École Normale Supérieure Paris-Saclay

Controle d'un groupe de panneau LED à l'aide d'un micro-controleur

Problématique Un ensemble de panneau LED d'origine inconnue ont été récupérées par l'association Son et Lumens. Le but de ce mini-projet est tout d'abord de comprendre comment fonctionne les panneau led (phase de retro-engineering) pour pouvoir ensuite générer à l'aide d'un micro-controleur un signal de commande.

Melchior Personnaz
Pierre-Louis Filoche

27 mai 2025

Table des matières

1 Présentation des panneau LED

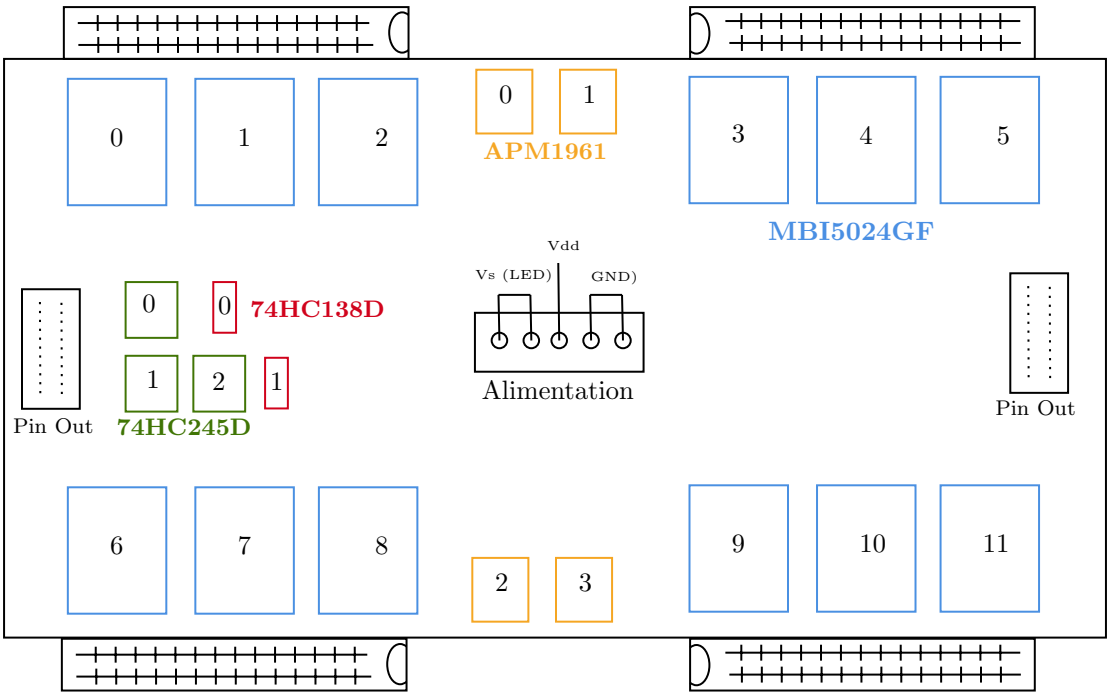
Les panneaux (voir figure **??**) sont constitués d'un assemblage de 9 panneaux élémentaires (voir figure **??**). Tous ces panneaux élémentaires sont reliés en série. Ne disposant d'aucune information sur le fonctionnement de ces panneaux à priori pas si commun, un travail préliminaire a donc consisté en l'analyse détaillée des composants d'un panneau élémentaire pour en percer les mystères.

2 Retro-engineering

2.1 Analyse des composants

Nous avons donc répertorié dans un premier temps chaque composant présent sur un panneau élémentaire et analysé à l'aide de leur documentation technique leur fonctionnement.

Schéma du Panneau LED



2.1.1 Driver LED : MBI5024GF

Le MBI5024GF est un driver LED à courant constant conçu pour piloter des matrices de LEDs dans des panneaux d'affichage. Son principe de fonctionnement repose sur la conversion de données série en signaux parallèles, tout en assurant un courant constant pour chaque canal, ce qui garantit une intensité lumineuse uniforme et stable, indépendamment des variations de tension des LEDs. **Principe de fonctionnement général**

Le MBI5024GF utilise un registre à décalage 16 bits pour recevoir les données série et les transférer vers les sorties parallèles. Chaque sortie est capable de fournir un courant constant réglable par une résistance externe, adaptée à la luminosité souhaitée. Ce composant est particulièrement adapté aux panneaux LED nécessitant une commande précise et rapide grâce à sa fréquence d'horloge pouvant atteindre 25 MHz. **Description des broches principales**

