Rapport de Conception

**École supérieure – Fixme**

Totem Lumineux

**Réalisé par :**

Nicolas Fürst -> V0\_1

PBY (Philou) -> V1\_1

**Date :**

Version 0\_1 : 20 Juin 2019

Version 1\_1 : xx Mars 2024

Table des matières

[1. Cahier des charges 4](#_Toc150984795)

[2. SChéma bloC et Explication composantS 4](#_Toc150984796)

[2.1 Pic32 5](#_Toc150984797)

[2.2 Matrice X2 5](#_Toc150984798)

[2.2.1 Autres possibilités 6](#_Toc150984799)

[2.3 Module Wifi 6](#_Toc150984800)

[2.4 Application 7](#_Toc150984801)

[2.5 Batterie 7](#_Toc150984802)

[2.6 Panneau solaire / gestion de charge 7](#_Toc150984803)

[2.7 Port de charge 8](#_Toc150984804)

[2.8 Regulateurs 8](#_Toc150984805)

[9](#_Toc150984806)

[2.9 Capteur de luminosité 9](#_Toc150984807)

[3. éstimation des couts 9](#_Toc150984808)

[4. Description 11](#_Toc150984809)

[4.1 Apparence 11](#_Toc150984810)

[4.2 Fonctionnalité 11](#_Toc150984811)

[5. Choix effectué 11](#_Toc150984812)

[5.1 Pré-étude 11](#_Toc150984813)

[5.2 Supplément 11](#_Toc150984814)

[6. Dimenssionnement 13](#_Toc150984815)

[6.1 Mesure de la luminosité 13](#_Toc150984816)

[6.2 Régulateur 14](#_Toc150984817)

[6.2.1 Régulateur 5V 14](#_Toc150984818)

[6.2.2 Régulateur 3,3V 15](#_Toc150984819)

[6.3 Mesure de la batterie 17](#_Toc150984820)

[6.3.1 Mesure de la tension 17](#_Toc150984821)

[6.3.2 Mesure du courant 18](#_Toc150984822)

[7. Concept du logiciel 20](#_Toc150984823)

[7.1 logiciel de base 20](#_Toc150984824)

[7.2 Améliorations 20](#_Toc150984825)

[7.3 Idées Possibles 20](#_Toc150984826)

[7.4 géstion des LEd 20](#_Toc150984827)

[8. Boitier et PCB 21](#_Toc150984828)

[9. Schéma batterie 21](#_Toc150984829)

[10. Choix des pins 22](#_Toc150984830)

[11. PCB 23](#_Toc150984831)

[11.1 Dimenssionnement de la carte 23](#_Toc150984832)

[11.2 Placement 23](#_Toc150984833)

[11.2.1 Partie puissance 25](#_Toc150984834)

[11.2.2 Partie logique 25](#_Toc150984835)

[11.3 Routage 25](#_Toc150984836)

[11.4 Montage et problême 27](#_Toc150984837)

[12. Communication avec les LED 29](#_Toc150984838)

[13. Code 31](#_Toc150984839)

[13.1 harmony 31](#_Toc150984840)

[13.1.1 Pin settings : 31](#_Toc150984841)

[13.1.2 Timer 1 : 31](#_Toc150984842)

[13.1.3 SPI 32](#_Toc150984843)

[13.2 LED 33](#_Toc150984844)

[13.3 Message 35](#_Toc150984845)

[13.4 Autres fonctions 38](#_Toc150984846)

[13.5 Problème et module wifi 38](#_Toc150984847)

[14. Mecanique 39](#_Toc150984848)

[14.1 Montage 41](#_Toc150984849)

[15. Tests et mesures 43](#_Toc150984850)

[15.1 Alimentation 43](#_Toc150984851)

[15.2 SPI 44](#_Toc150984852)

[15.3 Durée de fonction 46](#_Toc150984853)

[15.4 Consommation 48](#_Toc150984854)

[16. Stade d’avancement 50](#_Toc150984855)

[17. Conclusion 50](#_Toc150984856)

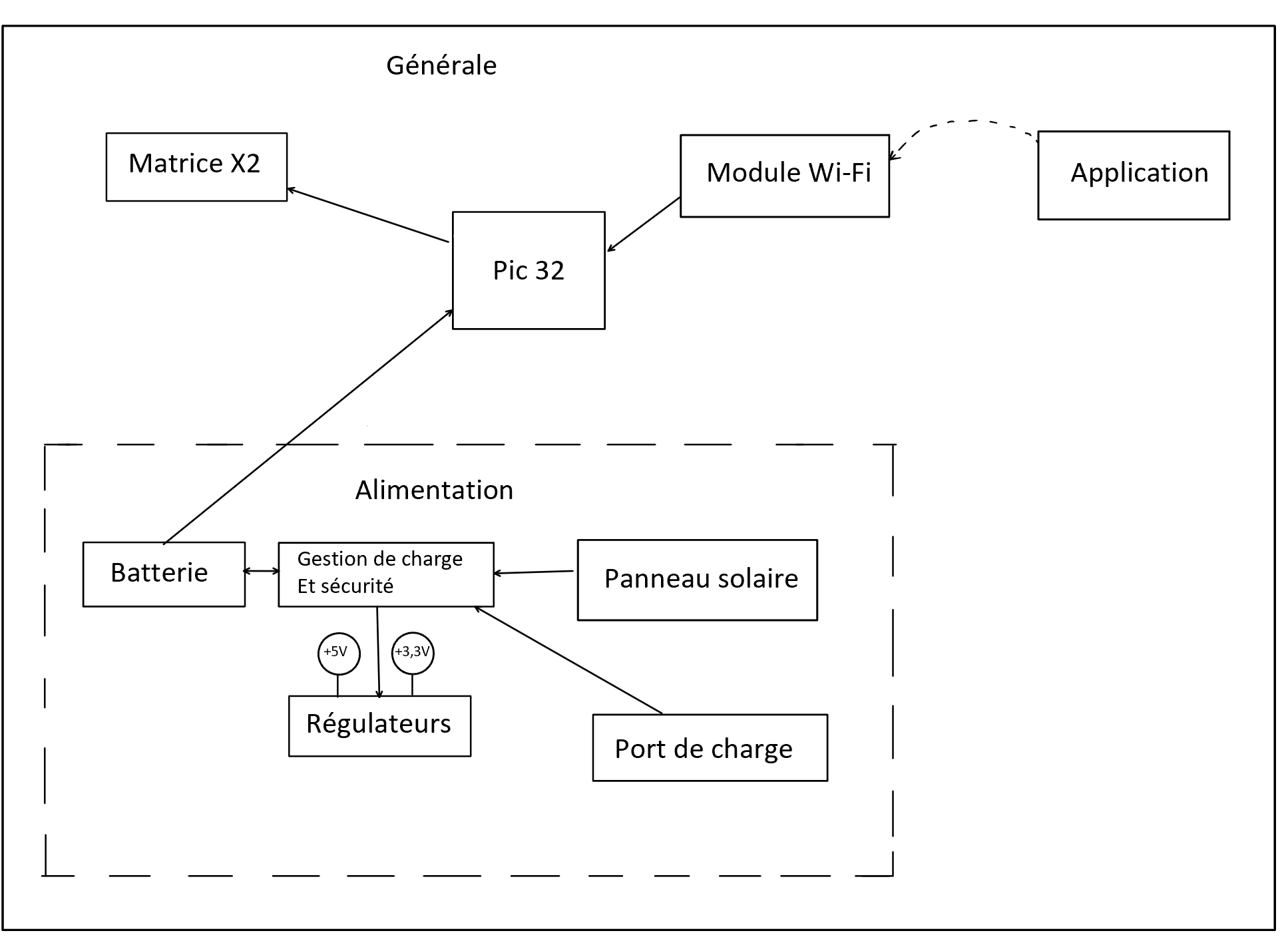
[18. Annexe 51](#_Toc150984857)

Phase Pré-étude

# Cahier des charges

* Totem lumineux à LEDs RGB
* Animation colorée.
* Possibilité d'afficher un mot ou un texte défilant.
* Lisible depuis toutes directions.
* Tête à monter sur un mat pour l'élever.
* Alimentation sur batterie, rechargeable par secteur et panneau solaire.
* Longue autonomie, environ 12H.
* Commande via Wifi, 4G ou Bluetooth.
* LED RGB - gestion de la batterie.
* Utilisable en extérieur.
* Gérer par un Pic32MX
* Affichage circulaire avec une matrice souple.

# SChéma bloC et Explication composantS





Capteur de luminosité

## ChoiX MicroControleur

Pour les versions V0\_1 et V1\_1, le choix du microcontrôleur se porte sur un PIC 32, mais il se peut que dans des versions futurs, le choix se porte sur un uC STM32,ou un ESP32.

* choix microcontrôleur : doit posséder au minimum ces périphériques/options:

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

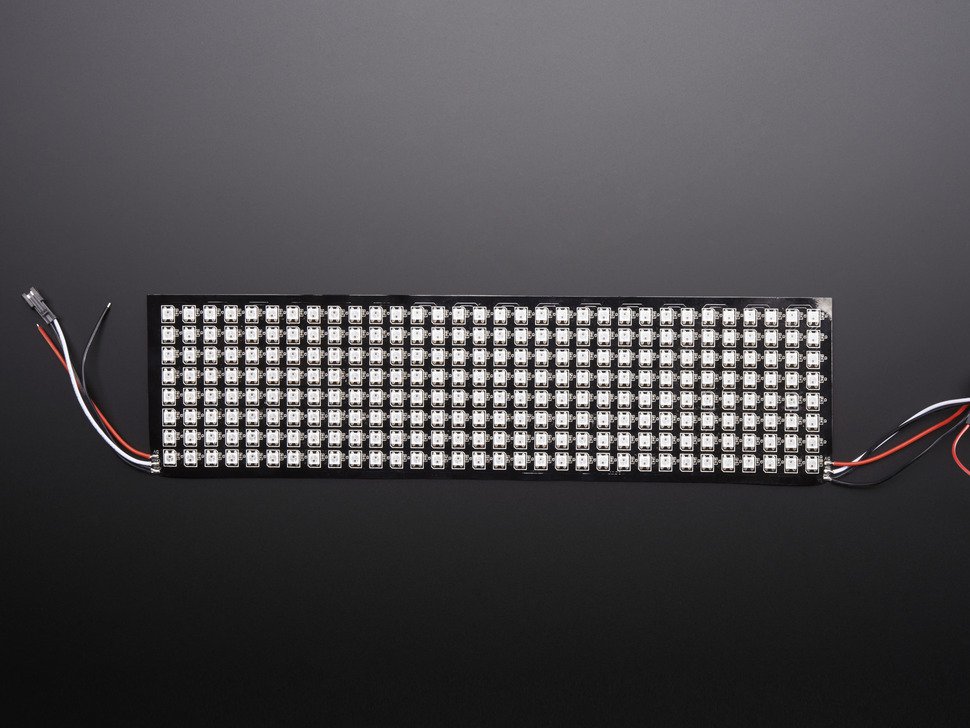
Le PIC32 – famille MX795H correspond à la requête minimale décrite ci-dessus.

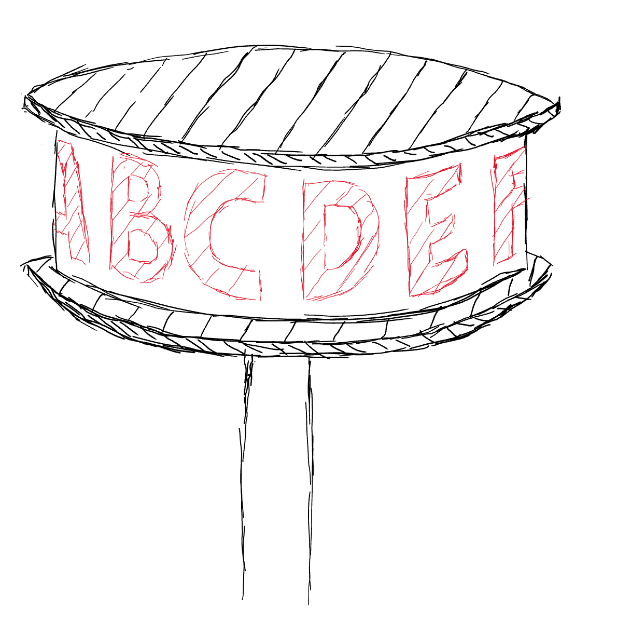
Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

## Panneaux de LEDS

Le choix des leds s’est basé sur deux panneaux de Leds flexibles de chez Adafruit ([produit 2294](https://www.adafruit.com/product/2294)) composé de 256 Leds RGB (modèle : ws2812b ou sk6812).



L’idée est de mettre ces deux matrices en cercle, comme le montre l’illustration ci-dessus.

Dimension de la matrice : 320mm x 80mm

Epaisseur : 2mm

Déterminer le rayon possible ? =>

p = périmètre r = rayon

Les deux matrices seraient disposées en cercle formant une sorte de soucoupe de 8 LED de haut et 64 de périmètre.

Il sera possible d’afficher entre 10 et 12 caractères en même temp et de faire tourner le texte pour en afficher plus et le rendre lisible de tous les coté.

Au niveau de la consommation après plusieurs mesures que l’on peut trouver dans le fichier Excel. Les LED peuvent consommer jusqu’à 64W ce qui est énorme mais pour quelle consomme autant il faut qu’elles soient toutes allumées en blanc et à la luminosité maximum ce qui fait un vrai lampadaire.

Je pense que pour une utilisation normale la consommation moyenne devrait se trouver entre 5 et 10 W ce qui est déjà beaucoup plus raisonnable.

D’après les mesures que j’ai faites il serait possible descendre en dessous de 2W et de toujours avoir un texte lisible, cependant la luminosité serais grandement atteinte et en plein soleil j’ai des doutes sur la lisibilité du texte.

Avec mes mesures j’ai également remarqué que les LED consomment même si elles sont éteintes, avec les deux matrice la consommation des LED éteintes vaut 1,73Wh.

En conclusion pour les deux matrices je pense qu’il sera définitivement possible d’avoir un excellent rendu à une consommation raisonnable. La consommation peut non seulement être réduite en réduisant la luminosité des LED mais également en n’affichant pas le texte en permanence par exemple. Comme je l’ai remarqué si les LED sont alimentées on ne peut pas consommer mois de 1,7Wh

, pour économiser encore plus il faudrait ajouter un système qui coupe l’alimentation des LED quand elles sont éteintes

### Autres possibilités

Parmi les autres possibilités, il y avait faire une seule matrice de LED qui tournerais sur elle-même. Mais cette méthode nécessite de faire passer au minimum la masse et le positive au travers d’un axe ce qui est très compliqué.

Il aurait également été possible d’utiliser 4 matrices formant un carré. Mais la consommation par rapport au deux matrices souples n’auraient pas changé dû au nombre de LED quasi identique donc quatre matrices en carré résulteraient juste en un affichage moins bien lisible.

## Module Wifi

Pour le module Wifi, j’ai trouvé trois modules potentiels le esp32 qui est le module que Patrick Favre va utiliser pour son projet.

La deuxième possibilité est assez similaire, s’est le RN171.

Et enfin la dernière et celle que je choisirais pour l’instant car par rapport au deux autres il est possible de si connecter depuis une page internet est le MRF24Wn0MA / ATWILC1000.

## Application

Il n’est pas prévu pour l’instant que je réalise l’application, je ne me suis donc pas encore renseigné sur le sujet.

## Batterie

La première chose à savoir pour pouvoir choisir la batterie est quel volume doit-elle faire. Pour le déterminer il me suffit de multiplier l’autonomie désirer [h] par la consommation moyenne de mon système :

Pour l’autonomie l’objectif est de 24H au mieux, mais pour la consommation s’est un peu plus compliquer. Premièrement je considère la consommation des LED comme l’unique consommation du système car le reste (Pic, module wifi) ont une consommation négligeable par rapport au LED. Comme l’autonomies est désirer dans le meilleur des cas je considère quelle seras possible à une consommation réduite. Je choisi donc 2Wh qui selon moi est une consommation atteignable.

La batterie devra donc avoir une capacité minimum de 24 \* 2 = 48Wh.

Une fois la capacité trouver deux solutions son possible : Batterie au plomb ou batterie au lithium. Cependant vus qu’il faut une grande capacité les batteries au lithium son moins envisageable.

Le prix/Wh des batteries au lithium est beaucoup plus élevé que pour celle au plomb ce qui en fait le premier mauvais point.

Ensuite due à la grande capacité nécessaire il faut plusieurs cellules ce qui rend la charge et la gestion de la sécurité des batteries au lithium très compliquer alors que la gestion de charge est un jeu d’enfant avec une batterie au plomb.

Les gros inconvénients d’une batterie au plomb sont sa taille et son poids, mais ce n’est pas un problème pour mon projet car le Totem na pas pour objectif d’être déplacer en permanence et il est parfaitement possible de faire une sorte de boite ou de cage à la base du totem pour y ranger la batterie.

Après quelque recherche j’ai trouvé cette batterie chez distrelec :

<https://www.distrelec.ch/fr/batterie-au-plomb-12-ah-fiamm-fg20722/p/30047216?q=batterie+au+plomb&page=6&origPos=6&origPageSize=50&simi=99.87>

Cette batterie fait 7,2Ah pour 12V ce qui fait 86,4Wh ce qui est largement au-dessus des 48Wh minimum calculer. Ça taille et son poids son également parfaitement raisonnable.

Avec cette batterie l’autonomie a consommation minimum :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Consommation | Minimum, 2Wh | Moyenne, 5Wh | Moyenne, 10Wh | Maximum 30Wh |
| Autonomie | 43,2h | 17,2h | 8,6h | 2,88h |

Avec la batterie on peut facilement atteindre 12h avec la consommation moyenne que j’ai estimée, en plus avec le panneau solaire cette autonomie peut être grandement améliorer. En plus si le Totem est utilisé en mode basse consommation, selon mes estimations il est presque possible d’atteindre 2 jour non-stop d’utilisation.

Mais plusieurs problèmes peuvent apparaitre avec une batterie de moto avec les charges décharge répéter. Voir la batterie à utiliser dans le point panneau solaire

## Panneau solaire / gestion de charge

Pour le panneau solaire, on m’a conseillé de prendre un kit qui comprend panneau, régulateur de charge et une batterie AGM qui est une batterie au plomb un peu spéciale adapté pour les panneaux solaires.

