Rapport de Projet

**Ecole supérieure**

Électronique

Salle R110

Projet portier

**Réalisé par :**

Ismaël Page

**A l’attention de :**

Serge Castoldi

Lionel Yersin

**Dates :**

Début du projet : 14 novembre 2017

Fin du projet : 21 juin 2018

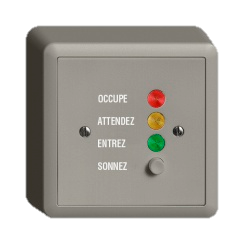


Table des matières

[1. Phase de pré-étude 4](#_Toc517356856)

[1.1. Introduction 4](#_Toc517356857)

[1.2. Cahier des charges 4](#_Toc517356858)

[1.2.1. Description du projet 4](#_Toc517356859)

[1.2.2 Schéma général du système 4](#_Toc517356860)

[1.3. Choix des composants 5](#_Toc517356861)

[1.4. Schéma bloc 7](#_Toc517356862)

[1.4.1. Partie out 7](#_Toc517356863)

[1.4.2. Partie in 7](#_Toc517356864)

[1.5. Système d'auto-alimentation circuit out 8](#_Toc517356865)

[1.6. Évaluation des coûts 8](#_Toc517356866)

[1.7. Planning 8](#_Toc517356867)

[1.8. Conclusion et perspectives 9](#_Toc517356868)

[2. Phase de design 10](#_Toc517356869)

[2.1. Logiciel 11](#_Toc517356870)

[2.1.1. Entrées / sorties 11](#_Toc517356871)

[2.1.2. Montage OUT 11](#_Toc517356872)

[2.1.3. Montage IN 11](#_Toc517356873)

[2.1.4. UART 12](#_Toc517356874)

[2.2. Schéma 12](#_Toc517356875)

[2.2.1. Power 12](#_Toc517356876)

[2.2.1.1. Auto-alimentation 12](#_Toc517356877)

[2.2.1.2. Mesure pile 13](#_Toc517356878)

[2.2.1.3. Alimentation µUSB 14](#_Toc517356879)

[2.2.2. µController 15](#_Toc517356880)

[2.2.2.1. PIC32MX130F064B-I/SS 15](#_Toc517356881)

[2.2.2.2. I/O 16](#_Toc517356882)

[2.2.2.3. Module 868MHz 17](#_Toc517356883)

[3. Phase de réalisation 18](#_Toc517356884)

[3.1. Hardware 18](#_Toc517356885)

[3.1.1. Consommation 20](#_Toc517356886)

[3.1.2. Problème du système d'auto alimentation 20](#_Toc517356887)

[3.1.3. Plaque frontale 20](#_Toc517356888)

[3.2. Software 21](#_Toc517356889)

[3.2.1. Fréquence Clock 21](#_Toc517356890)

[3.2.2. Timers 21](#_Toc517356891)

[3.2.3. UART 22](#_Toc517356892)

[3.2.4. Buzzer 23](#_Toc517356893)

[3.2.5. Module 868MHz 24](#_Toc517356894)

[3.2.6. Programme IN 25](#_Toc517356895)

[3.2.6.1. Vue générale du programme 25](#_Toc517356896)

[3.2.7. Programme OUT 26](#_Toc517356897)

[3.2.7.1. Vue générale du programme 26](#_Toc517356898)

[3.2.7.2. ADC 27](#_Toc517356899)

[3.2.7.3. Calcul valeur low 27](#_Toc517356900)

[3.3. Résultat 28](#_Toc517356901)

[4. Conclusion 29](#_Toc517356902)

[5. Annexes 30](#_Toc517356903)

# Phase de pré-étude

# 1.1. Introduction

Suite à notre stage, nous sommes de retour pour terminer notre 2ème et dernière année à l'ETML-ES à Lausanne. Nous avons tous reçu un travail à réaliser durant les cours de projet. Il se déroule en plusieurs phases que voici :

La pré-étude, qui est une analyse du projet. On prend connaissance du cahier des charges, les contraintes, on établit un schéma bloc avec les composants principaux. Une présentation de cette phase doit être effectuée le 12 décembre 2017.

La partie design et schéma, qui sont assez explicite de par leur nom. Une présentation doit également se faire le 30 janvier 2018.

Fin du projet, qui concerne toutes les phases qui suivent la schématique. C'est-à-dire routage, montage, software, test et mise au point. La présentation finale a lieu le 26 juin 2018.

# 1.2. Cahier des charges

# 1.2.1. Description du projet

Le but est de faire un portier wireless. Dans l'exemple de l'ETML-ES, l'étudiant appuie sur un bouton à la porte du bureau des professeurs et suite à la réponse faite par l'enseignant à l'intérieur, reçoit l'indication entrez, attendez ou occupé. Le circuit de réponse dans le bureau est alimenté via micro USB, avec un buzzer. Il doit avoir la possibilité d'en avoir plusieurs dans la pièce.

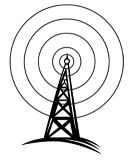
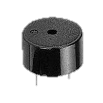
Le circuit qui se trouve à la porte fonctionne avec 2 piles AAA (Autonomie 2ans).

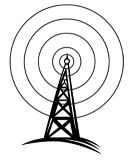
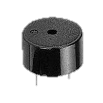
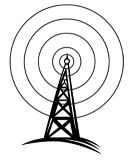
# 1.2.2 Schéma général du système

Module 868MHz projet 1623 Simonet

****

****

****

********

Module 868MHz projet 1623 Simonet

Module 868MHz projet 1623 Simonet

# 1.3. Choix des composants

Pour commencer, on nommera les deux circuits pour clarifier la suite de la documentation :

- Circuit dehors devant la porte : circuit **out**

- Circuit de réponse dans le bureau : circuit **in**

**Buzzer :**

On a décidé d'ajouter un buzzer sur le circuit out qui ferait un son lorsque la réponse a été faite ou que le timer s'est écoulé.

**Consommation du circuit out :**

Après avoir pris note des détails techniques concernant le module 868 MHz et le PIC32MX, on constate qu'ils peuvent être alimentés en 1V9 et 2V3. Les 3V fournis par les piles seront donc suffisant, à noter que chaque pile devra au minimum fournir 1V15.

On estime à deux utilisations par jour. Si le bureau est vide, l'appareil répondra "occupé" au bout de 30 secondes. Au pire des cas, nous avons 1 minute d'utilisation par jour.

Une LED sera allumé en permanence durant ce laps de temps : 20mA

Le PIC32MX, lui, sera en attente d'une réponse. Pire consommation estimée : 30mA

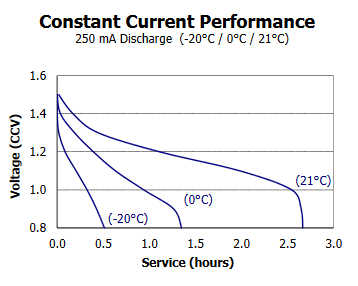
Il reste le module 868MHz (qui a également un PIC). Consommation selon datasheet : 34mA

La consommation du buzzer est négligeable car elle sera très courte.

Consommation totale : 84 mA

Chaque pile doit fournir au minimum 1V15 pour suffire au PIC32, grâce à l'entrée analogique, on va pouvoir mesurer la tension des piles et avertir l'utilisateur qu'il faut changer.

Datasheet pile Alkaline Energizer



On distingue que nous sommes à 1.2 heures de temps pour une décharge à 250 mA. Étant donné que nous avons 84 mA, on peut estimer ce temps 3 fois plus grand, soit 3.6 heures.

