Diplôme 2012

Représentation en 3D avec un cube à LEDs



Jonathan Aubert

Enoncé officiel

## Sujet Représentation en 3D avec un cube à LEDs

**Description** Pour réaliser une représentation en 3 dimensions, plusieurs solutions sont envisageables. Pour ce travail, le candidat va utiliser un cube composé de 512 LEDs (8 LEDs par côtés) pour créer une représentation en 3 dimensions.

Le but de ce travail consiste à réaliser l'interface de puissance permettant de commander les LEDs ainsi que la programmation d'un microcontrôleur réalisant la logique de commande afin de piloter chaque LEDs individuellement.

Un programme sur ordinateur doit permettre de constituer la matrice d'une image en 3D.

Si le temps le permet, le candidat étudiera la possibilité de visualiser une représentation de la variation du spectre de fréquence audio dans le temps. Pour cela, un filtre à 8 bandes doit être réalisé afin de déterminer l'amplitude dans chacune des bandes utilisées.

**Travail** Etude du système complet.

Etude, calculs et mise en point de l'interface de puissance.

Réalisation d'un programme sur microcontrôleur cygnal F320 permettant la gestion des LEDS.

Réalisation d’une application en C# sur ordinateur permettant de réaliser une image en 3 dimensions.

Etude, calculs et mise en point d'un filtres 8 bandes.

Réalisation d'un programme sur microcontrôleur cygnal F320 permettant la visualisation du spectre d'un signal audio.

**A remettre** Un mémoire dactylographié en 3 exemplaires, couverture rouge, comprenant:

- Une description technique complète, avec la motivation des choix effectués, les schémas et les calculs des composants.

- Une description des structures de données et du protocole de communication.

- Les descriptions, organigrammes et « listings » des programmes.

- Les résultats des essais et mesures effectuées.

- Les caractéristiques des composants spéciaux.

Le montage du dispositif, mécanique et électronique.

Les logiciels, schémas, organigrammes, etc., sur support informatique.

Patrick Tutt

Table des matières

[1.1 Sujet Représentation en 3D avec un cube à LEDs 2](#_Toc326587101)

[3 Précisions sur l’énoncé 5](#_Toc326587102)

[4 Etudes effectuées sur le thème 6](#_Toc326587103)

[5 Description générale 7](#_Toc326587104)

[5.1 Introduction 7](#_Toc326587105)

[5.2 Liste de matériel 7](#_Toc326587106)

[5.3 Représentation de l’ensemble 8](#_Toc326587107)

[5.4 Schéma bloc 8](#_Toc326587108)

[6 Description de détails 10](#_Toc326587109)

[6.1 Calcul des courants LEDs 10](#_Toc326587110)

[6.2 Choix des composants 10](#_Toc326587111)

[6.2.1 Génération du clock: 10](#_Toc326587112)

[6.2.2 Compteur en anneau 11](#_Toc326587113)

[6.2.3 Transistors de commande des étages 11](#_Toc326587114)

[6.2.4 Drivers pour les anodes 11](#_Toc326587115)

[6.2.5 Microcontrôleur 11](#_Toc326587116)

[6.2.6 Filtrage du spectre audio 12](#_Toc326587117)

[6.2.7 Connecteurs 12](#_Toc326587118)

[6.2.8 Interrupteurs 12](#_Toc326587119)

[6.3 Alimentation de la logique 13](#_Toc326587120)

[6.4 Réalisation du bloc drivers 14](#_Toc326587121)

[6.4.1 Générateur de clock 14](#_Toc326587122)

[6.4.2 Compteur Johnson 18](#_Toc326587123)

[6.4.3 Transistors de puissance 20](#_Toc326587124)

[6.4.4 Drivers sources 23](#_Toc326587125)

[6.4.5 Limitation du courant traversant les LEDs 26](#_Toc326587126)

[6.4.6 Spécifications des LEDs 28](#_Toc326587127)

[6.4.7 Choix de la tension d’alimentation de puissance 29](#_Toc326587128)

[6.4.8 Dimensionnement des câbles et pistes 30](#_Toc326587129)

[6.5 Analyse du spectre audio 32](#_Toc326587130)

[6.6 Connexions au microcontrôleur 33](#_Toc326587131)

[6.6.1 Schéma général 33](#_Toc326587132)

[6.6.2 Boutons poussoirs 34](#_Toc326587133)

[6.7 Programmation µC bas niveau 37](#_Toc326587134)

[6.7.1 Organisation du cube en mémoire 37](#_Toc326587135)

[6.7.2 Rafraichissement de l’affichage 38](#_Toc326587136)

[6.7.3 Transmission de données aux drivers sources (MIC5891) 42](#_Toc326587137)

[6.7.4 Correction du problème de persistance rétinienne 49](#_Toc326587138)

[6.7.5 Gestion de la luminosité 53](#_Toc326587139)

[6.7.6 Communication USB 58](#_Toc326587140)

[7 Logiciels 61](#_Toc326587141)

[7.1 Programme du microcontrôleur 61](#_Toc326587142)

[7.1.1 Introduction 61](#_Toc326587143)

[7.1.2 Organigramme de la boucle principale (Main) 62](#_Toc326587144)

[7.1.3 Liste des fonctions primitives 63](#_Toc326587145)

[7.1.4 Liste des fonctions niveau utilisateur 63](#_Toc326587146)

[7.2 Application PC 64](#_Toc326587147)

[7.2.1 Introduction 64](#_Toc326587148)

[7.2.2 Manuel d’utilisation de l’application « Cubizator » 64](#_Toc326587149)

[7.2.3 Diagrammes UML de l’application 69](#_Toc326587150)

[7.2.4 Organisation des fichiers .cube 75](#_Toc326587151)

[8 Résultats (schémas et mesures) 77](#_Toc326587152)

[8.1 Schéma de la carte 77](#_Toc326587153)

[8.2 Mesures du courant LED 77](#_Toc326587154)

[8.2.1 Mesure du courant avec la R79 de 30 [Ω] 78](#_Toc326587155)

[8.2.2 Mesures avec uniquement la colonne 64 allumée et R79 de 27 [Ω] 79](#_Toc326587156)

[8.2.3 Mesure avec tout le cube allumé (R79 de 27 [Ω]) 85](#_Toc326587157)

[8.3 Mesures de la consommation de la carte 88](#_Toc326587158)

[9 Conclusion 89](#_Toc326587159)

[10 Améliorations 90](#_Toc326587160)

[11 Mode d’emploi 91](#_Toc326587161)

[11.1 Alimentation 91](#_Toc326587162)

[11.2 Navigation dans le menu du cube 91](#_Toc326587163)

[11.3 Connexion à un PC 91](#_Toc326587164)

[12 Remerciements 91](#_Toc326587165)

[13 Bibliographie 92](#_Toc326587166)

[14 Annexes 93](#_Toc326587167)

# Précisions sur l’énoncé

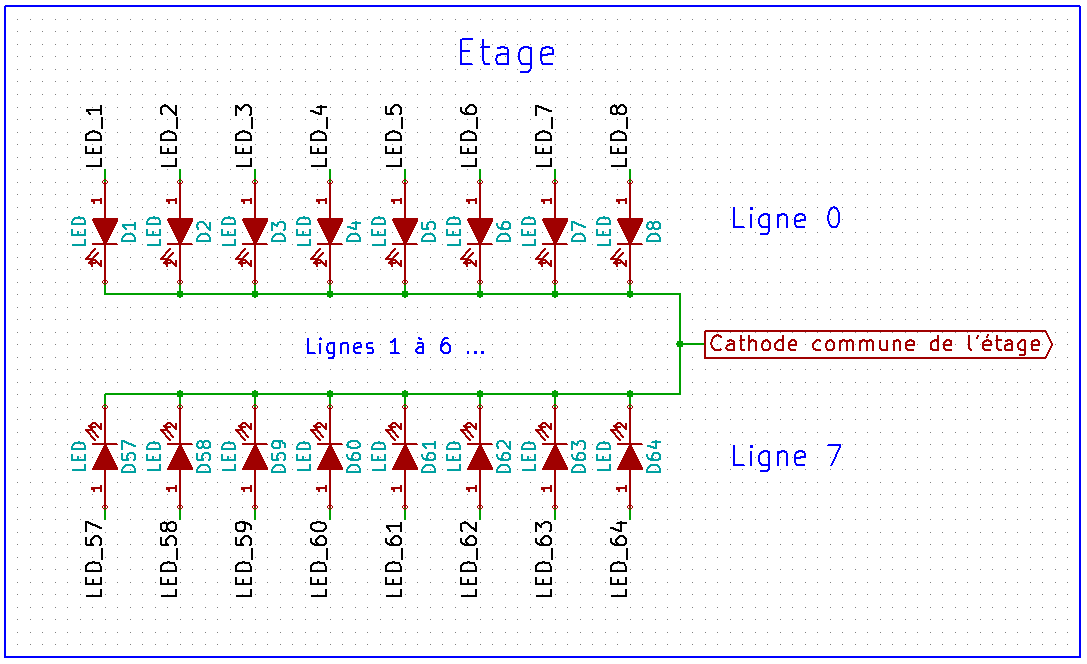
Tout d’abord, la structure cubique de 8x8x8 LEDs (bleues) fût réalisée avant le début du diplôme.

Celle-ci se trouve dans une configuration en cathodes communes. Le découpage se réalise via 8 étages. Le choix de la cathode commune provient d’une simple copie du cube à LEDs 8x8x8 réalisé sur le site « instructables.com ».



Illustration ci-dessus : Instructables.

Ci-dessous, le détail d’un étage vu de dessus:



Il y a donc 64 anodes à alimenter positivement via des drivers. Les cathodes communes seront reliées par étage. Il suffira de connecter l’étage à commander à la masse, pour allumer la ou les LEDs aux points définis.

Les anodes sont reliées verticalement, c’est-à-dire par exemple que toutes les anodes des LEDs n° 1 sont reliées.

Les étages sont numérotés selon l’ordre suivant : l’étage le plus bas est le n°0 et le plus haut, le n° 7.

Concernant l’interface d’interaction avec le cube, 3 boutons permettront de faire la sélection d’une animation dans un menu.

La connectique choisie pour charger les données à afficher dans la carte est l’USB.

Il y aura également une fiche « mini Jack » afin d’y connecter une source sonore et d’y afficher son analyse spectrale dans le cube.

# Etudes effectuées sur le thème

Toujours avant de commencer le diplôme, j’ai réalisé une série de tests me permettant de déterminer quelle serait la fréquence de rafraichissement et le courant par LED optimal pour l’affichage d’une image 3D.

Comme 8 étages seront présents, le rapport cyclique d’allumage = 1/8, soit 0.125.

Pour déterminer la fréquence, une LED provenant du même lot que celles du cube fût pilotée avec le rapport cyclique mentionné ci-dessus, d’abord à 50 [Hz], puis à 100 [Hz].

Résultat : Le scintillement est encore légèrement perceptible à 50 [Hz] alors qu’à 100 [Hz], l’œil ne fait plus la différence avec une LED alimentée en continu.

Le rafraichissement total du cube devra être réalisé à une fréquence de 100 [Hz], ce qui implique une fréquence de 800 [Hz] pour commander tous les étages.

Concernant le courant à injecter dans les leds (par impulsions), j’ai fait en sorte de mesurer 20 [mA] RMS avec l’aide d’un multimètre Amprobe 37XR-A (True rms) sous le rapport cyclique d’1/8, ce qui m’a donné 65 [mA] de valeur instantanée. Avec 65 [mA], l’intensité de l’éclairage rafraichi à 100 [Hz] est similaire à celui résultant avec un courant continu de 20 [mA]. La comparaison avec la valeur calculée sera effectuée dans la partie calculs.

La structure est câblée en cathodes communes, et, ce fût une belle erreur de ma part de recopier le cube trouvé sur le net. En effet, l’utilisation d’une configuration de type anodes communes m’aurait permis l’utilisation de drivers « sink » type STP08CP05 et ainsi permettre de réguler simplement et efficacement le courant pour toutes les LEDs (avec 1 résistance par driver).

Le montage réalisé utilise donc des drivers « sources » (voir le chapitre 6.2.4).

# Description générale

## Introduction

Dans le cadre de ce travail de diplôme, l’utilisateur pourra choisir l’animation à afficher. Pour cela, j’ai choisi la solution suivante : Trois boutons poussoirs rendront possible la navigation à travers un menu. Un mode « connexion au PC » permettra de dessiner, puis d’afficher une animation en 3D créée par l’utilisateur via un logiciel réalisé pour l’occasion. L’analyse des fréquences du spectre audio provenant d’une source sera également réalisée. Ainsi, la carte de commande se chargera de la communication avec le PC, du traitement audio, d’une partie de l’interface homme-machine, du calcul des animations et enfin, de l’affichage via les drivers de puissance. Par soucis de temps, l’alimentation du montage sera externe. Ce mémoire est le fruit de cette étude.

## Liste de matériel

1 Oscilloscope LeCroy WaveJet 324 SN : LCRY0101J22746

3 Sondes LECROY PP010

1 Analyseur logique Agilent Technologies MS07034B SN : MY50340262

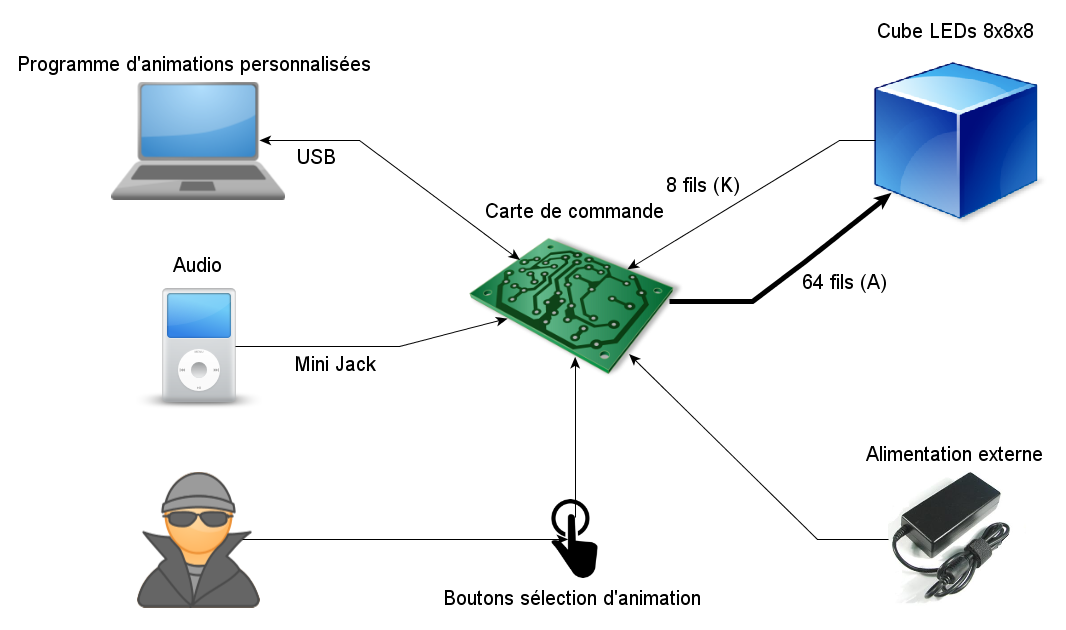
1 Multimètre Fluke 175 SN : 97101217

1 Multimètre Fluke 175 SN : 99040601

1 Alimentation GW INSTEK GPS-3303 SN : EL884441

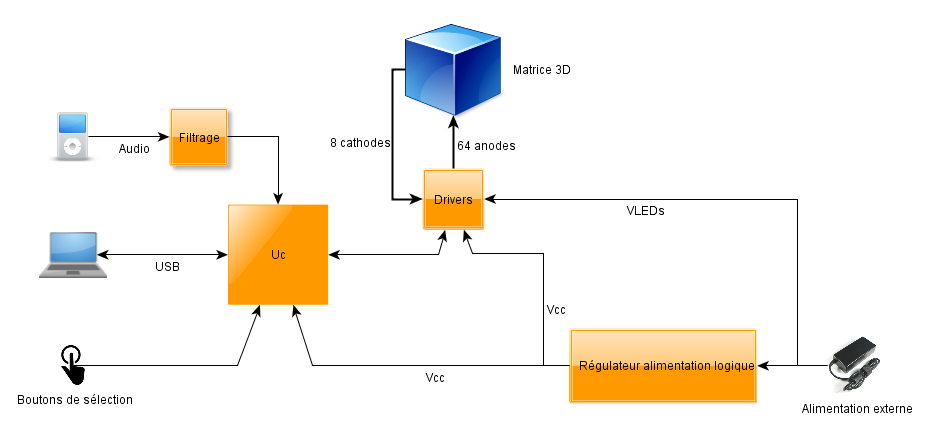
1 kit silabs C8051F320-TB

## Représentation de l’ensemble



## Schéma bloc

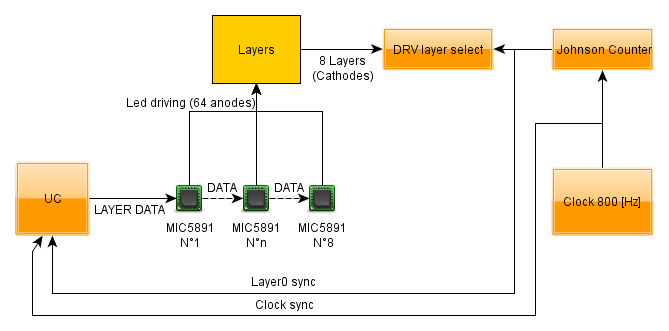
Voici les différentes parties constituant le montage :



Les leds seront alimentées via le bloc « Drivers ». Le balayage sera effectué de bas en haut en reliant successivement les cathodes à la masse. La tension appliquée sur l’étage de puissance proviendra directement de l’alimentation externe. Un régulateur de tension se chargera de la partie logique afin d’éviter au maximum les variations de

tensions lors d’appels de courant. Un composant spécialisé aura la tâche de filtrer le spectre audio et de transmettre l’amplitude de chaque bande de fréquence en une tension analogique mesurable par le convertisseur A/D du microcontrôleur.

Détail des drivers :



Ce que j’ai à tout prix voulu éviter était d’avoir un blocage la sélection d’un étage puisque dans le cas d’un courant de plus de 30 [mA] dans une LED, celle-ci grillerait. (Voir la datasheet de celle-ci). La solution envisagée est une gestion de la sélection des étages de manière indépendante au µC. Un clock à 800 [Hz] « anime » un compteur en anneau qui pilote les transistors de commande des étages un à un. Le µC reçoit le signal du clock ainsi que la pulsation de commande de l’étage 0, autrement dit, l’étage tout en bas du cube, ce qui permettra une synchronisation du clock et des étages. Les 8 drivers recevront les données à afficher via le µC à chaque changement d’étage.

