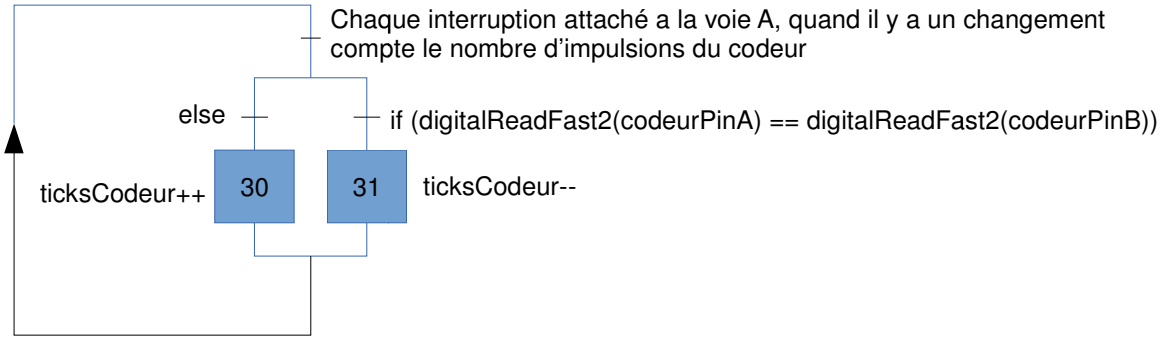
isrt()



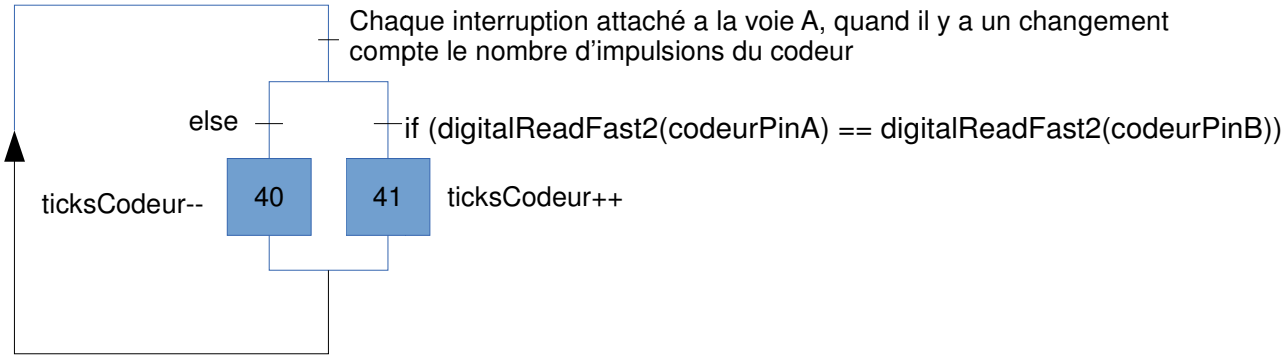
Commande moteur()



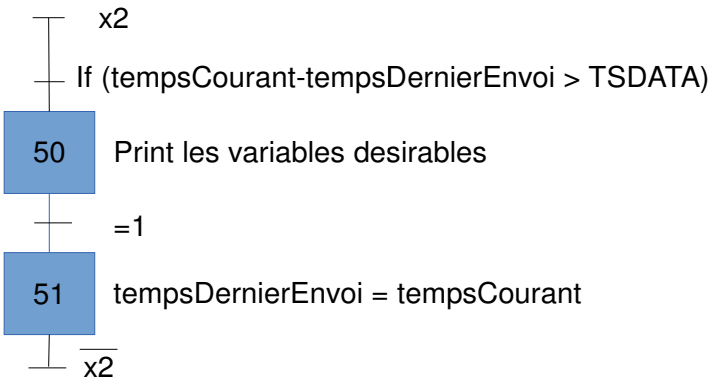
GestionInterruptionCodeurPinA()



GestionInterruptionCodeurPinB()



ecritureData()



Une variant du programme qu'on a testé était faire le contrôle de la vitesse et le sens de rotation avec un potentiomètre. De ce manière, la vitesse 0 correspond à la position central du potentiomètre et quand on le tourne vers un coté, la vitesse augmente jusqu'à le maximum et la même chose vers l'autre coté mais dans le sens contraire. Les changements nécessaires dans le programme sont les suivants :

```
//Potenciometre
const int potPin = A0;
double potVal;

// Boucle principale
void loop() {
  potVal = analogRead(potPin);
  vref=(potVal/1023-.5)*30; //rad/s
}
```

La version définitive est faire le contrôle avec l'ordinateur parce qu'est plus fiable. Ça veut dire qu'on mit la vitesse désire du moteur dans le moniteur série pour réguler la vitesse (signe positive ou négative pour changer les sens de rotation) et si on veut l'arrêter il faut mettre un 0. Ce numéro qu'on écrit doit être entre 0 et 255 et il aura une relation avec la vitesse en rpm. Pour faire ça, on a modifié le programme en créant les variables nécessaires et mettent dans le boucle une commande qui crie une fonction pour lire le numéro qu'on a écrit dans le moniteur série. Une fois le numéro est reçu, le moteur tourne à la vitesse et au sens correspondent. Le programme avec les modifications expliquées est le suivant:

```
#define directionMoteur1  4
#define directionMoteur2  5
#define pwmMoteur  9
int vref = 0;

//Variables nombreReçu
int octetReception=0;
long nombreReception=0;
long nombreReception0=0;
```

```
String chaineReception="";

// Initialisations
void setup(void) {
  Serial.begin(115200);
  Serial.flush();
  pinMode(directionMoteur1, OUTPUT);
  pinMode(directionMoteur2, OUTPUT);
  pinMode(pwmMoteur, OUTPUT);

  digitalWrite(pwmMoteur,HIGH);
  digitalWrite(directionMoteur1,LOW);
  digitalWrite(directionMoteur2,LOW);
}

// Boucle principale
void loop(){
  if (Serial.available() == true) {
    nombreReception=lecture_port();
  }
  if (nombreReception!=nombreReception0) {
    nombreReception=constrain(nombreReception,-255,255);
    Serial.println(nombreReception);
    nombreReception0=nombreReception;
    if (nombreReception>0) {
      analogWrite(pwmMoteur,nombreReception);
      digitalWrite(directionMoteur1,HIGH);
      digitalWrite(directionMoteur2,LOW);
    }
    if (nombreReception==0) {
      digitalWrite(pwmMoteur,LOW);
      digitalWrite(directionMoteur1,LOW);
      digitalWrite(directionMoteur2,LOW);
    }
    if (nombreReception<0) {
      analogWrite(pwmMoteur,-nombreReception);
      digitalWrite(directionMoteur1,LOW);
      digitalWrite(directionMoteur2,HIGH);
    }
  }
}

int lecture_port() {
  int octetRecu=0;
  int compt=0;
  boolean signe=true;
  long nombreRecu=0;
  while (Serial.available()>0) {
    octetRecu=Serial.read();
    if (octetReception==10) {
      break;
    }
    else {
```

```
    if ((octetRecu=='-') && (compt==0)){
      signe=false;
    }
    compt=compt+1;
    octetRecu=octetRecu-48;
    if ((octetRecu>=0)&&(octetRecu<=9)){
      nombreRecu = (nombreRecu*10)+octetRecu;
    }
  }
  delay(1);
 }
 if (signe==false){
  nombreRecu=nombreRecu*(-1);
 }
 return(nombreRecu);
}
```

3. Matériaux pour réaliser le banc

3.1. Moteur

Pour faire le banc d'essai à échelle réduite, on utilisera un moteur à courant continu (24V C.C). Cet propulseur est le DOGA DO 162.4122.3B.00 qui a un courant nominal de 2,5A et un pic de courant de 8,5A. C'est important aussi savoir que il a un pair motrice de 0,18 Nm et que il peut arriver à une vitesse de 1500 rpm. [3]

Pour commander la vitesse du moteur on doit connecter un circuit sur une carte Arduino Uno et aussi un programme qui permet contrôler les choses qui fa le moteur. Sur la carte Arduino Uno nous mettons les composants suivants:

- Une résistance de 1Ω
- 2 diodes
- 2 condensateurs de 100 µF
- 1 composant de puissance (Multiwatt 15, L298N)
- Le moteur.
- 1 batterie de 24 V

Pour le bon fonctionnement du mécanisme c'est très important la connexion du composant de puissance.

3.2. Multiwatt15

Le multiwatt15 est haute tension, haute courant, double pont complet driver conçu pour accepter de standard niveaux logiques et conduire la charge inductive. Les émetteurs de transistors inférieurs de chaque pont sont connectés ensemble et le correspondent terminal extérieur s'utilise pour les connections d'une résistance de détection. [4]

Est important aussi savoir que comme on a seulement un moteur, on utilise seulement un de les deux ponts complets, les pins de le 1 à le 9.

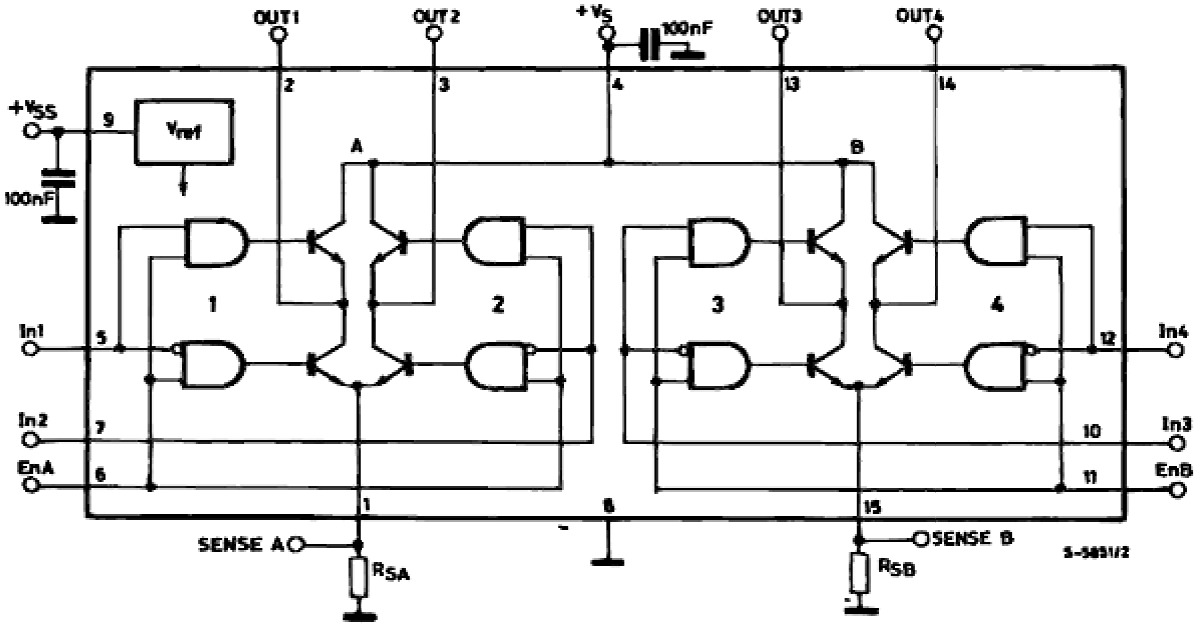


Image 2: schéma multiwatt15 [5]

Les connexions qu'on met sont les suivants :

- Pin 1: Avec une résistance (Rsa) de 1Ω et à la masse.
- Pin 2 et 3: Chaque un à une borne du moteur
- Pin 4: à 24V
- Pin 5 et 7: les directions du moteur, un au Pin 4 et l'autre au 5 de l'Arduino
- Pin 6: Est le Eneable il va à le Pin 9 de l'Arduino
- Pin 8: à la masse de l'Arduino
- Pin 9: à le 5V de l'Arduino

Si on utilise seulement un pont on a une intensité maximum de 2A et avec ce courant le moteur a beaucoup de difficultés pour rouler et quand il tourne il marche très lentement. Pour éviter ce problème on doit trouver une solution pour augmenter le courant.

Après voir que le montage initial ne marchait pas parce que le composant n'avait suffisant intensité et conséquemment il chauffait, on a changé les connexions pour avoir plus d'intensité. Ce composant en pont H, a 2 ampères à chacun des deux ponts, pour obtenir les 4 ampères désirables on a connecté les deux ponts en parallèle comme montre la figure ci-dessous [5]:

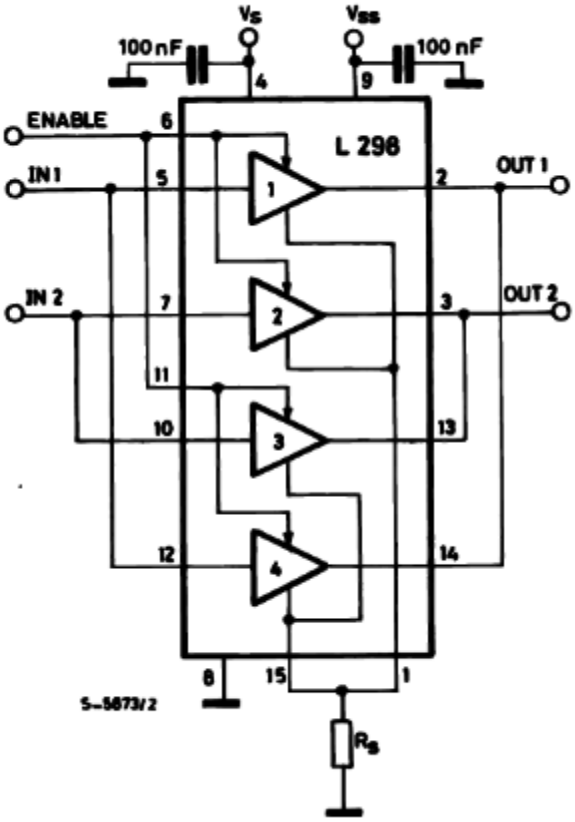


Image 3: Connexions en parallèle du composant L298N [5]

Maintenant ce possible faire fonctionner avec plus de vitesse le montage grâce à le double de capacité de courant au pont.

Un autre changement était éliminer la résistance, les diodes et les condensateurs. On a pu faire ça parce que le circuit est alimenté pour un élément qui limite le voltage et aussi l'ampérage et de cette manière on ne dépassera le limite de 4A.

Finalement, on a vu aussi que l'accouplement n'était pas correct. On doit changer ce pour réduire la friction et améliorer la jonction entre le moteur et le disque. À sa place on a mis un système de poulies qui augmentait le pair moteur du disque grâce à la relation de transmission.

3.3. Système de poulies

Pour monter un système de poulies on a calculait tous les paramètres nécessaires pour acheter dans le catalogue de HPC celles plus appropriés. Les dimensions doivent être les plus petites pour s'adapter a l'échelle du montage et la relation de transmission choisit est de 1/3. La poulie petite, que corresponde au moteur a un diamètre de 20,15mm avec 26 dents(référence 26T2,5-8) et la grosse que est mis à l'arbre du disque d'inertie a un diamètre de 63,15mm avec 80 dents(référence 80T2,5-8). Les deux sont poulies dentées de pas 2,5 avec une largeur de courroie de 4mm. La courroie a une longueur de 380mm (référence T2,5/380/4). [6]

Première on doit choisir les diamètres des deux poulies. On veut que la relation de transmission (i) soit de 0,3333 alors si la vitesse du moteur est de 1500 rpm (n_1), la vitesse du disque sera:

$$i = \frac{n_2}{n_1} = 0{,}333 \rightarrow n_2 = n_1 \cdot 0{,}333 = 500rpm$$

Pour les dimensions du moteur on a choisit le diamètre de ce poulie de 20,15mm et un pas (t) de 2,5 mm. Avec le même pas et avec la suivant équation on trouve l'autre mesure :

$$n_1 \cdot d_1 = n_2 \cdot d_2 \rightarrow d_2 = \frac{1500 \cdot 20{,}15}{500} = 60{,}45mm$$

Le mesure de 60,45mm ne se trouve pas dans le catalogue de HPC pour ça on a choisit le plus proche que c'est la mesure de 63,15mm. Il y a des autres paramètres importants à calculer mais avant on doit savoir les constants suivants [7]:

Entraxe souhaité: e=120 mm
Pas: t=2,5 mm
Z_1 = 26
Z_2 = 80
C_2 = 1 parce que c'est une fonction continue
C_3 = 0 parce que i<1

$$\text{Longueur: l} = t \cdot \frac{z_1 + z_2}{2} + 2 \cdot e + \frac{\left(\frac{t \cdot (z_2 - z_1)}{\pi}\right)^2}{4 \cdot e} = 375\ mm$$

$$F_u = \frac{\tau_{max}}{\frac{d_2}{2} \cdot 10^{-3}} = \frac{1}{31{,}575 \cdot 10^{-3}} = 31{,}67N = F_{umax}$$

$$C_1 = \frac{z_1}{\pi} \cdot acos \frac{(z_2 - z_1) \cdot t}{2 \cdot \pi \cdot e} = 11{,}21$$

$$F'_{ureq} = \frac{F_{umax}}{C_1} = 2{,}83$$

$$F_v = \frac{F_{umax}}{2} = 15{,}84N$$

$$F_b = F_v + F_{umax} = 15{,}84 + 31{,}67 = 47{,}51N$$

Les courroies qu'on a acheté ont les trous très petits et est impossible de mettre les axes dans les trous. On doit agrandir les trous et avoir un ajustage serré.

