SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

POLYTECH SORBONNE

Projet Court:

Electronique et traitement du signal embarqué

Yining BAO Qianhui JIN
Liyun YANG Oumeima ESSAID

Année: 2019-2020

Sommaire

I. Introduction

Ce projet est basé sur les connaissances qu'on a étudié pendant cette année de EISE, surtout sur l'électronique, le microcontrôleur, le management et le traitement du signal. Pour la partie de microcontrôleur et le traitement du signal, on a deux choix: soit la transformé de fourier rapide soit le filtrage. Parmi ces deux choix on a choisi de traiter sur la transformé de fourier rapide. Pour la partie électronique, on doit améliorer le module Grove Sound Sensor et Grove Speaker. Et le management est basé sur notre organisation du groupe.

Donc on va traiter ce projet en 4 parties. Dans un premier temps, on va étudier sur les deux modules. Ensuite, on va réfléchir sur l'implémentation du transformé de fourier. Puis on représente les différents moyens pour améliorer ces deux modules. Ensuite, on va critiquer sur notre organisation, ce qui est bien fait et ce qui est mal passé. A la fin, une petite conclusion basée sur ce projet.

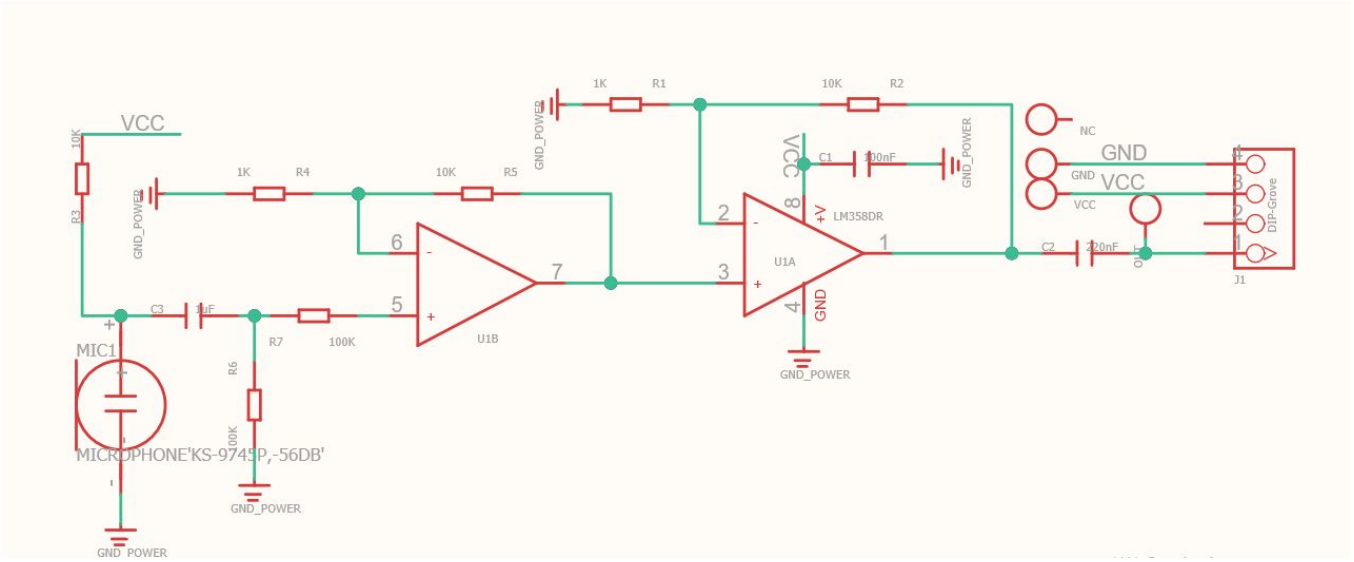
II. Prise en main du matériel

Cette première partie du projet consiste de prendre connaissances sur les deux modules ce qu'on va utiliser pendant notre projet.

Le module Grove Sound Sensor:

Le Grove Sound Sensor peut détecter l'intensité sonore de l'environnement. Il est basé sur l'amplificateur LM386 et un microphone à électret. La sortie de cette module est analogique. Son tension utilisé est de 5V, l'impédance de microphone est de 2,2 kΩ et la fréquence dominante est de 16 Hz à 20 kHz ce qui correspond bien la domaine audible de l'humain.

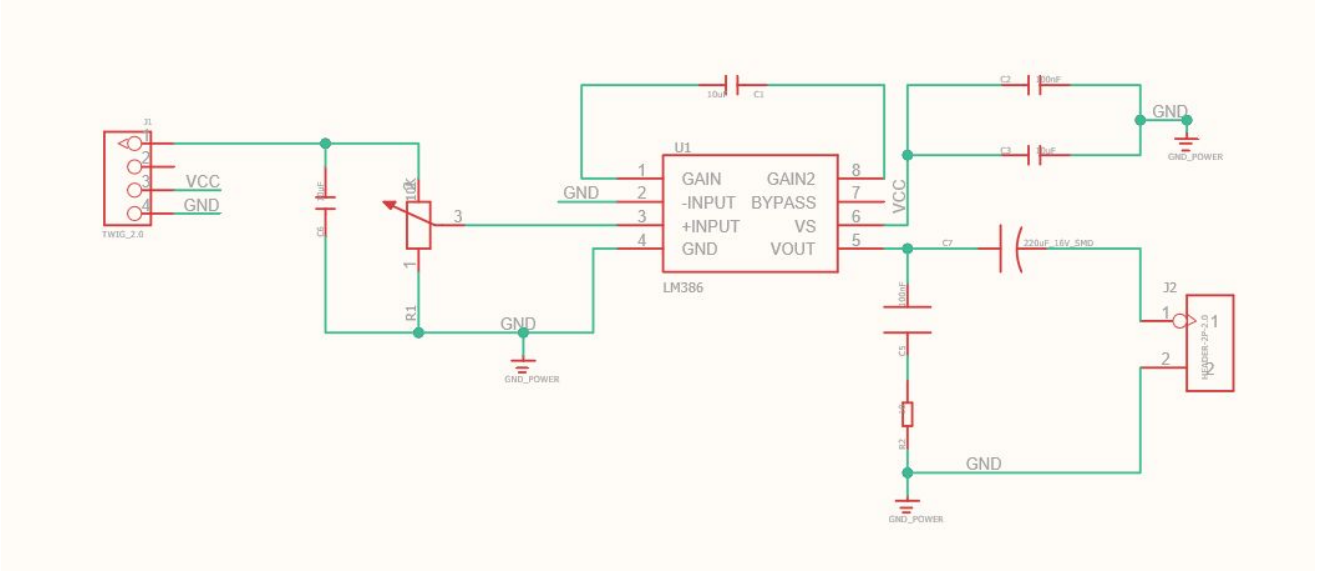
Dans son document technique, on a aussi trouvé son schéma qui donne les informations sur son mode de fonctionnement:



Le module Grove Speaker:

Le Grove Speaker peut émettre les différentes sons selon les différentes fréquence entrée, maximum 20 kHz. Il se compose par une amplification de puissance et des sorties vocales. Le volume peut ajuster par le potentiomètre intégré.

Voici son schéma:



Dans ce schéma, on peut détecter que le Grove Speaker est aussi intégré l'amplificateur LM386.

L’amplificateur LM386:

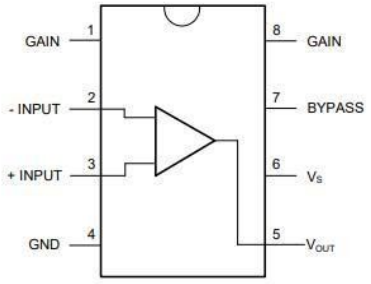
L’amplificateur LM386 a un gain de 20 jusqu’à 200, il se compose par peu de composant externe et avoir un petit distorsion de 0.2%. En plus il sont utilisé sur différentes applications. Voici son tableau caractéristique:

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_S	Operating Supply Voltage	LM386N-1, -3, LM386M-1, LM386MM-1	4		12	V
		LM386N-4	5		18	
I_Q	Quiescent Current	V_S = 6 V, V_{IN} = 0		4	8	mA
P_{OUT}	Output Power	V_S = 6 V, R_L = 8 Ω, THD = 10% (LM386N-1, LM386M-1, LM386MM-1)	250	325		mW
		V_S = 9 V, R_L = 8 Ω, THD = 10% (LM386N-3)	500	700		
		V_S = 16 V, R_L = 32 Ω, THD = 10% (LM386N-4)	700	100		
A_V	Voltage Gain	V_S = 6 V, f = 1 kHz		26		dB
		10 µF from Pin 1 to 8		46		
BW	Bandwidth	V_S = 6 V, Pins 1 and 8 Open		300		kHz
THD	Total Harmonic Distortion	V_S = 6 V, R_L = 8 Ω, POUT = 125 mW f = 1 kHz, Pins 1 and 8 Open		0.2%		
PSRR	Power Supply Rejection Ratio	V_S = 6 V, f = 1 kHz, CBYPASS = 10 µF Pins 1 and 8 Open, Referred to Output		50		dB
R_{IN}	Input Resistance			50		kΩ
I_{BIAS}	Input Bias Current	V_S = 6 V, Pins 2 and 3 Open		250		nA

Cet amplificateur se compose par 8 pins, chaque pin a son propre fonction.

5 Pin Configuration and Functions

D Package
8-Pin MSOP
Top View



Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
GAIN	1	–	Gain setting pin
–INPUT	2	I	Inverting input
+INPUT	3	I	Noninverting input
GND	4	P	Ground reference
V_{OUT}	5	O	Output
V_S	6	P	Power supply voltage
BYPASS	7	O	Bypass decoupling path
GAIN	8	–	Gain setting pin

D’après ces tableaux et les recherches sur internet pour les deux modules. A cause de l’intégration du l’amplificateur LM386, ils ont les caractéristiques similaires. Donc pour l’amélioration, on va principalement traiter sur cette amplificateur LM386.

En plus la module Grove Sound Sensor peut détecter l’intensité sonore, donc on va l’utiliser dans la partie de transformé de fourier rapide.

