

# Rapport de laboratoire

---

**Ecole supérieure**

Électronique  
PROJ  
SLO2

---

## Phase de Design Affichage Matriciel

N° 2126

---

**Réalisé par :**

Ricardo Crespo

**A l'attention de :**

Professeur M. Castoldi  
Professeur M. Moreno

**Dates :**

Début du laboratoire : 15 décembre 2022  
Fin du laboratoire : 2 février 2022

# Table des matières

1.	Description du produit .....	3
2.	Principaux choix effectuées .....	3
3.	Choix effectués et dimensionnement.....	4
3.1.	USB .....	4
3.2.	Alimentation .....	5
3.2.1.	5V .....	5
3.2.2.	3.3V .....	5
3.3.	Microcontrôleur .....	6
3.3.1.	Communication .....	6
3.3.2.	Quartz .....	7
3.3.3.	Reset .....	8
3.3.4.	JTAG.....	8
3.3.5.	Monitoring .....	8
3.3.6.	LED de vie .....	9
3.3.7.	Découplage.....	9
3.4.	Multiplexeur.....	10
3.5.	Matrice à LEDs.....	11
3.6.	Liaison inter-carte.....	12
4.	Concept software .....	13
5.	Concept firmware .....	14
6.	Conclusion .....	15
7.	Références.....	16
8.	Annexes .....	16

## 1. Description du produit

Lors de la rentrée des premières années à l'ETML-ES, ils doivent écrire leur nom sur une feuille de papier pour que les enseignants puissent retenir leurs noms. Afin de simplifier cette étape et de la normaliser, un affichage matriciel à installer derrière les écrans des PCs, permettra d'afficher le nom lié au profil utilisateur de l'étudiant logué.

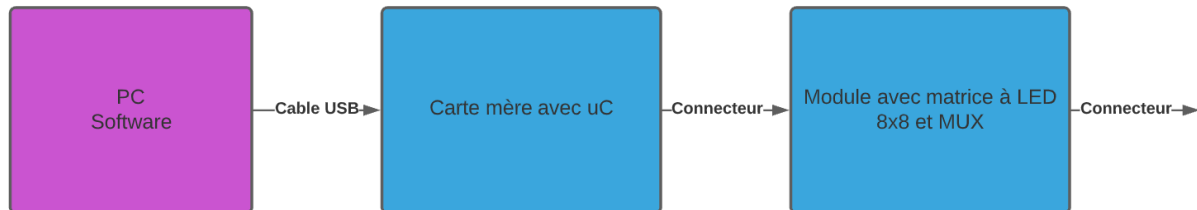


Figure 1 Schéma bloc du système complet

Vous trouverez plus de détails sur le schéma bloc dans la pré-étude et dans le CDC.

Globalement on est sur deux cartes, une avec le microcontrôleur qui récupère les informations qui viennent via l'USB du PC. Puis les deuxièmes types de cartes, sont les modules avec les matrices à LEDs.

## 2. Principaux choix effectués

Suite aux retours de la pré-étude, je suis passé sur les familles MX, car on peut également dédier une partie de la mémoire comme EEPROM.

J'aurais besoin d'une communication UART, et d'une communication SPI. Je n'aurais donc pas besoins de beaucoup de ports pour faire fonctionner le tour. C'est pourquoi j'ai pris un microcontrôleur adapté à mon usage. Suite à la vérification des disponibilités en stock, j'ai opté pour le microcontrôleur « PIC32MX130F064B » de 28 pins.

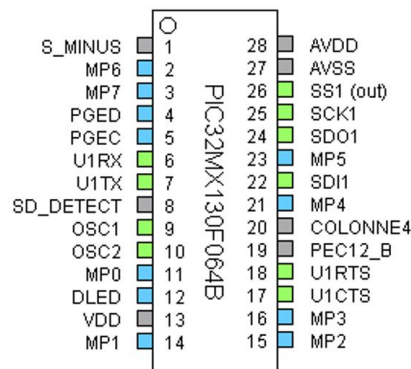


Figure 2 Microcontrôleur PIC32MX130F064B

Le reste des choix pour les autres composants seront détaillées dans leurs sections dédiées.

Prenez en compte les valeurs de tension des condensateurs, comme une valeurs minimum.

## 3.Choix effectués et dimensionnement

### 3.1. USB

Pour permettre de faire la communication entre le pc et ma carte, je vais utiliser une communication USB. C'est également via ce lien que je vais pouvoir alimenter tout le dispositif, et je n'aurais donc pas à mettre une alimentation externe, et donc respecter le cahier des charges.

Le microcontrôleur pic32 utilisé ne possède pas de module pour la communication USB. C'est pourquoi pour les projets le choix de prendre un convertisseur USB to UART a été fait.

Pour cela nous avons un modèle de prédilection à l'ES, le « FT230XS », mais malheureusement il n'est plus disponible dans le monde actuellement. C'est pourquoi j'ai cherché une alternative, et prenant le circuit « CY7C64225 ».

Je me suis basé sur le schéma proposé par le fabricant, dans la configuration avec le microcontrôleur alimenté avec un régulateur externe. Dans mon cas il sera alimenté en 3.3V, comme ci-dessous, et donc les sorties UART seront également en 3.3V

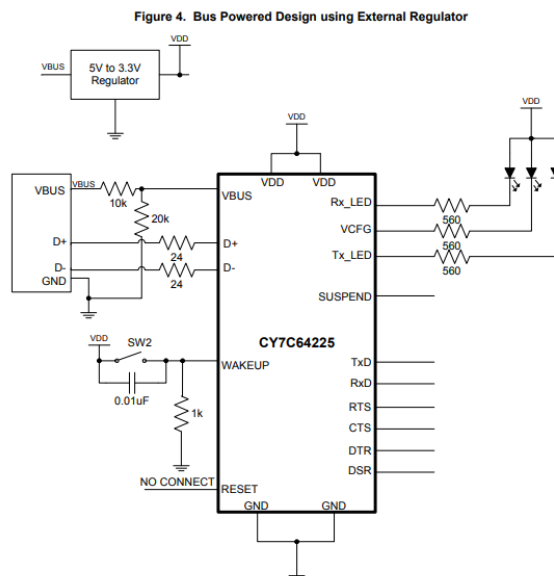


Figure 3 Schéma proposé par le fabricant pour el composant « CY7C64225 »