Courbe à 21°C

On a donc 1 minute par jour à 84mA de consommation, donc il faut 60 jours pour écouler 1 heure.

Si on compte maintenant un seul appui par jour, on arrive à 432 jours.

**Module 868MHz :**

Pour utiliser le module 868MHz, il doit être alimenté, commandé en UART et on doit y connecter un request, un statut et le reset. Il travaille à 115'200 baud.

Contrôle des piles :

Grâce aux comparateurs internes au PIC, on va pouvoir contrôler si les piles sont encore assez chargées ou non. Si c'est le cas, on donne l'info à circuit in qui activera la LED "Low Battery".

**Circuit in :**

Le circuit in va devoir produire l'alimentation 3V3 depuis son 5V. On va utiliser un MAX1793. Il a uniquement besoin de deux condensateurs (définis dans le datasheet) pour fonctionner.

Concernant les poussoirs, nous mettrons un pull-up avec une connexion à la masse lors d'une pression. Pour les LED et les buzzers, on inverse le principe : on commande la mise à la masse avec un MOS.

Explication en schéma :

PIC32MX

SW

RSW

+3V3

RD

+3V3

D

IRLML2402

Poussoir

LED



Chaque LED consomme 20mA sous 2V.

On a décidé d'inclure trois LED pour indiquer les cas suivants :

* Low battery : Indique que les piles du circuit out doivent être changées.
* Fail transmission : Problème de communication entre les modules.
* Request : Indique que quelqu'un a sonné et qu'il faut répondre.

# 1.4. Schéma bloc

On va faire 2 schémas bloc, un pour chaque circuit.

# 1.4.1. Partie out

Buzzer

Switch

3 LED

PIC32MX575F512H

Alim via 2 piles AAA (2x 1.5V)

Module 868MHz

UART

GPIO

GPIO

Alim.

Signaux

4

2

# 1.4.2. Partie in

Buzzer

3 Switchs

3 LED

PIC32MX575F512H

Alimentation 5V via micro USB

Module 868MHz

UART

GPIO

Convertisseur 3V3

Alim.

Signaux

6

2

# 1.5. Système d'auto-alimentation circuit out

PIC32MX

R

SW

BAS16

C

FDN306P

+1V5

+1V5

+VCC

GND



Explications :

Lorsqu'on presse sur le switch, on alimente le PIC en ouvrant le MOSFET canal P. Ensuite, le µC va prendre le relais pour "s'autoalimenter". Le condensateur est là pour garder l'alimentation active le temps que le PIC se réveil.

# 1.6. Évaluation des coûts

L'évaluation des coûts est calculée pour les 3 PCB ainsi que les 3 modules. En partant du principe que l'on part à zéro (aucuns composants en stock).

Prix des composants : 63.74 CHF

Prix des PCB : 50.00 CHF 🡪 3 pièces : 150.00 CHF

Prix du module 868MHz : 55.00 CHF 🡪 3 pièces : 165.00 CHF

Total : **378.74 CHF**

# 1.7. Planning

Voir en annexe

# 1.8. Conclusion et perspectives

Le projet dans son ensemble est prêt à démarrer. Grâce à ces quelques jours que l'on a pris pour organiser le projet, on gagne énormément de temps dans les prochaines phases car il n'y a presque plus besoin de réfléchir. Tous les choix sont définis.

Pour la suite, le schéma va prendre le plus de temps à mon avis. Je prévois de faire une librairie détaillée pour avoir une BOM directement prête depuis Altium. Je prendrai, dans la mesure du possible, mes composants d'Altium chez Würth.

# Phase de design

Pour commencer, voici les différents changements qui sont survenus durant cette phase.

Le circuit **IN** et **OUT** étant très semblable, il est plus simple de faire un seul type de PCB. On montera ensuite les composants selon le choix de la carte.

Exemple :

**Piles et mesure tension**

**uUSB & régulateur**

Module RF

PIC32MX130F

Buzzer

3 LED

**3 (0)**

**Switchs**

***Montage IN (intérieur du bureau) :***

- µUSB & régulateur 3V3

- 3 Switchs

***Montage OUT (extérieur du bureau) :***

- Piles et mesure de tension

- Aucun switch

Le reste est identique dans les deux cas.

Vous trouverez en annexe le pinout du PIC pour les deux situations. Il n'y a qu'un seul GPIO qui est inutilisé. Il sera branché à un point de test pour d'éventuels modifications ou tests lors de la partie programmation.

# Logiciel

# Entrées / sorties

|  |  |  |  |
| --- | --- | --- | --- |
| IN | | OUT | |
| **Entrées** | **Sorties** | **Entrées** | **Sorties** |
| Switch1 | LEDRED | Switch1 | LEDRED |
| Switch2 | LEDRED | Tension piles | LEDYLW |
| Switch3 | LEDRED | / | LEDGRN |
| / | Buzzer | / | Buzzer |
| RS232 | RS232 | RS232 | RS232 |

N'est pas compté comme entrée le reset du PIC

# Montage OUT

Pour que le système s'autoalimente, il faut que le démarrage du PIC soit le plus rapide possible. Selon le datasheet, il est plus avantageux d'utiliser l'oscillateur interne. Une fois l'alimentation assurée, on pourra switch sur l'oscillateur externe.

L'entrée du switch permet d'alimenter le système en entier. Ensuite le PIC effectue la requête tout en activant la LED rouge "occupé".

On utilise un timer pour commencer l'attente de la réponse. Au bout de 30 secondes sans réponse, le buzzer émet un son et la LED rouge clignote trois fois. Si on appuie à nouveau sur le bouton, le timer se reset à 0.

Si on obtient une réponse en retour, le buzzer va émettre un son la LED correspondante (jaune = "Attendez", vert = "Entrez").

Lorsque les piles atteignent un niveau trop bas, l'information sera transmise au circuit IN.

# Montage IN

Le système à l'état de repos sera totalement éteint (non alimenté).

Lors d'une requête, le buzzer avertira la personne à l'intérieur ainsi qu'une LED "Resquest" qui restera active tant qu'aucune réponse ne sera rendue. Il retournera en état de repos lorsque le circuit OUT lui indiquera que le délai est dépassé.

Si l'information des piles plate est communiquée, la LED "low batt" s'enclenchera et un signal sonore sera émis.

Pour finir, en cas d'échec de communication des modules RF, la LED "error" s'allumera jusqu'à réparation de l'erreur.

# UART

Baudrate : 57'600 baud

Pour économiser de la communication, le signal de requête sera en réalité la tension qu'il reste dans les piles. De cette façon, la gestion de l'alarme se fera dans le PIC du circuit IN.

Une initialisation est requise pour le module 868MHz. Il faudra le préparer à avoir plusieurs circuits IN.

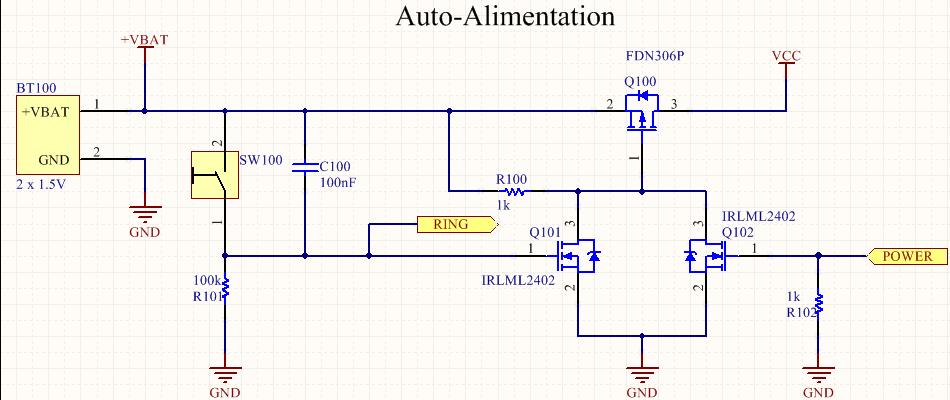
# Schéma

Voir en annexe.