# Description de détails

## Calcul des courants LEDs

Comme vu lors de l’étude effectuée lors du chapitre 4, le test réalisé dans la pratique montre que le courant traversant une LED devrait être limité à 65 [mA] (sous le rapport cyclique d’1/8) pour obtenir une luminosité égale à celle visible avec un courant continu de 20 [mA].

Calcul du courant RMS avec une limitation à 65 [mA] :



Constat : le courant calculé est légèrement supérieur à 20 [mA]. Malgré tout, le développement de la carte s’est effectué en prenant en compte la valeur pratique (65 [mA] de valeur instantanée).

Calcul du courant nécessaire pour l’allumage complet :

Puisque tous les étages seront allumés, les drivers sources (pour les anodes) devront fournir continuellement 65 [mA] par anode.



## Choix des composants

### Génération du clock:

La fréquence du clock de pilotage des étages sera de 800 [Hz], puisque l’étude réalisée avant le début du diplôme montre qu’à une fréquence de rafraichissement de 100 [Hz] dans une LED aucun scintillement n’est perceptible. Avec 8 étages, la valeur est multipliée par huit.

Après diverses recherches, (LM555, Oscillateur à quartz avec diviseurs, « vco » etc...) la solution retenue est l’utilisation du récent LTC6990. Il s’agit d’un « vco » commandé très simplement dans une plage allant de 440 [Hz] à 2 [MHz] et alimentable entre 2.25 et 5.5 [V]. Il ne nécessitera que 3 résistances dans la configuration choisie et possède une sortie compatible avec de la logique CMOS.

### Compteur en anneau

Le but réaliser le balayage de l’image en commandant les 8 transistors de sélection des étages les uns après les autres. J’ai trouvé deux types de compteur Johnson (en anneau) : le CD4017 (possédant 10 sorties) et le CD4022 (possédant 8 sorties). Le plus simple est d’utiliser directement le CD4022.

### Transistors de commande des étages

Les conditions sont : assurer l’allumage d’un étage avec une tension de déchet la plus faible possible, tout en ayant une tension de commande compatible avec la sortie du compteur Johnson.

Le courant (instantané) maximum d’un étage sera de 4.16 [A] durant la pulsation de commande.

Après diverses recherches dans le stock de l’école, l’exploitation d’un MOSFET à enrichissement canal N « STP16NF06L » semble possible : RDS(on) = 0.07 [Ω], VGSth entre 1 et 2 [V] (commande via la logique) et IDmax = 16 [A].

### Drivers pour les anodes

Il faut des drivers capables d’alimenter les 64 LEDs en permanence sous 65 [mA].

L’utilisation de 8 circuits MIC5891 en chaine est possible (Ioutmax = 500 [mA]). La transmission des informations en série (type registre à décalage) peut s’effectuer rapidement (garantie à 3.3 [MHz]).

### Microcontrôleur

Le c8051F320 utilisé à l’école semble viable pour cette application. Etant doté d’une connectique USB, la connexion à un PC s’en retrouvera aisée. Sa vitesse d’horloge (avec le mode USB actif) à 24 [MHz], suffira pour l’envoi des données dans les temps. Cependant, la réalisation de certaines fonctions de calcul matriciel (à réaliser dans le futur) pourrait-être limitée.

### Filtrage du spectre audio

Afin de réaliser la séparation des fréquences du spectre audio, le MESG7 est idéal puisqu’il transmet directement une valeur analogique pour chaque bande fréquence. Note : Le spectre est divisé en 7 fréquences. Il faudra sûrement réaliser un filtre supplémentaire pour l’affichage complet dans le cube. (Ou trouver un subterfuge !)

### Connecteurs

Alimentation : le courant maximum devrait se situer aux alentours des 4.16 [A] (sans la logique). Un Jack d’alimentation SwitchCraft RASM712 de 2.5/5.5mm permet de laisser passer un courant nominal de 5 [A], ce qui laisse une certaine marge de sécurité.

Cathodes communes des étages : ici le courant RMS maximum sera de 1.47 [A] (voir le calcul dans la partie 6.2.3 Transistors de puissance). Un couple de connecteurs 5566-08A / 3928-1083 et 5557-08R / 3901-2080 sera utilisé. Ceux-ci permettent la circulation d’un courant constant de 7[A] maximum.

Anodes leds : chaque led étant alimentée par un courant maximum de 65 [mA], un câble plat à 64 pins reste la meilleure alternative. Les connecteurs utilisés sont de types DIN 41651: le 6-828582-4 de chez TE Connectivity et le C3001-64YYGB00R fabriqué par Hsuan Mao.

### Interrupteurs

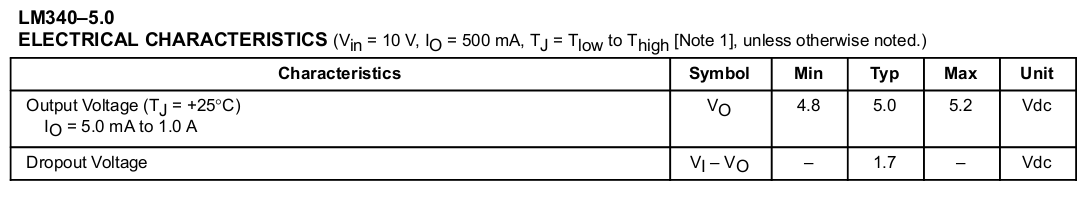
Les boutons de commande du menu seront de simples micros interrupteurs de type JPM 1990-0101.

L’alimentation du cube se fera au travers d’un interrupteur à bascule type 1801.6102 fabriqué par Marquardt. Il est capable de laisser passer 10 [A] ce qui est largement suffisant.

## Alimentation de la logique

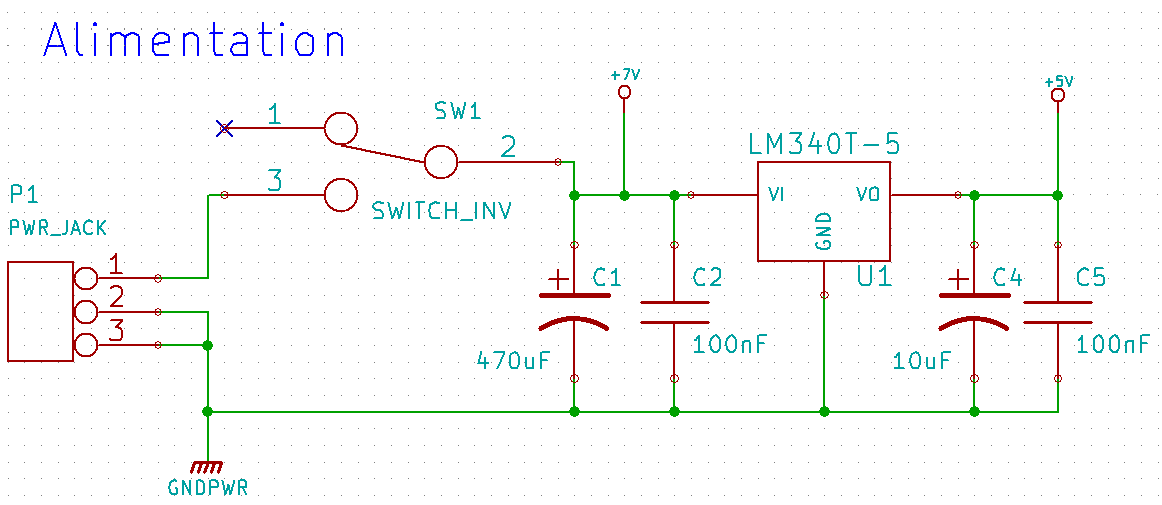
Toute l’électronique installée sur la carte est capable de fonctionner sous une tension de 5 [V].

Le régulateur de tension utilisé sera un LM340T-5 pouvant fournir un courant continu maximum d’1 [A], ce qui sera amplement suffisant pour la logique de la carte (consommation estimée à 30 [mA] avec marge). En conséquence, le calcul de la dissipation de puissance fût négligé.

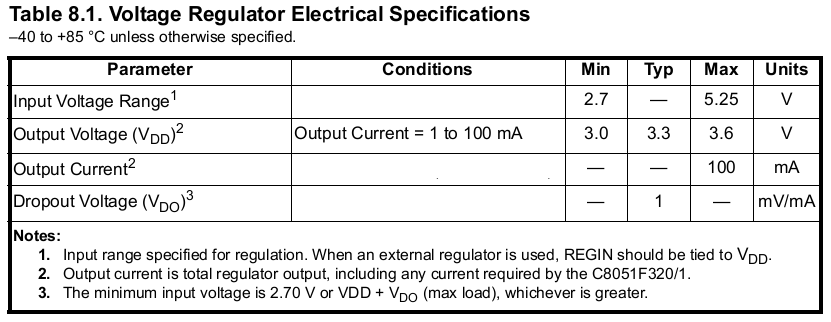


La tension d’alimentation minimale devra être de 6.7 [V] (Uout + Udropout).

Schéma :



Le µC possède un régulateur interne de 3.3 [V]. Il est possible de l’utiliser pour alimenter d’autres composants :



## Réalisation du bloc drivers

### Générateur de clock

Cette partie du montage est délicate, dans la mesure où il faut garantir son fonctionnement, et ce, même en cas de variations de la tension d’alimentation.

Schéma bloc interne au LTC6990:



Pouvant être alimenté entre 2.25 et 5.5 [V], le LT6990 est construit autour d’un générateur de fréquences « Maître » réglable en fonction du courant sortant de la pin SET. Une différence de potentiel d’1 [V] est fixée grâce à une contre réaction et permet à l’utilisateur d’ajuster la fréquence de cet oscillateur La fréquence maximale atteint le Mégahertz. Celle-ci peut ensuite être divisée jusqu’à 128 grâce à une tension fixée sur la pin DIV.

Voici la formule de calcul alliant la fréquence de sortie, RSET et le facteur de division :



La datasheet précise que le courant ISET doit se situer typiquement entre 1.25 et 40 [µA], avec un minimum à 500 [nA].

Puisque le clock à générer est de 800[Hz], j’ai directement choisi le facteur de division le plus lent soit 128.



RSET = 470 [kΩ] + POT de 20 [kΩ]

Le courant sortant de la pin SET sera approximativement de :

=> C’est suffisant !

Pour diviser la fréquence du clock par 128, le code de division doit être de 7 (0b0111). Le MSB sert à spécifier l’état de la sortie OUT lorsque OE est à l’état bas : OUT se retrouve mis à la masse si le MSB est à 0, dans le cas contraire, il sera à haute impédance.

Dans mon cas, je préfère avoir un état bas, même si cela ne semble pas changer énormément…

Le convertisseur A/D incorporé utilise la tension d’alimentation en tant que référence. En reliant un diviseur de tensions à la même alimentation, les variations de celle-ci n’auront pour ainsi dire aucune influence sur le facteur de division !

Calcul du diviseur :



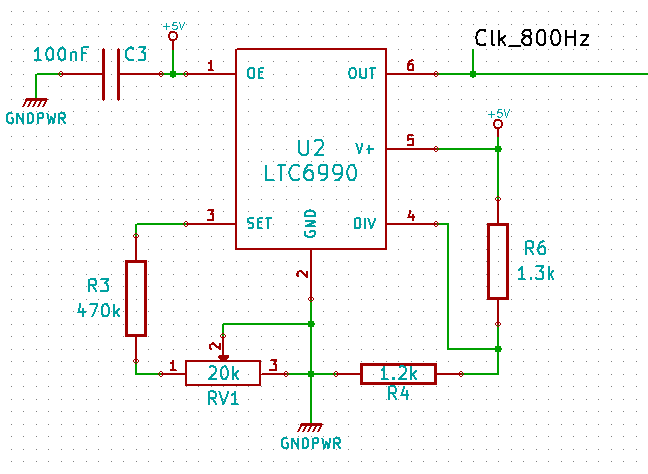


Constat : la tension UDIV devra se situer entre ces deux seuils. En cas de variations de celle-ci, je préfère me retrouver avec un code de division à 6 (divisant la fréquence du clock par 32, et qui, par conséquent induirait une fréquence de 400 [Hz]) plutôt qu’un code 8 (qui me fournirait un clock de 102.4 [kHz]). J’ai donc fixé le potentiel du diviseur à 2.4 [V] et la résistance R2 à 1.2 [kΩ].



Sur le schéma ci-dessous, R1 calculée correspond à R6 et R2 à R4.

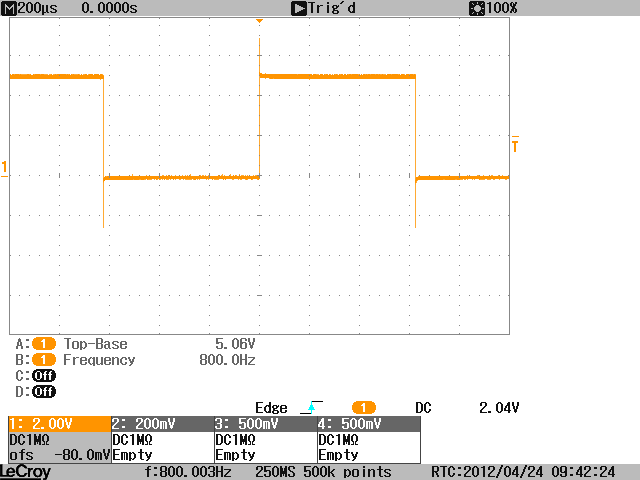
Schéma réalisé :



Mesure de la fréquence générée

La tension d’alimentation fût mesurée à 5.036[V] avec le FLUKE 175.

CH1 (Clk\_800[Hz]) correspond à la fréquence générée à 2 [V/Div]. Le canal est en DC avec un offset de -80 [mV]. Le trigger est synchronisé avec CH1 sur front montant à 2.04 [V]. La période de temps est de 200 [µs/Div].



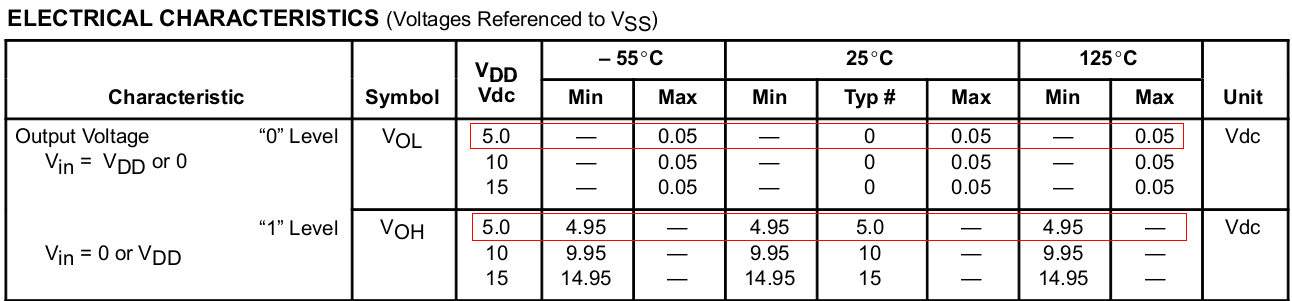
Constat : La fréquence peut-être réglée très précisément grâce au potentiomètre. Après avoir tenté plusieurs essais en faisant varier la tension d’alimentation entre 4.7 et 5.3 [V], la fréquence n’a pour ainsi dire pas changé.

### Compteur Johnson

Afin de commander les transistors dans l’ordre des étages (étage le plus bas) n°0 au n°7 (le plus haut), le plus simple est de me servir du légendaire CD4022.

Dans notre stock il ne nous restait que le MC14022CL (équivalent élaboré par Motorola). C’est donc celui-ci qui fût utilisé.

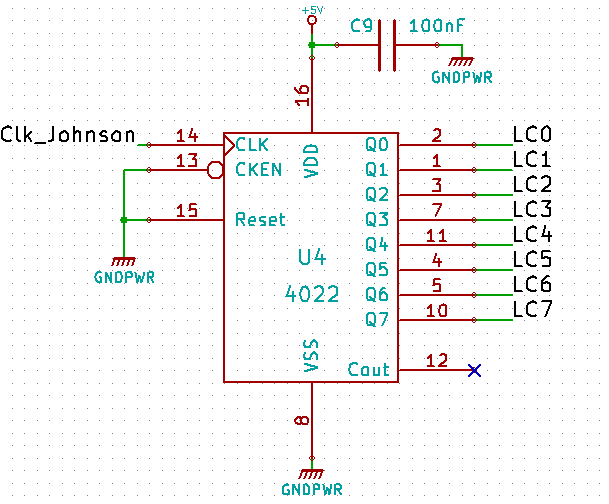
Tensions de sorties selon la datasheet du MC14022:



Celles-ci sont très proches de la tension d’alimentation et de la masse. Puisque la logique est alimentée en 5 [V], je vais être en mesure de définir le type de transistor à utiliser pour assurer la mise à la masse d’un étage (voir le chapitre 6.4.3) !

Ci-dessous, le clock généré à 800 [Hz] est présent sur « Clk\_Johnson », soit l’entrée CLK du 4022. Les sorties Q0 à Q7 vont passer successivement à 1 à chaque front montant du clock (avec Q0 à 1 au démarrage).

