Amplificateur audio de classe D

1. Introduction

Ce document décrit la réalisation d'un amplificateur audio de classe D ([1]), piloté directement par un signal à modulation de largeur d'impulsion (MLI) délivré par une sortie numérique d'un arduino. Le dispositif permet de réaliser un synthétiseur de sons sans passer par un convertisseur numérique-analogique, non disponible sur les arduinos standard (UNO, MEGA, etc). De plus, la puissance délivrée par ce type d'amplificateur peut être très importante, avec une très grande efficacité.

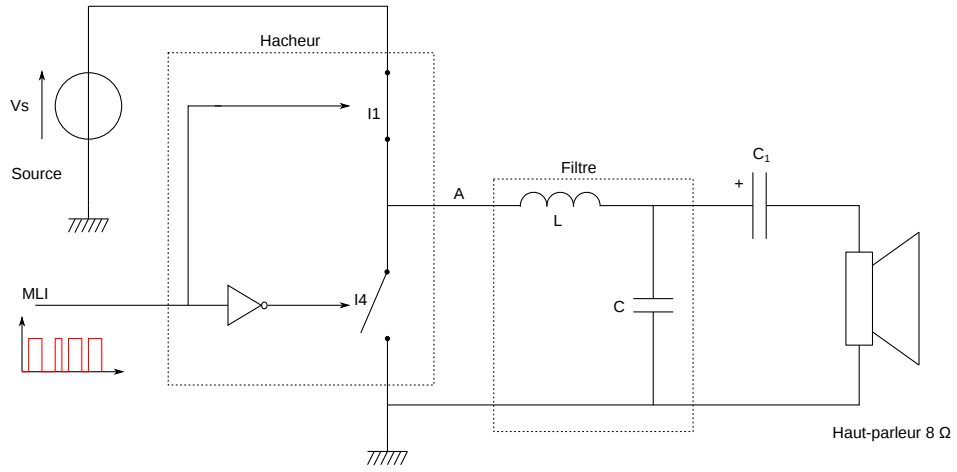
On commence par l'amplificateur en demi-pont, d'une grande efficacité mais nécessitant l'emploi d'un filtre LC.

On verra dans un second temps la structure en pont-complet, qui permet de délivrer plus de puissance pour la même alimentation. Pour cette structure, il existe une méthode de modulation de largeur d'impulsion à trois niveaux, qui permet éventuellement de se passer du filtrage LC. Cette solution est aujourd'hui la plus répandue sur les appareils portables.

2. Amplificateur avec hacheur en demi-pont

2.a. Principe

Dans son principe, un hacheur demi-pont (hacheur réversible en courant) est constitué de deux interrupteurs commandés en opposition. La commande se fait avec un signal MLI : le signal audio module le rapport cyclique d'un signal carré. La génération d'un signal de ce type avec un arduino est expliqué dans Génération d'un signal par modulation de largeur d'impulsion.



Lorsque le signal de commande est à l'état haut, l'interrupteur I_1 connecte la charge à la source d'énergie alors que I_4 est ouvert. Lorsque le signal de commande est à l'état bas, la charge est connectée à la masse alors que I_1 est ouvert. La tension au point A est en principe la réplique du signal MLI avec l'amplitude V_s, si la source se comporte parfaitement comme une source de tension.

La fréquence de la porteuse (la fréquence de découpage) doit être supérieure à la plus grande fréquence audible, c'est-à-dire 20 kHz. Dans les amplificateurs de classe D intégrés, la fréquence est au moins de 250 kHz, mais nous nous limiterons à des fréquences comprises entre 20 kHz et 50 kHz, en raison de limites dues à la fois à l'arduino et au pont utilisé. Il est en principe possible de brancher directement le haut-parleur au point A, car la fréquence de la porteuse est assez élevée pour ne pas donner de son audible. Dans cette configuration où une alimentation simple est utilisée, il faut néanmoins interposer un condensateur C_1 de couplage AC pour enlever la composante continue. Cependant, il est préférable d'interposer un filtre LC passe-bas pour éviter que des courants de haute fréquence ne traverse le haut-parleur et augmentent les pertes. D'autre part, nous avons constaté que pour une fréquence de porteuse basse de l'ordre de 50 kHz les résultats sont à l'oreille bien meilleurs avec un filtre LC.

2.b. Filtre LC

Le filtre passe-bas LC doit en principe avoir tout le spectre audible dans sa bande passante, et atténuer fortement à la fréquence de la porteuse. En pratique, on peut aussi atténuer les hautes fréquences audibles (au dessus de 1000 Hz) pour compenser la hausse de réponse des petits haut-parleurs dans les aiguës.

Une approche simplifiée consiste à considérer le haut-parleur comme une résistance $R = 8\ \Omega$, ce qu'il n'est que grossièrement dans le domaine des fréquences moyennes (500 Hz). La fonction de transfert du filtre est alors :

$$H(f) = \frac{1}{1 - \frac{f^2}{f_0^2} + j\frac{1}{Q}\frac{f}{f_0}} \tag{1}$$

$$f_0 = \frac{1}{2\pi\sqrt{LC}} \tag{2}$$

$$Q = \frac{R}{L2\pi f_0} = R\sqrt{\frac{C}{L}} \tag{3}$$

La fonction de transfert optimale pour un filtrage passe-bas est celle de Butterworth, obtenue lorsque le facteur qualité est :

$$Q = \frac{1}{\sqrt{2}} \tag{4}$$

Dans ce cas le gain est :

$$G(f) = \frac{1}{\sqrt{1 + \frac{f^4}{f_0^4}}} \tag{5}$$

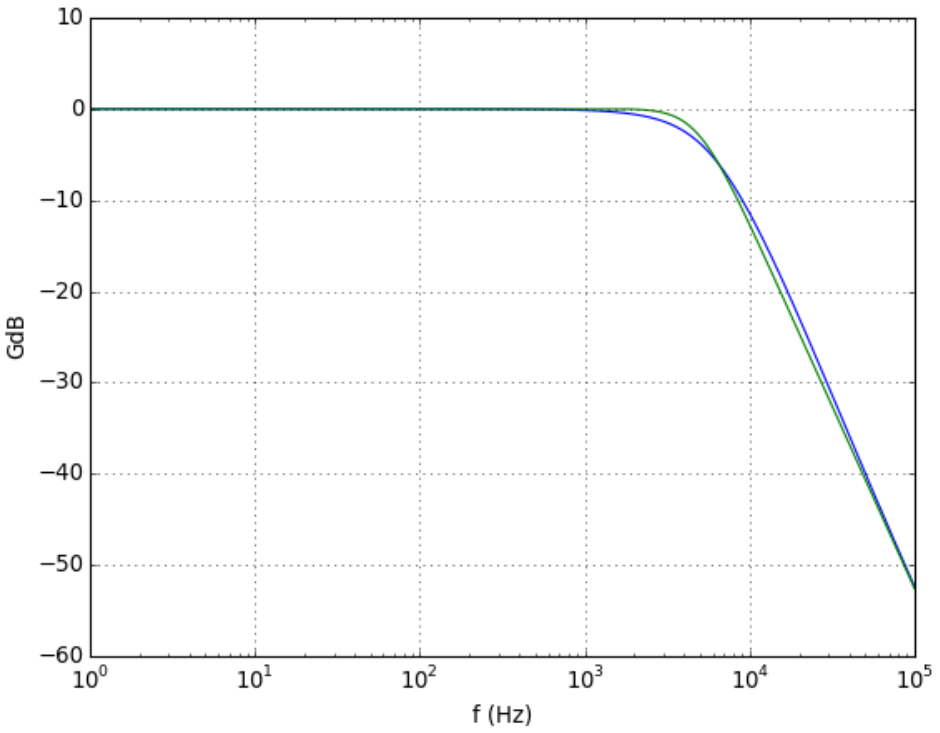
La fréquence de coupure est alors f_0 et la décroissance après la coupure est de $-40\ dB$ par décade.

Si par exemple on recherche une fréquence de coupure de 20 kHz, on calcule tout d'abord l'inductance $L = 90\ \mu H$, puis la capacité $C = 0,7\ \mu F$. Si la porteuse est à plus de 200 kHz, elle est atténuée d'au moins 40 dB, ce qui est largement suffisant car l'ondulation résiduelle est de tout manière inaudibe.

Pour notre application, où la fréquence de la porteuse est au maximum de 50 kHz, il faut abaisser la fréquence de coupure. On choisit $f_c = 5\ kHz$. Pour avoir un filtre

de Butterworth, il faut alors : $L = 360\ \mu H$ et $C = 3\ \mu F$. Voici le tracé de la réponse fréquentielle, avec un haut-parleur modélisé par une inductance et une résistance en série (modèle non valable à basse fréquence).

