ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

Mercredi 1er Juin

Professeurs encadrants : M.PAPAZOGLOU Nicolas
M.MARTIN Alexis

1ère année
Semestre 6
Promo 2024

Projet d'électronique :

Projecteur Led asservi

Ludovic Sercy - Tom Charbonneau
Thomas Pham - Quentin De la Chaise
Bassem Barakat - Mathys Daniel
Karim Jerjoub - Thomas Mouneyrat

ENSEA
ETABLISSEMENT PUBLIC
Cergy-Pontoise(95)
Ministère de l'Enseignement Supérieur et de la Recherche

How we made it ?

Partie I : Définition du système

- I.1 Description du fonctionnement
- I.2 Architecture du système
- I.3 Définition des composants

Partie 2 : Bloc d'alimentation

- II.1 Valeurs de tensions et de courants
- II.2 Pilotage et PCB
  - II.2.1 Réalisation du schéma PCB
  - II.2.2 Réalisation du PCB

Partie 3 : Pilotage des moteurs

- III.1 Commande des moteurs

Partie 4 : Pilotage des Leds

- IV.1 Choix des Leds et des drivers
- IV.2 Réalisation
  - IV.2.1 Réalisation du schéma PCB
  - IV.2.2 Réalisation du PCB
- IV.3 Commande des Leds

Partie 5 : Création de la carte principale

- V.1 Choix du microprocesseur
- V.2 Communication avec le PC
- V.3 Réalisation
  - V.3.1 Réalisation du schéma PCB
  - V.3.2 Réalisation du PCB
- V.4 Définition des ports et code du microprocesseur

Partie 6 : Assemblage structurel

- VI.1 Réalisation d'une structure

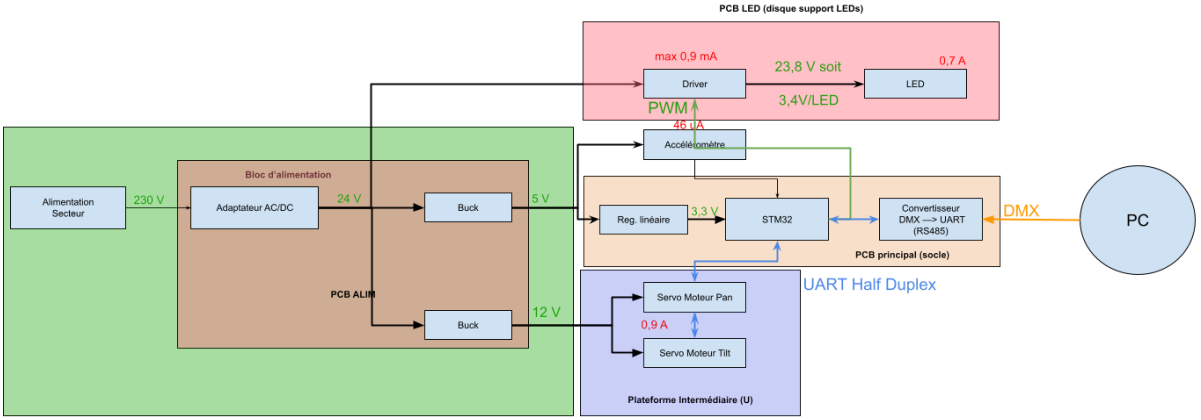
Partie 1 : Définition du système

I.1 Description du fonctionnement

Le système produit au cours de ce projet est un projecteur LEDs asservi. Notre projecteur peut se mouvoir suivant deux axes, le tilt et le pan et peut changer la couleur projetée. Il sera piloté par ordinateur, à travers une interface adaptée où l'on pourra choisir sa position et la couleur voulue. Par ailleurs, ce projecteur étant asservi, l'utilisateur aura accès à sa position suivant chaque axe par le biais d'un capteur.

I.2 Architecture du système

Le projecteur, dans sa fabrication a donc été segmenté en plusieurs blocs : un bloc alimentation qui prenant en entrée une tension secteur sera en charge de redistribuer de manière adaptée la tension à l'ensemble du système ; un bloc contrôle des leds, un bloc moteur et un bloc microprocesseur.
Le bloc alimentation sera alors composé d'un adaptateur alternatif vers continu ainsi que différents buck permettant l'adaptation de tension à chacune des trois sorties.
Le bloc contrôle des leds sera composé de drivers et de leds, les drivers par variation du signal envoyé aux leds permettront le contrôle de ces dernières.
La partie microprocesseur est le cœur du système c'est à travers ce bloc que les moteurs, ainsi que les leds sont contrôlés. Il reçoit les ordres donnés par l'utilisateur à travers l'adaptateur DMX→UART (uart en entrée du microprocesseur) et les transmets au reste du système. A noter que le microprocesseur n'est pas alimenté directement par le bloc d'alimentation, la tension d'entrée passe d'abord par un régulateur linéaire.
Comme indiqué dans son nom, on retrouvera les deux servomoteurs tilt et pan dans le bloc moteur. De plus, on observe un dernier bloc entre les autres, celui de l'accéléromètre, ce dernier est traité à part, il reçoit la même tension d'entrée que le régulateur linéaire et communique ses informations au microprocesseur.



I.3 Définition des composants

Adaptateur AC/DC : Permet la conversion d'une tension alternative en tension continue.

Buck : Convertit une tension continue en une tension continue plus faible.

Driver : Régule les systèmes d'éclairage (de type led), les alimente et contrôle leurs modes de fonctionnement.

Led : Composant électronique émettant de la lumière lorsqu'il est parcouru par un courant.

Accéléromètre : Capteur qui mesure une accélération et la convertit en signal électrique proportionnel.

Régulateur linéaire : Permet de réguler la valeur d'une tension continue.

STM32 : microprocesseur, il exécute et traite les informations données par le programme.

Convertisseur DMX → UART : Convertit une entrée de type DMX en sortie UART.

Servomoteurs : c'est un moteur capable de maintenir une opposition à un effort statique et dont la position est vérifiée en continu et corrigée en fonction de la mesure.