Schéma réalisé :

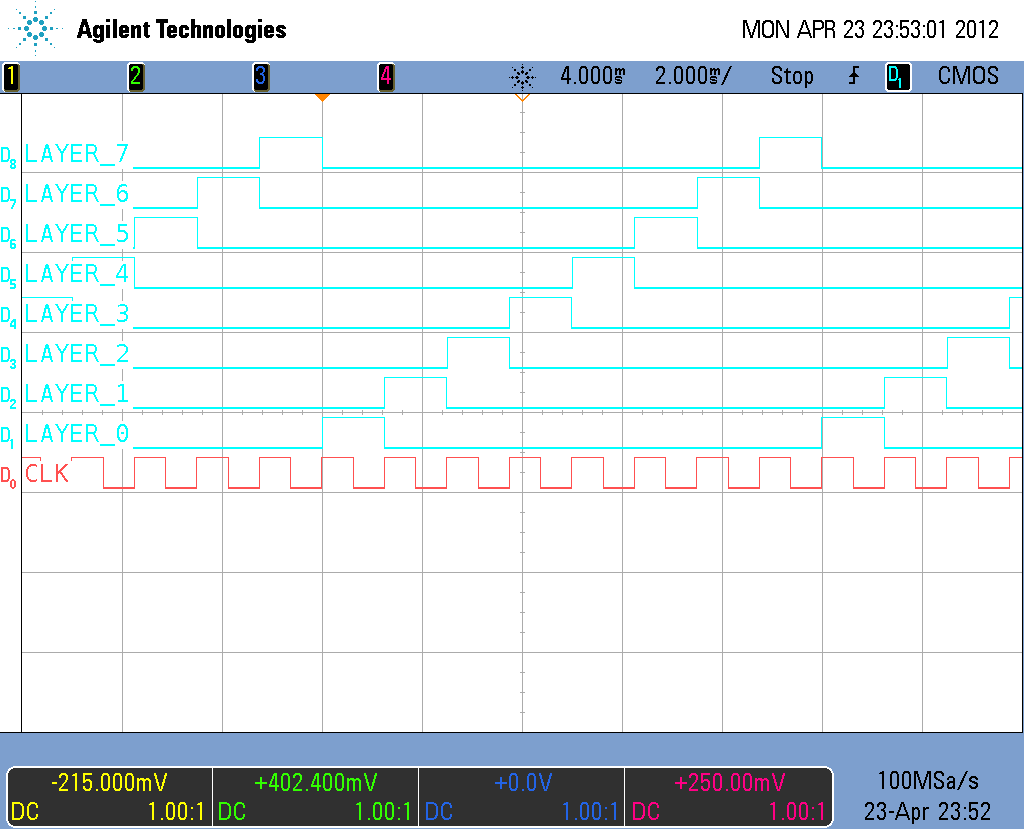


Note : l’abréviation LCn correspond à Layer Commande. La sortie CarryOut n’est pas utile dans cette application.

Mesure de la séquence résultante :

Ci-dessous, les sorties LAYER\_n sont directement celles du Johnson (sur le schéma LCn).

Le trigger est synchronisé avec Layer\_0 sur front montant pour des niveaux logiques CMOS. La période de temps est de 2 [ms/Div].



Constat : le comportement est bien celui attendu, les commandes d’étages passent successivement à l’état haut, le principe de balayage

### Transistors de puissance

Des transistors de technologie MOS ont été choisis puisqu’il me fallait une commande réalisable avec des niveaux logiques, tout en alliant une tension de déchet la plus faible possible !

Lors de la conduction (mise à la masse d’un étage), la valeur du courant instantané maximum sera de :



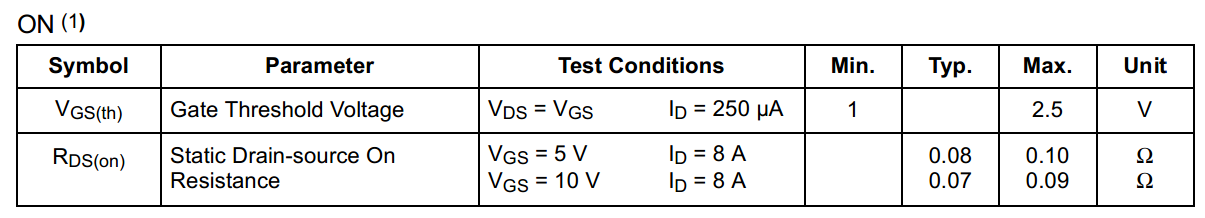
La valeur RMS sera :

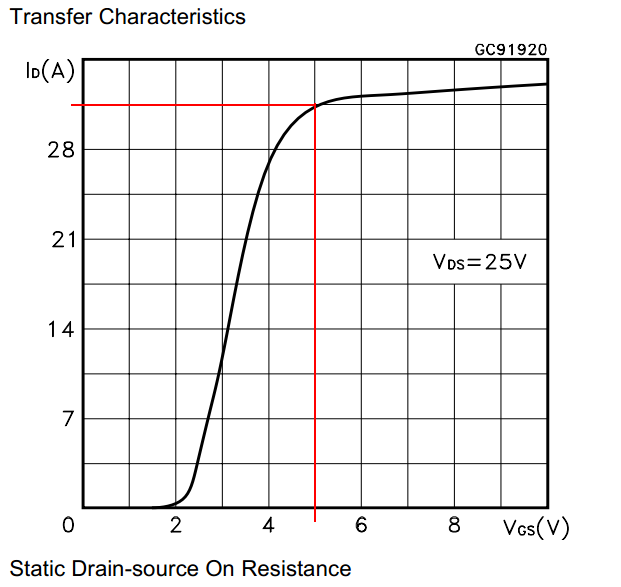


En cherchant dans le stock de l’école, j’ai trouvé le STP16NF06L !

Voici les indications utiles fournies par la datasheet :

Le transistor supporte au maximum : un VGSTH de ±16 [V], un courant IDcontinu de 16 [A] ainsi qu’une tension VDSmax de 60 [V].





Selon les tensions délivrées par le 4022 (voir chapitre 6.4.2), je constate qu’il n’y a aucun risque pour que le transistor conduise lors d’une commande à bas niveau (la tension VGS se situera en dessous d’1 [V]). Un niveau logique haut permettra aisément de faire passer le courant de 4.16 [A] (voir sur la courbe ci-dessus).

Par ailleurs, si je choisis le RDS(ON) maximum (de 0.10 [Ω]), je devrais obtenir une tension de déchet de :



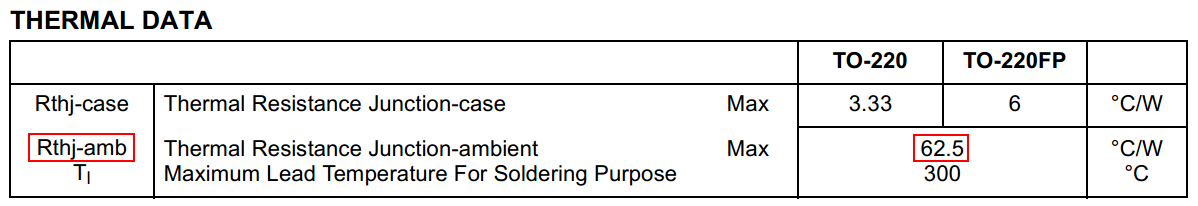
Calculons maintenant la puissance instantanée dissipée par le transistor :



Puis en RMS :



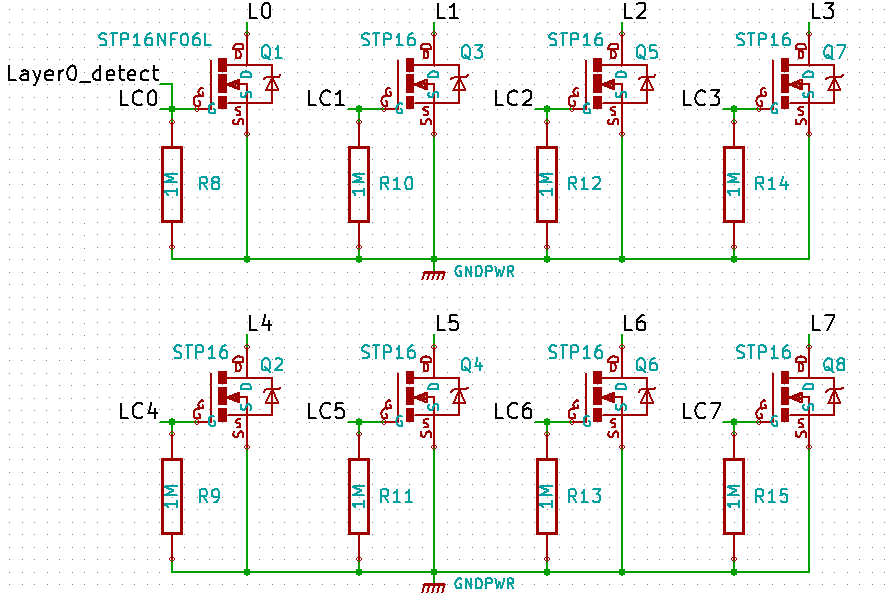
Calcul de la température de jonction à 616 [mW] pour une température ambiante de 30 [°C] :





Constat : il n’y aura pas besoin de radiateur !

Schéma de câblage :



Le drain de chaque transistor sera relié à la cathode commune de l’étage correspondant.

Afin de mesurer la tension de saturation réelle d’un transistor avant le test final, j’ai appliqué une tension continue VGS de 3.3 [V], puis de 5 [V]. Les étages furent remplacés par un rhéostat afin de contrôler le courant ID mesuré par le Fluke 175 SN : 99040601. La tension, quant à elle fût mesuré par le Fluke 175 SN : 97101217

Voici les résultats obtenus:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Avec VGS = 3.3 [V] | |  | Avec VGS = 5 [V] | |
| VDSsat [mV] | ID [mA] |  | VDSsat [mV] | ID [mA] |
| 2 | 10.49 |  | x | x |
| 4 | 50.54 |  | x | x |
| 8 | 110.4 |  | x | x |
| 24 | 315 |  | x | x |
| 31 | 404.3 |  | x | x |
| 73 | 947 |  | x | x |
| 180 | 2180 |  | x | x |
| 282 | 3117 |  | 205 | 3100 |
| 420 | 4300 |  | 330 | 4300 |

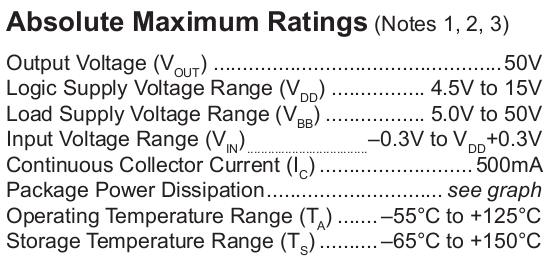
Conclusion : Finalement, comme la tension VGS utilisée est de 5 [V], VDSsat maximal se situera plutôt aux alentours de 330 [mV].

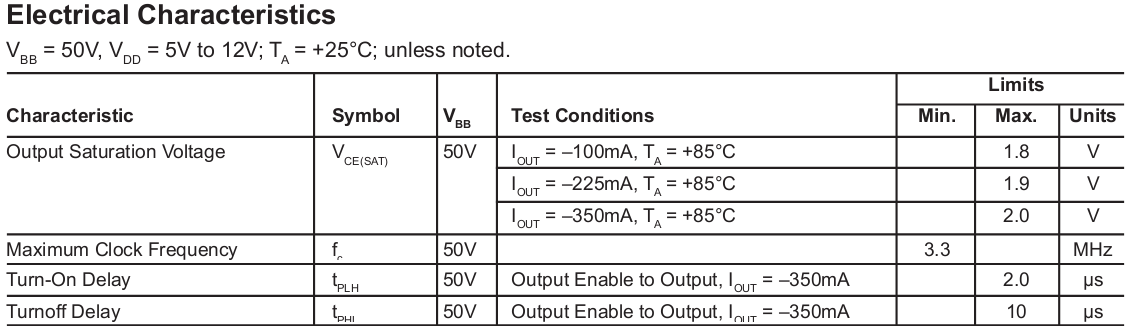
En ce qui concerne le temps de conduction du transistor, voici ce que la datasheet annonce : turnon à 10[ns], turnoff à 20[ns], un temps de montée de 37 [ns] et temps de descente de 12.5 [ns], ce qui est bien assez rapide !

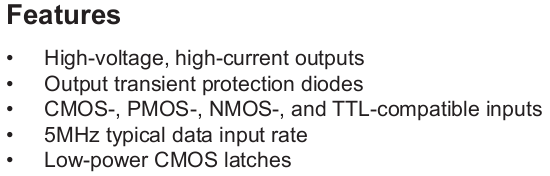
### Drivers sources

Après de longues recherches, il s’avère que le meilleur driver source pour cette application soit le MIC5891.

Il s’agit d’un registre à décalage muni de sorties de puissances capables de fournir chacune un maximum de 500 [mA].

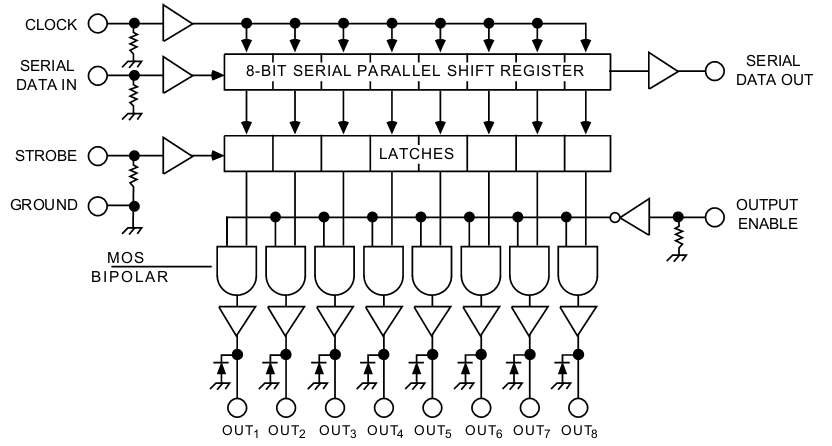




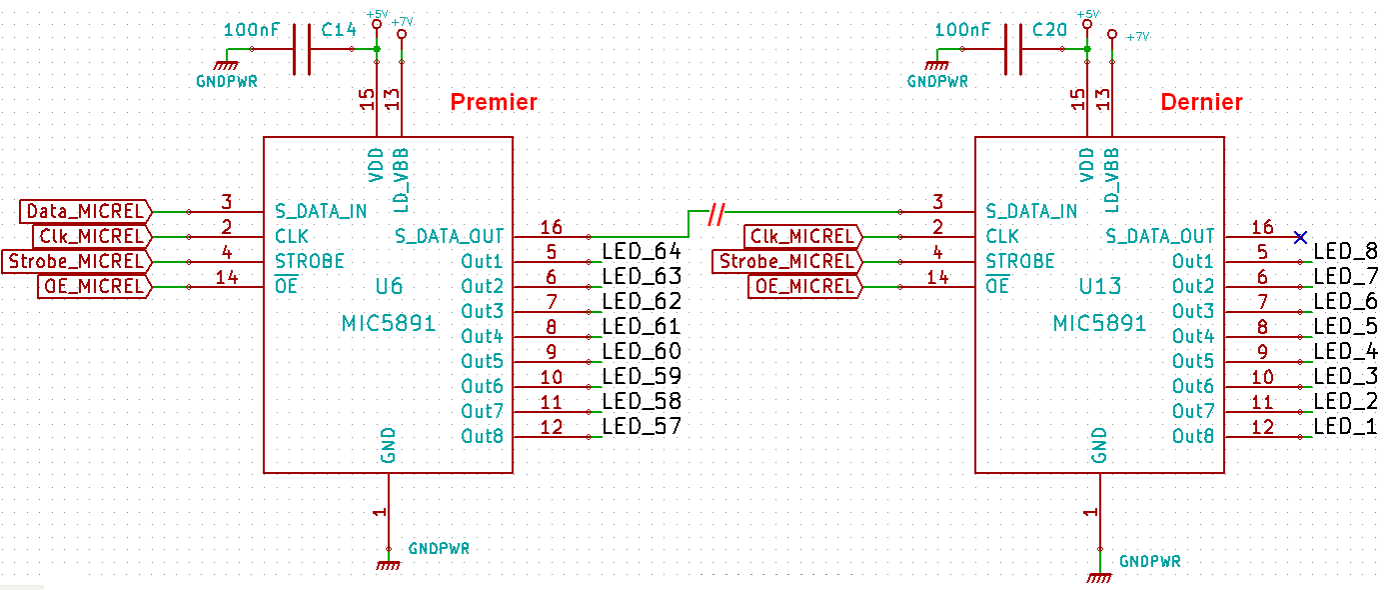


Les entrées sont compatibles TTL. Les niveaux de sortie du µC étant dans les pires conditions au minimum de 2.6 [V] pour VOH et au maximum à 0.6 [V] pour VOL, l’interfaçage sans adaptation est possible.

Voyons la structure interne :



Puisqu’il y a 64 LEDs à piloter, il faudra utiliser 8 MIC5891 enchainés comme il suit :



Les données seront envoyées via le µC sur le NET « Data\_MICREL ». Les autres pins de commandes et clock sont communs à tous les MIC5891.

Le premier byte envoyé agira sur la ligne 0 de leds. Le dernier byte, quant à lui pilotera la ligne 7 de l’étage en cours. (Rappel concernant la structure de l’étage au chapitre 3).

Pour plus de détails concernant la communication aux drivers, veuillez-vous référer au chapitre 6.7.3.

Calcul du courant nécessaire à l’alimentation des 8 LEDs :

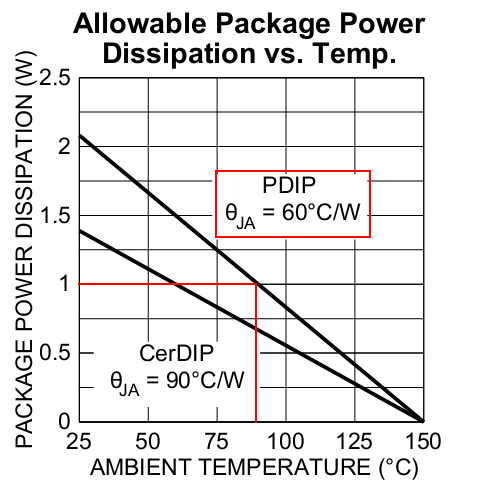


Calcul de la puissance dissipée :

D’après la datasheet, la tension de déchet par sortie serait typiquement égale à 1.8 [V] lors d’un courant de sortie de 100 [mA].



D’après la courbe de puissance ci-dessous, (avec de la marge), il serait possible d’atteindre une température ambiante d’environs 88 [°C] avant de griller le composant.



Conclusion : Il n’y a pas de souci de refroidissement de ce côté.

### Limitation du courant traversant les LEDs

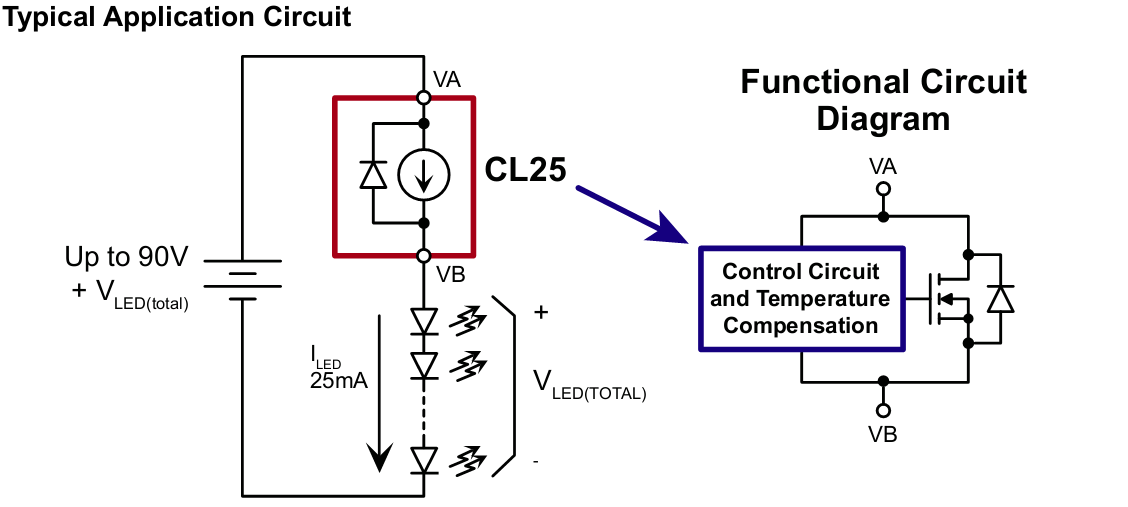
Pour limiter le courant, il existe plusieurs solutions plus ou moins onéreuses.

