# SAÉ 5.01 - ESE - Transmission de données à très grande distance

Rapport

Taha Aziz, Valentin Mercenaro, Guillaume Seimandi et
Alexis Wattignies

Groupe ESE A

# Sommaire

# 1 Introduction

La transmission de données à très grande distance représente un défi technique, notamment dans des environnements où les ressources énergétiques sont limitées et où la fiabilité des communications est primordiale. Le protocole WSPR (Weak Signal Propagation Reporter) légèrement modifié s'impose comme une solution pour répondre à ces contraintes. Conçu pour fonctionner avec une puissance d'émission extrêmement faible, de l'ordre de la centaine de milliwatts, WSPR permet d'étudier la propagation des ondes radio sur de longues distances, en exploitant notamment la réflexion ionosphérique. Ce protocole, développé par Joe Taylor (K1JT), prix Nobel de physique, est largement utilisé par les radioamateurs pour tester les performances des équipements dans des conditions de signal très faibles.

Dans ce contexte, notre projet à pour but de concevoir et à mettre en œuvre un système de transmission de données basé sur le protocole WSPR, en utilisant un microcontrôleur STM32L051x pour la gestion des émissions et une carte de transmission spécialement conçue pour garantir une stabilité et une précision optimales. L'objectif est de réaliser un système autonome, capable de fonctionner sur batterie pendant de longues périodes, tout en respectant les contraintes du protocole WSPR, notamment en termes de synchronisation temporelle et de gestion de l'énergie.

Ce rapport détaille les différentes étapes du projet, depuis la conception du système de transmission jusqu'à la réalisation des tests en conditions réelles. Nous aborderons notamment la conception du module de génération du signal et de l'amplificateur, l'intégration du microcontrôleur STM32, la gestion des modes basse consommation, et la mise en place d'un logiciel embarqué capable de générer et d'envoyer des messages WSPR de manière autonome. Enfin, nous présenterons les résultats des tests effectués, ainsi que les améliorations potentielles pour optimiser les performances du système.

# 2 Présentation du protocole WSPR

Le protocole WSPR (Weak Signal Propagation Reporter) est un système de communication radio numérique conçu pour la transmission de messages à très faible puissance sur de longues distances.

Le protocole WSPR a plusieurs objectifs principaux :

- **Transmission à faible puissance :** Il permet de transmettre des messages avec une puissance d'émission très faible (quelques milliwatts), ce qui le rend idéal pour des applications où l'énergie est limitée.Développé par Joe Taylor (K1JT), un prix Nobel de physique, WSPR est particulièrement utilisé par les radioamateurs pour étudier la propagation des ondes radio et tester les performances des équipements dans des conditions de signal extrêmement faible
- **Étude de la propagation des ondes :** WSPR est utilisé pour analyser comment les signaux radio se propagent sur de longues distances, en particulier via la réflexion sur l'ionosphère.

De plus, il est défini par plusieurs caractéristiques techniques :

- **Bandes de fréquences :** WSPR opère sur des bandes de fréquences radioamateur allant de 1,8 MHz (bande des 160 mètres) à 28 MHz (bande des 10 mètres). La bande des 30 mètres (10,1 MHz) est souvent privilégiée pour les transmissions à très longue distance.
- **Durée des transmissions :** Chaque transmission WSPR dure environ 110,6 secondes.
- **Format des messages :** Un message WSPR contient trois éléments principaux. Ces éléments sont décris dans le tableau ci-dessous.

| Élément | Longueur | Description |
|---|---|---|
| Indicatif d'appel (Call Sign) | 6 caractères | L'indicatif de la station émettrice. Si l'indicatif est plus court, il est complété par des espaces. |
| Localisateur Maidenhead (Grid Locator) | 4 caractères | La position géographique de la station, exprimée en notation Maidenhead (par exemple, "JN33"). |
| Puissance d'émission (dBm) | 2 chiffres | La puissance du signal en dBm (décibels par rapport à 1 milliwatt). Par exemple, "37" correspond à 5 watts. |

Les messages WSPR sont encodés sur 162 symboles, chacun représentatnt une tonalité (ou une fréquence) spécifique propre à la modulation utilisée : le 4-FSK (Frequency Shift Keying).

Avant la transmission, les 162 symboles sont réorganisés selon un schéma d'entrelacement prédéfini (interleaving). Par exemple, au lieu de transmettre les symboles dans l'ordre [S1, S2, S3, ..., S162], ils sont réorganisés en [S1, S54, S107, S2, S55, S108, ...]. Cette réorganisation disperse les symboles adjacents dans le temps.

**Fonctionnement de WSPR** Le fonctionnement de WSPR repose sur trois étapes. Tout d'abord, la génération du message consiste à encoder les informations essentielles, telles que l'indicatif d'appel, le localisateur et la puissance d'émission, en une séquence de symboles. Ensuite, le message est transmis sur une fréquence spécifique pendant une durée d'environ 110,6 secondes. Cette transmission est synchronisée pour débuter au commencement des minutes paires (0, 2, 4, etc.). Enfin, les stations réceptrices utilisent des logiciels spécialisés pour décoder les messages WSPR. Les données ainsi obtenues sont partagées sur des bases de données en ligne, comme WSPR Rocks, permettant aux utilisateurs de visualiser et d'analyser la propagation des signaux à faible puissance.

**Avantages du Protocole WSPR** Le protocole WSPR présente plusieurs avantages. Tout d'abord, sa faible consommation d'énergie permet des transmissions à très faible puissance, ce qui est idéal pour les stations autonomes fonctionnant sur batterie, assurant une utilisation prolongée. Ensuite, grâce à la réflexion ionosphérique, les signaux WSPR peuvent atteindre de très grandes distances, parcourant des milliers de kilomètres avec une puissance d'émission minimale. Enfin, la simplicité du protocole facilite sa mise en œuvre, notamment dans des microcontrôleurs faible consommation comme ceux de la gamme STM32L0x.

# 3 « Proof of concept » : émission de trame WSPR

Comme son nom l'indique, l'objectif d'un « Proof of concept » est de réaliser une version volontairement simplifiée du projet pour s'assurer de sa faisabilité. L'architecture utilisée est schématisée dans la figure 3.1.2. Ici, le but a été d'émettre des trames WSPR et d'être reçu.

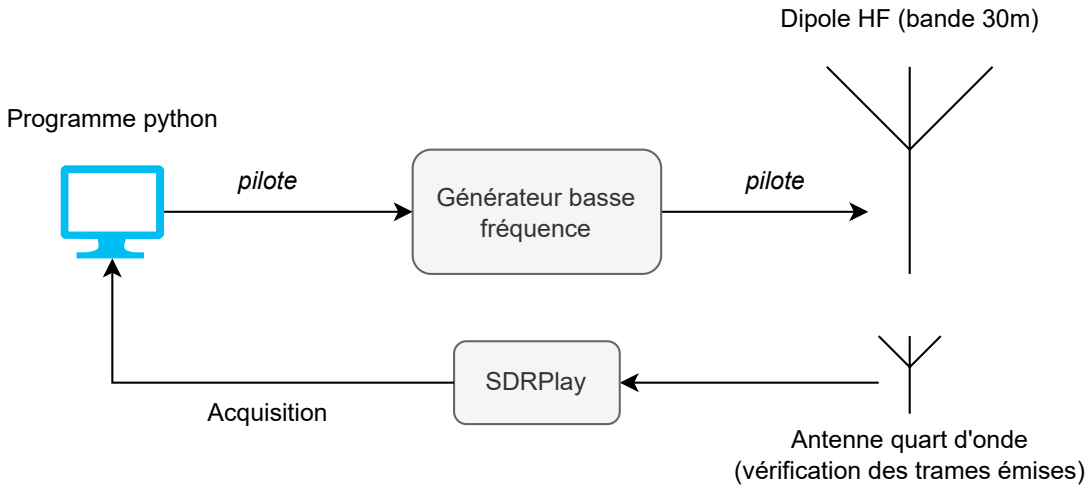



FIGURE 3.0.1 – Architecture du « Proof of concept »

Il est à noter que cette étape a déjà été réalisée et décrite dans le compte-rendu de l'année dernière. Ainsi, le présent document n'entrera pas dans les détails. Il se contentera de décrire le protocole de la manipulation qui à permis de valider le « Proof of concept ».

## 3.1 Conception du programme Python

L'objectif du programme Python est de piloter un GBF (Générateur Basse Fréquence) via le réseau TCP/IP. Pour ce faire, il exploite la librairie pyvisa et communique avec l'appareil sous forme de commande. Les commandes respectent la norme SCPI (Standard Commands for Programmable Instruments). La figure 3.1.1 présente les trois commandes principales utilisées dans le programme. Elles permettent respectivement de désactiver la sortie du channel 1, de « set » une fréquence et d'activer la sortie du channel 1.

```
C1:OUTPUT OFF
C1:BSWV FRQ, <freq>
C1:OUTPUT ON
```

FIGURE 3.1.1 – Exemple de commandes SCPI

Ensuite, il ne reste qu'à mettre en place l'algorithme qui permet d'émettre en respectant le protocole WSPR.

Plusieurs points sont à considérer. Dans un premier temps, les trames doivent être envoyées dans la première seconde d'une minute paire, de manière non-consécutive et avec une fréquence d'envoi inférieur à 20%. Ainsi, la gestion du temps et de la synchronisation de l'ordinateur,

qui exécute le programme, avec l'heure international est primordiale.

D'autre part, le protocole WSPR est modulé en 4-FSK, avec un shift de 1,46 Hz. D'où l'importance de convertir le message en symbole pouvant être envoyé. Á ce stade, un encoder en ligne [1] est utilisé, mais à terme la conversion du message utile en symbole pouvant être envoyé sera faite en local. Cette partie du projet est abordée plus loin dans ce document.

Tout semble prêt. La trame à envoyer est présentée ci-dessous. Cette trame est convertie en une liste de symboles ; soit une liste de chiffres compris en 0 et 3.


FIGURE 3.1.2 – Trame WSPR considérée

## 3.2 Algorithme d'envoi

L'algorithme implémenté en Python est présenté dans la figure 3.2.1

1. WSPR Code Generator - https://swharden.com/software/wspr-code-generator/

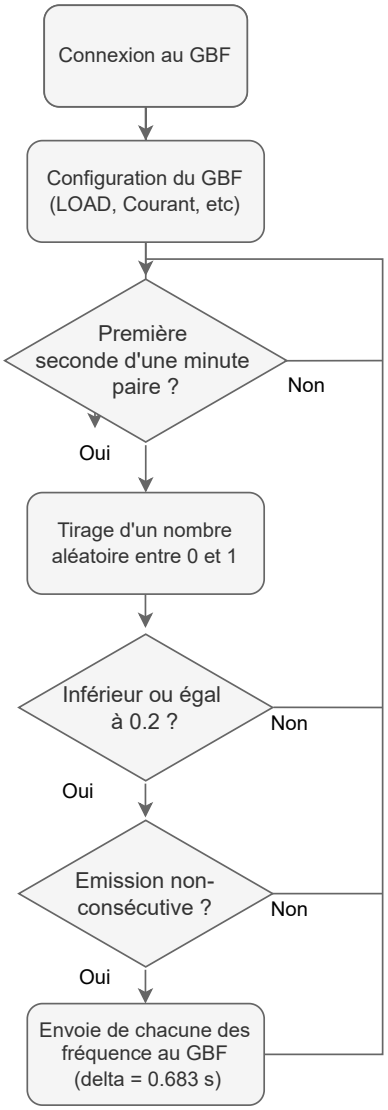



FIGURE 3.2.1 – Algorithme d'envoi

## 3.3 Synthèse des résultats

Une fois l'architecture décrite dans la figure 3.1.2 mise en place, le système a émis entre 12h58 et 21h40 le 19/09/2024. De plus, il a été reçu un total de 25 fois sur 4 stations différentes. La figure 3.3.1 [2] montre, en blanc, les stations ayant reçu le signal émis.

2. WSPR Rocks - https ://wspr.rocks/


FIGURE 3.3.1 – Carte des stations ayant reçu les trames WSPR

En conclusion, le « Proof of concept » est un succès. Il ouvre la voie au développement du projet.

# 4 Carte de Transmission de données WSPR

## 4.1 Cahier des charges de la carte de transmission

Cette année la carte de transmission a pour objectif de répondre à plusieurs exigences techniques et fonctionnelles dans le cadre du projet de transmission de données à très grande distance.

### 4.1.1 Solution STM32

Le choix du microcontrôleur STM32L051x est motivé par sa polyvalence, sa faible consommation d'énergie et ses performances élevées. Ces caractéristiques sont essentielles pour un système de transmission radio à bas débit et très longue distance, où l'efficacité énergétique et la stabilité du signal sont primordiales.

### 4.1.2 Etude de consommation

La carte doit être conçue pour minimiser la consommation d'énergie, notamment grâce à la gestion des modes de veille (SLEEP MODE , STANDBY MODE et STOP MODE) et des réveils programmés via l'horloge interne (RTC). Cela permet de prolonger la durée de vie de la batterie et de réduire l'impact énergétique du système.

### 4.1.3 Gestion des réveils suivant l'horloge interne

La gestion des réveils est assurée par l'horloge interne (RTC) du STM32. Cette horloge temps réel est cadencée par un quartz de 32.768 KHz.

Cette fonctionnalité permet de planifier des réveils périodiques pour l'envoi des messages WSPR, tout en maintenant la carte en mode veille le reste du temps pour économiser l'énergie.

### 4.1.4 Génération stable et précise du signal radio

La génération du signal radio est assurée par le module qui sera créé avec la PLL SI5351 (modèle SI5351A-B-GTR) et le TCXO 536L25001IT5O (voir section 4.2.1 Conception du module de génération du signal), qui permet de produire un signal stable et précis sur la bande des 30 mètres. La précision du signal est cruciale pour garantir une transmission fiable et sur de longues distances, en utilisant la réflexion ionosphérique. La faible dérive du signal est également cruciale pour garantir une transmission en accord avec le protocole WSPR. En effet, la bande de fréquence utilisée s'étend de 10.1398 MHz à 10.1402 MHz et les fréquences correspondantes aux quatre symboles de la modulation 4-FSK sont espacées de 2 Hz. Ainsi, le système doit garantir une dérive minimale et maximiser la précision.

### 4.1.5 Amplification du signal

Il faut choisir un amplificateur adapté à notre application RF soit un amplificateur adapté aux très hautes fréquences. Il sera alimenté en 12V, devra fournir une puissance maximale en sortie tout en garantissant un spectre de signal propre, pour un signal d'entrée carré d'amplitude 3.3V, avec une atténuation suffisante des harmoniques indésirables, en particulier une différence d'au moins 47 dB entre la fondamentale et la deuxième harmonique.

## 4.2 Conception du système de transmission

### 4.2.1 Conception du module de génération du signal

**Introduction** Cette sous-section aborde la conception du module de génération du signal. La création de ce module est conditionnée par plusieurs contraintes inhérentes au protocole, mais également au milieu dans lequel sera amené à évoluer le système.

Les mots d'ordre sont précision et stabilité. En effet, le protocole WSPR exploite une modulation 4-FSK sur une bande de fréquence allant de 10.1398 MHz à 10.1402 MHz et un intervalle entre chaque symbole de maximum 2 Hz. Ainsi, le module devra avoir une précision suffisante pour être en mesure de différencier les quatre symboles de la modulation, mais également avoir une bonne stabilité pour ne pas dévier en fréquence : la bande de fréquence utilisée ayant une largeur de 400 Hz. De plus, les émissions sont longues ainsi, la contrainte de stabilité s'applique à court-terme (le temps de l'émission d'une trame) et à long-terme.

En outre, le système est amené à être embarqué, entre autres, sur une station de mesure en mer. Ainsi, le module doit avoir une faible déviation face aux changements de température ou plus généralement de milieu.

**Choix de conception** Pour respecter les contraintes de conception du module, celui-ci est basé sur un circuit intégré : la PLL (Phase Lock Loop) SI5351A en boitier 10-MSOP. Cet IC (Intergrated Circuit) est cadencé par un oscillateur à quartz de 25 MHz régulé en température : le TCXO 536L25001IT5O (Temperature Controlled Oscillator). En effet, l'utilisation d'un oscillateur controllé en température permet de résoudre le problème de déviation en fréquences dû à la variation de température.

De nombreux circuits exploitant le SI5351 sont disponibles à la vente comme celui proposé par AdaFruit ; c'est sur ce dernier que la conception du module PLL va se baser. En effet, ce circuit utilise une PLL programmable via un bus I2C. Cependant, le quartz utilisé n'est pas régulé. D'où la nécessité de créer un autre circuit qui implémente un TCXO.

**Schéma électrique du module** La figure 4.2.1 montre le schéma électrique du module. [3]. Contrairement, à l'implémentation classique proposée par AdaFruit [4], le module utilise un oscillateur à quartz et non uniquement un quartz. Ainsi, le schéma diffère légèrement du circuit de référence pour « bypass » le mécanisme d'oscillation interne inclut dans le SI5351A. À ce titre la broche XB de la PLL est laissée en l'air.

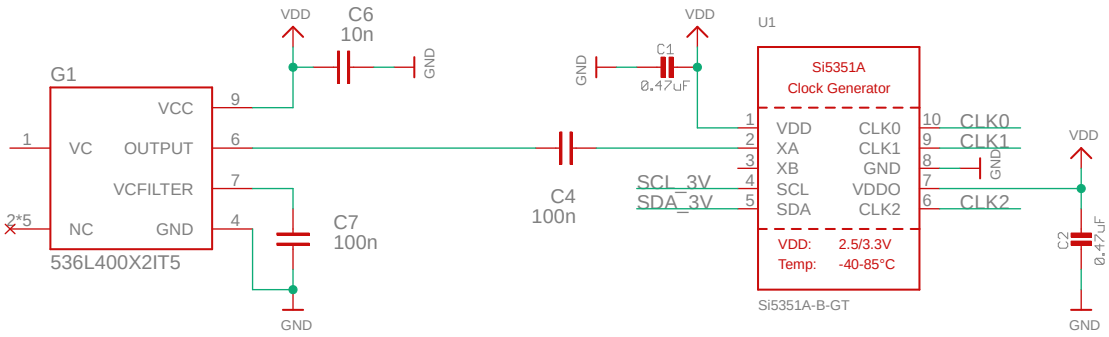



FIGURE 4.2.1 – Schéma électrique du module de génération du signal

**Routage du circuit** La figure 4.2.2 présente le routage du circuit. Il est à noter que la sortie des trois signaux du module (CLK0, CLK1 et CLK2) se fait sur des ports SMA.