La poulie petite est placée dans l'axe du moteur en utilisant une petit vis qui la serre et fa impossible la démonter. Au contraire, la poulie grande est placée dans l'axe du volant avec une vis qui traverse la poulie et les rondelles nécessaires et qui est mis dans un trou fait dans l'axe, ce poulie est possible le démonter si nécessaire.

Image 4 : Poulie petite

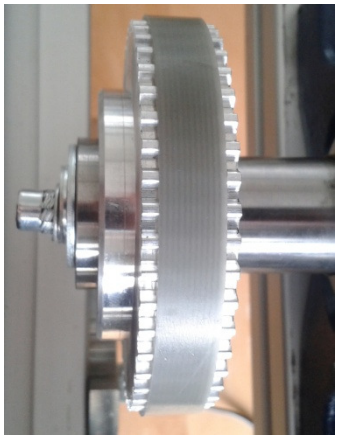


Image 5 : Poulie grand

3.4. Support moteur

Pour mettre le système de poulies dans le banc d'équilibrage on devra faire des changements en la collocation du moteur. Il ira sous la banque, parallèle à l'axe du disque, avec la courroie aligné avec le disque d'inertie. Pour faire ça, on a modifié la plaque initiale en faisant les trous plus larges pour la mettre dans la bonne position. La nouvelle plaque support sera:

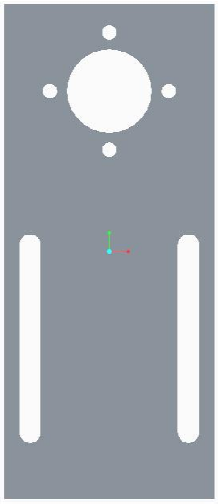


Image 6 : Plaque de support final[1]

Pour faire que la courroie de longueur 570 mm reste tendue, on a mis le moteur sous la banque, fixé avec le support à l'structure et un peu incline pour éviter le frottement avec les vis qui le fixent. On peut voir le montage final à l'image suivant:

[1] Voir les mises en plan à l'annexe I

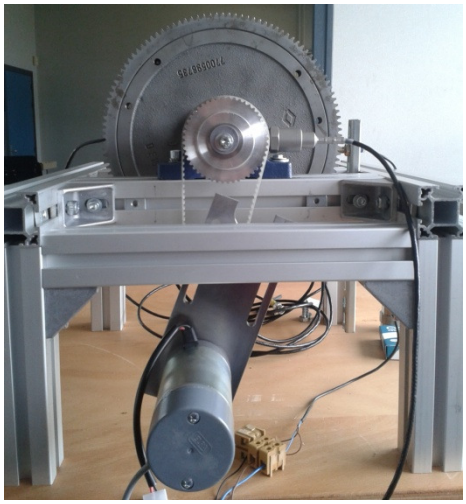


Image 7 : Emplacement final

3.5. Protection supérieur

Le moteur a la capacité pour faire rouler le disque à 500 rpm qui est une vitesse élevée et qui peut causer de mal s'il y a quelque problème et la roue perd la maîtrise de soi. Pour éviter ces complications on a réalise une carcasse de sécurité. Il est une caisse qu'on peut ouvrir pour un côté grâce à deux charnières (XDAH 110 A) qui se trouvent au latéral du banque.

La structure est faite avec le profilé de 44 x 44 mm d'aluminium (XCBL 3x44):

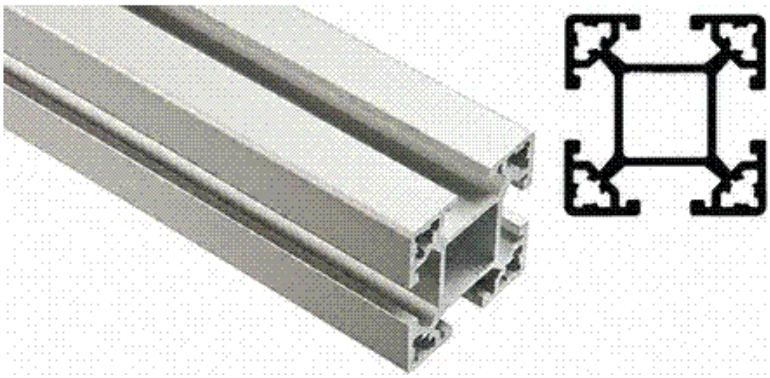


Image 8: Profilé de 44 x 44 mm

Entre les barres il y a une lamine de plastique de 6 mm qui est colloqué dans les rainures et fixé avec pièces de plastique. Pour faire l'union entre les barres on a utilisé des équerres (type XLFA 44B). Pour faire la structure on a utilisé 12 barres de profilé 44x44 mm. Il y a 4 bars de 200 mm de longueur, 4 de 300 mm de longueur et les 4 dernières sont de 430 mm. Voici la structure finale[2] :

[2] Voir les mises en plan à l'annexe I

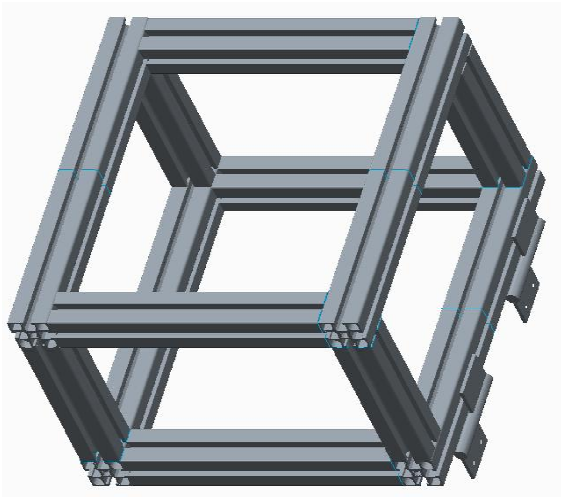


Image 9: Structure finale de la protection

Ce structure a été dessiné mais on ne l'a pas fini d'assembler (il a manquait quelques pièces qui n'ont pas arrivé après être commandées) et comme conséquence on ne l'a pas utilisé pendant les essais réalisées non plus.

4. Components de mesure

4.1. Capteur courant

Le capteur de courtant LA25-NP[3] c'est un capteur de courant de type fermé et d'effet Hall. L'effet Hall est l'apparition d'un camp électronique (V_H) dans un conducteur quand il est traversé par un courant (I) qui se trouve perpendiculaire à un camp magnétique (β). [8]

Pour déterminer quantitativement ce phénomène on utilise la résistance de Hall:

$$R_H = \frac{V_H}{I} = \frac{\beta}{\rho \cdot e \cdot c}$$

R_H: résistance de Hall
V_H: voltage de Hall
I: courant
β: camp magnétique
e: charge électrique
c: vitesse de déplacement de les charges libres
ρ: densité superficiel

Le capteur a 13 pins:

- 1-10: connexions qui dépendent du courent maxime qui circule pour le élément à calculer.
- +: 15V
- - : -15V
- M: Connecté a une résistance et ce résistance à la masse.

Comme le moteur a un courent maximum de 8,5A on utilisera la connexion suivant (2mA/1A) qui résiste un courant nominal de 12 A. Avec ce connexions on obtenu une bobine avec deux tours.

[3] Voire datasheet page 58 (Annexe III)

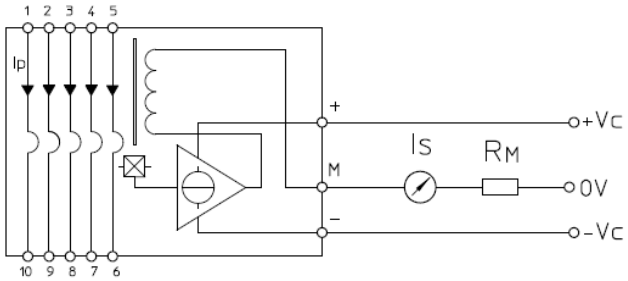


Image 10 : Schéma intérieur du capteur [8]

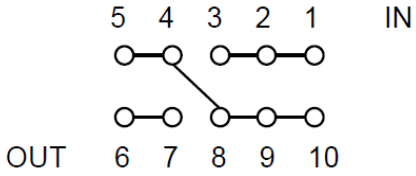


Image 11 : Schéma connexions initiales [8]

Une autre chose important est savoir quelle résistance doit aller entre le pin M et la masse (R_M). Implique le signal obtenu du condensateur de sortie pour obtenir la variable dVc/dt.

$$V_M = I_{pn} = R_m \cdot I_{sn} \quad \rightarrow \quad R_m = \frac{I_{pn}}{I_{sn}} = \frac{12}{24 \cdot 10^{-3}} = 500\,\Omega$$

Où les courants sont :

- I_{pn} : Courant nominal du primaire
- I_{sn} : Courant nominal de sortie

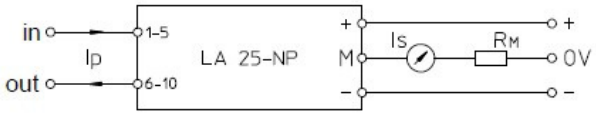


Image 12 : Schéma connexions capteur [9]

Pour mesurer le courant du moteur on doit mettre le IN au Pin 1 et le OUT au Pin 6. Avec ce connexion entre l'entré et la sortie il y a deux tours de bobine. Si on met le IN au Pin 5 et le OUT au Pin 10 il ne marche pas.

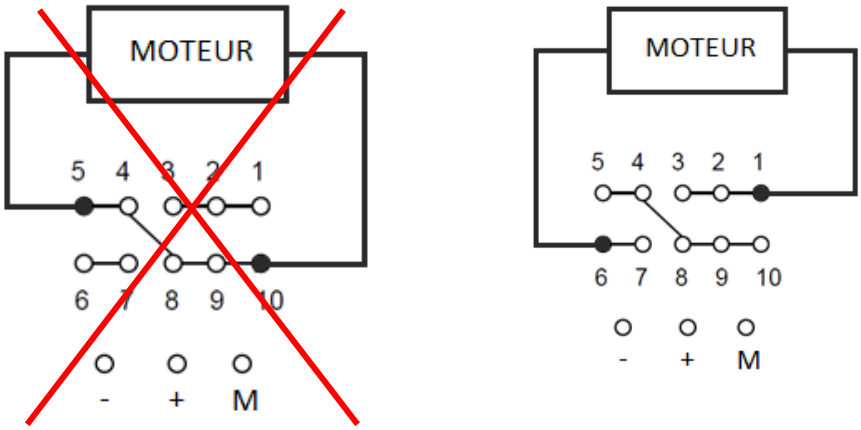


Image 13 : Changement des connexions

On doit mettre aussi comme avec le capteur de position un filtre passe bas pour avoir une meilleure image du signal avec le PicoScope au le LabView. Voici l'schéma du filtre passe bas [10]:

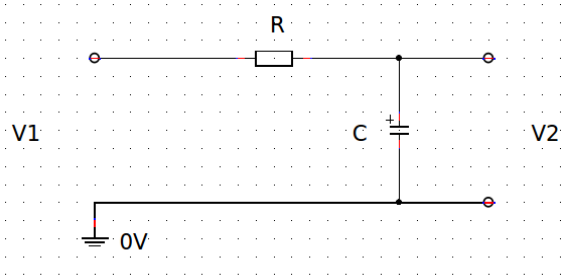


Image 14 : Schéma filtre passe bas

Les calcule pour trouver la résistance et le condensateur qu'on veut mettre pour filtrer à une fréquence de 50 Hz:

$$f = \frac{1}{2 \cdot \pi \cdot R \cdot C} = 50\ Hz$$

On utilise un condensateur de C = 100 · 10^{-9}

$$50 = \frac{1}{2 \cdot \pi \cdot R \cdot 100 \cdot 10^{-9}} \rightarrow R = 31830\ \Omega \cong 30\ K\Omega$$

$$w = 2 \cdot \pi \cdot f = 100 \cdot \pi = 314{,}16\ rad/s$$

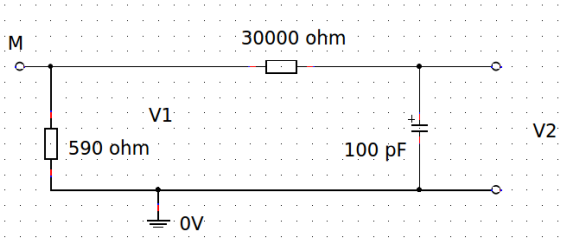


Image 15 : Schéma filtre final

Pour voir si on a obtenu la même lecture directe du courant avec le voltage de la sortie du capteur et après le filtre on analyse le diagramme de Bode.

$$\begin{cases} e = s + V_R \\ V_R = i \cdot R \\ i = C \cdot \dfrac{dU_c}{dt} = C \cdot \dfrac{ds}{dt} \end{cases}$$

$$\mathcal{L}\left[e = s + R \cdot C \frac{ds}{dt}\right]$$

$$E(p) = S(p) \cdot [1 + R \cdot C \cdot p]$$

$$\boldsymbol{\frac{S(p)}{E(p)} = \frac{1}{1 + R \cdot C \cdot p}}$$

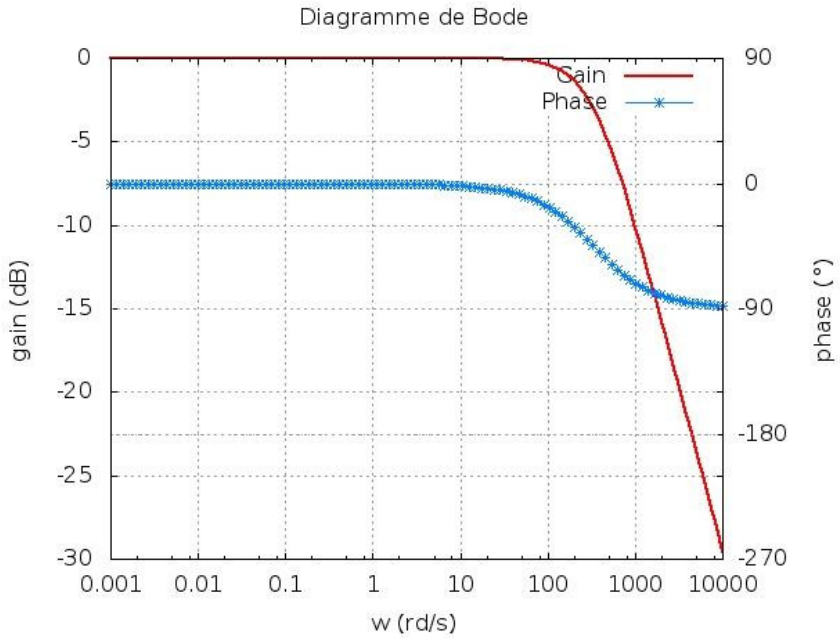


Image 16 : Diagramme de Bode

On peut observer qu'à la fréquence maxime de 50 Hz (314,16 rad/s) on a un gain de -2,5dB.

$$20 \cdot logx = -2{,}5 \rightarrow x = 0{,}75$$

Alors qu'entre les fréquences de 50 Hz et 16 Hz on a une petite variation du résultat avant et après le filtre, pour fréquences inferieurs à 16 Hz la lecture et la même à les deux côtés.

$$20 \cdot logx = 0 \rightarrow x = 1$$

Pour n'avoir pas problème de câblage on a créé une plaque imprimé avec le capteur de courant et des fiche bananes.