III. Transformé de fourier rapide

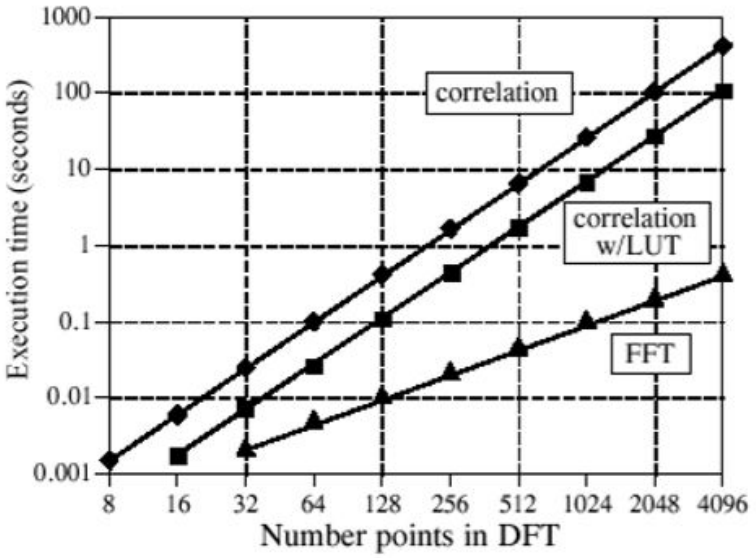
Cette deuxième partie consiste à implémenter une transformé de fourier rapide sur la carte Arch Pro à l'aide le module Grove Sound Sensor et Grove LCD qui permet afficher les caractéristique du son. Au début, on va chercher les différentes algorithme de FFT et leur performance sur internet. Après ces recherches, on décide d'utiliser l'algorithme de Cooley-Tukey. Avant d'implémenter, on va détermine la fréquence d'échantillonnage de signal qu'on va traiter.

Les différentes d'algorithmes de FFT:

La transformé de fourier rapide (FFT) est un algorithme basée sur la transformé de fourier discrète (TFD). Le FFT est plus rapide que le TFD qui a une complexité en O(n^2), pour le FFT la complexité est en O(n log n). Par exemple, pour n égale 1024 points, le temps d'exécution pour le FFT est plus de 100 fois plus courte que TFD.

$$f_j = \sum_{k=0}^{n-1} x_k e^{-\frac{2\pi i}{n} jk} \qquad j = 0, \ldots, n-1.$$

<formule de TFD>



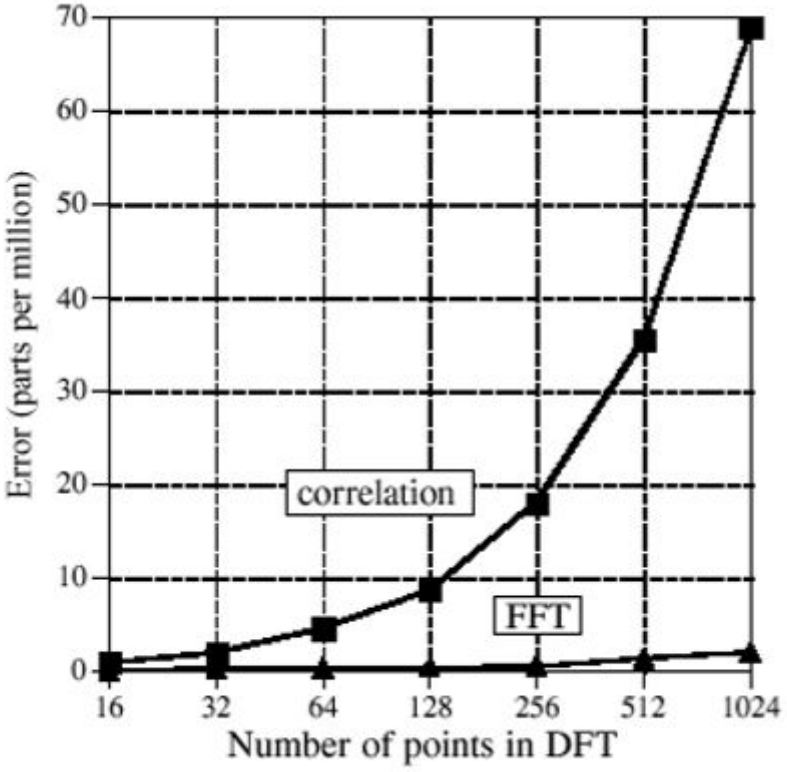
<Temp d'exécution pour le FFT et DFT>

En effet, il existe plusieurs algorithmes différentes pour l'implémentation de FFT. En 1965, Cooley et Tukey a proposé l'algorithme de Cooley-Tukey qui est le plus utilisé. Il a raccourci l'étape de calcule. Par exemple, pour TFD il faut N étape pour transformer le signal de domaine temporel à domaine fréquentiel, alors à partir de cette méthode, on n'a besoin que M=log2(N) étapes.

En 1976, M.Rader a proposé l'algorithme de Rader-Brenner, qui est similaire avec l'algorithme Cooley-Tukey. La différence est qu'il a utilisé la partie purement imaginaire de facteur Wn=exp(-2*pi*i/N). Cela sert à augmenter le nombre de l'addition et diminuer la stabilité numérique en réduisant le nombre de multiplication. Mais après cette méthode est remplacer par split-radix de Cooley-Tukey, ce qui garde tous les avantages sans diminuer la stabilité et la précision de calcule. En plus, il est réalisé avec moins de calcule de l'addition.

En 1978, Shmuel Winograd a proposé l'algorithme de Winograd. Cette méthode sert à trouver la calcule de FFT avec minimum de multiplication. Mais cela rend aussi l'augmentation de nombre de l'addition, en plus il n'est pas stable.

Selon l'analyse ci-dessus, on choisit d'utiliser l'algorithme de Cooley-Tukey qui est stable. En plus, il est plus précise que TFD, car il effectue moins d'opération, donc il y a moins de valeur arrondi.



<précision pour le FFT et DFT>

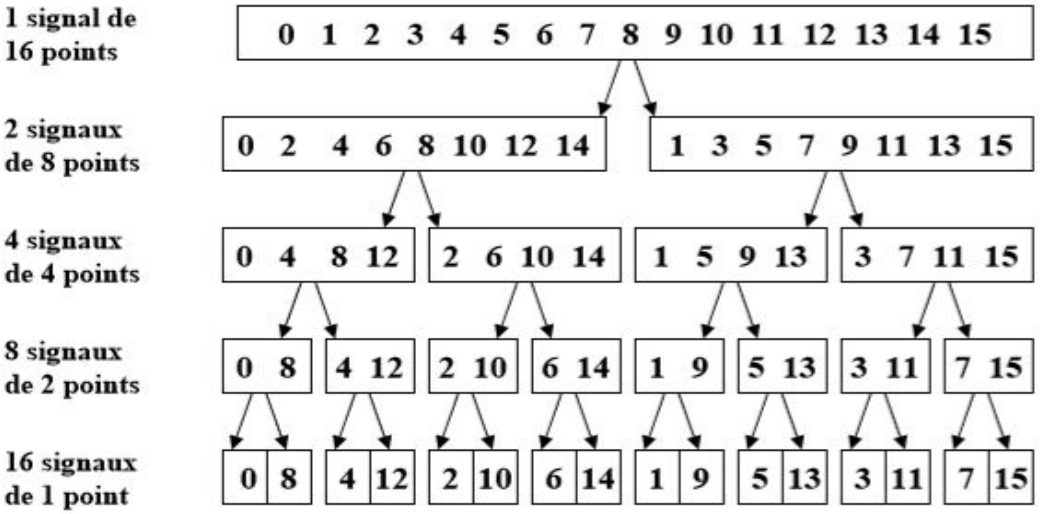
Échantillonnage du signal:

Avant de l'implémentation de FFT, il est important de déterminer la fréquence d'échantillonnage constante. Selon le théorème de Shannon qu'on a étudier en cours de traitement du signal, on sait que la fréquence d'échantillonnage d'un signal doit être supérieur ou égale au double de la différence entre les fréquences minimale et maximale. C'est à dire Fe ≥ 2*fmax. Dans notre cas, la fréquence maximale est 20 kHz et la fréquence minimale est 0 Hz. Donc on doit choisir une fréquence d'échantillonnage supérieur ou égale 40 kHz.

Alors pour effectuer l'échantillonnage, on a utilisé un Ticker qui vient de "mbed.h" et permet de lancer une fonction d'interruption à un intervalle de temps qui égale 1/Fe dans notre cas. Cette fonction va récupère la valeur analogique lu par le Grove Sound Senor et le stocker dans un tableau de complex. Lorsque cette tableau est complet, le Ticker va désactiver et on va traiter cette tableau avec l'algorithme de FFT.

L'algorithme Cooley-Tukey de FFT:

Selon l'algorithme qu'on a choisi, on doit d'abord décomposer le signal de domaine temporel à N points en N signaux de domaine temporel composés chacun d'un seul point.