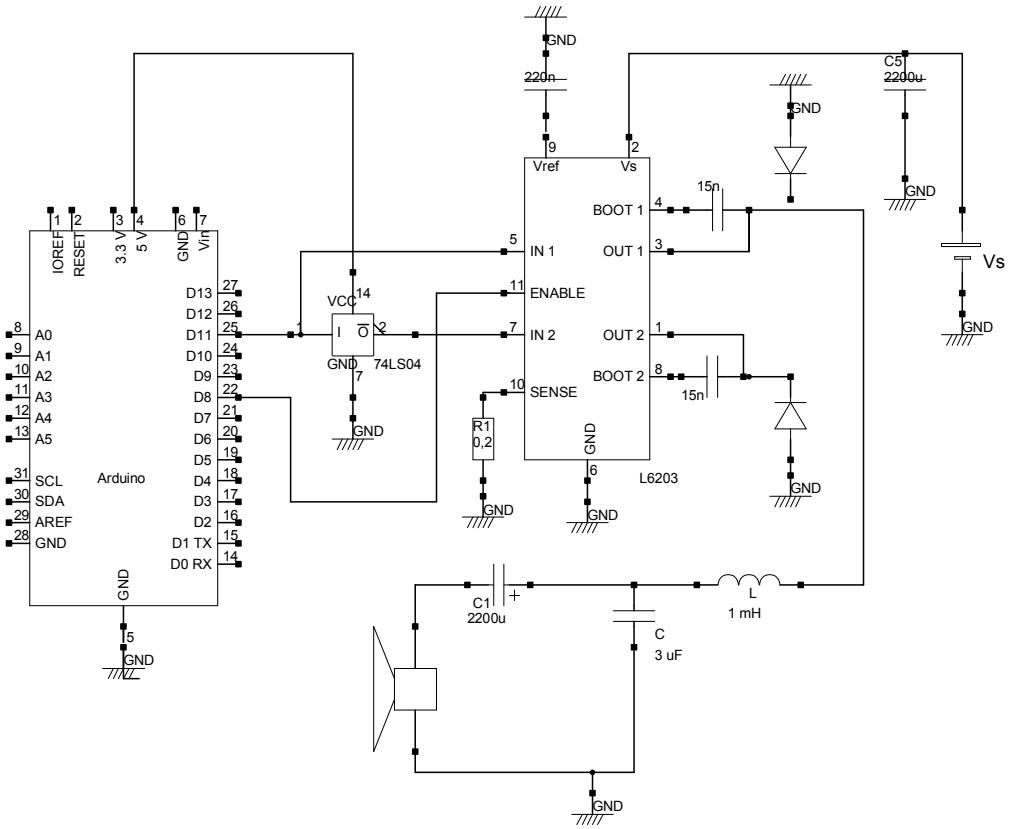
```
import numpy
from matplotlib.pyplot import *


def bodeFiltreLC(L,C,Rhp,Lhp):
    def H(f):
        w=2*numpy.pi*f
        return 1.0/(1+(1j*L*w)*(1.0/(Rhp+1j*Lhp*w)+1j*C*w))
    freq = numpy.logspace(start=0,stop=5,num=1000)
    h = H(freq)
    GdB = 20*numpy.log10(abs(h))
    plot(freq,GdB)

figure()
L=0.36e-3
C=3e-6
Rhp=7.2
Lhp=0.035e-3
bodeFiltreLC(L,C,Rhp,Lhp)
bodeFiltreLC(L,C,8,0)
xscale('log')
xlabel('f (Hz)')
ylabel('GdB')
grid()
```



On voit que l'inductance du haut-parleur a très peu d'effet. Une porteuse à 20 kHz sera atténuée de 20 dB. Une porteuse à 50 kHz sera atténuée de 40 dB.

2.c. Réalisation

On utilise un pont MOSFET L6203 (ST microelectronics). Ce pont est en principe prévu pour alimenter un moteur mais il convient aussi pour notre usage, car sa fréquence de découpage peut aller jusqu'à 100 kHz. Nous n'utilisons que la moitié du pont.



La source d'énergie est une alimentation stabilisée en tension. La tension délivrée est $V_s = 12\ V$, qui est le mininum pour que le pont MOSFET fonctionne correctement. On peut monter la tension jusqu'à $48\ V$ si l'on souhaite augmenter la puissance sonore.

Il faut remarquer que le gain d'un amplificateur de classe D est directement proportionnel à la tension d'alimentation (pas de réjection du mode commun). Avec une alimentation stabilisée, ce gain est bien constant. Si la source d'énergie est un accumulateur, il en est tout autrement, et la variation de la tension d'alimentation peut engendrer des distorsions. Dans ce cas, il est nécessaire d'ajouter une boucle de rétroaction pour contrôler le gain.

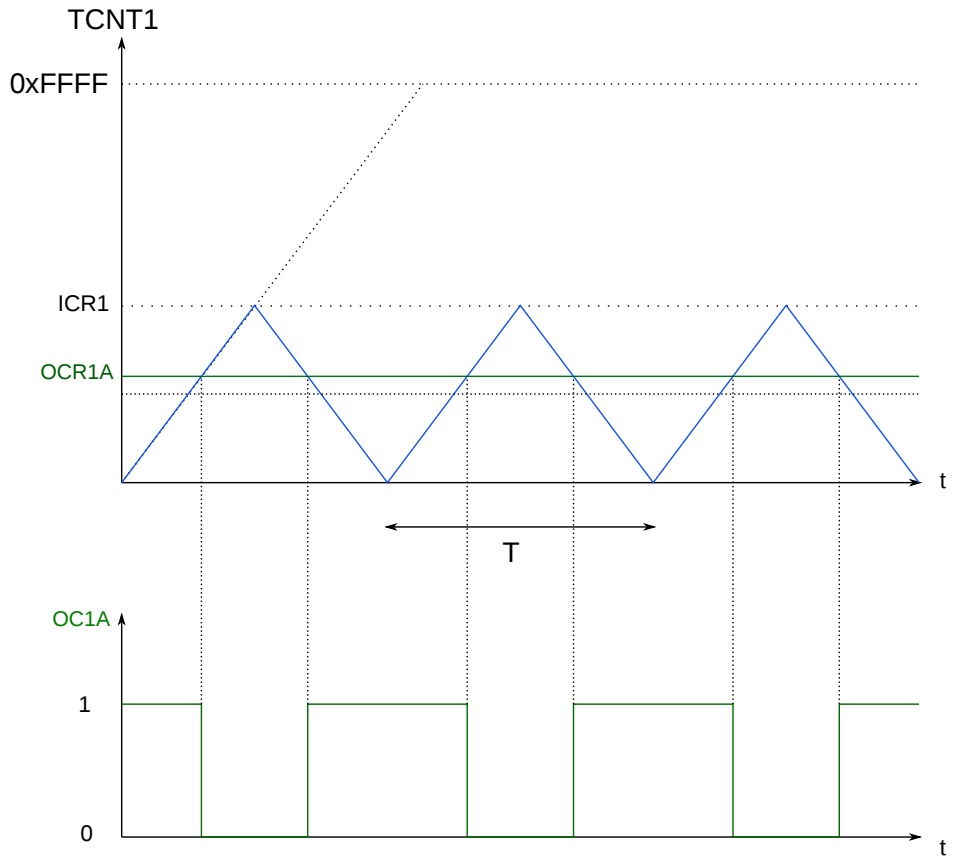
Le signal MLI est délivré par la sortie D11 d'un arduino MEGA, ou la sortie D9 d'un arduino UNO, et le pont est activé (ENABLE) par la sortie D8. La sortie D11 est reliée directement à l'entrée IN1 et à l'entrée IN2 via une porte NON.

La capacité $C = 3\ \mu F$ du filtre LC est réalisée par 3 condensateurs de $1\ \mu F$ en parallèle. Pour réaliser l'inductance L, on enroule un fil (8 spires) autour d'un tore en ferrite (type 3C90).

2.d. Programme arduino

On reprend le programme décrit en détail dans Génération d'un signal par modulation de largeur d'impulsion, en limitant à une seule sortie MLI. Le timer1 est programmé pour générer le signal MLI (ou PWM). Le rapport cyclique est modifié périodiquement à la fréquence de la porteuse. Pour cela, le timer déclenche une interruption à chaque fois que le compteur atteint sa valeur maximale.

La figure suivante montre la génération du signal sur la sortie A du timer 1 :



Dans ce programme de test, la modulation se fait avec un signal périodique stocké dans une table. Un accumulateur de phase 32 bits permet de faire varier finement la fréquence.

Dans le cadre de la réalisation d'un synthétiseur avec une interface homme-machine, bien d'autres stratégies de modulation sont bien sûr possibles, selon le type de son que l'on veut produire (c'est un très vaste sujet). D'une manière générale, si la fréquence de la porteuse est 20 kHz, il faut modifier le rapport cyclique à cette fréquence.

synthetiseurSon.ino

```
#include "Arduino.h"
#define ENABLE 8
#define NECHANT 512
#define SHIFT_ACCUM 23
uint32_t icr;
uint32_t period_pwm;
uint32_t table_onde[NECHANT];
uint32_t accum1,increm;
uint16_t diviseur[6] = {0,1,8,64,256,1024};
```

La fonction suivante programme le timer1 pour la génération du signal MLI. On active aussi une interruption (overflow interrupt).

```
void init_pwm_timer1(uint32_t period) {
    char clockBits;
    TCCR1A = 0;
    TCCR1A |= (1 << COM1A1); //Clear OC1A on compare match when upcounting, set OC1A
    TCCR1B = 1 << WGM13; // phase and frequency correct pwm mode, top = ICR1
    int d = 1;
    icr = (F_CPU/1000000*period/2);
    while ((icr>0xFFFF)&&(d<6)) { // choix du diviseur d'horloge
        d++;
        icr = (F_CPU/1000000*period/2/diviseur[d]);
   }
   clockBits = d;
   ICR1 = icr; // valeur maximale du compteur
   OCR1A = 0; // rapport cyclique
   TIMSK1 = 1 << TOIE1; // overflow interrupt enable
   sei(); // activation des interruptions
   TCNT1 = 0; // mise à zéro du compteur
   TCCR1B |= clockBits; // déclenchement du compteur
}
```

La résolution de variation du rapport cyclique est déterminée par la valeur de ICR1. Par exemple, si l'on choisit une période $T = 20\mu s$ (porteuse à 50 kHz), on a $ICR1 = 160$. Il y a donc 160 valeurs de rapport cyclique possibles, ce qui n'est pas très fin et peut engendrer des distorsions. Avec une période de 40 ms, il y a 360 valeurs possibles.

