Ticketing

N° Projet 1811

Réalisé par: Martins David

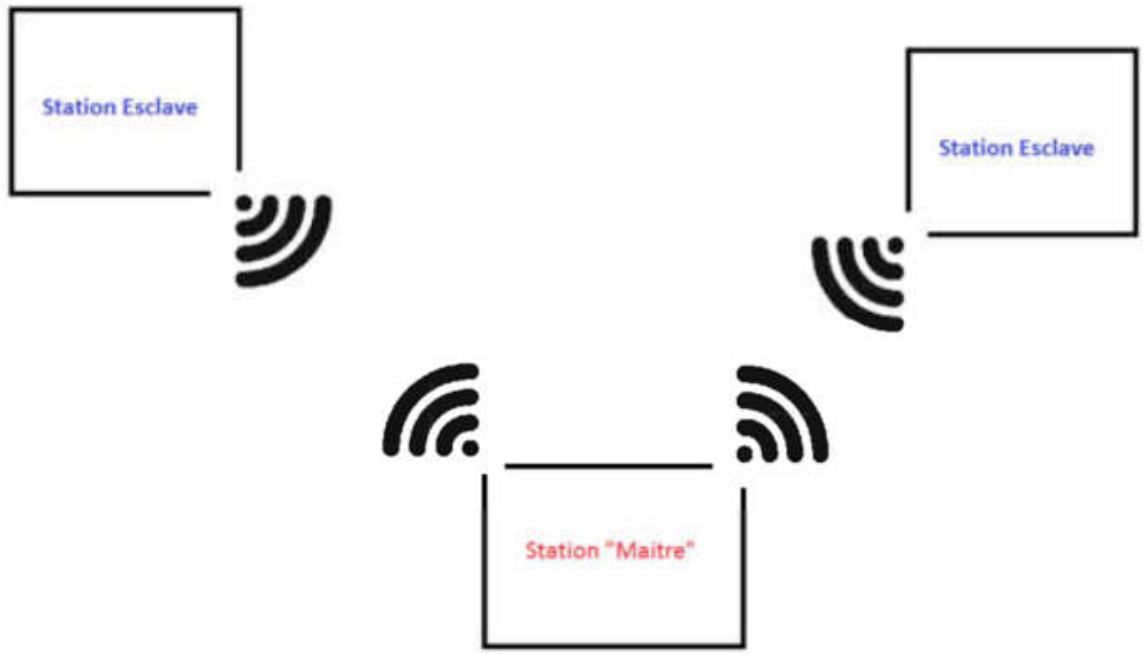
Table des matières :

1 Cahier des charges

1.1 Description du projet

Ce travail de diplôme fait suite à la demande des étudiants de première année SLO (2017/2018). Avoir un système de ticketing électronique qui permettra aux enseignants de mieux gérer les demandes des étudiants lors des cours ou travaux pratiques.

Le système de ticketing sera sans-fil et pour ce faire il faudra utiliser les modules RFs développé à l'ES (Projet ES 1623). Il faudra donc fournir au minimum 2 modules esclaves (pour les élèves) et 1 module maitre (pour les enseignants).



Tâches de l'esclave:

- Envoi de ticket.
- Affichage de l'état du module (Prêt à envoyer, attente, bloqué, non connecté).
- Annulation de ticket.

Tâches du maitre:

- Récupération des infos des étudiants dans la carte SD.
- Affichage de la liste des tickets en attentes.
- Accepter un ticket, le refuser, ou tous les resets.
- Via un menu, configurer la limite de questions par élèves.

1.2 Cahier des charges

- **Station "Maitre" comprenant:**

  - Un boitier
  - Un afficheur LCD minimum 4 lignes 20 colonnes
  - Boutons de commande (à impulsion ou rotatif) - doit permettre de sélectionner des menus ou valider des tickets
  - Alimentation 5V via µ-USB.
  - Emplacement pour module sans-fil -> voir projet 1623
  - Prévoir une carte SD (pour pré-configuration future, par exemple appairage des stations esclaves avec la station maitre)

  - **Firmware du microcontrôleur principal maitre**

    - Appairage des différents modules esclaves
    - Gestion des tickets (nombres par modules / validation des tickets)
    - Remise à zéro de tous les tickets

- **Station Esclave (minimum 2 à monter) comprenant:**

  - Un boitier
  - Boutons poussoir pour l'envoi du ticket
  - Système lumineux - 3 couleurs
    Rouge => bloqué, pas d'envoi possible (limite atteinte)
    => Clignotement: pas de lien RF
    Orange => envoi possible de ticket, ticket en cours.
    Vert => envoi possible de ticket, aucun en cours.
  - Alimentation 5V via via µ-USB.
  - Emplacement pour module sans-fil -> voir projet 1623

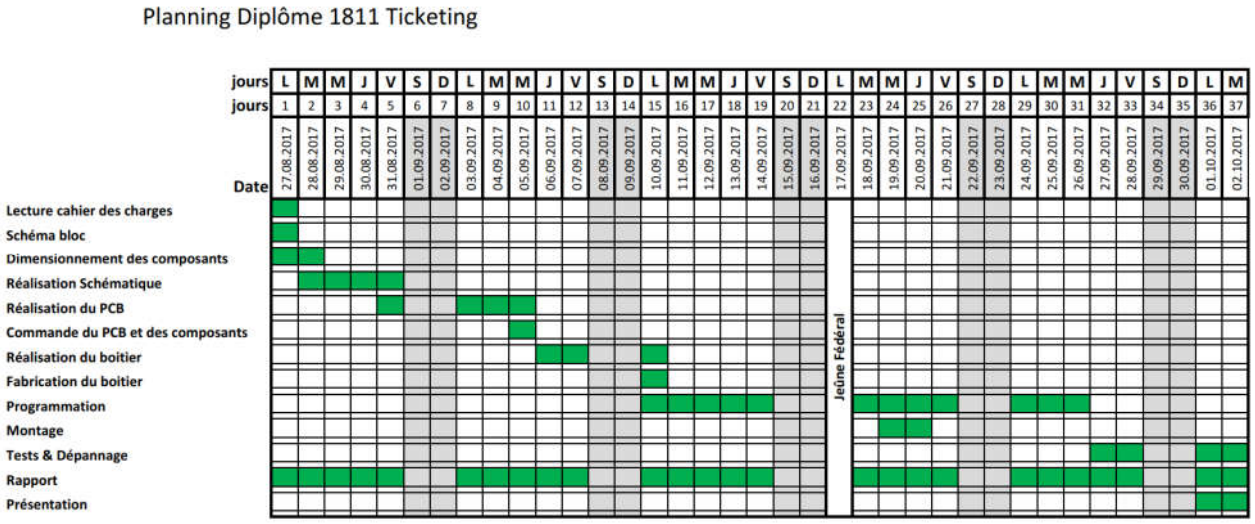
  - **Firmware du microcontrôleur principal esclave**

    - Envoi de ticket
    - Gestion des signaux lumineux

- **Firmware du module RF à mettre au point (HW fourni, FW existant Beugé)**

2 Pré-étude

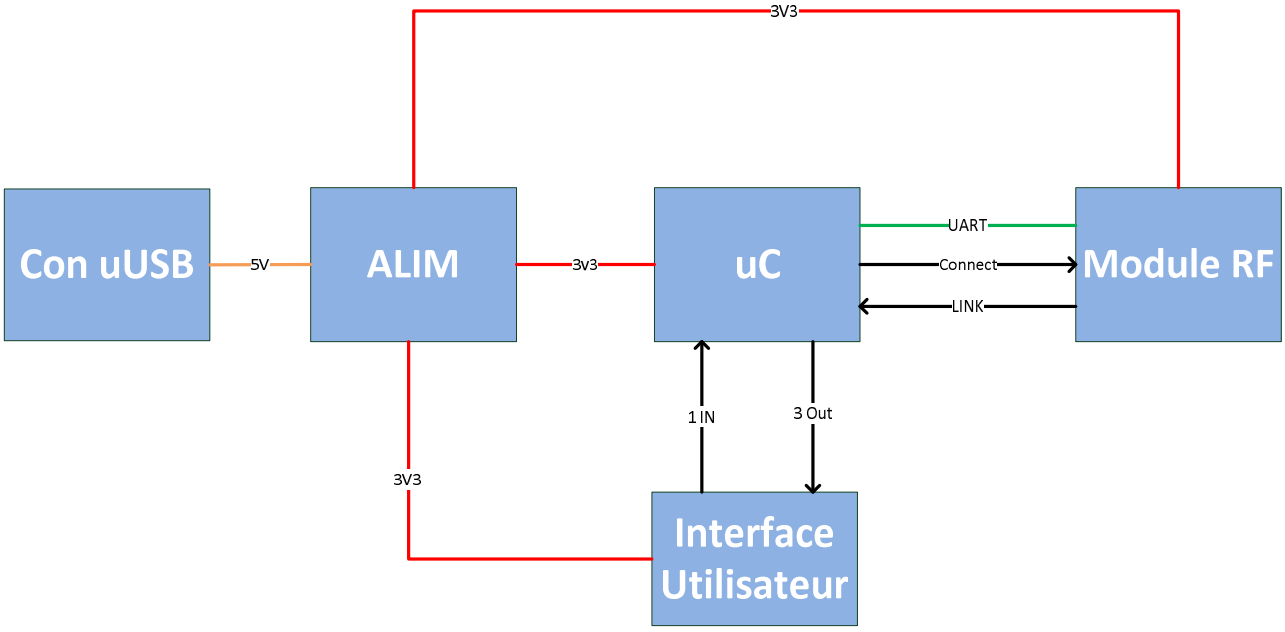
2.1 Planning



Remarque: en conclusion nous retrouverons ce planning, comparé au planning réel.

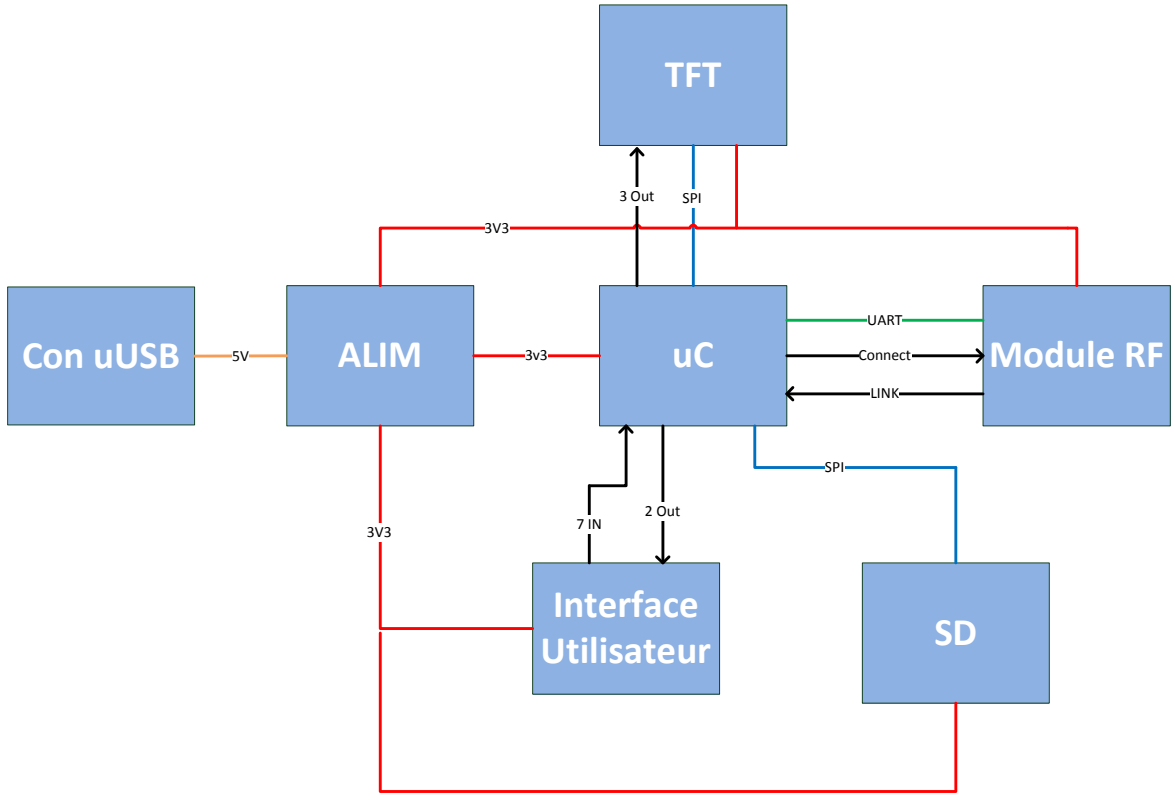
2.2 Schémas bloc

2.2.1 Schéma bloc module esclave



Ici nous pouvons voir que mon module esclave va nécessiter d'utiliser uniquement 1 Bus UART. Nous avons 1 entrée sur le microcontrôleur qui sera dédié au bouton d'envoi de ticket et nous aurons 3 sorties pour les 3 LEDs d'états.

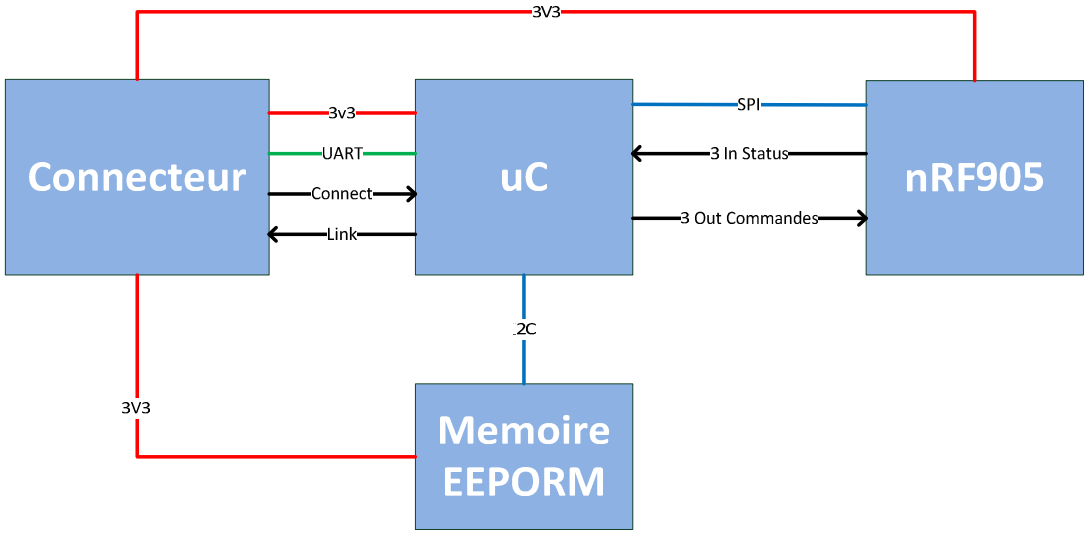
2.2.2 Schéma bloc module maitre



Pour le module maitre nous aurons besoin de :

2 bus SPI (SD et écran TFT)
1 bus UART
7 entrées pour les boutons de commandes
2 sorties pour le pilotage d'un buzzer et d'une LED

2.2.3 Schéma bloc Module RF



Remarque: je n'ai pas designé cette carte, cependant pour produire un code il faut en connaitre la construction.

Nous voyons donc que le module RF utilise un nRF905. Ce dernier est piloté via SPI et possède 3 pins de commandes ainsi que 3 pins d'indication de statuts.

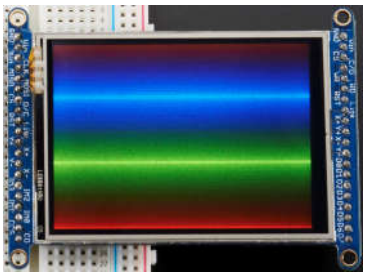
Une mémoire pilotée par I2C nous permet d'y stocker l'ID du module (son adresse RF).

Le contrôle du module se fera via UART.