Voici les principales broches du MBI5024GF et leur rôle :

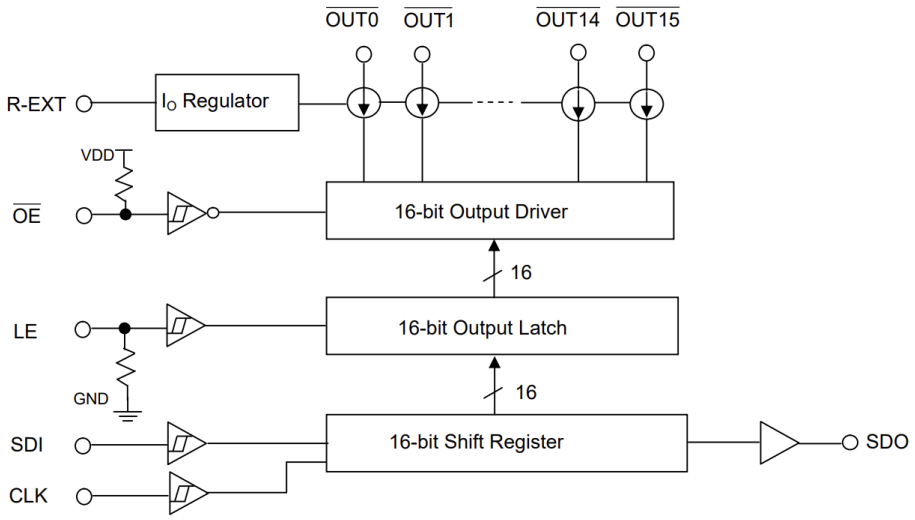


FIGURE 1 – Diagramme de Bloc du MBI5024GF

— **SDI (Serial Data Input)** : Entrée des données série, chaque bit représentant l'état d'une sortie LED (ON/OFF).
— **CLK (Clock)** : Signal d'horloge synchronisant le transfert des données série. Les bits sont lus à chaque front montant.
— **LE (Latch Enable)** : Active le verrouillage des données dans le registre de sortie. Lorsque ce signal passe de haut à bas, les données série sont transférées vers les sorties parallèles.
— **OE (Output Enable)** : Permet d'activer ou de désactiver toutes les sorties. Lorsque cette broche est haute, les sorties sont désactivées (mode blanc).
— **OUT0 à OUT15** : Canaux de sortie fournissant un courant constant pour piloter les LEDs.
— **R-EXT** : Entrée pour connecter une résistance externe définissant le courant constant des sorties.
— **VDD et GND** : Alimentation du composant.

Intégration dans un panneau LED
Dans un panneau LED, le MBI5024GF joue un rôle central en recevant les données de commande (provenant généralement d'un microcontrôleur ou d'un FPGA) via les broches `SDI` et `CLK`. Ces données déterminent l'état de chaque LED dans une matrice. Le signal `LE` synchronise l'affichage en transférant les données vers les sorties `OUT0` à `OUT15`.

Le courant constant, ajusté par la résistance connectée à `R-EXT`, assure une luminosité uniforme des LEDs, indépendamment de leurs caractéristiques de tension. Enfin, la broche `OE` permet de désactiver l'ensemble des LEDs pour des besoins de contrôle global. **Diagramme fonctionnel**

Le schéma suivant (issu de la documentation officielle) illustre l'organisation interne du MBI5024GF :

Résumé des caractéristiques principales

— 16 canaux de sortie à courant constant, ajustables de 3 à 45 mA.
— Fréquence d'horloge maximale : 25 MHz.
— Conversion série-parallèle pour un contrôle précis des LEDs.
— Alimentation compatible 3,3 V et 5 V.

Sur un panneau élémentaire (16x16 LED) on compte 12 MBI5024GF permettant de contrôler $16 \times 12 = 192$ sorties contrôlables.

2.1.2 APM1961 (MOSFET)

L'APM1961 est un transistor MOSFET utilisé comme interrupteur électronique pour contrôler la puissance et activer ou désactiver des segments du panneau LED. Il optimise l'efficacité énergétique et permet de gérer rapidement les courants nécessaires aux LEDs. Ce composant est essentiel pour isoler les sections de la matrice ou gérer des lignes spécifiques.

2.1.3 74HC245D (Transceiver bidirectionnel)

Le 74HC245D agit comme un tampon bidirectionnel entre deux bus numériques, assurant une transmission stable et isolée des données. Dans le panneau LED, il connecte le microcontrôleur aux composants de contrôle, protégeant les circuits contre les interférences ou les pertes de signal.