La plus simple consiste en l’utilisation de résistances séries entre les sorties des MIC5891 et les anodes des LEDs. Le désavantage de cette manière de procéder réside dans le fait que selon les tensions de déchet du MOSFET de commande d’étage en fonction du nombre de LEDs allumées, la tension aux bornes des résistances variera et ainsi affectera la valeur du courant !

La deuxième solution serait d’utiliser des diodes limitatrices de courant tels que la CL20B ou CL25. L’une des compagnies spécialisée dans ce type de composants, se nomme « Supertex inc. ». Il peut-être intéressant de découvrir quelques autres de leurs produits tel que CL6 qui permet une limitation à 100 [mA].

Note : Les diodes telles que CL20, CL25… peuvent-êtres mises en parallèle.

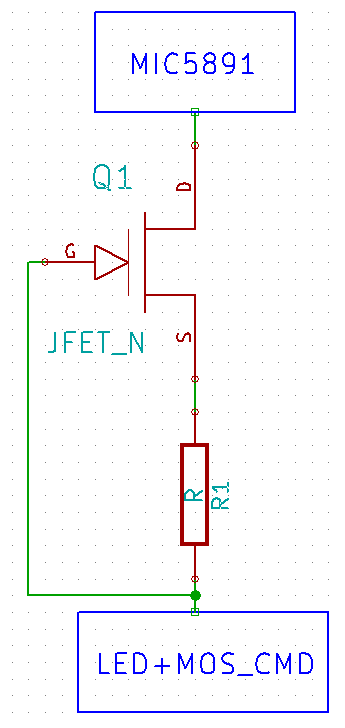
Ci-dessous, l’application typique du composant :



Comme je n’ai pas trouvé le même type de composant capable de limiter, cette fois-ci un courant d’environ 60 [mA], la solution aurait-été l’utilisation de trois CL20B en parallèle, soit un total de 192 pièces pour 64 LEDs !

Malheureusement encore un autre paramètre rentre en compte. Celui du prix ! En effet, chez « DigiKey », l’achat des 192 pièces reviendrait à près de 170 CHF !

Une autre solution que je n’ai pas essayée aurait été de réguler le courant (un peu comme à la manière des diodes CLxx) grâce à un JFET canal N :



En fonction des caractéristiques du transistor, du courant à limiter et de la valeur de la résistance R1, la tension VGS se créera lors de la conduction et le montage se stabilisera.

Finalement, la solution optée reste l’utilisation de résistances séries. Puisque le courant RMS spécifié à 23 [mA] par LED est relativement élevé, une baisse de quelques [mA] n’est pas gênant pour l’œil humain !

Calcul de la résistance nécessaire :

La tension aux bornes de la résistance de limitation est fixée à 2 [V].

 => normalisée à 30 [Ω].

Le courant sera donc de :



Et la puissance :



Il serait possible d’utiliser des résistances SMD au format 1206 puisqu’elles supportent généralement ¼[W].

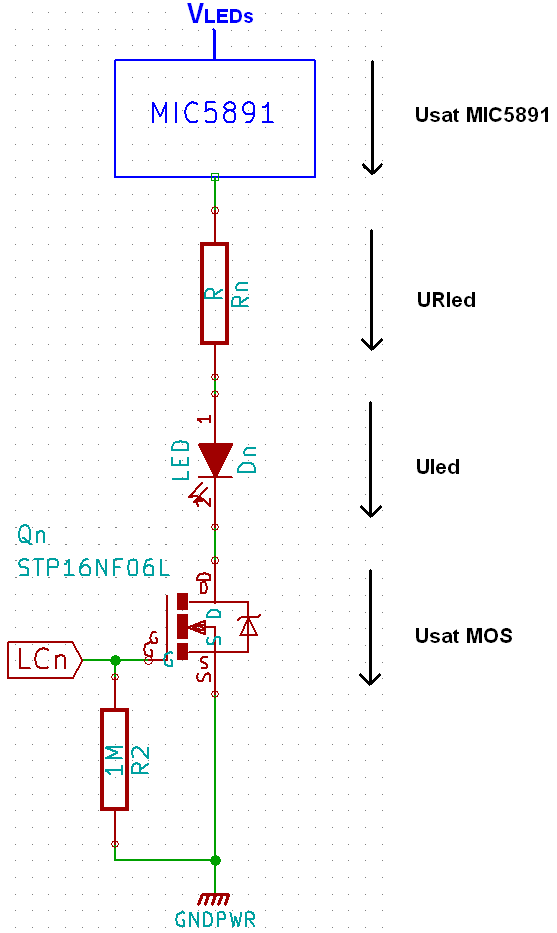
### Spécifications des LEDs

Le modèle conçu par Heibei IT (560LB7D) procure une lumière bleue diffuse. La tension UF est spécifiée entre 2.8 et 3.4 [V]. L’angle de réflexion quant à lui se situe entre 60 et 70 [°], ce qui procurera un meilleur effet visuel.

### Choix de la tension d’alimentation de puissance

Rappel : La tension administrée aux LEDs (VLEDs) proviendra directement de l’alimentation externe et sera connectée sur VBB des MIC5891.

Chaine d’’alimentation :



D’après les valeurs théoriques abordées précédemment, voici le calcul de la tension nécessaire à l’alimentation externe :



Note : La tension aux bornes de URLED fût spécifiée à 2 [V] et USAT\_MOS à 0.1[V]. Le choix de cette faible valeur de saturation permettra de calculer la résistance pour un courant proche du « maximum ». Bien sûr toutes ces valeurs ont une certaine tolérance. Les différentes mesures effectuées en pratique vérifieront la justesse des calculs.

Conclusion : Avec 7 [V], le la tension Vdrop sera suffisante pour utiliser le régulateur de tension 5 [V] (Type LM340T-5). Il faudra tout de même prévoir une alimentation externe avec une faible tension d’ondulation de charge afin d’éviter de descendre en dessous de 6.7 [V] (voir chapitre 6.3).

### Dimensionnement des câbles et pistes

Rappels concernant les courants (valeurs RMS):

Le courant d’alimentation pour toutes alimenter toutes les LEDs équivaudrait à 4.16 [A] (voir chapitre 6.1).

Le courant RMS maximum en sortie d’un étage serait d’1.47 [A] (calculé au chapitre 6.4.3).

L’alimentation pour chaque MIC5891 demanderait au maximum 520 [mA] (calculé au chapitre 6.4.4).

Calcul des sections minimales de câblage :

La section du fil reliant les étages à la carte de commande peut-être calculée selon la formule suivante :



Rappel : 

Section pour alimenter les anodes :

L = 40 [cm] I = 65 [mA] Chute de tension = 30 [mV]



L’idéal serait de se trouver au-dessus de cette valeur, ce qui sera le cas puisque la connexion aux anodes se fera grâce à un câble plat formés de conducteurs de type AWG26 (0.128 [mm2])

Section pour la sélection d’étages :

L = 40 [cm] I = 1.47 [A] Chute de tension = 30 [mV]



Les câbles utilisés auront une section de 0.5 [mm2].

Section pour l’interrupteur :

L = 40 [cm] I = 4.16 [A] + marge pour alimentation logique de 30 [mA]

Chute de tension = 30 [mV]



Le câble aura une section d’1.5 [mm2]

Calcul de la largeur des pistes :

Pour me simplifier la tâche, le logiciel « PCB trace calculator » fût utilisé afin de calculer la taille des pistes.

Vous pourrez vous en servir à l’adresse suivante : <http://www.4pcb.com/trace-width-calculator.html>

Dans son principe, le programme calcule l’aire de la piste selon l’épaisseur du cuivre, la température et l’ampérage. Finalement, la largeur est calculée.

L’épaisseur du cuivre fût fixée à 35 [µm]. La température quant à elle à 10 [°C] (par sécurité). A savoir qu’une diminution de celle-ci implique une augmentation de la largeur calculée.

Piste principale :

Son utilité sera de fournir le courant aux drivers MIC5891 ainsi qu’au régulateur 5 [V].

Le courant nécessaire pour un allumage complet du cube est de 4.16 [A].

La largeur minimale fût calculée à 5.58 [mm]. Une marge fût prise (5.8 [mm]) afin de baisser la résistance et assurer le passage d’un courant supplémentaire suffisant pour la logique et l’ajout de futurs périphériques.

Pistes secondaires pour les drivers sources :

Les drivers étant reliés par groupement de trois (au maximum), le courant à travers la piste équivaudra 1.56 [A] .

Largeur de la piste secondaire calculée à 1.44 [mm].

La sous-piste servant à relier chaque MIC5891 à la piste secondaire fût calculée à 0.317 [mm].

Pistes destinées aux transistors de commandes des étages :

Avec 1.47 [A], la largeur obtenue égale 1.33 [mm].

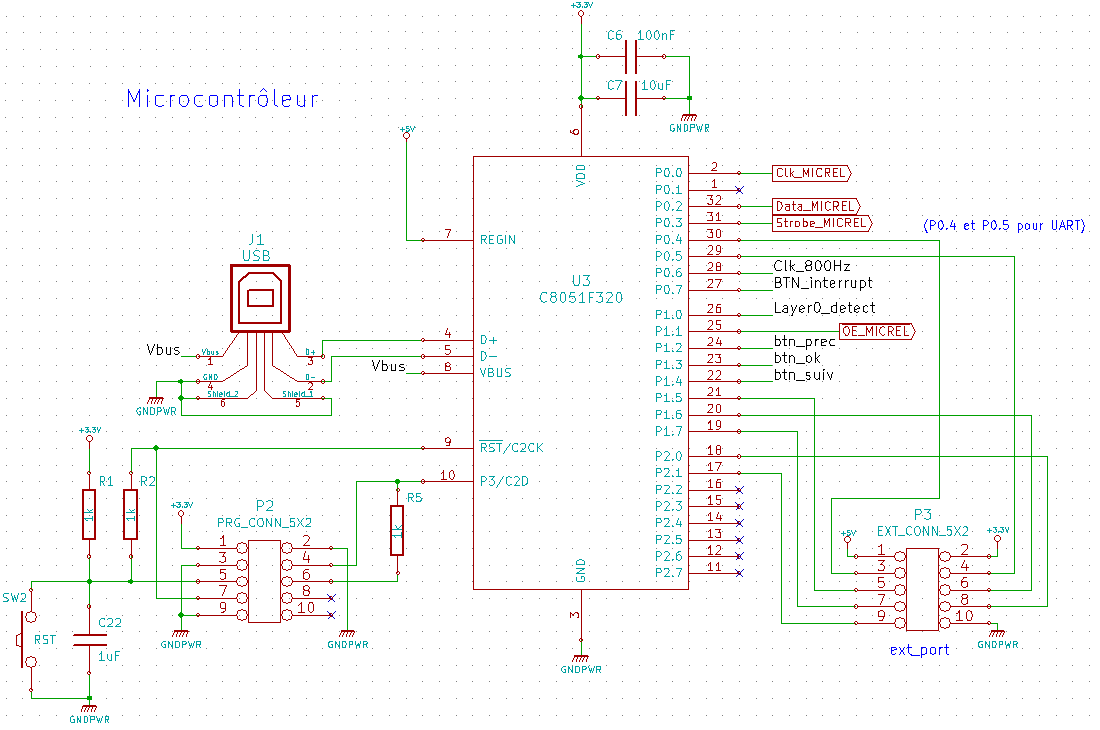
Remarque : toutes les tailles utilisées sur le circuit imprimé dépassent les valeurs calculées.

## Analyse du spectre audio

Par manque de temps, la partie Hardware concernant l’analyse des fréquences du spectre audio n’a pas pu être réalisée dans le cadre du diplôme. Cependant grâce à un port d’expansion disponible sur le PCB, cette amélioration reste possible !

## Connexions au microcontrôleur

### Schéma général



Pour la gestion de l’affichage « Clk\_800Hz » est de type INT0 front montant, tandis que « BTN\_interrupt » est sur INT1 front montant. En vue de futures améliorations, un connecteur 10 pôles (P3) permettant l’ajout de périphériques (manette de jeu etc..) fût implémenté.

### Boutons poussoirs

3 boutons poussoirs de type « microswich » permettront la sélection de l’animation à afficher ou le mode de fonctionnement du cube.

Le bouton tout à gauche « suivant » permet d’incrémenter le menu. « Précédent » permet de le décrémenter, et enfin « ok » confirmera la sélection effectuée. Pour revenir au menu de sélection, il suffira de presser sur l’un des trois.

Schéma :



Lorsqu’un un bouton est pressé, la sortie de la porte passe à 1. Les trois boutons sont reliés à trois entrées du µC. « BTN\_interrupt » est reliée sur une entrée avec INT1 sur front montant (P0.7).

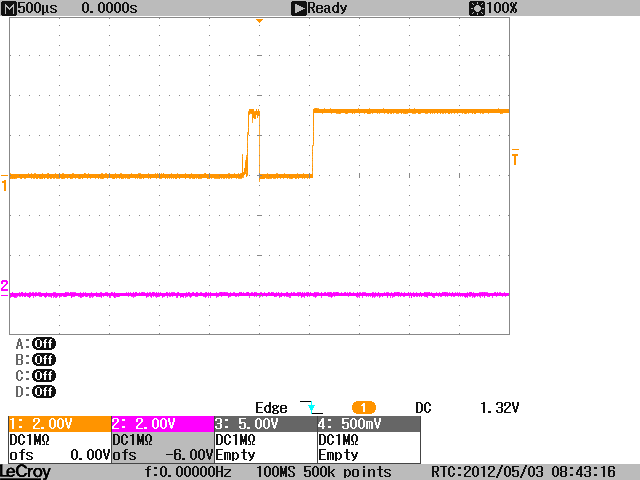
Une fois le programme entré dans l’interruption, il suffira de vérifier l’état des boutons.

Remarque : Les pull-up sont internes au microcontrôleur. La valeur du condensateur de 100[nF] est inspirée du montage du KIT F320 fourni par « Silabs ».

Mesure des rebonds sans condensateur :

CH1 (btn\_ok) correspond à l’état du bouton lors d’un relâchement à 2 [V/Div], CH2 ne représente rien.

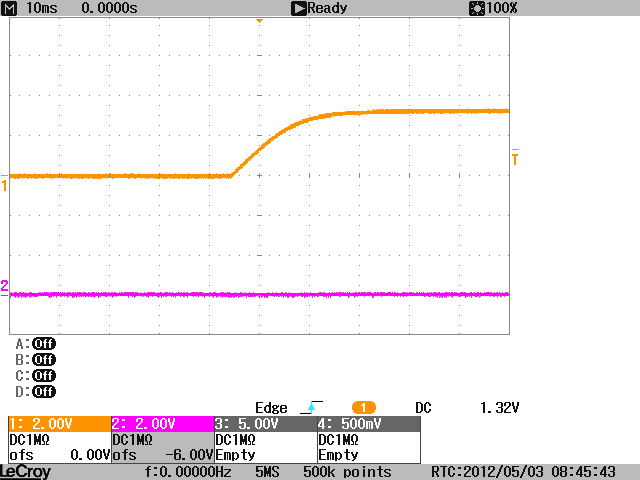
Les canaux sont en DC avec le trigger synchronisé avec CH1 sur front descendant à 1.32 [V]. La période de temps est de 500 [µs/Div].



Constat : les rebonds sont visibles !

Mesure avec le condensateur :

L’oscilloscope à la même configuration que précédemment, sauf que la base de temps est à 10 [ms/Div].



Constat : Charge du condensateur visible en lieu et place des rebonds. La solution est viable !

## Programmation µC bas niveau

Premièrement, puisque l’USB sera utilisé, le clock système du microcontrôleur est obligatoirement configuré à 24 [MHz].

Pour afficher une image, un rafraichissement des données à afficher doit être effectué à chaque changement d’étage. C’est au µC que cette tâche sera confiée.

Les données étant envoyées aux drivers via spi, il y aura un total de 8 envois à effectuer pour envoyer les 64 bytes totaux formant l’étage.

Rappel : le premier byte envoyé correspondra à l’état des LEDs de la ligne 0 de l’étage concerné.

Le mécanisme de rafraichissement s’effectuera à travers une interruption sur front montant « INT0 » qui se chargera d’interrompre le programme lors d’un changement d’étage (NET Clk\_800Hz).

### Organisation du cube en mémoire

Dans un premier temps, je voulais réaliser un tableau à trois dimensions [8][8][8]. La valeur contenue à l’emplacement spécifié représenterait l’intensité lumineuse de la LED en question. Après discussion avec M. Tutt, cette solution prendrait trop d’espace en mémoire et serait trop longue à réaliser. Finalement par simplification, un tableau à deux dimensions représentera le cube.



Le byte sélectionné ci-dessous représente l’état des LEDs sur la ligne 0 de l’étage n (vue au dessus du cube) :



Constat : La valeur obtenue égale 0x43. Un « 1 » allume la LED, 0 l’éteint.

### Rafraichissement de l’affichage

Le mécanisme de rafraichissement de l’affichage ira lire les informations contenues dans le tableau à deux dimensions de manière périodique (à 800 [Hz]).

Afin de réaliser un changement d’image propre (donc pas de rafraichissement pendant la mise en place de données dans le tableau), l’utilisation d’un bit d’état, de deux pointeurs ainsi que de deux tableaux (à deux dimensions) fût requise.

//démarrage sur le tableau 1 (si à 0, ce sera le tableau 2)!

bit flag\_buffcube1 = 1;

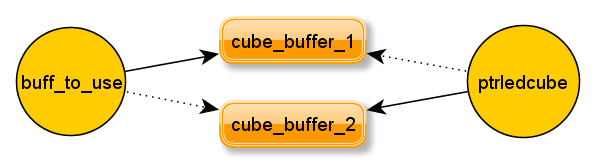
xdata unsigned char (\*ptrledcube)[8]; //ptr pour affichage mémoire du cube!

xdata unsigned char (\*buff\_to\_use)[8]; //pointeur sur le tableau à utiliser //en tant que buffer

xdata unsigned char cube\_buffer1 [8][8]; //tableau de traitement

xdata unsigned char cube\_buffer2 [8][8]; //tableau de traitement

Note : à l’initialisation, le pointeur « buff\_to\_use » pointe sur l’adresse du tableau «  cube\_buffer1 ».



