

Projet Vuvuzela :

Mise en place d'un filtre «anti-vuvuzela» pour atténuer les sonorités de celui-ci

Tristan Londe
Florian Pasco

Liam Vasseur-Viton
S4A-A



Table des matières

1	Analyse du fichier sonore vuvuzela.wav	3
2	Détermination du nombre L de composantes sinusoïdales significatives	3
3	Détermination des fréquences f_l	5
4	Détermination approximative de m	6
5	Mise en œuvre du filtre	7
5.1	Choix d'une topologie de filtre rejeteur	7
5.2	Détermination des composants électroniques	7
5.3	Test LTSpice de la cascade de filtres choisis	8
5.4	Réaliser en physique de l'un des filtres	9

Table des figures

1	Spectre de magnitude de la partie de l'enregistrement choisie	3
2	Spectre de magnitude paramétré	4
3	Spectre de magnitude paramétré et avec relevé de valeur	5
4	Extrait du Chapitre 2 : Filtres d'ordre 2 - Cours de S4 Électronique	6
5	Circuit réalisé sur LTSPICE	8
6	Analyse de spectre par FFT	9
7	Cablage du filtre propre à la composante fréquentielle f_l pour 1631Hz	9
8	Oscilloscope comparant l'entrée et la sortie du circuit	10

Introduction

Lors de la coupe du Monde de Football 2010 (et de rugby de 1995), les téléspectateurs du monde entier ont pu découvrir un instrument de musique d'Afrique du Sud nommé le **Vuvuzela**. Lors de la diffusion des matchs, le son continu produit par cet instrument est rapidement devenu gênant. (exemple : <https://www.youtube.com/watch?v=bKCIFXqhLzo>).

Dans ce projet, nous allons **mettre en place un filtre « anti-vuvuzela »** pour atténuer les sonorités de cet instrument et améliorer la restitution audio des commentaires sportifs.

La solution proposée dans ce document consistera à :

- Mettre en cascade L filtres rejeteur d'ordre 2.
- Faire en sorte que chaque rejeteur supprime une composante fréquentielle f_l particulière.

1 Analyse du fichier sonore vuvuzela.wav

Nous commençons par réaliser une lecture de l'audio "vuvuzela.wav". À la suite de cette écoute, nous constatons qu'il est effectivement difficile d'entendre les commentateurs. L'amplitude du son du vuvuzela étant plus importante que celle de la voix des commentateurs son écoute n'est pas aisée.

2 Détermination du nombre L de composantes sinusoidales significatives

Afin de déterminer le nombre L de composantes sinusoidales significatives (nombre correspond à la quantité de sinusoïde servant à former le son d'un Vuvuzela), nous commençons par isoler une partie de l'enregistrement "vuvuzela.wav" où cet instrument est seul afin de minimiser le bruit créé par la voix des commentateurs. Ensuite, nous réalisons un spectre de magnitude sur cet enregistrement à l'aide de la librairie matplotlib dont nous faisons usage dans le code suivant :

```
1 from scipy.io import wavfile
2 import matplotlib.pyplot as plt
3 import IPython.display as ipd
4 import numpy as np
5
6 path = "../wav/vuvuzela_only.wav"
7 Fs, data = wavfile.read(path)
8
9 spectrum, freqs, line = plt.magnitude_spectrum(data, Fs=Fs, color='C1')
```

Et l'on obtient désormais ce spectre :

