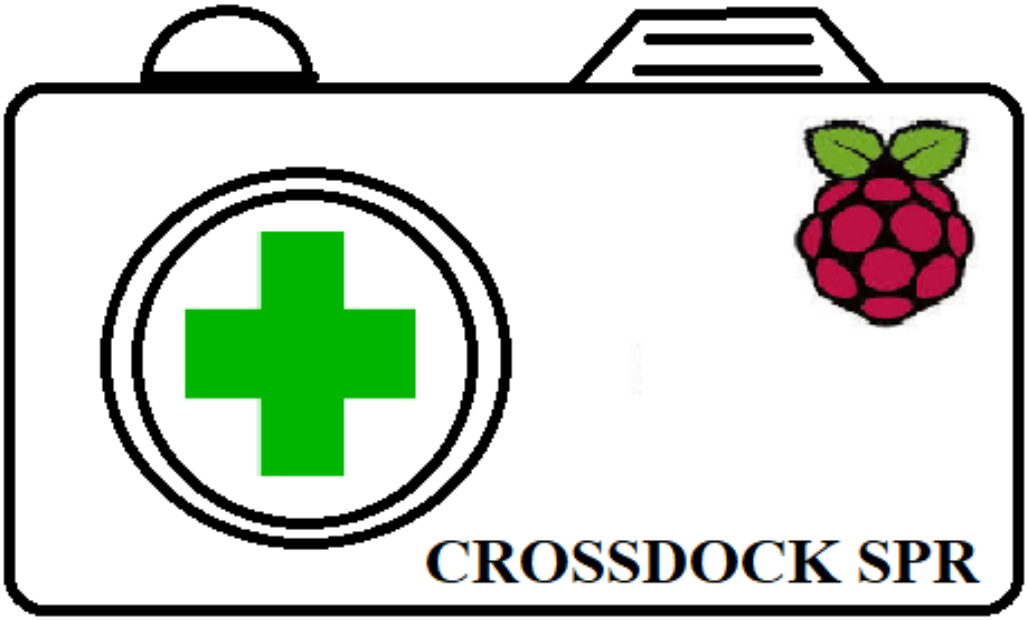
PROJET CROSSDOCK SPR



Date du début du projet : 06 / 01 / 2020

Candidats :
IR : Melki Lucas / Vives Vinny / Arnoux Adrien
EC : Guénard Gauthier / Pamatto Thomas / Leroy Lucas

Chefs de projet : Mr Hortolland / Mr Defrance

Entreprise concernée par le Projet : CrossDock Cavaillon

Responsable : Mr BIJOU

Sommaire :

Présentation de l'entreprise :……. P.4
- Pourquoi ont-ils fait appel a nous ?……. P.5

Partie Commune :....……. P.6
- Présentation du projet……..……... P.6
- Digramme des cas d'utilisation…….……. P.7
- Architecture Matérielle et Logicielle (réduite)…….. P.7
- Répartition des tâches par étudiant…… P.8
- Réunion du 27/01……..…….. P.9
- Réunion du 11/03…… P.10

Partie IR1 : Studio Photo (shield multi cam + éclairage + IHM Opérateur) :… P.11

Partie IR2 : Studio Photo (plateau tournant + IHM Opérateur) :…… P.13
- Planning prévisionnel + Planning des tâches réalisés..…… P.14
- Présentation du moteur pas à pas utilisé....…….. P.15
- Schéma de câblage..……. P.17
- BUS I2C…… P.19
- Programme utilisé..…….. P.20

Partie IR3 : Application de gestion & consultation des clichés :…… P.22
- Diagrammes de séquences…… P.23
- Diagramme de bloc…….. P.25
- Planning prévisionnel + Planning des tâches réalisés…… P.26
- Présentation Docker……. P.27
- Partie pratique Docker…….. P.28
- Base de données + Conclusion……. P.30

Partie EC1 : Rpi Plateau tournant :……. P.31
- Planning prévisionnel…… P.32
- Avancement de la solution étape par étape + Communication BUS I2C Raspberry Pi et Arduino…… P.33
- Programme RPI…… P.35
- Programme Arduino + conclusion partie……. P.36
- Création du schéma électrique……. P.37
- Explication du schéma électrique…….. P.38
- Création du circuit imprimé (PCB)…… P.40

Partie EC2 : Gestion RPI plateau tournant :……. P.41
- Avancement du projet :…….. P.42
- Level Shifter…….. P.43
- Schéma de câblage……. P.44
- Calcul de dissipation thermique……. P.45
- Le moteur pas à pas…… P.46
- Trames I2C relevées……... P.47
- Programmation du moteur pas à pas…….. P.48
- Le routage…….. P.49
- Liste du matériel……. P.49
- Listes des tâches…… P.50

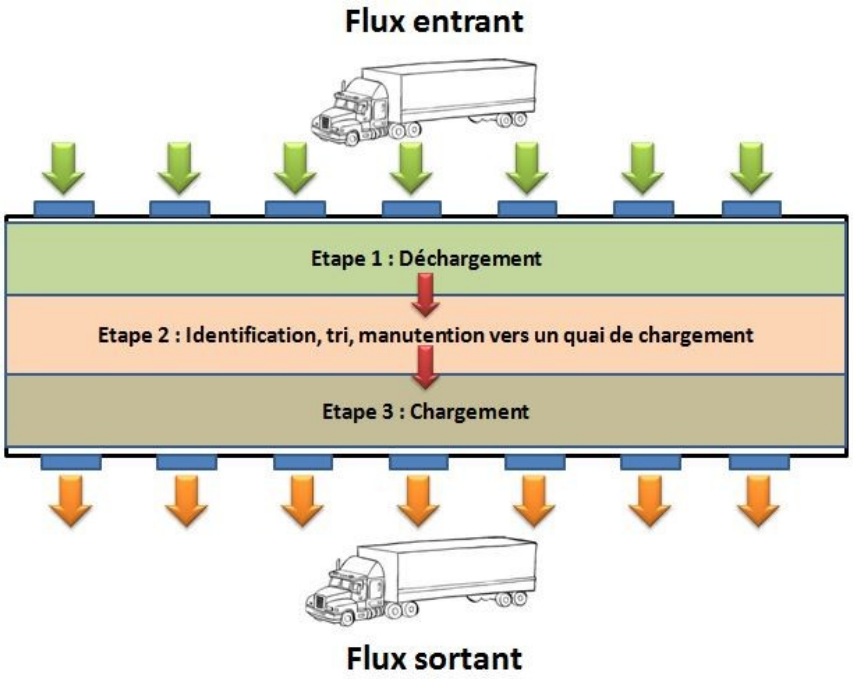
Partie EC3 : Gestion de l'éclairage et des prises de vues :…… P.51
- Choix de la caméra utilisée……. P.52
- Choix de l'éclairage……. P.54
- Partie DMX RDM panneau LED…… P.57
- Choix du capteur de luminosité…….. P.60
- Test TSL2561…….. P.60
- Test TMG39931…….. P.63
- Partie Kicad :…….. P.65
- Schéma de câblage…….. P.65
- Solutions routage..…….. P.66
- Planning prévisionnel + Listes des tâches…….. P.71
- But pour la prochaine revue…… P.73

Présentation de l’entreprise CrossDock

Crossdock est une entreprise située a Cavaillon dans le Vaucluse , pratiquant le cross-docking .

Le cross-docking c’est quoi ?

Le cross-docking est un mode d’organisation des flux logistiques qui permet de faire croiser flux d’approvisionnement venant de fournisseurs avec des flux de livraison terminale.



Concrètement il s'agit de faire passer des marchandises des quais d'arrivée aux quais de départs, sans passer par le stock.

Le cross-docking est une opération qui permet de consolider les colis par commande à partir d'une plate-forme de tri. Cette technique permet d'effectuer une préparation de commande sans pour autant s'appuyer sur un entrepôt.

Il n'est donc pas nécessaire de stocker les marchandises pour effectuer le picking des articles commandés (un colis arrivant sur une plate-forme de cross-docking y reste moins de 24 heures).

Pourquoi ont-ils fait appel a nous ?

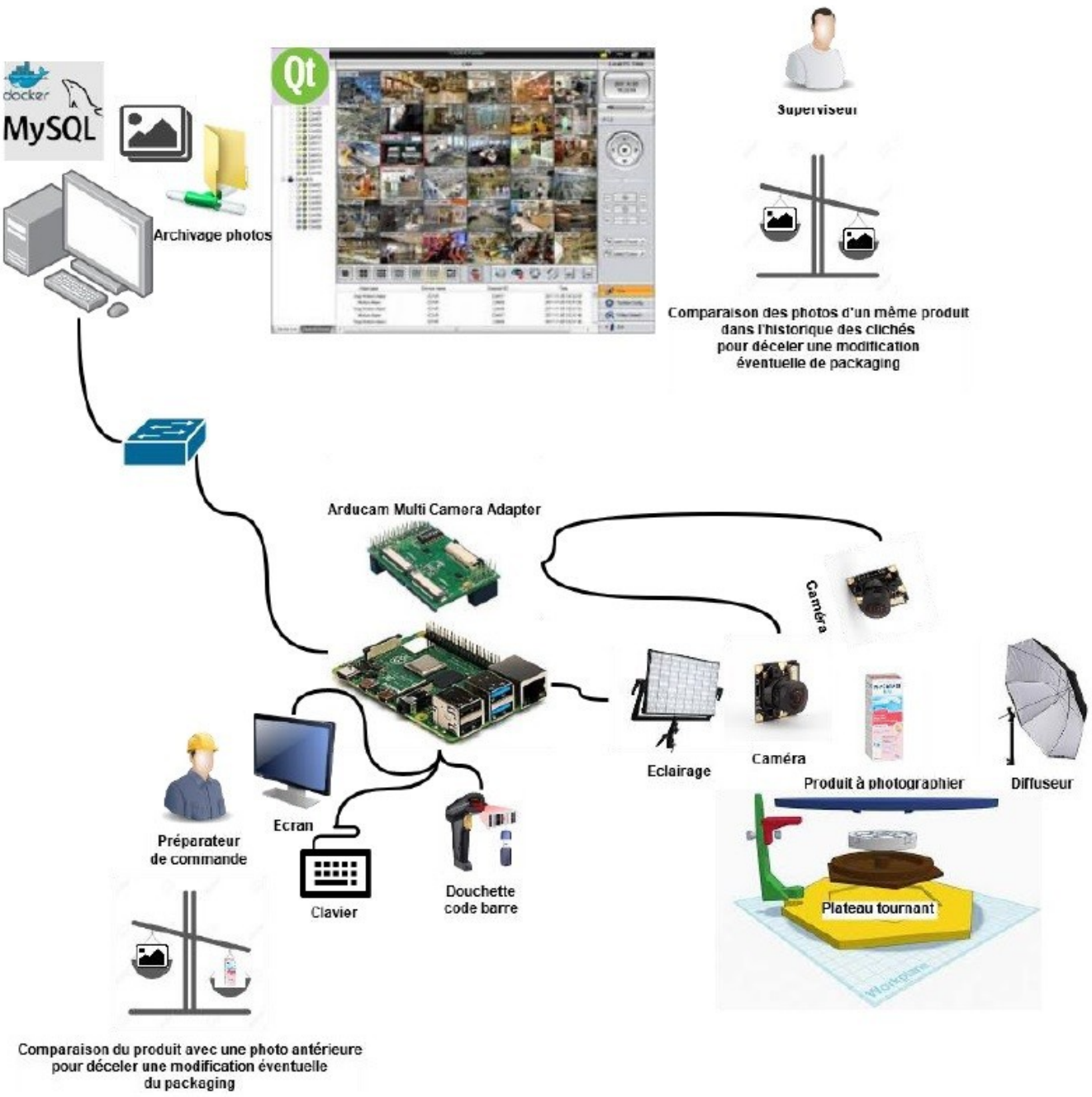
L'entreprise CrossDock aimerait évoluer encore et encore , et nous sollicite pour un projet qui pourrait permettre d'offrir un service supplémentaire a ses clients.

Pour se faire M.Bijou souhaiterait mettre en place un système capable de prendre des photos d'un objet, afin de pouvoir prendre connaissance de la contenance du paquet , ainsi que de pouvoir identifier les informations concernant le produit.

Le but étant de pouvoir comparer les paquets et de pouvoir prévenir le client en cas d'éventuelles modifications du paquet (produits principalement issus de l'industrie cosmétique et pharmaceutique)

Pour répondre à ce cahier des charges nous avons imaginé un système de mini studio photo, composé d'un plateau tournant, d'un éclairage DMX et de caméras Pi reliés a une seule et même Raspberry Pi 2 . Une base de donnée Docker sera mise en place afin de pouvoir effectuer des comparaisons avec les captures d'emballages antérieurs si les produits qu'ils contiennent sont similaires.

Présentation du projet :

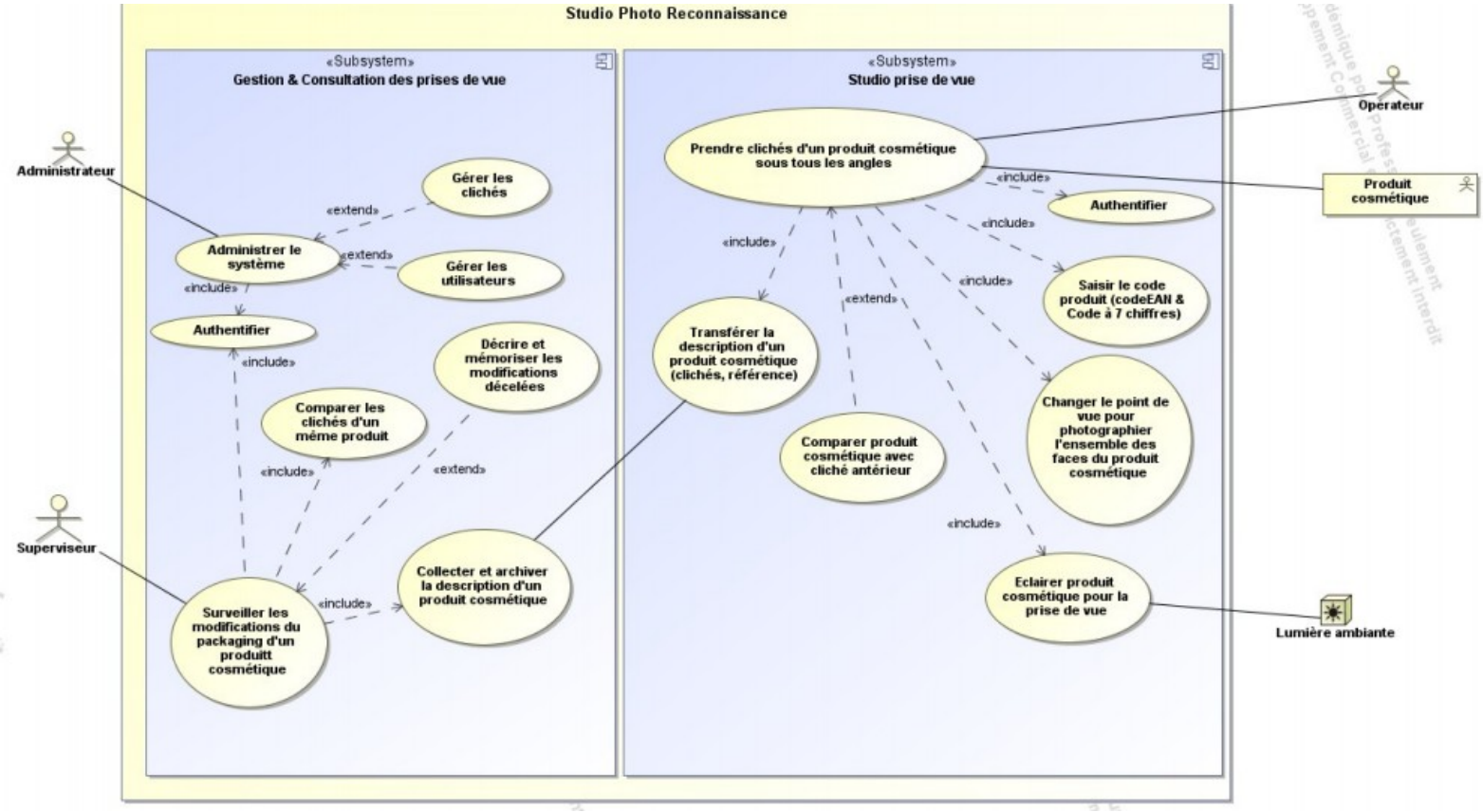


Le projet se présentera a l'utilisateur sous la forme d'une interface graphique conçue sous QT creator, pour faciliter l'utilisation du système.

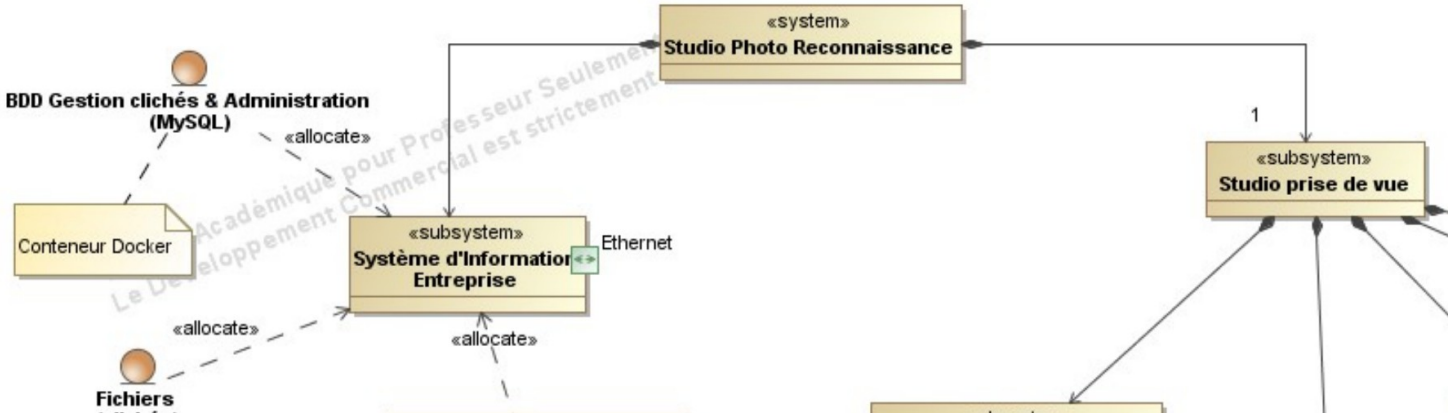
L'idée est de prendre en photo le produit sous tous les angles à l'aide d'un plateau tournant où le paquet sera positionné, face aux caméras.

Une fois les captures effectuées, elles se retrouveront archivées dans la base de données.
Une comparaison pourra être effectuée avec d'anciennes captures afin de détecter tout changement d'emballage.

Diagrammes des cas d'utilisation



Architecture Matérielle et Logicielle (Réduite)



Répartition des tâches par étudiants :

IR1 : Studio Photo (shield multi cam + éclairage + IHM Opérateur)

IR2 : Studio Photo (plateau tournant + IHM Opérateur)

IR3 : Application de gestion & consultation des clichés

EC1 : Rpi Plateau tournant

EC2 : Rpi Plateau tournant

EC3 : Gestion de l'éclairage et des prises de vues

Réunion du 27/01/2020

Lors de cette réunion nous avons pu constater l'avancement de chacun dans le projet, malgré les contraintes qui se sont présentées au fur et a mesure , nous avons quand même réussi a faire fonctionner un minimum des parties du système :

- Prise de connaissance des commandes de la caméra Pi

- Installation de Qt Creator

- Début de création d'Interface Graphique Finale

- Solution DMX envisageable pour l'éclairage

- Plateaux tournants sur la bonne voie (premiers essais prometteurs)

- Capteur de luminosité testé , répondant au cahier des charges

- Prise de connaissance sur le système Docker

En revanche après l'élaboration d'une esquisse rapide du système , nous nous sommes rendu compte d'un contrainte supplémentaire qui est la prise de vue et l'éclairage de la partie supérieure du paquet , comment trouver une structure fiable avec un éclairage qui pèse assez lourd tout en restant compact ?

N'ayant pas trouvé de solution en cet instant, nous envisageons peut-être de supprimer la partie supérieure et de faire intervenir l'utilisateur pour repositionner le paquet de manière a prendre en photo la partie supérieure et inférieure.

Pressé par le temps, nous n'avons pas encore effectué de tests regroupant tout les différents composants en 1 seul et même système , cela sera notre but pour la prochaine Revue ;

Établir un Prototype du Système Fonctionnel*

*si aucune nouvelle contrainte survient

Réunion du 11/03/2020

Lors de cette réunion nous avons fait le point avec toute l'équipe du projet pour savoir comment aborder la revue n°2 et nous avons établi un état d'avancement pour chacun des membres.

Pour la partie Plateau tournant :
Concernant IR2 , EC1 et EC2 le fonctionnement du moteur n'est plus le problème principal mais cette fois ci le programme l'est , un problème d'adaptation du programme rend les tests plus difficiles pour comparer les différents moteurs.

Un protocole devra être établi.

Pour la partie Base de Données :
L'initialisation et le fonctionnement de la plateforme Docker doit être réalisé dans un délai convenable afin de pouvoir tester le système avant la revue n°3

Pour la partie Éclairage et Caméra :
Le fait d'opter pour 1 seule caméra a été adopté par les chefs de projet pour une question d'ergonomie et d'encombrement , cependant le module arducam reste tout de même configuré , prêt a piloter de nouvelles caméra si le projet évolue dans l'avenir.

Une solution de fabrication va intervenir pour l'éclairage de l'objet, elle consistera a piloter une LED de 10W via une Arduino nano , tout cela avec un protocole DMX.
Lors de la phase « d'assemblage » des différentes parties (qui se tiendra après la revue n°2) , nous allons trancher entre le projecteur Stairville ou Cameo pour l'éclairage de fond après des tests

Un nouveau capteur de luminosité va être testé, il s'agit du TMG39931.
Un protocole devra être établi

Le projet avance mais de nouvelles contraintes sont arrivées, le choix de certains composants n'est pas définitifs cependant ils le seront entre la 2eme et la 3eme revue.

Un nouveau programme devra être établi sur le TSL2561 si le TMG39931 ne correspond pas a ce que l'on recherche.

Une recherche de documentation (Température de couleur + intensité lumineuse produite) sur le Stairville Wild Wash semble impossible suite a la non-réponse du fournisseur malgré un entretien par mail.

Les schémas de câblage devront-être terminés pour la 2eme revue pour lancer la fabrication des PCB.

PARTIE IR1 : Arnoux Adrien

Studio Photo (shield multi cam + éclairage + IHM Opérateur)

L'entreprise CrossDock est une entreprise de logistique dont l'activité consiste à préparer puis à expédier des commandes constituées de produits cosmétiques et/ou de parapharmacie. Le projet consiste à enregistrer le packaging complet de tous les produits distribués, en prenant des photos sur chacune de leurs faces permettant de détecter toute modification du conditionnement des produits.

La recherche de solution a été une partie importante durant les premières semaines du projet car le matériel n'était pas toujours imposé par le client comme le système d'éclairage que nous avons du définir, choix du projecteur, quel type de communication (DMX, I2C) et bien sûr, en prenant en compte le budget alloué a notre projet. Pour le système de prise de photo, plusieurs type de caméra étaient à disposition. J'ai donc fait des tests et nous avons fini par ne choisir qu'une seule caméra car les autres présentaient un inconvénient, la photo capturé était déformé.

Une étape de configuration à été obligatoire pour pouvoir travailler sur la raspberry. Puis, en ligne de commande j'y ai installer l'environnement de développement Qt Creator.

La documentation a aussi représenté une partie importante avant de pouvoir se lancer dans la réalisation du système. J'ai donc débuté des recherche pour savoir comment interagir avec l'adaptateur Arducam pouvant contrôler jusqu'à 4 caméras. Le code que j'ai développé permettant de piloter les caméras via cet adaptateur nécessite l'utilisation de la librairie bcm2835 dont la documentation est disponible en ligne. Cette librairie donne accès aux entrées/sorties de la raspberry afin de piloter les caméras grâce à la commande «raspistill».

Dans un premier temps, j'ai réalisé une succession de test depuis un terminal pour me familiariser avec les commandes avant de créer un code plus long et plus complexe.

La commande «raspistill» peut être suivi d'un ou plusieurs paramètres tel que «-o» pour indiquer le lieu ou sera stocker le cliché, «-t » suivi d'un nombre en milliseconde pour indiquer combien de temps à attendre avant de prendre un cliché ou encore «-d» pour lancer une démo. Ainsi la commande «raspistill -t 3000 -o home/pi/Desktop/monImage.jpg» permet de prendre une photo au bout de 3secondes qui sera enregistré sur le bureau au nom de «monImage» au format «.jpg».