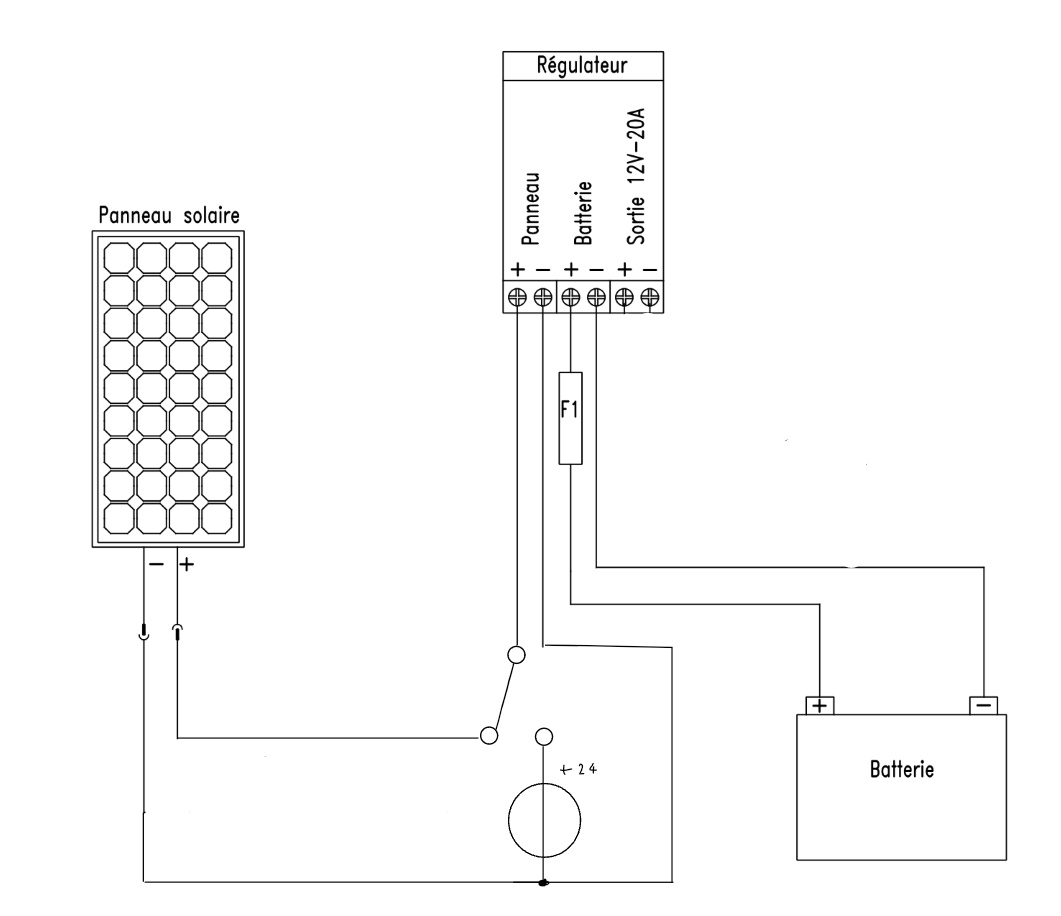
J’ai assez vite trouvé ce kit de chez Swiss green :

<https://www.swiss-green.ch/fr/kits-solaires-standard/39286460-kit-solaire-420-wh-12v.html>

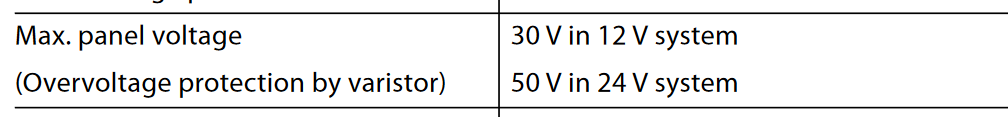
Il comprend un panneau solaire de 20Wpeak, un régulateur de charge de 5A et une batterie au plomb AGM de 12V 8Ah ce qui fait pile le double en capacité du minimum que j’ai calculer.

## Port de charge

En plus du panneau solaire il faut un port de charge secteur pour pouvoir le charger sans dépendre du soleil. Le seul problème est que le régulateur de charge qui viens avec le panneau solaire ne possède pas d’entrer secondaire. Ma solution est donc de directement brancher une alimentation de 24V sur l’entrée du panneau solaire comme ceci :



Vus que l’entrée du panneau solaire doit être d’un maximum de 30V pour une batterie de 12V :



Une alimentation de 24V comme celle-ci fait parfaitement l’affaire :

<https://www.amazon.fr/LED-Alimentation-151W-MeanWell-LPV-150-24/dp/B01CGQRYZY/ref=sr_1_18?ie=UTF8&qid=1544627932&sr=8-18&keywords=alimentation+24V>

Sachant que c’est un régulateur de charge de 5A pour une batterie de 12V ça ne sert à rien d’avoir une alimentation de plus de 3A pour 24V.

## Regulateurs

Une image contenant diagramme, Plan, Dessin technique, schématique

Description générée automatiquement

Le régulateur de 3,3V est simplement là pour le PIC se seras donc le même que pour les autres s’est à dire un : LM2674M

Mais pour les LEDs s’est différent il faut un régulateur capable de fournir au minimum 3A à 5V constant et jusqu’à 12,8A au cas où toutes les LEDs devaient être allumé à la luminosité maximum.

Vus qu’il y aura deux matrices il est possible si nécessaire de faire deux régulateurs chacun de 6,5A.

Après de multiple recherche je n’ai pas trouvé de composant tous en un avec un courant de sortie aussi élever. Mais à la suite du conseil de Yann Dacosta j’ai utilisé un des outils de Texas Instrument pour désigner un step down buck converter.

Voici un des montages qui tourne autour du TPS40345DRC :

<http://www.ti.com/lit/ds/symlink/tps40345.pdf>

## 

Pour retrouver la simulation il faut mettre sur ce lien : <https://webench.ti.com/power-designer/switching-regulator> les paramètres suivants : VinMin = 12V VinMax = 12V, Vout = 5V et IoutMax = 13A. Ensuite s’est le premier résultat.

Cette solution semble adéquate mais elle pose des problèmes au niveau du routage et du montage.

C’est pourquoi comme autre solution il y aurais un ou deux convertisseur DCDC comme celui-ci : <https://www.mouser.ch/ProductDetail/GE-Critical-Power/NQR010A0X4Z?qs=sGAEpiMZZMsc0tfZmXiUnbqX60T3gN%2fSlNT9RInDDys%3d>.

## Capteur de luminosité

Le capteur de luminosité servira à mesurer l’intensité lumineuse de l’environnement dans lequel se trouve le totem, et adapte la luminosité des LEDs pour qu’elles restent visible. Plus l’environnement est lumineux plus les LED le seront.

Pour ça il y a une multitude de capteur disponible. Mais mon choix s’est tourné vers un capteur de luminosité ambiante sous forme de LED :

<https://www.distrelec.ch/fr/capteur-de-luminosite-ambiante-570-nm-vishay-tept-4400/p/17522426?q=capteur+de+luminosit%C3%A9&page=4&origPos=4&origPageSize=50&simi=99.06>

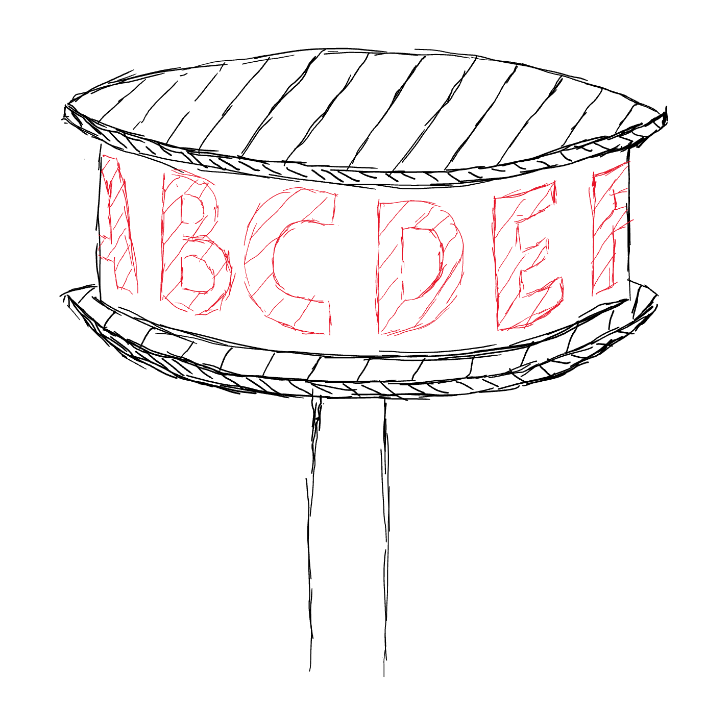
# éstimation des couts

|  |  |
| --- | --- |
| Composant | Prix |
| PIC | 10 .- |
| Module WIFI | 5 à 18.- |
| Régulateurs | 5 à ~20.- |
| Panneau solaire, batterie et régulateur | 230.- |
| Matrices X2 | 200.- |
| Composant divers | 5 à 20.- |
| Total | 455 à 498.- |

Phase de design

# Description

## Apparence



En apparence, la tête du totem fera un peu plus de 20 cm de diamètre pour 10 de haut, les deux matrices de 256 LED feront tout le tour afin de pouvoir afficher du texte lisible depuis toutes les directions. Le PCB avec le pic et tout le reste se situera dans la tête.

La base sera moins complexe, elle devra déjà maintenir le mât et la tête droits et stables et contenir la batterie, le panneau solaire, les deux régulateurs de charge avec le port de charge et finalement l’interrupteur on/off. La base pourra donc simplement être une boîte contenant le tout et maintenant le panneau solaire à l’extérieur.

## Fonctionnalité

En plus de pouvoir afficher un texte par Wifi, il sera possible de changer la couleur du texte, choisir d’afficher différentes animations, modifier la luminosité soit automatiquement ou alors à l’aide d’un capteur et finalement le Totem mesurera la tension de la batterie régulièrement pour pouvoir signaler le niveau de batterie et adapter automatiquement la luminosité ou la fréquence d’affichage pour allonger l’autonomie du Totem et le maintenir allumé plus longtemps.

# Choix effectué

## Pré-étude

Ce que j’avais déjà choisi lors de la pré-étude est resté. Que ce soit pour les matrices de LED, le kit batterie plus panneau solaire, le capteur de luminosité ou le design du Totem il n’y a pas eu de modification.

Cependant certains points n’étaient pas complétement résolus, notamment pour les régulateurs de tension 12V à 5V et pour le module Wifi. Depuis ses deux points ont été éclaircis .

Le module Wifi sera un ATWILC1000 de chez Microchip car il fallait un module ayant la possibilité de s’y connecter depuis une page Web. Le MRF24WN0MA a été mon premier choix car il a déjà été utilisé pour un précèdent projet, mais Microchip le considère obsolète et conseille d’utiliser son équivalent, le ATWILC1000.

Pour le régulateur, j’ai choisi de prendre deux NQR010A0X4. Je les ai choisis principalement pour une question de simplicité par rapport à celui créé sur mesure sur Texas Instrument et en fonction du coup.

## Supplément

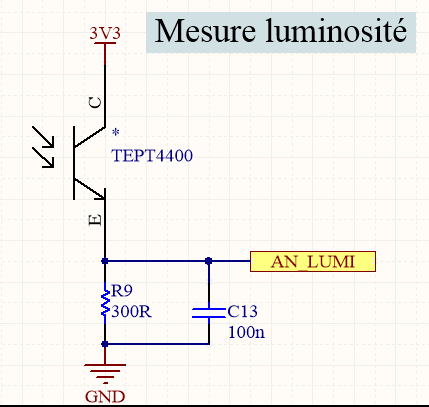
En plus de ce que j’ai prévu dans la pré-études, j’ai ajouté un LMP8601 qui permet de mesurer un courant. Cet ajout, en complément de la mesure de tension, permettra de mesurer le niveau de la batterie avec précision malgré les chutes de tension.

Le deuxième gros ajout est un deuxième régulateur de charge qui s’avère nécessaire pour avoir une charge par panneau solaire et par secteur.

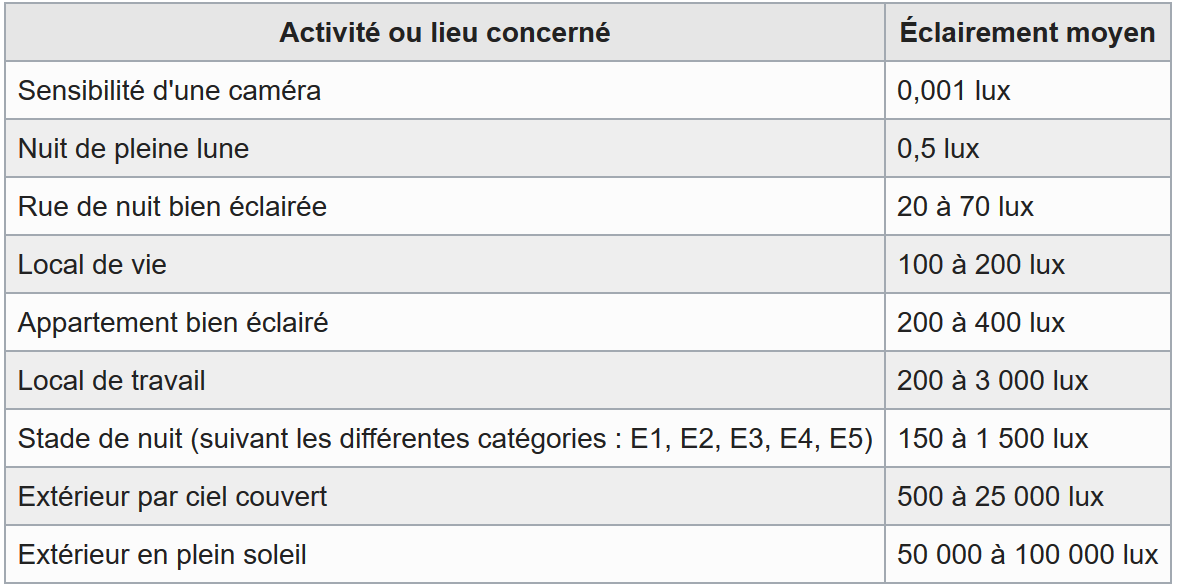
# Dimenssionnement

## Mesure de la luminosité

Voici le schéma du capteur de luminosité :



Son fonctionnement est plutôt simple. Le phototransistor utilisé comme capteur fournit un courant en fonction de la luminosité qu’il perçoit, il suffit donc de faire passer ce courant dans une résistance pour pouvoir mesurer une tension sur un port analogique du pic. Pour dimensionner la résistance il faut déterminer la luminosité maximum que l’on veut pouvoir détecter en Lux, pour me donner une idée de ce que représente une valeur de luminosité je suis allé voir sur Wikipédia :



Grace à ce tableau je peux déterminer à peu près qu’il ne sera pas nécessaire de connaitre la luminosité ambiante si elle passe le cap des 5000 Lux. Une fois ce cap atteint l’affichage sera certainement déjà à sa luminosité maximale.

Ensuite, dans le datasheet du capteur on peut facilement trouver la résolution du capteur (le nombre de mA par Lux) :

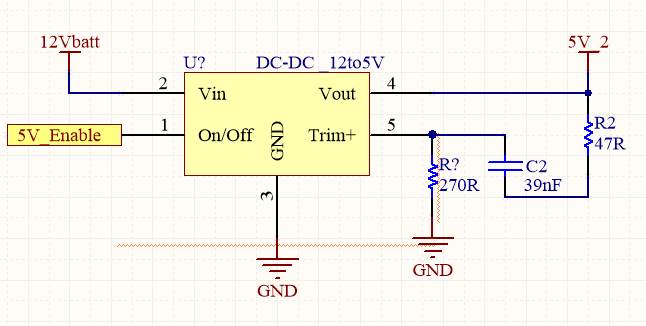


Malheureusement, dans le tableau on peut voir trois différents groupes de capteurs (A,B et C) qui ont chacun un rapport différent, mais Distrelec ne précise pas à quel groupe appartient leur capteur ce qui fait que selon le type du capteur la valeur de la résistance peut être amenée à changer. Pour l’instant j’ai considéré que le capteur se situera à 45 uA/20Lux ce qui donne les résultats suivants :

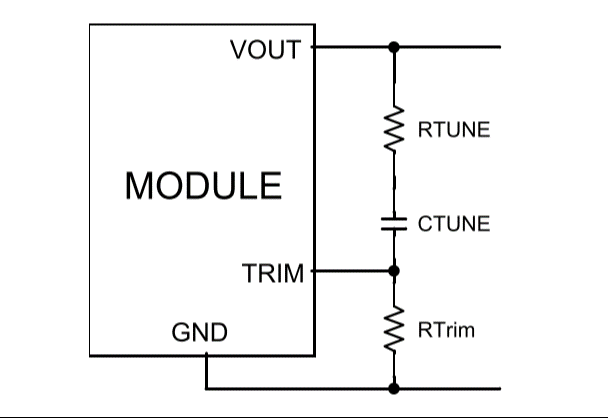
## Régulateur

### Régulateur 5V

Schéma pour les deux régulateurs 5V :

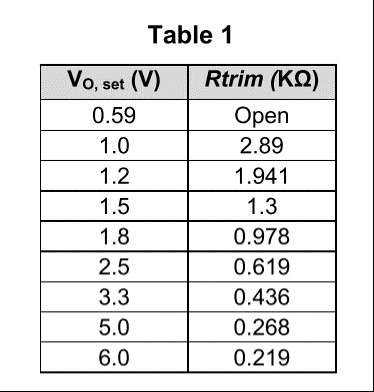
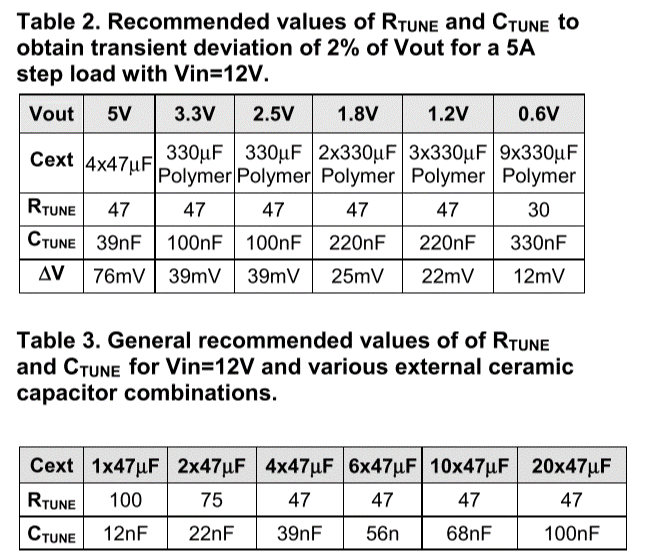