Voici la fonction appelée lors de l'interruption, qui se charge d'attribuer au rapport cyclique une valeur lue dans la table. Celle-ci est indexée avec les 7 premiers bits (de poids fort) de l'accumulateur de phase. L'incrément de l'accumulateur dépend de la fréquence du signal.

```
ISR(TIMER1_OVF_vect) { // Timer 1 Overflow interrupt
  accum1 += increm;
  OCR1A = table_onde[accum1 >> SHIFT_ACCUM];
}
```

La fonction suivante permet de remplir la table par une fonction périodique avec au plus trois harmoniques, de rang 1, 3 et 5 :

```
void set_table_harmoniques(float amp, float a1, float a3, float a5) {
    int i;
    float dt = 2*3.1415926/NECHANT;
    for(i=0; i<NECHANT; i++) {
            table_onde[i] = icr*0.5*(1.0+amp*(a1*sin(i*dt)+a3*sin(3*i*dt)+a5*sin(5*i*
        }
}
```

La fonction suivante permet de modifier la fréquence du signal, en modifiant la valeur de l'incrément de l'accumulateur de phase :

```
void set_frequence(uint32_t frequence) {
    increm = (uint32_t) (((float)(0xFFFFFFFF))*((float)(frequence)*1e-6*(float)(perio

}
```

Dans la fonction `setup`, on fixe la période de la porteuse (ici 40 ms), on programme le timer et on remplit la table (dans cet ordre) :

```
void setup() {
  pinMode(ENABLE,OUTPUT);
  digitalWrite(ENABLE,LOW);


#if defined(__AVR_ATmega2560__)
    pinMode(11,OUTPUT); // sortie PWM sur arduino MEGA
    digitalWrite(11,LOW);
#else
    pinMode(9,OUTPUT); // sortie PWM sur arduino UNO ou YUN
    digitalWrite(9,LOW);
#endif
    period_pwm = 20; // en microsecondes
    accum1 = 0;
    set_frequence(400);
    init_pwm_timer1(period_pwm);
    float amplitude = 0.1;
    set_table_harmoniques(amplitude,1.0,0.0,0.0);
    digitalWrite(ENABLE,HIGH);
}
```

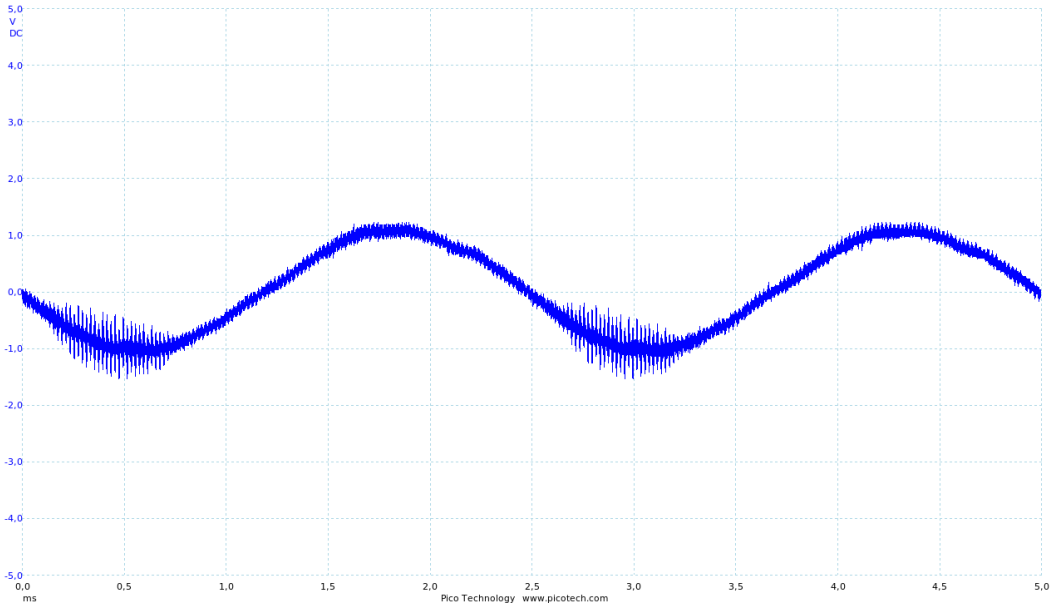
Voici un exemple de fontion `loop` dans laquelle on change la fréquence toutes les secondes.

```
void loop() {
    int d = 1000;
    delay(d);
    set_frequence(100);
    delay(d);
    set_frequence(200);
    delay(d);
    set_frequence(400);
    delay(d);
    set_frequence(800);
}
```

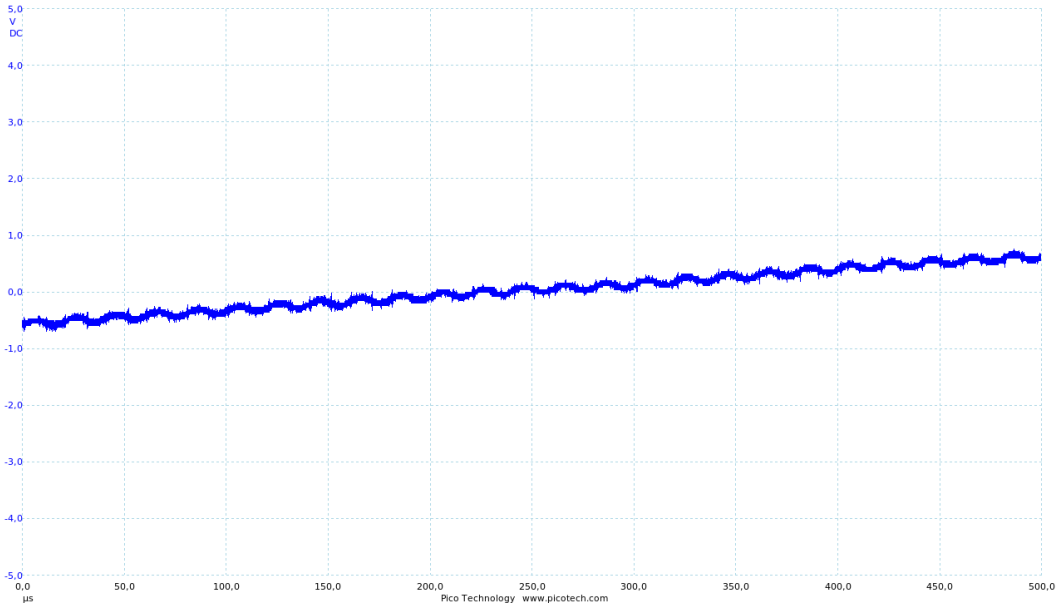
2.e. Tests de l'amplificateur

Les test sont faits avec un oscilloscope qui enregistre la tension aux bornes du haut-parleur. Le signal audio est contrôlé avec un microphone d'enregistrement et un logiciel qui trace le signal et son spectre en temps réel.

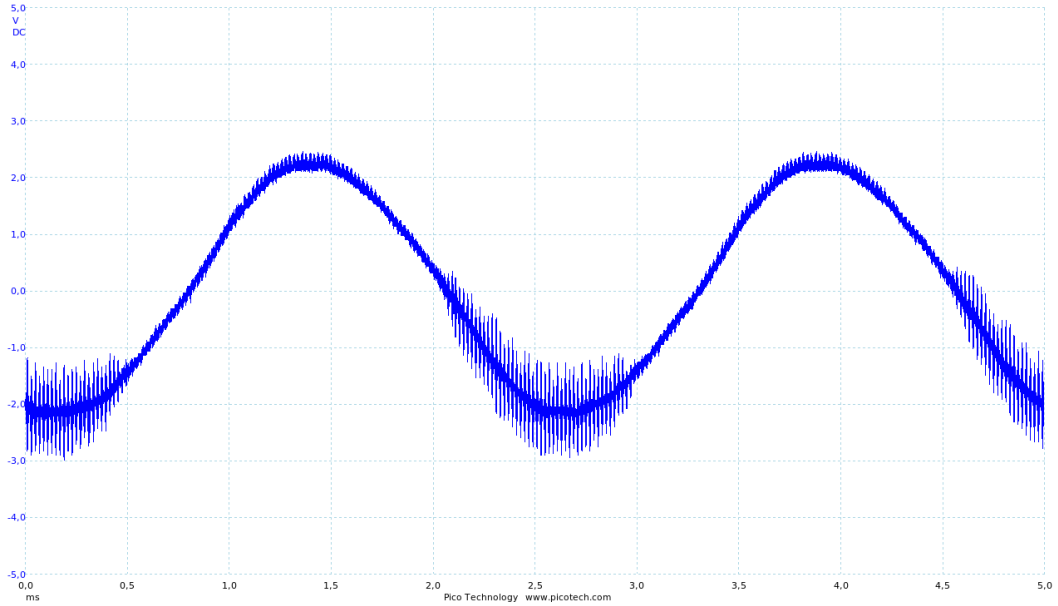
Voici le résultat pour une fréquence de porteuse (fréquence de découpage) de 50 kHz. Le signal audio est une sinusoïde de 400 Hz. L'amplitude est de $0,2$, ce qui donne déjà un bon niveau sonore pour une tension d'alimentation de 12 V.



Voici un détail, qui permet de bien voir les ondulations à la sortie du filtre LC :



Voici la tension pour une amplitude de $0,4$:

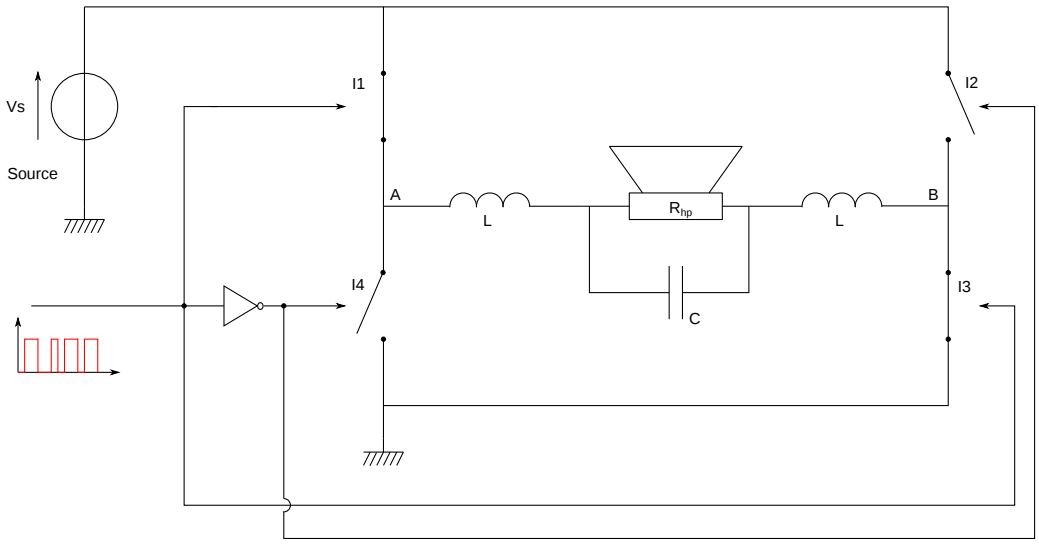


Le signal délivré par le microphone est bien sinusoïdal, avec une très faible distorsion. En augmentant le nombre de spires autour du tore pour obtenir une inductance de 1 mH (14 spires), nous avons observé une très forte distorsion de la tension aux bornes du haut-parleur. Il est donc important de faire fonctionner la ferrite dans son domaine linéaire, en limitant le nombre de spires. Le courant dans la bobine est bien plus élevé que celui dans le haut-parleur, car le courant oscillant à la fréquence de la porteuse traverse l'inductance L et le condensateur.

