Rapport de pré-étude

**Ecole supérieure**

Électronique

Laboratoire PROJ

Salle R110

Emetteur DCF

**Réalisé par :**

Culand Julie

**A l’attention de :**

M.Moreno & M.Castoldi

**Dates :**

Début du laboratoire : 21 novembre 2018

Fin du laboratoire : 20 juin 2019

**Table des matières :**

Emetteur DCF 1

1 Pré-étude 7

1.1 Cahier des charges 7

1.2 Schéma bloc du système 7

1.3 Schéma bloc du hardware 7

1.4 Choix technologiques 7

1.4.1 Microcontrôleur 7

1.4.2 Oscillateur 8

1.4.3 Antenne DCF 8

1.4.4 Affichage LCD 9

1.4.5 Amplificateur 9

1.4.6 Régulateur 5V à 3.3V 9

1.4.7 Connecteur RJ45 9

1.4.8 Connecteur micro-USB 9

1.4.9 Switchs 10

1.4.9.1 Choix mécaniques 10

1.4.9.1.1 Affichage LCD 10

1.4.9.2 Fixation de l'antenne 10

1.4.9.2.1 Dimensions PCB 11

1.4.9.2.2 Boîtier 11

1.5 Consommation 12

1.5.1 PIC32MX795F512H 12

1.5.2 Affichage LCD - DEM 20488 SYH-PY 12

1.5.2.1 Rétro-éclairage 12

1.5.2.2 Ecran 12

1.6 Interaction du système avec l'extérieur 13

1.7 Evaluation des coûts 13

1.8 Evaluation des coûts 15

1.9 Conclusion et perspectives 17

2 Design 19

2.1 Description du produit voulu 19

2.2 Choix technologiques du système 21

2.2.1 Microcontrôleur 21

2.2.2 Affichage LCD 21

2.2.3 Antenne DCF 21

2.2.4 Analog switch 21

2.2.5 Amplificateur 21

2.2.6 Régulateur 5V à 3V3 21

2.2.7 Connecteur RJ45 21

2.2.8 Connecteur micro-USB 22

2.2.9 Switchs 22

2.3 Choix des composants 22

2.3.1 Taille des boîtiers 22

2.3.2 Coûts 22

2.4 Consommation 25

2.5 Dimensionnement du hardware 26

2.5.1 Alimentation uUSB 26

2.5.2 Touches menu 26

2.5.3 Régulateur 5V à 3V3 27

2.5.4 Affichage LCD & uC 28

2.5.5 Signaux DCF 30

2.5.6 RESET du uC & Connecteur de programmation 33

2.5.7 Contrôleur Ethernet 34

2.6 Concept du logiciel 35

2.6.1 Partie DCF 35

2.6.2 Mise à jour de l'heure 35

2.6.3 Menu 35

2.7 Conclusion 36

3 Réalisation 37

3.1 Modifications apportées au schéma électrique 37

3.1.1 Affichage 37

3.1.2 Touches menu et RESET 37

3.1.3 Signaux DCF 38

3.2 PCB 39

3.2.1 Routage 39

3.2.2 Placement des composants 41

3.2.3 Sérigraphie 43

3.2.4 Eléments particuliers du PCB 43

3.3 Mécanique 44

3.3.1 Choix du boîtier 44

3.3.2 Emplacement de l'antenne 44

3.4 Montage 45

3.5 Test du PCB - Alimentations 46

3.6 Réalisation du software 46

3.6.1 Détails du protocole 46

3.6.2 Programmation sous MPLAB 47

3.6.2.1 Paramétrage de MPLAB Harmony 47

3.6.2.1.1 Pin Settings 47

3.6.2.1.2 Timer 1 48

3.6.2.1.3 Timer 2 49

3.6.2.1.4 Timer 3 50

3.6.2.1.5 OC3 51

3.6.2.1.6 OC4 51

3.6.3 Affichage LCD 52

3.6.3.1 Problèmes rencontrés 53

3.6.4 Ethernet 54

3.6.4.1 Programme de test 54

3.6.5 Tests 58

3.6.6 Gestion des menus 60

3.6.7 Signaux DCF 63

3.6.7.1 Porteuse 63

3.6.7.2 Explications 63

3.6.7.3 Analog switch 64

3.6.7.4 Test de la partie analogique 65

3.6.7.4.1 Possibilité 1 65

3.6.7.4.2 Possibilité 2 66

3.6.7.5 Mesure partie analogique 66

3.6.7.5.1 Mesure 67

3.6.7.6 Dépannage 68

3.6.7.6.1 Test de l’amplificateur OPA2347UA 69

3.6.7.6.1.1 Mesure @ 10kHz 69

3.6.7.6.1.2 Mesure @ 77.5kHz 70

3.6.7.6.1.3 Explication des mesures 70

3.7 Codage de l'heure 71

3.7.1.1 Machine d’état 73

3.7.1.2 Conversion des minutes et de l’heure en BCD 75

3.7.1.3 Extraction des bits 75

3.7.1.4 Vérification de la parité 77

3.7.2 Envoi des données 79

3.8 Travail restant à faire 80

3.9 Conclusion 80

3.10 Annexes 81

# Pré-étude

## Cahier des charges

Voir annexe I : 1819\_emetteurDcf-CDC-v1.doc

## Schéma bloc du système

Voir annexe I : 1819\_emetteurDcf-CDC-v1.doc

## Schéma bloc du hardware

Boîtier

PCB

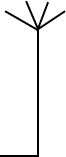
Affichage LCD

Câble

RJ-45

Câble

uUSB



Antenne

## Choix technologiques

### Microcontrôleur

Comme l'ES travaille avec la famille de microcontrôleur PIC32MX, j'ai décidé de reprendre le même modèle utilisé pour la carte de développement de l’ES, mais la taille en dessous, soit un 64pins (PIC32MX795F512H) au lieu d'un 100pins, car cela représente un nombre de patte beaucoup plus important que ce dont j'ai besoin.

Ce uC a également été choisi car il est muni d’une interface Ethernet.

Et de plus, l'environnement de programmation utilisé (MPLAB X IDE) m'est familier, et donc plus facilement utilisable et compréhensible.

Le PIC32MX795F512H a donc la signification suivante :

32 Le uC travaille sur 32bits

MX Type d’architecture utilisée pour le uC

795 Numéro d’identification

F Indique la famille de mémoire, ici Flash

512 Taille en [KB] de la mémoire programmable

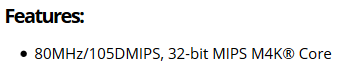
L Correspond au nombre de patte, ici 64 pins

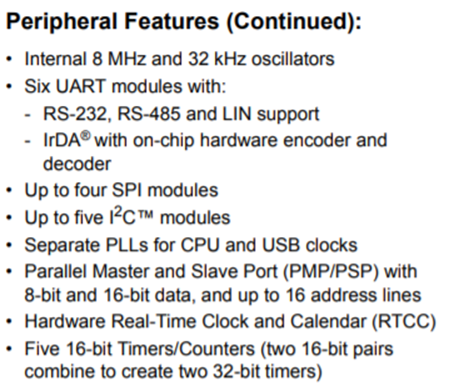
### Oscillateur

L'oscillateur d'une fréquence de 77.5kHz, sera généré par un timer du microcontrôleur.

Les Timers du PIC32MX795F512H, ont une résolution de 216, soit 65'536.

Le clock du microcontrôleur étant de 80MHz, celui-ci sera donc largement suffisant pour générer la fréquence de la porteuse





fporteuse = 77.5kHz fclock = 80MHz

= = 12.90us

= = 12.5ns

= = 1032

Après calcul, on voit que le nombre de tick devra être de 1032, afin de pouvoir générer une fréquence de 77.5kHz, à partir d'un Timer.

Comme l'échelle de travail de ceux-ci est de 0 à 65'535, la valeur est donc largement dans cette gamme.

### Antenne DCF

L'antenne pour émettre le signal, sera reprise directement d'une platine de réception DCF 2.5-15V/DC - 3mA, car il est difficile de trouver une antenne seule, sans module.

De plus, l'antenne choisie possède déjà un condensateur de filtrage.

### Affichage LCD

Pour l'affichage LCD, j'ai choisi la famille DEM 20XXX, car c'est celui utilisé sur le PIC32 de l'ETML-ES, son environnement de travail m'est donc connu, au niveau de la programmation et de son utilisation.

Le modèle DEM 20488 SYH-PY 4x20 caractères, me semble adapté pour mon application, de par sa taille (77x47), qui permettra à l'utilisera d'avoir un affichage ni trop grand, ni trop petit, et qui sera donc confortable à regarder.

De plus, la tension doit être comprise entre 4.5V et 5.5V, il pourra ainsi être alimenté par l'alimentation principale du circuit (micro-USB +5V).

### Amplificateur

Pour définir quel type d’amplificateur opérationnel sera utilisé dans l’étage de sortie, nous devons tenir compte des caractéristiques suivantes :

* Transmission de données numériques : besoin de gérer des flancs raides => Faire attention au Slew Rate.
* Circuit alimenté en 5V : choisir un AOP compatible avec une alimentation 0-5V asymétrique.

### Régulateur 5V à 3.3V

Afin d’alimenter le microcontrôleur en +3.3V, j’ai décidé d’utiliser un régulateur à découpage, car ceux-ci ont un meilleur rendement que les régulateurs linéaires.

J’ai ainsi choisi le modèle TPS54228DR, qui a une valeur d’entrée entre 4.5 et 18V, donc qui pourra être alimenté par l’alimentation principale, qui est de +5V.

La tension de sortie est ajustable entre 0.76V et 7V, et le courant est de 2A.

Il a été choisi en fonction du courant de consommation du circuit (voir point 5). Nous avons un courant de consommation d’approximativement 150mA. Mais prenons une marge à 2A car à ce stade nous ne connaissons pas encore la puissance d’émission du signal.

### Connecteur RJ45

Pour le connecteur RJ45, pour le câble Ethernet, j'ai repris celui déjà présent sur le Kit PIC32MX, comme il correspond à la connexion standard utilisée par l'école. C'est donc le modèle SI-60062-F.

### Connecteur micro-USB

Le connecteur micro-USB choisi est de type B, car actuellement c'est le type qui est le plus couramment utilisé (chargeur smartphone, etc...).

### Switchs

Pour les switchs, j'ai décidé de prendre ceux de chez Würth Elektronik, car ils sont simples d'utilisation et de plus, la gamme possède de nombreux switchs avec des hauteurs différentes.

Cela me sera très pratique du fait que, comme le PCB sera placé dans un boîtier, cela signifie que les touches devront être accessible par l'utilisateur, et donc présenter une certaine hauteur pour permettre cela.

#### Choix mécaniques

##### Affichage LCD

Au début, je pensais placer l'affichage LCD sur le circuit avec une barrette à broche pour réaliser les connexions, ainsi que des colonnettes pour fixer celui-ci.

Cela engendre que les switchs, qui permettront de régler les menus du LCD, doivent avoir une certaine hauteur, c'est-à-dire être plus élevé que l'affichage, afin d'être accessible par l'utilisateur, lorsque la carte sera dans le boîtier.

Or, il s'est avéré difficile de trouver des touches suffisamment hautes. J'ai donc pensé à la solution des touches encastrées, soit à 90°. Ce qui signifie que celles-ci seraient placées sur l'un des côtés du boîtier, au lieu de la face avant.

Après réflexion, cette solution ne m'a pas parue optimale, de par le fait que leurs emplacements ne seraient pas agréables à l'utilisateur, et que de plus, celles-ci ont un coût relativement élevé.

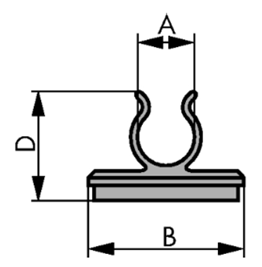
Pour remédier à ces problèmes, j'ai décidé que mon affichage LCD serait placé, non pas sur la face TOP de mon circuit, mais sur la face BOTTOM:

Cela veut dire qu'il y aurait une ouverture faite pour que seulement l'affichage puisse dépasser et la barrette à broche, ainsi que les colonnettes seraient également sur l'autre face.

#### Fixation de l'antenne

Comme l'antenne représente une surface importante, celle-ci ne sera pas directement placée dans le boîtier.

J'ai donc pensé à la placer dans un support de fixation, qui serait vissé sur l'un des côtés du boîtier, aux deux extrémités de l'antenne.



##### Dimensions PCB

Comme le composant le plus encombrant est l'affichage (77x47mm), et que le côté TOP n'aura certainement que l'affichage et les touches, j'ai donc estimé approximativement la grandeur du PCB.

J'ai donc, dans la longueur, et dans la largeur, laissé une marge d'environ 20mm, ce qui donnerait les dimensions suivantes : 100mm x 70mm

**AFFICHAGE LCD**

**100**

**70**

**77**

**47**

**Dimensions : mm**

**6.2**

**6.2**

##### Boîtier

Concernant le boîtier, plusieurs idées sont possibles, j'ai donc pensé aux propositions suivantes :

* Boîtier plastique avec face avant transparente
* Boîtier plastique couleur uniforme
* Boîtier plastique « pupitre »

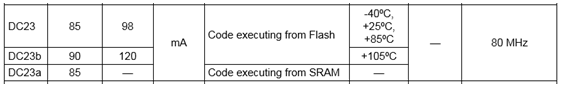
Plusieurs critères seront donc pris en compte lors du choix du boîtier :

* Agréable d’utilisation pour l’utilisateur (boîtier pas trop grand)
* Hauteur (PCB avec connecteur RJ-45 et micro-USB + Affichage et switchs)

## Consommation

|  |  |
| --- | --- |
| **COMPOSANT** | **CONSOMMATION** |
| PIC32MX795F512H | 120mA (pire cas) |
| Affichage LCD - DEM 20488 SYH-PY | Rétro-éclairage 30mA - Ecran 0.5mA  => 30.5mA |

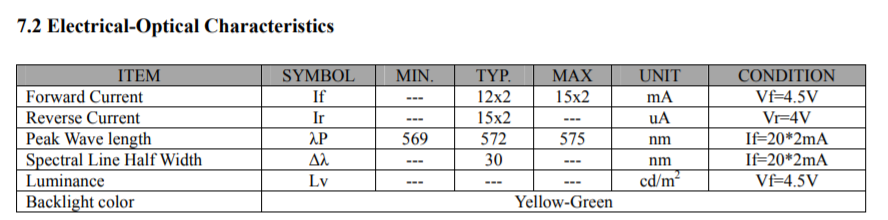
### PIC32MX795F512H



Fréquence = 80MHz Pire cas +105° soit 120mA

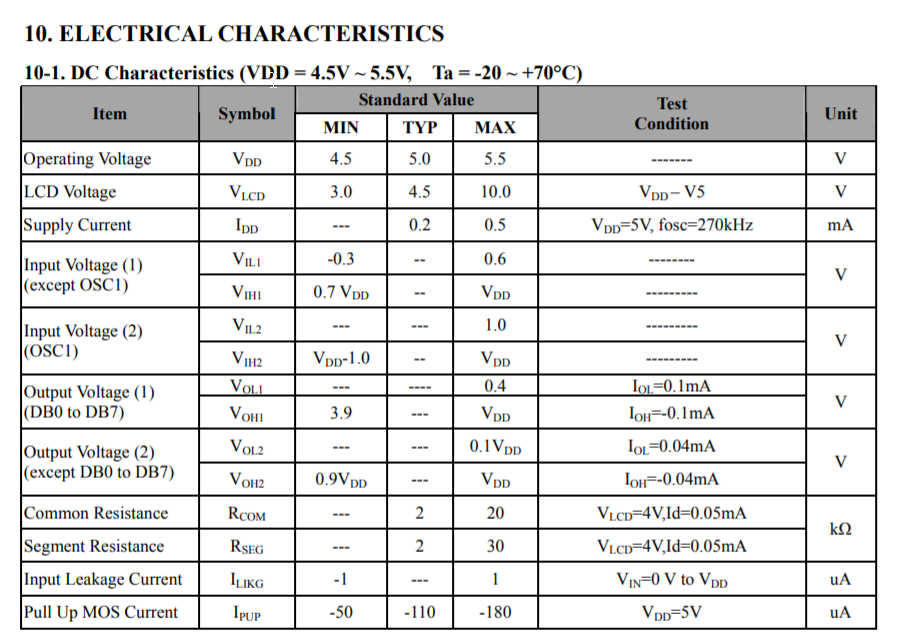
### Affichage LCD - DEM 20488 SYH-PY

#### Rétro-éclairage



Forward Current : If = 2 \* 15mA = 30mA

#### Ecran



Supply Current : IDD (MAX) = 0.5mA

Consommation totale : If + IDD (MAX) = 30\*10-3 + 0.5\*10-3 = 30.5mA

## Interaction du système avec l'extérieur

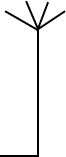
**Boîtier avec PCB**

Câble

RJ-45

Câble

uUSB



Antenne

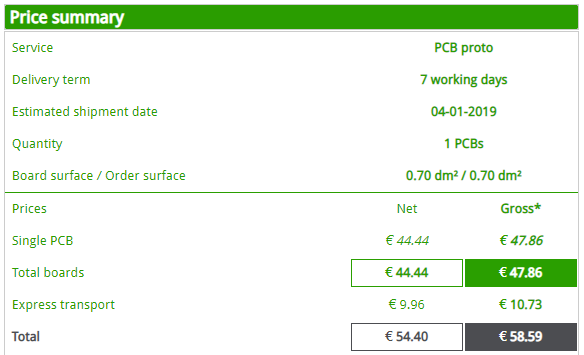


Le système va interagir avec les éléments extérieurs suivants :

* Le connecteur Ethernet recevra l'heure via un câble RJ45
* Le circuit est alimenté par un câble micro-USB
* L'antenne à l'extérieur du boîtier va permettre de transmettre le signal qui aura les données de l'heure, pour régler automatiquement les horloges de l'école

## Evaluation des coûts

D'après la taille que j'ai estimée pour le PCB, j'ai réalisé, sous Euro-circuit, une "simulation" afin d'estimer le prix du circuit.



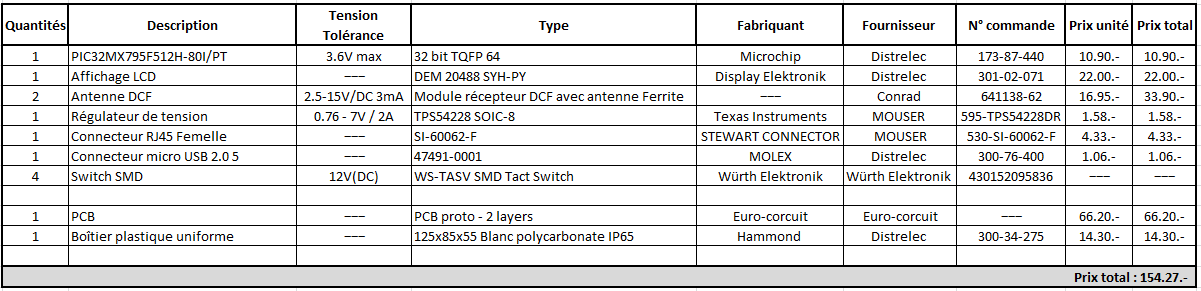
1euro = 1.13 Franc suisse

Le PCB coûterait donc : 58.59 \* 1.13 = 66.20.-

## Evaluation des coûts

Pour l'évaluation des coûts, j'ai, dans le tableau ci-dessous, mis tous les éléments principaux du système, comme le microcontrôleur, l'affichage, etc...

Le choix de certains composants n'est pas encore définitif, mais cela permet d'avoir un ordre d'idée des prix, comme pour le PCB et le boîtier, qui seront défini dans la phase suivante de design.

****

## Conclusion et perspectives

Cette première phase du projet m’a permis de me familiariser avec, et d’étudier de nouvelles notions, comme le DCF.

Celle-ci s'est bien passée, et est à mon avis importante, non seulement pour étudier les divers aspects du système, mais aussi les diverses technologies qui composeront celui-ci.

Pour la suite du projet, il va falloir choisir les composants définitifs pour les diverses parties du montage. Ce choix se fera en fonction de certaines contraintes (alimentation, consommation, etc...), mais aussi en fonction des autres projets.

C'est-à-dire que comme il y a plusieurs projets en cours, et que nous sommes plusieurs à avoir besoin de certains composants similaires, comme un affichage LCD, par exemple, le choix sera donc regroupé. Ceci pour non seulement simplifier les commandes, mais également pour se mettre dans une situation de travail en entreprise.

Ensuite, une fois les composants choisis, il faudra réaliser le dimensionnement de toutes les parties qui vont composer le système, pour la réalisation du schéma électrique.

Concernant la partie mécanique, il faudra prêter attention au choix du boîtier, car il faut que les connecteurs micro-USB et RJ45 soient accessible, et également les touches.