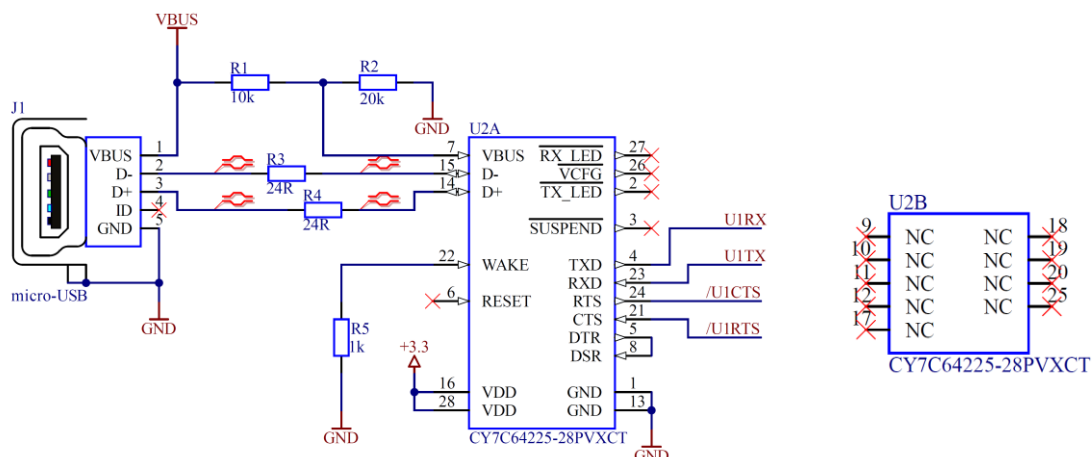


Figure 4 Schéma effectué pour la conversion du USB en UART

## 3.2. Alimentation

### 3.2.1. 5V

La liaison établie entre le pc et la carte principale étant faire par une communication USB, je dispose donc d'une alimentation en entrée de 5V et 500mA.

Pour filtrer les éventuelles perturbations IFTB, j'ai protégé le circuit avec un filtre en T avec les ferrites et un condensateur. Ce filtre passe bas permet de couper toutes les hautes fréquences potentielles de venir perturber l'alimentation. Le dimensionnement de ce filtre a été fait pour de précédent projet et stage effectué.

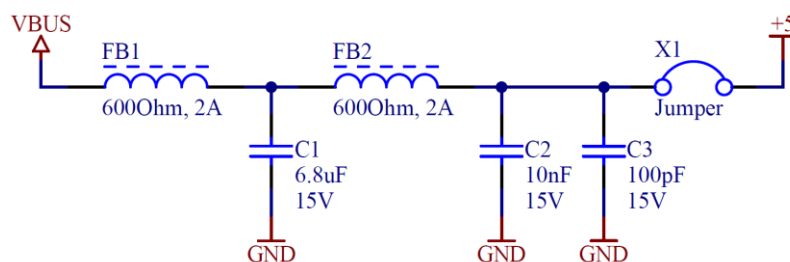


Figure 5 Filtrage de l'alimentation USB VBUS qui fournit le 5V propres à la carte

Après le filtre en T, vous remarquerez la présence d'autres deux condensateurs, ils sont là pour découpler l'alimentation sur une grande plage de fréquences.

Un jumpeur a également été placé pour pouvoir alimenter la carte avec une alimentation externe lors de la mise en service, et du développement du firmware.

### 3.2.2. 3.3V

Pour pouvoir alimenter le microcontrôleur j'ai besoin d'une tension d'alimentation de 3.3V. J'ai donc pris, en consensus avec tous ces qui devaient alimenter un microcontrôleur le même composant. C'est donc le LDO « MAX1793 » qui a été choisi. Il est simple à implémenter et nous avons du stock à l'ES.

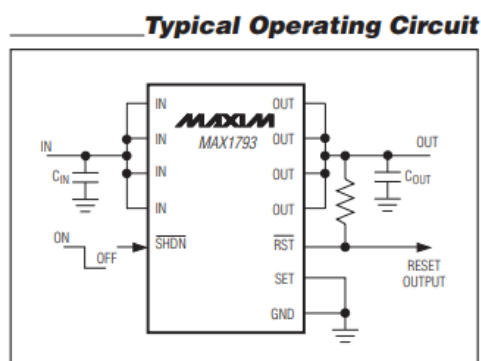


Figure 7 Recommandations du fabricant du LDO

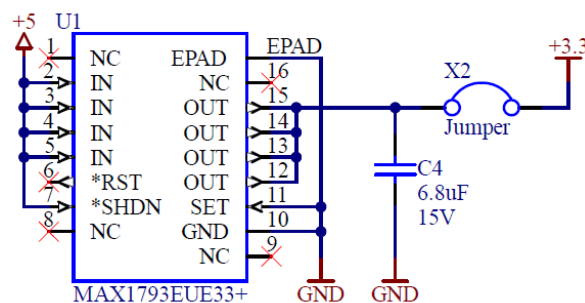


Figure 6 Schéma effectue pour le LDO

Vous pourrez remarquer que je n'ai pas mis le condensateur CIN, car j'arrive avec les 5V du USB, et on ne peut pas charger cette ligne avec plus de 10uF. En effet lors du filtrage de l'alim 5V j'utilise déjà un condensateur de 6.8uF

### 3.3. Microcontrôleur

#### 3.3.1. Communication

##### 3.3.1.1. UART

Comme précédemment dit, c'est donc une communication UART qu'il faudra décoder avec mon microcontrôleur les informations envoyées par le PC via USB. On aura notamment le nom de l'étudiant qu'il faudra afficher.