2.1.4 74HC138D (Décodeur 3-to-8)

Le 74HC138D est utilisé pour sélectionner une des 8 sorties en fonction de 3 bits d'entrée. Dans une matrice LED, il permet d'activer des lignes ou des colonnes spécifiques, simplifiant ainsi le câblage et le contrôle du panneau. Ce composant joue un rôle clé dans l'adressage rapide des LEDs.

2.2 Analyse du câblage des composants

Le travail suivant consiste à identifier où chaque pin de chaque composant est relié. Pour ceci, on utilise un multimètre en mode détection de court-circuit et on note soigneusement les connections entre chaque composants.

2.3 Fonctionnement d'un panneau élémentaire

On conclut le fonctionnement suivant, sur les 20 pin du HUB73, on peut répertorier l'utilité de chaque (voir figure **??** :

⇒ 4 pin pour les Led rouges

⇒ 4 pour les Led vertes

⇒ 4 enfin pour les Led bleus

⇒ 2 pin sont réservé au contrôle des lignes, en réalité on ne peut allumer que 4 lignes en même temps, c'est à l'aide de ces deux pin qu'on contrôle quels lignes sont allumées.

⇒ Une pin pour l'horloge des MBI (CLK)

⇒ Une pin pour activer ou non les sorties des MBI (output enable)

⇒ Une pin LE qui controle le fonctionnement des MBI

⇒ Et enfin 3 pins de masse

3 Génération de la commande via LPC804

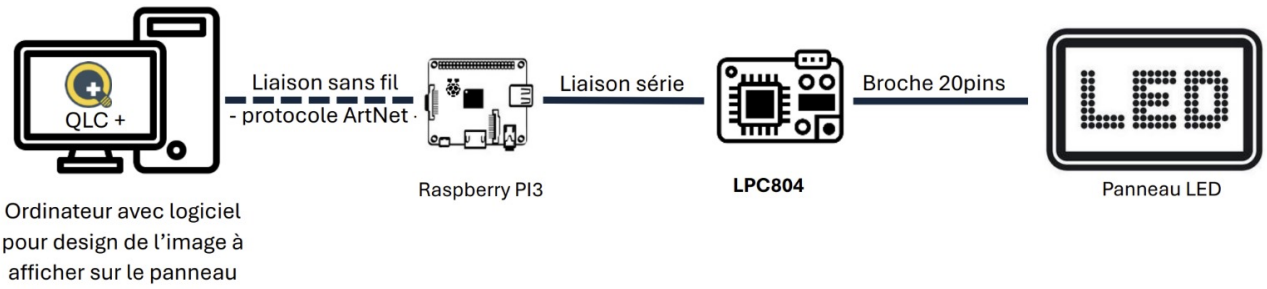


FIGURE 2 – Schéma de principe

3.1 Protocole Artnet

Le protocole Art-Net est une norme largement utilisée pour la transmission de données DMX sur un réseau IP. Conçu spécifiquement pour répondre aux besoins de l'industrie de l'éclairage, il permet de transmettre des données d'éclairage et de contrôle sur Ethernet en utilisant UDP (User Datagram Protocol). Cette méthode offre une solution robuste et flexible pour contrôler des équipements tels que des panneaux LED, des projecteurs ou des appareils DMX depuis un ordinateur ou une console. Ce protocole est donc très pratique pour notre cas d'utilisation.

Schéma d'une trame Art-Net

Une trame Art-Net typique est structurée de la manière suivante :

Champ	Taille (octets)	Description
ID Art-Net	8	Identifiant de la trame (`Art-Net\0`)
OpCode	2	Type de trame (par exemple, ArtDMX = `0x5000`)
Version du protocole	2	Version utilisée (habituellement 14 : `0x000E`)
Sequence	1	Numéro de séquence pour synchronisation
Physical	1	Numéro de port physique (souvent ignoré)
Universe	2	Identifiant de l'univers DMX
Length	2	Longueur des données DMX (0 à 512 octets)
Données DMX	Variable (*512*)	Valeurs des canaux DMX (0-255)

Pourquoi choisir le protocole Art-Net ?

Le choix du protocole Art-Net provient de deux contraintes principales :

Premièrement, une contrainte logicielle. Une autre solution aurait été de transférer directement, via une liaison série, le vecteur de données du PC vers le microcontrôleur **LPC804**. Cependant, de nombreux logiciels de contrôle de panneaux LED ne proposent pas cette possibilité. Nous avons donc préféré mettre en place une solution basée sur un protocole réseau.

