Institut de Technologie
Enseignement de Promotion sociale

Microcontrôleurs

Dossier récapitulatif

Programmation en C de filtres numériques sur PIC18F8722

Etudiante : Laura BINACCHI
Professeur : Eric BOSLY
Cours : Microcontrôleurs 2020-2021

Table des matières

1 Cahier des charges

1.1 Travail demandé

Ecrire un programme en langage C sous MPLABX XC8 pour filtrer un signal sonore mono envoyé à partir d'un fichier .wav sur l'entrée analogique d'un PIC. Ce programme devra tourner sur un PIC18F8722 en simulation Proteus. Ce PIC pilotera un convertisseur NA de sortie permettant d'écouter le résultat des filtres. On respectera les règles de bonne syntaxe vues au cours, notamment en termes de commentaires.

Un fichier Proteus d'exemple et des fichiers son ont été fournis. L'interface utilisateur du programme est laissée à l'appréciation du développeur, je propose ci-dessous un menu circulaire utilisant le LCD et des boutons poussoirs. Ne perdez cependant pas trop de temps dans ce domaine, ce n'est pas le principal. Le plus important dans ce dossier, sera la mise au point des quatre filtres demandés, programmation et tests.

1.2 Cahier des charges de l'application

Un signal sonore en mono, généré par une application de type Audacity avec une fréquence d'échantillonnage de 8 ou 16 kHz, est injecté sur une entrée analogique du PIC. Le signal est numérisé sur 8 bits par le module CAN du PIC et filtré en temps réel selon le mode de fonctionnement courant. Le signal filtré est recomposé par un CNA MCP4922 ou DAC0808 au choix, à piloter. Le résultat est audible dans un graphe audio, exportable.

Connecter un afficheur LCD 4 lignes de 20 caractères, le programme aura plusieurs modes de fonctionnement, résumés dans le tableau ci-dessous.

La fréquence d'échantillonnage F_e du signal sera configurable à 8kHz ou 16kHz. La fréquence d'échantillonnage conditionne évidement les fréquences de coupure des filtres.

Le mode « Configuration » et les menus sont laissés à l'appréciation du concepteur. Attention aux conflits potentiels entre le LCD et le convertisseur CNA, si ils utilisent tous les deux une connexion SPI.

1.3 Description des 4 filtres demandés

Variables	Y : tampon sortie N : dimension du tampon A, B : coefficients	X : tampon entrée
Moyenne glissante	$Y_j = \sum_{i=0}^{N-1} \frac{X_{j-i}}{N}$	pour N = 2 à 8
Filtre récursif	$Y_j = \sum_{i=0}^{N-1} A_i X_{j-i} - \sum_{i=1}^{N-1} B_i X_{j-i}$	voir fichier tableur joint
Echo	$Y_j = AX_j + BX_{j-del}$	avec $A + B = 1$

Le menu décrit ci-dessous est exemplatif, pour éviter de perdre trop de temps avec Proteus, vous pouvez limiter la complexité de l'interface homme-machine.

Mode courant	Valeurs	Exemples d'affichage
Configuration	Fréquence d'échantillonnage F_e = 8000 ou 16000 Hz	Configuration Fe : 16000Hz
1.Filtre moyenne glissante	Nombre de valeurs sommées, on peut se limiter à 2, 4, 8 pour profiter des divisions par rotation de bits.	Moyenne Nr Val : 4
		MG Running
2.Filtre récursif passe bas	On choisira la fréquence de coupure en accord avec la fréquence d'échantillonnage (Shannon) dans une gamme préétablie.	Passe Bas FC : 1000Hz
		PB Running
3.Filtre récursif passe haut	On choisira la fréquence de coupure en accord avec la fréquence d'échantillonnage (Shannon) dans une gamme préétablie.	Passe Haut FC : 300Hz
		PH Running
4.Echo	Sélectionner la durée de l'écho et le nombre d'échos (1, 2 ou 3).	Echo Delay : 500 msec
		Echo Running

2 Organigramme de haut niveau

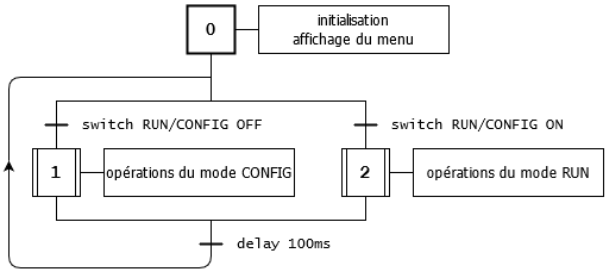
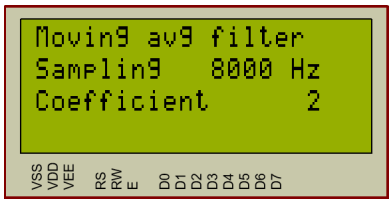


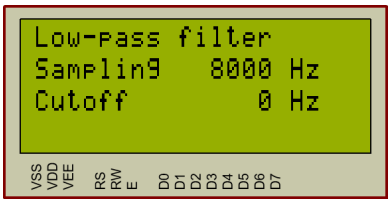
FIGURE 1 – Organigramme : boucle principale

Le programme démarre par l'initialisation du PIC **[0]**. La configuration du PIC et des périphériques (écran LDC et DAC) est détaillée dans la section 4. Les paramètres de traitement du signal et les variables de gestion du menu sont initialisés avec des valeurs par défaut définies pour la plupart (quand cela est possible) par des variables du préprocesseur. Puis le menu est affiché sur l'écran LCD :

- le type de filtre est affiché sur la première ligne ;
- la fréquence d'échantillonnage est affichée sur la deuxième ligne ;
- les paramètres du filtre sont affichés sur la troisième et, uniquement dans le cas du filtre echo, sur la quatrième ligne.



(a) Filtre par moyenne glissante



(b) Filtre passe-bas

(c) Filtre passe-haut

(d) Filtre echo

FIGURE 2 – Vues des différents menus

En fonction de l'état du switch RUN/$\overline{\text{CONFIG}}$, le programme fonctionne soit en mode de configuration **[1]**, soit en mode de traitement du signal **[2]**. L'état du switch est scanné en boucle après l'exécution des instructions du mode de fonctionnement sélectionné et une temporisation de 100 ms.

2.1 Mode de configuration

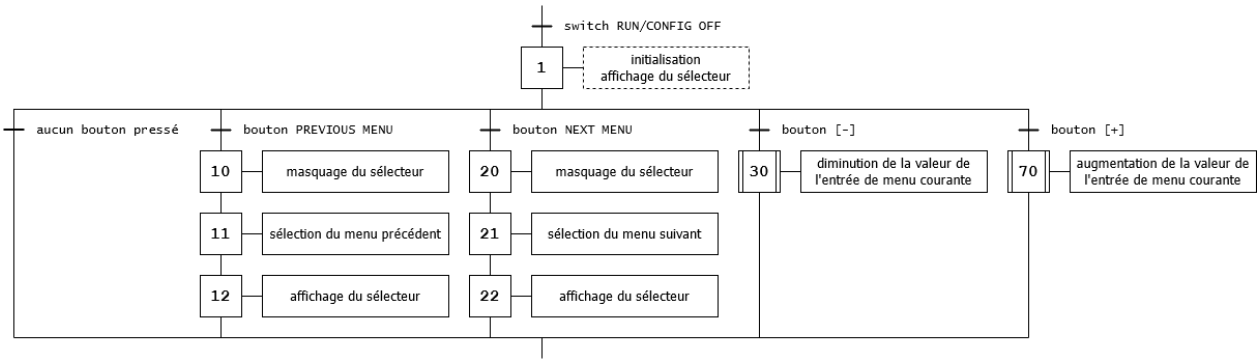


FIGURE 3 – Organigramme : mode de configuration

A son initialisation **[1]**, le mode de configuration désactive les interruptions de haute priorité, i.e. le traitement du signal, l'entrée de menu active est initialisée au type de filtre et le sélecteur de menu < est affiché sur la ligne correspondante. Grâce à une variable conservant l'état précédant du switch RUN/$\overline{\text{CONFIG}}$, le mode de configuration n'est initialisé que lors du passage du mode RUN vers le mode CONFIG.

Les différents boutons de paramétrage du filtre sont scannés un à un. Si aucun bouton n'est pressé, le programme continue de scanner les entrées tant que le mode de configuration est actif. Si plusieurs boutons sont pressés en même temps, seul le premier est pris en compte selon l'ordre dans lequel ils sont scannés.

Les boutons `PREVIOUS` et `NEXT` permettent de naviguer dans le menu. Lorsque le bouton `PREVIOUS` est pressé puis relâché, le sélecteur de menu est effacé **[10]**. L'entrée de menu précédente est sélectionnée **[11]** en mettant à jour la variable du programme qui garde en mémoire l'entrée de menu courante. Puis le sélecteur est affiché sur la nouvelle entrée active **[12]**. Si la première entrée du menu est active lorsque le bouton `PREVIOUS` est pressé, c'est la dernière entrée du menu qui devient active.

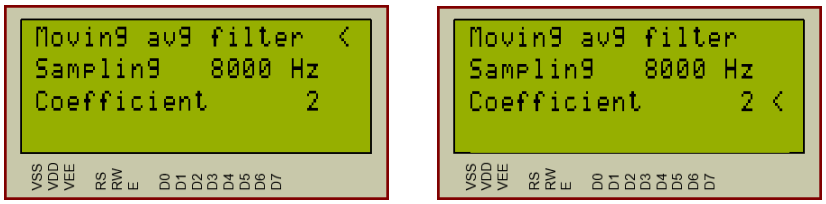


FIGURE 4 – Passage du sélecteur de la première à la dernière entrée lorsque le bouton `PREVIOUS` est pressé

De la même manière, lorsque le bouton `NEXT` est pressé puis relâché, le sélecteur de menu est effacé **[20]**, l'entrée de menu suivante est sélectionnée **[21]** puis le sélecteur est affiché sur la nouvelle entrée de menu active **[22]**.

Les boutons `[-]` et `[+]` permettent de modifier le type de filtre et ses paramètres en diminuant **[30]** ou augmentant **[70]** la valeur de l'entrée de menu active.