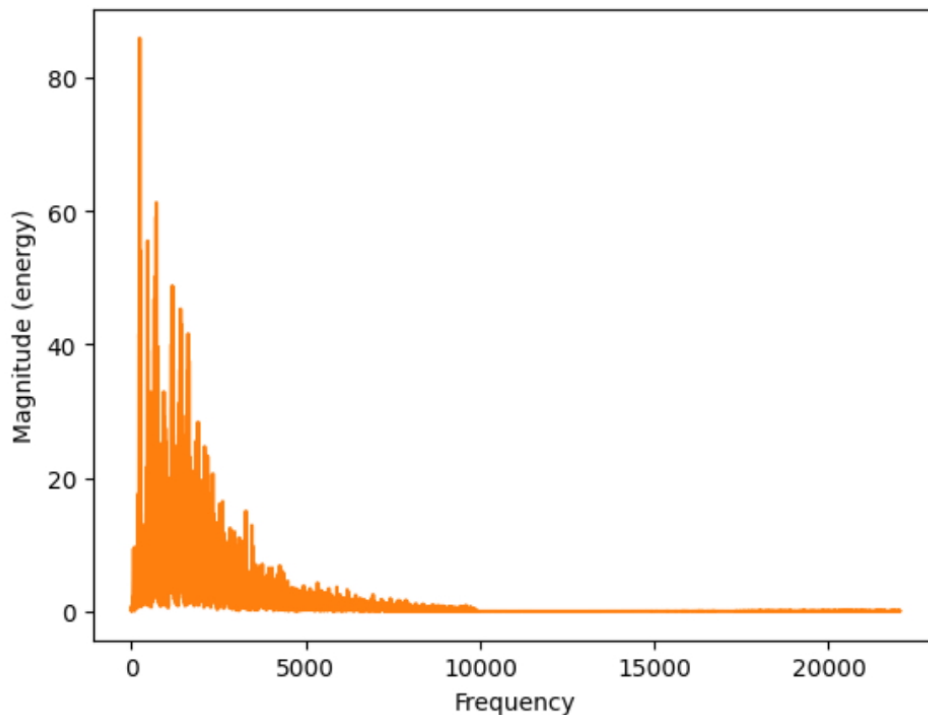


FIGURE 1 – Spectre de magnitude de la partie de l'enregistrement choisie

Afin de se concentrer sur les fréquences dont la magnitude est la plus élevée, on vient analyser les fréquences entre 0Hz et 1750Hz. Et n'afficher que les magnitudes uniquement si elle dépasse 35. Pour cela, on ajoute ceci dans notre code :

```
1 plt.xlim(0, 1750)
2 plt.ylim(35, 100)
```

Et l'on obtient désormais ce spectre :

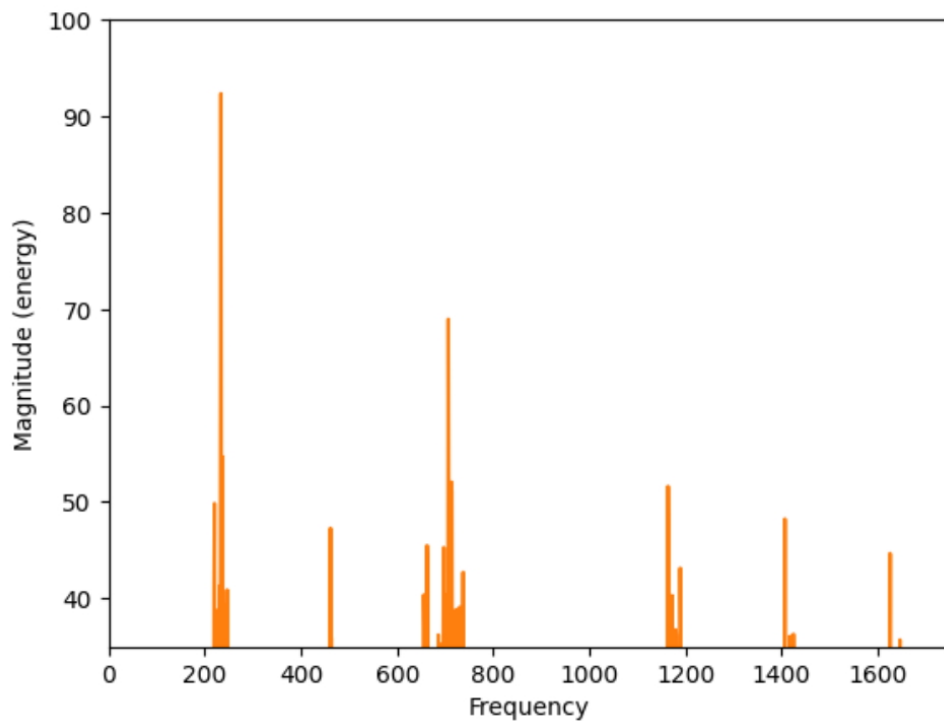


FIGURE 2 – Spectre de magnitude paramétré

Désormais que le spectre est correctement paramétré, on peut déterminer le nombre L de composantes sinusoïdales significatives sans aucun soucis. En effet, $L=6$ comme le montre les 6 pics ci-dessus.