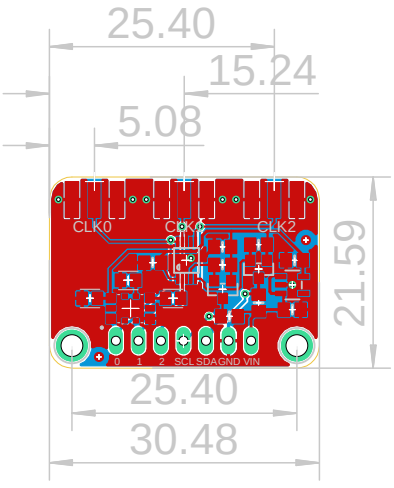



FIGURE 4.2.2 – Routage du module de génération du signal

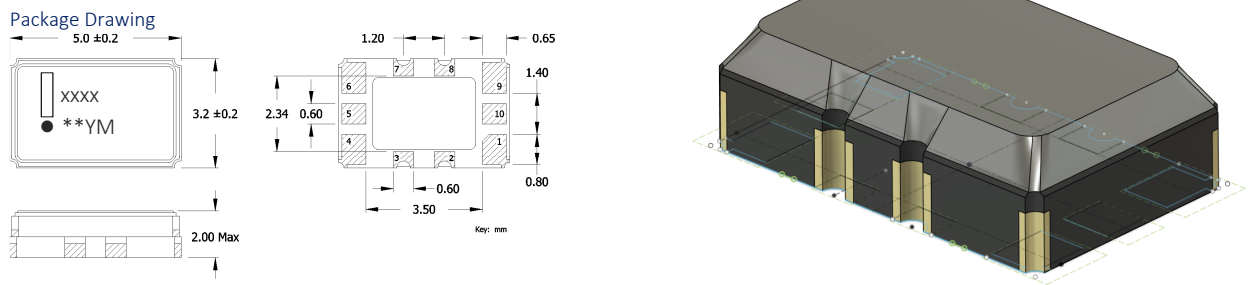
3. Pour une meilleure clarté seul le TCXO et le SI5351 sont montrés, l'ensemble du schéma électrique est disponible dans le dossier joint à ce rapport

4. Adafruit-SI5351A-Clock-Generator-Breakout-PCB : https://github.com/adafruit/Adafruit-Si5351A-Clock-Generator-Breakout-PCB

**Création d'un support de test** Finalement, afin de garantir le bon fonctionnement du module, il a été choisi de créer un support via impression 3D pour réaliser des tests. Ces tests sont décrits dans la section Qualification du module de génération du signal.

L'objectif de ce support est de maintenir le module en face d'un capteur de température pour mesurer la déviation en fréquence du module en fonction de la température. Les composants tels que le TCXO ou le capteur de température ayant une inertie thermique faible à cause de leur faible taille, il est nécessaire que ceux-ci soient alignés et maintenus à distance constante pendant la manipulation. C'est pourquoi, il a fallu dans un premier temps créer un modèle 3D du module de génération du signal et du capteur de température.

**Création du package du TCXO** Dans la majorité des cas, les packages (modèles 3D) des composants utilisés sont disponible en ligne. Toutefois, le TCXO fait exception, il a donc fallu le créer. La figure 4.2.3 montre la package du TCXO 536L25001IT5O avec un extrait de la documentation technique montrant son encombrement.

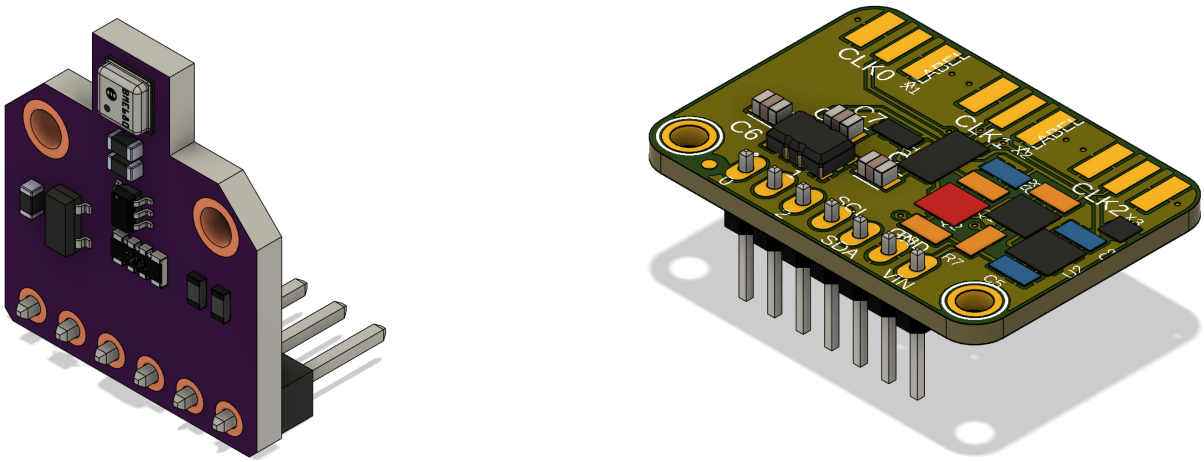



(a) Schéma d'encombrement du TCXO

(b) Modèle 3D du package du TCXO

FIGURE 4.2.3 – Représentation du TCXO : schéma d'encombrement et modèle 3D

**Modèle 3D du module PLL et du thermomètre I2C** La figure 4.2.4 montre les modèles 3D du module de génération du signal, sans les connecteurs SMA, et de la carte de mesure de température I2C : le BME680.



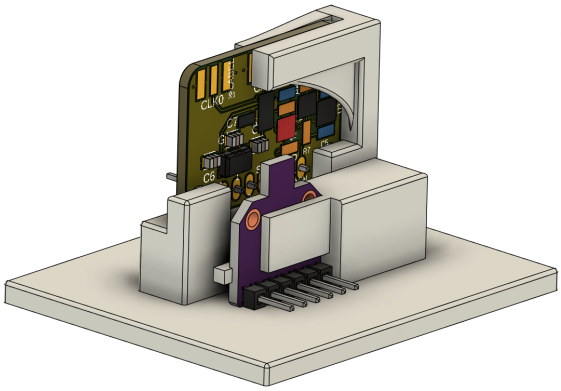

(a) Modèle 3D du module d'acquisition de température

(b) Modèle 3D du module de génération du signal

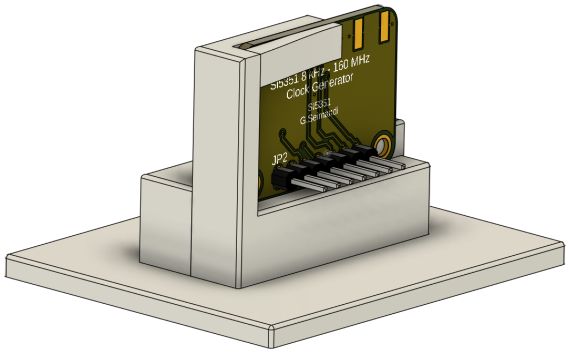
FIGURE 4.2.4 – Modèles 3D du module de génération du signal et d'acquisition de température

-

La figure 4.2.5 montre le support complet avec le module PLL et le module d'acquisition de la température. Il est à noter que le support a été conçu de manière à ne pas avoir de vis. Les modules sont maintenus en place avec des « languettes » qui joue sur l'élasticité du plastique.



(a) Modèle 3D du support de test (Vue de derrière)




(b) Modèle 3D du support de test (Vue de devant)

FIGURE 4.2.5 – Visualisation 3D du support de test

### 4.2.2 Conception du module d'amplification

**Les différents types d'amplificateur** Dans le projet l'amplificateur, abrégé ampli, se place juste avant l'antenne, il amplifie le signal de la PLL avant de le transmettre dans l'antenne afin de diffuser le signal. Cela permet au signal d'avoir une très large portée et d'être capté jusque dans des pays éloignés. Il existe différents types d'amplificateur.

- **Audio** : Classes A, AB, B, D, G, H et T
- **RF** : Classes C et E

Chaque ampli possède des spécificités et des applications différentes, dans notre projet l'objectif est d'émettre en HF soit à 10,14 MHz. Le choix d'ampli se tournera donc vers un classe C ou E.

**L'amplificateur de classe C** Les amplificateurs de classe, C sont conçus pour fonctionner avec des transistors qui ne conduisent que pendant une petite partie du cycle du signal d'entrée.

Cette conduction partielle est obtenue en polarisant les transistors en dessous de leur seuil de conduction, ce qui signifie qu'ils ne s'activent que lorsque le signal d'entrée dépasse un certain niveau. Le signal de sortie est fortement déformé, un circuit supplémentaire est nécessaire pour reconstituer le signal.

**L'amplificateur de classe E** Le signal d'entrée est modulé en impulsions, l'ampli utilise un circuit de résonance qui est soigneusement ajusté pour que les transitions entre les états de commutation se produisent à des moments où la tension ou le courant est nul. Cela réduit encore davantage les pertes d'énergie.

Optimisé pour les radiofréquences (RF) avec des circuits de résonance pour minimiser les pertes. Transforme des V en W pour un rendement bien supérieur aux classes C.

C'est pourquoi, l'amplificateur choisit pour être implanté dans le projet est un amplificateur de classe E. [5]

Le signal d'entrée est le signal de la PLL. Le signal d'entrée est donc un signal carré d'amplitude 3,3V, d'offset 1,65V et de fréquence 10,14 MHz ; le système est alimenté en 12V. L'objectif de cet ampli étant de générer le plus de puissance possible en sortie.

Ce résultat sera observé sur un analyseur de spectre, à la place de l'antenne sur la figure 4.2.6, la fondamentale doit être le plus haut possible tout en ayant un écart d'au minimum 47 dB avec la 2e harmonique. La fondamentale doit avoir son pique à une fréquence de 10,14 MHz et la 2e harmonique à une fréquence d'environ 20 MHz.

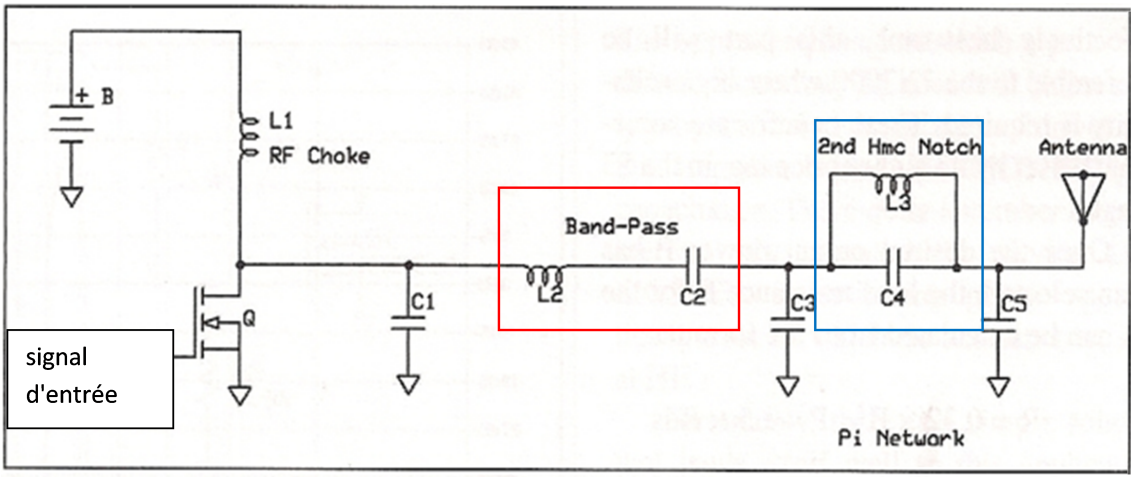



FIGURE 4.2.6 – Schéma de l'amplificateur de classe E

Un ampli de classe E se réalise comme en figure 4.2.6, il est constitué de différentes parties :

- Le ou les transitor MOSFET qui sert/servent à moduler le signal
- `L2` et `C2` qui forment un filtre passe-bande à la fréquence de la fondamentale permettant ainsi d'amplifier celle-ci au maximum.
- `L3` et `C4` qui forment un filtre coupe-bande à la fréquence de la 2e harmonique, réduisant fortement la puissance du signal à cette fréquence. Cela assure l'écart de 47 dBm entre la fondamentale et la 2e harmonique.
- `C3` et `L5` qui font office d'un 2e filtre-bas de sûreté.

5. La conception et l'analyse de l'amplificateur s'appuient sur le document : Nathan O. Sokal, *Class-E RF Power Amplifiers*. QEX, 2001.

- `C1` contribue à ajuster la forme d'onde de tension et de courant pour réduire les chevauchements, ce qui diminue les pertes de commutation.

**Fonctionnement des MOSFET**

**À quoi sert le déphasage des MOSFET ?** Dans les amplificateurs de classe E, le déphasage des MOSFET est une étape cruciale : le principal objectif de ce déphasage est de minimiser les pertes de puissance dans les transistors MOSFET. Cela est particulièrement vrai lors des commutations (état « Power on » vers « Off » et inversement). Pour la bonne raison que, dans ces moments-là, la puissance consommée par le transistor est considérablement importante.

Le déphasage permet entre autres de :

- Éviter les pertes de commutation : en retardant l'arrivée de la tension jusqu'à ce que la charge soit nul, et en raccourcissant la charge jusqu'à que le courant recule à zéro, on s'assure de réduire au maximum les pertes durant la charge et décharge du condensateur de grille du MOSFET.
- Obtenir deux formes d'onde optimisées : la forme d'onde de la tension et du courant sont naturellement sculptées par le réseau de charge. Elle garantit toujours que la tension est nulle quand la charge est la plus importante et que le courant est nul lorsque la tension est la plus grande.
- Analyser le compromis optimal du déphasage : il existe d'ailleurs un compromis, si l'on regarde l'efficacité. Suivant le déphasage choisi, vous pourrez voir que l'efficacité peut glisser entre 85 et 95%, le tout, autrement, étant perdu sous forme de chaleur dans vos MOSFET, les calories ne jouant pas un rôle de haut rang dans la grille du calcul de l'efficacité.

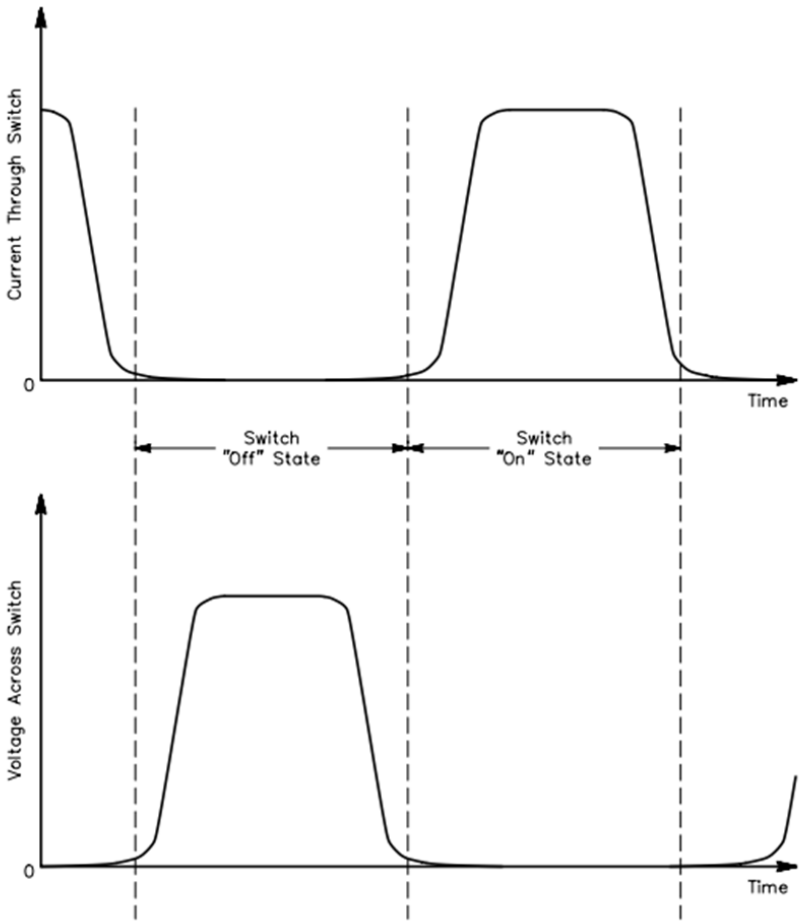



FIGURE 4.2.7 – Représentation conceptuelle de la tension et du courant dans le transistor

**Comment le déphasage est-il effectué ?** Le déphasage est réalisé grâce au réseau de charge (composé d'inductances et de capacités) qui est soigneusement conçu pour façonner les formes d'onde de tension et de courant. Voici les étapes clé pour y parvenir :

1. **Conception du réseau de charge** :
   - Le réseau de charge est conçu pour retarder la montée de la tension jusqu'à ce que le courant ait chuté à zéro.
   - Il assure également que la tension retombe à zéro avant que le courant ne commence à augmenter.
   - Ce réseau est composé d'éléments tels que des inductances (`L1`, `L2`) et des capacités (`C1`, `C2`) qui sont calculés pour obtenir les formes d'onde souhaitées.
2. **Formes d'ondes cibles** :
   - État "on" : Le transistor agit comme un interrupteur fermé, avec une tension presque nulle et un courant élevé.
   - État "off" : Le transistor agit comme un interrupteur ouvert, avec une tension élevée et un courant nul.
   - Transitions : Les temps de commutation sont gérés pour que la tension et le courant ne soient jamais simultanément élevés.

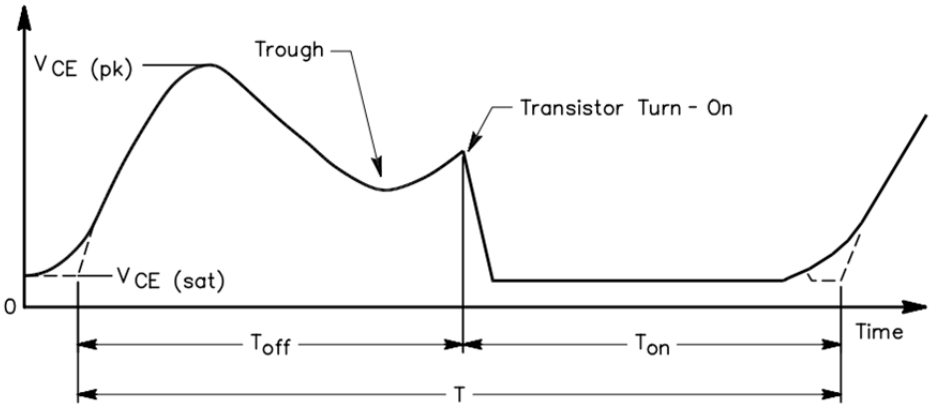



FIGURE 4.2.8 – Exemple de signal mal accordé

3. **Ajustement des composants** :
   - Les valeurs des composants (`L1`, `L2`, `C1`, `C2`) sont ajustées pour obtenir les formes d'onde optimales. Par exemple, augmenter `C1` déplace le creux de la forme d'onde de tension vers le haut et la droite, tandis qu'augmenter `C2` le déplace vers le bas et la droite.
   - Le choix du facteur de qualité chargé du réseau de charge influence également le déphasage et l'efficacité (voir figure 4.2.9).