# Power

Pour pouvoir s'y retrouver plus aisément, les composants qui concernent l'alimentation ont leur annotation qui commence à 100.

# Auto-alimentation



***C100 & R101***

Le système d'auto-alimentation a légèrement évolué depuis la phase de pré-étude. Un RC est mis en place pour augmenter le temps activé lors d'une pression sur la sonnette.

Le calcul a été fait pour que tau soit 10x plus long que le temps qu'il faut au reset pour se mettre à 1.

RCRST : τ = 1 ms

RCAA : τ = 10 ms

***Signal RING***

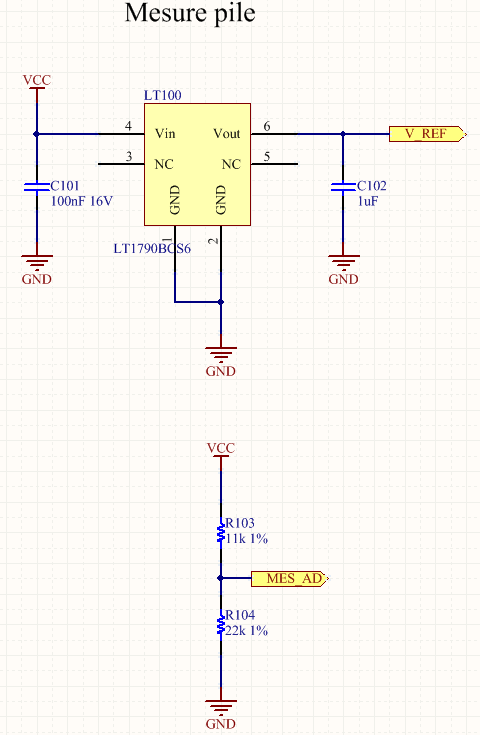
Le signal RING est utilisé pour savoir si l'utilisateur appuie à nouveau sur la sonnette. Dans ce cas le timer va se réinitialiser.

***IRLML2402***

Ce sont les MOSFET à canal N qui vont servir à ouvrir ou non le FDN306P qui lui est le dernier élément qui retient l'alimentation au reste du montage. Le signal POWER viendra prendre le relais à l'activation du PIC pour s'alimenter le temps de la séquence.

# Mesure pile

***LT1790***

C'est une référence de tension qui va fournir une tension de 2.048V au PIC. Il va fonctionner jusqu'à Vout + 0.2V donc jusqu'à l'alarme des piles plates. Les condensateurs sont définis sur le datasheet. La tension de sortie servira de référence pour comparer à la mesure des piles. Cette mesure sera faite à travers un diviseur de tension duquel on ressort le ⅔ de la tension.

Valeur du pont :

R103 = 11k

R104 = 22k

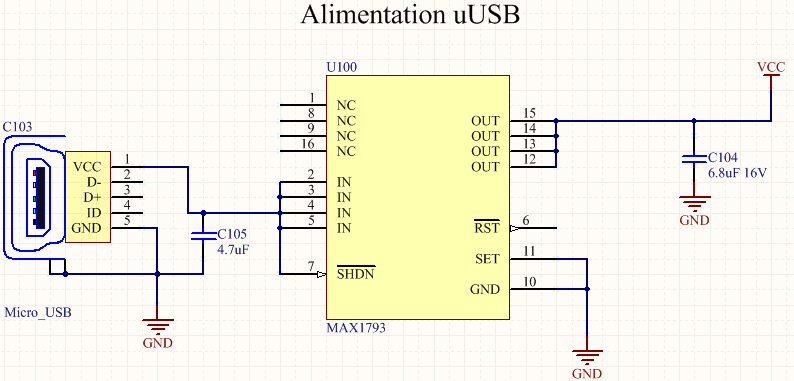
Rappel :

UPILEmin = 2.3V

On cherche maintenant à quelle valeur on donnera l'alerte.

A noter que pour assurer la communication, on prendra une valeur légèrement plus élevée, soit 1.6V.

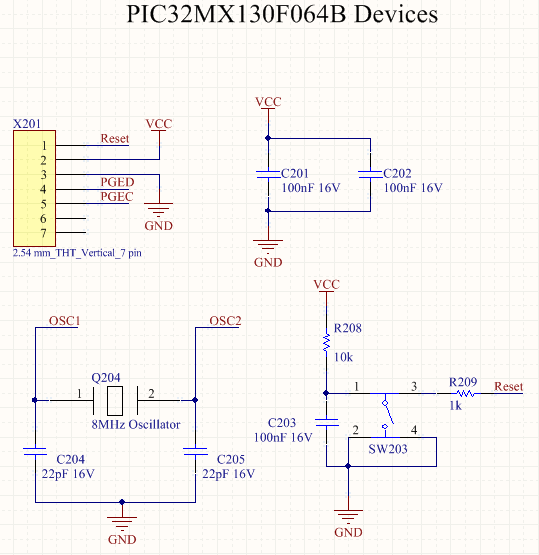
# Alimentation µUSB



***MAX1793***

Permet de convertir les 5V de l'USB en 3V3. Le schéma vient directement du datasheet. A noter que l'on connecte uniquement l'alimentation du µUSB.

# µController

Maintenant, la deuxième partie du schéma qui concerne les I/O, et les signaux de communication.

# PIC32MX130F064B-I/SS

Pour mieux se repérer, l'annotation commence à 200.

***RJ12***

Un connecteur RJ12(J200) pour la programmation du PIC. On y connecte l'alimentation, le reset ainsi que PGED(21) et PGEC(22).

***Oscillateur***

Un oscillateur 8MHz (Q204) dont les condensateurs ont étés définis dans le datasheet.

***Reset***

On trouve un schéma de principe dans le datasheet du PIC. On a un RC qui va définir en combien de temps le reset va se mettre à 1.

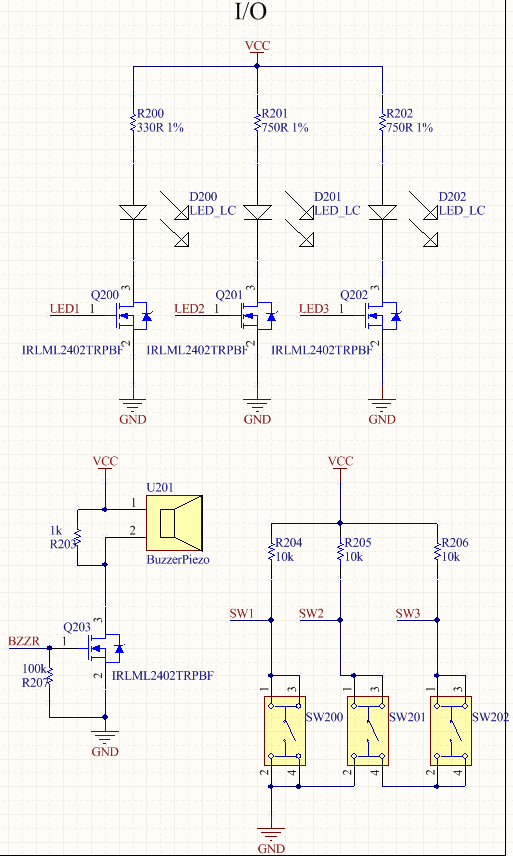
Calcul :

On peut grossièrement dire qu'après 3 tau, le PIC commencera son fonctionnement. C'est-à-dire 3 ms.