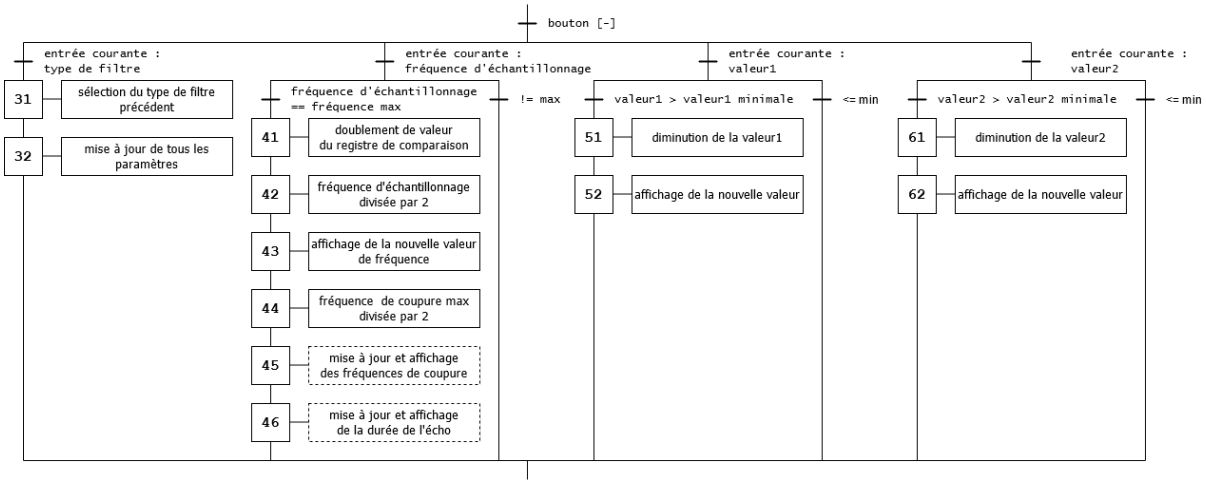


FIGURE 5 – Organigramme : diminution de la valeur de l'entrée de menu active

La modification du type de filtre **[31]** se fait, comme la navigation, de manière circulaire : il tourne en boucle en appuyant sur le bouton `[+]` ou sur le bouton `[-]`, dans un sens de défilement ou dans l'autre. Lorsque le type de filtre est modifié, c'est tout l'affichage qui est mis à jour **[32]** car les paramètres configurables dépendent du type de filtre actif.

La fréquence d'échantillonnage n'a que deux valeurs possibles : 8 kHz ou 16 kHz. Elle n'est donc diminuée que si elle est au maximum (de la même manière, elle ne sera augmentée que si elle est au minimum). La valeur de comparaison de l'interruption est doublée **[41]** et la valeur de la fréquence d'échantillonnage est divisée par deux **[42]** puis mise à jour sur le LCD **[43]**. La fréquence d'échantillonnage détermine la fréquence de coupure maximale qui peut être utilisée par les filtres : ce maximum est fixé à la moitié de la fréquence d'échantillonnage **[44]** (théorème de Shannon). Si les fréquences de coupure des filtres passe-bas et passe-haut dépassent cette valeur maximale, elles prennent cette nouvelle valeur et, si filtre actif est un filtre passe-bas ou passe-haut, le LCD est mis à jour **[45]**. La fréquence d'échantillonnage détermine également la valeur maximale que peut prendre la durée de l'écho : lorsque la fréquence est divisée par deux, la durée de l'écho est doublée et si le filtre écho est sélectionné, la nouvelle valeur est affichée **[46]**.

La diminution de la première valeur met à jour **[51]**, en fonction du filtre actif, soit : le coefficient du filtre par moyenne glissante ; la fréquence de coupure du filtre passe-bas ; la fréquence de coupure du filtre passe-haut ; le délai du filtre echo.

La deuxième valeur **[61]** ne concerne que le filtre echo et met à jour le nombre d'échos.

Ces valeurs ne peuvent être diminuées que si elles sont strictement supérieures aux valeurs minimales définies par des variables du préprocesseur. Elles sont diminuées d'un pas également défini par des variables du préprocesseur. Le même mécanisme est implémenté pour l'augmentation des valeurs.

2.2 Mode de traitement du signal

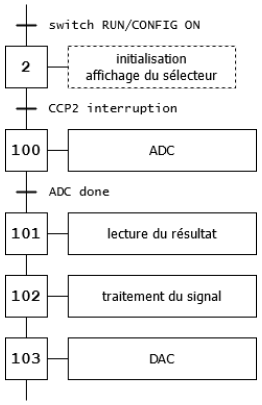


FIGURE 6 – Organigramme : mode de traitement du signal

Comme le mode de configuration, le mode de traitement du signal n'est initialisé que lors du passage du mode `CONFIG` au mode `RUN` **[2]** : le sélecteur de menus < est masqué et le signal est initialisé en fonction du type de filtre actif et de ses paramètres.

Une interruption survient toutes les 125 µs (fréquence d'échantillonnage à 8 kHz) ou toutes les 62.5 µs (fréquence d'échantillonnage à 16 kHz). Elle déclenche la conversion analogique-numérique (ADC) **[100]**. Quand elle est terminée, son résultat est lu dans le tampon d'entrée **[101]**. Le signal est alors traité selon le type de filtre et ses paramètres **[102]** et la résultat du traitement est placé dans le tampon de sortie pour pouvoir être transmis au convertisseur numérique analogique (DAC) **[103]**.

3 Fréquences utilisées

Le cahier des charges demande d'implémenter des fréquences d'échantillonnage de 8 et de 16 kHz. Conformément au théorème de de Nyquist-Shannon les fréquences utiles des signaux sont donc respectivement limitées à 4 et à 8 kHz.

Les différents signaux testés ont été échantillonnés à 8 ou à 16 kHz permettant de se passer d'un filtre anti-repliement à l'entrée du signal. En sortie, les filtres de Laplace sont réglés à la fréquence utile du signal testé.

Les analyses de Fourier présentent également le spectre de fréquence sur la plage de fréquence utile du signal, soit 8 ou 16 kHz.

La fréquence d'échantillonnage est implémentée par le programme en déclenchant une interruption à un intervalle déterminé : toutes les 125 µs pour du 8 kHz ou toutes les 62.5 µs pour du 16 kHz. La configuration de l'interruption est détaillée à la section 4.2.

4 Configuration du PIC

Le PIC est configuré par les instructions d'initialisation de ses différents registres aux lignes 250 à 270 du fichier *main.c* et par les directives de préprocesseur (*pragmas*) aux lignes 28 à 100.

4.1 Oscillateur

La première directive de préprocesseur configure l'oscillateur en mode HSPLL. Ce mode multiplie par 4 la fréquence de l'oscillateur. Une fréquence de 10 MHz correspond donc à un temps de cycle CPU de 0.1 µs.

```
[29]    #pragma config OSC = HSPLL
```

4.2 Interruption

Le traitement du signal est effectué par une interruption générée par le module CCP2 et le Timer1. Le module CCP2 est activé en mode comparateur avec special event trigger : à chaque interruption, le timer est réinitialisé et la conversion analogique-digitale est déclenchée automatiquement.

```
[258]    CCP2CON = 0x0B;         // CCP2 interruption in compare mode
                                         with trigger special event
```

Le Timer1 est activé avec un prescaler 1 :1 : une valeur de comparaison de 1 déclencherait une interruption tous les cycles CPU, i.e. toutes les 0.1 µs.

```
[261]    T1CON   = 0x01;         // Timer1 prescale 1:1
[262]    TMR1H   = 0;            // Timer1 -> 0
[263]    TMR1L   = 0;
```

La valeur de comparaison est initialisée à 1250 (`0x04E2`). L'interruption surviendra donc toutes les 125 µs, ce qui correspond à une fréquence de 8 kHz. Pour fonctionner à une fréquence d'échantillonnage de 16 kHz, cette valeur est divisée par 2.

```
[259]    CCPR2H  = 0x04;         // 1250us -> 8kHz
[260]    CCPR2L  = 0xE2;
```

Les interruptions prioritisées sont activées et les interruptions de haute et de basse priorité sont désactivées. Les interruptions de haute priorité seront activée après l'initialisation si le mode de traitement du signal est actif.

```
[264]    RCONbits.IPEN   = 1;    // enable priority levels on interrupts
[265]    INTCONbits.GIEH = 0;    // high priority interrupts disabled
[266]    INTCONbits.GIEL = 0;    // low priority interrupts disabled
```

Une haute priorité est accordée à l'interruption par le module CCP2. L'interruption est activée (et son flag effacé). Elle ne sera effectivement active que quand les interruptions de haute priorité auront été activées.

```
[267]    IPR2bits.CCP2IP = 1;    // CCP2 interrupt: high priority
[268]    PIE2bits.CCP2IE = 1;    // CCP2 interrupt enabled
[269]    PIR2bits.CCP2IF = 0;    // CCP2 interrupt flag cleared
```

4.3 Conversion analogique-digitale

La conversion analogique-digitale est activée par le registre `ADCON0` qui sélectionne également l'entrée analogique au convertisseur : la canal 0. Le registre `ADCON1` configure la PIN AN0 en analogique (le minimum nécessaire) et garde les tensions de référence par défaut : A_{VDD} et A_{VSS} (0 et 5 V).

```
[255]    ADCON0  = 0X01;         // ADC enabled on AN0
[256]    ADCON1  = 0x0E;         // AN0 -> analogic
[257]    ADCON2  = 0x01;         // Left justification, 0 Tad, Fosc/8
```

Le registre `ADCON2` sélectionne la justification à gauche du résultat de la conversion dans les registres `ADRESH` et `ADRESL` : le programme travaille en 8bits et ne conserve donc que les 8 bits de poids forts lus dans le registre `ADRESH`.

Le registre `ADCON2` permet aussi de définir le temps d'acquisition et la fréquence de conversion. Le temps d'acquisition est configuré à 0 T_{AD} car les conversions sont effectuées selon la fréquence des interruptions (maximum à 16 kHz) : il se sera donc toujours passé assez de temps entre les conversions. La fréquence de conversion est choisie en fonction de la fréquence de l'oscillateur et du type de PIC conformément à la table 21-1 (*TAD vs. DEVICE OPERATING FREQUENCIES*) de la datasheet. Elle est de 8 T_{OSC} pour une fréquence d'oscillateur de 10 MHz en respectant la recommendation de la datasheet de choisir le plus petit temps supérieur au minimum requis.

4.4 Bus SPI et périphériques

Le bus SPI et les périphériques qui y sont associés sont initialisés par la fonction `init_SPI()` du fichier *SPI.c*. Cette fonction est reprise de la librairie de gestion de la communication d'un PIC avec un LCD fournie pour les projets précédents.

4.5 I/O

Les boutons de gestion du menu sont connectés directement au PORTE du PIC. Le registre TRISE est donc configuré en input.

```
[250]    TRISE   = 0xFF;         // PORTE (menu buttons) -> input
```

Le projet aurait pu implémenter une lecture des entrées via le bus SPI et une interruption de faible priorité. Ce n'est pas le cas car le programme principal utilise déjà le bus SPI pour communiquer avec le LCD et les communications entreraient en conflit.