3 Détermination des fréquences f_l

Ensuite, nous cherchons à déterminer les composantes fréquentielles f_l particulière que nous devons supprimer afin d'entendre les commentateurs.

Pour cela, nous allons venir placer par tâtonnement des points sur notre courbe et les considérer comme des relevés de valeur. Voici le code qui nous le permet :

```
1 spectrum, freqs, line = plt.magnitude_spectrum(data, Fs=Fs, color='C1')
2 plt.xlim(0, 1700)
3 plt.ylim(35, 100)
4
5 #-----
6
7 plt.scatter(233, 92)
8
9 plt.scatter(2*233, 48)
10
11 plt.scatter(3*233, 70)
12
13 #plt.scatter(4*233,49)
14
15 plt.scatter(5*233,52)
16
17 plt.scatter(6*233, 48)
18
19 plt.scatter(7*233, 45)
20
21 print("Voici la liste des frequences (elles ont ete normalises pour etre des multiples de la
22     fondamentale) :")
23 print(f"[233, {2*233}, {3*233}, {5*233}, {6*233}, {7*233}]")
24 print(f"L'absence de la frequence {4*233}Hz est remarquable.")
```

Et l'on obtient désormais ce spectre :

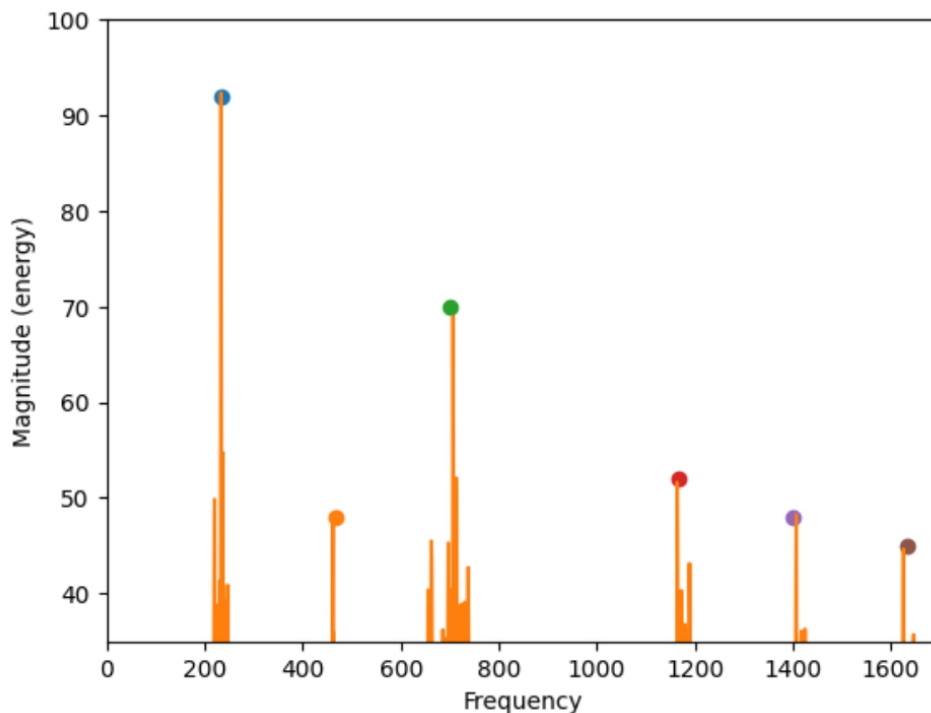


FIGURE 3 – Spectre de magnitude paramétré et avec relevé de valeur

Il nous est affiché par le code : " Voici la liste des fréquences (elles ont été normalisés pour être des multiples de la fondamentale) : [233, 466, 699, 1165, 1398, 1631]

L'absence de la fréquence 932Hz est à noter. "

Nous avons donc désormais nos composantes fréquentielles f_l particulière que voici : **233Hz, 466Hz, 699Hz, 1165Hz, 1398Hz et 1631Hz.**

On peut remarquer, comme le mets en évidence le code, qui nous avons une fréquence fondamentale à 233Hz ainsi que ses harmoniques (la fréquence étant un multiple entier de la fréquence fondamentale).