Une fois documenté et l'environnement de développement installé, j'ai commencé a coder mon IHM qui, à terme, permettra de :
- Saisir la référence d'un produit
- Vérifier l'éclairage correct du produit à photographier
- Le déclenchement de prises de vue
- L'affichage des clichés obtenu
- L'affichage des clichés dans un dossier partagé Windows
- Référencer les clichés dans la base de données MySQL

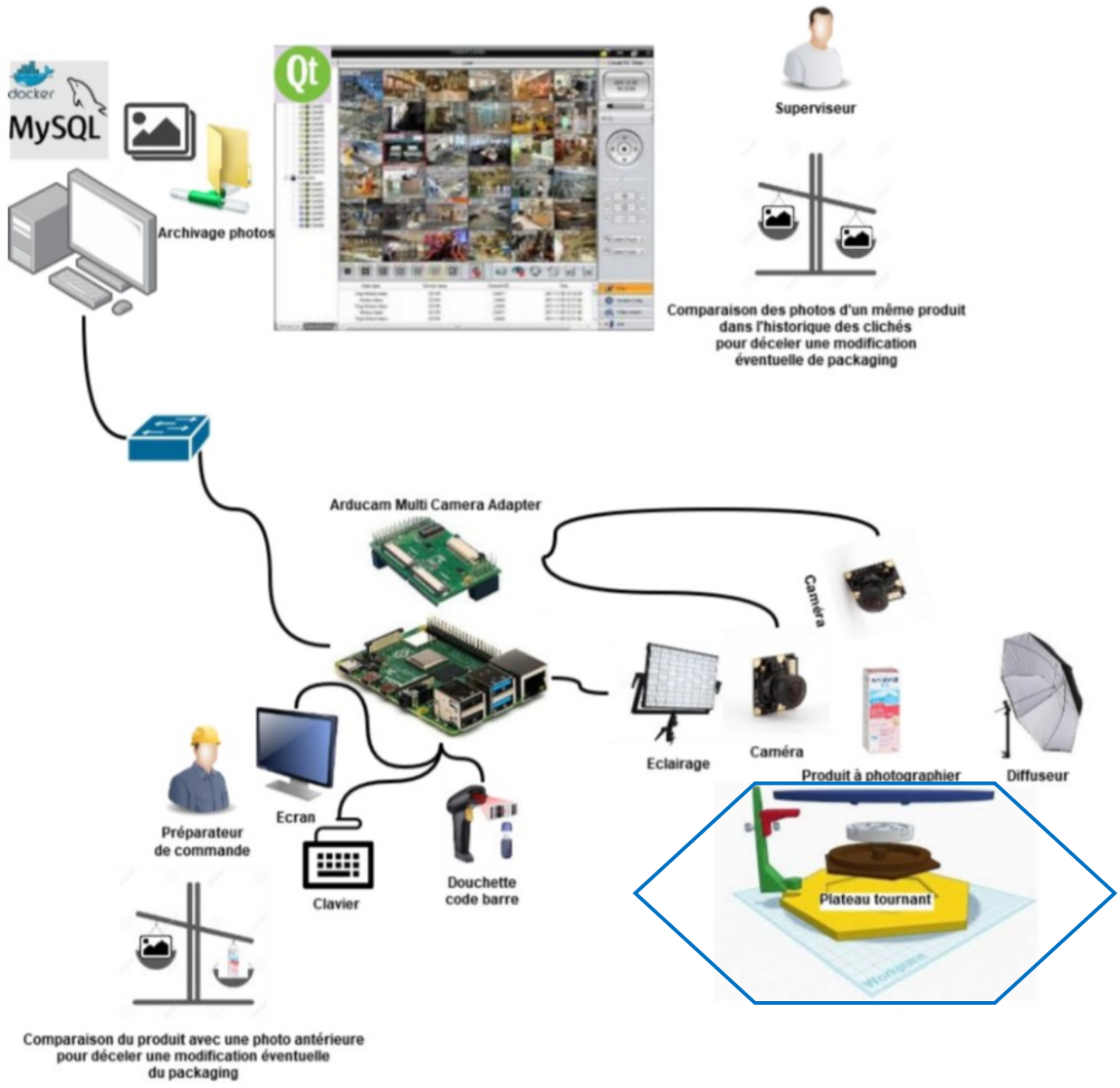
Pour le moment, l'IHM permet seulement de prendre un cliché, le visualise en direct, lui attribut un nom en fonction de la date du jour (format : année-mois-jours) plus un compteur permettant de différencier plusieurs clichés pris le même jour, le sauvegarde sur le bureau car le dossier partagé Windows n'est pas encore créé.

Pour facilité le travail de l'utilisateur j'ai opté pour une interface comportant des onglets plutôt qu'une seul fenêtre regroupant toutes les fonctionnalités. Un premier onglet est dédié à la prise des photos, un autre onglet offrira la possibilité de contrôler la luminosité ambiante en réglant le niveau de luminosité du projecteur, un dernier onglet permettra de visualiser l'entièreté des clichés et une fonctionnalité de reconnaissance sera ajouté au troisième onglet ou utilisera un quatrième onglet à part entière.

Pour conclure, les tâches qu'il me reste à accomplir sont :
- Créer un espace dans l'IHM permettant de saisir la référence d'un produit à la main
- Vérifier l'éclairage correcte du produit à photographier
- Créer un dossier partagé Windows
- Pouvoir dialoguer avec la base de donnée depuis l'IHM

Partie IR2 : Melki Lucas

Studio Photo (plateau tournant + IHM Opérateur)



Je m'occupe de la partie plateau tournant du projet.

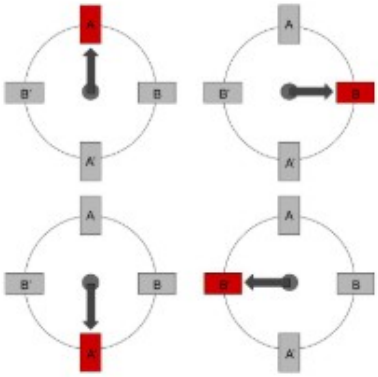
Planning prévisionnel :

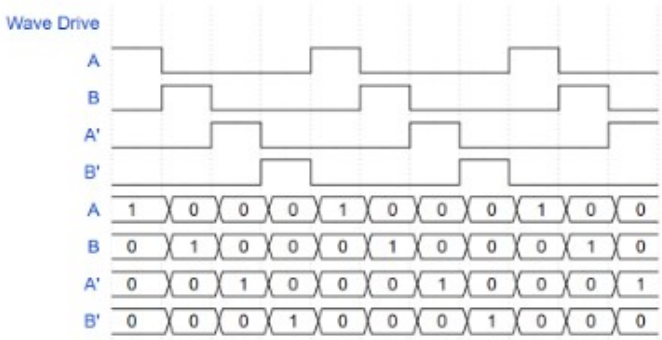
Projet SPR	35,13 days?	Mon 13/01/20	Mon 15/06/20
Revues de projet	27,63 days?	Mon 10/02/20	Mon 15/06/20
Revues de projet 1	0,38 days?	Mon 10/02/20	Mon 10/02/20
Revues de projet 2	0,38 days?	Mon 04/05/20	Mon 04/05/20
Remise de projet	1 day	Fri 29/05/20	Fri 29/05/20
Soutenance finale	0,38 days	Mon 15/06/20	Mon 15/06/20
Réunions Projects	1,75 days?	Mon 13/01/20	Thu 16/01/20
Réalisation des tâches	14,13 days?	Mon 13/01/20	Tue 17/03/20
S'approprier le fonctionnement du moteur pas à pas	2 hrs	Mon 13/01/20	Mon 13/01/20
Cabler un dispositif provisoire simulant le plateau tournant	1,75 days	Mon 13/01/20	Mon 20/01/20
Installer l'OS sur une RaspBerry	2 hrs	Thu 16/01/20	Thu 16/01/20
Coder une classe c++ QT pour la commande du plateau tournant	12,13 days?	Thu 16/01/20	Tue 17/03/20
Intégrer le programme IHM dans le programme c++	3 hrs	Tue 17/03/20	Tue 17/03/20

Planning taches réalisés :

Projet SPR	35,13 days?	Mon 13/01/20	Mon 15/06/20	
Revues de projet	27,63 days?	Mon 10/02/20	Mon 15/06/20	
Revues de projet 1	0,38 days?	Mon 10/02/20	Mon 10/02/20	
Revues de projet 2	0,38 days?	Mon 04/05/20	Mon 04/05/20	3
Remise de projet	1 day	Fri 29/05/20	Fri 29/05/20	4
Soutenance finale	0,38 days	Mon 15/06/20	Mon 15/06/20	5
Réunions Projects	1,75 days?	Mon 13/01/20	Thu 16/01/20	
Réalisation des tâches	20,5 days	Mon 13/01/20	Thu 09/04/20	
S'approprier le fonctionnement du moteur pas à pas	4 hrs	Mon 13/01/20	Mon 13/01/20	
Cabler un dispositif provisoire simulant le plateau tournant	3,63 days	Tue 14/01/20	Mon 27/01/20	9
Installer l'OS sur une RaspBerry	2 hrs	Thu 16/01/20	Thu 16/01/20	
Prendre connaissance du fonctionnement du bus I2C	13 hrs	Thu 16/01/20	Wed 22/01/20	11
Programmation c++ d'une classe plateauTournant sur qt creator (non terminé)	17 days	Thu 23/01/20	Thu 09/04/20	12

Le plateau tournant utilise un moteur pas à pas donc je me suis d'abord renseigné sur le fonctionnement d'un moteur pas à pas bipolaire.

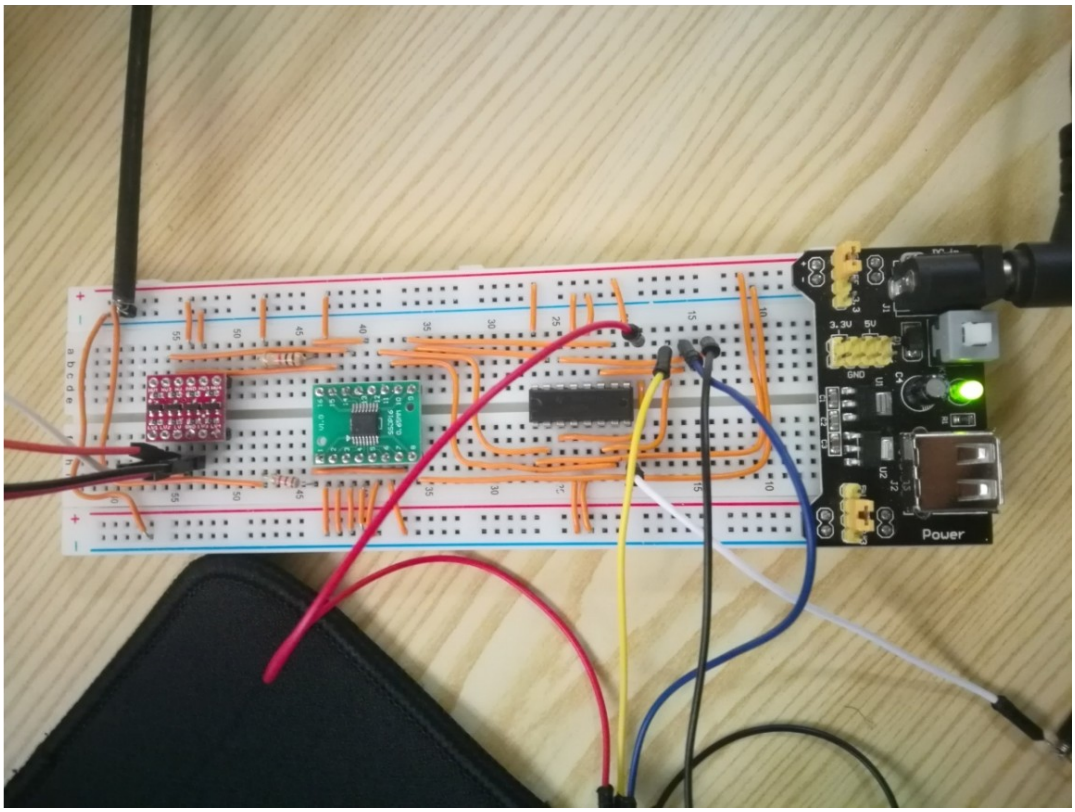




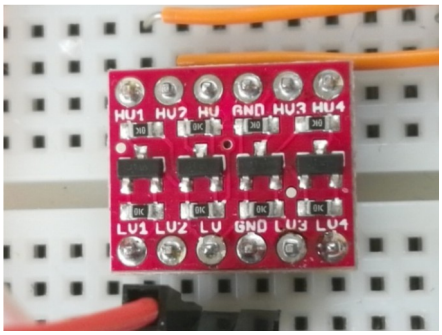
Ce moteur fonctionne en envoyant une impulsion électrique dans une bobine puis l'autre, à nouveau dans la première mais dans l'autre sens c'est-à-dire A <- C au lieu de A -> C puis pareil pour la seconde bobine, dans l'ordre A>B>C>D.

J'ai câblé un dispositif provisoire pour faire mes tests, la communication ce fait d'une Raspberry Pi via le bus I2C.

Le bus I2C est une voie de communication synchrone qui envoie sur un fil des trames de donnée et sur un deuxième fil il y a une horloge.

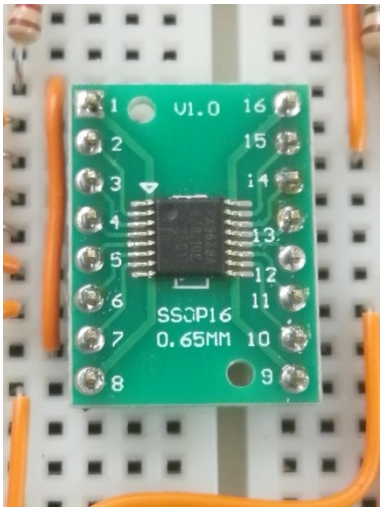


Il y a d’abord le level shifter :



Il permet d’augmenter ou de diminuer la tension du signal d’entrée jusqu'à 4 signaux.
Actuellement la tension du signal passe de 3.3V à 5V.

Il y a le PCA9629A :



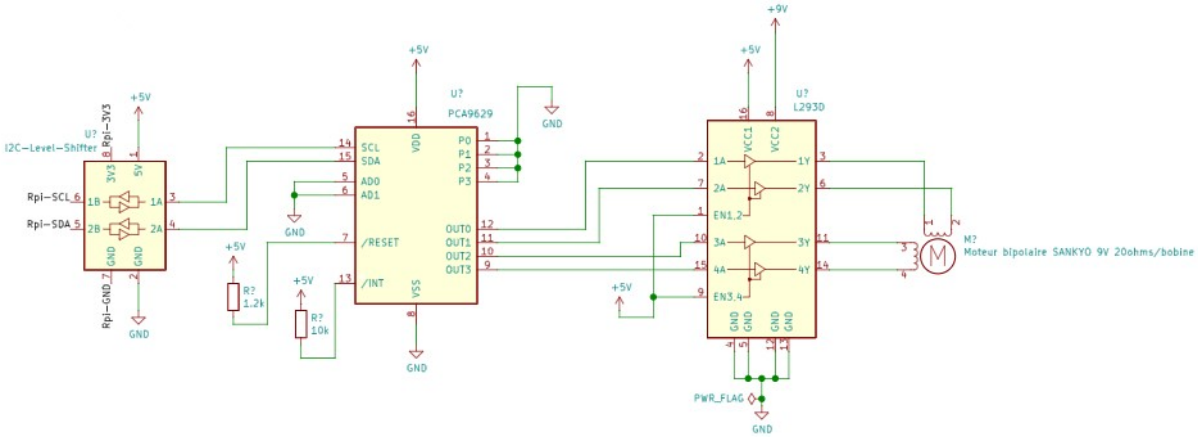
Ce composant permet de contrôler le moteur pas à pas en envoyant via le bus I2C des trames directement dans les registres du composant.
C'est-à-dire qu’il y a par exemple un registre pour une pente d’accélération/décélération, la vitesse du moteur, et le nombre de pas à effectuer.

Et il y a le L293D :

Ce composant permet que le moteur pas à pas puisse tourner dans le sens horaire et dans le sens antihoraire.

Le dispositif temporaire est principalement câbler en 5V.

Voici le schéma du câblage :



J’ai installé l’OS Raspbian sur la raspberryPI, la classe C++ sera fait sur QT sur ce mini ordinateur.

Nous avons utilisé une raspberryPI car nous avons besoin d’un environnement linux ce qui convient bien à ce qui est voulu.

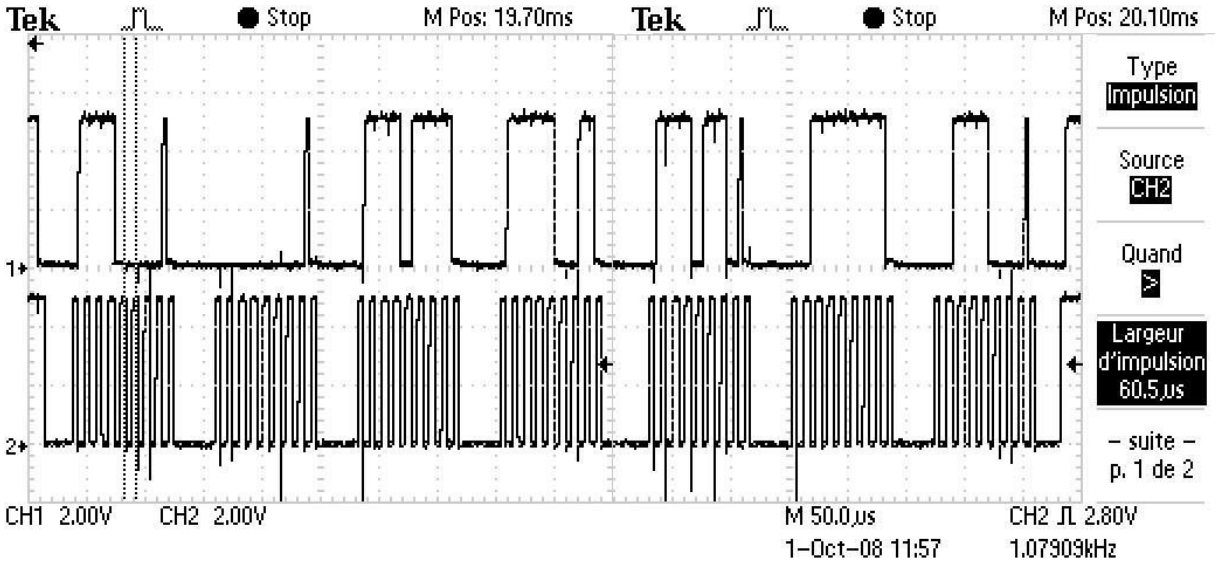
J’ai fait un programme en bash dans la raspberryPI pour envoyé des trames sur le PCA9629A pour écrire des directives dans les registres.
Cela a été réalisé grâce à « i2cset » qui permet en précisant l’adresse de l’esclave qui ici est le PCA9629A d’envoyer les directives.

```
#!/bin/bash
# Configurer le PCA9629A en fonction du nombre de pas désirés pour
# les rotations horaire et anti-horaire, de la vitesse désirée,
# des pentes des rampes d'accélération/décélération désirées... :

# 1/ Configurer la pente d'accélération
i2cset -y 1 0x20 0x0d 0x25

# 2/ Configurer la pente d'accélération
i2cset -y 1 0x20 0x0e 0x25

# 3/ Configurer la vitesse de rotation horaire
i2cset -y 1 0x20 0x96 0x080a w

# 4/ Configurer le nombre de pas désirés pour la rotation horaire
i2cset -y 1 0x20 0x92 0x0330 w

# 5/ Configurer la vitesse de la rotation anti-horaire
i2cset -y 1 0x20 0x98 0x080a w

# 6/  Configurer le nombre de pas désirés pour la rotation anti-horaire
i2cset -y 1 0x20 0x94 0x0330 w

# 7/ Lancer rotation horaire
echo "rotation horaire"
i2cset -y 1 0x20 0x1a 0xc0

# Attendre fin de rotation (1200/266 => t >= 4s)
sleep 12

# 8/ Lancer rotation anti-horaire
echo "rotation anti-horaire"
i2cset -y 1 0x20 0x1a 0xc1

# Attendre fin de rotation (1200/266 => t >= 4s)
sleep 12

# 9/ Stopper le moteur
i2cset -y 1 0x20 0x1a 0xa0
```

J’ai câblé un dispositif provisoire pour faire mes tests, la communication ce fait d’une raspberryPi via le bus I2C.

Le bus I2C est une voie de communication synchrone qui envoie sur un fil des trames de donnée (SDA), et sur un deuxième fil il y a une horloge (SCL) qui définie la vitesse des informations.

Voici un exemple de trame I2C où en 1 le signal, et en 2 l’horloge.



Pour envoyer des données, un périphérique maître envoie des informations aux périphériques esclaves.
Les esclaves sont ceux qui reçoivent les données et il peut y avoir plusieurs maîtres.

Dans la classe plateau Tournant il y a une fonction setSpeed qui permet de régler la vitesse de transmission sur le bus I2C en baud :

```
Void I2cComm ::setSpeed(speedMode_t speed)   {
        Int br;

        Switch(speed) {
              Case I2cComm ::STANDARD :
                    Br = 100000 ;         //100 000 bauds
                    Break ;
              Case I2cComm ::FULL :
                    Br = 400000 ;         //400 000 bauds
                    Break ;
              Case I2cComm ::fast :
                    Br = 1000000 ;        //1 000 000 bauds
                    Break ;
              Default :
                    Br = -1 ;
                    Break ;
        }
        If(br > 0)   {
               Bcm2835_i2c_set_baudrate(br);
        }
}
```

Il y a le choix entre la vitesse standard, full ou fast.
Dans la classe plateauT il y a le contrôle du moteur, c'est-à-dire la pente d'accélération, le nombre d'impulsion par seconde (vitesse) et le nombre de pas qui vont êtres écrit dans les registres du composant PCA9629A:

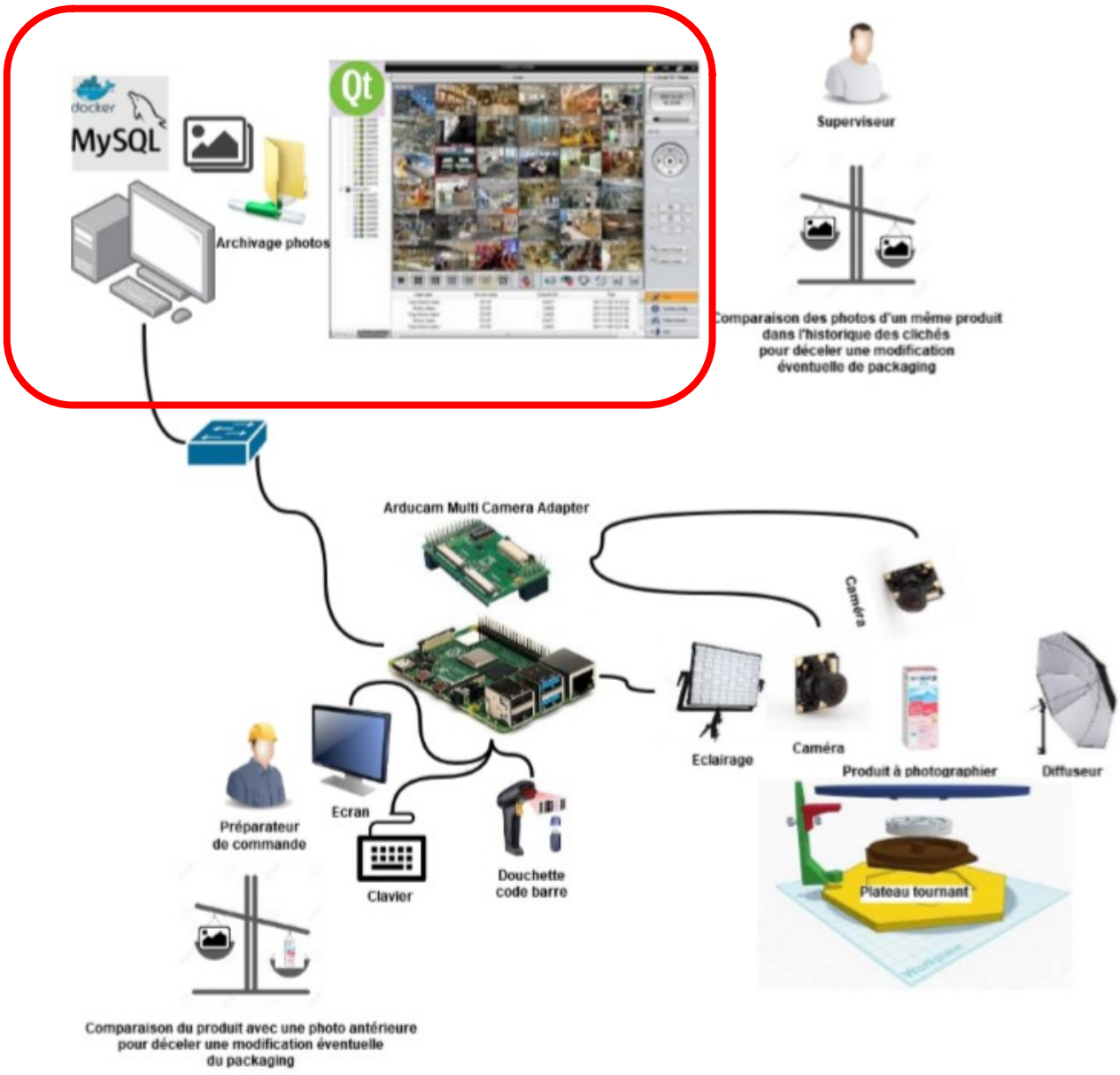
```
   m_motor = new PCA9629A();      //PCA9629A est defini comme m_motor
   m_motor->setRampSlope(PCA9629A::UP, 5);    //pente d'acceleration
   m_motor->setRampSlope(PCA9629A::DOWN, 5);    //pente de décélération
   // On fixe le nb de pas et la vitesse pour la rotation horaire désirée.
   m_motor->setPulsePerSecond(PCA9629A::CW, 800);   //impulsions par seconds (800)
   m_motor->setStepCount(PCA9629A::CW, 400 * 3 - 485);   //nombre de pas en 3 tours - les pentes (715)
   // CW = sens horaire,  CCW = sens antihoraire
   m_motor->setPulsePerSecond(PCA9629A::CCW, 800);
   m_motor->setStepCount(PCA9629A::CCW, 400 * 3 – 485);
```

Dans l’interface home machine il y a les fonctions qui vont faires tournés le moteur:

```
void ihm::on_btnCW_clicked()     //bouton sens horaire cliqué
{
   m_motor->start(PCA9629A::CW);    //le moteur commence a tourner
   m_timer.start();          //lance un timer qui permet de vérifier si le moteur tourne toujours et
   //controle donc la possibilité de pouvoir appuyer sur un bouton ou non.
   ui->btnCW->setEnabled(false);   // le bouton est désactivé
   ui->btnCCW->setEnabled(false);   //idem

}

void ihm::on_timer_elapsed()       //après la fin du timer
{
   if(m_motor->isRunning() == false) {       //vérification de si le moteur tourne ou non.
      m_timer.stop();      //stop la relance du timer
      ui->btnCW->setEnabled(true);      //le bouton est réactiver
      ui->btnCCW->setEnabled(true);    //idem
   }
}

void ihm::on_btnCCW_clicked()       //bouton sens antihoraire cliqué
{
   m_motor->start(PCA9629A::CCW);    //le moteur commence a tourner
   m_timer.start();          //lance le timer
   ui->btnCW->setEnabled(false);   // le bouton est désactivé
   ui->btnCCW->setEnabled(false);   //idem
}
```