J'ai donc choisi d'utiliser l'UART numéro 1 du microcontrôleur, qui se trouve sur les pins suivantes :

U1CTS	PPS	PPS	PPS	PPS	I	ST	UART1 clear to send
U1RTS	PPS	PPS	PPS	PPS	O	—	UART1 ready to send
U1RX	PPS	PPS	PPS	PPS	I	ST	UART1 receive
U1TX	PPS	PPS	PPS	PPS	O	—	UART1 transmit

Figure 8 Tableau pour les connections de l'UART numéro 1 du microcontrôleur

Comme vous pouvez le voir, on a la mention « PPS », qui veut dire « Peripheral Pin Select ». Cela signifie que l'on peut choisir sur quel pin on veut mettre nos signaux.

J'ai donc pu choisir les pins suivantes pour l'UART1 sur MPLAB pour fixer leur position.

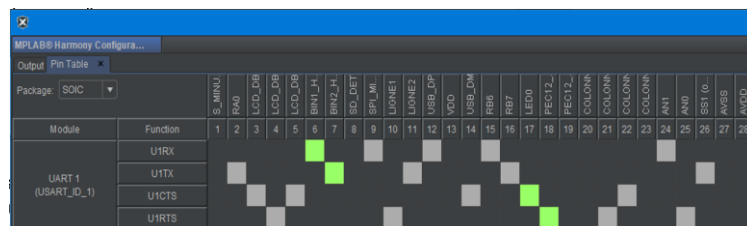


Figure 9 Configuration des pins du UART1 dans MPLAB

Puis pour la connexion au microcontrôleur, il ne faut pas oublier de croiser les signaux, comme ci-dessous.

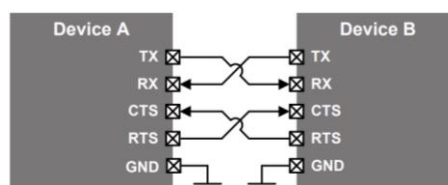


Figure 10 Connexion de l'UART Master et Slave

En effet, le PC étant le master, une fois les trames USB convertis en UART, il faut les réceptionner avec le microcontrôleur. C'est pourquoi l'envoi de données « TX » du PC est connecté à la réception de données « RX » du microcontrôleur. De même dans l'autre sens de communication, même s'il ne sera pas fondamental d'envoyer des données du microcontrôleur au PC.

Il faudra faire la même chose pour les signaux « RTS » et « CTS ».

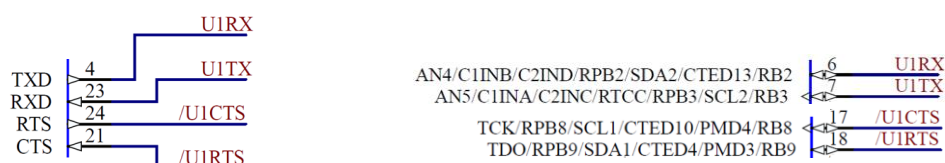


Figure 11 Pins utilisées pour la communication UART

### 3.3.1.2. SPI

Pour communiquer entre la carte principale et les modules de matrices à LED, j'utiliserais le SPI. Cela est dû au multiplexeur utilisé sur chaque module, qu'on verra plus loin dans le rapport.

J'ai donc choisi de prendre le premier SPI disponible, dans ce cas j'ai pris le SPI numéro 1, qui se trouve sur les pins suivantes :

SCK1	22	25	28	14	I/O	ST	Synchronous serial clock input/output for SPI1
SDI1	PPS	PPS	PPS	PPS	I	ST	SPI1 data in
SDO1	PPS	PPS	PPS	PPS	O	—	SPI1 data out
SS1	PPS	PPS	PPS	PPS	I/O	ST	SPI1 slave synchronization or frame pulse I/O

Figure 12 Tableau pour les connections du SPI numéro 1 du microcontrôleur

J'ai donc pu choisir les pins suivantes pour le SPI1 sur MPLAB pour fixer leur position.

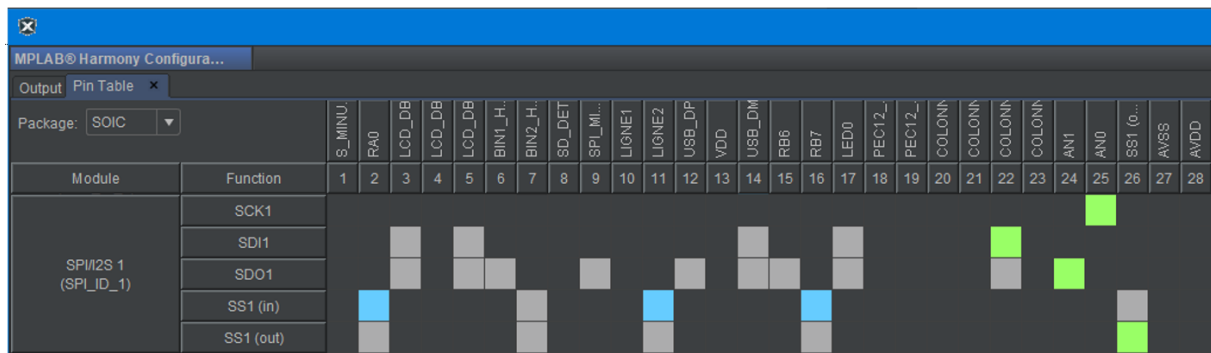


Figure 13 Configuration des pin du SPI1 dans MPLAB



Figure 14 Pins du microcontrôleur utilisées pour la communication SPI1

### 3.3.2. Quartz

Pour le choix de la fréquence du quartz, je me suis basé sur le kit micro que l'on utilise en classe, et j'ai repris la même fréquence de 8MHz. Cette fréquence sera modifiée grâce à la PLL du micro pour monter jusqu'à 80MHz, pour travailler avec les mêmes fréquences que pour les autres travaux fait au labo. J'ai opté pour l'utilisation d'un quartz car j'utilise plusieurs protocoles de communication, et car l'affichage très rapide et successif de tous les modules connectés à ma carte principale doit se faire sans accros.