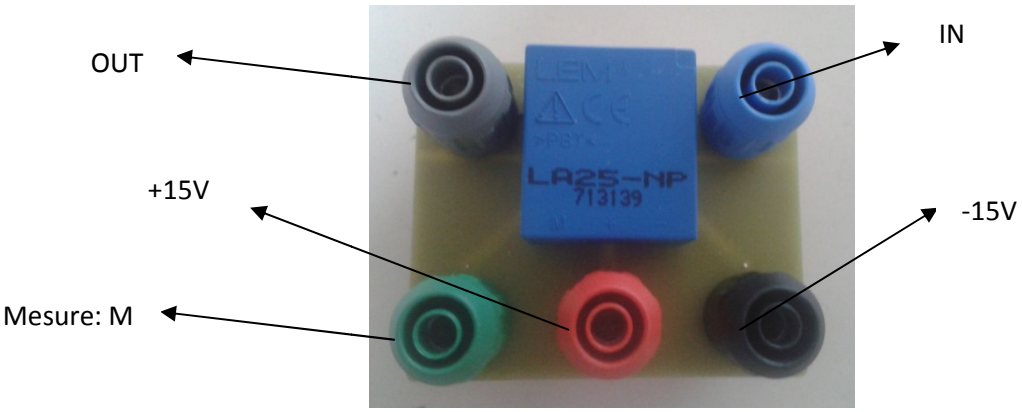


Image 17 : Plaque imprimé du capteur de courant LA25-NP

4.2. Capteur proximité

Le capteur de proximité inductif XS1N08PA340 c'est un capteur de proximité type PNP qui sert à détecter la présence ou non présence d'un matériel métallique à une distance de 1,5 mm. Son range de voltage pour le fonctionnement est de 12 à 24 volts avec une fréquence d'opération de 5000 Hz, ça veut dire que sa résolution est de 0,2 ms. [11]

Il a 3 câbles, deux sont pour l'alimentation et le troisième est qui donne l'information reçu et que dans ce cas il est connecté à l'Arduino pour lire les données. L'Arduino ne peut pas lire ces voltages, ils doivent être plus bas alors, on doit changer le voltage du pin qui est connecté avec la plaque pour pouvoir lire ce valeur, on a mis des résistances pour diviser le voltage comme est expliqué ci-dessous :

$$I = \frac{24V}{1440\Omega} = 0{,}0166\ A$$

$$V_A = 0{,}0166 \cdot 440 = 7{,}33\ V$$

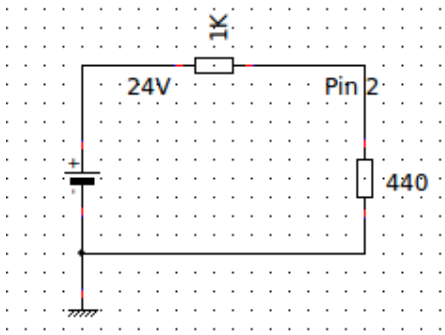


Image 18 : Schéma diviseur voltage

Maintenant, au point 2 on a 7,33 V qui est une valeur que l'Arduino peut lire et traduire comme 0 ou 1. Dans ce projet, on veut utiliser ce composant pour calculer la vitesse de rotation du disque d'inertie et aussi savoir la position dans laquelle il se trouve. Pour faire ça, on a utilisé le programme ci-dessous :

```
const int hallPin = 7;
int hallState = 0;
int hallState0 = 0;
float temps0 = 0;
float temps1 = 0;
float v = 0;
int N = 0;

void setup(void) {
        Serial.begin(19200);
        pinMode(hallPin, INPUT);
}
void loop(){
  while(N<80){
        hallState = digitalRead(hallPin);
        temps1 = millis();
        if(hallState != hallState0){
          N = N + 1;
          hallState0 = hallState;
        hallState0 = hallState;
        }
  }
  v = ((3.1416*1000)/(temps1-temps0));
  w= ((v*60)/(2)) ;
  Serial.println(w);
  N = 0;
  temps0 = temps1;
}
```

La manière de calculer la vitesse est calculer les changements que le capteur détecte (il lit un 1 s'il détecte présence et 0 autrement). Le volant d'inertie a 40 dents, pourtant le capteur détecte 80 changements (de 0 à 1 quand il commence a voir une dent et de 1 à 0 au final), quand il a détecté tous les 80 changements ça veut dire que le volant a fait un tour complète et pourtant on peut calculer la vitesse de rotation comme:

$$w = \frac{(1000 \cdot \pi)\ \cancel{rad}}{(t_1 - t_0)\ \cancel{s}} \cdot \frac{1\ rev}{2 \cdot \pi\ \cancel{rad}} \cdot \frac{60\ \cancel{s}}{1\ min} = \frac{30000}{(t_1 - t_0)}\ rpm$$

Le problème de ce montage est la limitation de vitesse, il peut lire comme maximum 264 rpm. Ce problème n'est pas du capteur ou du programme, c'est de la résolution de l'Arduino qui seulement peut lire informations chaque 1ms et les données que le capteur envoie sont plus rapides. Pour cette raison, le capteur ne peut être utilisé pour grandes vitesses et il faut trouver une solution.

La solution trouvé était faire des interruptions chaque fois qu'il y a un changement de 0 a 1 (*rising*), de ce manière le *loop* du programme ne doit pas lire autant de donnés que avant et il peut

aussi aller plus vite parce qu'on a éliminé le *while*, le compteur est maintenant l'interruption et le programme est :

```
//Capteur proximite
const int hallPin = 2;
#define codeurInterruption 0
int hallState0 = 0;
float temps0 = 0;
float temps1 = 0;
float v = 0;
int N = 0;
int hallState = 0;

// Initialisations capteur
void setup(void) {
  pinMode(hallPin, INPUT);
  attachInterrupt(codeurInterruption, GestionInterruptionHallPin, RISING);
}

// Boucle principale
void loop(){
  if (N>20){
    temps1 = millis();
    v = ((3.1416*1000)/(temps1-temps0));
    N=1;
    Serial.println(v);
    temps0 = temps1;
  }
}

// Routine de service d'interruption attachée à la lecture du capteur
void GestionInterruptionHallPin(){
  N = N + 1;
}
```

Maintenant, dans les données il y a du bruit et le signal n'est pas la correcte. À une vitesse haute mais ne pas la maxime on trouve beaucoup d'interférences et à une vitesse baisse il y a moins mais elles restent :

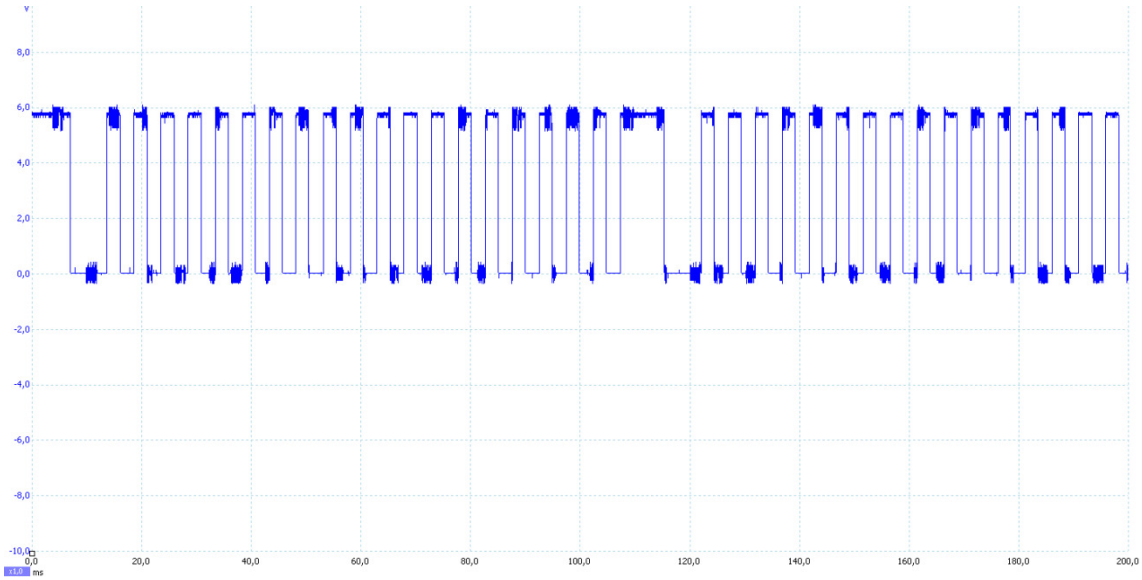


Image 19 : Signal du capteur dans le PicoScope (368 rpm)

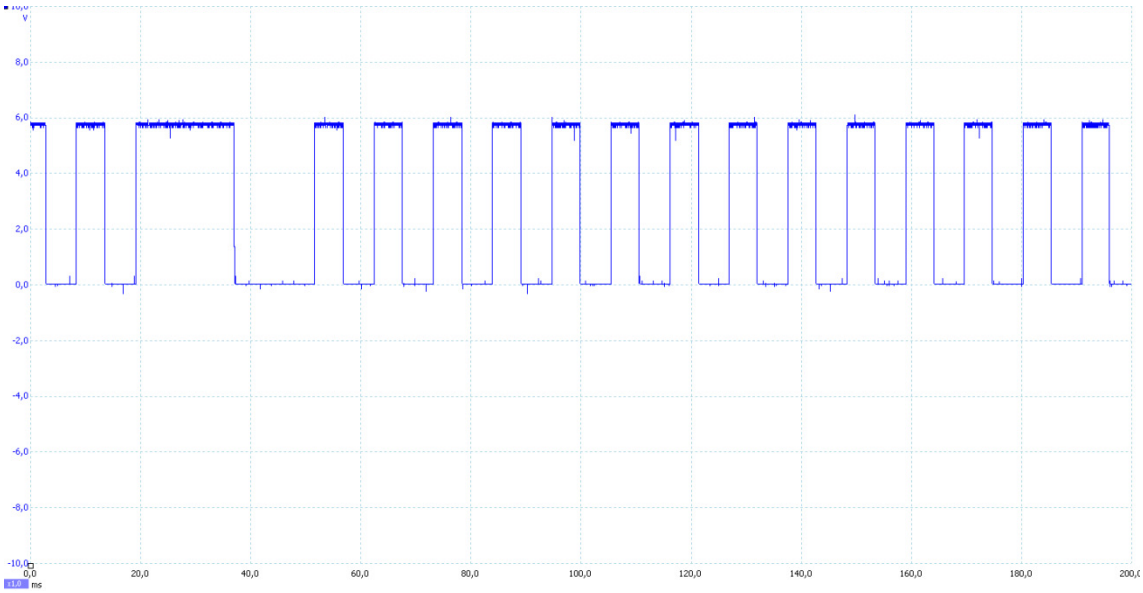


Image 20 : Signal du capteur dans le PicoScope (223,84 rpm)

La solution a ce problème a été résolu en posant des condensateurs à le câble qui est la sortie de donnés du capteur, de ce manière on a réduit le bruit et la captation est beaucoup meilleur.

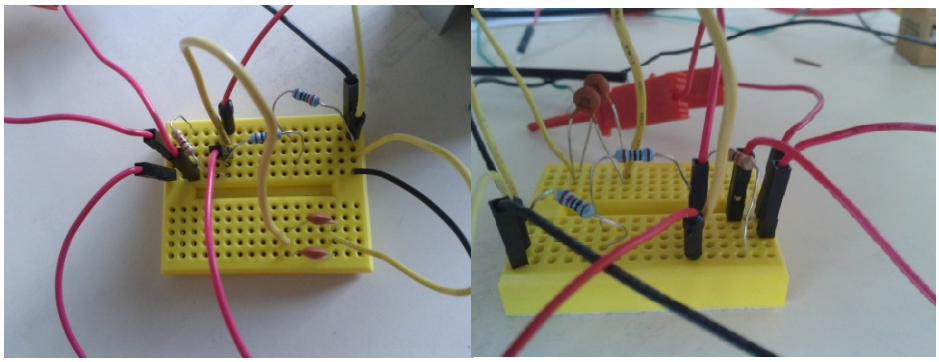


Image 21 : Connexions filtre

Finalement, les condensateurs choisis ont été deux d'une valeur de 100pF mis en parallèle. Si on pose une valeur de condensateur plus grand, le signal n'est pas bien lit parce qu'il n'a pas du temps pour se charger et se décharger. On peut voir a continuation les calculs faits pour avoir la meilleur filtration et les images de les filtrations avec condensateurs de valeur trop grand :

$$f_c = \frac{1}{2\pi RC} \qquad 55\text{kHz} = \frac{1}{2\pi RC} \qquad RC = \frac{1}{55 \cdot 10^3 \cdot 2\pi} = 2{,}89 \cdot 10^{-3}$$

R= 1440 Ω
C= $2 \cdot 10^{-10}$

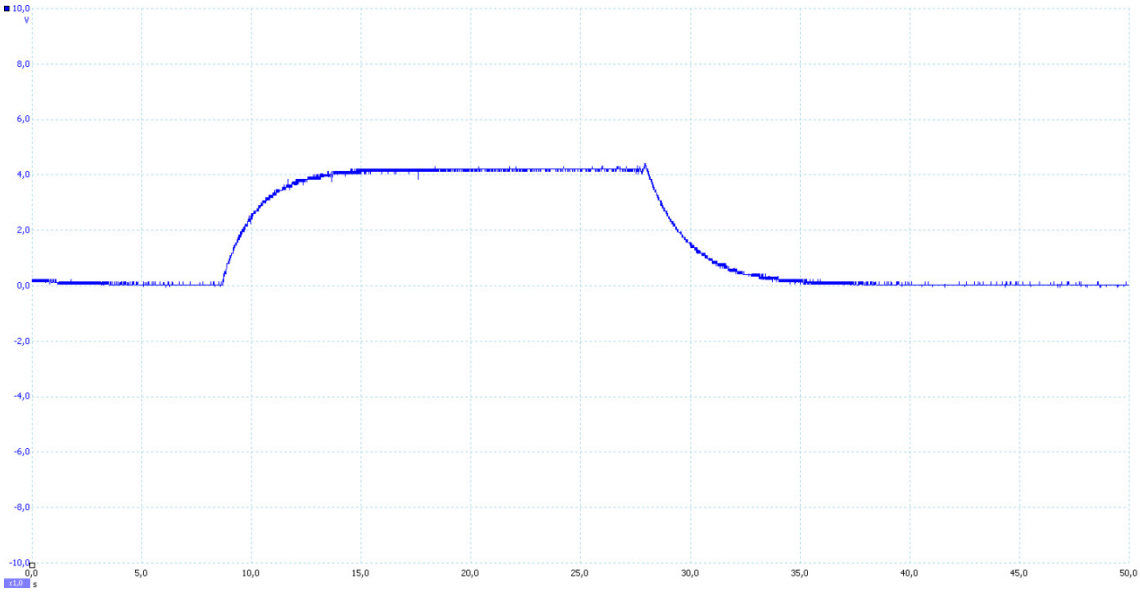


Image 22 : Signal du capteur dans le PicoScope avec condensateur de 3300µF (223,84 rpm)

Ici en 20 seconds on seulement voit un changement, on ne peut pas voire les changements réels parce que le condensateur est trop gros et la charge et décharge est très lente.

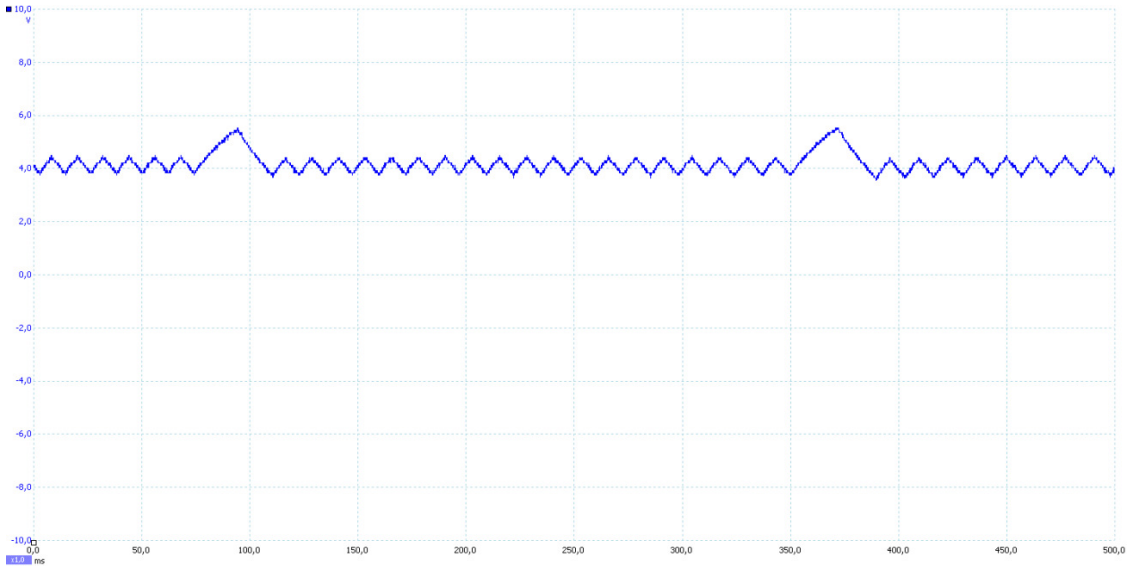


Image 23 : Signal du capteur dans le PicoScope avec condensateur de 100µF (368 rpm)

Avec ce condensateur, on peut voire les changements mais le signal n'est pas la bonne parce que il est encore trop gros et les changements ne sont pas assez rapides.