La PIN permettant de vérifier le temps d'exécution de l'interruption est configurée en output et initialisée à 0.

```
[251]    TRISGbits.TRISG0 = 0;   // TICK -> output
[252]    TICK    = 0;            // TICK -> 0
```

Enfin, bien que le projet permette de générer un signal via le convertisseur digital-analogique MCP4922, le DAC0808 n'a pas été supprimé car tous les tests présentés ont été fait avec ce DAC.

```
[253]    TRISD   = 0;            // DAC0808 -> output
[254]    DAC0808 = 128;          // DAC0808 -> 0 + offset
```

5 Tests

Tout au long du développement, le programme a été testé avec la simulation Proteus et les différents outils à disposition. L'oscilloscope branché sur les différentes PIN a permis de vérifier la génération d'un signal en sortie des DAC et en sortie des filtres de Laplace mais aussi de s'assurer que la durée de traitement ne dépasse pas la durée de l'interruption grâce à la PIN activée durant le traitement du signal ou encore de débugger les communications via le bus SPI.

Une fois la programmation du filtre terminée, la sortie des filtres a été validée avec les analyses de Fourier et les analyses de signaux audio générées dans Proteus. Ce sont les résultats de ces tests qui sont présentés dans cette section. Ils ont été produits en changeant les paramètre d'initialisation du filtre avant la compilation du programme. Tous les tests réalisés se trouvent dans le code source et il suffit de décommenter le test souhaité pour le reproduire.

Pour les différents tests, les fréquences de coupures des filtres de Laplace en sortie des DAC ont été réglées soit à 8 kHz soit à 4 kHz en fonction de la fréquence d'échantillonnage testée (ce paramètre est affiché sous le filtre dans le projet Proteus).

Tous les signaux utilisés pour les tests et tous les signaux générés par les tests se trouvent dans le dossier *wav* du projet.

5.1 Filtre par moyenne glissante

Les tests du filtre par moyenne glissante ont été effectués sur l'audio *drake_30s_delay_1s_8k.wav* pour les tests faits avec une fréquence d'échantillonnage de 8 kHz et *drake_30s_delay_1s_16k.wav* pour une fréquence d'échantillonnage de 16 kHz. Le délai d'une seconde permet de laisser le temps au programme de s'initialiser.

Le filtre par moyenne glissante implémente assez peu de combinaisons possibles de coefficient et de fréquence d'échantillonnage que pour pouvoir les tester toutes.

Que ce soit en 16 kHz ou en 8 kHz, les tests mettent en évidence que le filtre est d'autant plus filtrant que son coefficient est élevé. Plus le coefficient augmente, plus le signal de sortie est atténué par rapport au signal d'entrée. Les fréquences du signal de sortie sont également de plus en plus affaiblies à mesure que le coefficient du filtre augmente.

Les hautes fréquences sont par contre plus puissantes pour le signal de sortie que pour le signal d'entrée : il s'agit du bruit de quantification. Ce bruit n'est pas lié au filtre mais au traitement numérique d'un signal analogique : il apparaît également quand le signal lu est directement reporté sur le DAC.

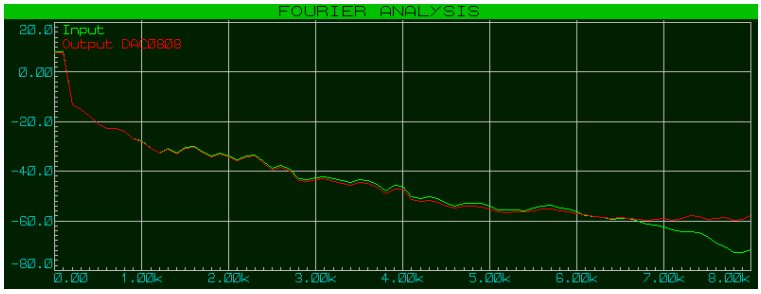


FIGURE 7 – Spectre des signaux d'entrée et de sortie, pas de filtre (16kHz)

Les fichiers audio produits permettent également d'entendre la dégradation progressive du signal : à mesure que le coefficient du filtre augmente, le son paraît de plus en plus étouffé. Les signaux produits sont enregistrés dans les fichiers :

- *test_mov_avg_2_16k.wav* (filtre d'ordre 2 en 16 kHz)
- *test_mov_avg_4_16k.wav* (filtre d'ordre 4 en 16 kHz)
- *test_mov_avg_8_16k.wav* (filtre d'ordre 8 en 16 kHz)
- *test_mov_avg_2_8k.wav* (filtre d'ordre 2 en 8 kHz)
- *test_mov_avg_4_8k.wav* (filtre d'ordre 4 en 8 kHz)
- *test_mov_avg_8_8k.wav* (filtre d'ordre 8 en 8 kHz)

Dans tous les tests, le signal de sortie apparaît légèrement déphasé par rapport au signal d'entrée. Cet effet est dû au filtre de Laplace placée en sortie en plus du temps de traitement du signal. Plus le coefficient du filtre est élevé, plus le temps de traitement augmente et plus les signaux sont déphasés.

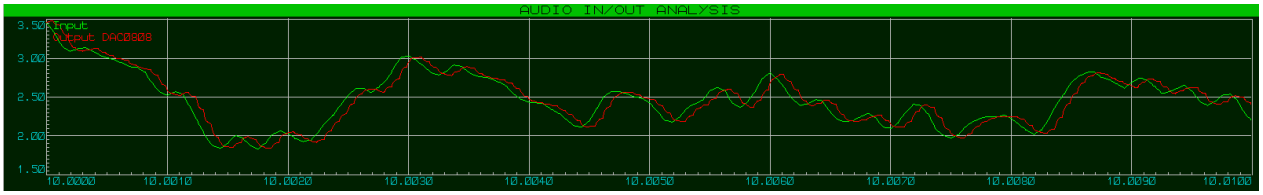
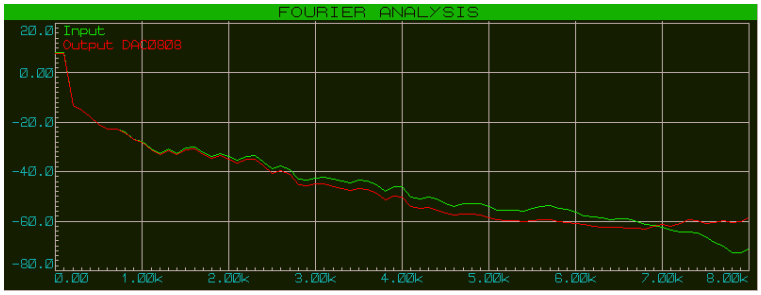
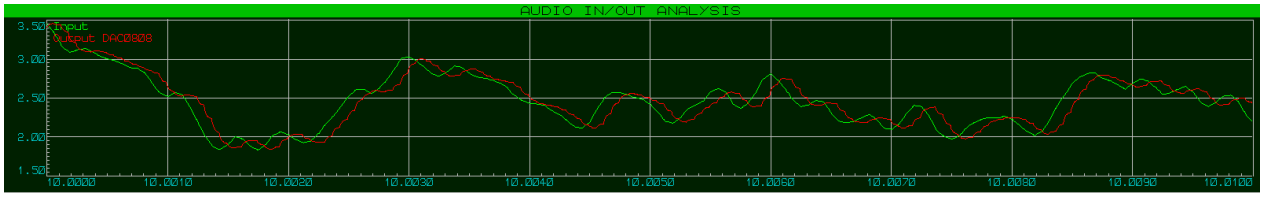


FIGURE 8 – Signal de sortie déphasé par rapport au signal d'entrée, pas de filtre (16kHz)

5.1.1 Test : filtre d'ordre 2 (16kHz)



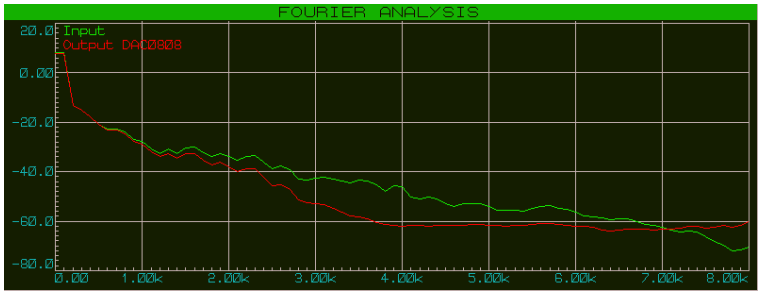
(a) Analyse fréquentielle



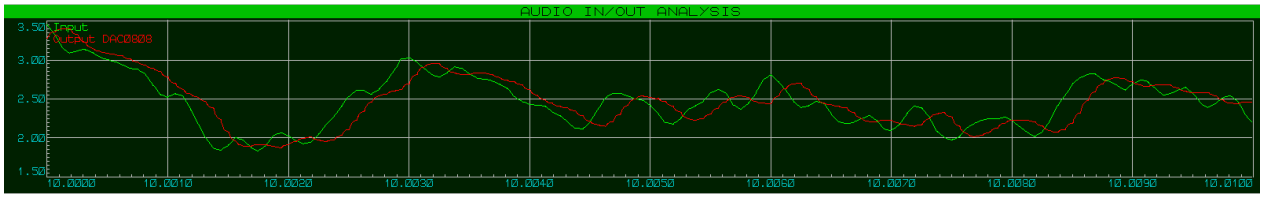
(b) Analyse temporelle

FIGURE 9 – Test du filtre par moyenne glissante d'ordre 2 (16kHz)

5.1.2 Test : filtre d'ordre 4 (16kHz)



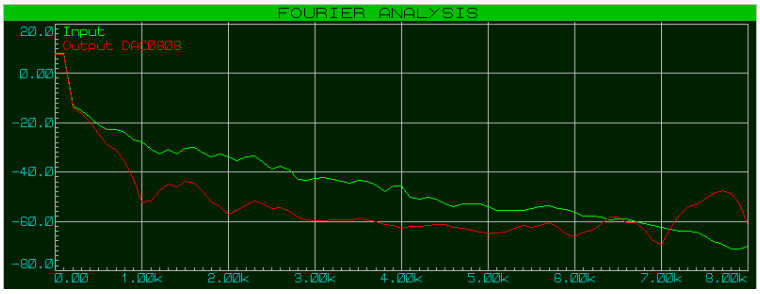
(a) Analyse fréquentielle



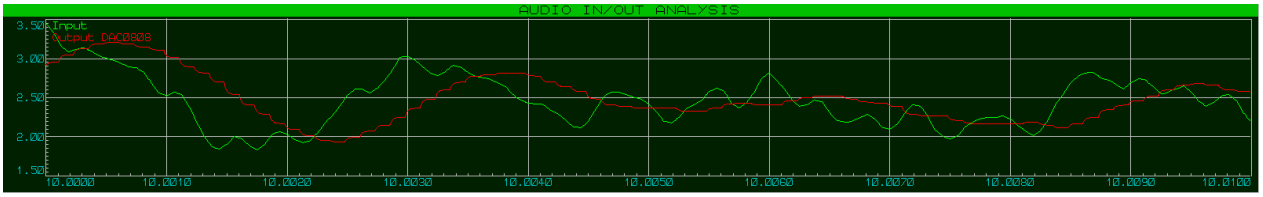
(b) Analyse temporelle

FIGURE 10 – Test du filtre par moyenne glissante d'ordre 4 (16kHz)