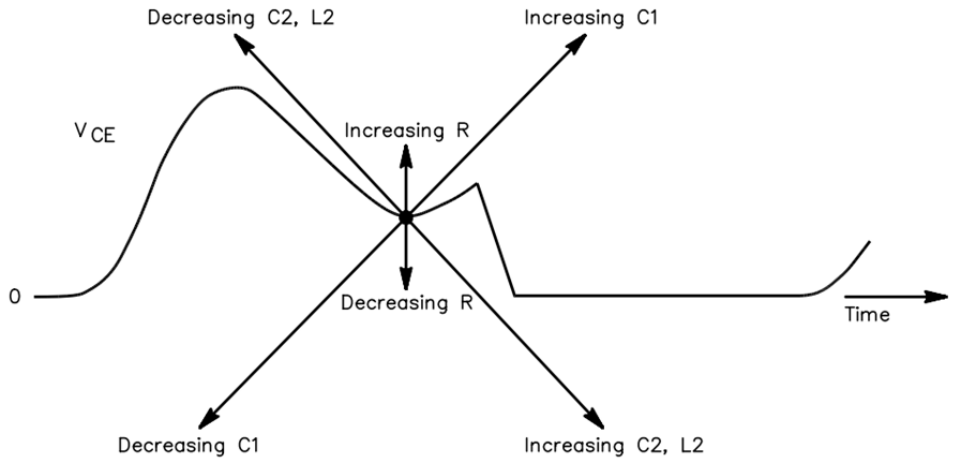



FIGURE 4.2.9 – Impact de l'ajustement des composants

**Puissance maximale théorique et puissance actuelle produite**

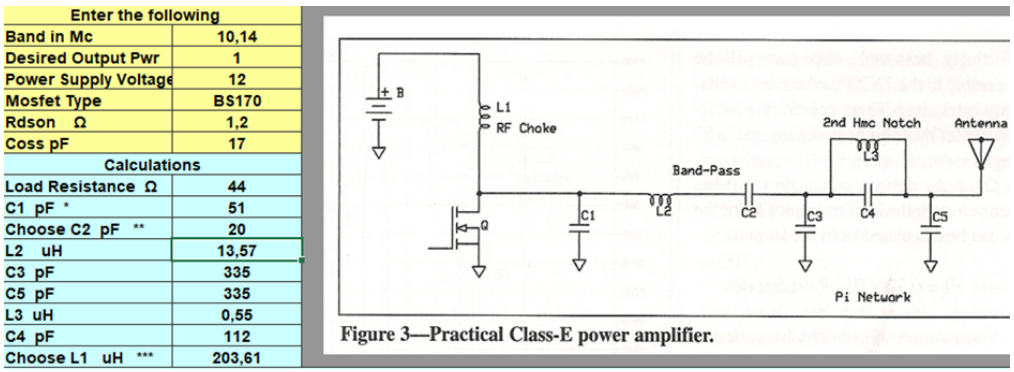
$$\mathrm{P_{max}} = \frac{\mathrm{V_{supply}^2}}{2 \cdot \mathrm{Z_{load}}}$$

Avec, $\mathrm{V_{supply}} = 12\mathrm{V}$ et $Z_{load} = 50\,\Omega$

$$\mathrm{P_{max}} = \frac{12^2}{2 \cdot 50} = \frac{144}{100} = 1{,}44\ \mathrm{W}$$

En théorie, un amplificateur parfait devrait pouvoir générer 1,44 W en sortie.

**Test de fonctionnement de l'ancienne carte** Dans un premier temps, le travail réalisé par l'équipe de l'année passée a été étudié. Plusieurs logiciels sont à disposition afin de réaliser l'étude théorique du système. Dans un premier temps un fichier `Excel` permettant de calculer les composants. Les calculs sont très longs et complexes, or ces calculs ne nous intéressent pas dans cette étude et prendrait trop de temps à être réalisé manuellement en plus d'inclure de potentielles erreurs de calcul. Les valeurs sont entrées dans le `Excel` comme en figure 4.2.10



| Enter the following | |
|---|---|
| Band in Mc | 10,14 |
| Desired Output Pwr | 1 |
| Power Supply Voltage | 12 |
| Mosfet Type | BS170 |
| Rdson Ω | 1,2 |
| Coss pF | 17 |
| **Calculations** | |
| Load Resistance Ω | 44 |
| C1 pF * | 51 |
| Choose C2 pF ** | 20 |
| L2 uH | 13,57 |
| C3 pF | 335 |
| C5 pF | 335 |
| L3 uH | 0,55 |
| C4 pF | 112 |
| Choose L1 uH *** | 203,61 |



FIGURE 4.2.10 – Excel pour le calcul des composants

Une fois les valeurs des composants calculés un logiciel de simulation de système, `LTSpice`, est utilisé afin de réaliser le circuit et pouvoir observer les théoriques résultats qui seraient obtenus.

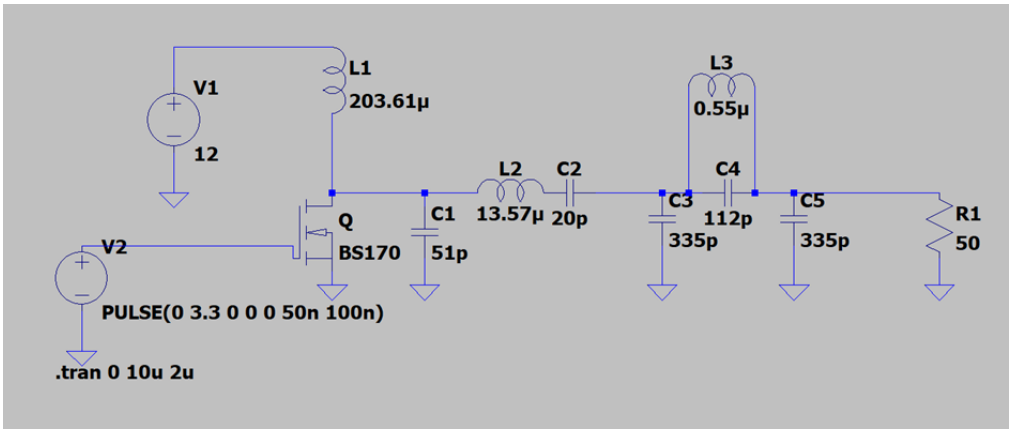




FIGURE 4.2.11 – Schéma de l'amplificateur sous `LTSpice`

Une fois la simulation lancée, le résultat obtenu est le suivant :

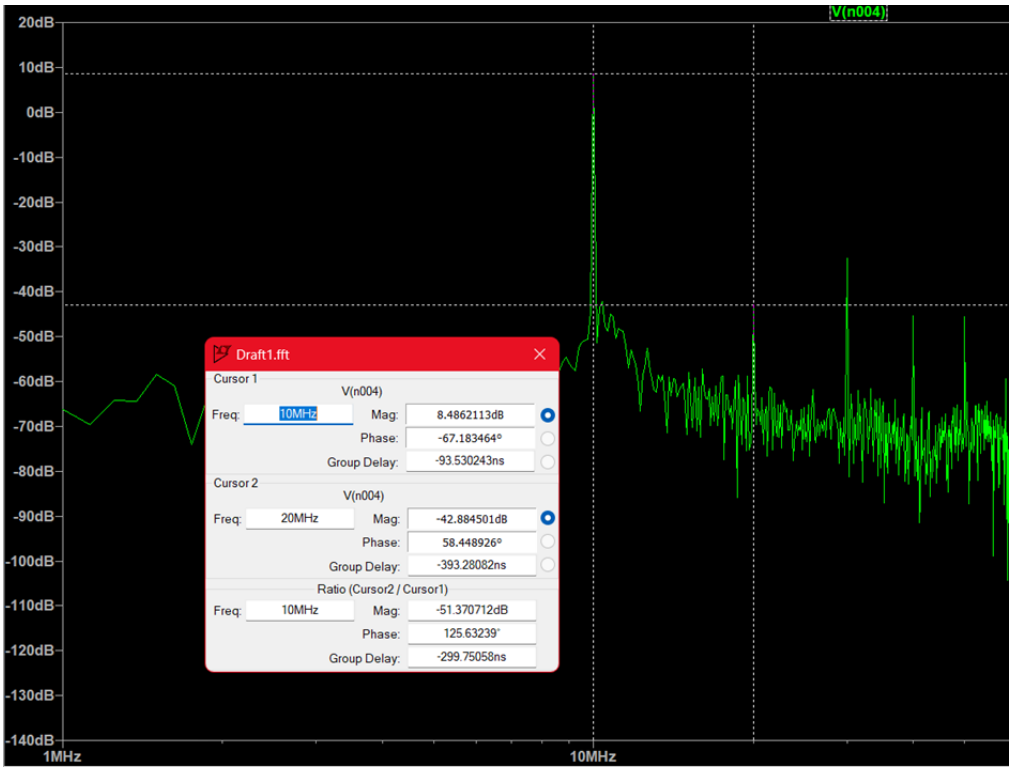



FIGURE 4.2.12 – Simulation du spectre de l'amplificateur sous `LTSpice`

D'après la simulation, à 10.14 MHz la puissance obtenue est d'environ 8,5 dB avec bien un écart de 47 dB avec la 2e harmonique.

Une fois, la simulation réalisée la carte est testé afin de constater les différences.

Le matériel utilisé est une alimentation, un GBF, un analyseur de spectre, un coupleur et une charge. Le GBF sert à simuler le module de génération du signal soit un signal carré de 3.3V avec un offset de 1.65V. L'alimentation réglée sur 12V continue permettra en plus de calculer la consommation du système.

Afin de réaliser la mesure et ne pas endommager l'analyseur de spectre, il est nécessaire d'installer en coupleur 20 dB en sortie du système comme le montre la figure 4.2.13.

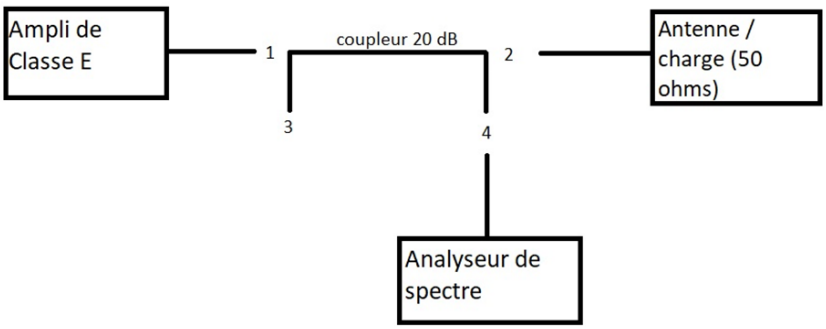



FIGURE 4.2.13 – Schéma de cablage du coupleur

Le coupleur peut avoir de nombreuses utilisations, ici, il servira uniquement à réaliser des

mesures durant le fonctionnement du système. Le signal de sortie est injecté en 1, en 2 est placé une charge de 50 Ω. Une autre charge est placée en 3, l'analyseur de spectre lui est placé sur la sortie 4. Cette installation permet de réduire de 20 dB la puissance injectée dans l'analyseur de spectre.

Lors de la lecture des résultats, il faut ajouter 20 dB qui ont été atténués par le coupleur, ici en figure 4.2.14 la puissance mesurée sur la fondamentale est de 22,01 dB. Une différence considérable est observable avec la simulation.

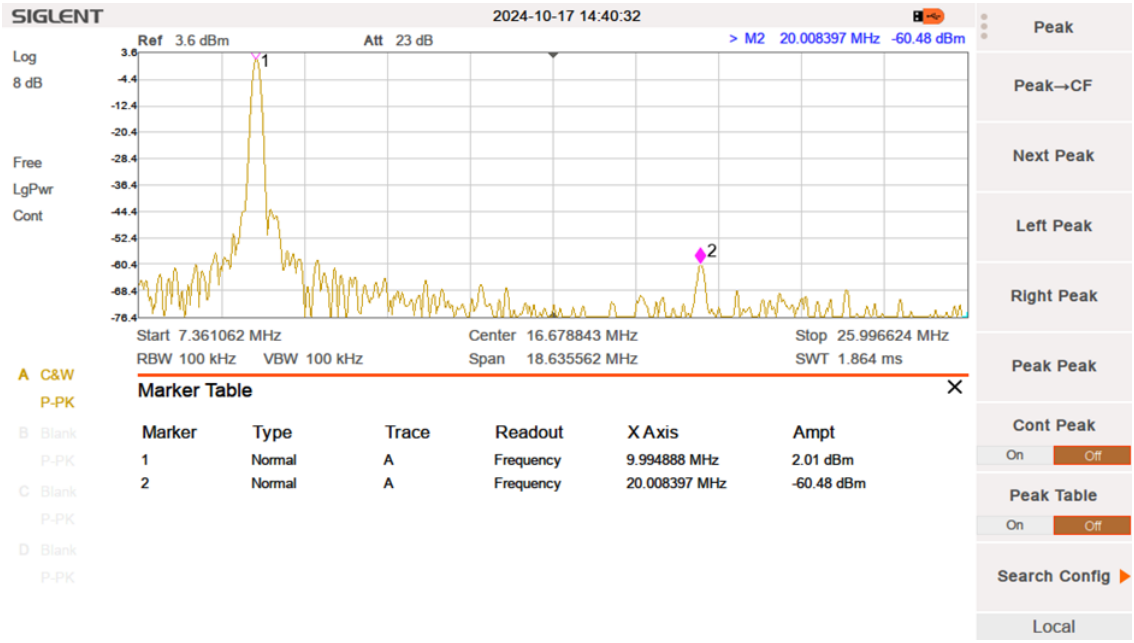



FIGURE 4.2.14 – Mesure à l'analyseur de spectre de la puissance de sortie de l'ancienne carte

**Modification des valeurs** Une fois le sujet maîtrisé, il convient à présent de concevoir un système optimisé répondant à nos exigences. La première étape consiste à recalculer les valeurs des composants en s'appuyant sur les données du fichier `Excel`.

Pour une puissance désirée de 1,44W et une alimentation de 12V les valeurs des composants sont tels qu'indiqués en figure 4.2.15.

| Enter the following | |
|---|---|
| Band in Mc | 10,1 |
| Desired Output Pwr | 1,44 |
| Power Supply Voltage | 12 |
| Mosfet Type | BS170 |
| Rdson Ω | 1,2 |
| Coss pF | 17 |
| **Calculations** | |
| Load Resistance Ω | 30 |
| C1 pF * | 83 |
| Choose C2 pF ** | 45 |
| L2 uH | 6,38 |
| C3 pF | 407 |
| C5 pF | 407 |
| L3 uH | 0,46 |
| C4 pF | 136 |
| Choose L1 uH *** | 95.63 |

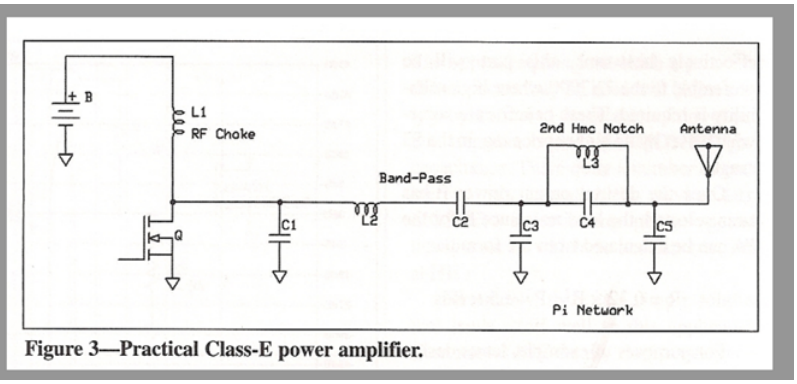


Figure 3—Practical Class-E power amplifier.

FIGURE 4.2.15 – Calcul des composants sous `Excel`

Les valeurs sont appliquées au modèle `LTSpice`, le résultat obtenu après simulation est représenté en figure 4.2.16.

Sur la fondamentale, le pic de puissance le plus haut la valeur est de 7,43 dB, soit un peu moins que la valeur obtenue via la simulation avec les valeurs de l'équipe précédente. Cependant, il y a bien l'écart de 47 dB entre la fondamentale et la 2e harmonique la plus haute.

Malgré des résultats de simulation moins concluants, les valeurs calculées à l'aide du fichier `Excel` seront appliquées à la carte, car elles ont été déterminées pour permettre le développement d'une puissance plus importante. Par ailleurs, lors des travaux précédents, une divergence a été observée entre les résultats pratiques et les simulations, ce qui justifie de privilégier les valeurs calculées pour cette réalisation.

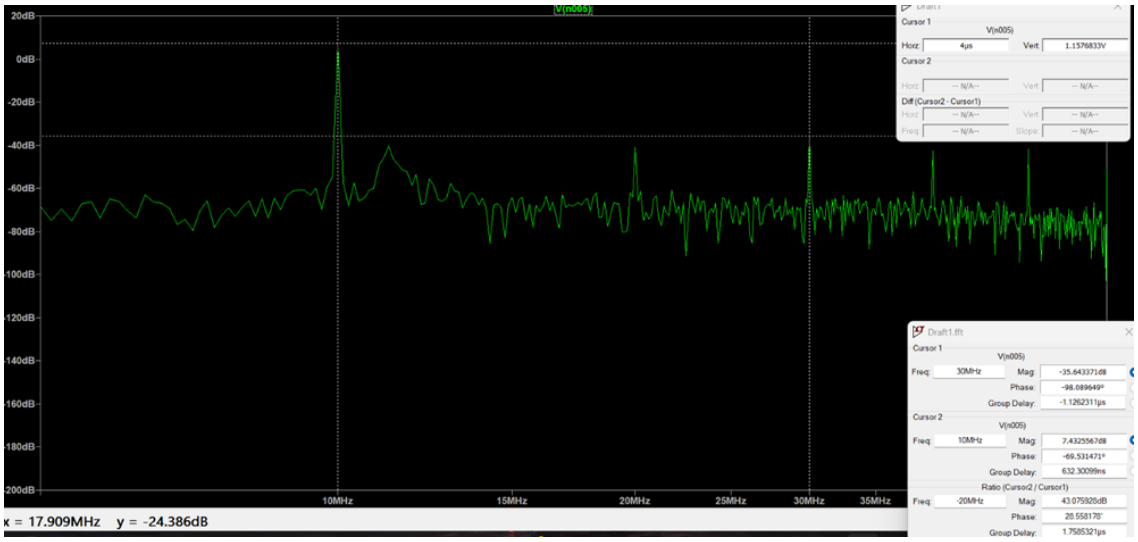



FIGURE 4.2.16 – Simulation du spectre de l'amplificateur sous `LTSpice`

**Réalisation de la carte** La première étape de la réalisation est la modification du PCB sur le `KiCAD` de la carte. Des éléments peuvent être améliorés que cela soit pour réduire les pertes ou aider aux tests de la carte.

