



EA07– TP projet

Banc hybride

**Pierre Perocheau – Benjamin Redzic - Muhammad Ieyan -
M'Monwal WABEDO - Maxime Dai Pra - ADJINOU Kodjo
Paul Vahana**

Automne



utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

I. Introduction

Dans le cadre des TP-Projets de l'UE EA07, nous avons travaillé sur la réparation et la mise en fonctionnement d'un banc d'essai simulant une motorisation hybride. Ce banc a déjà fait l'objet de nombreux projets au sein de diverses UE comme TN12, EA02 et CS02. Le banc est constitué de différents composants mécaniques et électriques que nous allons détailler. L'objectif du projet était de réparer et de rendre le banc d'essai opérationnel pour la réalisation de TP en EA01, EA07, etc...

Dans un premier temps nous présenterons l'état initial du projet. Nous verrons précisément l'objectif du banc d'essai, l'ensemble de ses composants, son fonctionnement et son pilotage. Puis, nous détaillerons le travail que nous avons effectué durant tout ce semestre, et nous donnerons des recommandations sur la suite des travaux à réaliser.

Il est important de noter que ce rapport vient en complément de celui réalisé par l'équipe A23. Ainsi, nous recommandons à la prochaine équipe de prendre connaissance de ces deux rapports.

II. Sommaire

I.	Introduction	2
II.	Sommaire.....	3
III.	Découverte du TP, remise en route de l'armoire électrique et schéma	4
	Remise en route de l'armoire électrique :	4
IV.	Réalisation du système de contrôle des freins	6
	Contexte	6
	Choix des composants.....	7
	Mise en œuvre du circuit expérimentale et numérique	8
	Tests expérimentaux.....	9
V.	Récupération du signal du couplemètre.....	11
VI.	Récupération du signal du capteur de courant.....	13
VII.	Amplification du signal Arduino pour contrôle du driver brushless.....	15
	Conception et réalisation du circuit :	15
	Programmation et contrôle :	16
	Pilotage du moteur :	17
VIII.	Conception d'une carte électronique pour centraliser les capteurs	18
IX.	Bilan et suite	25
	Bilan du semestre :	25
X.	Conclusion	26
XI.	Annexes avec contributions individuelles.....	27
	Contributions individuelles :	27
	Annexe 1 : Courbes condensateur 6800 μ F.....	28
	Annexe 2 : Courbes condensateur 22000 μ F.....	29
	Annexe 3 : Codes pour l'amplification du signal à destination du driver brushless	30
	Annexes 4 : Récupération du signal du capteur de courant	33

III. Découverte du TP, remise en route de l'armoire électrique et schéma

Nous reprenons donc la suite du TP projet Banc Hybride dont l'objectif à terme est de simuler le fonctionnement d'un véhicule automobile à motorisation hybride (type Toyota Prius). Nous avons consacré notre première séance de TP à la compréhension et la prise en main du banc d'essai, avant de définir nos objectifs, en nous basant sur les recommandations du groupe de travail de l'année passée. Ainsi, les objectifs de ce semestre étaient :

- Comprendre le fonctionnement de l'armoire, la remettre en marche et la documenter (schéma électrique sous format numérique)
- Récupération des données du couplemètre
- Assurer un contrôle total du moteur brushless
- Implanter un capteur de courant
- Mettre au point le contrôle des freins
- Intégrer l'ensemble des modifications dans l'armoire électrique et faire une alimentation commune pour l'ensemble des capteurs.

Remise en route de l'armoire électrique :

La première étape fut d'identifier et de comprendre le rôle des différents composants du banc hybride. On peut diviser le banc d'essai en trois parties : la partie mécanique, la partie alimentation et commande, et les capteurs.

Composants mécaniques principaux :

- Un **moteur brushless**, qui simule le fonctionnement du moteur électrique du véhicule
- Un **moteur asynchrone**, alimenté en triphasé, qui simule le fonctionnement du moteur thermique du véhicule
- Un **volant d'inertie**, simulant la masse du véhicule, relié mécaniquement aux deux moteurs précédents
- Des **freins électromagnétiques** : un premier qui bloque la rotation du moteur asynchrone et un second qui bloque la rotation du volant d'inertie

Composants électriques principaux :

- Une **armoire électrique**, qui alimente et permet la commande du moteur asynchrone, et des freins électromagnétiques. Elle est elle-même alimentée en triphasé 380V. Le contrôle du système se fait en partie (contrôle des freins, de la vitesse de rotation du moteur asynchrone) via la façade de cette armoire.
- Un **générateur de tension 13V**, qui alimente le moteur brushless et un **générateur de tension +24/-24V** pour l'alimentation des différents capteurs
- Un **driver** de moteur brushless, qui gère le fonctionnement du moteur brushless.
- Une **carte Arduino Uno** qui permet de récupérer les signaux et de contrôler le brushless

Capteurs :

- Un **couplemètre**, qui permet de récupérer les valeurs de couple du moteur asynchrone
- Un **codeur incrémental**, qui sert à mesurer la vitesse de rotation de l'arbre post réducteur
- Un **capteur de courant** qui permet de mesurer la puissance absorbée par le moteur asynchrone

Pour faciliter la lecture des câblages et l'identification des composants dans l'armoire électrique, nous réalisons un schéma électrique sur le logiciel QElectrotech. Il est à noter que ce schéma est disponible sous format numérique sur Windchill et peut être modifié.

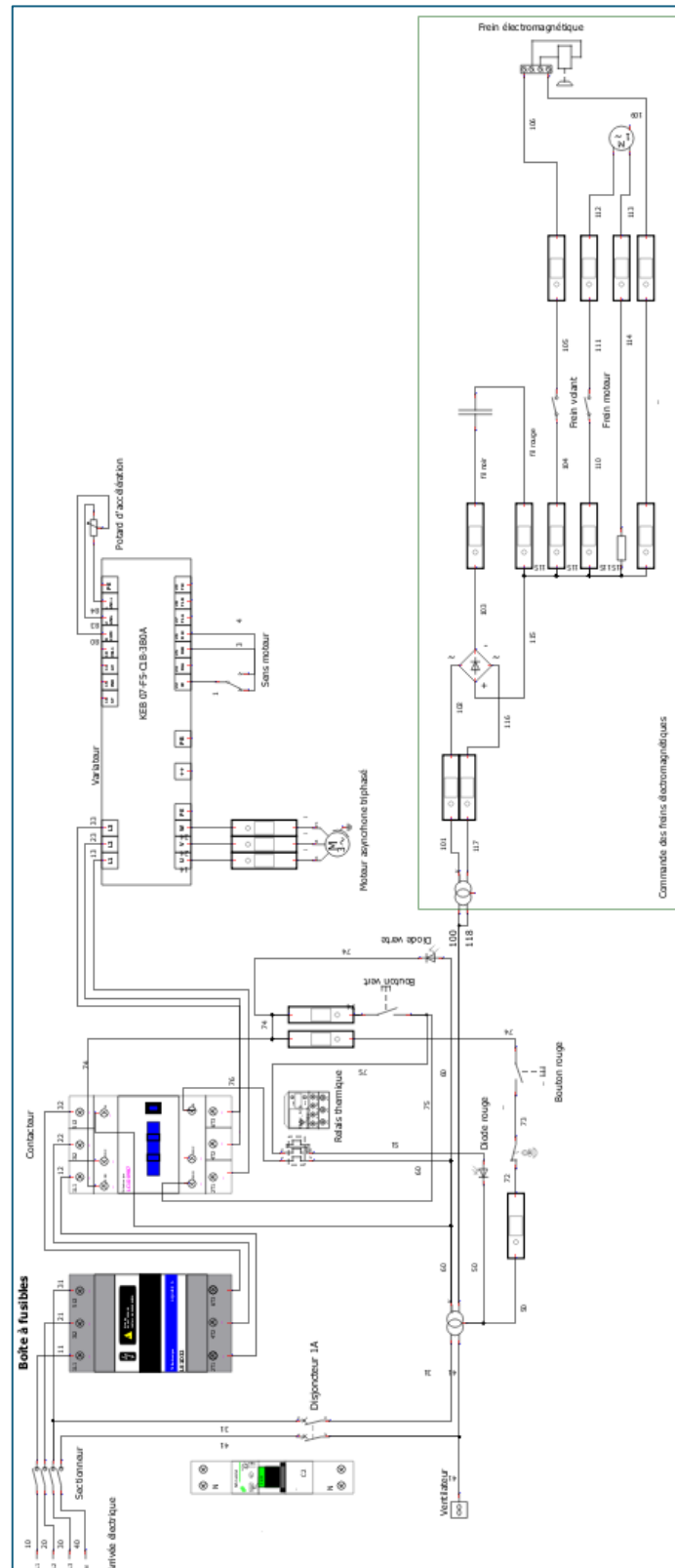


Figure 1 : Schéma de câblage de l'armoire électrique

IV. Réalisation du système de contrôle des freins

Contexte

Des freins électromagnétiques sont installés en amont du moteur asynchrone et du volant d'inertie pour pouvoir simuler les différents modes de fonctionnement du véhicule hybride qu'est censé reproduire le banc d'essai (le tableau récapitulant les différents modes de fonctionnement est disponible dans le rapport de l'année précédente).

Il faut préciser ici qu'il n'est pas question de stopper les moteurs lancés à grande vitesse avec les freins. Les freins sont utilisés pour stopper les moteurs quand ceux-ci sont arrêtés ou avec une vitesse de rotation très faible. Voici un tableau rappelant les caractéristiques des freins électromagnétiques intégrés à ce banc d'essai

Spécification	Tension d'alimentation	Puissance électrique
Frein du moteur asynchrone ROBA® - Quick, Type 3/520.202.0.	24VDC $\pm 10\%$ (IEC 60038)	13W
Frein du volant d'inertie ROBA® - Quick, Type 5/520.202.0.	24VDC $\pm 10\%$ (IEC 60038)	31W

Pour le groupe nous ayant précédé, l'enjeu était de faire parvenir à ces freins une tension de 24VDC afin de freiner le moteur ou le volant d'inertie sur commande. Après transformation de la tension de réseau de 230VAC en 18VAC (25V en tension de crête) grâce à un transformateur, le groupe précédent avait commencé un prototype d'un circuit redresseur-lisseur de tension combinant un pont de diode et un condensateur.

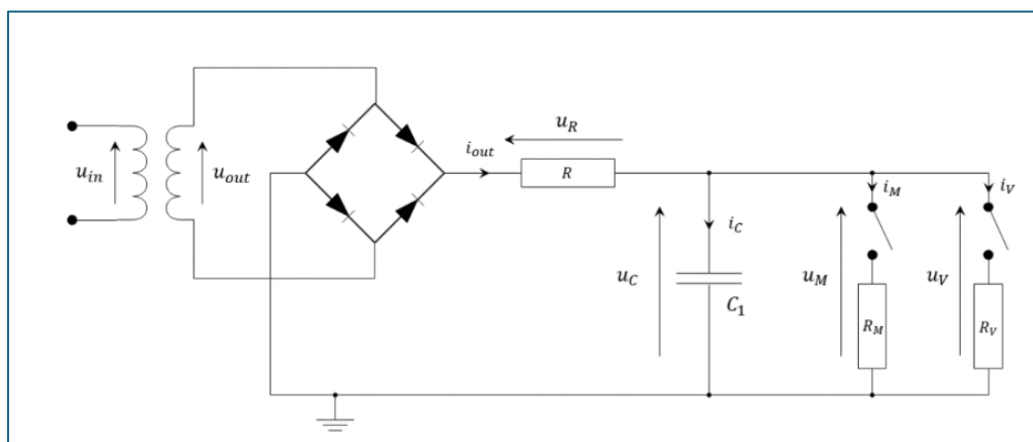


Figure 2 : Schéma électrique du premier prototype du circuit redresseur-lisseur de tension

Après analyse de la liste des solutions de lissage post-redressement proposée par nos prédécesseurs, nous estimons que les solutions ayant le meilleur rapport simplicité de mise en œuvre/performance pour nous sont :

- Le lisseur capacitif
- Le régulateur de tension LDO

Toutefois, nos TP ayant commencé tard, il n'était plus possible de commander le régulateur de tension. Ainsi nous avons choisi de mettre en œuvre le circuit redresseur-lisseur capacitif.