Avec ces régulateurs, pour avoir une sortie à 5V il faut dimensionner le condensateur et les deux résistances. Pour ce faire on trouve dans le datasheet un schéma qui montre comment les câbler :



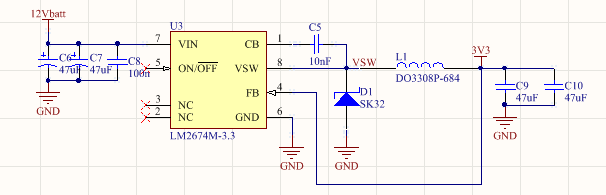
Et des tableaux nous donnant les valeurs de résistance et de condensateur pour des tensions communes telles que 3,3V ou dans mon cas 5V :

Avec les tableaux on peut déterminer toutes les valeurs nécessaires : RTune (R2) = 47 Ohm, CTrim (C2) = 39nF et RTrim = 268 Ohm. Pour RTrim la valeur ne tombe pas sur une valeur facile à trouver, j’ai donc pris la valeur la plus proche dans la série E12 ce qui donne 270 Ohm. Pour vérifier l’impact que le changement de valeur a eu sur la tension de sortie, j’ai retourné la fonction servant à trouver RTrim en cas de tension hors valeur des tableaux :



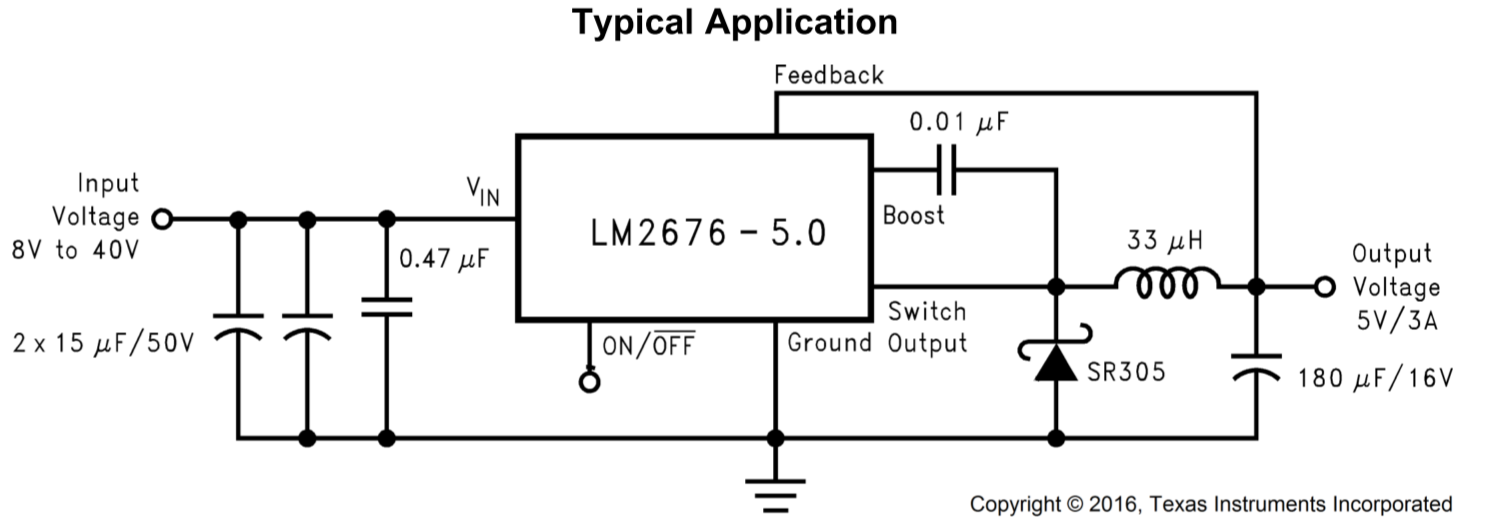
On peut voir que la tension reste extrêmement proche des 5V.

### Régulateur 3,3V

Schéma du régulateur 3,3V :

Comme pour le régulateur 5V toutes les informations sur le dimensionnement des composants se trouvent dans le datasheet. Cette fois-ci cependant la tension de sortie est fixée à 3,3V dans le chip. Tous les dimensionnements se feront par rapport au courant nécessaire en sortie.

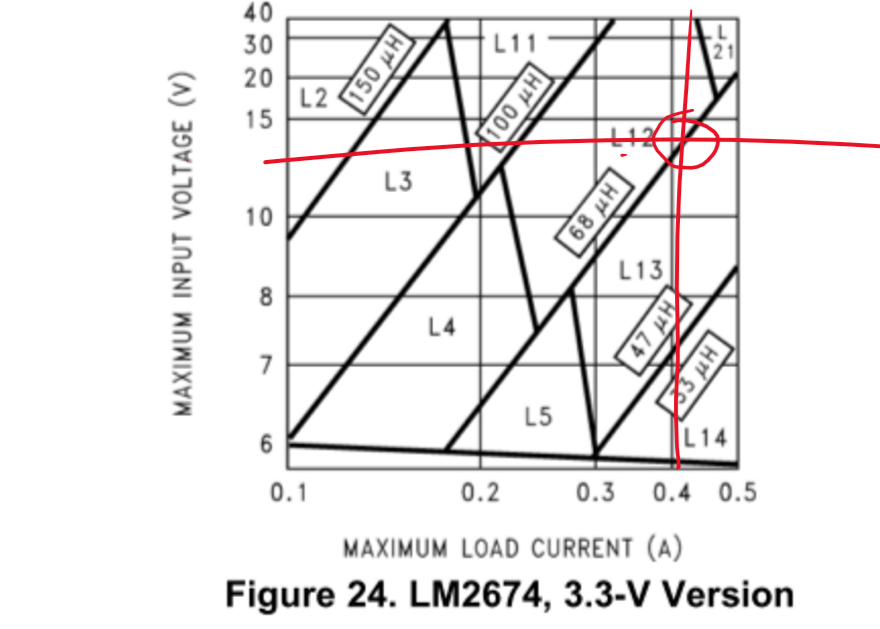
La première chose est le schéma exemple :



On peut voir que tous les composants sont déjà dimensionnés pour une application typique. J’ai repris la valeur des condensateurs tels quels mais j’ai quand même redimensionné la bobine et la diode.

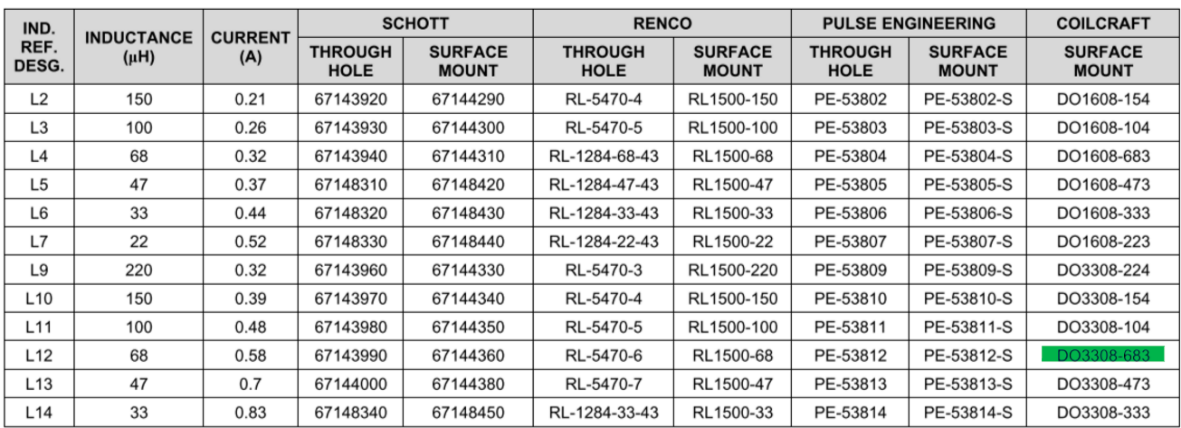
Que ce soit pour la bobine ou la diode il faut commencer par déterminer le courant max de sortie nécessaire. Pour estimer le courant max j’ai simplement considérer environ 200mA pour le Pic et pour le module Wifi ce qui correspond à ce que j’ai pu trouver dans les datasheets, j’ai ensuite arrondi à 500mA pour compenser les autres éventuelles consommations.

Une fois ce courant max déterminé il suffit de trouver les valeurs correspondantes dans les tableaux :



Le tableau tombe pile entre deux bobines pour avoir de la marge j’ai choisi la bobine L23.

Le datasheet fournit un tableau avec des bobines correspondent aux références du premier tableau :



J’ai choisi la bobine DO3308-683 car je voulais une bobine SMD et celle-ci se trouvait dans le vault de Altium.

Pour la diode on peut trouver un tableau similaire à la bobine :

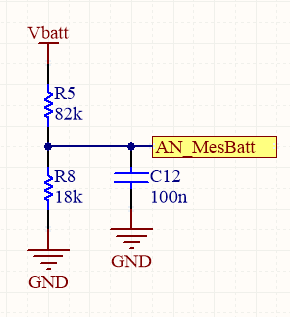


Pour les diodes, toutes feraient l’affaire je me suis donc contenté de prendre la plus basique que le datasheet propose en SMD.

## Mesure de la batterie

### Mesure de la tension

Schéma du diviseur :

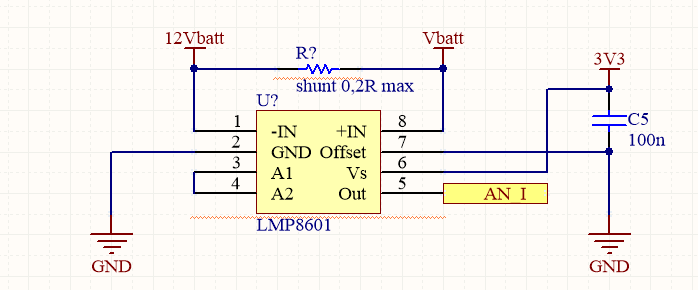


La mesure de la tension est constituée d’un simple diviseur de tension pour avoir une tension variable d’un maximum de 3,3V lisible par une des entrées analogiques du Pic. En considérant que la batterie aura une tension maximale de 14V on peut facilement dimensionner le diviseur en déterminant que Rtot vaux 100 kOhm pour limiter un maximum le courant :

Pour R8 22kOhm fonctionnerait mais pour avoir une marge d’erreur j’ai choisi la valeur inférieure de 18kOhm. On peut ensuite trouver la résistance R5 qui vaudra 82 kOhm, ce qui fait parfaitement 100 kOhm au total.

### Mesure du courant

Schéma du LM8601 :



Pour mesurer le courant j’utilise un LM8601. Il s’agit basiquement d’un ampli OP. Le dimensionnement à faire ici est la résistance. Le but de cette résistance est de créer une chute de tension en fonction du courant qui la traverse pour créer une différence entre les deux entrées de l’ampli OP.

Pour la dimensionner, il faut que la valeur de la résistance soit suffisamment haute pour que la chute de tension au courant maximum soit exploitable, mais la plus basse possible pour minimiser les pertes. Dans cette optique, on m’a conseillé de ne pas dépasser 1V de chute de tension aux bornes de la résistance lorsque le courant est maximum. Maintenant en divisant cette chute de tension par le courant que la batterie aura à fournir au maximum dans le pire des cas, on trouve la valeur de la résistance :

Calcul du courant :

Le courant max ne devrait pas dépasser les 5,14A mais j’ai fait la suite des calculs en considérant le courant max à 5A. Car comme c’est un courant estimé et extrême qui ne devrait jamais survenir plus de quelques secondes. L’arrondir à 5A donne une valeur de résistance plus simple.

Calcul de la résistance :

# Concept du logiciel

## logiciel de base

Le Totem pourra être contrôlé uniquement au travers du module Wifi. Voilà une liste de ce qu’il sera possible de faire :

* Pouvoir afficher tous les caractères de l’alphabet en majuscule ainsi que tous les chiffres.
* Mettre à jour le texte à afficher.
* Modifier la luminosité ou demander au Totem de la régler lui-même.
* Modifier la fréquence d’affichage, faire en sorte que le texte tourne en permanence ou que une fois toutes les 5 secondes par exemple.
* Demander au Totem d’afficher le niveau de batterie.

## Améliorations

Si aucun problème majeur ne se présente lors de la réalisation du projet, le logiciel de base avec les fonctions citées ci-dessus sera fait et fonctionnel. Cependant, je pense qu’il y a beaucoup d’améliorations qui peuvent être ajoutées. Si mon avancement le permet voici la liste des fonctions que j’aimerais pouvoir ajouter de la plus importante à la moins importante selon moi :

1. Pouvoir modifier la couleur du texte voire de chaque caractère.
2. Etendre le nombre de caractère possible à celui de la table ascii.
3. Ajouter des animations.
4. Faire un mode d’affichage optimisé à différentes consommations permettant d’estimer précisément combien de temps encore le Totem pourra rester allumé.

## Idées Possibles

Il est très peu probable que je réussisse à faire le logiciel de base avec toutes les améliorations que j’aimerais y amener mais si j’y arrive et qu’il me reste du temps, voici des idées que j’ai eues pour la suite :

* Augmenter le nombre de caractères à la table ascii étendue.
* Ajouté un mode d’édition de la matrice depuis l’application wifi. Ce mode permettrait de modifier indépendamment chaque LED des deux matrices. Ce mode permettrait aussi de modifier la matrice de manière automatique depuis un PC par exemple sans devoir modifier le code dans le Pic.

## géstion des LEd

Les LED des matrices sont des WS2812B qui sont des LED RGB avec un chip permettant une communication en série, ce qui permet de contrôler les 512 LED avec une seule ligne de commande. Pour pouvoir les commander il faut donc respecter un protocole, chaque LED prend 24 bits, les bits sont envoyés à une vitesse de 800kHz, chaque bit que ce soit un 1 ou un 0 possède un temps à l’état haut et un temps à l’état bas, ce qui permet entre autres de transmettre l’horloge.

C’est pourquoi l’utilisation d’une ligne SPI pour contrôler les LED fonctionne bien, en utilisant un SPI travaillant à une fréquence 8 fois plus élevée que celle nécessaire pour gérer les LED un byte du SPI correspondra à un bit pour les LED et donc en transmettant 1000’0000 on obtiendras un 0 et 1111’1110 un 1.

# Boitier et PCB

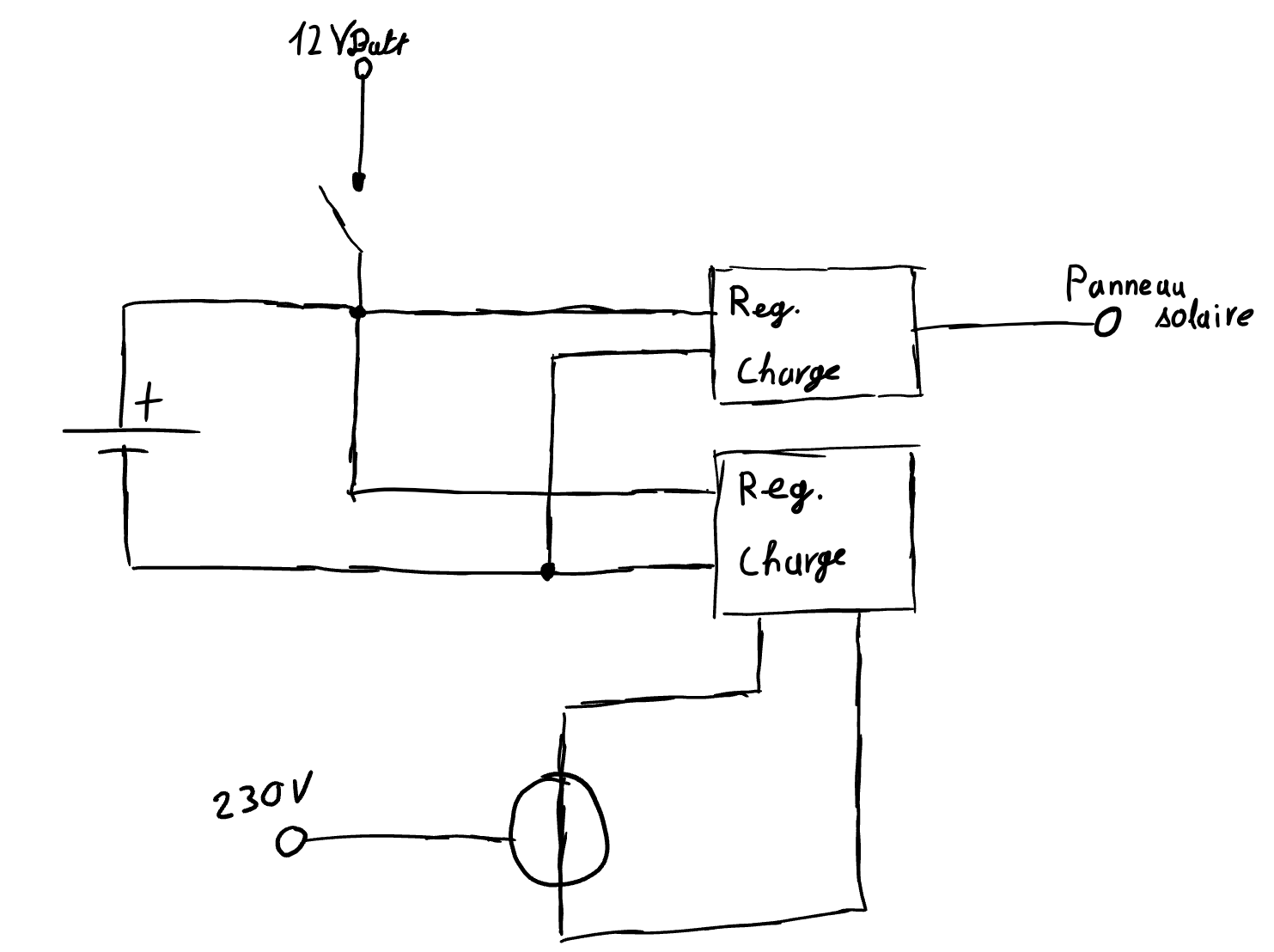
Pour le boîtier, la forme sera normalement très semblable à ce que j’ai dessiné. Pour déterminer les dimensions du boîtier, il faut partir du diamètre que les deux matrices vont créer une fois mises en cercle. Vu que la longueur des matrices est connue, le diamètre peut se trouver facilement :

La hauteur de la tête est en partie déterminé par la hauteur des matrices qui font 8cm de haut, À la hauteur des matrices s’ajoute l’épaisseur des disques inférieurs et supérieurs. Au total la tête fera environ 10cm de haut.