Deuxièmement, une contrainte pratique. Le protocole Art-Net sans fil répond parfaitement à la nécessité d'une solution adaptable et facilement utilisable dans des contextes variés, comme une cafétéria ou une salle de concert. Une communication sans fil entre le PC et le panneau LED simplifie considérablement l'installation et réduit les contraintes matérielles, tout en offrant une flexibilité accrue dans l'utilisation du système.

3.2 Traitement de données

En réception du protocole *ArtNet* on reçoit donc une trame complète. On la nettoie pour ne garder que l'information qui nous intéresse : le DMX. On les opérations suivantes : on place les données dans [0 ;253] car les valeurs de start et stop sont 254 et 255 (on choisit de mettre des valeurs de start et stop pour faciliter la réception sur la LPC804). On doit ensuite faire atte,tion aux différents Univers DMX et leur agencement dans le vecteur final que l'on envoie au microcontrôleur. En effet une trame DMX contient 512 valeurs, or le panneau élémentaire sur lequel on travail a 16×16 LED $\times 3$ (RGB) $= 768$ canaux. Il nous faut donc deux univers pour envoyer toutes les informations. Lorsqu'on nettoie la trame reçue on fait donc bien attention à l'univers (0 ou 1) qui est envoyé pour les remettre dans l'ordre. L'agencement des 768 canaux entre les deux univers se règlent facilement dans le logiciel en patchant la matrice de LED, il faut toutefois être vigilent lors de la reconstruction à être cohérent avec le patch (on s'est fait avoir longtemps sur ça) !

3.3 Liaison Série

Pour envoyer le vecteur de données de la RPI3 au microcontrôleur. Partie à finir après implémentation...

3.3.1

Pour chaque pin contrôlant les led, l'information est envoyée en série sur 16 bits, chaque bit correspondant à une led d'une ligne (1 la led sera allumée au prochain front montant de la pin LE). On joue ensuite sur les deux pins de contrôle des lignes pour allumer la bonne ligne.

On envoie donc l'information série synchrone avec le signal d'horloge correspondant à une ligne puis on envoie le signal au led en envoyant un front montant sur LE. On réeffectue la même opération pour la ligne suivante en changeant les valeurs des pins de contrôle des lignes. On contrôle simultanément 4 lignes des 16 grâce aux 4 pins dédiés.

4 Génération de commande via STM32

On a maintenant deux principaux objectifs : porter le code de la LPDC804 vers la STM32 et mettre en place une connexion IP entre stm32 et PC pour avoir un système autonome et ne plus réutiliser de RPI.

4.1 LPC804 ⇔ STM32

Le code d'origine de la LPC804 reçoit un mot (= image à afficher) en RGB et le traduit en commande avec le panneau LED avec gestion des timers et du séquençage du panneau. Il faut traduire ce code pour la STM32 avec gestion des timers, des interruptions et GPIO. On choisit arbitrairement le timer 2, en mode Internal Clock sur interruptions. Calcul de f_{Timer2} :

$$f_{Timer2} = \frac{f_{max}}{(1 + PreScaler) \times (1 + HRR)}$$

Pour les ports GPIO on redéfinit tout via le `.ioc`

Fonction	Port STM32
CLOCK_PIN	PH9
OUTPUT_E	PH11
LE_PIN	PH14
SEQ_1	PE5
SEQ_2	PE6

LED (RED)	Port STM32
1	PA6
2	PA4
3	PG9
4	PH13

TABLE 1 – Correspondance entre les ports GPIO de la STM32 et les signaux du panneau LED

Et on a les associations avec le panneau LED :