3. Amplificateur avec hacheur en pont complet

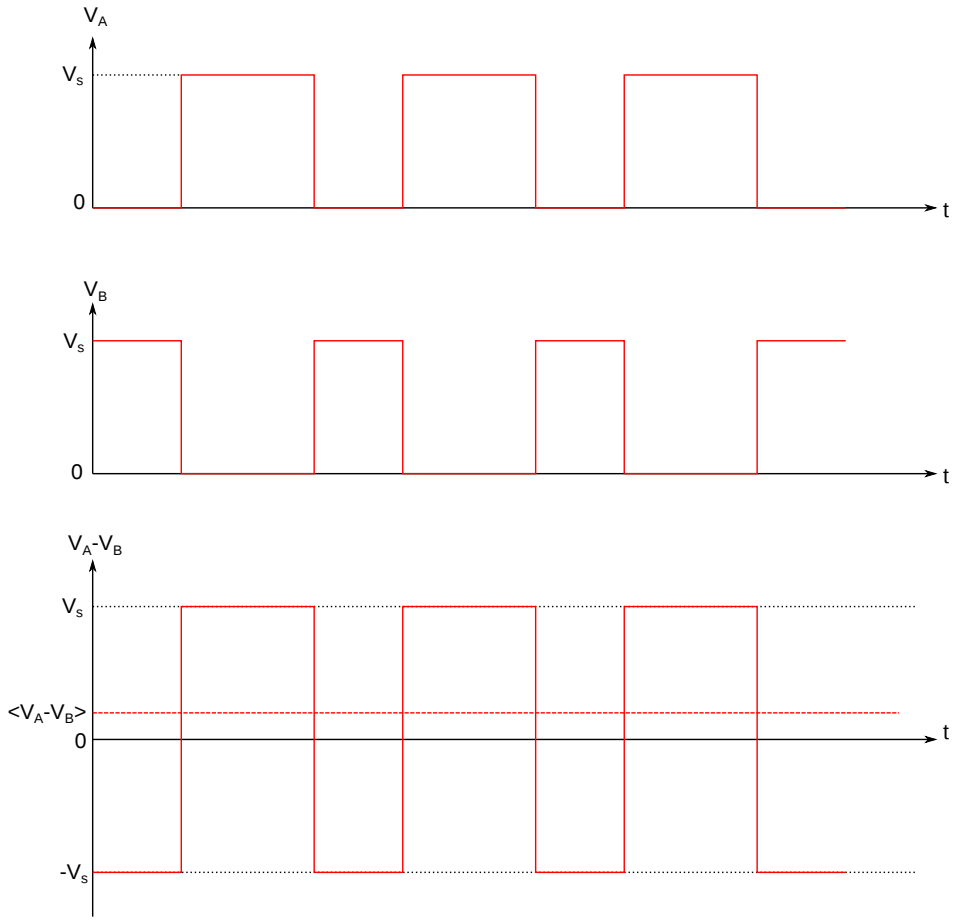
3.a. Principe

Voici le schéma de base de l'amplificateur de classe D en pont complet, pour une modulation MLI standard.



Les interrupteurs I1 et I3 sont commandés par le signal MLI alors que I2 et I4 sont commandés par son complémentaire. Le pont complet permet d'appliquer une tension alternative entre les bornes A et B, tout en permettant le passage du courant dans les deux sens. Contrairement au montage en demi-pont à alimentation simple, il n'y a plus besoin de condensateur de couplage AC. En contrepartie, il faut que le pont soit bien équilibré, car un haut parleur ne supporte pas de composante continue dans son courant (un fort courant DC peut le détruire).

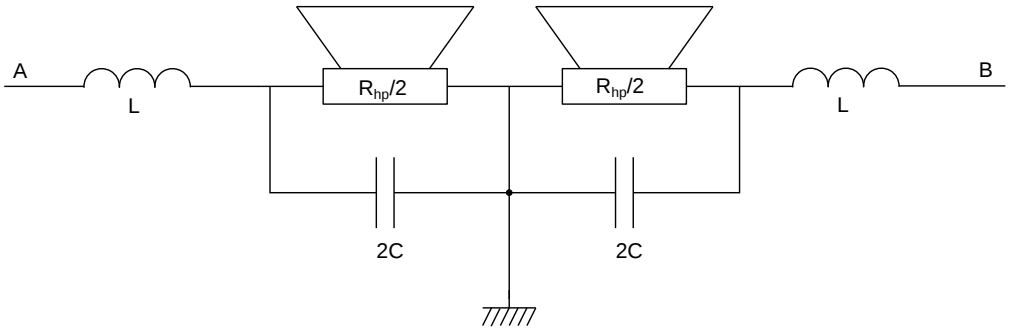
Avec une modulation MLI standard (celle utilisée ci-dessus), voici l'état des tensions pour une tension moyenne de sortie positive :



Les tensions V_A et V_B sont complémentaires. La tension différentielle en sortie a soit la valeur $+V_s$, soit la valeur $-V_s$. Cette modulation nécessite un filtre LC pour éviter la circulation d'un fort courant à la fréquence de la porteuse dans le haut-parleur. Même lorsque la tension différentielle moyenne est nulle, il y aurait un courant alternatif de forte amplitude sans ce filtre. L'avantage de cette modulation est le fonctionnement avec un seul signal de commande. L'inconvénient est la nécessité du filtre LC.

3.b. Filtre LC

Pour calculer les valeurs de L et C en fonction de la fréquence de coupure souhaitée, il faut représenter un schéma équivalent ([2]) :

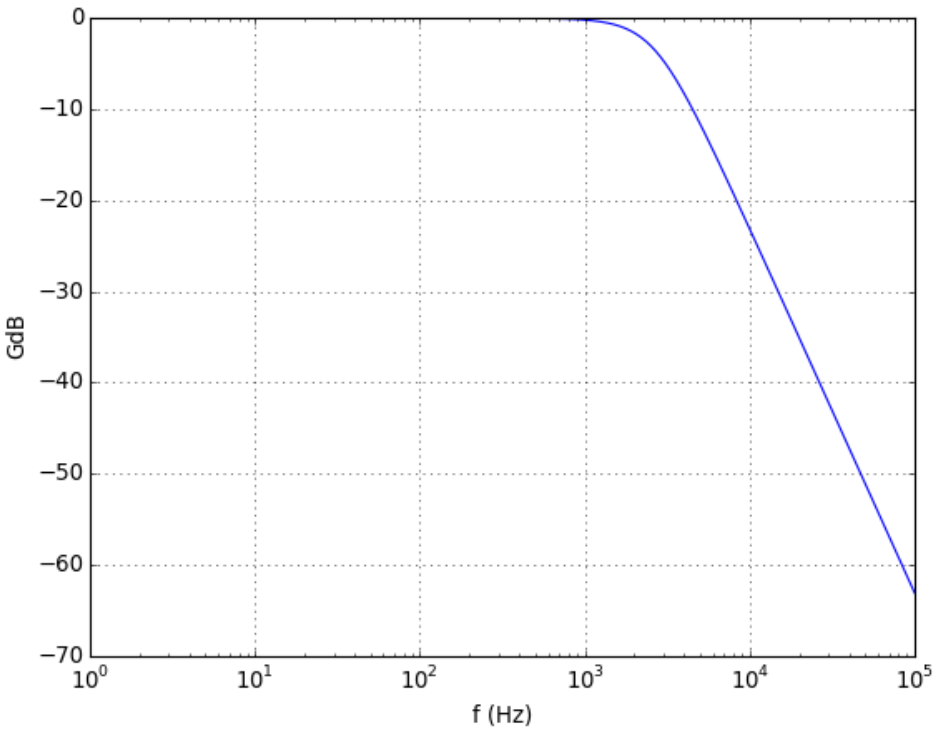


Le haut-parleur est assimilé à une résistance. Pour un haut-parleur de 8 Ω, il faut concevoir le filtre LC comme précédemment, mais avec un haut-parleur de 4 Ω. La capacité obtenue devra être divisée par deux pour le circuit final.

Par exemple, pour obtenir un filtre de Butterworth avec une fréquence de coupure $f_0 = 2500\ Hz$, il faut $L = 0,36\ mH$ et $C = 11\ \mu F$. La capacité à placer dans le montage en pont complet est donc $5,5\ \mu F$.