<Décomposition>

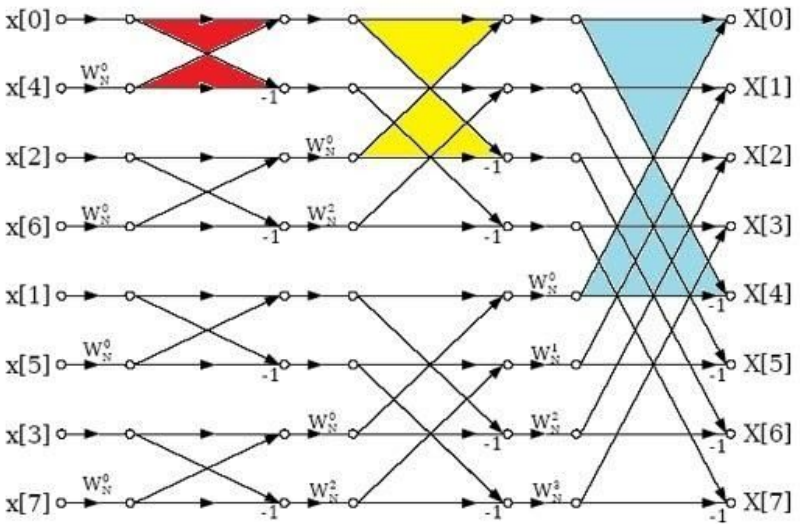
Cette décomposition est réalisé à l'aide la méthode Reder qui sert à inverser les bits. Par exemple, le nombre 1 en binaire est 0001, après la technique d'inversion de bits, il devient 1000 égale 8. Voici le schéma:

Sample numbers in normal order			Sample numbers after bit reversal	
Decimal	*Binary*		*Decimal*	*Binary*
0	0000		0	0000
1	0001		8	1000
2	0010		4	0100
3	0011		12	1100
4	0100		2	0010
5	0101		10	1010
6	0110	⇨	6	0100
7	0111		14	1110
8	1000		1	0001
9	1001		9	1001
10	1010		5	0101
11	1011		13	1101
12	1100		3	0011
13	1101		11	1011
14	1110		7	0111
15	1111		15	1111

<Inversion des bits>

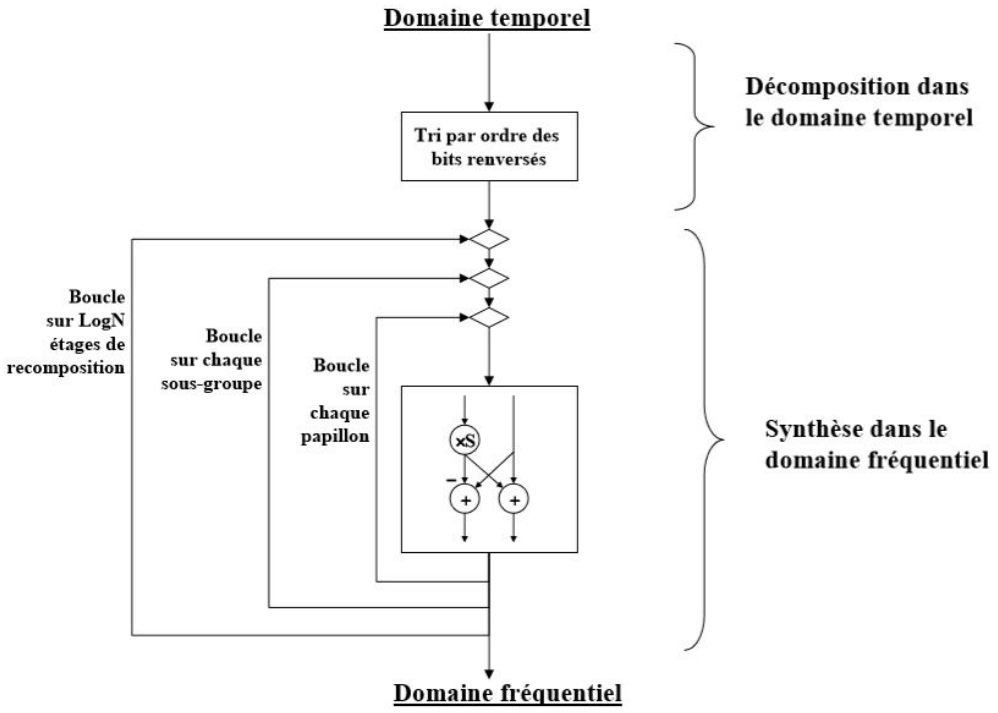
Après la décomposition, la valeur du signal à un point dans le domaine temporel est égale à sa valeur dans le domaine fréquentiel. Donc on n'a besoin aucune transformation pour modifier un tableau de point de domaine temporel à fréquentiel.

Ensuite, pour avoir un spectre de fréquence de N-points, on doit combiner ces N points en un seul signal à l'aide de calcul de "papillon". Ce qui réduit le nombre d'opération effectué pour la transformation. Par exemple, comme cet organigramme de 8 points, la reconstruction est durée de 3 étapes et pas 8. Et on obtient bien les signal en ordre.



<Organigramme de 8 points>

Diagramme entier de l'algorithme FFT :



L'implémentation de FFT:

En pratique, on a rencontré beaucoup de problèmes. Au début, on suit les étapes qu'on a indiqué dans la paragraphe précédente. Tout d'abord, on écrit une fonction Reder qui sert à inverser les bits pour séparer le tableau en partie pair et impair:

```
void Echange(complex a, complex b){
    complex tmp = a;
    a = b;
    b = tmp;
}

// methode reder qui faire inverser les bits
void Reder(complex *x, int N){

    int j=0;
    int k;

    for(int i = 0;i < N/2; i++){

        if(i < j){
            Echange(x[i], x[j]);
        }

        k = N >> 1;  //toujours égale 512 qui a un 1 en MSB

        while(j >= k){  // pour comparer si j a un 1 en MSB, sioui j est supérieur ou égale k

            j -= k;          // mettre MSB en 0
            k >>=1;          // k/2 pour comparer l'autre bits de j jusqu'à LSB
        }

        j += k; //sinon on met MSB de j en 1, j est le indice de taleau qui va faire échange
    }
}
```

Il est suffit de traiter que la moitié partie du tableau, quand la valeur d'indice i est inférieur de j, on va échanger leur position dans le tableau. Pour déterminer la valeur de j qui va être changé, on le compare avec la valeur k qui égale la moitié taille de tableau. Si j a un 1 en MSB, c'est à dire si j est supérieur ou égale à k, on va mettre MSB de j en 0, sinon, on va le mettre en 1. Mais quand on le teste dans le terminal avec un tableau de complex, les résultats sorties ne correspond pas ce qu'on attend, il n'y a rien qui a changé. Donc on modifie un peu notre code, on ajoute un nouveau tableau qui sert à stocker la valeur après échange. Cette fois on obtient un résultat correct, dans les photos ci dessous, on peut voir que le tableau a bien séparé en deux partie. Les indices des 4 valeur avant sont 0, 2, 4, 6 et celui des 4 valeurs derrière sont 1, 3, 5, 7.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
typedef struct Complex
{
    float re;
    float im;
} complex;

void Reder(complex *x, complex *r, int n, int l)
{
    int i = 0;
    int j = 0;
    short ind = 0;
    complex *temp = 0;

    temp = (complex *)malloc(sizeof(complex) * n);
    if (!temp) {
        return;
    }

    for(i=0; i<n; i++) {
        ind = 0;
        j = 0;
        do {
            ind |= (i>>(j++)) & 0x01;
            if(j<l)
            {
                ind <<= 1;
            }
        }while(j<l);

        if(ind < n) {
            temp[ind] = x[i];
        }
    }
    for (i=0; i<n; i++) {
```

```
38          r[i] = temp[i];
39      }
40      free(temp);
41
42  }
43  int main(void)
44  {
45      complex a[8] = {{1,0}, {2,1}, {3,2}, {4,3}, {5,4}, {6,5}, {7,6}, {8,7}};
46      complex r[8];
47      Reder(a, r, 8, 3);
48      printf("Inverser Bits\n");
49      for(int i=0; i<8; i++)
50      {
51          printf("Re:%g Im:%g\n", r[i].re, r[i].im);
52      }
53
54      return 0;
55  }
56
```

```
Inverser Bits
Re:1 Im:0
Re:5 Im:4
Re:3 Im:2
Re:7 Im:6
Re:2 Im:1
Re:6 Im:5
Re:4 Im:3
Re:8 Im:7
```

Ensuite, dans la fonction FFT, on stocke la valeur lu par le module Sound Sensor dans un tableau Complex et inverse les bits. Puis on fait 3 boucles de *for,* un de M étages, un de L étages qui sert à initialiser le facteur Wn et un de N/nbs étages qui utilise le calcul de papillons pour donner la valeur de signal en domaine fréquence en ordre.

```
for (int i=0; i<N; i++){  //boucle initialisation de X
    X[i].Re=x[i];
}

Reder(X,N);

//méthode papillon
for (stage=1; stage<=M; stage++){

    nbS =(int)(pow(2.0,stage));
    L = nbS/2;
    nbWn = N/nbS;

    for(int k=0; k<L; k++){

        Wn.Re=(double)cos(2*Pi*nbWn*k/N);
        Wn.Im=(double)sin(2*Pi*nbWn*k/N);


        for(iMax=k; iMax<N; iMax+=nbS){  // Boucle pour calculer les valeurs des indices avec le meme Wn

            pH = pX + iMax;        // Pointeurs des indice pour la valeur haute et basse du tableau X
            pL = pH + nbWn;

            //Multiplication entre Wn et indice
            tmp.Re = (pL->Re*Wn.Re) - (pL->Im * Wn.Im);
            tmp.Im = (pL->Re*Wn.Im) + (pL->Im * Wn.Re);
```

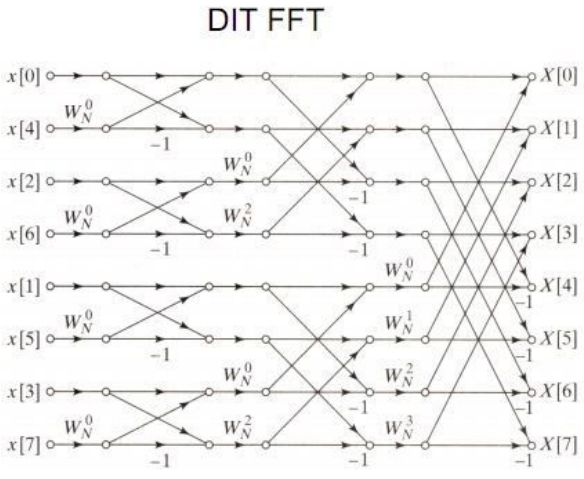
Et on l'utilise *read()* pour lire le signal d'entré et stocké dans un tableau de complex. Puis on utilise la Ticker qui attache la fonction *lireSignal()* et les fonctions qui servent à initialiser l'écran LCD et afficher les informations du signal sur l'écran dans le *main*.

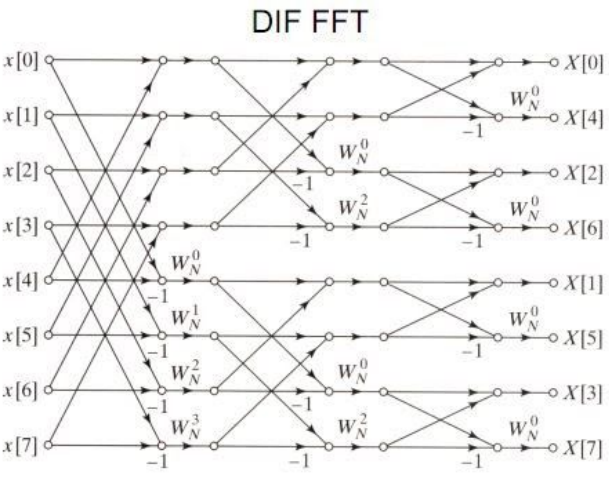
```
void lireSignal(){
    x[i] = sound.read();
    i++;