Le calcul des condensateurs a été fait grâce au  $C_L = 18\text{pF}$  et le  $C_0 = 7\text{pF}$  du quartz utilisé.

$$C7 = C8 = 2 * (C_L - C_0) = 2 * (18 - 7) = 22\text{pF}$$

Figure 16 Dimensionnement selon le site de Microchip, voir références

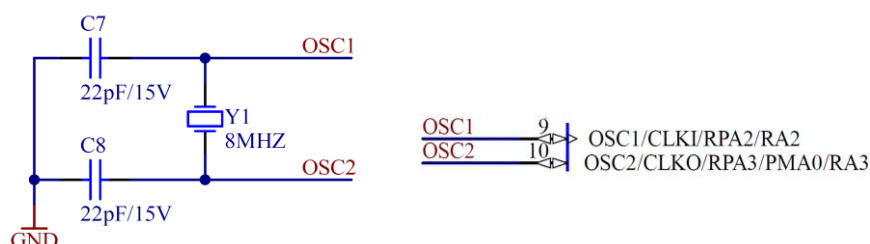


Figure 15 Pins utilisées pour le quart du microcontrôleur

### 3.3.3. Reset

L'implémentation d'un bouton poussoir sur le reste est là pour faciliter le développement, et de réinitialiser le tout s'il y a un changement d'hardware. Notamment le nombre de modules de matrices connectées. En plus d'un reset manuel, lors de la mise sous tension du circuit un reset est automatiquement effectué également.

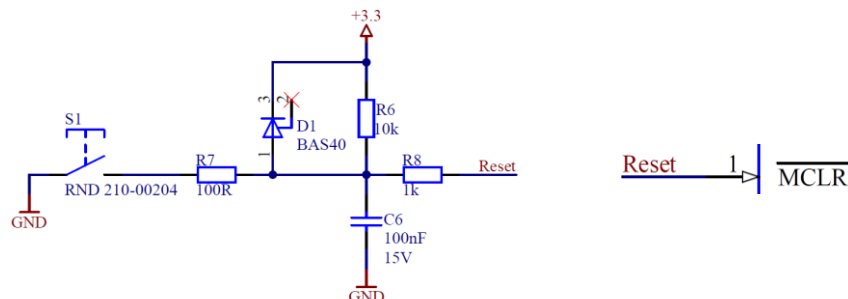


Figure 17 Schéma du montage de reset et la pin de reset

### 3.3.4. JTAG

Le JTAG est là pour pouvoir programmer le microcontrôleur en chargeant le firmware à l'intérieur. Je pourrai également connecter le débogueur pour déboguer en temps réelle notre firmware.

Le dimensionnement a été repris également du kit micro utilisé au labo. Est a été designé pour accueillir un debugger « ICD3 ».

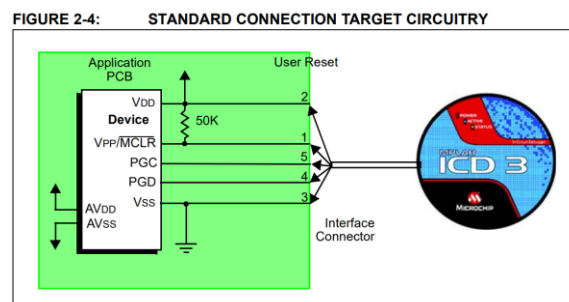


Figure 19 Connexions du « ICD3 »

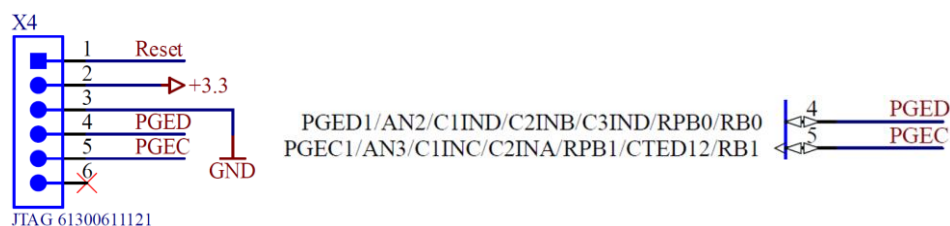


Figure 18 Connection du port JTAG avec le microcontrôleur

### 3.3.5. Monitoring

Pour faciliter le développement du firmware, et le test de fonctionnalités, j'ai sorti toutes les pinnes restantes sur un connecteur pour y avoir un accès direct. Cela va me permettre d'y mettre le signal que je veux pour tester certaines fonctionnalités. J'y ai également mis le VCC et le GND.

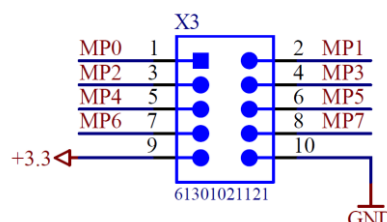


Figure 20 Port pour le monitoring



### 3.3.6. LED de vie

Pour s'assurer que les deux alimentations fonctionnent correctement et que le circuit est allumé, j'ai ajouté deux LEDs de vie.