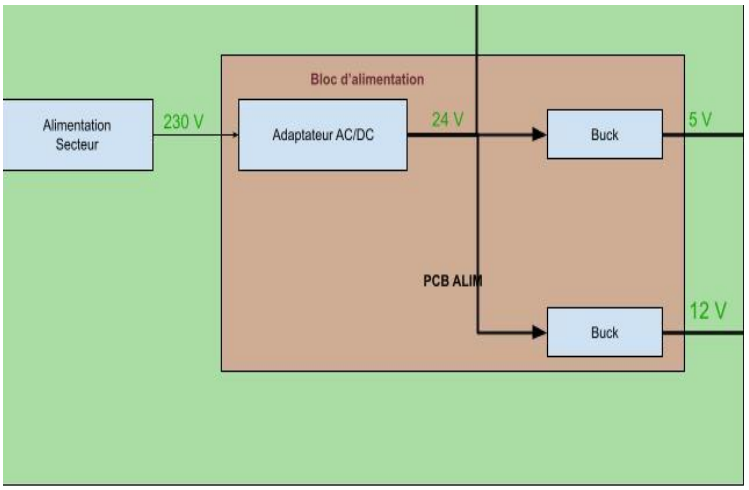
Pour cette partie on se référera à l'annexe partie 1

Partie 2 : Bloc d'alimentation

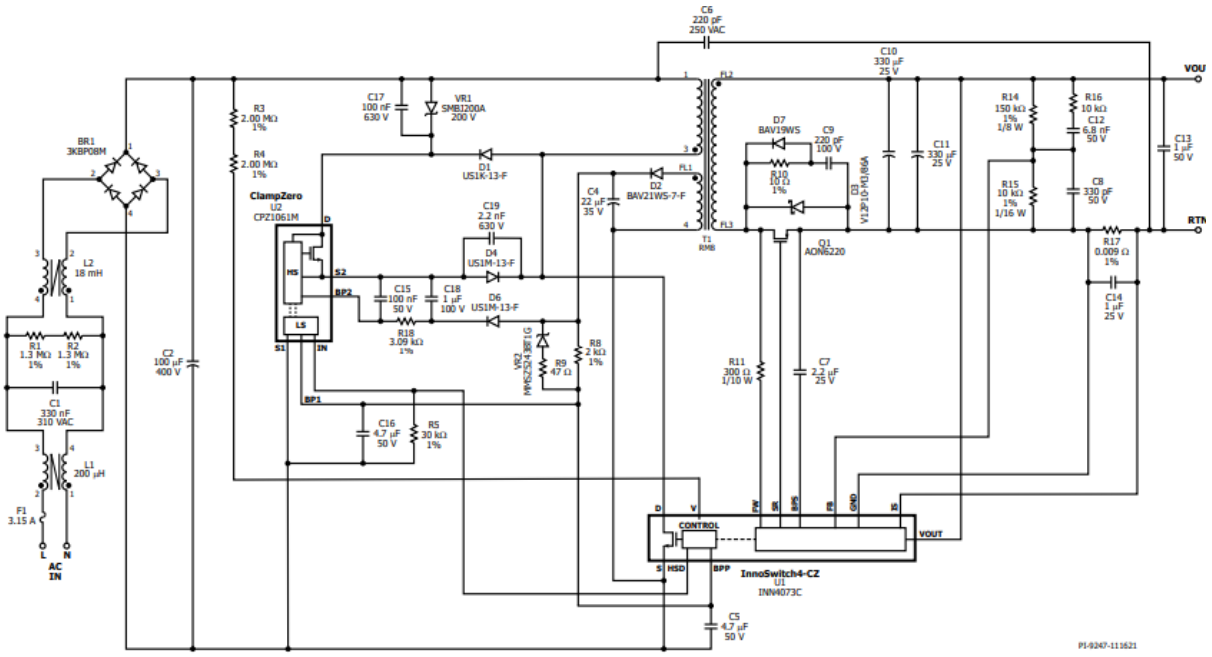
L'objectif de cette partie est d'élaborer un montage qui alimente chaque module du système en adéquation avec leurs besoins. Nous avons opté pour un montage Flyback, c'est-à-dire un montage modifié qui fournit via des montages Buck les valeurs de tensions et d'intensités désirées. En principe, le montage Flyback permet de convertir une tension analogique en une tension continue et d'adapter la tension à 24V, et le montage Buck permet de convertir une tension continue en une autre tension continue.

II.1 Valeurs de tensions et de courants

Les besoins en tensions et en courants de chaque module du système ont changé plusieurs fois.Il était donc important de modifier en adéquation à leurs attentes. Notre montage a dû s'y adapter aussi.
Les tensions et les intensités exigées sont récapitulées dans le schéma suivant:



On envoie une tension de 24V continue au moteur, donc on n'utilisera pas de Buck pour cela, puisque le Driver a besoin de 24V. De plus, les 24V sont envoyés à deux Buck, d'une part pour convertir la tension continue vers les LED en une tension abaissée de 5V et d'autre part cette tension est envoyé au microprocesseur avec une tension voulue de 12V. Pour répondre aux attentes, nous avons étudié et essayé de faire 4 montages à flyback différents auxquels s'ajoutent des montages de type Buck pour alimenter les modules du système.Suite à un manque de composant et aux délais de livraisons long nous avons finalement choisi le montage suivant en shuntant la partie filtrage du secteur.



Montage retenu validant les exigences

Le montage fonctionne via un pilotage switch en entrée du transformateur. Un feedback est prélevé en sortie pour pouvoir adapter les impulsions du switch envoyées au transformateurs. L'Innoswitch4 permet via le MOSFET de synchroniser la rectification en tension opérée sur la partie CONTROL à l'intérieur de l'Innoswitch4.
Plusieurs composants passifs et actifs servent à protéger le ClampZero notamment des retours de tension redressée qu'il peut subir.
Il y a également des zones de filtrages pour prélever un signal optimisé pour l'Innoswitch 4.

II.2 Pilotage et PCB

II.2.1 Réalisation du schéma PCB

Dès lors, nous avons procédé à la création de notre PCB sous eagle. Il a fallu créer des patterns pour les composants pilotant le système qui étaient manquantes, tels que le ClampZero, l'Innoswitch. Nous avons choisi des composants équivalents pour le transformateur notamment tout en respectant les caractéristiques du montage. La bibliothèque en ligne search components nous a été d'une grande aide pour trouver certaines patterns de composants. Cependant, nous n'avons pas dessiné tout le circuit électrique comme affiché plus haut, on a shunté quelques éléments nous permettant de s'adapter au mieux aux attentes des autres groupes. Le circuit en parallèle avec les diodes, le condensateur et le transistor plus bas après le transformateur n'a pas été pris en compte puisqu'il ne rentre pas dans nos critères. Ces changements ont été faits en accord avec la documentation qui explique le fonctionnement lorsque certaines parties étaient shuntées.
Les difficultés rencontrées lors du schématic ont été tout d'abord le temps qu'il nous a fallu pour trouver ce schematic adéquat avec le reste du groupe, puis la prise en main d'Eagle qui nous a fait perdre beaucoup de temps. Il faut éviter de recopier le circuit électrique tel

quel, mais de le simplifier au mieux en séparant le circuit électrique en plusieurs groupes de fonctions. Ainsi la visibilité en est meilleure et les erreurs qui y suivent sont moindre.