2.3 Choix des composants

2.3.1 Module maitre

2.3.1.1 LCD



Décision portée sur un module TFT de chez adafruit:

- Alimentation 5V ou 3V3
- Communication SPI
- Dimension de 240x340 pixels pour 2.8" de diagonale
- Utilisé dans projet de semestre

2.3.1.2 Interface utilisateur

Décision d'utiliser un buzzer:

- Sonne à la réception d'un nouveau ticket
- Fréquence de 440Hz
- Couplé à un trimer permettant d'en régler le volume

LED Orange:

- Boitier 5mm, pour qu'elle puisse ressortir du boitier.

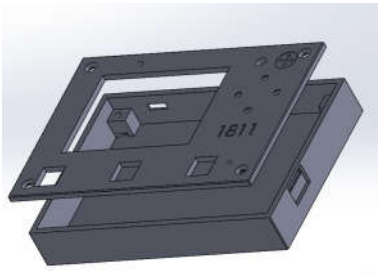
Boutons:

- 3 boutons principaux colorés
  - Vert ticket accepté
  - Rouge ticket refusé
  - Noir tous les tickets sont reset

- 4 boutons
  - 4 boutons pour le control de menu

2.3.1.3 Boitier

Le boitier sera fait à l'ES grâce à l'imprimante 3D.



2.3.1.4 Microcontrôleur

Besoins:

- 2 Ports SPI (Carte SD et écran TFT)
- Port UART (Module RF)
- 15 GPIO
- Alimentation 3V3
- Mémoire minimum de 126kB

Décision portée sur le PIC32MX170F256 en boitier TQFP 44 pattes.

2.3.2 Module esclave

2.3.2.1 Interface utilisateur

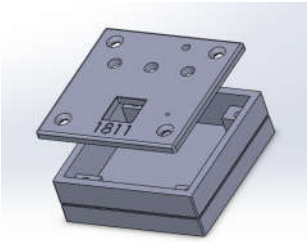
LEDs:

- 3 Leds de couleurs différentes
  - Verte, indique que le module est prêt à l'envoi d'un ticket.
  - Orange, indique état d'attente de réponse.
  - Rouge, ticket refusé ou module non linké.
- Boitier 5 mm pour qu'elles puissent dépasser du boitier
- Décision de prendre 3 LED pour être plus userfriendly, 3 LED alignées verte, orange et rouge expliquent instinctivement leurs utilités.

Bouton:

- Un bouton rouge pour l'envoi de ticket et l'annulation (maintient) 

2.3.2.2 Boitier

Le boitier sera fait à l'ES grâce à l'imprimante 3D. 

2.3.2.3 Microcontrôleur

Besoins:

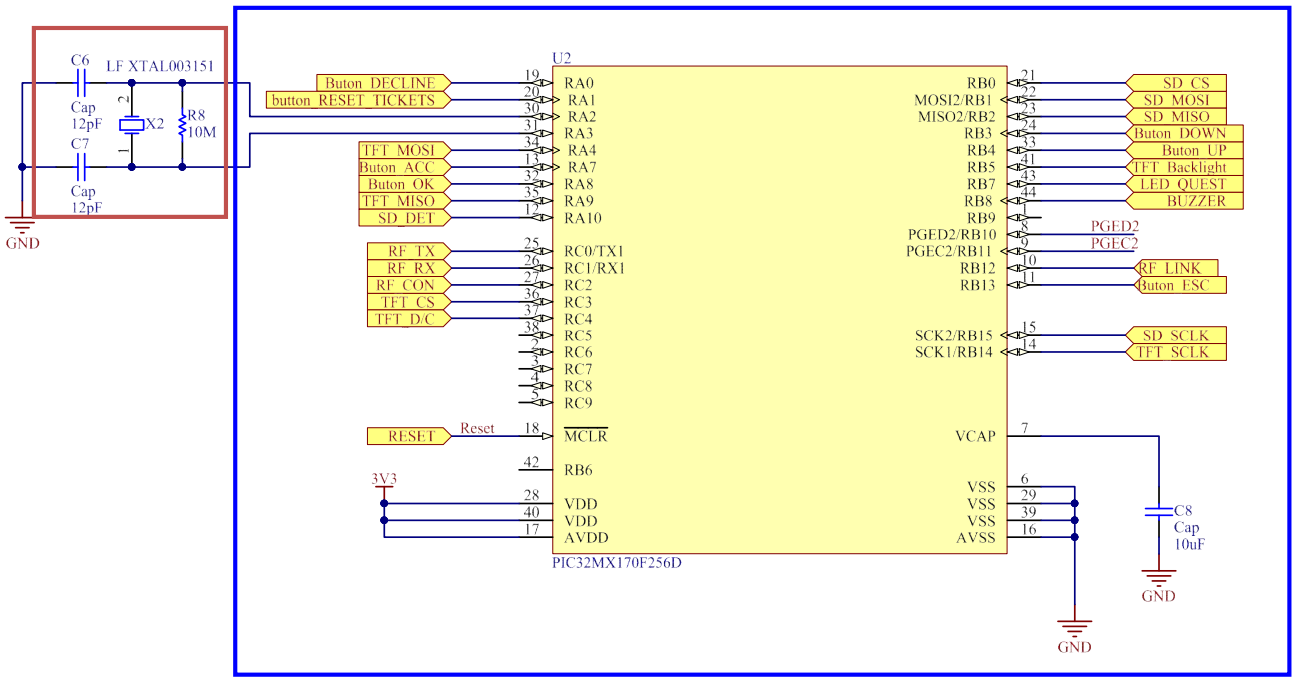
- Port UART (Module RF)
- 6 GPIO
- Alimentation 3V3
- Mémoire moins de 126kB

Décision portée sur le PIC32MX130F64 en boitier QFN 28 pattes.

3 Design

3.1 Module maitre

3.1.1 Microcontrôleur



Microcontrôleur: PIC32MX170F256D

- SPI 1: Pilotage de l'écran TFT
- SPI 2: Pilotage de la carte SD

Oscillateur:

Ne sera pas monté dans un premier temps mais si besoin j'aurais les footprints disponibles pour le monté. Cela sera un oscillateur 8MHz.

Calcul des condensateurs:

- CIN = PIC32_OSC2_Pin Capacitance = ~4-5 pF
- COUT = PIC32_OSC1_Pin Capacitance = ~4-5 pF
- C1 and C2 = XTAL manufacturing recommended loading capacitance
- Estimated PCB stray capacitance, (i.e.,12 mm length) = 2.5 pF

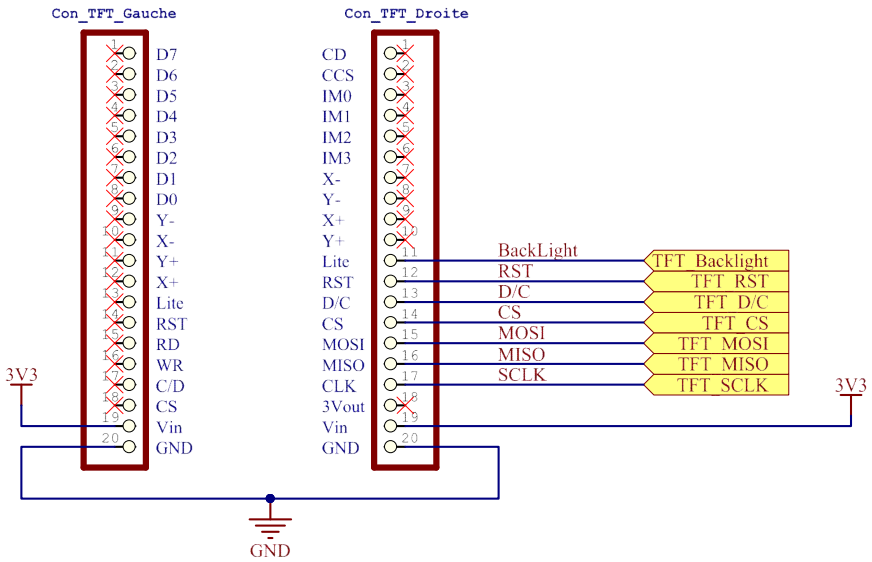
$$C_{LOAD} = \{([C_{IN} + C1] * [C_{OUT} + C2]) / [C_{IN} + C1 + C2 + C_{OUT}]\} + \text{estimated oscillator PCB stray capacitance}$$

C1 et C2 = 16 pF (datasheet Quartz)

CLoad = {([5pF + 16pF]*[5pF + 16pF])/[5pF + 16pF + 16pF + 5pF]} + 2.5pF = 13pF

La résistance de 1M est recommandée par le datasheet lorsque la fréquence du quartz est entre 4M et 10M Hertz

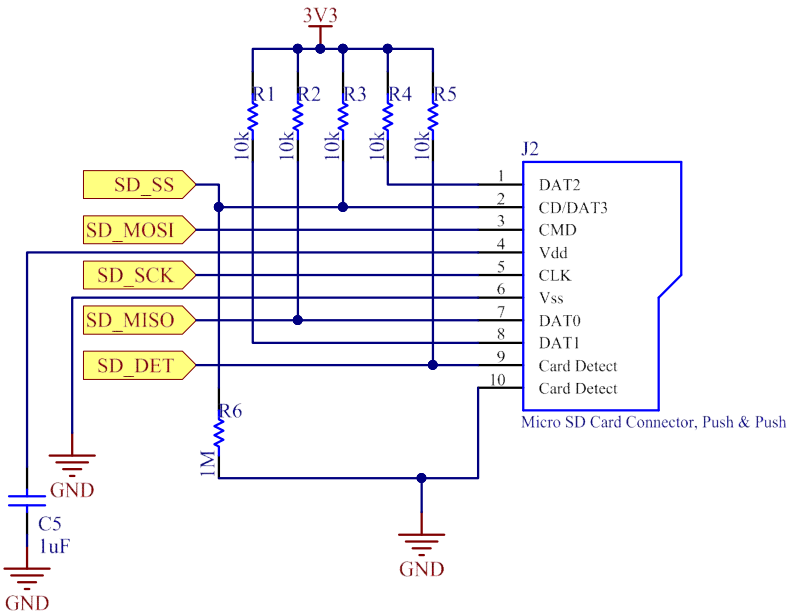
3.1.2 TFT



Ici nous avons le pinning de l'écran TFT. L'écran propose plusieurs possibilité de communication, j'utilise la communication SPI car plus simple et déjà utilisé dans un précédent projet.

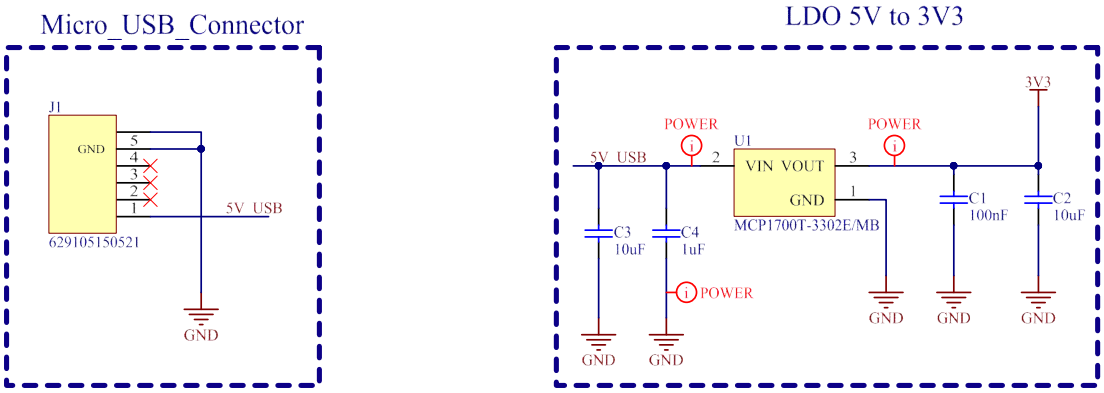
"Lite" permettra l'allumage du rétro-éclairage.
"Reset" est liée au reste de la carte.
"D/C" permettra de choisir si l'on envoie des données ou des commandes.

3.1.3 SD



Ici nous avons le holder de la carte SD, le design est repris du starter-kit ES.

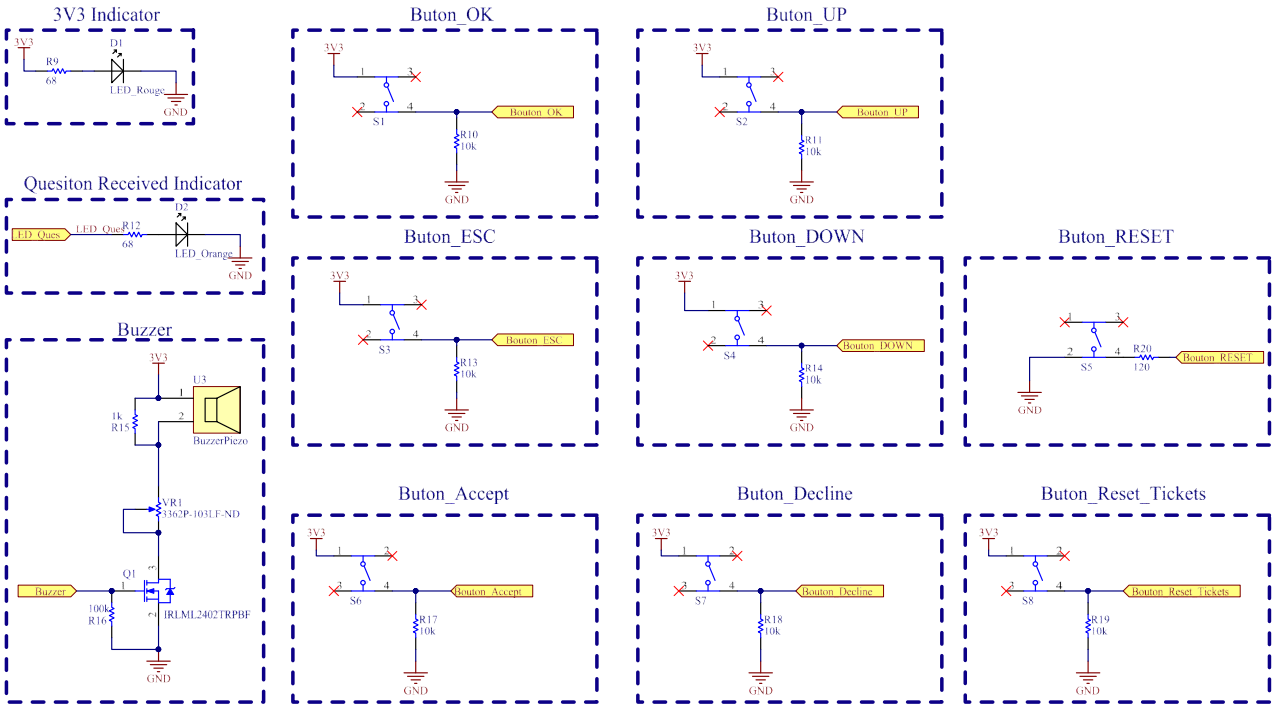
3.1.4 Alimentation 3.3V



Pour la gestion de l'alimentation j'ai utilisé leMCP1700T-330 qui est un régulateur de tension linéaire 3V3 volts.

Ce régulateur peut me fournir un courant de 200mA maximum à 3.3V (datasheet).
Le module RF consomme moins de 50 mA (estimation tirée de la documentation du projet 1623)
L'écran TFT consomme au maximum 100mA (estimation tirée de la description de l'écran sur le site adafruit)
Ce qui laisse 50mA de consommation pour mon PCB ce qui sera suffisant car mon microcontrôleur consommera 50mA au maximum.