Partie IR3 : Vives Vinny

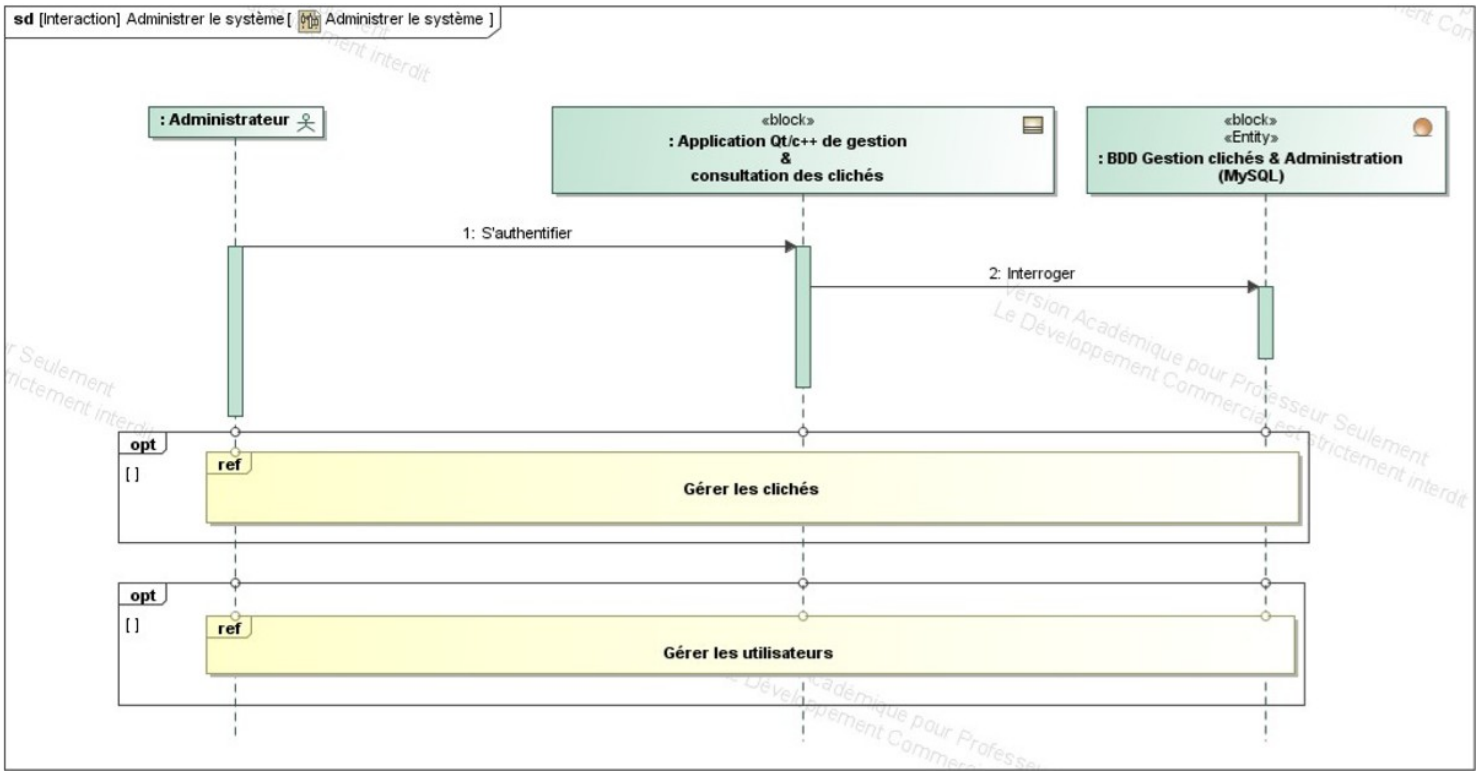
Application de gestion & consultation des clichés



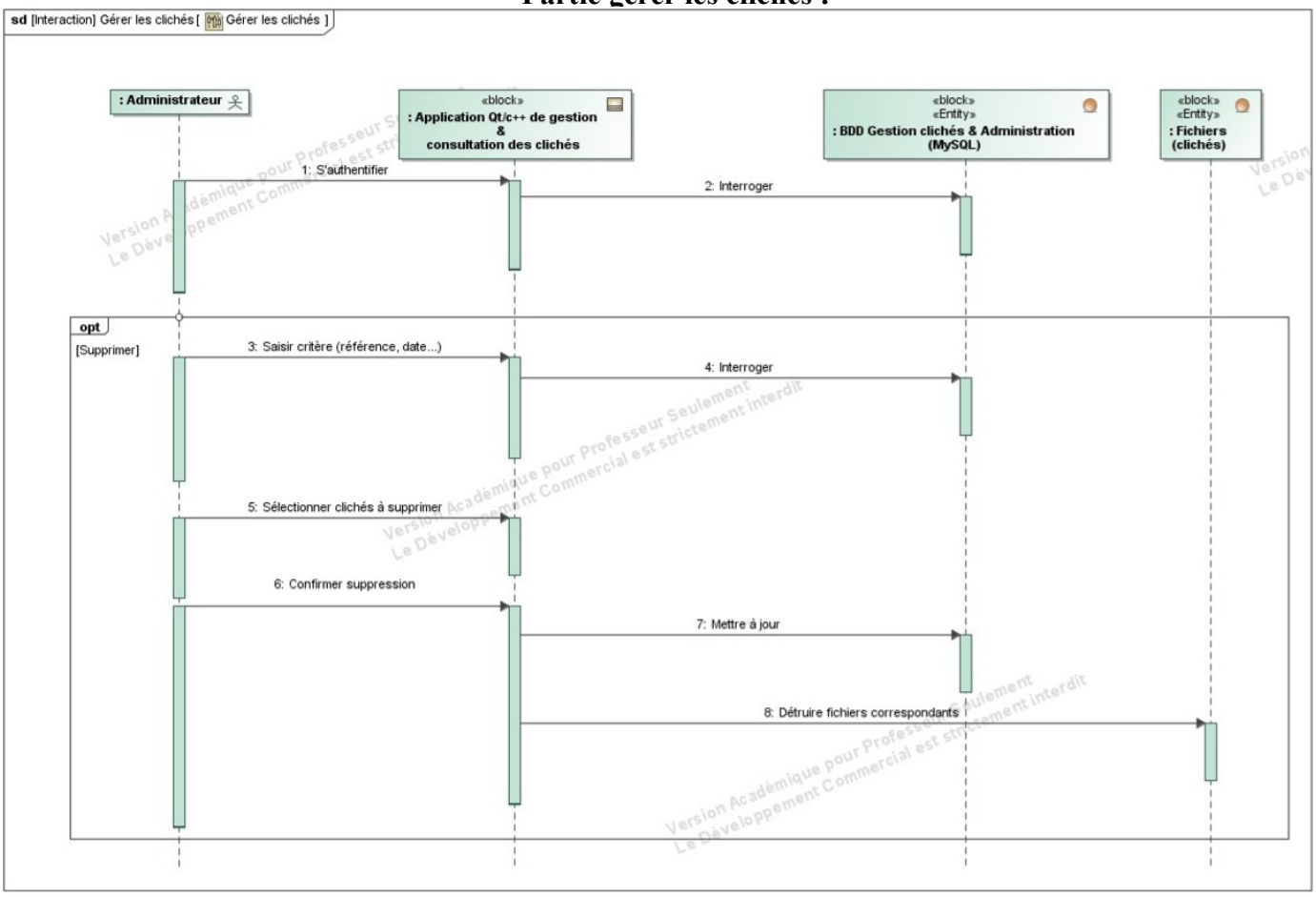
Présentation de ma partie

Mon but dans ce projet est tout d'abord de mettre en place une base de données qui se situera dans un conteneur (grâce à Docker). Ensuite je dois réaliser une IHM QT en C++ sous Windows qui permettra de visualiser toutes les photos afin de déceler des modifications de packaging.

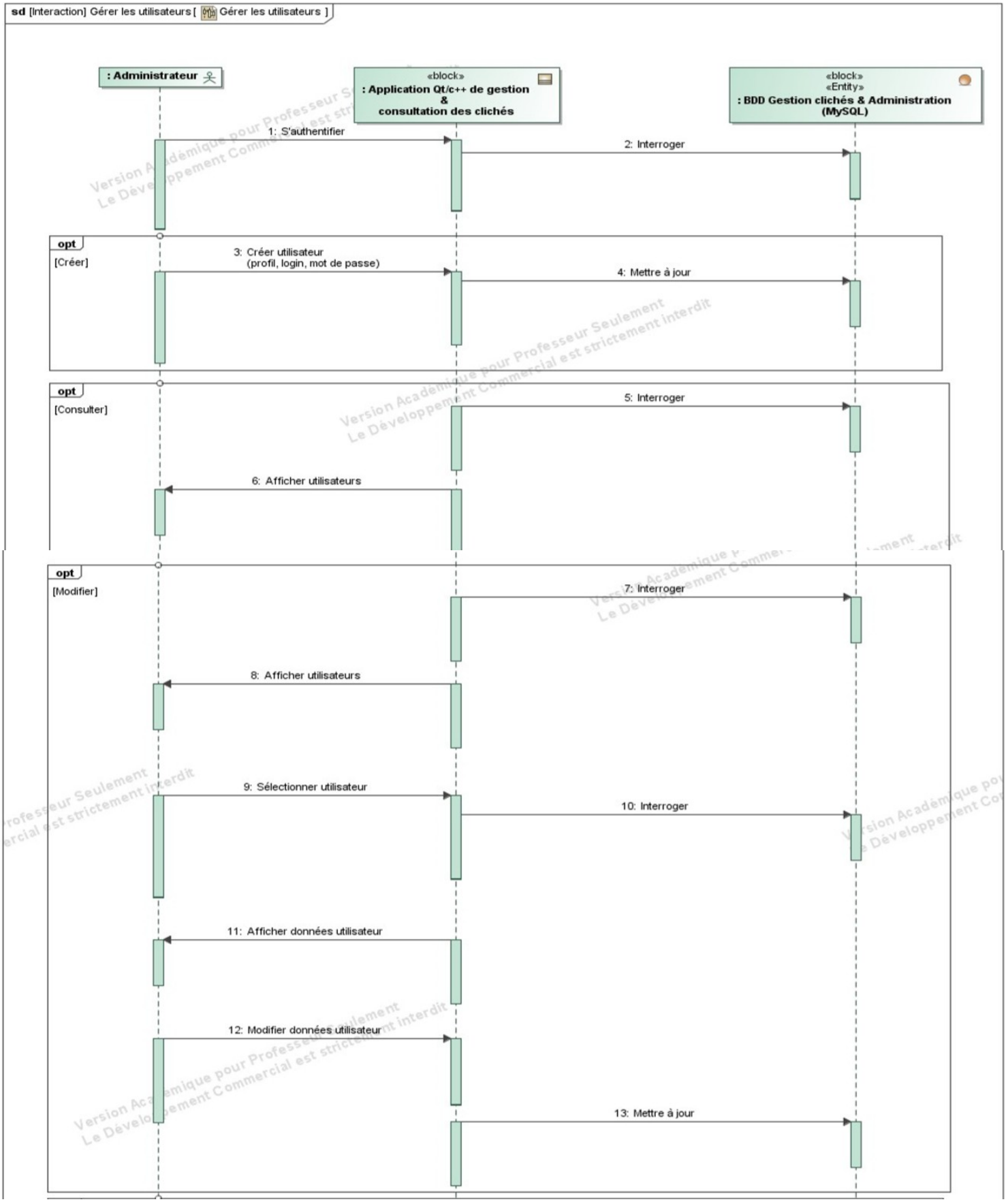
Diagrammes de séquences :



Partie gérer les clichés :



Partie gérer les utilisateurs :



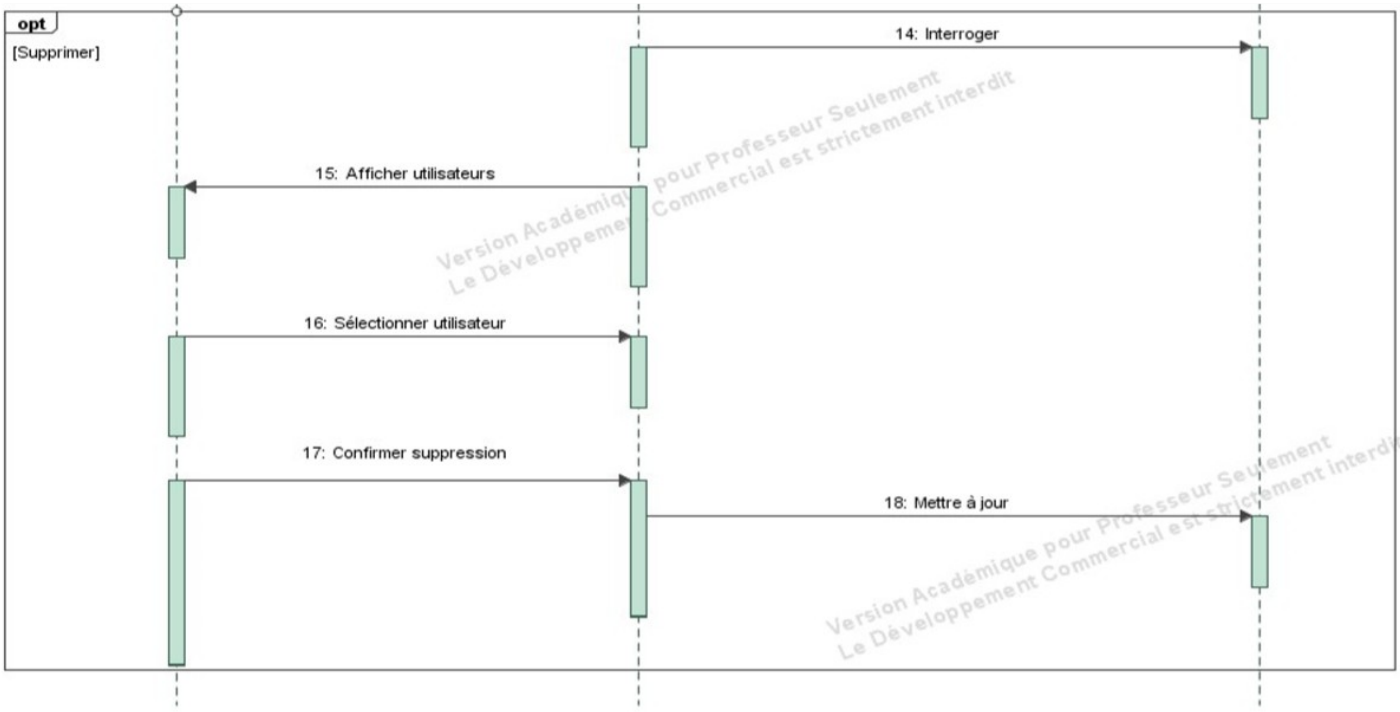
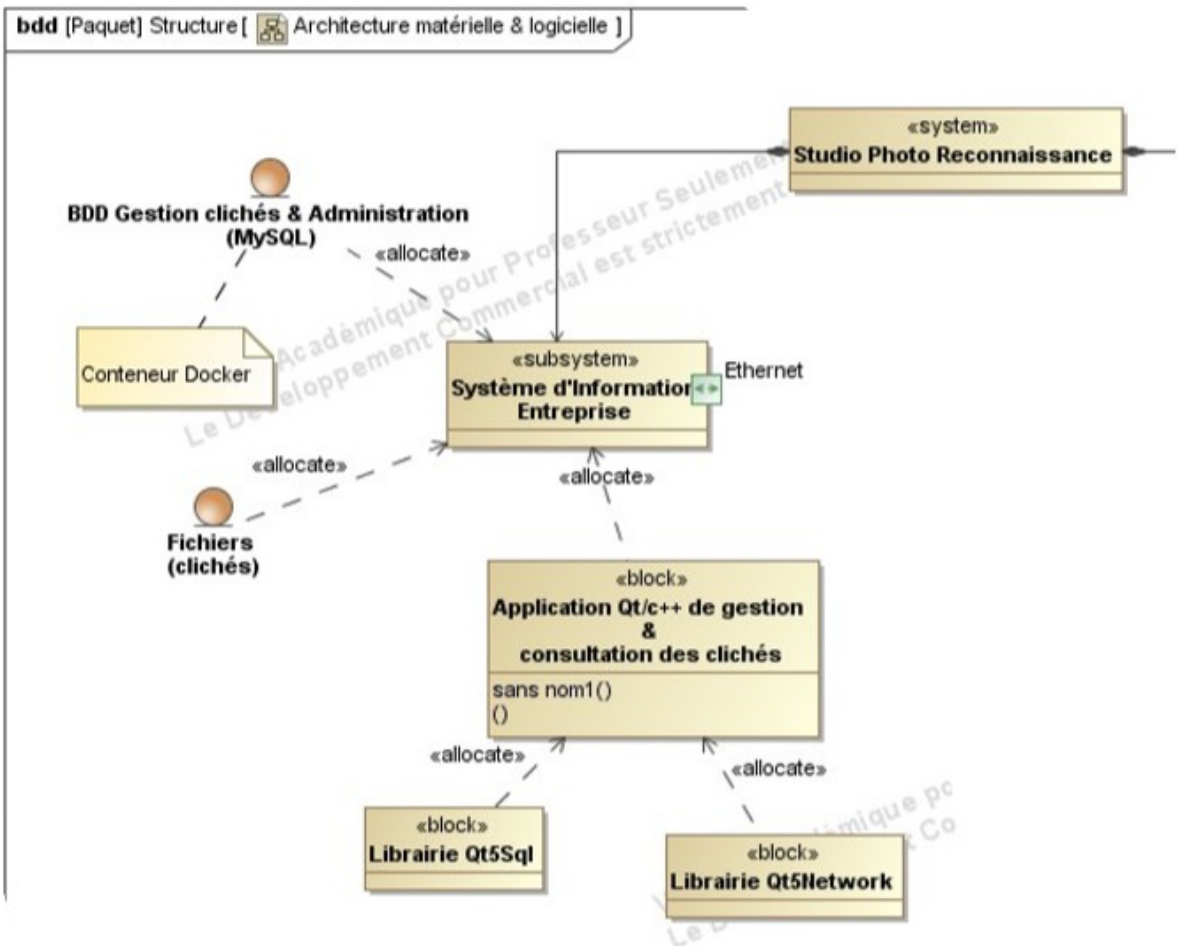


Diagramme de bloc :



Planification prévisionnelle

Projet SPR	43,43 jrs?	Lun 06/01/20	Ven 19/06/20
Revues de projets	42,86 jrs?	Lun 06/01/20	Ven 19/06/20
Revue de projet 0	3,43 jrs	Lun 06/01/20	Mer 15/01/20
Revue de projet 1	2,14 jrs?	Lun 10/02/20	Ven 14/02/20
Revue de projet 2	2,14 jrs	Lun 04/05/20	Ven 08/05/20
Remise du projet	2,14 jrs?	Lun 25/05/20	Ven 29/05/20
Soutenance avec le jury	2,14 jrs?	Lun 15/06/20	Ven 19/06/20
Réalisation du projet	34,86 jrs?	Lun 06/01/20	Lun 25/05/20
S'approprier le fonctionnement de Docker	3 jrs?	Lun 06/01/20	Mar 14/01/20
Installer le SGBD dans le conteneur Docker	8 jrs	Mar 14/01/20	Lun 10/02/20
Concervoir une BDD destinée à stocker les informations liées aux clichés réalisés avec le studio photo	2 jrs	Lun 10/02/20	Lun 02/03/20
QT: Concevoir l'IHM	2 jrs	Lun 02/03/20	Jeu 05/03/20
QT: Coder l'IHM	17 jrs	Lun 09/03/20	Jeu 14/05/20
QT: Tester l'IHM	2 jrs	Mar 19/05/20	Lun 25/05/20

Planification réelle

S'approprier le fonctionnement de Docker	2,5 jrs	Lun 06/01/20	Lun 13/01/20
Installer le SGBD dans le conteneur Docker	8 jrs	Lun 13/01/20	Jeu 06/02/20

Docker

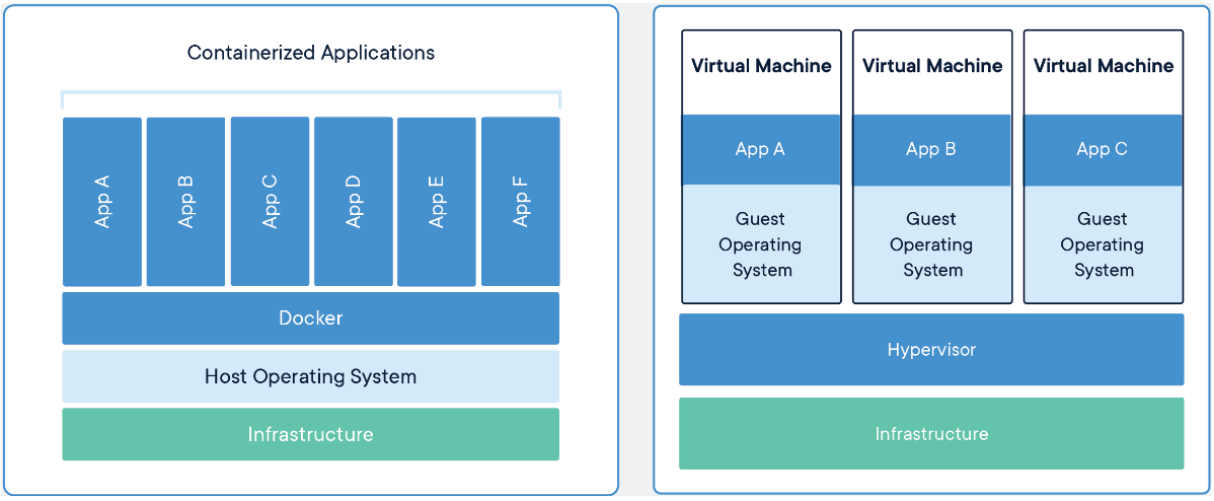
Docker c'est quoi ?

Pour commencer il a fallut que je m'approprie le fonctionnement de Docker et pour ça il est essentiel de comprendre ce qu'est Docker et pourquoi l'utiliser.

Docker est un logiciel pouvant fonctionner sur Linux, Windows et MAC. Docker est un gestionnaire de conteneurs, pour être plus précis, Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur.

A présent il faut comprendre ce qu'est un conteneur,

Docker + conteneurs **Machines virtuelles**



Comme on peut le voir ci-dessus Docker, facilite énormément la chose que ce soit en termes de compréhension mais aussi en performance. Docker s'occupe seulement de gérer les conteneurs, alors que si on utilise des machines virtuelles qu'on doit faire tourner en arrière plan, donc faire tourner des OS ce qui est clairement plus lourd et donc beaucoup moins optimisé.

Docker a aussi l'avantage d'être très mobile car les environnements contenus dans les conteneurs sont extrêmement léger, pour exemple l'image d'Ubuntu fait 70 Mo et pour un Opensuse l'image fait 103 Mo. Ce qui est extrêmement léger comparé à la taille réelle.
Mais il faut se dire que si l'image est si petite c'est que il y a eu des restrictions, c'est-à-dire que l'image contient seulement les fonctionnalités nécessaire et donc si on veut rajouter des fonctionnalités il faudra installer des packages supplémentaires.

L'un des grands avantages de Docker est aussi la facilité d'obtenir des images, par exemple pour obtenir l'image de la dernière version de Mysql il suffit de saisir dans PowerShell :

```
docker pull mysql:latest
```

Pourquoi utiliser Docker ?

Comme on l'a vu Docker est très léger, mobile et une fois configuré facile à déployer peu importe l'OS de l'host.