Choix des composants

Nos prédécesseurs avaient rencontré deux problèmes en simulant leur premier prototype sur Scilab:

- La résistance R abaissait trop la tension aux bornes des freins.
- La tension aux bornes des freins était trop instable ($\pm 17\%$).

Pour la résistance R, nous en venons à la même conclusion que nos prédécesseurs et choisissons tout simplement de l'enlever.

Pour améliorer le lissage de la tension, il convient ici de choisir un condensateur avec une capacité plus grande. Pour avoir une idée de l'ordre de grandeur de la capacité à utiliser lorsque l'on met en placeur un lisseur capacitif, on peut utiliser cette relation simplifiée venant de la formule du courant dans un condensateur :

$$i = C \frac{V}{T}$$

$$C = \frac{i}{V * f}$$

Avec :

- i le courant passant dans la charge
- V l'amplitude de l'ondulation de tension que l'on souhaite ne pas dépasser
- f la fréquence du signal

On a :

- $V = 4,8V$ (10% de 24V multiplié par 2),
- $i = 1,83A$ (courant consommé par les deux moteurs)
- $f = 100Hz$ (fréquence du signal en sortie du pont de diode)

Ainsi on peut commencer à envisager d'utiliser des condensateurs ayant une capacité supérieure à :

$$C \approx 3800 \mu F$$

On voit dans cette relation que plus la capacité du condensateur est grande et plus la chute de tension à ses bornes sera atténuée ; nous pourrions déduire ainsi qu'en prenant un condensateur avec une capacité énorme, la tension serait parfaitement lisse et le problème serait complètement résolu.

Ce serait un peu trop simple. En effet, en regardant la relation précédente, on voit qu'en augmentant la capacité, avec V et f constant, c'est le courant I qui se met à augmenter. Cela se traduit par des appels de courants dans le condensateur qui induisent de l'effet Joule impactant négativement la durée de vie du condensateur. Dans la doc technique, le constructeur assure une durée de vie de X h pour une amplitude maximale d'ondulation (ripple) de courant avec une fréquence et une température ambiante donnée.

Dans le rapport de l'année précédente on peut voir qu'un condensateur de 4700 μF donnait un résultat satisfaisant sur les simulations. En fouillant dans les stocks disponibles en halle, nous avons

trouvé deux condensateurs pouvant satisfaire notre besoin : un condensateur électrolytique de $6800\mu\text{F}$ et un autre de $22000\mu\text{F}$.

Mise en œuvre du circuit expérimentale et numérique

Nous mettons ainsi en œuvre le lisseur capacitif dont le schéma se trouve ci-dessous :

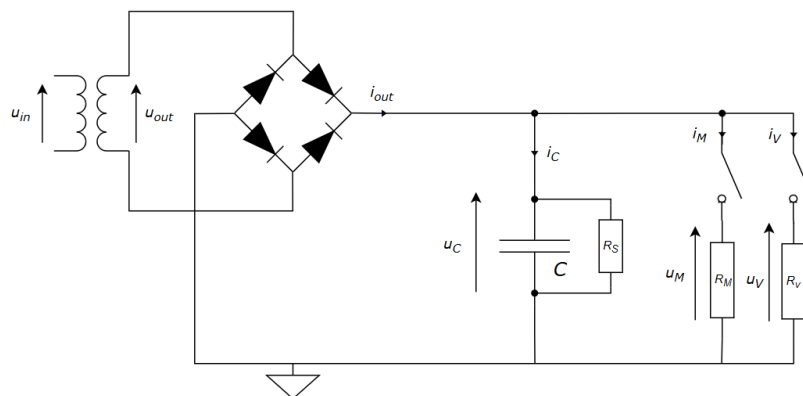


Figure 3 : Schéma électrique de notre prototype de circuit redresseur-lisseur de tension

On peut remarquer sur le schéma que nous avons fait le choix de ne pas utiliser de condensateur de petite capacité en parallèle servant à améliorer le filtrage. On rajoute une résistance de sécurité R_S pour que le condensateur se décharge dans celle-ci une fois la tension d'entrée coupée.

Nous allons maintenant tester les deux condensateurs trouvés ($6800\mu\text{F}$ et $22000\mu\text{F}$) l'un après l'autre, et relever puis comparer la tension se trouvant à leurs bornes lorsque les deux freins sont en route. Pour accompagner ces mesures expérimentales, nous proposerons aussi quelques courbes issues de simulations faites à l'aide du logiciel LTSpice ; ceci dans un but de comparaison et de prise en main de ce logiciel. Nous effectuerons ensuite un choix définitif du condensateur.

Pour les relevés de tensions sur le circuit, nous utilisons un oscilloscope *PicoScope* couplé au logiciel PicoScope 6 de l'entreprise Pico Technology qui permet la visualisation des courbes de tension sur le PC. Petite précision, notre outil de prise des mesures divise la tension par 10, ainsi la tension affichée sur le diagramme est à multiplier par 10 pour obtenir la tension mesurée.



Figure 4 : Oscilloscope utilisé pour les relevés de tension

Sur LTSpice nous créons le schéma suivant :

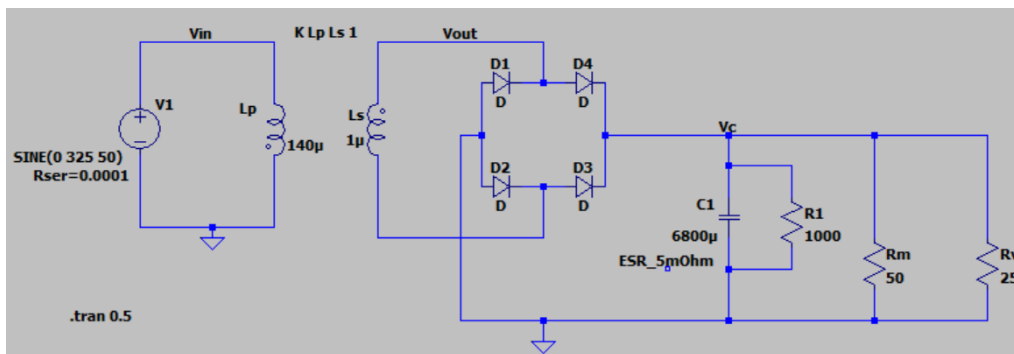


Figure 5 : Circuit modélisé sur LTSpice

La partie compliquée dans la mise en œuvre de ce circuit était la modélisation du transformateur étant donné que l'on doit créer celui-ci nous-même et qu'il y'a quelques subtilités accompagnant sa création.

Tests expérimentaux

Pour chacune des courbes, les deux freins sont actifs. Elles peuvent être consultée dans [les annexes](#). Voici le tableau résumant les valeurs trouvées :

	Condensateur 6800µF		Condensateur 22000µF	
	Oscilloscope	LTSPICE	Oscilloscope	LTSpice
Uc(max)	24,8V	25,5V	24,4V	25,5V
Uc(min)	22,9V	23,8V	23,3V	24,8V
Ic max (petits pics)		11A		14A
Ic max (grand pic au début)		53A		154A

On voit qu'il y'a un léger décalage entre les valeurs que nous donne LTSpice et les valeurs obtenues sur l'oscilloscope. Cet écart peut être dû :

- A la résistance en série du condensateur qui peut être mal estimée ici (je me suis basé sur des valeurs trouvées sur internet)
- Au transformateur qui n'est pas modélisé avec assez de précision, le transformateur réel donne peut être légèrement plus ou moins que 24V ce qui n'est pas pris en compte dans la simulation.

Ce qui choque sur le résultat des simulations sont les pics de courant d'entrée au tout début. Il semble assez difficilement imaginable que le disjoncteur de l'armoire laisse passer de telles intensités. Cela est potentiellement à investiguer.

Pour les petits pics venant à chaque charge, la valeur du courant est plus raisonnable. Le fabricant et donc les spécificités techniques des condensateurs ne nous étaient pas connues au moment du choix définitif du condensateur. Nous avons donc choisi le condensateur de 22000µF de capacité en raison de la meilleure stabilité de la tension à ses bornes.

Pour l'installation dans l'armoire, nous avons compacté le prototype et l'avons vissé ; nous avons mis le condensateur dans une boîte plastique vissée à côté de l'armoire. Après installation, nous avons retesté les freins et ceux-ci marchaient correctement. On peut retrouver le schéma de l'installation dans la partie 1. Ci-dessous une photo où l'on peut voir le lisseur :

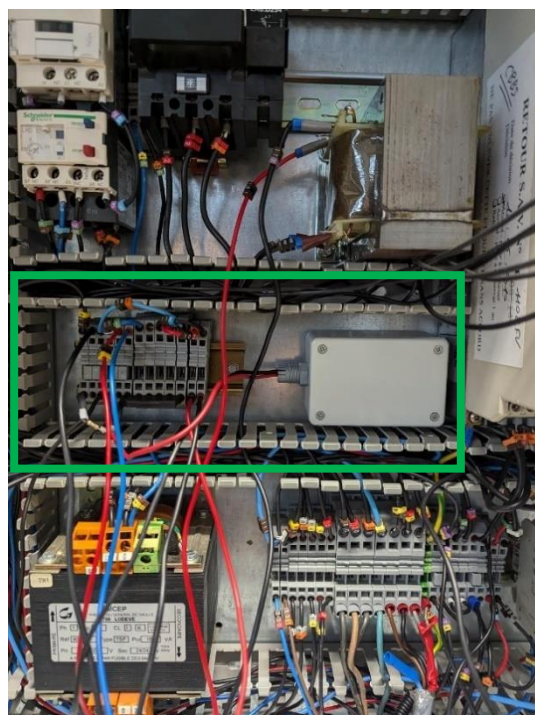


Figure 6 : Redresseur lisseur implémenté dans l'armoire

Il reste simplement à rendre les branchements plus propres en rajoutant des cosses où il n'en a pas et en rajoutant des colliers de serrage plastique pour les câbles allant vers les interrupteurs.

V. Récupération du signal du couplemètre

Un couplemètre est installé sur le banc de test afin de récupérer l'information du couple transmit par le moteur asynchrone. Ce couplemètre, alimenté en 12V, renvoi des signaux entre -5 et 5V. Cela permet d'obtenir les couples pour les deux sens de rotation :

- Moteur est à l'arrêt : la valeur renvoyée par le couplemètre sera 0V
- Moteur tourne dans le sens horaire : la valeur renvoyée par le couplemètre sera comprise entre 0 et 5V
- Moteur tourne dans le sens anti-horaire : la valeur renvoyée par le couplemètre sera comprise entre 0 et -5V

Le détail des branchements étant donné dans le rapport 2023 et dans la documentation, on précise ici simplement la partie traitement du signal. L'objectif étant de lire les valeurs du couplemètre par le biais d'un Arduino, il est nécessaire de traiter le signal pour passer d'une plage -5/5v à une plage 0/5V. Ce changement de tension se réalise en 2 temps :

- On réalise premièrement un décalage par l'intermédiaire d'un AOP (montage sommateur) pour obtenir un signal sur la plage 0-10V
- On utilise ensuite un Pont Diviseur de Tension (PDT) pour obtenir le signal final sur la plage 0-5V.