Date : 13 décembre 2018

Signature : Julie Culand

# Design

## Description du produit voulu

**Microcontrôleur**

uC

Oscillateur

77.5kHz

Affichage LCD

Switchs

Modulation

uUSB

uC

Ethernet

uC

Amplificateur

Potentiomètre

Antenne

uC



Le système a pour but de créer un signal de remplacement, de l'émetteur DCF77, qui va permettre de régler les horloges de l'école ES.

Le signal qui est recréé, doit comporter une porteuse d'une fréquence de 77.5kHz, qui sera générer par l'un des Timer du microcontrôleur. De même pour la modulation, celle-ci sera réalisée par le microcontrôleur.

Le signal modulé sera ensuite, amplifier et transmis via une antenne à l'extérieur du circuit.

La puissance d'émission pourra être réglée à l'aide d'un potentiomètre.

## Choix technologiques du système

### Microcontrôleur

Comme il m'a été imposé de travailler avec la famille PIC32, j'ai choisi de prendre le modèle PIC32MX795F512H (64 pins), car il possède les connexions nécessaires pour l'Ethernet. Et de plus, un précédent projet a été réalisé avec ce microcontrôleur et une partie Ethernet, que je vais donc reprendre de ce projet, comme c'est une partie du système qui n'est pas étudiée à l'ES, et qui ne m'est ainsi pas connue.

### Affichage LCD

Concernant l'affichage LCD, le choix c'est porter sur une décision de groupe, comme nous sommes plusieurs à en avoir besoin. L'écran choisi a une tension d'entrée entre 3V et 3.6V, il pourra donc être alimenté par le régulateur 3V3. Il est doté d'une interface parallèle et de 4x20 caractères. Le modèle est le LCD MOD CHAR 4X20 GRY TRANSF STN.

### Antenne DCF

L'antenne pour émettre le signal sera reprise directement d'une platine de réception DCF 2.5-15VDC - 3mA, car il est difficile de trouver une antenne seule, sans module.

De plus, l'antenne choisie possède déjà un condensateur de filtrage.

### Analog switch

L'analog switch choisi est le modèle TS12A4515. Il est alimenté entre 2.7V et 12V, donc il pourra être alimenté par le uUSB (+5V).

Celui-ci est d'une extrême simplicité, puisque il ne possède qu'un switch, et de plus, il est peu coûteux.

### Amplificateur

L'amplificateur opérationnel choisi, possède une tension d'alimentation asymétrique entre 2.3V à 5.5V, ce qui permet de l'alimenter avec celle principale (+5V).

De plus, il est rail-to-rail, cela implique qu’il peut atteindre les niveaux de tensions d’alimentation en sortie (pas tension de déchet).

I’IC possède également deux amplis, c’est le modèle OPA2347UA

### Régulateur 5V à 3V3

Pour avoir une tension de 3V3, pour alimenter le microcontrôleur, j'ai repris le régulateur linéaire utilisé sur le Kit PIC32 (Projet 1102x\_SK32MX775F512L), qui est un MAX1793, car il est en stock à l'ES.

Celui-ci a une tension d'entrée entre 2.5V et 5.5V, ce qui va permettre de l'alimenter avec l'alimentation principale du circuit (micro-USB +5V). De plus, la tension de sortie est ajustable entre 1.25V à 5.5V/3.3V. Le courant en sortie est de 1A, ce qui suffira largement pour alimenter le circuit. (Voir Annexe IV : Consommation)

### Connecteur RJ45

Comme l'ES utilise un connecteur RJ45 sur la carte de développement Kit PIC32 (Projet 1102x\_SK32MX775F512L), j'ai décidé de reprendre le même, car celui-ci correspond à la connexion standard utilisée par l'école. C'est donc le modèle SI-60062-F.

### Connecteur micro-USB

Le connecteur micro-USB utilisé pour alimenter le circuit, est de type B, car actuellement, c'est le modèle le plus utilisé (chargeur de smartphone, etc...), il est donc facile de se procurer un câble d'alimentation, et à moindre coût.

Le modèle choisi est le 47346-0001 de chez MOLEX.

### Switchs

Comme les touches doivent être accessibles par l'utilisateur à l'extérieur du boîtier, j'ai choisi de prendre les switchs de chez Würth Elektronik, car ceux-ci présentent un choix au niveau des hauteurs. J'ai donc pris le modèle 430152095836, qui a une hauteur de 9.5mm.

## Choix des composants

### Taille des boîtiers

Pour les composants passifs (résistances et condensateurs), j'ai décidé de prendre une taille de boîtiers de 0805. Ce choix c'est en premier lieu porter sur le fait que l'ETML-ES possède la majeure partie de son stock de composants en cette taille, ce qui va permettre de limiter les frais. Et deuxièmement, ce sont des composants que l'on peut aisément braser à la main.

### Coûts

En ce qui concerne directement les composants mêmes, j'ai privilégié le stock de l'ETML-ES, afin de me mettre dans des conditions réelles de travail, comme dans une entreprise, où il est important d'utiliser d'abord le stock, puis de commander ce qui n'est pas disponible.

Pour l’estimation des coûts, j’ai regroupé les principaux composants du montage et j’ai réalisé une estimation pour le PCB, le boîtier, ainsi que pour les petits composants (résistances, condensateurs, etc..).

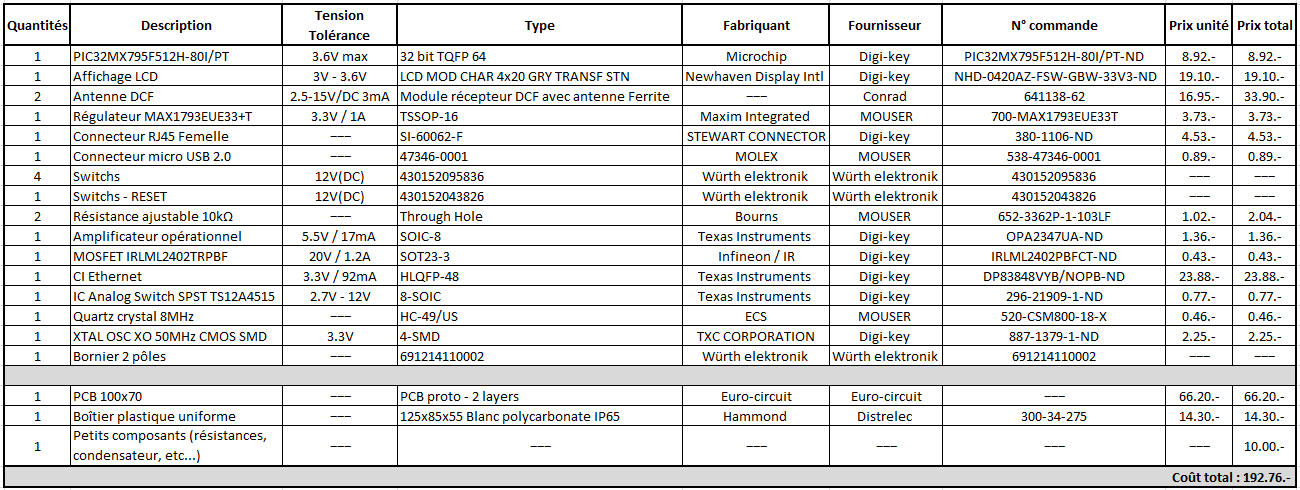
Concernant les choix des fournisseurs, je me suis axée sur Digi-key, qui est souvent moins cher que Distrelec et MOUSER. Autrement, pour les composants non disponible chez Digi-key, je me suis retournée auprès des autres fournisseurs.

**Estimation des coûts**

Pour l'évaluation des coûts, j'ai dans le tableau ci-dessous, mis tous les éléments principaux du système, comme le microcontrôleur, l'affichage, l'antenne, etc...

Concernant le prix des petits composants (résistances, condensateurs, ...), j'ai réalisé une estimation d'environ 10.-.

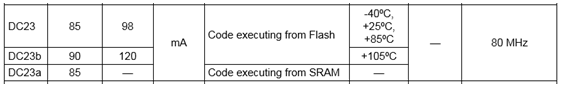
Pour le PCB et le boîtier, j'ai réalisé une approximation comme leurs caractéristiques finales ne sont pas encore totalement connues (tailles, matières, etc..).



## Consommation

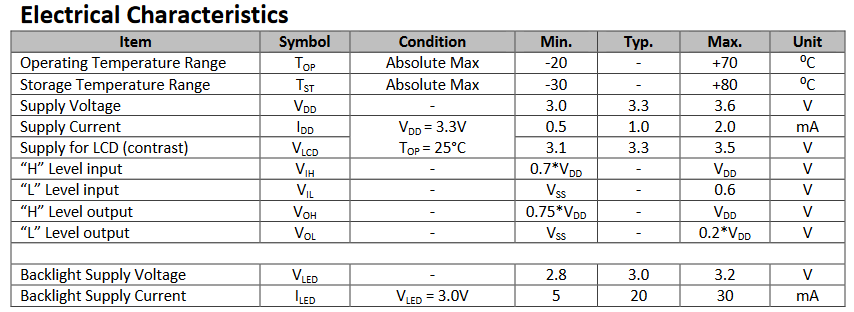
|  |  |
| --- | --- |
| **COMPOSANT** | **CONSOMMATION** |
| PIC32MX795F512H | 120mA (pire cas) |
| Affichage LCD - NHD-0420AZ-FSW-GBW | Rétro-éclairage 30mA  Ecran 2mA  => 32mA |
| Contrôleur Ethernet - DP83848VYB | ~100mA |
| **Consommation totale :** | ~ 252mA |

**PIC32MX795F512H**



Consommation => Fréquence = 80MHz Pire cas +105° soit 120mA

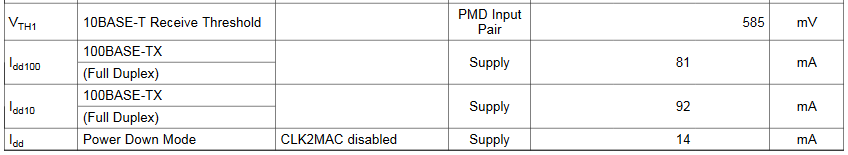
**Affichage LCD - DEM 20488 SYH-PY**

****

Consommation => Rétro-éclairage + Ecran : ~ 32mA

**Contrôleur Ethernet**





Consommation => ~ 100mA

## Dimensionnement du hardware

### Alimentation uUSB

Comme le connecteur micro-USB est utilisé

seulement pour alimenter le circuit, seules

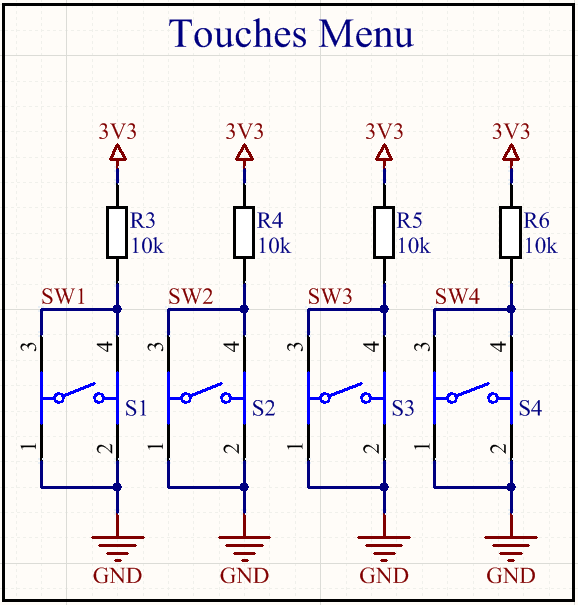
les pattes VBUS (+5V) et GND (0V) sont utiles.

### Touches menu

Lorsque l’on regarde dans le datasheet du microcontrôleur (PIC32MX795F512H), on peut voir que l’entrée d’une pin consomme maximum 25mA, et la tension est de 3.3V.



On peut donc en déduire la valeur de la résistance :



R = = = 132Ω

Or, une consommation de 25mA sur une pin est

relativement élevé, c’est pourquoi j’ai placé des

résistances de 10kΩ, qui est un choix arbitraire et

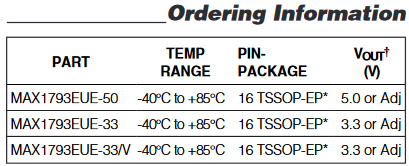
un standard. Cela permet notamment, de réduire

la consommation.

I = = = 330uA

### Régulateur 5V à 3V3

Lorsque l'on regarde dans le datasheet du MAX1793, on peut voir que le type choisi (MAX1793EUE33+), à une sortie qui est déjà par défaut fixée à 3.3V, il n'y a ainsi pas besoin de l'ajuster.



Les connexions sont donc les suivantes :

VIN1 - VIN2 - VIN3 - VIN4 Correspondent à la tension d'alimentation +5V

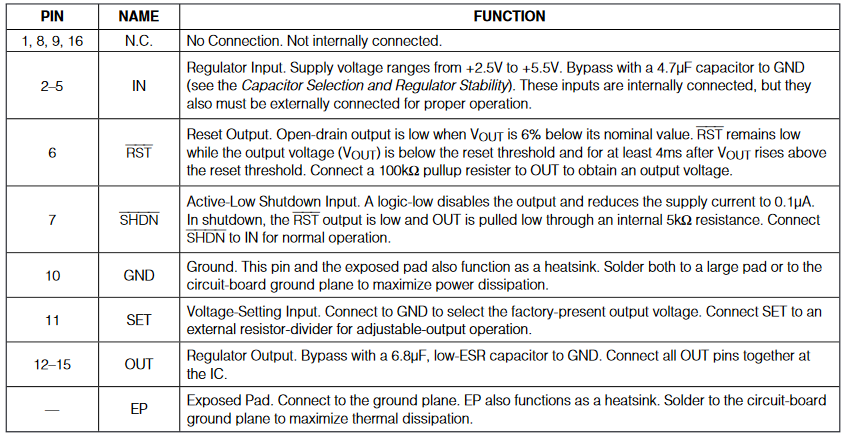
VOUT1 - VOUT2 - VOUT3 - VOUT4 Correspond à la tension fixée en sortie +3.3V

/SHDN Connecté à la tension d'alimentation pour un fonctionnement normal

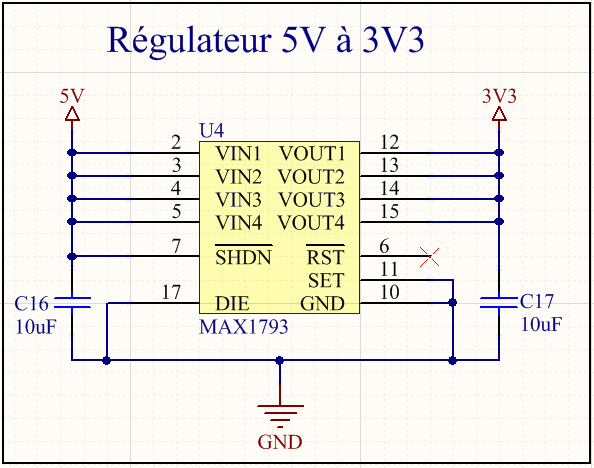
DIE Patte située sous le circuit, pour la dissipation thermique, donc relié à la masse

SET Connecté au GND pour avoir la tension de sortie fixe

/RST Permet de réinitialiser la sortie, mais pas utilisé



Le MAX1793 est donc connecté de la manière suivante :



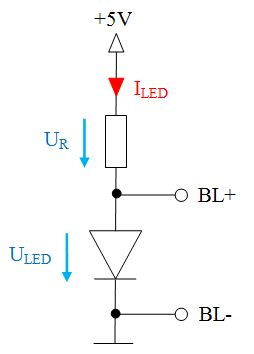
Le régulateur MAX1793 est de type linéaire. Il a l’avantage d’engendrer moins de bruit sur l’alimentation, par rapport à une configuration à découpage.

De plus, il présente une moins grande complexité, donc un coût peu élevé.

### Affichage LCD & uC

Lorsque l'on regarde dans le datasheet de l'affichage LCD, concernant le backlight, on peut que la Led doit avoir une chute de tension entre 2.8V et 3.2V et que le courant doit être compris entre 5mA et 30mA.



J'ai décidé de prendre les valeurs typiques, afin de ne pas être, d'un côté à la limite du fonctionnement, et dans l'autre dans un état critique.

Vcc = 5V ILED = 20mA ULED = 3.0V

UR = Vcc - ULED  = 5 - 3 = 2V

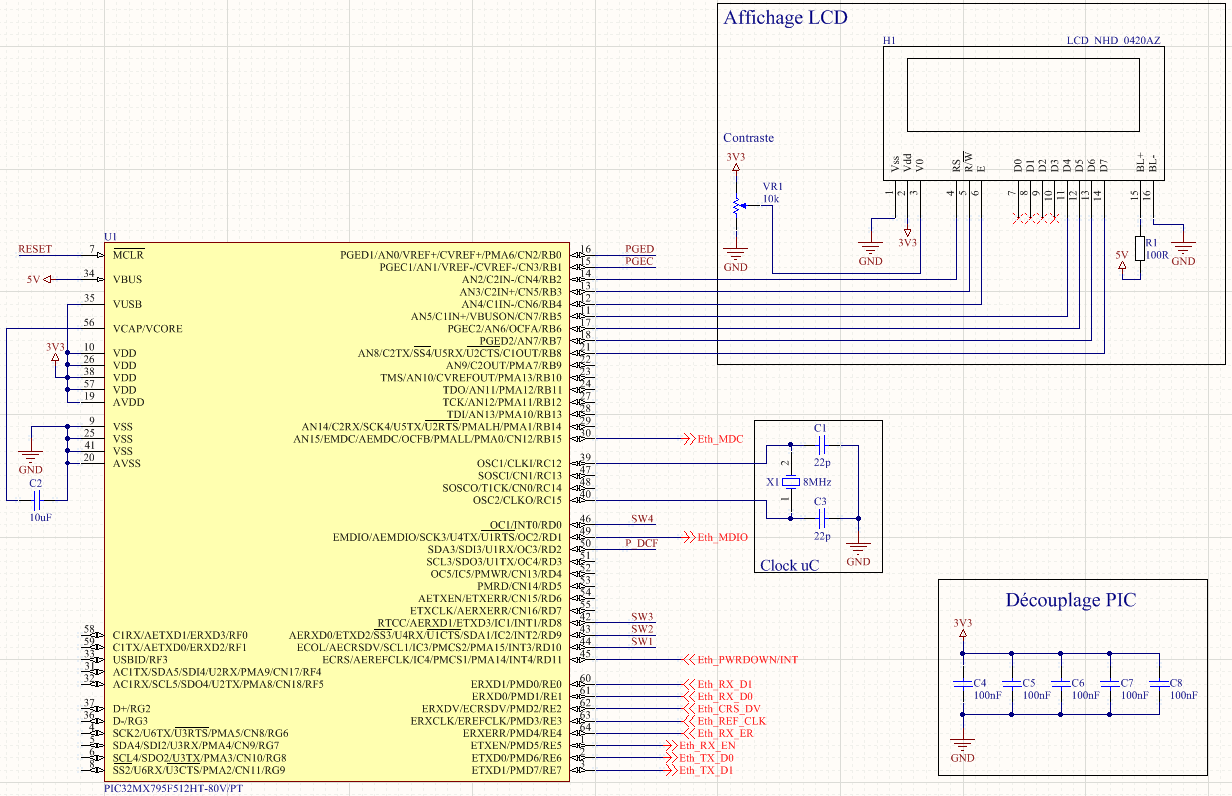
R = = = 100Ω

PR = R \* ILED2 = 100 \* (20\*10-3)2 = 40mW

Ce qui donne la configuration suivante :

L'affichage LCD travaille avec une interface parallèle, il est donc connecté sur le port B du microcontrôleur.