Partie pratique Docker

Pour faciliter la suite du projet j'ai décidé d'utiliser un conteneur PhpMyAdmin afin d gérer la base de donnée plus efficacement. Et donc pour utiliser plusieurs conteneurs on peut soit lancer les deux indépendamment ou utiliser ce qu'on appelle Docker Compose qui permet de regrouper plusieurs conteneurs en seul service afin de faciliter leurs manipulations.
Pour utiliser Docker compose il faut un fichier docker-compose.yml contenant la configuration voulue, pour mon cas mon fichier ressemble à ça :

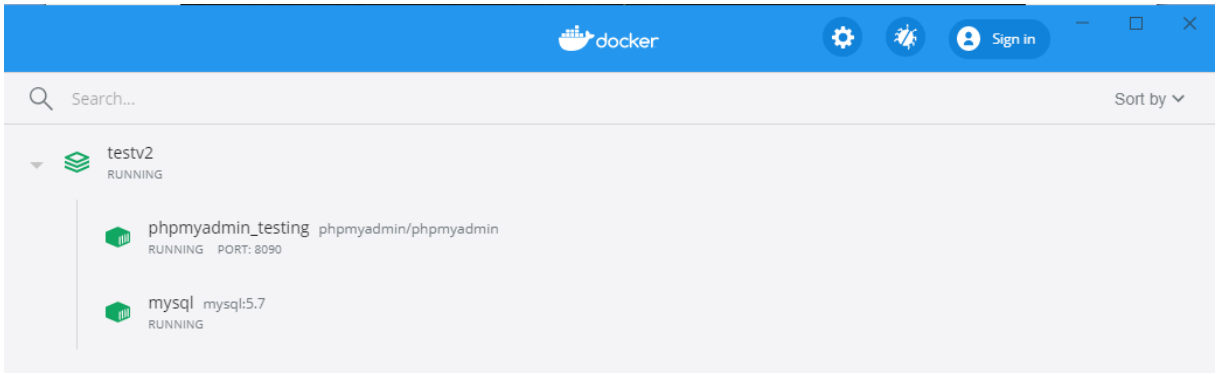
```yaml
version: '3.1'    #Version de docker-compose

services:
  db_server:
    image: mysql:5.7    #image mysql sous la version 5.7
    container_name: mysql   #Nom du conteneur
    volumes:                                #partage d'un dossier en temps réel
      - ../testbasededonnee:/var/liv/mysql    #entre l'hôte et le conteneur
    environment:
      MYSQL_ROOT_PASSWORD: admin    #Définir le mot de passe du root
      MYSQL_DATABASE: crossdock   #Définir le nom de la BDD

  phpmyadmin:
    image: phpmyadmin/phpmyadmin    #dernière version de phpmyadmin
    container_name: phpmyadmin_testing    #nom du conteneur
    ports:
      - 8090:80   #redirection des ports
    environment:
      PMA_HOST: mysql   #définir l'hôte
      PMA_PORT: 3306    #port de Mysql
    depends_on:
      - db_server    #dépendance de Mysql

```

Et pour déployer le docker-compose rien de plus simple :

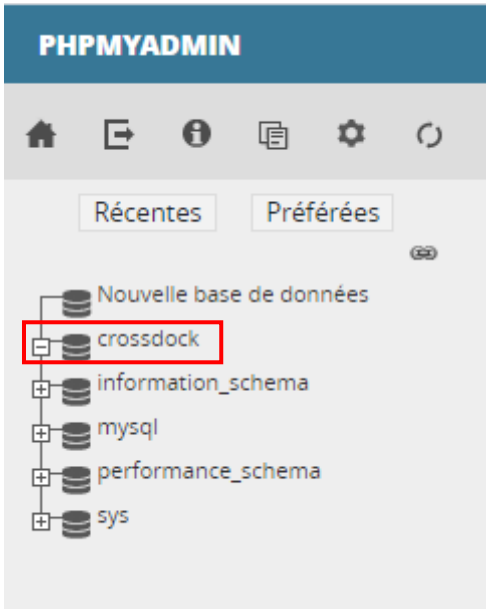
```
PS C:\Users\vives\Desktop\Docker\testv2> docker-compose up -d
Creating network "testv2_default" with the default driver
Creating mysql ... done
Creating phpmyadmin_testing ... done
```

(–d permet de lancer en arrière plan)

Grâce à l'interface graphique de Docker on peut avoir un petit aperçu des conteneurs, et pour moi ça donne ceci :

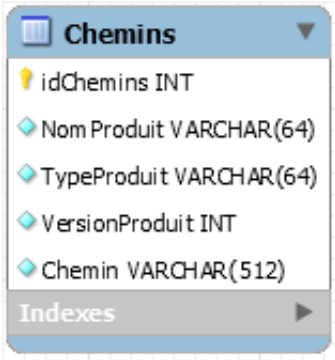


Et on peut également accéder à la base de données via PhpMyAdmin :



Base de données

Pour commencer j'ai conçu une base donnée temporaire extrêmement simple afin de voir si tout fonctionne avec Mysql, et donc la base de données subira sûrement des changements plus tard. Pour le moment elle ressemble à ceci :



Numéro désignant chemin d'accès du produit
Le nom du produit
Le type du produit
La version du produit (qui changera si y'a changement du packaging)
Le chemin d'accès aux clichés (dossier partagé sur le réseau)

Le script SQL qui va désigner la base de données ressemble à ceci :

```
CREATE TABLE bdd(
    idChemins INT PRIMARY KEY,
    NomProduit VARCHAR(64),
    TypeProduit VARCHAR(64),
    VersionProduit INT,
    Chemin VARCHAR(512)
);
```

Conclusion

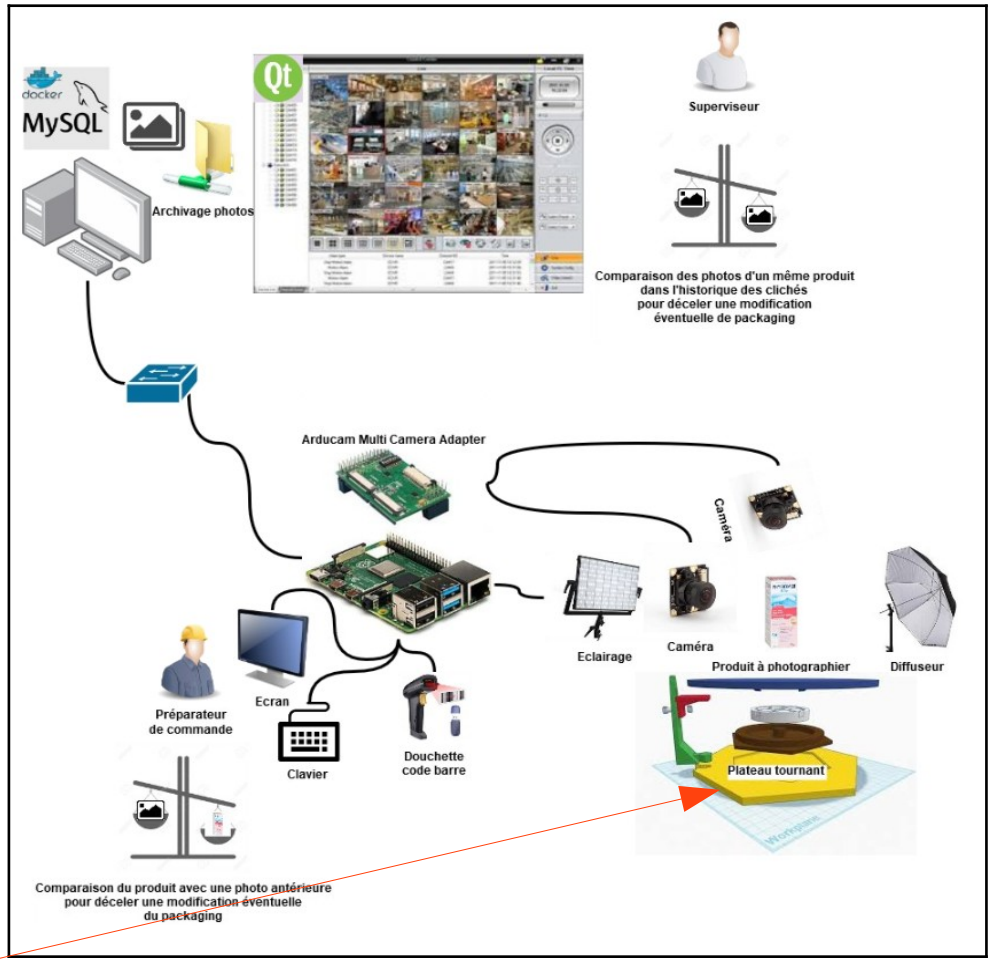
Pour conclure, je dirai que pour l'instant ayant beaucoup utilisé Docker qui est un outil très utile mais que je trouve parfois pas assez aboutie, pour apprendre de ce logiciel il faut se renseigner sur internet étant donné que je suis sous Windows et que la plupart des personnes utilisent ce logiciel sur Linux, qui rend compliqué de trouver les informations qu'on recherche. Le logiciel donne très peu d'informations sur les erreurs qu'on a.
J'ai commencé un peu à faire l'IHM surtout à essayer de se connecter à la BDD à partir de QT.

Sur l'avancée du projet j'ai réalisé la partie « s'approprier le fonctionnement de Docker » ainsi que « installer le SGBD dans un conteneur Docker ».
Je suis en train de réaliser la partie « concevoir une base de données destinée à stocker les informations liées aux clichés réalisés avec le studio photo »

Il me restera donc à concevoir➔coder➔tester l'IHM Qt.

Partie EC1 : Pamatto Thomas

Rpi Plateau tournant



Voici la partie “plateau tournant” du système que je dois gérer en parallèle de l’étudiant EC2. Mon devoir pour la revue 3 est de concevoir un PCB permettant de s’adapter au GPIO de la RaspBerry Pi et de contrôler le moteur avec précision.

Planning Prévisionnel

Specifications générales	**Lun 06/01/20**	**Lun 06/01/20**		
Planning prévisionnel	Lun 06/01/20	Mar 07/01/20		
Convenir charte graphique	Lun 06/01/20	Mar 14/01/20		
Essais et validation des structures (prototypage rapide)	**Lun 06/01/20**	**Jeu 06/02/20**	**12**	
Essai carte Arduino et driver A4988 + moteur pas à pas	Jeu 09/01/20	Mar 14/01/20		
Essai Arduino + RPI	Mar 14/01/20	Mar 21/01/20		
Essai avec ATtiny85	Mar 21/01/20	Mar 28/01/20		
Réléchir sur les solutions de driver, de moteur etc	Mar 28/01/20	Lun 03/02/20		
Illustrer les essais avec Fritzing	Lun 03/02/20	Mar 04/02/20		
Dresser la liste des composants etc	Mar 04/02/20	Jeu 06/02/20	20	
Prototypage rapide et routage	Lun 09/03/20	Jeu 09/04/20	9	
Fabrication et essais	Lun 04/05/20	Jeu 11/06/20	10	
Fichier de fabrication PCB finalisés	Lun 06/04/20	Jeu 09/04/20		
Rédaction du dossier	Jeu 09/01/20	Mer 27/05/20		

Diagramme de Gantt qui permet de planifier les tâches à effectuer jusqu'au 09/04/2020

Avancement de la solution étape par étape

Pour commencer, j'ai décidé d'utilisé le driver pour moteur pas à pas A4988. Pour le tester j'ai du le câbler avec un moteur pas à pas et une Arduino Uno.
Le fabricant du module précise que celui ci fonctionne avec des tensions allant de 3V à 5,5V. La carte Arduino Uno fournissant du 5V, on peut les brancher sans crainte. Il nous indique aussi que l'alimentation du moteur par le module se fait de 8V à 35V. Notre moteur étant un moteur bipolaire de 12V, pourra donc être correctement alimenté.

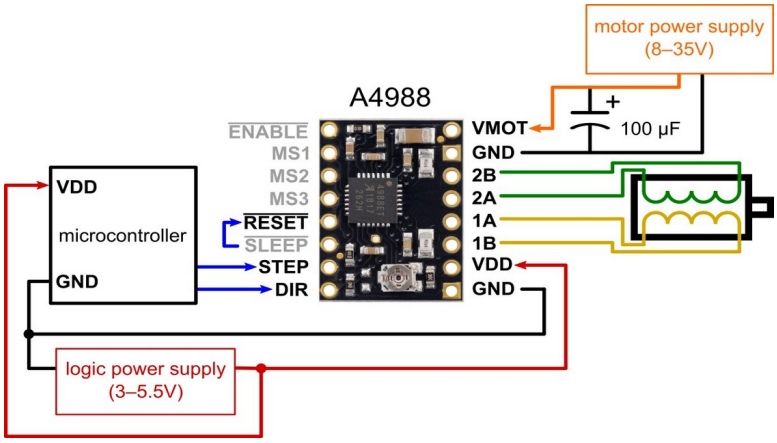


Figure 1:Module A4988

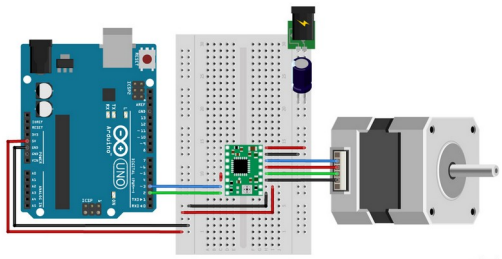


Figure 2:Branchement de la carte Arduino du driver et du moteur

Le branchement de la carte Arduino Uno, du driver A4988 et du moteur pas à pas se fera sur une plaque de prototypage rapide (breadboard) en suivant ce schéma fritzing. Une fois correctement câblé et en ayant téléversé dans la carte Arduino le programme qui commande le moteur celui ci tourne dans un sens puis dans l'autre.

Une fois cette étape réalisé, il faut que j'arrive à faire communiquer ma Raspberry Pi et Arduino grâce au bus I2C

Communication BUS I2C Raspberry Pi et Arduino

Voulant faire communiquer Raspberry Pi et Arduino grâce au bus I2C, je me heurte à un problème : la Raspberry Pi fournit une tension de 3,3V tandis qu'Arduino fournit une tension de 5V. Pour le surmonter et grâce à l'aide de mon professeur j'ai décidé d'utiliser un élévateur de tension (on utilisera le terme level shifter pour la suite). J'ai utilisé la structure ci-dessous pour le level shifter. Les transistors utilisés sont des VN2222.

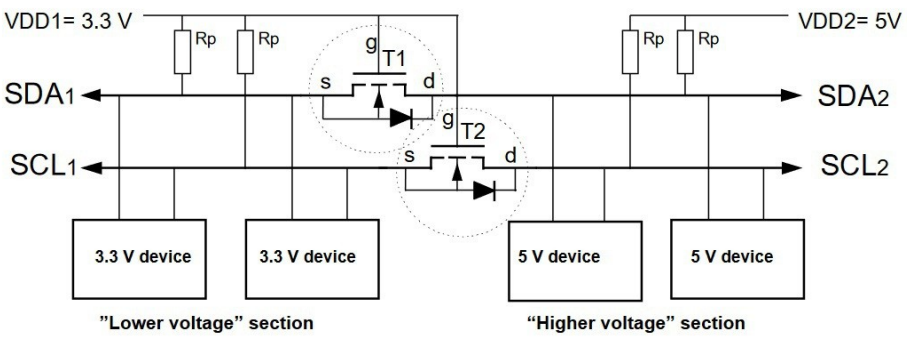


Figure 3:Structure du level shifter

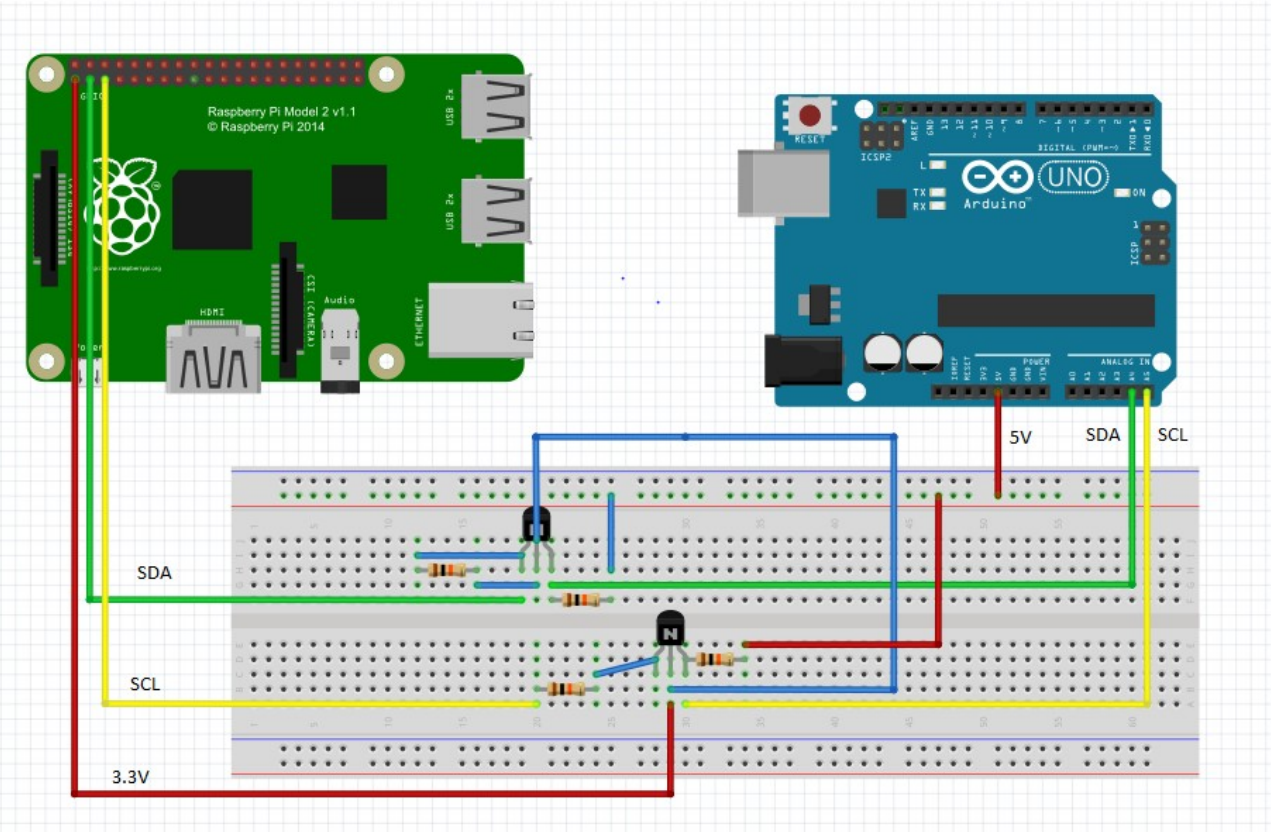


Figure 4: Schéma de câblage Arduino RPI en i2c

J'ai réalisé ce montage en accord avec le schéma de prototypage précédent.Pour faire fonctionner ce montage, j'ai du m'inspirer d'un programme d'un TP effectué au cours de l'année. Le programme Arduino à aussi été récupéré sur un ancien TP de communication Arduino i2c.

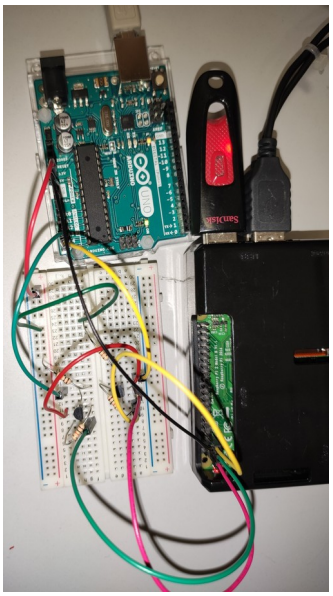


Figure 5: Câblage I2C Arduino Esclave RPI Maître

Programme RPI

```
#include <iostream>
#include <bcm2835.h>
using namespace std;

#define clk_div BCM2835_I2C_CLOCK_DIVIDER_2500
```

Le programme suivant est le programme permettant de lire les 6 octets envoyés par l'esclave.
La librairie bcm2835 est utilisée pour la communication i2c dans le programme du maître en C++

```
void init_I2c_bcm2835()
{
        if (!bcm2835_init())
                {
                printf("bcm2835_init failed. Are you running as root??\n");
                }
        if (!bcm2835_i2c_begin())
                {
                printf("bcm2835_i2c_begin failed. Are you running as root??\n");
                }
        bcm2835_i2c_setClockDivider(clk_div);                   // Division de l'horloge Rpi pour obtenir une vitesse de 100KHz
}
```

Initialisation de le librairie bcm2835

```
int main(void)
{
        //double mesure;
        unsigned int slave_address=0x08; // Adresse du DS1621
        char i2cOut[3];         // Tableau des données I2C à sortir
        char i2cIn[6];          // Tableau des données I2C entrées

        init_I2c_bcm2835();
        bcm2835_i2c_setSlaveAddress(slave_address);     // Affectation de l'adresse de l'esclave


        delay(1000);                                    // 1 seconde avant première lecture

        while(1)
                {

                bcm2835_i2c_read(i2cIn,6); // Lis les 6 bytes que l'eclave envoie
                cout<<'valeur reçue : '<<i2cIn[0]<<i2cIn[1]<<i2cIn[2]<<i2cIn[3]<<i2cIn[4]<<i2cIn[5]<<endl;
                delay(1000);                                                                    // 1 seconde entre chaque conversion
                }

        bcm2835_close();
        return 0;
}
```

char i2cIn[6]:Les données attendues par le maître sont de 6 octets.

bcm2835_i2c_setSlaveAdress(slave_adress) : configure l'adresse de l'esclave sur 0x08.

bcm2835_i2c_read(i2cIn,6) : Le maître demande à lire les 6 octets que l'esclave envoie.
La dernière ligne « cout » affiche les valeurs reçues c'est à dire les 6 octets.

Programme Arduino

Ce programme permet d'envoyer le mot « hello » codé sur 6 octets a la RPI maître.

```
#include <Wire.h>
```

Librairie gérant l'i2c sur Arduino.

```
void setup()
{

Wire.begin(8); // join i2c bus with address #8

Wire.onRequest(requestEvent); // register event

}


void requestEvent()
{

Wire.read("hello "); // respond with message of 6 bytes

// as expected by master

}
void loop()
{

delay(100);
}
```

Wire.begin() configure l'esclave a l'adresse 0x08.

Création de la fonction requestEvent() qui envoie « hello » sur le bus i2c.

Entre chaque envoie du mot « hello », on attend 0,1 seconde.

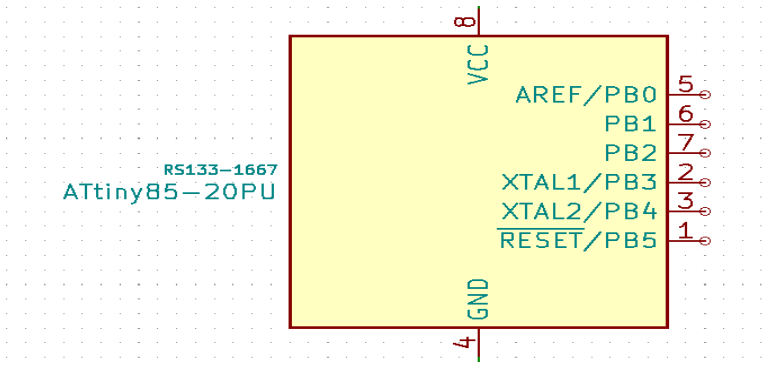
```
pi@raspberrypi
Fichier  Édition  Onglets  Aide
pi@raspberrypi:~ $ sudo ./RPI_ARD_1
sudo: ./RPI_ARD_1 : commande introuvable
pi@raspberrypi:~ $ ls
bcm2835-1.60       Desktop    Downloads  Music     Public     Travail
bcm2835-1.60.tar.gz  Documents  MagPi      Pictures  Templates  Videos
pi@raspberrypi:~ $ cd \ Travail
bash: cd:  Travail: Aucun fichier ou dossier de ce type
pi@raspberrypi:~ $ cd \Travail
pi@raspberrypi:~/Travail $ ls
Capteur_DS1621  DS1621_V1_figoureux.deazevedo  Projet  StoreMotorise
pi@raspberrypi:~/Travail $ cd \Projet
pi@raspberrypi:~/Travail/Projet $ ls
Programme  progrpi_arduino1.cpp
pi@raspberrypi:~/Travail/Projet $ sudo ./RPI_ARD_1
sudo: ./RPI_ARD_1 : commande introuvable
pi@raspberrypi:~/Travail/Projet $ cd \Programme
pi@raspberrypi:~/Travail/Projet/Programme $ ls
progrpi_arduino1.cpp  RPI_ARD_1
pi@raspberrypi:~/Travail/Projet/Programme $ sudo ./RPI_ARD_1
valeur reçue : 
valeur reçue : 
valeur reçue : 
valeur reçue : 
^C
pi@raspberrypi:~/Travail/Projet/Programme $ sudo ./RPI_ARD_1
valeur reçue : hello
valeur reçue : hello
valeur reçue : hello
valeur reçue : hello
valeur reçue : hello
valeur reçue : hello
valeur reçue : hello
^C
pi@raspberrypi:~/Travail/Projet/Programme $
```

Figure 6: Programme "hello" en fonctionnement

Création du schéma électrique

La création du schéma électrique se fait avec le logiciel KiCad.

Pour commencer à créer un schéma électrique sur KiCad, notre professeur d'électronique nous à donner des fichiers contenant les symboles et les empreintes des composants.
Certains composant n'étant pas inclut directement dans KiCad, il a du les créer.



Symbole du composant « ATtiny85 »

Symbole: Attribution Empreintes

1	A1	Breakout_A4988	0_ModulesProjet:A4988_Breakout_sans_dissipateur
2	C1	100nF	Capacitor_SMD:C_0805_2012Metric
3	C2	100nF	Capacitor_SMD:C_0805_2012Metric
4	C3	100nF	Capacitor_SMD:C_0805_2012Metric
5	C4	100nF	Capacitor_SMD:C_0805_2012Metric
6	C5	100µF	Capacitor_Tantalum_SMD:CP_EIA-3216-10_Kemet-I
7	D1	LED	LED_SMD:LED_0805_2012Metric
8	D2	LED	LED_SMD:LED_0805_2012Metric
9	D3	PMEG3010ER	Diode_SMD:D_SOD-123
10	FU1	Polyfuse	0_ModulesProjet:Polyswitch_MICROSMD110F-2
11	J1	Conn_02x13_Odd_Even	Connector_PinSocket_2.54mm:PinSocket_2x13_P2.54mm_Vertical
12	J2	Barrel_Jack	0_ModulesProjet:Fiche_alimentation_DC
13	J3	SDA	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
14	J4	STEP	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
15	J5	DIR	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
16	J6	M1	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
17	J7	M2	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
18	J8	M3	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
19	J9	M4	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
20	J10	GND	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
21	J11	+12V	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
22	J12	3.3V	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
23	J13	SCL	Connector_PinHeader_2.54mm:PinHeader_1x01_P2.54mm_Vertical
24	J14	JST B4B-XH-A	0_ModulesProjet:Embase_JST_Angle_Droit_S4B-XH-A_LF_SN
25	J15	Barette mâle 4pts	Connector_PinHeader_2.54mm:PinHeader_1x04_P2.54mm_Vertical
26	R1	330	Resistor_SMD:R_1206_3216Metric
27	R2	2,2k	Resistor_SMD:R_1206_3216Metric
28	U1	ATtiny85-20PU	Package_DIP:DIP-8_W7.62mm

Nom du symbole

Empreinte du symbole

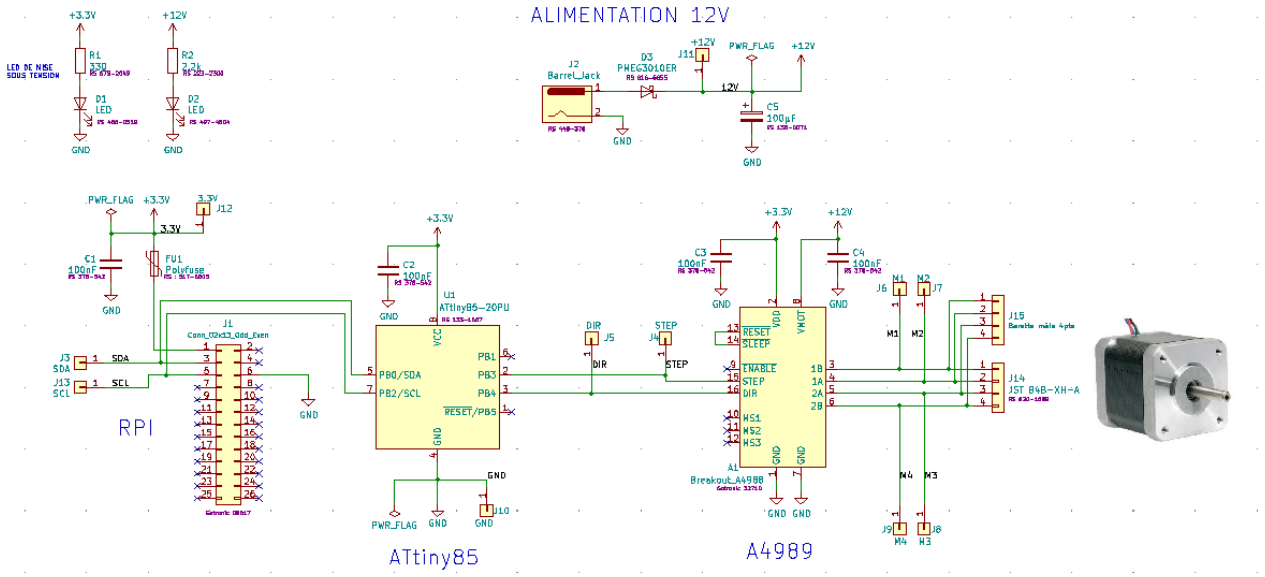
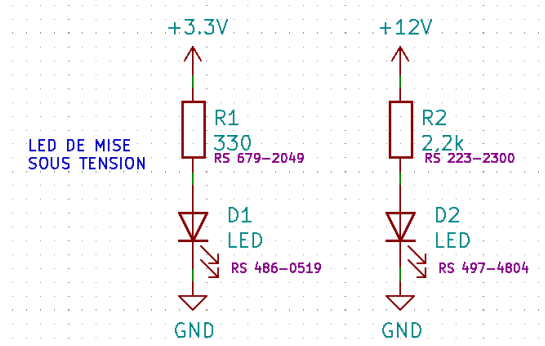
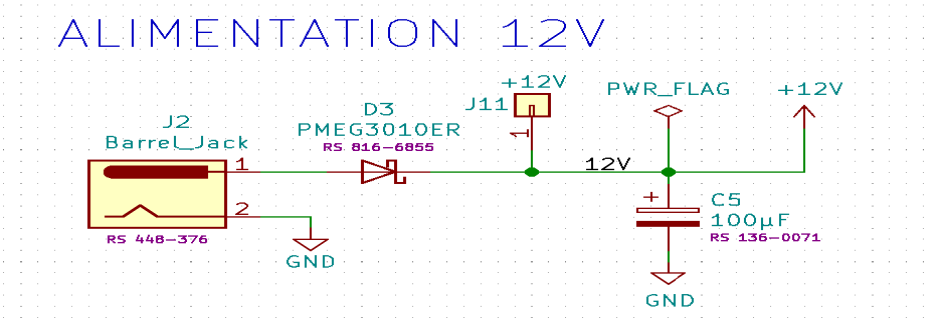


Figure 7: Vue d'ensemble du schéma électrique fini

Explication du schéma électrique



Les led de mise sous tension qui va nous renseigner sur l'état du circuit. Elles sont aux nombres de deux : une pour le 3,3V et une pour le 12V.