Voici la réponse fréquentielle pour $C = 10\ \mu F$:

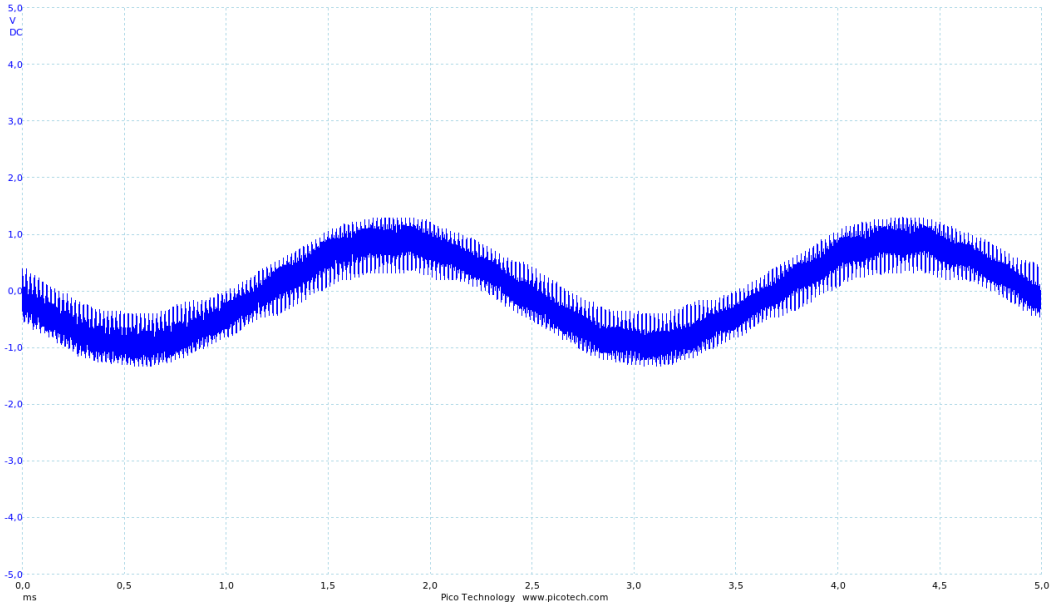
```
figure()
L=0.36e-3
C=10e-6
Rhp=4
Lhp=0
bodeFiltreLC(L,C,Rhp,Lhp)
xscale('log')
xlabel('f (Hz)')
ylabel('GdB')
grid()
```

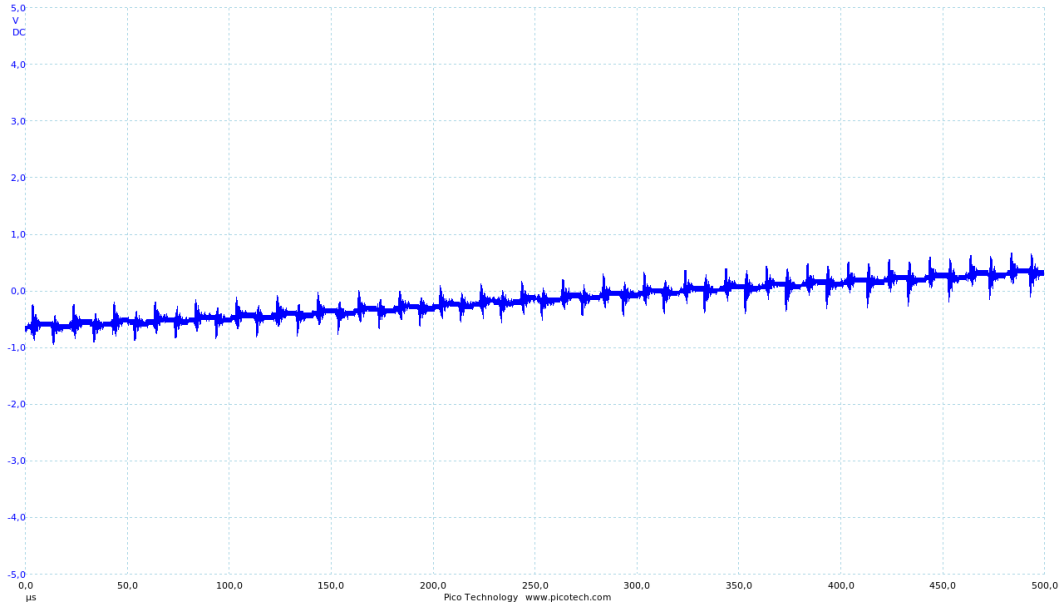


3.c. Tests de l'amplificateur

Les deux bobines L sont réalisées comme précédemment avec un tore en ferrite. Elles sont testées avec le montage en demi-pont pour vérifier qu'elles sont bien identiques. Pour visualiser la tension aux bornes du haut-parleur, on ajoute un amplificateur différentiel réalisé avec un AO et 4 résistances identiques (voir schéma plus loin).

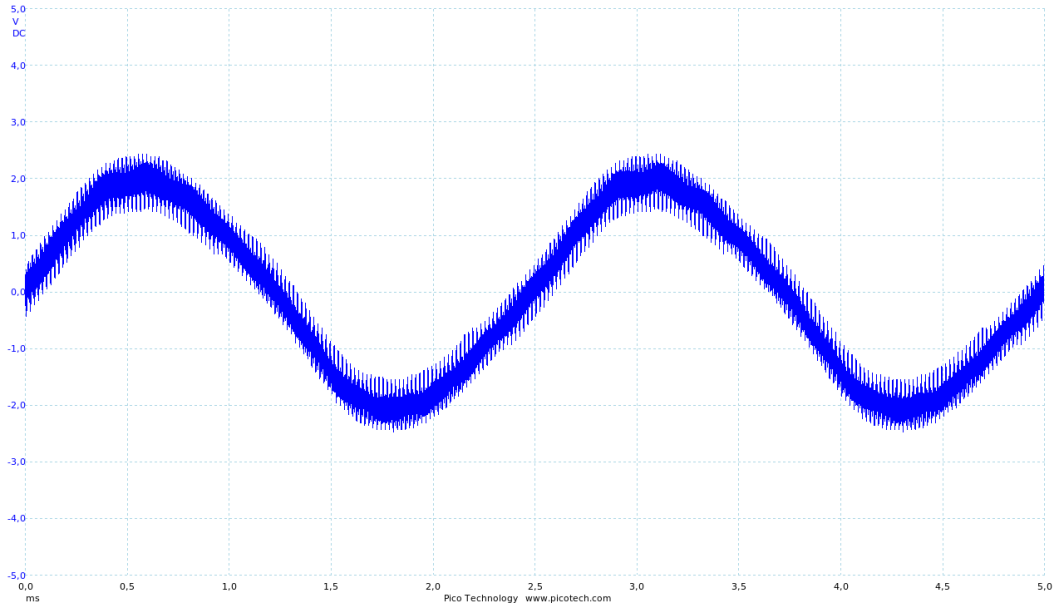
Pour une amplitude de $0,1$, le son est nettement plus intense qu'avec le montage en demi-pont et le signal audio est bien sinusoïdal. Voici la tension aux bornes du haut-parleur :





Comme prévu, la même amplitude donne une tension deux fois plus élevée. L'ondulation est très faible mais il y a des pics de tension bien visibles au moment de la commutation des transistors. Ceux-ci n'ont aucun effet sur la qualité du son émis.

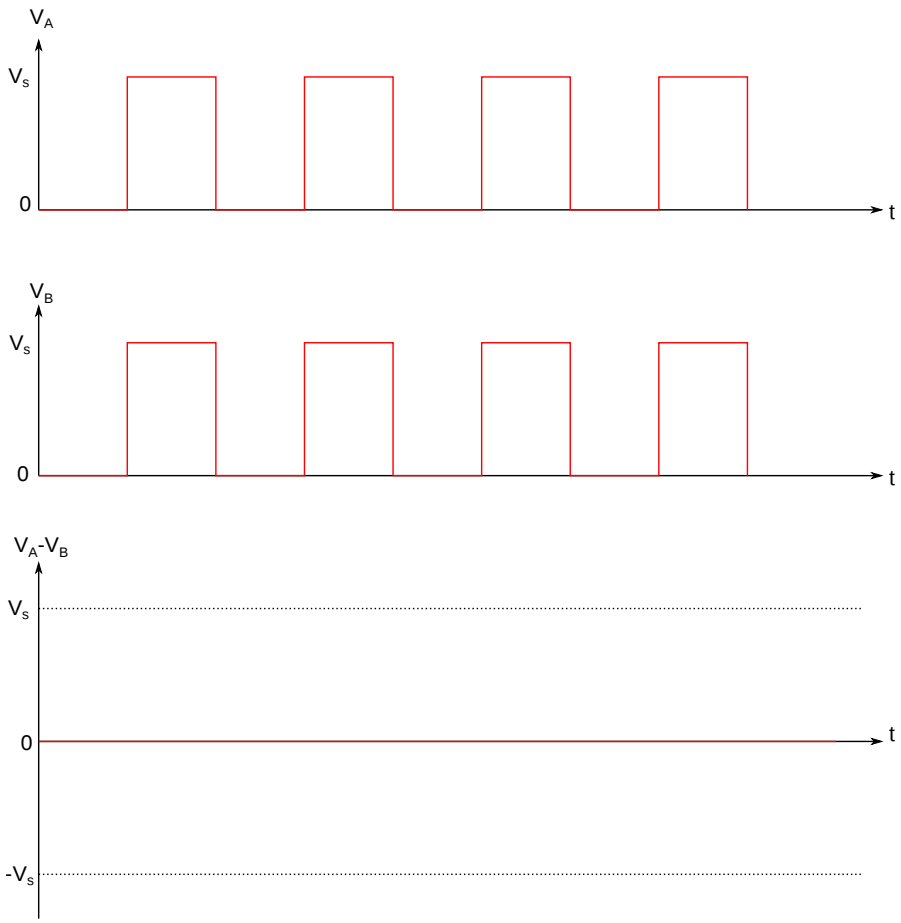
Voici la tension pour une amplitude de $0,2$:



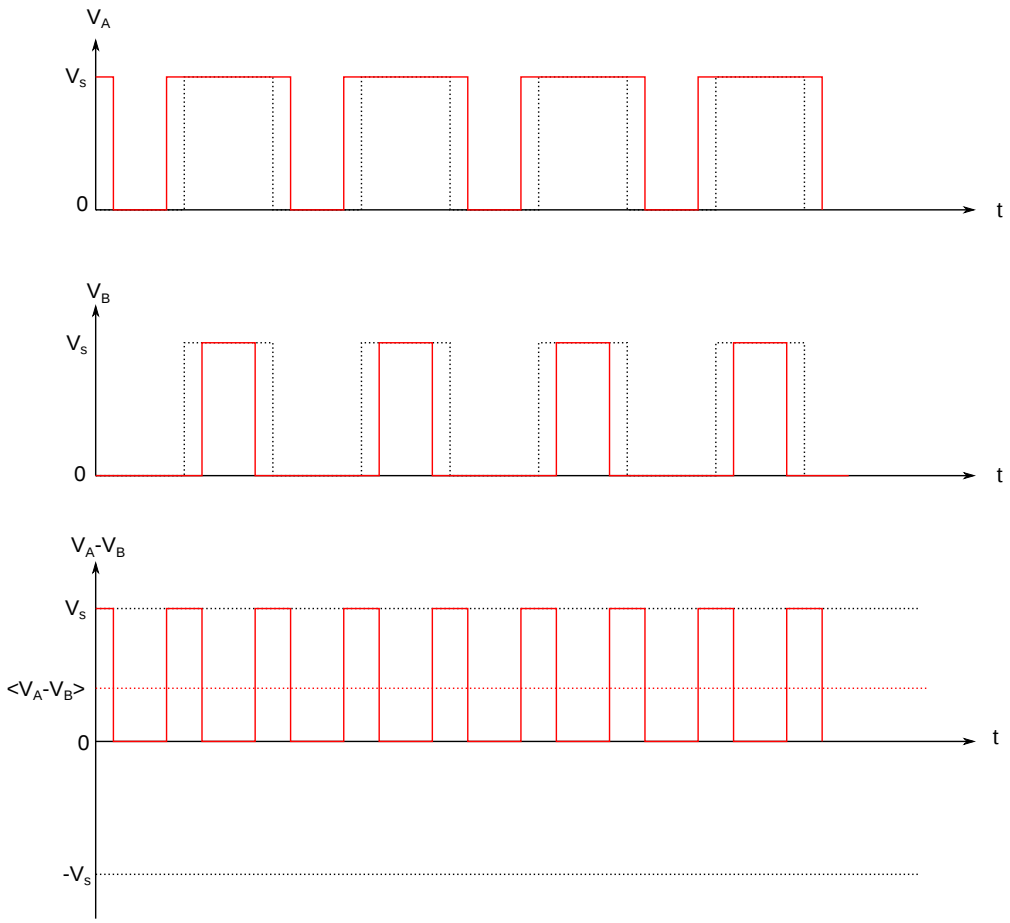
4. Amplificateur en pont complet avec modulation à trois niveaux

4.a. Principe

Avec le montage en pont complet, il existe une méthode de modulation qui permet de réduire fortement le courant d'ondulation dans le haut-parleur. Pour obtenir une tension différentielle nulle, on génère deux tensions en phase avec un rapport cyclique $1/2$:

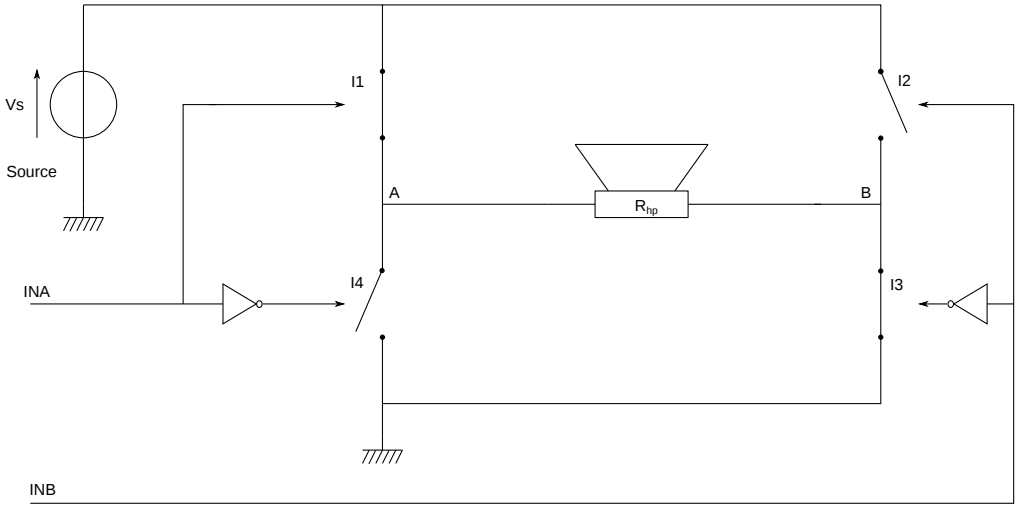


Pour obtenir une tension différentielle moyenne positive, il faut alors augmenter le rapport cyclique de V_A tout en réduisant celui de V_B :



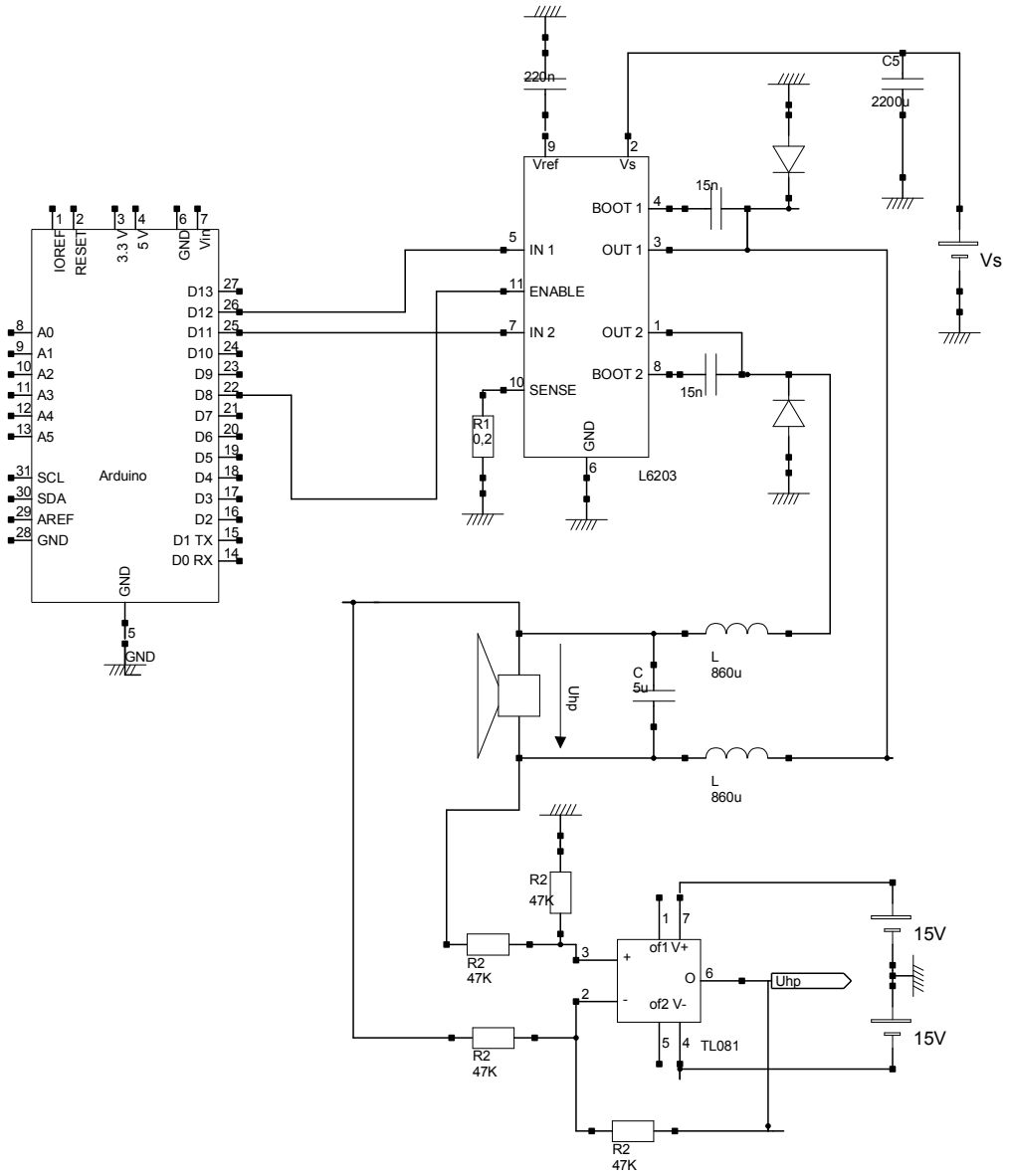
Comparé à la modulation précédente, la tension entre les deux points A et B varie deux fois moins. Il y trois état de sortie : 0, -Vs et +Vs. De plus la fréquence de découpage finale est le double de la fréquence des signaux de commande.