Ci-dessous, le montage réalisé. La figure 7 présente un schéma de principe tandis que la figure 8 présente un schéma plus formel. Le schéma de principe est disponible en pdf sur Windchill.

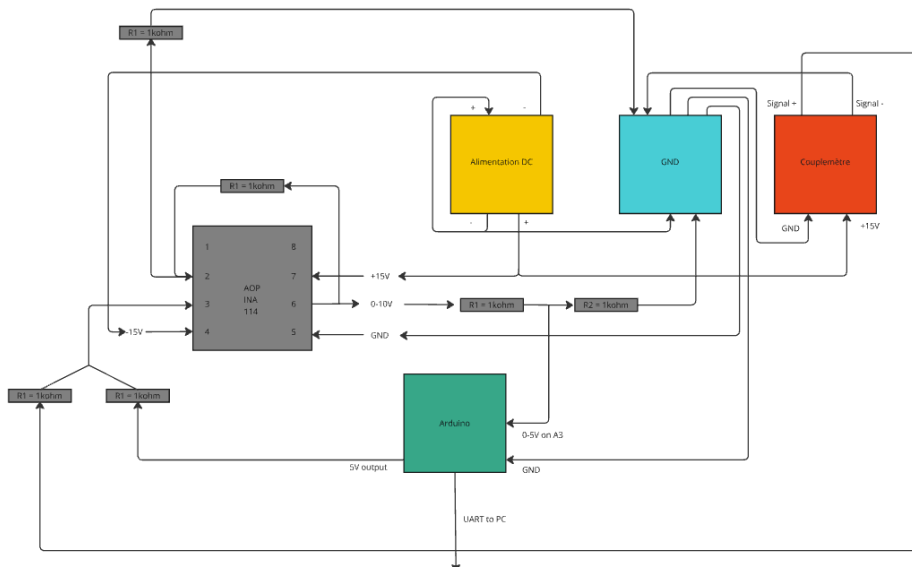


Figure 7 : Schéma de principe pour la récupération et le traitement du signal du

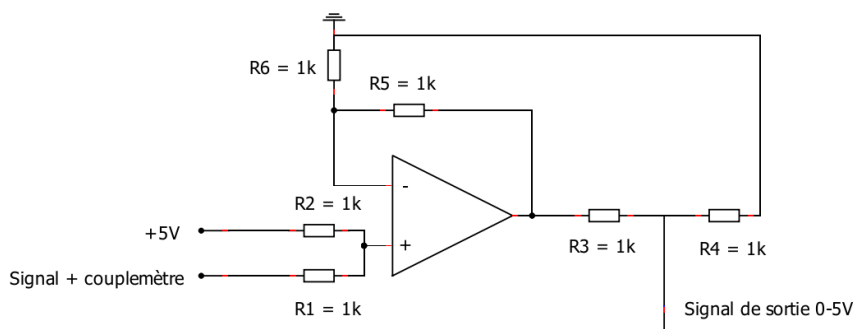


Figure 8 : Schéma pour la récupération et le traitement du signal du couplemètre

On réalise ensuite un code qu'on injecte dans l'Arduino pour récupérer le signal du couplémètre. Ce dernier est retraité dans le code pour afficher directement la valeur de couple mesurée. Notons que les fluctuations observées sont peu importantes puisque le couple nominal du couplémètre est de 15N.m tandis que le couple nominal du moteur asynchrone est de 0,9 N.m. Cela laisse donc présager une erreur plus importante sur les mesures par rapport à un couplémètre d'un couple nominal inférieur.

VI. Récupération du signal du capteur de courant

De manière analogue à la partie précédente où on récupérait et traitait le signal du couplemètre, on cherche ici à récupérer et à traiter le signal du capteur de courant LPSR-50NP. On repart pour cela d'un prototype laissé par nos prédécesseurs. Ce prototype n'ayant pas été testé, on commence par vérifier son fonctionnement.

Petite précision sur le branchement du capteur qui sera peut être utile à nos successeurs : le capteur utilisé possède un courant nominal de 50A. Or le courant absorbé par le moteur asynchrone ne devrait pas excéder quelques ampères en pics (moins d'1 ampère en fonctionnement nominal). Afin de réduire l'erreur lors de la mesure, le courant passe 4 fois dans le capteur avant de repartir en direction du moteur asynchrone (le capteur étant monté en série). Cela correspond donc au dernier montage (4 primary turns) sur la figure 9.

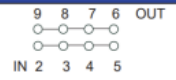
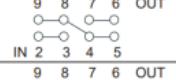
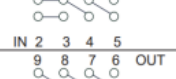
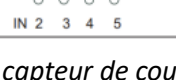
Number of primary turns	Primary Nominal RMS current	Output voltage U_{out}	Primary resistance R_p [mΩ]	Recommended connections
1	$\pm I_{PN}$	$U_{ref} \pm 0.625$	0.18	
2	$\pm I_{PN}/2$	$U_{ref} \pm 0.625$	0.72	
3	$\pm I_{PN}/3$	$U_{ref} \pm 0.625$	1.8	
4	$\pm I_{PN}/4$	$U_{ref} \pm 0.625$	2.88	

Figure 9 : Branchements possibles pour le capteur de courant

Ci-dessous, le détail des différents branchements possibles :

- U_c : c'est le + du capteur, on y branche le 5V
- GND : c'est le - du capteur, on y branche la masse
- V_{ref} : ce pin permet de centrer la valeur de la tension. Par défaut, les valeurs de courant sont centrées sur 2,5V. Cela signifie que si un signal de 2,5V est émis par le pin V_{out} , un courant nul parcourt le capteur. De manière analogue, si un signal de 5V est émis par le pin V_{out} , un courant de 50A parcourt le capteur. On comprend ici le potentiel d'erreur assez important : un dixième d'erreur sur la tension renvoyée correspond à 2A soit plus que notre courant nominal !
- OCD : c'est un pin d'alerte. En cas de surcharge de courant, un signal sera renvoyé (valeur digitale 0 ou 1)
- V_{out} : c'est par ce pin que la tension 0-5V nous renseignant sur le courant passant dans le capteur nous parvient (valeur analogique 0-1023)

On réalise ensuite un code pour récupérer et traiter le signal envoyé par le couplemètre. Ce signal étant directement renvoyé en 5V aucune modification n'est nécessaire. Le code consiste donc en quelques manipulations afin d'afficher la bonne valeur de courant dans le moniteur série de l'IDE Arduino ne partant de la tension V_{out} . Le script du code est disponible en annexes.

```
Code_captur_courant | Arduino 1.8.19 (Windows Store 1.8.57.0)
Fichier Édition Croquis Outils Aide

Code_captur_courant
#define VOLT_MEAS A2
#define OCKT_ALARM A1

float voltage = 0;
float mapped_voltage = 0;
float intermediate_voltage = 0;
float current = 0;
const int nominal_current = 50;

void setup() {
  Serial.begin(9600);
  pinMode(VOLT_MEAS, INPUT);
  pinMode(OCKT_ALARM, INPUT);
}

void loop() {
  // Lecture de la tension analogique qui représente le courant :
  voltage = analogRead(VOLT_MEAS);
  Serial.println(voltage);

  // La valeur lue est sur la plage 0-1023
  // Il faut la passer sur la plage -5V à 5V pour l'utiliser dans la formule I = U/RH issue de la documentation

  // Etape intermédiaire pour vérifier la tension reçue par l'arduino
  mapped_voltage = map(100*voltage, 0, 102300, 0, 500); // Multiplication par 100 car la fonction map n'accepte que les entiers
  Serial.println("The voltage transmitted to Arduino is: ");
  Serial.println(mapped_voltage/100, 2); // Affiche 2 décimales
  Serial.println(" V");

  // Passage à la tension sur la plage -5 à 5V
  intermediate_voltage = map(mapped_voltage, 0, 500, -500, 500);
  //Serial.println(intermediate_voltage);

  // Calcul du courant (exemple : 0-5V correspond à 0-15A)
  current = 20*mapped_voltage/100-50;
  // current = intermediate_voltage/(4*100*0,0125);
  Serial.println("The current transmitted to induction motor is: ");
  Serial.println(current, 2);
  Serial.println(" A");

  // Vérification de surcharge
  if (digitalRead(OCKT_ALARM) == HIGH) {
    Serial.println("Warning! Overcurrent to induction motor");
  }

  delay(500); // Pause de 500 ms
}
```

Figure 10 : Code réalisé pour récupérer la valeur de courant absorbé par le MAS (script en annexe ...)

On teste ensuite le prototype avec une alimentation de laboratoire pour connaître la valeur d'intensité passant par le capteur et ainsi comparer avec la valeur fournie par le code. Les tests sont concluants : nous obtenons une précision au dixième pour la valeur de courant récupérée.

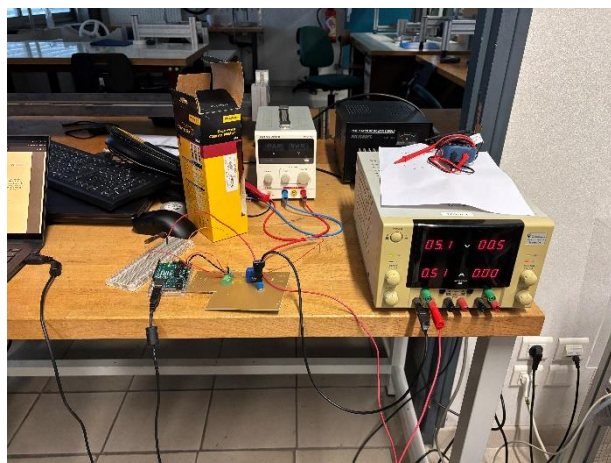


Figure 11 : Test du prototype de nos prédecesseurs

On verra dans la partie VIII l'actualisation du prototype pour obtenir un PCB propre.

VII. Amplification du signal Arduino pour contrôle du driver brushless

Dans cette partie du projet, nous avons travaillé sur l'amplification du signal de sortie de la carte Arduino afin de piloter le moteur brushless via un driver, tout en exploitant toute la plage de tension disponible. Cette partie présente les travaux réalisés lors de nos trois séances, les problèmes rencontrés et les solutions envisagées pour la suite.

Conception et réalisation du circuit :

Schéma d'amplification :

Durant la phase initiale, plusieurs options technologiques ont été envisagées. Nous avons finalement opté pour l'utilisation d'un amplificateur opérationnel (AOP) en raison de sa simplicité d'intégration et de son adaptabilité à notre application. Le modèle LM386N-3 a été choisi pour réaliser le circuit d'amplification. Avant la mise en œuvre, les caractéristiques fréquentielles de l'AOP ont été analysées pour garantir une compatibilité avec la fréquence PWM de l'Arduino, qui est inférieure ou égale à 490 Hz.