L'alimentation 12V est protégée par une diode shottky et un condensateur pour éviter une alimentation à l'envers de la carte.

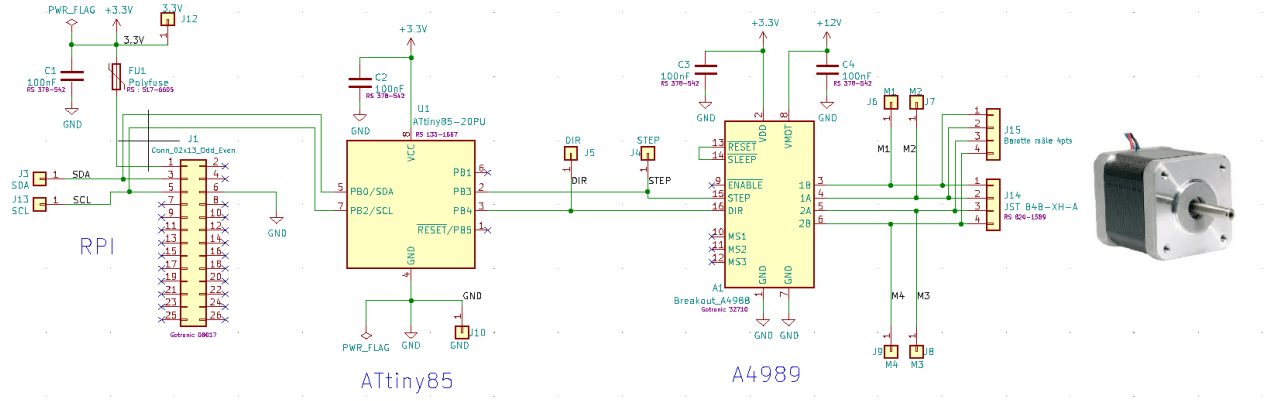
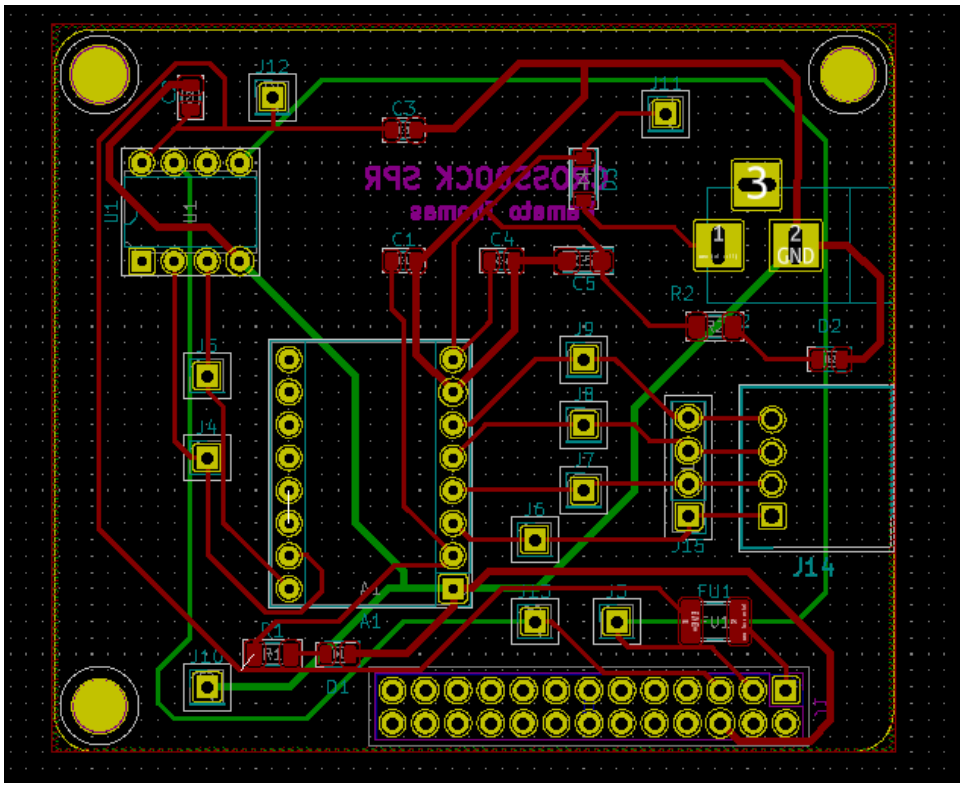


Figure 8: RPI,ATtiny85 et A4889

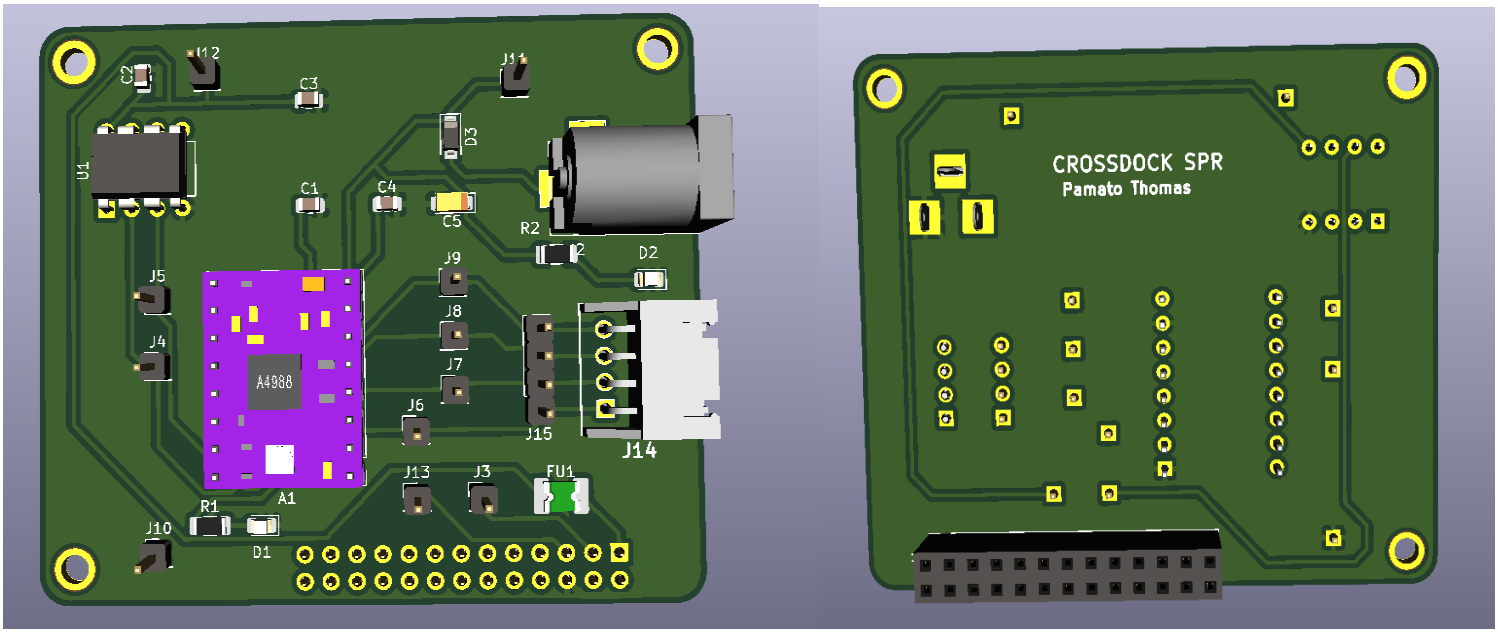
A gauche : le GPIO de la RPI est protégé par un condensateur de découplage et un polyfuse.
Au centre : l'ATTiny85 et le A4988 sont reliées. Les 2 composants sont tous les deux protégées par des condensateur de découplages.
A droite : Les connecteurs du moteur sont reliés à l'A4988. Un deuxième connecteur est mis à disposition pour d'éventuelle changement de connectiques du moteur.
Des points de test sont aussi mit à disposition pour effectué des mesures si besoin.

Création du circuit imprimé (PCB)

La création du circuit imprimé ce fait sur PCBnew qui est un outils de conception de circuit imprimé de KiCad.



Ici, on peut voir la carte quasiment finie avec tout les composants qui sont reliés entre eux.



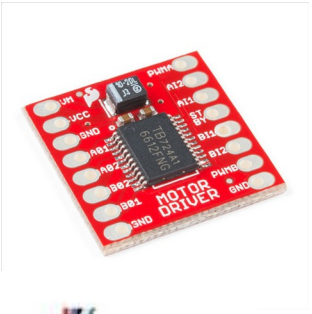
Grâce à la visualisation 3D, on peut voir notre carte tel qu'elle sera quand on l'aura soudé. Ainsi, on peut déceler d'éventuelle problème de fiabilité avant de commander la carte.

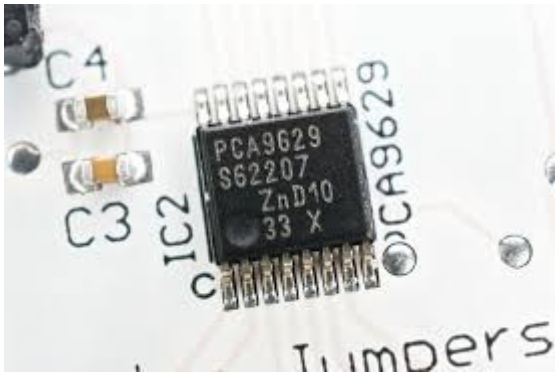
Partie EC2 : Guénard Gauthier

Gestion RPI plateau tournant

Présentation :

Mon Objectif dans le groupe est d'établir la communication et le contrôle entre la RPI et le moteur pas à pas.
Le système devra faire tourner un plateau avec un colis dessus, puis une caméra devra prendre des photos des différentes faces.
La RPI communiquera avec un Driver de moteur pas à pas, par L'I2C et celui ci fera faire des rotations au moteur.





Avancement du projet :

Choix des composants utilisés :

Je dois utiliser une RPI qui communiquera avec des drivers (breakout) qui contrôlerons un moteur pas à pas.

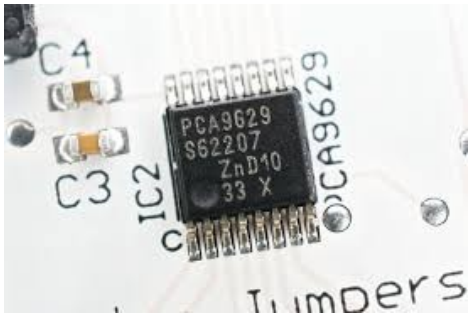
L'une des premières solution était un PCA9269 avec un TB6612FNG.

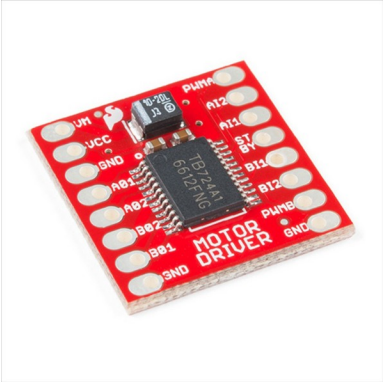
La conception d'un level shifter était impératif pour établir une communication,car la RPI communique en 3,3V sur l'I2C et le PCA9629 est en 5V.

Table 38. Static characteristics

V_{DD} = 4.5 V to 5.5 V; V_{SS} = 0 V; T_{amb} = −40 °C to +85 °C; unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Supply						
V_{DD}	supply voltage		4.5	-	5.5	V
I_{DD}	supply current	operating mode; no load; f_{SCL} = 1 MHz; V_{DD} = 5.5 V	-	6	10	mA
I_{stb}	standby current	no load; f_{SCL} = 0 kHz; MODE[6] = 1; OSC off; V_I = V_{DD} or V_{SS}; V_{DD} = 5.5 V	-	600	800	µA
V_{POR}	power-on reset voltage	no load; V_I = V_{DD} or V_{SS}	-	2.3	-	V
V_{PDR}	power-down reset voltage	no load; V_I = V_{DD} or V_{SS} [1]	-	2.0	-	V
Input SCL; input/output SDA						
V_{IL}	LOW-level input voltage		−0.5	-	+0.3V_{DD}	V
V_{IH}	HIGH-level input voltage		0.7V_{DD}	-	5.5	V
I_{OL}	LOW-level output current	V_{OL} = 0.4 V; V_{DD} = 5.0 V	30	40	-	mA
I_L	leakage current	V_I = V_{DD} or V_{SS}	−1	-	+1	µA
C_i	input capacitance	V_I = V_{SS}	-	6	-	pF

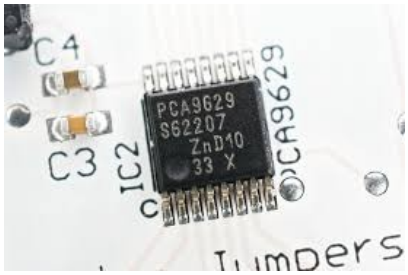




Pour la seconde il y a le PCA9629 et le L298 avec toujours un level shifter entre la RPI et le PCA9629.

Pour l'instant la première a été fini d'être câblée et testé, il ne reste plus qu'a faire quelques correction sur le software et commander un autre moteur pas a pas qui consomme moins, cette solution fonctionne et elle est viable.

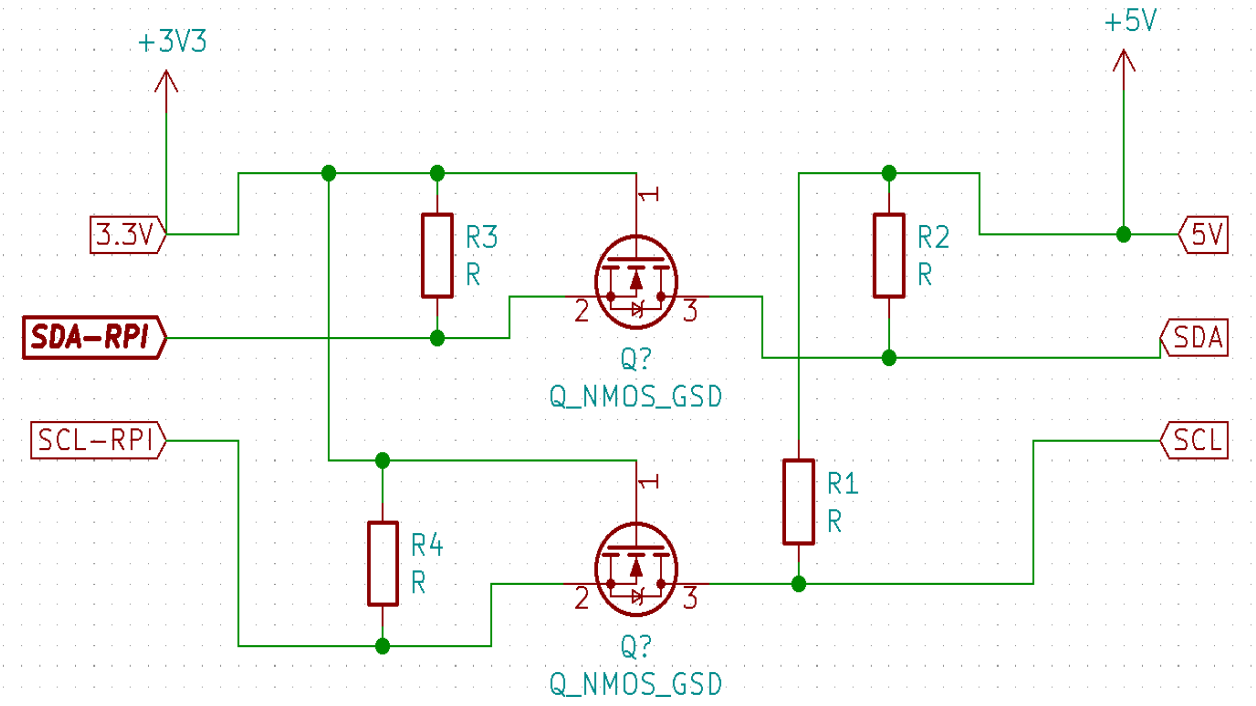
La seconde solution n'as pas encore été tester, mais déjà nous savons que le L298, mais nous savons que sont architecture est composé de transistors MOS et de porte logique &, font qu'il consomme plus de courant et chauffe encore plus de ce fait, ce qui rend cette solution non viable



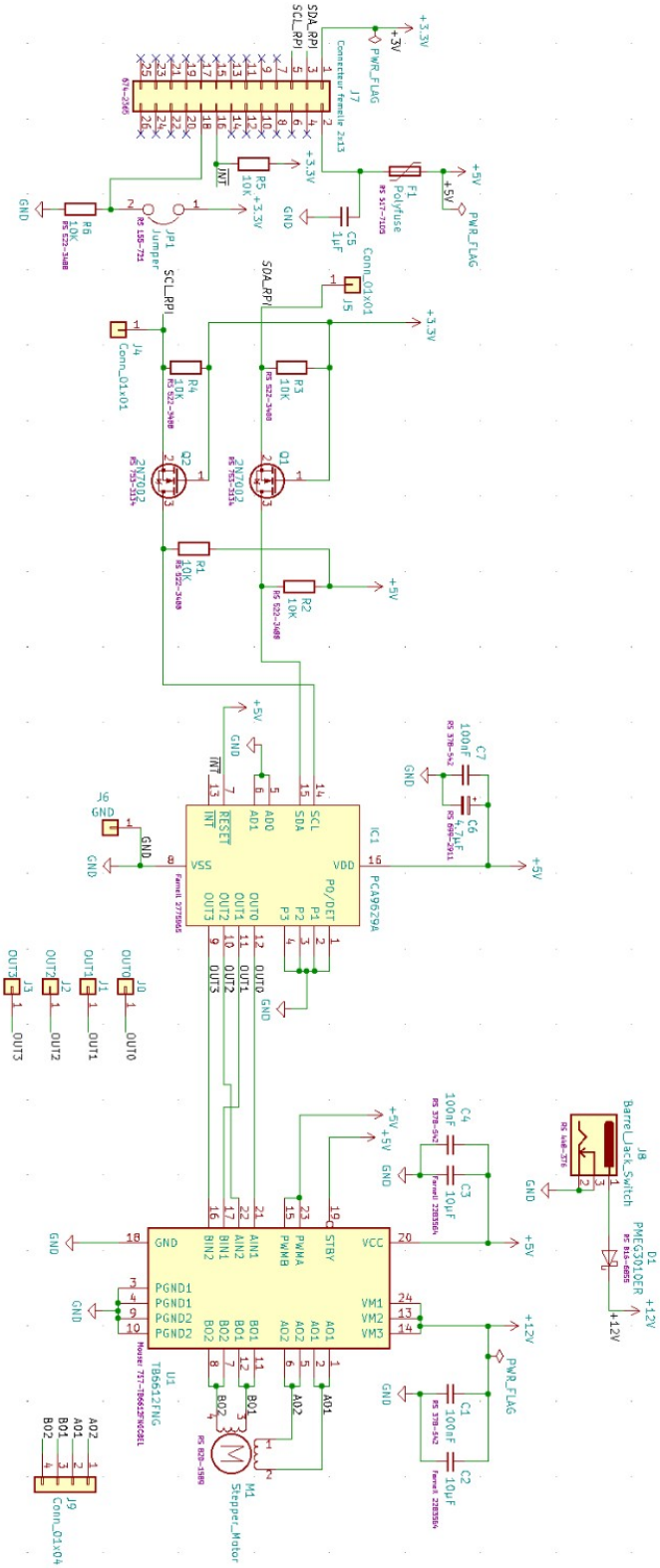
Le programme m'a été fourni par l'étudiant IR2

Level Shifter :

Pour la liaison entre la RPI et le PCA9629 nous avons du utiliser un level shifter.
Pour celui-ci j'ai utiliser des transistor MOSFET CANAL N VN2222 ainsi que 4 Résistance de pull up de 10KΩ chacune.



Le schéma :



Calcul de dissipation thermique

Ta = 20°C
Tj = 175°C
Pmax = (0,7*1)*2=1,4 *2 car nous avons 2 transistor
Rthja = (175- 20)/1,4 = 110°C/W

Thermal shutdown circuit operating temperature	TSD	(Designed value)	---	175	---	℃
Thermal shutdown hysteresis	△TSD		---	20	---	

Dans la documentations nous avons 160°C/W, il faudra donc un dissipateur.
Nous n'avons pas d'informations sur RthJc.
Et les 1A sont théorique sur les deux phase du moteur.
Comme notre système est rarement en rotation il ne consomme rien, et quand il l'est, il fait un pic d'intensité au démarrage et ce stabilise par la suite, pour diminuer la consommation j'ai penser a modifier les rampes d'accélération et de décélération dans le programme, et aussi choisir un autre moteurs qui consomme moins mais qu'il ait du couple quand même.
On prendra certainement un petit dissipateur car il en faut un mais, dans la théorie on ne peut pas le calculer précisément.

Je n'ai calculer qu'une seule des deux parie du circuit, car du coté de l'alimentation du moteur c'est là qu'il y a un fort courant, la partie logique est insignifiante comparée a celle ci donc chauffe très peu voir quasiment pas

Le moteur pas à pas

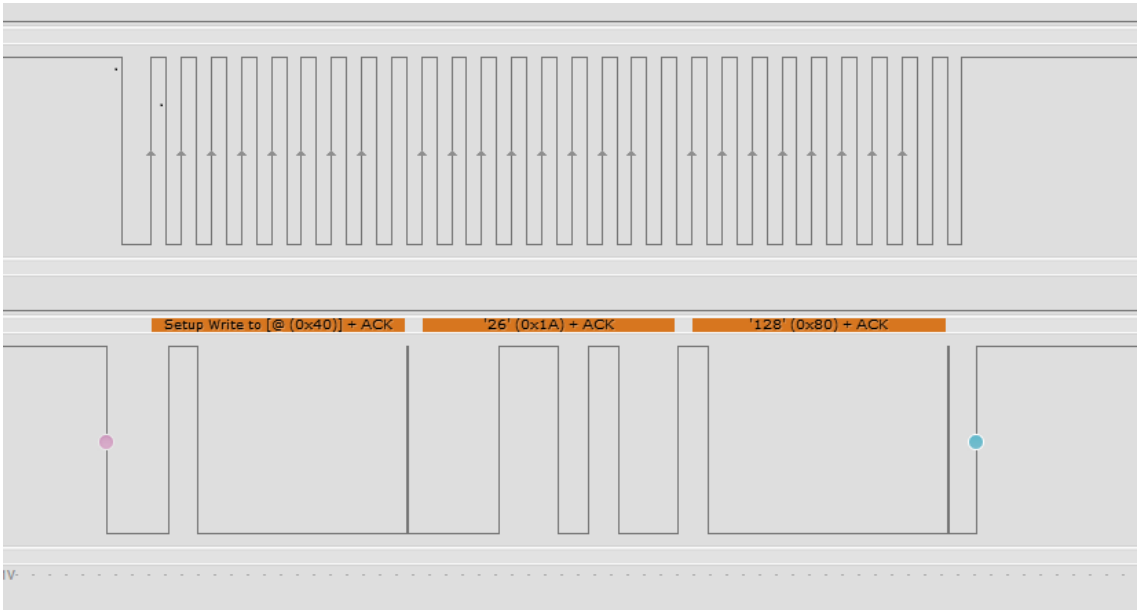
Nous utiliserons un moteur bipolaires
Cela on 4 fils et deux bobines
il y a plusieurs possibilités pour contrôler les pas d'un moteur.

-Full Step (Pas complet)
-Half Step (Demi pas)
-Microstepping (Multi pas)

avec le full step nous utilisons la totalité des pas d'un moteur
exemple avec un moteur de 200 pas, avec le Half Step il en ferait 400
et le Microstepping permet de pousser la résolution(précision) encore plus loin, certain microcontroller
peuvent aller (dans certaine cas) jusqu'à 256 fois la résolution de base d'un moteur
Dans le cas de notre microcontroller, la datasheet ne nous donne aucune information sur le Microstepping ou même Half Step.
Ce qui nous laisse que le Full Step comme seule possibilité.