Le diamètre des matrices permet de connaitre le diamètre maximal du PCB, environ 20cm de diamètre. Cependant pour le diamètre du PCB je vais partir sur un PCB rond de 18cm de diamètre ce qui devrait être amplement suffisant au cas où cela ne suffit pas il reste encore un peu de marge.

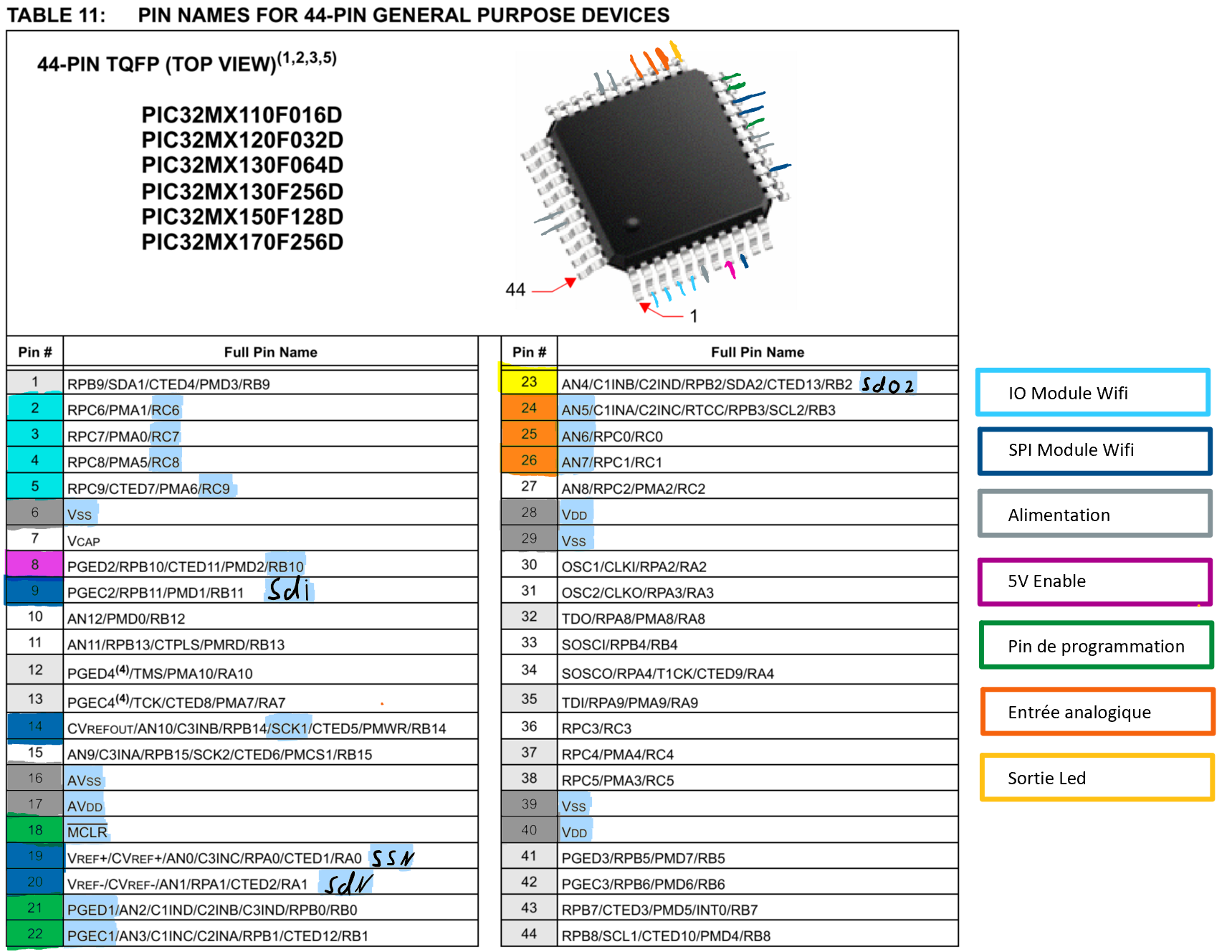
# Schéma batterie

La partie charge de la batterie, interrupteur et panneau solaire se trouvera dans et sur la base. Cette partie ne sera pas faite sur PCB mais câblée. Voici le schéma :



# Choix des pins

Pour visualiser le choix des pins sur le pic j’ai fait ce tableau :



La première chose à faire pour correctement choisir les pins est de savoir ce qu’il faut et pour quel élément :

* Trois entrées analogiques, deux pour la mesure de la batterie et une pour la luminosité.
* Un SPI à 4 fil et 4 IO pour le module Wifi.
* Une IO pour activer, désactiver l’alimentation 5V.
* PGED, PGEC.
* Les alimentations.

Une fois déterminé il faut les placer. Le plus simple est de d’abord placé celle qui on le plus de restriction. Les alimentations ne peuvent pas être placé ailleurs que sur les pattes d’alimentation, de ce fait la question de ou les mettre ne se pose même pas. Ensuite le module Wifi est celui qui monopolise le plus de pattes, de plus il a besoin d’un SPI, pour le SPI 1 la patte SCK1 est la seul à être imposer les autres sont des pps ce qui fait quel peuvent être placer presque librement. J’ai donc essayé de placer les huit pattes destiner au module wifi sur le moins de côté du Pic possible. J’ai placer les deux pattes de programmation sur PGEC1 et PGED1 car elles étaient libres. Finalement j’ai placé les pattes restantes en essayant d’imaginé ou seront placer les composant sur le PCB.

Réalisation

# PCB

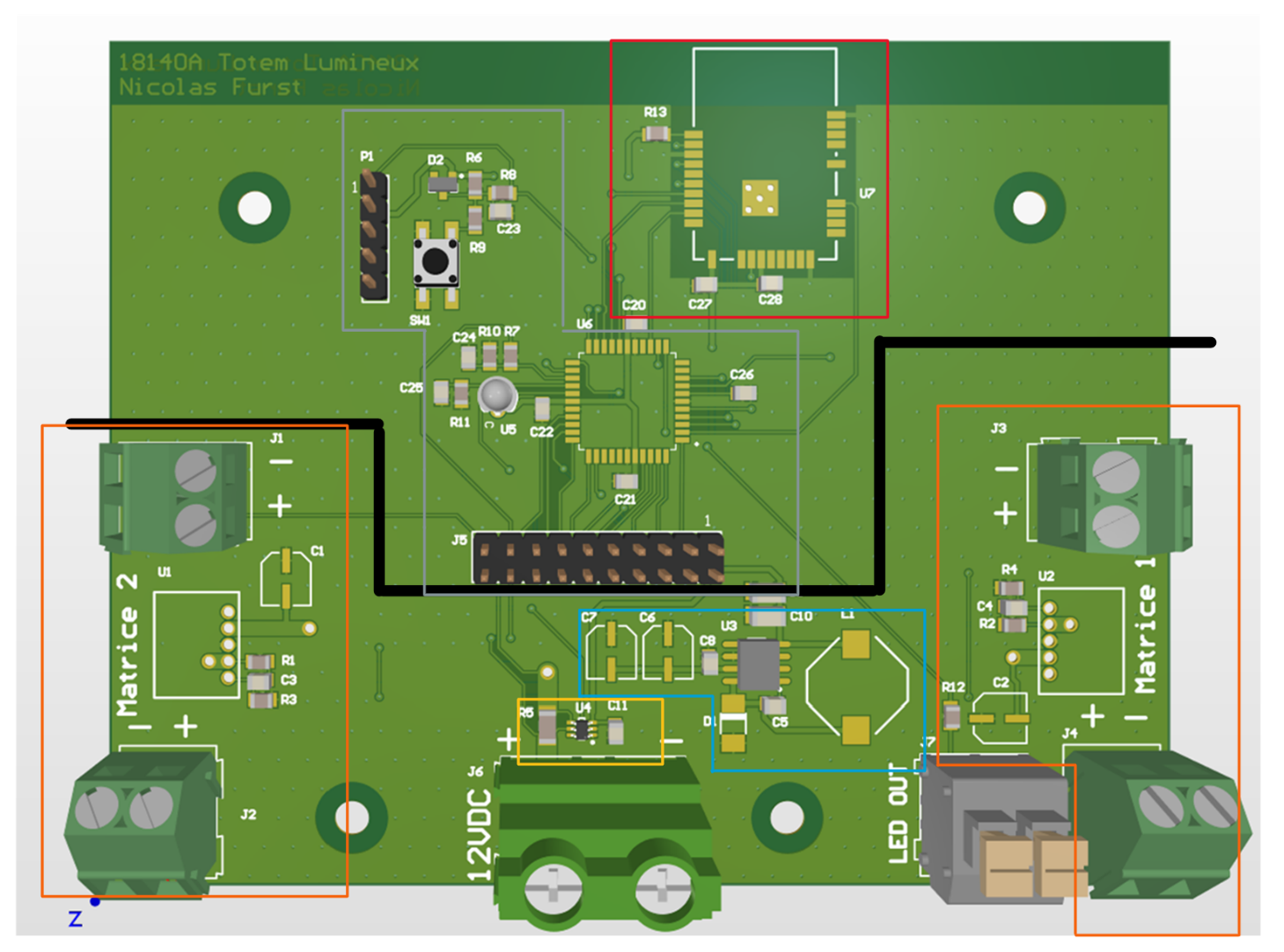
## Dimenssionnement de la carte

Vus que aucun boitier ou design particulier n’a été imposé dans le cahier des charge, il n’y avait pas réellement de contrainte concernant la taille du PCB. Cependant de légère contrainte son apparu lors de la phase de design. Il a été décider de placer le PCB au centre des deux matrices de LEDs ce qui limite la taille du PCB au périmètre des matrices :

Due aux matrices le PCB ne pouvait pas être plus grand qu’un rond de 204mm de diamètre ou un carré de 144mm x 144mm.

Ces dimension calculé sont plus que suffisante pour les composant qu’il fallait monter, j’ai donc fait le PCB au plus petit et finalement c’est un rectangle de 110mm par 87mm.

## Placement

Pour la plus parts des composants il n’y avait pas de grande contrainte pour le placement ou pour le routage, seul le module wifi et les connecteurs avait un placement restreint et un routage particulier.

Voilà à quoi ressemble la carte. La séparation délimite, en haut la partie logique et en bas la partie puissance et alimentation.

### Partie puissance

En bas dans les encadrés orange j’ai placé les borniers d’alimentation des matrices, les deux sont identique et indépendant, deux bornier à droite pour alimenter la matrice droite et pareil du côté gauche pour alimenter la deuxième matrice chacun ayant un régulateur 5V.

De nombreuse contrainte on mener à les placer là, la première est qu’ils doivent être au bord pour droite et gauche pour être plus proche des matrice, ensuite si les borniers sont aussi proches de leur régulateur 5V respectif s’est pour limiter la résistance dans les piste 5V et les effet de résistance dans le plan de masse menant à une masse non uniforme.

À droite il y a également le bornier de data out pour les LEDs car c’est la matrice une qui reçois les informations pour les LEDs.

Enduite au centre j’ai placer le bornier de la batterie et tout de suite après dans l’encadré jaune la mesure du courant. Si la mesure du courant se fait aussi proche s’est simplement pour simplifier le routage du 12V ensuite.

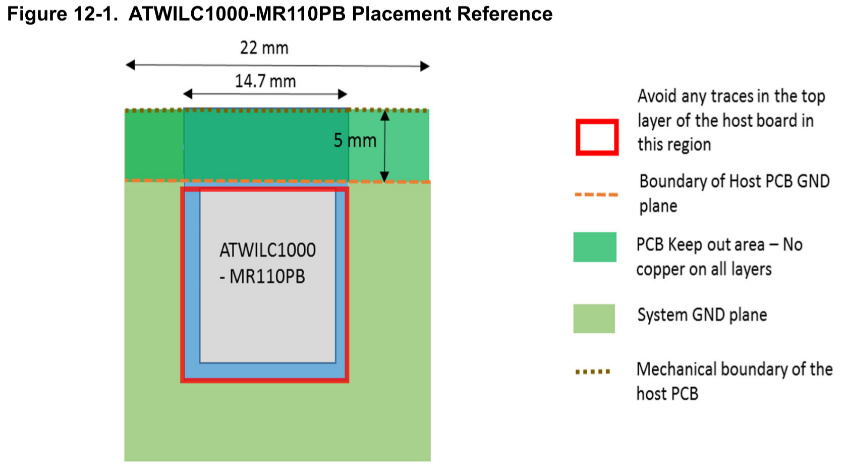
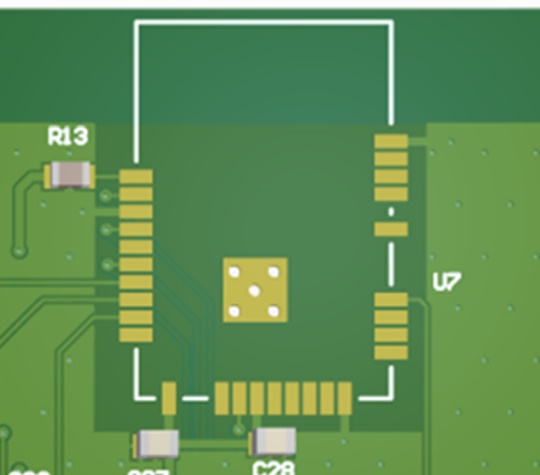
Et finalement il ne restait plus qu’à mettre la partie régulateur 3,3V (encadré bleu) dans l’espace qu’il restait.

### Partie logique

La seul contrainte pour la partie logique concerne le module wifi (dans l’encadrer rouge) qui, pour fonctionner de manière optimale, doit se trouver au bord du PCB.

Le reste des composants (dans l’encadrer gris) a été placé en fonction de la place restante en suivant une logique de base c’est à dire les capacités de découplage le plus proche des composant à découpler etc...

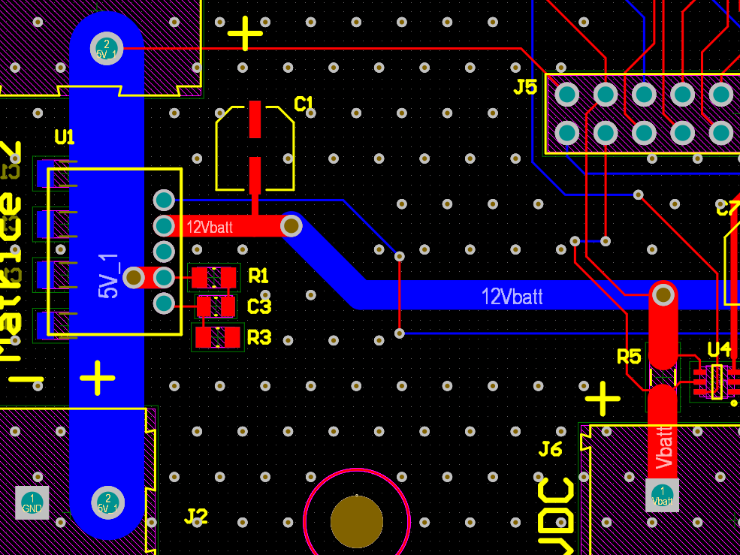
## Routage

Le routage de la partie logique était assez classique, du moins pour toute la partie dans l’encadré gris. En revanche pour le module wifi les contrainte ne s’arrêtait pas à le placer au bord de la carte, le datasheet conseil aussi vivement de retiré tous le cuivre de la couche top directement en dessous du module et de faire une zone sous l’antenne sans cuivre sur aucune des couches :

Consigne du datasheet. Résultat sur le PCB.

Le routage de la partie puissance lui était un peu plus particulier, il a fallu faire attention à la largeur des pistes.

Vus que le 3,3V est prévu uniquement pour la partie logique, le courant traversant les piste serras relativement faible. Les piste font donc une largeur standard pour des alimentation de 0,5mm.



En revanche pour la partie 12V et 5V vus que la puissance nécessaire pour alimenter les matrices de LEDs doit passer dans le piste il faut les dimensionner en conséquence :

Voilà ce que ça donne sur le PCB. On peut voir, en particulier pour le 5V, que les piste qui entre et qui sorte du régulateur 5V sont plus petite quelle devrait s’est dû à l’entraxe du régulateur, si les piste était plus large elles seraient trop proche des pad. Pour le 12V la réduction est négligeable mais pour le 5V elle est importante, j’ai donc routé la piste au top et au bot et j’ai fait la piste le plus court possible.

Après avoir tous routé j’ai ajouté un plan de masse au top et au bot avec un stiching.