    if(i >= N){
        i=0;
        FFT(N,x,fft);
    }
}
```

Mais quand on compile la fonction dans le logiciel keil, cela ne marche pas. On est sur que la fonction affichage et initialisation est correcte, parce que ces fonctions sont des fonctions qu'on a déjà utilisé dans le TP. Donc on en déduit que la problème est sur la fonction de FFT. Après plusieur modifications qu'on a effectué, il ne marche encore pas.

Donc on décider de chercher sur internet, et on trouve un site qui donne un exemple de l'implémentation de FFT de l'algorithme de Cooley-Tukey sur les différentes langages (donné dans la bibliographie). Il utilise la formule TFD pour calculer chaque signal et puis il inverse les bits. Comme cette schéma représente, il utilise un DIF FFT d'abord, après il utilise un décimation pour tirer les signaux en ordre. C'est un étape inverse qu'on a fait avant.

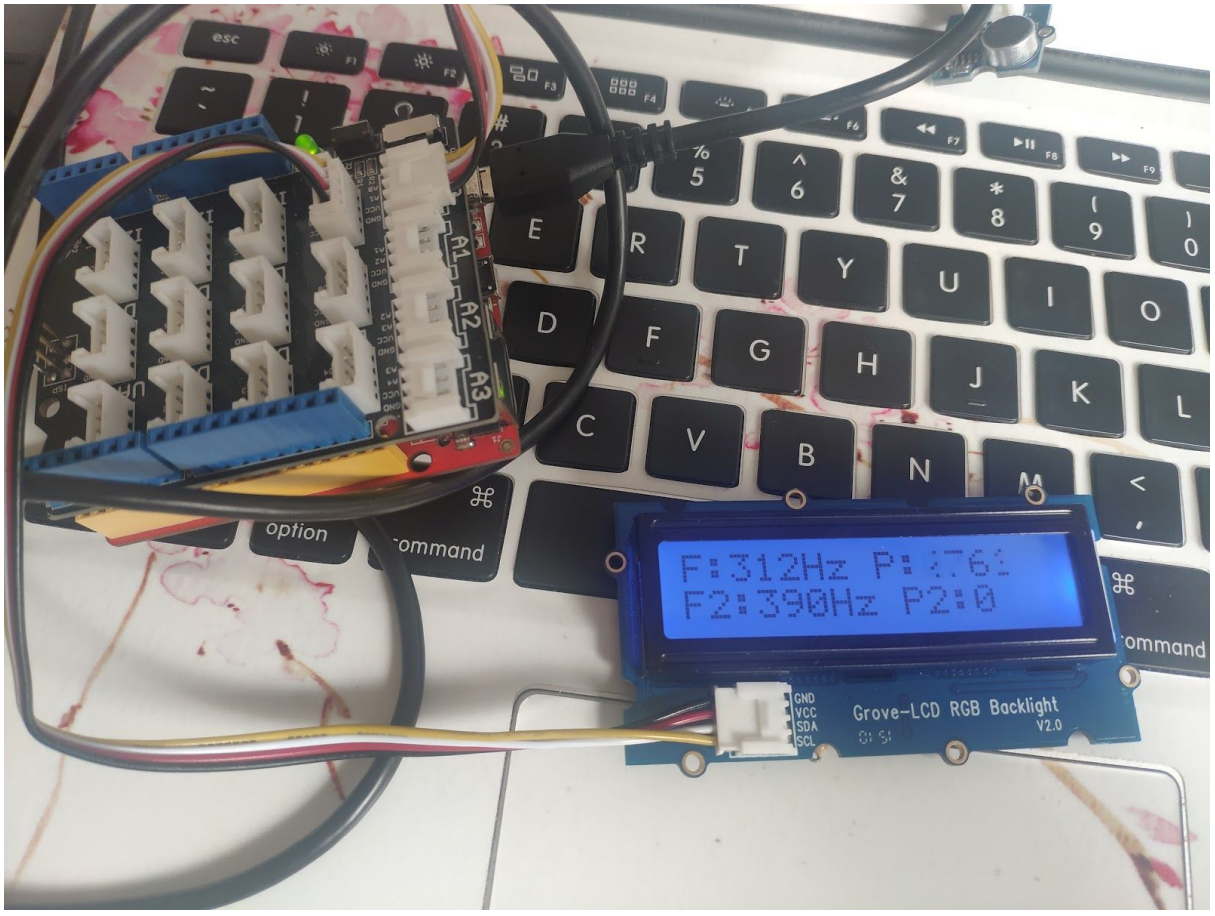




Ensuite, pour obtenir les caractéristiques des signaux, on a fait une fonction *spectre().* Dans cette fonction, on a fait un boucle *for* pour calculer le vecteur de fréquence et la tableau de puissance pour un signal et trouver la fréquence fondamentale de cette signal et son puissance maximal. Puis on utilise un libraire "Grove_LCD_RGB_Backlight.h" qui sert à afficher sur l'écran LCD, c'est plus simple à utiliser que notre fonction avant. On afficher la fréquence fondamental de cette signal et son puissance sur la premier ligne, et sur la deuxième ligne on affiche la puissance et la fréquence de la deuxième harmonique qui est la prochaine point dans le vecteur de fréquence. C'est à dire si la fréquence fondamentale est en indice i, alors la deuxième est en indice i+1.

A la fin, dans le *main* on initialise l'écran LCD et attache la fonction *liresignal()*.

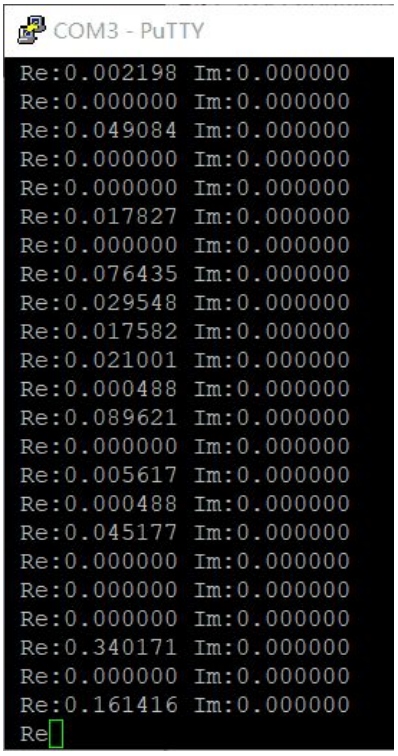
Cette fois, on a réussi à compiler mais on n'a rien afficher sur l'écran, c'est à cause le nombre de points utilisé. Donc on modifie le nombre de points du signal qui passe de 1024 à 512. Enfin, on a obtenu bien un résultat.

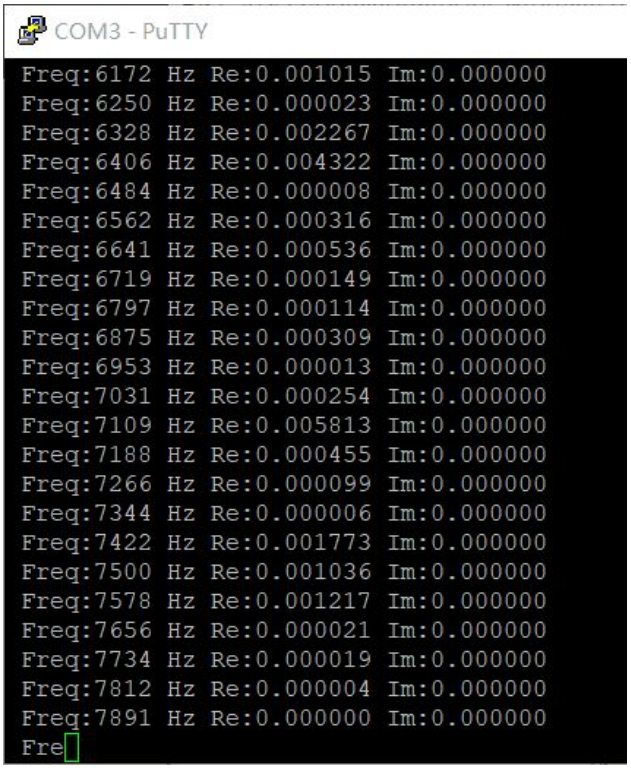


Voici un lien de vidéo pour notre affichage:

https://drive.google.com/file/d/1YPtL7KSSv5TsdMre3cXXrWTPyf4QKBY1/view?usp=sharing

On a aussi testé dans le terminal putty:





En effet, les résultats obtenue n'est pas trop précise, cela est peut être à cause de peu de nombre de points et la rapidité de l'implémentation de FFT. On doit encore améliorer sur cette partie.

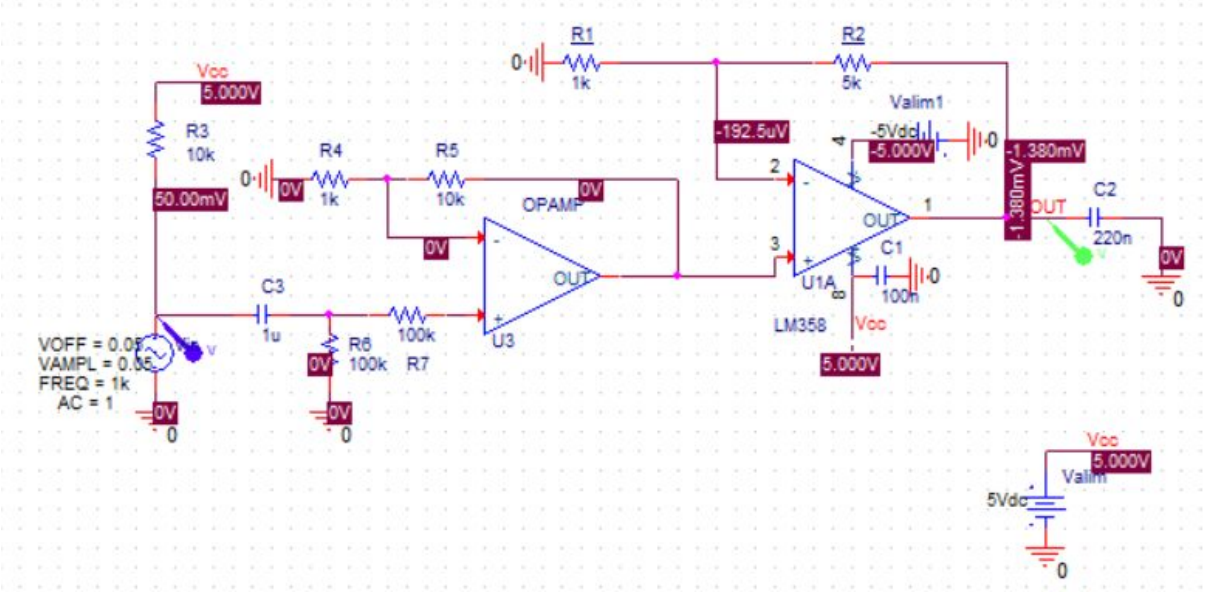
IV. Amélioration

L'implémentation de FFT permet d'analyser les signaux à travers des modules grove sound sensor ou grove speaker, mais ces modules peuvent être améliorer. On peut proposer les différentes solutions pour ajuster les performances des modules. Par exemple varier manuellement ou dynamiquement le gain d'amplificateur LM358 ou LM386, trouver les alternatives de classe AB et D pour remplacer LM386. Donc dans un premier temps, on analyse LM358 puis LM386 et à la fin les alternatives de classe AB et D. Pour simuler les circuits on utilise le logiciel Pspice.

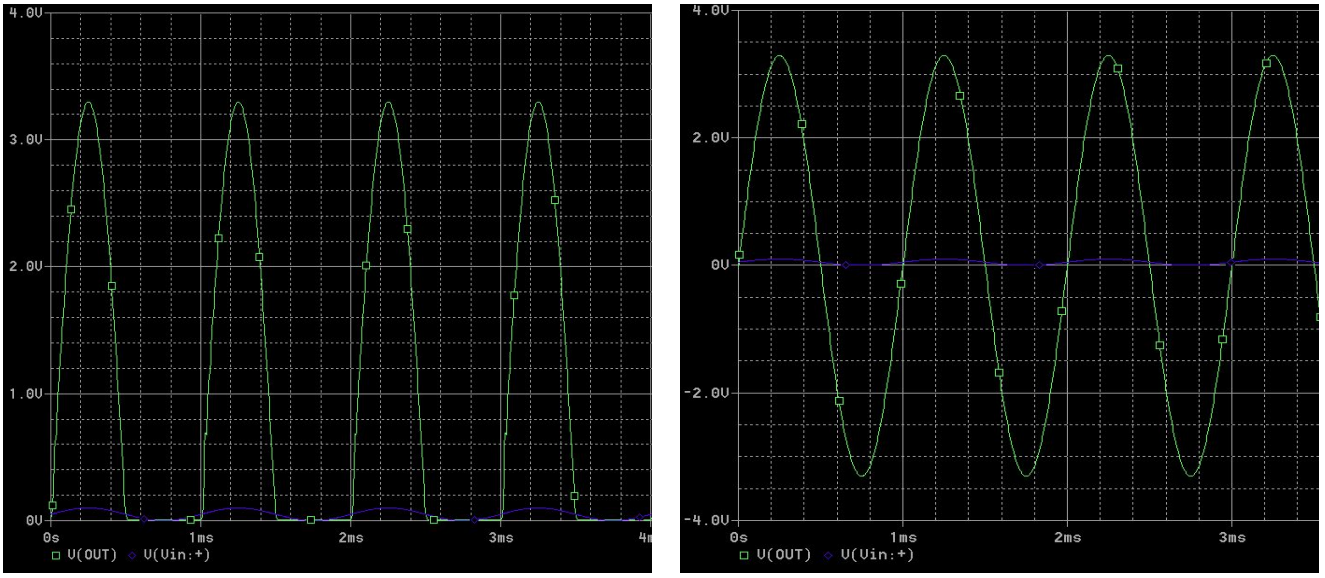
LM358:

- **Amplifier bien la composante alternative négative:**

Or < VOUT = AOP (IN+ - IN-) >, donc pour amplifier la composante négative de signal, il faut alors mettre une tension à Vin- avec une valeur de -Valim. D'ici, parce que la tension d'alimentation de LM358 est 5V, donc on met Vin- à une valeur de -5V:



< Le montage de <Sound Sensor> avec Vin- = 0v et Vin- = "-Valim" >



< La simulation de circuit avec Vin- = 0V et avec Vin- = - Valim >

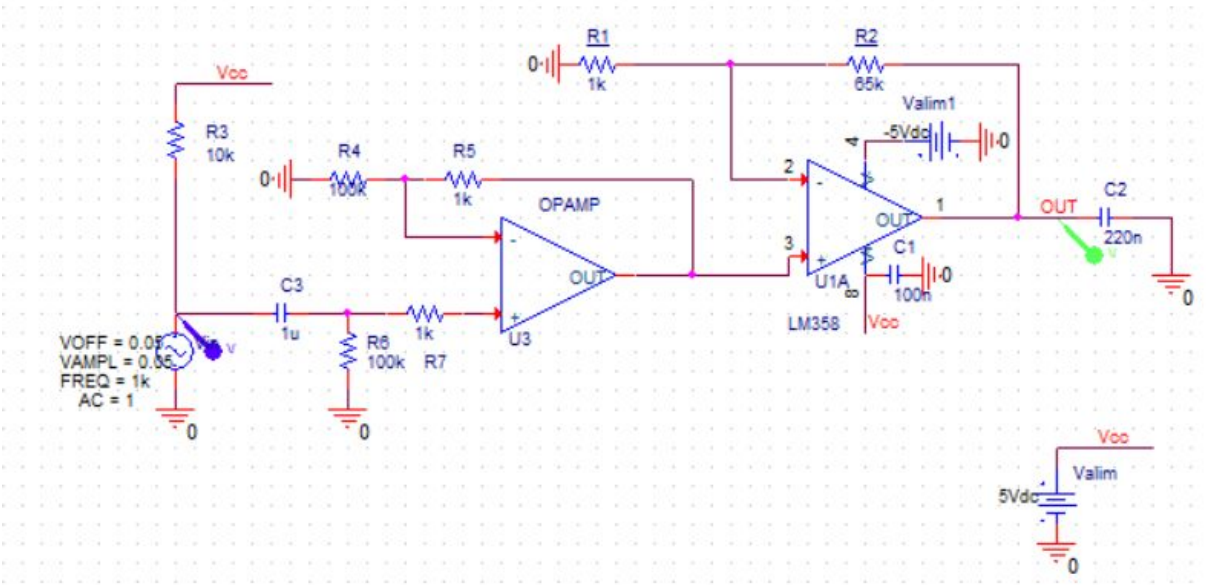
Dans le graphique du similation, on voit bien que la composant négative du signal d'entrée est amplifiée.

- **Ajuster manuellement le gain de l'amplificateur:**

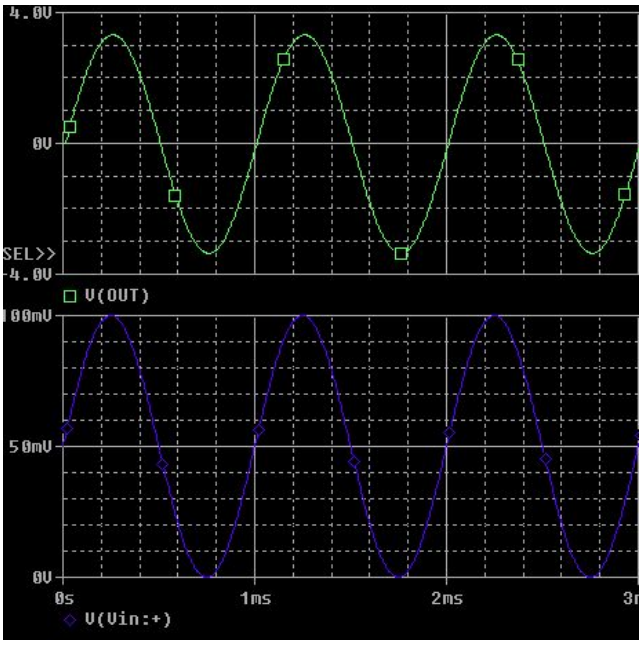
On trouve que la bande de gain du LM358 est de 25 à 100, avec la formule (R1+R2)/R1. D'ici le signal d'entrée est de 0.1V et la valeur maximale de l'entrée du CAN de microcontrôleur est 3.3V, c'est à dire que la sortie du montage doit inférieur à 3.3V. Dans ce cas là, on a un gain A = 33, ce qui est compris dans la bande de gain, donc c'est cohérent.

La formule de gain total du montage est (1+ R2/R1) * (1+R5/R4) *(1+R7/R6), qui doit être inférieur à 33. Donc on essayait de mettre R1 =1K, R2 = 32K, R4 = 100K, R5 = 1K, R6 = 100K et R7 = 1K pour mettre le gain de U3 à 1 et le gain de LM358 à 33. En ce moment, le Ath = 33, par contre, le signal sortie est à 1.65V, donc le Aexp est 16.5. On obtient une proportion de 2 entre le gain théorique et le gain expérimental.