5.1.3 Test : filtre d'ordre 8 (16kHz)



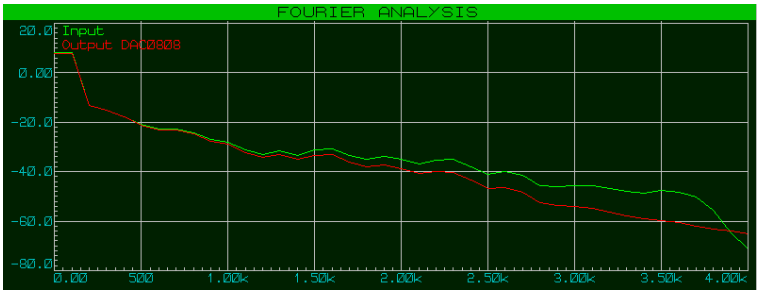
(a) Analyse fréquentielle



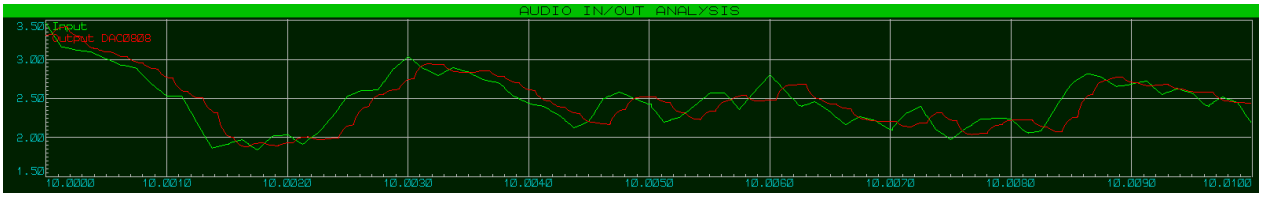
(b) Analyse temporelle

FIGURE 11 – Test du filtre par moyenne glissante d'ordre 8 (16kHz)

5.1.4 Test : filtre d'ordre 2 (8kHz)



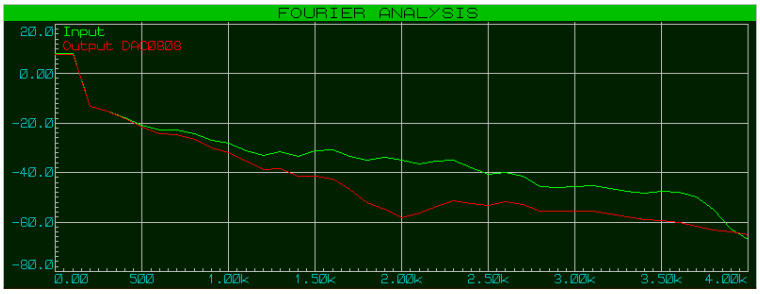
(a) Analyse fréquentielle



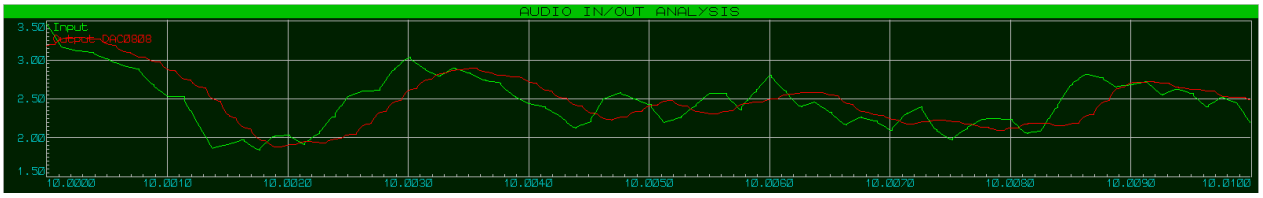
(b) Analyse temporelle

FIGURE 12 – Test du filtre par moyenne glissante d'ordre 2 (8kHz)

5.1.5 Test : filtre d'ordre 4 (8kHz)



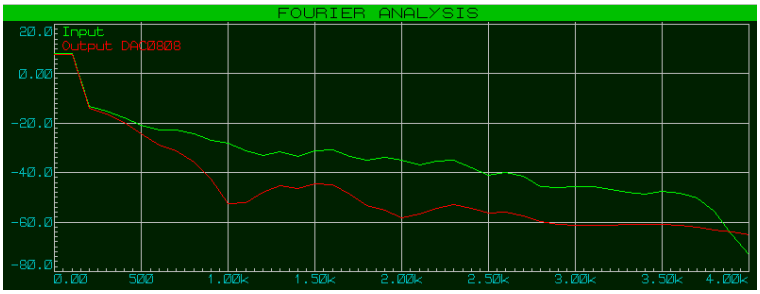
(a) Analyse fréquentielle



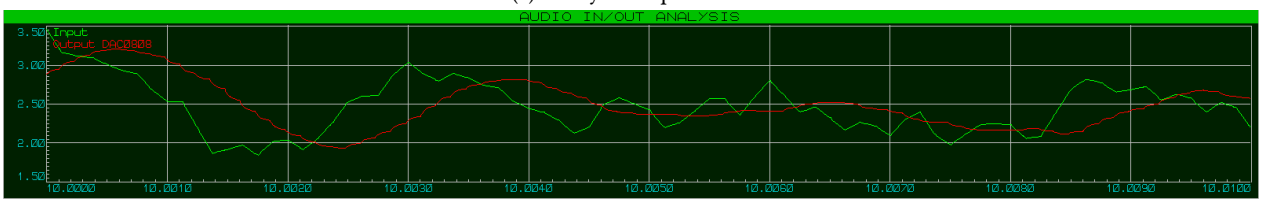
(b) Analyse temporelle

FIGURE 13 – Test du filtre par moyenne glissante d'ordre 4 (8kHz)

5.1.6 Test : filtre d'ordre 8 (8kHz)



(a) Analyse fréquentielle



(b) Analyse temporelle

FIGURE 14 – Test du filtre par moyenne glissante d'ordre 8 (8kHz)

5.2 Filtre passe-bas

Le filtre passe-bas a été testé avec des valeurs de fréquence de coupure progressives entre les deux valeurs limites : 0 et 8 kHz pour une fréquence d'échantillonnage de 16 kHz ou 4 kHz pour une fréquence d'échantillonnage de 8 kHz.

Les analyses de Fourier sont les plus intéressantes ici puisqu'elles permettent de comparer les fréquences du signal d'entrée et du signal de sortie : le filtre laisse bien passer les fréquences qui se trouvent sous la fréquence de coupure et filtre celles qui sont au-dessus.

Que ce soit en 16 ou en 8 kHz, au fur et à mesure que la fréquence de coupure augmente, le filtre devient de moins en moins filtrant et l'audio devient de plus en plus net : avec une fréquence de coupure de 0 Hz, le filtre ne laisse passer aucun signal puis le signal se rapproche de plus en plus du signal original.

Les signaux produits par les différents tests sont enregistrés dans les fichiers suivants :

- *test_lp_0_16k.wav* : fréquence de coupure de 0 Hz (16 kHz)
- *test_lp_500_16k.wav* : fréquence de coupure de 500 Hz (16 kHz)
- *test_lp_1000_16k.wav* : fréquence de coupure de 1000 Hz (16 kHz)
- *test_lp_2000_16k.wav* : fréquence de coupure de 2000 Hz (16 kHz)
- *test_lp_4000_16k.wav* : fréquence de coupure de 4000 Hz (16 kHz)
- *test_lp_8000_16k.wav* : fréquence de coupure de 8000 Hz (16 kHz)
- *test_lp_0_8k.wav* : fréquence de coupure de 0 Hz (8 kHz)
- *test_lp_250_8k.wav* : fréquence de coupure de 250 Hz (8 kHz)
- *test_lp_500_8k.wav* : fréquence de coupure de 500 Hz (8 kHz)
- *test_lp_1000_8k.wav* : fréquence de coupure de 1000 Hz (8 kHz)
- *test_lp_2000_8k.wav* : fréquence de coupure de 2000 Hz (8 kHz)
- *test_lp_4000_8k.wav* : fréquence de coupure de 4000 Hz (8 kHz)

5.2.1 Test : fréquence de coupure progressive (16kHz)

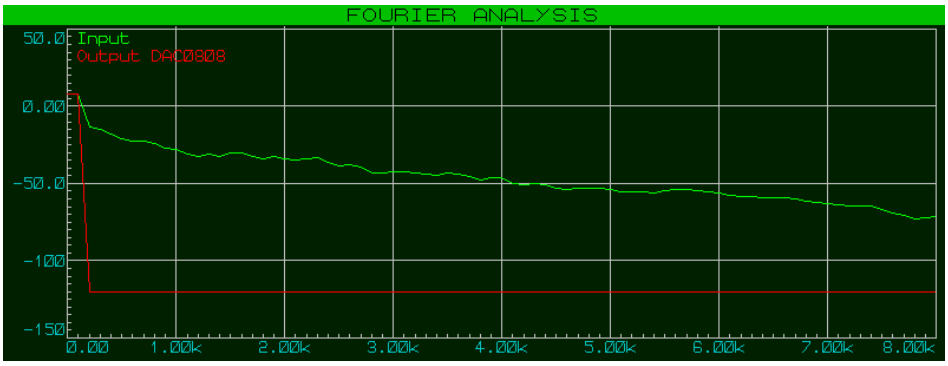


Figure 15 – Test du filtre passe-bas : fréquence de coupure de 0Hz (16kHz)

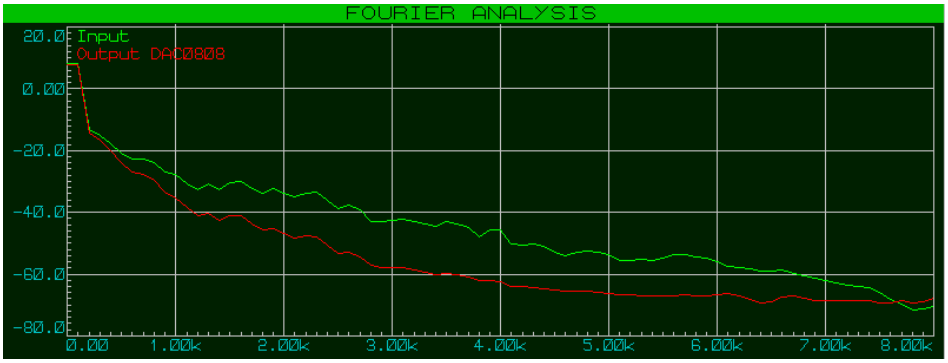


FIGURE 16 – Test du filtre passe-bas : fréquence de coupure de 500Hz (16kHz)