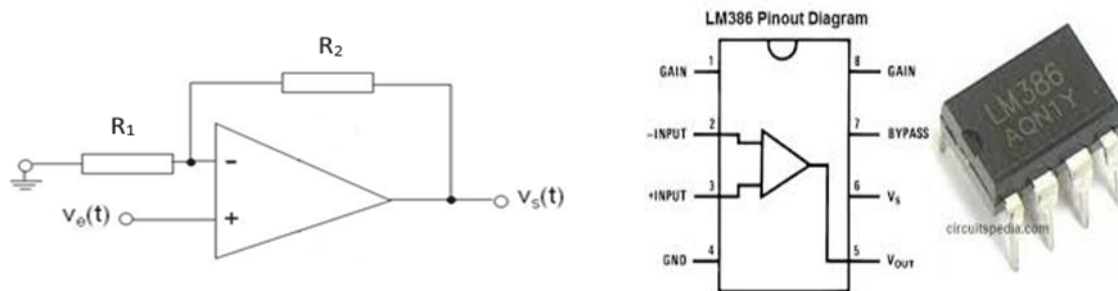


Figure 12 : Schéma du circuit d'amplification et AOP utilisé

Avec ce schéma, nous avons : $V_s(t) = (1 + \frac{R_2}{R_1})V_e(t) \rightarrow$ le gain : $G = 1 + \frac{R_2}{R_1}$. Où $V_e(t)$ est la tension d'entrée et $V_s(t)$ la tension de sortie de l'AOP. Nous avons donc choisi $R_1=R_2$ pour obtenir enfin le gain $G = 2$ et $V_s(t) = 2V_e(t)$.

Réalisation du circuit d'amplification :

Le circuit a été monté sur une plaque d'essai en suivant le schéma conçu. Avant de connecter le moteur brushless, le circuit a été testé avec une charge résistive afin de vérifier la tension de sortie. Une mesure à l'aide d'un multimètre a confirmé une amplification correcte à 10V, validant l'efficacité du montage.

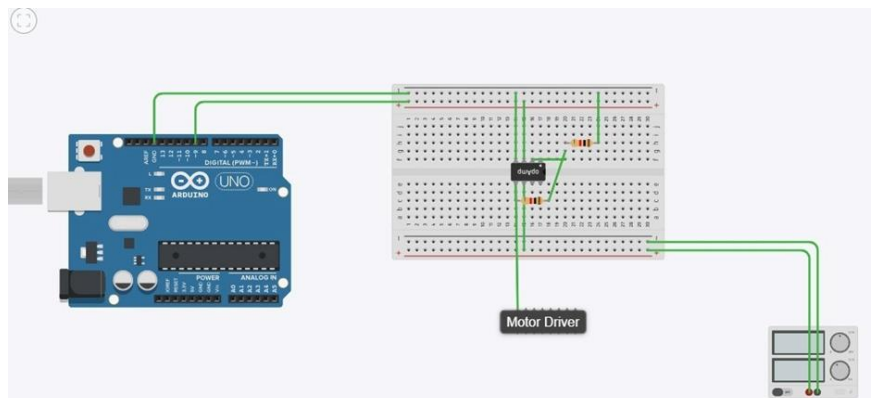


Figure 13 : Circuit d'amplification

- **Problème Rencontré** : Le signal amplifié obtenait des valeurs fixes de 0V ou 10V, ce qui est inadapté pour le contrôle progressif du moteur.
- **Solution Proposée** : Écrire un programme Arduino capable de moduler progressivement la tension de sortie de 0V à 10V (démarrage) et de 10V à 0V (arrêt), assurant un contrôle fluide.

Programmation et contrôle :

Algorithme de modulation :

Un algorithme de modulation par largeur d'impulsion (PWM) a été implémenté pour contrôler la tension appliquée au moteur. L'algorithme prévoit :

- Une augmentation progressive de la tension de 0V à 10V au démarrage.
- Une diminution progressive de la tension de 10V à 0V lors de l'arrêt.

Réimplantation du code de pilotage du moteur :

Nous avons adapté le code de pilotage existant en y ajoutant une modulation de tension. Cette amélioration permet un contrôle plus précis de la vitesse du moteur. Les modifications incluent :

- L'utilisation de la modulation PWM pour ajuster la tension appliquée.
- Le calcul des cycles de service pour moduler la vitesse du moteur selon différents niveaux.

```

sketch_dec16a | Arduino IDE 2.3.4
Fichier Modifier Croquis Outils Aide
Arduino Uno

sketch_dec16a.ino
1 int increm = 0; // Variable pour l'augmentation/diminution progressive de la tension const int choixPin = 0; // Pin pour la commande de vitesse
2 const int motorOnOffPin = 8; // Broche pour allumer/éteindre le moteur (digitale) const int motorDirectionPin = 7; // Broche pour le sens de rotation (digitale)
3 bool motorForward = true; // Direction du moteur
4 void setup() {
5   Serial.begin(9600);
6   pinMode(choixPin, OUTPUT);
7   pinMode(motorOnOffPin, OUTPUT); // Configure la broche d'allumage/éteignage en sortie
8   pinMode(motorDirectionPin, OUTPUT); // Configure la broche de direction en sortie digitalWrite(motorOnOffPin, LOW); // Éteint initialement le moteur digitalWrite
9   Serial.println("Entrez 'on' pour allumer le moteur, 'off' pour l'éteindre."); Serial.println("'forward' pour avancer, 'backward' pour reculer."); }
10 void loop() {
11   if (Serial.available() > 0) {
12     String input = Serial.readString();
13     input.trim(); // Supprimer les espaces en début et en fin de chaîne
14     if (input == "on") {
15       motorOn = true;
16       Serial.println("Moteur allumé.");
17     } else if (input == "off") {
18       motorOn = false;
19       Serial.println("Moteur éteint.");
20       digitalWrite(choixPin, 0); // Assure que le moteur est éteint } else if (input == "forward") {
21       motorForward = true;
22       Serial.println("Direction : Avant.");
23     } else if (input == "backward") {
24       motorForward = false;
25       Serial.println("Direction : Arrière.");
26     } else {
27       Serial.println("Commande invalide. Entrez 'on', 'off', 'forward' ou 'backward'."); }
28   }
29   // Si le moteur est allumé et une direction est définie
30   if (motorOn) {
31     digitalWrite(motorOnOffPin, HIGH); // Allume le moteur
32     // Changer la direction du moteur
33     if (motorForward) {
34       digitalWrite(motorDirectionPin, HIGH); // Avant
35     } else {
36       digitalWrite(motorDirectionPin, LOW); // Arrière
37     }
38     // Augmente la tension de 0 à 5V (0 à 255) en 5 secondes
39     for (increm = 0; increm <= 255; increm++) {
40       analogWrite(choixPin, increm);
41       delay(20); // 20 ms pour chaque pas -> 255 * 20 ms = ~5 secondes }
42     // Maintient la tension à 5V pendant 10 secondes
43     analogWrite(choixPin, 255);
44     delay(10000);
45     // Diminue la tension de 5V à 0V en 5 secondes
46     for (increm = 255; increm >= 0; increm--) {
47       analogWrite(choixPin, increm);
48       delay(20); // 20 ms pour chaque pas -> 255 * 20 ms = ~5 secondes }
49     // Éteint le moteur une fois le cycle terminé
50     motorOn = false; // Réinitialise l'état
51     digitalWrite(motorOnOffPin, LOW); // Éteint le moteur
52     Serial.println("Cycle terminé, moteur éteint.");
53   }
54 }

```

```

sketch_dec16a | Arduino IDE 2.3.4
Fichier Modifier Croquis Outils Aide
Arduino Uno

sketch_dec16a.ino
29 // Si le moteur est allumé et une direction est définie
30 if (motorOn) {
31   digitalWrite(motorOnOffPin, HIGH); // Allume le moteur
32   // Changer la direction du moteur
33   if (motorForward) {
34     digitalWrite(motorDirectionPin, HIGH); // Avant
35   } else {
36     digitalWrite(motorDirectionPin, LOW); // Arrière
37   }
38   // Augmente la tension de 0 à 5V (0 à 255) en 5 secondes
39   for (increm = 0; increm <= 255; increm++) {
40     analogWrite(choixPin, increm);
41     delay(20); // 20 ms pour chaque pas -> 255 * 20 ms = ~5 secondes }
42   // Maintient la tension à 5V pendant 10 secondes
43   analogWrite(choixPin, 255);
44   delay(10000);
45   // Diminue la tension de 5V à 0V en 5 secondes
46   for (increm = 255; increm >= 0; increm--) {
47     analogWrite(choixPin, increm);
48     delay(20); // 20 ms pour chaque pas -> 255 * 20 ms = ~5 secondes }
49   // Éteint le moteur une fois le cycle terminé
50   motorOn = false; // Réinitialise l'état
51   digitalWrite(motorOnOffPin, LOW); // Éteint le moteur
52   Serial.println("Cycle terminé, moteur éteint.");
53 }
54 }
55
56
57
58
59

```

Figure 14 : Code de régulation de tension, script en annexe

Pilotage du moteur :

Lors du test, le moteur a été connecté à une alimentation stabilisée. Nous avons réussi à démarrer le moteur brushless sur toute la plage de tension 0-10V. Pour ce faire, nous avons utilisé le circuit amplificateur de l'AOP, présenté précédemment. Dans le code de pilotage, nous avons défini la commande de vitesse sur la broche 9 de l'Arduino. Tout d'abord, notre code Arduino permet une variation de la tension entre 0-5V à la sortie de la broche 9. Nous avons mis un second circuit d'amplification de l'AOP en aval au circuit de l'Arduino. Il faut noter qu'il s'agit d'une amplification avec un gain égal à 2. Finalement, nous avons une tension qui varie entre 0-10V à la sortie de l'AOP. Plus précisément, dans le code Arduino, on a trois phases, une phase où la tension augmente progressivement pendant 5 secondes pour atteindre 5V, ensuite la tension reste constante (5V) pendant 10 secondes, puis décroît progressivement jusqu'à s'annuler : 0V. Le dernier bloc du code se charge d'éteindre le moteur, pour pas continuer le cycle inutilement. Dès lors qu'on exécute le code, l'interface Arduino nous demande d'entrer premièrement On ou Off pour allumer le moteur ou le laisser éteint. Ensuite on doit entrer « forward » ou « backward », pour définir le sens de rotation. La principale difficulté qu'on a eu pour la réalisation des deux circuits est l'agencement du circuit d'amplification à la suite de l'Arduino et la connexion de la connexion de la sortie de l'Arduino à la commande de vitesse de la carte de commande du moteur. Mais nous avons réussi à le faire, pour ça il faut juste utiliser un fil relié à la broche 9 pour l'entrée non-inverseuse de l'AOP, utiliser un fil sortant de la masse de l'Arduino pour le connecter à l'entrée inverseuse de l'AOP. Ensuite pour l'alimentation de l'AOP, il faut prendre du 0V et 15V au générateur mis à disposition. L'alimentation de la carte de commande n'a pas posé de problème, car il y avait un deuxième générateur. Cette manière de commander la tension de sortie au niveau de la broche 9 de l'Arduino résulte sur une variation de la tension à la sortie de l'AOP comme suit : une première augmentation entre 0 à 10V pendant 5 secondes suivi d'une tension constante à 10V pendant 10 secondes, puis une diminution progressive pour s'annuler 0V pendant 5 secondes.