Dans ce schéma de modulation, le microcontrôleur doit donc générer deux signaux INA et INB. Voici le schéma du circuit (sans filtre LC) :



4.b. Réalisation

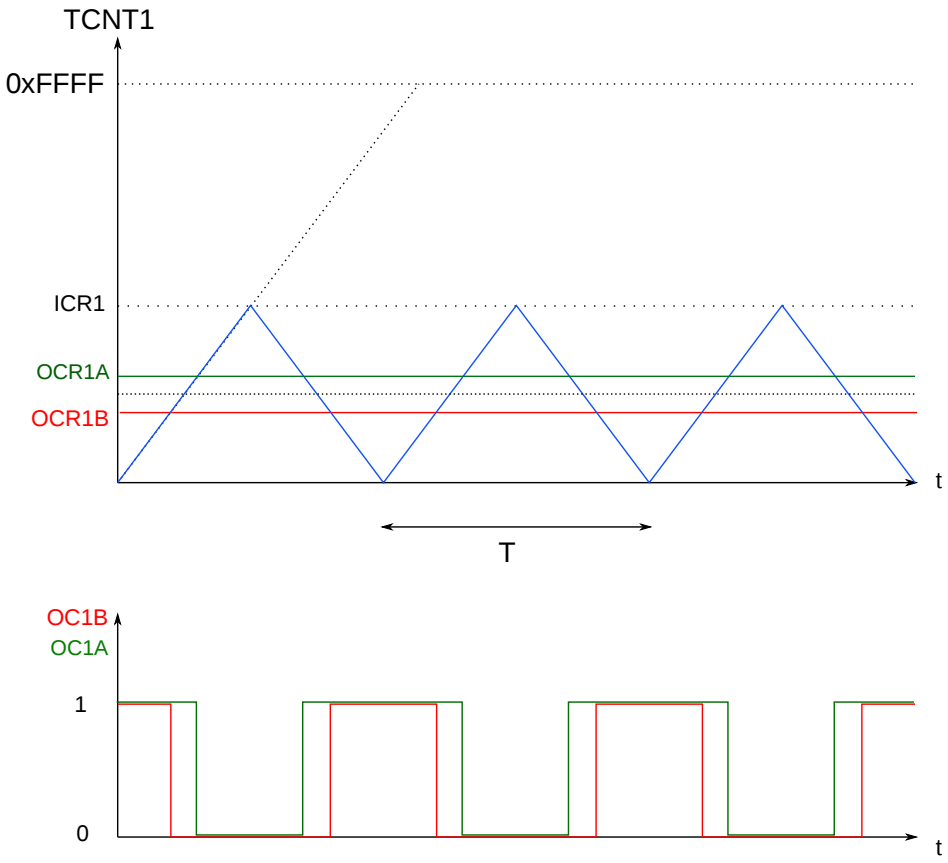
Voici le circuit réalisé autour d'un pont MOSFET L6203, avec une filtre LC. L'amplificateur différentiel permet d'obtenir la tension U_{hp} aux bornes du haut-parleur sur l'oscilloscope.



La commande des deux bras de pont (IN1 et IN2) se fait avec les sorties D11 et D12 pour un arduino MEGA. Pour un arduino UNO ou équivalent, il faut connecter les sorties D9 et D10.

4.c. Programme arduino

La génération des deux signaux de commande se fait avec les deux sorties A et B du Timer 1. La figure suivante représente l'état des seuils de déclenchement lorsque la tension différentielle moyennée sur une période est positive.



La sortie A est au niveau haut lorsque le compteur est en dessous du seuil OCR1A. La sortie B est au niveau haut lorsque le compteur est en dessous du seuil OCR1B.

Voici les bornes connectées à ces deux sorties :

- ▷ Arduino MEGA : OC1A : 11, OC1B : 12, OC1C : 13
- ▷ Arduino UNO : OC1A : 9, OC1B : 10
- ▷ Arduino YUN : OC1A : 9, OC1B : 10, OC1C : 11

synthetiseurSon-2.ino

```
#include "Arduino.h"
#define ENABLE 8
#define NECHANT 512
#define SHIFT_ACCUM 23
uint32_t icr;
uint32_t period_pwm;
uint32_t table_onde_A[NECHANT];
uint32_t table_onde_B[NECHANT];
uint32_t accum1,increm,indice;
uint16_t diviseur[6] = {0,1,8,64,256,1024};
```

```
void init_pwm_timer1(uint32_t period) {
    char clockBits;
    TCCR1A = 0;
    TCCR1A |= (1 << COM1A1); //Clear OC1A on compare match when upcounting, set OC1A
    TCCR1A |= (1 << COM1B1); //Clear OC1B on compare match when upcounting, set OC1B
    TCCR1B = 1 << WGM13; // phase and frequency correct pwm mode, top = ICR1
    int d = 1;
    icr = (F_CPU/1000000*period/2);
    while ((icr>0xFFFF)&&(d<6)) { // choix du diviseur d'horloge
        d++;
        icr = (F_CPU/1000000*period/2/diviseur[d]);
   }
   clockBits = d;
   ICR1 = icr; // valeur maximale du compteur
   OCR1A = 0; // rapport cyclique
   OCR1B = 0;
   TIMSK1 = 1 << TOIE1; // overflow interrupt enable
   sei(); // activation des interruptions
   TCNT1 = 0; // mise à zéro du compteur
   TCCR1B |= clockBits; // déclenchement du compteur
}
```

Lors de l'interruption, on doit mettre à jour les rapports cycliques des deux sorties A et B :

```
ISR(TIMER1_OVF_vect) { // Timer 1 Overflow interrupt
  accum1 += increm;
  indice = accum1 >> SHIFT_ACCUM;
  OCR1A = table_onde_A[indice];
  OCR1B = table_onde_B[indice];
}
```

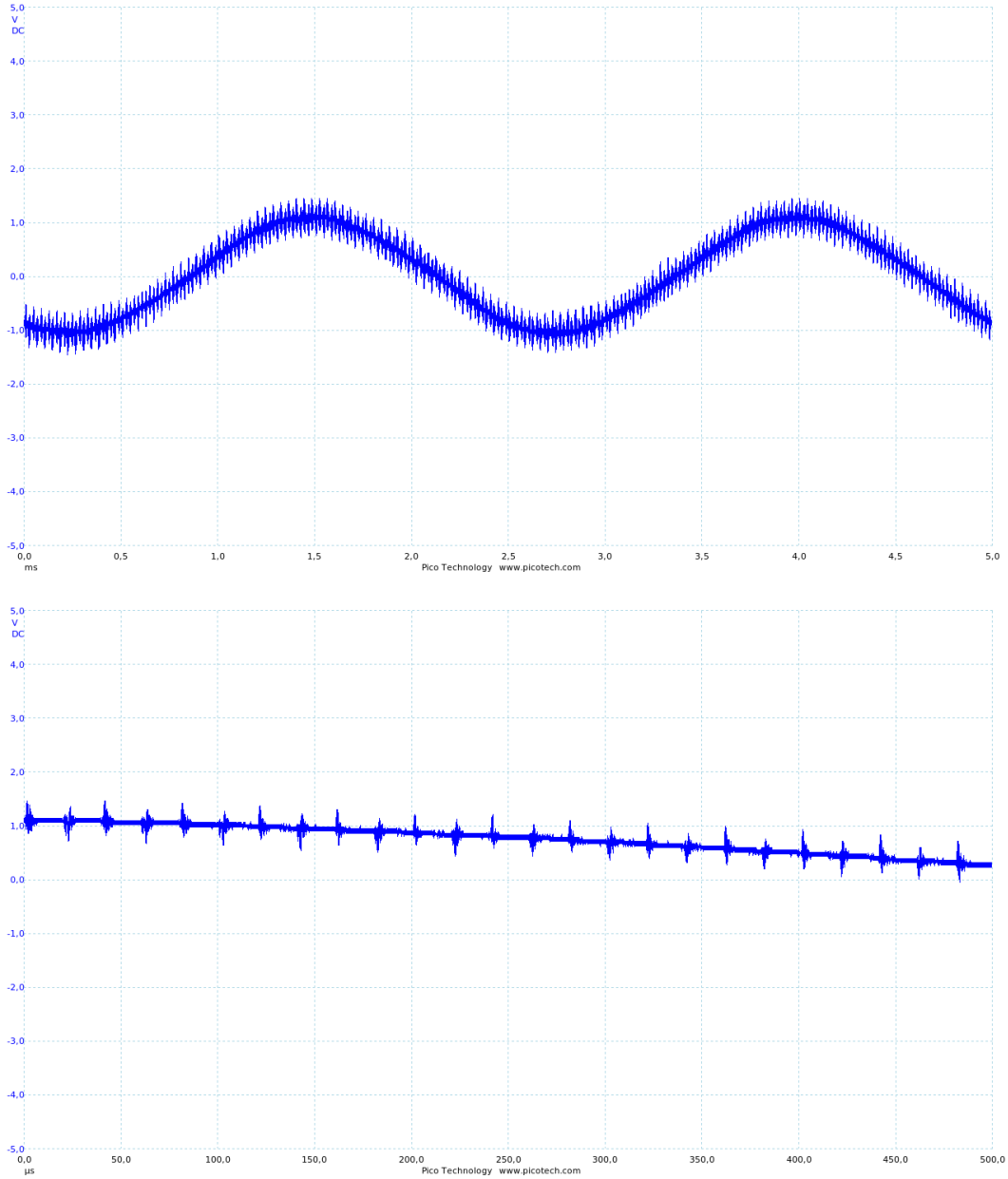
Il y a deux tables à générer :

```
void set_table_harmoniques(float amp, float a1, float a3, float a5) {
    int i;
    float x;
    float dt = 2*3.1415926/NECHANT;
    for(i=0; i<NECHANT; i++) {
            x = amp*(a1*sin(i*dt)+a3*sin(3*i*dt)+a5*sin(5*i*dt));
            table_onde_A[i] = icr*0.5*(1+x);
            table_onde_B[i] = icr*0.5*(1-x);
        }
}


void set_frequence(uint32_t frequence) {
```