4 Détermination approximative de m

Pour déterminer la valeur de m , nous allons partir d'une valeur de bande rejetée que l'on va servir. La formule de la largeur de la bande rejetée est présentée ci-dessous :

- Largeur de la bande rejetée à -3dB ($|H(j\omega_c)| = |T_0|/\sqrt{2}$):

$$\left. \begin{aligned} \omega_{c1} &= \omega_0(-m + \sqrt{m^2 + 1}) \\ \omega_{c2} &= \omega_0(m + \sqrt{m^2 + 1}) \end{aligned} \right\} \Rightarrow \Delta\omega = \omega_{c2} - \omega_{c1} = 2m\omega_0$$

FIGURE 4 – Extrait du Chapitre 2 : Filtres d'ordre 2 - Cours de S4 Électronique

On a donc :

$$\Delta\omega = \omega_{c2} - \omega_{c1} = 2m\omega_0$$

Ce qui nous permet donc d'exprimer m ainsi :

$$m = \frac{2\Delta\omega}{\omega_0}$$

Par lecture graphique, on détermine les largeurs de bande rejetée aux alentours nos composantes fréquentielles f_i particulière suivante :

1. 233Hz : 80 Hz $\Rightarrow m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.68$
2. 466Hz : 10 Hz $\Rightarrow m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.04$
3. 699Hz : 90 Hz $\Rightarrow m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.25$
4. 1165Hz : 45 Hz $\Rightarrow m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.08$
5. 1398Hz : 33 Hz $\Rightarrow m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.04$
6. 1631Hz : 30 Hz $\Rightarrow m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.04$

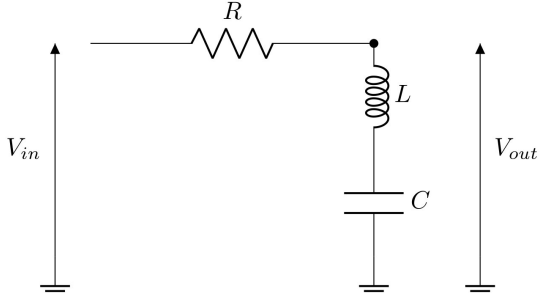
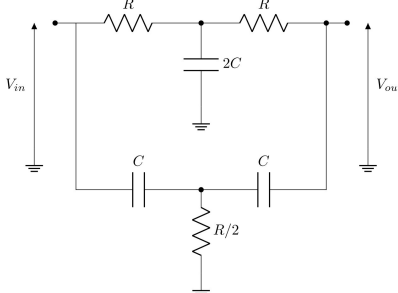
5 Mise en œuvre du filtre

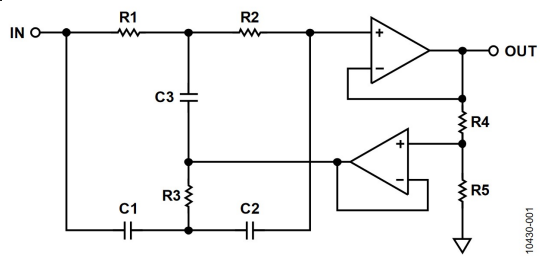
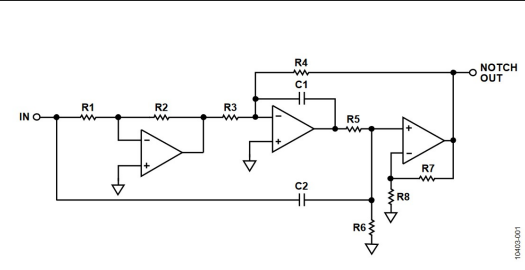
Nous avons désormais déterminé les paramètres des différents filtres. Il est à noter que nous mettrons en place un filtre pour chaque valeur de ω_0). À savoir :

$$T_0 = 1, \omega_0 = [233, 466, 699, 1165, 1398, 1631], m = \frac{2\Delta\omega}{\omega_0}$$

5.1 Choix d'une topologie de filtre rejeteur

Afin de comparer les différents filtres, nous plaçons leurs principales caractéristiques dans un tableau (Toutes les informations présentées ici proviennent des Jupyter Notebook et des simulations LTSpice liées à chaque filtre).