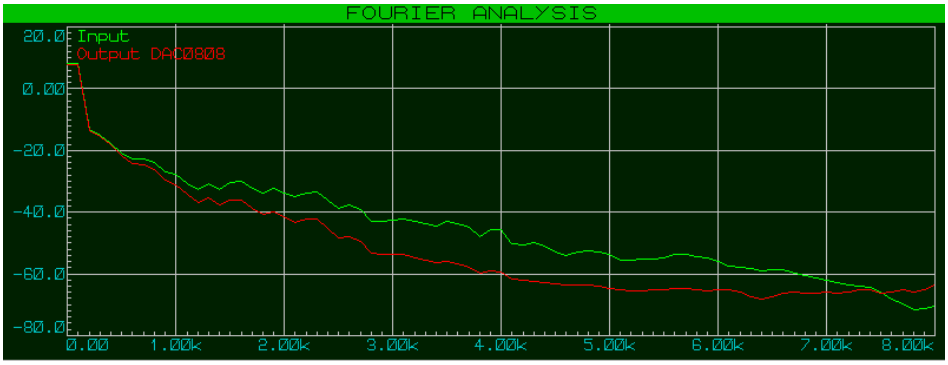


FIGURE 17 – Test du filtre passe-bas : fréquence de coupure de 1000Hz (16kHz)

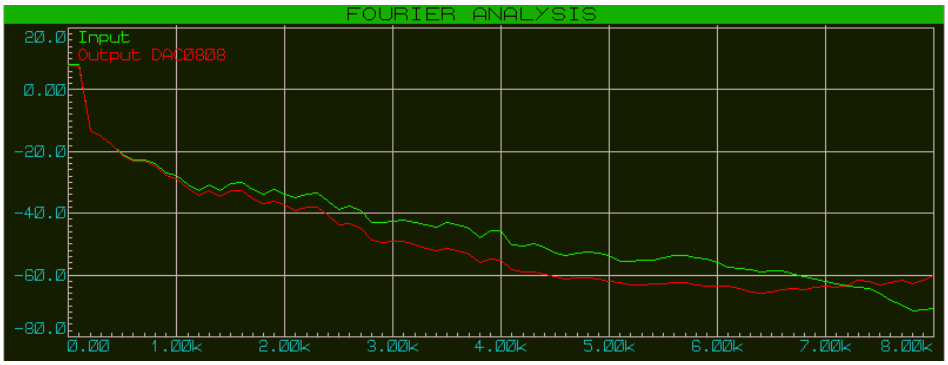


FIGURE 18 – Test du filtre passe-bas : fréquence de coupure de 2000Hz (16kHz)

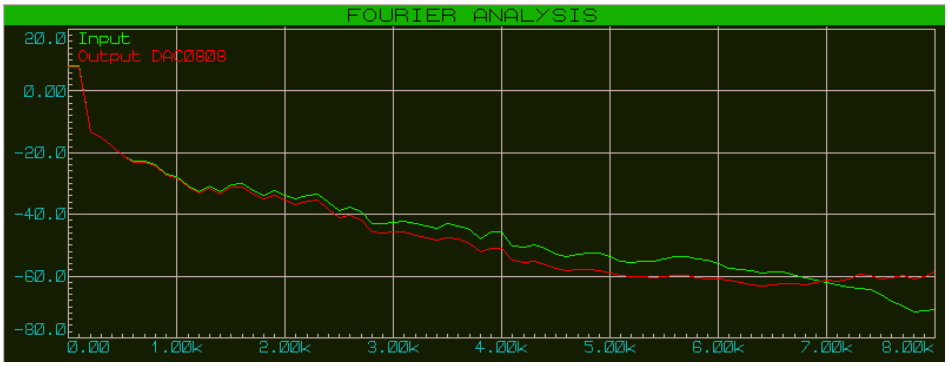


FIGURE 19 – Test du filtre passe-bas : fréquence de coupure de 4000Hz (16kHz)

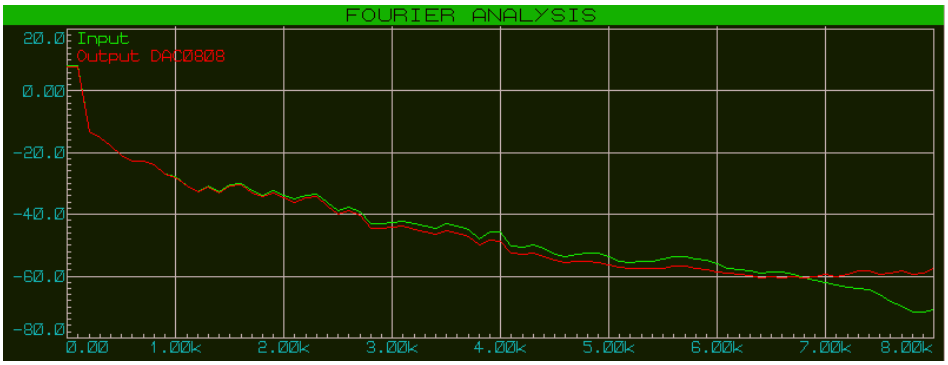


FIGURE 20 – Test du filtre passe-bas : fréquence de coupure de 8000Hz (16kHz)

5.2.2 Test : fréquence de coupure progressive (8kHz)

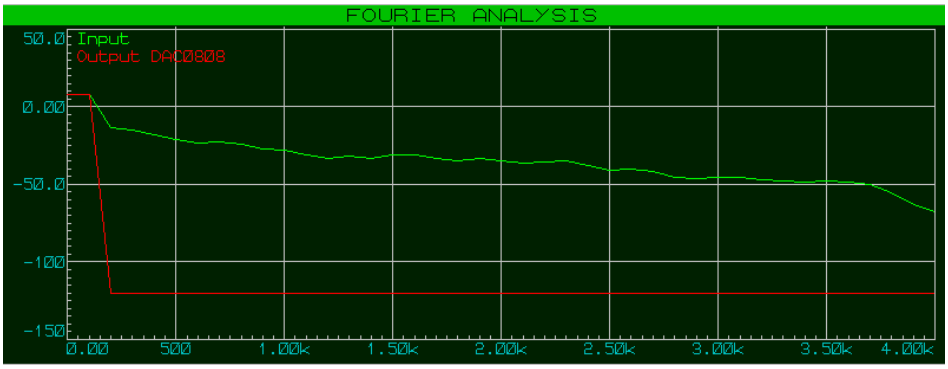


FIGURE 21 – Test du filtre passe-bas : fréquence de coupure de 0Hz (8kHz)

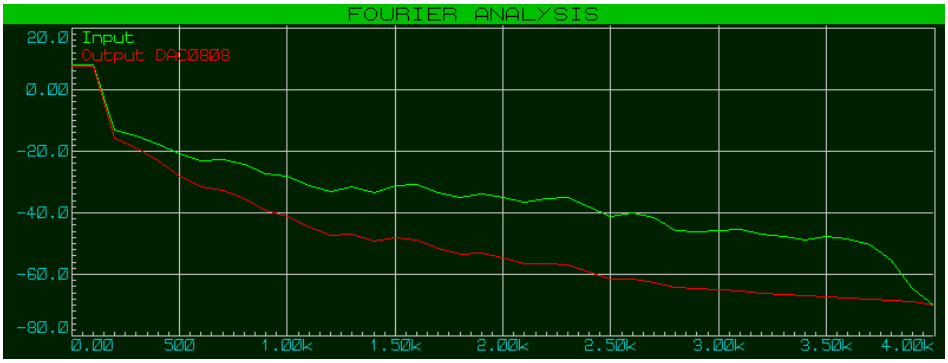


FIGURE 22 – Test du filtre passe-bas : fréquence de coupure de 250Hz (8kHz)

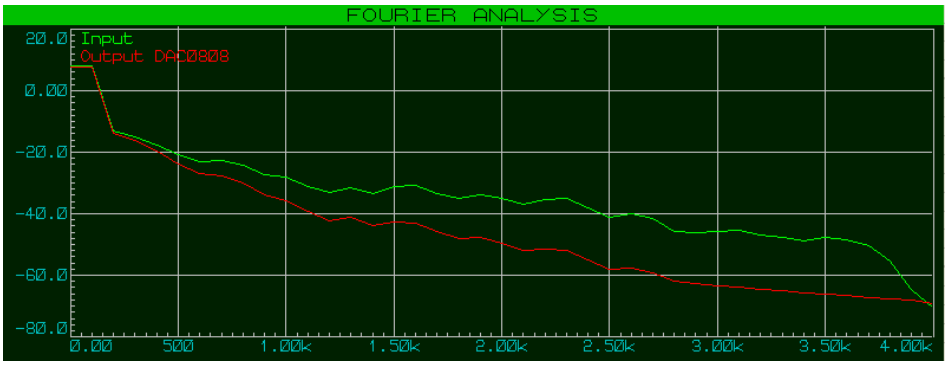


FIGURE 23 – Test du filtre passe-bas : fréquence de coupure de 500Hz (8kHz)

FIGURE 24 – Test du filtre passe-bas : fréquence de coupure de 1000Hz (8kHz)

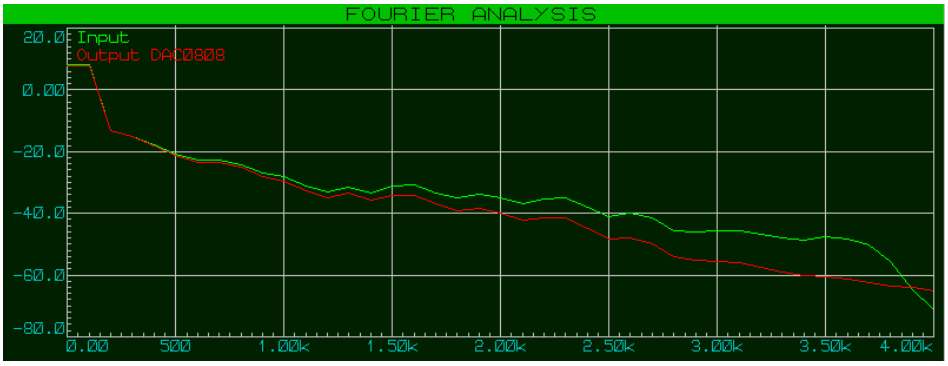


FIGURE 25 – Test du filtre passe-bas : fréquence de coupure de 2000Hz (8kHz)