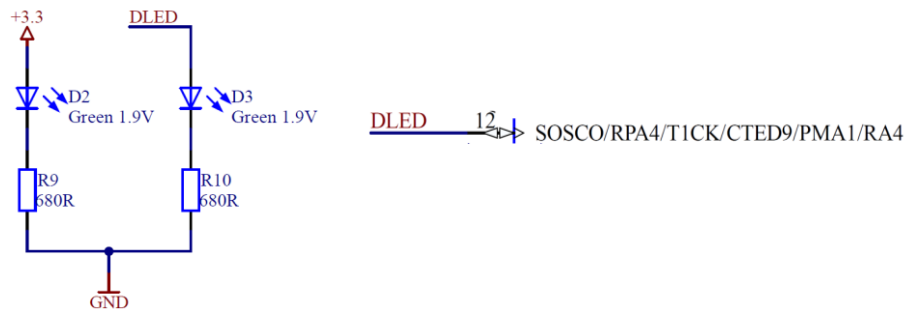


Figure 21 Connection des LEDs de vie

Pour une alimentation de 3.3V, avec sa résistance dimensionnée comme ceci :

$$R_x = \frac{(U_{3.3V} - U_{LED})}{I_{LED}} = \frac{(3.3 - 1.9)}{2 * 10^{-3}} = 700\Omega \Rightarrow E24 \ 680\Omega$$

### 3.3.7. Découplage

Comme indiqué et conseillé sur le datasheet du fabricant du microcontrôleur, j'ai ajouté un certain nombre de condensateurs de découplages entre le VCC et le GND des différentes pates du microcontrôleur.

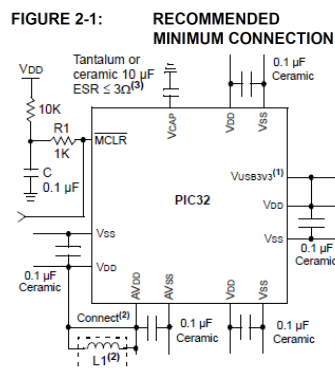


Figure 22 Recommandations de découplage du fabricant du microcontrôleur

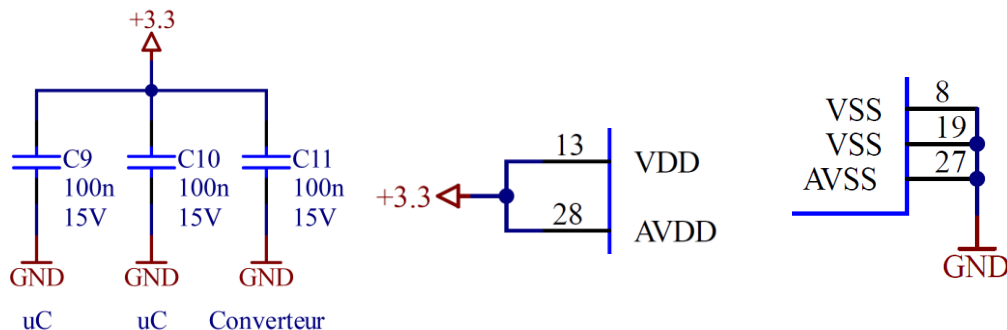


Figure 23 Découplage du microcontrôleur et du convertisseur USB to UART

En plus des deux condensateurs de découplage pour les alimentations « VDD » et « AVDD » du microcontrôleur, j'ai un troisième condensateur qui découple le convertisseur USB to UART.

## 3.4. Multiplexeur

### 3.4.1.1. Data et Adresse

Pour réduire le nombre de connexions à transmettre entre chaque module de matrice, un multiplexeur sera utilisé. Il me permettra également de directement driver les LD en courant avec une seule résistance dimensionnée par la suite.

C'est ce composant qui va décoder la trame SPI envoyée depuis le microcontrôleur.

J'ai donc 8 pins qui seront connectées sur les anodes des LEDs de la matrice, qui se trouveront connectées ensemble sur les colonnes. Et j'ai également 8 pins qui seront connectées sur les cathodes des LEDs de la matrice, qui se trouveront connectées ensemble sur les lignes.

Les colonnes « ROW » seront les pins « SEG » du multiplexeur, et les lignes « LINES » seront les pins « DIG », comme le propose un exemple dans le datasheet du fabricant.

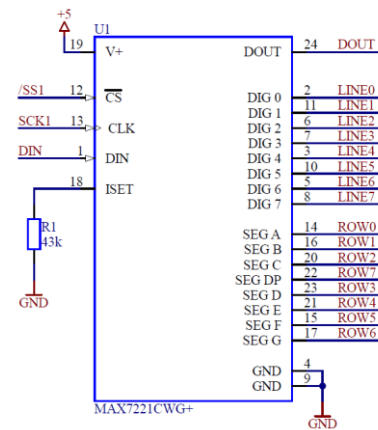
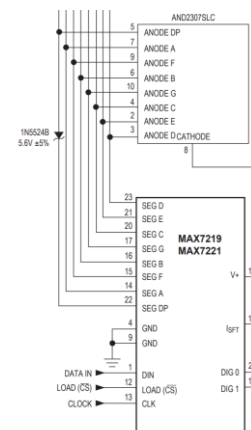


Figure 24 Recommandations du fabricant du convertisseur Figure 25 Connections du convertisseur

### 3.4.1.2. Résistance ISET

Pour dimensionner la résistance qui va nous permettre de fixer le courant pour driver les LEDs, je n'ai pas eu accès à un calcul me permettant de calculer la valeur exacte. C'est pourquoi je me suis basé sur le seul tableau avec des valeurs que le fabricant propose. J'ai donc tracé une courbe de la tendance de ces valeurs, et j'y est estimé la valeur pour une consommation de 16mA. Car j'afficherais au maximum une ligne de 8 LEDs à la fois, consommant chacune 2mA.