***Condensateurs de découplage***

On a un condensateur pour chaque alimentation du PIC.

# I/O

***LED***

Consommation : 2 mA

Tension LED rouge : 1V75

Calcul de la résistance :

Tension LED jaune : 1V85

Calcul de la résistance :

Tension LED vert : 2V65

Calcul de la résistance :

Avec ce montage, elles seront actives à 1.

***Buzzer***

Une résistance 100k est placée pour éviter les sons indésirables lors du démarrage. R203 vient du datasheet.

***Switch***

Ce sont de simples switchs avec des pull-up. Ils seront donc actifs à 0.

# Module 868MHz

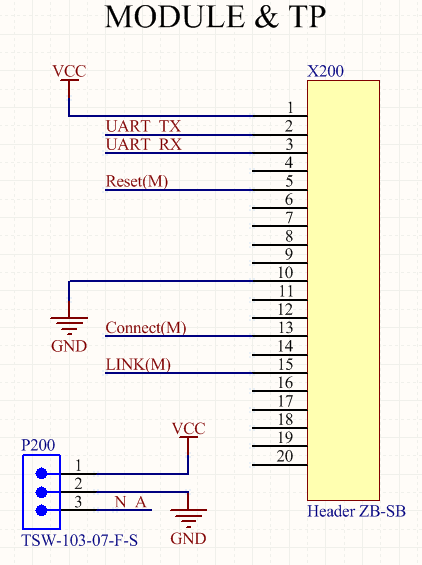
***Connecteur Xbee***

On a besoin de 5 signaux ainsi que l'alimentation. Les signaux qui sont assignés au module sont accompagnés d'un (M).

|  |  |
| --- | --- |
| Signaux | Connecteur Xbee |
| +3V3 | 1 |
| GND | 10 |
| UART\_TX | 2 |
| UART\_RX | 3 |
| Reset(M) | 5 |
| Connect(M) | 13 |
| LINK(M) | 15 |

***Barrettes à broche***

Simple points de mesure avec l'alimentation ainsi que le GPIO non utilisé du PIC(12).



# Phase de réalisation

# Hardware

Pour la réalisation de mon PCB, 2 couches suffiront amplement ! Le cahier des charges ne comporte pas de restriction concernant la taille du circuit. Une taille de 10cm x 10cm conviendra à son utilisation.

Les Design Rules ont été prises chez Eurocircuit.

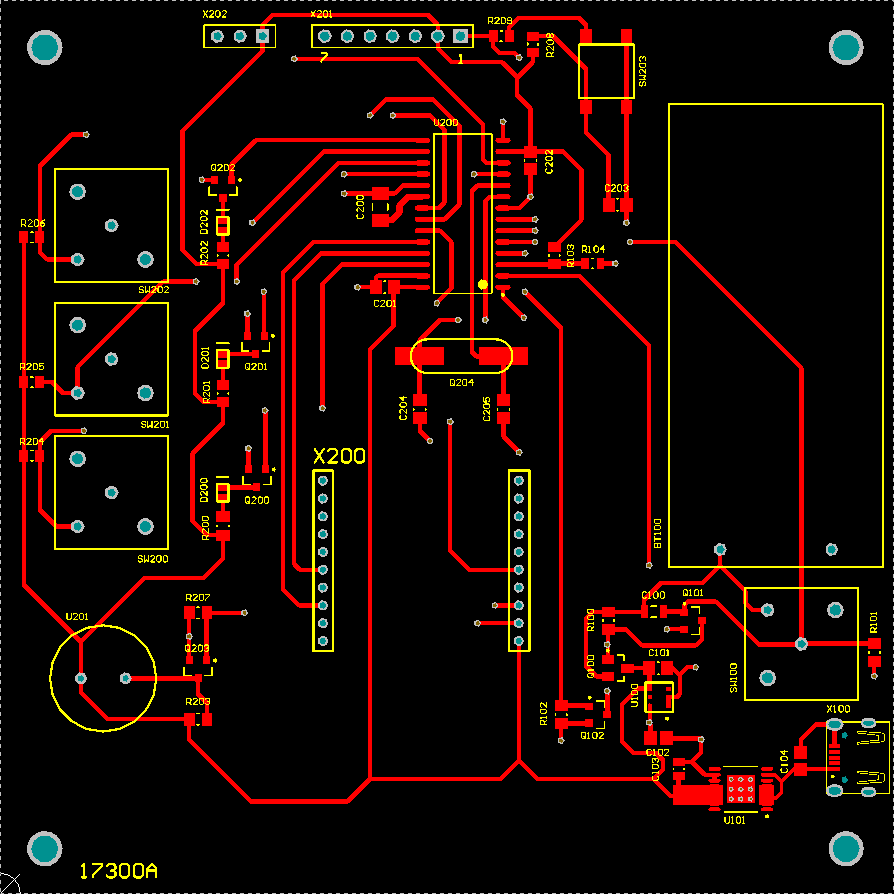
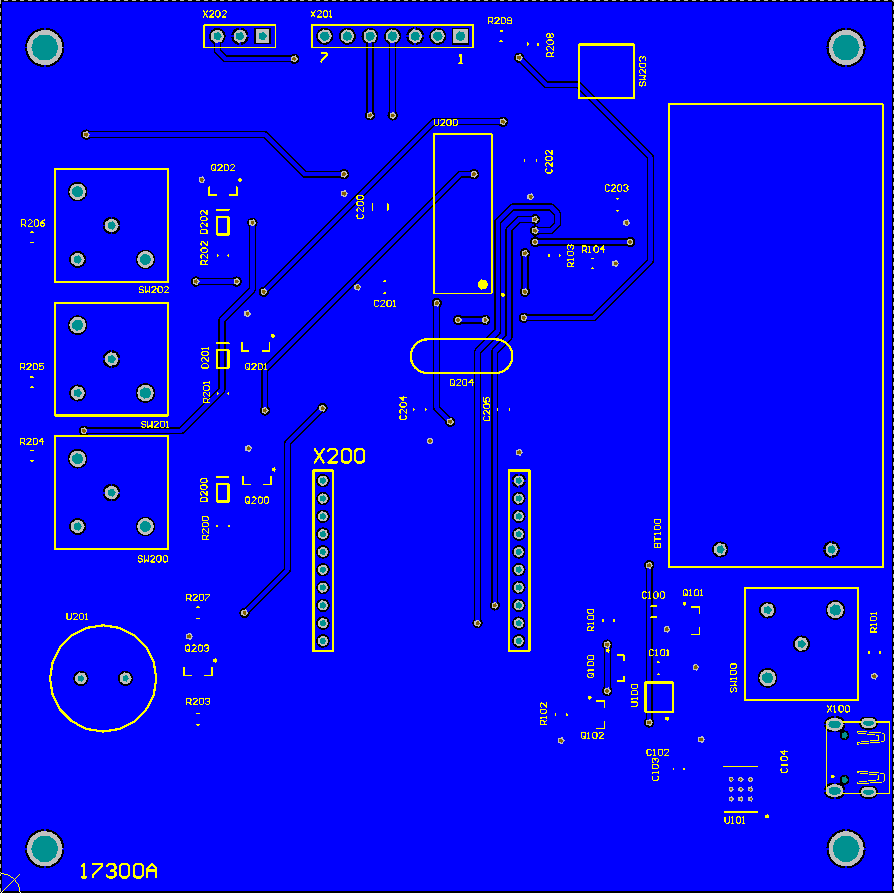
***Top Layer :***

Tous les composants sont sur cette face. Il n'y a pas de plan sur ce côté.