De plus, un élément a été omis par l'équipe précédente, dans les indications de la réalisation de la carte, il est indiqué :

> « Must be mounted vy close to the MOSFET. Directly bridge the drain and source if possible »

pour le composant `C1` indiquant que cette capacité doit être installée le plus proche possible des MOSFET.

Une autre modification est l'ajout des pads CMS pour la plupart des composants, cet ajout permet de pouvoir modifier aisément la valeur des composants testés ou les remplacer si celui-ci devient inutilisable. Cette modification permet un gain de temps énorme en plus d'améliorer la praticité des tests.

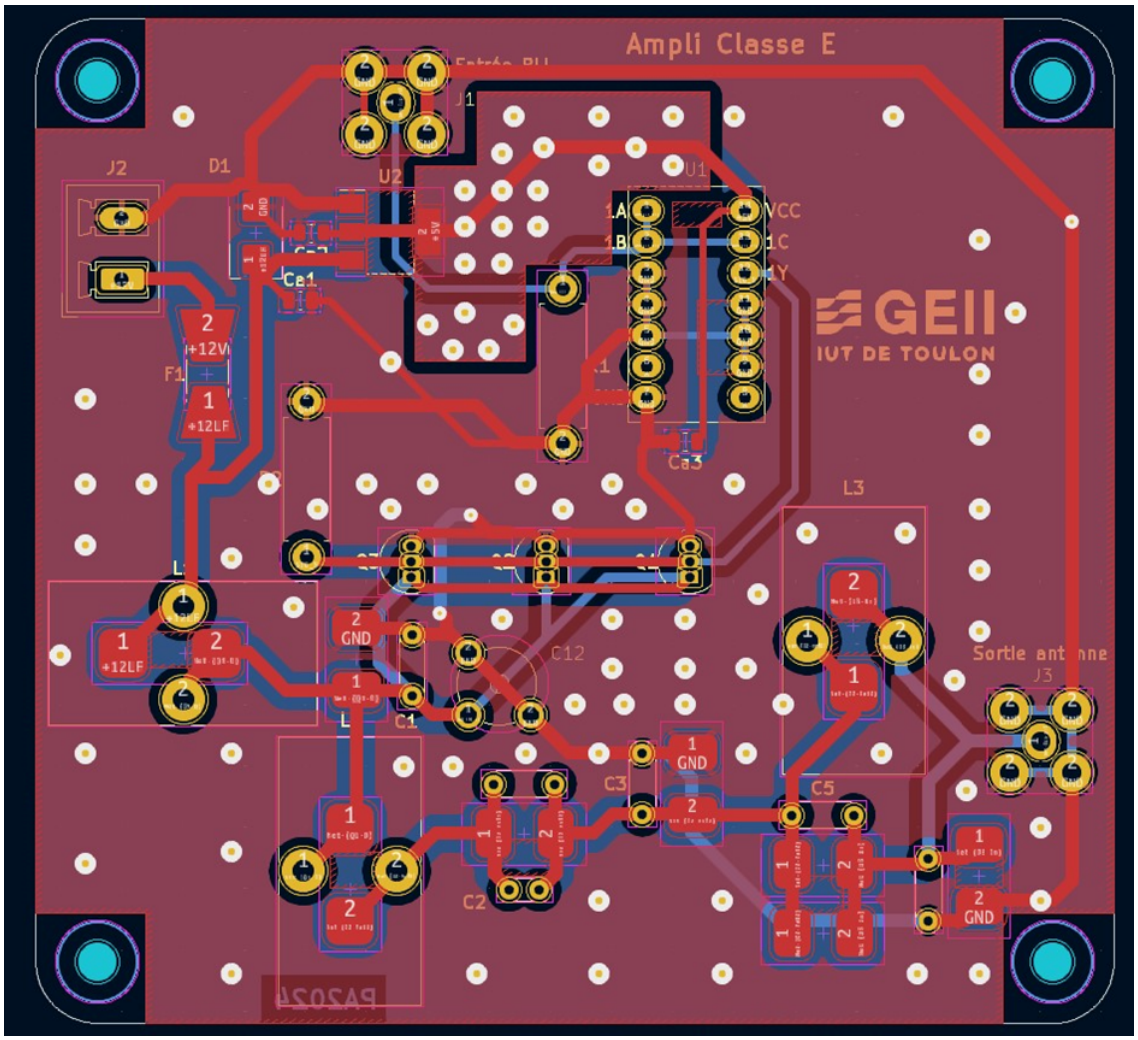



FIGURE 4.2.17 – Routage de la carte sous `LTSpice`

Enfin, l'élément le plus complexe à modifier, les bobines. Chaque bobine devait être refaite avec ses nouvelles valeurs. Pour cela, un logiciel appelait `mini Tore Calculator` a été utilisé. Ici aussi, pour une raison de gain de temps, les calculs des composants ont été réalisés via un logiciel, l'étude sur le calcul des bobines n'étant pas nécessaire à la réalisation du projet.

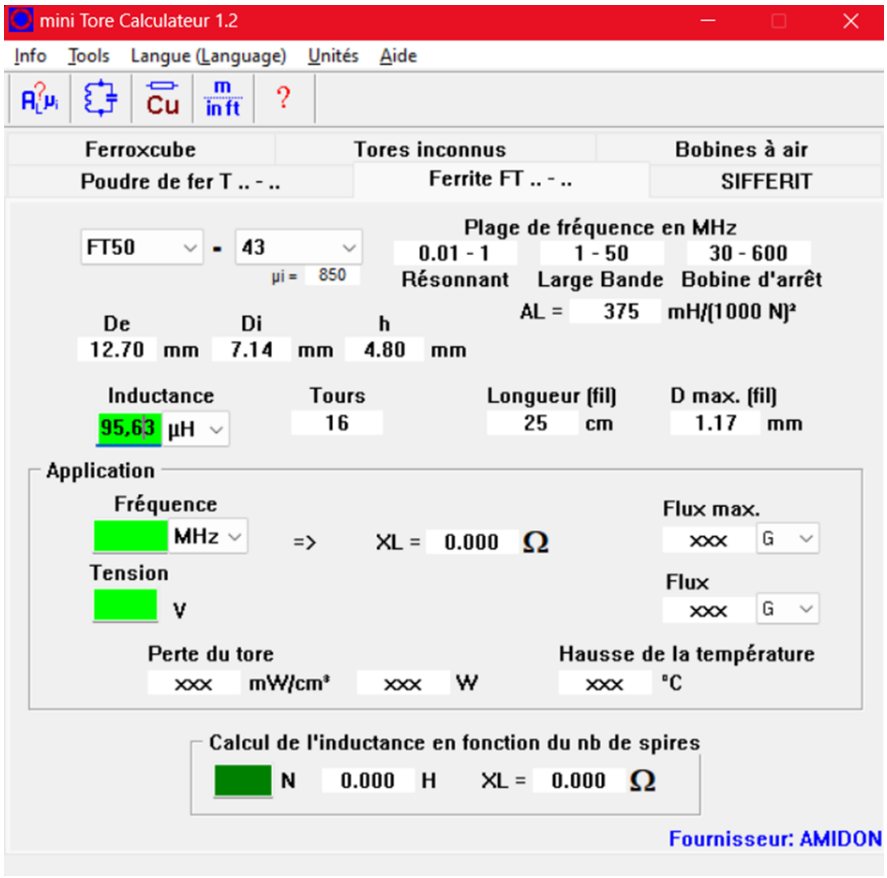



FIGURE 4.2.18 – Interface de conception de bobine - `mini Tore Calculator`

Le logiciel `mini Tore Calculator` se présente comme en figure 4.2.18, il suffit de sélectionner le composant du tore qui sera utilisé puis de saisir l'inductance désirée. En entrant seulement ces valeurs le logiciel calcul le nombre de tours qu'il faudra faire autour du tore, la longueur de fil nécessaire ainsi que le diamètre de fil maximum qui peut être utilisé. Il est cependant important d'utilisé le diamètre de fil le plus proche possible de la valeur indiquée.

Une fois les bobines réalisées, il peut être intéressant de vérifier leur facteur de qualité.

Le facteur de qualité, ou **facteur Q**, est un paramètre clé caractérisant les performances d'une bobine dans un circuit électrique. Il est défini comme le rapport entre l'énergie stockée et l'énergie dissipée par cycle, ce qui permet d'évaluer les pertes dans l'inductance. Mathématiquement, il s'exprime par la relation suivante :

$$\mathrm{Q} = \frac{\omega \mathrm{L}}{\mathrm{R}} \tag{1}$$

Avec, $\omega$ la pulsation (omega $= 2\pi$f), L l'inductance [H] et R la résistance en série de la bobine [Ω].

Ici,

$$\omega = 2\pi \mathrm{f} = 2\pi \cdot 10{,}14 \cdot 10^{6} = 63{,}71\,\mathrm{MHz}$$

Les facteurs de charge sont décrit dans l'énumération suivante :

— `Bobine L1` : Le facteur de qualité n'est pas important, le seul impact de cette bobine sur le système est le temps de charge.

— `Bobine L2` : L'inductance désirée est 6,38 µH, son inductance mesurée est 6,55 µH pour une résistance de 41 Ω. Soit un facteur qualité de 10,18.

— `Bobine L3` : l'inductance désirée est 0,46 µH, son inductance mesurée est 0,63 µH pour une résistance de 3,88 Ω. Soit un facteur qualité de 7,55.

### 4.2.3 Conception de la carte principale

La carte principale est basée sur un microcontrôleur de la gamme STM32L051x en boitier LQFP32. Elle dispose de quatre principales fonctionnalités : une alimentation, une IHM rudimentaire, une liaison I2C et une interface de programmation de compatible ST-Link.

**Alimentation de la carte et du microcontrôleur** L'étage d'alimentation est assuré par un régulateur de tension LDO qui délivre une tension de 3.3V. De plus, pour un fonctionnement optimal du STM32, il faut mettre en place des capacités de découplages sur l'ensemble des brôches d'alimentation (voir figure 4.2.19). Pour ce faire, il faut se référer au *Power-supply schema* fournit dans la documentation du constructeur montré dans la figure 4.2.20.

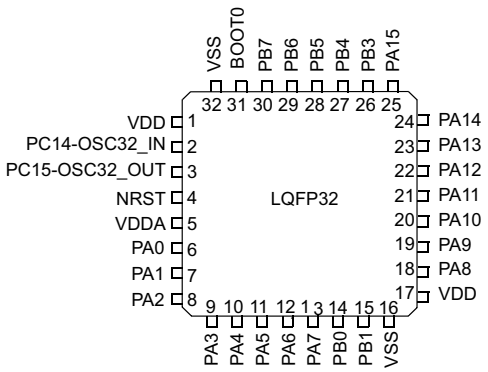



FIGURE 4.2.19 – Pinout du STM32L051x en boitier LQFP32

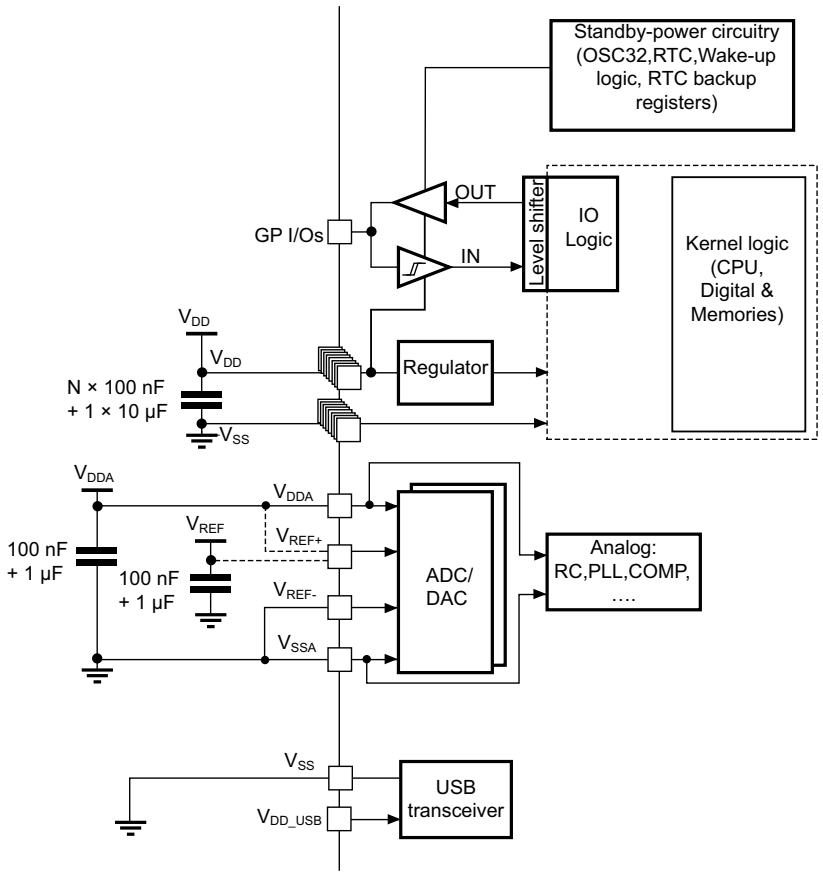



FIGURE 4.2.20 – Schéma d'alimentation du STM32L051x

La schéma d'alimentation indique comment alimenter notre microcontrôleur, et combien de capacité de découplage il faut pour chaque pin d'alimentation présent dans le pinout du STM32 utilisé pour le projet. Ici le schéma montre une architecture plutôt basique avec une capacité de découplage de 100nF au plus proche de chacune des broches d'alimentation et d'une capacité de type « bulk » de 1µF en amont.

**Interface de programmation** Cependant, cette carte ne pourra être programmée sans ST-LINK.

Le ST-LINK est le principal outil de programmation et de débogage pour les cartes STM32. Il permet de charger du code et de déboguer à distance via JTAG ou SWD.

Pour savoir comment brancher notre ST-LINK au STM32, il faut consulter la datasheet concernant l'attribution des pins. Ainsi que la datasheet du ST-LINK pour voir la connexion STM32 / ST-LINK. Ces informations ont été rassemblé dans le tableau présenté en figure 4.2.21.

| Pin Number | Pin name | Pin functions |
|---|---|---|
| 24 | PA14 | SWCLK |
| 26 | PB3 | SWO |
| 23 | PA13 | SWIO |
| 4 | NRST | Reset |
| 32 / 15 | GND | GND |

FIGURE 4.2.21 – Attribution des pins utiles à la liaison STM32/ST-LINK

**Interface I2C** De plus, pour permettre la communication avec le module de génération du signal, il a fallu implémenter un bornier compatible avec ce module. La correspondance des pins est donnée dans la figure 4.2.22.

| STM32 Pin | Pin function | Module Pin |
|---|---|---|
| - | - | 0 |
| - | - | 1 |
| - | - | 2 |
| PB6 | I2C1_SCL | SCL |
| PB7 | I2C1_SDA | SDA |
| GND | GND | GND |
| VDD | VDD | VIN |

FIGURE 4.2.22 – Attribution des pins utiles à la liaison STM32/Module PLL

**Routage complet** Un schéma de la connexion finale a donc été réalisé sur KiCad, et le câblage a été fait sur une breadboard pour tester la connexion et vérifier si le STM32 est reconnu par le ST-LINK V2 via STM32 Programmer. La figure 4.2.23 montre une vue 3D de la carte principale routée. L'ensemble du schéma est présent dans le dossier joint à ce rapport dans le fichier : `hardware/SCH_Master_board.pdf`.

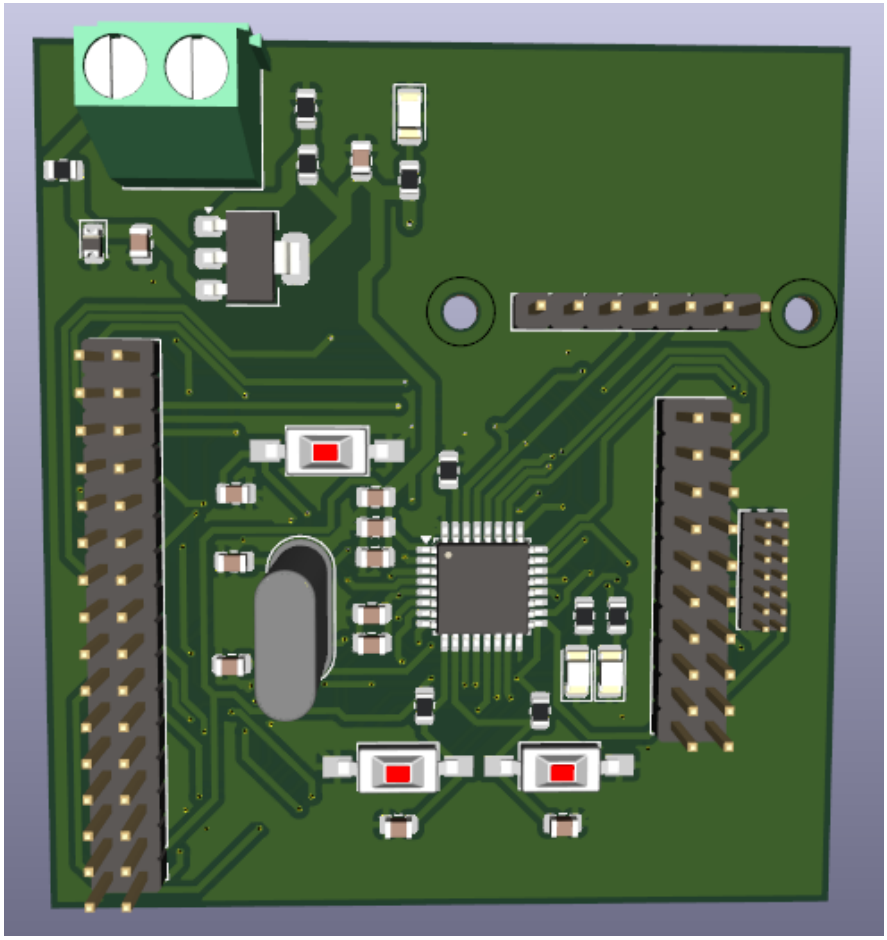


FIGURE 4.2.23 – Visualisation 3D du routage de la carte principale

Il est à noter qu'une résistance de tirage bas a été placée sur le pin 32 (BOOT0). Celle-ci permet de démarrer le microcontrôleur sur sa flash interne.