Filtre	Filtre RLC	Filtre Twin T Passif
Schéma		
Fonction de transfert	$H(p) = \frac{1+LCp^2}{LCp^2+RCp+1}$	$H(p) = \frac{(RC)^2 p^2 + 1}{(RC)^2 p^2 + 2RCp + 1}$
Commentaire	Le modèle du filtre est très intéressant et facilement manipulable mais malheureusement la résistance interne de la bobine vient perturber le modèle et le rendre incorrect.	Le modèle du filtre nous permet de constater que le coefficient d'amortissement m y est fixe. Cela nous empêchant de choisir la largeur de la bande rejetée ce filtre n'est pas retenu. (Ici notre fonction de transfert est probablement fautive au vu de nos tests mais il semble évident que m sera constant)

Filtre	Filtre Twin T Actif	Filtre Bainter Actif
Schéma		
Fonction de transfert	$H(p) = \frac{p^2 + (\frac{1}{RC})^2}{p^2 + p(\frac{1}{RC})(\frac{1}{1 + \frac{R5}{R4}}) + (\frac{1}{RC})^2}$	$H(p) = H * \frac{p^2 + 2w_z}{p^2 + \frac{w_0}{Q} + w_0^2}$
Commentaire	Ce filtre nous semble être le meilleur compromis, aucun réel reproche ne peut lui être fait.	Ce filtre bien que très puissant s'avère dur à régler et des incohérences sont constatées en certains lieux du filtre lors de la simulation.

À la suite de cette analyse, le filtre nous semblant le plus approprié à notre projet est le **Twin T Actif**.

5.2 Détermination des composants électroniques

Pour déterminer les composants électroniques nécessaires à la réalisation du filtre, on utilise le code ci-dessous (Les différents paramètres sont affichés en commentaire en bas du code).

```

1 def compute_component(f0, C1, R4, df):
2     m=df/(2*f0)
3     R1 = 1/(2*np.pi*C1*f0)
4     R2 = R1
5     R3=R1/2
6     C3=2*C1
7     Q = 1/(2*m)
8     R5=R4*(4*Q-1)
9     return {"C1": "{:.2e}".format(C1), "C2": "{:.2e}".format(C1), "R1": "{:.2e}".format(R1), "R2": "{:.2e}".format(R1), "R3": "{:.2e}".format(R3), "R4": "{:.2e}".format(R4), "R5": "{:.2e}".format(R5)}
10
11
12 # example

```

```

13 C1=1*(10**-9)
14 R4=1*(10**3)
15 freqs= [
16     {"f0": 233, "df": 100},
17     {"f0": 466, "df": 250},
18     {"f0": 699, "df": 175},
19     {"f0": 1165, "df": 290},
20     {"f0": 1398, "df": 250},
21     {"f0": 1631, "df": 300},
22 ]
23
24 for freq in freqs:
25     print(compute_component(freq["f0"],C1,R4,freq["df"]))
26
27 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '6.83e+05', 'R2': '6.83e+05', 'R3': '3.42e+05', 'R4':
28   '1.00e+03', 'R5': '8.32e+03'}
29 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '3.42e+05', 'R2': '3.42e+05', 'R3': '1.71e+05', 'R4':
30   '1.00e+03', 'R5': '6.46e+03'}
31 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '2.28e+05', 'R2': '2.28e+05', 'R3': '1.14e+05', 'R4':
32   '1.00e+03', 'R5': '1.50e+04'}
33 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '1.37e+05', 'R2': '1.37e+05', 'R3': '6.83e+04', 'R4':
34   '1.00e+03', 'R5': '1.51e+04'}
35 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '1.14e+05', 'R2': '1.14e+05', 'R3': '5.69e+04', 'R4':
36   '1.00e+03', 'R5': '2.14e+04'}
37 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '9.76e+04', 'R2': '9.76e+04', 'R3': '4.88e+04', 'R4':
38   '1.00e+03', 'R5': '2.07e+04'}

```

5.3 Test LTSpice de la cascade de filtres choisis

Afin de réaliser une simulation LTSPICE de notre solution technique, nous utilisons réalisons un filtre pour chaque composante fréquentielle f_i particulière et les mettons en cascade.