***Bottom Layer :***

J'ai mis un plan de masse sur cette couche.

- Top Layer - - Bottom Layer -

**Note :** La version que j'ai commandée contenait des erreurs au niveau des connecteurs ZB-SB. Le Rx et Tx étaient inversés et le pinout du connecteur était faux. Cependant, on peut connecter le module en mettant l'antenne contre le bas (elle dépasse) étant donné que j'ai fait la correction du Rx/Tx.

Ceci est maintenant corrigé sur Altium. Comme j'ai été un peu vite dans ma correction, je n'ai pas gardé le schéma exact de la carte actuelle. Cependant la seule modification est l'inversion de Rx-Tx ainsi que le léger déplacement du connecteur ZB-SB.

La place pour un oscillateur externe est prévue, mais il n'est pas prévu de le monter.

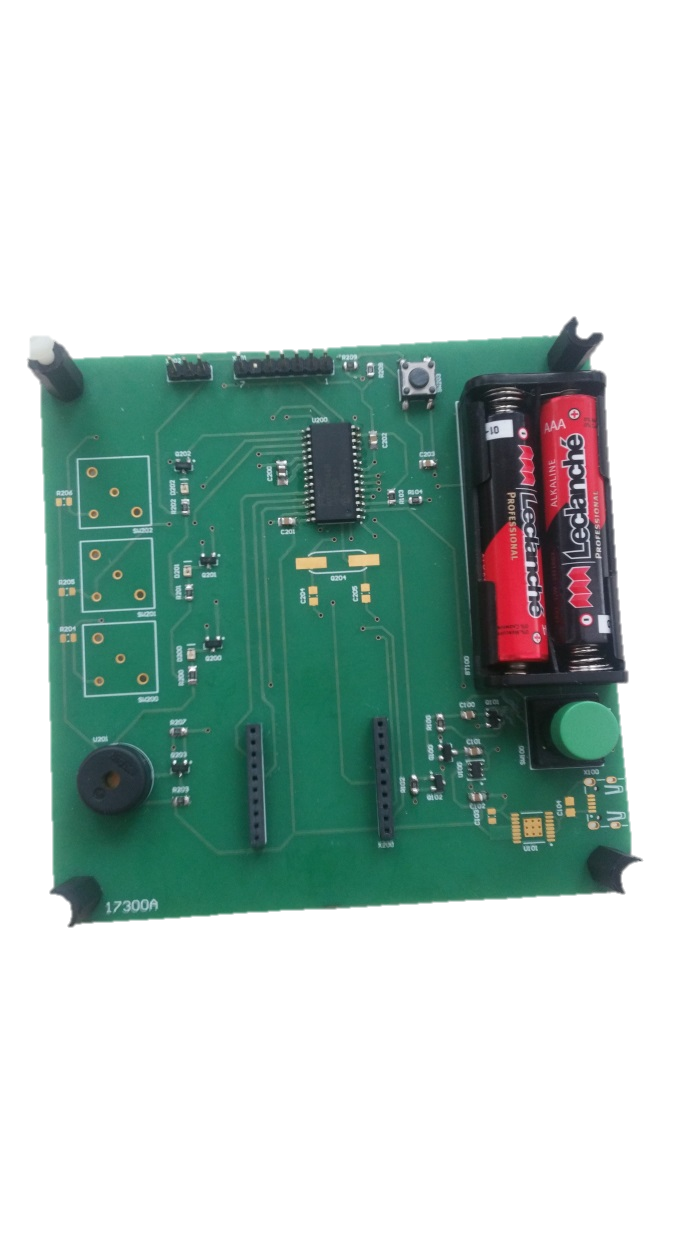
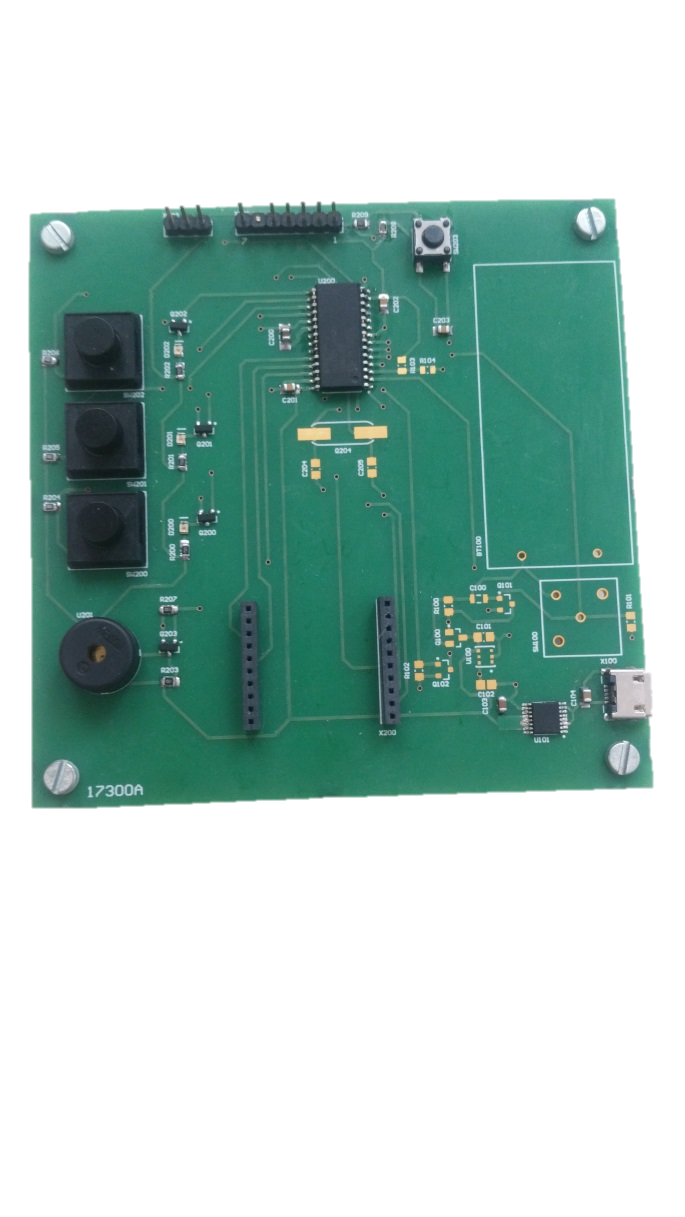
***Circuits montés :***

+3V3

Connecteur de programmation

RA4

GND

- Circuit OUT - - Circuit IN -

= Sonnette

"occupé"

Requête

LED

btn

LED "Occupé"

"attendez"

Erreur communication

LED

btn

LED "Attendez"

"entrez"

Batterie faible

LED

btn

LED "Entrez"

Les montages ont été testés au niveau hardware et tout fonctionne parfaitement. Deux erreurs à notifier, le connecteur pour le module 868MHz(Rx/Tx) et l'auto alimentation.

La pin RA4 sera utilisée pour faire diverses mesures.

# Consommation

La consommation de mon circuit OUT (piles) est d'environ **81 mA**. Elle augmente à 91 mA lorsque le buzzer fonctionne. On est donc légèrement plus élevé que l'estimation faite durant la pré-étude.

Ces mesures ont été faites depuis l'alimentation externe sous 3V.