Finalement, des leds et des boutons ont été ajoutés et constitues un IHM rudimentaire au besoin.

# 5 Création du logiciel de transmission

Le logiciel embarqué sur la carte principale doit remplir plusieurs fonctions clés pour assurer la transmission des messages WSPR de manière efficace et conforme au protocole.

## 5.1 Cahier des charges du logiciel de transmission

### 5.1.1 Création du message WSPR

Le logiciel doit être capable de générer des messages WSPR en respectant les spécifications du protocole. Cela inclut l'encodage des informations telles que l'indicatif d'appel, la localisation géographique, et la puissance d'émission en dBm.

### 5.1.2 Respect des aspects temporelles du protocole

Le protocole WSPR impose des contraintes strictes sur la temporalité des transmissions afin d'optimiser l'utilisation du spectre radio et de minimiser les interférences. Premièrement, les transmissions ne doivent pas excéder 20 % du temps sur une période donnée. Deuxièmement, les transmissions doivent être synchronisées pour débuter exclusivement à la première seconde des minutes paires (0, 2, 4, etc.), ce qui nécessite une synchronisation temporelle précise, assurée par la RTC (ou à terme par une référence GPS). De plus, le protocole interdit les transmissions consécutives, c'est-à-dire qu'il ne doit pas y avoir deux transmissions successives sur des minutes paires adjacentes (par exemple, éviter les minutes 2 et 4). En conséquence, sur une période d'une heure, seulement 6 transmissions WSPR sont autorisées.

### 5.1.3 Gestion des phases de réveil et de veille

Le logiciel doit gérer les réveils programmés via l'horloge interne pour activer la carte uniquement lors des minutes paires et pendant les périodes autorisées pour l'envoi des messages. Le reste du temps, la carte doit être en mode veille (sleep mode) pour économiser l'énergie.

### 5.1.4 Configuration des alarmes pour l'envoi du message

Le logiciel doit configurer des alarmes basées sur la RTC pour déclencher l'envoi des messages WSPR aux moments appropriés. Cela inclut la gestion des intervalles de temps entre les transmissions pour éviter les envois consécutifs.

### 5.1.5 Ajout d'une phase de synchronisation

Le logiciel doit inclure une fonction d'initialisation permettant à l'utilisateur de configurer l'horloge RTC en entrant une commande du type '[HH,MM,SS]' sur le port série. Cela permet de synchroniser l'horloge interne avec l'heure réelle et de garantir que les transmissions WSPR sont effectuées aux bons moments.

### 5.1.6 Couche d'abstraction de la PLL

Finalement, le logiciel doit inclure une librairie qui permet l'abstraction du fonctionnement de la PLL. Cette libraire doit être en mesure d'initialiser la PLL, d'abstraire l'utilisation de la modulation 4-FSK, de permettre la sélection d'un jeu de fréquences (ici 4 fréquences correspondant aux quatre symboles de la modulation) et d'activer ou de désactiver les sorties.

## 5.2 Création des librairies

Pour faciliter le développement et la maintenance du logiciel, les fonctions essentielles sont regroupées dans plusieurs librairies.
Ces librairies incluent les fonctions suivantes :

- La gestion de l'horloge RTC et des alarmes.
- La gestion d'un tableau listant les minutes paires d'envoi.
- La génération des messages WSPR.
- La configuration et le contrôle du module SI5351 pour la génération du signal radio.
- La gestion des modes de veille et des réveils.

### 5.2.1 La librairie d'abstraction du module de génération du signal

**Introduction** Cette sous-section aborde la création d'une librairie qui constitue une couche d'abstraction pour l'utilisation du module de génération du signal, en accord avec le cahier des charges décrit dans « Génération stable et précise du signal radio » . La création du module est décrite dans la sous-section « Conception du module de génération du signal ».

**Fonctionnement du circuit intégré Si5351A** Le `Si5351` est un circuit intégré qui propose de nombreuses options. Toutes les possibilités ne sont pas exploitées et donc ne sont pas explicitées dans ce document. Les pages qui suivent s'appuient majoritairement sur deux documents : l'`AN619` [6] et la documentation proposée par le constructeur.

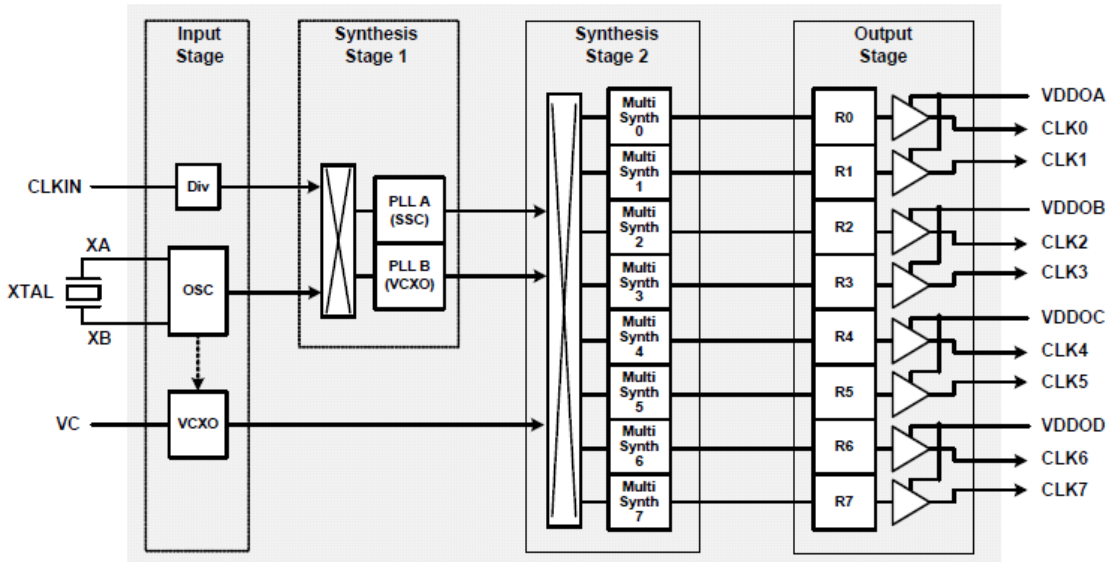



FIGURE 5.2.1 – Diagramme bloc du `Si5351A/B/C` - AN619

La figure 5.2.1 montre le diagramme bloc pour tous les modèles de `Si5351`. Ainsi, certaines fonctionnalités ne sont pas disponibles sur le `SI5351A` en boitier `10-MSOP`.

- **Input Stage :** La figure 5.2.1 montre trois possibilités d'entrée. Toutefois, pour le `SI5351A` seule l'entrée sur un oscillateur est disponible (`XA` et `XB`). Un quartz [7] de 25 MHz a été choisi.
- **Output Stage** La figure 5.2.1 présente un étage de sortie avec huit sorties. Dans le cas du `SI5351A` seules, trois sont disponible : `CLK0`, `CLK1` et `CLK2`.

De plus, il a été vu que la sortie du module de génération du signal est une des quatre fréquences de la modulation 4-FSK ou rien. Ainsi, une seule sortie est utilisée : `CLK0`. Par extension, une seule PLL et un seul Multisynth sont utilisés : la `PLL A` et le `MultiSynth 0`.

Avant de plonger dans la configuration à proprement parler, il convient de s'attarder sur les formules qui permettent d'exprimer la fréquence de sortie du signal fourni par le Si5351.

6. AN619 - Manually Generating an Si5351 Register Map for 10-MSOP and 20-QFN Devices - Skyworks
7. Dans les faits, un oscillateur est branché sur `XA` et `XB` est laissé en l'air pour permettre l'utilisation d'un TXCO. Toutefois, cela n'impacte pas le raisonnement qui va suivre (voir « Conception du module de génération du signal »)

$$f_{out_x} = \frac{f_{vco}}{N_A \cdot R_x} \tag{2}$$

Avec,

$$f_{vco} = f_{XTAL} \cdot M_x \qquad \text{« PLL Feedback Multisynth divider »} \tag{3}$$

Ici, $R_x$ est le diviseur de sortie. La figure 5.2.2 montre l'ensemble de la chaîne de traitement de la fréquence. Le schéma est issu du document AN619 et montre la sortie et le Multisynth 3. Toutefois, cette illustration s'applique également pour CLK0 et Multsynth 0.

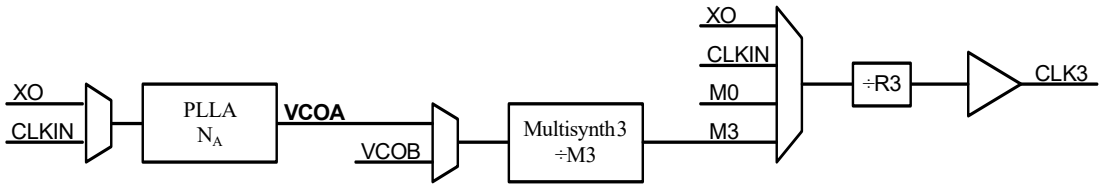



FIGURE 5.2.2 – Illustration de la chaine de traitement du signal pour CLK3 - AN619

Il est à noter que les Multisynth peuvent agir soit en mode entier (Integer) ou en mode réel (Fractionnal). Toutefois, ce dernier augmente le tremblement de la fréquence du signal, ce qui peut être problématique dans la mesure où le signal de sortie doit avoir une précision d'au moins 1 Hz sur 10 MHz.

**Configuration du circuit adaptée au WSPR** Il existe plusieurs options pour la configuration du `Si5351`. La majorité des librairies existantes calculent les registres de configuration du circuit directement pendant l'exécution. Toutefois, cela nécessite d'approcher des nombres flottants par des fractions sous la forme :

$$x = a + \frac{b}{c} \tag{4}$$

C'est pourquoi, une autre solution a été choisie. Les registres de configuration du circuit sont déterminés à l'avance, en utilisant un programme fournit par le constructeur du `SI5351` : « ClockBuilder Pro ». Les registres de configuration sont séparés dans le programme en plusieurs catégories.

- **Les registres de configuration généraux** Ils sont utilisés dans la phase d'initialisation du circuit. Ils configurent le comportement général du circuit
- **Les registres de configuration fréquentiels** Pour correspondre au protocole WSPR, chaque « Jeu de fréquence » est composé de quatre ensembles de registres de configuration fréquentiels (un par fréquence). Ces registres déterminent la valeur des diviseurs et des multiplicateurs de la chaine de traitement du signal ($M_x$, $R_x$ et $N_A$) présentés dans la figure 5.2.2

La figure 5.2.3 montre un exemple de sortie de « ClockBuilder Pro ». Chaque sortie se fait sous la forme d'un « Frequency Plan » qui assigne à chaque composant de la chaine de traitement une valeur en se basant sur une fréquence spécifique. Attention, les coefficients de la PLL et du Multisynth, respectivement M et N sont intervertis entre la figure 5.2.3 et 5.2.2. Pour clarifier les choses, dans la suite du rapport les coefficients sous la forme $M_x$ sont ceux de la PLL et $N_x$ ceux du Multisynth.

```
Frequency Plan
==============

PLL_A:
   Enabled Features = None
   Fvco             = 899.94275 MHz
   M                = 35.99771
   Input0:
      Source           = Crystal
      Source Frequency = 25 MHz
      Fpfd             = 25 MHz
      Load Capacitance = Load_08pF
   Output0:
      Features       = None
      Disabled State = StopLow
      R              = 1  (2^0)
      Fout           = 10.1402 MHz
      N              = 88.75
```

FIGURE 5.2.3 – Exemple de sortie du logiciel « ClockBuilder Pro »

Il est alors possible de déterminer les valeurs à donner à chaque coefficient pour obtenir chacune des quatre fréquences voulues. Ces quatre fréquences constituent alors un jeu de fréquence WSPR. Le tableau ci-dessous montre la configuration à mettre en place pour un jeu de fréquence WSPR sur la bande des 30m avec un décalage de 1,46 Hz entre chaque fréquence. Il est à noter que les diviseurs/multiplicateurs du Multisynth ($N_A$ et $R_0$) ne changent pas suivant la fréquence. En effet, cela permet de limiter le tremblement en fréquence du signal car de cette manière, le Multisynth est utilisé en mode « Integer ».

| **Symbole** | **Fréquence[MHz]** | **$M_0$** | **$R_0$** | **$N_A$** |
|---|---|---|---|---|
| 0 | 10,140 2 | 35,997 71 | 1 | 88,75 |
| 1 | 10,140 201 46 | 35,997 715 2 | 1 | 88,75 |
| 2 | 10,140 202 92 | 35,997 720 32 | 1 | 88,75 |
| 3 | 10,140 204 38 | 35,997 725 44 | 1 | 88,75 |

FIGURE 5.2.4 – Tableau de configuration du module pour un jeu de fréquence

Maintenant que la façon de configurer le `SI5351A` a été expliqué, il ne reste qu'à écrire la librairie. Il est à noter que la librairie dont il est question ici, s'appuie sur la HAL fournit par l'environnement STM32.

**Arborescence de la librairie** La librairie se décompose en plusieurs fichiers :

**Inc** Dossier contenant tous les fichiers headers

**assert.h** Macros pour la gestion des erreurs

**errors.h** Définition du type `errcode_t`

**si5351.h** Prototypes et définition des types de la librairie

**wspr_pll.h** Prototypes et définition des types pour la gestion des jeux de fréquence WSPR

**wspr_frequency_def.h** Déclaration des jeux de fréquences WSPR

**Src** Dossier contenant tous les fichiers sources

**si5351.c** Déclarations des fonctions principales

**wspr_pll.c** Déclarations des fonctions de gestion des jeux de fréquences

### Routine I2C

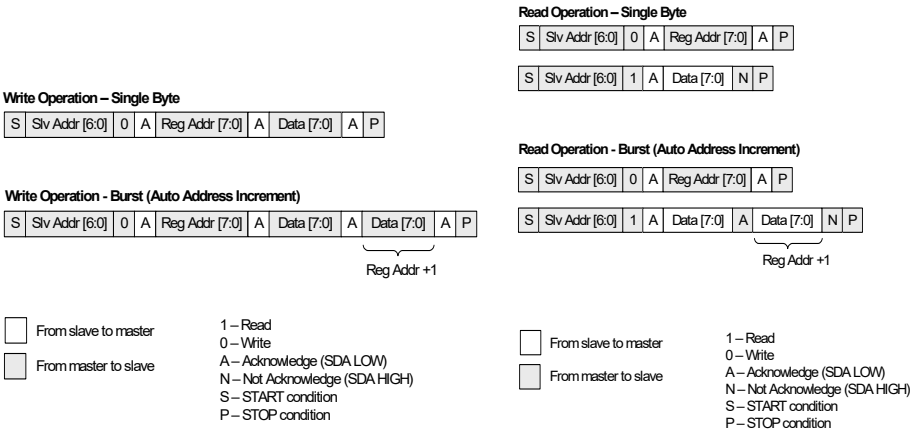
**Import des headers** Dans un premier temps, il faut importer les fichiers header de la HAL. Malgré l'apparente simplicité, il y a quelques subtilités. Il est tentant d'importer la HAL comme suit, n'ayant besoin que du module I2C.

```
#include "stm32f3xx_hal_i2c.h"
```

Agir, de la sorte provoque une avalanche d'erreurs, environ une centaine. En effet, les fichiers d'header de la HAL de STM32 s'importent les uns les autres. Ainsi, si un fichier est importé de manière indépendante, cela provoque des erreurs. La bonne manière d'importer les headers de la HAL est la suivante. Il est à noter que ces directives pré-processeur peuvent changer légèrement suivant la famille du microcontrôleur utilisé. Par exemple, si la librairie est exploitée par un `STM32L051K8T6`, il faudra importer les fichiers de la forme `stm32l0xx.h`

```
#include "stm32f3xx.h"
#include "stm32f3xx_hal.h"
```

**Lecture et écriture sur le Si5351** Comme beaucoup de périphérique I2C, le `Si5351` dispose de deux modes d'écriture et de lecture : le mode octet par octet et le mode « burst » qui permet d'écrire ou de recevoir plusieurs registres contigus en mémoire. La figure 5.2.5, issue de la documentation du `Si5351`, montre les deux types de lecture/écriture.




(a) Écriture sur le bus I2C (b) Lecture sur le bus I2C

FIGURE 5.2.5 – Modes de lecture/écriture sur le `Si5351`

De plus, les communications sur le bus I2C doivent se faire de manière non-bloquante. En d'autres termes, le maître ne doit pas garder la ligne SDA à l'état haut. C'est pourquoi, la librairie exploite les fonctions `HAL_I2C_Mem_Write` et `HAL_I2C_Mem_Read`. Les fonctions d'interaction de la librairie ne sont que des encapsulations de la HAL.

```
errcode_t Si5351_write_register(Si5351_Descriptor_t *si5351handle
      , uint8_t reg, uint8_t value) {
   HAL_StatusTypeDef ret = HAL_I2C_Mem_Write(si5351handle->i2cHandle,
            (uint8_t)(SI5351_BUS_ADDRESS),
            (uint8_t)reg,                  // Adresse du registre
            I2C_MEMADD_SIZE_8BIT,          // Le SI5351 utilise des adresses de 8bit
            (uint8_t*)(&value),            // Octet(s) à écrire
            1,                             // Nombre d'octet à envoyer
            HAL_MAX_DELAY);
   ASSERT(ret == HAL_OK, ERRCODE_I2C_WRITE);

   return ERRCODE_NONE;
}

errcode_t Si5351_read_register(Si5351_Descriptor_t *si5351handle
      , uint8_t reg, uint8_t* data) {

   HAL_StatusTypeDef ret = HAL_I2C_Mem_Read(si5351handle->i2cHandle,
         SI5351_BUS_ADDRESS,
         reg,
         I2C_MEMADD_SIZE_8BIT,
         data,
         1,
         HAL_MAX_DELAY);
   ASSERT(ret == HAL_OK, ERRCODE_I2C_READ);
   return ERRCODE_NONE;
}
```

Le lecteur attentif aura remarqué la gestion des erreurs avec `ASSERT` et le type de retour de la fonction : `errcode_t`. La gestion des erreurs est traitée dans la sous-section suivante.