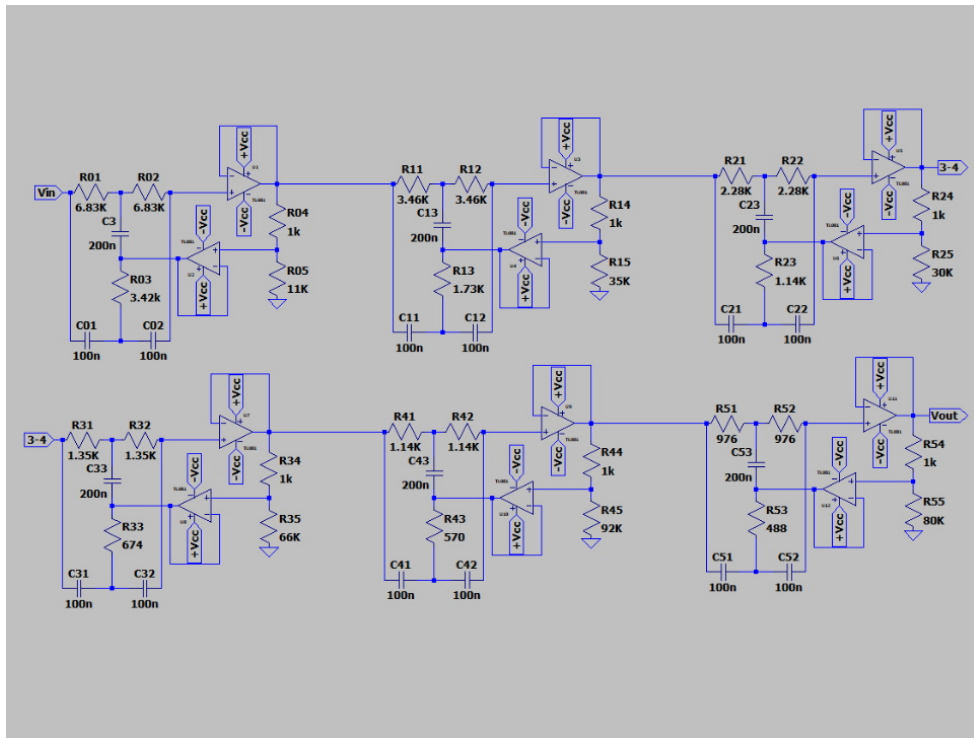


FIGURE 5 – Circuit réalisé sur LTSPICE

Afin de vérifier le fonctionnement correct du circuit, on réalise une analyse de spectre par FFT.

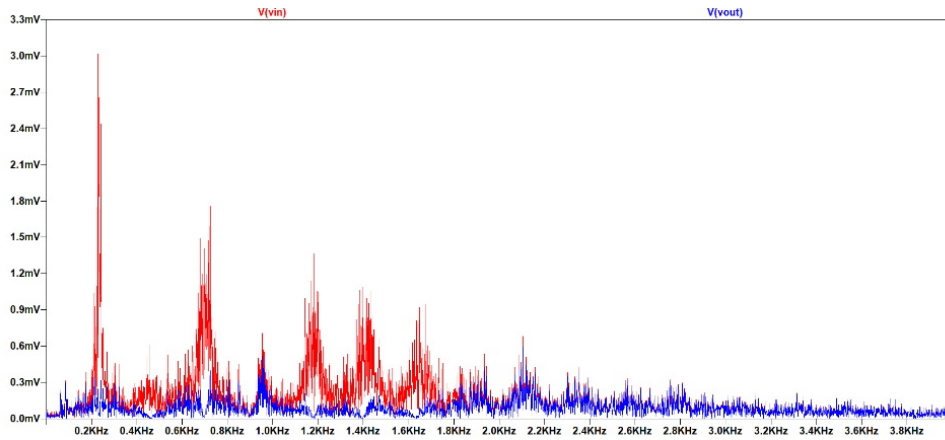


FIGURE 6 – Analyse de spectre par FFT

Lorsque l'on regarde l'analyse on constate bien que nous avons retiré les pics de magnitude propre à notre audio "vuvuzela-only" qui insolait uniquement l'instrument au sein de l'audio.(il est également possible d'écouter le fichier audio "output.wav" ci-joint afin de constater le filtrage).

5.4 Réaliser en physique de l'un des filtres

Ici il est question de tester l'un des filtres en réel afin de vérifier son fonctionnement réel.