***Courant :***

Imoy : 81 mA

Imax : 91 mA

***Puissance :***

P = U x I

Pmoy : 243 mW

Pmax : 273 mW

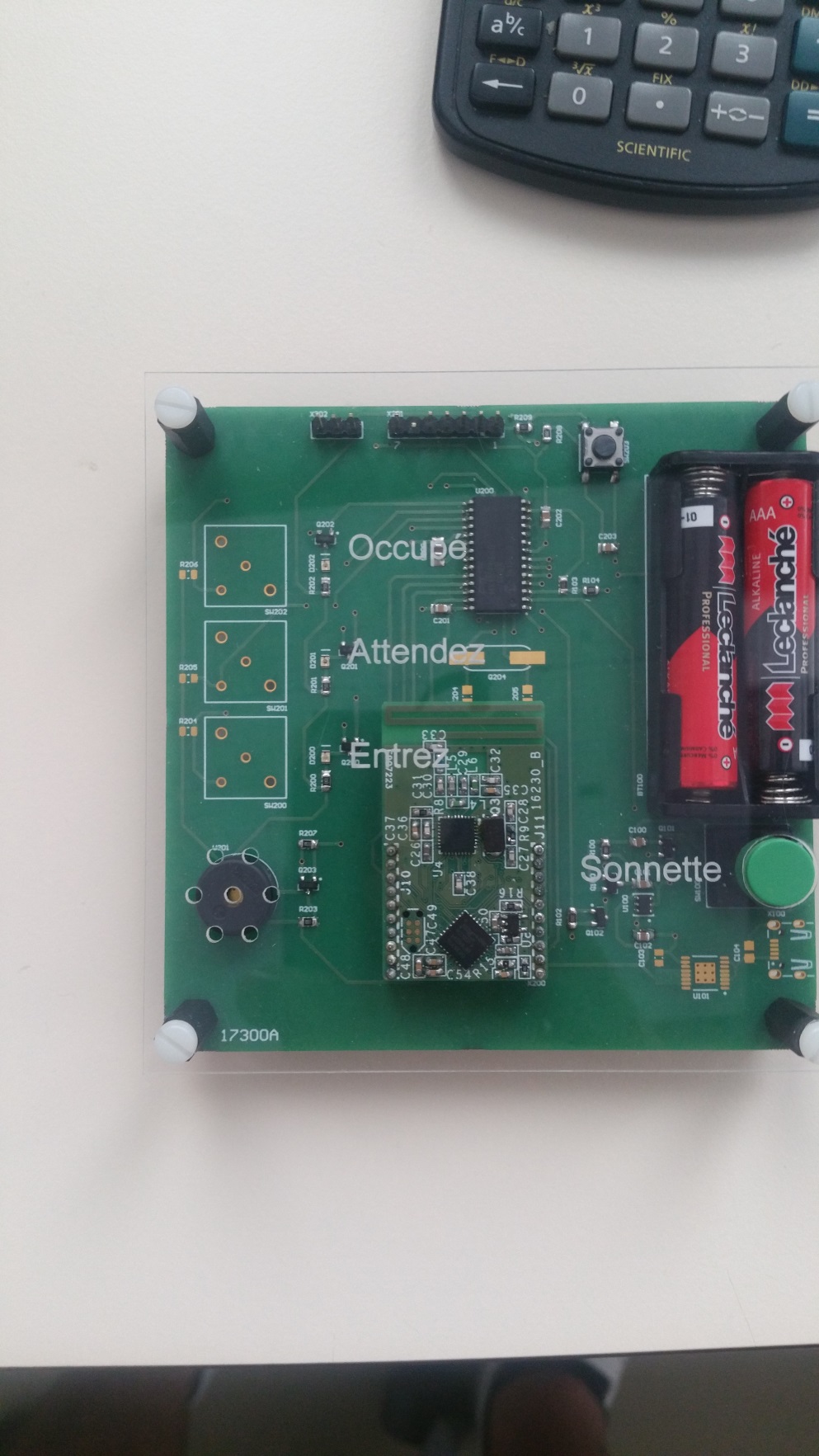
# Problème du système d'auto alimentation

Apparemment, mon circuit refuse de démarrer en dessous de 2,97V. C'est erreur vient du système d'auto alimentation qui empêche uniquement le démarrage du système. Cependant s'il est déjà démarré, on peut descendre comme prévu jusqu'à 2.3V. C'est d'ailleurs pour cela que j'ai remarqué l'erreur si tard.

# Plaque frontale

J'ai décidé de les faire en plexiglas que ça soit esthétique.

Voici un résultat :



Pour les colonnettes, j'ai dû faire par rapport au composant le plus haut, qui est le support à piles. J'en ai prises en plastique de 15mm de haut type femelle-femelle.

Pour élever le circuit, j'ai également mis des colonnettes, mais cette fois pas en plastique. Elles mesurent 12mm et sont femelle-mâle.

J'ai aussi dû faire attention à bien avoir des poussoirs plus haut que les colonnettes avec l'épaisseur du plexiglas. On arrive donc à une taille de 17mm.

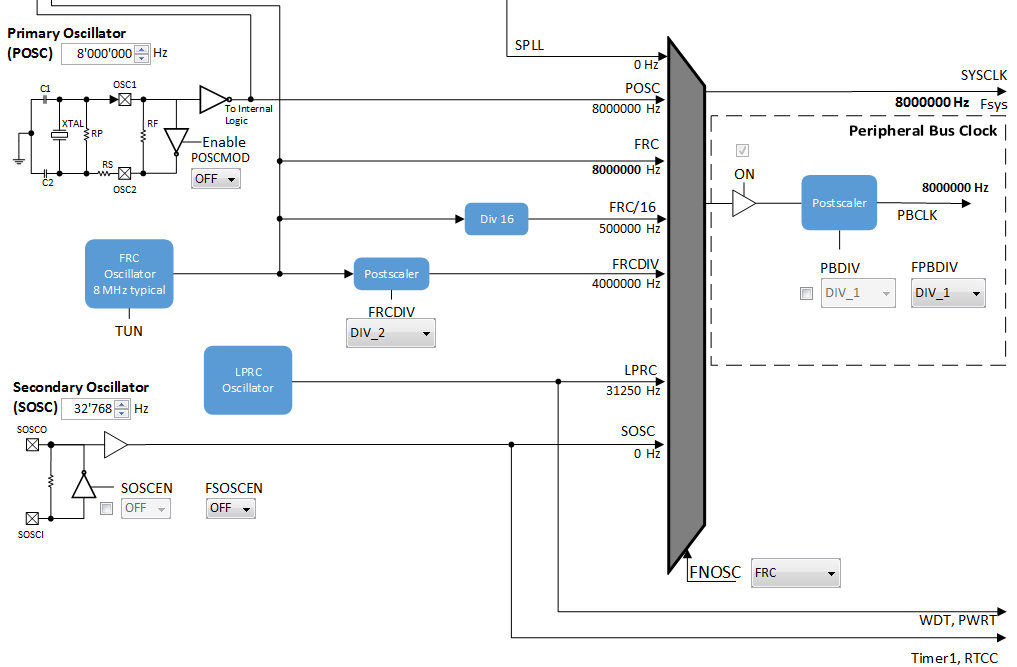
Pour finir, des vis plastique pour faire tenir le tout.

# Software

Pour cette partie, nous avons donc 2 codes différents.

Cependant, la configuration de harmony est semblable pour 2 projets.