VIII. Conception d'une carte électronique pour centraliser les capteurs

L'ensemble des montages pour la récupération et le traitement des signaux des différents capteurs étant désormais connus, on décide de proposer une carte électronique permettant de centraliser les fonctions suivantes :

- Alimentation et récupération du signal du couplemètre
- Alimentation et récupération du signal du capteur de courant
- Alimentation et récupération du signal du capteur de vitesse
- Envoi du signal de contrôle pour le moteur brushless

La carte électrique, aussi appelée PCB, permettra de graver des pistes dans une plaque de cuivre afin d'éviter les câbles présents dans les montages Arduino sur breadboard. Ceci en prévision de la réalisation d'une boîte de contrôle complémentaire à l'armoire électrique déjà réalisée. Le circuit sera gravé sur une plaque de cuivre de dimension 100*160mm donnée par l'enseignant. Notons que ce PCB est le premier réalisé par l'étudiant chargé de cette mission. Aussi il est fort probable que les conventions du milieu ne soit pas totalement respectée. Cela ne devrait néanmoins pas empêcher le bon fonctionnement de la carte. Dernière précision : c'est le logiciel Fusion 360 qui a été utilisé pour créer ce PCB. Si des modifications devaient se faire sur ce dernier, les fichiers de conception sont disponibles sur Windchill.

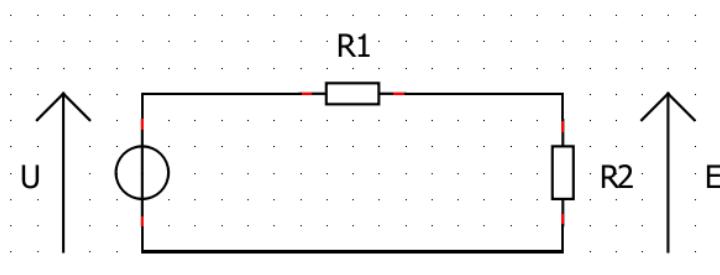
Afin de débiter sur de bonnes bases, on commence par récapituler les différents composants et leurs alimentations associées :

Composants	Plage d'alimentation théorique	Tension d'alimentation choisie
Capteur de vitesse	24V et GND	24V et GND
Couplemètre	12 à 28V et GND	12V et GND
AOP pour couplemètre	-18V à 18V et GND	-12V, 12V et GND
Capteur de courant	5V et GND	5V et GND
AOP pour signal brushless	4 à 12V et GND	12V et GND

Figure 15 : Tension d'alimentation de chaque composant du système

On observe donc que plusieurs tensions sont utilisées pour l'alimentation de nos composants. Pour faciliter l'utilisation et éviter d'utiliser plusieurs générateurs, on choisit d'alimenter l'ensemble avec un +24V, un -24V ainsi qu'un GND. Des conversions de tension seront ensuite réalisées par l'intermédiaire de PDT dans le PCB. On recense donc 3 ponts diviseurs :

- Pont n°1 : passage d'un signal +24V à un signal +12V. En utilisant la formule classique de pont diviseur obtenue par loi des mailles, on trouve que $R_1 = R_2$. On choisit ensuite des résistances de 2kOhm pour ne pas dépasser les 0,25W admissibles par la résistance.



$$E = \frac{R_2}{R_1 + R_2} U$$

Figure 16 : Pont diviseur de tension et formule associée

- Pont n°2 : passage d'un signal -24V à un signal -12V. Exactement comme précédemment, on choisit $R_1 = R_2 = 2\text{k}\Omega$.
- Pont n°3 : passage d'un signal +24V à un signal +5V. Dans ce cas, on résout le petit système suivant. En utilisant des valeurs usuelles de résistance, on trouve : $R_1 = 5,6\text{ k}\Omega$ et $R_2 = 1,5\text{ k}\Omega$.

$$\begin{cases} \frac{R_2}{R_1 + R_2} = \frac{5}{24} \\ \frac{U^2}{R_1} < 0,25 \\ \frac{E^2}{R_2} < 0,25 \end{cases}$$

Pour générer les signaux +24V, -24V et GND, on utilisera un générateur à 2 sorties. Le lien entre le générateur et la carte sera assuré par des connecteurs bananes comme présenté ci-dessous. De même, pour récupérer les signaux, on utilisera des borniers 2 pins, également présenté ci-dessous.



Figure 17 : Connecteur banane femelle, bornier 2 pin et header pin utilisés sur le PCB

Enfin, pour envoyer les signaux traités à l'Arduino, une série de 10 pin sera disponible grâce à un header pin. Ci-dessous le détail des signaux à renvoyer à l'Arduino :

- Sortie du capteur de vitesse (2 pins)
- Connection Arduino to GND (1 pin)
- Sortie couplemètre (1 pin)
- Sortie capteur de courant (2 pins : OCD et Vout)
- Entrée pour contrôle driver brushless (1pin)
- Entrée pour capteur de courant (1pin : Vref)

Une fois ces calculs préliminaires et le choix de matériels réalisés, on passe à la conception sur le logiciel Fusion 360, module électronique. Le logiciel propose une conception en plusieurs temps : on commence par attacher la conception électrique à une esquisse dont on a préalablement défini la taille (100x160mm dans notre cas). On réalise ensuite un schéma électrique de l'ensemble avant de réaliser le routage puis de générer le rendu 3D de la carte.

La principale difficulté rencontrée fût le fait que les composants (capteurs de courant, AOP...) n'étaient pas tous présents dans la bibliothèque de composants du logiciel. Il a donc fallu créer les composants en leur créant un schéma électrique, un « footprint » (positionnement des pins sur le PCB) et en associant à l'ensemble une CAO. Ci-dessous l'exemple de la création du capteur de courant dans Fusion 360.

- Création du schéma électrique : cette étape est relativement simple, il s'agit de déclarer au logiciel le nombre de pin présents sur le composant et de positionner ces derniers de manière judicieuse pour se faciliter la tâche lors de l'intégration dans le schéma électrique global.

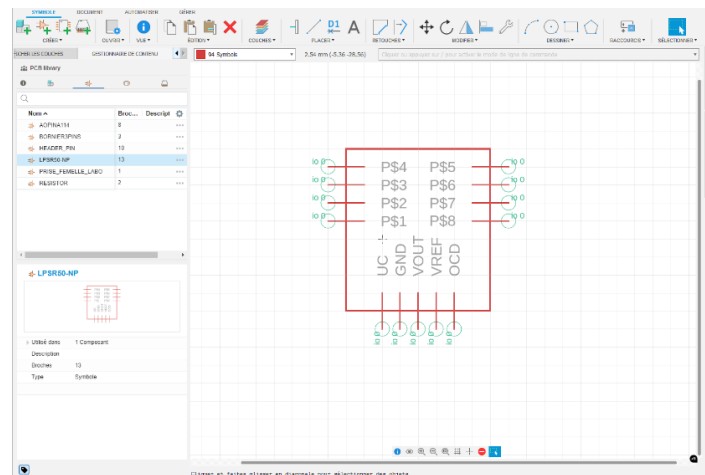


Figure 18 : Représentation schématique du capteur de courant

- Création du footprint : pour cela, on reprend la documentation du composant et on place nos trous de manière à respecter cette dernière.

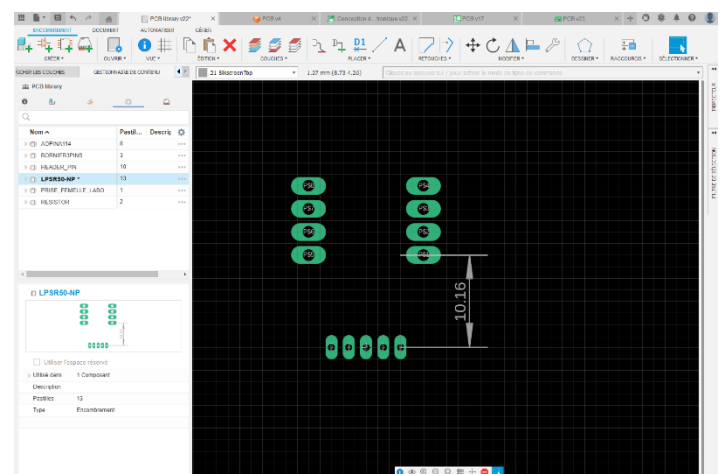
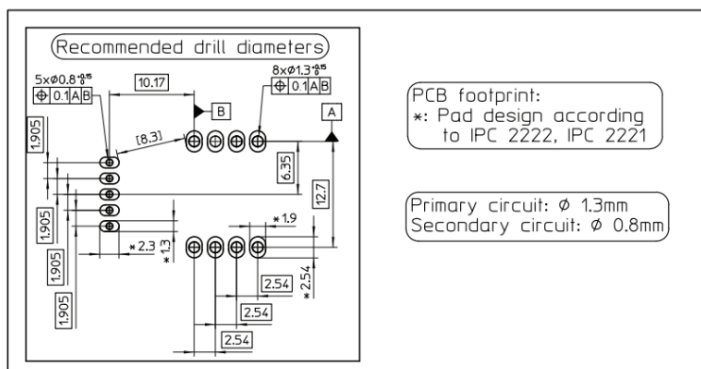


Figure 19 : Réalisation du footprint du capteur de courant

- Association d'une CAO à l'ensemble : pour cela on se rend sur le site constructeur, on télécharge le fichier step associé au composant et on place ce dernier sur l'empreinte précédemment réalisée (convertie en esquisse pour le placement du composant).

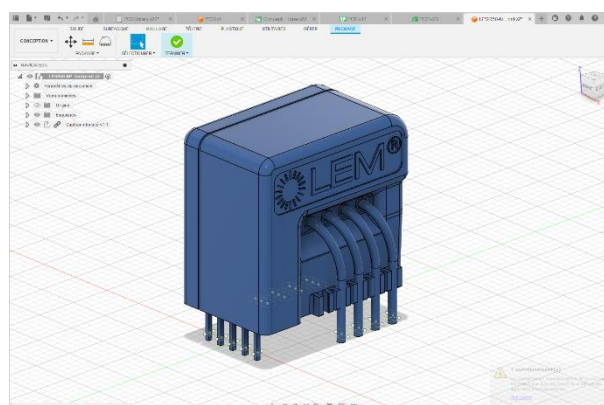


Figure 20 : Association d'un modèle 3D avec le footprint réalisé

En procédant de manière analogue pour les autres composants (résistance, AOP, trou pour les connecteurs bananes), on arrive à créer notre propre bibliothèque de composant. Une fois cette dernière constituée, on passe à la réalisation du schéma électrique global. Il s'agit ici de relier les pins préalablement défini lors de la conception de chaque composant. Ces liaisons seront ensuite utilisées pendant la phase de routage.