**Propagation des erreurs** La gestion et par extension la propagation des erreurs, est primordiale pour assurer un code robuste et faciliter le débogage. Pour ce faire, deux fichiers ont été ajoutés : `assert.h` et `errors.h`.

Le fichier assert.h ajoute deux macros. La première, `ASSERT`, permet de retourner une erreur si une condition n'est pas vérifiée. La deuxième, `PEDDLE_ASSERT` permet de retourner une erreur selon la valeur passée en paramètre. La deuxième macro pourrait paraître inutile. Toutefois, elle permet de faire remonter une erreur sans alourdir le code avec des « if » à répétition. Il est à noter que ces macros ne peuvent s'utiliser que dans des fonctions qui retournent le type `errcode_t`.

```
#ifndef ASSERT_H
#define ASSERT_H

#include "errors.h"

/*
 * SHOULD BE USED IN FUNCTION WITH RETURN TYPE : errocde_t <errors.h>
 *
 *    When condition is not met, the function
 *    which this macro is into return specified
 *    error code.
 */
#define ASSERT(condition, errcode) \
   if (!(condition)) {            \
      return (errcode);           \
   }

/*
 * SHOULD BE USED IN FUNCTION WITH RETURN TYPE : errocde_t <errors.h>
 *
 *    Check error code and peddle error
 */

#define PEDDLE_ERRCODE(errcode)    \
   if (errcode != ERRCODE_NONE) { \
      return (errcode);           \
   }
#endifreturn ERRCODE_NONE;
}
```

FIGURE 5.2.6 – Contenu du fichier `assert.h`

En parallèle, le fichier `errors.h` fournit une « enum » (`errcode_t`) qui formalise les différents code d'erreur.

**Initialisation du module** L'initialisation du `Si5351` est réalisée en plusieurs étapes. Dans un premier temps, il faut initialiser le périphérique lui-même et désactiver les fonctionnalités non désirées. Ensuite, c'est les « PLL » et les « Multisynth » qui sont initialisés. La figure 5.2.7 montre clairement les différentes étapes.

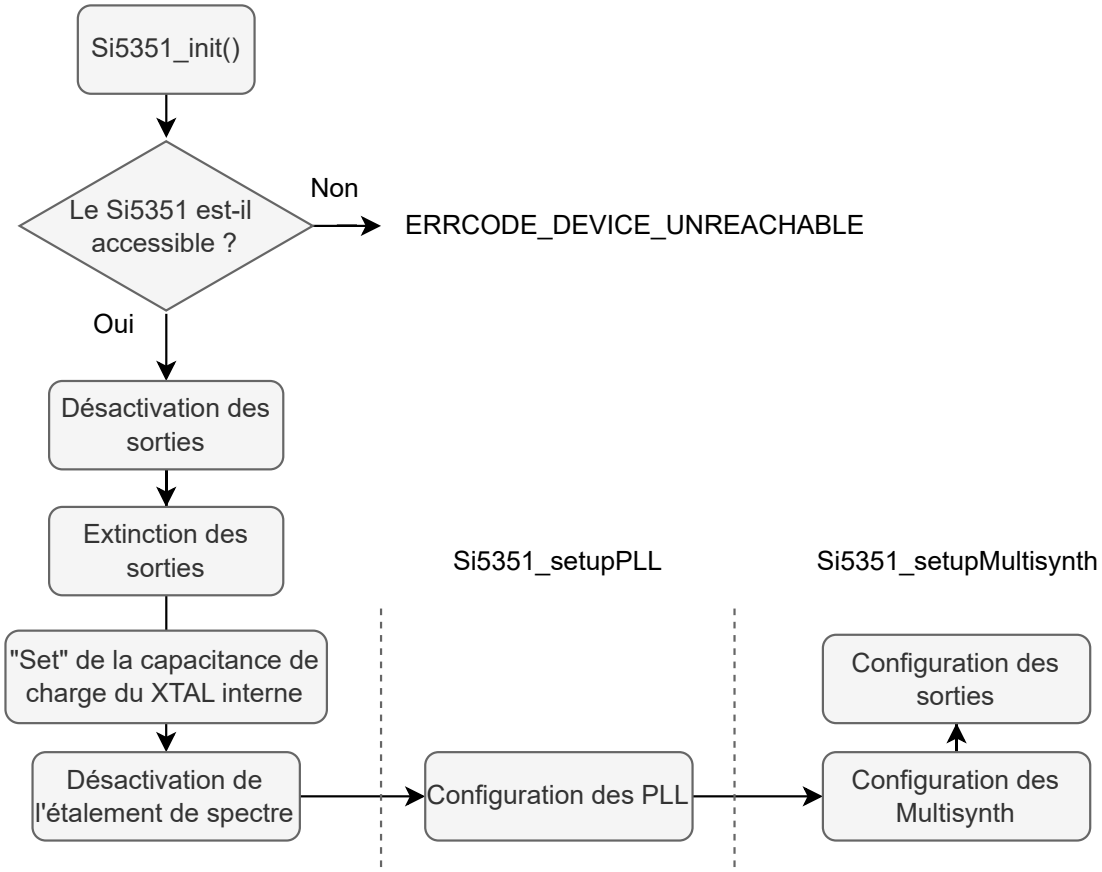



FIGURE 5.2.7 – Procédure d'initialisation du Si5351

**Utilisation de la librairie** La librairie a été conçue pour ce projet ainsi, elle n'implémente pas toutes les fonctionnalités disponibles. L'utilisation de la librairie s'appuie sur des structures qui décrivent la configuration des PLL et des Multisynth, mais également sur trois fonctions principales ; les prototypes de ces fonctions sont présentés ci-dessous.

```
errcode_t Si5351_applyPLLConfig(Si5351_Descriptor_t *si5351handle
      , Si5351_PLL_Descriptor_t *si5351plldesc);

errcode_t Si5351_applyMSConfig(Si5351_Descriptor_t *si5351handle
      , Si5351_MS_Descriptor_t *si5351msdesc);

errcode_t Si5351_applyConfig(Si5351_Descriptor_t *si5351handle
      , Si5351_Config_t *si5351config);
```

Ainsi, il suffit de remplir ces structures en s'appuyant sur le logiciel « ClockBuilder Pro ». Le code source ci-dessous montre la création d'une configuration pour générer un signal de

10,140 2 MHz, sur la sortie CLK0.

```
Si5351_PLL_Descriptor_t pllDesc = {
      SI5351_PLL_A,
      0x00FFF, // Si5351_MSNx_P1
      0x11420, // Si5351_MSNx_P2
      0x186A0 // Si5351_MSNx_P3
};
Si5351_MS_Descriptor_t msDesc = {
      SI5351_CLK0, // Channel
      SI5351_PLL_A, // Source PLL
      0x02A60, // Si5351_MSx_P1
      0x00000, // Si5351_MSx_P2
      0x00004, // Si5351_MSx_P3
      SI5351_R_DIV_1,
      0 // MS0_INT = 0 -> Fractionnal mode

};
Si5351_Config_t si5351Config = {&pllDesc, &msDesc};
Si5351_applyConfig(&si5351Desc, &si5351Config);
```

**Conclusion** La librairie d'abstraction du module de génération du signal est conçue pour être simple d'utilisation et adaptée au protocole WSPR. De plus, sa capacité, à lever et à propager des erreurs la rend robuste et aisée à déboguer, d'une certaine manière la gestion des erreurs faite ici est proche de celle réaliser par la HAL STM32.

### 5.2.2 La librairie d'encodage WSPR

Nous commencerons par la mise en place d'une librairie permettant la transmission de données en basse puissance en utilisant le protocole WSPR. À partir de la librairie JTEncode disponible sur GitHub (https://github.com/etherkit/JTEncode), qui est utilisée pour la création de messages WSPR sur Arduino.

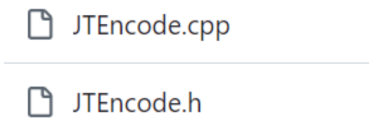



FIGURE 5.2.8 – Fichiers inclues dans la librairie

Une nouvelle version plus universelle a été développée ici. En effet, la librairie JTEncode, initialement structurée autour des fichiers JTEncode.h et JTEncode.cpp, a été simplifiée et condensée en un seul fichier C. Les fonctions qui ne servaient pas à l'encodage WSPR ont été retirées, ainsi que la classe *JTEncode*, ne conservant que les fonctions spécifiques au protocole WSPR. Cette approche a permis de créer une nouvelle librairie en C, facilitant ainsi son portage tout en optimisant les performances pour l'encodage WSPR :

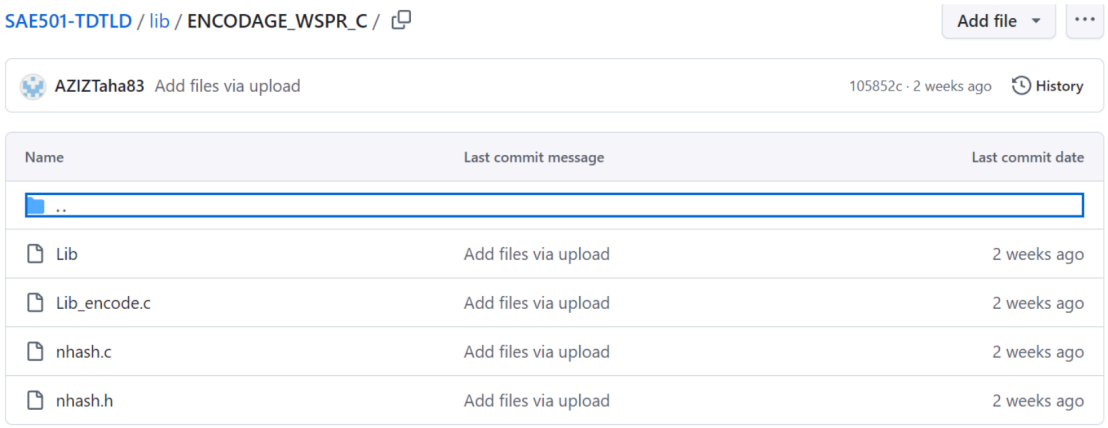



FIGURE 5.2.9 – Librairie en langage C , Encodage WSPR - Github

**Test et comparaison de la librairie avec des données conformes** Afin de valider la conformité de l'encodage.

```
int main() {
   wspr_encode("F4KIX", "JN33", 24, &buffer[0]);

   for(int i = 0 ; i < WSPR_SYMBOL_COUNT ; i++) {
      printf("\%d, ", buffer[i]);
   }
}
```

FIGURE 5.2.10 – Test de la réponse de wspr_encode

Dans le fichier d'encodage WSPR, la fonction principale wspr_encode est testée pour s'assurer de son bon fonctionnement. Cette fonction a pour but d'encoder un message WSPR en suivant un processus précis, prenant en entrée les trois paramètres suivants :

- Indicatif d'appel : un code d'identification unique attribué à une station de radioamateur.
- Localisation : un code représentant la position géographique de la station (sous la forme d'un « locator »). ;
- Niveau de puissance d'émission en dBm : la puissance d'émission du signal exprimée en décibels par rapport à un milliwatt (dBm).

La fonction suit les spécifications du protocole WSPR pour générer un message binaire encodé. Ce message peut ensuite être utilisé pour transmettre des données via des bandes de radio amateur en très basse puissance, tout en maximisant la portée de la transmission. L'encodage se fait en plusieurs étapes, incluant la vérification et la conversion des données d'entrée en un format compatible avec les contraintes du protocole WSPR.

Pour s'assurer de la fiabilité de la librairie, le résultat de l'encodage obtenu avec nos paramètres a été comparé à celui généré par le site WSPR Code Generator [8] à l'aide d'un comparateur de texte :

8. WSPR Code Generator : https://swharden.com/software/wspr-code-generator/

En premier lieu, dans l'invite de commande, le fichier C de l'encodage WSPR tiré de notre librairie a été exécuté.

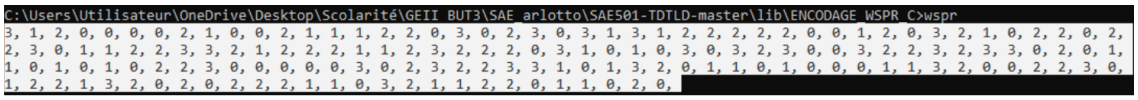



FIGURE 5.2.11 – Retour du main sur l'invite de commande

D'autre part, l'encodage a également été simulé sur le site WSPR Code Generator avec les mêmes paramètres :

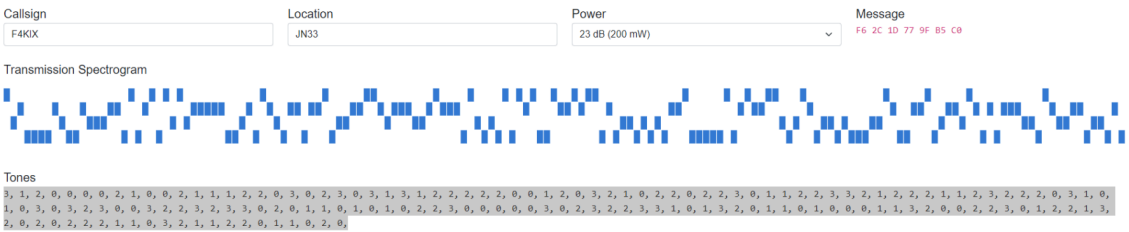



FIGURE 5.2.12 – Retour du site web WSPR Code Generator

Synthèse des résultats : le seul changement entre les deux textes est la source de l'encodage.

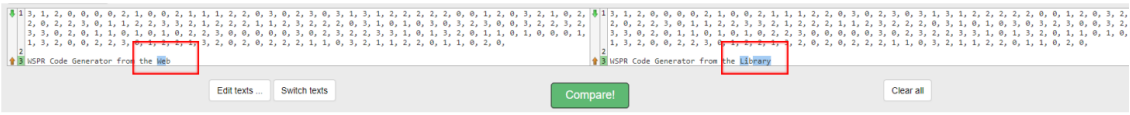



FIGURE 5.2.13 – Retour du comparateur de string

La fonction wspr_encode de notre librairie est donc sûre.

### 5.2.3 Librairies de configuration de la RTC et des alarmes

L'objectif de cette librairie est d'encapsuler les fonctions de manipulations de la RTC fournit par la HAL (Hardware Abstract Layer) STM32. Elle est constituée des quatre fonctions suivantes :

- **RTC_read_print** Fonction qui lit le compteur de la RTC et l'envoi sur le port série.
- **set_time** Fonction qui configure le temps de la RTC.
- **set_date** Fonction qui configure la date de la RTC.
- **set_alarm** Fonction qui met en place une alarme à un temps donné.

Leur prototype est présenté dans la figure 5.2.14.Il est à noter que dans les fonctions set_date et set_alarm, l'argument « day » fait référence au numéro du chiffre dans la semaine (de 0 à 7) et que l'argument « year » encode les deux premiers chiffres de l'année spécifiée ; par exemple year = 25 pour 2025.

```
void RTC_read_print(void);
void set_time(uint8_t hr, uint8_t min, uint8_t sec);
void set_date(uint8_t year, uint8_t month, uint8_t date, uint8_t day);
void set_alarm(uint8_t hr, uint8_t min, uint8_t sec, uint8_t date);
```

FIGURE 5.2.14 – Prototypes des fonctions de la librairie de configuration de la RTC et des alarmes

### 5.2.4 Librairie de génération des « minutes » d'envoi

L'intérêt de cette librairie est de fournir un ensemble de fonction qui permettent de générer les minutes auxquelles seront envoyé les trames WSPR. Afin de respecter la spécification, six minutes seront choisies aléatoirement sur une période d'une heure afin de respecter une émission qui occupe 20% du temps. De plus, ces minutes ne doivent pas être consécutives.

Cette section s'appuie sur quatre fonctions décritent ci-après. Leur prototype est présenté dans la figure 5.2.15.

- **initialize_random_seed** Fonction d'initialisation du générateur de nombre aléatoire. En d'autres termes, elle choisit une « seed » unique qui va servir pour le mécanisme de génération de nombres pseudo-aléatoire de la librairie standard du C (stdlib.h).
- **generate_random_minutes** Cette fonction génère aléatoirement six minutes et remplit un buffer situé en variable globale ; d'où l'absence d'argument.
- **is_unique** Cette fonction vérifie l'unicité du buffer de minutes.
- **sort_minutes** Cette fonction trie un tableau passé en paramètre. Elle exploite l'algorithme de tri à bulles ou « Bubble Sort » en anglais. Le tri à bulles est un algorithme simple qui compare et échange répétitivement des éléments adjacents jusqu'à ce que la liste soit triée.

```
void initialize_random_seed(void);
void generate_random_minutes(void);
uint8_t is_unique(int value, int count, int *rand_transmit_minutes);
void sort_minutes(int *array, int size);
```

FIGURE 5.2.15 – Prototypes des fonctions de génération des « minutes » d'envoi

### 5.2.5 Librairie d'envoi du message WSPR

L'objectif de cette librairie est composé d'une fonction. Celle-ci est chargé de générer un message WSPR et de l'envoyer en s'appuyant sur le module de génération du signal. Le prototype de la fonction est le suivant :

```
void wspr_ready_to_transmit(void);
```

## 5.3 Création du logiciel de la carte principale

### 5.3.1 Découverte de l'environnement STM32

Comme décrit plus haut, le microcontrôleur choisit pour ce projet est de la gamme STM32. Le fabricant fournit également tout un environnement constitué de plusieurs logiciels qui

permettent d'exploiter les capacités des microcontrôleurs au maximum. L'environnement est décrit dans la figure 5.3.1.


FIGURE 5.3.1 – Environnement STM32

Durant ce projet, trois logiciels ont été utilisés :

- **STM32Cube IDE :** Un IDE gratuit basé sur Eclipse, fourni par STMicroelectronics qui permet de faciliter la programmation et la configuration du STM32 utilisé.
- **STM32CubeMX :** Un outil graphique pour configurer les broches, les périphériques et générer du code d'initialisation.
- **STM Programmer :** Les STM32 peuvent être programmés via des interfaces comme SWD (Serial Wire Debug) ou JTAG à l'aide de programmateurs dédiés (ST-Link, J-Link, etc.) STM Programmer est un logiciel qui permet le formatage ou la programmation des STM32 à partir de fichier HEX. Cependant, il peut être utilisé pour vérifier que le microcontrôleur est bien reconnu par ST-Link, ce qui permet de s'assurer rapidement et sans la lourdeur de l'IDE que le microcontrôleur est reconnu et peut être programmé.