II.2.2 Réalisation du PCB

La réalisation du PCB a nécessité une formation au logiciel eagle. Il a fallu un temps d'adaptation. Après plusieurs tentatives, nous avons abouti à une réalisation correspondant aux exigences.Malheureusement, à la suite d'un nouveau manque de composant nous avons dû réadapter le montage et donc réaliser un nouveau PCB. Les patterns ont été elles aussi adaptées tout au long de la conception et nous ont pris beaucoup de temps. De plus, les largeurs de pistes sont fonction de la tension qui les traverse. C'est pourquoi nos différents routages ont été défaillants. Il faut prendre en compte la largeur des pistes avant de procéder au routage. De plus l'espacement entre chaque piste doit être de 1.5mm minimum pour éviter tout phénomène d'arcs électriques entre les pistes. Ceci est en particulier vrai pour les pistes de tensions redressées. Durant notre réalisation PCB, nous nous sommes rendus compte qu'un composant n'avait pas sa bibliothèque propre, il a fallu le rajouter dans la librairie, cependant lorsqu'il fallait remplacer ce composant dans le schematic, le composant en question ne pouvait pas s'y ajouter au board. Ainsi nous avons pris beaucoup de temps à refaire le PCB et à prendre en compte la largeur des pistes afin d'avoir le PCB optimal. Malheureusement nous n'avons pas pu aboutir à un PCB correct dû à une mauvaise gestion du temps. Il est en effet important de ne pas négliger le temps de la conception du schematic et du PCB, puisque cette conception est la plus grande partie du travail. Mais avant cela il faut bien choisir ses composants pour ne pas retourner à un travail déjà fait au préalable.

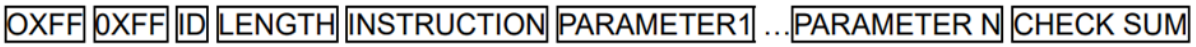
Notre plus grand défaut lors de ce projet a été de ne pas avoir fait plus attention au choix des composants qui nous a été fatal lors de la conception du PCB. Puisque le temps de réalisation du board a été considérable, nous avons dû retourner plusieurs fois sur le travail des recherches de composants. Temps qu'il fallait consacrer prioritairement avant toute chose tel que le schematic.

Pour cette partie on se référera à l'annexe partie 2

Partie 3 : Pilotage des moteurs

III.1 Commande des moteurs

Les servomoteurs Dynamixel AX-12 sont commandés via le protocole UART half duplex. Ce protocole utilise un seul fil pour les données : Tx qui permet d'envoyer des données au moteur mais aussi d'en recevoir (contrairement à l'UART ou deux fils Rx et Tx sont utilisés pour envoyer et recevoir des données). Pour commander le moteur on lui envoi donc des trames constituées de la manière suivante :



Chaque trame commence par deux headers à 0xFF suivi de l'ID du moteur, de la longueur de la trame, de l'instruction à exécuter ainsi que des paramètres utiles à cette instruction et se termine par un "check sum".

Chaque moteur possède un ID défini compris entre 0 et 253. Le moteur recevra la trame uniquement si son ID correspond à celui fourni dans la trame. Heureusement si on ne connaît pas l'ID d'un moteur il est possible d'envoyer une trame avec l'ID 254 (0xFE) qui est l'ID de broadcast et permet de communiquer avec tous les moteurs.

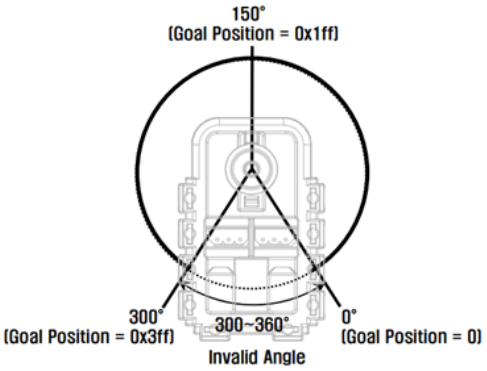
Le "checksum" est un paramètre servant à vérifier l'intégrité de la trame, il est calculé selon la formule suivante :

Check Sum = ~(ID+Length+Instruction+Parametre1+…+ParametreN)&0xFF

Si la trame reçue ne correspond pas au checksum l'instruction n'est pas exécutée et le moteur renvoie une erreur.

Les moteurs présentent beaucoup de paramètres sur lesquels agir ainsi que des données accessibles. Mais dans notre cas on souhaite seulement commander la position et la vitesse du moteur. De plus, nous n'avons pas réussi à faire fonctionner le UART en half duplex sur le STM32, nous utilisons donc de l'UART full duplex en se servant seulement du Tx pour envoyer des trames au moteur sans jamais lire ce que celui-ci renvoie.

Pour commander la position du moteur on vient écrire dans sa mémoire grâce à l'instruction **Write (0x03)** à l'adresse 0x1E et 0x1F la nouvelle valeur de position comprise entre 0x00 et 0x3FF. On veille à séparer cette valeur en deux dans la trame avec le MSB à l'adresse 0x1E et le LSB à l'adresse 0x1F.



De la même manière on peut changer la vitesse du moteur en écrivant dans les deux adresses suivantes de sa mémoire des valeurs entre 0x00 et 0x3FF avec 0 la vitesse maximale atteignable grâce à l'alimentation et 1 la vitesse la plus faible.

Enfin pour pouvoir communiquer avec les moteurs il faut que le baudrate utilisé par le microprocesseur soit le même que celui utilisé par le moteur. Par défaut, le moteur utilise un baudrate de 1000000 bps mais il peut être changé. Nous avons dû tester différentes valeurs de baudrate jusqu'à trouver la bonne. Dans notre cas, il est de 115200 bps.

Interface de TITAN ONE
(interface permettant le contrôle des Leds et des moteurs)

Afin de configurer les paramètres sur l'interface TITAN ONE nous procédons de la manière suivante :
Patch → Dimmers, puis on configure les dimmers en précisant la quantité et l'adresse d'origine.
Les Dimmers sont créés, il suffit alors de les renommer en revenant à l'interface d'origine (touche échap) et de cliquer sur "set Legend".

On peut alors modifier le paramètre que l'on veut en cliquant sur ce dernier et en faisant varier la molette en bas à droite

Avant de réaliser le montage complet, il est conseillé de comprendre chaque partie du système et de les faire fonctionner indépendamment. On peut par exemple réaliser différentes missions :

- Envoyer une trame choisie et l'observer sur l'oscilloscope grâce au décodage UART
- Envoyer une trame choisie et l'observer dans le terminal putty grâce au port UART
- Envoyer une trame d'instruction au moteur pour vérifier la compréhension du mécanisme de communication micro contrôleur-moteur. (Allumage de la LED moteur, rotation du moteur etc..)

- Configurer l'interface de commande sur TITAN ONE
- Initialiser les paramètres configurés et observer la sortie grâce à une paire différentielle DMX sur l'oscilloscope
- Effectuer un code permettant de lire la trame envoyée par le logiciel TITAN one et l'afficher dans le terminal putty

Il ne reste alors plus qu'à tout mettre en commun et de piloter les moteurs grâce à l'ordinateur

Pour cette partie on se référera à l'annexe partie 3

Partie 4 : Pilotage des LEDs

IV.1 Choix des LEDs et des drivers

La seule contrainte émise par le cahier des charges de départ était de faire un projecteur LED RGBW de l'ordre de quelques dizaines de Watt.

Pour des questions d'esthétisme, nous avons choisi de mettre les LED en hexagone (1 au centre et 6 autour). Ce qui était raisonnable en terme de nombre de LED car assez pour appeler ça un projecteur et assez peu pour ne pas exploser le budget et notre limite de puissance.