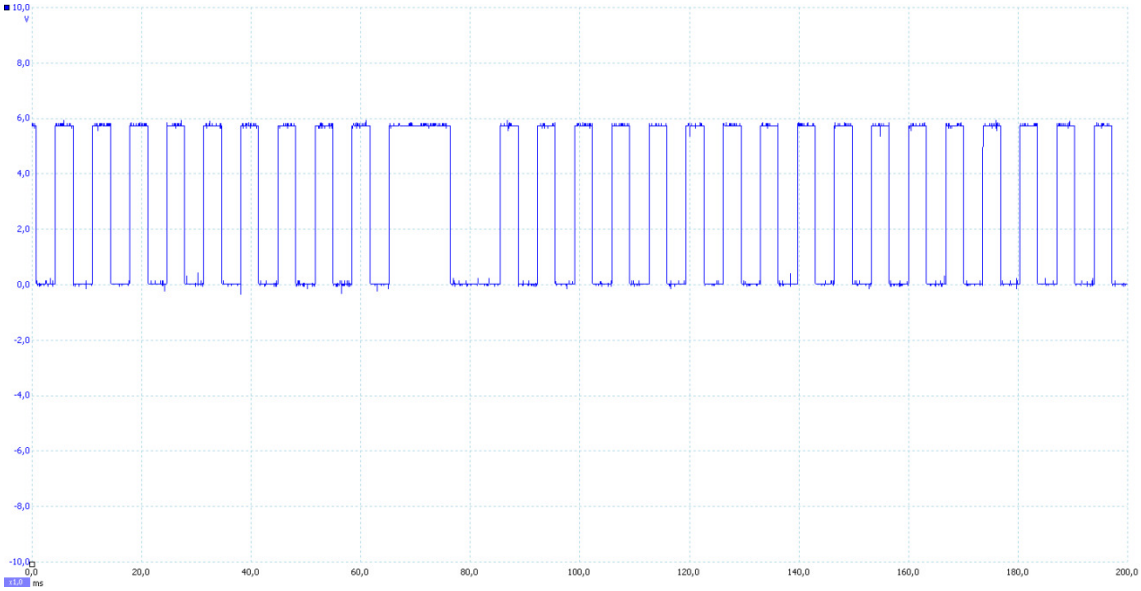


Image 24 : Signal du capteur dans le PicoScope avec deux condensateurs de 100pF (368 rpm)

Ici, avec une vitesse intermède on voit un signal assez bonne, il y a encore un peu de bruit mais il est très bas et le changement est aussi bien enregistré.

4.3. LED

Le PicoScope qu'on utilise a seulement deux entrées analogiques. Comme on les veut utiliser pour mettre les deux accéléromètres, on doit chercher une manière pour savoir la position avec un signal digital. Ce qu'on a décidé de faire est mettre un LED qui s'allume quand le capteur de proximité passe pour les deux dents larges.

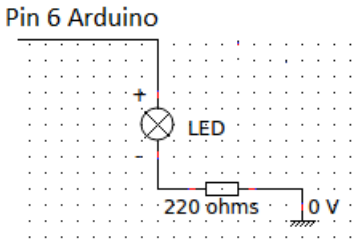


Image 25 : Schéma montage du LED

Les commandes qu'on a incluses dans le programme sont [2]:

```
const int led = 6;
// Initialisations
void setup(void) {
  pinMode(led, OUTPUT);
}
// Boucle principale
void loop(){
  if (N= =20){
    digitalWrite(led, HIGH);
    }
  else{
    digitalWrite(led, LOW);
  }
}
```

Dans la boucle principale on mit deux conditions. La première dit que quand le capteur de proximité compte la dent numéro vingt le LED doit s'allumer. Alors que, pendant le reste de temps il doit être éteint. L'utilisation d'un compteur permet changer la vitesse et aussi arrêter le moteur sans perdre l'information du signal digital du LED. Si on redémarre l'Arduino on doit placer la rue à la position qu'on veut pour compter bien les vingt dents, cet est a une de les deux lents larges. Comme on veut faire des preuves pour faire l'équilibrage dynamique, le fait de mettre on place la roue chaque fois que on arrêt la machine n'est pas un problème.

En ces deux images suivantes on peut voir ce qu'il passe quand la roue à une bonne collocation quand le capteur commence à compter et quand la collocation n'est pas la bonne.

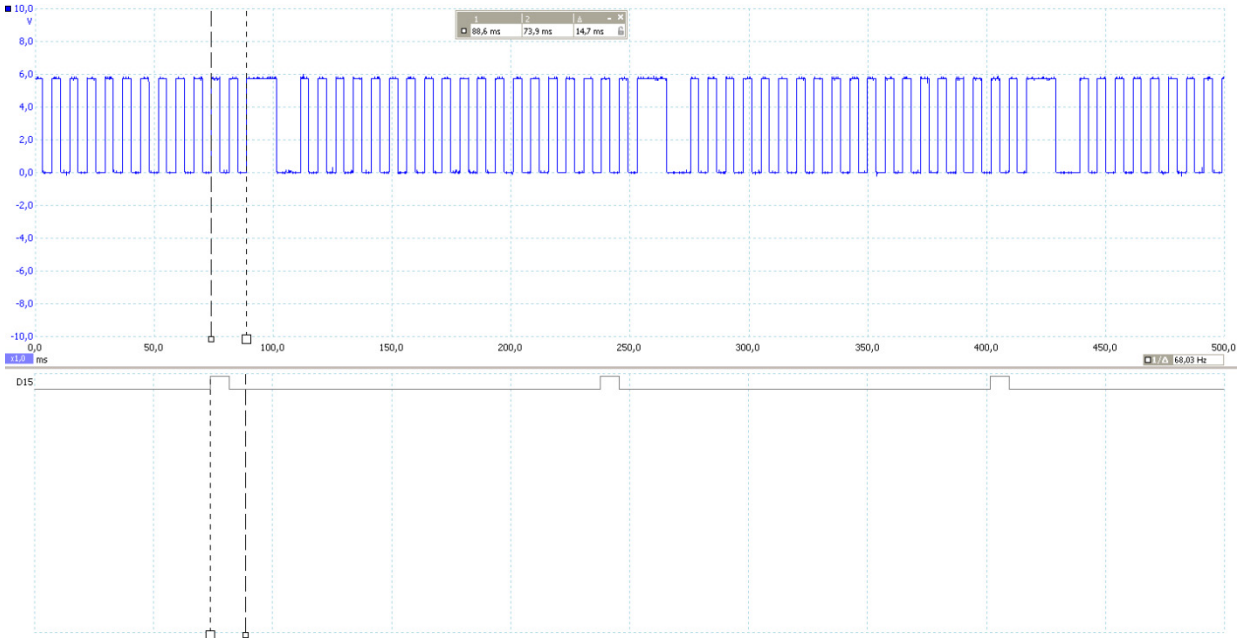


Image 26 : LED déphasé

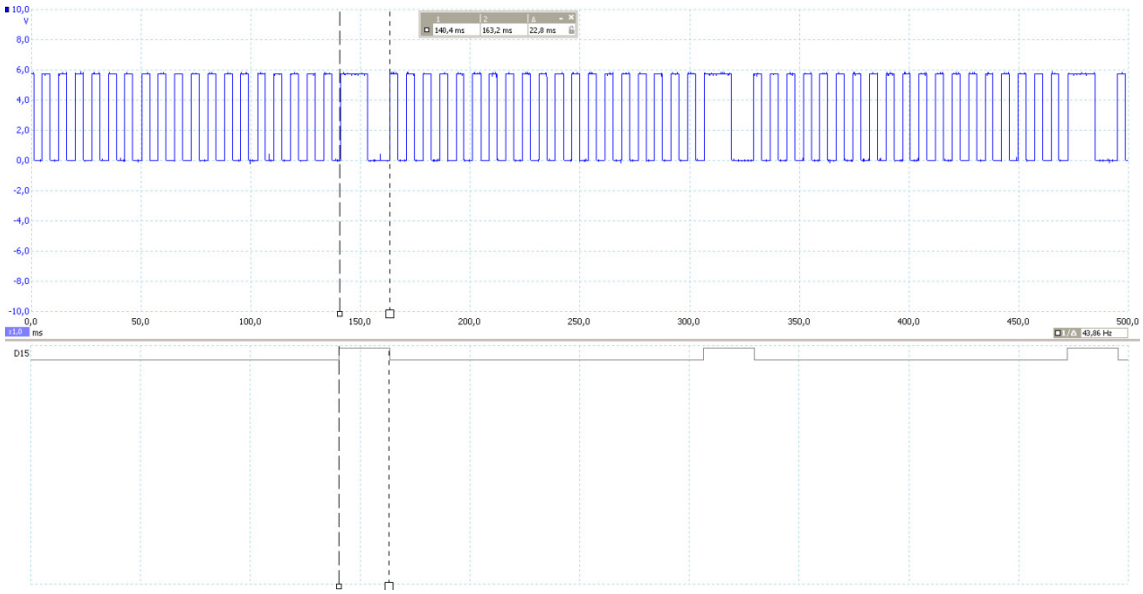


Image 27 : LED ne pas déphasé

4.4. Bouton arrêt d'urgence

Pour arrêter le système immédiatement en cas d'urgence on a mis un bouton d'arrêt pour le moteur. Il est connecté à l'Arduino et dans le programme on a ajouté une petite interruption qui est lancé quand le bouton est appuyé comme on peut voir ci-dessous [2]:

```
#define motorInterruption 1
void setup(void) {
  attachInterrupt(motorInterruption, arretEmergence, HIGH);
}
void arretEmergence(){
  nombreReception = 0;
  long nombreRecu=0;
  nombreRecu = 0 ;
}
```

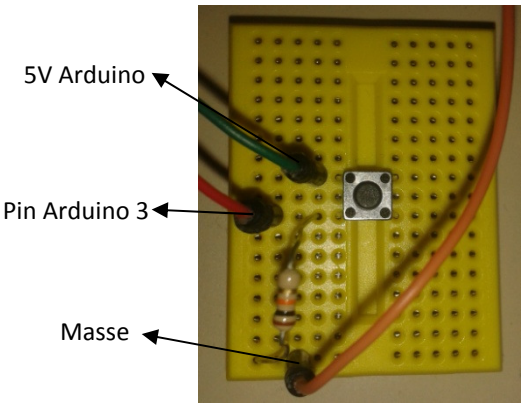


Image 28 : Connexions bouton arrêt émergence

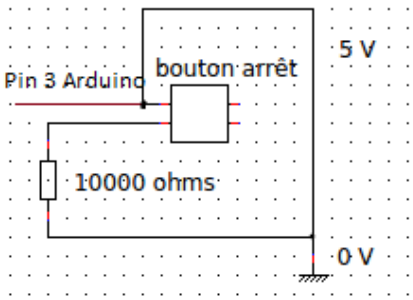


Image 29: Connexions bouton arrêt émergence

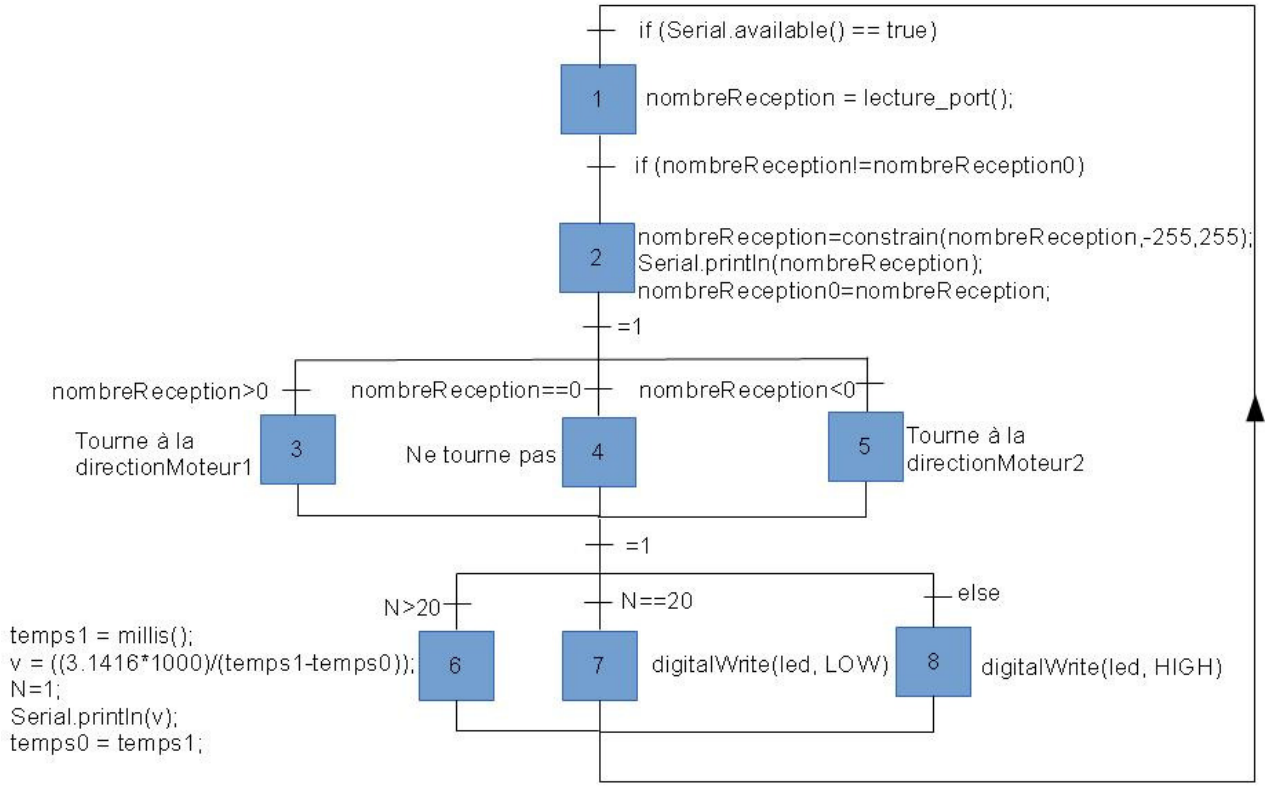
5. Banc définitif

5.1. Programme définitif

La version définitive du programme permet du réguler la vitesse du moteur avec le moniteur série. Il aussi calcule et montre la vitesse à la que roule le disque grâce à le capteur de proximité. Le LED et le capteur de proximité aident à savoir la position des dents grandes. Pour finir le bouton d'arrêt peut arrêter complètement le système en cas d'émergence.

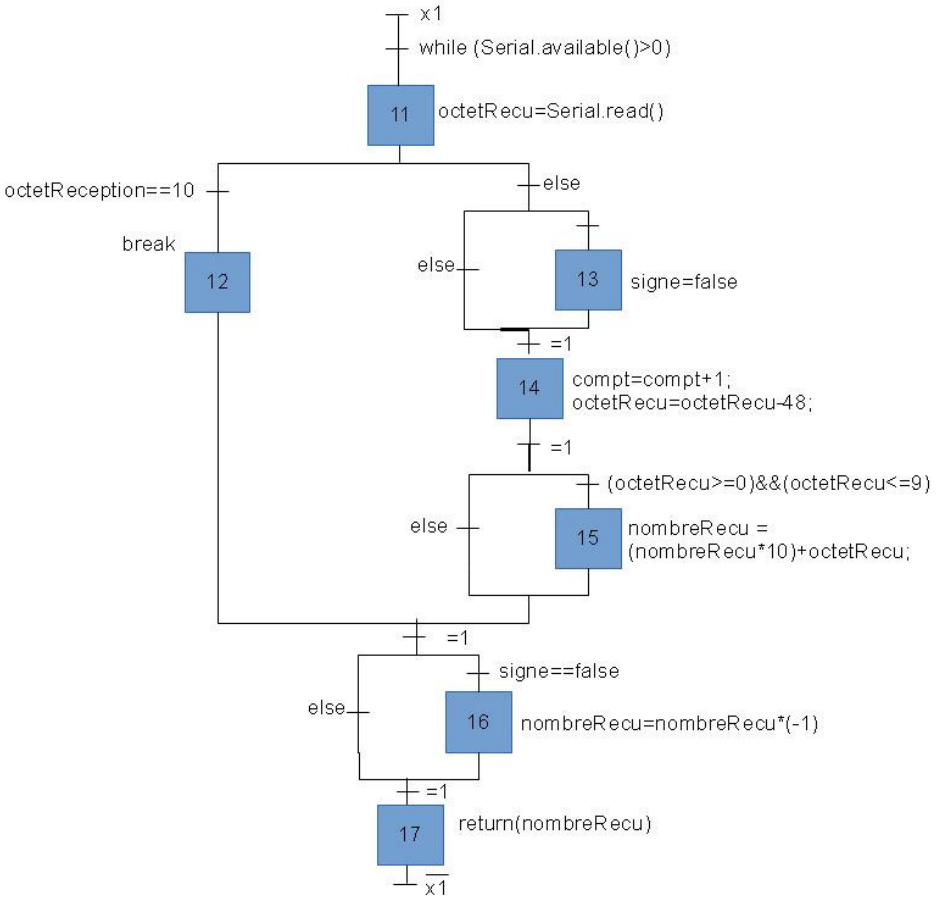
Voici l'organigramme du programme définitif[4]:

loop()

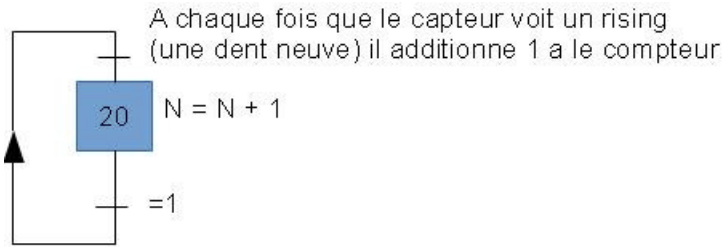


[4] Voire code programme définitif dans l'Annexe II

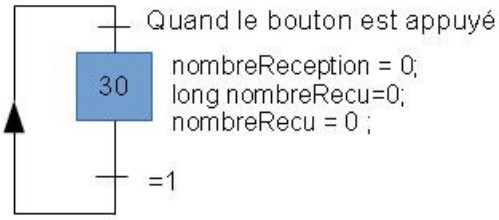
lecture_port()



gestionInterruptionHallPin()



arretEmergence()



5.2. Montage définitif

Voici le schéma final après l'incorporation du capteur de proximité et son filtre, un LED et un bouton d'arrêt.