Une fonction nommée « Tab\_to\_use() » se servant du flag «  flag\_buffcube1 » fût écrite afin de mettre à jour le pointeur « buff\_to\_use ». Celle-ci se devra d’être appelée avant un besoin d’écriture en mémoire.

Exemple avec l’affichage d’une nouvelle image:

Tab\_to\_use(buff\_to\_use,cube\_buffer1,cube\_buffer2);//choix du buffer à utiliser

anim\_base\_up(buff\_to\_use); //animation de base montante!

while(memory\_lock); //Attendre libération de la mémoire ptrledcube = buff\_to\_use; //mise à jour du pointeur d’affichage

flag\_buffcube1^=1; //Avertissement de changement de buffer

Le système d’envoi se charge de placer un flag « memory\_lock » à 1 pendant la transmission des informations aux drivers (opération de 60.5 [µs]), ce qui permettra de ne pas mettre à jour le pointeur d’affichage en même temps.

Organigramme de l’interruption INT0



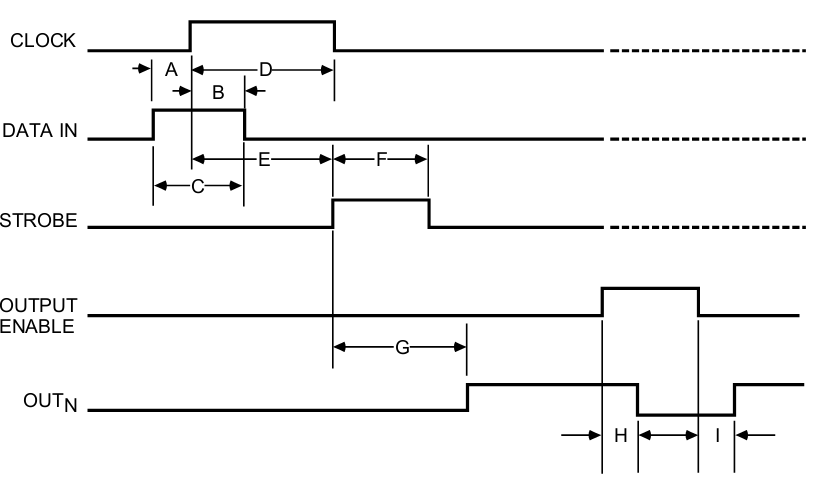
Organigramme de l’interruption SPI



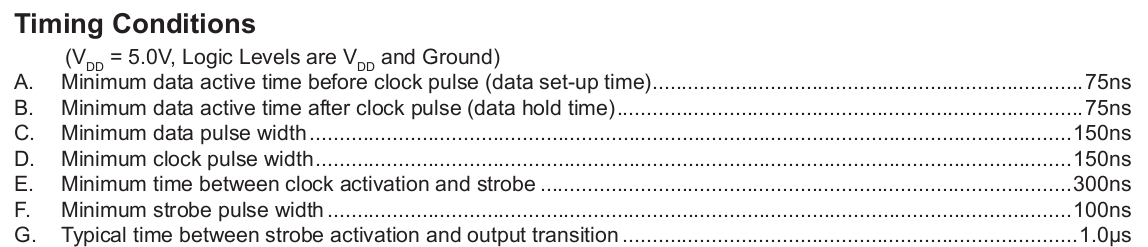
### Transmission de données aux drivers sources (MIC5891)

Comme vu lors du chapitre concernant les drivers sources (6.4.4), la transmission de données s’effectuera dans un seul sens (µC => MIC5891).

En termes de trames de données, voici ce que présente la datasheet du MIC5891 :

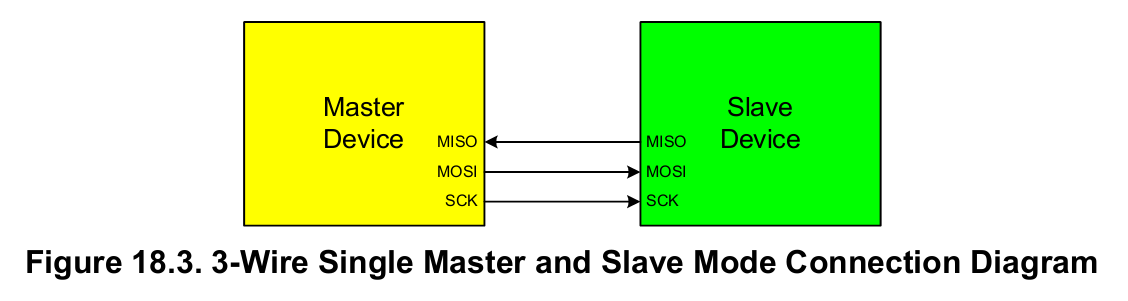


Les timings à respecter sont les suivants :



Le microcontrôleur utilisé au cours de ce travail possède une interface spi, interface qui sera utilisée pour l’occasion !

Celle-ci peut-être configuré de plusieurs manières et la configuration la plus appropriée semble être le mode « single master » 3 fils. En effet, le µC ne recevra pas d‘ordre de la part des drivers.



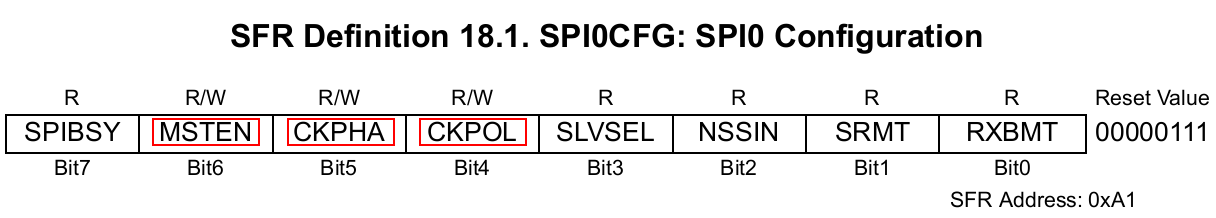
Le clock et les données (SCK et MOSI) transmettront les informations aux drivers sources tandis que les pins « Strobe » et « Output Enable » seront gérés manuellement. Puisqu’inutile dans cette application, la pin MISO sera laissée non connectée.

Concernant la fréquence du clock « SCK », le minimum garanti par les drivers étant fixé à 3.3 [MHz], j’ai choisi une fréquence de 3 [MHz], qui me semble suffisante puisque théoriquement, l’envoi de 8 bytes (64 bits) s’effectuera en :



Bien entendu, cela ne prends pas en compte les temps d’accès aux interruptions ainsi qu’aux instructions d’envoi.

La configuration du SPI au sein du microcontrôleur s’effectue au moyen des registres suivants :

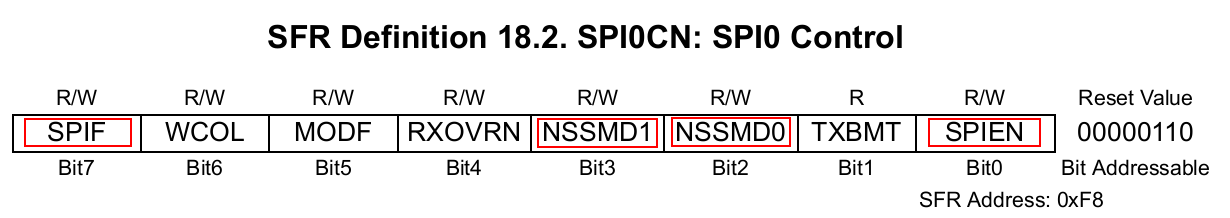


MSTEN = sélection du mode (1 pour master, 0 pour slave).

CKPHA = Phase du SCK (0 pour front montant, 1 pour front descendant).

SCKPOL = niveau du clock au repos (0 pour low, 1 pour high).

Le registre SPI0CFG sera configuré à 0x40 !



Les bits 7 à 4 se mettent à 1 lors d’un événement SPI, ce qui abouti à une interruption.

Afin de se rendre compte si l’interruption fût générée par une fin de transmission de byte, le bit SPIF devra être testé.

NSSMD1 et NSSMD0 = sélection du mode de connexion :

0 – 0 = mode 3 fils, NSS n’est pas relié au port.

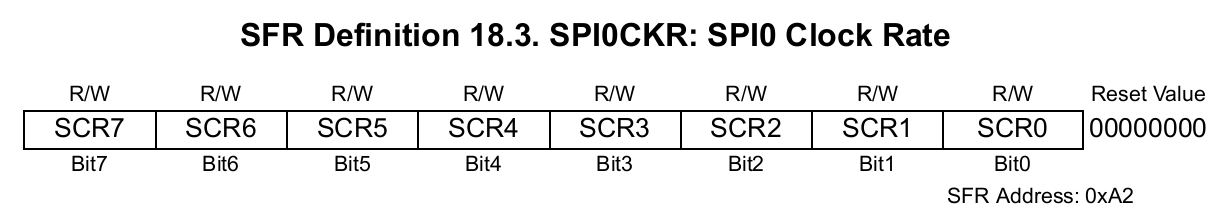
0 – 1 = mode 4 fils (slave ou multi master) NSS est en entrée.

1 – x = mode 4 fils (single master) NSS est en sortie et prendra la valeur de NSSMD0

SPIEN = mise en fonction du module interne SPI (1 = activation, 0 = désactivation).

Le registre SPI0CN sera configuré à 0x01 !

Le registre SPI0CKR permet de définir la vitesse de SCK :



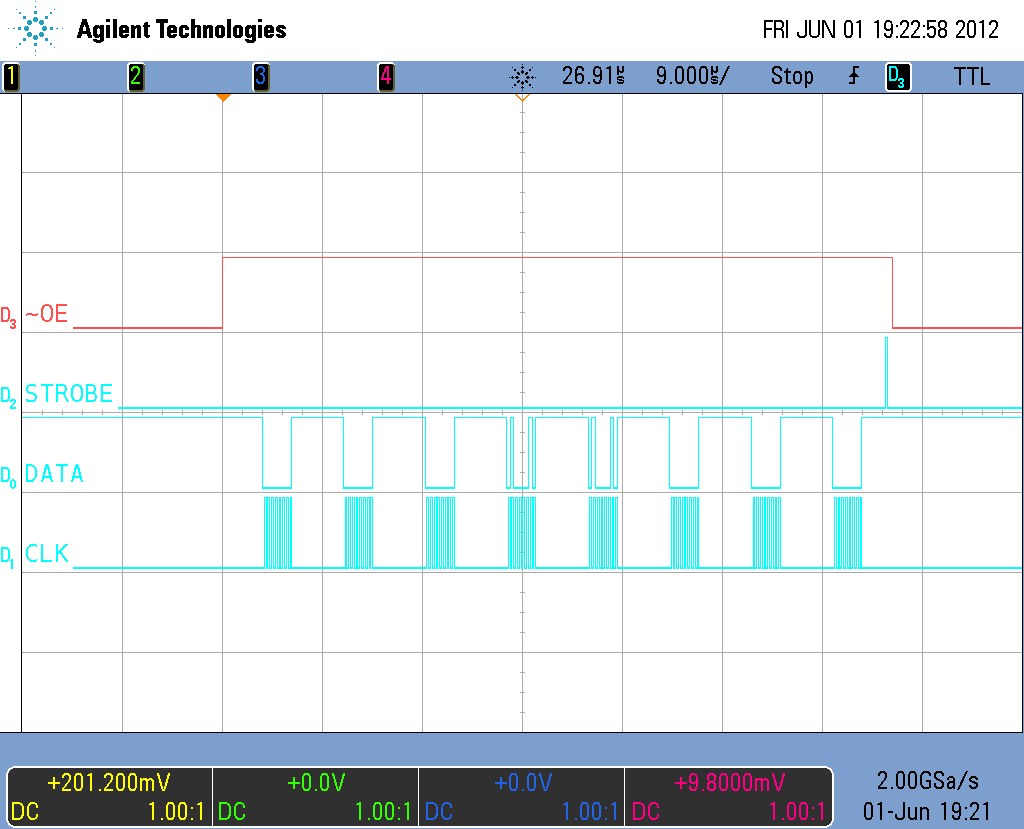
Formule de calcul :



Le registre SPI0CKR sera chargé à 0x03.

Analyse logique d’un envoi de 8 bytes :

Le trigger est synchronisé avec le clock sur front montant pour des niveaux logiques CMOS. La période de temps est de 20 [µs/Div].

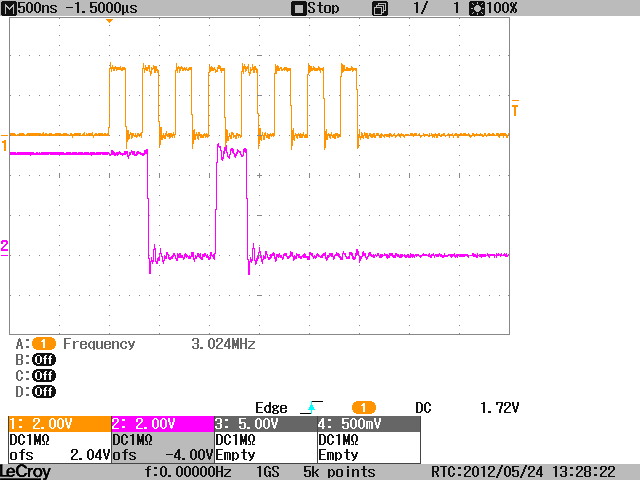


Constat : La forme de l’envoi des données est correcte !

Mesure de la fréquence du clock SPI :

CH1 correspond au clock SPI (P0.0 du µC) à 2 [V/Div], CH2 représente les données (P0.2).

Les canaux sont en DC et le trigger est synchronisé avec CH1 sur front montant à 1.72 [V]. La période de temps est de 500 [ns/Div].

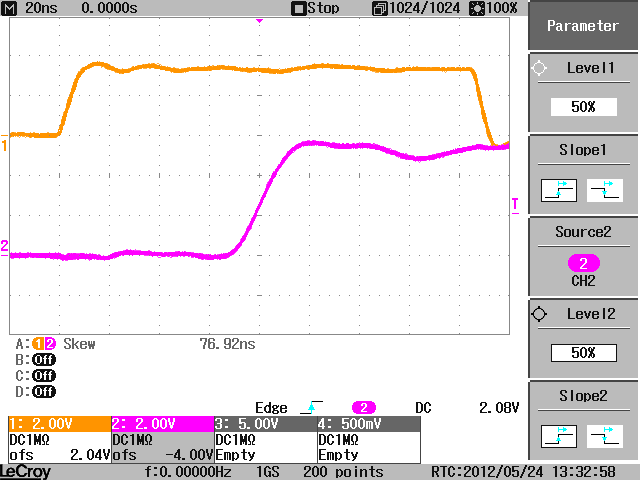


La fréquence est proche de la valeur désirée, 3.024 [MHz] à la place de 3 [MHz].

Mesure du temps entre le clock et la préparation de la nouvelle donnée:

CH1 correspond au clock SPI (P0.0 du µC) à 2 [V/Div], CH2 représente les données (P0.2).

Les canaux sont en DC et le trigger synchronisé avec CH2 sur front montant à 2.08 [V]. La période de temps est de 20 [ns/Div].

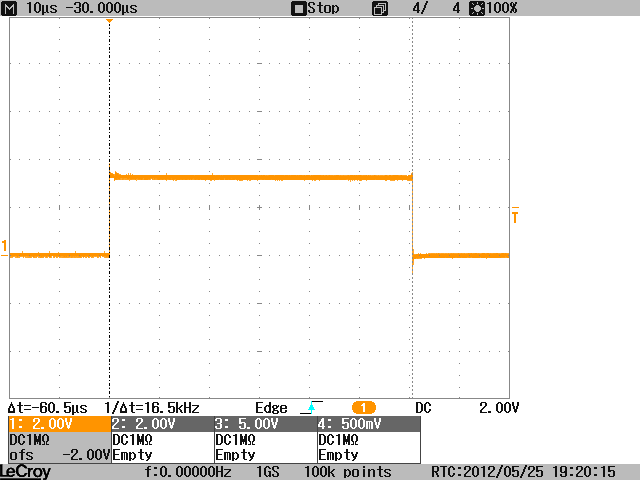


Le temps « data hold time » (Timing B de la datasheet MIC5891) est correct : 76.92 [µs], ce qui respecte le temps minimal de 75 [µs]. Le phénomène est similaire avec le temps « data set-up time ». Les temps en rapport avec le « Strobe », furent également concluants.

Mesure du temps d’envoi réel :

Afin de mesurer le temps réel d’envoi des 8 bytes, la pin P2.0 du microcontrôleur fût mise à « 1 » pendant ce laps temps.

CH1 (P2.0) correspond à la pin de test à 2 [V/Div]. Le canal est en DC avec le trigger synchronisé avec CH1 sur front montant à 2 [V]. La période de temps est de 10 [µs/Div].



Constat : l’envoi de 64 bits prend 60.5 [µs], c’est bien plus que les 21.3 [µs] calculé précédemment, mais cela reste normal car il faut prendre en compte les temps d’accès aux interruptions.

### Correction du problème de persistance rétinienne

A la suite des premiers tests d’affichage d’une image, une certaine persistance fût observée sur les étages supérieurs. Exemple illustrant la problématique : lorsqu’une LED éteinte (sur l’étage 7) se retrouvait au dessus d’une autre allumée (étage 6), la LED éteinte était en réalité légèrement éclairée.

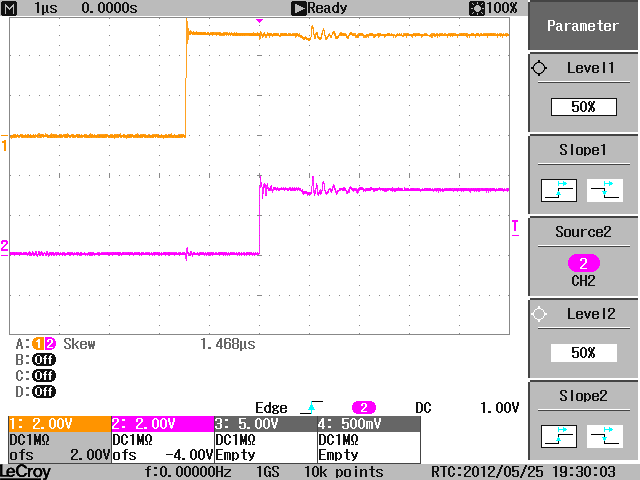
Après quelques recherches et mesures, j’ai remarqué que cela provenait du temps que le µC prenait avant d’entrer dans l’interruption INT0. A chaque changement d’étage, le nouveau transistor commutant à « 1 » avait de l’avance la mise à « 0 » des sorties des drivers sources. Ainsi l’image inférieure était visible sur l’étage supérieur.