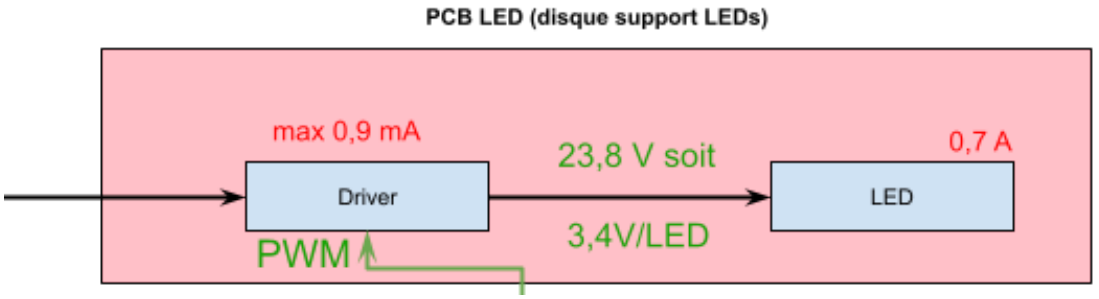
Ainsi, on a choisi la LED LERT WS2WM-JBLA-1+LBMA-P+4V5A-P+MBNA-CQ, qui a une tension seuil entre 2,1 et 3,4V selon la couleur et qui supporte 700mA. On a donc une LED de l'ordre de la dizaine de W, qui n'était pas trop chère par rapport au reste du marché et qui était plutôt puissante.

Ce choix a alors défini notre choix de driver, puisqu'on devait choisir un driver capable de fournir 7 fois la tension maximale nécessaire, soit 23,8V tout en fournissant du 700mA. Et qui reçoit une tension capable d'être fourni par le bloc d'alimentation (même si, au contraire, c'est le choix de driver qui a le plus influencé la tension devant être fournie par le bloc d'alimentation)

On a alors trouvé le LM3404MA/NOPB qui respectait les conditions précédemment données. Il fournit jusqu'à 1mA, il renvoie du 40V max, et est capable de recevoir entre 6 et 42V.

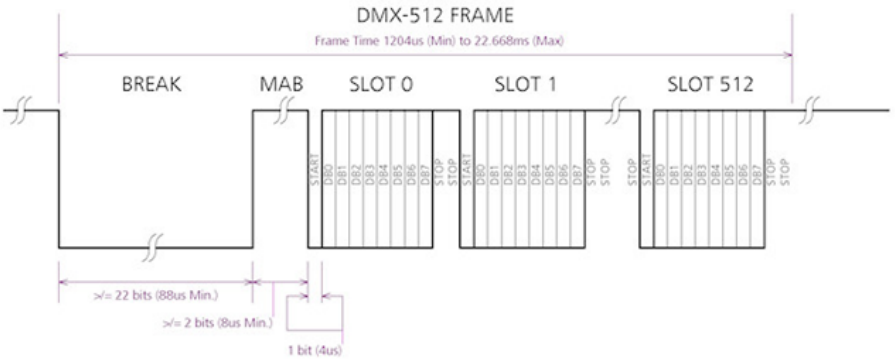
Finalement, on a décidé de lui fournir une tension d'entrée de 24V pour pouvoir être proche de la tension de sortie tout en ayant une valeur référence pour le bloc d'alimentation.

On se retrouve alors avec ce schéma:



IV.3 Commande des LEDs

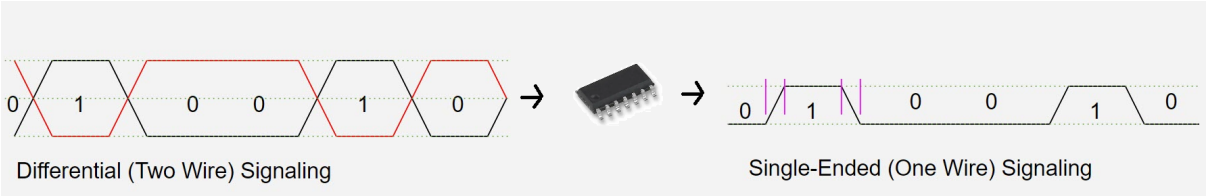
Afin de communiquer avec notre projecteur donc de lui fournir les angles de TILT et de PAN ainsi que les couleurs et intensités des LEDs on utilise le protocole DMX. Ce protocole nous permet d'envoyer des valeurs comprises entre 0 et 255 sur 512 canaux différents à l'aide d'un seul fil. On peut donc contrôler jusqu'à 512 paramètres mais dans notre cas on n'en utilise que 13. TILT1, TILT2, PAN1 et PAN2 pour contrôler le tilt et le pan du projecteur via les moteurs (il y 2 variables par paramètre car le tilt et le pan prennent des valeurs sur deux octets et sur un canal on ne peut envoyer qu'un octet). R, G, B et W pour les intensités et les couleurs des LEDs. Et 3 Checksums pour détecter la fin de la trame désirée et donc vérifier la bonne lecture de la trame. Ces informations sont envoyées dans une trame constituée de la manière suivante :



Chaque trame commence par un break d'au moins 22 bits à 0 puis la map avec au moins 2 bits à 1 et enfin 513 frames de 8 bits commençant par un bit de start à 0 et finissant par deux bits de stop à 1. La frame 0 ne peut pas contenir d'informations.

Afin d'envoyer des trames DMX on utilise un adaptateur USB-DMX Avolites TITAN-ONE. Grâce au logiciel TITAN ONE on peut définir des dimmers permettant de contrôler les valeurs de chaque canal. Pour installer le logiciel il faut veiller à prendre la version 11.4 ou antérieure car elle ne nécessite pas une AvoKey. Enfin après installation il faut générer un token et l'uploader sur le site internet d'Avolites afin de récupérer une clef d'activation du logiciel.

Le DMX utilise une paire différentielle pour transmettre le signal, pour pouvoir l'interpréter grâce au processeur il faut convertir le signal différentiel en signal sur un seul fil. Pour cela, on utilise un Transmitter RS-485.



On peut ensuite lire ce signal en UART sur le STM32 en veillant à bien configurer les bits de start et de stop et en détectant le début de la trame pour savoir à quel canal correspond quelle valeur.

Nous avons eu des problèmes pour extraire les bonnes valeurs de la trame, en effet pour savoir quelle valeur correspond à quel canal, il faut trouver le début de la trame. Pour cela on se sert de l'espace entre les différentes trames et on impose une valeur de timeout légèrement inférieure à cet espace. Pourtant lors des tests on a observé que toutes nos trames commençaient par 00, nous décodons donc notre trame à partir du 3ème octet.

Pour effectuer les tests on choisit les valeurs de différents canaux sur TITAN ONE et dans le stm32 on extrait la trame et on stocke les valeurs des canaux désirés dans une autre trame qu'on envoie en UART à l'ordinateur. Grâce à putty on peut enregistrer ces trames dans un log que l'on ouvre ensuite dans un éditeur de hex pour décoder les trames et les comparer aux valeurs désirées. Grâce à cette technique nous avons pu vérifier que nous pouvions bien isoler les valeurs des différents canaux.