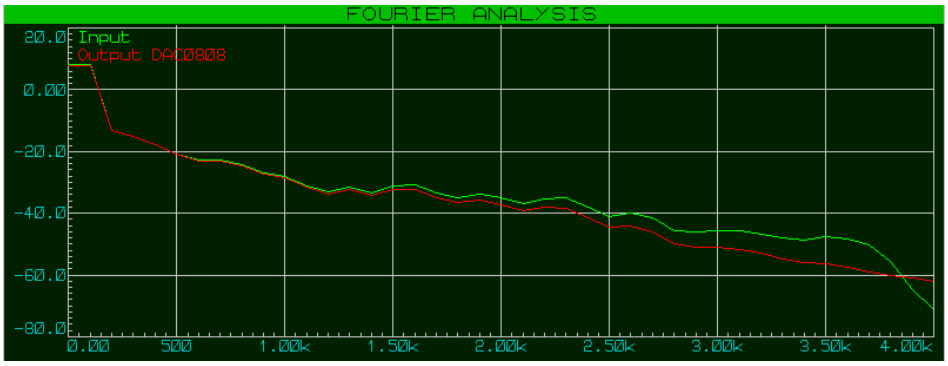


FIGURE 26 – Test du filtre passe-bas : fréquence de coupure de 4000Hz (8kHz)

5.3 Filtre passe-haut

Le filtre passe-haut a été testé comme le filtre passe-bas avec des valeurs de fréquence de coupure progressives.

Les analyses de Fourier mettent cette fois en évidence que ce sont les fréquences au-delà de la fréquence de coupure qui sont conservées et les fréquences plus basses qui sont filtrées. A l'opposé du filtre passe-bas, plus la fréquence de coupure augmente, plus le signal original est dégradé : leurs spectres de fréquences sont de plus en plus éloignés et le son paraît de plus en plus aigu.

Les signaux produits par les différents tests sont enregistrés dans les fichiers suivants :

- *test_hp_0_16k.wav* : fréquence de coupure de 0 Hz (16 kHz)
- *test_hp_500_16k.wav* : fréquence de coupure de 500 Hz (16 kHz)
- *test_hp_1000_16k.wav* : fréquence de coupure de 1000 Hz (16 kHz)
- *test_hp_2000_16k.wav* : fréquence de coupure de 2000 Hz (16 kHz)
- *test_hp_4000_16k.wav* : fréquence de coupure de 4000 Hz (16 kHz)
- *test_hp_8000_16k.wav* : fréquence de coupure de 8000 Hz (16 kHz)

- *test_hp_0_8k.wav* : fréquence de coupure de 0 Hz (8 kHz)
- *test_hp_250_8k.wav* : fréquence de coupure de 250 Hz (8 kHz)
- *test_hp_500_8k.wav* : fréquence de coupure de 500 Hz (8 kHz)
- *test_hp_1000_8k.wav* : fréquence de coupure de 1000 Hz (8 kHz)
- *test_hp_2000_8k.wav* : fréquence de coupure de 2000 Hz (8 kHz)
- *test_hp_4000_8k.wav* : fréquence de coupure de 4000 Hz (8 kHz)

5.3.1 Test : fréquence de coupure progressive (16kHz)

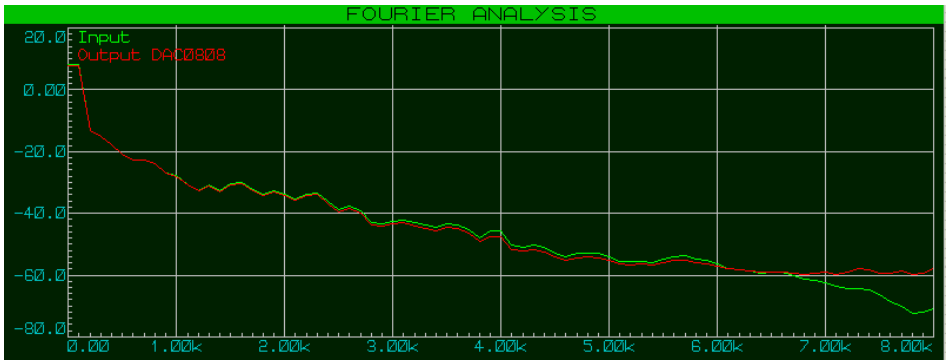


FIGURE 27 – Test du filtre passe-haut : fréquence de coupure de 0Hz (16kHz)

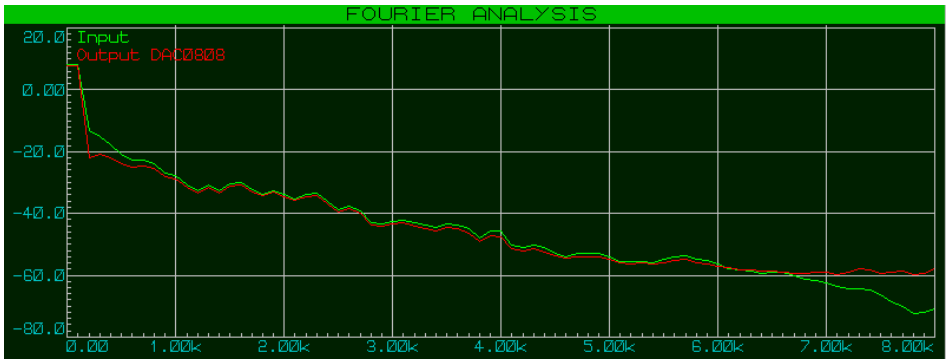


FIGURE 28 – Test du filtre passe-haut : fréquence de coupure de 500Hz (16kHz)

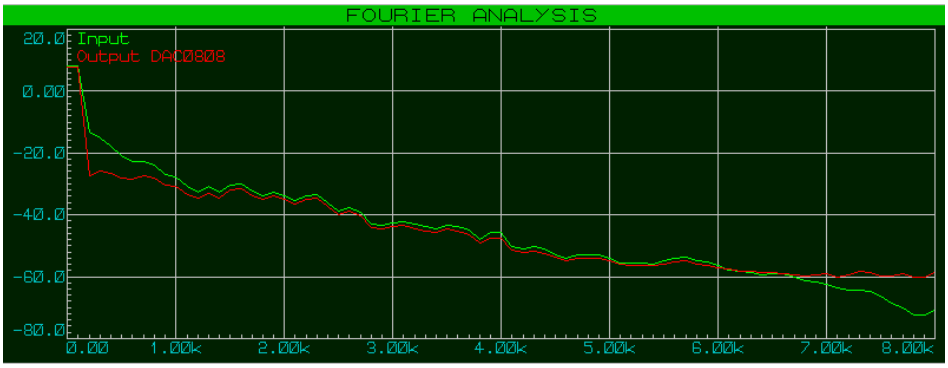


FIGURE 29 – Test du filtre passe-haut : fréquence de coupure de 1000Hz (16kHz)

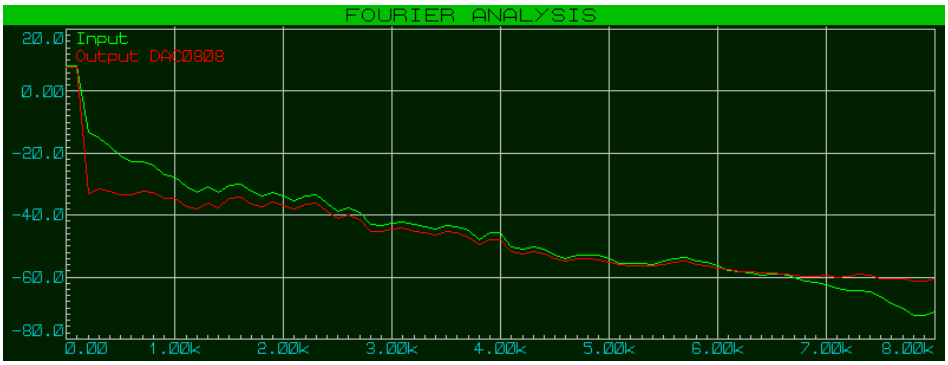


FIGURE 30 – Test du filtre passe-haut : fréquence de coupure de 2000Hz (16kHz)

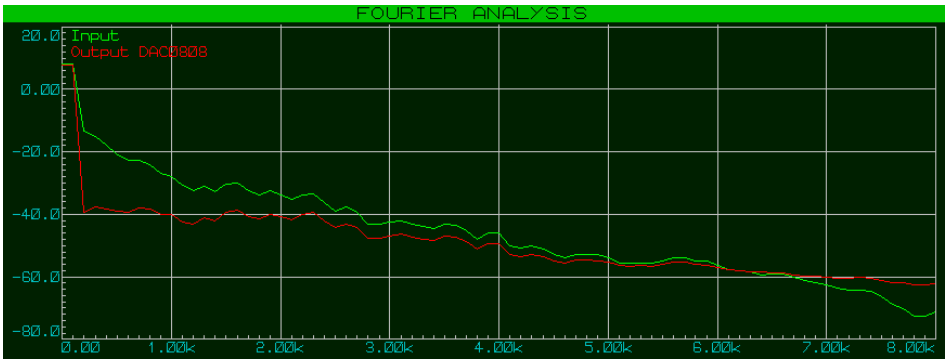


FIGURE 31 – Test du filtre passe-haut : fréquence de coupure de 4000Hz (16kHz)

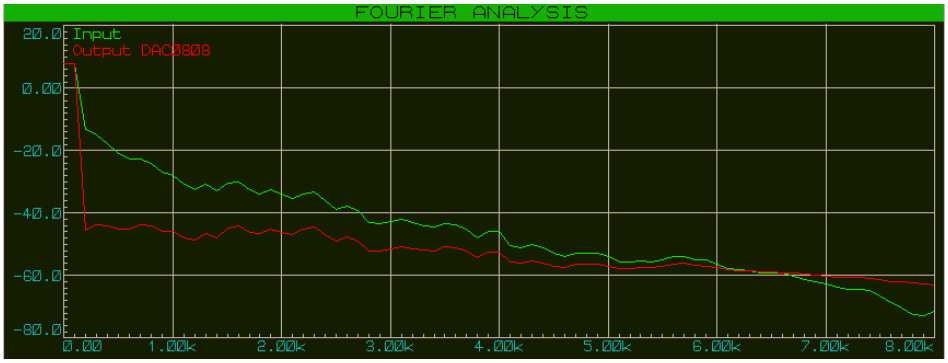


FIGURE 32 – Test du filtre passe-haut : fréquence de coupure de 8000Hz (16kHz)