## Montage et problême

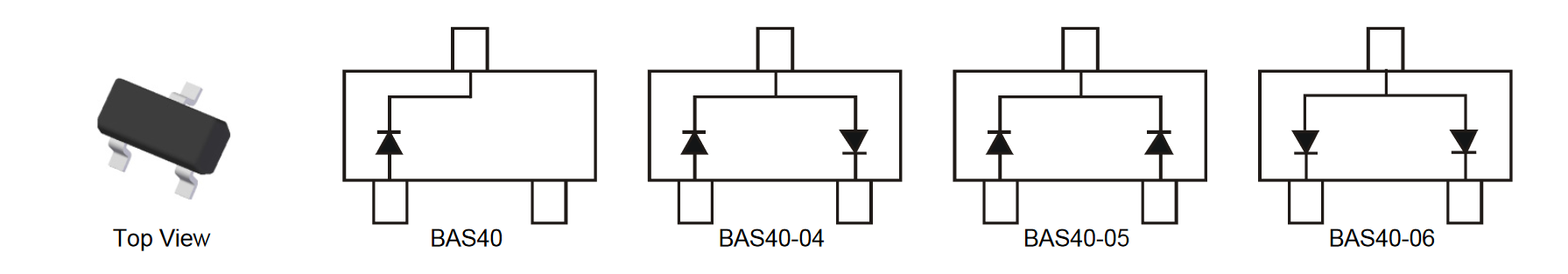
Après avoir reçus le PCB et les composants j’ai vite remarqué qu’il y avait des problème avec certain footprint que j’ai réalisé :

Tout d’abord je me suis trompé sur les dimension de la résistance shunt R5, la résistance que j’ai commandée est du 2512 mais le footprint que j’ai utilisé n’est que du 1206, par chance grâce au via en dessus de la résistance j’ai pu souder la résistance en limitant un maximum l’ajout de résistance indésiré.

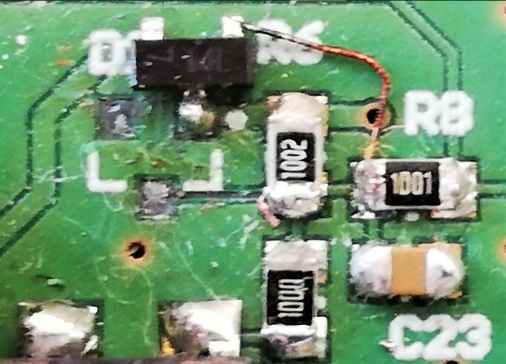


Le deuxième footprint qui à poser un problème est celui des condensateur polarisé de 47uF, en plus d’avoir assigné les pin à l’envers, je n’ai pas mis suffisamment de tolérance par rapport aux indications du datasheet ce qui fait qu’ils n’étaient pas faciles du tous à souder, exactement le même problème est survenu pour la bobine. Au finale cette erreur n’impact pas le produit finale mais elle a posé beaucoup de problème lors du montage.

La première chose que j’ai soudé et testé sont les alimentation ce qui m’a très vite permis de remarquer que j’ai également assigné les pin des régulateurs 5V à l’envers. Par chance en avoir testé un à l’envers ne la pas détruit et le dessouder puis le resouder à l’endroit à résolue le problème.

La dernière erreur commise lors du montage concerne le reset du pic qui ne fonctionnais pas avec le bouton. J’ai assez vite remarqué que le problème venait de la diode D2. Pendant la confection du PCB j’ai décidé de prendre la même diode que celle utilisé pour la carte de développement pic 32 de l’ETML-ES parce qu’il y en avait en stock, la diode BAS40. La diode était disponible dans le vault de Altium ce qui m’a épargné la confection du schéma et du footprint, seulement la diode BAS40 qui est dans le vault est une BAS40-06 et on peut voir dans le datasheet que la BAS40 n’a pas la même configuration que la BAS40-06 chose que j’ai remarqué trop tard :

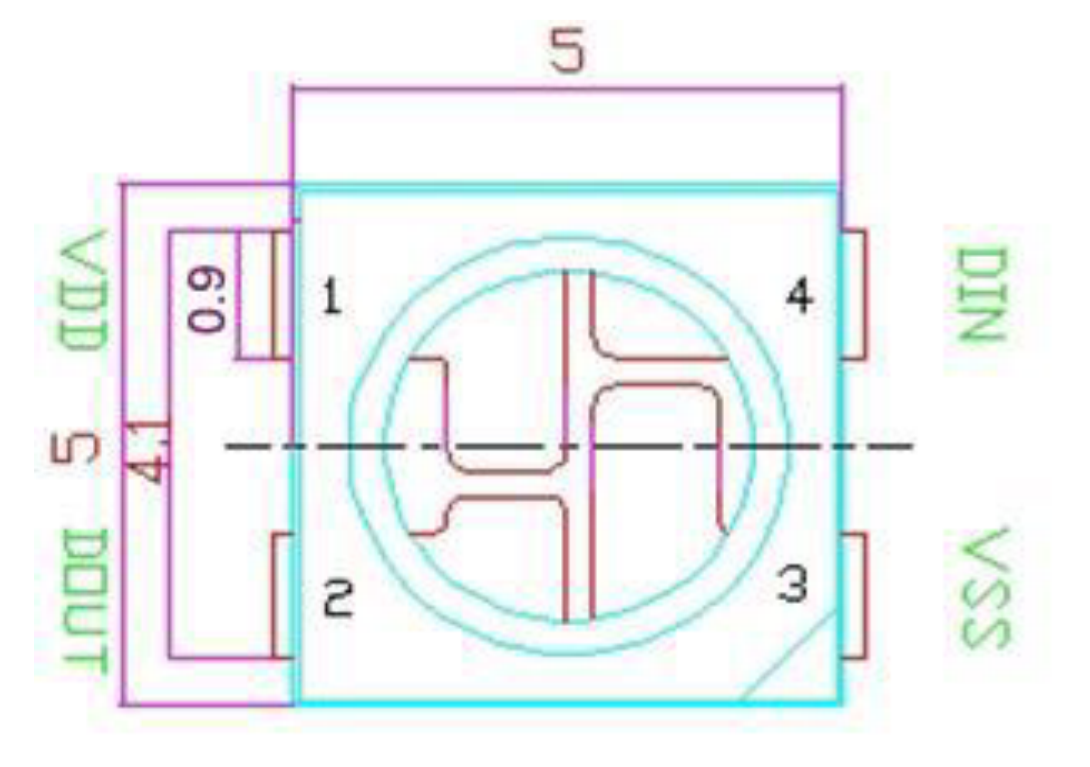
J’ai cependant réussi à corriger cette erreur efficacement en recablent une diode BAS40 pour éviter de perdre du temps avec la commande d’un nouveau composant :



En revanche le footprint le plus sensible que j’ai dû réaliser, celui du module wifi est juste et j’ai réussi à le souder sans difficulté.

# Communication avec les LED

La gestion de la communication des LEDs est un des point les plus important et le premier auquel je me suis attaquée.

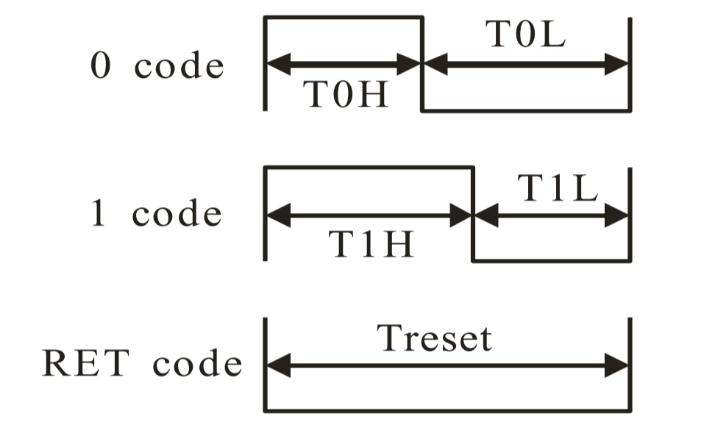
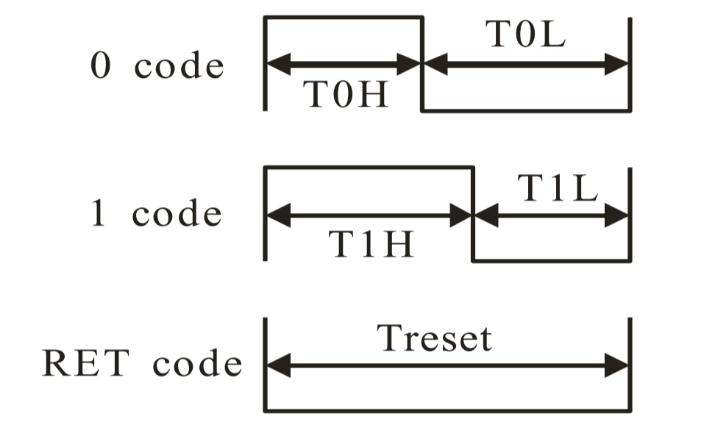
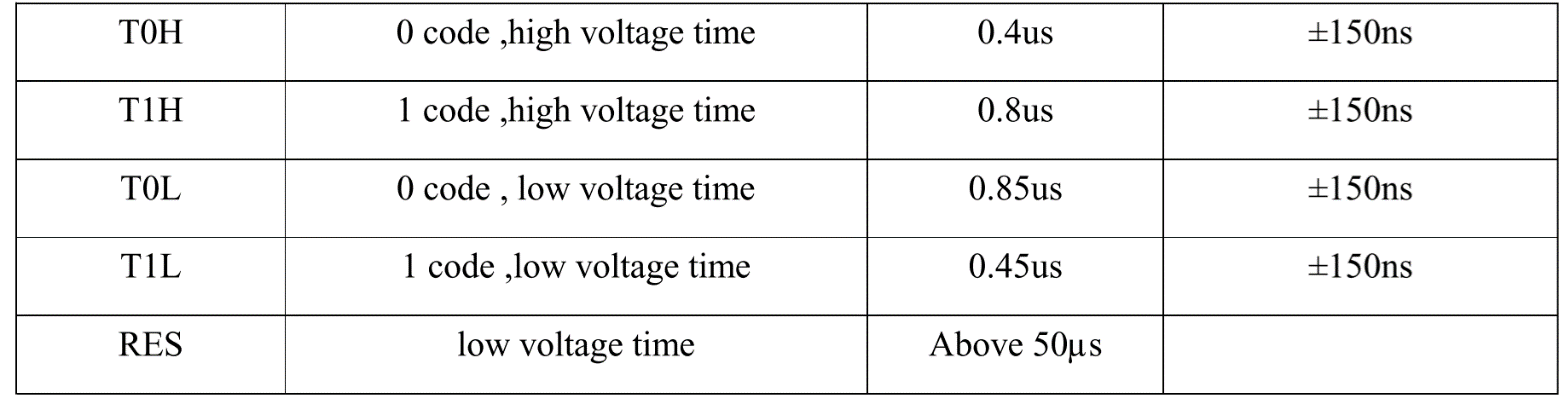
Les matrices sont composées chacune de 256 LEDs RGB adressable WS2812B ou de l’équivalant SK6812 pour un total de 512 LEDs. Ces LEDs comporte 4 pins :

Les pins d’alimentation VDD et VSS et le pins de données Din et Dout respectivement data in et data out.

La ou ces LEDs sont particulière par rapport à des LEDs RGB classique c’est la manière dont elles sont commandées. Chacune des LEDs est composer d’un chip qui vas lui-même adapter la tension de chacune des composante de la LED RGB en fonction des donné reçus sur sa pin Din. Une LED a besoin de 24 bit pour être commander, 8 bit par couleur, ce qui donne un total de plus de 16 million de couleurs.

Sur la matrice les LEDs sont simplement placée en séries. Pour les commandée il faut envoyer une tram du nombre totale de LEDs fois 24 bit. Chacune des LEDs vas lire les 24 premier bit puis transmettre le reste de la tram à la suivante, ce qui fait que la tram vas en quelque sort se consommer un peu plus après chaque LEDs.

Les LEDs utilises leurs propre protocole :



Vu qu’il n’y a pas de transmission d’horloge il faut un moyens de synchroniser les données et un moyens de différencier un 0 de aucune donnée. C’est pour ça que le 0 comme le 1 ont un temps high voltage et un temps low voltage.

Pour générer ces 1 et ces 0 j’ai utilisé le SPI 2 et plus précisément le SDO2 que j’ai placer sur la pin 23 du Pic grâce à Harmony. Si j’ai choisi un SPI pour générer le signale de commande des LEDs s’est parce que s’est facile de générer les 1 et les 0. Il suffit d’utiliser un byte de SPI comme bit du signale, par exemple en envoyant 11000000 sur SDO2 avec le bon baud rate la première LED vas le comprendre comme un 0 et en envoyant 24 bytes d’affiler la première LED vas mettre ses trois couleur à 0.

Maintenant que la théorie est expliquée il faut passer au calculs. La première chose que j’ai fait est de trouver un temps identique pour T0H, T1L et idem pour T1H, T0L maintenant ses temps ne peuvent pas être choisi arbitrairement, il faut qu’il soit dans la tolérance et également qu’ils respectent la règle suivante

:

Après quelque essais j’ai trouvé Valeurs trouvé qui respecte toutes les conditions. Avec ces valeurs et la formule juste en dessus on peut également trouver le nombre de bits du SPI correspondant pour chacun des temps. T0H et T1L corresponde à 2 bits, T1H et T0L corresponde eux au reste du byte donc 6 bits. Donc en envoyant 11000000 on obtiendra un 0 au LEDs et si on envois 11111100 un 1.

La deuxième chose à réglé est le baud rate du SPI. En sachant que les LEDs travail à 800kHz ça veut dire que le SPI doit envoyer les bytes à 800kHz. On trouve donc facilement le baud rate avec le calcul suivant :

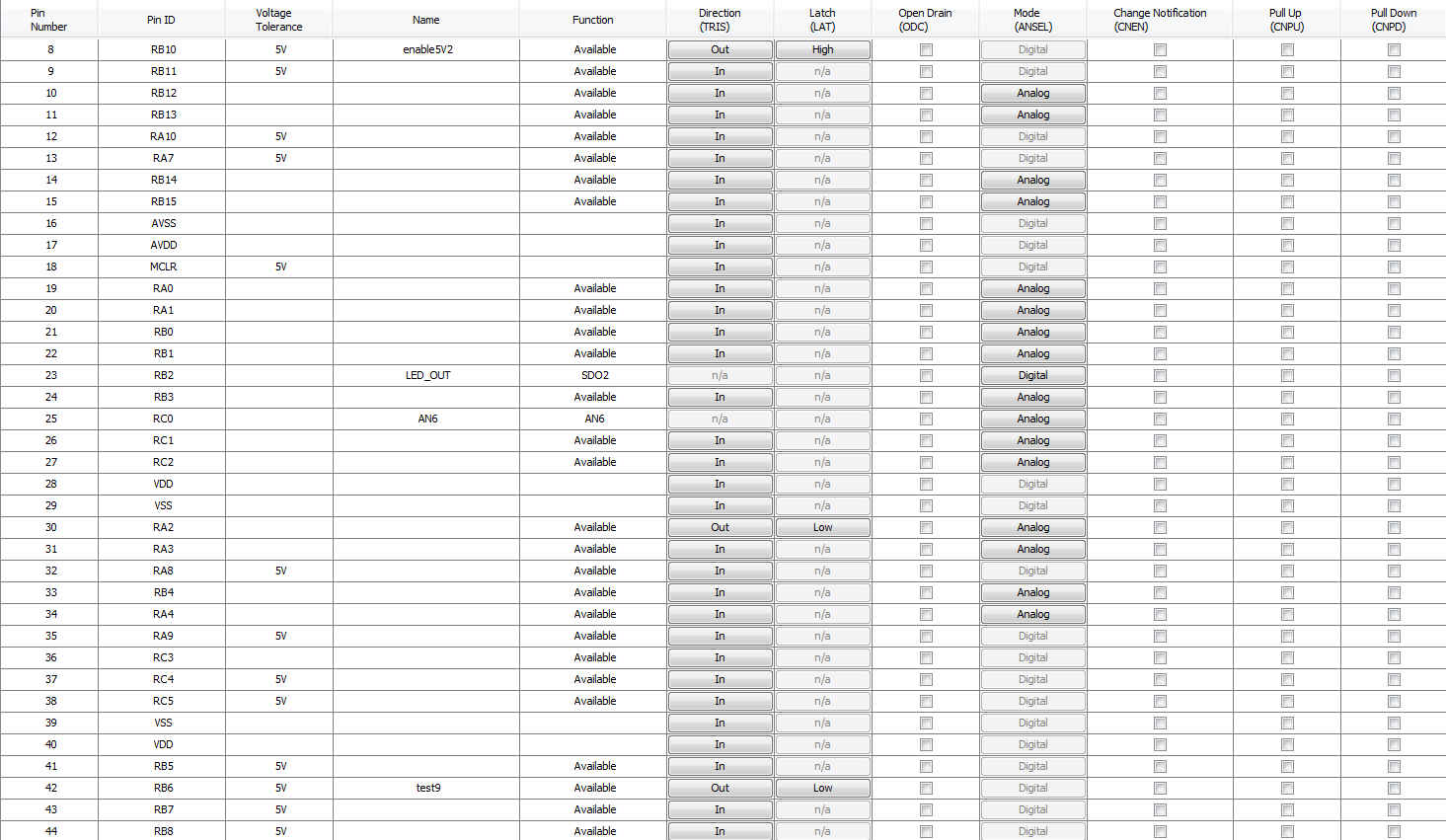
Cette configuration fonctionne mais on peut réduire le baud rate du SPI sans perdre en précision. Il est possible de réduire le baud rate de moitié, car les ratio de bits nécessaire pour que les LEDs interprète un 1 ou un 0 peuvent être divisé par deux sans que les délai T0H,T0L,T1H et T1L ne change. Ce changement implique qu’il faut envoyer les donner par paire de bits car avec un byte du SPI on envois désormais deux bits mais vus que les LEDs ont besoin chacune de 24 bits le nombre de bit à envoyer serras toujours paire quelle que soit le nombre de LEDs à commander. En plus de permettre au SPI de travailler moins vite cette solution à un autre avantage qui est celui de réduire le code. Le baud rate du SPI serra donc deux fois plus petit à 3,2MHz à la place de 6,4MHz, et désormais pour un 0 il faudra écrire 1000XXXX et pour un 1 il faudra écrire1110XXXX.