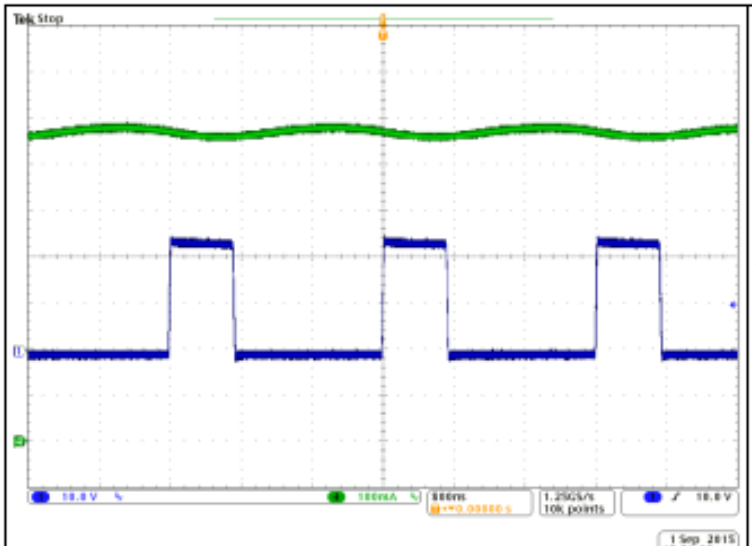
Malheureusement lorsque nous avons rajouté le code permettant de contrôler les moteurs plus rien ne fonctionnait. Cela était dû au fait que l'envoi des trames au moteur ajoutait un délai et décalait la réception de toutes nos trames. La valeur du canal 1 n'étant plus la troisième valeur de la trame reçue. Pour corriger ce problème nous avons ajouté des valeurs fixes à la trame depuis TITAN ONE et écrit un code vérifiant les indices de ces valeurs dans la trame pour traiter uniquement les trame dont ces valeurs sont situées là où on les attends.

Ensuite, les valeurs entre 0 et 255 vont être transformées par le processeur en signaux PWM (un par valeur) qui vont chacun gérer une couleur (R,G,B,W). Ce signal PWM va être transmis au driver LED qui va gérer l'intensité de sortie pour allumer chaque couleur de LED. Ainsi, on peut gérer l'intensité lumineuse et les variations de couleur en envoyant seulement un nombre entre 0 et 255 par couleur (0 correspondant à une intensité de couleur nulle et 255 une intensité de couleur maximale).

On a donc 4 paramètres envoyés en DMX, 4 signaux PWM générés par le processeur, 4 drivers, et 4 intensités pour les 4 couleurs (toujours un par couleur).

Pour ceci, on a dû écrire un code sur CubeIDE permettant au processeur de générer les 4 signaux PWM que l'on souhaitait à partir des paramètres envoyés en DMX.

Par la suite, chaque driver transforme le signal reçu en une variation d'intensité dans les LED. Les signaux ressemblent à ceci:



En bleu, on peut observer le signal renvoyé par le microprocesseur et en vert l'intensité envoyé dans les LED

après passage (et donc lissage) par la bobine.

Les LED ont été mises en série pour qu'elles reçoivent toutes le même signal. En effet, il aurait été possible d'envoyer des intensités différentes dans chaque LED RGBW, mais il aurait alors fallu 4 drivers par LED. Ce qui n'était pas envisageable pour des raisons évidentes de budget, de difficulté et d'ergonomie. Le défaut de ce choix est que les LED doivent toutes afficher simultanément la même chose, elles ne sont pas indépendantes entre elles. C'est donc un choix qu'on assume pour pouvoir mener à bien notre projet.

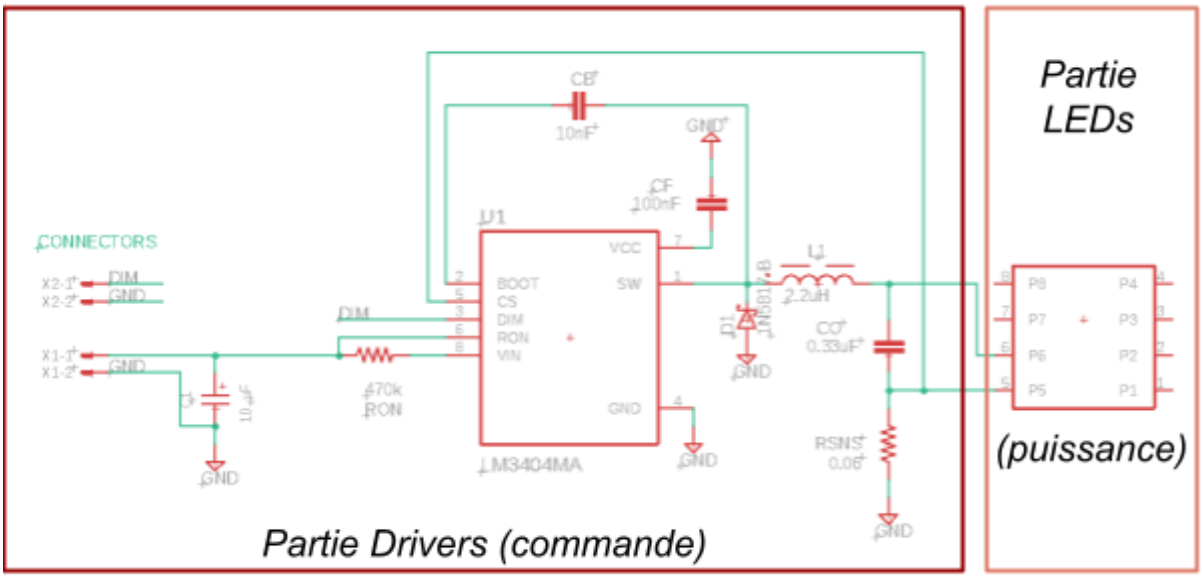
IV.2 Réalisation

- IV.2.1 Réalisation du schéma PCB

On commence par une conception de PCB test pour vérifier le fonctionnement de chaque étape et corriger les potentielles erreurs. On se base donc sur 1 driver qui commande 1 Led et 1 couleur (vert), on lui envoie un PWM via le programme sur CubeIDE qui fait varier l'intensité de la Led de son minimum à son maximum. Pour le montage du driver on reprend un montage détaillé dans la DataSheet du driver, il faut reprendre les calculs qui y sont présent pour adapter les valeurs des composants selon la tension d'entrée et de sortie (dépendant du nombre et type de Led) que l'on possède (dans notre cas V_{in} = 24 V et V_{out_max} = 23,8 V).