### 5.3.2 Conception de l'algorithme principal

La figure 5.3.2 présente l'algorithme principal d'émission.

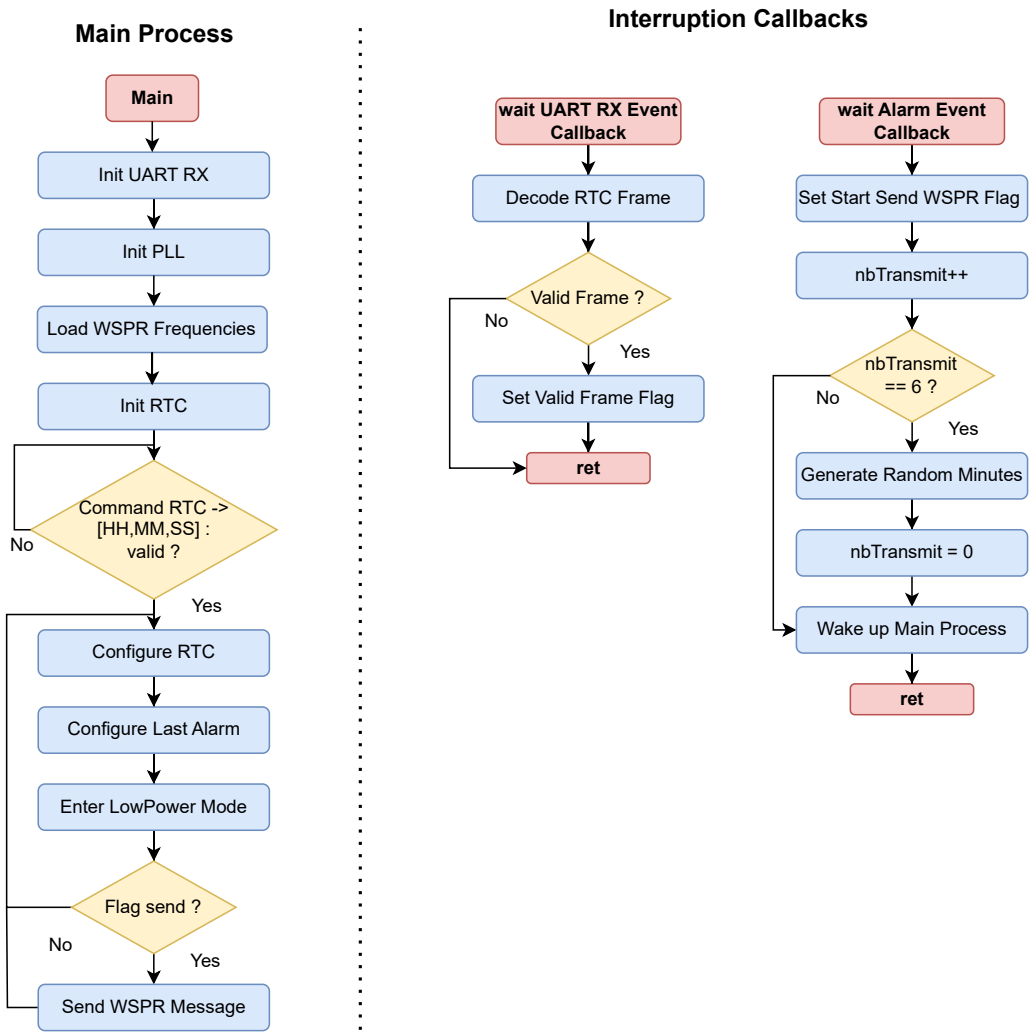



FIGURE 5.3.2 – Algorithme principal d'émission

L'algorithme principal du logiciel est structuré autour des étapes suivantes. Dans un premier temps, il entre dans une phase d'initialisation. Il configure la RTC, l'UART, le module PLL et les jeux de fréquences WSPR nécessaires à la transmission. Ensuite, il attend que la mise à jour de l'heure par l'utilisateur. Pour ce faire, il boucle jusqu'à ce qu'une trame de configuration soit reçue sur le port série. Ceci fait, il met en place la gestion des réveils. En effet, la carte se réveille uniquement pendant les minutes paires et vérifie si une transmission est autorisée (respect de la règle des 20% du temps d'occupation et des minutes non-consécutives). Après avoir configuré les alarmes, le microcontrôleur entre dans un mode de veille. Il se réveille en cas d'interruption de la RTC : si cela arrive il réalise une émission WSPR et reconfigure les alarmes avant de retourner en mode veille.

# 6 Synthèse des résultats

## 6.1 Qualification du module de génération du signal

Cette section décrit les différents tests mis en place pour s'assurer que le module de génération du signal est conforme au cahier des charges décrit dans la section « Génération stable et précise du signal radio » :

> [...] Le module [...] permet de produire un signal stable et précis sur la bande des 30 mètres. La précision du signal est cruciale pour garantir une transmission fiable et sur de longues distances [...]. La faible dérive du signal est également cruciale pour garantir une transmission en accord avec le protocole WSPR. En effet, la bande de fréquence utilisée s'étend de 10.1398 MHz à 10.1402 MHz et les fréquences correspondantes aux quatre symboles de la modulation 4-FSK sont espacées de 2 Hz. Ainsi, le système doit garantir une dérive minimale et maximiser la précision.

Pour garantir, le bon fonctionnement du module plusieurs tests ont été mis en place. Dans un premier temps, un test comparatif de dérive fréquentiel en fonction de la température a été réalisé. Ce test permet également de juger de la stabilité du signal. Dans un second temps, un test a été réalisé pour mesurer la longueur de la transition entre la mise sous tension du module et la génération d'un signal stable.

### 6.1.1 Création d'un environnement de tests

La majorité des tests réalisés nécessite un environnement contrôlé en température pour être réalisés. De plus, les tests nécessitent une mesure de la fréquence de sortie, une mesure de la température et une gestion de la température du système.

La figure 6.1.1 montre l'environnement de test réalisé. Il est constitué d'une glacière, d'un GBF en mode « counter » pour mesurer la fréquence, cadencé à 10 MHz par une source satellite extrêmement précise et d'un support de test qui intègre un capteur de température `BME680` et le module de génération du signal [9].

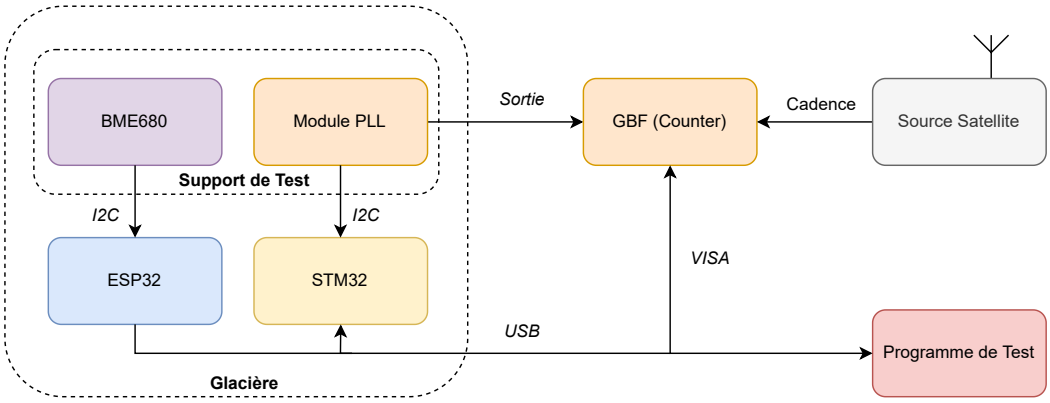



FIGURE 6.1.1 – Schéma de l'environnement de test

9. La création du support de test est décrit dans la section « Conception du module de génération du signal »

Comme le montre la figure 6.1.1, toutes les informations sont centralisées sur un PC avec un programme de test. Ce programme de test est réalisé en Python. Toutes les deux secondes, c'est-à-dire à chaque rafraîchissement du fréquencemètre, il récupère la fréquence de sortie de la PLL et la température mesurée par le `BME680`. Toutes les informations sont sauvegardées dans un fichier `CSV` pour permettre un traitement ultérieur.

Finalement, la figure 6.1.2 montre un exemple d'affichage du programme de test. Il permet de mettre en évidence une caractéristique problématique du matériel à notre disposition : le fréquencemètre a une précision de 1 Hz. D'où les sauts dans le graphique « Frequency deviation vs Time ».

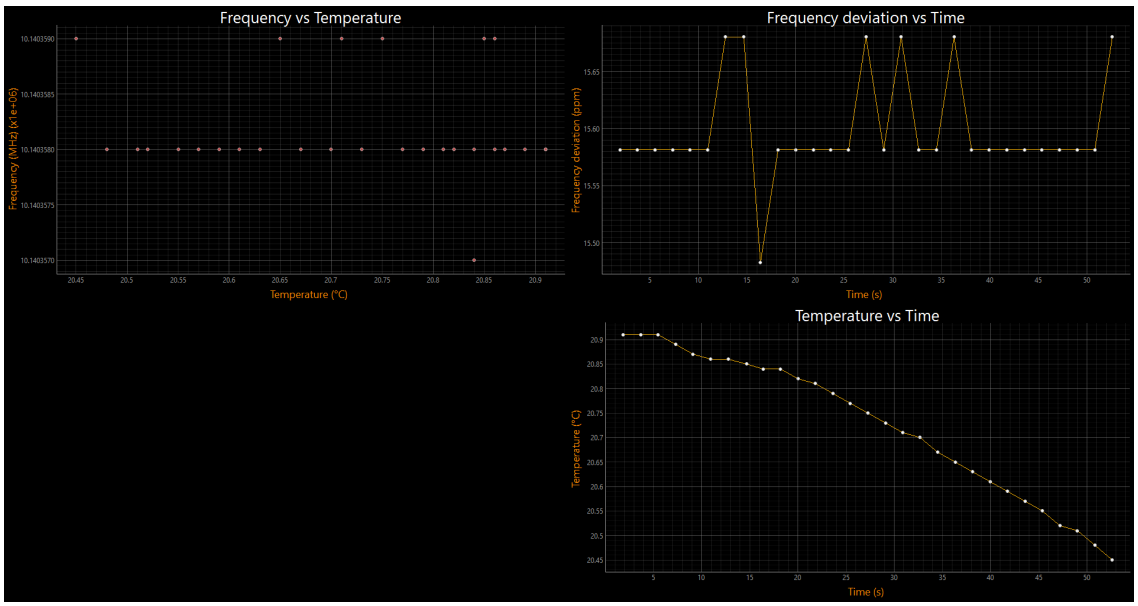



FIGURE 6.1.2 – Exemple d'affichage du programme de test

### 6.1.2 Test comparatif de déviation en fréquence en fonction de la température

Plusieurs relevés ont été réalisés sur un module de génération du signal avec et sans `TCXO`. Afin de mettre en évidence le comportement fréquentiel de la sortie en fonction de la variation de température. Le système de test a été exposé à une plage de température variant entre 8.5 °C et 21.8 °C. La figure 6.1.3 montre la déviation en fréquence de la sortie du module en fonction de la température. Il est à noter que les données présentées ne sont pas brutes. Un décalage a été appliqué pour normaliser les décalages relatifs à 12 °C (température où il y a le plus de relevé). En effet, cette manipulation permet de supprimer les décalages absolues dûs à la mauvaise calibration des quartz et de se concentrer uniquement sur les décalages relatifs dûs à la variation de température. Ici, la courbe « XO » est décalée de +7.37 et la courbe « TCXO » de + 15.443.

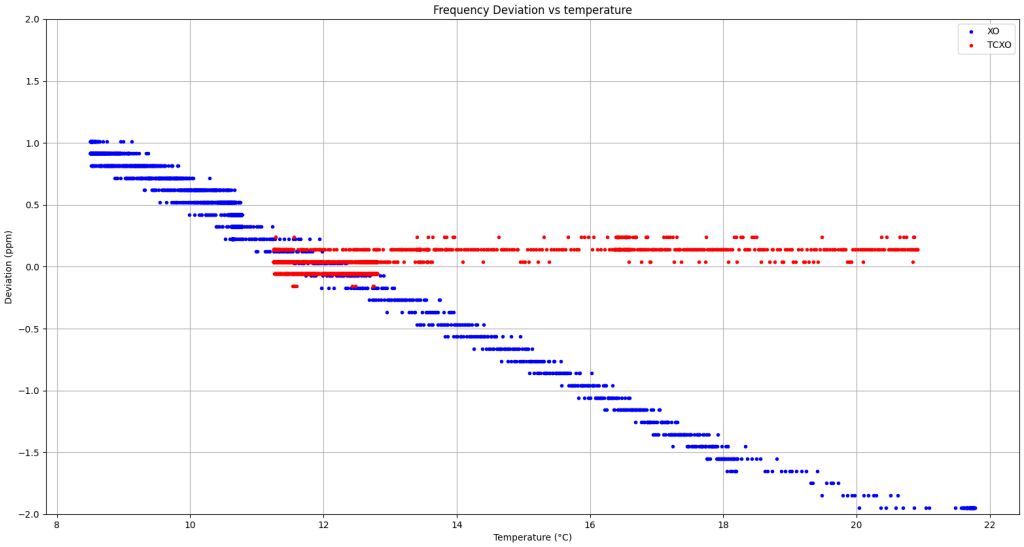



FIGURE 6.1.3 – Déviation en fréquence par rapport à la température pour un module avec ou sans TCXO

**Synthèse des résultats** Dans un premier temps, il est clair que l'utilisation d'un TXCO est primordiale pour limiter le déviation en fréquence. Par ailleurs, sur la plage de température mesurée la déviation du TCXO varie entre 0.25 et -0.151 ppm soit une amplitude de 0,401 ppm, ce qui correspond à une variation de $\pm 2,033$ Hz. Ce résultat est surestimé. En effet, il faut prendre en compte l'imprécision du fréquencemètre et le fait que le graphique représente des données acquises sur plusieurs heures de mesure. En outre, la figure montre que les points minimum et maximum (aux alentours de 12 °C) sont très peu nombreux. Ainsi, si ceux-ci sont négligés, il est possible d'obtenir une variation en fonction de température de $\pm 0.994$ Hz.

Or, l'écart entre les fréquences de la modulation est de 2 Hz. Ainsi, peu importe la température, le module fournit la précision suffisante pour discriminer les différentes fréquences de la modulation.

### 6.1.3 Test de mise sous tension

L'objectif de ce dernier test est de vérifier le temps nécessaire au module pour fournir une fréquence stable. En d'autres termes, l'objectif est de mesurer le temps de chauffe du TCXO. Pour ce faire, le module a été placé au congélateur pendant 30 min, puis il a été mis sous tension. Il est à noter que la mesure de température n'est pas significative ici car le module seul n'est pas en capacité de porter l'air de l'environnement de test à sa température.

Dans la mesure où le système doit être énergiquement sobre, cette manipulation permet de déterminer le temps de chauffe du module et par extension le moment où il faut le mettre sous tension avant une transmission. En effet, le TCXO à lui seul consomme 10mA : il est exclu de le laisser constamment sous tension.

La figure 6.1.4 montre le temps de chauffe du module.

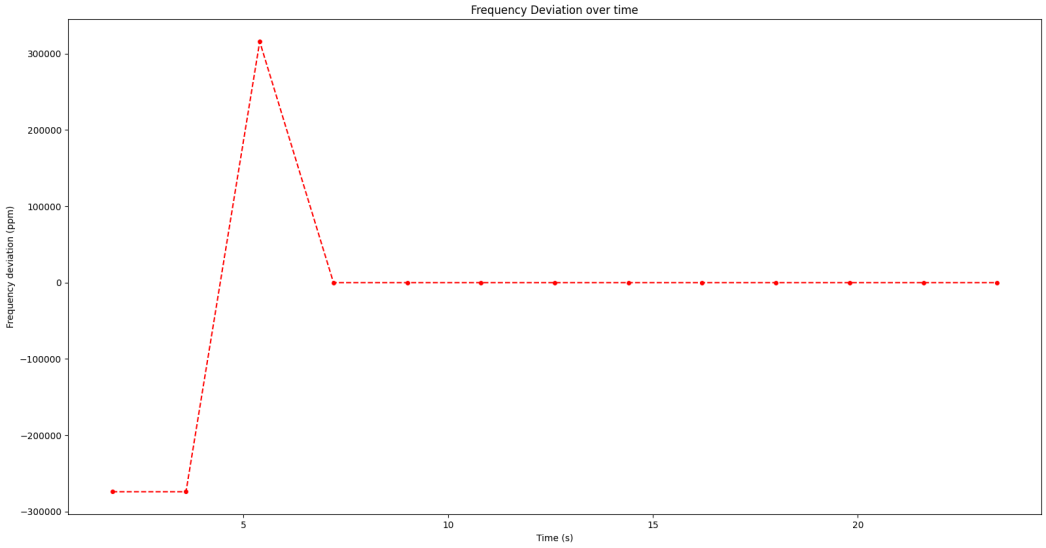



FIGURE 6.1.4 – Déviation en fréquence en fonction du temps à partir de la mise sous tension

**Synthèse des résultats** La figure montre que le temps de chauffe est au maximum de 4 s. Bien que cette manipulation permette d'obtenir un temps maximal de chauffe, la période d'échantillonnage du fréquencemètre rend les mesures de temps imprécises. En effet, avec une période d'échantillonnage de 2 sec, il est impossible de connaître précisément le comportement fréquentiel du module.

Toutefois, le résultat n'est pas sans intérêt. En effet, cette manipulation montre qu'il faut mettre le module sous tension au mois 4 s avant chaque émission.

## 6.2 Qualification du module d'amplification

Pour les premiers tests réalisés la capacité `C1` a été fixé à une valeur de 39 pF et un seul transistor a été mis en place.

Le résultat obtenu à l'issue de ce premier test déjà meilleur que celui de l'année précédente, l'objectif d'optimisation est donc déjà atteint. Cependant, en jouant avec la capacité variable, on peut constater que la puissance maximale est atteinte pour la capacité réglée au minimum.

Pour une alimentation de 12V continue le courant consommé est de 0,07A soit une consommation de 0,84W. Une fois de plus, une divergence est observée entre la simulation et la pratique, les valeurs obtenues en théorie sont bien supérieures à celle pratique.