Le schéma de l'affichage, avec le microcontrôleur, est le suivant :

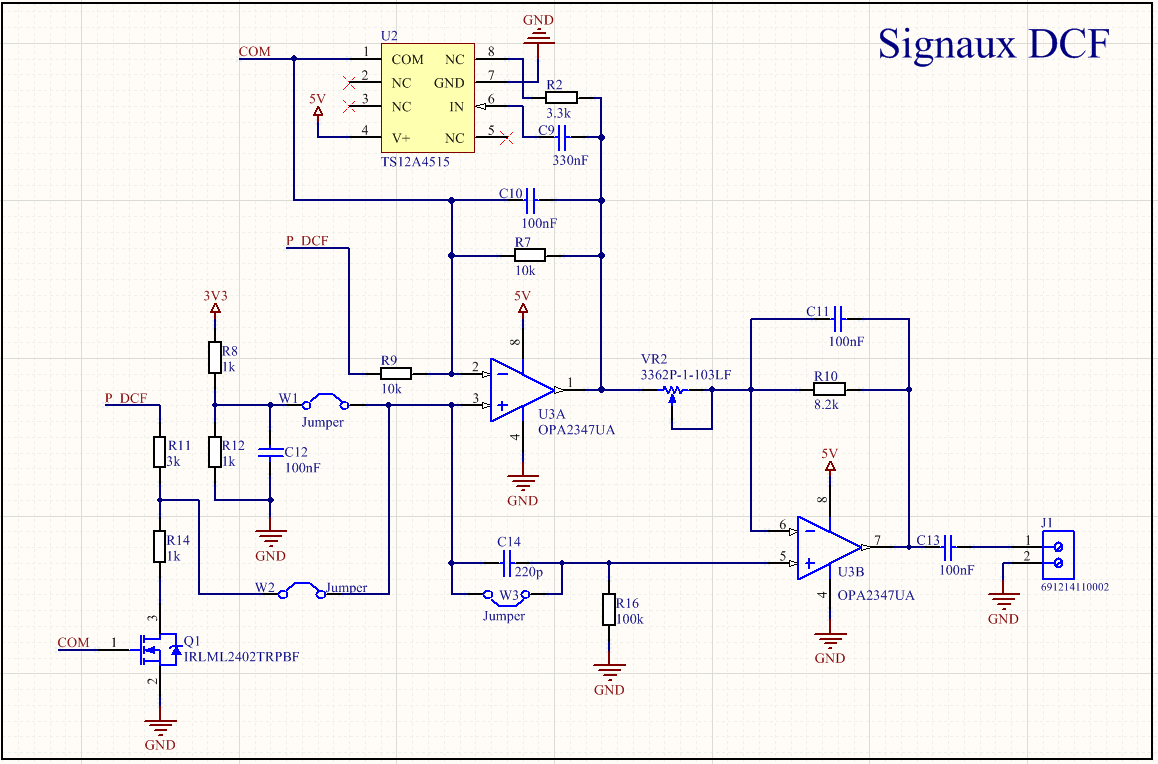


### Signaux DCF

Cette partie du schéma va recréer le signal DCF. C'est-à-dire que la porteuse est modulée par des impulsions, au rythme d'une par seconde. Sa fréquence sera de 77.5kHz.

Ces impulsions se traduisent chaque seconde par une diminution de 25% de l'amplitude du signal reçu.

Voici le schéma complet du système :



Sur le schéma, j'ai décidé de mettre deux possibilités quant à la création du signal DCF.

**1er possibilité**

Le pont diviseur permet d'obtenir la tension d'alimentation (3.3V), divisée par deux, comme les résistances possèdent la même valeur.

Le condensateur en sortie permet de lisser le signal.

U = = = 1.65V

U = \* VDD = \* 3.3 = 1.65V

Cette partie permet, lorsque le signal sera abaissé à 25%, d'avoir un signal centré (Figure 1), car dans le cas contraire, celui-ci restera à 0V (Figure 2).

0V

3.3V

25%

3.3V

25%

0V

**Figure 1.**

**Figure 2.**

Le premier amplificateur (UA3) permet d’abaisser l’amplitude de 25% à l’aide d’un analog switch.

Lorsque l’analog switch sera ouvert, seul le premier filtre sera actif (C10 et R7), ainsi le signal sera donc à 100%. La constante de temps est donc la suivante :

C10 = 100nF R7 = 10kΩ Tau1 = R7 \* C10 = 10\*103 \* 100\*10-9 = 1[ms]

Lorsque l’analog switch sera fermé, cette fois-ci, les deux filtres sont actifs, ce qui permet d’abaisser la tension du signal à 25%, soit ¼ de celle-ci.

Que l’analog switch soit ouvert ou fermé, la valeur de Tau doit rester la même, c’est pourquoi, il faut multiplier la valeur du condensateur par trois, et diviser celle de la résistance par la même valeur.

C9 = 3 \* C10 = 3 \* 100\*10-9 = 300nF Série E24 : 330nF

R2 = R7 / 3 = 10\*103 / 3 = 3.3kΩ Série E24 : 3.3kΩ

Tau2 = \* 4C = R \* C=> Ce qui permet d'éliminer le facteur 4 et donc d'avoir un Tau de même valeur que précédemment

Le second amplificateur (U3B) permet d’amplifier le signal, et aussi de pouvoir faire varier la puissance d’émission en sortie, grâce à un potentiomètre (VR2).

Le condensateur en sortie de l'amplificateur permet d’éliminer la composante continue.

En sortie, l'antenne possède déjà un condensateur de filtrage, ce qui forme un circuit bouchon. On peut donc estimer que l'impédance du circuit est d'environ 200-300Ω.

Si le condensateur en sortie a une valeur de 100nF, la réactance capacitive Xc vaudra :

Xc = = = 20.53Ω

On voit donc que l’impédance du condensateur est très faible comparé à celle du circuit bouchon, la valeur du condensateur est donc correcte et acceptable, et la réactance de celui-ci est très minime.

Concernant les gains des amplificateurs, et comme il y a deux étages, ceux-ci vont donc être multiplié, et doivent ainsi être inférieurs à 1, afin de limiter la puissance d'émission, qui est de l'ordre du [mW].

Comme la configuration est une rétroaction négative, l’amplificateur travaille en mode opérationnel.

Ampli 1 : Av1 = - = - = -1

Ampli 2 : Av2 = - = = - 0.82

Gaintot = Av1 \* Av2 = -1 \* -0.82 = 0.82

**2ème possibilité**

La seconde partie consiste à placer un pont diviseur (R11 et R14) de façon à avoir l'amplitude abaissée à 25%.

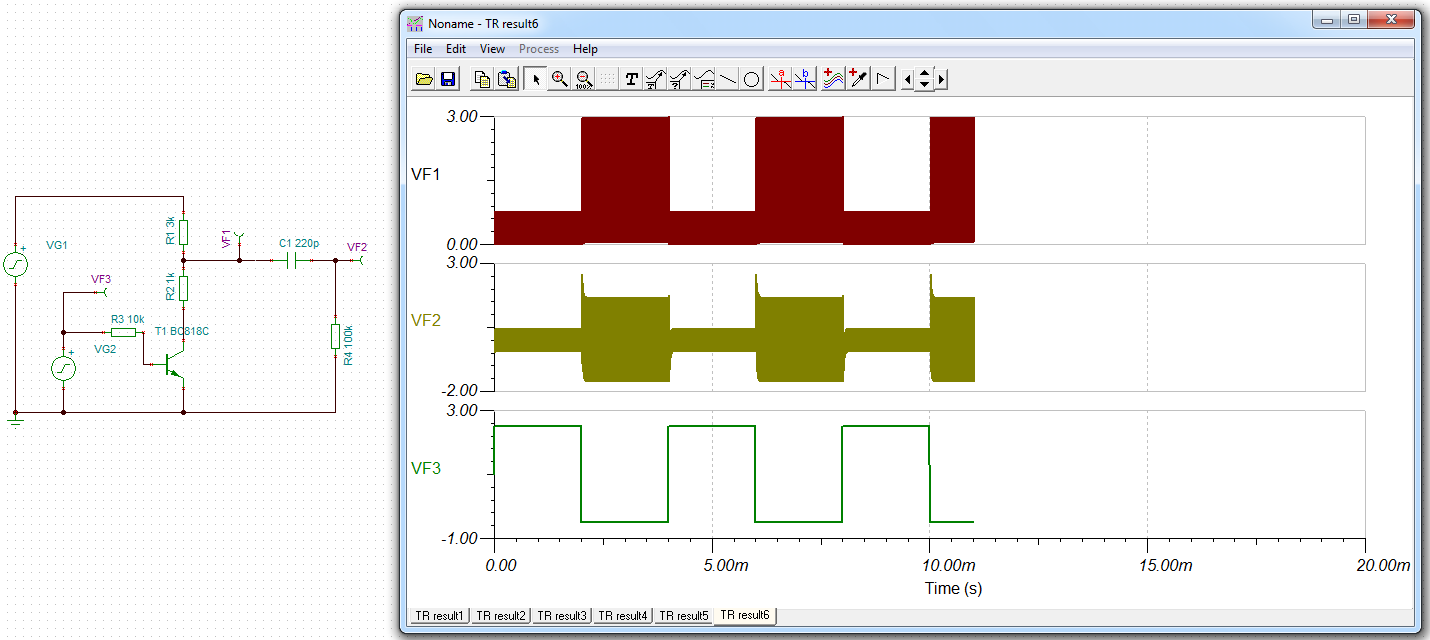
Rtot = 4k R14 => ¼ de la résistance totale

¼ de 3.3V = 825mV

On place un transistor MOSFET (Q1), en bas du pont diviseur, et à l'entrée, le signal de la porteuse à 77.5kHz.

Ensuite, pour centrer le signal, on place un filtre passe haut (C14 et R14).

Puis le signal sera amplifié, comme pour la première possibilité, via l'amplificateur (U3B).

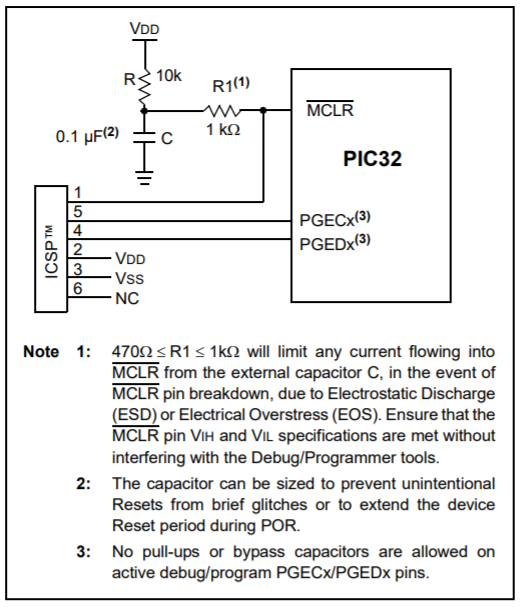


La simulation, ci-dessus, montre la situation avec la 2ème possibilité, où l'on voit bien que sur VF1, le signal est abaissé à 25%, mais il n'est pas centré.

L'ajout du filtre passe haut permet ainsi, de conserver l'amplitude abaissée, tout en centrant le signal.

### RESET du uC & Connecteur de programmation

Lorsque l’on regarde dans le datasheet du PIC32MX795F512H, sous la catégorie Master clear (/MCLR) Pin, on trouve le schéma avec la configuration suivante :



La broche /MCLR fournit la fonction

qui permet de réinitialiser l’appareil.

Or, il est important de respecter certaines

contraintes lors du dimensionnement du

RESET, qui est donné dans la même rubrique.

La résistance R1 doit être comprise entre

470Ω et 1kΩ, ce qui permet de limiter le

courant entrant dans la pin /MCLR du

condensateur externe C, en cas de rupture de

la broche, due à une décharge électrostatique

ou d’une surcharge électrique.

On voit que la résistance R, a une valeur dix

fois supérieure à celle de R1. Et que la valeur

du condensateur est de 0.1uF.

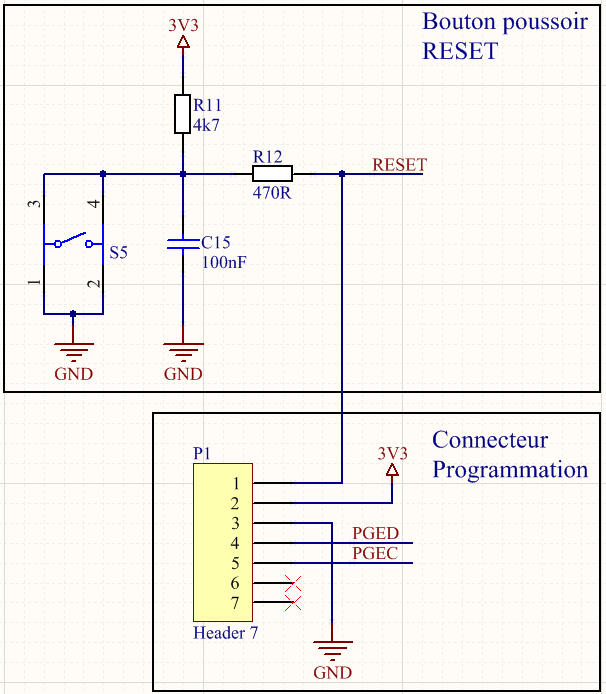
J’ai donc décidé de reprendre la configuration

du Kit PIC32, où R11 vaut 4.7k, R12 470R et

C15 100nF. Ce qui répond aux critères donnés.

Concernant le connecteur de programmation, on voit qu’il faut simplement connecter les pattes de programmation sur celles PGECx et PGEDx du microcontrôleur.

Cela donne la connectivité suivante :

****

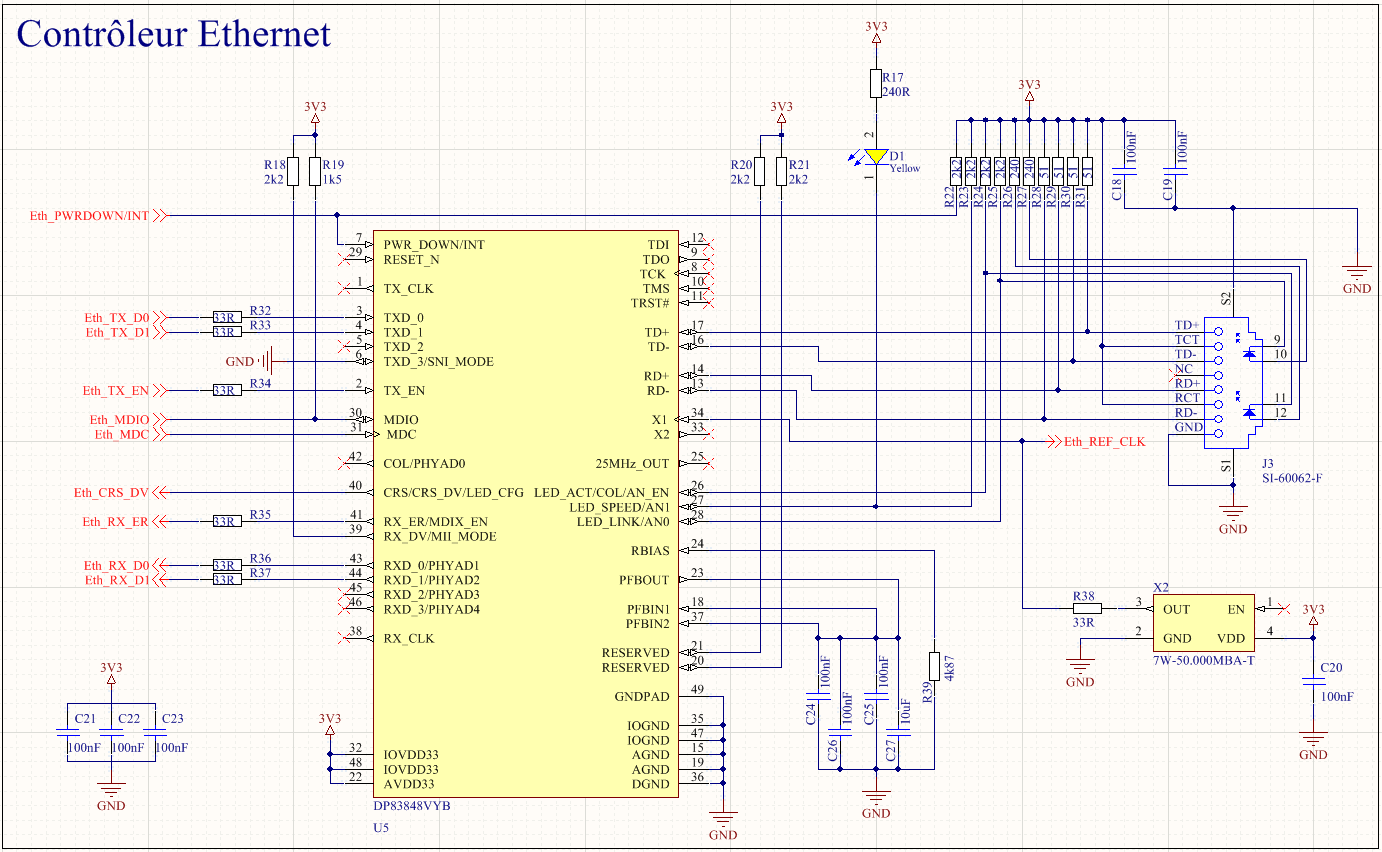
### Contrôleur Ethernet

La partie Ethernet a été reprise d’un précédent projet (1630x\_TimbreuseBadgeRFID). Aucune modification n'a été apportée au schéma, comme celui-ci est fonctionnel.

Néanmoins, l’élément principal est le composant DP83848VYB, qui permet de faire le pont entre le microcontrôleur et le RJ45.

Ce dernier communique via une interface RMII.

Le contrôleur Ethernet est donc le suivant :

****

## Concept du logiciel

### Partie DCF

La partie principale consiste à la reconstitution du signal DCF.

Un Timer du microcontrôleur va permettre de recréer le signal de la porteuse, d’une fréquence de 77.5kHz, et d’une amplitude de 0V à 3.3V.

L'onde porteuse est modulée par des impulsions, au rythme d'une par seconde, les informations sont donc transmises sous forme binaire, à raison d'un bit par seconde, contenant ainsi les données de l'heure.

Ces impulsions se traduisent chaque seconde par une diminution de 25% de l'amplitude du signal, qui est réalisé à l'aide d'un analog switch et à une disposition de condensateurs et de résistances, de par le choix des valeurs.

L’impulsion qui est émise au début de chaque seconde dure 100ms pour un ‘0’ logique et 200ms pour un ‘1’ logique. Seule la 59ème seconde n’est pas modulée et permet d’annoncer le début d’une nouvelle trame.

### Mise à jour de l'heure

L'heure sera mise à jour via le protocole SNTP (Simple Network Time Protocol).

Ce protocole permet de synchroniser, via un réseau informatique, l'horloge locale d'ordinateurs sur une référence d'heure.

### Menu

Les quatre switch vont permettre de régler un menu, qui va gérer les ajustements suivants :

* Réglage du fuseau horaire
* Réglage de l'heure été/hiver
* Possibilité de désactiver le changement d'heure
* Emission du signal (5min par jour à midi)

## Conclusion

La partie Design m'a permis d'établir le schéma électrique complet, et ainsi de fixer les technologies qui seront utilisées, ainsi que les différents types de communications.

Lors du choix des technologies, de nombreux composants ont été choisi de par leurs caractéristiques électriques, mais certains aussi par choix de groupe, afin de limiter les coûts de certains éléments, comme l'affichage.

Lors de cette étape, je n'ai pas eu de problèmes majeurs, mise à part la réalisation de la partie DCF, qui posait problème, de par le centrage des signaux, lorsque l'on a l'abaissement de l'amplitude. Or ce problème a pu être résolu avec plusieurs solutions, et c'est notamment pour cela, que j'ai décidé de mettre en place deux systèmes différents sur ma partie DCF.

La deuxième solution nécessite l'ajout de peu de composants, donc un coût très bas.

La suite est donc positive, puisque le PCB va pouvoir être réalisé, ainsi que la fin du projet.

Lausanne, le 6 février 2019

Signature : Julie Culand

# Réalisation

## Modifications apportées au schéma électrique

Suite à la partie précédente réalisée, donc du Design, les éléments suivants ont été relevés pour être modifié. **(Voir Annexe II : Schéma électrique)**

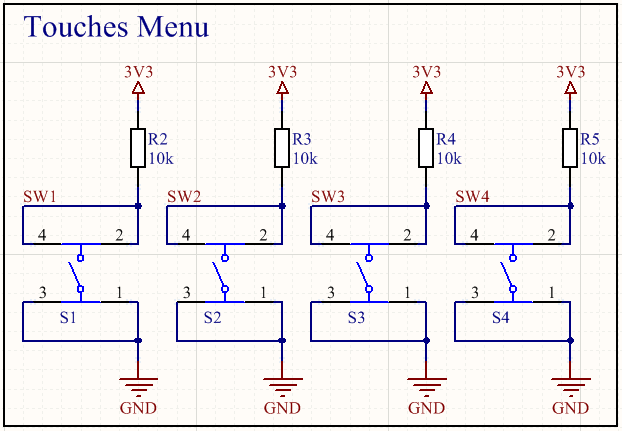
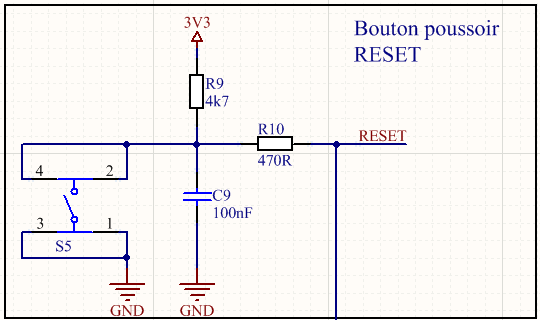
### Affichage

Ajout d'une source de courant pour

le backlight, et commutation on-off

### Touches menu et RESET

Rotation de 90° des switchs

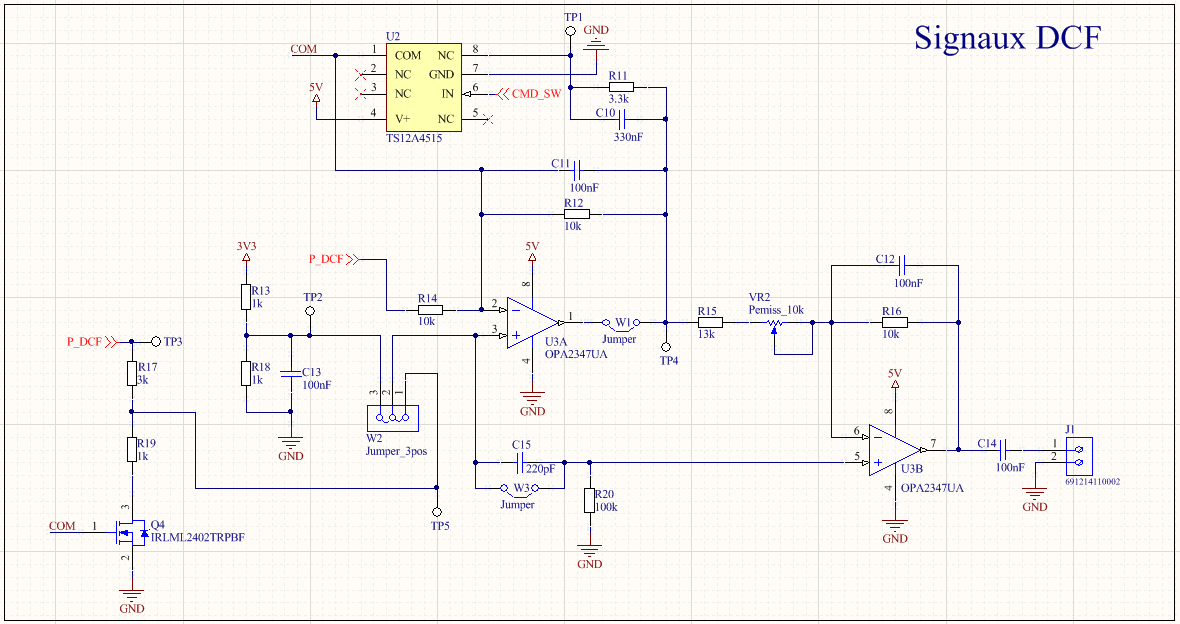
****

### Signaux DCF

Remplacer les deux jumpers à une position, par un seul à deux positions.