Alors que l'on peut modifier la résistance R2 à 65K pour obtenir un gain expérimental de LM358 de 66, au lieu un gain de 33 en théorique et un tension de sortie est à 3.3V.



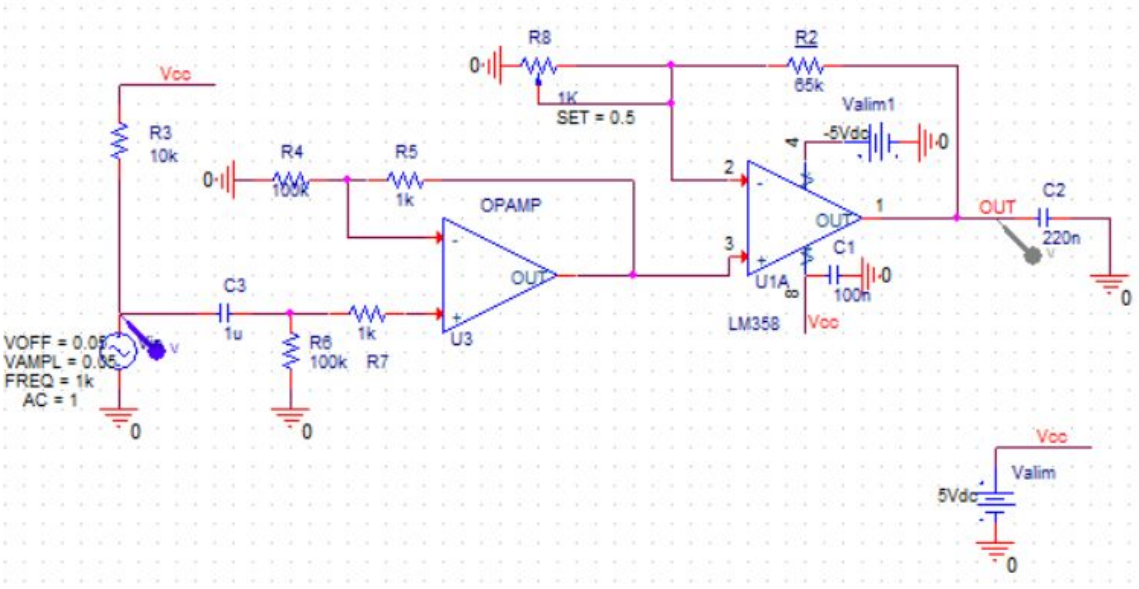
< Le montage modifié pour avoir la plus grande d'entrée de CAN >



<Simulation du montage précédant >

- **Un gain variable contrôlé par le microcontrôleur:**

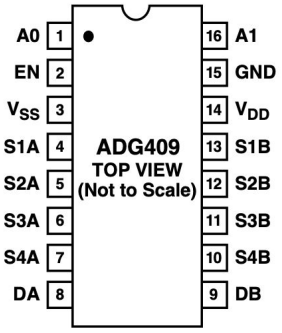
Tout simplement on peut mettre un potentiomètre ou un MUX à la place de R1. On préfère d'utiliser un potentiomètre de 1k avec un coefficient de x à y. Par apport à MUX, on préfère d'utiliser un composant ADG409 qui est un multiplexeur avec 8 channels. Ce composant change l'un des 8 différents inputs à un output commun qui est déterminé par l'adresse binaire de 2 bits A0 et A1. le EN input contrôle l'activation et la désactivation de device, lorsqu'il est en mode "Disable", tous les channels ont changé à mode OFF.



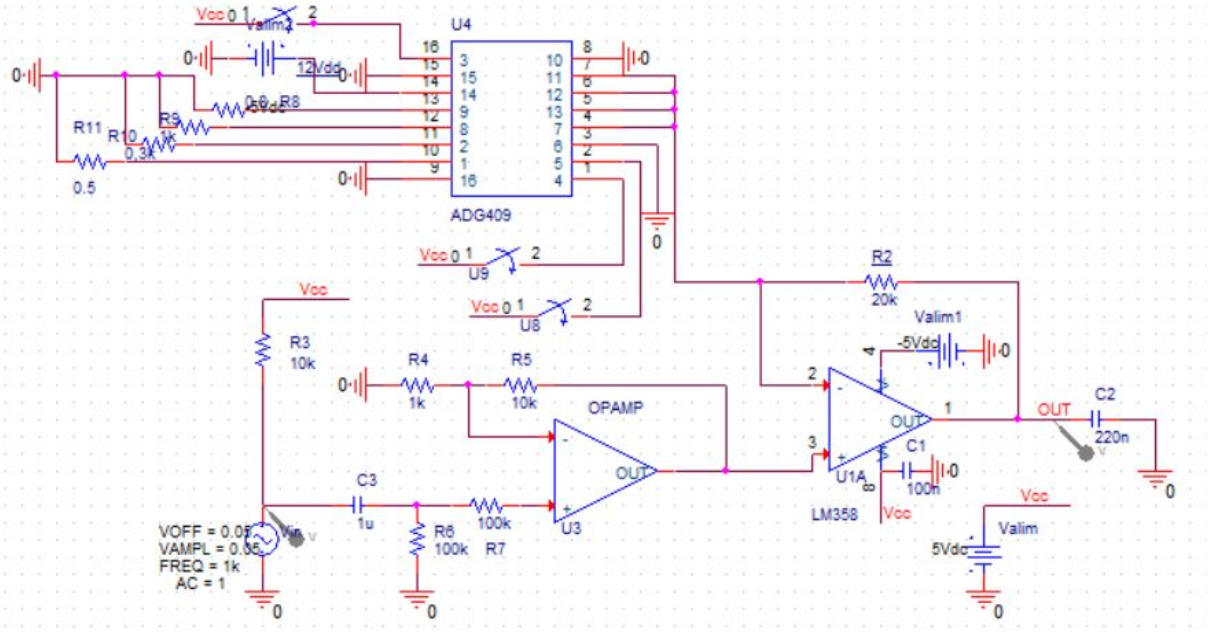
< Le montage modifié avec un gain variable avec le potentiomètre >

ADG409 Truth Table

A1	A0	EN	ON SWITCH PAIR
X	X	0	NONE
0	0	1	1
0	1	1	2
1	0	1	3
1	1	1	4



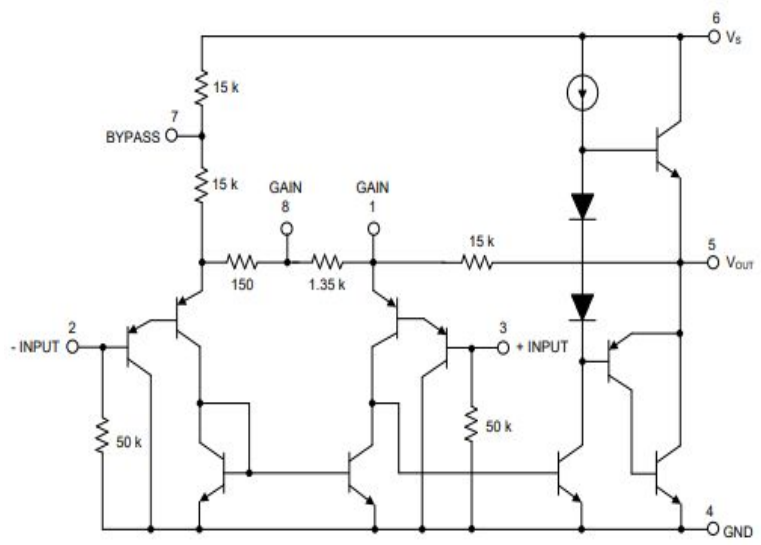
< Tableau de vérité de ADG409 et sa configuration >



< Le montage modifié avec un gain variable avec le multiplexeur >

LM386:

D'après le fiche technique et le schéma de LM386 , on peut voir 3 exemples de montage pour le gain qui égale 20, 50 et 200.



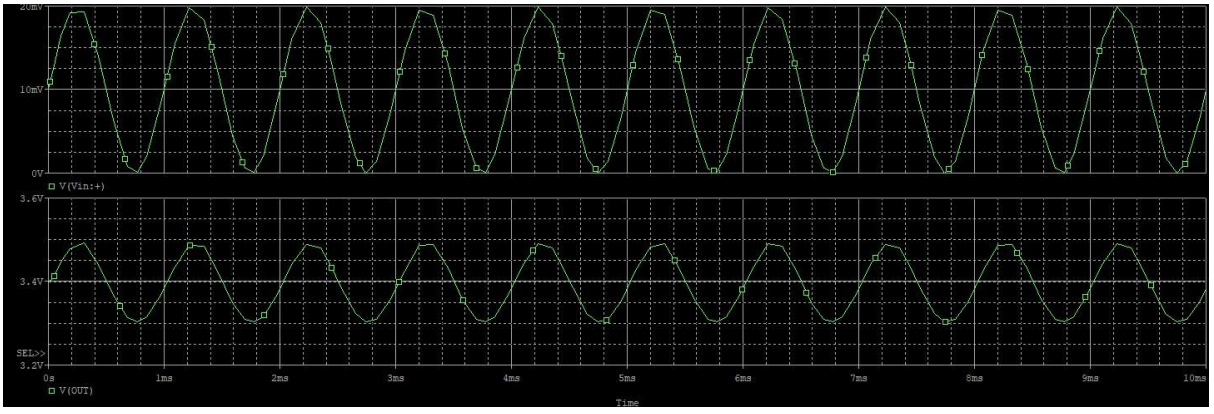
<Figure de schéma LM386>

On voit qu'il existe une résistance de 1,35kΩ entre pin1 et pin8 et une résistance de 150Ω près de pin8.

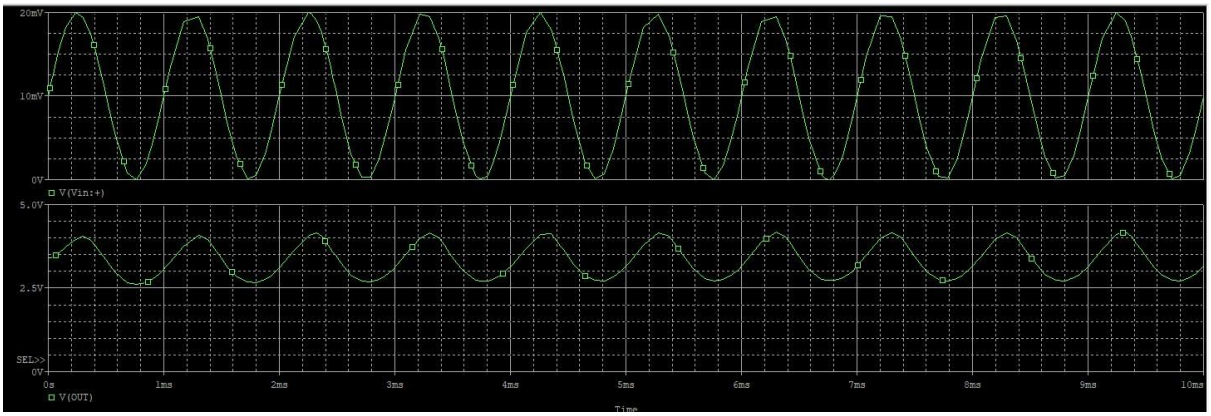
Donc le gain on peut calculer par la résistance effective Reff qui est le résultat de la résistance en parallèle avec 1,35kΩ, Reff=(R*1,35k)/(R+1,35k), le gain est donc (2*15kΩ/Ref).

Si il n'y a rien de composant entre pin1 et pin8, le gain est donc 20. Si on place une capacité de 10uF entre pin1 et pin8, le gain est maximum de 200. Donc le gain est ajustable entre 20 et 200. Si on place une résistance de 1,2kΩ et une capacité 10uF entre pin1 et pin8 le gain est 50.

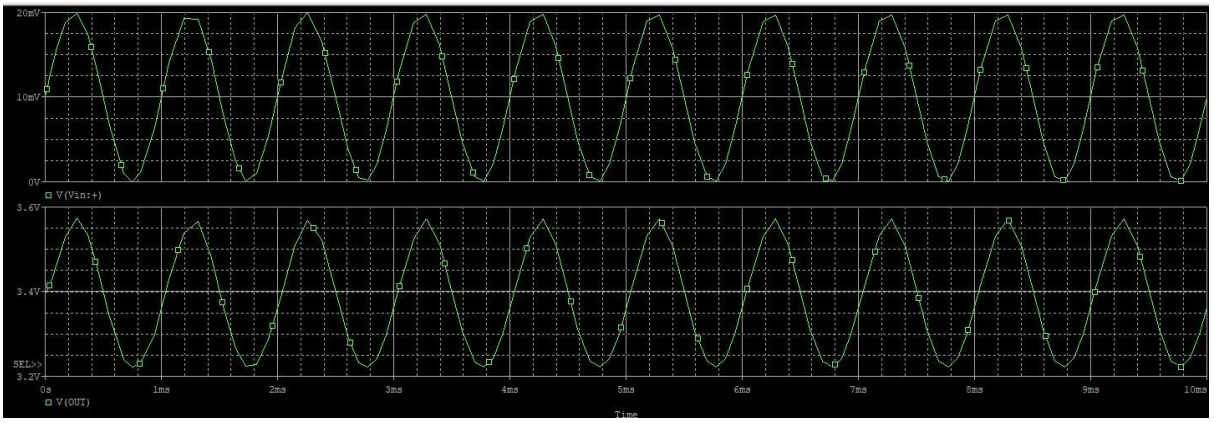