5.3.2 Test : fréquence de coupure progressive (8kHz)

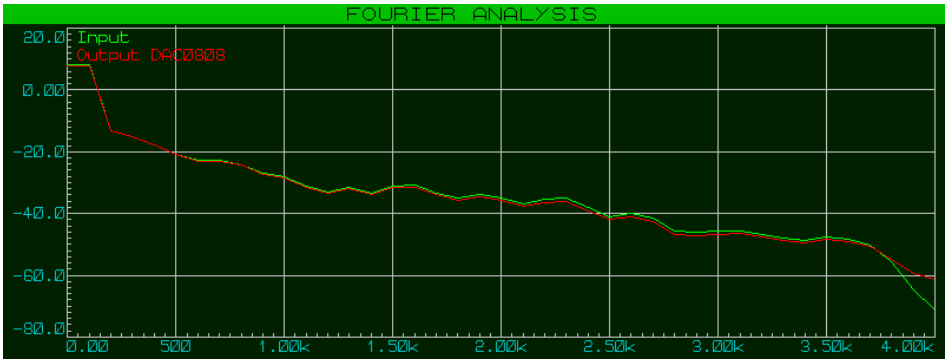


FIGURE 33 – Test du filtre passe-haut : fréquence de coupure de 0Hz (8kHz)

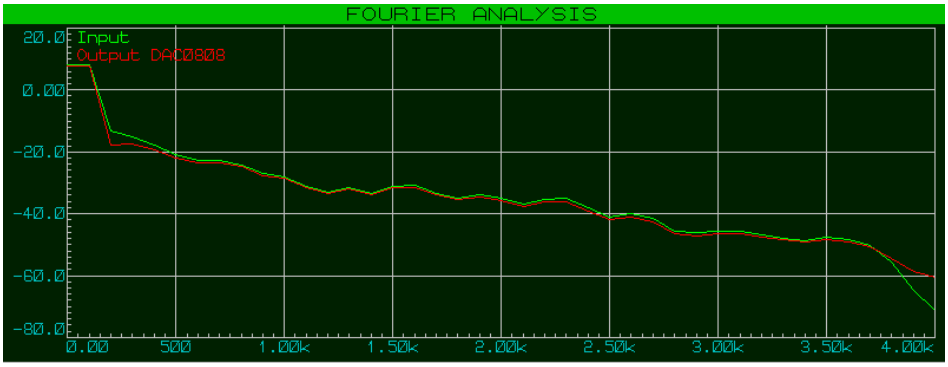


FIGURE 34 – Test du filtre passe-haut : fréquence de coupure de 250Hz (8kHz)

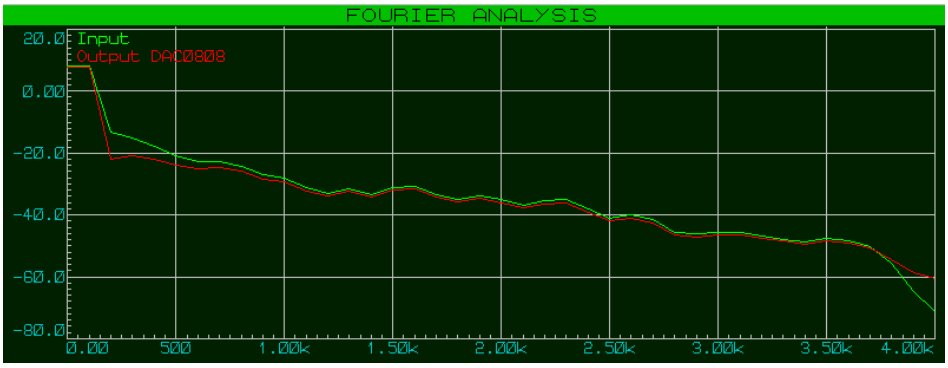


FIGURE 35 – Test du filtre passe-haut : fréquence de coupure de 500Hz (8kHz)

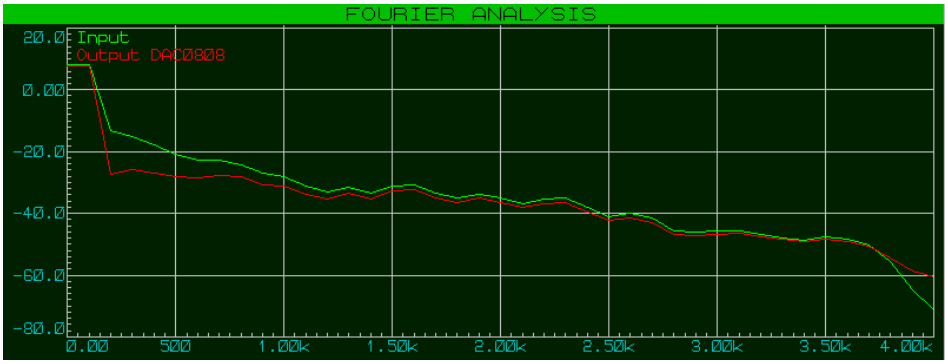


FIGURE 36 – Test du filtre passe-haut : fréquence de coupure de 1000Hz (8kHz)

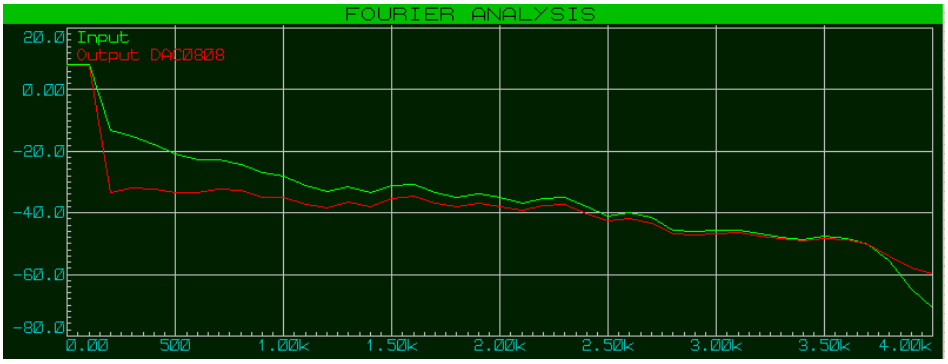


FIGURE 37 – Test du filtre passe-haut : fréquence de coupure de 2000Hz (8kHz)

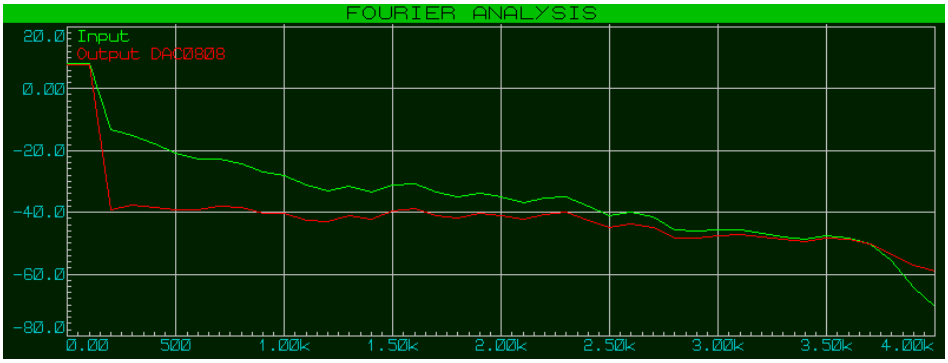


FIGURE 38 – Test du filtre passe-haut : fréquence de coupure de 4000Hz (8kHz)

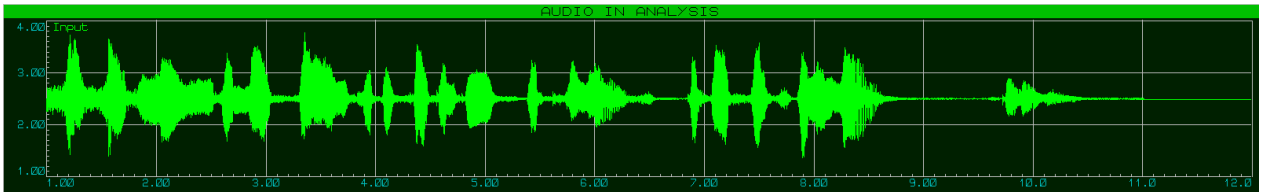
5.4 Filtre écho

Le filtre écho a été testé avec les fichiers *angelou_10s_delay_1s_8k.wav* (pour une fréquence d'échantillonnage de 8 kHz) et *angelou_10s_delay_1s_16k.wav* (pour 16 kHz).

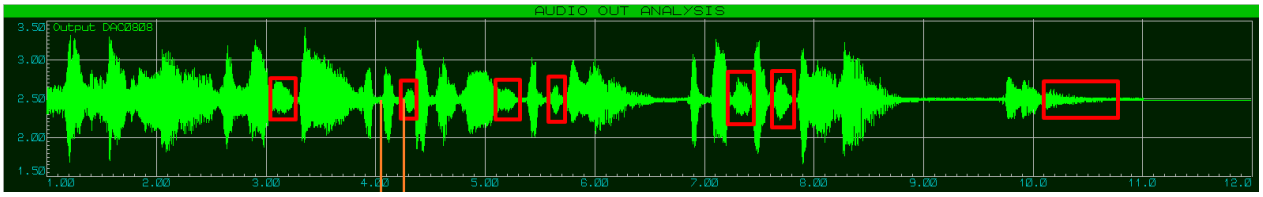
5.4.1 Test : 1 écho, durée de 200ms (16kHz)

Avec une fréquence d'échantillonnage de 16 kHz, le filtre écho peut fonctionner avec une durée maximale de 200 ms. L'audio produit permet d'entendre un léger écho (*test_echo_1_200ms_16k.wav*).

La comparaison des signaux d'entrée et de sortie de la figure 39 met en évidence l'apparition de cet écho (entouré en rouge). Cette figure vérifie également que l'écho apparaît bien 200 ms après le signal original : e.g. le deuxième écho mis en évidence sur la figure 39b démarre à 4.25 s, répercutant le signal original qui démarre à 4.05 s (lignes oranges).



(a) Signal d'entrée



(b) Signal de sortie

FIGURE 39 – Test du filtre écho : 1 écho, durée de 200ms (16kHz)

L'analyse de Fourier des signaux d'entrée et de sortie montre des spectres similaires. Dans les plus hautes fréquences, le spectre du signal de sortie s'éloigne du spectre du signal d'entrée à cause du bruit de quantification.