Ajouter un jumper en sortie de l'amplificateur U3A.

Adapter le gain de l'amplificateur U3B, en fonction du potentiomètre (Avmin = 0.8).



**Ajustement du gain**

**+**

**-**

R3

R1

R2

R2 = 10kΩ Gain min : 0.8

R3 = 10kΩ

Gain maximum si R2 est au minimum (0Ω) :

Av = - = 0.8 => R1 = = = 12.5kΩ Série E24 : 13kΩ

Gain minimum si R2 est au maximum (10kΩ) :

Av = - = - = 0.43

Vérification du gain après normalisation

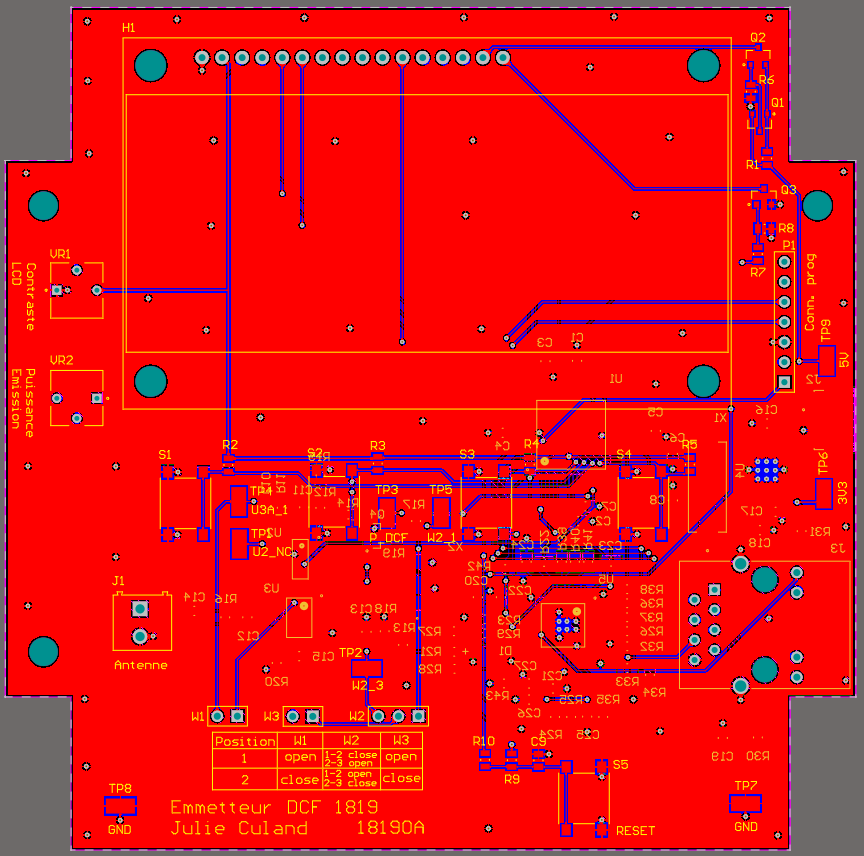
Avmax = - = - = -0.77

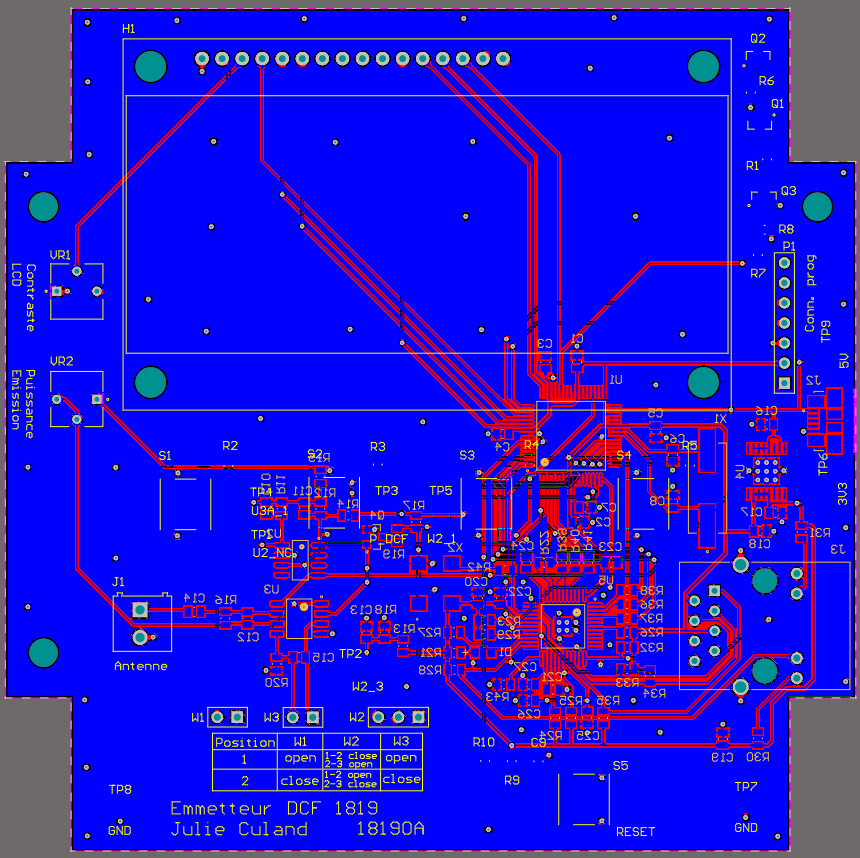
## PCB

### Routage

Pour le routage, les tailles des pistes et vias ont été paramétré en fonction des normes euro-circuit donnée. (**Voir Annexe VIII : Normes euro-circuit)**

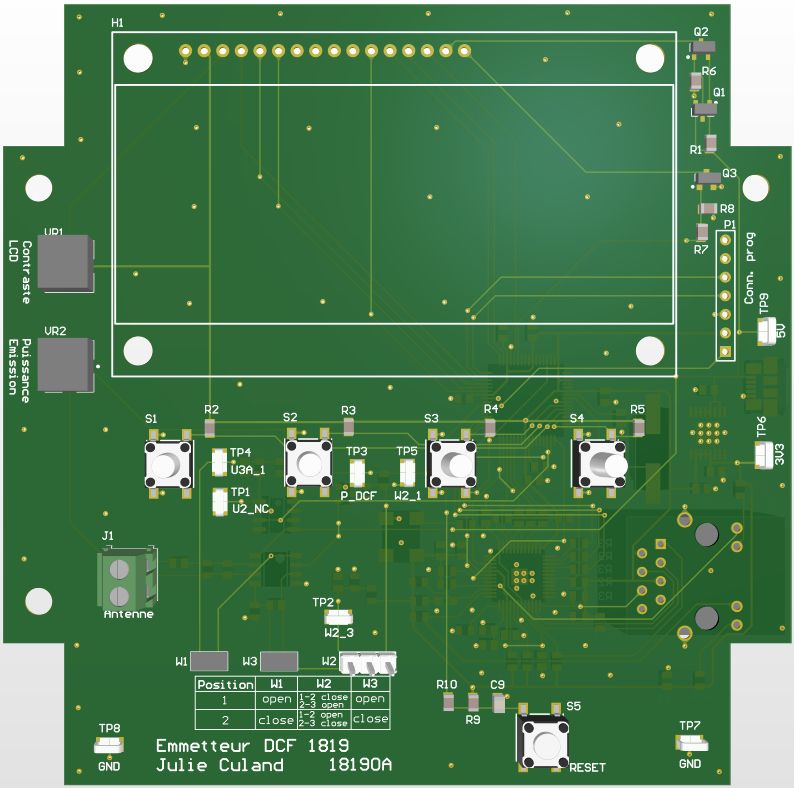
Des plans de masse ont été placés sur le TOP, ainsi que sur la face BOTTOM.





### Placement des composants

Concernant le placement des composants, sur le layer TOP, on était placé tous les éléments dont il faut avoir accès, qui permettent de réaliser l'interface homme-machine, soit : L'affichage LCD, switchs, potentiomètres, jumpers, Reset, points de test, connecteur de programmation, bornier pour l'antenne.

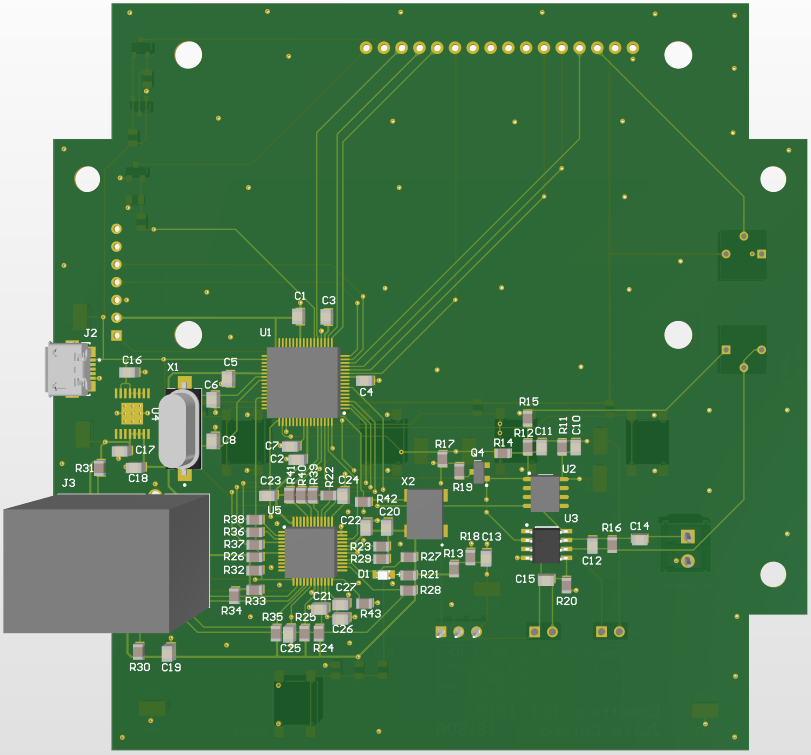


Pour faciliter l'utilisation des divers éléments, j'ai décidé de mettre une description à ceux-ci. J'ai également fait pareil pour le positionnement des jumpers, selon le circuit DCF que l'on souhaite utiliser.

Sur le layer BOTTOM, tous les autres composants on était placé, et notamment les deux connecteurs (micro-USB et Ethernet).

Comme le PCB sera surélevé dans le boîtier, et que le connecteur Ethernet a une hauteur plus ou moins importante, cela permettra de ne pas gêner l'affichage sur le TOP.

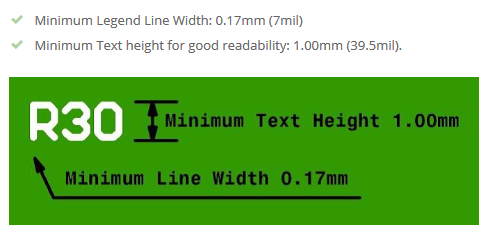
C'est donc pour cela, que ceux-ci ont été placés au bord du PCB (J2 et J3).



Concernant les fixations pour le PCB, l'une d'entre elles n'a pas été faite, car la place est prise par le connecteur RJ-45. Malgré tout, cela n'est pas gênant car il y a les trois autres trous de fixations qui sont utilisables.

### Sérigraphie

Pour la sérigraphie, les normes Euro-circuits ont également été reprises (dimensions texte).



### Eléments particuliers du PCB

|  |  |
| --- | --- |
| **Via sur plan de masse :**  Le plan de masse peut à certain endroit être relativement fin, on place donc des vias un peu partout, afin d'assurer une bonne connexion entre le plan de masse du TOP et du BOTTOM, ceci pour éviter tout bruit ou parasites. |  |
| **PAD du plan de masse :**  Afin d'assurer une bonne connexion entre le plan des layers, tous les PADs concernés (GND), sont reliés par un via.  La liaison entre le PAD et le via doit être le plus court possible (idem pour les alimentations). |  |
| **Teardrops :**  Consiste en l'ajout de matière, afin que les connexions soient plus solides entre les pistes. L'objectif principal est d'améliorer l'intégrité de la structure en présence de contraintes thermiques ou mécanique, mais surtout d'éviter des défauts de gravure, notamment aux endroits où il y a des angles droits. |  |
| **Dissipation de chaleur** :  Des zones avec plusieurs vias on était placé notamment sous le régulateur 3V3, ainsi que le CI Ethernet, afin de mieux dissiper la chaleur. |  |

## Mécanique

### Choix du boîtier

Pour le choix du boîtier, j'ai pris en compte les contraintes suivantes :

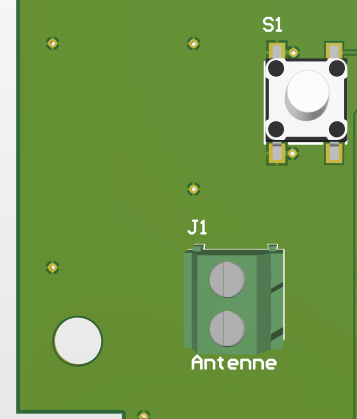
* Hauteur du PCB avec l'affichage et le connecteur Ethernet
* Facile à transporter et déplacer

J'ai ainsi choisi un boîtier de chez Hammond, avec un couvercle transparent afin de ne pas avoir besoin d'usiner une fenêtre pour l'affichage LCD.



### Emplacement de l'antenne

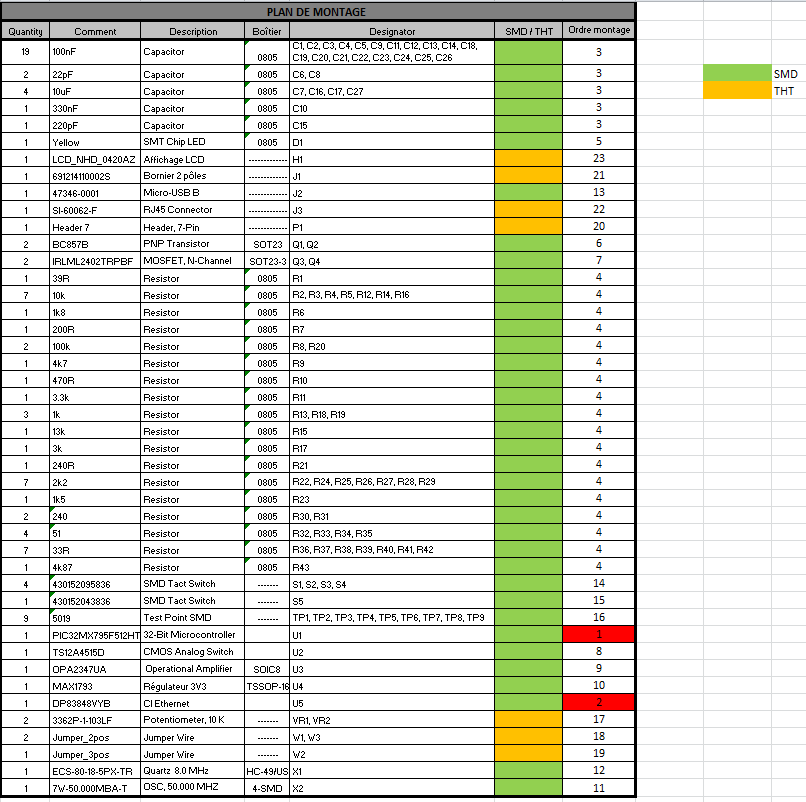
Dans la partie de l'étude du projet, j'avais comme première idée de placer l'antenne à l'extérieur du boîtier. Or comme celle-ci est relativement fragile et ne présente pas une importante surface, j'ai décidé de la positionner directement dans le boîtier sur le PCB, comme la place me le permet.



## Montage

Pour le montage, j'ai décidé de réaliser un plan, afin d'indiquer l'ordre dans lequel les composants vont être montés. Ceci afin de commencer par les composants les plus bas au plus haut (hauteur).

En revanche, les deux premiers composants qui seront montés sont le PIC32 ainsi que l'IC Ethernet, car ce sont ceux les plus difficile et risquer à braser.



## Test du PCB - Alimentations

Le but de cette étape est de tester que le circuit fonctionne sans le software, c'est-à-dire que les alimentations de la carte soient toutes fonctionnelles, ainsi que sur les différents ICs.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Valeur attendue** | **Valeur mesurée** | **Erreur absolue** |
| Alim 5V - TP9 | 5V | 5.08V | 1.6% |
| Alim 3.3V - TP6 | 3.3V | 3.292V | -0.24% |
| uC - VDD (patte 10) | 3.3V | 3.290V | -0.30% |
| uC - VBUS (patte 34) | 5V | 5.08V | 1.6% |
| U2 - V+ (patte 4) | 5V | 5.08V | 1.6% |
| U5 - IOVDD33 (patte 22) | 3.3V | 3.284V | 0.48% |
| LCD - Vdd (patte 2) | 3.3V | 3.290V | -0.30% |

On peut voir que les alimentations du circuit sont toutes fonctionnelles.

## Réalisation du software

### Détails du protocole

La fréquence porteuse est de 77.5kHz. Le principal défaut de cette fréquence est sa sensibilité aux parasites. Il est donc important de vérifier la cohérence des données reçues, par le biais de bits de parité.

Les informations sont transmises sous forme binaire à raison d'un bit à chaque seconde. Celles-ci sont codées en BCD (décimal codé en binaire), leur décodage fournit au récepteur les éléments comme la date et l'heure.

Les impulsions se traduisent chaque seconde par une diminution de 25% de l'amplitude du signal reçu. La durée d'impulsion détermine le niveau du bit reçu, à savoir qu'une impulsion de 100ms représente un '0' et qu'une impulsion de 200ms, représente un '1'.

Les informations horaires sont émises par trame d'une minute. Chaque trame est divisée en soixante secondes, chacun d'entre elles débutant par le front de l'impulsion.

Il faut noter que, pour la 59e seconde, il n'y a pas d'impulsion afin de permettre au décodeur de repérer le début d'une trame.

Ainsi, l'impulsion suivante détermine le début de la trame suivante. Toute absence d'impulsion plus grande que 999ms doit donc être considérée comme le début d'une nouvelle trame.

La synchronisation des récepteurs se fait sur le premier bit (bit n°0). L'apparition de la première modulation marque alors le début d'une nouvelle minute.

L'heure sera reçue via Ethernet et sera mise à jour via le protocole SNTP (Simple Network Time Protocol).

Ce protocole permet de synchroniser, via un réseau information, l'horloge locale d'ordinateurs sur une référence d'heure.

En revanche, ce protocole ne s'occupe pas du changement de l'heure dû au fuseau horaire, ainsi que du passage à l'heure d'été et d'hiver.