# Fréquence Clock



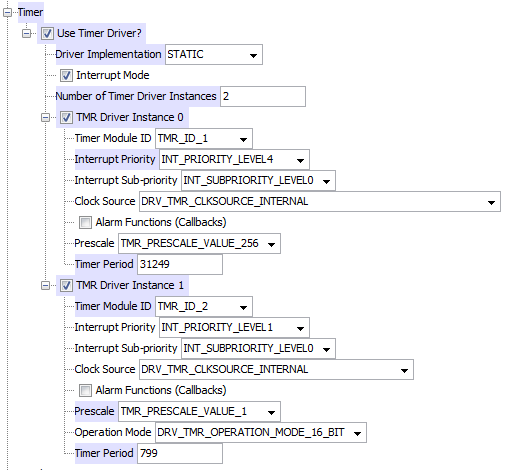
On utilise directement l'oscillateur interne du PIC. Pour cela, on sélectionne "FRC".

# Timers

J'utilise 2 timers, le premier va servir pour mesurer le temps qui passe. Il sera donc réglé à 1s.

Le deuxième sera utilisé pour le buzzer, il sera à une un temps de 100us. A l'intérieur, on fait compter une variable et nous l'utiliserons pour faire notre fréquence de son. Explications plus détaillées à la page 23.

La priorité sera sur la fréquence pour le son. L'utilisateur ne va pas se rendre compte si il a attendu 15.5 ou 14.5 secondes.

***Timer 1 :***

Interruption chaque seconde :

Période :

***Timer 2 :***

Interruption chaque 100us :

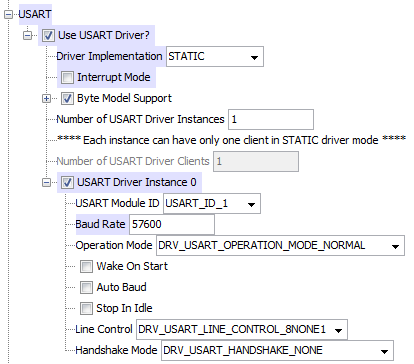
Période :

# UART

**Remarques** :

Baud rate : 57'600 (pour aller au module)

Sans interruptions (travaille en machine d'etat)

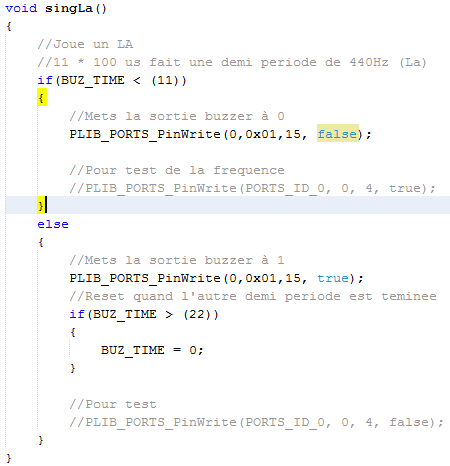
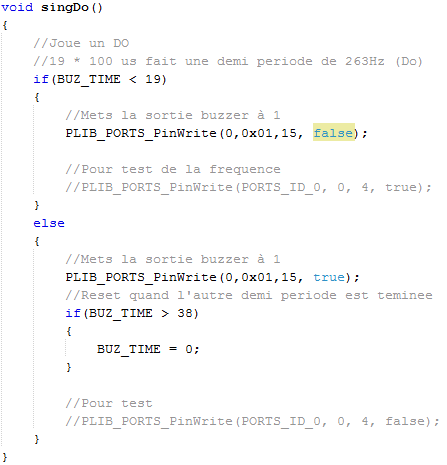


Sans contrôle de flux

# Buzzer

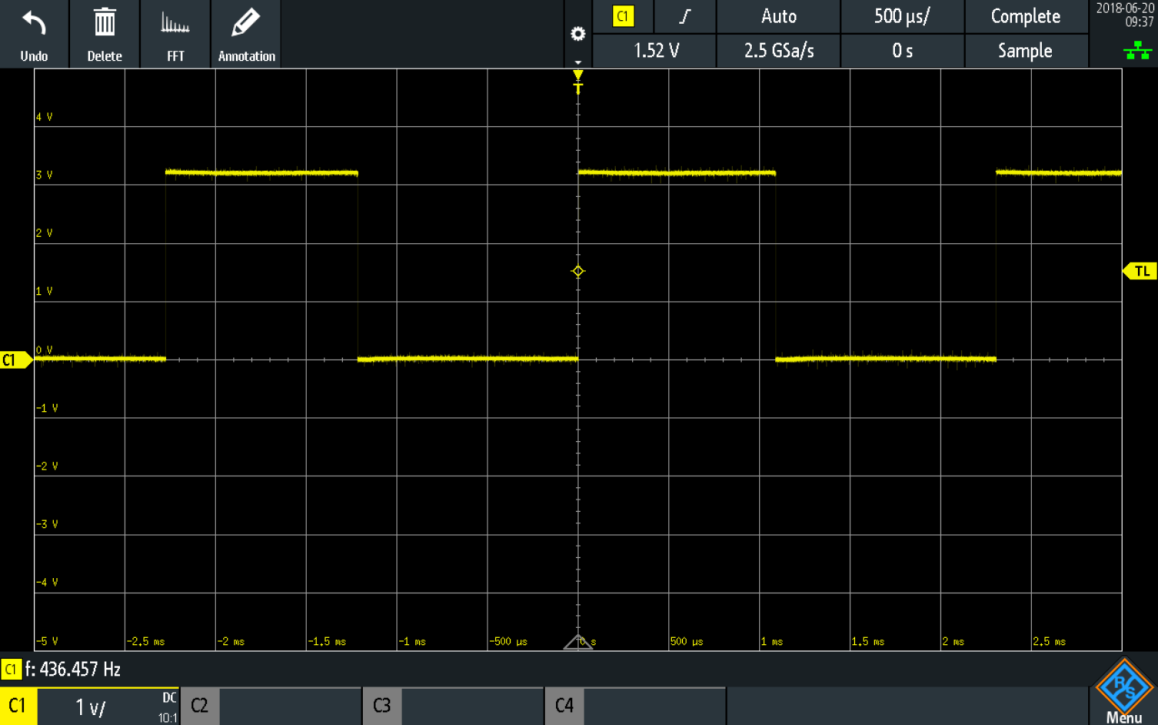
Le timer 2 incrémente BUZ\_TIME tous les 100 us. La fonction singLa() gérer la fréquence en fonction de la valeur BUZ\_TIME.

Le même principe est utilisé pour faire un Do. Lui sera servi dans les cas où la réponse est négative.