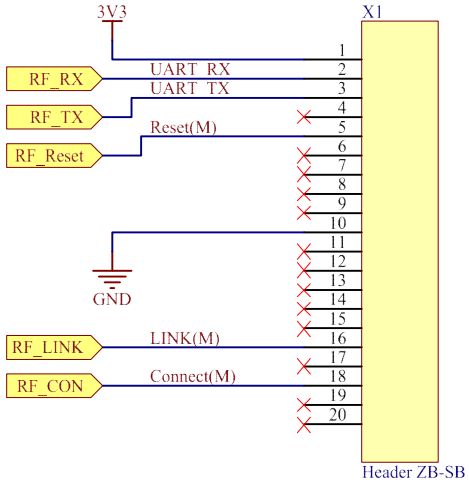
3.1.5 Interface utilisateur



Led Orange = Led rouge => consommation 20mA pour tension de 2V

RLed = (Vcc - VLed) / ILed = (3.3 - 2) /20m = 65ohm -> E12 -> 68ohm

3.1.6 XBee

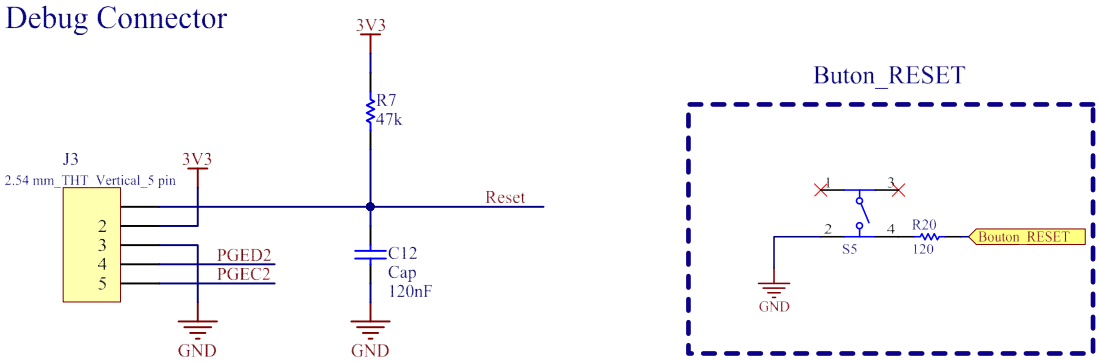


Ici nous avons le connecteur Xbee, 2 barrettes à broches femelle

Attention: pitch => 2mm

Sa ligne Reset est liée au reste de la carte.
Les lignes "LINK" et "Connect" sont des lignes destinées à être utilisées par de simples GPIO.

3.1.7 Circuit de reset

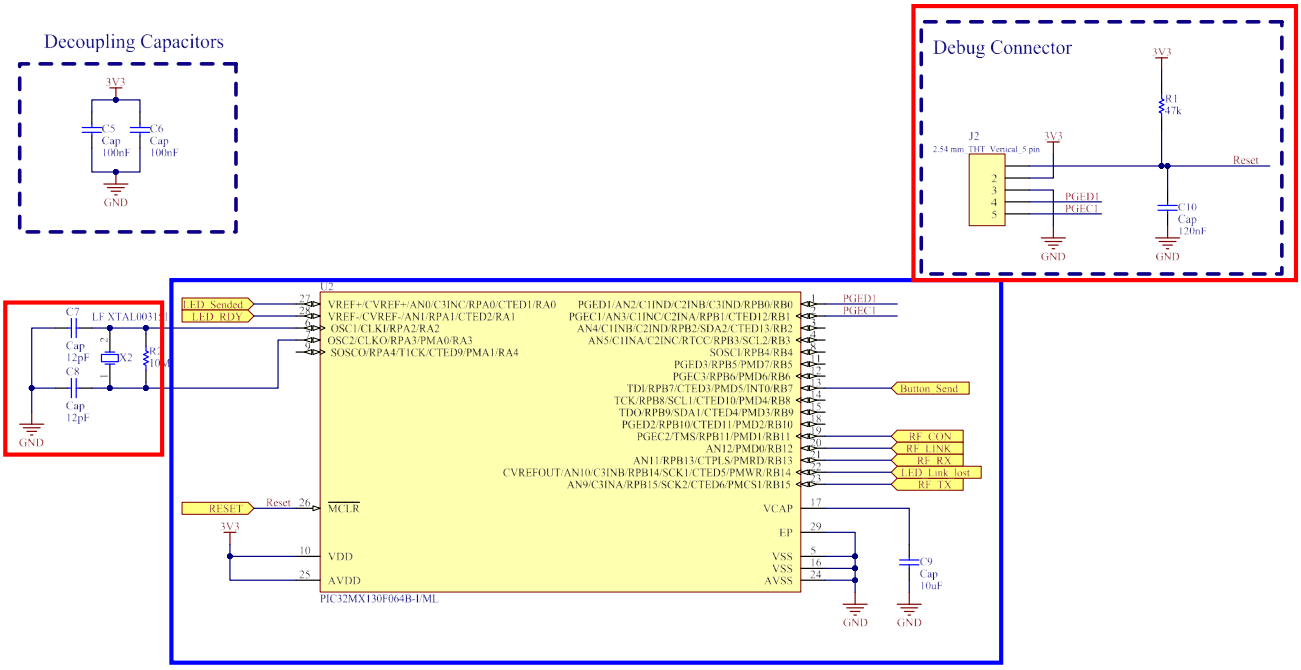


Ici lors de la mise sous tension le condensateur C12 permettra à la carte de se maintenir en reset pendant minimum 25ms lors de la mise sous tension.

t = R7 * C12 = 47k * 120n = 5.64ms
Pour charger le condensateur à son maximum on estime qu'il faut 5 fois ce temps.
5t = 28.2ms.

3.2 Module esclave

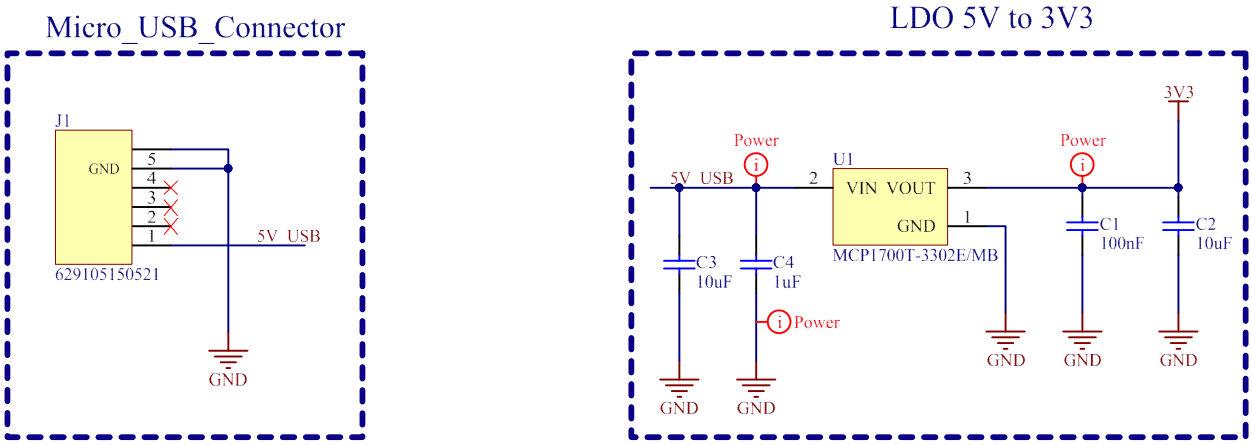
3.2.1 Microcontrôleur



Microcontrôleur: PIC32MX130F064

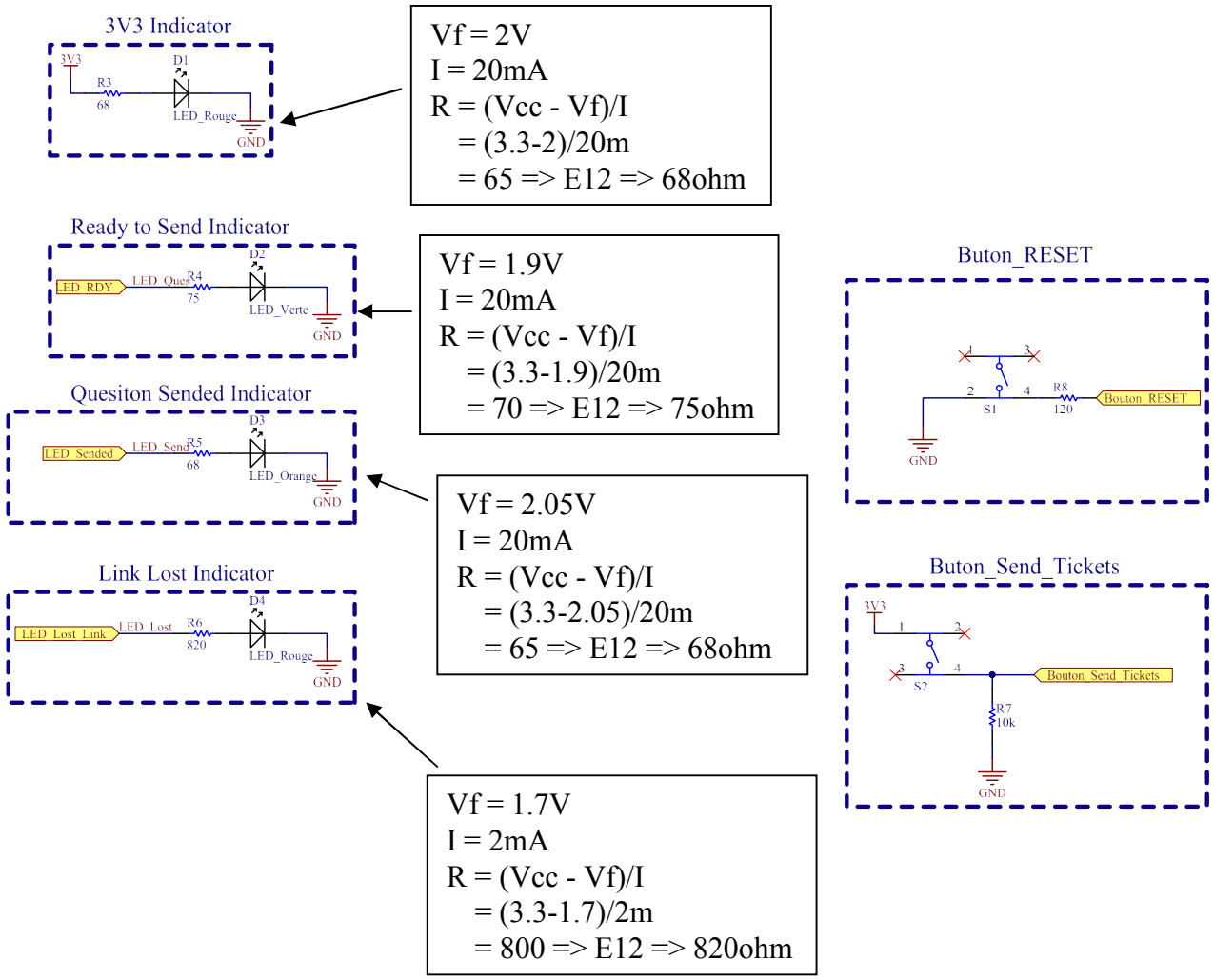
Ici, à part le microcontrôleur, les dimensionnements de composants périphériques sont les mêmes que sur la carte maitre. Voir page 3-9.

3.2.2 Alimentation 3.3V

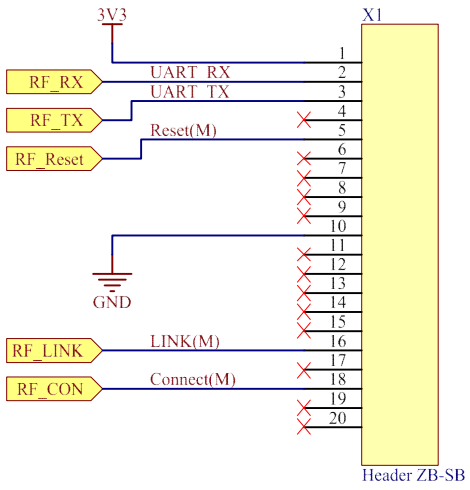


Même dimensionnement que l'alimentation de la carte maitre. Voire page 3-11

3.2.3 Interface utilisateur



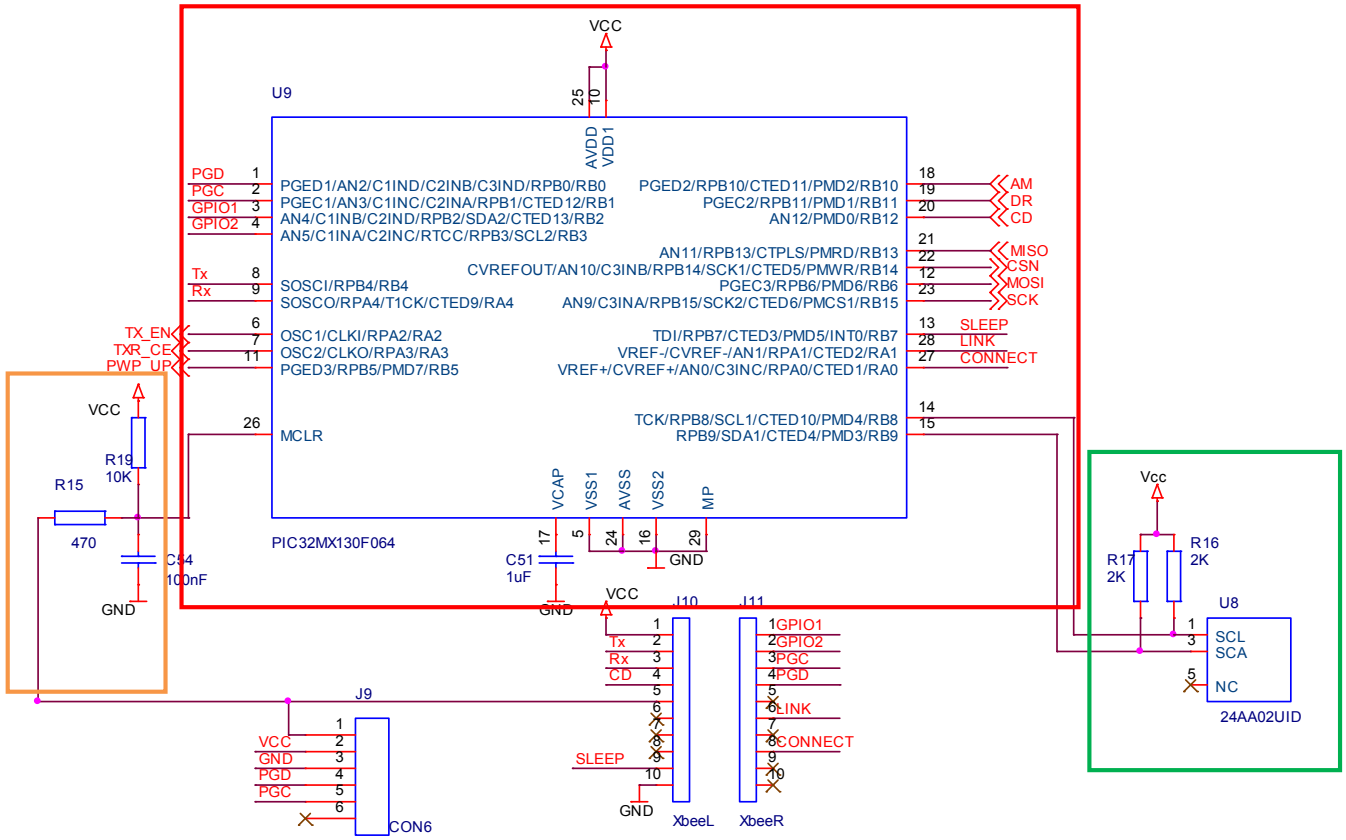
3.2.4 XBee



Même que sur la carte maitre. Voir page 3-12

3.3 Analyse du module RF

3.3.1 Microcontrôleur

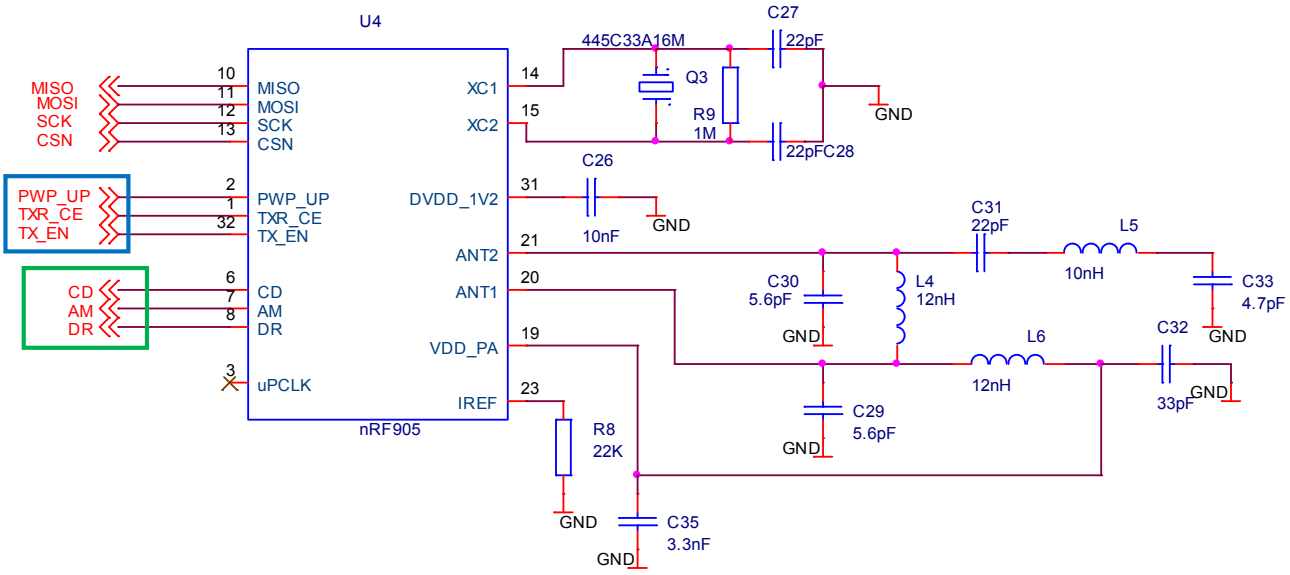


Microcontrôleur: PIC32MX130F064 en boitier QFN

Mémoire EEProm: Cette mémoire EEprom permet d'enregistrer l'ID de la carte.
Communication en I2C

Circuit de reset: Ce circuit de Reset maintient le microcontrôleur en reset durant un minimum de 5ms lors de sa mise sous tension.
On peut remarquer que la décharge du condensateur ne se fera que si l'on a un élément connecté sur la broche "reset" du connecteur Xbee.