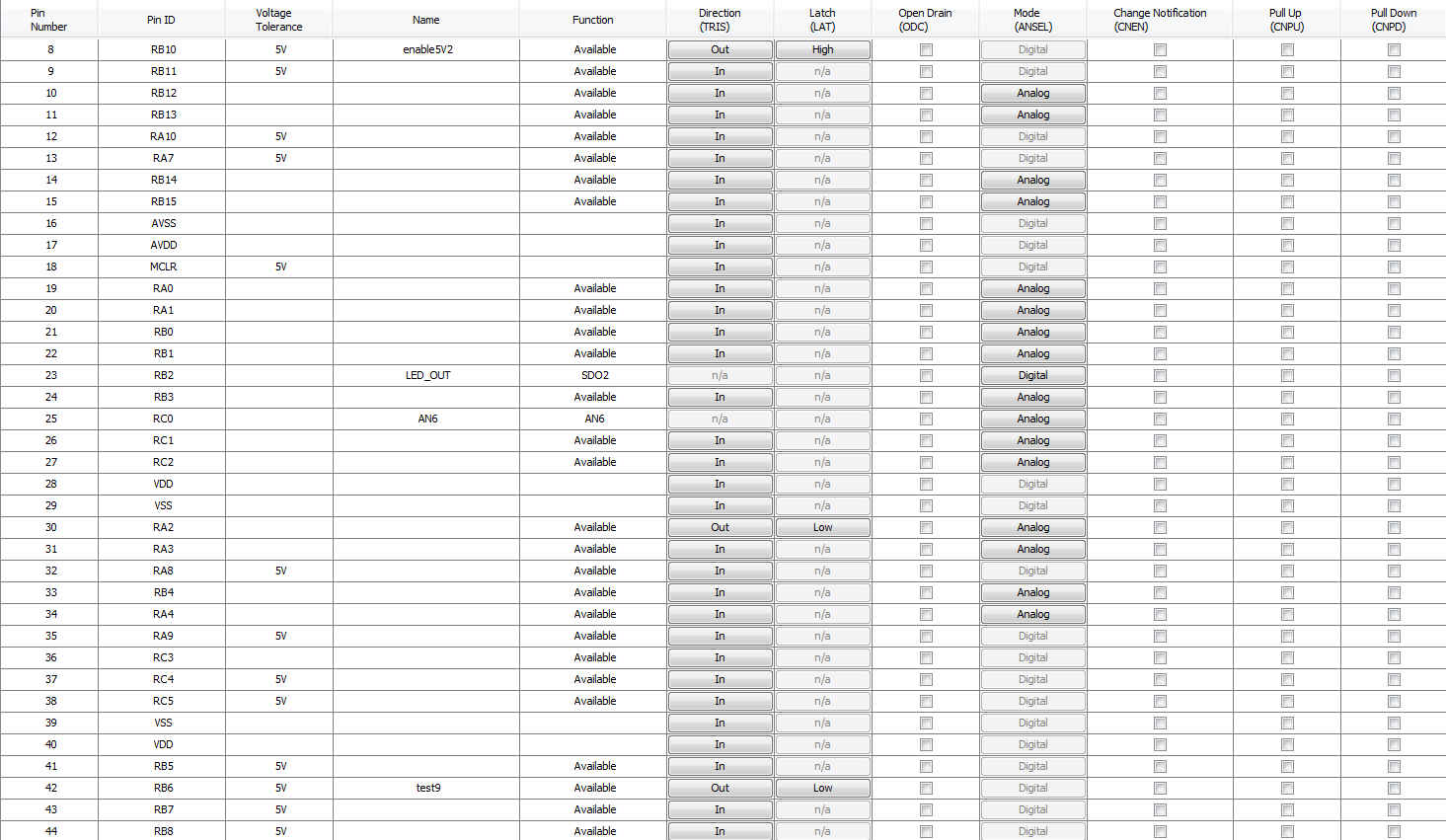
# Code

## harmony

### Pin settings :

Il y avait plusieurs chose à régler avec Harmony, ici on peut voir la configuration des pins :

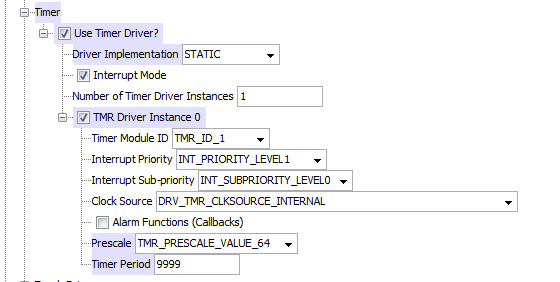




Les deux seul pin vraiment importantes sont la pin 8 qui est l’enable des deux régulateur 5V et la pin 23 qui est la pin de commande des LEDs.

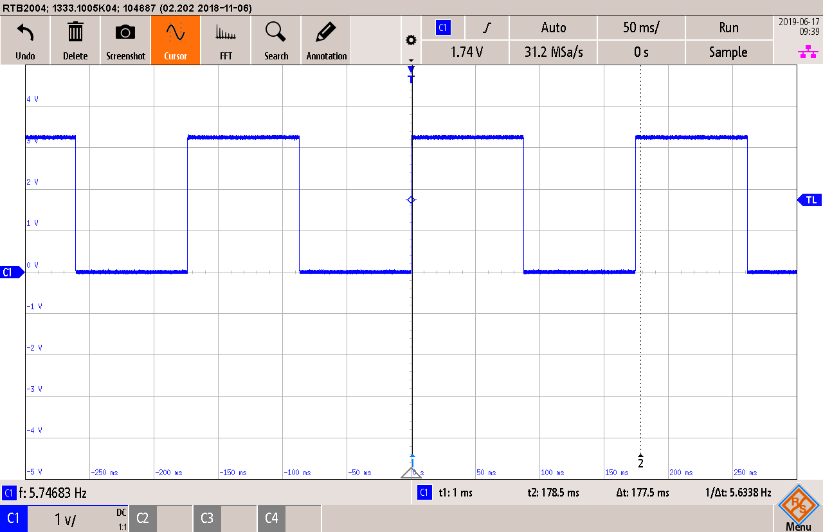
### Timer 1 :

Le reste de la configuration se fait dans les options. Tous d’abord il a fallu un Timer pour cadencer l’application, s’est lui qui vas déterminer la vitesse de défilement du texte :



Ici le Timer Period n’est pas très important car il est tout de suite modifié dans le code mais le Prescaler lui a été choisi de manière que la fréquence du timer ne soit pas trop haute ce qui ferais que le code n’a pas le temps de mettre à jour les matrice qu’il doit déjà les remettre à jour et pas trop faible non plus pour éviter que le texte mette 10 minute pour faire le tour du totem. Après plusieurs test j’ai remarqué que le prescaler idéal été 64 ce qui permet une fréquence du Timer 1 entre :

Et :



Test du Timer 1, Timer Period = 65535, toggle de la pin RB6 à chaque interruption donc f = 2\*f mesuré.

Sur le Screenshot on voit une fréquence de 5,74Hz ce qui fait une interruption à 11,48Hz ce qui est très proche des 11,44Hz calculé.

### SPI

Le projet contient deux communication SPI mais n’ayant pas eu le temps de m’attaquer sérieusement à la programmation du module wifi je n’ai pas initialisé le SPI servant à la communication avec le module wifi.

Vus qu’il ne faut que le SDO2 pour communiquer avec les LEDs la configuration du SPI n’est pas très compliquer :



Le plus important a changé est le Clock/Baud Rate qui a été mis à 3,2MHz.

## LED

L’écriture sur les matrice est faite avec deux fonction. La fonction Led\_Set() et la fonction WriteToLED.

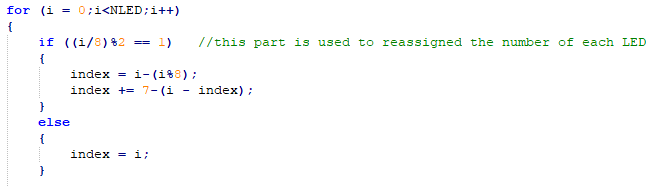
La Fonction Led\_Set() est très basic, la seul chose qu’elle fait s’est écrire des 0 dans le SPI pendant environ 50us le temps de reset des LEDs. Cette fonction va permettre de dire au LEDs d’afficher ce qu’elles ont reçus.

La fonction WriteToLED elle est un peu plus compliquer, elle s’occupe de prendre un tableau, colorMatrix, d’un nombre de case égale au nombre de LEDs contenant une structure, colorStruct, de trois uint8 par case représentant les trois couleurs RGB de chaque LEDs et de convertir ces valeurs en une tram valide pour les LEDs à envoyer sur le SPI. Ce tableau permet de modifier facilement la couleurs de chacune des LEDs avec des fonctions relativement simple.

La première chose que fait la fonction est de réassigner les cases du tableau, car dans mon code je considère que les LEDs sont router comme ça sur les matrices :

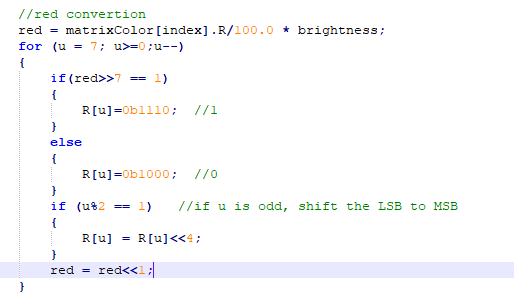
Alors qu’elles sont réellement routées comme ça :

La première configuration permet de grandement simplifier le faite de faire bouger les lettres ou les animations sur la matrice. Dans le code au lieu de réassigner tous le tableau j’ai juste adapté l’index utilisé pour le lire :



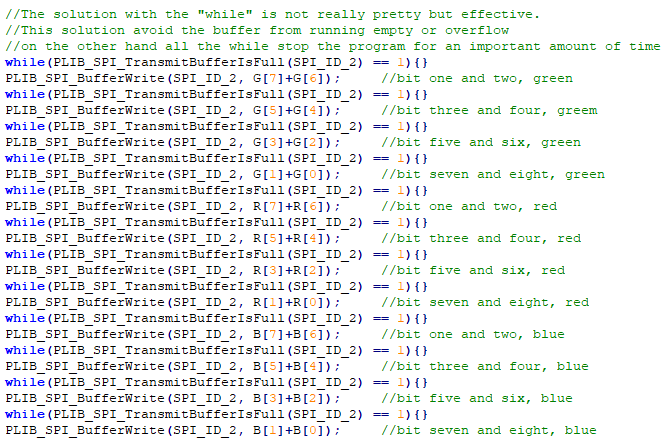
Voilà ce que fait cette partie de code, si la colonne sur laquelle pointe i est impaire l’index est adapté, sinon il vaut i.

La deuxième chose que vas faire la fonction est de convertir la valeur 8 bit de chaque couleur de la case pointer par l’index :



On peut voir ici la couleur rouge mais les trois couleur marche exactement de la même manière. La valeur rouge du tableau est testé bit par bit puis selon si c’est un 1 ou un 0 vas être converti en 1110 ou en 1000 puis assigné à la case correspondent au bit d’un tableau de 8 case. Puis vus que l’on envois deux bit par byte de SPI si le bit converti est impair il serra shift de 4 bit en direction du MSB pour simplifier sont addition plus tard.

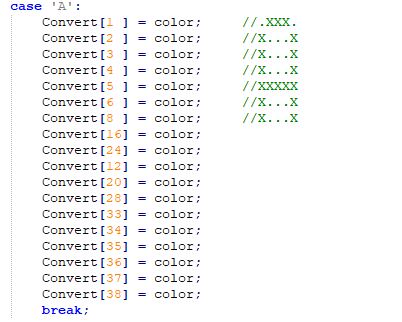
La dernière partie mais pas des moindre est celle qui s’occupe de remplir le buffer du SPI :



Cette partie m’a causé beaucoup de problème car bien que le datasheet du pic est la conception du SPI suggère que en cas de non transfère de donner le SDO garde son dernière état réalité le SDO se met à l’état haut ce qui pose de gros problème pour les LEDs. Il faut donc s’assuré que le buffer ne soit jamais plein et encore moins vide. La fonction s’exécute bien assez vite pour que le buffer ne soit jamais vide cependant il se remplit très vite d’où les while qui font attendre le code tant qu’il n’y a pas au moins une place dans le buffer. Cette solution stop le program ce qui n’est pas parfait mais l’écriture se fait quand même suffisamment vite pour ne pas gêner le reste du programme.

## Message

Pour écrire des massages sur les matrices il faut une librairie de caractères. Une fonction qui, une fois appelé, vas retourner un tableau contenant les LEDs à allumer pour faire la lettre voulu. Voilà ce que ça donne dans le code pour la lettre A :

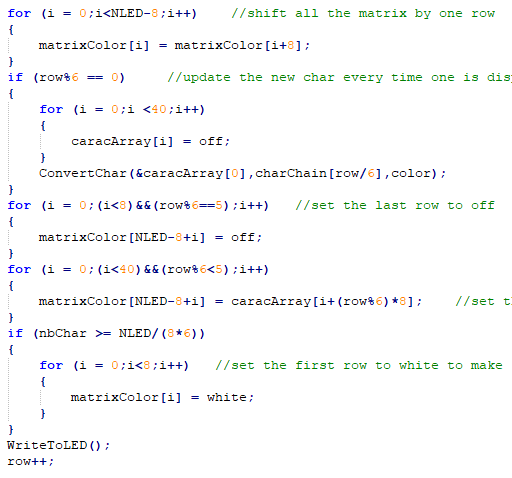


La fonction qui s’occupe de faire c’est la fonction convertChar() qui avec l’adresse du tableau de 40 case, un caractère de la table ascii et une couleur vas remplir le tableau avec le caractère d’entré converti. Pour l’instant la fonction peut convertir 53 caractère faisant partie de cette liste :

.',-\_/\*+=:()%°$!?0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ

À cette liste s’ajoute l’espace qui enfaite est le tableau vide et un damier si le caractère voulu ne correspond pas à la liste. Il est possible de mettre à jour cette « librairie » de caractère avec d’autre caractère mais sans trop la modifier il faut que les caractère respectent les règle suivantes : ils doivent faire partie de la table ascii non-étendue, ils ne doivent pas une foi converti prendre plus de place sur la matrice que 5x8 LEDs et ça ne peut pas être le caractère ‘\0’.

Une fois la librairie de caractère faite il faut afficher s’est caractère sur les matrices, s’est ici que la fonction Text() entre en jeu. Cette fonction fonctionne de la manière suivante : Elle prend le premier caractère de la chaine qu’il lui a été passé en paramètre et le convertit avec la fonction CaracLib() ; ensuite elle décale tous le tableau matixColor d’une colonne vers la gauche ; puis elle met à jour la dernière colonne du tableau matrixColor avec la première colonne du tableau de conversion du caractère.



La fonction va également analyser la taille du message et s’il est suffisamment petit pour être afficher en entier sur les matrices il tournera à l’infinie Contrairement à une phrase un mot suffisamment court de mois de X caractère est lisible tout autour du totem. Si le message est trop grand la première colonne s’allumeras en blanc pour marquer la différence entre le début du message et la fin car là où le début et la fin du message se rencontre si-il n’y a pas d’indication les lettres donne l’impression de se mélanger et le résultat est très perturbant.

La fonction Text() a besoin en entré d’une chaine de caractère contenant à la fin le caractère ‘\0’ sinon la fonction ne sait pas où se trouve la fin du message et il risque d’y avoir des overflow avec les index de tableau.

## Autres fonctions

Mon code est composé d’autres fonction un peu plus secondaire que je vais un peu moins détailler que les précédentes mais s’est quand même intéressent de s’arrêter dessus.

Dans le fichier Anim.c en plus de la fonction Text() j’ai fait une fonction Nyan() et une fonction Battery(). Les deux fonctions marchent de manière assez similaire.

La première, Nyan(), est une animation de la célèbre vidéo Nyancat, je l’ai principalement réalisé pour faire une sorte de démonstration de ce que l’on peut faire avec les matrices. En plus de la fonction Nyan() je voulais implémenter une fonction PacMan() mais le manque de temps m’en a empêché.

La fonction Battery() elle affiche des batterie toute autour du totem qui vont afficher le niveau de l’alimentation. Cette fonction est surtout là pour montrer que la lecture de la tension fonctionne il manque toute la calibration, pour l’instant les batteries afficher son plein si la tension vaut 14V et diminue de manière linéaire jusqu’à 10V pour afficher des batteries vides.

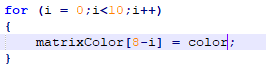
J’ai réalisé une autre fonction qui a besoin de réglage c’est la fonction AutoBrightness() qui comme son nom l’indique règle la luminosité des matrices en fonction de la lumière ambiante, cette fonction marche étonnamment bien mais elle a besoin d’être calibrer. Il faudra peut-être même changer la résistance R11 sur le PCB pour modifier la sensibilité du capteur.

Les fonction d’initialisation et de lecture des ADCs ont été copier des anciens TP et exercice réaliser autour des ADCs. Presque aucune modifications n’a été requise.

Les dernières fonctions sont des fonctions plutôt utilitaire comme la fonction FullColor() qui permet de mettre la même couleur dans tous le tableau matrixColor. Cette fonction est utilisée principalement pour remettre tous le tableau dans un état LEDs éteinte. La fonction RainbowPick() elle est une fonction qui permet de return une couleur, au format colorStruct, faisant partie du spectre de l’arc-en-ciel. Cette fonction est très utile pour faire une boucle couvrant toutes les couleurs ou choisir une couleur en étant sûr qu’elle soit jolie et à la même luminosité.

## Problème et module wifi

En réalisant ce programme j’ai rencontré plusieurs problème, certains relativement classique comme des oubli d’initialisation de variable etc. et d’autre plus embêtent et qui mérite qu’on si attarde. Le premier était celui concernant la communication des LEDs et le SPI que j’ai déjà détaillé en dessus. Le deuxième gros problème que j’ai rencontré lui concerne l’écriture des message sur la matrice. Pendant l’implémentation de la fonction Text() j’ai fait une erreur qui faisait que l’index du tableau matrixColor réalisait une formule pouvant donner un résultat négatif. Voilà à quoi ressemblais le problème en simplifier dans le code :



L’ennui s’est que MplabX compilait le code sans erreur est l’écrivait dans la mémoire mais il ne fonctionnait pas du tout. En plus de compiler sans erreur en mode debug le code ne plantais que plus tard à un endroit qui n’avait aucun problème. J’ai assez vite remarqué que certaine variable qui ne devait pas être modifié prenait de valeurs absolument absurde pour mon code. Et j’ai fini par remarquer qu’un tableau avait parfois un index négatif mais qu’au lieux de planter écrivais la valeur que je voulais lui attribuer à l’adresse de la case en considérant l’index comme non signé et les adresses en question été par hasard celle d’autre variable qui elles, ont causé des problème juste en dessous car elle n’était pas réassignée. Bien que ce problème m’ait fait perdre deux jours j’ai eu de la chance car les variable modifier aurait pu être utile bien plus loin dans le code rendant la détection du problème encore plus compliquer.