### 6.2.1 Premier test fonctionnel

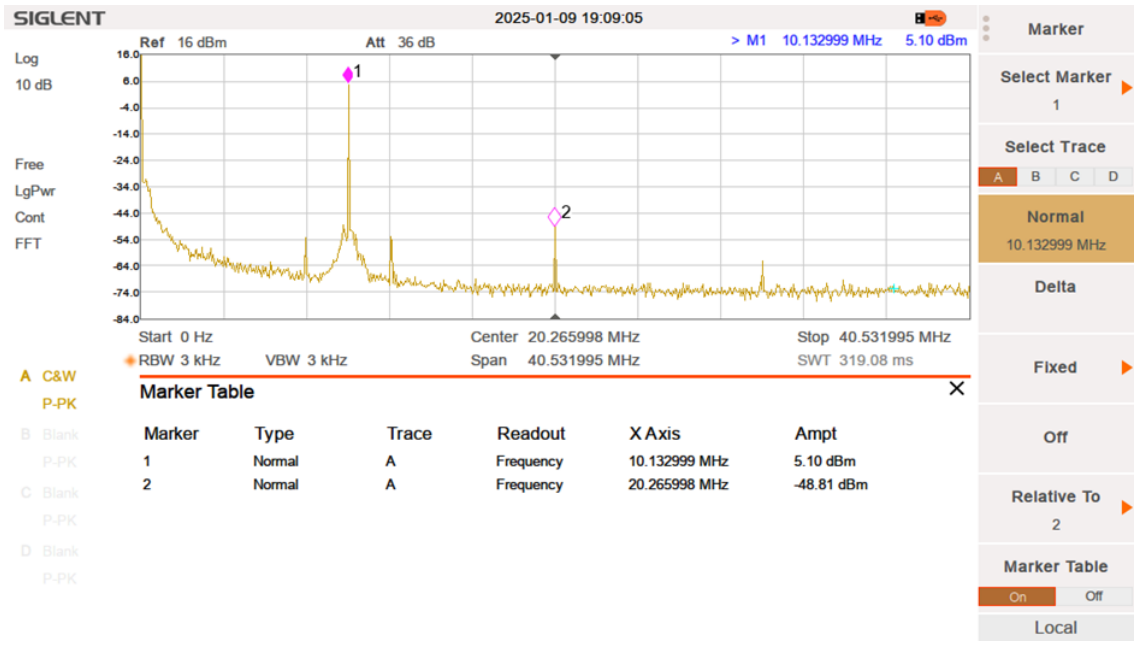



FIGURE 6.2.1 – Mesure du spectre de sortie lors du premier test fonctionnel

En sachant que la capacité variable est au minimum cela veut dire qu'on peut diminuer la valeur de `C1` pour essayer d'avoir plus. Lors du test suivant 2 `BS170` ont été ajouté en parallèle avec celui déjà installé. Chaque transistor ayant une capacité de 17 pF, il est théoriquement possible de réduire la valeur de `C1` de 34 pF. Afin de ne pas sauter des étapes et d'étudier l'évolution du système la capacité `C1` a été réduite à 12 pF.

En conséquence à la diminution de `C1` le système à gagner presque 2 dBm, passant de 25 dBm (316 mW) à 27 dBm (500 mW). La capacité variable était cependant toujours réglée au minium indiquant qu'il était possible de gagner encore en puissance en diminuant la capacité.

### 6.2.2 Le test de trop

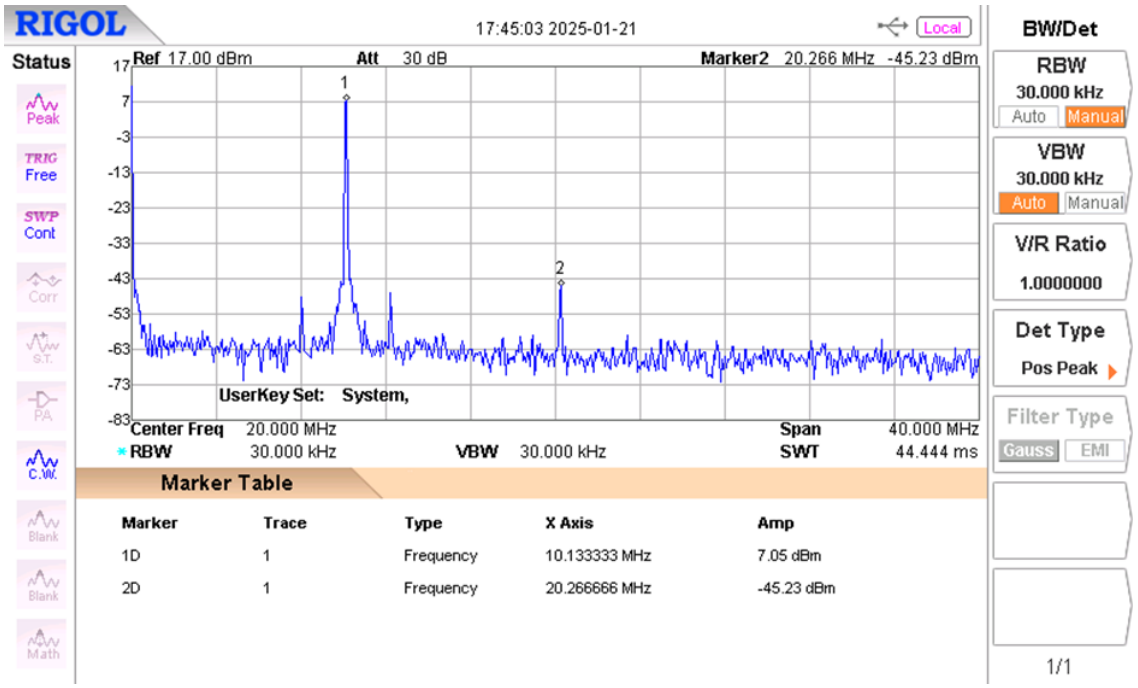



FIGURE 6.2.2 – Mesure du spectre de sortie de l'amplificateur

Lors des tests suivants, 3 capacités ont été testées, 5 pF, 8 pF et 10 pF, cependant à chacun de ces essais le système ne supportait pas la surtension, 13 MOSFET et une carte furent perdu durant la quête de puissance.

> « Tel Icare qui voulut voler trop près du soleil, Valentin Mercenaro a brûlé ses MOFSET » - Valentin Mercenaro

Le système possède un potentiel de production de puissance théoriquement supérieur. Toutefois, en dépit des modifications effectuées, il reste des limites matérielles. Certains composants atteignent leurs seuils de tolérance et brûlent, rendant ainsi le système inexploitable. Par conséquent, la puissance actuelle représente la limite maximale exploitable avec la carte en l'état.

Le meilleur résultat, l'amplificateur qui a été gardé pour être intégré au circuit complet juste avant l'antenne est donc l'amplificateur ayant une puissance de 27 dBm soit environ 0,5W.

Pour cette valeur l'alimentation 12V consomme un courant de 0,1 A soit une consommation de 1,2 W.

$$\eta = \frac{0.5}{1.2} = 0{,}416\,7$$

Soit un rendement de 41,67%.

La limite des composants nous empêchant de produire plus de puissance provoque en retour un mauvais rendement. Là où un bon rendement aurait été autour de 80% nous n'avons ici que 41,67%.

### 6.2.3 Conclusion

Les résultats obtenus cette année sont en progression par rapport à l'an dernier, mais restent en dessous des attentes et peuvent encore être améliorés. Une première optimisation peut être réalisée directement sur `KiCAD` en ajoutant des pads CMS pour les MOSFET, facilitant ainsi leur remplacement.

De plus, le choix des MOSFET utilisés peut être une piste d'amélioration importante. Actuellement, des `BS170` ont été employés, mais d'autres alternatives sont proposées dans le fichier `Excel` méritent d'être étudiées. Il pourrait être utile d'analyser ces composants afin d'évaluer leur résistance aux surtensions et leur potentiel à accroître la puissance délivrée par le système.

| Data (typical) for popular MOSFETS for QRP Pwr Amps | | | | | |
|---|---|---|---|---|---|
| | 2N7000 | BS170 | IRF510 | IRF520 | STP16NF06 |
| Voltage | 60 | 60 | 100 | 100 | 60 |
| Id | 200ma | 500 | 5.6A | 9,2 | 16 |
| Rdson Ω | 1,2 | 1,2 | 0,54 | 0,25 | 0,08 |
| Coss * pF | 11 | 17 | 81 | 130 | 70 |
| Pwr Out W | 1 to 1.5 | 1 to 2 | 2 to 10 | 2 to 20 | 2 to 20 |
| Case | TO-92 | TO-92 | TO-220 | TO-220 | TO-220 |
| *Output capacitance i.e. Drain to Source Capacitance<br>Can be paralleled for more output. Calculate the parallel Coss and Rdson | | | | | |

FIGURE 6.2.3 – Extrait du `Excel` de calcul des composants

La valeur de `C2` pourrait également subir des tests de variation de capacité, sa valeur doit être entre 1 à 2 fois la valeur de la résistance de charge (Load Resistance), ici elle a été mis à 1,5 fois la LR, cependant comme vu en figure 4.2.9 sa valeur impact le MOSFET.

## 6.3 Qualification du système

Finalement, pour s'assurer que le système est conforme aux attentes du cahier des charges, plusieurs tests ont été réalisé sur le système, notamment des tests de consommation électrique et des tests d'émission WSPR.

### 6.3.1 Tests de consommation électrique

La consommation électrique est une contrainte forte de ce projet, c'est pourquoi elle a donné lieu à plusieurs tests pour obtenir la consommation du système.

**Étude de la consommation de la carte principale** Le microcontrôleur propose plusieurs modes basse consommation : le « Stop Mode » ou le « Sleep Mode » entres autres. La première manipulation va consister à tester plusieurs mode « Low Power » pour déterminer lequel est le plus performant. Pour ce faire, la carte est reliée à un micro-ampèremètre pour mesurer

précisément la consommation de la carte en mode « Low Power ». Les résultats sont rassemblés dans le tableau ci-dessous.

| Mode | Consommation [µA] |
|---|---|
| Sleep | 22,4 |
| Stop | 18,3 |

Le Mode de basse consommation qui est retenu, est celui, où le moins de courant est consommé. Donc le Stop Mode soit une consommation de 18 µA est conservé pour le passage en « Low Power » (voir Figure 5.3.2).

Ce qui est raisonnable dans le cadre de ce projet où la carte sera sur batterie sur une station de mesure en mer munie d'un panneau solaire.

Toutefois, il est à noter que la documentation du microcontrôleur annonce une consommation typique en « Stop Mode » de 2,1 µA avec la RTC et la rétention de la RAM. Plusieurs facteurs peuvent expliquer ce décalage. Une première piste pourrait être une erreur dans la mesure. La piste la plus probable est sans doute une consommation excessive provenant du régulateur de tension, des leds ou des résistances présentent sur la carte.

**Étude de consommation du système** Par la suite, des mesures de consommation ont été effectuées sur l'ensemble des composants du système. Les résultats sont synthétisés dans la tableau ci-dessous.

| Composant | Consommation en émission [mA] | Consommation au repos [mA] |
|---|---|---|
| Amplificateur | 100 | – |
| Générateur de signal | 24 | 18 |
| µC | 2,112 | 0,018 3 |
| Carte principale sans µC | 0,017 5 | 0,017 5 |
| Total | 126,130 | 18,035 8 |

### 6.3.2 Tests en conditions réels

Afin de mettre un terme aux tests du système, celui-ci a été testé dans des conditions réels. Les tests se sont étendus sur 5 jours du 21/01/2025 au 25/01/2025. La figure 6.3.1 montre la carte des stations ayant reçu le signal. Cette figure est issue du site `https://wspr.rocks/`.

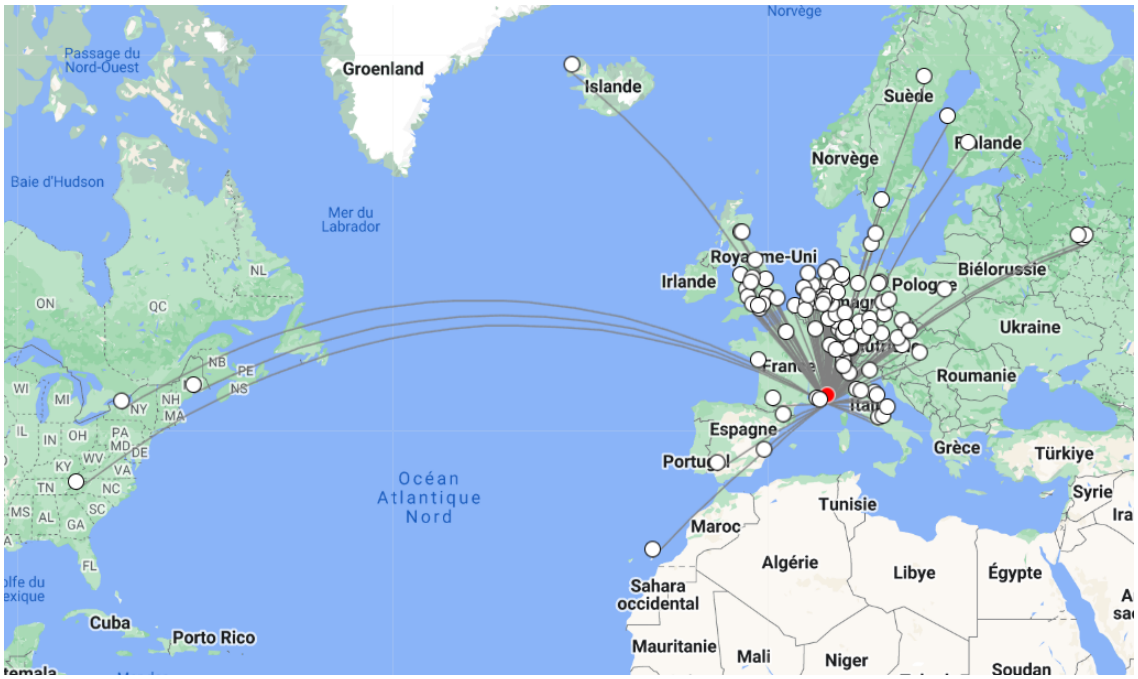



FIGURE 6.3.1 – Carte des stations ayant reçu notre signal

Plus précisément, notre signal a été reçu 2705 fois sur 5 jours. De plus, la station la plus éloignée ayant reçu notre signal se trouve à Waynesville dans le Tennnessee soit à plus de 7 340 km. Toutefois, bien que satisfaisant les résultats sont en dessous des attentes. En effet, le programme a planté plusieurs fois et la puissance du signal émis semble être inférieur à celle émise par l'amplificateur. La principale piste qui pourrait expliquer ce décalage est la mauvaise qualité de la connectique de l'antenne. La figure 6.3.2 présente une analyse de la part des messages WSPR reçu au moins une fois dans le monde pour chaque heure de la journée. Sachant, que l'absence de réception à 17h correspond au moment où le programme a planté ou au moment où nous avons dû intervenir sur le système.

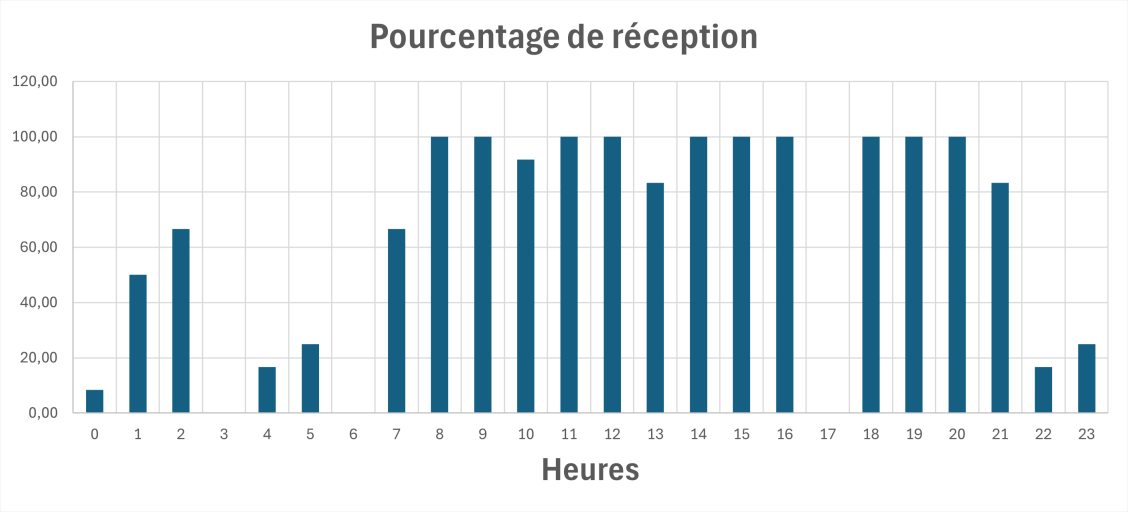



FIGURE 6.3.2 – Analyse heure par heure de la part des messages reçus au moins une fois

Ce graphique montre que quasiment aucune réception n'a été enregistrée entre 3 et 6 heures du matin. Il est vrai que la transmission HF sur la bande des 30m peut être altérée suivant

qu'il fasse nuit ou jour. Toutefois, les taux sont bien trop faibles pour que la propagation des ondes soit la seule explication. À ce jour, nous n'avons pas trouvé d'explication.

# 7 Conclusion

Le projet de transmission de données à très grande distance basé sur le protocole WSPR a permis de mettre en œuvre un système fonctionnel, capable d'émettre des messages avec une puissance faible tout en atteignant des distances, dépassant les 7 000 kilomètres. L'utilisation du microcontrôleur STM32L051x, combinée à une carte de transmission spécialement conçue, a permis de répondre aux exigences de faible consommation d'énergie et de stabilité du signal, essentielles pour des applications autonomes et de longue durée.

Les tests réalisés ont démontré l'efficacité du système, avec des résultats encourageants en termes de portée et de fiabilité des transmissions. Cependant, certaines limitations ont été identifiées, notamment en ce qui concerne la stabilité du logiciel embarqué et la qualité de la connectique de l'antenne, qui ont impacté les performances globales du système. Ces points pourront faire l'objet d'améliorations futures, notamment en optimisant le code embarqué pour éviter les plantages et en renforçant la qualité des connectiques de l'antenne.

En conclusion, ce projet a permis de valider la faisabilité d'un système de transmission WSPR autonome et basse consommation, tout en ouvrant des perspectives intéressantes pour des applications pratiques, notamment dans le domaine des stations de mesure en mer ou des systèmes de surveillance environnementale.