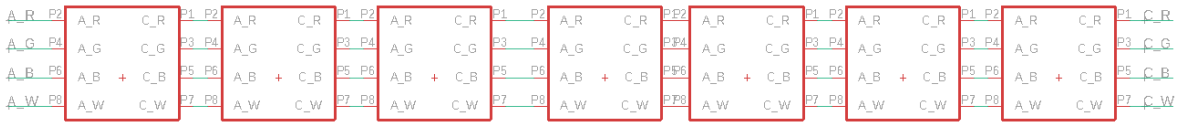
Quelques exemples de formules, elles ne sont pas toutes explicites dans la datasheet ce qui rajoute un travail de déduction.

$$R_{ON} = \frac{V_O}{1.34 \times 10^{-10} \times f_{SW}} \qquad L_{MIN} = \frac{V_{IN} - V_O}{\Delta i_L} \times t_{ON} \qquad R_{SNS} = \frac{0.2 \times L}{I_F \times L + V_O \times t_{SNS} - \frac{V_{IN} - V_O}{2} \times t_{ON}}$$

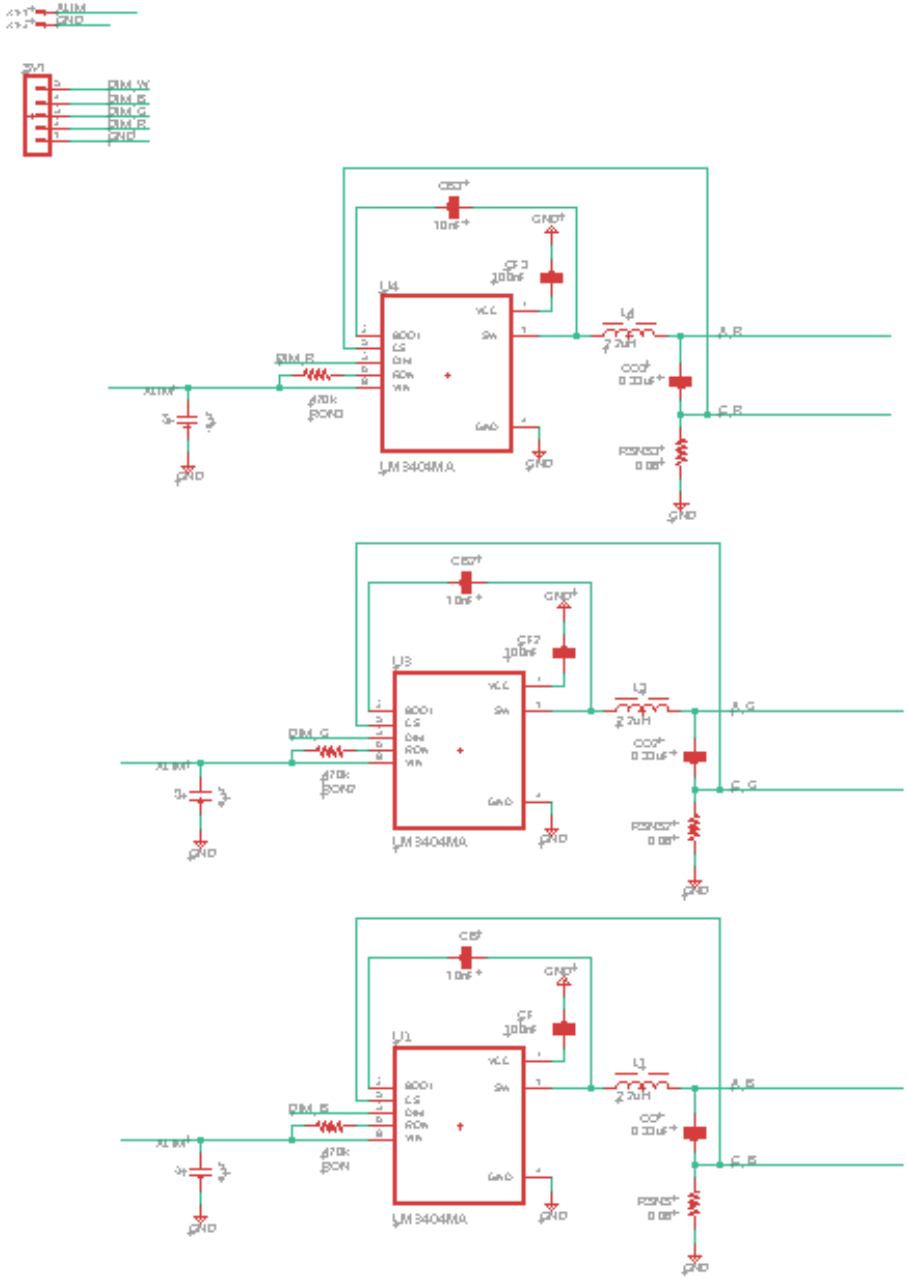


Par la suite on reproduit 4 fois ce montage (car il y a 4 couleurs) en reliant cette fois ci à 7 Leds séries, c'est-à-dire reliant les cathodes et anodes de même couleur sur 2 Leds différents. Cette étape nous a été compliquée pour plusieurs raisons. Nous avons fait quelques erreurs d'étourderie sur le Schématic, le courant de 700 mA circulant dans les Leds impose des pistes assez larges sur le PCB et l'ordre des pins de la Led ne sont pas optimisés pour un montage en série. Ainsi le montage en série avec les 4 montages des drivers peut devenir très vite inextricable. De plus, la création du composant de la Led sur Eagle a ralenti notre progression en générant des erreurs d'inversion cathode/anode et de couleur.

Schematic : les 7 Leds en série, les pins ne sont pas disposés de cette façon réellement, nous les avons modifiés pour simplifier le Schématic.

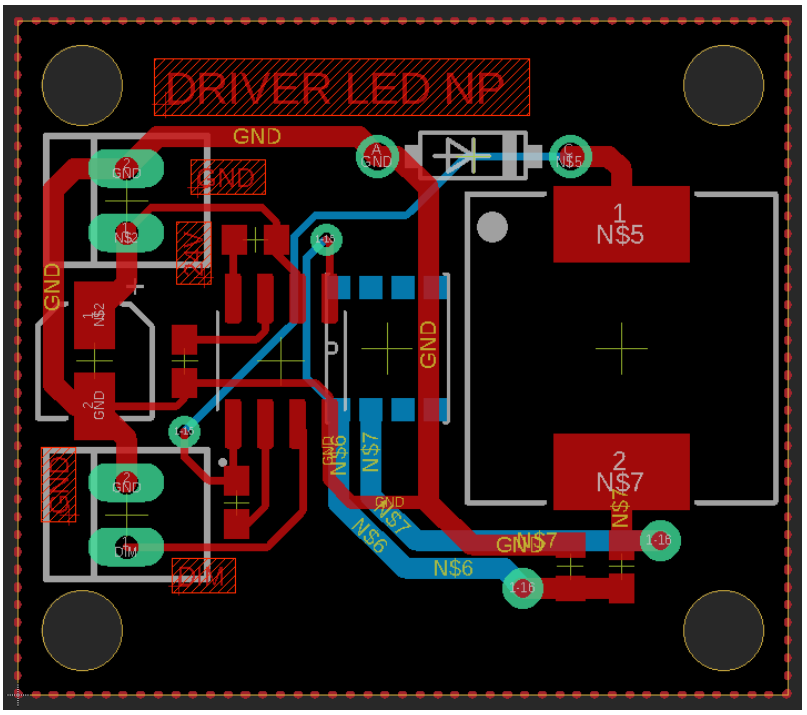


Schematic : Les 4 Drivers, l'utilisation des labels évite d'utiliser des fils et rendent donc le Schématique plus simple a réaliser.