Concernant le module wifi qui était une grosse partie du cahier des charges, j’ai essayé sans succès à le faire fonctionner, d’un point de vue hardware toute à l’air correcte s’est du côté software qu’il y a des problèmes. Pour commencer si j’ai choisi le module ATWILC1000 s’est parce que monsieur Castoldi ma conseiller d’utiliser le module MRF24Wn0MA seulement sur le site de Microchip il conseil de prendre le module ATWILC1000 car l’autre est devenus obsolète. En faisant des recherches j’ai remarqué que Microchip fournissait un exemple spécifique pour le ATWILC1000 ce qui aurait dû rendre l’utilisation du module très similaire au dernier TP de MINF sur la communication Ethernet.

Dans un premier temps j’ai essayé d’adapter l’exemple à mon projet. J’ai très vite remarqué que le code était grandement basé sur FreeRTOS après avoir analyser le code j’ai demandé de l’aide à monsieur Bovey qui en analysant le code à pus me renseigner sur le fonctionnement de l’exemple. Tout d’abord l’exemple est prévu pour fonctionner avec une carte de développement faite par Microchip grâce à l’laquelle le module est programmer par Uart.

C’est ici qu’est la première difficulté, il faut trouver les commande SPI pour programmer le module que je n’ai pas réussi à trouver. C’est à ce stade que j’ai abandonné faute de temps.

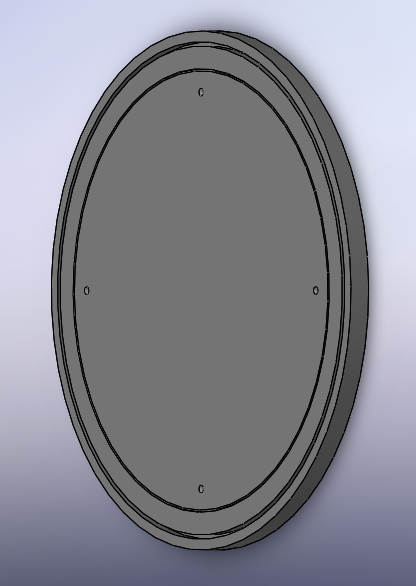
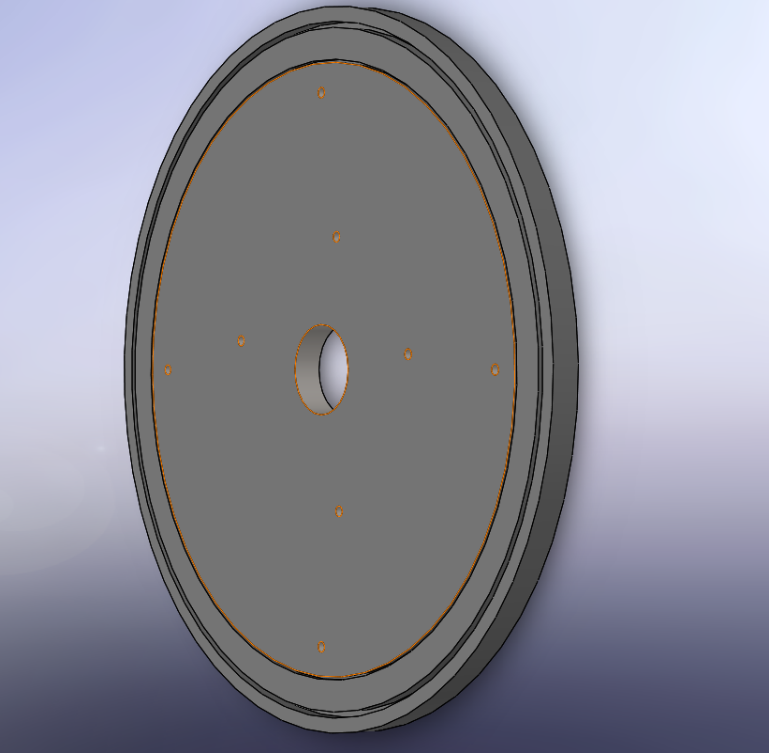
Pendant mes recherche j’ai eu pas mal de peine à trouver des gens ayant fait un projet fonctionnel sur le module, le forum le plus constructif que j’ai trouvé est celui-ci :

https://www.avrfreaks.net/forum/atwilc1000-sd-evaluation-code

Ici la personne essaye comme moi d’utiliser le module via SPI sans succès et conseille vivement d’utiliser un autre module comme le ATWINC1500. Je regrette de ne pas avoir pris le module que l’on m’a conseillé.

# Mecanique

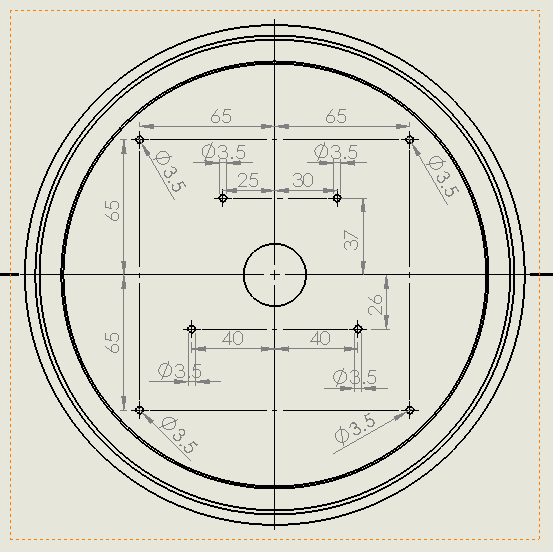
Après avoir fait le PCB avec Altium j’ai commencé le design du boitier. J’ai donc suivi l’idée développer dans la phase de pré-étude qui était de faire deux disque qui prennent en sandwich les matrices laissant de la place au milieu pour monter le PCB. J’ai donc développé les deux disque en 3D sur Solid Works l’utilisation du programme était assez intuitive, je n’ai pas rencontré de problème particulier. Voilà le résultat :



Disque du bas. Disque du haut.

Les deux disque sont presque identique, la seul différence entre les deux est les quatre trous de Montage du PCB et le gros trous au centre qui sert à faire passer les câbles d’alimentation. Sur les disques on peut apercevoir deux rainure au tour, la première sert à tenir les matrice correctement en rond el la deuxième sert à mettre un plexiglass souple pour d’une part essayer de rendre le Totem imperméable et de pouvoir dissipé un peu les LEDs en ponssant le plexiglass ce qui rend également les lettres et animation plus lisible. Finalement les quatre trous qu’il y a sur les deux disques servent à maintenir les disques avec des entretoise.

Après avoir coté les disques j’ai transmis les plan au secteur mécanique de l’ETML qui on accepter de les réaliser pour moi sur une CNC. Une fois les deux disques reçus j’ai remarqué que malheureusement l’étudiant qui s’est occuper de les usiner à lu les dessins en miroir ce qui cause les trous de fixation du PCB à ne pas être aligner, seulement deux trous correspondes à la fois cette erreur n’est pas spécialement gênante, les deux tous aligner plus le support qu’offre les entretoises suffisent à maintenir le PCB. En vérifiant d’où venait l’erreur j’ai également remarqué que même si le dessin avait été interpréter correctement seul trois trous aurait été aligné car j’ai mal placé un des trous :

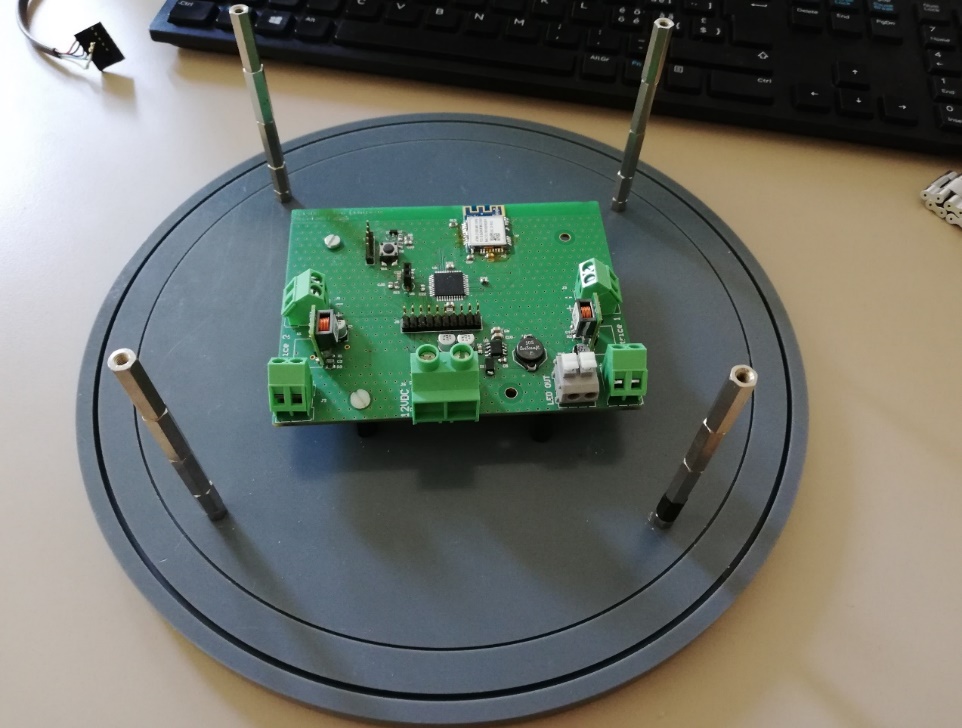


Cette côte aurait dû être à 15 au lieu de 25.

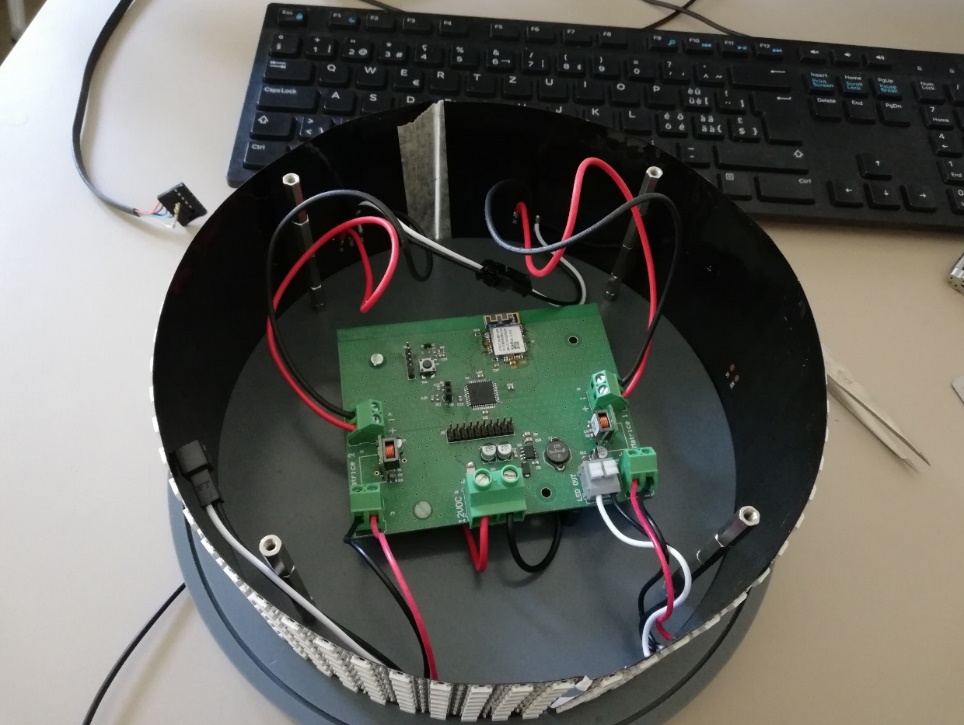
## Montage

L’or du design du boitier le montage me paraissait simple et évident mais finalement il s’avère bien plus compliquer que prévu.

1. Placé les entretoises et monté le PCB sur le disque du bas.



Il est important que les longues entretoises soient plus courtes que la hauteur des matrices mais plus longue que les matrices lorsqu’elles sont placé dans les disque pour éviter de les écraser. Ici on peut voir un élégant empilement d’entretoises forment quatre longues entretoises de 78mm de haut chacune.

1. Placer les matrices et mettre les câbles dans les bon borniers.

Pour cette étape brancher les câbles d’alimentations avant de placer les matrices est plus simple. Ensuite il faut faire attention de brancher les deux matrices sur leurs alimentations en suivants les légendes sur le PCB.

1. La dernière étape et de très loin la plus compliquer consiste à mettre le plexiglass dans la deuxième rainure, de mettre le disque du haut et de le vissé dans les entretoise.

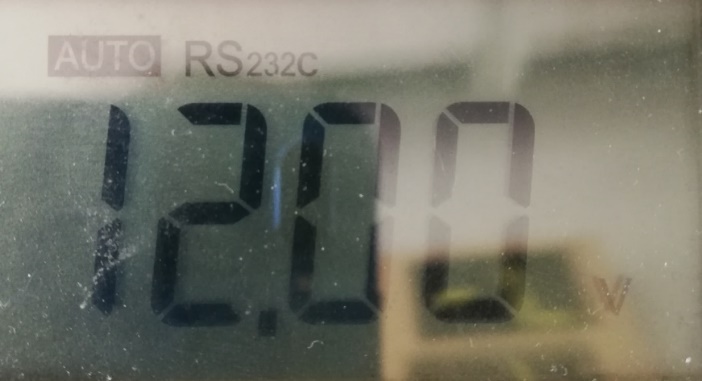
C’est ici qu’on peut remarquer les premier défaut du design, il est très dur de faire rentrer les matrices et le plexiglass dans leur rainure. Il est vraiment important d’être sûr d’avoir placé les matrices correctement dans la rainure avant de visser les deux disques ensemble, car si elles ne sont pas correctement mises elles vont se plier et il sera encore plus dur de les placer correctement par la suite.

Une fois complétement monté et en état de marche on peut remarquer l’autre défaut du boitier. À hauteur d’œil la visibilité des matrices est parfaite mais si le Totem est plus haut ou plus bas certaine rangée de LEDs deviennent cacher par les disques.

# Tests et mesures

## Alimentation

Mon PCB est composé de trois régulateurs, deux de 5V servant d’alimentation pour les matrices et un de 3,3V pour alimenter tout le reste.



D’abords la tension d’alimentation qui est régler à 12V



Ensuite le régulateur de la matrice 1.



Et le régulateur de la matrice 2.

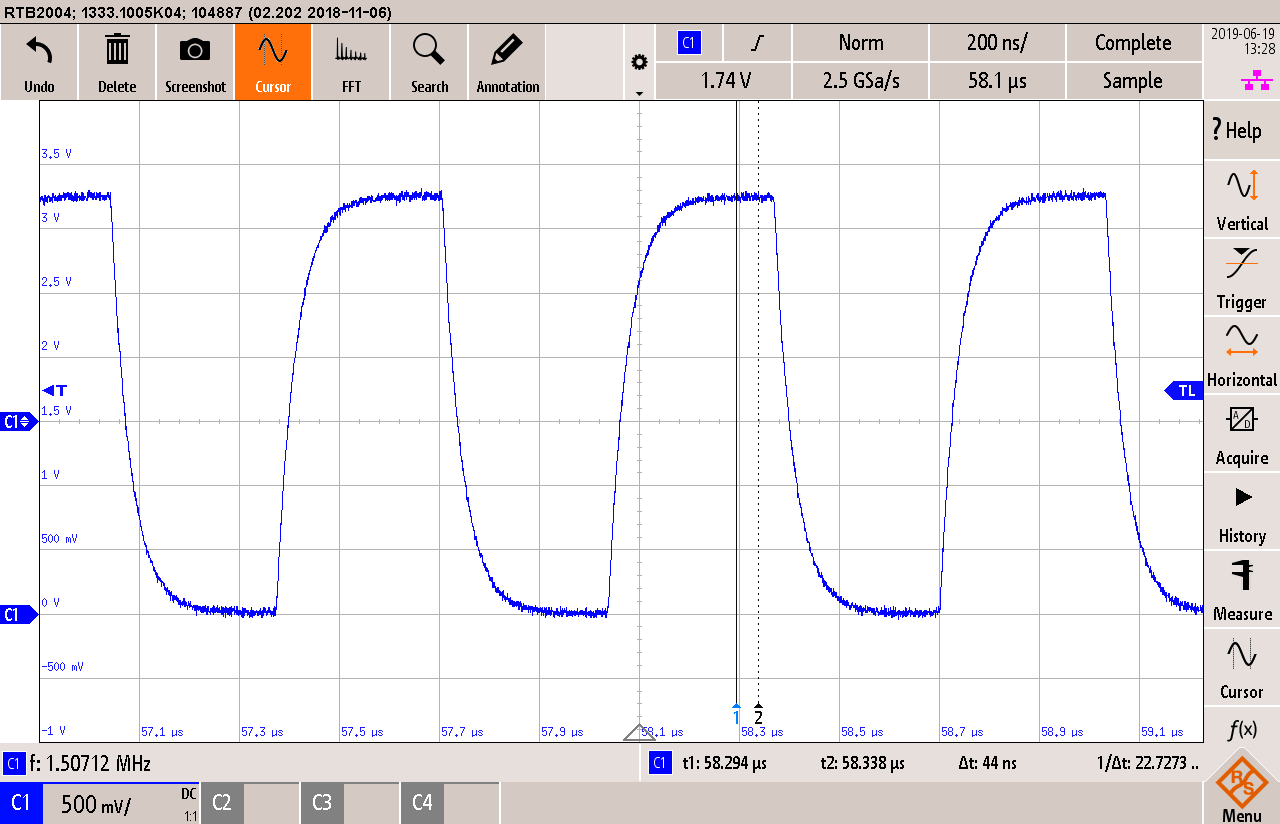


Finalement le régulateur 3,3V.

On peut voir avec ces mesure que le dimensionnement des alimentations est juste et que les tension sont celle calculé.