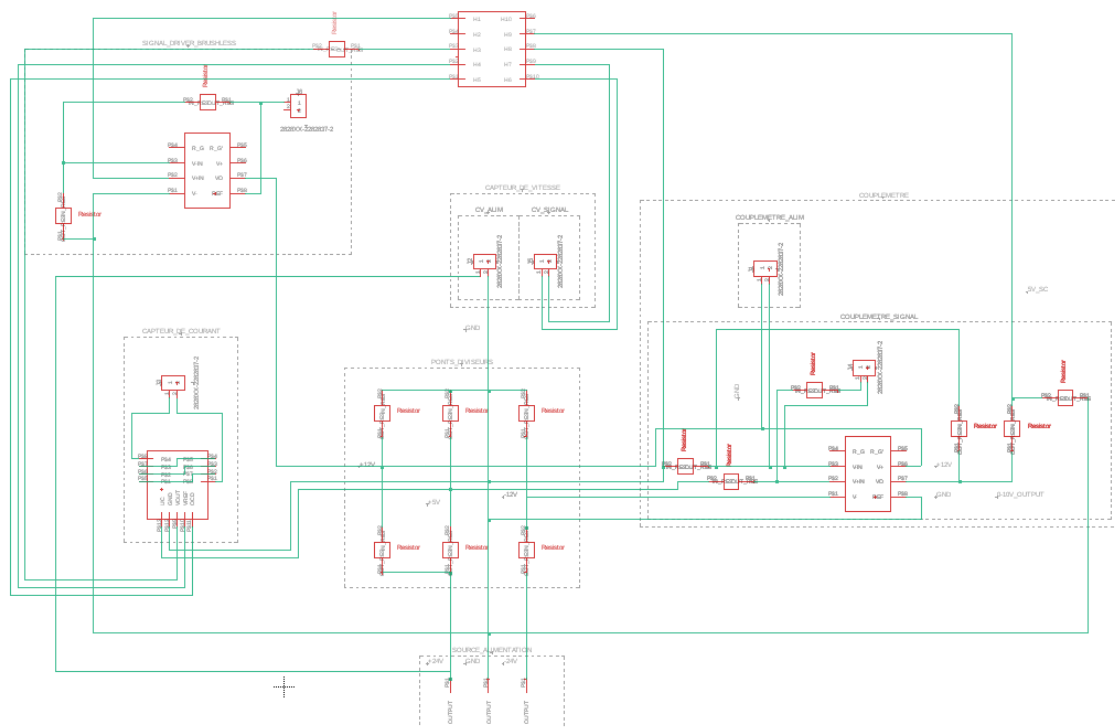


Figure 21 : Représentation schématique du PCB

La phase de routage consiste ensuite à matérialiser les liaisons entre les différents composants. Cette phase est quelque peu fastidieuse puisqu'il faut s'assurer que les pistes n'interfèrent pas entre elles. Ci-dessous, une première capture d'écran pour un PCB réalisé avec la technologie et une seconde pour un PCB réalisé en gravant une plaque de cuivre. Notons que les pistes font ici une largeur de 6mil soit environ 0.16mm.

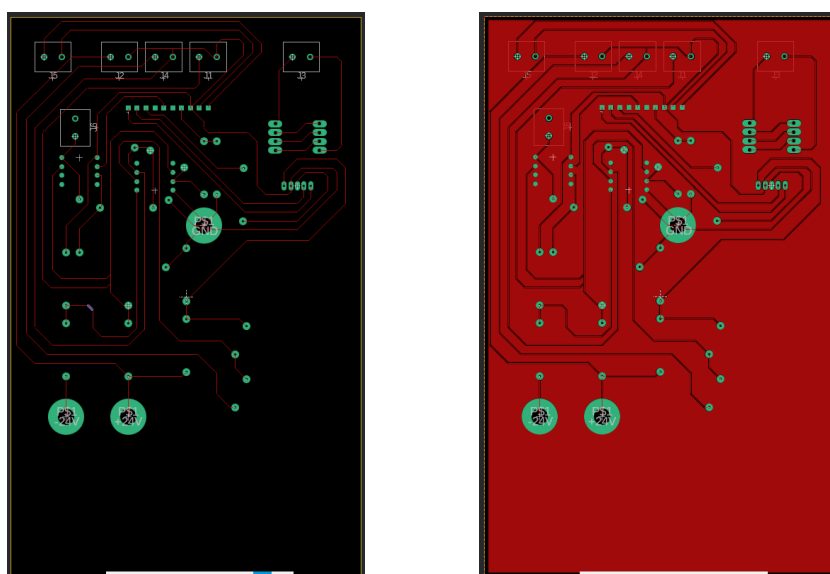


Figure 22 : Routage du PCB, V1

On conclut ensuite en générant le rendu 3D de la carte qui permet de vérifier le placement des composants et d'éventuellement ajuster ce dernier.

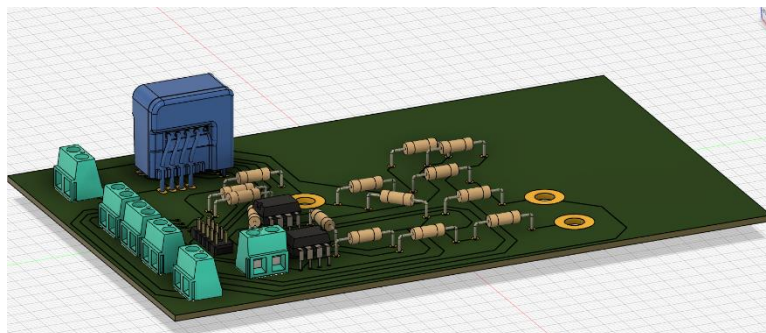
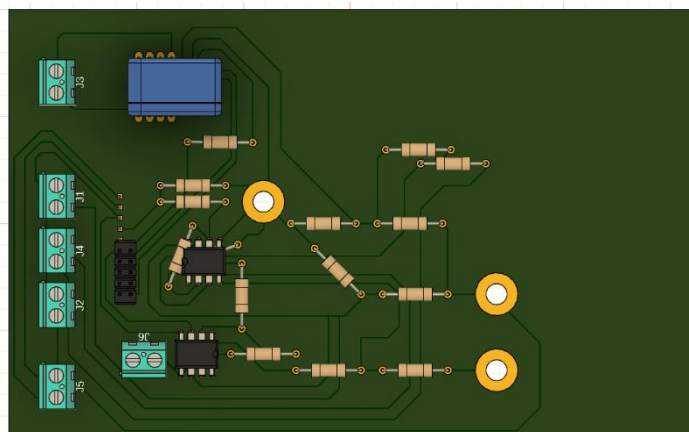


Figure 23 : Modélisation 3D de la carte, V1

La version présentée dans les pages ci-dessus a néanmoins connue une évolution suite à quelques échanges avec M. Didon qui gère la graveuse présente au MindTech. Voici les points modifiés :

- La gravure était auparavant réalisée sur la couche supérieure (top) de la plaque. C'est sur la couche inférieure que ces dernières doivent être réalisées. Sur le logiciel, cela se matérialise par des pistes bleues et non rouge
- Intégration d'un plan de masse qui évite les pistes pour relier toutes les terres : désormais l'ensemble du cuivre sur la carte est relié à la terre
- Agrandissement des pads de soudure
- Augmentation de la largeur des pistes : passage de 6 à 16mil (millième de pouces)
- Amélioration du routage (pistes plus directes, conventions usuelles davantage respectées)

Ci-dessous, la nouvelle version de la carte réalisée :

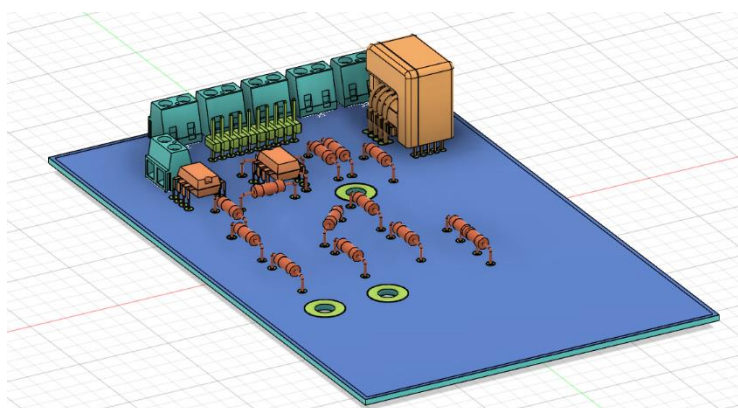
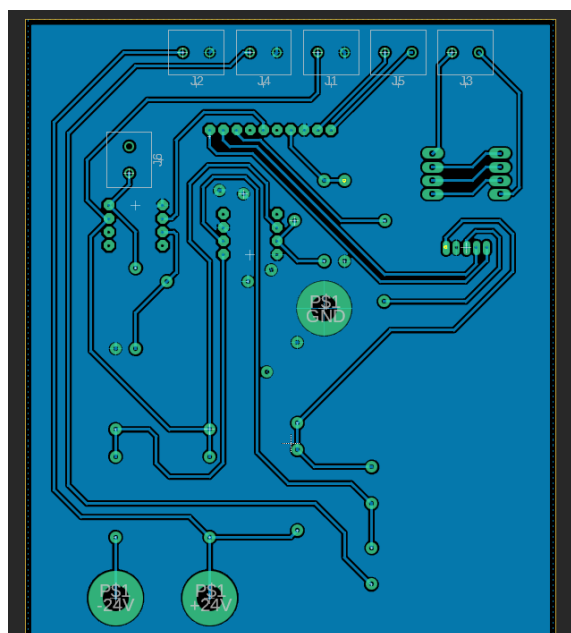


Figure 24 : Routage et Modélisation 3D de la carte, V2

Malheureusement, la graveuse n'étant pas fonctionnelle en fin de semestre A24, il revient à l'équipe A25 de graver et d'assembler le PCB. Voici pour cela le détail de l'assemblage de ce dernier :

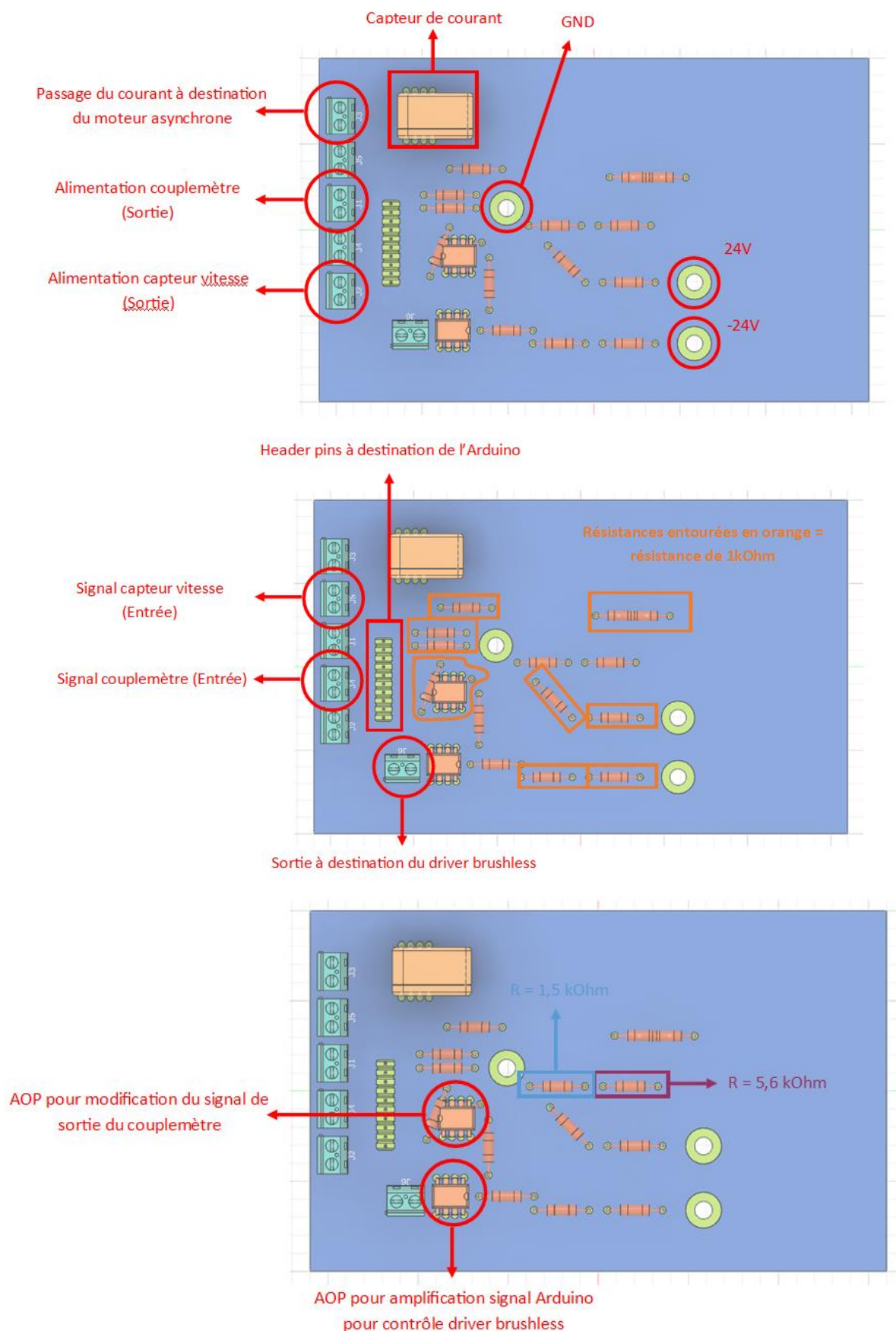


Figure 25 : Indications de montage

Ordre de branchements pour les pins de l'Arduino :