Table 11.  $R_{SET}$  vs. Segment Current and LED Forward Voltage

$I_{SEG}$ (mA)	$V_{LED}$ (V)				
	1.5	2.0	2.5	3.0	3.5
40	12.2	11.8	11.0	10.6	9.69
30	17.8	17.1	15.8	15.0	14.0
20	29.8	28.0	25.9	24.5	22.6
10	66.7	63.7	59.3	55.4	51.2

Note:  $R_{SET}$  values are in Kilo Ohms (k $\Omega$ )

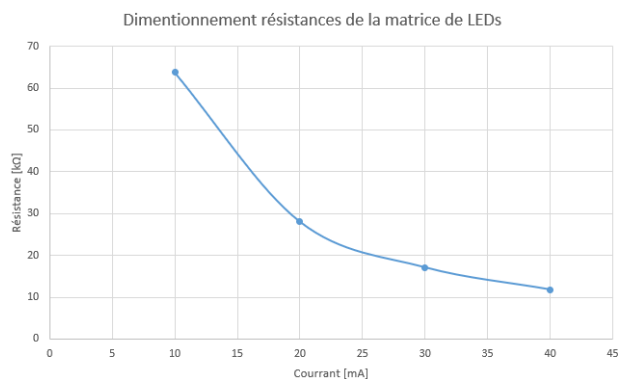


Figure 27 Table pour dimensionner  $R_{SET}$  Figure 26 Courbe extraite du tableau pour dimensionner  $R_{SET}$

En prenant du principe que la LED soit à environ 2V, et une consommation de 16mA, je tombe donc sur une valeur de 43k $\Omega$  pour la résistance.

### 3.5. Matrice à LEDs

Afin de respecter les contraintes de consommation, j'ai opté pour la réalisation des matrices à LEDs par moi-même, avec des LEDs séparées, lors de la pré-étude.

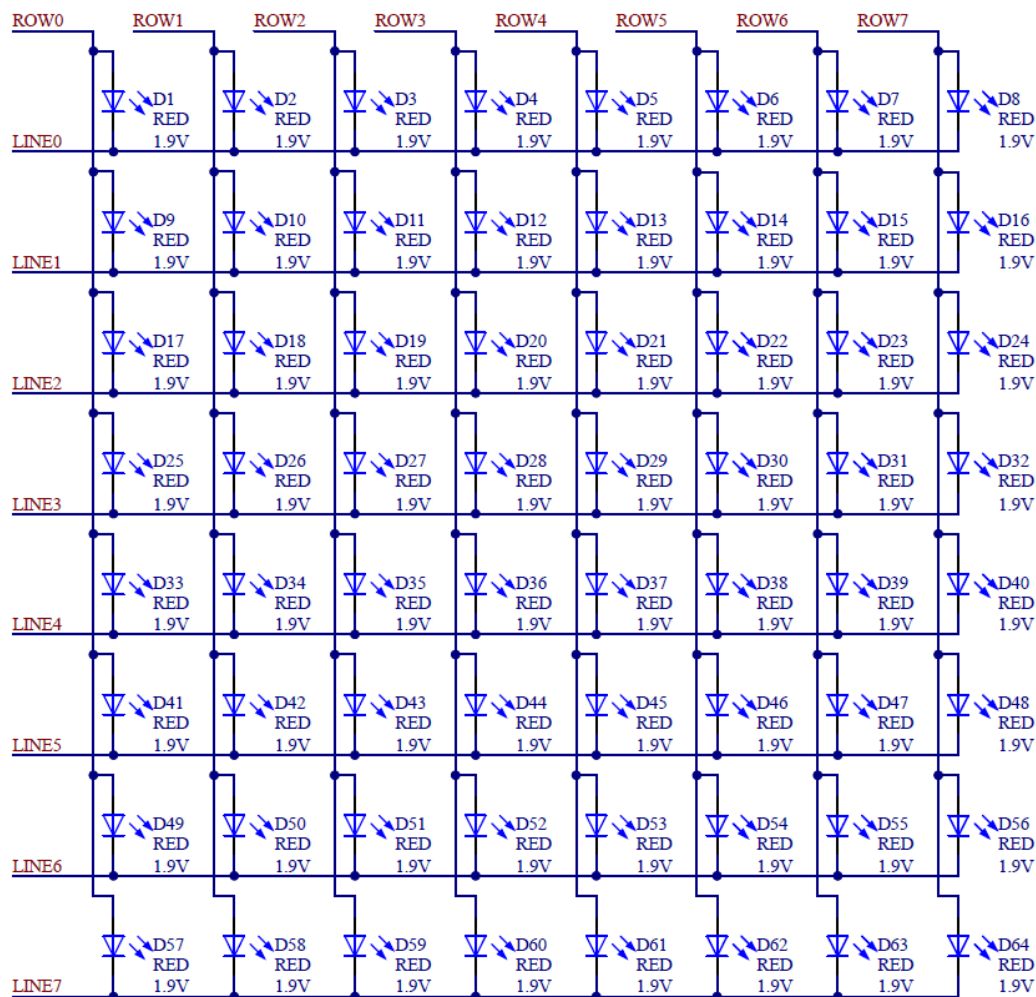


Figure 28 Connections de la matrice de LEDs

C'est donc avec un system de ligne et de colonnes que je vais pouvoir commander la LED que je veux indépendamment des autres.

Pour cela je commanderais ligne par ligne de 8 LEDs. Par exemple si je veux allumer la deuxième LED de la première ligne, je mettrais la ligne « LINE0 » à 0, et la colonne « ROW1 » à 1. Cela fonctionne de la même manière pour toutes les autres combinaisons.

Une fois la première ligne affichée, je passerais à la deuxième ligne, et ainsi de suite jusqu'à afficher les 8 lignes.

Je ferais ensuite tourner en boucle l'affichage des 8 lignes en permanence. Une ligne représentera toutes les LEDs connectées sur cette même ligne. C'est-à-dire que si j'ai trois modules de matrices chaînées, j'afficherais à chaque fois sur une ligne les 24 LEDs au même temps. Je synchroniserais le tout grâce au chip sélec, qui va me permettre de d'abord charger les informations sur tous les modules, et de déclencher l'affichage au même temps pour tous les modules.