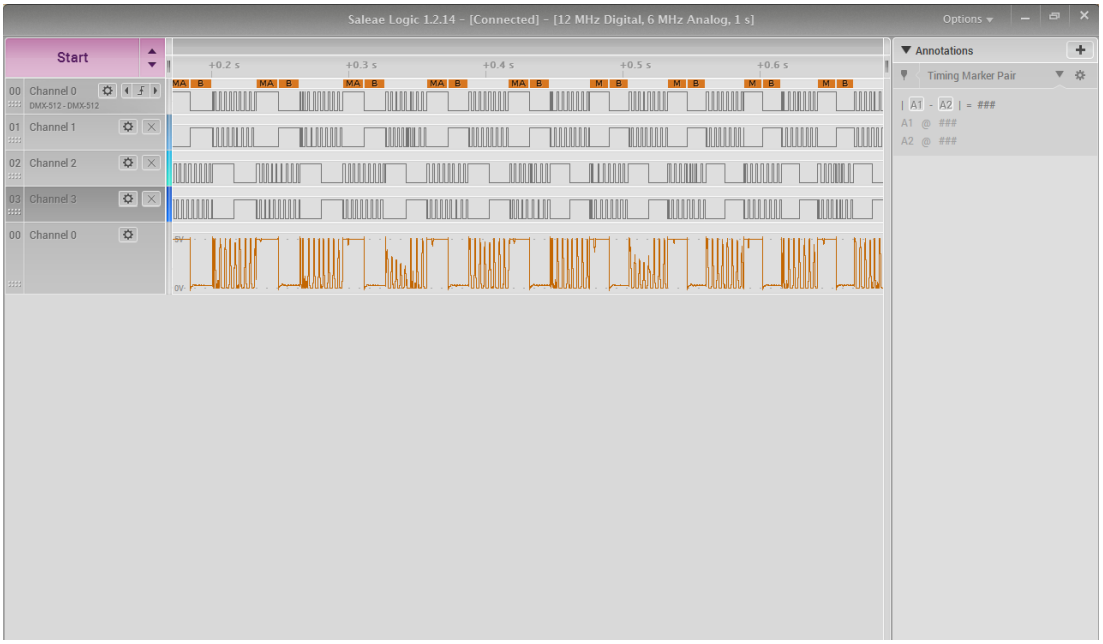
Pour notre projet, pas besoin de Half Step ou de Microstepping.
Nous n'avons pas besoin d'être précis au micron prés.

La trame I2C



i2cset -y 1 0x20 0x1a 0x80
Pour cette trame ci, c'est juste un exemple de communication entre la RPI et le PCA9629A

Voici la trame I2C de la RPI allant vers le PCA9269Tout d'abord il transmet a l'adresse 0x20 l'ordre d'aller dans le registre MCNTL 0x1A et finit par transmettre le bit de poids fort a 1 et les 7autres a 0



Voici les 4 trames du TB6612 qui sortent par A01 A02 B01 et B02 allant au moteur avec l'ancien programme .

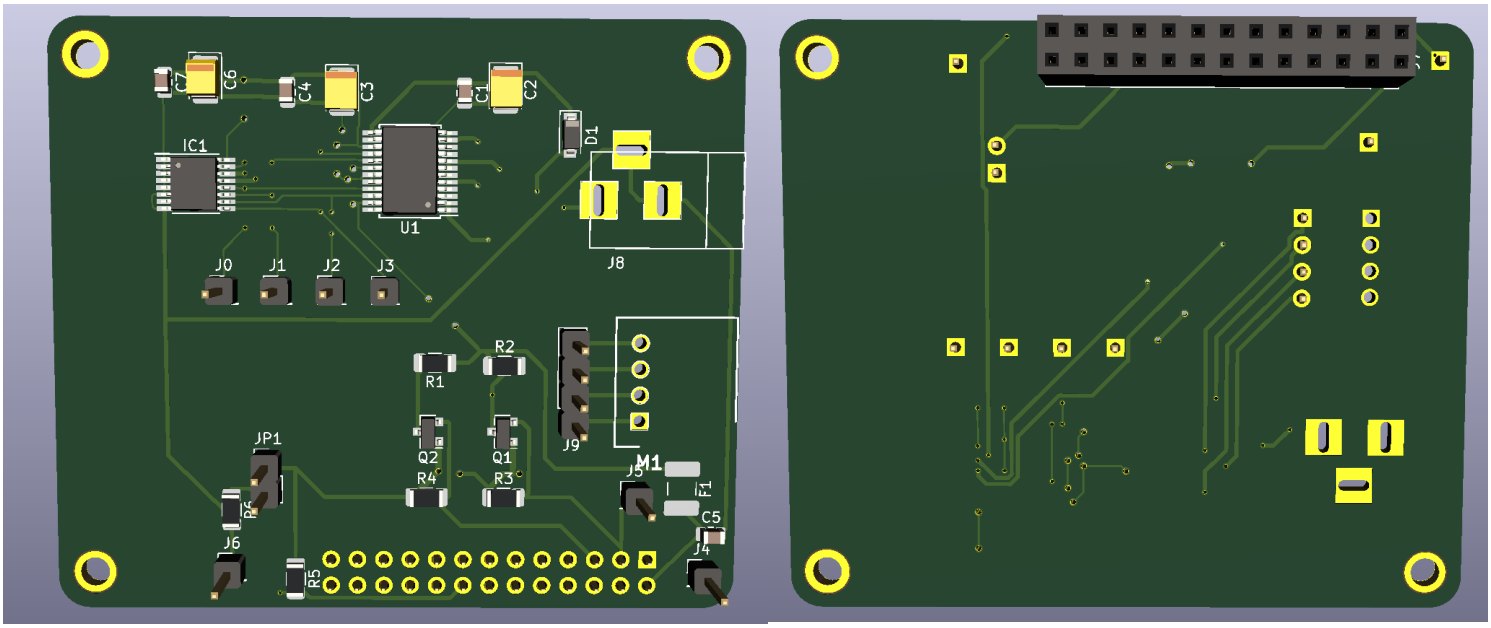
La programation :

Voici quelques lignes de code essentielles au fonctionnement du moteur pas à pas

```
# 3/ Configurer la vitesse de rotation horaire
# . CWPWL & CWPWH : Step pulse width for CW rotation (Low & High bytes)
#     CWPW = 0x010a => Prescaler = 0 & Pulse width = 0x10a
#                   => 266 impulsions de 3µs par seconde
i2cset -y 1 0x20 0x96 0x0a01 w

# 4/ Configurer le nombre de pas désirés pour la rotation horaire
# . CWSCOUNTL & CWSCOUNTH : Number of steps CW (Low & High bytes)
#     CWSCOUNT = 0x02cb => 715 pas
#     !! ATTENTION !! le nb de pas spécifié doit prendre en compte ceux
#     "`consommés`" pour l'accélération => 3 tours avec un moteur 400pas
#     nécessite en tout 3*400=1200pas mais 485 pas sont déjà utilisés par
#     l'accélération pour atteindre la vitesse nominale de 266 impulsions/s
#     avec une rampe de 5 (cf. valeur de RUCNTL). Donc il faut effectuer 715 pas
#     à la vitesse nominale pour faire 1200 pas en tout.
#     La valeur 485 a été déterminée expérimentalement par lecture des registres
#     STEPCOUNT0 à STEPCOUNT4 qui mémorise le nombre de pas effectifs.
i2cset -y 1 0x20 0x92 0xcb02 w


# 7/ Lancer rotation horaire
# . MCNTL : Motor start/stop and rotate direction control
#     MCNTL[7] = 1 => start motor
#     MCNTL[6] = 1 => re-start motor for new speed and operation
#     MCNTL[1:0] = 00 => rotate clockwise
i2cset -y 1 0x20 0x1a 0xc0

# Attendre fin de rotation (1200/266 => t >= 4s)
sleep 5


# Attendre fin de rotation (1200/266 => t >= 4s)
sleep 5

# 9/ Stopper le moteur
# . MCNTL : Motor start/stop and rotate direction control
#     MCNTL[7] = 1 => start motor
#     MCNTL[5] = 1 => emergency stop motor
i2cset -y 1 0x20 0x1a 0xa0
```

***Source M.DEFRANCE**

Le routage :



Voici la carte router, pour la conception je ne devais pas avoir un PCB de plus de 10cm par 10cm, comme on peut le voir il y a encore de la place.
Je vais continuer de la miniaturiser et de corriger le souci software qui m'empêche de choisir une plus grosse largeur de piste d'un coté du boîtier U1

Liste du matériel

Id	Référence	Boitier	Quantité	Désignation	Fournisseur et réference	Prix Unité TTC	Prix total TTC	Vendu par
1	J7	PinSocket_2x13_P2.54mm_Vertical	1	Connecteur femelle 2x13	Gotronic 8017	0,7	0,7	
2	M1	Embase_JST_Angle_Droit_S4B-XH-A_LF_SN	1	Stepper_Motor	RS 820-1589	0,542	2,735	5
3	,,	RPi_Hat_Mounting_Hole	3					
4	C1,C4,C7	C_0805_2012Metric	3	100nF	RS 264-4450	0,161	4,025	25
5	C5	C_0805_2012Metric	1	1Ï¼4F	RS 378-542			
6	C6	CP_EIA-3528-21_Kemet-B	1	4.7ÃµF	RS 378-542	0,028	0,7	25
7	F1	Polyswitch_MICROSMD110F-2	1	Polyfuse	RS 517-7105	0,33	3,3	10
8	IC1	TSSOP-16_4.4x5mm_P0.65mm	1	PCA9629A	Farnell 2775965			
9	J0	PinHeader_1x01_P2.54mm_Vertical	1	OUT0	Gotronic 08014	0,5	0,5	1
10	J1	PinHeader_1x01_P2.54mm_Vertical	1	OUT1	Gotronic 08014	0,5		1
11	J2	PinHeader_1x01_P2.54mm_Vertical	1	OUT2	Gotronic 08014	0,5		1
12	J3	PinHeader_1x01_P2.54mm_Vertical	1	OUT3	Gotronic 08014	0,5		1
13	J4,J5	PinHeader_1x01_P2.54mm_Vertical	2	Conn_01x01	Gotronic 08014	0,5		1
14	J6	PinHeader_1x01_P2.54mm_Vertical	1	GND	Gotronic 08014	0,5		1
15	J8	Fiche_alimentation_DC	1	Barrel_Jack_Switch	RS 448-376	2,88	2,88	1
16	J9	PinHeader_1x04_P2.54mm_Vertical	1	Conn_01x04				
17	JP1	PinHeader_1x02_P2.54mm_Vertical	1	Jumper				
18	Q1,Q2	SOT-23	2	2N7002	RS 535-9622	3,92	3,92	1
19	R1,R2,R3,R4,R5,R6	R_1206_3216Metric	6	10K	RS 125-1192	0,014	1,4	100
20	U1	SSOP-24_5.3x8.2mm_P0.65mm	1	TB6612FNG				
21	C2,C3	CP_EIA-3528-12_Kemet-T	2	10Ï¼4F	Farnell 2283564	0,391	1,955	5
22	D1	D_SOD-123	1	PMEG3010ER	RS 816-6855	0,284	14,2	50
					TOTAL		36,35 €	

Liste des tâches :

Date		Tâche	Durée
06/01/20		Briefing de départ + recherche documentation caméra PI	3 h
07/01/20		Révision schéma+ test nouveau moteur	4 h
13/01/20		étude du moteur pas a pas	2-3 h
14/01/20		étude des microcontroller cnc-shield	2 h
21/01/20		étude du microstepping	3 h
		aide entre EC1 et EC2	3 h
28/01/20		Teste et ajustement des rampes	3 h
03/03/20		Teste avec les deux moteurs et codes	3 h
09/03/20		Compréhension du coode avec IR2	1 h
16/03/20		Correction de mon shéma au totale	5h
19/03/20		Rédaction de la revue 2	2h30
		Routage de 3 cartes avec différent emplacements des composants	7h
30/03/20		recherche solution de chauffe	1 h
02/04/20		Ajouter les empreintes	1h30
06/04/20		Création Liste matériel	1 h
07/04/20		Début du routage finale (perte du fichier)	3 h
14/04/20		Routage finit et ajustement dossier de revue 2	4 h
		TOTAL	46h

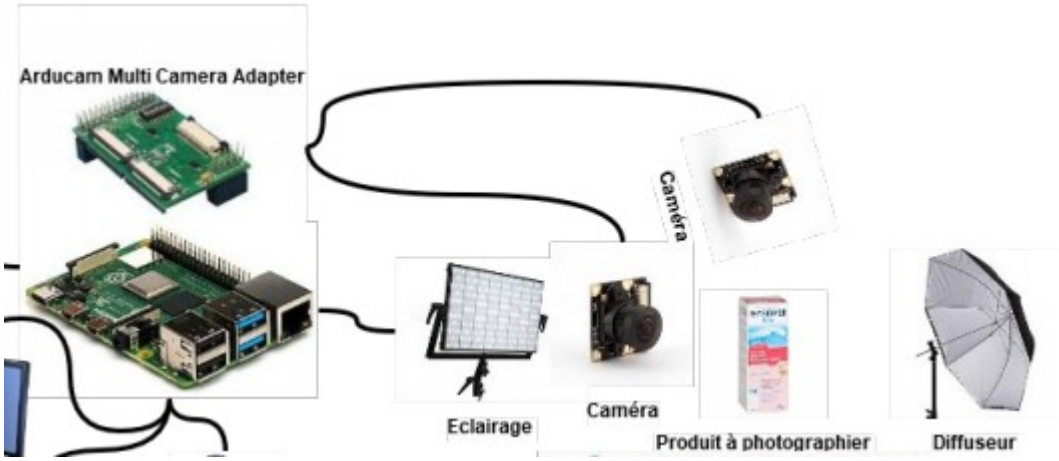
Beaucoup d'heures n'ont pas été comptées.

Partie EC3 : Leroy Lucas

Gestion de l'éclairage et des prises de vues

Présentation :

Mon rôle dans le projet est d'effectuer des recherches et des tests sur les solutions d'éclairage, sur les solutions potentielles d'acquisition de la luminosité afin de pouvoir effectuer au mieux les captures des produits concernés ainsi que d'effectuer un choix dans le type de caméra utilisé pour les captures qui vont être réalisées.



Mes solutions devront pouvoir être pilotées par Raspberry Pi via une interface graphique QT Creator, je travaille étroitement avec IR1 qui s'occupe de cette IHM.

Une IHM avec 3 onglets sera élaborée pour une meilleure prise en main:

- Un onglet pour la partie Éclairage

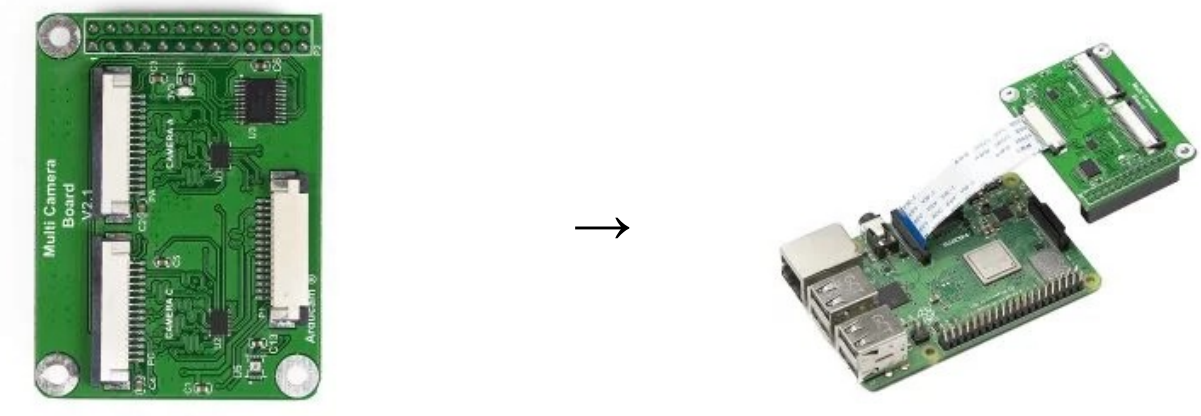
- Un onglet pour la partie Base de Données

- Un onglet pour la partie photo

Avancement du projet :

Choix de la caméra utilisée :

Pour ce qui est du choix de la caméra RPI, avec IR1 nous avons effectué une série de tests comparant 2 types de caméras RPI fournies par l'entreprise Crossdock avec un module Arducam qui vient se brancher sur la Raspberry (ce module permet de connecter jusqu'à 4 caméras RPI).



En ce qui nous concerne tout se passe autour de la commande concernant les prises de vue : **raspistill**
Il existe une commande **raspivid** qui concerne les prises de vidéos mais pour le projet nous n'utiliserons pas cette option.

Voici un exemple basique permettant de prendre une capture au bout de 2 secondes :

raspistill -t 2000 -o image.jpg

→ la valeur après l'instruction « -t » détermine le temps en milliseconde avant la capture (il peut bien évidemment être modifié).
→ après l'instruction « -o » se trouve le nom que l'on donne a la capture qui va être effectuée , ne pas oublier de préciser le format de l'image (.JPG dans le cas présent)
Cette capture va par défaut s'enregistrer sur le bureau.

Il existe différentes librairies pour mettre en œuvre les caméras , celle en C++ sera retenue du fait que l'on utilise QT Creator qui est un outil de développement en C++ .
(la libraire python a été utilisée pour les premiers tests car elle est plus simple d'utilisation).

Liens sources :
https://github.com/ArduCAM/RaspberryPi/tree/master/Multi_Camera_Adapter/Multi_Adapter_Board_4Channel/Multi_Camera_Adapter_V2.1_C%2B%2B
https://www.raspberrypi.org/documentation/raspbian/applications/camera.md

Les tests effectués ont révélé que la Caméra 5MP avec un objectif où l'on peut régler la netteté offre une image de meilleure qualité que la caméra 5MP sans ce réglage.

Caméra Pi 5 MP sans mise au au point manuelle (1)

Caméra Pi 5MP avec mise point manuelle (2)

Les commandes sont les mêmes pour ces 2 caméras.

- La prise de vue de gauche concerne la **caméra Pi 5MP sans mise au point manuelle (1)**
Distance : 20cm / Vue de face
- La prise de vue de droite concerne la **caméra Pi 5MP avec la mise au point manuelle (2)**
Distance : environ 40cm (rendu après un léger zoom) / Vue de face

On observe une qualité assez satisfaisante au niveau de la lisibilité des inscriptions (réglages a affiner dans le futur pour la caméra avec mise au point manuelle pour une prise optimale).

Néanmoins le défaut majeur de la caméra sans mise au point manuelle est ce rendu avec un léger FishEye (Oeil de poisson : la photo semble arrondie) , par élimination sur une plus grande distance , nous avons donc décidé avec IR1 de nous baser sur une structure avec la **caméra Pi 5MP avec la mise au point manuelle (2)**, ses résultats sont plus prometteurs.

Choix de l'éclairage :

Le choix de l'éclairage fût assez laborieux, beaucoup de solutions avaient l'air envisageable mais avec le temps et les recherches, ce panel s'est réduit considérablement.

Parmi les contraintes a respecter , l'éclairage devait :
- Pouvoir être dimmable
- Avoir une lumière blanche de température 6000k (ou très proche)
- Avoir une documentation complète pour analyser tous les paramètres
- Avoir un prix raisonnable et qui rentre dans le cahier des charges
- Avoir un délai de livraison convenable pour effectuer les essais

En premier lieu j'avais donc décidé de lancer des recherches sur les différentes possibilités d'éclairage , ce qui m'avait conduit à 3 solutions :

- DMX via Interface USB → Simple d'utilisation / Compatible RPI / Installation Propre
- Fabrication → Moins Coûteux / Compatible RPI / Encombrement moindre
- Projecteur Dimmable 1/10V → Compatible RPI / Solution via Convertisseur Analogique Numérique

Après quelques heures de recherches supplémentaires , **la solution du projecteur dimmable 1/10V** a été abandonné pour plusieurs raisons :

- Manque d'informations sur certains paramètres tel que l'intensité lumineuse
- Projecteurs dimmables sous conditions (nécessite variateur 85 % du temps)
- Aucun schéma disponible pour éventuelle modification
- Qualité produit ? (Sites inconnus du grand public)

La solution de fabrication semblait donc la plus adaptée a nos besoins avec son prix peu élevé, mais hélas ce sont cette fois les délais de livraisons qui engendraient un problème, environ 5 à 7 semaines selon le site. Un manque de documentation était aussi constaté mais le prix abordable aurait pu nous laisser le droit a l'erreur , des tests auraient pu être effectués.
Cette solution aurait consisté a fabriquer un projecteur White « maison » composé de quelques leds sur un PCB , ces dernières pilotées par une Arduino Nano
Les délais de livraisons trop importants auraient menés a un manque de temps pour le projet.

Par élimination , **la solution DMX via interface USB a été retenue,** cela aura un coût supplémentaire mais permettra de respecter les délais imposés pour les tests et sa simplicité d'utilisation permettra a un utilisateur novice de pouvoir piloter l'éclairage.

J'ai recherché différents éclairages pouvant correspondre a ce que nous cherchons, en voici une liste comparative :

Projecteur	Dimensions	Temperature	Puissance	Angle Diffusion	Canaux	Puissance Lumineuse	Prix
Stairville Wild Wash 132 LED CW	310x70x70mm		132L x 0,2W	75°	1 à 3		79,00 €
Stairville DCL Flat Par 5x4W 2in1 CW/WW	255x235x105mm	2700-6500 K	5L x 4W	25°	2 ou 6	2000 lx (froid)	69,00 €
Eurolite LED SLS-98 Strobe SMD B-Stock	120x215x245mm	5900 K	X	X	4 à 9		82,00 €
Stairville Mini Stage Par CW/WW/A	200x140x260mm	2800-6000K	7L x 6W	30°	4 à 5		89,00 €
Cameo Flat PAR Can 1 TW IR	175x180x115mm	3400-7000K	7L x 4W	25°	1 à 7	6000 lx	85,00 €
Stairville LF-6 LED Flash 6 COB Strobe	225x170x118mm		6L x 5W	71°	1 à 3		79,00 €
Cameo Thunder Wash 100W	310x70x70mm		132L x 0,2W	80°	1 à 3		99,00 €

Parmi toutes ces solutions, le **CAMEO FLAT PAR CAN 1 TW IR** a été retenu comme solution la plus élaboré pour éclairer l'objet car :
- il dispose de toutes les informations dont nous avons besoin,
- une puissance lumineuse très satisfaisante,
- une plage de température de couleur qui remplit les critères
- une documentation accessible et complète sur le site de Thomann.fr

Quant au **Starville Wild Wash 132 LED CW**, nous avons décider de retenir cette solution comme secondaire pour effectuer l'éclairement de l'arrière plan

Pour ce second choix je me suis penché sur l'angle de diffusion , qui est 3 fois supérieur que celui du projecteur Cameo choisi.

Ces 2 projecteurs seront donc utilisés lors du projet si la solution DMX est celle retenue :
Le **Cameo** pour éclairer l'objet et le **Stairville** pour éclairer le fond

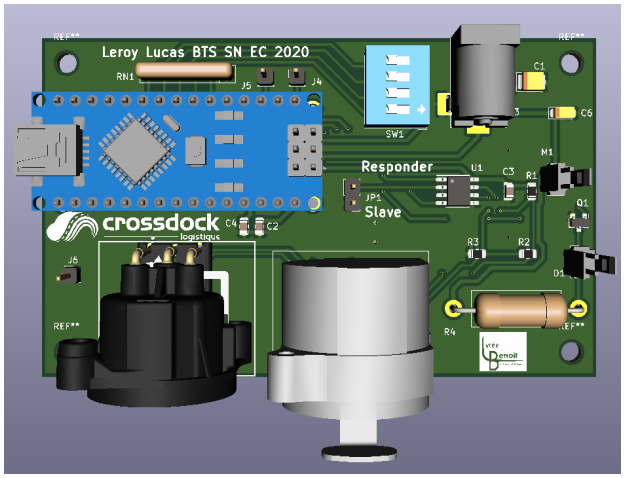
Des tests seront effectués lors de la réception du matériel.

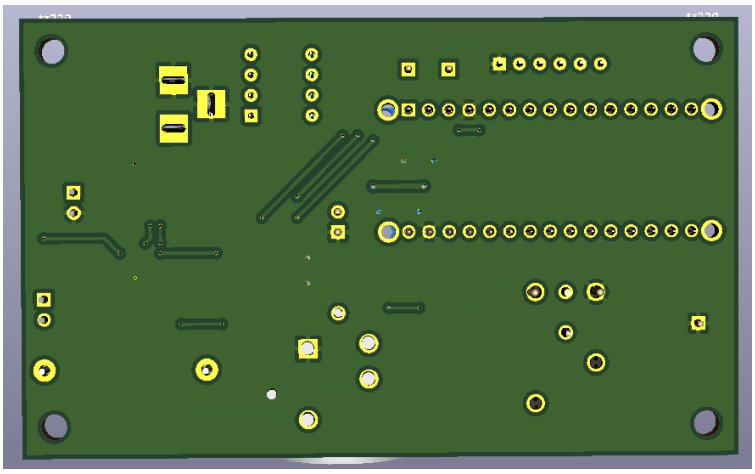
CAMEO FLAT PAR CAN 1 TW IR

STAIRVILLE WILD WASH 132 LED CW

Une nouvelle solution imposée a émergée , celle d'un projecteur DMX fabriqué , composé d'une LED Blanche de 10W pilotée par une Arduino Nano.
Tout ceci sur le même PCB avec les 2 connecteurs XLR afin de pouvoir communiquer en DMX via une interface USB Enttec.

Cette solution a été imposée car dans le cas contraire, je n'aurai pas eu de PCB a réaliser, ce qui n'est pas le but.





Voici un aperçu de ma carte ci dessus , elle remplacera donc l'ensemble de cartes qui a été testé , l'éclairage sera relié a un ventilateur afin de ne pas créer de surchauffe dans le futur boîtier de la carte. Le prototype est là , il est susceptible de changer (certaines empreintes vont changer).

Partie DMX_RDM panneau LED:

Voici le programme test utilisé pour l'exemple d'un petit panneau Leds , connecté a un module RDM lui même relié a une carte Arduino Uno.