Mesure du temps d’accès à l’interruption :

La pin P2.0 est mise à 1 pendant l’envoi des données aux MIC5891.

CH1 correspond au clock de 800 [Hz] à 2 [V/Div], CH2 représente P2.0 à 2 [V/Div].

Les canaux sont en DC, le trigger est synchronisé avec CH2 sur front montant à 1 [V]. La période de temps est de 1 [µs/Div].



Constat : cela prend 1.468 [µs] avant d’accéder à la première instruction de l’INT0.

La solution mise en place pour corriger le problème fût de rajouter un circuit de retard entre la sortie du générateur de fréquence à 800 [Hz] et l’entrée clk du compteur Johnson (4022).

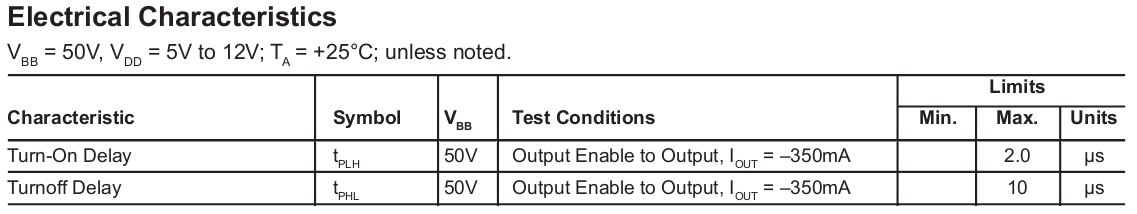
Le microcontrôleur serait directement connecté au générateur et aurait une avance sur l’information reçue par le compteur en anneau.



La méthode de retard la plus simple consiste en un simple circuit « RC ». La datasheet du 4022 indique qu’il n’y a pas de limite avec les temps de montée et descente du clock d’entrée. Ceci rend la solution possible sans l’ajout de porte (à base trigger de Schmidt par exemple).

Pour calculer le temps de retard sécurisé, il est reste encore un paramètre à prendre en compte : le temps d’action de la pin OE des drivers sources sur leurs sorties.

Voici ce que la datasheet présente :



Le temps d’extinction des sorties égal 10 [µs].

Si je néglige le délai d’action du 4022 d’1 [µs] maximum (entre le clock qu’il reçoit et le changement des sorties) ainsi que le temps de conduction des transistors de sélection d’étage, le calcul sera le suivant :



Le niveau logique « 1 » en sortie du circuit de retard sera perçu aux 2/3 de la tension d’alimentation logique (5 [V]). Cela tombe bien puisque c’est à cette tension que le niveau haut CMOS est détecté. La formule de charge du condensateur se limite donc à la simple charge à 1 [T].

Un pourcentage de marge (30%) est ajouté au temps TRC par sécurité (marge de précision du condensateur).



Le condensateur est fixé à 10 [nF].

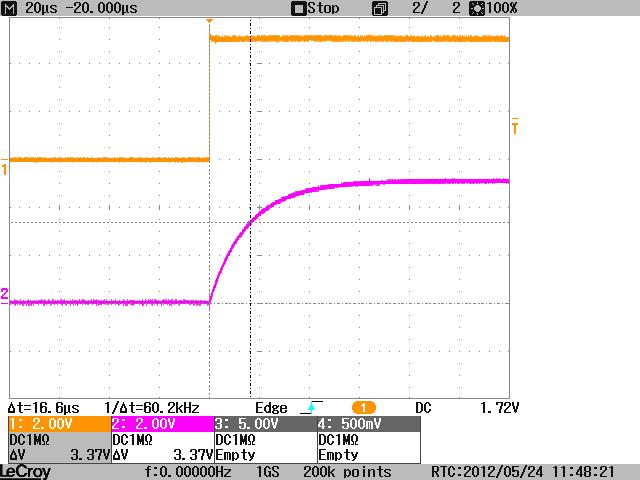
=> Normalisée à 1.5 [kΩ].

Avec la valeur normalisée, le délai devrait être de 15 [µs].

Mesure du temps de charge à 1 [T] :

CH1 correspond au clock 800 [Hz] à 2 [V/Div], CH2 représente le clock allant au 4022 à 2 [V/Div].

Les canaux sont en DC avec le trigger synchronisé avec CH1 sur front montant à 1.72 [V]. La période de temps est de 20 [µs/Div].

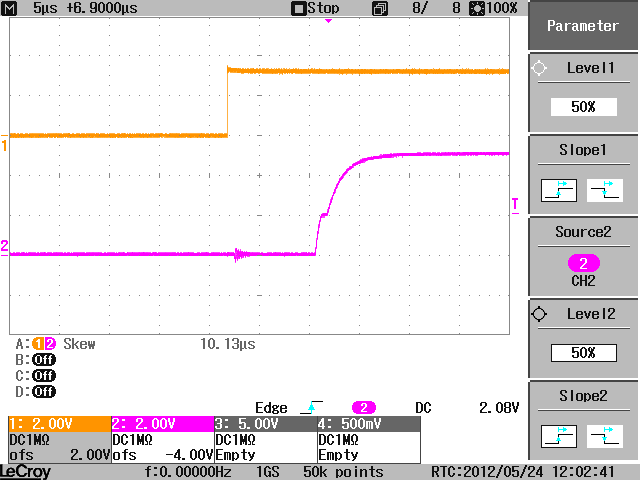


Constat : le temps de charge du condensateur à 1 [T] prend 16.6 [µs], c’est proche de la valeur calculée (tolérance des composants).

Mesure du nouveau temps entre l’accès à l’INT0 et le changement d’étage :

CH1 correspond à P1.2 (ou OE\_MICREL) à 2 [V/Div] et CH2, la commande de conduction du MOSFET de l’étage en question.

Les canaux sont en DC, le trigger est synchronisé avec CH2 sur front montant à 2.08 [V]. La période de temps est de 5 [µs/Div].



Constat : une fois entré dans l’interruption (et donc désactivé les drivers sources), le laps de temps avant la nouvelle commande du MOSFET est de 10.13 [µs].

Lors de la synchronisation entre le n° de l’étage et le clock de 800 [Hz] (effectuée au démarrage du programme, le microcontrôleur devra attendre plus de 10.13 [µs] avant de lire l’état de la « Layer0\_detect » (P1.0). Le problème de persistance est résolu !

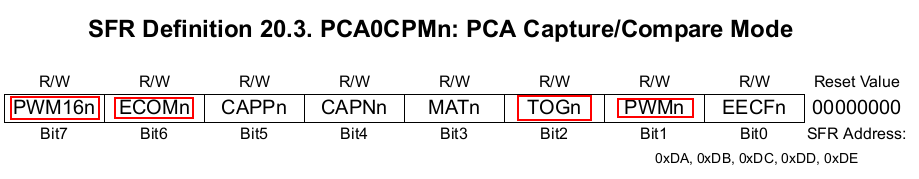
### Gestion de la luminosité

Dû à la limitation de l’organisation du cube en mémoire, un PWM commun sera pour toutes les leds.

Le microcontrôleur possède un PCA (Programmable Counter Array) lui-même possédant 5 modules (Capture/Compare). Il est possible de se servir de l’un d’entre eux en mode PWM, ce qui me simplifiera la tâche en termes de programmation et n’occupera que très peu le processeur.

Le PWM sera câblé en interne à la pin P1.1 (OE des drivers sources) grâce à CEX0 du PCA0. Il suffira de désactiver le PCA0 et de déconnecter CEX0 de la pin physique lors de l’envoi des données pour de ne pas interférer avec le mécanisme de rafraichissement.

La configuration du PWM s’effectue selon les registres suivants :



PWM16n = sélection du mode de comptage (1 pour 16 bits, 0 pour 8 bits).

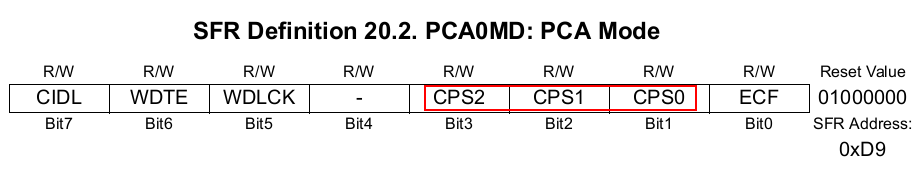
ECOMn = activation du comparateur module n du PCA0 (actif à 1).

TOGn = mode génération de fréquence si PWM actif (actif à 1).

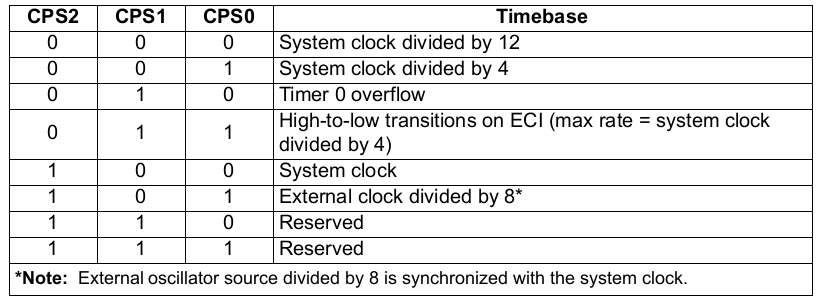
PWMn = enclenche le mode PWM (actif à 1)

Un PWM de 8 bits suffira amplement pour gérer le niveau de luminosité.

Le registre PCA0CPM0 sera configuré à 0x42 !



CPS2 à CPS0 = choix du clock PCA0:



Dans mon cas, j’ai choisi de laisser la prédivision à 12, ce qui implique un clock à 2 [MHz]. Puisque le PCA0 est configuré en mode 8 bits, la base de temps du rapport cyclique sera de 128 [us].

Le nombre de PWM durant le temps de conduction de l’étage sera de :

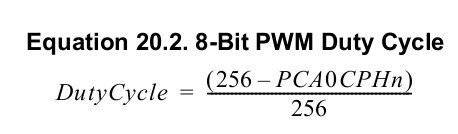


Remarque : Le dernier PWM sera interrompu.

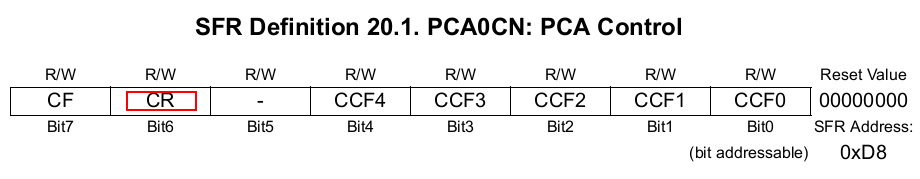


Grâce au registre ci-dessus, le rapport cyclique du PWM pourra être modifié.

Voici l’équation de calcul du rapport :



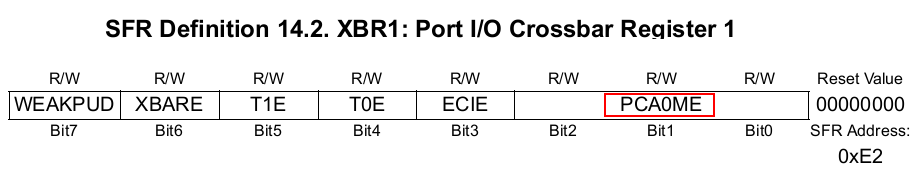
Remarque : La variation s’effectuera entre 100 et 0.39 %. Force est de constater que puisque la commande OE des MIC5891 est active à 0. Les LEDs pourront être éteintes mais jamais totalement allumées durant le laps de temps où aucune communication avec les drivers n’a lieu. Heureusement, il existe un moyen de forcer le PWM à 0% en désactivant le comparateur du module 0 via le bit ECOM0 du registre PCA0CPM0. Mais cependant, il faut faire attention au fait qu’une écriture dans le PCA0CPH0 réactivera automatiquement le comparateur.



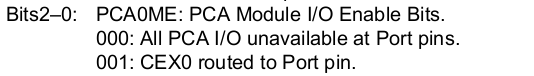
CR permet d’activer/désactiver le module PCA0 (actif à 1).

PCA0CN sera configuré à 0x40 !

Ce bit qui sera mis à 0 lors de l’enclenchement du système de rafraichissement.



Le périphérique PCA0 peut-être relié à un pin physique via CEX0 via PCA0ME.



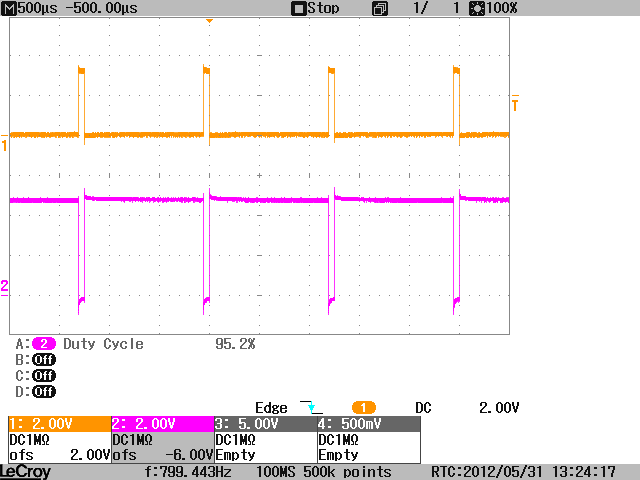
A chaque nouvel envoi de données aux drivers, PCA0ME sera mis à 000.

Le reste du temps il sera à 001.

Mesure du PWM à 100 %:

CH1 (OE\_MICREL ou P1.1) correspond à l’état de commande des sorties envoyé par le µC. CH2 représente la tension en sortie de la pin 5 (LED\_64) du MIC5891 U6.

Les canaux sont en DC à 2 [V/Div] avec le trigger synchronisé avec CH2 sur front descendant à 2 [V]. La période de temps est de 500 [µs/Div].



Constat : A cause de la communication aux drivers, le PWM atteint un maximum de 95.2%. L’erreur est de 4.8%.

Mesure du PWM à 60 %:

Même configuration que précédemment, sauf que la période de temps est de 200 [µs/Div].

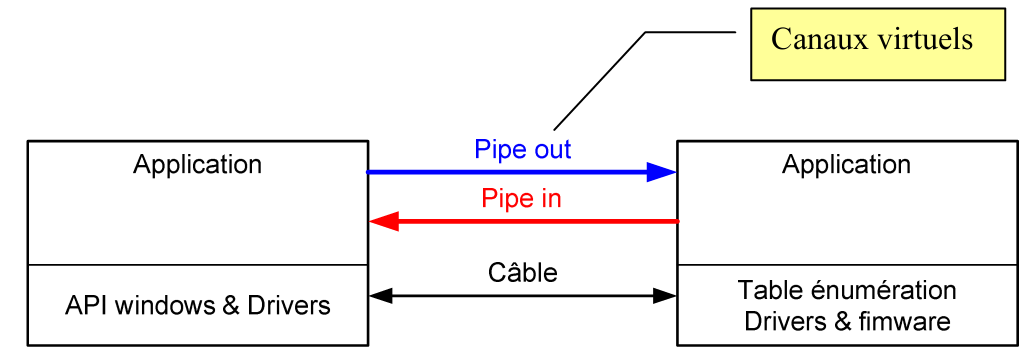


Constat : Avec 60.3 % à la place de 60, l’erreur est faible (0.5%).

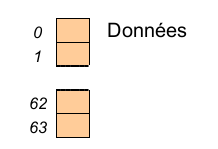
Conclusion : Le mécanisme de gestion de la luminosité est opérationnel et le résultat visuel est convaincant. La plus grande erreur entre la consigne et le résultat obtenu intervient lors de d’une demande de luminosité maximale (4.8%).

### Communication USB

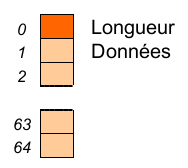
L’USB à la caractéristique d’être configurable dans plusieurs modes nécessitant parfois des drivers. Afin de rendre la connexion à un PC enfantine, le mode HID (déjà utilisé dans le cadre des cours) sera utilisé. Il ne nécessite pas de drivers et grâce aux librairies fournies par M. Neuhaus, il sera configuré pour transmettre les données par paquets de 64 bytes utiles (+ 1 byte servant à indiquer la longueur du paquet) avec un temps de scrutation garanti à 10 [ms].



Concernant la gestion des trames, du côté du microcontrôleur, le byte indiquant la longueur du paquet est transparent, le tableau à donc une taille de 64 :



Du point de vue API Windows, le tableau à une taille de 65 :



Note : les images ci-dessus sont extraites du « Principe du Protocole (cfpt-usbhid) » réalisé par M.Neuhaus.

Pour plus d’informations concernant l’utilisation du firmware « cfpt-usbhid » veuillez-vous référer aux documents de M.Neuhaus.

L’application PC sera en mesure de transmettre des images et des valeurs de luminosités au cube.

Le protocole de communication USB est le suivant :



Une fois le mode « PC » sélectionné dans le menu du cube, celui-ci envoie la trame « #ready$ ». A partir de cet instant, l’application PC est en mesure d’envoyer une valeur PWM ou une image à afficher via le biais d’une trame de commande, puis d’une trame de données.

En ce qui concerne la valeur du PWM, celle-ci est transmise sous le format ASCII.

Le flot de données étant de 64 bytes, une image complète (8 étages de 8 bytes chacun) peut-être transmise en une seule étape.

L’organisation des données de l’image complète s’effectue, des lignes 0 à 7 mises à la suite pour chaque étage (de l’étage 0 à 7) :





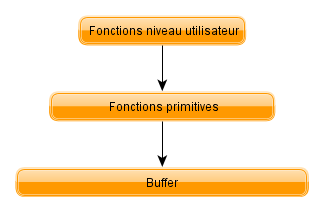
# Logiciels

## Programme du microcontrôleur

### Introduction

Les fonctions d’interruptions abordées précédemment (chapitre 6.7.2) sont pour ainsi dire presque au niveau « Hardware » puisqu’elles n’agissent que sur la transmission des données d’un tableau en mémoire aux drivers sources ainsi que sur le rafraichissement du PWM.