## SPI

En envoyant en boucle le byte suivant 10101010 on peut mesurer le baud rate du SPI exactement de la même manière que pour le timer c’est-à-dire à la moitié de sa fréquence réel :



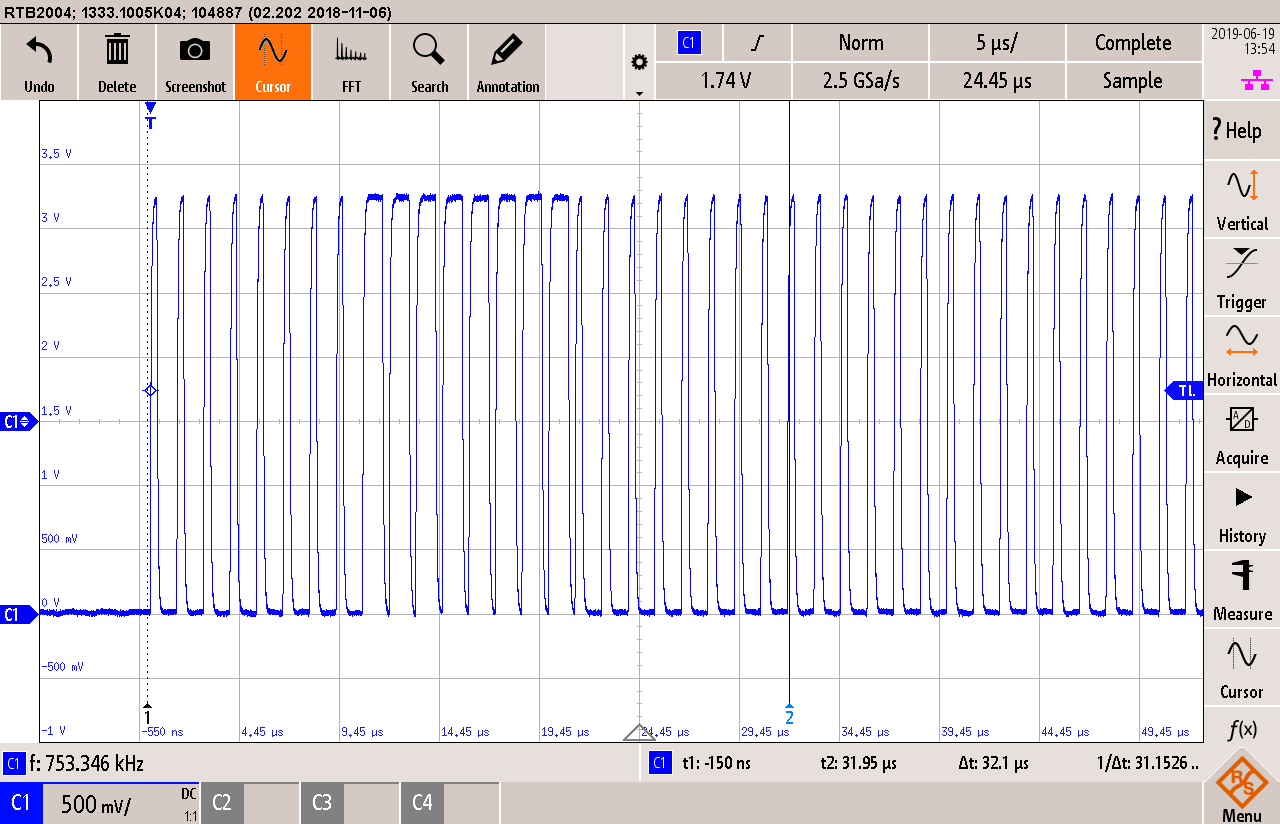
On voit que le SPI atteint 1,5MHz ce qui fois deux fait 3MHz ce qui est assez proche des 3,2MHz.

Si maintenant on allume la première LED en rouge à la luminosité maximum, la tram du SPI devrais ressembler à ça :

8 bits à 0 vert 8 bits à 1 rouge 8 bits à 0 bleu

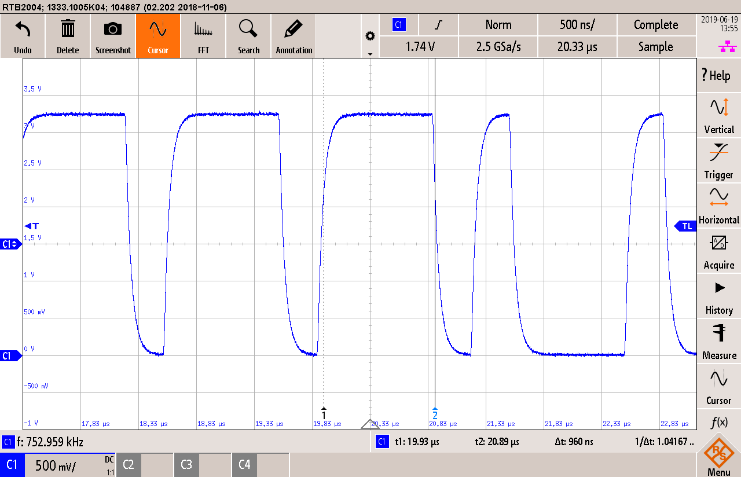
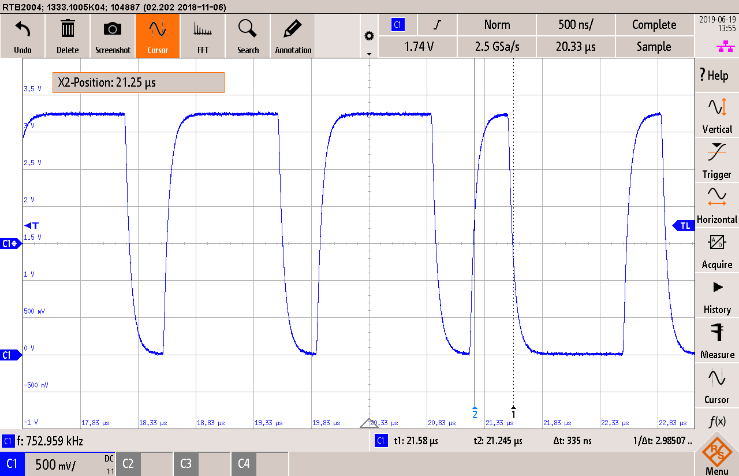
10001000 10001000 10001000 10001000 11101110 11101110 11101110 11101110 10001000 10001000 10001000 10001000

Voilà ce qui est mesuré à l’entrée des matrices :



On voit effectivement la même tram ainsi que les donné pour la suite des LEDs qui sont toutes éteinte c’est donc uniquement des 0 ou des 1000 pour le SPI. Sur le Totem la première LED s’est bien allumé en rouge

Avec cette tram on peut directement vérifier les temps calculer avant et vérifier s’ils correspondent :

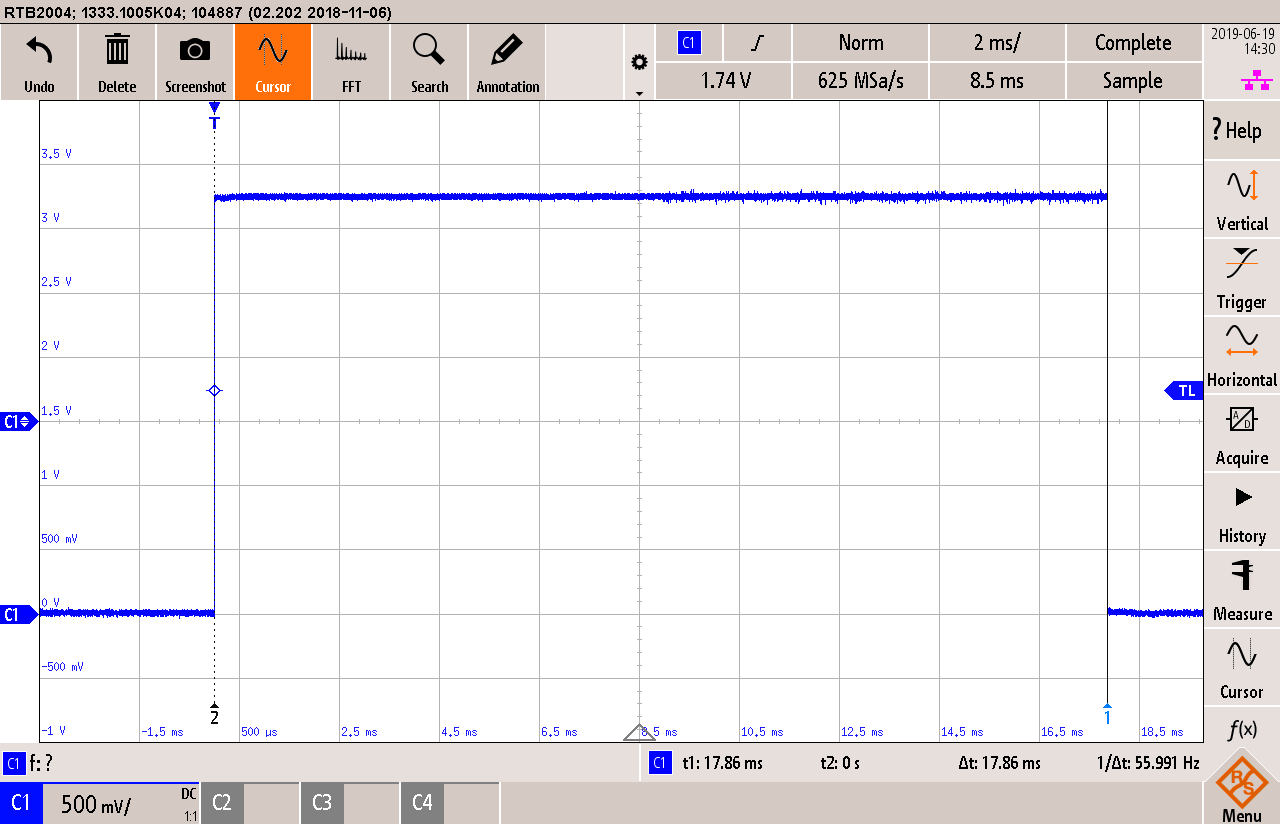


Les temps sont respectés, on est bien dans les tolérance du datasheet, il y à certes une déviation par rapport au calcule mais les valeurs sont encore dans les tolérances et la communication fonctionne. Cette différence avec les valeurs calculer viens du faite que le baud rate du SPI n’est pas pile 3,2MHz mais 3MHz.

## Durée de fonction

Le code varie selon la taille du message à envoyer ou simplement si le totem affiche un message ou un animation il n’est donc pas facile ni vraiment utile de mesurer la durée du code, cependant il y a une fonction qui prend beaucoup de temps, qui est bloquante et qui prend toujours le même temps quel que soit le reste du code, c’est la fonction d’écriture sur les LEDs WriteToLED().

En faisant passé de l’état bas à l’état haut la pin RB6 juste avant l’appelle de la fonction et en faisant l’inverse juste après on auras une bonne idée de la durée de la fonction :

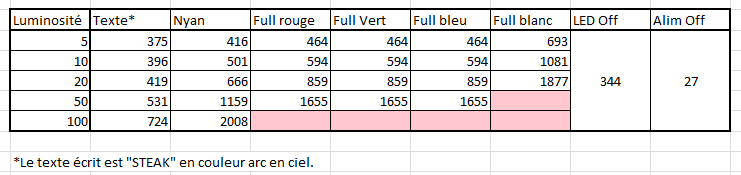


La fonction dure 17,86ms cette information peut paraitre un peu inutile elle nous permet déjà de réaliser le temps qu’il faut pour mettre à jour les matrices et en plus grâce à ce temps on sait que si le Timer 1 crée une interruption plus souvent que toute les 17,86ms le texte n’ira pas plus vite. En sachant ça on peut calculer la valeur minimum du Timer 1 :

La valeur sous laquelle il ne faut pas aller pour le Timer Period est 13395.

## Consommation

Pour mesurer la consommation j’ai fait un tableau sur Excel, toutes les valeur sont en mA, le courant a été mesuré juste avant le PCB avec une tension d’entrée de 12V :

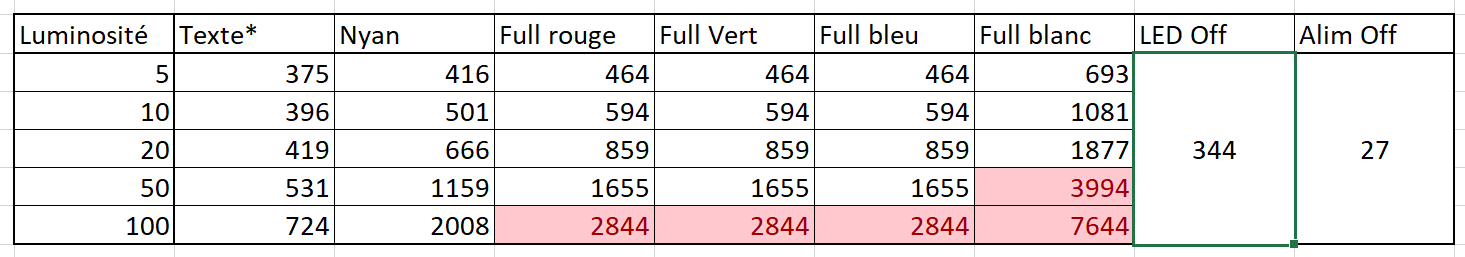


Dans ce tableau on peut voir que j’ai tester plusieurs mode à plusieurs luminosité. Pour commencer on peut voir que quand les LEDs sont éteinte mais alimenté (LED Off) on consomme quand même 344mA ce qui n’est pas négligeable, en mesurant le courant mais cette fois en désactivant les régulateurs 5V on voit que l’on consomme plus que 27mA on peut donc en conclure que rien que d’alimenter les deux matrices par les régulateurs 5V consomme 317mA.

Mise à part ça, on peut voir que quand les matrices sont allumées d’une des trois couleurs RGB (Full rouge, Full rouge, Full bleu) on obtient exactement la même consommation pour la même luminosité ce qui prouve que les LEDs sont bien calibré à l’intérieur. Les cases en rouge indiquent que le courant est supérieur à 3A la limite de l’alimentation. L’alimentation ne peut pas fournir plus de 3A ce qui peut paraitre être un inconvenant, avec une tel consommation les matrices chauffent considérablement et d’après Adafruits qui sont les fabricants les matrices peuvent allez jusqu’à la destruction.

Ce tableau est pratique pour estimer l’autonomie du Totem en fonction du mode, de la luminosité et de la batterie. Vus que la batterie n’a pas été commander et quelle ne serras surement pas la même que celle prévu ça n’est pas très utile d’essayer d’estimer l’autonomie mais on peut déjà voir si la consommation est linéaire en fonction de la luminosité. Pour faire le graph j’ai soustrait la consommation des LEDs éteintes.

Ce graph montre que la consommation des matrices est parfaitement linéaire en fonction de la luminosité. Ce qui peut me permettre de compléter le tableau en extrapolant les valeurs :



Après avoir extrapoler ces valeurs on voit qu’en théorie le Totem peut atteindre une consommation de plus de 7,6A a 12V si toutes les LEDs son allumer en blanc ce qui fait environ 9A par régulateur 5V à fournir.

# Stade d’avancement

La base du Totem fonctionne, il affiche un message sur les matrices. La grosse partie manquante du projet est, évidemment, la communication par wifi mais mise à part ça à condition d’avoir un pc et un programmeur le projet respect le cahier des charge, il est capable d’afficher un message, des animation, le niveau de la batterie, changer sa luminosité en fonction de la lumière ambiante, etc… En plus de la partie wifi toute la partie batterie et panneau solaire a été mise de coter dû à plusieurs chose notamment le prix des composant et le temps restreint. La partie panneau solaire et batterie n’est pas très compliquer commander les composant et les brancher au projet devrais suffire.

Si ce projet est repris je conseil vivement de changer le module wifi.

# Conclusion

En conclusion je suis content de mon travail bien que déçus de ne pas avoir réussi à faire fonctionner le module wifi. Le projet a atteint un stade qui le rend selon moi fonctionnelle et qui respecte les points essentielle du cahier des charge. Le totem est capable d’afficher un message sur les matrices, de mesuré sa tension d’alimentation, d’adapter la luminosité des LEDs en fonction de la lumière ambiante et d’afficher une animation.

J’ai acquis beaucoup de nouvelle compétence grâce à ce projet. J’ai pu voir comment fonctionne la communication des LEDs à communication série et réaliser des fonction capable de communiquer avec les LEDs avec succès, mais là où j’ai appris le plus s’est au niveau de la gestion de sont temps et de la gestion d’un projet. J’ai très mal géré mon temp, j’ai été très optimiste lors de la pré-étude sur le temps que prendrais certaine chose et j’ai oublier pas mal de chose, j’ai oublié de prendre en compte les délai de commande des composant et du PCB ce qui m’a un peu retardé pour le code, je n’ai pas non plus pensé au temps que je passerais à régler des erreurs comme pour le module wifi. Ce manque d’organisation à fait que je n’ai pas réaliser que la fin du projet approchait à grand pas ce qui a grandement réduit le temps nécessaire pour travailler sur le rapport.

Bien qu’ayant fait beaucoup d’erreurs sur l’organisation et la gestion du temps je suis sûr que je réaliserais mais future projet bien mieux grâce à celui-ci.

Je tiens également à m’excuser pour le nombre de fautes que le lecteurs de ce rapport à dû subir malheureusement dû à la mauvaise gestion de mon temps je n’ai pas eu le temps de faire corriger mon rapport ma dysorthographie se reflète donc pleinement au travers de mon rapport.

Nicolas Fürst

# Références

Datasheets utilisés :

* + 1. PIC32MX795H - Microchip

<http://ww1.microchip.com/downloads/en/DeviceDoc/80000480P.pdf>

* + 1. Panneaux de Leds – Adadruit

<https://www.adafruit.com/product/2294#description>

* + 1. Led WS2812B lié aux panneaux de Leds

<https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf>

* + 1. Led SK6812 lié aux panneaux de Leds

<https://cdn-shop.adafruit.com/product-files/1138/SK6812+LED+datasheet+.pdf>

# Annexe

**Annexe 1** : Schéma Alim.

**Annexe 2** : Schéma Pic.

**Annexe 3** : Liste des composant.

**Annexe 4** : Planning réel.

**Annexe 5** : Plan mécanique, disque du bas.

**Annexe 6** : Plan mécanique, disque du haut.

**Autres annexe** : Code.