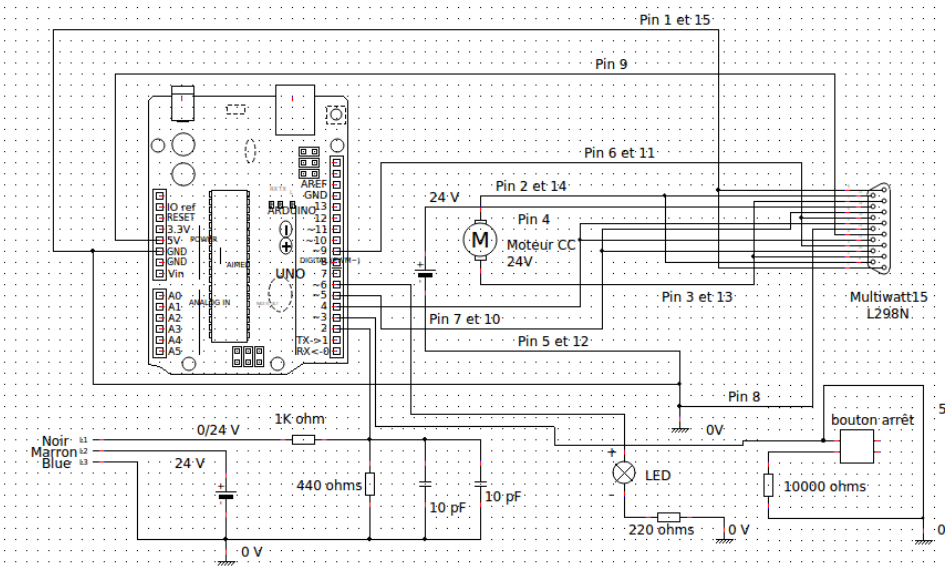


Image 30: Schéma définitif du montage électrique

6. Équilibrage

6.1. Équilibrage statique et dynamique

Le déséquilibrage statique est celui qu'on peut trouver quand on place le disque d'inertie entre deux barres parallèles et on la laisse tourner jusqu'à il s'arrête. La partie avec plus de masse du montage du disque se trouvera à la base et la plus léger à la partie supérieure du rotor. Si lui a seulement ce type de déséquilibrage c'est-à-dire qu'il a l'axe de rotation et d'inertie séparés mais parallèles. Pour compenser ce déséquilibrage on peut mettre une masse a chaque côté et en sens contraire à ce problème ou une masse de valeur l'addiction de tour le déséquilibrage statique à le centre du rotor. [12] Avec une petite expérience on peut déterminer si l'axe a quelques problèmes de compensation. On fait roter le disque d'inertie doucement et le laisse arrêter, après on fait un petit signal à le point plus bas. On répète ça quelques fois, si tous les signaux se trouvent à la même partie du disque on a déséquilibrage statique, si par contre ils ont en différents n'y a pas déséquilibrage. Les signaux indiquent la position angulaire du déséquilibrage en ce qui concerne le système xy. [13]

On a réalisé cette petite expérience pour tester si le moteur a des problèmes d'équilibrage statique et on a vérifié qu'il n'y a pas. Les signaux ce trouvent en différents point de le disque.

Bien que le disque d'inertie soit équilibre statiquement il peut avoir des problèmes de compensation dynamique important et provoquer des vibrations. Le déséquilibrage dynamique est lequel que apparaît quand le disque est en rotation. Pour lui détecter on doit faire une mesure "in situ" ou mettre le disque dans une machine d'équilibrage. La complication est que après la compensation

statique on a des masses sur le disque et elles provoquent des forces et ces forces un coup de force. Ça implique le mouvement des axes de rotation et d'inertie qu'on a déjà corrigé. Pour ça toujours si on fait un équilibrage statique après on doit faire le dynamique ou en deux plans. Par contre on peut faire seulement le dynamique parce que il inclure le statique.

6.2. Software utilisé

6.2.1. LabView

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) est une plate-forme de conception de systèmes de mesure et de contrôle appartenant à National Instruments. Est utilisé principalement pour la mesure par acquisition de données et s'exécute sous différents systèmes d'exploitation comme Microsoft Windows, Linux et Mac OS X. Le langage graphique utilisé dans cette plate-forme est appelé "G". Le code est déterminé par un diagramme, le programmeur connecte les différentes fonctions sous forme d'icônes avec des fils dessinés entre les terminaisons des blocs. Chaque fil propage les variables du code en permettent l'exécution de code en multitâche. Dans le diagramme on peut aussi modifier et manipuler le signal saisi pour voir dans le graphique ce qu'on veule. [14]

Dans ce cas en particulière on veule capter l'information des accéléromètres. Ce sera nécessaire un processeur temps réel pour le contrôle, l'enregistrement de données et l'analyse, particulièrement le NI cRIO-9073 qui est connecté à l'ordinateur grâce à un réseau croisé. N'oublier pas d'alimenter le LabView cRIO-9073 à 24V. Dans ce processeur on doit mettre les modules NI 9211 et NI 9205 qui sont utilisés pour lire les donnés analogiques et digitaux, le module NI 9402 qui sert à lire les signaux analogiques et le NI9232 qui sert à connecter les accéléromètres. Ce dernier est un module FPGA, plus efficient et rapide que les autres mais aussi plus complexe pour le connecter avec l'ordinateur parce que la connexion n'est pas directe, on doit aussi installer un logiciel additionnel.

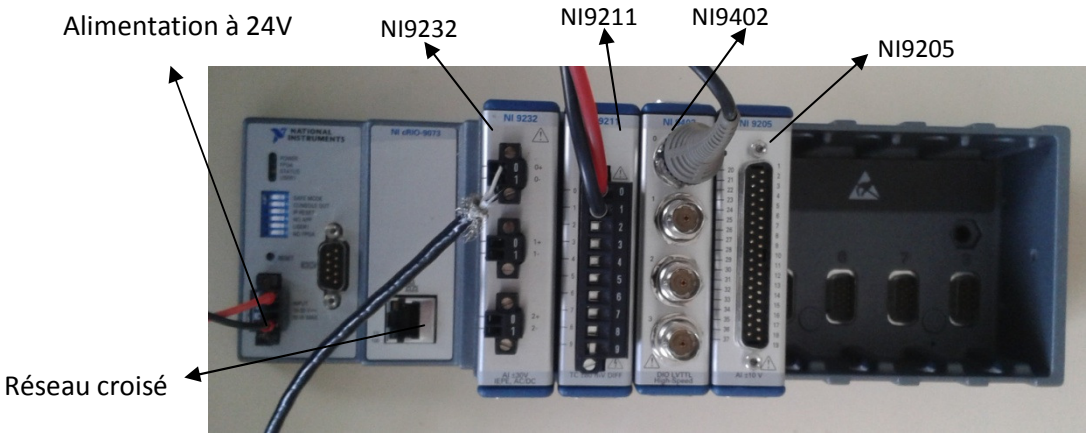


Image 31 : NI cRIO-9073n avec les modules

6.2.2. PicoScope

Le PicoScope est un programme qui fait la fonction d'oscilloscope. Un oscilloscope est un instrument de mesure qui permet représenter les signaux électriques analogiques et digitaux. On utilisera un PicoScope de Mixed Signal Oscilloscope avec 2 entrées analogiques et 16 digitales. [15]

Premièrement, on a utilisé les deux entrées analogiques. Une pour connecter le LED que s'allume quand le capteur se trouve sur les dents larges en permettant savoir la position où on se trouve. Sur l'autre, on a connecté le capteur de présence qui permet voir touts les changements et pourtant toutes les dents. Avec ce deux signales, on peut voir si le LED fonctionne bien puisque sa signal doit coïncider avec la que donne le capteur.

Une fois cet est correct, les 2 entrées analogiques on les a utilisés pour connecter les accéléromètres. Dans les digitales on a utilisé une entré dans laquelle on a connecté le LED et une autre avec le capteur. Ces signaux permettent étudier les problèmes d'équilibrage et leur résoudre. [16]

Pour éviter problèmes à cause de les interférences, les perturbations et le bruit que le câblage peut provoquer, on utilise deux alimentations de 24 V différents, une pour alimenter le capteur de proximité et l'autre pour le moteur et le composant multiwatt15. De plus, on fabrique une plaque imprimé pour le composant pour réduire aussi le numéro de câbles utilisés et assurer une bonne connexion.

Voici l'image des fiche banane:

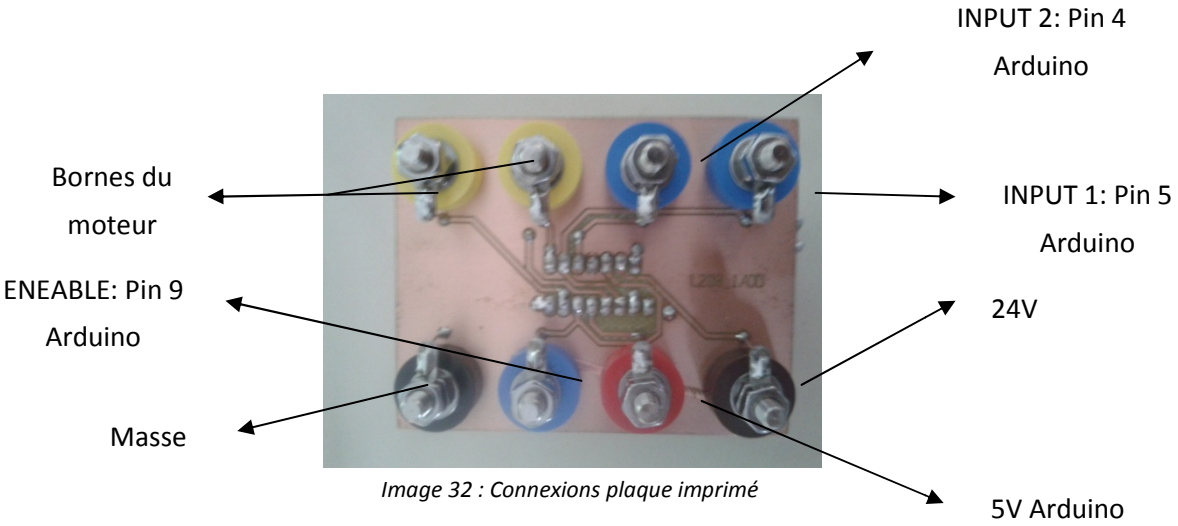


Image 32 : Connexions plaque imprimé

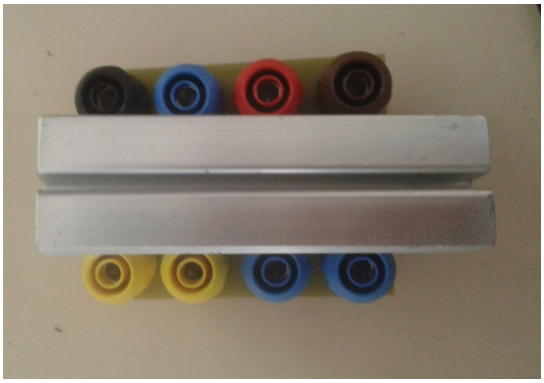


Image 33 : Plaque imprimé du Multiwatt15 avec un radiateur

6.3. Accéléromètre

6.3.1. Accéléromètre Brüel & Kjaer type 8325

Pour réaliser l'équilibrage du disque on doit utiliser des accéléromètres. On utilisera un accéléromètre Brüel & Kjaer type 8325. Cette absolu-vibration piézoélectrique mesure accéléromètre fait partie d'une famille de capteurs industriels robustes et il se sert pour mesurer de vibrations. Le 8325 est un accéléromètre robuste avec une grande sensibilité, parfaitement adapté pour la vibration permanente et *off-line* la surveillance d'une grande variété de machines. Et à cause de sa *currentdrive* intégrante amplificateur, il offre une excellente solution de prix attrayant pour le grand-systèmes multicanaux usage. [17]

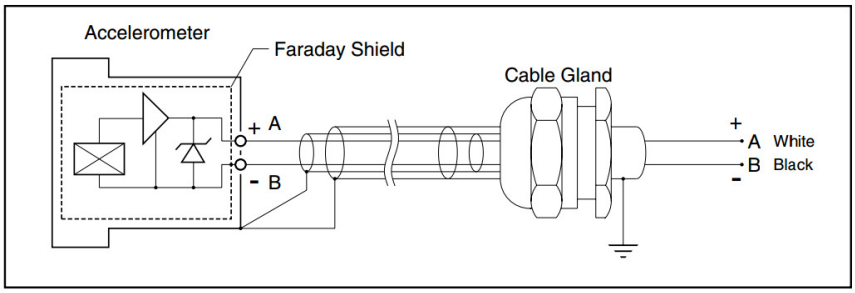


Image 34 : Intérieur accéléromètre [18]

Les accéléromètres doivent être placés parallèles à la terre et perpendiculaire à l'axe du disque. On a fait des trous dans les supports pour mettre les accéléromètres. Les accéléromètres permettent mesurer l'accélération quand il y a une force centrifuge due à une masse placé sur le disque.

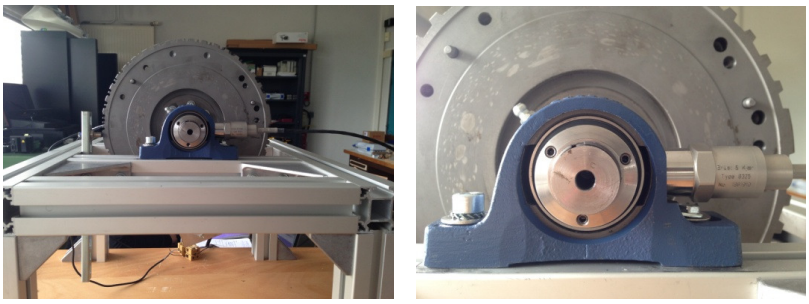


Image 35 : Montage accéléromètres

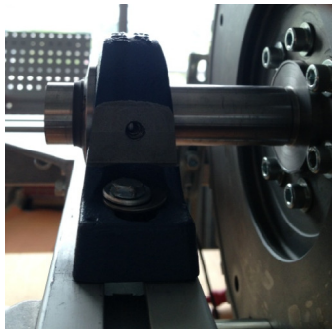


Image 36 : Trou pour l'accéléromètre

On a utilisé le PicoScope pour réaliser les premières preuves de l'accéléromètre. On a mis une masse de 2N au disque pour voir si l'accéléromètre détecte ce déséquilibrage. On a vu que le PicoScope ne voit pas l'accélération de la masse. Il seulement détecte les coups que on fa avec un objet métallique contre le support du banc (le support est aussi métallique) et aussi quand on frôle le support avec un objet métallique.

Les images acquis avec le PicoScope montrent le signal de l'accéléromètre. Les changements de plus voltage sont la réponse aux coups qu'on fa sur le structure métallique et les plus bais sont les frôlements mais ce n'est pas ce qu'on veut observer.