Dans le but simplifier la vie de l’utilisateur lorsqu’il désirera gérer l’état des LEDs dans l’espace tridimensionnel, il est nécessaire de réaliser diverses fonctions situées à différents niveaux :



Les fonctions du niveau utilisateur permettront, de manière relativement simplifiée l’écriture d’un caractère, d’effectuer une animation, etc…

Le niveau primitif effectuera des opérations sur un des tableaux à deux dimensions servant de buffer. Exemple : Décalages pour recréer un effet de déplacement, mise à 0 du tableau, etc...

L’avantage d’une imbrication des fonctions par niveau résulte par une optimisation du code et d’organisation. Certaines fonctions du niveau utilisateur pourront combiner diverses primitives. Exemple : La fonction « anim\_pluie() » dans son premier mode de fonctionnement effectue une image aléatoire sur l’étage 7 et crée un défilement vertical via la primitive « Depl\_hauteur() ».

Le programme principal aura pour tâche la gestion du déplacement dans le menu selon le bouton appuyé, l’affichage du menu ou de l’animation en cours, puis finalement, si le mode PC a été sélectionné, la gestion des trames reçues via USB.

### Organigramme de la boucle principale (Main)



Dans la version actuelle du programme, le « Main » effectue la gestion de l’emplacement dans le menu, la gestion de l’affichage et la gestion du mode USB sans appeler les blocs sous forme de fonctions. Ces opérations devraient être séparées en fonctions afin de rendre une meilleure lisibilité et transportabilité.

### Liste des fonctions primitives

La plupart des ces fonctions ont la particularité d’avoir leur propre algorithme de traitement d’un tableau, qu’il soit monodimensionnel ou bidimensionnel.

void rotat\_h(unsigned char ptr\_ledcube[][8],unsigned char ptr\_newcube[][8], bit sens);

void Depl\_profond(unsigned char ptr\_newcube[][8], bit sens);

void Depl\_largeur(unsigned char ptr\_newcube[][8], bit sens);

void Depl\_hauteur(unsigned char ptr\_newcube[][8], bit sens);

void tabtoface(unsigned char ptr\_tab[],unsigned char ptr\_newcube[][8],unsigned char face\_nb);

void tabtolayer(unsigned char ptr\_tab[],unsigned char ptr\_newcube[][8],unsigned char etage\_nb);

void Copy\_cube(unsigned char tab\_src[][8],unsigned char tab\_dst[][8]);

void Clear\_cube(unsigned char tab[][8]);

void Or\_cube(unsigned char tab\_src[][8],unsigned char tab\_dst[][8]);

void Wr\_coord\_cube(coord \*ptr\_coord,bit state,unsigned char ptr\_tab[][8]);

bit Rd\_coord\_cube(coord \*ptr\_coord,unsigned char ptr\_tab[][8]);

### Liste des fonctions niveau utilisateur

void Cube\_car(unsigned char ptr\_newcube[][8],unsigned char face\_nb ,char caract);

void anim\_pluie(unsigned char ptr\_newcube[][8], unsigned char anim\_nb);

void anim\_base\_l\_r(unsigned char ptr\_newcube[][8], bit new);

void anim\_base\_up\_dn(unsigned char ptr\_newcube[][8], bit new);

void anim\_base\_up(unsigned char ptr\_newcube[][8]);

void luminosite(unsigned char lum);

## Application PC

### Introduction

Le langage de programmation utilisé lors du développement de l’application fût le C# accompagné de l’environnement « Visual Studio ».

Il y existe diverses manières de transmettre une animation. La première à laquelle j’ai pensé consistait à envoyer directement toutes les images de l’animation à afficher ainsi que les paramètres de luminosité et temporisation au microcontrôleur. Cela aurait demandé l’ajout d’une mémoire (flash par exemple) externe puisque pour accéder que cela soit en lecture ou en écriture à la flash du microcontrôleur, il est obligatoire de désactiver les interruptions de manière générale (bit EA), ce qui implique de nouvelles limitations dans la gestion de la mémoire…

Au vu du temps restant pour ajouter cette fonctionnalité, j’ai dû me remettre à une idée plus simple : celle où le PC allait lui, garder les informations à afficher en mémoire et les transmettre une à une au cube. Dans ce mode, le µC ne fait que d’attendre la trame lui indiquant une réception d’image ou de PWM suivi de la donnée disponible dans une seconde trame.

### Manuel d’utilisation de l’application « Cubizator »

Connexion du cube:

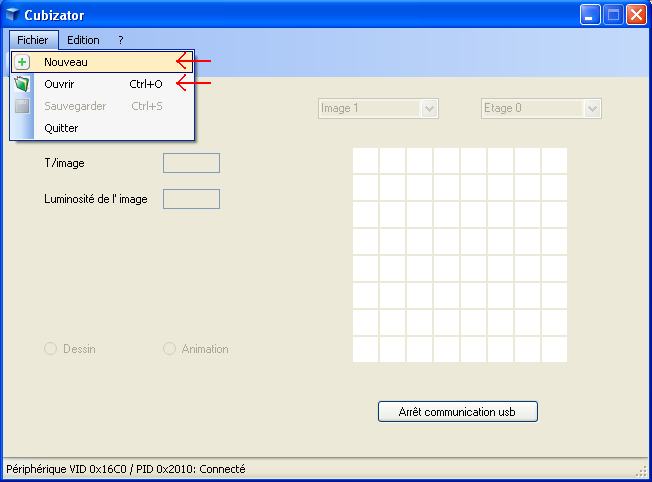
* Brancher le cube au PC grâce au câble USB.
* Sélectionner le mode PC depuis le cube afin qu’il se retrouve en mode communication.

Note : Une fois le mode PC actif du côté cube, celui-ci attend les informations via USB. Il devient alors impossible de revenir au menu de sélection tant que l’autorisation n’a pas été donnée par le logiciel PC.

Une fois l’application « Cubizator » exécutée et le cube connecté en mode PC, la fenêtre d’avertissement suivante devrait apparaître :

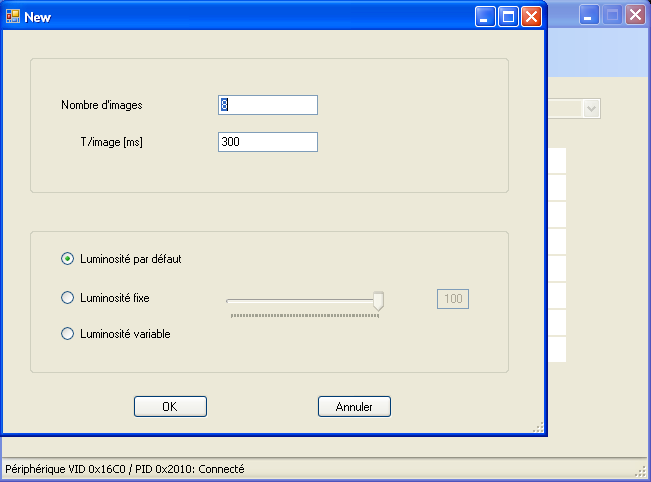


Vous allez maintenant être confronté à la fenêtre suivante :

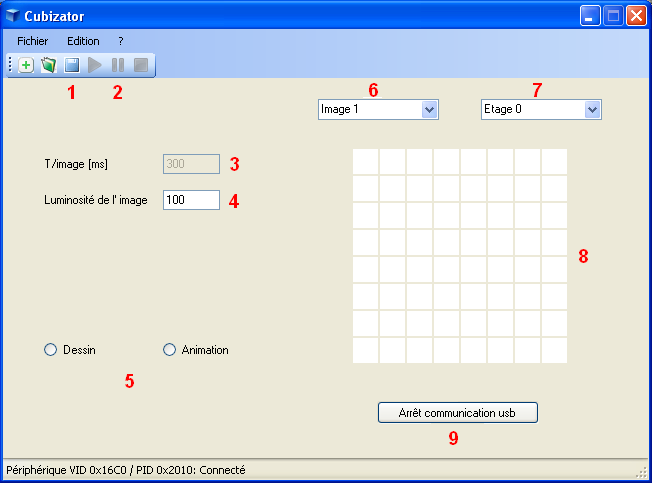


Dès lors, deux choix vous sont imposés : créer une nouvelle animation ou ouvrir un fichier d’animation existant.

Dans le cas d’une demande de nouvelle animation, la fenêtre ci-dessous se présentera :



A partir de maintenant, le programme vous invite à définir les spécificités de l’animation. Une fois les réglages effectués et une validation par le bouton « OK », la fenêtre principale se retrouve modifiée :



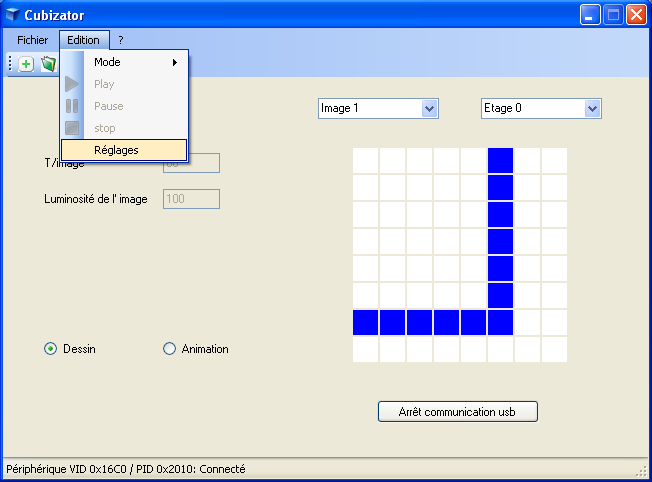
Description des éléments :

1. Enregistrer l’animation en cours dans un fichier situé à l’emplacement que vous aurez choisi.
2. Boutons de contrôle de l’animation (disponibles en mode animation).
3. Permet d’avoir un œil sur la temporisation entre chaque image.
4. Si le mode luminosité variable fût sélectionné, c’est dans cette textbox que vous pourrez choisir la luminosité de l’image en cours. Le cas échéant, la valeur de la luminosité fixe sera affichée.
5. Sélection du mode dessin ou animation. Le mode dessin affiche les données de l’image en cours sur le cube et permet ainsi de dessiner directement sur celui-ci. Le mode animation masque la zone de dessin et laisse apparaître l’icône « play » permettant le lancement de l’animation.
6. Sélection de l’image, c’est ici que vous sélectionnerez l’image à éditer/afficher.
7. Sélection de l’étage à afficher sur la zone de dessin (n°7).
8. Zone de dessin représentant le cube avec une vue de dessus. Un clic gauche allume la LED sélectionnée par la case alors qu’un clic droit l’éteint.
9. Ce bouton vous permet de stopper le dialogue avec le cube, ce qui donnera à nouveau contrôle au menu de celui-ci (un message d’information sera affiché).

Remarque : avec les éléments de sélection n°5 et 6, une astuce pratique (surtout lors de la sélection d’images en mode dessin) consiste à utiliser la molette de la souris pour visualiser un aperçu de ce que donnera leur enchainement.

Si à un moment donné, vous désirez modifier les propriétés de votre animation (nombre d’image, temporisation ou luminosité), une option prévue à cet effet est située dans le menu d’édition. La fenêtre affichée sera similaire à celle vous invitant à créer une nouvelle animation.

Attention : si l’utilisateur décide de réduire le nombre d’images, celles excédentaires seront supprimées sans préavis !



Note : cette option n’est pas disponible lorsque le mode animation est sélectionné.

### Diagrammes UML de l’application

































### Organisation des fichiers .cube

Le fichier est segmenté en différentes parties :

La première consiste en un en-tête présentant les caractéristiques de l’animation (en ASCII). S’ensuit les données images mises à la suite les unes après les autres selon le même procédé d’envoi que par l’USB. Le dernier champ n’est autre que la (ou les) luminosité(s) sous forme de byte(s).



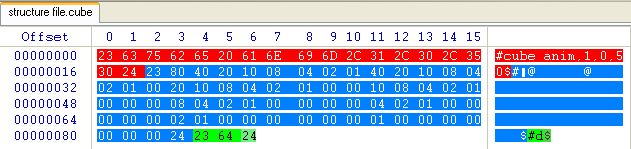
Note : le mode de luminosité peut prendre les valeurs suivantes : ‘0’ = mode par défaut ; ‘1’ = fixée ; ‘2’ = variable.

Rappel concernant la constitution d’un étage :



Par convention, chaque champ commence par le caractère ‘#’ et se termine par ‘$’. Même si cela n’est pas très utile lors de la navigation dans le fichier lorsqu’il s’agit de bytes d’images (d’un point de vue logiciel), vous pourrez constater que lors d’une analyse de fichier via un éditeur hexadécimal, il devient beaucoup plus facile de s’y retrouver.

Voici l’analyse grâce au logiciel « WinHex » d’un fichier représentant l’animation d’une image avec luminosité fixe :



En rouge se trouve l’en-tête du fichier pour 1 image, mode de luminosité par défaut avec une temporisation par image de 50 [ms]. En bleu, les 64 bytes de données de l’image1, puis en vert, la valeur de la luminosité (ici à 100%).

# Résultats (schémas et mesures)

## Schéma de la carte

Le schéma étant réalisé sur des feuilles au format A3, celui-ci se trouvera en annexe.

## Mesures du courant LED

Les mesures ci-dessous se référeront au schéma suivant :



Note : La plupart des oscillogrammes ci-dessous ont une mesure RMS volontairement affichée en plus de la valeur instantanée. En effet, celle-ci peuvent donner à réflexion !

### Mesure du courant avec la R79 de 30 [Ω]

Mesure des tensions d’alimentation au multimètre Fluke 175 SN : 97101217

VLEDs = 7.05 [V]

VCC = 5.034 [V]

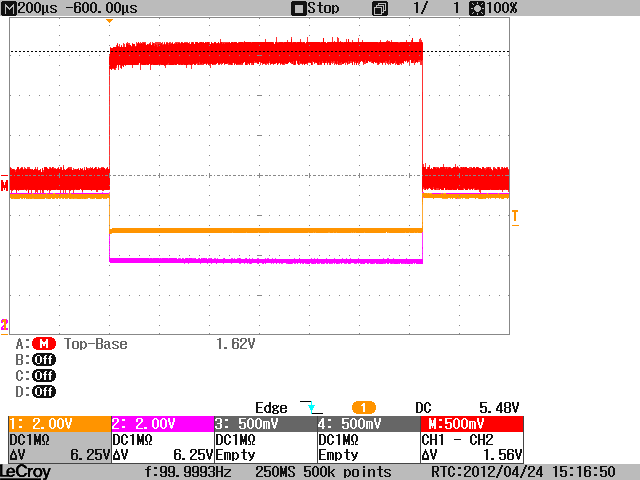
La résistance fût mesurée à 30.2 [Ω].

Pour cette mesure, seule la LED concernée fût allumée dans l’espace cubique.

Oscillogramme de la tension UR79 :

CH1 est connecté sur Pt2 à 2 [V/Div] et CH2 sur Pt3. Les canaux sont en DC avec un offset de -8 [V]. Le trigger est synchronisé avec CH1 sur front descendant à 5.48 [V]. Pour mesurer directement la tension UR79, la fonction maths fût appliquée (Pt2-Pt3) à 500 [mV/Div], sans offset.

La période de temps est de 200 [µs/Div].



Constat : La tension aux bornes de la résistance fût mesurée à 1.56 [V] lors de la conduction de la LED (sous le rapport cyclique d’1/8).

Le courant est limité à :



En valeur RMS :



L’on pourrait se contenter de cette valeur de courant, cependant, elle diminuera significativement lorsque toutes les LEDs seront allumées. J’ai donc abaissé la valeur des résistances à 27 [Ω].

### Mesures avec uniquement la colonne 64 allumée et R79 de 27 [Ω]

Mesures au multimètre Fluke 175 SN : 97101217

VLEDs = 7.04 [V]

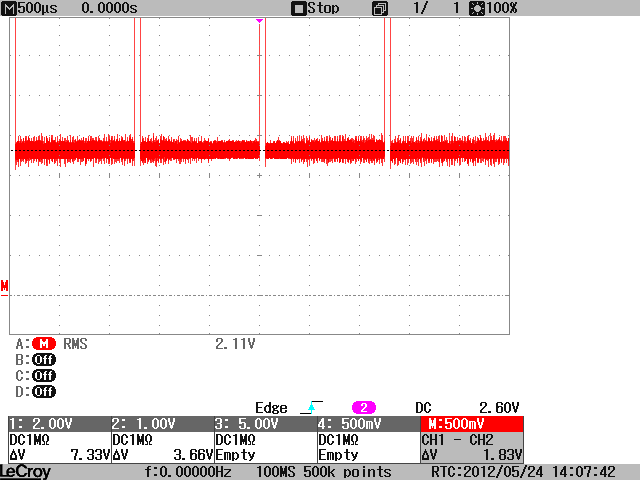
VCC = 5.079 [V]

R79 = 27.1 [Ω]

Tension de déchet MIC5891 :

CH1 (Pt1) correspond à la tension d’alimentation VLEDs de 7.04 [V]. CH2 (Pt2) représente la tension en sortie de la pin 5 (allant sur R79) du MIC5891 U6. La fonction maths représente (Pt1-Pt2) à 500 [mV/Div].

Les canaux sont en DC avec le trigger synchronisé sur CH2 sur front montant à 2.6 [V]. La période de temps est de 500 [µs/Div].

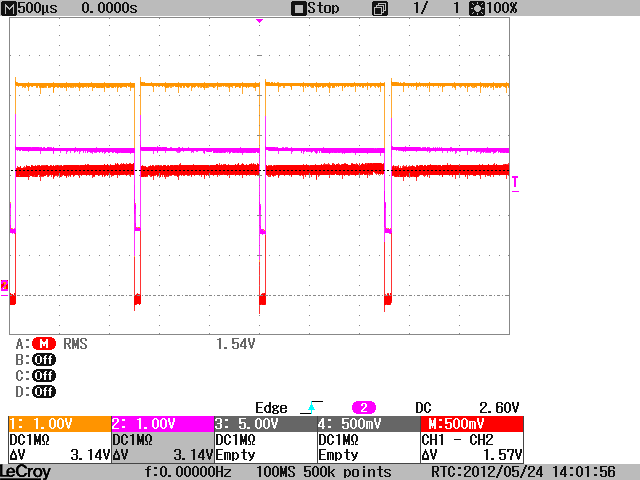


Constat : La tension de déchet mesurée à 1.83 [V] est proche de la valeur d’1.8 [V] pour I = 100 [mA] mise en évidence dans la datasheet du constructeur.

Tension aux bornes de R79 :

CH1 (Pt2) et CH2 (Pt3) sont à 1 [V/Div]. La fonction maths représente la chute de tension aux bornes de la résistance (Pt2-Pt3) à 500 [mV/Div].

Les canaux sont en DC avec le trigger synchronisé sur CH2 sur front montant à 2.6 [V]. La période de temps est de 500 [µs/Div].