3.3.2 nRF905



Ici nous avons le circuit RF. Le chip communique en SPI.

Lignes commandes:

PWR_UP	TRX_CE	TX_EN	Operating Mode
0	X	X	Power down and SPI programming
1	0	X	Standby and SPI programming
1	X	0	Read data from RX register
1	1	0	Radio Enabled - ShockBurst™ RX
1	1	1	Radio Enabled - ShockBurst™ TX

Mode lecture RF

Mode d'envoi RF

Lignes de status:

CD: Carrier Detect
En mode de lecture RF la pin CD se met à "1" lorsque que le chip détecte une porteuse.

AM: Address Match
En mode de lecture RF la pin AM se met à "1" lorsque que le chip detecte son adresse dans la tram RF.

DR: Dataready
En mode de lecture RF la pin DR se met à "1" lorque le chip aura vérifié que les données recues sont juste (CRC).

En mode d'envoi la pin se met à "1" lorsque un packet de donnée à été envoié.

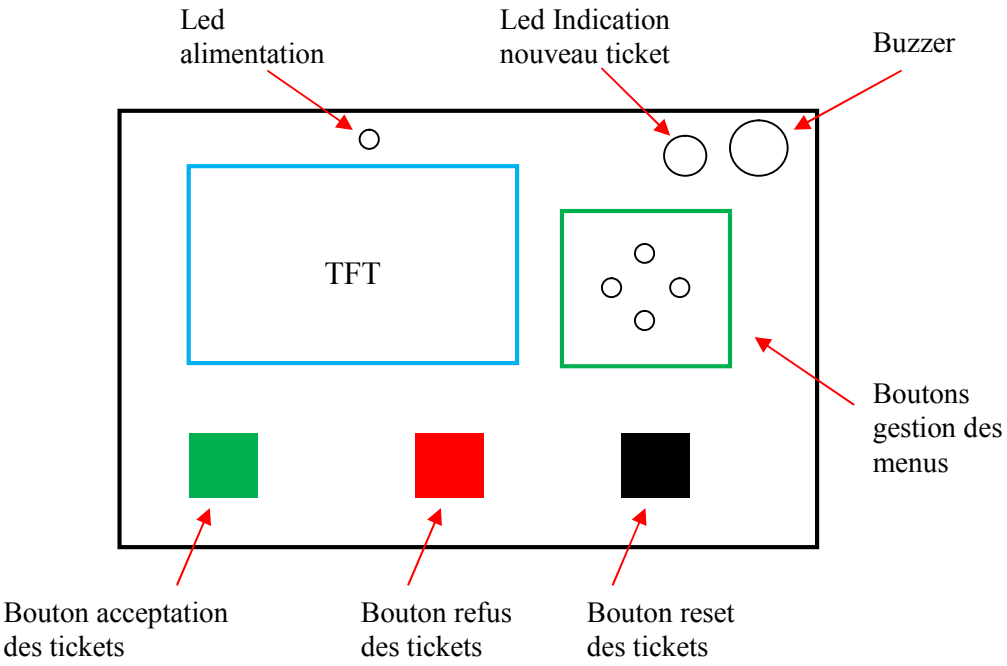
En reception si le module nous informe qu'il à recu une bonne adresse (AM à "1") mais que la pin DR n'es pas passée a "1" avant que la pin AM redescende à "0", c'est qu'une erreure c'est glissée dans la tram et que le CRC est faux.

3.4 Boitiers

Contrairement à ce qui était prévu dans le planning, j'ai préféré commencer par un design de boitier avant de créer le PCB afin d'avoir une réalisation plus esthétique, et cela me permettra d'avoir une première version de boitier à tester dès que j'aurais monté le PCB, ce qui me permettra de facilement faire des retouches et envoyer la production des boitiers finaux.

3.4.1 Boiter Maitre

3.4.1.1 Idée du design



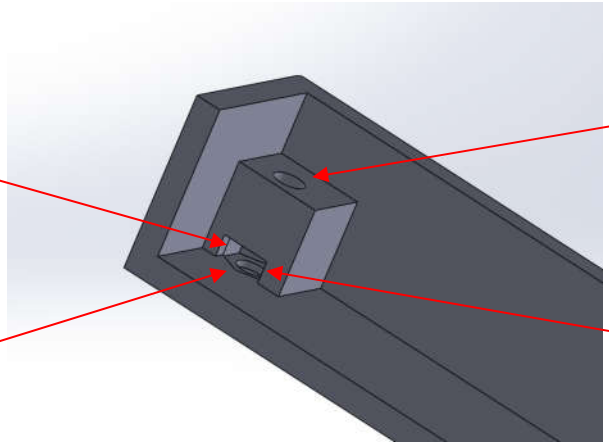
3.4.1.2 Création

Concernant la création du boitier, j'ai pris contact avec M. Muller, qui s'occupe de l'impression 3D à l'ES.

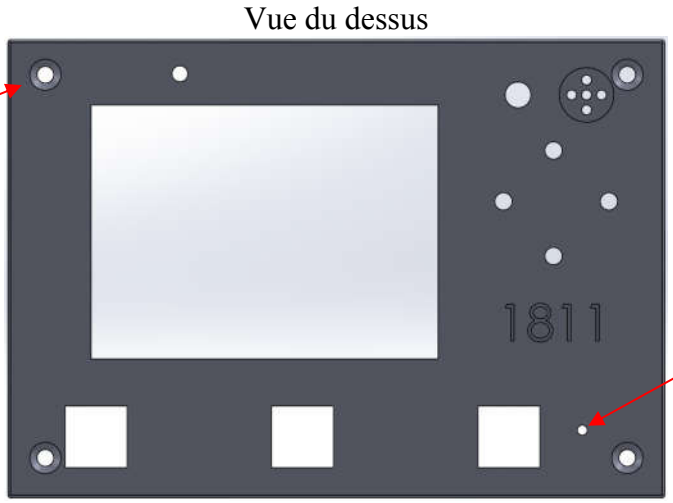
Dans un premier temps il a fallu prendre en compte les éléments suivants:

- L'imprimante a une précision de 0.1 mm
- L'épaisseur des parois du boitier doit être de 3mm minimum pour garantir une certaine solidité.
- Le taraudage du boitier risque de casser le boitier donc à éviter

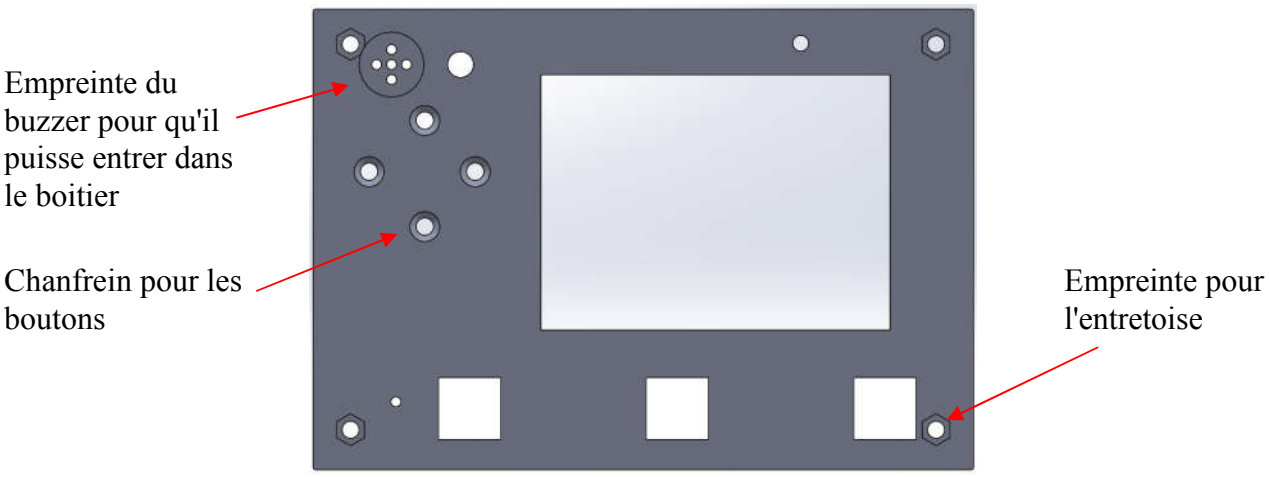
A propos de la visserie M. Muller m'as conseillé sur une façon simple de procéder:



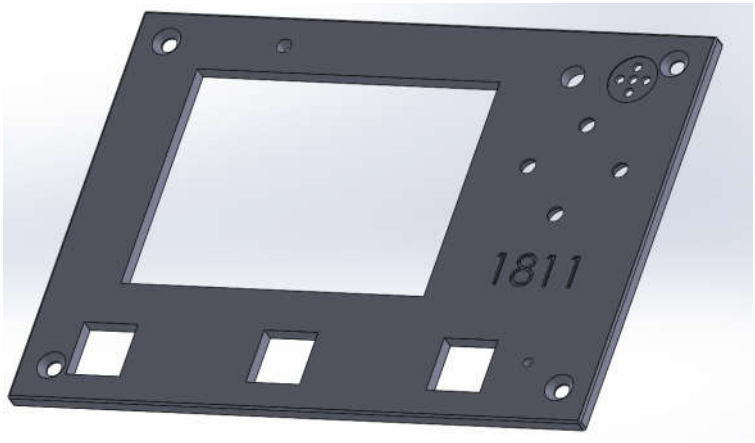
3.4.1.2.1 Top



Vue du dessous

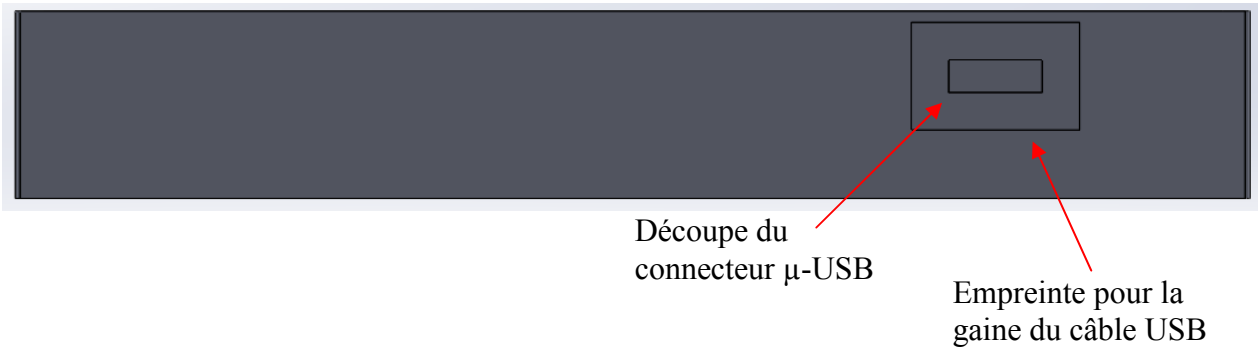


Vue 3D



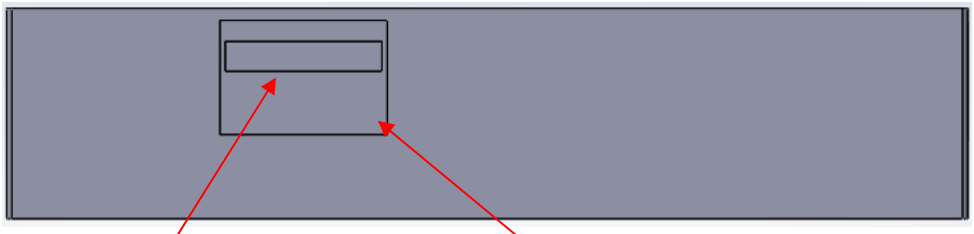
3.4.1.2.2 Bot

Vue de derrière

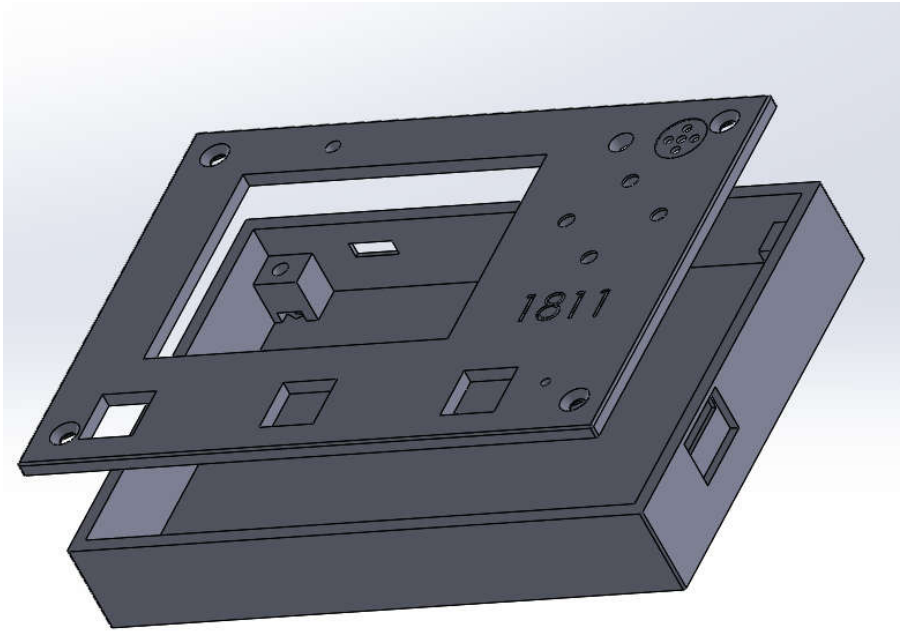


Vue de dessus

Vue de droite

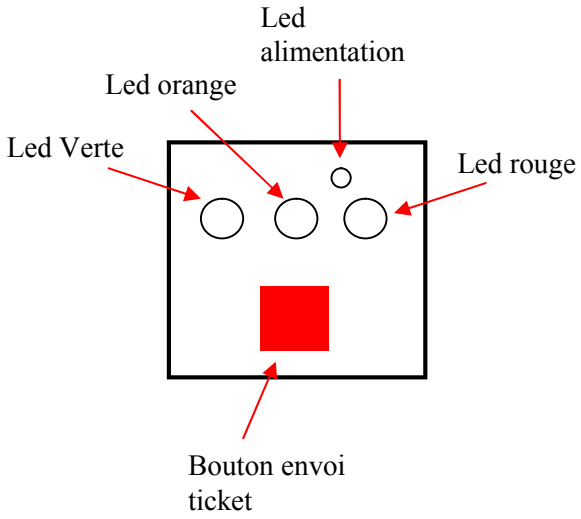


3.4.1.2.3 Assemblage

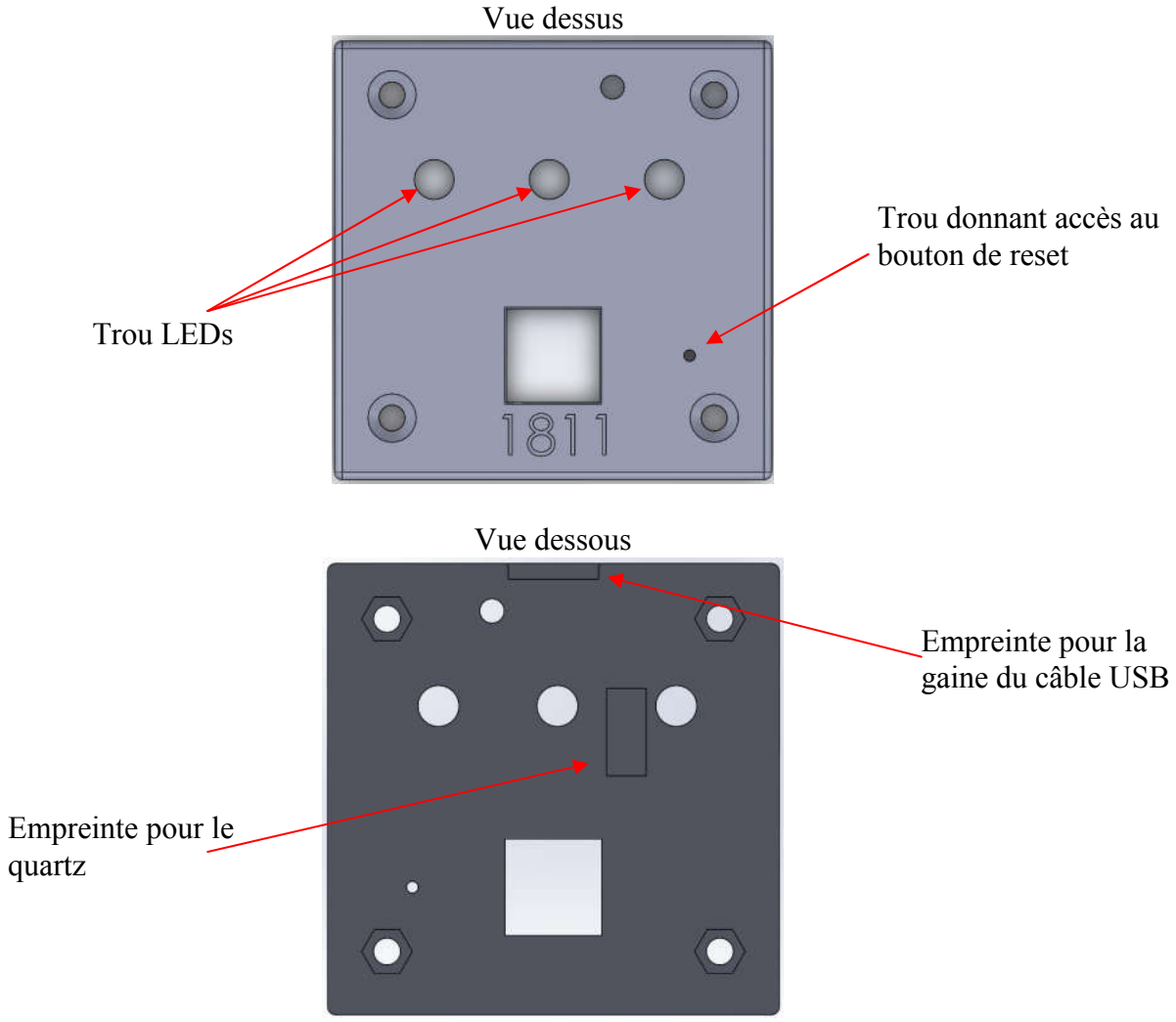


3.4.2 Boitier Esclave

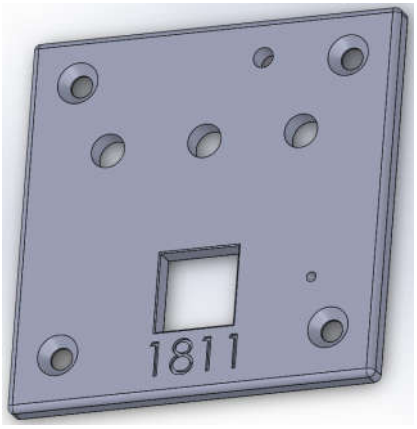
3.4.2.1 Idée du design



3.4.2.1.1 Top

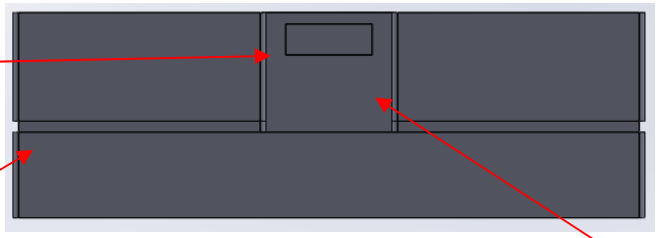


Vue 3D



3.4.2.1.2 Bot

Vue de derrière

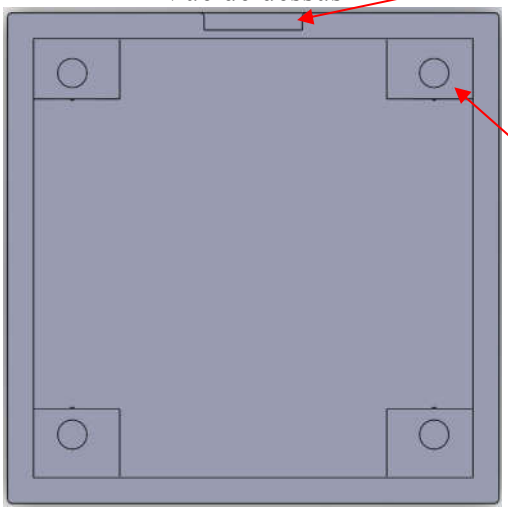


Découpe du connecteur µ-USB

Rainure esthétique

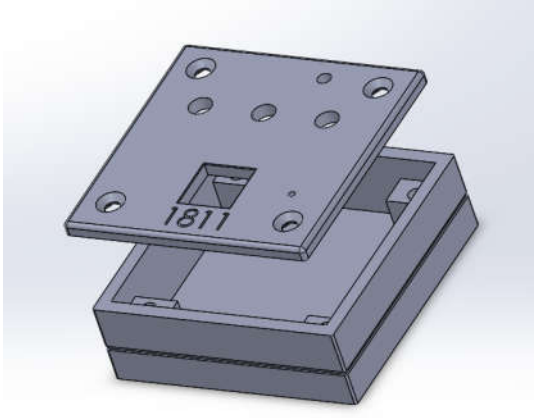
Empreinte pour la gaine du câble USB

Vue de dessus



Trou de passage des vis

3.4.2.1.3 Assemblage

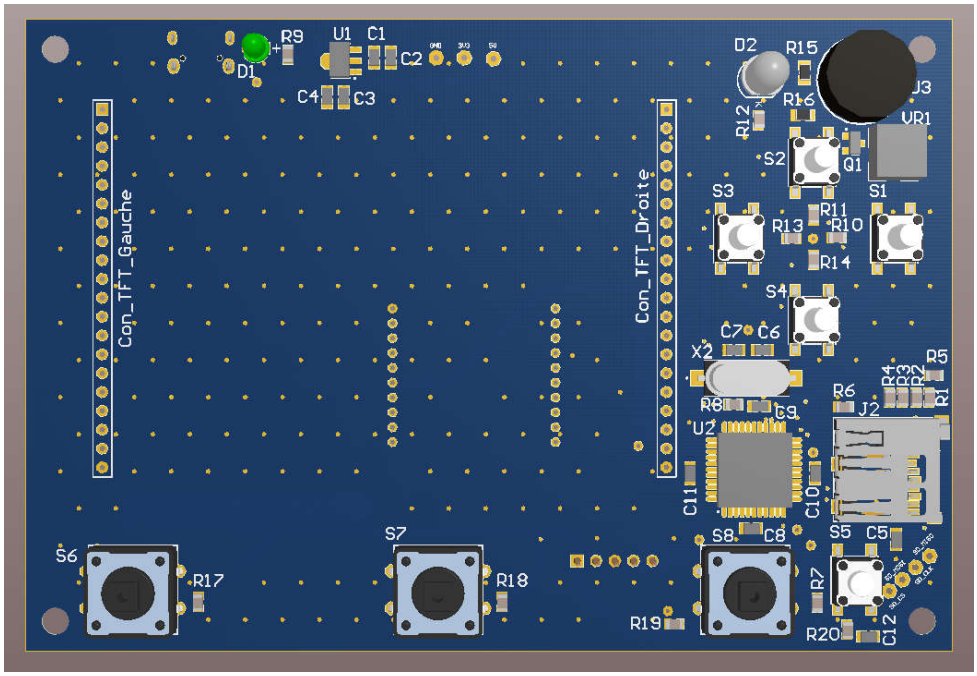


3.5 PCB

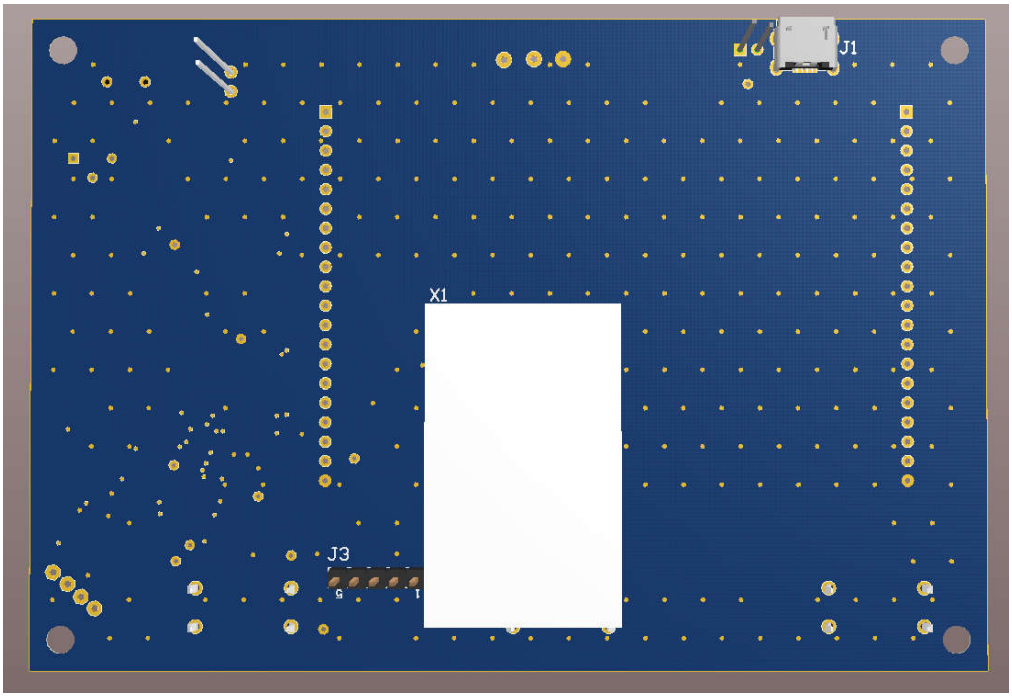
3.5.1.1 Contraintes

Seule une contrainte doit être respectée et c'est celle de la position des composants.

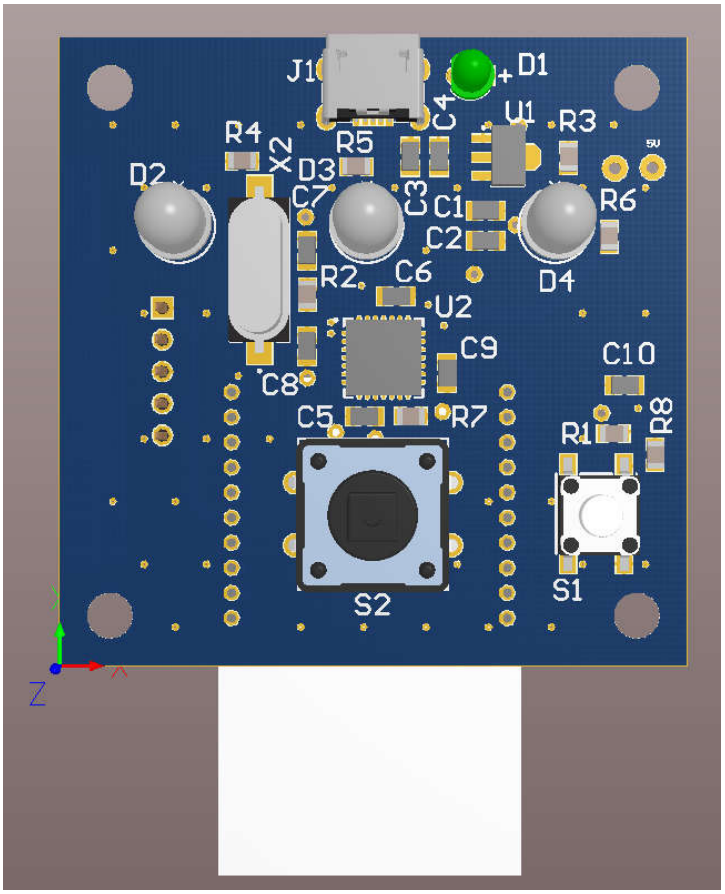
3.5.1.2 Vue TOP



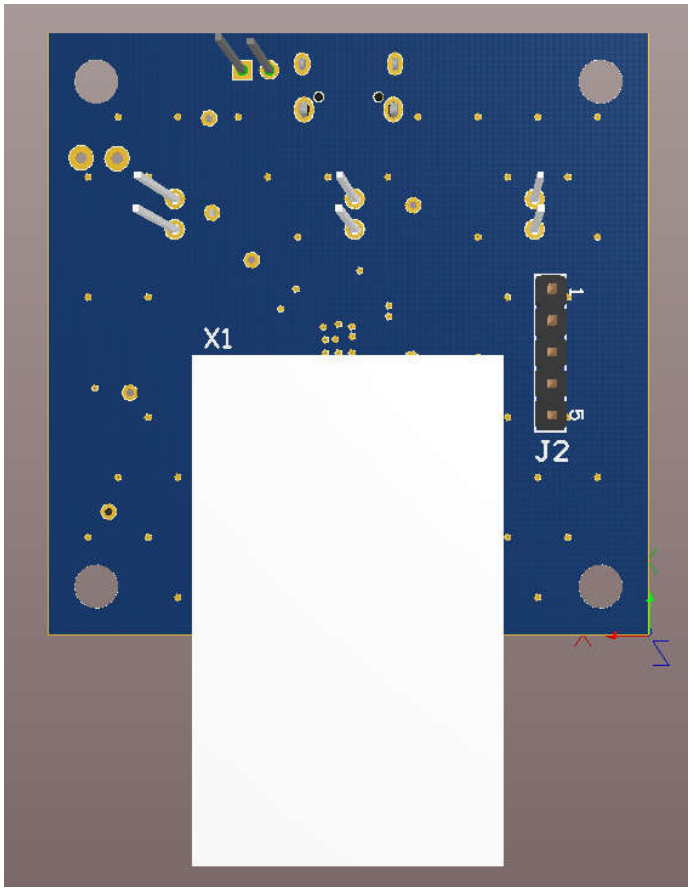
3.5.1.3 Vue BOT



3.5.1.4 Vue TOP



3.5.1.5 Vue BOT



3.6 Montage et Mise en Service

Pour le montage, l'alimentation a été montée en premier lieu. Une fois montée, elle a été testée. Quand l'alimentation est OK, on monte le microcontrôleur.
Une fois monté, test de la programmation, si le microcontrôleur répond bien alors nous pouvons monter le reste de la carte.