### 3.6. Liaison inter-carte

Afin de connecter tous les modules à la carte principale, et de pouvoir chaîner les matrices, j'ai dû créer une disposition des signaux minimums à avoir. J'ai donc sorti le « MOSI » et le « MISO » du SPI, le clock du SPI et sont chip sélec. En plus des signaux de communication j'envoie également l'alimentation, dont le 5V et le GND, comme ci-dessous.

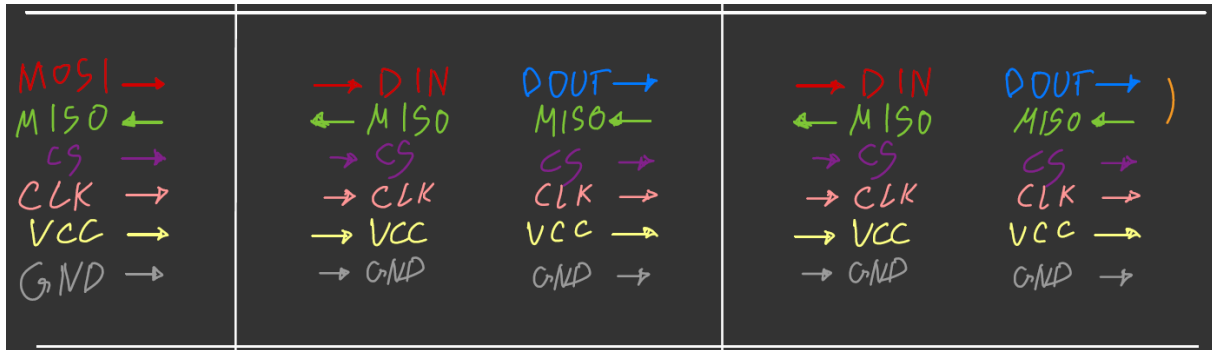


Figure 29 Pinning des connecteurs inter-carte, carte principale à gauche, suivi de deux modules à matrices

Ce qui se traduit par ça sur la carte principale, et les deux connecteurs sur le module avec la matrice.

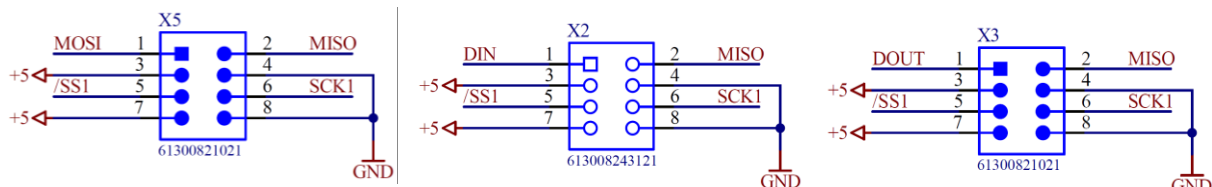


Figure 30 Connecteur de la carte principale, suivi des deux connecteurs des modules

L'utilisation du signal « MISO » pourrait être mise en doute quant à son utilité. C'est grâce à ce signal que je vais pouvoir déterminer le nombre de modules de matrices qui seront chaînées à l'allumage ou au reset du dispositif.

C'est en plaçant un jumper entre le « DOUT » et le « MISO » du dernier module de matrices, qu'une boucle sera créée. Ensuite ça sera grâce au firmware que je pourrais faire un test successif en envoyant des données sur les registres à décalage du driver de LEDs, et relire les informations que je reçois en retour.

Pour un exemple où je connecte deux modules, je devrais envoyer deux fois les packs utiles à leur utilisation, une fois pour les remplir, et une fois pour les revider et lire les informations reçues. Dans le cas où j'envoie que les informations nécessaires pour un seul module, je ne recevrais pas en retour toutes les informations envoyées.

Le choix d'utiliser un connecteur avec deux rangées des quatre pinnes a été fait, pour que le tout soit compatible avec les connecteurs 8 pins pour les câbles plats. Cela facilite très grandement la portabilité et l'adaptation du system.

## 4. Concept software

Afin de récupérer le nom de la session de l'élève connecté sur sa session, un logiciel à installer sur le PC sera nécessaire. Une fois le nom récupéré, il faudra qu'il l'envoie sur le port USB qui sera connecté à la carte principale.

Afin de faciliter la démarche, j'ai sou traité la réalisation d'un programme en C# fait par un informaticien de quatrième année de CFC. Je lui ai demandé de récupérer le nom de la session et de l'afficher dans une fenêtre sur le bureau. De mon côté je devrais envoyer l'information récupérée, et l'envoyer via USB ver la carte principale de mon system.

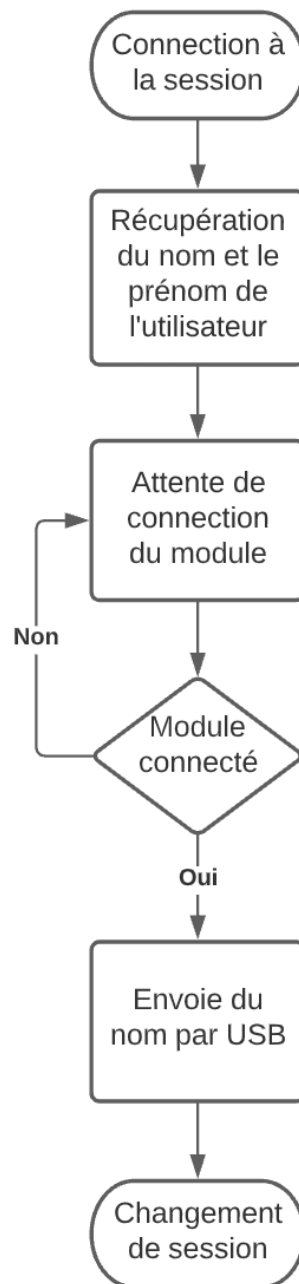


Figure 31 Flowchart du Software

## 5. Concept firmware

Pour la partie firmware, je devrais à l'allumage du system récupérer le nombre de matrices qui sont connectés à la carte principale.