Constat : La tension lors de la conduction est établie à 1.57 [V].

Le courant est limité à :



En valeur RMS :



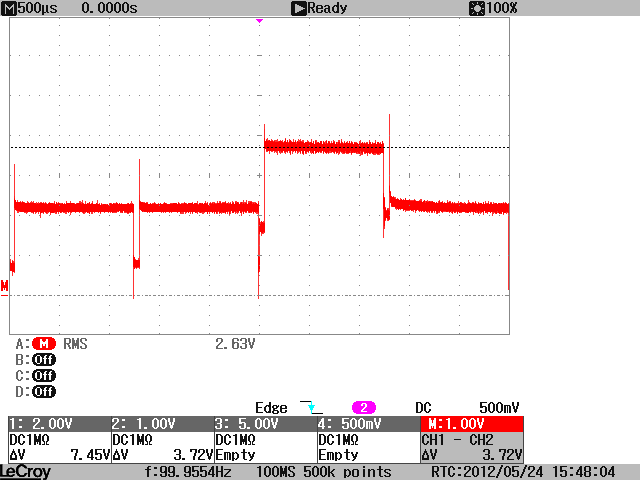
Constat : C’est déjà plus proche de la valeur désirée ! La nouvelle valeur de R79 sera conservée.

Tension aux bornes la colonne 64 (étage 0) :

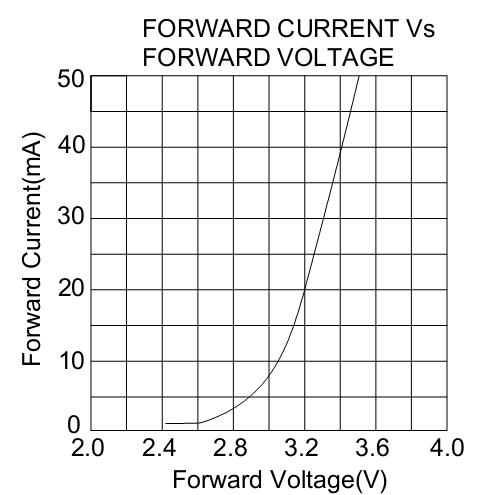
L’oscilloscope est connecté aux bornes de la LED 64 (de l’étage 0) à travers les câbles de l’anode et la cathode (donc les points les plus proches de la carte afin de prendre également en compte la chute de tension à travers les fils).

CH1 (Pt3) correspond à l’anode et CH2 (Pt4) la cathode. La fonction maths représente la chute de tension aux bornes de la LED (Pt3-Pt4) à 1 [V/Div].

Les canaux sont en DC avec le trigger synchronisé sur CH2 sur front descendant à 500 [mV]. La période de temps est de 500 [µs/Div].



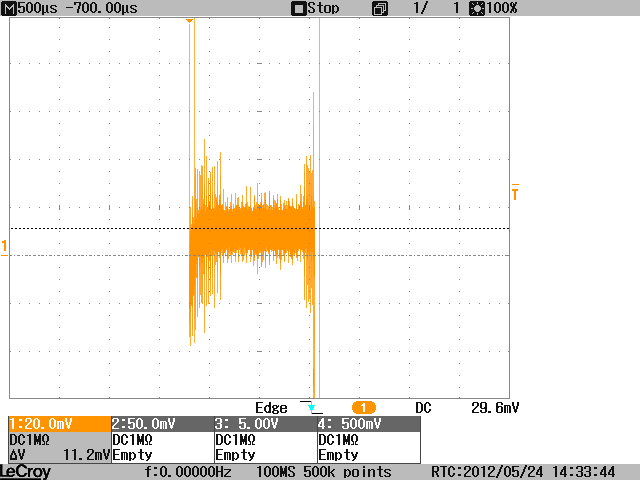
Constat : la tension atteint 3.72 [V], ce qui est supérieur à la valeur estimée à 3.1 [V] et à la valeur maximale proscrite dans la datasheet qui est de 3.4 [V] ! Cependant le graphique provenant de la même datasheet (ci-dessous) montre bien qu’une élévation de courant à travers la LED a pour effet d’augmenter la tension VF.



Tension UDSsat:

CH1 (Pt4) correspond à la tension UDrain à 20 [mV/Div].

Les canaux sont en DC avec le trigger synchronisé sur CH1 sur front descendant à 29.6 [mV]. La période de temps est de 500 [µs/Div].



Constat : avec aussi peu de courant, la tension de saturation n’est pas vraiment mesurable (de l’ordre de 11 [mV])…

Addition des tensions mesurées :

L’addition de toutes les tensions mesurées devrait équivaloir à 7.04 [V].

Calcul :



Conclusion : Il y a quelques petites erreurs de mesure mais cela reste cohérant.

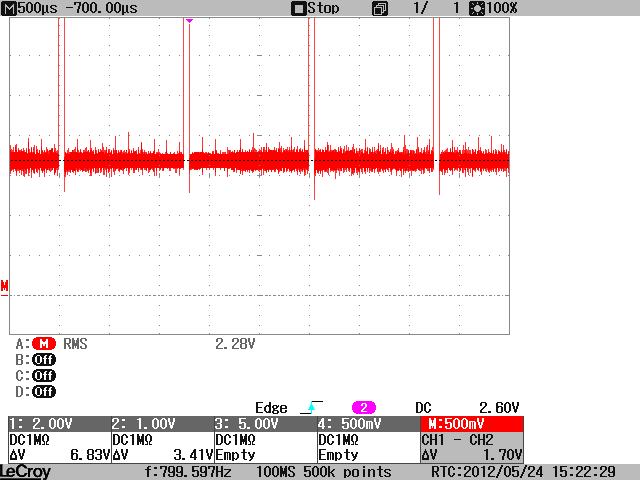
### Mesure avec tout le cube allumé (R79 de 27 [Ω])

A savoir : avec cette charge, la tension d’alimentation VLEDs mesurée sur le pcb a chuté pour finalement atteindre les 6.71 [V]. VCC quant à lui est resté stable : 5.082 [V]. (Le même multimètre que précédemment fût utilisé).

Tension de déchet MIC5891 :

CH1 (Pt1) correspond à la tension d’alimentation VLEDs. CH2 (Pt2) représente la tension en sortie de la pin 5 (allant sur R79) du MIC5891 U6. La fonction maths représente la tension de déchet (Pt1-Pt2) à 500 [mV/Div].

Les canaux sont en DC avec le trigger synchronisé sur CH2 sur front montant à 2.6 [V]. La période de temps est de 500 [µs/Div].

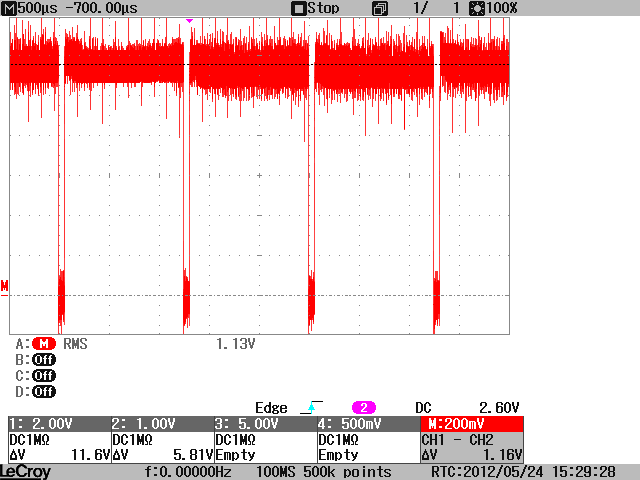


Constat : Il est intéressant de s’apercevoir que la tension de déchet est cette fois plus faible que précédemment (1.7 [V] en lieu et place d’1.83 [V]). Ceci provient de la température des composants ayant augmenté puisque toutes les sorties alimentent les LEDs.

Tension aux bornes de R79 :

CH1 (Pt2) est à 2 [V/Div] et CH2 (Pt3) à 1 [V/Div]. La fonction maths représente la chute de tension aux bornes de la résistance (Pt2-Pt3) à 200 [mV/Div].

Les canaux sont en DC avec le trigger synchronisé sur CH2 sur front montant à 2.6 [V]. La période de temps est de 500 [µs/Div].



Constat : La tension lors de la conduction est établie à 1.16 [V], ce qui fait une différence de 410 [mV] par rapport à la mesure du chapitre précédent (1.57 [V]).

Le courant est maintenant limité à :



En valeur RMS :



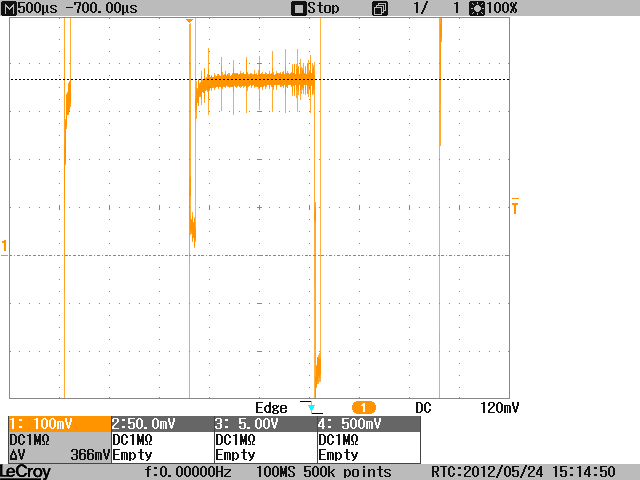
Tension aux bornes la colonne 64 (étage 0) :

La tension mesurée n’a pas varié depuis l’ancienne mesure (ULED64 = 3.72 [V]).

Tension UDSsat:

CH1 (Pt4) correspond à la tension UDrain à 100 [mV/Div].

Les canaux sont en DC avec le trigger synchronisé sur CH1 sur front descendant à 120 [mV]. La période de temps est de 500 [µs/Div].



Constat : Cette fois-ci, la tension de saturation n’est plus négligeable : 366 [mV]!

Addition des tensions mesurées :

L’addition de toutes les tensions mesurées devrait équivaloir à 6.71 [V].

Calcul :



Conclusion : Il y a une contradiction avec la mesure effectuée à l’aide du Fluke 175. Cependant, je suis certain de la tension UR79 et donc de la nouvelle valeur du courant !

## Mesures de la consommation de la carte

La consommation avec toutes les LEDs allumées et la luminosité maximale fût mesurée avec le FLUKE 175 (tensions SN: 97101217 ; courant SN: 99040601) :

Tension à vide mesurée à 7.05 [V].

Tension en charge mesurée à 6.7 [V].

Intensité mesurée à 2.372 [A].



La puissance de 15.89 [W] est valable dans ce cas précis où la tension d’alimentation chuterai à 6.7 [V].

Mesure avec la tension en charge à 7 [V] sur le pcb :

Tension en charge mesuré à 7 [V].

Intensité mesurée à 2.822 [A].



Cette valeur de 19.75 [W] est « idéale », et sera celle à utiliser lors du choix de l’alimentation.

# Conclusion

En définitive, le cube est pourvu d’un menu défilant permettant de choisir l’animation à afficher (grâce aux trois boutons de sélection) et d’un mode de connexion PC via l’USB. L’animation personnelle pourra être gérée et lancée depuis l’ordinateur grâce à l’application « Cubizator ».

Les mesures de courants à travers les LEDs montrent que l’intensité fluctue entre

20.48 [mA] (dans le cas d’une colonne (LED 64) « ON ») et 15.13 [mA] (lorsque tout le cube est allumé), ce qui exprimé en pourcentage donne la variation suivante :



Cela représente plus d’un quart de baisse par rapport au courant spécifié, ce qui n’est pas négligeable ! Cette variation est principalement causée par la tension UDSsat du MOSFET de commande d’étage qui augmente avec le courant, ainsi qu’à la baisse de la tension d’alimentation en charge. Heureusement, la luminosité perçue par notre œil n’est pas uniquement sensible au courant de chaque LED, mais aussi au nombre de LEDs allumées. Finalement, même avec la baisse de courant constatée, l’utilisateur percevra une plus forte luminosité.

L’usage du PWM permet de contrôler simplement et précisément la luminosité avec une erreur maximale de 4.8 %.

Pour ce qui concerne la communication aux MIC5891, c’est en réalisant cette rédaction que je me suis rendu compte (malheureusement trop tardivement) qu’il y a une contradiction dans la datasheet.

En effet, les informations fournies dans la partie « Features » indiquent la compatibilité des entrées avec les niveaux logiques TTL et CMOS. Cependant, dans la description générale, le fait est que la possibilité de relier une sortie logique TTL sur les entrées nécessite une adaptation de niveaux. Même après avoir passé du temps à lire cette documentation, j’ai continué à être persuadé que première affirmation était correcte tout en étant certain d’avoir bien fait attention à la compatibilité des niveaux. Ainsi, je pense avoir eu beaucoup de chance quant au fonctionnement de la communication (grâce à la tolérance des circuits). L’adaptation par l’ajout de simples résistances pull-up aux sorties du microcontrôleur n’est guère possible puisque la fréquence des signaux de communications est trop rapide pour les sorties open drain (testée sans succès). Il faudrait alors utiliser un circuit de la famille 74HCTxx à interconnecter entre le µC et les drivers sources.

Alors que je pensais être tiré d’affaire avec le problème de persistance rétinienne (chapitre 6.7.4), le simple branchement du câble USB au PC recréait un nouveau retard (cette fois-ci de durée variable) interférant avec le temps d’accès à l’INT0. La cause provient du fait que l’USB répond aux requêtes dans un délai maximal de 10

[ms], et ainsi, semble prendre la main sur toutes les interruptions, sauf celle mise en mode haute priorité !

La solution mise en œuvre consiste à fixer l’interruption INT0 en priorité maximale grâce à la mise à 1 du bit PX0.

Constat : Le problème de persistance n’est plus visible. Malgré tout, l’USB reste maintenant prioritaire sur les interruptions SPI, ce qui rend le temps d’envoi des données variable (pouvant prendre jusqu’à 216 [µs]) et, par conséquent le temps d’extinction des LEDs aussi, ce qui se perçoit par de légères variations de la luminosité. Les deux solutions envisageables seraient soit de changer de mode de communication (exemple : connexion au port série du PC), soit d’opter pour un microcontrôleur disposant de meilleures performances.

Concernant l’application de gestion d’animation sur PC, la communication via USB s’effectue convenablement.

L’alimentation utilisée devra être capable de délivrer au minimum 19.75 [W] sous une tension de 7 [V].

Si j’analyse mon travail de manière critique, je pense avoir répondu aux exigences demandées. Le chapitre explicatif du programme C aurait pu être bien plus détaillé, mais je n’avais qu’à mieux gérer mon temps !

# Améliorations

Il reste encore beaucoup de travail à fournir pour améliorer l’ergonomie de l’application PC. L’ajout de fonctionnalités telles que la copie de l’étage (ou image) courant(e) sur un(e) autre, effectuer des traitements d’image, rendre le temps de remplissage de la zone de dessin simplifié en bougeant la souris avec un clic maintenu, etc…

D’un point de vue microcontrôleur, de nouvelles fonctions « primitives » permettant la gestion (glissement, rotation) de l’image en degrés par application de matrice de calculs serait intéressante mais certes gourmande en ressources. Le programme principal devrait être remanié afin de n’appeler que quelques fonctions de gestion (gestion du menu, affichage, puis USB). L’ajout d’un jeu de casse-brique (ou du serpent) commandé par joystick ainsi que l’analyse des fréquences audibles reçues d’un baladeur sont en vue.

Une adaptation des niveaux logiques entre le microcontrôleur et les drivers sources ainsi que la mesure des tensions correspondantes aux seuils logiques réels (à titre indicatif) devraient-être effectués.

Un boitier carré en aluminium traité par microbillage sera utilisé de manière à ce que toute l’électronique se retrouve à l’intérieur et que le cube soit fixé sur l’extérieur du boitier.

# Mode d’emploi

## Alimentation

Après avoir sélectionné une alimentation d’au moins 20 [W] sous 7 [V], vous pourrez la connecter à la carte de commande via un connecteur jack de puissance 2.5/5.5 [mm].

## Navigation dans le menu du cube

Par la mise sous tension du cube (basculement de l’interrupteur principal), un menu défilant se présentera. Le chiffre affiché indique le numéro de l’animation. Grâce aux boutons « Précédent » et « Suivant », le menu défilera. Pour afficher l’animation, il suffira de presser le bouton « Ok ». Le retour au menu de sélection s’effectue par l’appui de n’importe quel bouton (sauf en mode connexion PC où la déconnection doit être émise par l’application PC).

## Connexion à un PC

Le branchement s’effectue simplement grâce au câble USB. Aucun driver n’est nécessaire. Je vous invite à lire le chapitre 7.2.2 pour prendre le logiciel en main.

# Remerciements

Je tiens à remercier tout particulièrement la bonne humeur apportée par les personnes m’ayant entouré durant ces sept semaines.

Merci à M. Tutt pour m’avoir suivi et conseillé lors de cette expérience, M. Neuhaus pour ses recherches sur l’USB et M. Weiss pour les commandes de matériel.

# Bibliographie

Liens internet :

Pour le calcul des pistes :

<http://www.4pcb.com/trace-width-calculator.html>

Pour la navigation dans les fichiers binaires :

<http://www.x-ways.net/winhex/index-f.html>

Utilisation des pointeurs sur tableaux à deux dimensions :

<http://forum.hardware.fr/hfr/Programmation/C/tableau-dimensions-pointeur-sujet_78983_1.htm>

<http://www.siteduzero.com/tutoriel-3-321485-tableaux-pointeurs-et-allocation-dynamique.html>

Structure du Cube à LEDs sur le site « instructable » :

<http://www.instructables.com/id/Led-Cube-8x8x8>

Autre cube à LEDs :

<http://www.hownottoengineer.com/projects/lc.html>

Pour l’obtention d’icones :

<http://www.icons16x16.com/>

<http://www.iconfinder.com/>

Pour les listes de circuits logiques :

<http://en.wikipedia.org/wiki/List_of_7400_series_integrated_circuits>

<http://en.wikipedia.org/wiki/List_of_4000_series_integrated_circuits>

Livres :

C. Cimelli, R. Bourgeron. *Guide du Technicien en électronique.* Ed. HACHETTE Technique, 2007.

Claude Rosset, Yvan Siggen, René Rathgeb. *Formules pour la résolution de problèmes* Ed. LEP Edition Loisirs et Pédagogie SA, 2002.

# Annexes

* Schéma
* Liste des sources
* Journal de bord
* Planning
* Listing du programme C
* Listing du programme C#
* Dossier de fabrication
* Datasheet du MIC5891