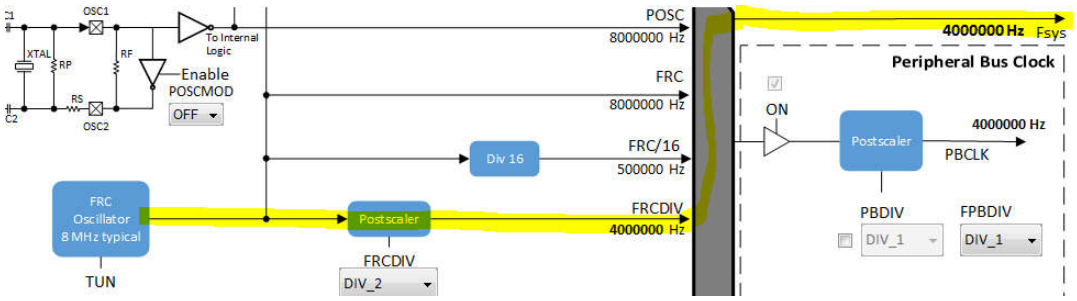
4 Programme

4.1 Programme module RF

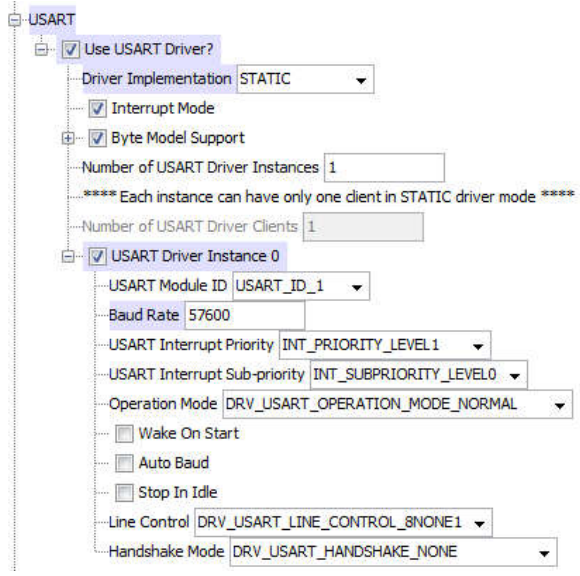
4.1.1 Configuration

La configuration du PIC est la même que sur le projet 1623.

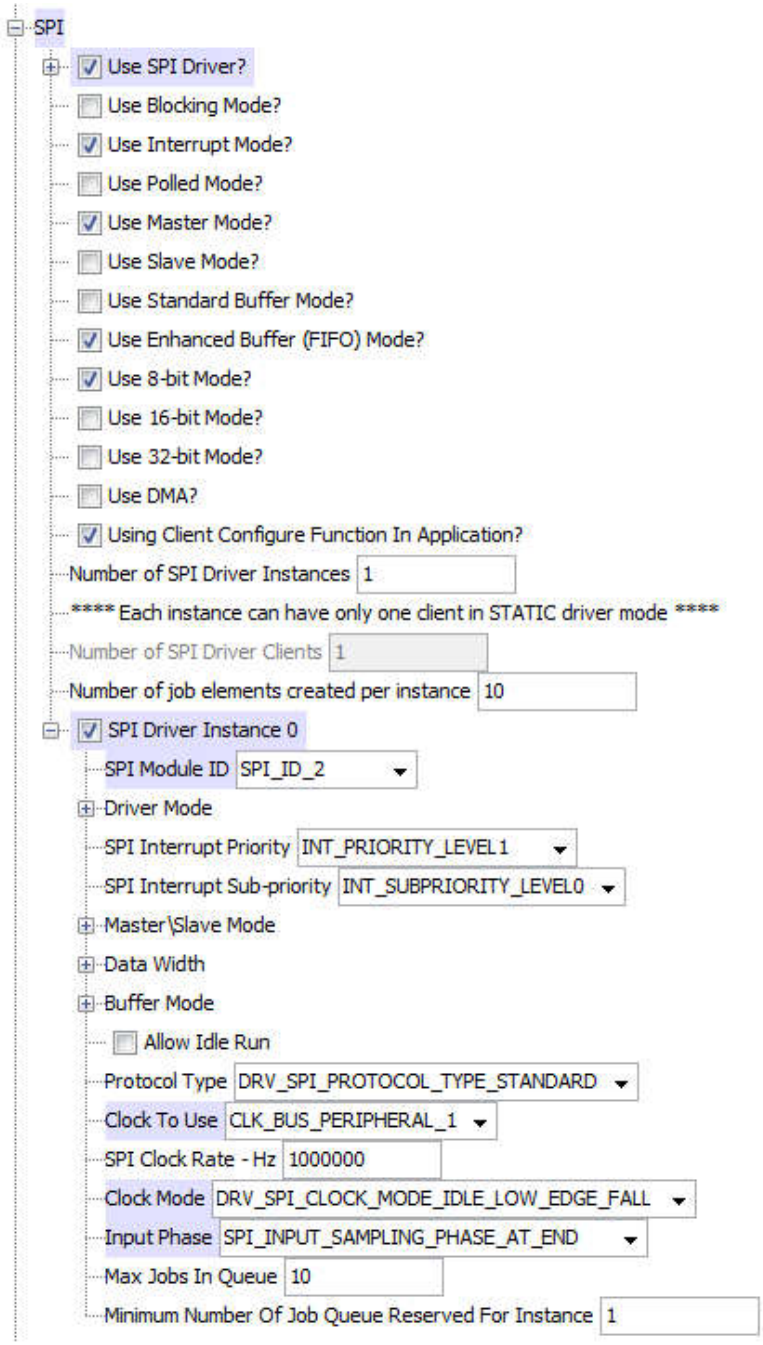
L'horloge est réglée en mode oscillateur interne 4MHz.



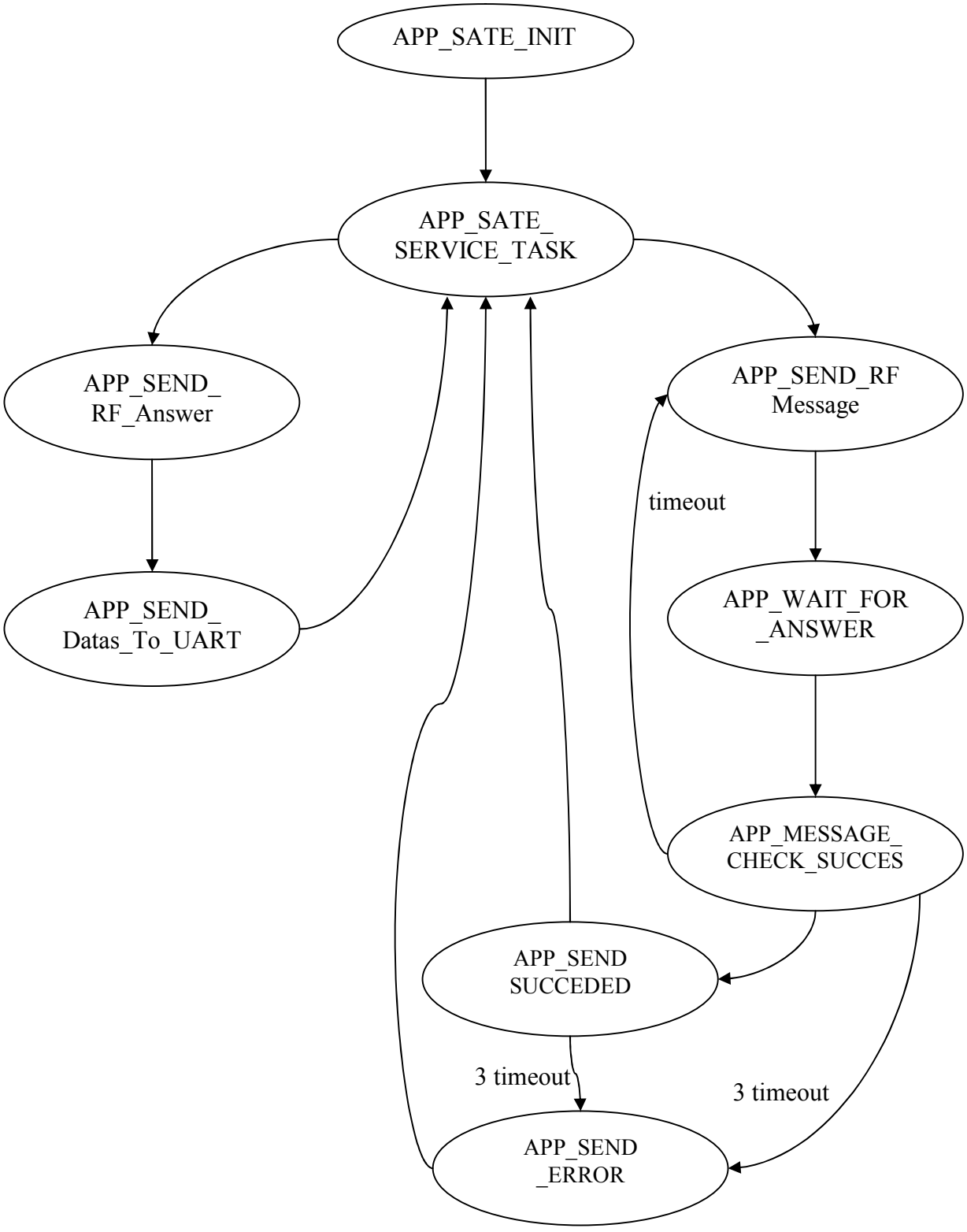
UART:



SPI:


SPI
Use SPI Driver?
Use Blocking Mode?
Use Interrupt Mode?
Use Polled Mode?
Use Master Mode?
Use Slave Mode?
Use Standard Buffer Mode?
Use Enhanced Buffer (FIFO) Mode?
Use 8-bit Mode?
Use 16-bit Mode?
Use 32-bit Mode?
Use DMA?
Using Client Configure Function In Application?
Number of SPI Driver Instances 1
**** Each instance can have only one client in STATIC driver mode ****
Number of SPI Driver Clients 1
Number of job elements created per instance 10
SPI Driver Instance 0
SPI Module ID SPI_ID_2
Driver Mode
SPI Interrupt Priority INT_PRIORITY_LEVEL1
SPI Interrupt Sub-priority INT_SUBPRIORITY_LEVEL0
Master\Slave Mode
Data Width
Buffer Mode
Allow Idle Run
Protocol Type DRV_SPI_PROTOCOL_TYPE_STANDARD
Clock To Use CLK_BUS_PERIPHERAL_1
SPI Clock Rate - Hz 1000000
Clock Mode DRV_SPI_CLOCK_MODE_IDLE_LOW_EDGE_FALL
Input Phase SPI_INPUT_SAMPLING_PHASE_AT_END
Max Jobs In Queue 10
Minimum Number Of Job Queue Reserved For Instance 1

4.1.2 Diagramme des états



4.1.3 Configuration du nRF904

Le chip nRF904 nécessite une configuration avant son utilisation.
Cette configuration est tirée du projet 1623

```
unsigned char Tab_config [10] = {0x75,0x0E,0x44,0x08,0x08,0x00,0x00,0x00,0x00,0x5E};
```

Chaque Byte sera écrit dans le registre de configuration.
(Se référer au datasheet du nRF904 pages 23-24)

Le chip possède donc 5 registres

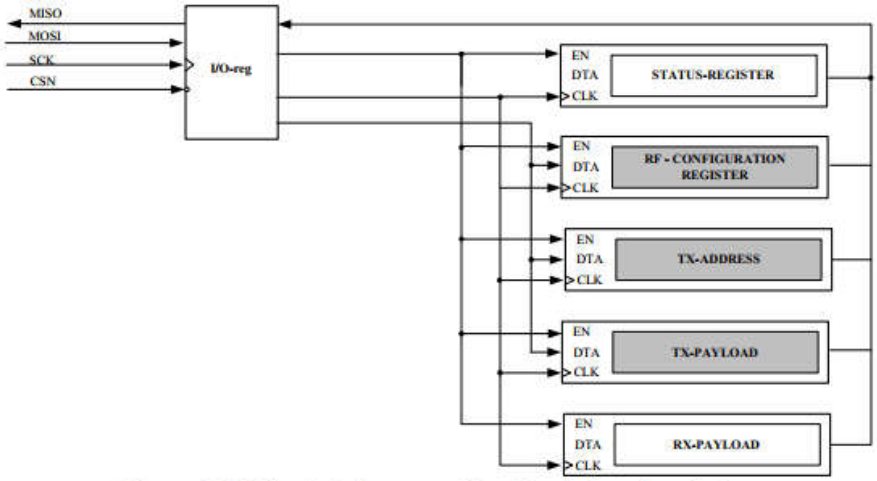


Figure 5. SPI – interface and the five internal registers.

Internal registers	Description
Status – Register	Register contains status of Data Ready (DR), Address Match (AM).
RF – Configuration Register	Register contains transceiver setup information such as frequency and output power ext.
TX – Address	Register contains address of target device. How many bytes used is set in the configuration register.
TX – Payload	Register containing the payload information to be sent in a Shock-Burst™ packet. How many bytes used is set in the configuration register.
RX – Payload	Register containing the payload information derived from a received valid ShockBurst ™ packet. How many bytes used is set in the configuration register. Valid data in the RX-Payload register is indicated with a high Date Ready (DR) signal.

pour aller écrire dans ces registres il suffi d'envoier une première tram SPI conntent les infos suivantes:

```
#define WC  0x00  // Write configuration register command
#define RC  0x10  // Read  configuration register command
#define WTP  0x20 // Write TX Payload  command
#define RTP  0x21 // Read  TX Payload  command
#define WTA  0x22 // Write TX Address  command
#define RTA  0x23 // Read  TX Address  command
#define RRP  0x24 // Read  RX Payload  command
```

En ayant les pins de commande en STAND BY (voir page 3-16) sauf pour la lecture du registre de payload RX qui neccecite d'avoir les pin de commande en RX shockburst (voir page 3-16)

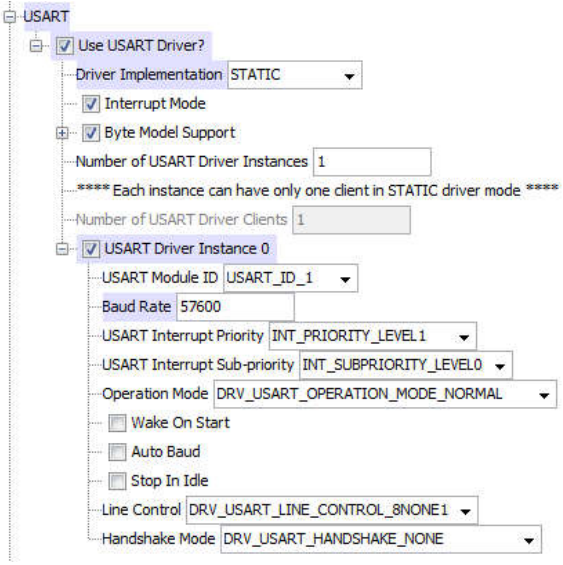
4.1.4 Traitement des informations

Le module RF sera presque transparent au niveau des informations transmises.