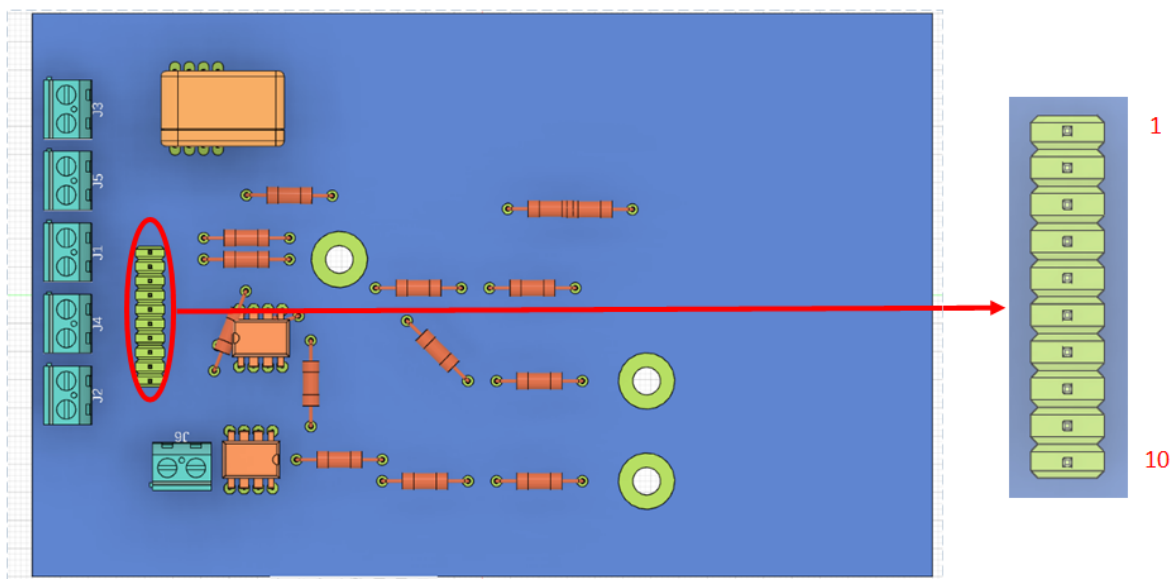


Figure 26 : Indications de branchements

- Pin 1 : Entrée A capteur de vitesse
- Pin 2 : Entrée B capteur de vitesse
- Pin 3 : GND
- Pin 4 : Signal de sortie couplemètre
- Pin 5 : libre
- Pin 6 : Signal pour le contrôle du moteur brushless
- Pin 7 : libre
- Pin 8 : Signal de sortie pour capteur de courant (indication de la valeur de courant, analogique)
- Pin 9 : Signal de sortie pour capteur de courant (indication d'une surcharge, digital)
- Pin 10 : Signal d'entrée pour capteur de courant ($V_{\text{réf}}$)

La solution vient donc presque clé en main pour la prochaine équipe : il reste simplement à positionner des trous (M4) permettant le maintien de la carte. La boîte d'intégration de la carte n'étant pas encore choisie, nous laissons soin à la prochaine équipe de positionner les trous selon où bon lui semble. **Attention également à s'assurer que les borniers et connecteurs bananes considérés ici soient toujours disponible en halle J1. Auquel cas, il faudra potentiellement modifier les empreintes sur fusion (taille et positionnement des trous les un par rapport aux autres).**

IX. Bilan et suite

Bilan du semestre :

Concrètement, tous les objectifs fixés en début de documents ont été atteints. Cela a nécessité un peu plus de temps que les seules 5 séances proposées mais l'ensemble des signaux des capteurs sont désormais récupérés via une unique plateforme (PCB), le moteur brushless peut être exploité à ses pleines capacités et le système de frein est fonctionnel.

Pour la suite, il convient de tester le PCB réalisé en le gravant au MindTech lorsque la machine sera réparée. Nous invitons les prochains étudiants à comprendre en détail le fonctionnement de ce dernier avec la documentation proposée. Il n'est pas impossible que des erreurs soient présentes sur la PCB et des tests permettront de les révéler.

En parallèle des tests, la réalisation d'une boîte mécanique afin de stocker le PCB et l'Arduino pourra être réalisée, de manière analogue à ce qui a été fait sur les autres bancs d'essais.

X. Conclusion

Ce TP projet fût très enrichissant et a permis de mettre en application les concepts vu en CM et TD, notamment la régulation de tension via transformateur et redresseur. L'ensemble de l'équipe a progressé, chacun à son niveau : pour certains ce fût la découverte de l'environnement Arduino, pour d'autres l'apprentissage de la conception de carte électronique.

Il reste encore du travail mais le projet arrive sur la fin !

XI. Annexes avec contributions individuelles

Contributions individuelles :

Pierre Perocheau :

- Récupération signal couplemètre (montage et code associé)
- Implantation du capteur de courant (test du prototype et développement du code)
- Réalisation du PCB
- Gestion d'équipe

Muhammad Ieyan :

- Amplification du signal Arduino pour contrôle du driver brushless (Circuits et codes associés)
- Implémentation du code Arduino pour contrôler la tension délivrée au driver du moteur

M'Monwal WABEDO :

- Amplification du signal Arduino pour contrôle du driver brushless (Circuits et codes associés)
- Proposition du circuit d'amplification pour le driver brushless

ADJINOUE Kodjo Paul Vahana

- Proposition du montage amplificateur à base d'AOP pour le driver brushless
- Correction du code de pilotage du moteur Brushless

Maxime Dai Pra

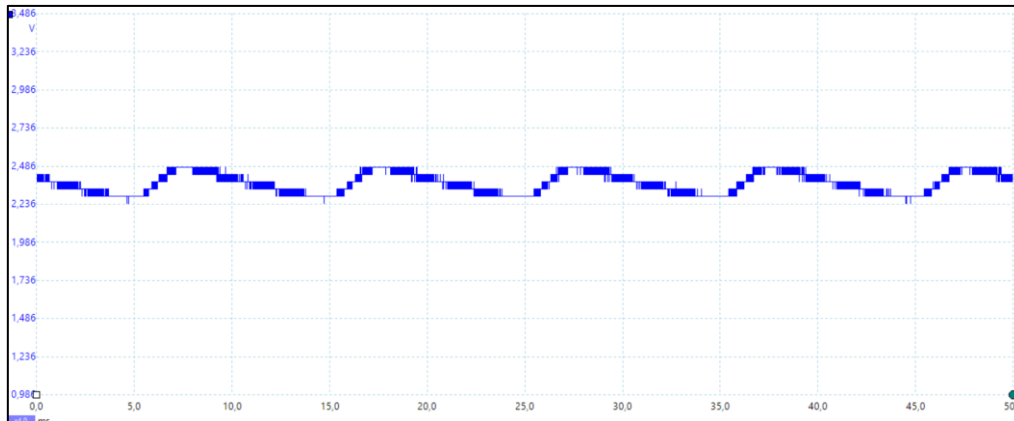
- Proposition d'un schéma électrique de l'armoire
- Assistance à la réalisation du système de contrôle des freins

Benjamin Redzic

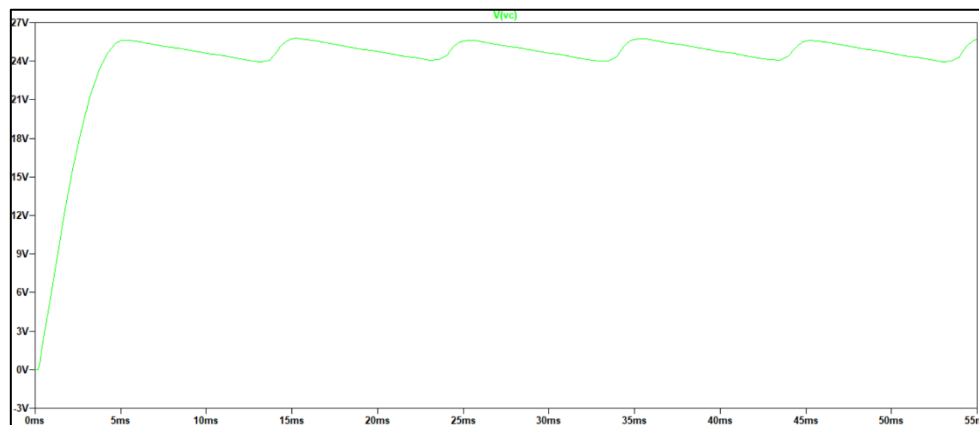
- Réalisation du circuit redresseur-lisseur pour le contrôle des freins.
- Implémentation du circuit dans l'armoire.
- Comparaison expérimentale (picoScope) et numérique (LTSpice).

Annexe 1 : Courbes condensateur 6800 μ F

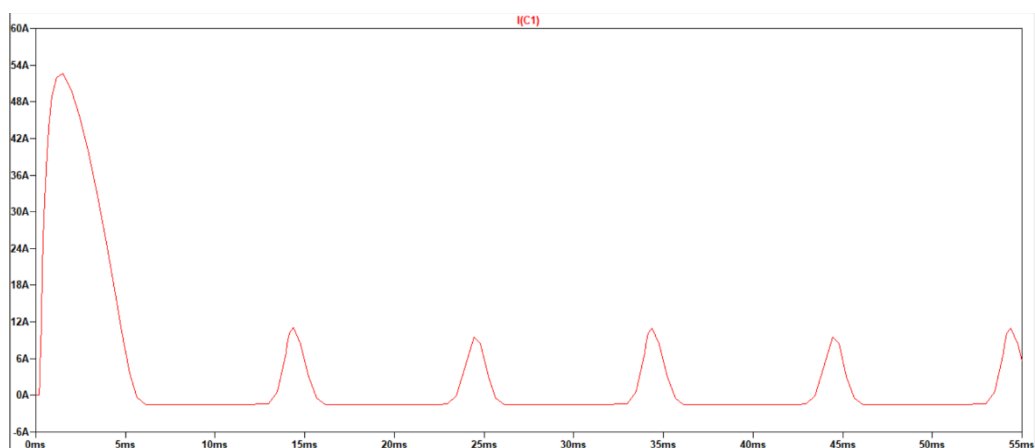
Uc mesurée à l'oscilloscope :



Uc obtenue sur LTSPice :