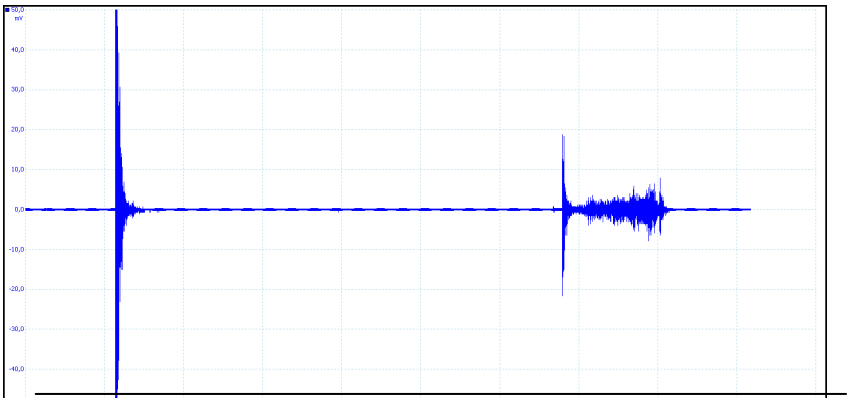


Image 37 : Signal de l'accéléromètre dans le PicoScope

De cet image on peut conclure que l'accéléromètre ne marche pas bien parce qu'il seulement détecte des coups forts qu'on provoque extérieurement. Pour solutionner ce problème, une alimentation de courant constant entre 2 et 20 mA est nécessaire. Ce courant doit être alimenté par un voltage entre +22 et +28 V. Comme cet alimentation n'est pas compris en l'intérieur de l'accéléromètre, on doit créer une font extérieur qui fourni ce courant.

6.3.2. Générateur de courant

On doit créer un générateur de courant pour alimenter les accéléromètres avec 4mA. On a essayé de le faire avec un amplificateur opérationnel (A.O.). Premièrement on a utilisé l'A.O. INA114AP [19] :

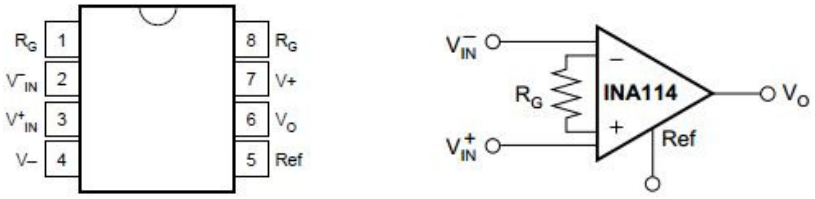


Image 38 : Connexions A.O. INA114AP [19]

L'amplificateur n'est pas un simple, il y a trois A.O. connectés entre eux dans le composant comme on peut voir ci-dessous:

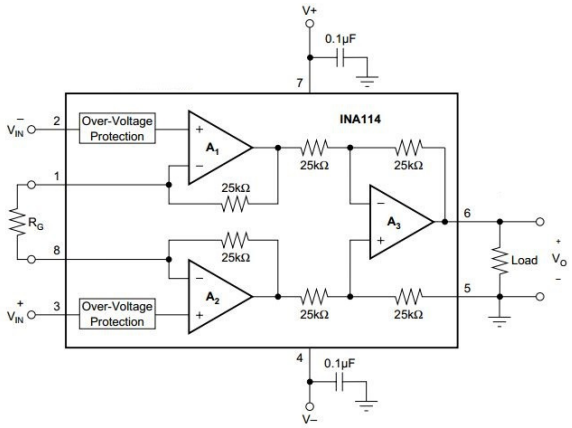


Image 39 : Intérieur A.O. INA114 [19]

Il y a plusieurs montages pour créer un générateur de courant. Les deux plus simples de faire avec un A.O. sont les suivants [20] :

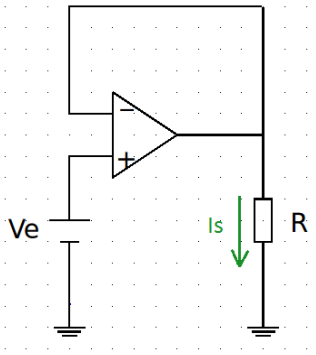


Image 40 : Configuration 1, seguidor

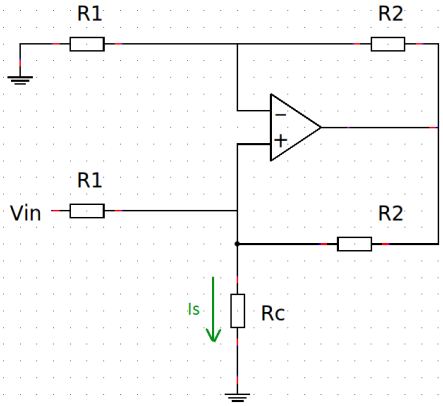


Image 41 : Configuration 2, Générateur Howland

Pour obtenir l'intensité désirée, on a mis, dans le première montage une résistance de 1kΩ et un voltage de 4V à V_e (V^+_{IN}). Après, on a connecté V- et V+ a -15V et +15V pour alimenter le composant, Ref et V^-_{IN} à la masse et rien connecté à R_G pour avoir un gain unitaire. Ça n'a pas marché, on a obtenu un voltage constant à la résistance mais pas une intensité constante. On a essayé le deuxième montage en faisant les mêmes connexions mais il n'a pas marché non plus.

En pensant que le problème pouvait être du composant, on a changé cet et a essayé les deux montages avec les A.O. TL081CP et LM324N mais le résultat n'a pas été bon. Avec la configuration 1, utilisant le première composant on a obtenu un voltage à la R plus petit de la Ve mais qu'il change si on change Ve et avec le deuxième, le résultat a été d'un voltage à la R plus grand qui varie aussi. Avec la configuration 2, avec les deux composants on a obtenu un voltage fixé mais d'une valeur incorrecte.

Une autre option pour obtenir la font de courant peut être utiliser un régulateur de tension. On a le composant LM338K mais toutes les configurations trouvées sont pour obtenir une intensité beaucoup plus grande que les 4mA désirées. [21]

Par conséquent, on doit changer le composant ou bien le montage. Finalement, on a découvert que le problème était l'alimentation du composant. En utilisant la première configuration et l'A.O. TL081CP, on a alimenté à 24V le composant, a mis aussi 24V à Ve et une R de 6 kΩ [22]. De cette manière quand on a branché l'accéléromètre à continuation de R, on a obtenu les 4mA désirées (on a 24V en bornes de la résistance).

Le point plus important à considérer en utilisant des A.O. est l'alimenter avec une tension plus haute que la tension d'entré parce que sinon il se sature et ne marche pas bien.

6.4. Résultats

6.4.1. Premiers résultats

Une fois l'accéléromètre est bien alimenté avec le générateur de courant, on peut commencer à fer l'équilibrage. Pour commencer, on a utilisé le PicoScope pour voir si le signal est la bonne puisque est plus simple que le LabView. Pour ça, on doit vérifier que la tension qu'il y a aux bornes de l'accéléromètre peut être supporté par le PicoScope (maximum 20V) et aussi on doit modifier l'échelle pour avoir le signal en 0 et pouvoir observer meilleur la variation. À l'image suivant on peut voir le signal bleue qui est l'original et qui est a 12V et le signal verte qui est la même mais centré en 0.

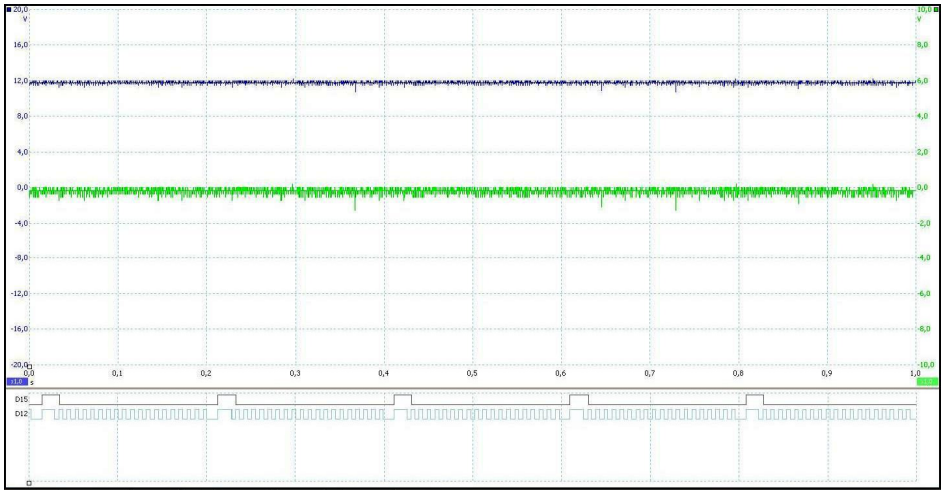


Image 42 : PicoScope avec le signal original (bleue) et la modifié (verte).

Comme on peut observer dans l'image antérieur, le signal n'est pas très claire et pour pouvoir l'étudier meilleur, on connecte l'accéléromètre au LabView. L'étude avec LabView est très complexe et avec l'aide d'un professeur, on a obtenu un programme (*test-9232-rt-level.lvproj*) pour voir le signal. Ce programme nous montre le signal directement dans l'écran et on doit seulement s'assurer qu'on voit le signal *DC coupled* (option pour étudier le courant continu) et que les échelles sont les bonnes (que l'axe de les Y est l'approprié pour le signal et que dan l'axe de les X on voit des tours complètes du volant). Dans l'image ci-dessous on peut observer le signal obtenu à 180 rpm (155 au terminal Arduino) :

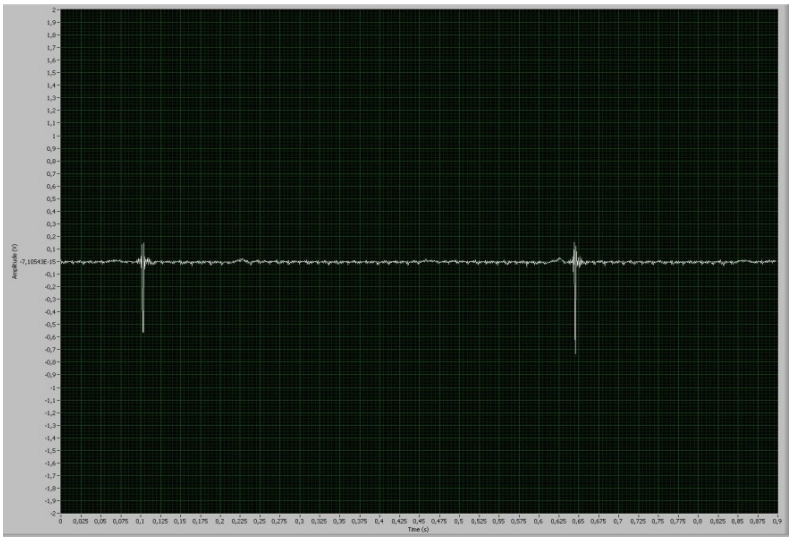


Image 43 : Signal à 180 rpm

Ce signal ne correspond pas à laquelle on doit obtenir, on voit une variation mais sans un patron de répétition et sans la détection de la masse ponctuel non plus. Dans l'image on voit 1s, à 180 rpm on devrait observer trois changements, un à chaque tour, causé par la masse mais à chaque fois on observe différents variations du signal. Il y a des fois qu'on voit 2 pics, d'autres 6, d'autres aucun, etc. Pour savoir si le problème viens de que la variation seulement est détecté pour plus hautes vitesses, on l'a augmenté. Dans ce moment on a eu un autre problème, la table a commencé à bouger et par conséquence, l'accéléromètre a détecté un signal mais que n'est pas le correcte, il est le causé par le mouvement horizontal de la table et a mesure qu'on augmente la vitesse, l'amplitude du signal augmente aussi en voulant dire que ce signal est causé pour le mouvement mais qu'il ne détecte pas la masse ponctuel dans la rue. Maintenant, on doit trouver une solution.

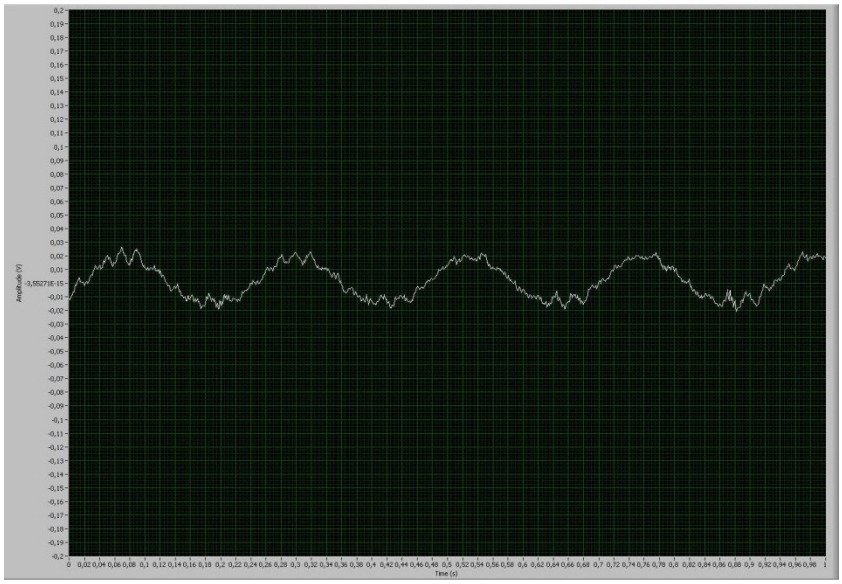


Image 44 : Signal LabView moteur à 12V

6.4.2. Filtrage

On a vu qu'il y a beaucoup de bruit si on commande le moteur avec l'Arduino et ça peut empirer la lecture, pour ça on essaye de créer un filtre passe bas pour éliminer les possibles parasites et aussi de brancher directement le moteur à la font de tension pour éliminer des possibles interférences. [10]

Si la maxime vitesse que le moteur peut avoir est 500rmp, on cherche la fréquence et on met la fréquence de coup à 10 fois cette fréquence.

$$w = 2 \cdot \pi \cdot f$$

$$500\ \cancel{rmp} \cdot \frac{\cancel{2} \cdot \cancel{\pi}\ rad}{1\ \cancel{rev}} \cdot \frac{1\ \cancel{min}}{60\ s} = \cancel{2} \cdot \cancel{\pi} \cdot f \ \rightarrow\ f = \frac{500}{60} = 8{,}33\ Hz$$

$$f_c = 10 \cdot f = 10 \cdot 8{,}33 = 83{,}33\ Hz$$

Maintenant qu'on a la fréquence de coup on peut calculer la résistance qu'on veut. On utilisera un condensateur de 100 nF.

$$f_c = \frac{1}{2 \cdot \pi \cdot C \cdot R}$$

$$83{,}33 = \frac{1}{2 \cdot \pi \cdot 100 \cdot 10^{-9} \cdot R} \rightarrow R = 19{,}1\ K\Omega \cong 20\ K\Omega$$

Avec ce filtre, l'image est un peu meilleure mais pas parfait, pour cette raison on a décidé de filtrer à 5 fois la fréquence de la vitesse maxime.

$$f_c = 5 \cdot f \rightarrow f_c = 5 \cdot 8{,}33 = 41{,}66\ Hz$$

$$f_c = \frac{1}{2 \cdot \pi \cdot C \cdot R}$$

$$41{,}66 = \frac{1}{2 \cdot \pi \cdot 100 \cdot 10^{-9} \cdot R} \rightarrow R = 38{,}2\ K\Omega$$

Comme 38,2 KΩ est une résistance très grande on met 2 condensateurs de 100 nF en parallèle.

$$f_c = \frac{1}{2 \cdot \pi \cdot C \cdot R}$$

$$41{,}66 = \frac{1}{2 \cdot \pi \cdot 2 \cdot 100 \cdot 10^{-9} \cdot R} \rightarrow R = 19{,}1\ K\Omega\ \cong 20\ \mathrm{K\Omega}$$

Voici les connexions d'un accéléromètre avec la font de courant et le filtre.

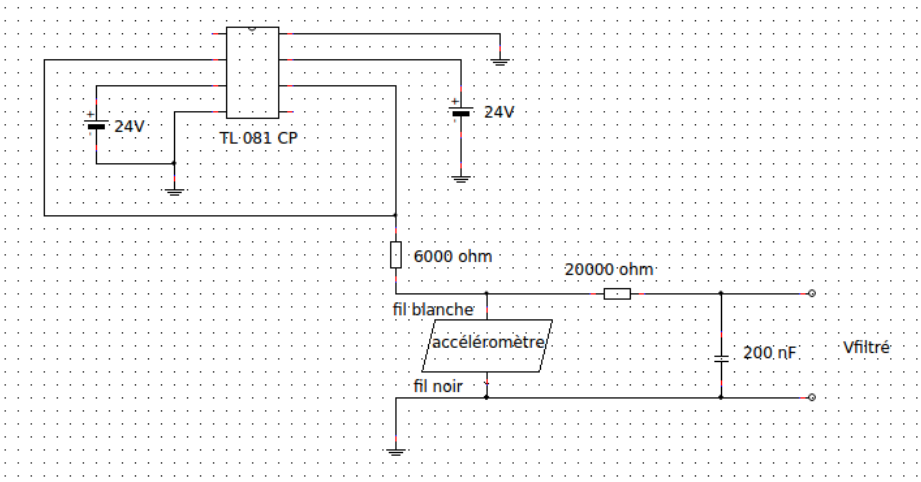


Image 45: Connexions d'un accéléromètre avec la font de courant et le filtre

Pour voir quel gain on a obtenu avec le filtre on a réalisé le diagramme de Bode de la réponse. De la même manière qu'avec le filtre du capteur de courant on a obtenu cette fonction de transfert après la transformation de Laplace.

$$\frac{S(p)}{E(p)} = \frac{1}{1 + R \cdot C \cdot p}$$

$$C = 200\,nF \qquad R = 20\,k\Omega$$

$$\frac{\boldsymbol{S(p)}}{\boldsymbol{E(p)}} = \frac{1}{1 + 20 \cdot 10^3 \cdot 200 \cdot 10^{-9} \cdot p} = \frac{1}{1 + 0{,}004 \cdot p} = \frac{\mathbf{1}}{\boldsymbol{p} + \mathbf{250}}$$

Voici le diagramme de Bode de l'equation de transference.