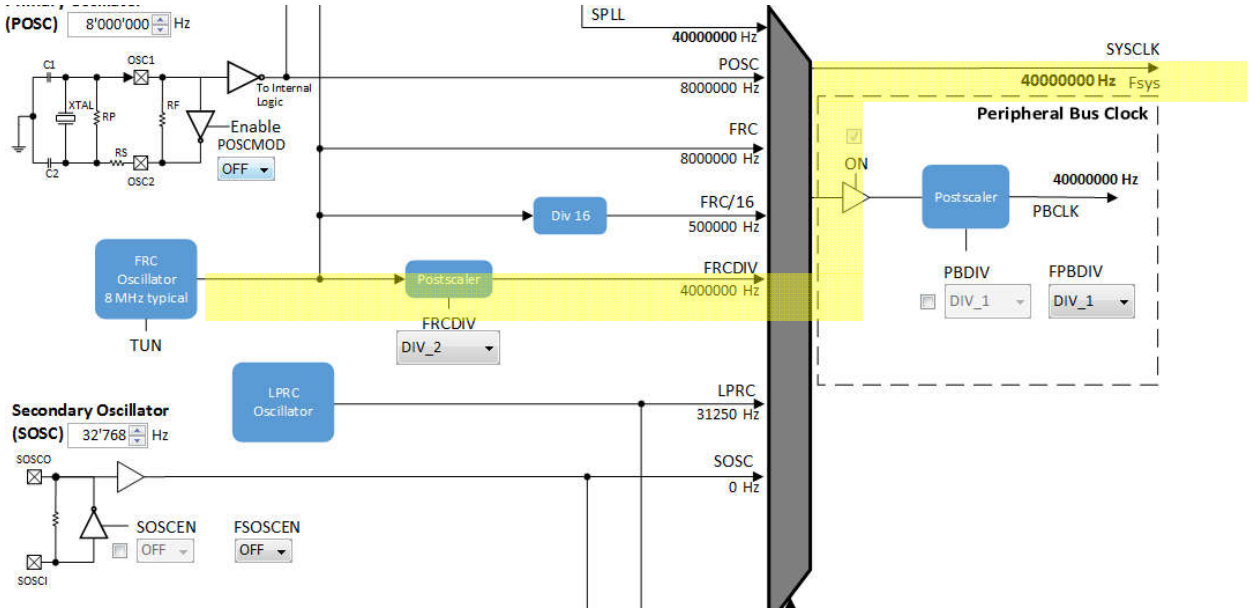
Du coté UART le module attendra une tram contenant l'adresse de la cible ainsi que les information à transmettre. Alors que du côté RF le module recevra l'adresse de la source ainsi que les données.

4.2 Programme Esclave

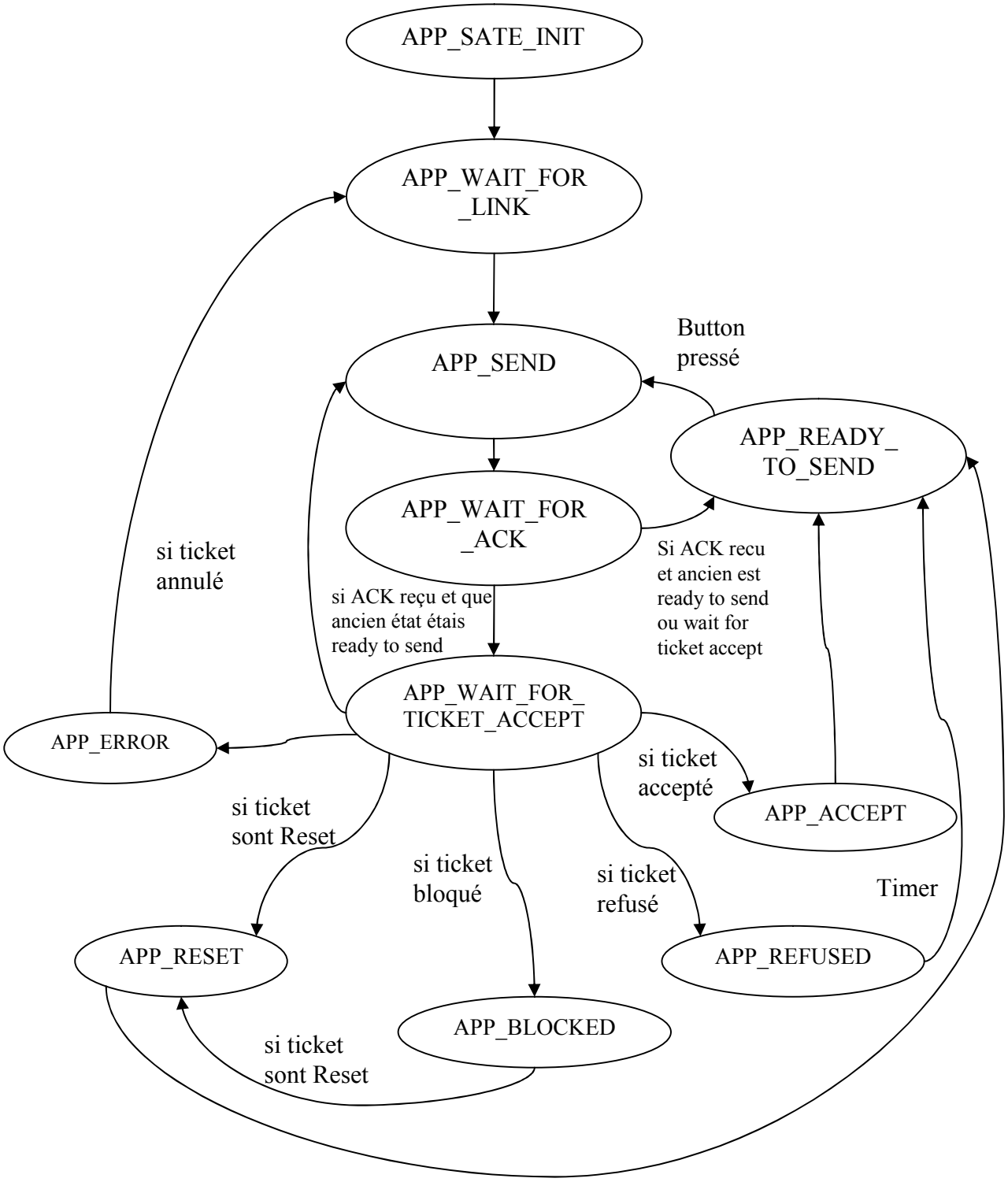
4.2.1 Configuration UART



4.2.2 Configuration Fréquence système

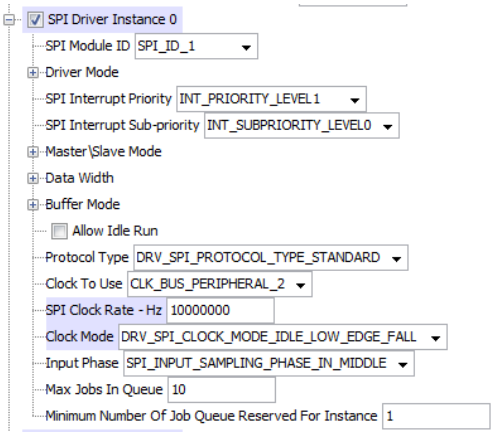


Utilisation de la RTC interne 8MHz pour créer notre fréquence système à 40MHz

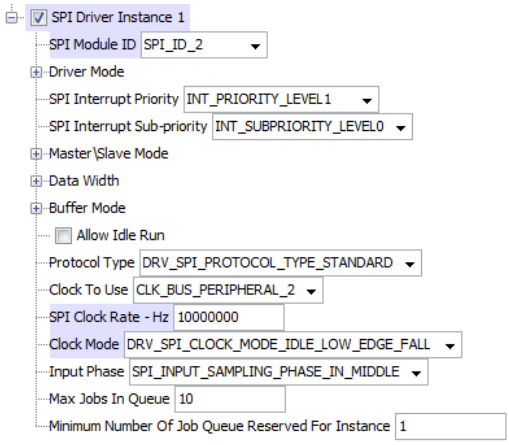

APP_SATE_INIT
APP_WAIT_FOR
_LINK
APP_SEND
Button
pressé
APP_READY_
TO_SEND
APP_WAIT_FOR
_ACK
si ticket
annulé
Si ACK recu
et ancien est
ready to send
ou wait for
ticket accept
si ACK reçu et que
ancien état étais
ready to send
APP_WAIT_FOR_
TICKET_ACCEPT
APP_ERROR
si ticket
accepté
APP_ACCEPT
si ticket
sont Reset
si ticket
bloqué
si ticket
refusé
Timer
APP_RESET
APP_REFUSED
APP_BLOCKED
si ticket
sont Reset