D'après la simulation de pspice :



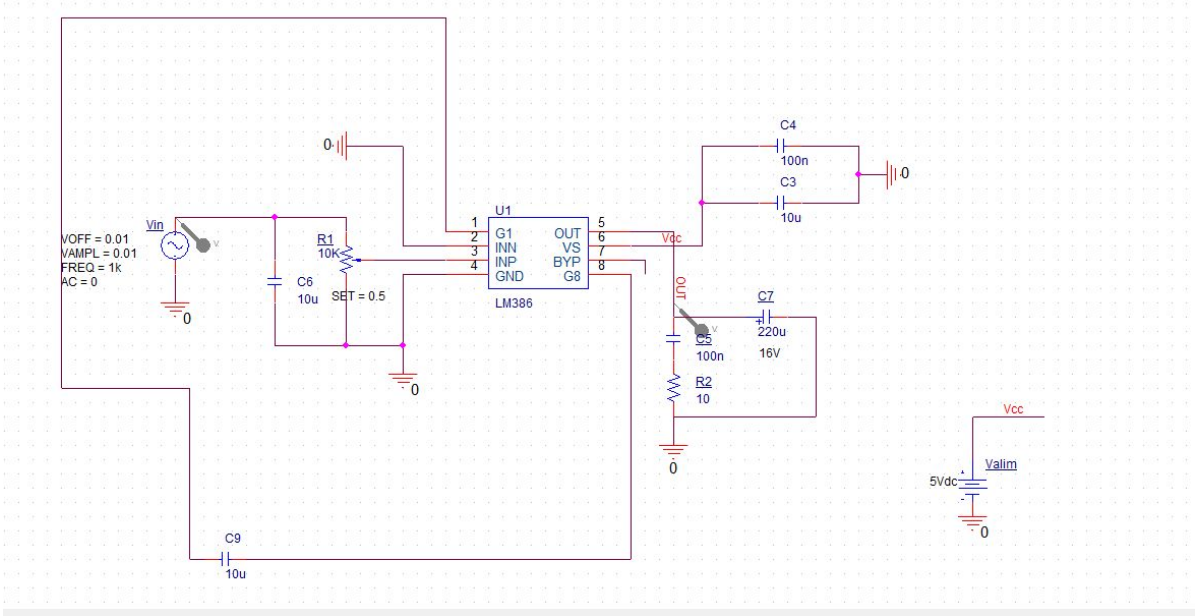
<Simulation de rien entre pin1 et pin8>



<Simulation de 10uF de capacité entre pin1 et pin8>



<Simulation de 10uF de capacité et 1,2kΩ de résistance entre pin1 et pin8>



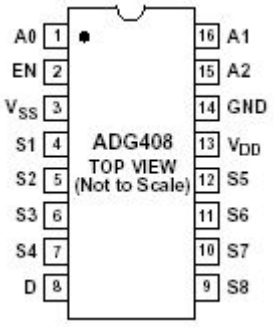
<figure de circuit>

- **Ajuster manuellement le gain de l'amplificateur:**

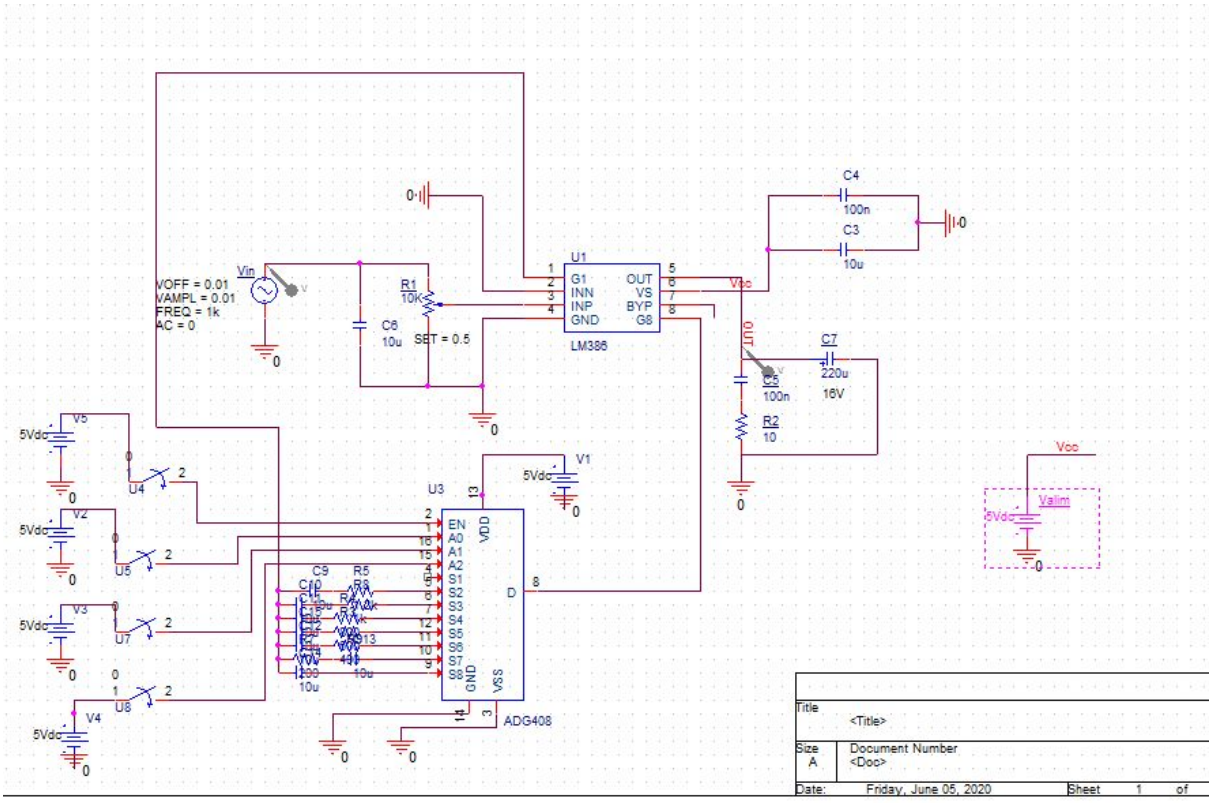
On peut ajouter un MUX de ADG408 de 3 entrées A0 A1 et A2 qui est contrôlés par 3 switch de SPST. Il permet d'autoriser une seule sortie avec la capacité de 10uF et la résistance entre 200Ω et 1,2kΩ ou vide, donc le gain est entre 20 à 200. On peut aussi utiliser une résistance variable pour contrôler manuellement le gain.

Table 6. ADG408 Truth Table

A2	A1	A0	EN	On Switch
X	X	X	0	None
0	0	0	1	1
0	0	1	1	2
0	1	0	1	3
0	1	1	1	4
1	0	0	1	5
1	0	1	1	6
1	1	0	1	7
1	1	1	1	8



<le tableau de vérité de ADG408 et le pin de ADG408>

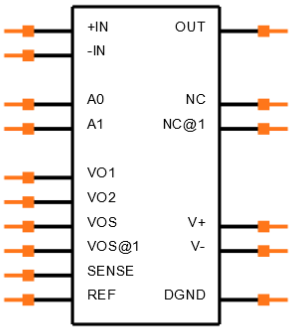


<figure de circuit de MUX>

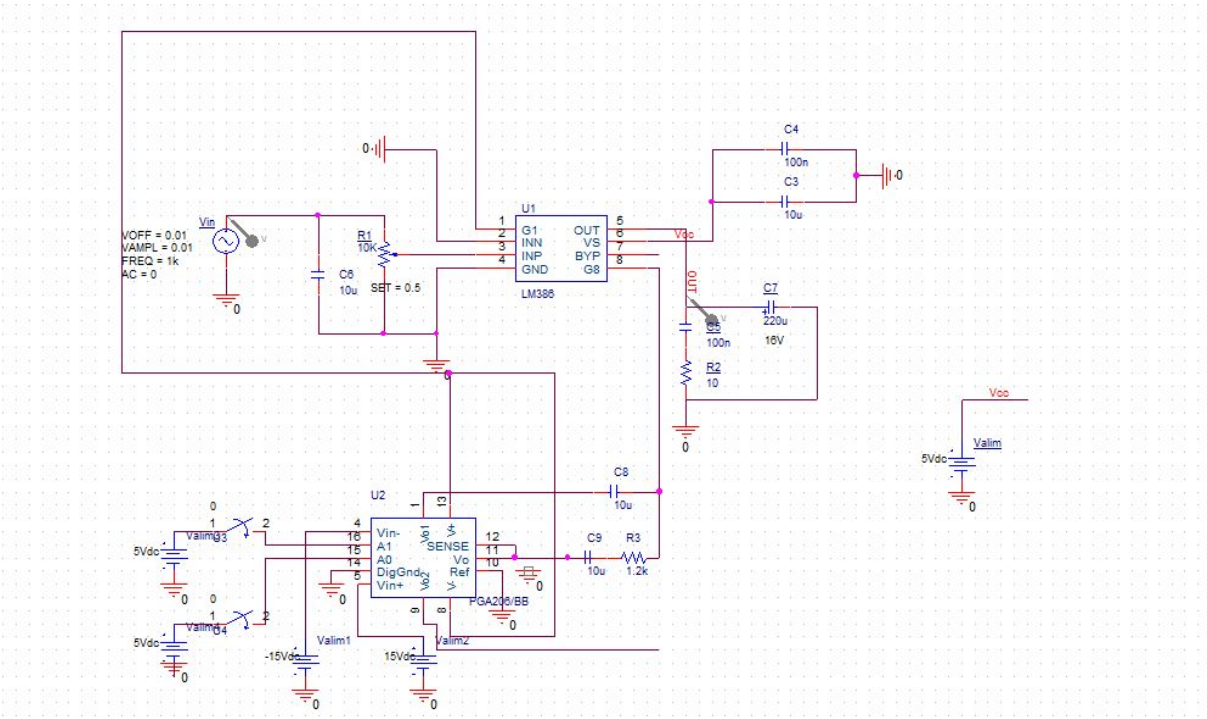
- **Ajuster dynamiquement le gain de l'amplificateur par le microcontrolleur**

Grâce au composant de PGA (Programmable-gain amplifier), on peut contrôler le gain de PGA par les différents des entrées, il ressemble MUX. Dans notre cas, on choisit PGA207. Le tableau de gain est ci-dessous:

GAIN (V/V)	A_1	A_0
1	0	0
2	0	1
5	1	0
10	1	1



<le tableau de gain et le pin de PGA207>



<figure de circuit de MCU>

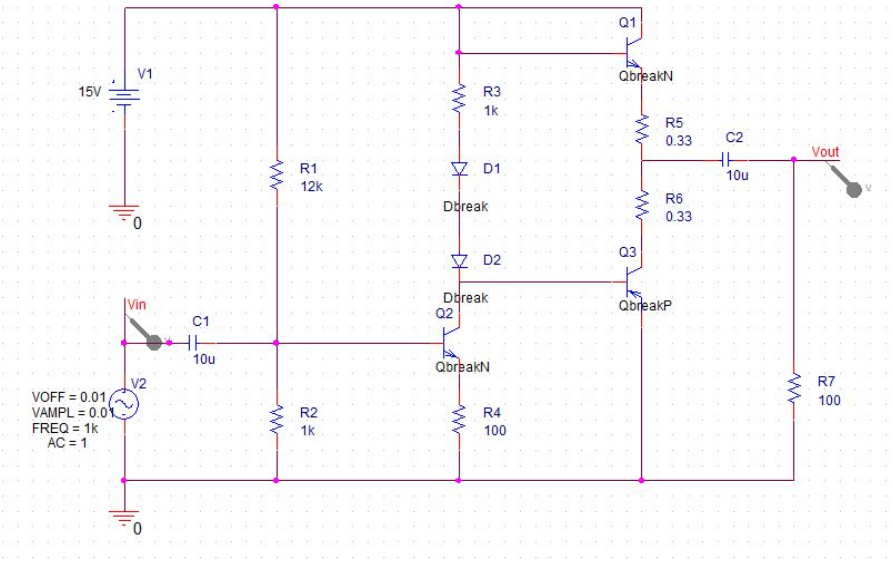
On ne peut pas simuler avec la carte, donc on remplace les entrées par Vdc et switch SPST.

Pour remplacer LM386 par les dispositifs de classe AB et de classe D, on a fait d'abord des recherches sur internet afin de comprendre les principes de la classe AB et de la classe D.

Classe AB

La classe AB comporte les BJT. Elle est obtenue en modifiant le circuit de la classe A ou de la classe B. En l'absence de signal, les BJTs sont toujours passantes selon le signe de l'alternance, c'est à dire qu'il y a encore des courants passent par les BJTs, et les courants circuits sont tous sorties à output. Cela permet donc d'économiser la consommation (soit avec la meilleur amplification soit avec le meilleur de la consommation d'électricité).