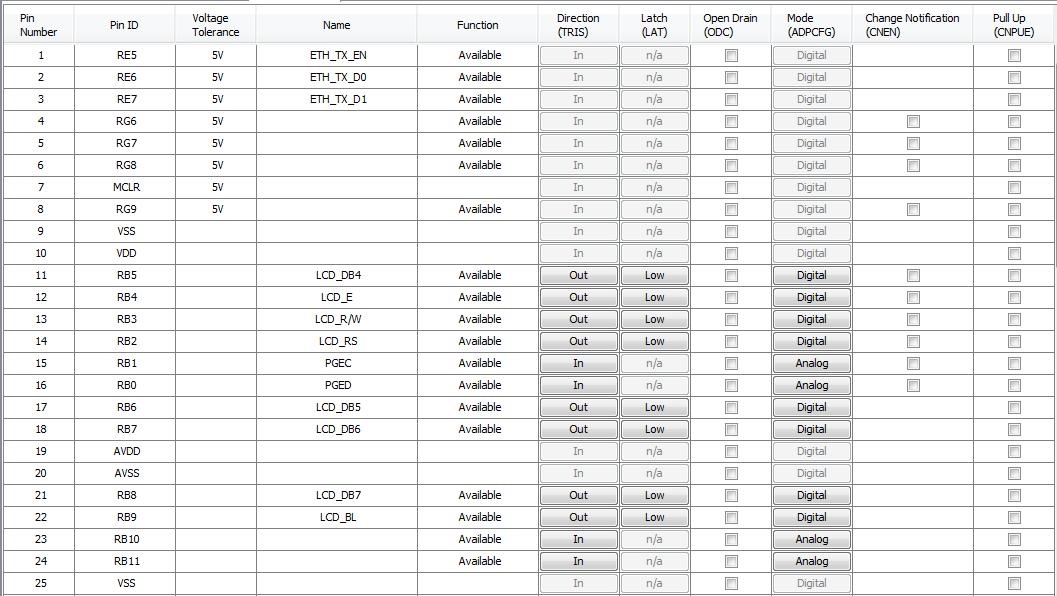
### Programmation sous MPLAB

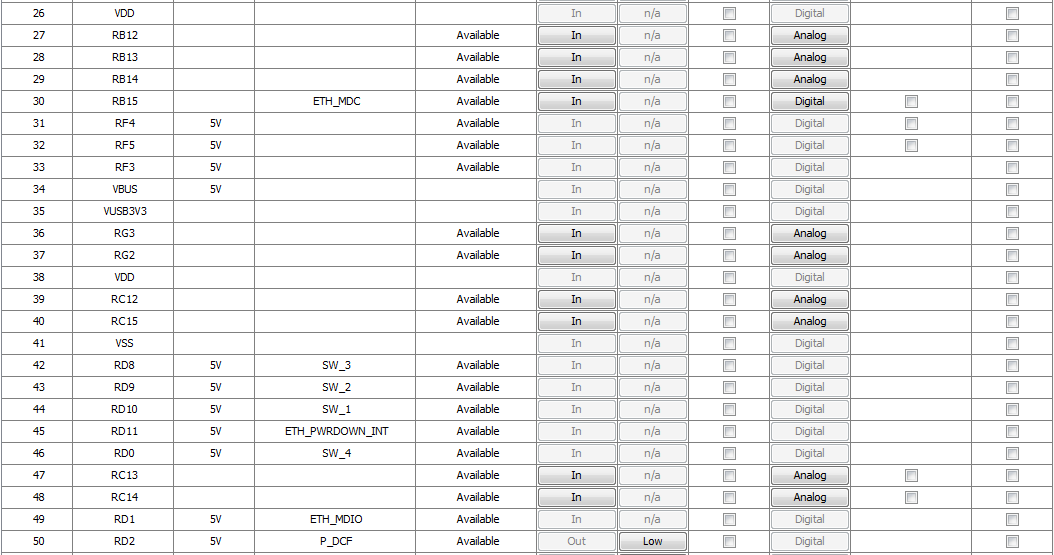
#### Paramétrage de MPLAB Harmony

##### Pin Settings

Avant de commencer à programmer, il faut paramétrer les pins settings, soit toutes les pins qui seront utilisées à la programmation.

Voici donc la configuration réalisée sous "Harmony Configurator > Pin settings" :







##### Timer 1

Le Timer 1 est paramétré pour avoir une période de 1ms avec un niveau de priorité à 3. Celui-ci va permettre de gérer les différents menus.

fosc = 80MHz TTIMER = 1ms

Tosc = = = 12.5ns

Nbcoup = = = 80'000

Comme le Timer a 216 pas, soit 0 à 65'535, la valeur calculée est trop élevée vu qu'elle dépasse la valeur max.

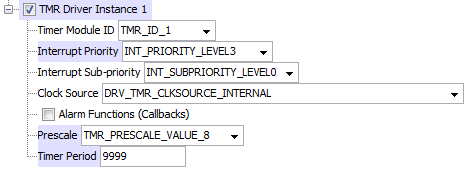
Pour remédier à ça, il faut utiliser un "prescale", soit diviser notre valeur, afin qu'elle soit dans l'échelle de travail du Timer.

Nbcoup = = = 10'000

On peut voir qu'avec un "prescale" à 8, la nouvelle valeur est de 10'000, ce qui est correct comme nous sommes dans la tranche de valeur du Timer.

Attention : Comme l'on commence à compter à 0, la valeur finale sera donc Nbcoup -1, soit 9999.

La configuration du Timer 1 est donc la suivante :



##### Timer 2

Il faut créer un Timer de façon à génerer la fréquence de 77.5kHz pour la porteuse du signal DCF.

fosc = 80MHz fporteuse = 77.5kHz

Tosc = = = 12.5ns

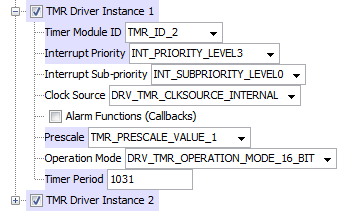
Tporteuse = = = 12.90us

Nbcoup = = = 1032.258 = 1032

Comme le Timer a une échelle de valeur de 216 (0 à 65'535), on peut voir que la valeur calculée se trouve dans la tranche de valeur, et qu'il n'y a ainsi pas besoin de prescale.

Attention : Comme l'on commence à compter à 0, la valeur finale sera de Nbcoup -1, soit 1031.

La configuration du Timer 2 est donc la suivante :



##### Timer 3

Le Timer 3, d'une période de 200ms, va permettre de gérer la modulation.

fosc = 80MHz TTIMER = 200ms

Tosc = = = 12.5ns

Nbcoup = = = 16'000'000

Comme le Timer a 216 pas, soit 0 à 65'535, la valeur calculée est trop élevée vu qu'elle dépasse la valeur max.

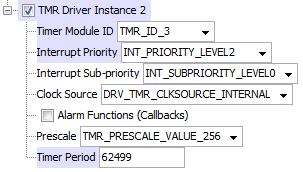
Pour remédier à ça, il faut utiliser un "prescale", soit diviser notre valeur, afin qu'elle soit dans l'échelle de travail du Timer.

Nbcoup = = = 62'500

On peut voir qu'avec un "prescale" à 256, la nouvelle valeur est de 31'250, ce qui est correct comme nous sommes dans la tranche de valeur du Timer.

Attention : Comme l'on commence à compter à 0, la valeur finale sera donc Nbcoup -1, soit 62'499.

La configuration du Timer 3 est donc la suivante :



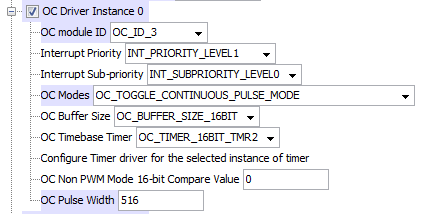
##### OC3

L'OC3 va permettre d'obtenir le signal pour la porteuse DCF à 77.5kHz, qui sera donc en relation avec le Timer2. Cet OC a été choisi car le signal DCF est directement connecté sur la pin OC3 (patte 50).

Celui-ci est en mode "PWM", est et paramétré pour fonctionner à 50% de la valeur du Timer 2,

soit : 1032 / 2 = 516

La configuration de l'OC3 est donc la suivante :

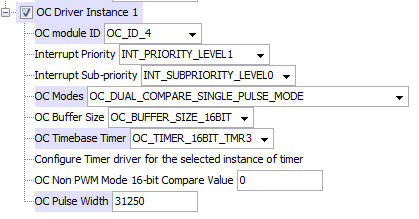


##### OC4

L'OC4 va permettre de gérer la durée des impulsions pour la modulation, en relation avec le Timer 3, qui a une période de 200ms.

Celui-ci va donc être réglé, pour être à une valeur de 50%, soit 62'500 / 2 = 31'250, qui correspond à 100ms. L'OC Modes doit être paramétré en "oneshot", soit en mode "OC\_DUAL\_COMPARE\_SINGLE\_PULSE\_MODE".

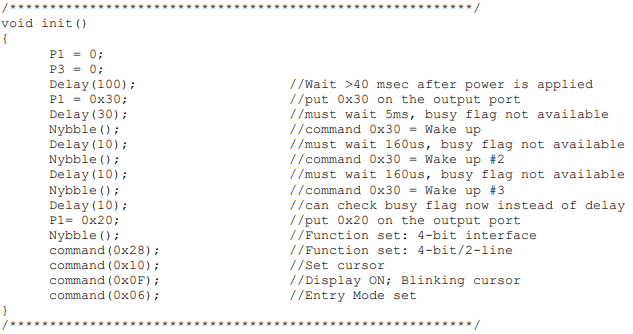
La configuration de l'OC4 est donc la suivante :



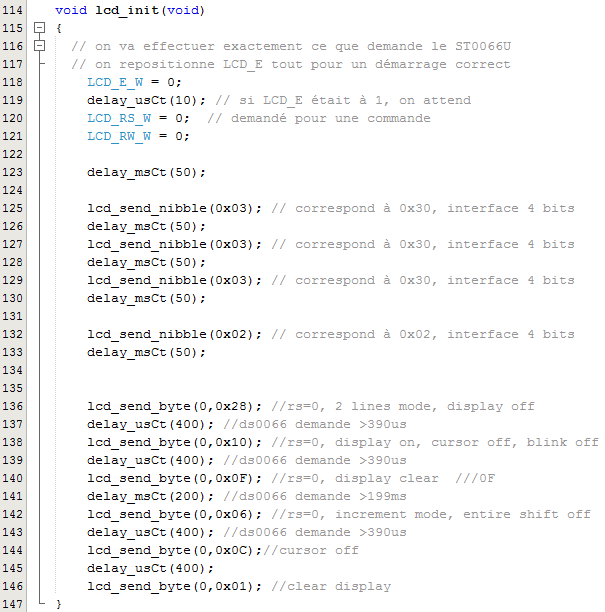
### Affichage LCD

Pour la programmation de l'affichage LCD, j'ai repris les fichiers "Mc32DriverLcd.c" et "Mc32DriverLcd.h", comme l'affichage que j'utilise et similaire à celui du kit PIC32.

Pour l'adapter, j'ai modifier les numéros des ports pour chaque pins et j'ai repris la configuration donnée dans le datasheet de l'affichage NHD-0420AZ-FSW-GBW-33V3, concernant l'initialisation[[1]](#footnote-1).



Ce qui donne le code suivant :



#### Problèmes rencontrés

Lorsque l'affichage LCD a été testé, le backlight s'allumait bien, or aucun caractère ne s'affichait.

L'erreur provenait du fait que les durées de temps paramétrées correspondaient à un clock de 40MHz du microcontrôleur, alors que celui de la carte est de 80MHz.

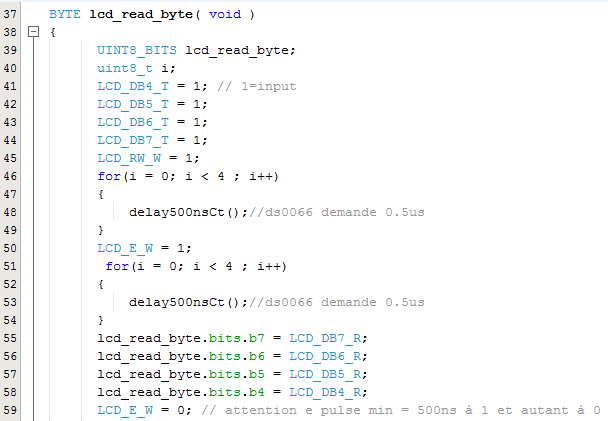
Cela signifie que le clock était beaucoup trop rapide pour les durées réglées et les caractères n'avaient ainsi pas le temps de s'afficher. Pour identifier le problème, j'ai réalisé un debug en mode pas à pas, afin de regarder où le programme restait bloquait.

J'ai ainsi pu observer que le programme restait bloqué dans la fonction "lcd\_send\_byte", à la ligne 105, soit en lien avec la fonction "lcd\_read\_byte()".



J'ai donc multiplié les durées par un facteur 4, grâce à un "for", afin de rendre le code plus lisible, comme de base le temps est de 500ns, ce qui est beaucoup trop court par rapport au clock de 80MHz.

Ce qui donne la modification suivante dans la fonction "lcd\_read\_byte" :



### Ethernet

#### Programme de test

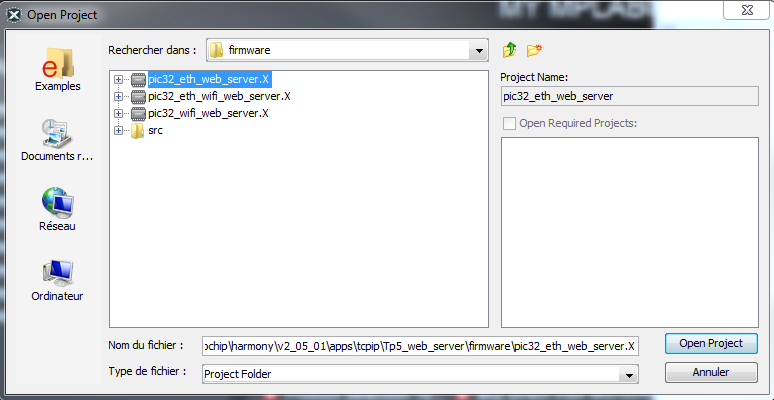
Pour tester la partie Ethernet, j'ai repris le projet "web\_server\_nvm\_mpfs", qui se trouve sous :

**C:\microchip\harmony\v2\_05\_01\apps\tcpip\web\_server\_nvm\_mpfs**

Afin de ne pas modifier le contenu de l'exemple, j'ai recopié celui-ci et est renommer le répertoire.

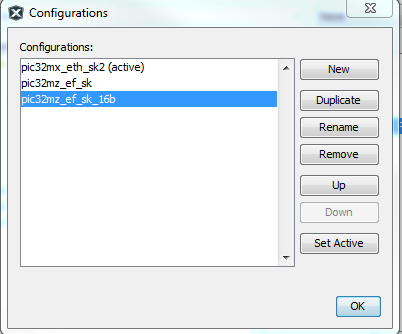


Ensuite, j'ai ouvert "MPLAB X IDE v4.15", puis suis allé chercher le projet que l'on vient de renommer, et sélectionner le projet "pic32\_eth\_web\_server.X".

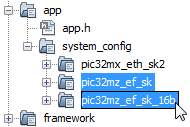


Ensuite, on va supprimer les configurations dont nous n'avons pas besoin. Pour ce faire, il faut refaire un clic droit sur le projet -> Properties -> Manage Configuration ...

On retire les configurations qui ne nous intéressent pas, soit celle en "mz", que l'on "Remove".



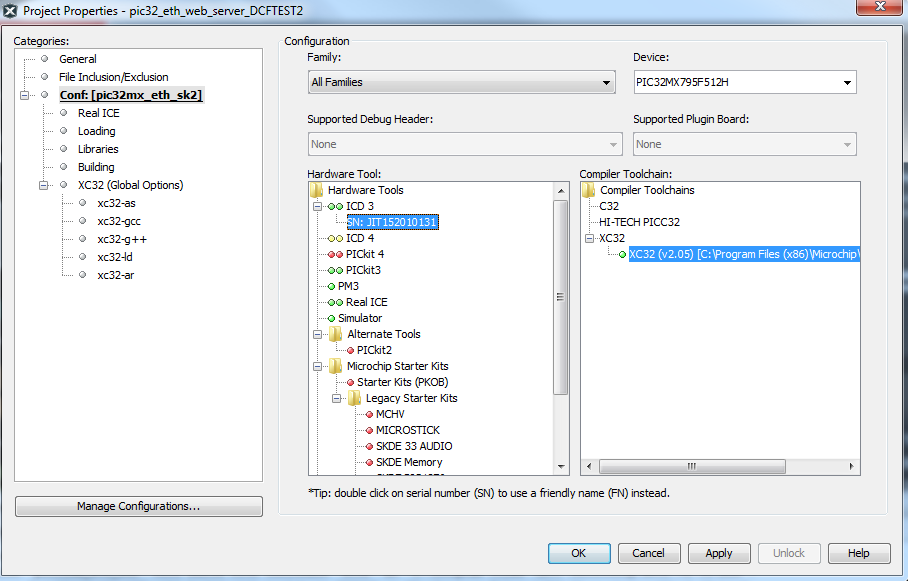
On réalise la même opération pour les fichiers sources "headers files" qui ne sont pas utiles (mz), que l'on "Remove" du projet.

****

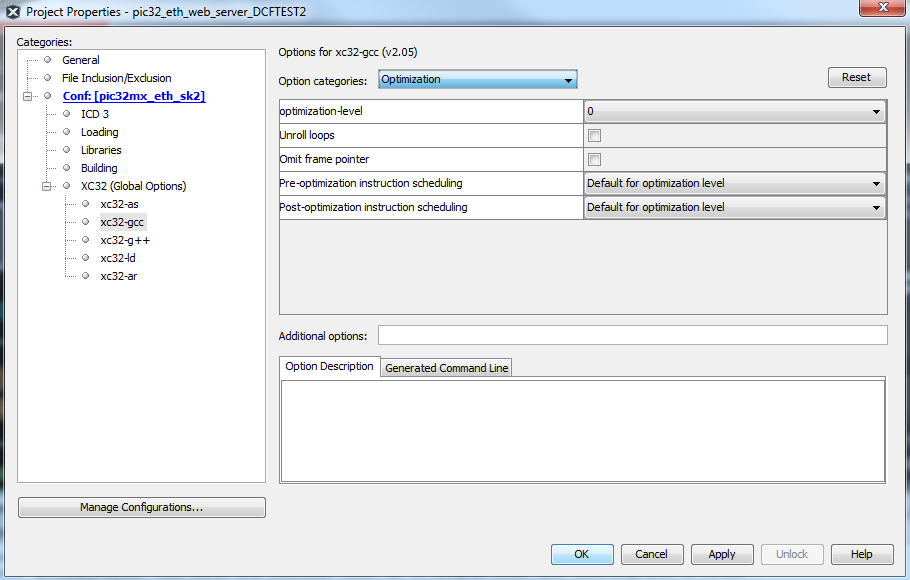
Dans les propriétés du projet, il faut se rendre dans les propriétés :

Clic droit sur le projet -> Properties

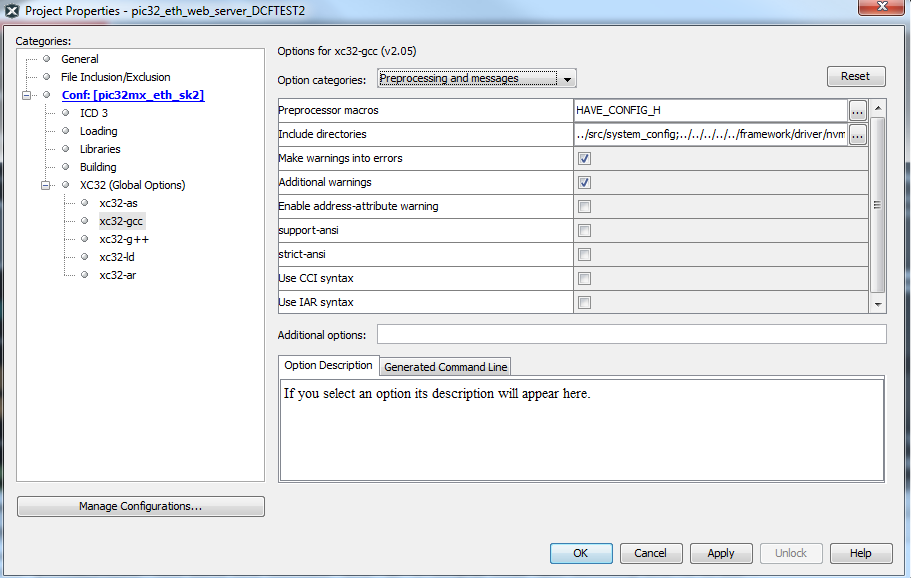
Dans "Device", il faut sélectionner le bon microcontrôleur : PIC32MX795F512H

****

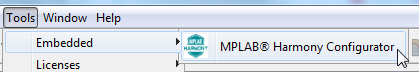
Dans "xc32-gcc", dans l'option "Option categories", il faut sélectionner "Optimization" puis mettre "optimization-level" à 0.

****

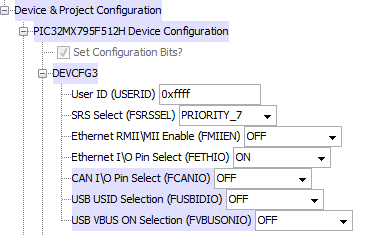
Toujours dans le "xc32-gcc", il faut sélectionner l'option "Preprocessing and messages", et cochez "Additional warnings".