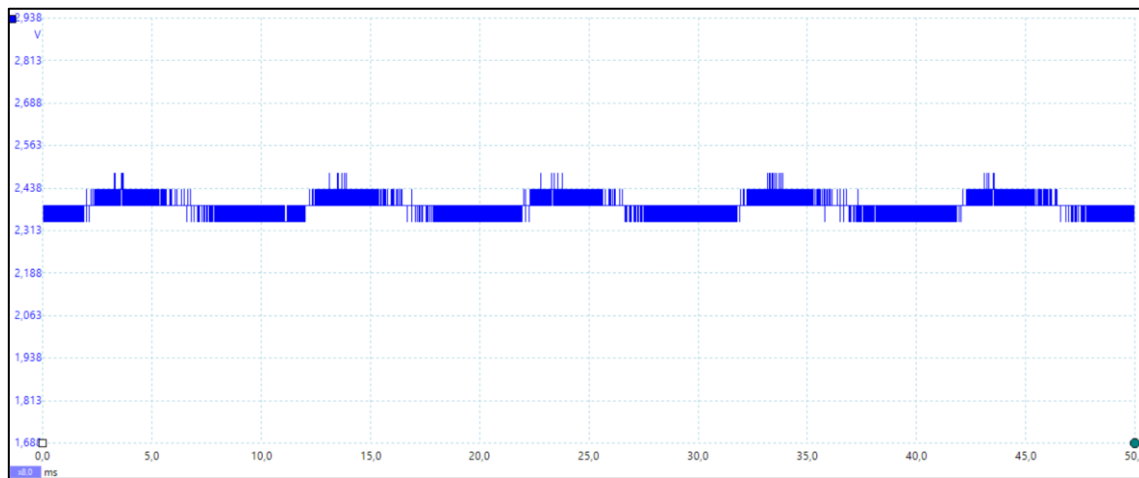


Ic obtenu sur LTSPice :

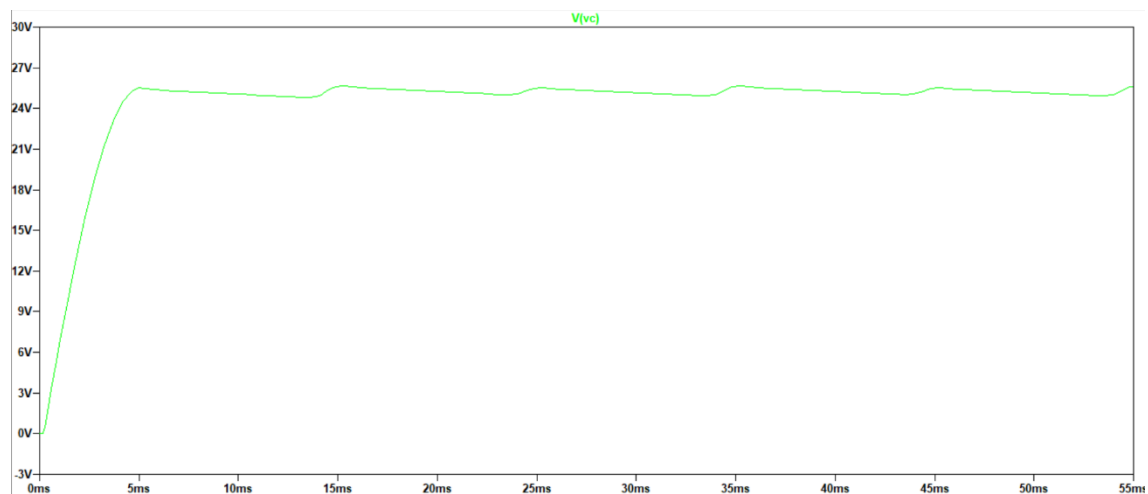


Annexe 2 : Courbes condensateur 22000 μ F

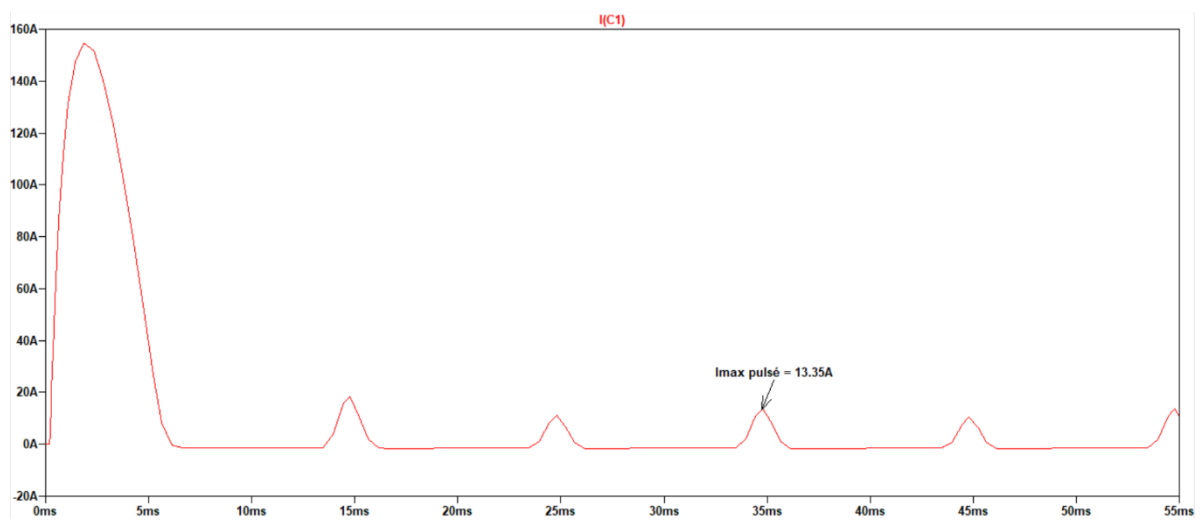
Uc mesurée à l'oscilloscope



Uc obtenue sur LTSPice :



Ic obtenu sur LTSPice :



Annexe 3 : Codes pour l'amplification du signal à destination du driver brushless

Code 1 : algorithme de modulation [-]

```
// Déclaration des constantes et variables

const int pwmPin = 9; // Pin PWM pour contrôler la sortie

int dutyCycle = 0; // Variable pour le cycle de service (0 à 255)

const int increment = 1; // Incrément pour la variation progressive

const int delayTime = 10; // Délai en millisecondes entre chaque incrément

const int pauseTime = 2000; // Temps de pause à 0% et 100% (en ms)

void setup() {

    // Initialisation de la communication série (facultative) et de la pin PWM
    Serial.begin(9600);

    pinMode(pwmPin, OUTPUT);
}

void loop() {

    // Augmentation progressive de la luminosité/tension
    while (dutyCycle < 255) {

        analogWrite(pwmPin, dutyCycle); // Applique le cycle de service actuel

        Serial.println(dutyCycle); // Affiche la valeur actuelle (facultatif)

        dutyCycle += increment; // Incrémente le cycle de service

        delay(delayTime); // Pause pour un effet progressif
    }

    delay(pauseTime); // Pause après avoir atteint 100%

    // Diminution progressive de la luminosité/tension
    while (dutyCycle > 0) {

        analogWrite(pwmPin, dutyCycle); // Applique le cycle de service actuel
```

```

Serial.println(dutyCycle); // Affiche la valeur actuelle (facultatif)

dutyCycle -= increment; // Décrémente le cycle de service

delay(delayTime); // Pause pour un effet progressif
}

delay(pauseTime); // Pause après être revenu à 0%
}

```

Code 2 : Code de Pilotage du Moteur modifié [-]

```

int increm = 0; // Variable pour l'augmentation/diminution progressive de la tension const int choixPin
= 9; // Pin pour la commande de vitesse

const int motorOnOffPin = 8; // Broche pour allumer/éteindre le moteur (digitale) const int
motorDirectionPin = 7; // Broche pour le sens de rotation (digitale) bool motorOn = false; // État du
moteur
bool motorForward = true; // Direction du moteur

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

  pinMode(choixPin, OUTPUT);
  pinMode(motorOnOffPin, OUTPUT); // Configure la broche d'allumage/éteignage en sortie
  pinMode(motorDirectionPin, OUTPUT); // Configure la broche de direction en sortie
  digitalWrite(motorOnOffPin, LOW); // Éteint initialement le moteur digitalWrite(motorDirectionPin,
  LOW); // Par défaut, tourne dans le sens "forward"

  Serial.println("Entrez 'on' pour allumer le moteur, 'off' pour l'éteindre,");
  Serial.println("'forward' pour avancer, 'backward' pour reculer."); }

void loop() {
  if (Serial.available() > 0) {
    String input = Serial.readString();
    input.trim(); // Supprimer les espaces en début et en fin de chaîne
    if (input == "on") {
      motorOn = true;
      Serial.println("Moteur allumé.");
    } else if (input == "off") {
      motorOn = false;
      Serial.println("Moteur éteint.");
    }
  }
}

```

```

analogWrite(choixPin, 0); // Assure que le moteur est éteint } else if (input
== "forward") {
    motorForward = true;
    Serial.println("Direction : Avant.");
} else if (input == "backward") {
    motorForward = false;
    Serial.println("Direction : Arrière.");
} else {
    Serial.println("Commande invalide. Entrez 'on', 'off', 'forward' ou 'backward'."); }
}

// Si le moteur est allumé et une direction est définie
if (motorOn) {
    digitalWrite(motorOnOffPin, HIGH); // Allume le moteur

    // Changer la direction du moteur
    if (motorForward) {
        digitalWrite(motorDirectionPin, HIGH); // Avant
    } else {
        digitalWrite(motorDirectionPin, LOW); // Arrière
    }

    // Augmente la tension de 0 à 5V (0 à 255) en 5 secondes
    for (incred = 0; incred <= 255; incred++) {
        analogWrite(choixPin, incred);
        delay(20); // 20 ms pour chaque pas => 255 * 20 ms = ~5 secondes }

    // Maintient la tension à 5V pendant 10 secondes
    analogWrite(choixPin, 255);
    delay(10000);
    // Diminue la tension de 5V à 0V en 5 secondes
    for (incred = 255; incred >= 0; incred--) {
        analogWrite(choixPin, incred);
        delay(20); // 20 ms pour chaque pas => 255 * 20 ms = ~5 secondes }

    // Éteint le moteur une fois le cycle terminé
    motorOn = false; // Réinitialise l'état
    digitalWrite(motorOnOffPin, LOW); // Éteint le moteur
    Serial.println("Cycle terminé, moteur éteint.");
}
}

```


Annexes 4 : Récupération du signal du capteur de courant

```
#define VOLT_MEAS A2

#define OCRT_ALARM A1


float voltage = 0;
float mapped_voltage = 0;
float intermediate_voltage = 0;
float current = 0;
const int nominal_current = 50;


void setup() {
  Serial.begin(9600);
  pinMode(VOLT_MEAS, INPUT);
  pinMode(OCRT_ALARM, INPUT);
}


void loop() {
  // Lecture de la tension analogique qui représente le courant !
  voltage = analogRead(VOLT_MEAS);
  Serial.println(voltage);


  // Etape intermédiaire pour vérifier la tension reçue par l'arduino
  mapped_voltage = map(100*voltage, 0, 102300, 0, 500); // Multiplication par 100 car la fonction map
n'accepte que les entiers
  Serial.print("The voltage transmitted to Arduino is: ");
  Serial.print(mapped_voltage/100, 2); // Affiche 2 décimales
  Serial.println(" V");


  // Calcul du courant (exemple : 0-5V correspond à 0-15A)
  current = 20*mapped_voltage/100-50;
```

```
// current = intermediate_voltage/(4*100*0,0125);  
Serial.print("The current transmitted to induction motor is: ");  
Serial.print(current, 2);  
Serial.println(" A");  
  
// Vérification de surcharge  
if (digitalRead(OCRT_ALARM) == HIGH) {  
    Serial.println("Careful! Overcurrent to induction motor");  
}  
  
delay(500); // Pause de 500 ms  
}
```