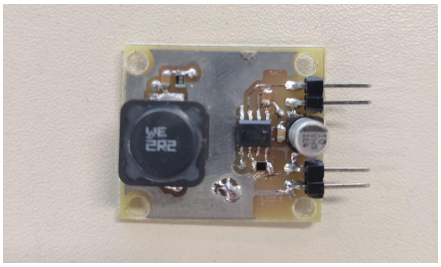
IV.2.2 Réalisation du PCB

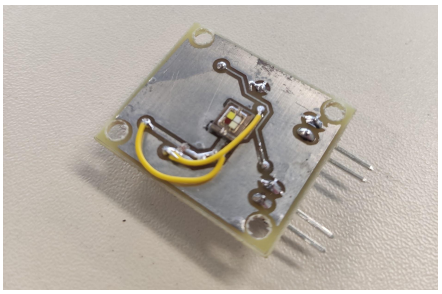
Board : Driver Led Test



Certains composants a valeurs très faibles comme Rsns sont difficilement trouvables.

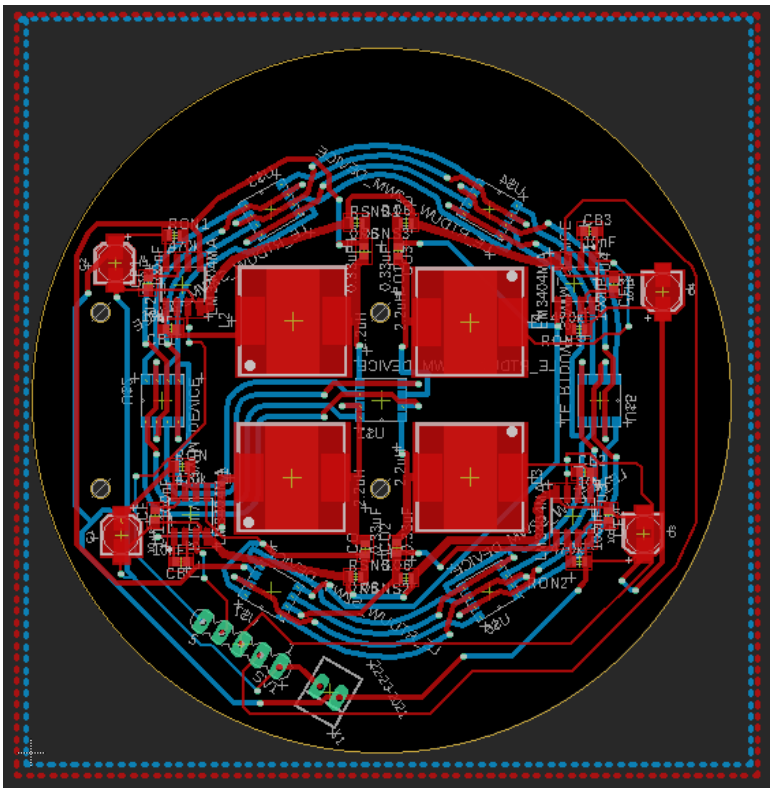
Ce pcb présente plusieurs erreurs, nous sommes connectés au Led bleu au lieu du vert et l'anode et la cathode sont inversées.



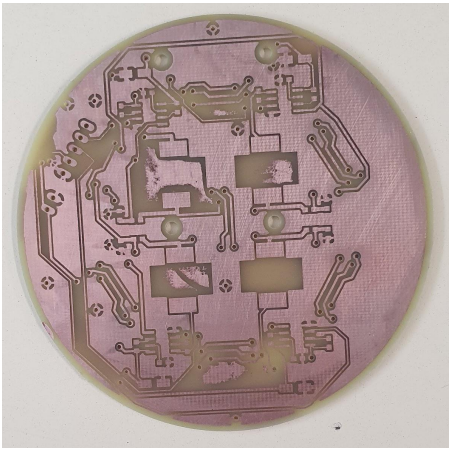


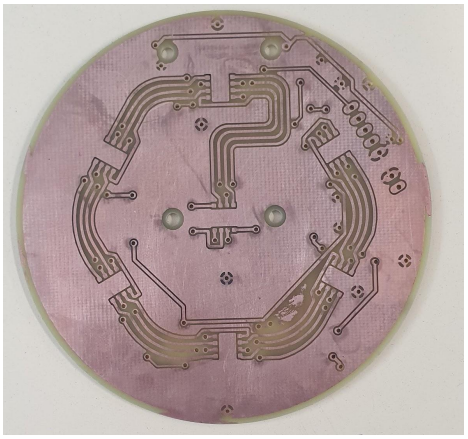
Après quelques corrections des pistes en les recoupant, resoudant et en utilisant des fils. Nous avons obtenu un Led vert fonctionnel sur lequel nous effectuons nos tests de code. Nous faisons par exemple augmenter graduellement l'intensité de la Led en boucle.

Board complète routé :



PCB final : il y a quelques restes de cuivre au mauvais endroit et un petit morceau n'est pas présent mais facile corrigeable. De plus, nous manquons de temps pour souder les composants, il y a un nombre important de via mais ce n'est pas infaisable. Il aurait été judicieux de ne pas superposer les composants sur le top avec les pistes du bottom pour réduire le nombre de via.





Pour cette partie on se référera à l'annexe partie 4

Partie 5 : Création de la carte principale

V.1 Choix du microprocesseur

Le microprocesseur choisi est un STM32L412KBT6 (STM de type M4). Notre choix s'est orienté vers ce microprocesseur car nous avions besoin de :

- 2 port uart (un en mode "receiver" → RX pour le DMX et un autre en "transmitter" →TX moteurs)
- 1 port I2C (Accéléromètre)
- 4 PWM (Driver LED)

Or le STM32 M4 possède les caractéristiques suivantes : 3 ports UART, 2 ports I2C, 1 SPI et 7 TIM (sachant qu'on ne se servira que d'un TIM pour les 4 PWM). Par ailleurs, le M4 possède 28kB de flash et 40kB de RAM, ce qui est suffisant pour ce que l'on souhaite réaliser.