****

Ensuite, il faut se rendre dans "MPLAB Harmony Configurator".



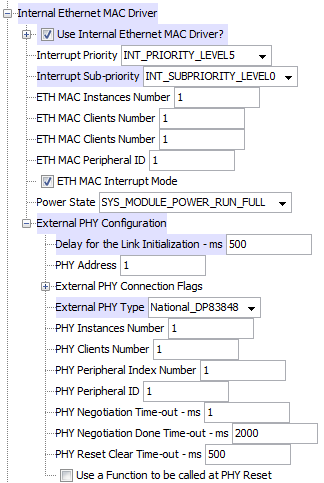
Dans les "Device & Project Configuration", il faut mettre à "ON" l'Ethernet I\O Pin Select (FETHIO), pour utiliser le port Ethernet par défaut.

****

On active le "DEBUG" et on sélectionne également "ICS\_PGx2".

****

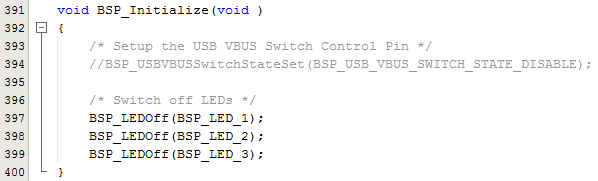
On sélectionner le bon cheap pour l'Ethernet "DP83848".

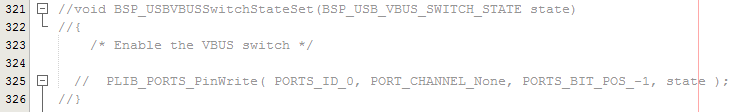
****

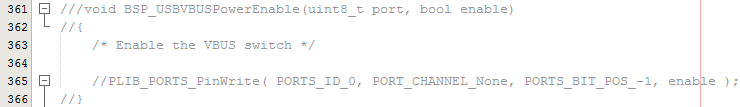
Lorsque les différents réglages sont terminés, j'ai compilé le projet et plusieurs erreurs sont survenues, qu'il faut corriger de la manière suivante :

Comme l'usb n'est pas utilisé dans le projet, les éléments suivants ont été mis en commentaire

**Fichier "bsp.c"**







Le port A n'existe pas sur le microcontrôleur utilisé, également le mettre en commentaire.

**Fichier "sys\_ports\_static.c"**



### Tests

Afin de voir si la partie Ethernet est fonctionnelle, il faut en premier lieu, simplement connecter la prise réseau dans le connecteur RJ-45.

Malheureusement le premier test n'a pas été concluant, car les Leds du connecteur ne réagissaient pas, donc ne s'allumait jamais.

En premier lieu, j'ai refait un contrôle des brasures de la partie concernée, et je me suis aperçue qu'il y a avait deux pattes du chip Ethernet qui n'étaient pas brasée correctement. Malgré cette correction, le fonctionnement était toujours le même.

J'ai donc décidé de vérifier toutes les alimentations de cette partie, et j'en suis arrivé à la conclusion que celles-ci étaient toutes correctes, soit une valeur de 3.3V, comme on peut le voir dans le tableau ci-dessous.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Valeur théorique** | **Valeur mesurée** | **Erreur absolue** |
| Résistance R22, R23 | 3.3V | 3.292V | -0.24% |
| Résistance R24, R25, R21 | 3.3V | 3.292V | -0.24% |
| Résistance R26, R27, R28, R29, R30, R31, R32, R33, R34, R35 | 3.3V | 3.293V | -0.21% |
| Condensateur C21, C22, C23 | 3.3V | 3.293V | -0.21% |
| IOVDD33 - patte 32 | 3.3V | 3.292V | -0.24% |
| IOVDD33 - patte 48 | 3.3V | 3.293V | -0.21% |
| AVDD33 - patte 22 | 3.3V | 3.293V | -0.21% |
| X2 VDD - patte 4 | 3.3V | 3.286V | -0.42% |

Erreur absolue = \* 100

**Schéma de mesure**

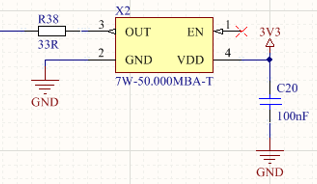
**X2**

**patte 3**

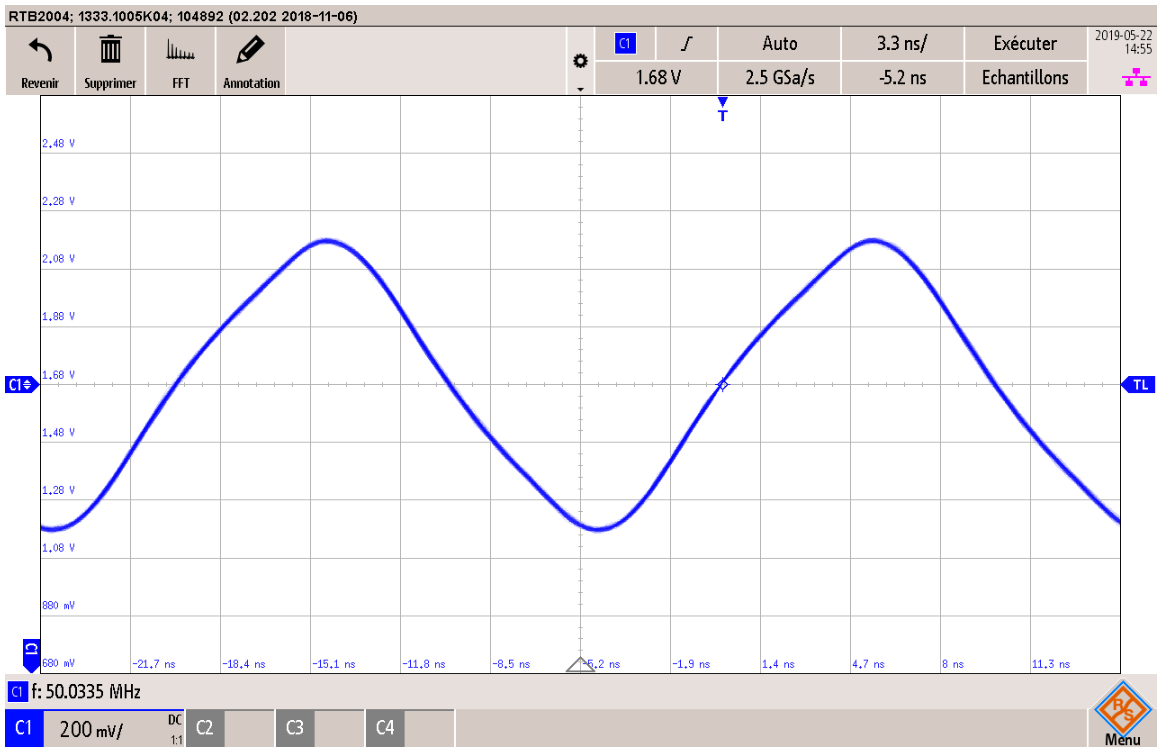
**CH4**

**CH1**

**P1**



Comme deuxième vérification, j'ai mesuré le signal en sortie de l'oscillateur, qui est la référence du clock à 50MHz. On observe ainsi que la fréquence de sortie est correcte.



Pour finir, j'ai effectué un contrôle de la valeur de tous les composants, et est revérifié le schéma, que j'ai également fait contrôler à des tierces personnes.

Après ces deux vérifications, je n'ai détecté aucun problème.

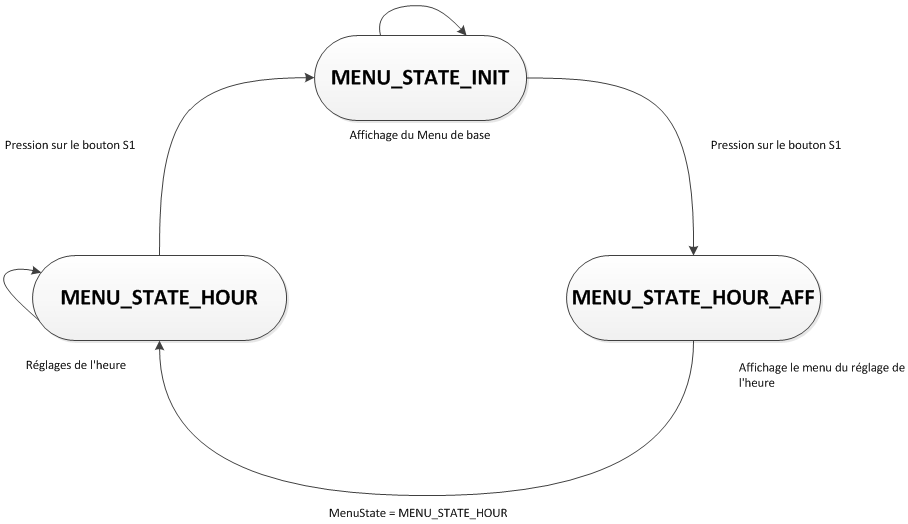
Afin de contrôler que le problème ne venait pas du software, j'ai repris le programme et changé le microcontrôleur, pour tester avec la carte du kit PIC32, qui c'est en effet avéré fonctionnel.

Je n'ai ainsi pas pu déterminer d'où venait le problème, car cette partie est relativement complexe.

Comme cette partie permettait de recevoir l'heure, celle-ci sera donc réglée à l'aide des boutons.

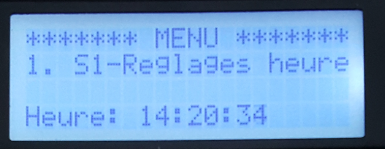
### Gestion des menus

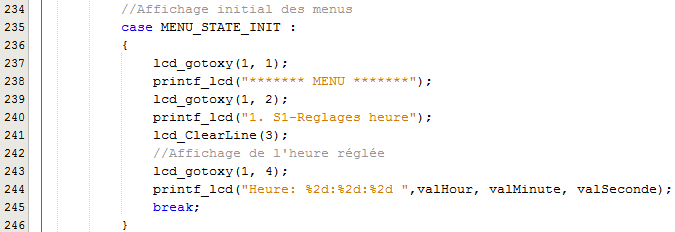
Pour la gestion des différents menus, j'ai décidé de réaliser une machine d'état.



**MENU\_STATE\_INIT :**

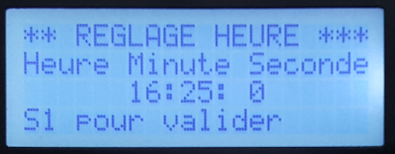
Affiche le menu initial, ainsi que l'heure avec les options de réglages possibles.

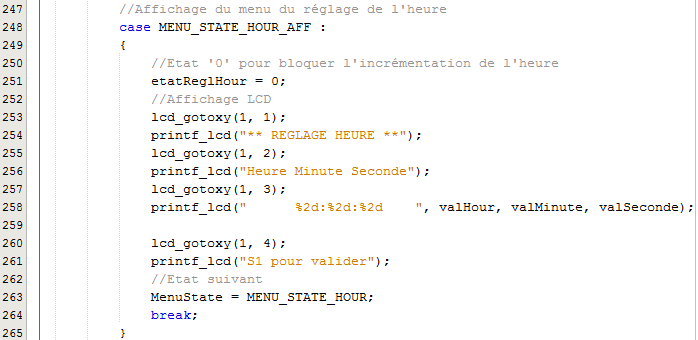
****



**MENU\_STATE\_HOUR\_AFF :**

Affiche le menu du réglage de l'heure.

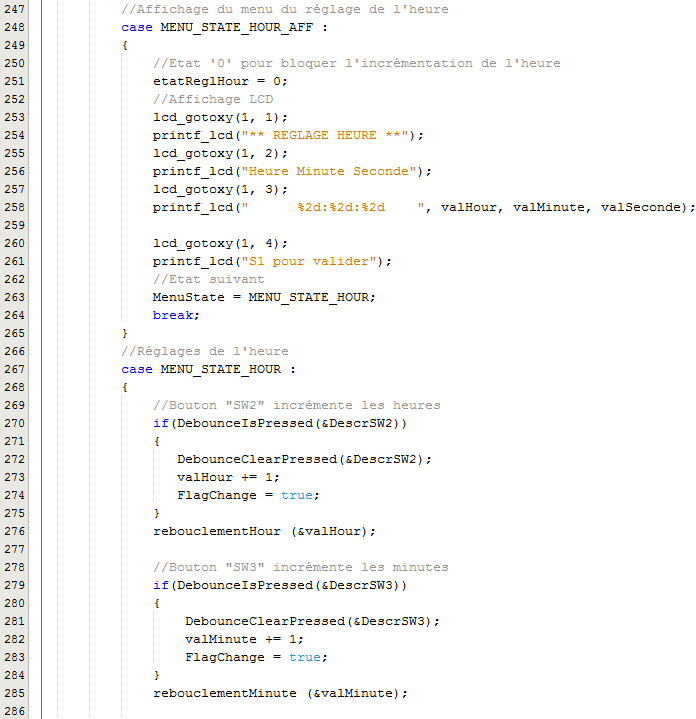
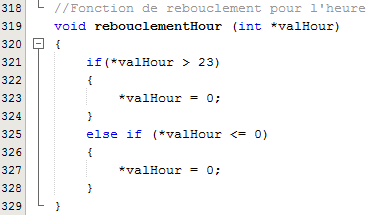
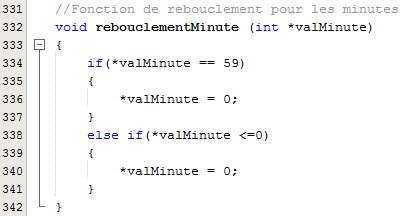
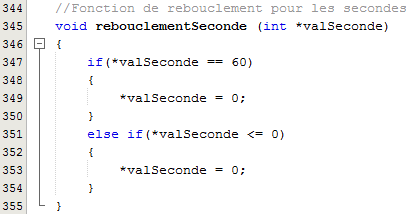
****



**MENU\_STATE\_HOUR** **:**

Permet de régler les paramètres de l'heure, à savoir les heures, minutes et secondes grâce aux boutons.

Pour les trois paramètres réglables, des fonctions de rebouclement ont été réalisées, afin de rester dans des plages de valeurs correctes.





### Signaux DCF

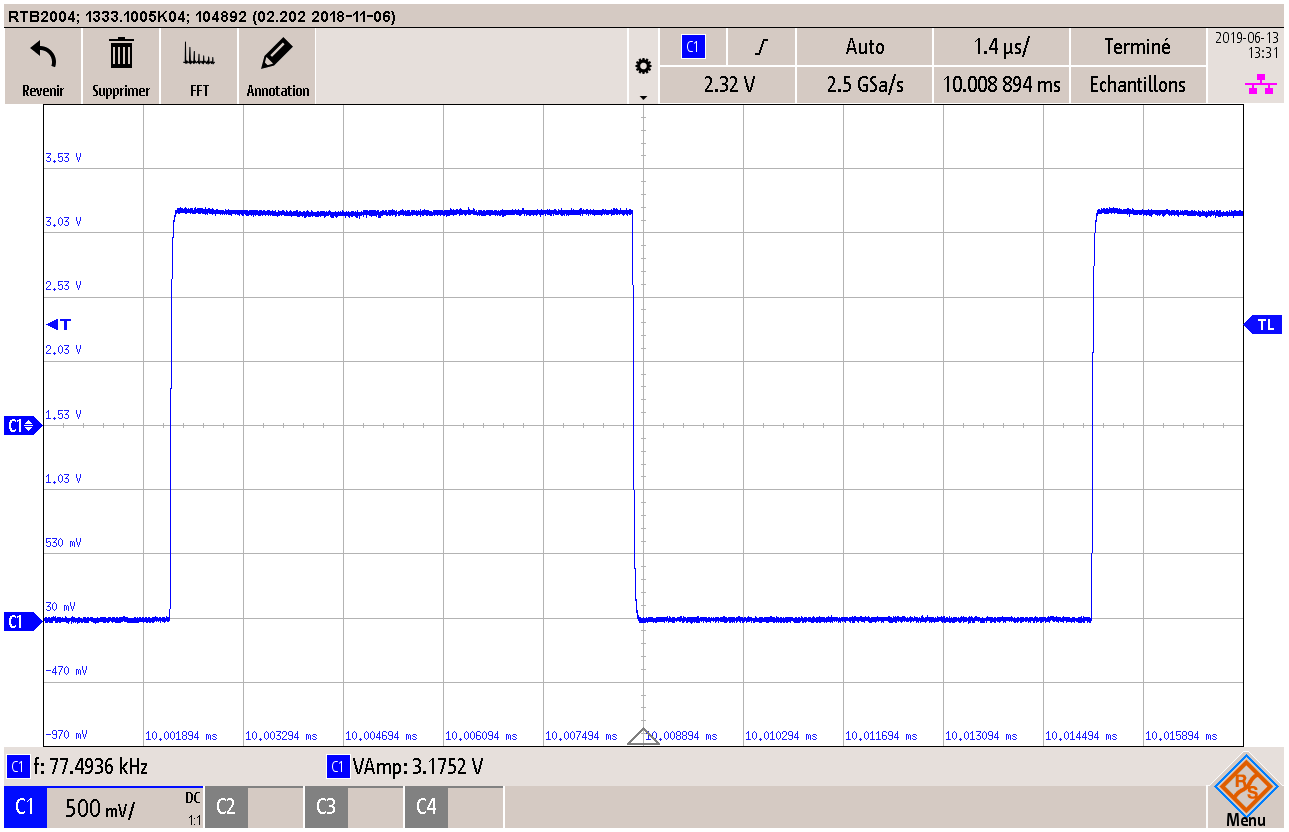
Cette partie permet de recréer le signal DCF, c'est-à-dire qu'une porteuse est modulée par des impulsions, au rythme d'une par seconde, avec une fréquence de 77.5kHz.

Ces impulsions se traduisent chaque seconde par une diminution de 25% de l'amplitude du signal reçu.

#### Porteuse

Le signal pour la porteuse a été créé sur la base d'un Timer, soit le Timer 2 et de l'OC3, qui permet de directement générer le signal à 77.5kHz, avec une amplitude de 3.3V. **(Voir point 3.6.2.1.3 Timer 2 & point 3.6.2.1.5 OC3).**

Comme le signal est directement sur la pin de l'OC3, celui-ci sera directement mis en sortie de celle-ci.



Fréquence : 77.4936kHz Amplitude : 3.1752V

#### Explications

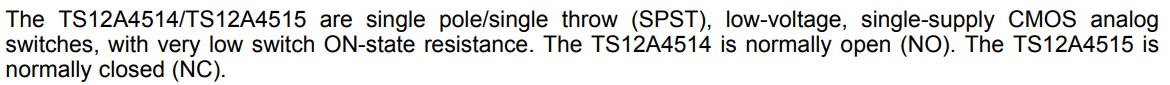
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Valeur théorique** | **Valeur mesurée** | **Erreur absolue** |
| Fréquence porteuse | 77.5kHz | 77.4936kHz | -0.008% |
| Amplitude | 3.3V | 3.1752V | -3.78% |

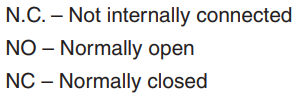
Erreur absolue = \* 100

On peut ainsi voir que la porteuse a bien les valeurs souhaitées.

#### Analog switch

Comme on peut le voir dans le datasheet, l'état initial du composant est "NC", donc fermé.

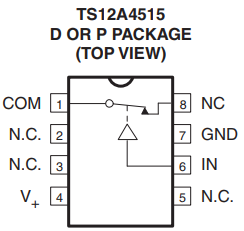


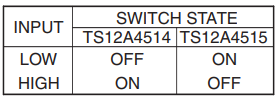


Dans le tableau ci-dessous, l'analog switch (TS12A4515) sera ouvert ou fermé selon l'état appliqué sur l'entrée IN.

