Rapport Projet Court FFT

Noé Dubois & Yasmine Amangar & Laith Alkhaer

1. La Recherche

Avant de commencer quoi que ce soit, il a fallu comprendre le problème et faire des recherches pour savoir comment avancer. Voici quelques points que nous avons identifié :

- Quel système pour échantillonner l'entrée micro ? A quelle fréquence échantillonner ?
- Quel algorithme pour la FFT ?
- Comment détecter les harmoniques maximales ?
- Comment mesurer le temps d'exécution ?
- Quelles limites la carte impose

Nous avons donc fait une série de recherches et avons réfléchi sur les problèmes, pour arriver à ces réponses :

- Il faut simplement récupérer le micro sur une entrée avec un objet AnalogIn, et lui demander un échantillon toutes les 25 us pour avoir une Fe de 40kHz, ce qui équivaut à 2 fois la fréquence max devant être détectée (20kHz). Il s'agit d'une simple application du théorème de Shannon. On se servira de la classe Ticker de Mbed qui nous permet de déclencher des interruptions à rythme régulier sans avoir à toucher aux timers nous-même.
- Pour la FFT, il existe de nombreux algorithmes, nous avons commencé par vouloir implémenter FFTw, qui a la particularité d'être rapide, mais il était défini dans une bibliothèque que nous n'aurions pas pu exporter sur la carte, et le code était trop gourmand en données. Nous nous sommes rabattu sur un code environ 3 fois moins rapide, donné sur la page web https://riptutorial.com/algorithm/example/27088/radix-2-fft. Il a l'avantage d'être très léger, et très compréhensible.
- Il suffit de regarder le module (ou module carré pour économiser des calculs) de chaque valeur du tableau que l'on obtiendra après FFT, et de sélectionner le(s) plus grand(s). Ensuite, pour retrouver la fréquence, il suffit de multiplier l'indice par Fe/N (N nombre d'échantillons).
- On pourra simplement utiliser la class TImer de Mbed.
- Nous avons une mémoire accessible limitée ! (32ko), Il faut donc faire attention à la mémoire que nous utilisons pour ne pas dépasser cet espace.

2. Le Développement

Nous avons créé un nouveau projet sur Keil, et commencé quelques tests pour s'échauffer : Tester l'écran, puis réaliser un simple code qui affiche sur l'écran la valeur sortant du micro. Tout fonctionnait, l'écran affichait des valeurs entre 0 et 1, et quand nous faisions saturer le micro, la valeur restait à 1.

Ensuite, nous avons mis en place l'échantillonnage, et les interruptions avec le Ticker. Le but était simplement d'afficher sur l'écran les premières valeurs du tableau d'échantillon à chaque fois

qu'on remplissait celui-ci. Simplement, rien ne fonctionnait, l'écran restait vide… Nous avons mis au moins une heure à trouver le problème : La fonction appelée par l'interruption à une fréquence de 40KHz était trop longue à exécuter ! Nous avions oublié la faible puissance de calcul de la carte quand nous avions évoqué ses limites. Le programme ne quittait donc jamais les interruptions pour revenir à l'exécution du main…

Alors, nous pouvions faire 2 choix différent : soit baisser la Fe, soit rendre la fonction d'interruption la plus courte à exécuter possible. En choisissant la deuxième option, le problème n'était pas tout à fait réglé. Nous avons décidé qu'à la fin de l'échantillonnage de tout le tableau, le programme d'interruption mettrait lui-même fin aux interruptions. Nous pensons donc que la carte n'échantillonne pas vraiment à 40kHz, limité par ses capacités, mais avec ce système, elle échantillonne le plus rapidement possible, afin de capter les plus hautes fréquences possibles.

Une fois ce problème réglé, nous avons importé dans notre projet le code de la FFT de la page https://riptutorial.com/algorithm/example/27088/radix-2-fft. IL était fonctionnel, mais il y avait un problème : ce code se sert de tableaux de complexes (étant constitués de deux doubles) comme entrée et sortie de la fonction FFT. Hors, notre mémoire est limitée à 32kHz, à raison de deux doubles (8o) par complexe, de deux tableaux (un pour les échantillons, un pour le résultat de la FFT) de complexe de taille N (nombre d'échantillons), on a besoin de N * 2* 2* 8 = 32N octets de mémoires. N doit être une puissance de 2, on avait alors au maximum N = 512…

Ce n'est pas le maximum de ce que l'on pouvait faire. Comme le code était très bien expliqué et documenté, nous l'avons bien compris. Il suffisait d'adapter sur cette base pour pouvoir augmenter N. Nous avons fait deux modifications majeures commentées dans le code :

1. La classe complexe prendra maintenant des champs float et non double, ce qui fait perdre un peut en précision mais libère beaucoup de mémoire
2. Nos échantillons étant toujours réels, pas besoin de les enregistrer dans un tableau de complexes, nous avons donc modifié le code de la fonction réalisant la FFT pour qu'elle acceptes un tableau de flottants plutôt que de complexes en entrée.

Avec ces changements, il ne fallait plus q'un tabeau de flottant et un tableau des nouveaux complexes en mémoire, soit N * (4 + 2*4) = 12N octets de mémoire. Avec N = 2048, on a donc environ 25ko d'utilisé sur 32 ! N a donc été quadruplé.

Ici, nous avons eu un problème avec Keil, notre version du logiciel ne permet pas de travailler sur d'aussi gros projet, nous sommes donc passé sur l'IDE en ligne Mbed.

Une fois la FFT implémentée et fonctionnelle, il ne restait plus qu'a trouver et afficher les fréquences des harmoniques maximales. Mais à ce moment, nous avons eu le problème le plus long à résoudre de tout le projet. L'écran affichait inexplicablement 40Hz en permanence, et « inf » comme puissance spectrale. Après bien 2h à se demander ce qui ne marchait pas, nous avons compris notre erreur, toute bête. Nous avions inversé les arguments dans la fonction pow(a,b) dans le calcul du module carré. Ainsi nous avions 2^X au lieu de X^2. Ceci donnait donc des résultats trop grands, et très faux, la suite du programme ne pouvant pas marcher.

Ensuite, le projet était quasiment fini, nous avons juste implémenté le Timer et un système qui switch régulièrement l'affichage de la seconde ligne entre 2^e harmonique maximale et temps de calcul.

3. Le Résultat

Nous avons produit une code entièrement commenté, se servant de la fonction FFT de https://riptutorial.com/algorithm/example/27088/radix-2-fft modifiée par nos soins. Le résultat est conforme aux exigences, la détection de l'harmonique principale commet des erreurs de l'ordre de quelques pourcents. Sur l'écran, il est alternativement affiché l'harmonique max sur la première ligne, et alternativement la deuxième harmonique max et le temps de calcul de la FFT sur la deuxième ligne.

Nous vous avons fourni le projet Mbed en annexe, si vous voulez une démonstration complète, nous vous avons aussi fourni une vidéo montrant le bon fonctionnement de la carte.