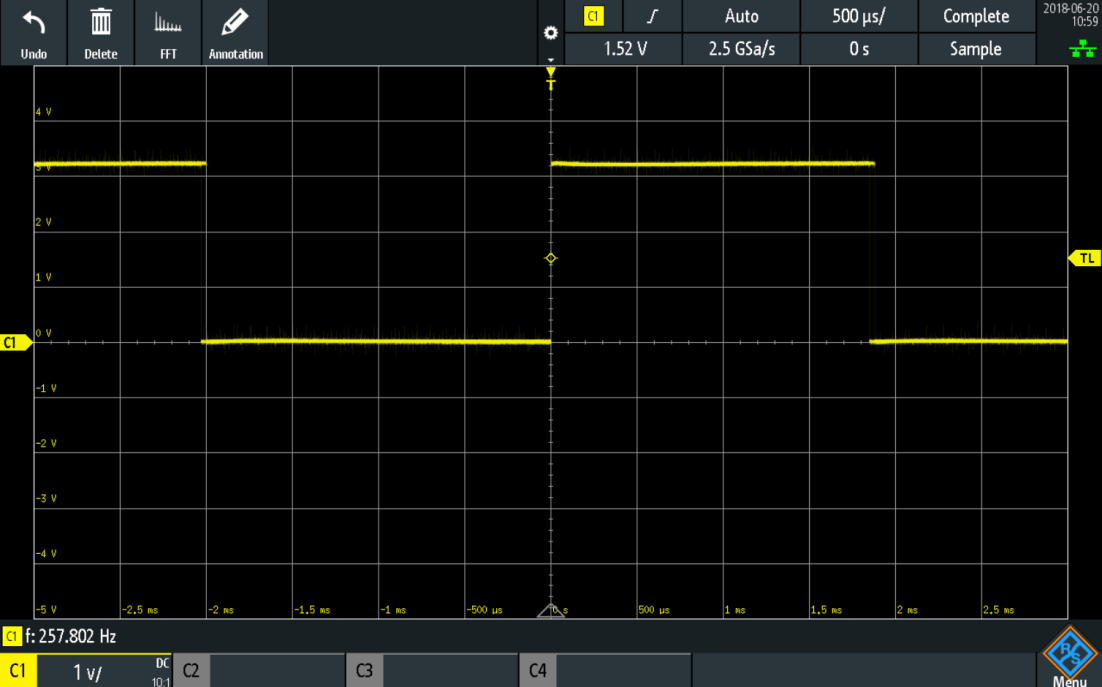
 

Pour tester, on fait osciller au même moment la pin de réserve de notre montage RA4 (voir p17).

- Mesure du LA sur RA4 -



- Mesure du DO sur RA4 -



**Remarque :**

L'erreur est négligeable pour l'utilisation dont nous avons besoin.

# Module 868MHz

Pour l'initialiser, il faut mettre la pin CONNECT à 1 puis attendre que le pin LINK passe à 1 également. Une fois cela fait, on peut communiquer en UART en envoyant la trame suivante :

Start

Com

Adresse

Data

**Start** : La valeur 0xAA est définie pour indiquer le début de la trame.

**Com** : Sert à indiquer le mode d'utilisation du module. Ici, on le met à 0x00 pour être en mode bidirectionnel.

**Adresse** : Indique l'adresse du module sur 4 bytes du MSB au LSB. Le contrôle est également fait à travers le module.

**Data** : Il doit être un caractère selon la plage ASCII, c'est-à-dire entre 0x41(A) et 7F(DEL). Si ce n'est pas le cas, les modules vont empêcher la transmission du caractère.

On a donc une trame de 7 bytes au total.

**Note** :

Les modules gèrent eux même le fait de faire une validation (ACK), il n'est donc pas nécessaire de le refaire une deuxième fois dans notre programme.

# Programme IN

# Vue générale du programme

On va diviser cette partie en plusieurs phases.

**Start**

**Initialisation**

**ASK**

**WAIT**

**WAIT ANSWER**

**Initialisation** :

On commence par une initialisation classique, les timers, les ports, les LED, etc...

**WAIT** :

Cette partie va attendre une valeur correcte du module 868MHz, c'est la phase de repos du système. Lorsqu'elle a reçu une information plausible, elle va passer en phase ASK.

Si on a une visite manquée enregistrée, on fait clignoter la LED "Request" jusqu'à la validation par une pression sur un poussoir.

**ASK** :

La parte ASK va indiquer à l'utilisateur que quelqu'un a sonné, en allumant "Request" et si l'information que les piles sont low, c'est à cet instant que "Low Bat" est allumé. C'est un état que l'on traverse une seul fois.

**WAIT\_ANSWER**

Lorsqu'un poussoir est pressé (ce qui indique une réponse "occupé", "attendez", "entrez"), on fait alors un envoi de cette réponse et l'on revient à WAIT.

Si on attend trop longtemps avant de répondre, on indique qu'une visite a été manquée, puis on repasse à WAIT.

Trouvez les détails sur le structogramme en annexe.

# Programme OUT

# Vue générale du programme

J'ai également fait une machine des états pour ce code.

**Start**

**Initialisation**

**WAIT**

**END**

**ANSWER**

**READ\_BAT**

**REQUEST**

**Initialisation** :

On fait la même que pour le IN, mais, on ajoute l'ADC ainsi que la pin d'auto alimentation.

**READ\_BAT** :

On lit l'AD et on regarde si les piles ont assez de tension.

**REQUEST** :

On envoie la requête qui se compose également de l'information sur les piles ('F' = Full et 'L' = Low)

**WAIT** :

On est maintenant dans la phase qui attend une réponse de montage IN. On lit si on reçoit une information correcte et on contrôle si on dépasse les 30 secondes.

Si on a une bonne réponse ('O' 'A' ou 'E'), on passe à ANSWER.

Si on dépasse les 30s, on sonne le buzzer avec un do, puis on passe à END.

On a également la possibilité de reset les 30s si on sonne une nouvelle fois.

**ANSWER** :

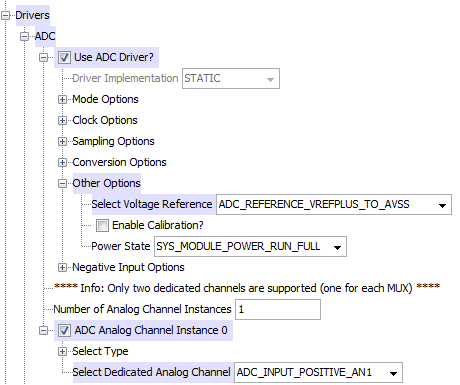
On allume la LED qui correspond à la réponse reçue, puis on fait sonner le buzzer(LA) pendant un bref instant. On peut ensuite avancer dans END

**END** :

Finalement on peut éteindre le circuit, c'est-à-dire, passer la pin d'auto alimentation à 0.

Le code est donc conçu pour être traversé une seule fois.

# ADC



**Remarque** :

Indiquer que l'on a une référence sur VREF+

Indiquer que l'on utilise l'entrée AN1.

# Calcul valeur low

Nous avons donc 1024 pas pour 2,048V. Nous pouvons donc faire une règle de 3 pour trouver à combien de pas nous seront à 1,6V.

Pas pour 1V : 1024 / 2.048 = 500

Pas pour 1,6V : 500 x 1.6 = 800 🡪 799

# Résultat

J'ai pu faire fonctionner mon circuit avec le module 868MHz, cependant il n'arrive pas toujours à se connecter. Il est très difficile de se d'avoir la connexion lorsqu'ils ne sont pas côte à côte.

Mon circuit OUT ne peut pas démarrer lorsque la tension se trouve en-dessous de 2.97V. Le problème vient de la partie auto alimentation, car si j'alimente de l'extérieur avec un 2V3, cela fonctionne.

Le reste de mon circuit fonctionne correctement avec une consommation dans la zone prévue, donc une autonomie probablement correcte.

# Conclusion

Ce projet n'a pas de zone particulièrement compliquée, si ce n'est la reprise du module 868MHz avec une documentation très peu claire. L'erreur du Rx/Tx m'a fait perdre beaucoup de temps. Sans cette perte, j'aurais certainement remarqué l'erreur de l'auto alimentation plus rapidement.

Le fonctionnement un peu aléatoire de la communication entre les cartes devrait pouvoir être réglée à travers le code. C'est le seul changement software à effectuer.

J'ai pu approfondir mes capacités dans le déroulement d'un projet, et j'ai pu noter l'importance d'avoir une bonne documentation.

Lieu, date : .................................................................

Signature : .................................................................

# Annexes

* A Schématique
* B Schéma d'implantation
* C BOM
* D Pinout PIC
* E Evaluation des prix
* F Planning
* G Journal de travail
* H Structogramme IN & OUT
* I.1 Listening IN
* I.2 Listening OUT
* I.3 Listening commun
* J Mode d'emploi
* K Résumé
* L Affiche