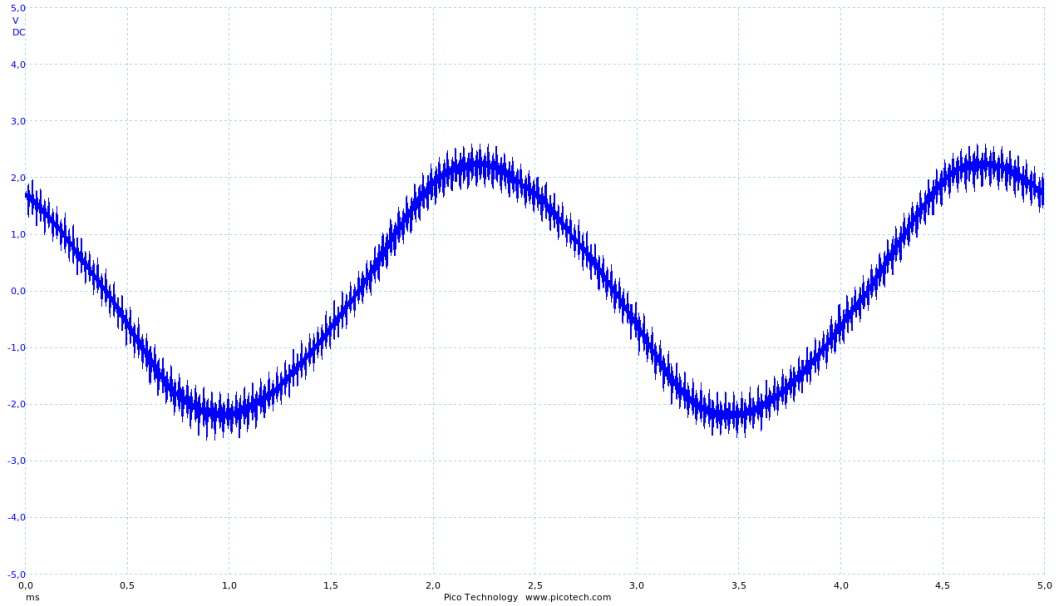
```
    increm = (uint32_t) (((float)(0xFFFFFFFF))*((float)(frequence)*1e-6*(float)(perio

}

void setup() {
  pinMode(ENABLE,OUTPUT);
  digitalWrite(ENABLE,LOW);


#if defined(__AVR_ATmega2560__)
    pinMode(11,OUTPUT); // sortie PWM sur arduino MEGA
    digitalWrite(11,LOW);
    pinMode(12,OUTPUT);
    digitalWrite(12,LOW);
#else
    pinMode(9,OUTPUT); // sortie PWM sur arduino UNO ou YUN
    digitalWrite(9,LOW);
    pinMode(10,OUTPUT);
    digitalWrite(10,LOW);
#endif
    period_pwm = 40; // en microsecondes
    accum1 = 0;
    set_frequence(400);
    init_pwm_timer1(period_pwm);
    float amplitude = 0.1;
    set_table_harmoniques(amplitude,1.0,0.0,0.0);
    digitalWrite(ENABLE,HIGH);
}

void loop() {

}
```

4.d. Test de l'amplificateur

On commence par tester l'amplificateur avec le filtre LC. On profite du doublement de la fréquence de découpage pour doubler la période du compteur, ce qui a l'avantage d'augmenter la résolution de variation du rapport cyclique. Nous avons en effet constaté qu'une résolution insuffisante peut donner un bruit de fond audible, mais invisible à l'oscilloscope. Voici la tension aux bornes du haut-parleur :

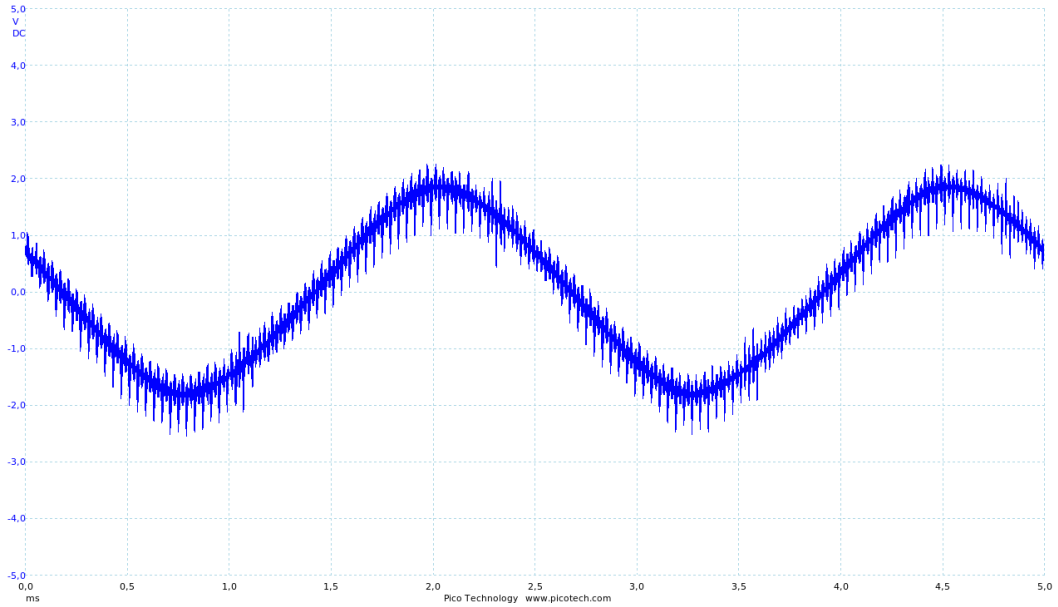


Voici la tension pour une amplitude de $0,2$ (le maximum est 1) :

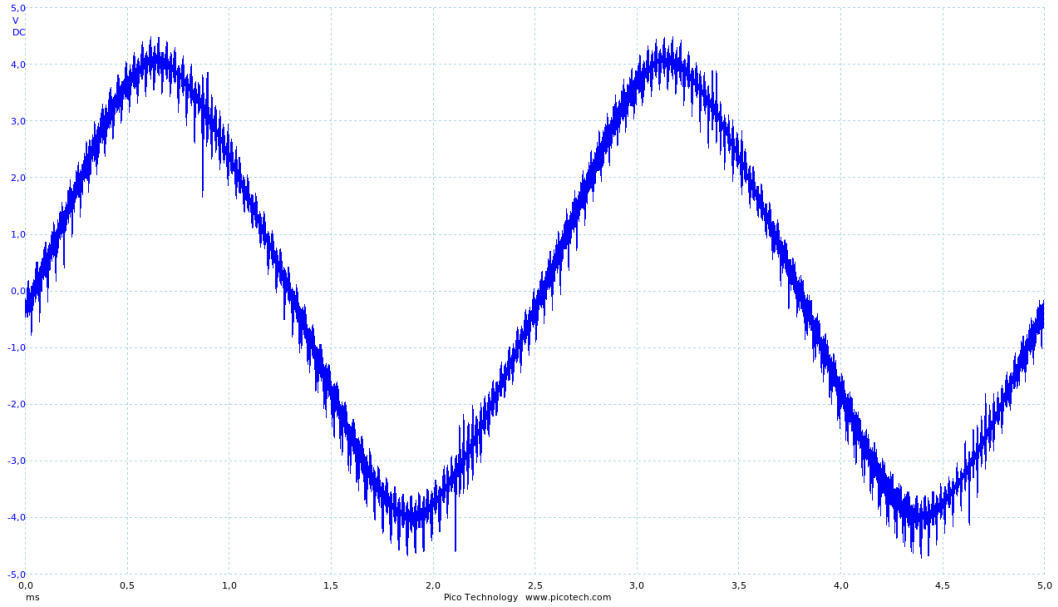


Comme dans les tests précédents, on observe à cette amplitude une légère distorsion dissymétrique, que nous attribuons à un début de saturation magnétique du tore des bobines. Nous avons finalement utilisé des bobines commerciales (Wurth elecktronik 7447075) $L = 860\ \mu H$ de courant maximal 3 A avec un tore en poudre composite de fer. La poude de fer a une perméabilité beaucoup moins élevée que la ferrite, d'où un nombre de spires plus grand pour la même inductance. Elle peut supporter un courant plus fort sans saturation, mais les pertes sont plus élevées.

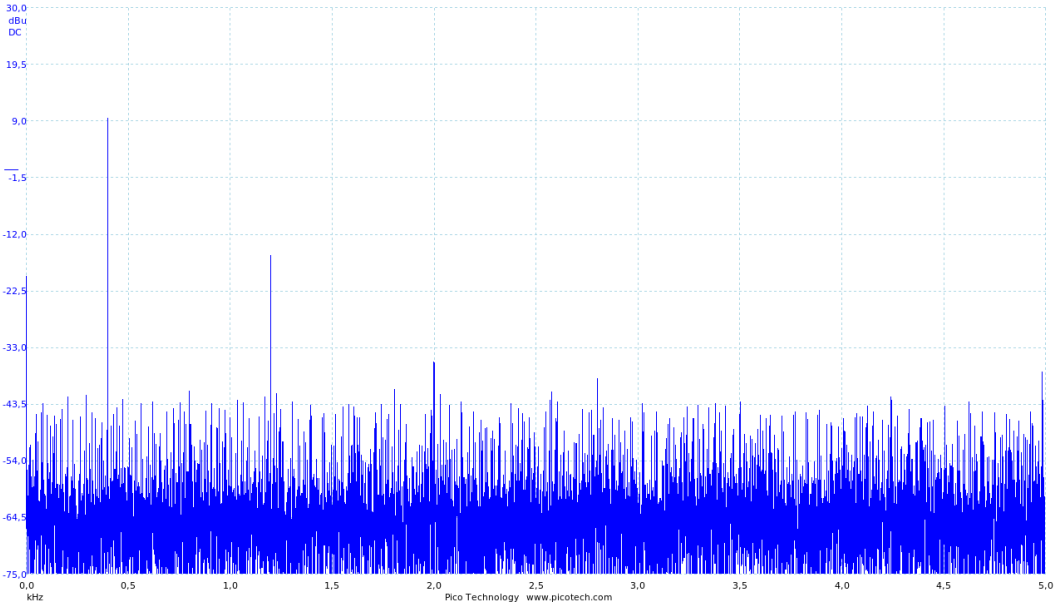
Voici le résultat avec ces bobines :



Avec les tores en poude de fer, on peut augmenter encore l'amplitude tout en restant à un niveau de distorsion faible. Voici la tension pour une amplitude de $0,5$ (la tension d'alimentation est toujours 12 V) :



Voici le spectre :



Il a une distorsion bien visible mais faible, avec une harmonique de rang 3 à $-25\ dB$ du fondamental.

À cette amplitude, le signal audio présente une distorsion très marquée, très probablement attribuable au haut-parleur lui-même.

Nous avons finalement fait un test sans le filtre LC. Pour un période de 40 ms, il y a un bruit de fond gênant, qui disparait presque complètement à 50 ms. Avec le haut-parleur que nous avons utilisé, le son est meilleur avec le filtre LC.

Références

Class D audio amplifier basics

Class D amplifier Design Basic

Designing Practical High Performance Class D Audio Amplifier

[1] A.P. Malvino, D.J. Bates, *Electronic Principles*, (McGraw-Hill, 2016)

[2] Texas Instruments, *Class-D LC Filter Design*, (Application Report, April 2006)