V.2 Communication avec le PC

Le projecteur LED doit être commandé entièrement à l'aide d'une interface graphique, ici nous avons à notre disposition le logiciel Titan One et un adaptateur USB → DMX pour communiquer avec notre maquette. Cela nous à imposé comme contrainte de devoir convertir la trame DMX en une trame uart pour le processeur. Le tout connecté l'interface nous permet d'ordonner différents mouvements aux moteurs.

screen logiciel

V.3 Réalisation du PCB

Comme pour les autres modules intégrant la création d'une PCB, un premier schéma référence et nécessaire pour l'impression des pistes, a été créé sur eagle. Une fois le PCB réalisé vient l'étape de soudure des composants, on commence par souder les vias (connectique) puis l'on soude les composants du plus petit au plus gros. A chaque fin de soudure on vérifie au voltmètre les valeurs de tension (vias, résistances, condensateur etc) pour s'assurer de la soudure correcte. A noter que les trous sont parfois à faire dans la PCB pour les ports.

Problèmes de la PCB :

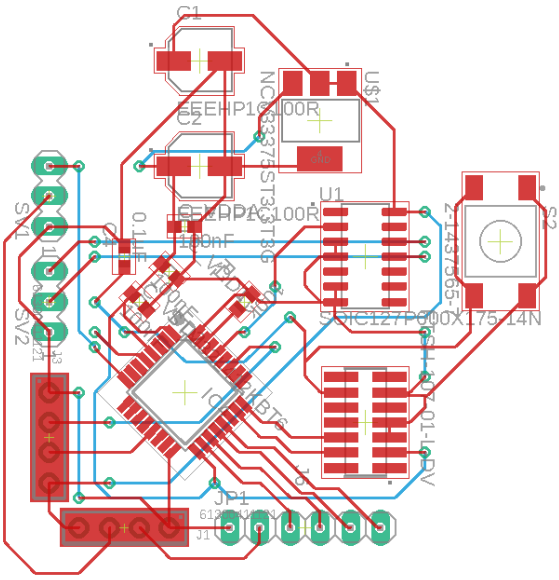
Les condensateurs ne sont pas au plus proche des pins du processeur ce qui peut créer résulter en des signaux qui ne seront pas filtrés.

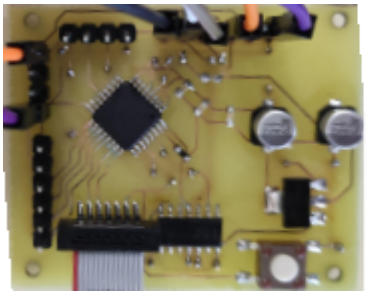
Nous avons également oublié d'intégrer un plan de masse sur la carte ce qui aura sûrement un impact sur la commande des différents modules (présence de bruit).

Le connecteur I2C est présent mais ne sera pas utilisé dans notre projet par manque de temps pour utiliser l'accéléromètre.

Nous avons remplacé la résistance de 100kΩ par une résistance de 200kΩ car nous n'avions pas de résistance de 100kΩ en stock.

Par précaution, nous avons également soudé tous les vias.

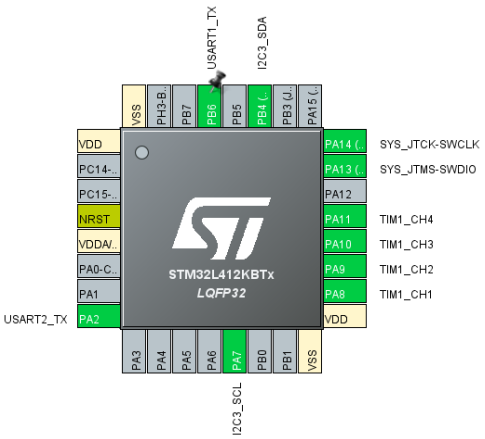




V.4 Définition des ports et code du microprocesseur

Pour la définition des ports du PCB et le code du microprocesseur on est passé par l'interface CubeIDE. Pour utiliser correctement ce logiciel on commence par choisir notre type précis de STM32 pour avoir un bon nombre de ports. Suite à cela on passe à l'étape de configuration des ports en sélectionnant les ports qui vont être actif ou non et en les paramétrant en conséquence ; on pourra alors être amené à rentrer un baud rate, un sens de la data (réception ou émission) ou encore à configurer les PWM sur un port TIM.

Une fois tous les ports paramétrés on peut passer à la partie code en C, dans un premier temps on veillera à ce que tous nos ports soient bien initialisés et notés aux bons paramètres. Après cette vérification on peut commencer à coder les instructions que le microprocesseur sera capable d'ordonner aux moteurs.

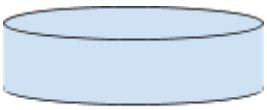


Pour cette partie on se référera à l’annexe partie 5

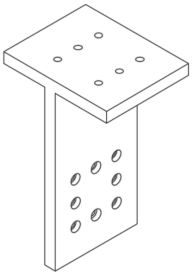
Partie 6 : Assemblage structurel

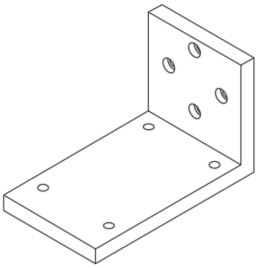
VI.1 Réalisation d'une structure

Pour la partie structure de notre projecteur LED nous avons utilisé une base d'un ancien projet pour ne pas avoir à réimprimer une pièce, et nous avons également conçue deux autres pièces, le T qui permet de maintenir les deux moteurs entre eux et un L qui à pour fonction de tenir le PCD LED.

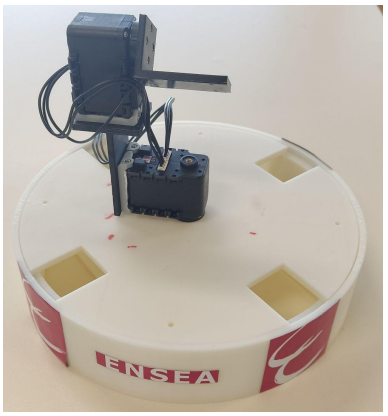


Base





Pour les réaliser, on les a d'abord conçues en modélisation 3D à l'aide du logiciel Fusion 360 puis après enregistrement au format STL (nécessaire à l'impression) on les a imprimées à l'imprimante.
A l'origine nous partions sur une forme de U pour lier les moteurs entre eux et s'assurer de la stabilité, mais par soucis de légèreté, de temps d'impression et d'esthétisme on a préféré opter pour cette solution.



Maquette final avec moteurs montés

Conclusion :

L'objectif d'obtenir une maquette de projecteur LED n'a pas été atteint, cependant le module Driver LED et communication ordinateur au moteur à été réussi. Pour la carte principale, plusieurs problèmes ont nui à son fonctionnement (plan de masse, soudure, proximité des condensateurs au processeur). Quant à la partie alimentation, la disponibilité des composants et la conception eagle ont fait défaut.

La réalisation de la maquette d'un projecteur LED a été très formateur tant d'un point de vue organisation qu'apprentissage. Grâce à ce projet nous avons pu faire le lien avec le cours et comment l'appliquer à la réalité.

Point à améliorer :

- Suivi du projet (mise à jour régulière du GitHub pour le versionning)
- Réalisation des cartes tardif délai, normalement délai de 2 semaines
- Se renseigner plus sur les norme de création des PCB quant il s'agit de 230V
- Rigueur

To be continued…