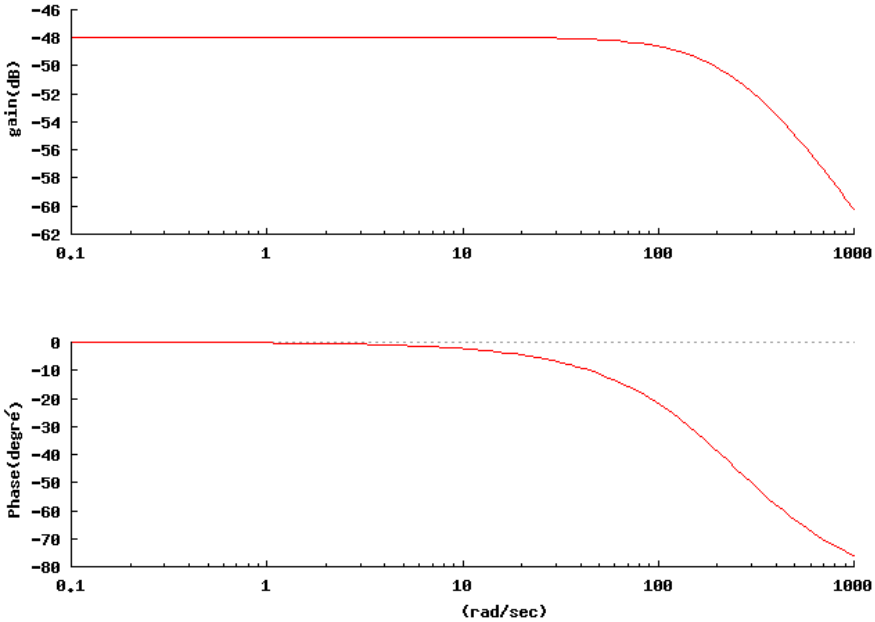


Image 46: Diagramme de Bode du filtre de l'accéléromètre

Avec le diagramme on voit qu'à la fréquence maxime de filtrage 41,66 Hz qu'est 261,8 rad/s, on a un gain de plus et moins -54 dB.

$$20 \cdot logx = -54 \rightarrow x = 0{,}002$$

Avec le filtre la réponse qu'on obtenu est 0,002 fois plus petite que la variation réel qu'on a, est pour ce raison qu'on voit une variation très petite.

Avec ce montage, on obtient le même résultat qu'on avait obtenu sans filtre comme on peut observer dans l'image suivante :

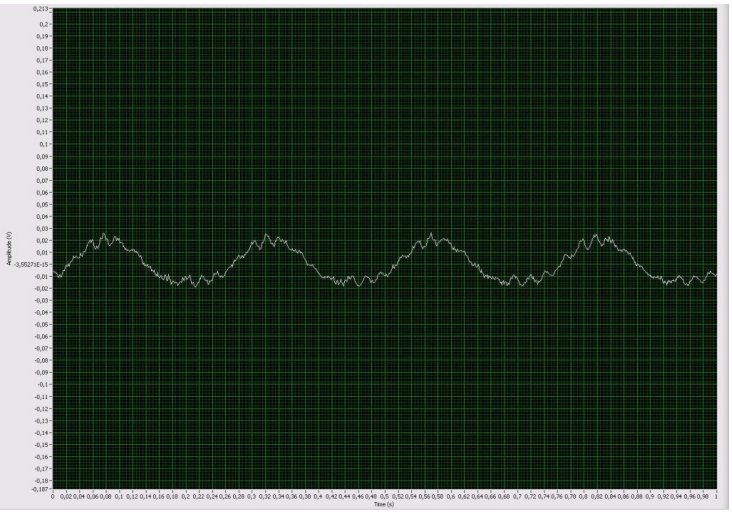


Image 47 : Signal accéléromètre avec filtre

6.4.3. Résultats finals

Pour être sur que le problème n'est pas de l'accéléromètre, on a connecté l'autre qui est nécessaire pour l'étude en utilisant 2 A.O du composant LM324N qui a 4 dedans [23].

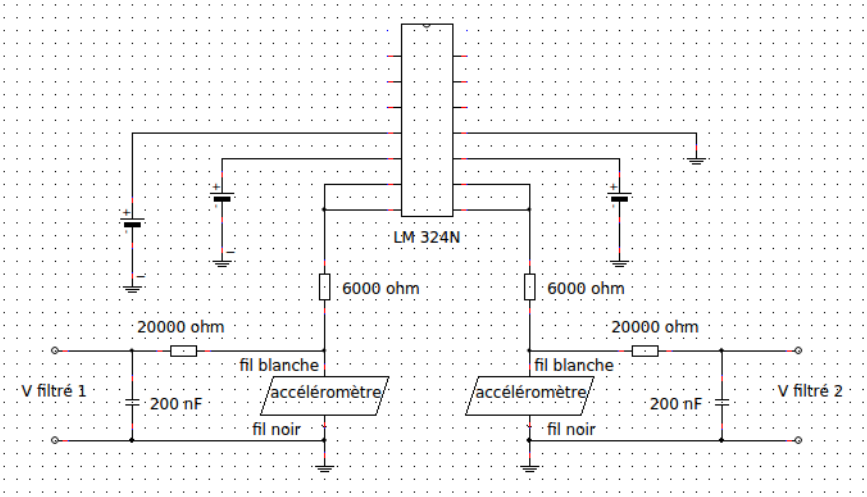


Image 48: Connexions de les deux accéléromètres avec ses font de courant et les filtres

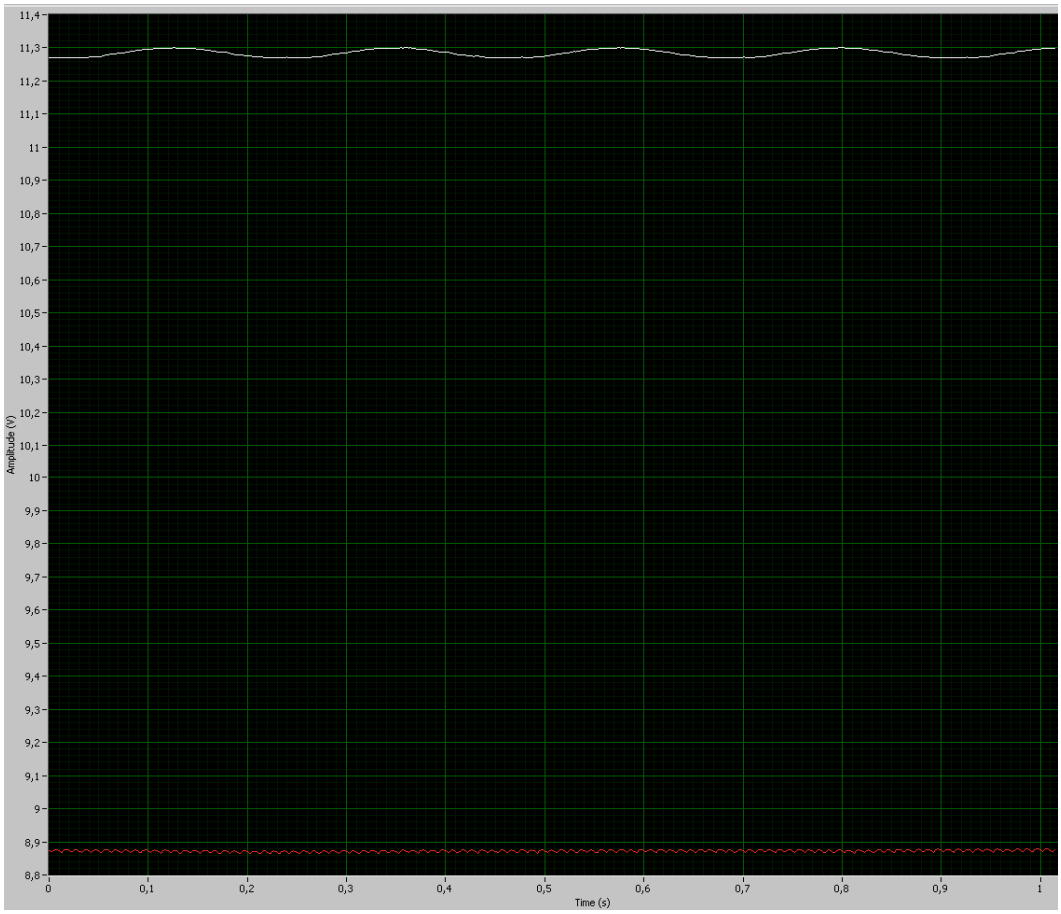


Image 49: Signal de l'accéléromètre de devant (rouge) et de derrière (blanc)

Maintenant on peut voir les deux signaux simultanément mais les résultats continuent étant incorrects. Le signal change seulement quand la table bouge et de plus, la variation est très petite en comparaison avec le mouvement. En plus, l'accéléromètre mis toujours devant voit moins les changements que l'un de derrière. On a essayé de séparer les deux montages en différents cartes pour réduire les possibles interférences; aussi d'utiliser différents A.O. pour chaque accéléromètre et de les échanger entre eux; on a mis la masse devant et derrière pour voir si le signal change et d'échanger la position (mettre l'un de devant derrière et vice versa) des capteurs mais le résultat est toujours le même et également incorrects.

On contemple deux fréquences d'oscillation, une plus grand qui est causé pour le bougement de la table et qu'elle ne change pas son valeur, seulement l'amplitude (augmente à mesure qu'augmente la vitesse et par conséquence le bougement). Et une autre plus petite et périodique que si on l'analyse, on peut conclure qu'elle n'est pas causée par la masse parce qu'elle existe alors que le moteur est arrêté et donc peut être des parasites que le filtre n'a pas éliminé.

On a observé aussi que l'accéléromètre mis toujours dans la fixation du devant ne voit pas aucun changement. Par contre, lequel on met en arrière seulement voit le mouvement de la table. Ça veut dire qu'il marche bien parce que c'est le mouvement qu'il doit capter mais il devait aussi détecter des plus petits. Quant à l'accéléromètre mis devant on n'a pas trouvé une explication logique pour justifier son comportement.

7. Conclusions

On est très content de pouvoir former part de ce projet qu'a commencé avec nous. On n'a pas eu le temps nécessaire pour finir touts les objectifs et intentions qu'on avait au début du semestre à cause des problèmes surgis pendant la réalisation du projet mais on a finalisé des points très importants pour la réalisation du banc:

- L'alimentation du moteur
- L'asservissement en vitesse
- L'acquisition du signal pour corriger les défauts
- Préparation du programme et montage pour l'équilibrage

On a réalisé les montages et la programmation nécessaires pour réaliser l'équilibrage du disc et corriger les défauts mais on n'a pas pu terminer de faire bien l'étude des donnés. On croit qu'avec les logiciels qu'on utilise, les masses sont trop petites et les accéléromètres ne perçoivent pas les petites accélérations qu'elles provoquent mais par contre, on ne peut pas mettre une plus grand. De plus, l'accélération causée par la masse sera visible seulement au moment que la masse se trouve dans le point plus haut et plus bas du mouvement circulaire parce qu'en ce moment l'accélération aura la même direction que l'accéléromètre. Peut être que ça sera pendant un instant trop court pour être capté.

Un autre possible problème peut être la mauvaise fixation du châssis, on doit essayer avec un châssis plus rigide ou bien de mettre le montage sur une superficie meilleur fixé donc maintenant, les résultats obtenus ont eu causés pour le mouvement de la table sur laquelle il se trouve et pas pour ce qu'on veule étudier.

On croit aussi, après avoir étudié toutes les configurations possibles, que les problèmes viennent des accéléromètres. On doit trouver un autre système pour pouvoir capter les accélérations. Peut être qu'on n'a pas fait une bonne utilisation de cet instrument ou bien qu'il n'est pas suffisamment précis et on doit essayer un autre type d'accéléromètre avec plus d'exactitude.

Avec ce projet on a appris que les choses ne sont jamais comme tu crois et que toujours il y a des petits problèmes avec lesquels tu ne comptes pas. Ces problèmes sont lesquels font ralenti tout le travail et dans un projet long est encore plus difficile planifier le travail. Ce projet on a permis aussi pendre conscience de comme fonctionne un projet pratique de génie et comme toutes les connaissances théoriques qu'on a appris pendant les courses dernières sont appliqués à la pratique.

8. Bibliographie

1] Arduino. *Arduino Uno* [en ligne]. [Consulté: 12 février 2014]. Disponible à: arduino.cc/en/Main/arduinoBoardUno

[2] S. Fitzgerald and M.Shilon. *Arduino Projects Book.* Torino: Setembre 2012.

[3] Ciiva. *Doga - DO162.4122.3B.00/3065* [en ligne].[Consulté: 25 février 2014]. Disponible à: http://ciiva.com/CloudSearch/Component/14955570/doga-do162-4122-3b-00-3065

[4] Radiospares. *SGS-Thomson L298N* [en ligne].[Consulté: 2 mars 2014]. Disponible à: http://docs-europe.electrocomponents.com/webdocs/0060/0900766b80060c26.pdf

[5] Blog-J. "Puente-H L298 en motores DC con algo de PWM parte I". *Blog-J* [blog].Disponible à: http://blog-j.marcano.net.ve/index.php/2010/09/24/puente-h-con-l298-motores-dc-y-algo-de-pwm/

[6] HPC. *Roues dentées* [en ligne].[Consulté: 11 mars 2014]. Disponible à : http://www.hpceurope.com/cat13HPC/HPC-T4-2013-RouesDentees.pdf

[7] Forbo-Siegling. *Calcul des courroies dentées* [en ligne]. [Consulté: 15 mars 2014]. Disponible à: http://www.forbo-siegling.com/fr/pages/brochures/technical/download/Forbo-Siegling_202_fr.pdf

[8] Radiospares. *Multi-Range Current Transducer LA25-NP Product Data* [en ligne]. [Consulté: 20 mars 2014]. Disponible à: http://docs-europe.electrocomponents.com/webdocs/002b/0900766b8002bd9d.pdf

[9] Datasheet catalog. *Current Transducer LA 25-NP Data* [en ligne]. [Consulté: 20 mars 2014]. Disponible à : http://pdf.datasheetcatalog.com/datasheet/lem/LA25-NP.pdf

[10] Wikipedia. *Filtre passe bas* [en ligne]. [Consulté: 17 abril 2014]. Disponible à: http://fr.wikipedia.org/wiki/Filtre_passe-bas

[11] Radiospares. *Inductive Proximity Sensors, Cylindrical, M8 threaded, Brass Body* [en ligne]. [Consulté: 26 mars 2014]. Disponible à: http://docs-europe.electrocomponents.com/webdocs/002c/0900766b8002c996.pdf

[12] A. Rava. *Desequilibrio estático y dinámico* [en ligne].[Consulté: 14 mars 2014]. Disponible à: http://www.elettrorava.es/espanol/docum/equilibrado/secc-02.html

[13] J.M.Santesteban Cizar. *Desequilibraje* [en ligne]. UPNa. [Consulté: 13 mars 2014]. Disponible à: http://academica-e.unavarra.es/bitstream/handle/2454/1861/577064.pdf?sequence=1

[14] National Instruments. *Logiciel de conception de systèmes LabVIEW* [en ligne]. [Consulté: 5 mai 2014]. *Disponible à : http*://www.ni.com/labview

[15] Wikipedia. *Oscilloscope* [en ligne]. [Consulté: 2 avril 2014]. *Disponible à :* http://fr.wikipedia.org/wiki/Oscilloscope

[16] Pico Technology. *PicoScope. Manuel d'utilisation* [en ligne]. [Consulté: 9 avril 2014]. *Disponible à:* http://www.picotech.com/document/pdf/psw.fr-1.pdf

[17] Brüel & Kjær. *Product data. Industrial Accelerometer-Type 8325* [en ligne]. [Consulté: 2 mai 2014]. *Disponible à:* http://bruel.ru/UserFiles/File/8325.pdf