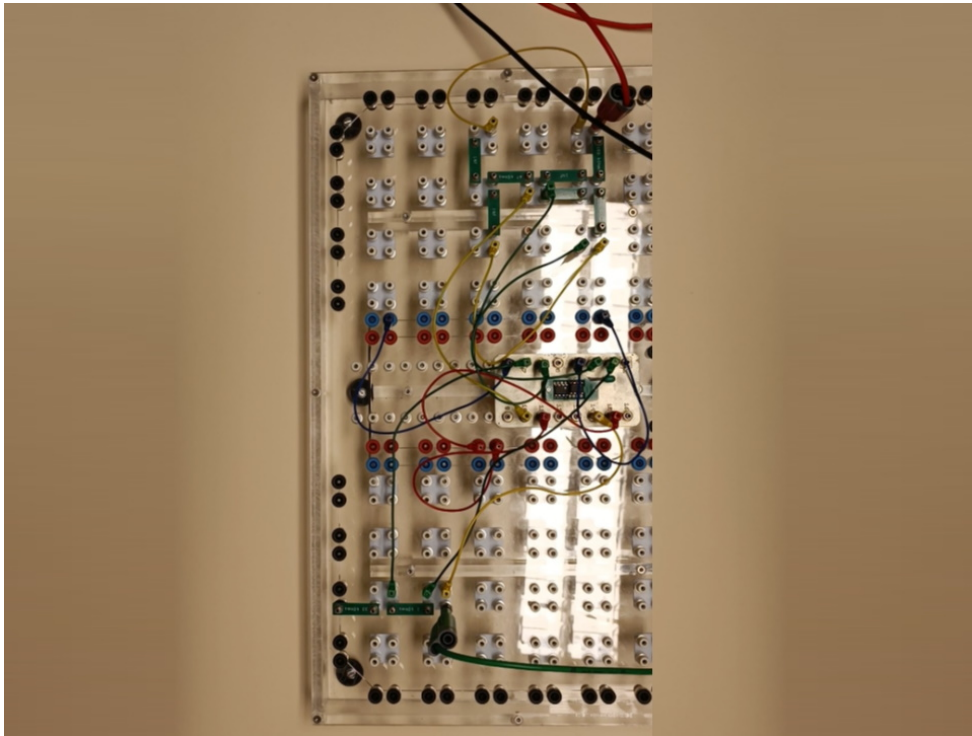


FIGURE 7 – Cablage du filtre propre à la composante fréquentielle f_i pour 1631Hz

Afin de vérifier son bon fonctionnement, on vient placer un signal sinusoïdal à la fréquence qui nous intéresse en entrée, si le filtre fonctionne correctement, on ne devrait plus le voir en sortie (ce qui est effectivement le cas ici).

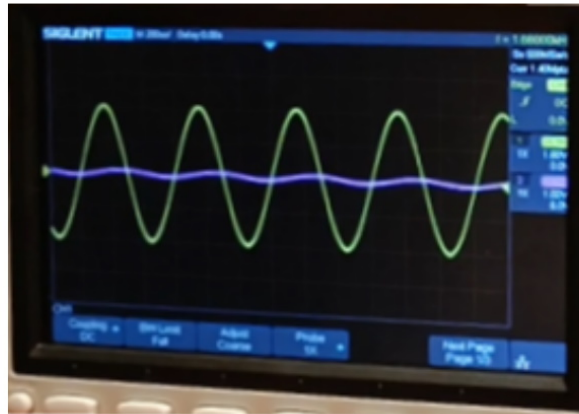


FIGURE 8 – Oscilloscope comparant l'entrée et la sortie du circuit

Conclusion

En conclusion, nous pouvons constater que notre solution technique à un résultat très significatif bien que le vuvuzela peut être encore très légèrement perçue (quelques sons aigus) et que les voix graves des commentateurs sont un peu atténués dû à des grosses bandes rejetées.

Nous sommes allés jusque 7 fois la fréquence fondamentale du vuvuzela, il est probablement que si nous allons plus loin et réalisons le $\text{freq-fond} \times 4$ le résultat sera meilleur mais cela complexifierait la conception du montage (notamment via une augmentation du temps de calculs de LTSpice qui est déjà assez élevé) et augmenterait les coûts en cas d'éventuelle production.