4.3 Programme Maitre

4.3.1 Configuration SPI

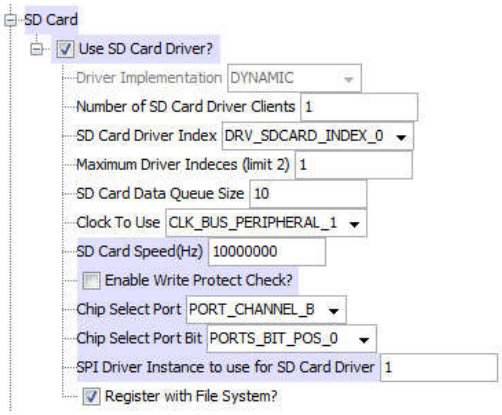


Configuration SPI TFT

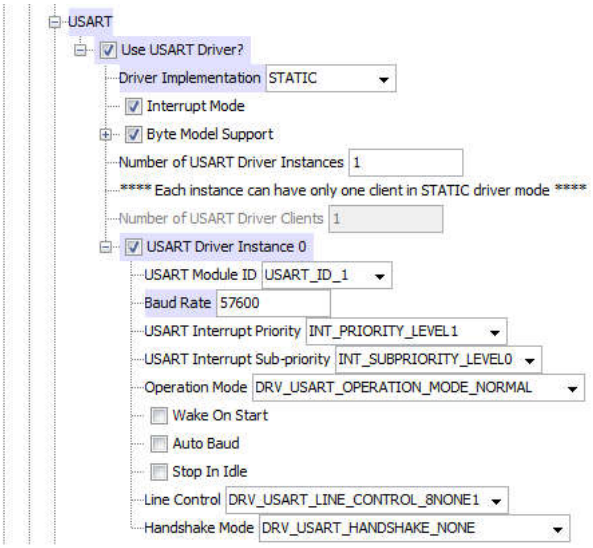


Configuration SPI carte SD

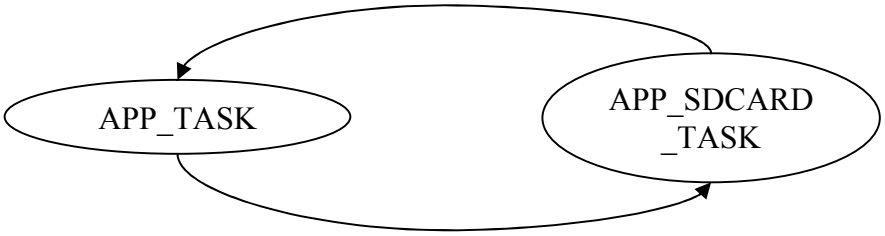
4.3.2 Configuration Carte SD



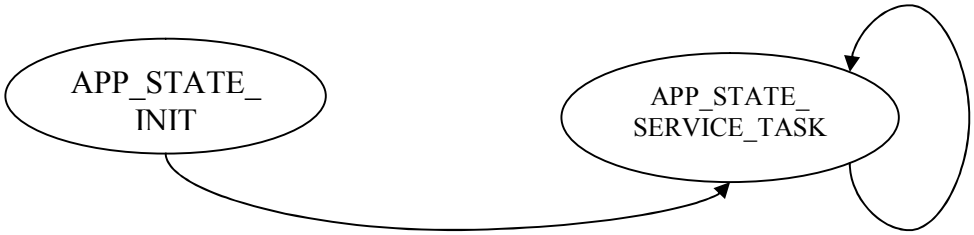
4.3.3 UART



4.3.4 Diagramme system task

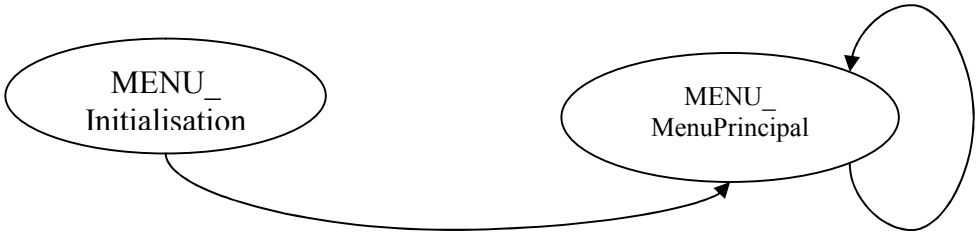


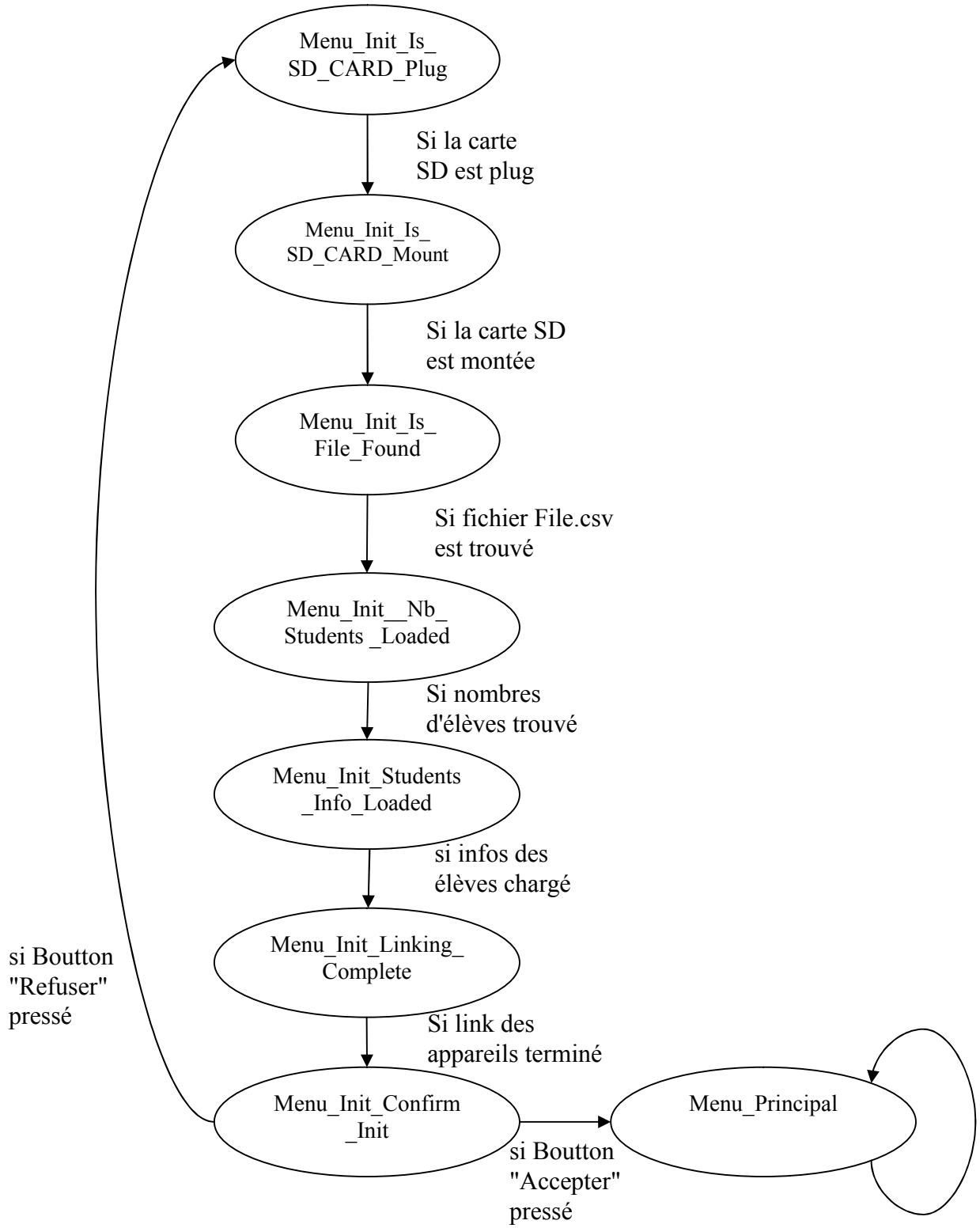
4.3.5 APP_TASK



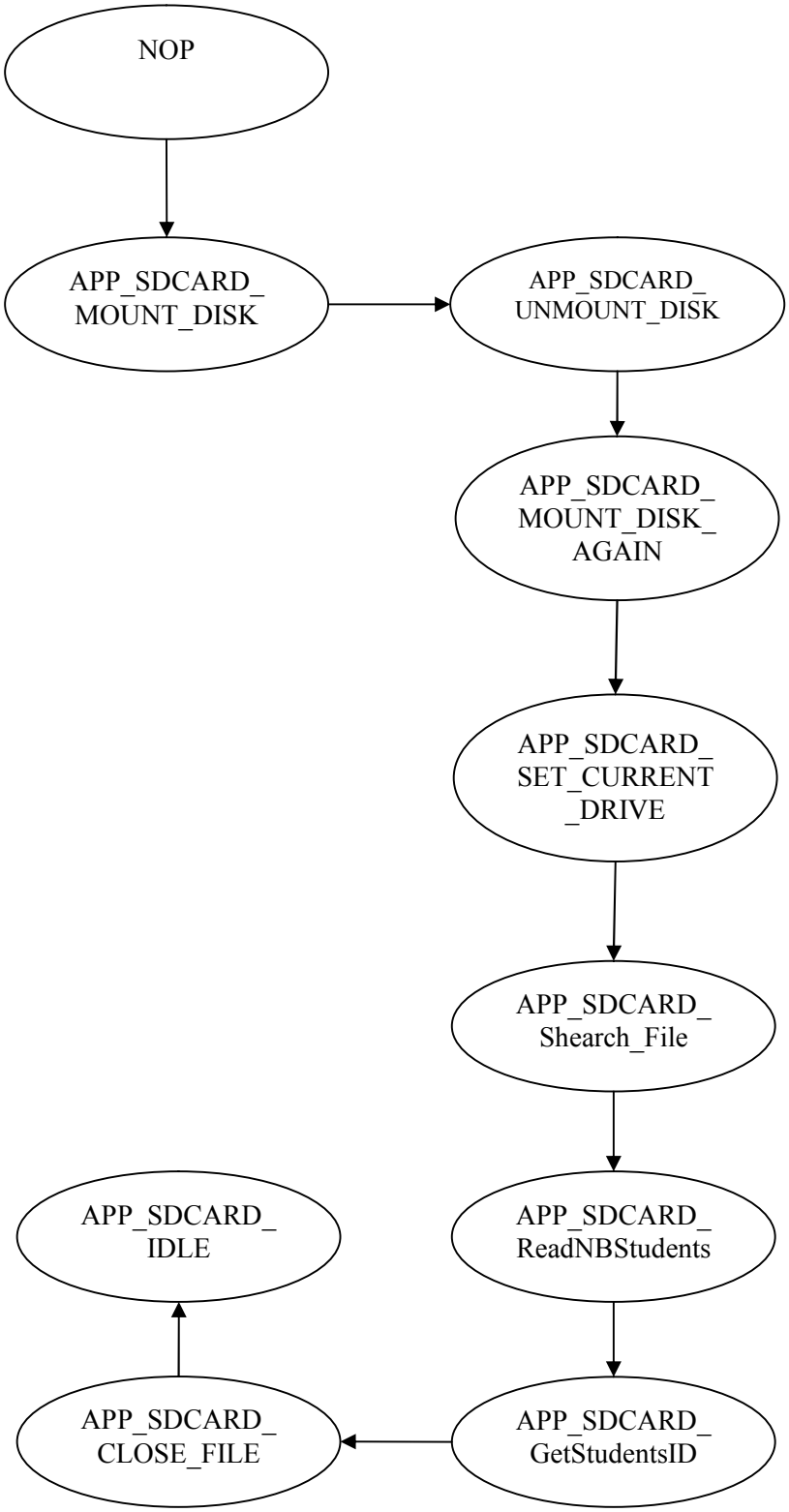
Dans le APP_STATE_SERVICE_TASK nous avons l'appel de la fonction de gestion des menus

4.3.6 Gest_Menu




Menu_Init_Is_
SD_CARD_Plug
Si la carte
SD est plug
Menu_Init_Is_
SD_CARD_Mount
Si la carte SD
est montée
Menu_Init_Is_
File_Found
Si fichier File.csv
est trouvé
Menu_Init__Nb_
Students _Loaded
Si nombres
d'élèves trouvé
Menu_Init_Students
_Info_Loaded
si infos des
élèves chargé
Menu_Init_Linking_
Complete
Si link des
appareils terminé
Menu_Init_Confirm
_Init
Menu_Principal
si Boutton
"Accepter"
pressé
si Boutton
"Refuser"
pressé

4.3.8 APP_SD_CARD_TASK



Dès que la carte SD est retirée, l'état sera forcément le NOP

Important: dans la configuration harmony, le bus SPI doit tourner à 10Mhz et doit être en configuration DRV_SPI_CLOCK_MODE_IDLE_LOW_EDGE_FALL.

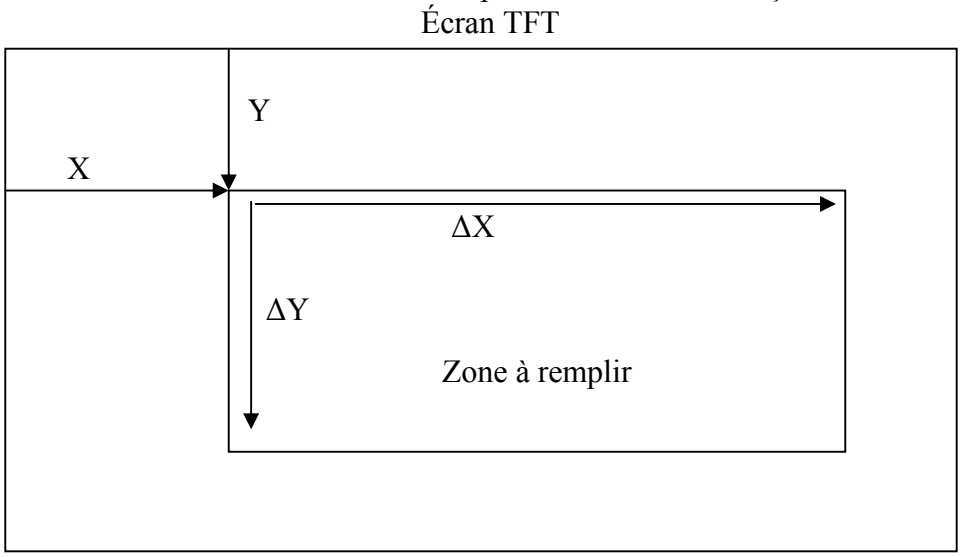
L'écran fait 320 pixels de large sur 240 de haut.

Pour initialiser l'écran il faut appeler la fonction "tft_begin" qui va faire une grosse initialisation de l'écran.

Suite à cela les fonctions qui ont été les plus utilisées sont les suivantes:

tft_fillRect(X,Y,ΔX, ΔY,Couleur)
Cette fonction remplis une zone définie d'une certaine couleur.

Le X,Y, ΔX et ΔY servent à définir la zone à remplir de couleur de la façon suivante:



tft_setTextSize (taille)
Défini la taille du texte qu'on va écrire

tft_setCursor(X,Y)
Défini la position du curseur

tft_setTextColor(Couleur)
Défini la couleur du texte

tft_drawFastHLine(X,Y,ΔX,Couleur);
Dessine une ligne horizontale de 1 pixel de largeur

tft_drawFastHLine(X,Y,ΔY,Couleur);
Dessine une ligne verticale de 1 pixel de largeur

tft_writeString(message);
Écrit sur l'écran le message

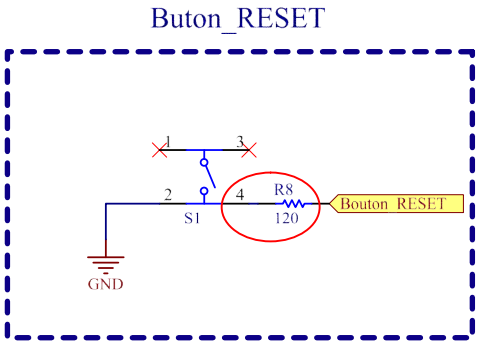
5 Corrections

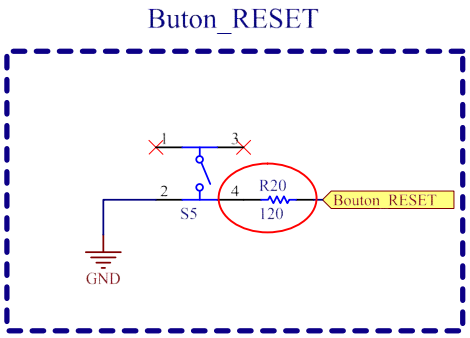
5.1 Module XBee

Footprint à l'envers.

5.2 Bouton Reset

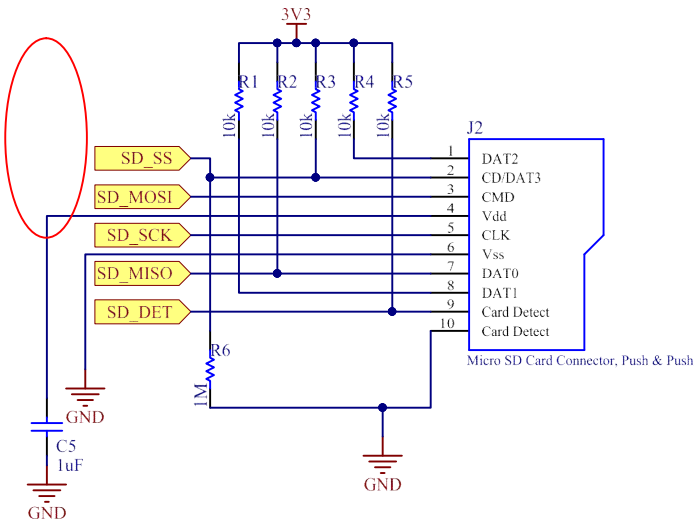
Lors de la programmation j'ai pu me rendre compte que lorsque j'uploadais mon programme en release, le programme ne se lançait pas.
Après un certain temps de recherche, j'ai constaté que j'ai mis la ligne de reset sur la mauvaise patte du bouton, ce qui fait que la ligne de reset était toujours à l'état bas.





5.3 Alimentation de la carte SD

J'ai oublié un symbole d'alimentation pour la carte SD



6 Validation selon cahier des charges

- **Station "Maitre" comprenant:** ✓

  - Un boitier
  - Un afficheur LCD minimum 4 lignes 20 colonnes
  - Boutons de commande (à impulsion ou rotatif) - doit permettre de sélectionner des menus ou valider des tickets
  - Alimentation 5V via µ-USB.
  - Emplacement pour module sans-fil -> voir projet 1623
  - Prévoir une carte SD (pour pré-configuration future, par exemple appairage des stations esclaves avec la station maitre)

  - **Firmware du microcontrôleur principal maitre** ✓

    - Appairage des différents modules esclaves ✓
    - Gestion des tickets (nombres par modules / validation des tickets) ~
    - Remise à zéro de tous les tickets ✓

- **Station Esclave (minimum 2 à monter) comprenant:** ✓

  - Un boitier
  - Boutons poussoir pour l'envoi du ticket
  - Système lumineux - 3 couleurs

    Rouge => bloqué, pas d'envoi possible (limite atteinte)
    => Clignotement: pas de lien RF
    Orange => envoi possible de ticket, ticket en cours.
    Vert => envoi possible de ticket, aucun en cours.
  - Alimentation 5V via via µ-USB.
  - Emplacement pour module sans-fil -> voir projet 1623

  - **Firmware du microcontrôleur principal esclave**

    - Envoi de ticket ✓
    - Gestion des signaux lumineux ✓

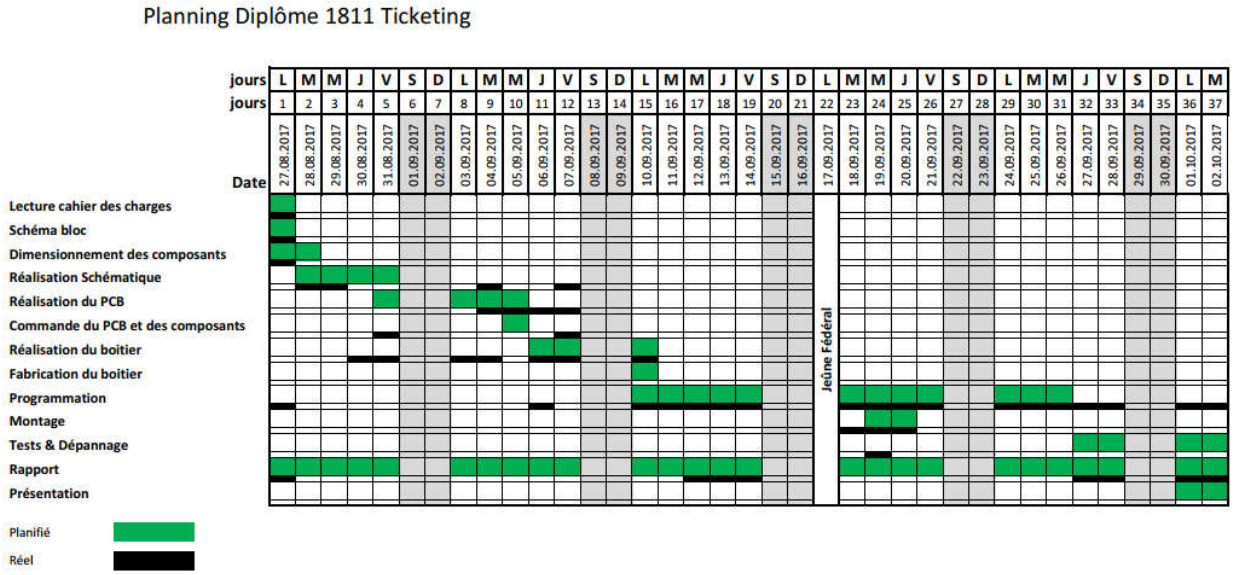
- **Firmware du module RF à mettre au point (HW fourni, FW existant Beugé)** ~

Le module RF est beugé sans comprendre pourquoi, car avant son implémentation sur mes cartes les modules communiquaient bien.

La communication entre le module maitre et le module esclave fonctionne si l'on modifie légèrement le code(car le module RF n'est pas 100% transparent)

Avec le retard dû aux problèmes liés au module RF j'e n'ai pas eu le temps de finir d'implémenter le mode de configuration, mais il sera fonctionnel pour la présentation

7 Comparaison des plannings



J'ai donc volontairement commencé par le boitier avant le PCB ce qui m'as permis de déjà faire le placement des composants principeaux avant le routage du PCB.

La programmation m'as pris beacoup plus de temps que prévu, car le module RF du XBee n'ont plus pu communiquer depuis leur implémentation à mes cartes.

8 Conclusion

Ce travail de diplôme était intéressant, c'était la première fois pour moi que je réalisais une communication sans-fil. Le fonctionnement du projet n'est pas total à cause, principalement, d'une mauvaise gestion du temps, je persistais à essayer de résoudre les problèmes modules RF alors que je n'avais pas fini de réaliser le code du module maitre et esclave.

J'ai apprécié faire un projet qui soit le plus user-friendly possible.

Lausanne, le 02.10.35 Martins David

A. Journal de travail

B. Cahier des charges

C. Listing Code Esclave

D. Listing Code Maitre

E. Listing Code Module RF

F. Procès-Verbaux

G. Mode d'emploi

H. Plannings

J. Documents de fabrication

K. Datasheet nRF904