Puis une fois les configurations effectuées, je devrais récupérer le nom de l'élève logué sur sa session.

Il ne me restera plus qu'à transmettre aux modules de matrices les LEDs qu'il faut allumer et éteindre pour afficher le nom correctement.

Si le nom ne rentre pas en entier sur le nombre de matrices connectées, un défilement sera effectué.

Après le premier affichage d'un nom, celui-ci est stocké dans l'EEPROM. Donc si le system est mis hors tension, puis remis sous tension, il affichera le dernier nom sauvegardé.

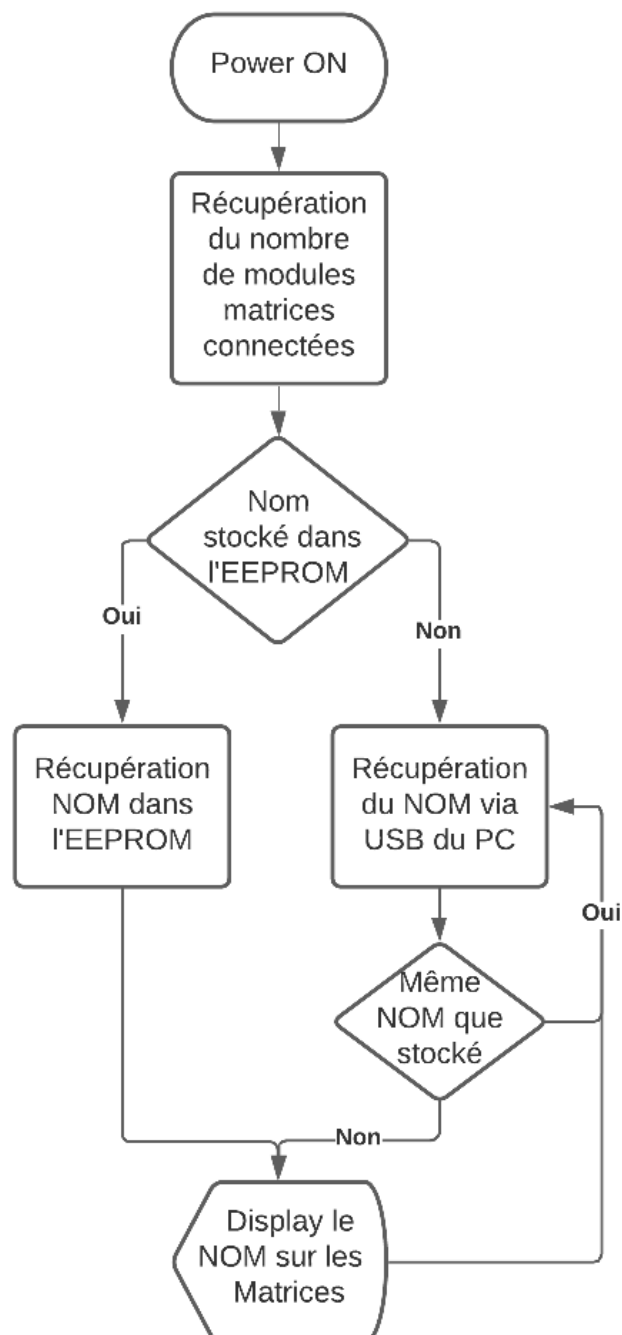


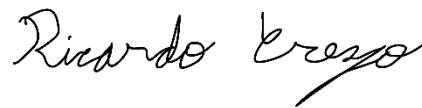
Figure 32 Flowchart du Firmware

## 6. Conclusion

Suite à cette phase de design, j'ai pu établir et définir tous les composants, J'ai également pu réaliser l'intégralité des deux schémas électriques des deux cartes.

J'ai également pu corriger les différentes erreurs faite lors de la phase de prés étude. Mais également put me confronter à des problématiques, et devoir trancher sur plusieurs choix décisifs pour le projet.

La prochaine étape sera de réaliser le PCB des deux cartes. Puis suite à leurs commandes les monter, et dans un deuxième temps développer le firmware.



ETML-ES, le 2 février 2022

Ricardo Crespo

## 7. Références

### **CY7C64225 - USB-to-UART Bridge Controller**

[https://www.infineon.com/dgdl/Infineon-CY7C64225\\_USB-to-UART\\_Bridge\\_Controller-DataSheet-v08\\_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ecca10346da](https://www.infineon.com/dgdl/Infineon-CY7C64225_USB-to-UART_Bridge_Controller-DataSheet-v08_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ecca10346da)

### **MAX1793 - Low-Dropout, Low IQ, 1A Linear Regulator**

<https://datasheets.maximintegrated.com/en/ds/MAX1793.pdf>

### **Image connexion UART master to slave**

<https://www.silabs.com/documents/public/application-notes/an0059.0-uart-flow-control.pdf>

### **Datasheet Quartz 8MHz CSM4Z-A2B3C3-60-8.0D18**

<https://www.cardinalxtal.com/uploads/files/csm4-csm5.pdf>

### **Dimensionnement des condensateur du quartz**

<https://microchipdeveloper.com/faq:937>

### **MPLAB® ICD 3 In-Circuit Debugger User's Guide For MPLAB X IDE**

<https://ww1.microchip.com/downloads/en/DeviceDoc/50002081B.pdf>

### **Datasheet PIC32MX130F064B**

<http://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX1XX2XX-28-36-44-PIN-DS60001168K.pdf>

## 8. Annexes

Vous retrouverez en annexes les deux schémas électriques.