Etat haut sur IN : Fermé → Actif

Etat bas sur IN : Ouvert → Repos

****

****

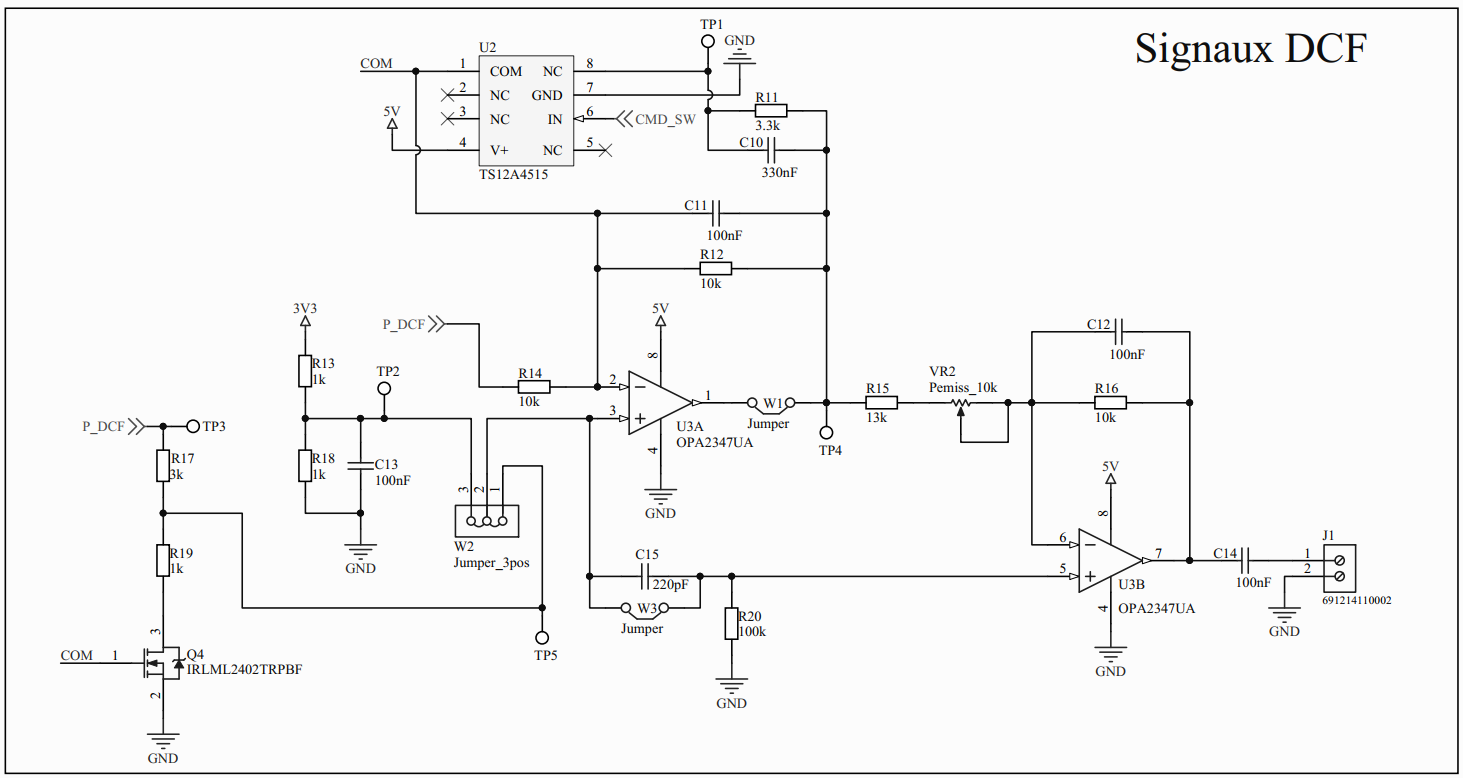
L'analog switch, lorsqu'il sera ouvert, va activer seulement le premier filtre (C10 et R7), ainsi le signal sera à 100%.

Dans le cas contraire, lorsqu'il sera fermé, cette fois-ci les deux filtres seront actifs, ce qui va permettre d'abaisser la tension du signal à 25%, soit ¼ de celle-ci.

#### Test de la partie analogique

Cette partie a pour but de réaliser un abaissement de 25% de l'amplitude du signal.

##### Possibilité 1



La première possibilité utilise la partie avec les composants suivants : Q4, R17, R19

Cela signifie que, pour que celle-ci fonctionne, il faut avoir les jumpers dans les états suivants :

W1 : Ouvert

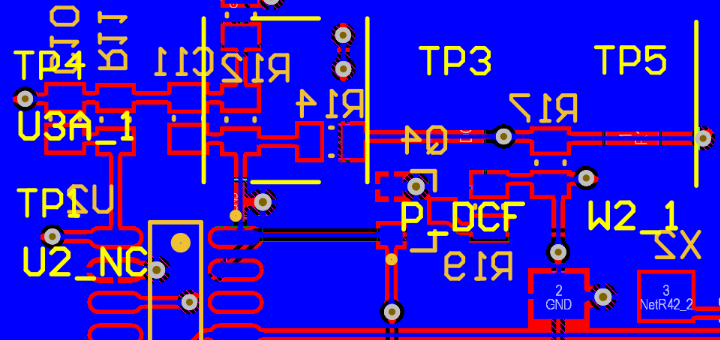
W2 : 1-2 Fermé

W3 : Ouvert

Or, après avoir réétudié le schéma, je me suis rendu compte que cette partie ne pourrait pas fonctionner, car elle présente un défaut au niveau du Hardware.

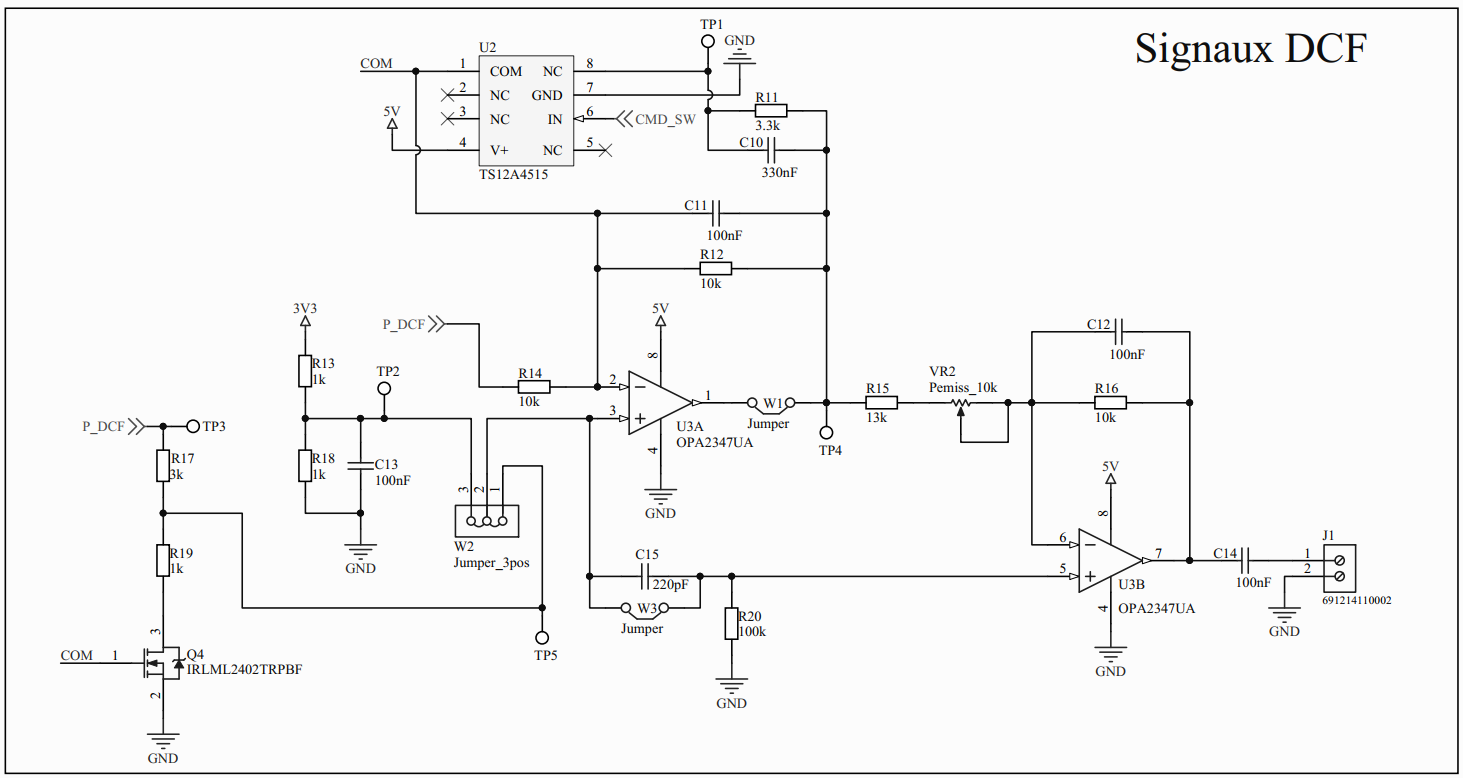
En effet, si le jumper W1 est ouvert, cela signifie qu'il n'y aura pas le signal voulu sur l'entrée de la partie concerné, soit 'COM'.

Afin que cette partie n'influence pas la seconde, j'ai décidé d'ôter les composants Q4 et R17.



**(Voir Annexe VI : Fichier des modifications)**

##### Possibilité 2



La seconde possibilité utilise la partie avec les composants suivants : R13, R18, C13

Cela signifie que, pour que celle-ci fonctionne, il faut avoir les jumpers dans les états suivants :

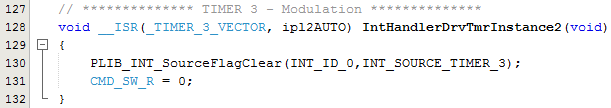
W1 : Fermé

W2 : 2-3 Fermé

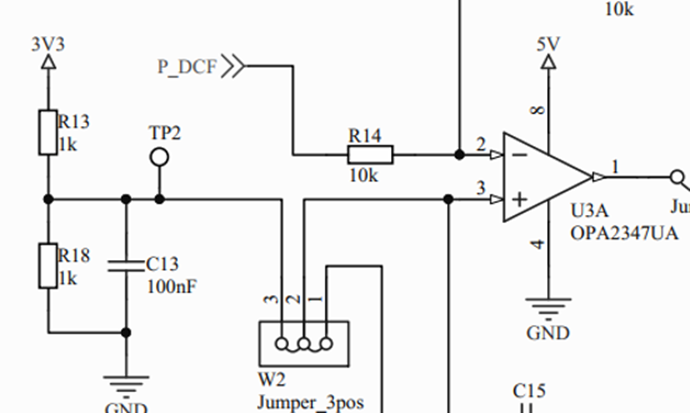
W3 : Fermé

#### Mesure partie analogique

Conditions : Analog switch à l'état "open" donc un état bas à l'entrée.



**Schéma de mesure**



**CH1**

**CH1**

P1

**U3**

**Patte 2**

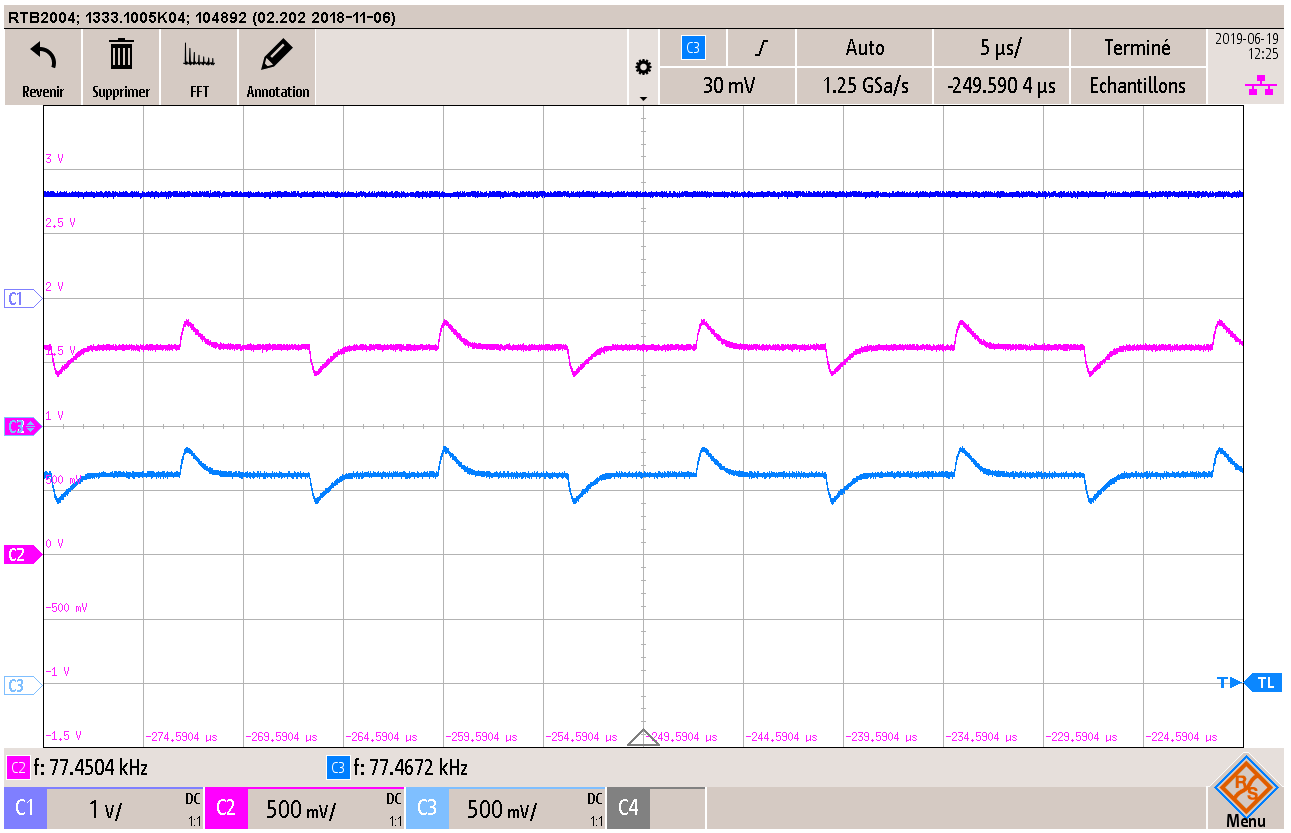
**CH2**

**CH3**

**TP4**

**TP2**

##### Mesure



Sur le canal 1, on retrouve bien la tension de 1.65V du pont diviseur (R13 et R18).

En revanche, on remarque que le signal sur la patte (-) de l'amplificateur, et identique à celui en sortie, ce qui n'est pas correct.

Cela permet d'émettre les hypothèses suivantes :

1. Il y a un court-circuit entre la sortie et l'entrée, ce qui créer un suiveur.

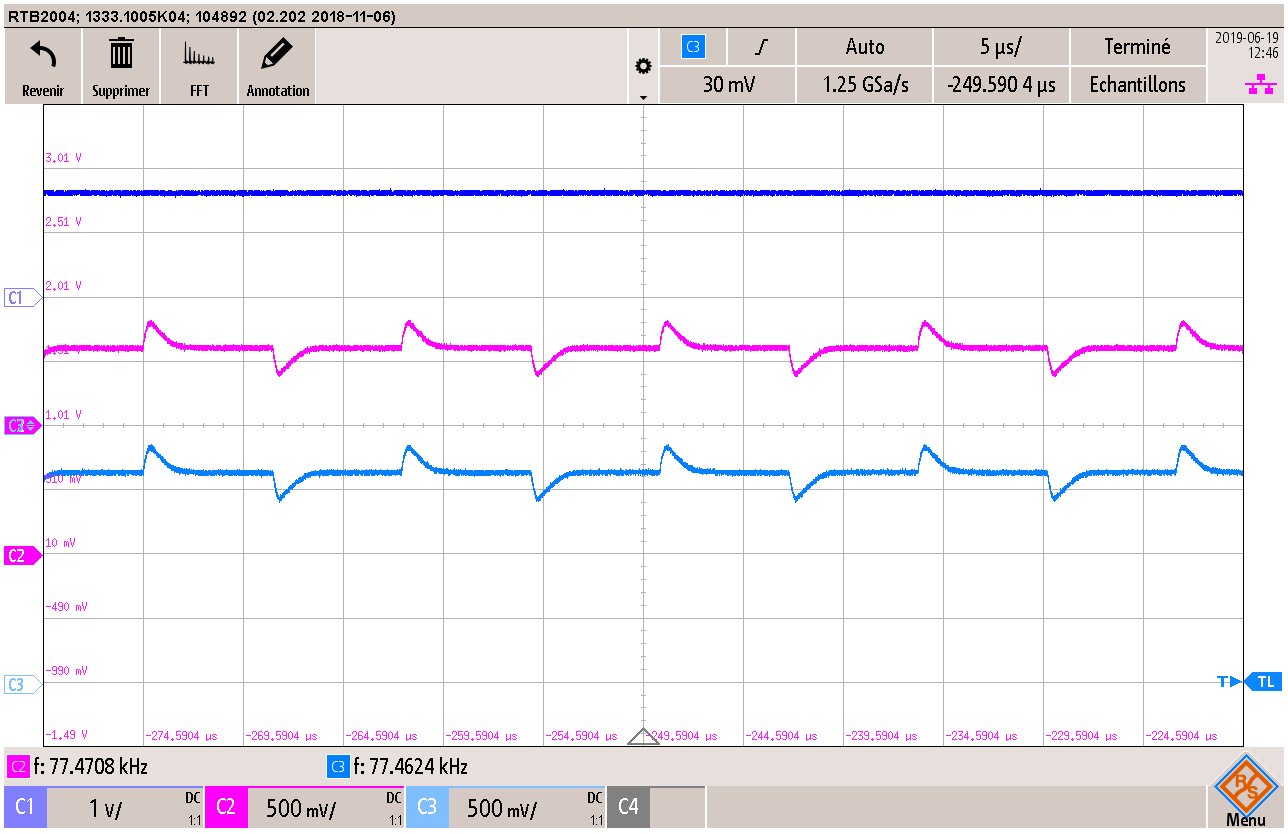
2. L'amplificateur U3 est KO ou n’est pas un choix judicieux.

#### Dépannage

En premier lieu, j'ai décidé d'ôter la résistance R11, afin que l'analog switch n'ait plus d'influence sur le système.



J'ai ainsi refait la même mesure que précédemment.



On observe que les signaux n'ont pas changé, et que l'analog switch était bien à l'état ouvert, et qu'il n'a pas d'influence sur le système.

Deuxièmement, j'ai recontrôlé toutes les brasures de la partie DCF, afin de vérifier qu'il n'y ait pas de court-circuit ou de pattes mal brasées, et là encore, tout est OK.

Troisièmement, j'ai vérifié toutes les valeurs des composants, et là tout était correct.

Comme dernière possibilité, je me suis penchée sur l'amplificateur U3. Je me suis ainsi demandé si la vitesse de balayage n'était pas trop importante.

##### Test de l’amplificateur OPA2347UA

Afin de vérifier que l’amplificateur utilisé supporte une fréquence de 77.5kHz, sans perturbation, j’ai décidé de prendre celui que j’avais en stock et de réaliser la configuration ci-dessous :

**+**

**-**

R12

R14

R12 = R14 = 10kΩ

Vref : 1.65V

1

2

3

**Schéma de mesure**

**+**

**-**

R12

R14

R12 = R14 = 10kΩ

Vref : 1.65V

1

2

3

**CH1**

**CH1**

P1

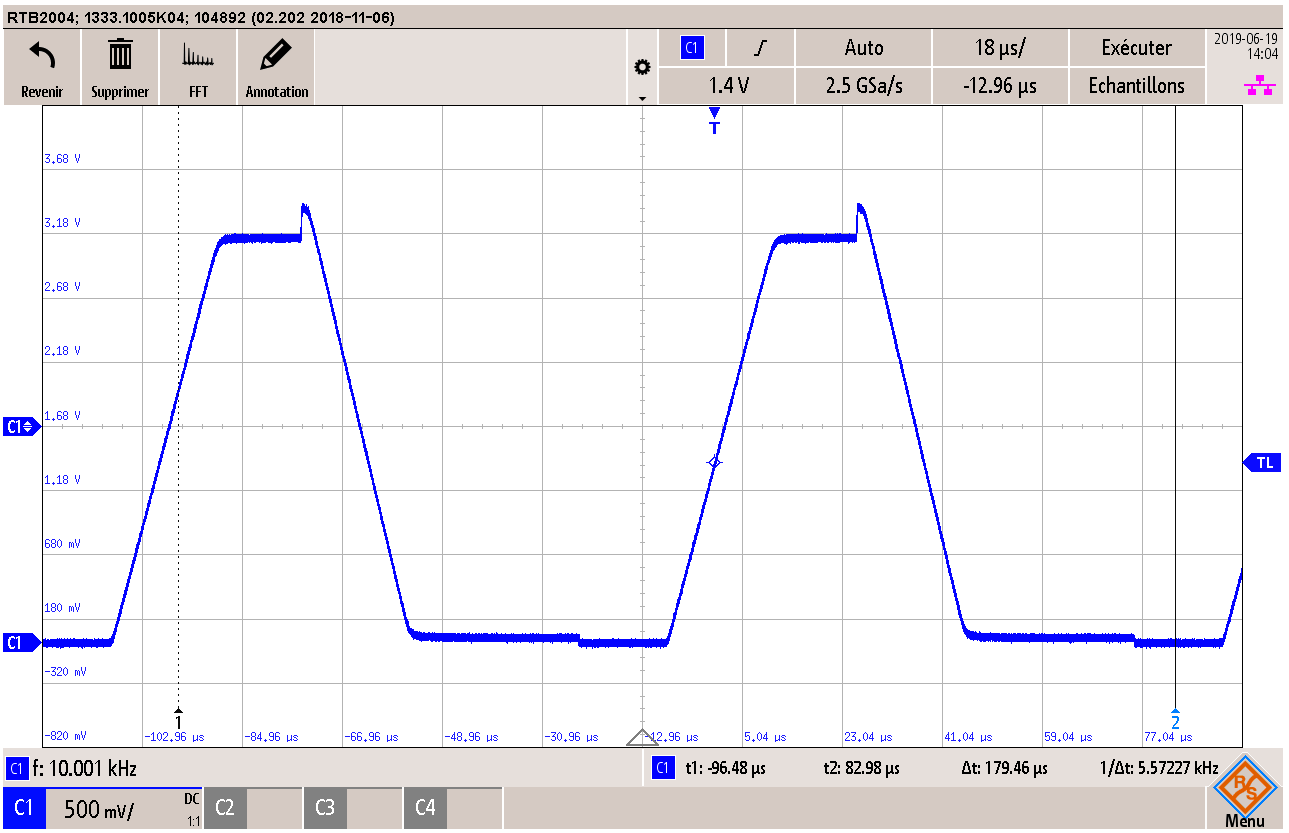
**U3 patte 1**

U3

G1

Le générateur est paramétré pour sortir un signal carré d’une amplitude de 3.3V, avec pour la première mesure une fréquence de 10kHz, et pour la seconde une fréquence de 77.5kHz.