Cette base peut nous servir d'exemple pour le futur projecteur , le programme devra bien évidemment être adapté a l'Arduino nano qui sera utilisé

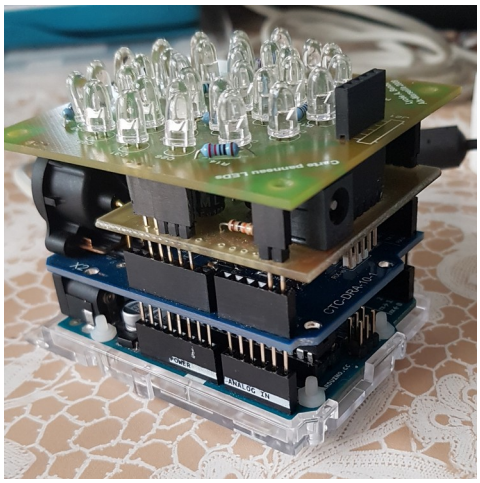
La classe **Conceptinetics.h** permet d'obtenir les commandes pour que l'Arduino se comporte comme un esclave

DMX_Panneau_Led §

```
#include <Conceptinetics.h>

// On configure 10 cannaux , la plage des cannaux s'étendra donc de 1 à 11
#define DMX_SLAVE_CHANNELS   10


// Configure a DMX slave controller
DMX_Slave dmx_slave ( DMX_SLAVE_CHANNELS );

// On initialise le GPIO correspondant aux leds et ses couleurs
const int ledPin_Rouge = 9;
const int ledPin_Vert = 10;
const int ledPin_Bleu = 11;

void setup() {
 dmx_slave.enable ();
 dmx_slave.setStartAddress (1);

 pinMode ( ledPin_Rouge, OUTPUT );
 pinMode ( ledPin_Vert, OUTPUT );
 pinMode ( ledPin_Bleu, OUTPUT );
}
void loop()
{
  // La valeur des cannaux 1 / 2 / 3 est communiqué au GPIO
  // des leds provoquant un éclairement plus ou moins intense
 analogWrite(ledPin_Rouge, dmx_slave.getChannelValue (1));
 analogWrite(ledPin_Vert, dmx_slave.getChannelValue (2));
 analogWrite(ledPin_Bleu, dmx_slave.getChannelValue (3));
}
```

On teste ici un panneau led avec 3 types de LEDs , rouges , vertes et bleues

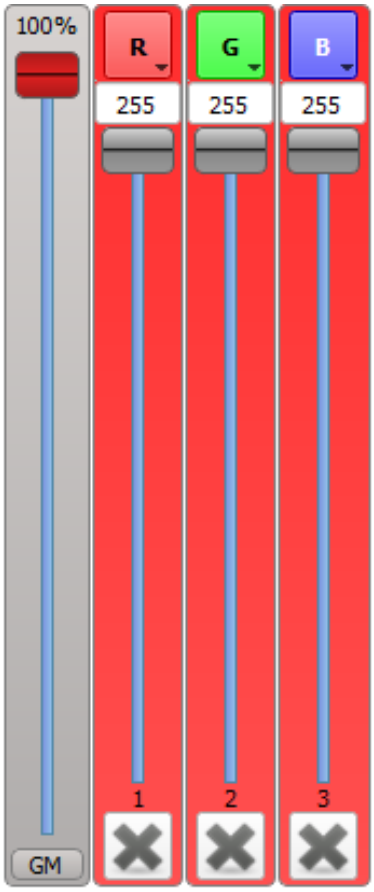
Voici a quoi ressemble la carte où les tests sont effectués , pour ma partie , mon but est de créer un équivalent plus ergonomique et en modèle réduit.



État de base

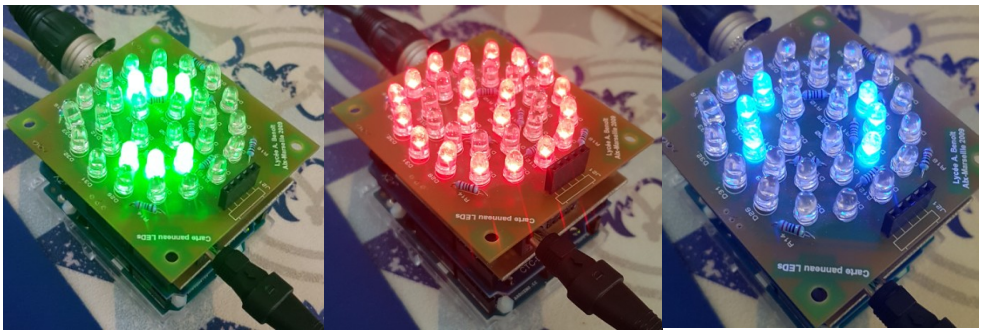
État lorsque les 3 canaux sont a 255

Ce programme permet tout simplement d'éclairer les LEDs en fonction de l'état des canaux.



Voici comment se présente le réglage de ces canaux sur le logiciel QLC+ (cette photo représente la carte avec toutes les LEDs en fonctionnement

Et voici les différents états obtenus lorsque l'on sélectionne 1 seul de ces canaux :



Selon le nombre binaire que l'on sélectionne (entre 0 et 255) , l'intensité lumineuse augmente , le programme joue donc bien son rôle et renvoie bien cette valeur binaire a notre carte.

Les essais sont assez satisfaisants, je pense que je pourrais m'inspirer de ce programme pour ma nouvelle carte , cependant je ne mettrais pas 10 canaux mais 1 seul car la couleur sera fixe , seule l'intensité lumineuse nous intéresse.

En voici un exemple sous la forme de conditions , un programme qui est testé avec 3 Leds

DMX_LED_RGB §

```
#include <Conceptinetics.h>

#define DMX_SLAVE_CHANNELS   10

// Configure a DMX slave controller
DMX_Slave dmx_slave ( DMX_SLAVE_CHANNELS );

const int ledPin1 = 9;
const int ledPin2 = 10;

void setup() {
  dmx_slave.enable ();
  dmx_slave.setStartAddress (7);
    pinMode ( ledPin1, OUTPUT );
    pinMode ( ledPin2, OUTPUT );
}
void loop()
{
  if ( dmx_slave.getChannelValue (1) > 100 ) digitalWrite ( ledPin1, HIGH );
    else digitalWrite ( ledPin1, LOW );
  if ( dmx_slave.getChannelValue (4) < 200 ) digitalWrite ( ledPin2, HIGH );
    else digitalWrite ( ledPin2, LOW );
}
```

Ce programme consiste a instaurer un éclairement sous condition , pour vérifier le bon fonctionnement de l'application QLC + avec notre module RDM , après essais ,
la communication se fait correctement car le résultat est bien cohérent avec le programme au canal près (pas de marge d'erreur)

Pourquoi opter pour cette solution si tard ? Nous nous sommes rendu compte que le capteur de luminosité ne nécessitait pas d'être câblé sur le PCB , son montage sur breakout est suffisant pour l'exploitation voulue, cependant il n'y avait plus de PCB a fabriquer pour ma partie, avec Mr Hortolland nous nous sommes donc penché sur une solution de fabrication ou nous avons trouvé des composants qui peuvent être expédiés rapidement contrairement a la première solution de fabrication énoncée.

En revanche les 2 projecteurs DMX dont nous disposons désormais peuvent toujours servir a l'éclairage de l'arrière plan mais ce n'est pas définitif , des tests sont encore a effectuer afin d'affiner la qualité de l'image que l'on souhaiterait avoir.

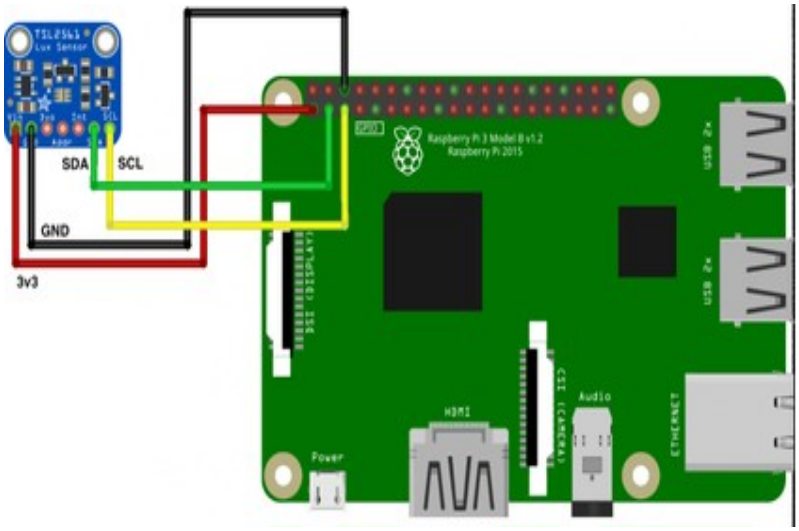
Choix du capteur de luminosité :

Le capteur de luminosité ne fût pas le choix le plus difficile , les contraintes étaient moindre , il suffisait que ce dernier communique en I2C (idéalement) et qu'il soit compatible avec le 3,3V de la Raspberry Pi.

	TSL2561	TMG39931	DFR0026	SEN0172
ALIMENTATION	2,7 V - 3,6 V	2,4 V - 3,6 V	3,3 V - 5 V	3 V - 5,5 V
PROTOCOLE DE SORTIE	I2C	I2C	SORTIE ANALOGIQUE	SORTIE ANALOGIQUE
DIGITAL OUTPUT	7 BITS	2 x 8-BIT → 16	X	X
SDA / SCL VIL	- 0,5 V / 0,8 V	NON PRECISE / 0,54 V	X	X
SDA / SCL VIH	2,1 V / 3,6 V	1,26 V / NON PRECISE	X	X
VENDU	CÂBLE SUR BREAKOUT	CÂBLE SUR BREAKOUT	CÂBLE SUR BREAKOUT	SUR BREAKOUT
LONGUEUR D'ONDE MESURABLE	500nm - 1000nm	465nm - 615nm	X	X
PLAGE DE MESURE	0,1 - 40 000 LUX	0,1 - 100 000 LUX ? (a vérifier)	1 - 6000 LUX	0 - 800 LUX
FOUNISSEUR			DFROBOT	DFROBOT

On souhaite tout de même un capteur de luminosité avec une plage de mesure pouvant comprendre la puissance lumineuse de l'éclairage choisi , dans le cas présent le choix de l'éclairage n'était pas encore définitif , j'ai donc choisi d'opter pour le TSL2561 pour sa plage de mesure , sa compatibilité avec le 3,3V de la RPI , un protocole I2C en sortie et surtout , il était déjà câblé sur Breakout, cela facilitera les tests lorsque le capteur aura été livré.

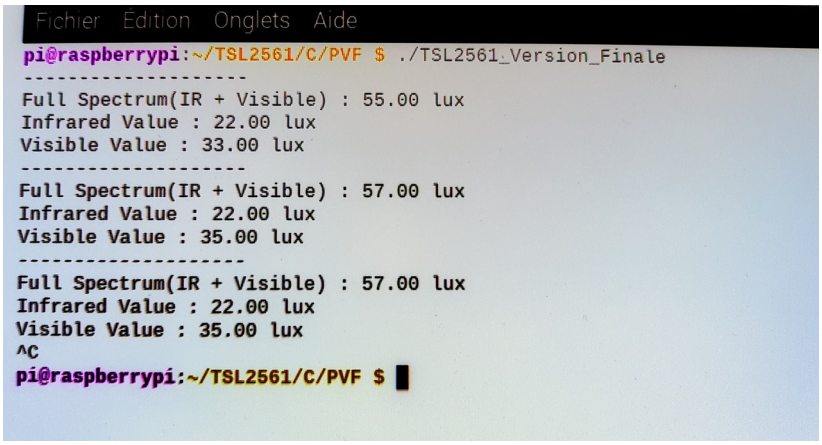
Test du capteur de luminosité TSL2561 :



Le câblage test est le suivant :

- Broche Vin reliée au 3,3V de la RPI
- Broche GND connectée au GND de la RPI
- Broche SDA reliée au SDA de la RPI
- Broche SCL reliée au SCL de la RPI

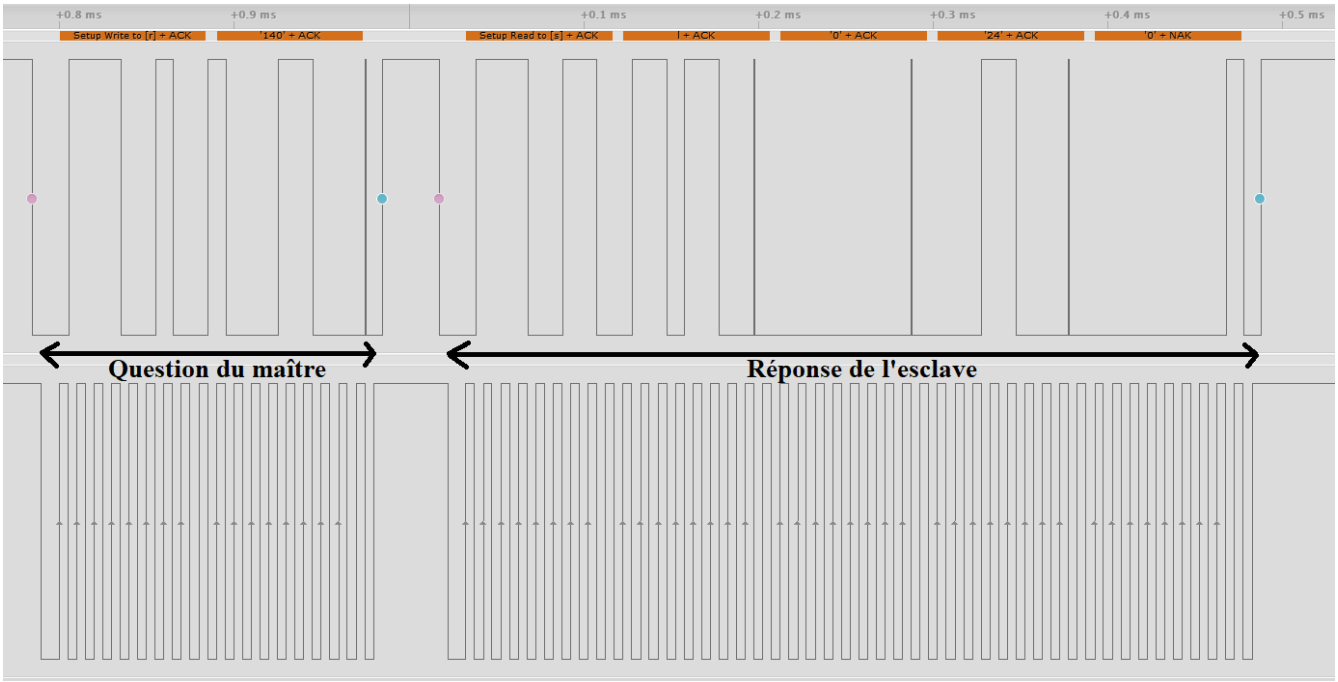
Pour tout test a effectuer sur le bus I2C , ne pas oublier d'activer ce dernier sur la Raspberry Pi :
Pour se faire : Framboise → Préférences Systèmes → Interfaces → Activer I2C
Une fois l'I2C activé , j'utilise la commande : **i2cdetect -y 1** pour m'assurer que mon capteur est bien détecté. L'adresse de base du capteur est 0x39.



Voici le résultat obtenu lorsque l'on utilise un programme en C pour faire l'acquisition des valeurs en lux.

Seules les valeurs « **Full Spectrum** » et « **Infrared Value** » sont mesurées par le capteur, la valeur « **Visible Value** » est seulement une déduction, le résultat correspond à une soustraction entre le Full Spectrum et l'Infrared Value.

Ici le programme d'origine a été modifié , j'ai rajouté une boucle « **while** » et un « **sleep** » afin que le programme se relance , toutes les 3 secondes (le délai est nécessaire sinon l'interprétation des résultats sera impossible lorsque l'utilisateur souhaitera régler l'intensité de son éclairage.
(Taux de rafraîchissement initial du capteur : 50 ms).



Ci dessus une trame I2C où l'on distingue l'instruction du maître (RPI) ainsi que la réponse de l'esclave (TSL2561).

Voici ce qui est écrit au dessus de la trame SDA (Trame du dessus, celle du dessous correspond a la trame SCL , l'horloge).

! : La mesure analysée ci dessous ne correspond pas aux valeurs affichées dans le terminal ci dessus , elle correspond ici a une intensité lumineuse relevée de 108lux dont 24 lux IR

Comment lire cette trame ? Tout simplement en analysant octet par octet les données :
<u>On commence par lire les 2 octets envoyées par le maître :</u>

« **Setup Write to [0x72] + ACK** » : On initialise l'écriture d'une instruction
L'esclave répond avec un bit d'acquittement

« **0x8C + ACK** » : 0x8C correspond a l'instruction Read dans la documentation, on souhaite lire une information / une donnée.
L'esclave répond avec un bit d'acquittement

```
Command = 0x8C                          //Address the Ch0 lower data register
ReadByte (Address, Command, DataLow)    //Result returned in DataLow
```

Les 2 octets envoyés par le maître signifient donc que le maître écrit dans le TSL2561 une requête de lecture, l'esclave lui, confirme la réception avec un bit d'acquittement a la réception de chaque octet.

On passe maintenant a la lecture des 5 octets concernant l'esclave où cette fois ci, le maître émet un bit d'acquittement a chaque octet pour confirmer la réception des données (hors dernier octet):

« **Setup Read to [0x73] + ACK** » : On initialise la lecture d'une donnée
Le maître répond avec un bit d'acquittement

Les 2 octets qui suivent concernent la donnée **Full Spectrum :**

« **0x6C + ACK**» : L'esclave renvoie la valeur de la donnée **Full Spectrum** qui est « 108 » lux
« **0x00 + ACK** » : La valeur suivante correspond a un nombre entre 0 et 255 que l'on multiplie ensuite par 256 , cette valeur restera a 0x00 si le **Full Spectrum** ne dépasse pas 255 lux

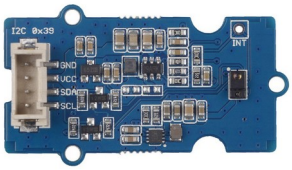
Les 2 octets qui suivent concernent désormais la valeur **Infrared Value :**

« 0x18 + ACK : L'esclave renvoie la valeur de la donnée **Infrared Value** qui est « 24 » lux
« 0x00 + NAK : De même que pour la valeur **Full Spectrum**, la valeur **Infrared Value** correspond a un nombre entre 0 et 255 que l'on multiplie ensuite par 256, sa valeur est aussi de 0 car **Infrared Value** vaut 24lux.

Le dernier octet lui , ne dispose pas de bit d'acquittement, d'où l'acronyme NAK.

Test TMG39931

Un second capteur de luminosité a été testé , il s'agit du TMG39931 câblé aussi sur breakout , les tests étaient moyennement satisfaisant , on ne comprenait pas toujours le programme et ses 5 valeurs renvoyées :



Red color ; ...lux **Blue color ; ...lux** **Green color ; ...lux** **IR ; ...lux** + une valeur estimant la distance du capteur mais on ne sait pas par rapport a quoi .
En conclusion ce capteur est + optimisé pour si l'on souhaite un éclairage dominant d'une couleur en particulier car l'on peu en effet surveiller l'intensité lumineuse de chaque couleur et ainsi ajuster les seuils a notre guise.

Un second point négatif est le schéma de câblage , qui est assez imposant comparé au TSL2561, il y a énormément de modifications de tension avec ce capteur , on peut supposer que c'est le fait qu'il puisse acquérir les couleurs qui requiert autant de composants.

Schéma du TMG39931

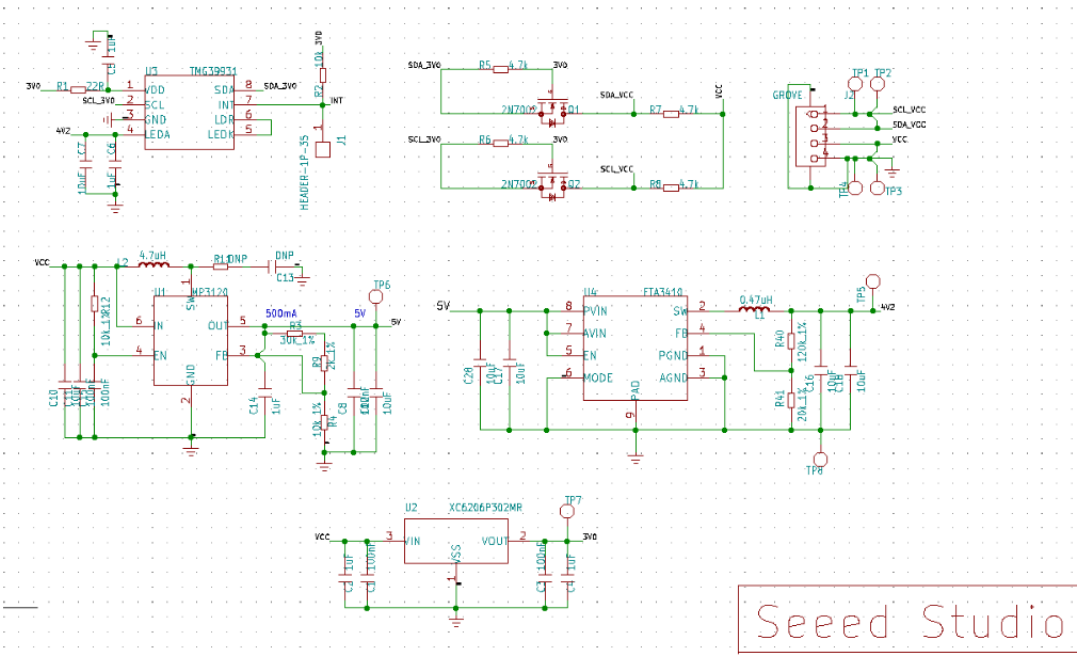
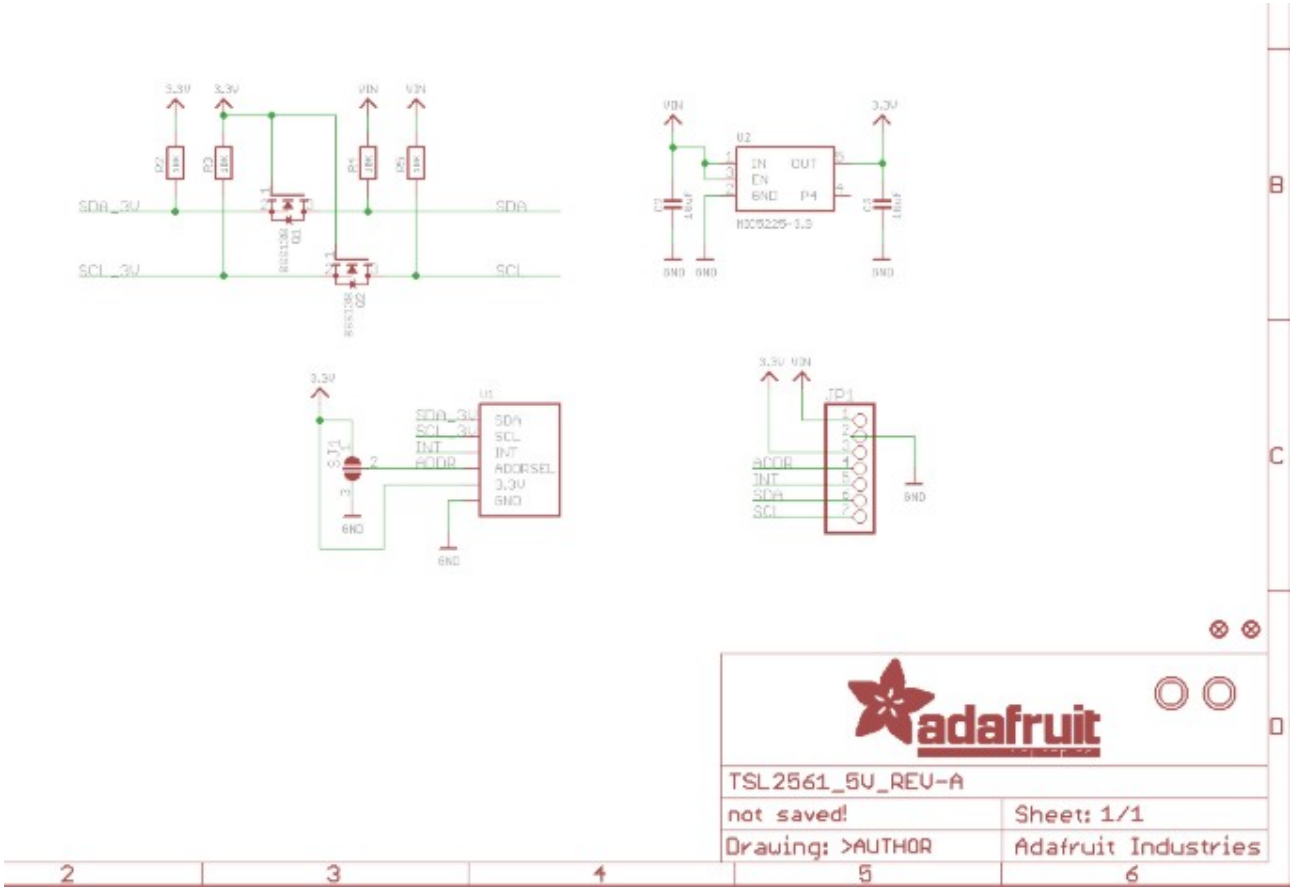


Schéma du TSL2561



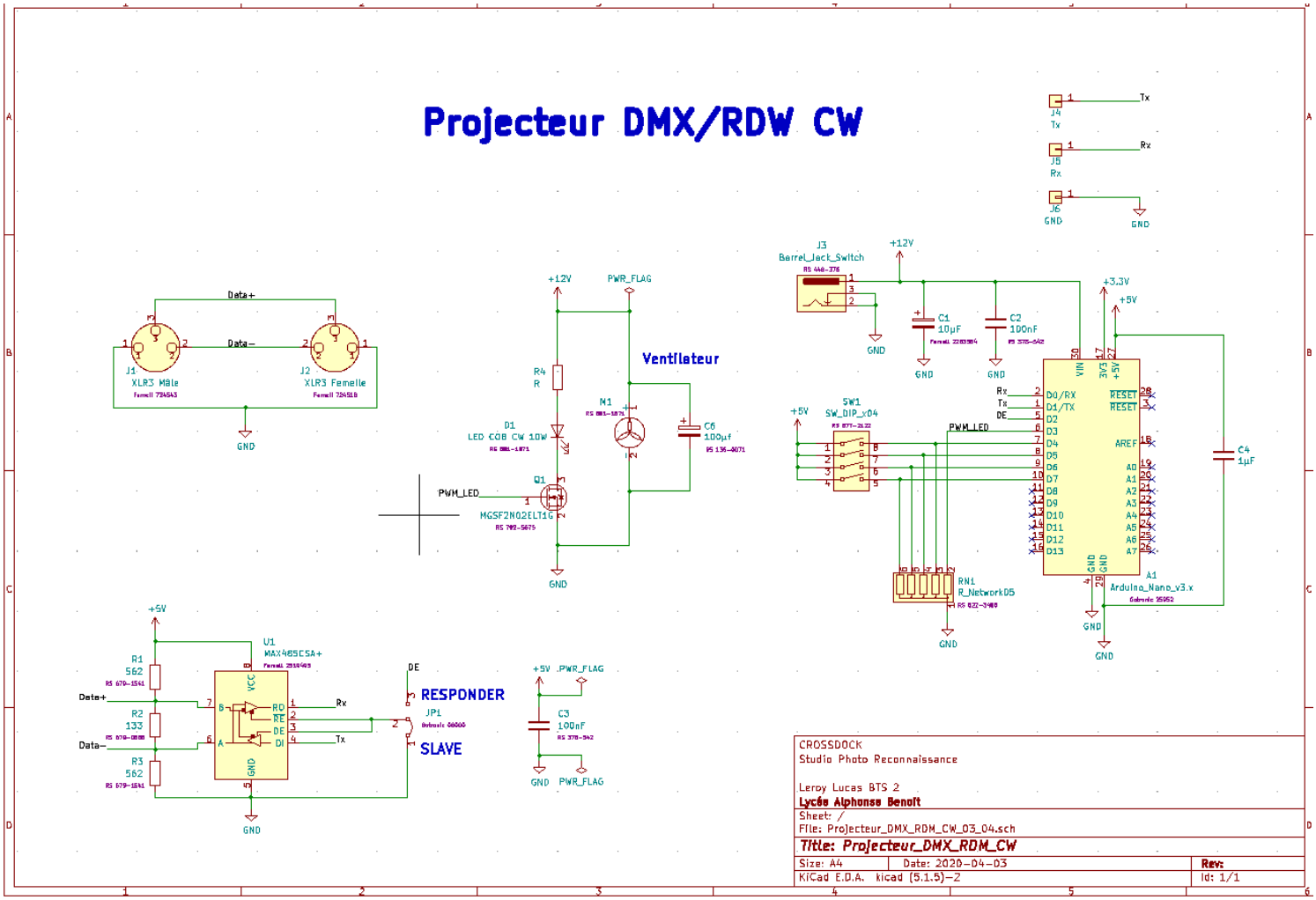
Je garde donc la solution précédente du TSL2561 qui lui proposait des valeurs compréhensibles pour l'utilisateur , un nouveau programme a été écrit avec la librairie BCM2835 (spécialement dédié pour l'I2C), un programme beaucoup plus épuré et commenté , 1 seule valeur est renvoyée , celle de l'intensité lumineuse totale.
C'est exactement ce que l'on recherche pour notre Interface graphique Qt afin de pouvoir surveiller et modifier cette intensité en temps réel lors des réglages pour la prise de vue.