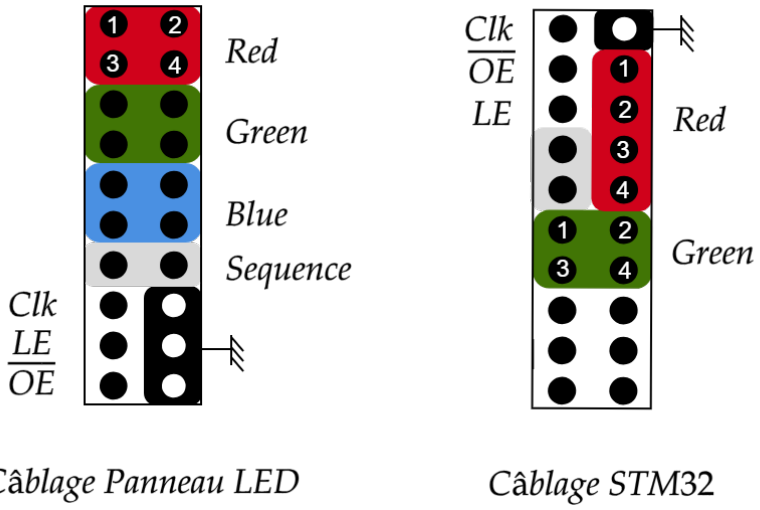


FIGURE 3 – Câblage entre panneau LED et STM32

A posteriori cette disposition n'est pas optimale pour le câblage, il faudrait mieux reproduire au mieux le schéma de câblage du panneau LED sur les ports de la STM32, à améliorer.

4.2 Communication IP ⇔ PC

4.2.1 Gestion de l'IP sur STM32

Ressources utiles :

— **Fiche Ethernet TCP/IP STM32 746NG** – A. Juton
Pour la configuration réseau via CubeMX et les bases de LwIP.

— **Retour d'expérience sur LwIP** – GitHub : `https://github.com/pierrot-cadeilhan/miniprojet-LwIP/blob/main/difficultes.md` - Pierrot Cadeilhan
Pour les principales difficultés avec la mémoire de la pile LwIP sur STM32.

La mise en place de la communication IP avec la STM32 a commencé par une configuration soignée dans STM32CubeMX. Nous avons d'abord activé le périphérique Ethernet (ETH) en choisissant l'interface RMII, adaptée à notre carte et au PHY utilisé. Ensuite, nous avons ajouté la pile réseau LWIP (Lightweight IP) depuis l'onglet Middleware, en activant l'option "Use Ethernet" dans les paramètres de LWIP. Dans l'onglet *LWIP*, nous avons désactivé le DHCP pour attribuer une adresse IP statique à la carte, et défini manuellement l'adresse IP.

Un problème fréquent lié à LWIP est la mauvaise allocation de la pile mémoire, notamment lorsque celle-ci est placée dans une zone inaccessible par le DMA. Pour résoudre cela, nous avons suivi la méthode documentée dans le Git de Pierrot : il faut forcer LWIP à utiliser une section mémoire spécifique. Concrètement, dans `STM32F746NGHX_FLASH.ld`(le linker script), on a redirigé l'allocation mémoire vers une région définie dans la SRAM1 (plutôt que dans les régions non compatibles DMA comme DTCM ou ITCM sur certains STM32).

```
MEMORY
{
  RAM     (xrw)     : ORIGIN = 0x20000000,   LENGTH = 288K
  LWIP_RAM      (xrw)    : ORIGIN = 0x20048000,   LENGTH = 32K
  FLASH     (rx)     : ORIGIN = 0x8000000,   LENGTH = 1024K
}
```

Listing 1 – Changements effectués (LWIP_RAM

4.2.2 Envoie de données

On test avec un petit script python `interface_graphique.py` l'envoie d'un mot à la STM32. Résultat concluant ! On essaye donc avec `Jinx!`, logiciel déjà utilisé au S1 pour la création d'animation graphique et envoie en IP. Cette fois on peut envoyer directement à la STM32 sans avoir besoin de passer par UN RPI3. On recoit bien sur la carte (problème d'affiche lié à la longueur de la trame à gérer).

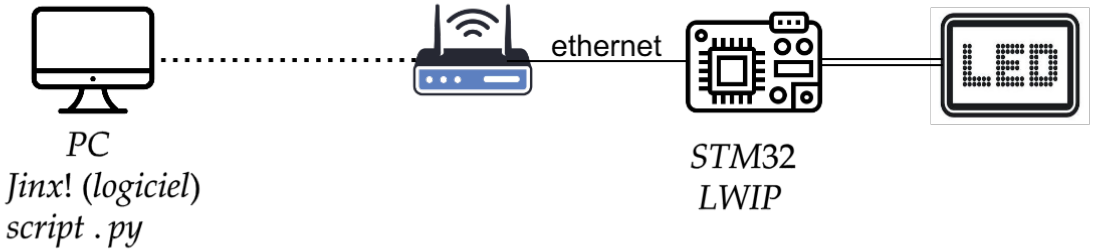


FIGURE 4 – Schéma de principe

A terme on aimerait bien mettre en place une liaison WiFi entre le routeur et la STM32 grâce a un pont WiFi-Ethernet (solution simple à mettre en place sans modifier le code, il faut aliment le composant).