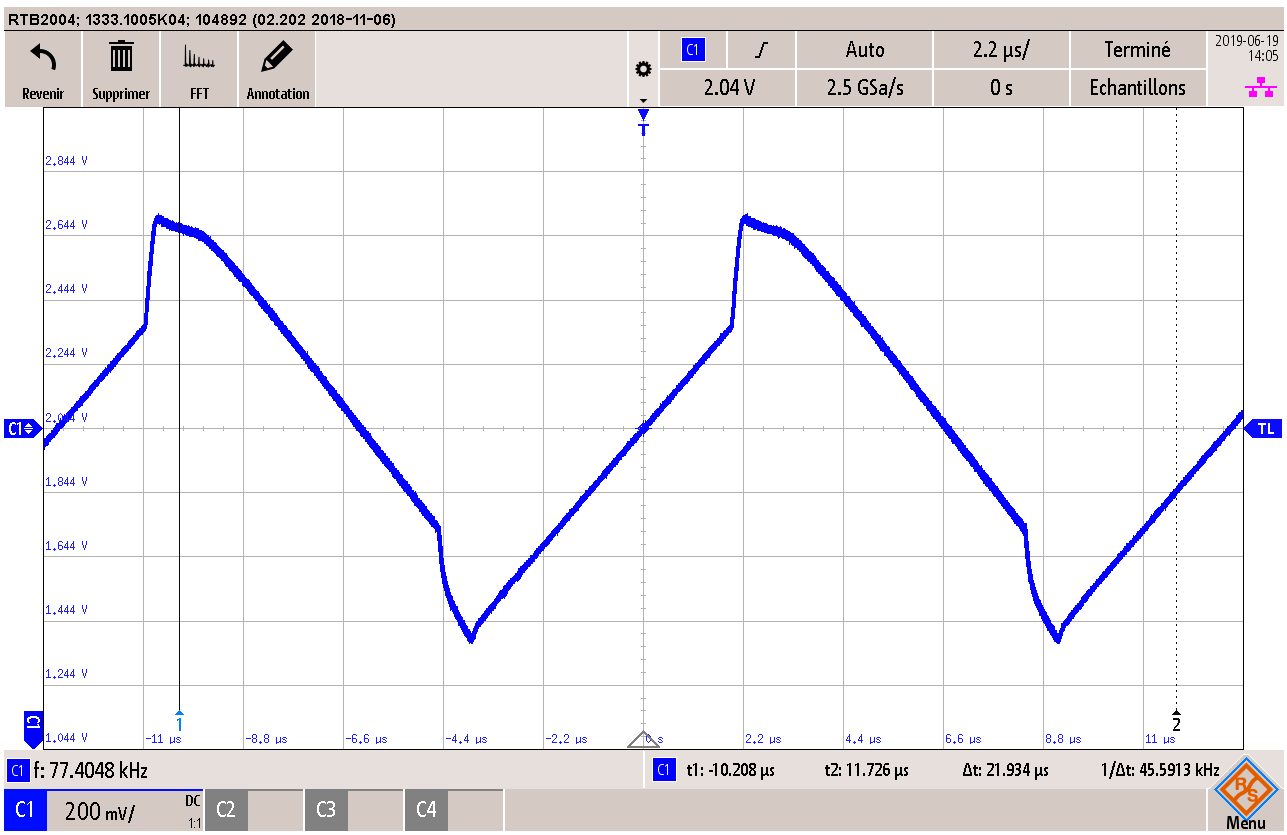
###### Mesure @ 10kHz



Δt

Δt = 18us ΔV = 3.18V

###### Mesure @ 77.5kHz

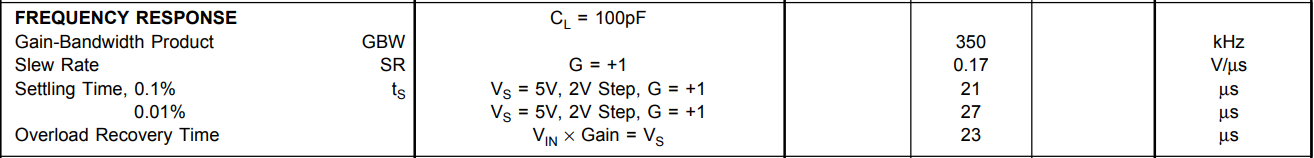


###### Explication des mesures

Sur la première mesure, à 10kHz, on peut voir que le delta en tension est de 3.18V et au niveau de la durée, celui-ci est de 18us.

On peut donc, à partir de ces valeurs, déterminer le slew rate : = = 0.176V/us

Lorsque l’on consulte le datasheet de l’OPA2347UA, pour un gain de 1, la valeur de slew rate est la suivante, 0,17V/us :



On voit donc que la valeur mesurée, correspond à celle donnée par le fabricant.

Sur la seconde mesure, on peut voir qu’à une fréquence de 77.5kHz, le signal n’a pas le temps de réagir correctement. C’est-à-dire que lorsqu’il a atteint sa pente maximum, il n’a pas assez de temps pour réagir et redescendre.

L’amplificateur choisi n’est donc pas adapté à ce niveau-là, ce qui explique le disfonctionnement de la partie analogique.

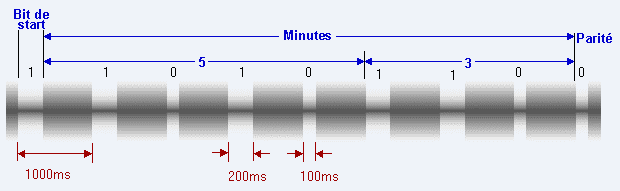
## Codage de l'heure

Comme l'heure est réglée manuellement, il faut la coder, afin que les informations envoyées puissent être reçues et lues par l'horloge.

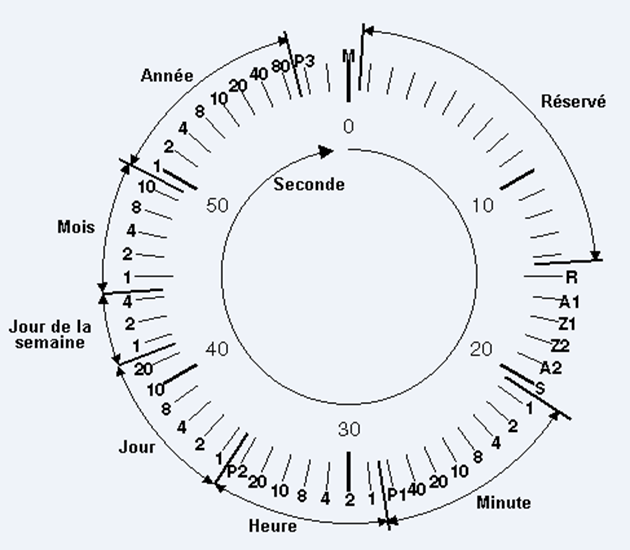
Les informations sont donc codées comme ci-dessous[[2]](#footnote-2) :



Exemple : codage de la minute 35



On peut voir les informations contenues dans le signal d’horloge, sous une forme schématique, où l’on voit la signification des 59 bits durant une minute.

****

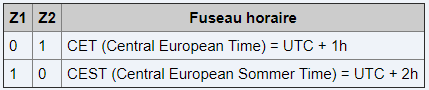
**0 (M) :** Début de trame (bit à 0)

**1 – 14 :** Réservé pour une utilisation future

**15(R) :** L’émetteur de réserve est actif lorsque ce bit est à 1

**16 (A1) :** Annonce de l’heure d’hiver

**17, 18 (Z1, Z2)** : Ces deux bits codent le fuseau horaire actuel



**19 (A2) :** Indique qu’une seconde va être supprimée pour

corriger les irrégularités de la rotation de la terre

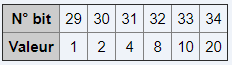
**20 (S) :** Bit de début de codage des informations horaire (toujours à 1)

**21 – 27 :** Minutes codées en BCD, bit de poids faible en premier



**28(P1) :** Bit de parité (parité paire) des minutes (bits 21 à 27)

**29 – 34 :** Heures codées en BCD, bit de poids faible en premier



**35 (P2) :** Bit de parité (parité paire) des heures (bits 29 à 34)

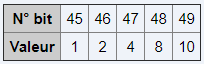
**36 – 41 :** Jour codé en BCD, bit de poids faible en premier



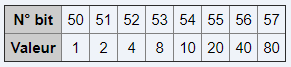
**42 – 44 :** Jour de la semaine codée en BCD, bit de poids faible en premier



**45 – 49 :** Mois codés en BCD, bit de poids faible en premier



**50 -57 :** Année (sur deux chiffres) codées en BCD, bit de poids faible en premier

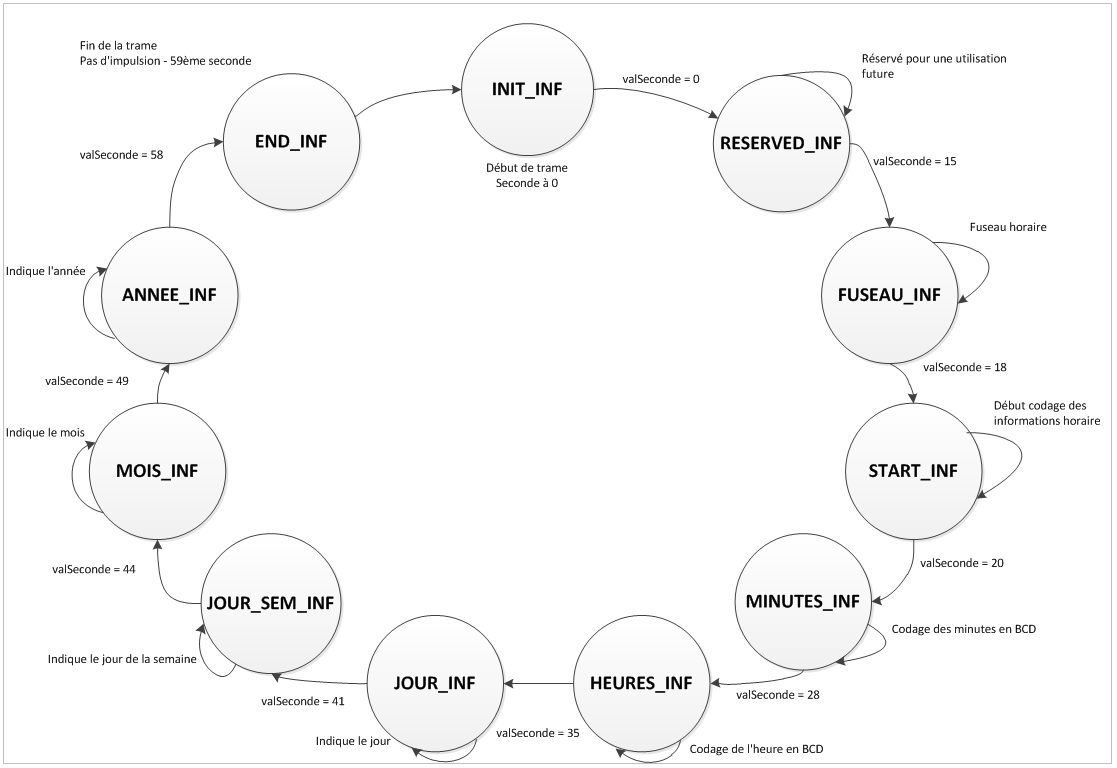


**58 (P3) :** Bit de parité (parité paire) de la date (bits 36 à 57)

**59 :** Pas d’impulsion

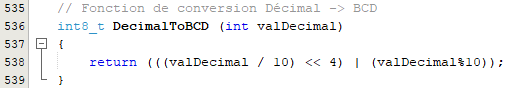
#### Machine d’état

Comme la trame est composée de plusieurs éléments, j'ai décidé de réaliser une machine d'état avec chacun de ceux-ci, qui sera cadencée à la seconde.



#### Conversion des minutes et de l’heure en BCD

Comme les valeurs de l’heure sont en décimal, il faut les convertir en BCD pour pouvoir les envoyer. J’ai ainsi créé une fonction, à laquelle j’envoie la variable contenant les minutes et l’heure :



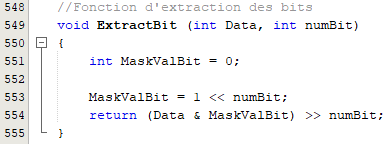
Chaque décimal doit être traitée individuellement, par exemple si les minutes valent « 52 », cela équivaut en BCD : 5 -> 0101 2 -> 0010 soit 0x52

En divisant, tout d’abord, par ‘10’ cela donne le premier chiffre, puis le modulo (%) par ‘10’ donnera le deuxième chiffre. Dès que le premier chiffre est obtenu, le décalage à gauche le déplace de 4 bits vers l’avant. L’opération ‘OU’ permet d’obtenir le résultat.

1. 52 / 10 = 5.2 => Comme nous travaillons avec des entiers, seul le ‘5’ est gardé
2. Décalage de 4 bits sur la gauche, donc 0x05 devient 0x50
3. 52 % 10 = 2 => Le modulo permet d’obtenir le second chiffre
4. L’opération ‘OU’ donne le résultat => 0x50 | 0x02 = 0x52

#### Extraction des bits

Comme les valeurs des minutes et des heures sont obtenues en BCD, après la conversion, il faut à présent extraire chaque bit pour pouvoir les envoyer les uns après les autres.



En premier lieu, on envoie la data, donc la valeur en BCD, puis le numéro du bit.

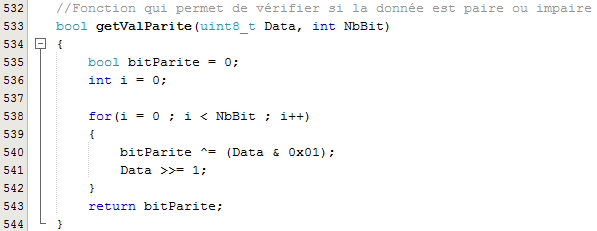
Si l’on reprend la valeur précédente, soit « 0x52 », qui donne 0101 0010, et qu’on envoie numBit à ‘2’, donc on extrait le troisième bit de poids faible.

1. 0000’0001 << 2 = 0000’0100 => Décalage en fonction du Bit
2. 0101’0010 & 0000’0100 = 0000’0000 = 0x00 => Opération ‘&’
3. 0000’0000 >> 2 = 0 => Décalage du résultat avec le bit

L’opération est donc répétée 7 fois pour les minutes et 6 fois pour les heures.

#### Vérification de la parité

Comme certaines données ont une parité (paire), j’ai décidé de créer une fonction qui permet de renvoyer, via une variable booléenne, un ‘1 si la parité est impaire pour qu’elle devienne paire, et un ‘0’, si celle-ci est déjà paire.



On va donc envoyer à la fonction, les datas donc les données des minutes ou de l’heure en BCD, ainsi que le nombre de bit (7 bits pour les minutes et 6 bits pour les heures).

Ensuite, la boucle ‘for’ va traiter les bits un par un.

La seconde opération, qui est un 'OU EXCLUSIF' va comparer le « bitParite » avec la valeur du premier, qui par la suite sera décaler de ‘1’ sur la droite.

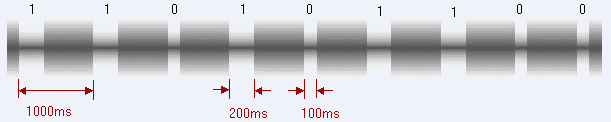
Exemple avec la valeur 0x52 : 0101 0010

|  |  |  |
| --- | --- | --- |
| **Variable ‘i’** | **Variable ‘bitParite’** | **Data (valeur BCD)** |
| 0 | 0 | 0010’1001 |
| 1 | 1 | 0001’0100 |
| 2 | 1 | 0000’1010 |
| 3 | 1 | 0000’0101 |
| 4 | 0 | 0000’0010 |
| 5 | 0 | 0000’0001 |
| 6 | 1 | 0000’0000 |

On peut donc voir que la valeur BCD de 0x52 à une parité impaire (3 bits) et que la valeur renvoyée et un ‘1’, ce qui permet d’obtenir une parité paire, comme souhaité.

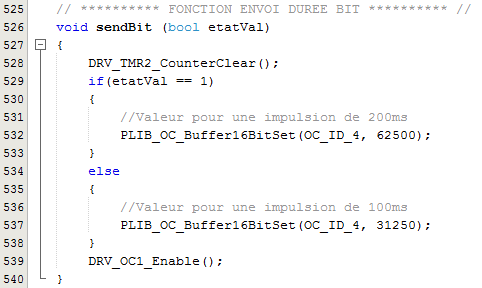
### Envoi des données

La durée d'une impulsion, détermine le niveau du bit reçu, à savoir qu'une impulsion de 100ms représente un bit à '0', et qu'une impulsion de 200ms représente un bit à '1'.



Pour ce faire, j’ai paramétré un Timer de 200ms, ainsi qu’OC, afin de gérer les durées des impulsions. De plus j’ai réalisé une fonction qui permet, selon l’état du bit, de définir la durée de l’impulsion.

**(Voir point 3.6.2.1.4 Timer 3 (page 50) & point 3.6.2.1.6 OC4 (page 51))**



Tout d’abord, on met le compteur du Timer 3 à ‘0’. Puis on regarde si la valeur à envoyer vaut ‘0’ ou ‘1’ et on fonction de celle-ci, on attribue à l’OC4 la valeur pour obtenir la durée de l’impulsion voulue.

## Travail restant à faire

* Usinage du boîtier (Boutons et micro USB)
* Dépanner la partie analogique (voir point 3.6.7.6)
* Implémenter la machine d'état, pour qu'elle soit appelée à la seconde
* Tester le code pour l'envoi des informations horaires

## Conclusion

Ce module m'a permis de réaliser un projet complet, et ainsi d'être confrontée aux difficultés que cela peut représenter.

L'un des premiers problèmes que j'ai rencontrés et la partie Ethernet qui n'est pas fonctionnelle. En effet, j'ai passé beaucoup de temps à essayer de trouver d'où venait le dysfonctionnement, mais sans succès. En sachant que c'est une partie qui est reprise d'un projet fonctionnel (1630 Timbreuse RFID).

J'ai donc perdu un certain temps sur cette partie, et aurait préférer l'investir dans la partie analogique, afin de la rendre fonctionnelle.

Car en effet, lors du dimensionnement de celle-ci, je n'ai pas pensé à vérifier que l'amplificateur était capable de fonctionner, avec la fréquence que j'injecte dans le système, soit la porteuse à 77.5kHz. Il faut donc trouver un autre composant pour remplacer l'amplificateur U3, tout en portant une intention particulière au fait que celui-ci doit être capable de fonctionner à la fréquence voulue.

Le projet n'est donc pas totalement fonctionnel, étant donné que l'envoi des informations horaires n'a pas pu être testé. Mais malgré ça, la majeure partie du programme a été réalisée, et il ne reste que peu de code à implémenter, il reste essentiellement des tests à réaliser, dès que la partie analogique sera totalement opérationnelle.

Néanmoins, ce projet m'a aidé à être encore plus autonome et à m'améliorer dans divers domaines de l'électronique comme le développement, le design, la programmation, etc...

Lausanne, le 20 juin 2019

Signature :

## Annexes

Annexe I : Cahier des charges

Annexe II : Schéma électrique

Annexe III : Journal de travail

Annexe IV : Planning

Annexe V : Liste de pièces

Annexe VI : Fichier des modifications

Annexe VII : Software (app.c ; app.h ; system\_interrupt.c ; Mc32DriverLcd.c ; bsp.h)

Annexe VIII : Normes euro-circuit

Annexe IX : Résumé du projet

Annexe X : Mode d'emploi

Annexe XI : Affiche

1. <http://www.newhavendisplay.com/specs/NHD-0420AZ-FSW-GBW-33V3.pdf> [↑](#footnote-ref-1)
2. <https://fr.wikipedia.org/wiki/DCF77> [↑](#footnote-ref-2)