Partie Kicad :

Pour la réalisation de la carte , nous avons utilisé le logiciel Kicad , qui permet de pouvoir créer un schéma de câblage et d'effectuer un routage sur ce dernier , avec la possibilité d'avoir un aperçu 3D.

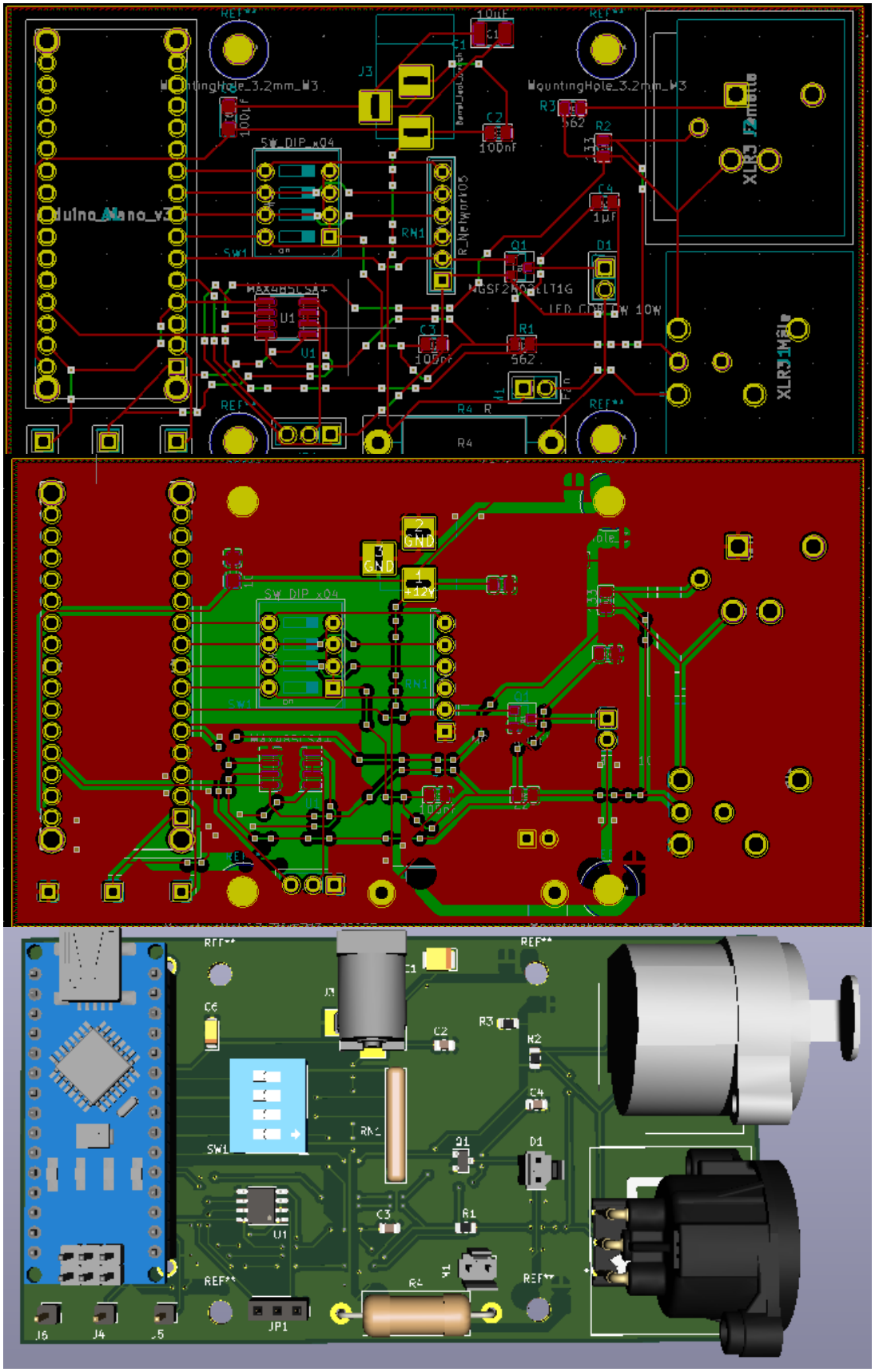
Le schéma a été dessiné puis complété , j'ai fait émerger 2 idées pour le routage avec différents placements des composants et des trous .

Voici le **schéma de câblage** définitif de la carte Projecteur DMX/RDM CW :

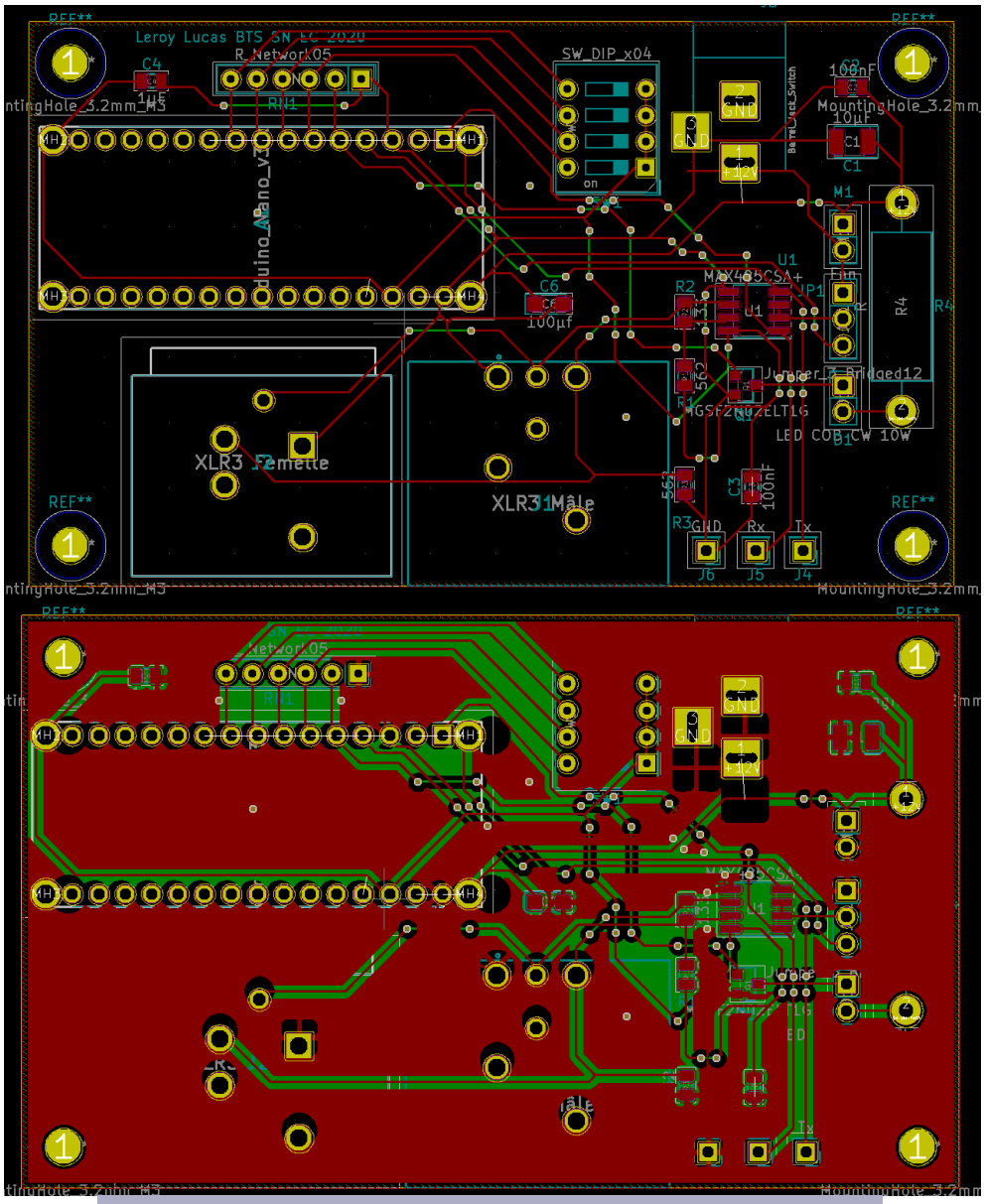


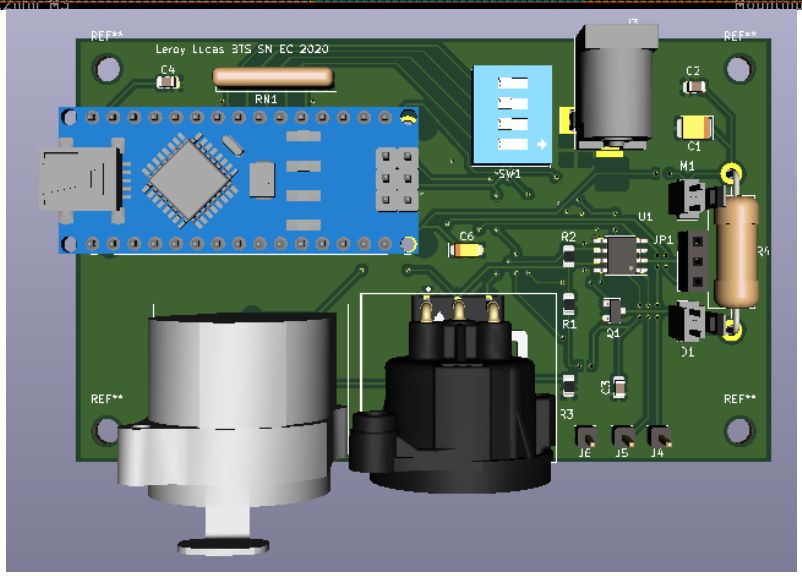
J'ai proposé 2 solutions brouillon de routage pour le PCB (routage/plan de masse /rendu 3d), afin d'effectuer une comparaison rapide des avantages et des inconvénients pouvant être rencontrés lors du routage final.

Solution routage 1 (trous au centre de la carte):



Solution routage 2 (trous aux 4 coins de la carte) :





La solution 2 sera retenue en vue du placements des trous et de la facilité de routage comparé a la première solution , le plan de masse est lui aussi « mieux » réalisé dans le 2eme cas, il nécessitera moins de modifications que le 1er cas.

L'inconvénient majeur sur la **solution 1** est un vide énorme du plan de masse « TOP » au niveau du SW_DIP_04 , il serait assez complexe d'arranger la situation sans trouver une autre solution de câblage , c'est donc pour cela que j'ai opté pour la **solution 2** , qui a première vue a un plan de masse mieux réparti sur la carte

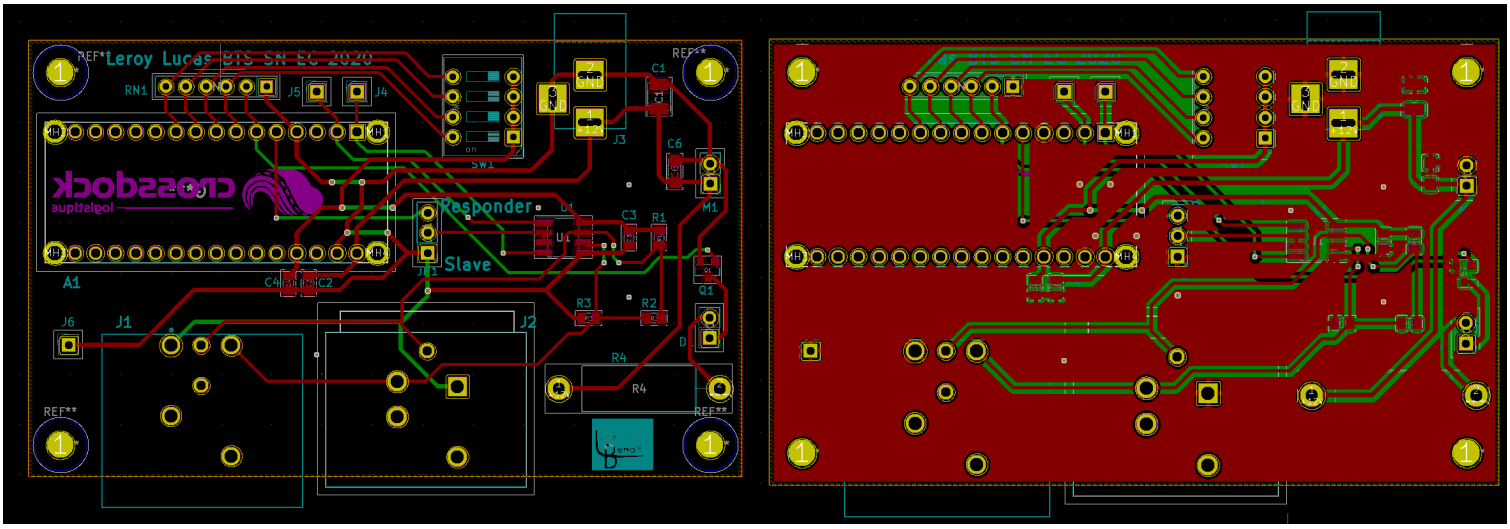
Attention ! Ces 2 schémas de routage ne représentent pas tout a fait la réalité , étant tout deux des brouillons , je n'ai pas tenu compte de la largeur des pistes imposées pour l'alimentation , qui sont normalement plus large ; des empreintes pourront éventuellement être changées au dernier moment.

Les composants sont placés de façon ergonomique , notamment les connecteurs d'alimentation et les points de tests , afin de grouper les fils lorsque que la carte sera câblé en réel.

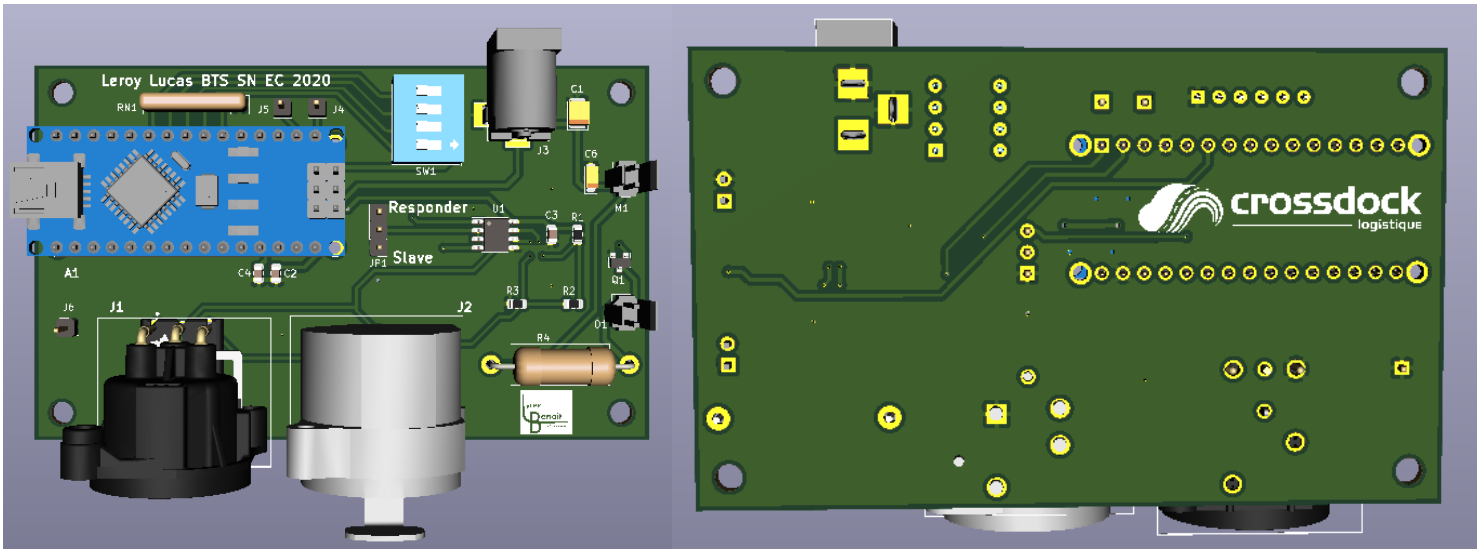
Les fichiers gerber devront être généré une fois le brouillon du routage mit au propre , ainsi que la liste du matériel. Les dernières empreintes ont été rééditées, cependant elles ne changent en aucun cas le positionnement des composants sur la carte.

Voici maintenant le routage définitif de la carte , avec les dernières modifications d'empreintes et ajout des logos du lycée + de l'entreprise :

Schéma de routage final Projecteur_DMX_RDM_CW :



A gauche le routage de la carte, a droite le plan de masse



Coté top et bottom en rendu 3D

Le placement des composants a été édité , les pistes ont été retravaillées a la toute fin afin d'avoir un plan de masse complet , les connecteurs XLR et le connecteur d'alimentation ont été légèrement sorti de la carte afin de pouvoir se marier avec le futur boîtier tout en ayant aucune difficulté pour brancher les câbles.

Le ventilateur et la LED sont représentés sous forme de connecteur car l'on souhaite en réel fixer la LED sur le dos du ventilateur , afin d'éviter toute surchauffe (inenvisageable si les 2 composants se situent sur la carte) , puis éventuellement si l'on souhaite changer de LED pour plus ou moins de puissance , cela reste a vérifier lors de futurs tests après réception de la carte et après l'association des différentes parties tels que le plateau tournant et l'arrière plan .

Mon but principal était de pouvoir créer un éclairage piloté par un programme Arduino , ce qui sera possible avec cette carte , il faudra simplement la relier a une interface DMX USB pour la mettre en œuvre.

Moins imposante et plus ergonomique que son ancienne version , la nécessité de déconnecter 2 cartes pour changer le cavalier de position ne sera plus indispensable , il est possible désormais de le changer a tout moment , sans avoir a débrancher l'alimentation et permet de gagner du temps.

Planning prévisionnel du 06/01/2020 au 09/04/2020

	Date	Jour	Tâche	Durée	Prédécesseur
2	06/01/20	Lundi	Briefing du projet / Premier Tests Caméra (Urgent)	3 h	
3	07/01/20	Mardi	Analyse des commandes Caméra Pi	3 h	2
4	09/01/20	Jeudi	Etude du module Arducam / Choix de la caméra	4 h	3
5	13/01/20	Lundi	Premier tests des éclairages fourni	3 h	
6	14/01/20	Mardi	Analyse des potentielles contraintes d'éclairage	3 h	5
7	16/01/20	Jeudi	Solutions proposées	4 h	6
8	20/01/20	Lundi	Etude des capteurs de luminosité proposés	3 h	
9	21/01/20	Mardi	Choix du capteur et recherche de documentation	3 h	8
10	23/01/20	Jeudi	Rédaction du dossier de projet	4 h	
11	27/01/20	Lundi	Test de l'éclairage	3 h	7
12	28/01/20	Mardi	Recherche de solution pour communication RPI	3 h	11
13	30/01/20	Jeudi	Premier tests TSL2561	4 h	9
14	03/02/20	Lundi	Elaboration d'un programme C pour le TSL2561	3 h	13
15	04/02/20	Mardi	Analyse de trames I2C du TSL2561	3 h	
16	06/02/20	Jeudi	Rédaction du dossier de projet	4 h	10
17	02/03/20	Lundi	Essai des nouveaux éclairages DMX	3 h	
18	03/03/20	Mardi	Essai du capteur TMG39931	3 h	
19	05/03/20	Jeudi	Elaboration d'un programme BCM2835 pour TSL2561	4 h	18
20	09/03/20	Lundi	Création du schéma du câblage Kicad	3 h	
21	10/03/20	Mardi	Importation des nouvelles librairies Kicad	3 h	20
22	12/03/20	Jeudi	Rédaction du dossier de projet	4 h	16
23	16/03/20	Lundi	Etude du TP DMX_RDM sur Arduino	3 h	
24	17/03/20	Mardi	Interprétation des programmes Arduino	3 h	23
25	19/03/20	Jeudi	Finalisation schéma de câblage Kicad	4 h	21
26	23/03/20	Lundi	Assignation des empreintes + vérifications rendu 3D	3 h	25
27	24/03/20	Mardi	Premier routage brouillon PCB	3 h	26
28	26/03/20	Jeudi	Deuxième routage brouillon PCB	4 h	27
29	30/03/20	Lundi	Rédaction du dossier de projet	3 h	22
30	31/03/20	Mardi	Edition des routages brouillon pour comparaison et choix du PCB	3 h	28
31	02/04/20	Jeudi	Routage Final du PCB	4 h	30
32	06/04/20	Lundi	Edition d'empreintes + pistes sur routage final	3 h	31
33	07/04/20	Mardi	Création liste du matériel + fichiers Gerber	3 h	32
34	09/04/20	Jeudi	Rédaction du dossier de projet	4 h	29
35			TOTAL	110 h	

La liste des tâches est réalisé en fonction des séances sur l'emploi du temps , soit environ 10h / semaine.

Liste des tâches du 06/01/2020 au 07/04/2020 (Réel)

Date	[redacted]	Tâche	Durée
06/01/20		Briefing de départ + recherche documentation caméra PI	3 h
07/01/20		Essai caméras Pi type 1 et 2	4 h
13/01/20		Test Shield Arducam	3 h
14/01/20		Recherche de solution pour éclairage	10 h
21/01/20		Comparaison et choix capteur de luminosité	3 h
27/01/20		Réunion + élaboration de nouveaux objectifs	3 h
28/01/20		Recherche de schéma de câblage rapide sur Fritzing	4 h
03/02/20		Essai du TSL2561	7 h
06/02/20		Rédaction du dossier de projet	3 h
02/03/20		Essai des nouveaux éclairages DMX	3 h
03/03/20		Essai du capteur TMG39931	3 h
05/03/20		Elaboration d'un programme BCM2835 pour TSL2561	4 h
09/03/20		Création du schéma de câblage Kicad + finitions	10h
16/03/20		Etude du TP DMX/RDM sur Arduino + Interprétation des programmes	6h
19/03/20		Rédaction du dossier de projet	4h
23/03/20		Routage des PCB brouillons	10h
30/03/20		Routage du PCB final + edition d'empreintes et de pistes	10h
06/04/20		Création Liste matériel + fichiers Gerber	2 h
07/04/20		Rédaction du dossier de projet	3 h
[redacted]		TOTAL	95 h

(Revue 1)
2 séances ont été annulées ce qui a instauré un retard de 7h dans le projet , en plus d'un délai rallongé de livraison des éclairages.
Un retard conséquent s'est installé lors de la recherche de solutions pour l'éclairage car de nombreuses contraintes m'ont poussé a approfondir mes recherches sur un éclairage pouvant être livré dans les temps, en plus de toutes les contraintes déjà énoncés.

En revanche les essais sur le capteur de luminosité se sont bien passé , cela m'a permis de gagner environ 1 à 2h que j'ai pu consacrer a la rédaction de mon dossier.

(Revue 2)
Le confinement a eu un effet négatif sur le projet , en effet un travail étroit avec IR1 devait être effectué et n'a pas pu aboutir pour l'intégration du capteur TSL2561 à l'IHM, un ralentissement de la progression est constaté avec une perte de 8h supplémentaire soit 15h au total.

But pour la prochaine revue :

Pour la prochaine revue , mon but sera d'affiner mes programmes afin de les rendre le plus compréhensible possible , créer un programme permettant la communication DMX_RDM sur l'Arduino nano , les essais effectué sur l'Arduino Uno sont satisfaisant.

Trouver avec IR1 une solution afin de pouvoir intégrer ma partie a l'IHM :
- Projecteur
- Capteur de luminosité

Trouver un agencement convenable des divers équipements pour que la structure soit le moins encombrante possible (une esquisse provisoire pourrait-être réalisée)

Créer des fiches techniques afin que les autres membres du projet puissent utiliser facilement ma partie lors de la revue finale.