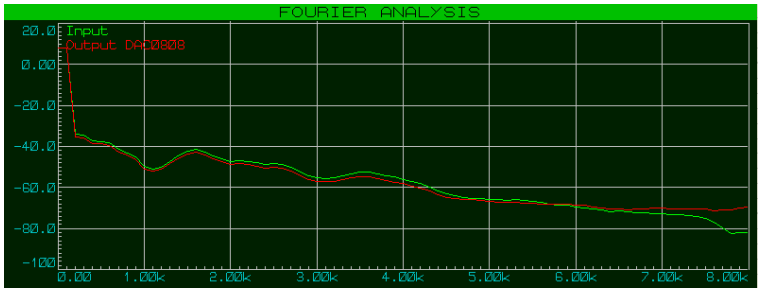
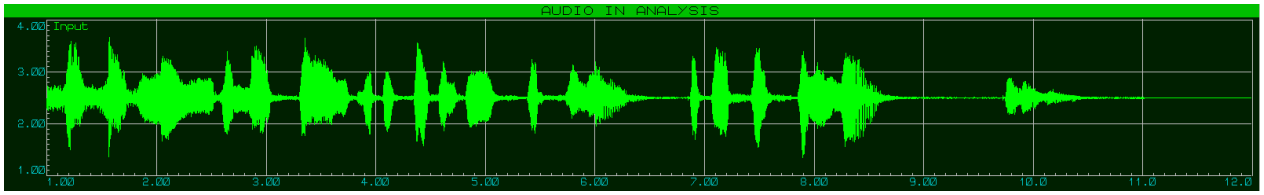


FIGURE 40 – Test du filtre écho : spectre de 1 écho, durée de 200ms (16kHz)

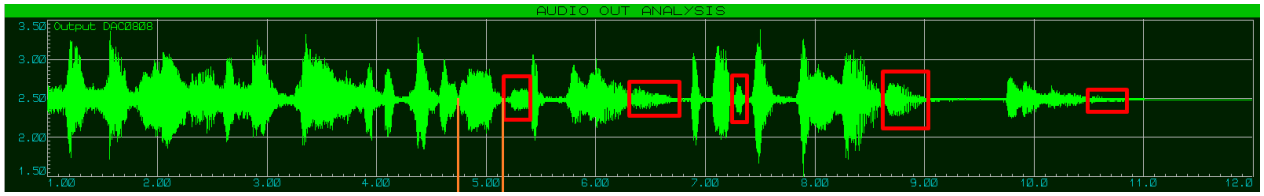
5.4.2 Test : 1 écho, durée de 400ms (8kHz)

Le filtre écho a également été testé avec un écho de la durée maximale possible pour une fréquence d'échantillonnage de 8 kHz, i.e. 400 ms. L'audio de sortie (*test_echo_1_400ms_8k.wav*) permet d'entendre clairement cet écho.

La comparaison des signaux d'entrée et de sortie met en évidence l'apparition de cet écho 400 ms après le signal original : e.g. le premier écho mis en évidence démarre à 5.12 s et répercute la portion du signal original qui débute à 4.72 s (lignes oranges).



(a) Signal d'entrée



(b) Signal de sortie

FIGURE 41 – Test du filtre écho : 1 écho, durée de 400ms (8kHz)

L'analyse de Fourier est similaire à la précédente : les spectres des signaux d'entrée et de sortie sont identiques jusqu'à 4 kHz où ils commencent à s'écarter l'un de l'autre à cause du bruit de quantification.

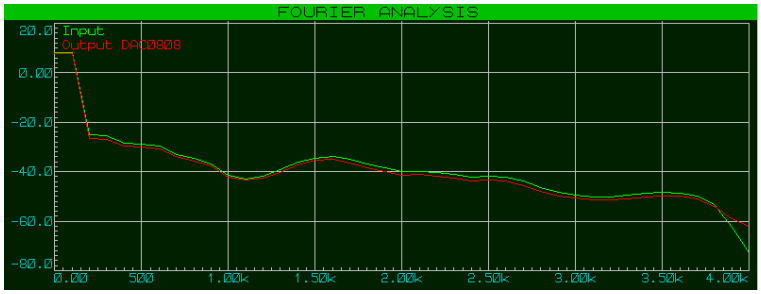


FIGURE 42 – Test du filtre écho : spectre de 1 écho, durée de 400ms (8kHz)

5.4.3 Test : 1 écho, durée de 200ms (8kHz)

Le filtre écho testé avec un écho d'une durée de 200 ms avec une fréquence d'échantillonnage de 8 kHz donne un résultat similaire au premier test. L'audio de sortie *test_echo_1_200ms_8k.wav* permet d'entendre le même écho malgré la qualité d'audio moindre. Le signal de sortie est similaire : les échos apparaissent aux mêmes endroits, décalés de 200 ms par rapport au signal original.

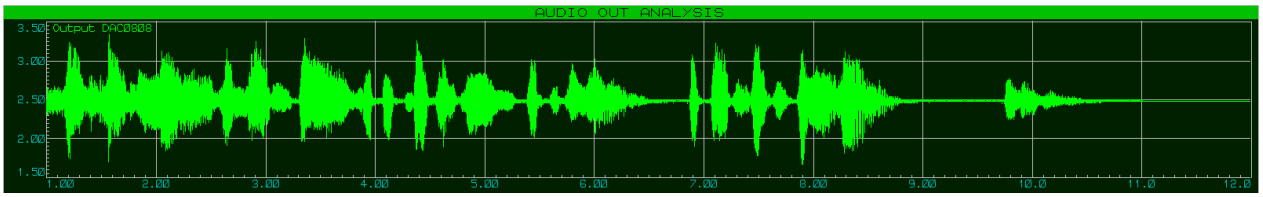


FIGURE 43 – Test du filtre écho : output de 1 écho, durée de 200ms (8kHz)

5.4.4 Test : 1 écho, durée de 100ms (8kHz)

Avec une durée de 100 ms, l'écho n'est presque plus perceptible (*test_echo_1_100ms_8k.wav*). Les échos générés sont bien visibles sur l'analyse du signal de sortie, décalés de 100 ms par rapport au signal original.

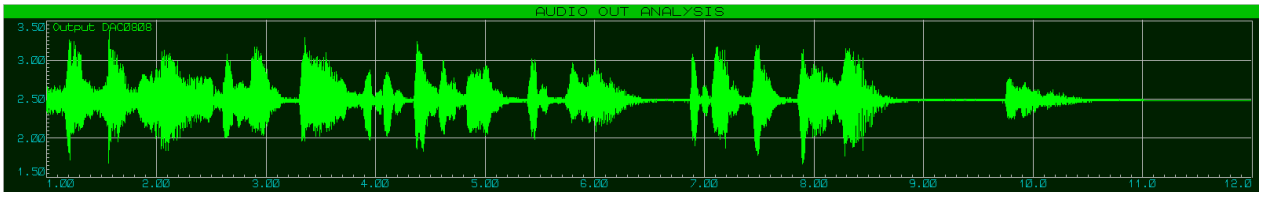


FIGURE 44 – Test du filtre écho : output de 1 écho, durée de 100ms (8kHz)

5.4.5 Test : 2 échos, durée de 400ms (8kHz)

Les échos multiples ont été testés avec une fréquence d'échantillonnage de 8 kHz puisqu'elle autorise une durée deux fois plus grande qu'à 16 kHz. La durée est réglée au maximum (400 ms) pour pouvoir distinguer au mieux le signal original et ses échos.

Comme avec un seul écho, le signal original apparaît affaibli. Les échos sont moins marqués mais apparaissent bien et surtout sont bien distinguables à l'écoute (*test_echo_2_400ms_8k.wav*).

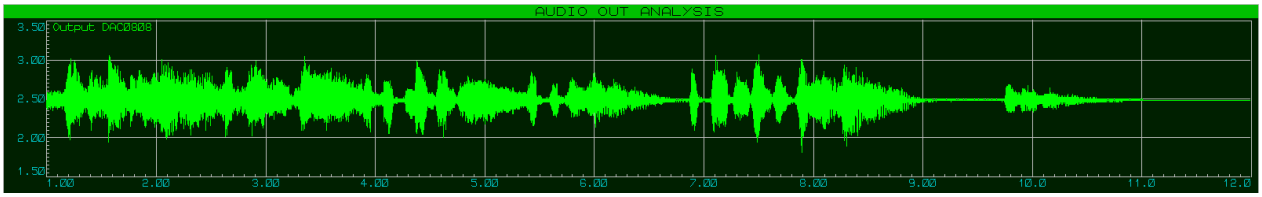


FIGURE 45 – Test du filtre écho : output de 2 échos, durée de 400ms (8kHz)

5.4.6 Test : 3 échos, durée de 400ms (8kHz)

Avec trois échos, le signal original est encore plus atténué et les échos sont plus fondus. Ils sont maintenant difficilement distinguables à l'écoute (*test_echo_3_400ms_8k.wav*).

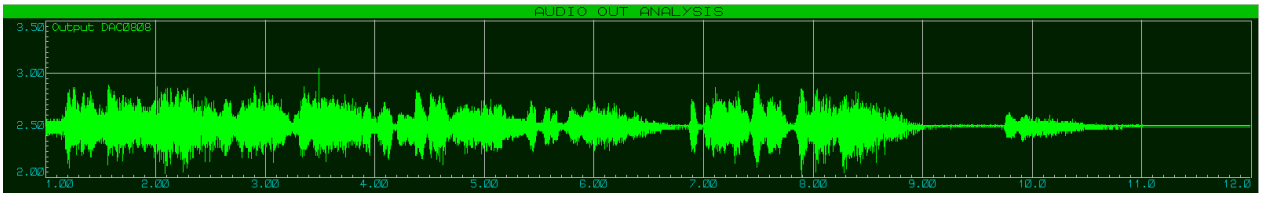


FIGURE 46 – Test du filtre écho : output de 3 échos, durée de 400ms (8kHz)

6 Difficultés rencontrées

Pour la réalisation de ce projet, j'ai principalement rencontré des difficultés avec Proteus. J'ai conçu un programme avec un menu circulaire qui s'est révélé peu adapté aux tests en simulation. J'ai donc dû recompiler le programme pour chaque test ce qui m'a fait perdre du temps (en plus des plantages réguliers de Proteus). Pour les tests en "temps réel" ou pour une application qui ne serait plus en simulation, ce type de menu reste pratique.

Le fait de proposer un menu assez riche a aussi le désavantage d'utiliser la mémoire qui pourrait être réservée aux filtres. L'espace en mémoire pour les données est limité à 3936 bytes et une programmation plus économe m'aurait permis d'implémenter un écho encore un peu plus long, sans toutefois pouvoir aller jusqu'à 500 ms. Puisque ce n'était pas un des critères demandés par le cahier des charges, je n'ai pas choisi de concevoir le programme de cette manière mais je suis consciente que des projets utilisant des ressources aussi limitée demande toujours d'être attentif·ve à leur utilisation.