[18] Brüel & Kjær. *Accelerometer Catalogue*[en ligne]. [Consulté: 3 mai 2014]. *Disponible à: http://www.midebien.com/_literature_104056/Aceler%C3%B3metro_industrial_8326-B*

[19] Datasheet catalog. *Precision Instrumentation amplifier* [en ligne]. [Consulté: 5 juin 2014]. *Disponible à:* http://pdf.datasheetcatalog.net/datasheet2/1/03e3o8y9sz5awge7sed0y1f4w0yy.pdf

[20] Ph Dondon. *Electronique Analogique modulaire* [en ligne]. [Consulté: 3 juin 2014]. *Disponible à:* http://dondon.vvv.enseirb-matmeca.fr/EAM/transpAOP2emepartie.pdf

[21] Rémy MALLARD. *Générateurs de courant constant*[en ligne]. [Consulté: 3 juin 2014]. *Disponible à:* http://www.sonelec-musique.com/electronique_bases_gene_courant.html

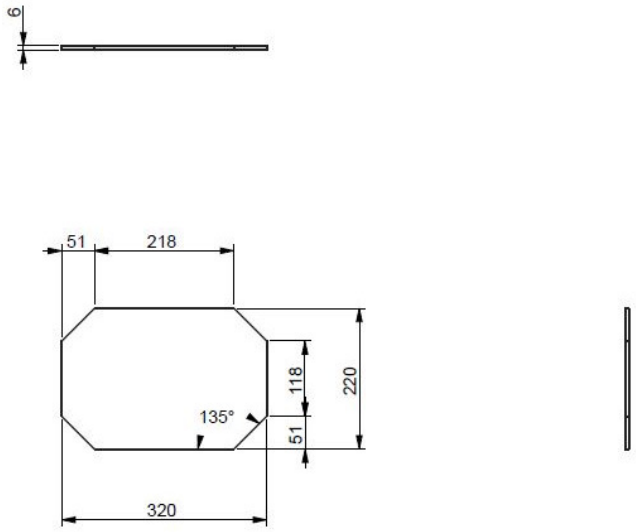
[22] Datasheetcatalog. *TL081CP datasheet* [en ligne]. [Consulté: 11 juin 2014]. *Disponible à:* http://www.datasheetcatalog.net/es/datasheets_pdf/T/L/0/8/TL081CP.shtml

[23] Datasheetcatalog. *Low power quad op amps* [en ligne]. [Consulté: 18 juin 2014]. *Disponible à:* http://pdf.datasheetcatalog.com/datasheet/philips/LM324N.pdf

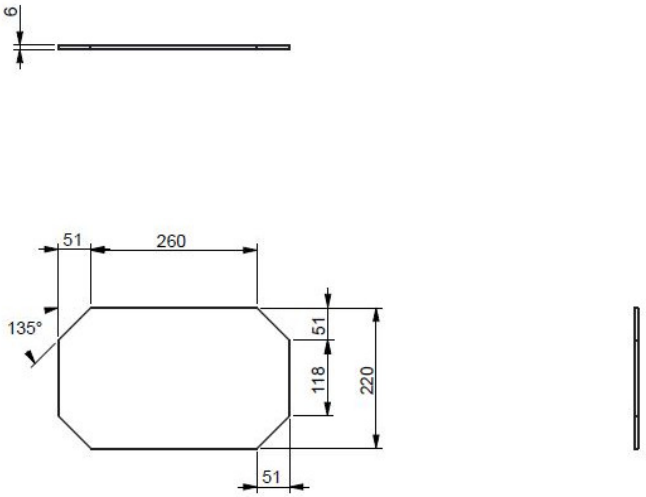
ANNEXE

I. Mises en plan

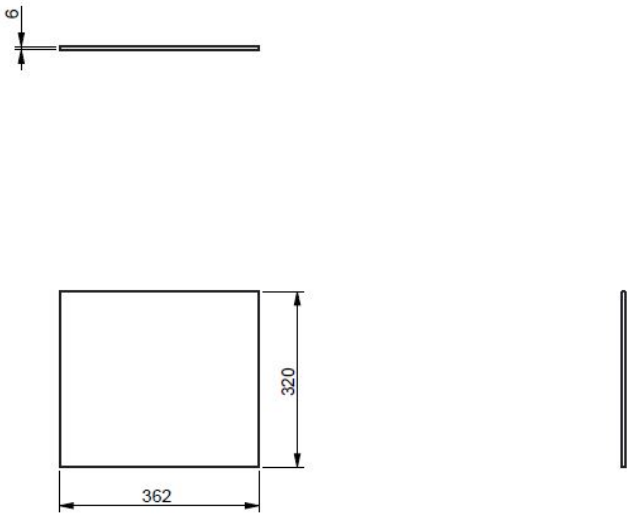
Plan 1 : Mur devant et derrière de la protection supérieur



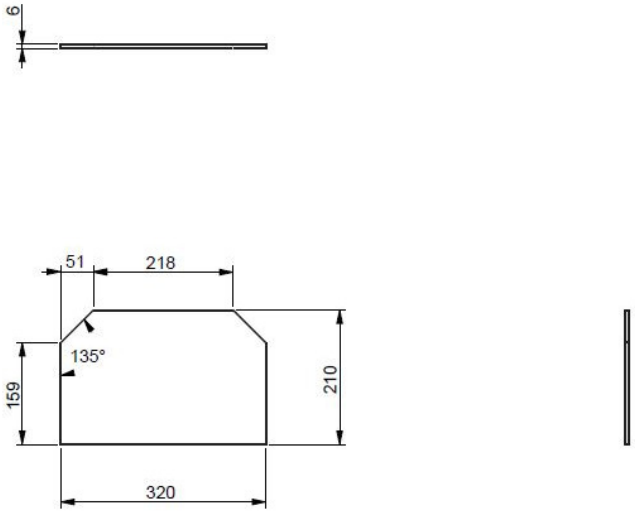
Plan 2 : Mur latéral de la protection supérieur



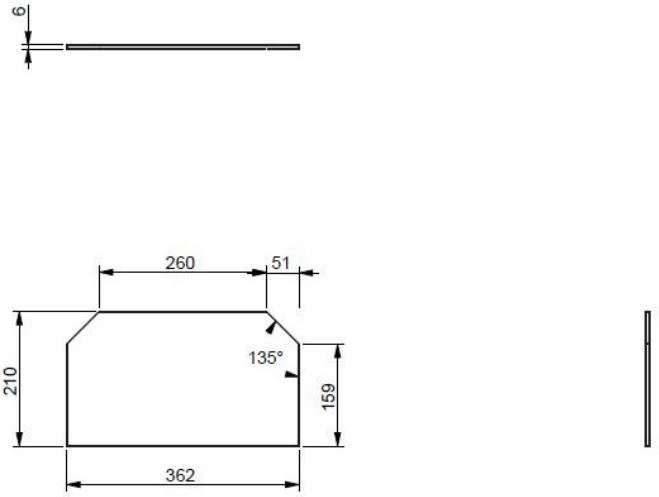
Plan 3: Mur supérieur de la protection supérieur



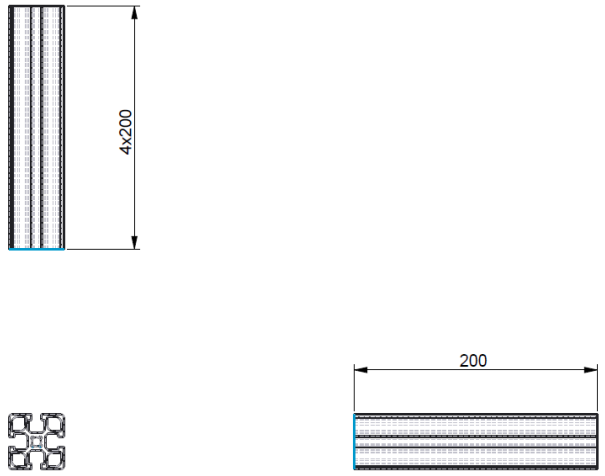
Plan 4 : Mur derrière du banc



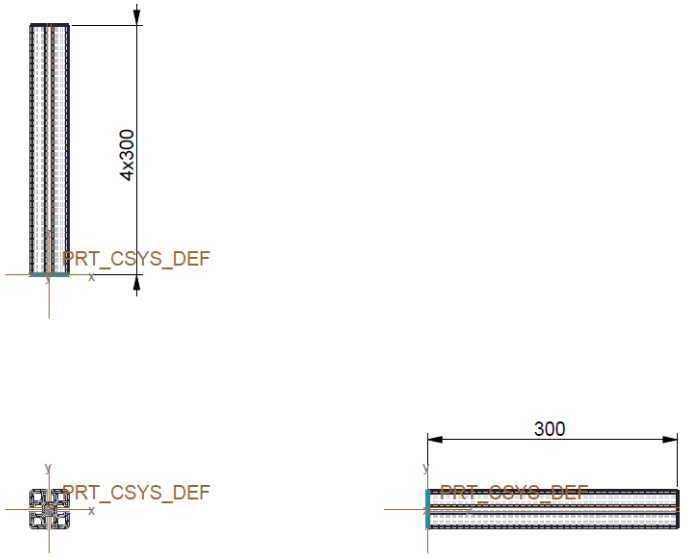
Plan 5: Murs latéraux du banc



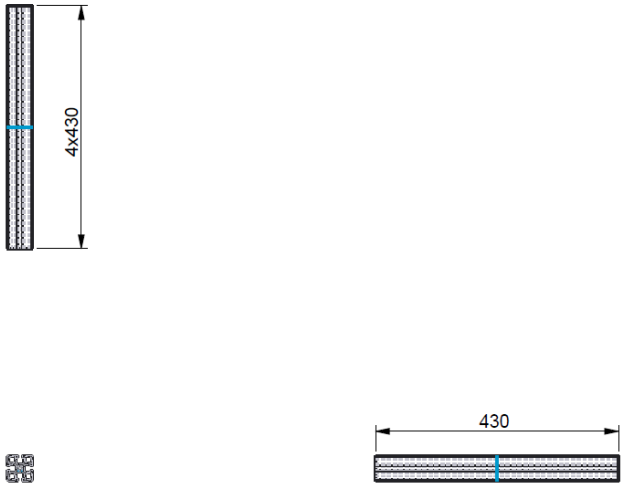
Plan 6: Bars de profilé 44x44 et 200 mm de longueur



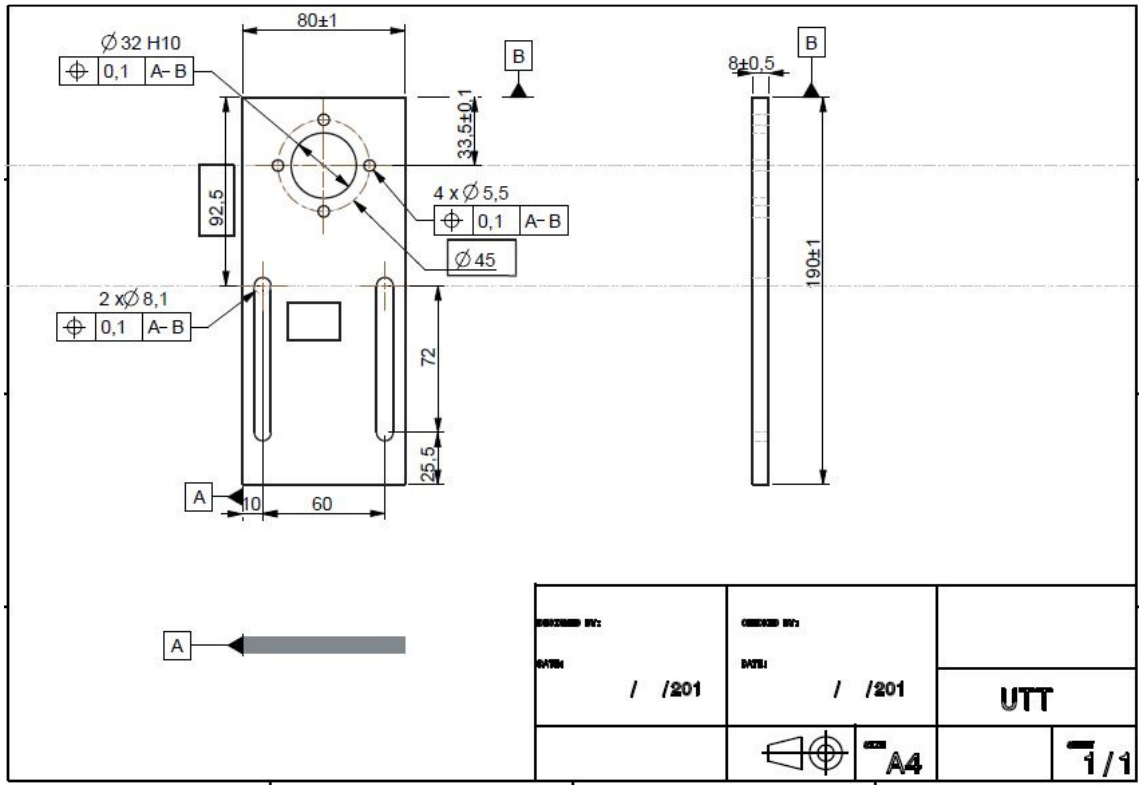
Plan 7: Bars de profilé 44x44 et 300 mm de longueur



Plan 8: Bars de profilé 44x44 et 430 mm de longueur



Plan 9: Support moteur



II. Code du programme définitif

```
#include <FlexiTimer2.h>
#include <digitalWriteFast.h>

#define directionMoteur1  4
#define directionMoteur2  5
#define pwmMoteur  9
#define codeurInterruption 0
#define motorInterruption 1
const int hallPin = 2;
const int led = 6;
const int motorPin = 3;

//Capteur proximite
int hallState0 = 0;
float temps0 = 0;
float temps1 = 0;
float v = 0;
int N = 0;

//variable capteur hall
int hallState = 0;

//Variables nombreReçu
int octetReception=0;
long nombreReception=0;
long nombreReception0=0;

// Initialisations
void setup(void) {
  Serial.begin(115200);
  Serial.flush();
  pinMode(directionMoteur1, OUTPUT);
  pinMode(directionMoteur2, OUTPUT);
  pinMode(pwmMoteur, OUTPUT);
  pinMode(led, OUTPUT);
  pinMode(motorPin, OUTPUT);

  // Codeur incremental
  pinMode(hallPin, INPUT);
  //digitalWrite(hallPin, HIGH);
  attachInterrupt(codeurInterruption, GestionInterruptionHallPin, RISING);
  //interruption moteur
  attachInterrupt(motorInterruption, arretEmergence, HIGH);

  digitalWrite(pwmMoteur,HIGH);
  digitalWrite(directionMoteur1,LOW);
  digitalWrite(directionMoteur2,LOW);
 }

// Boucle principale
```

```
void loop(){
  if (Serial.available() == true) {
  nombreReception=lecture_port();
  }
  if (nombreReception!=nombreReception0) {
    nombreReception=constrain(nombreReception,-255,255);
    Serial.println(nombreReception);
    nombreReception0=nombreReception;
    if (nombreReception>0) {
      analogWrite(pwmMoteur,nombreReception);
      digitalWrite(directionMoteur1,HIGH);
      digitalWrite(directionMoteur2,LOW);
    }
    if (nombreReception==0) {
      digitalWrite(pwmMoteur,LOW);
      digitalWrite(directionMoteur1,LOW);
      digitalWrite(directionMoteur2,LOW);
    }
    if (nombreReception<0) {
      analogWrite(pwmMoteur,-nombreReception);
      digitalWrite(directionMoteur1,LOW);
      digitalWrite(directionMoteur2,HIGH);
    }
  }
  if (N>20)
  {
    temps1 = millis();
    v = ((30000)/(temps1-temps0));
    N=1;
    Serial.println(v);
    temps0 = temps1;
  }
  if (N==20){
    digitalWrite(led, HIGH);
    }
  else{
    digitalWrite(led, LOW);
  }
}

int lecture_port(){
  int octetRecu=0;
  int compt=0;
  boolean signe=true;
  long nombreRecu=0;
  while (Serial.available()>0) {
    octetRecu=Serial.read();
    if (octetReception==10) {
      break;
    }
    else {
      if ((octetRecu=='-') && (compt==0)){
```

```
      signe=false;
     }
     compt=compt+1;
     octetRecu=octetRecu-48;
     if ((octetRecu>=0)&&(octetRecu<=9)){
       nombreRecu = (nombreRecu*10)+octetRecu;
     }
    }
    delay(1);
  }
  if (signe==false){
    nombreRecu=nombreRecu*(-1);
  }
  return(nombreRecu);
}

// Routine de service d'interruption attaché à  la voie A du codeur incrémental
void GestionInterruptionHallPin(){
  N = N + 1;
}
void arretEmergence(){
  nombreReception = 0;
  long nombreRecu=0;
  nombreRecu = 0 ;
}
```