On a pris le montage de Pull-up par pont, on alimente le montage en +-15V. Et on a pris ainsi des résistances reliant avec les diodes qui restent passantes en imposant un courant de pont non nul.



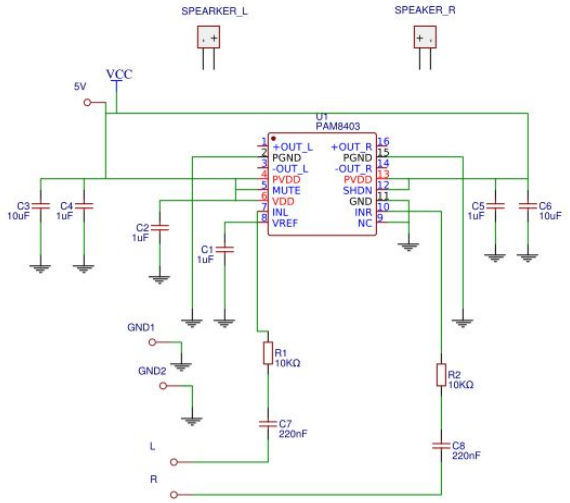
< Figure de circuit de la class AB >

Classe D

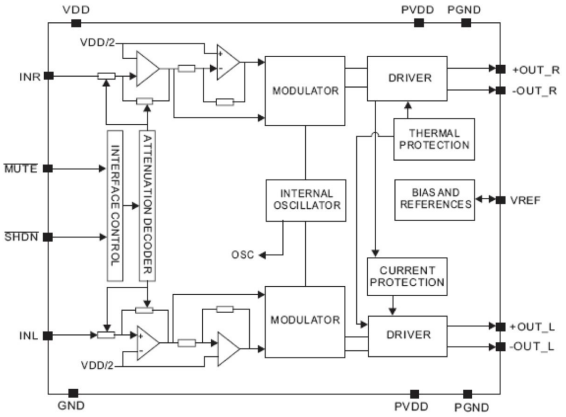
La classe D comporte 2 composantes MOSFETs qui permet d'économiser la consommation de l'électricité, or ils ne sont jamais passantes en même temps. Lorsque l'amplificateur est en cours de la fonction, la classe D transforme le signal d'entrée à un signal avec haute fréquence en utilisant le PWM, l'échantillonnage et les comparateurs. Cela est très efficace pour les signaux de basses fréquences.

Lorsque les MOSFETs sont tous bloquées, il n'y a pas de courant passante, cela permet d'économiser l'électricité, donc il y a moins de consommation.

On a trouvé un composant qui est PAM8403, il consiste 2 stages d'amplification et il travaille alternativement : le gain du 1er stage peut varier lorsque le gain de la 2e stage est fixé.



< Figure de circuit de la class D >



< Configuration de PAM8403 >

Remarque

Durant la partie d'amélioration, on a rencontré parfois des difficultés, par exemple: LM386 n'est pas fourni initialement dans le libraire, pour résoudre cette problème on a cherché le fichier cir sur Internet grâce au logiciel Model Editor d'Orcad on obtient le fichier olb et lib.

On n'avait pas fait attention aussi sur la tension d'entrée de composantes, et sur les valeurs limites de composants (par exemple la bande passante de gain). Parce que les sensors sont fournis au microcontrôleur, alors la tension limite de microcontrôleur impose les influences sur le choix de composant et les valeurs de tensions de sensor.

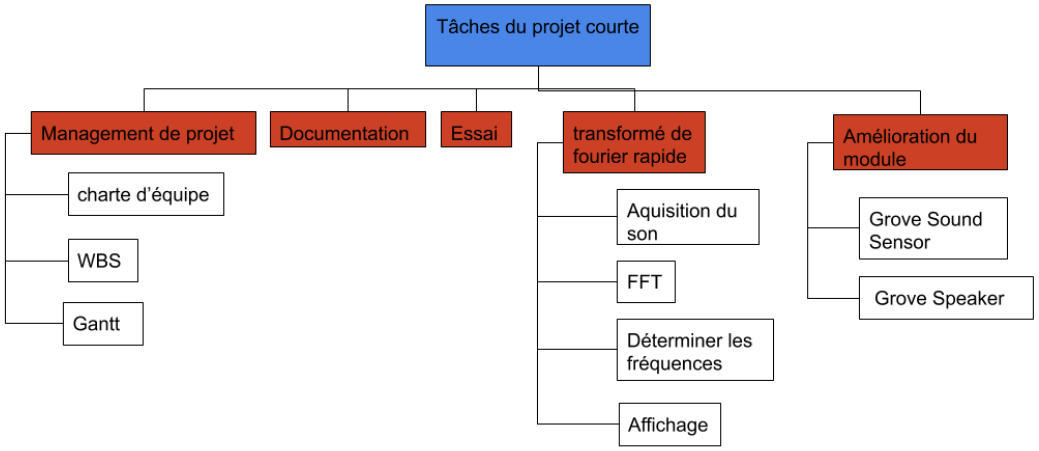
V. Organisation

Après l'amélioration, on peut avancer dans le cadre de critique de l'organisation. On va parler de 4 parties: fixer la problématique, répartir les rôles, gérer les tâches et organiser les réunions.

D’abord pour fixer la problématique, on confirme les solutions logicielles pour avoir une précision améliorée et les meilleures performances de temps réel en se basant sur une recherche scientifique qui contient des simulations et des calculs à effectuer.

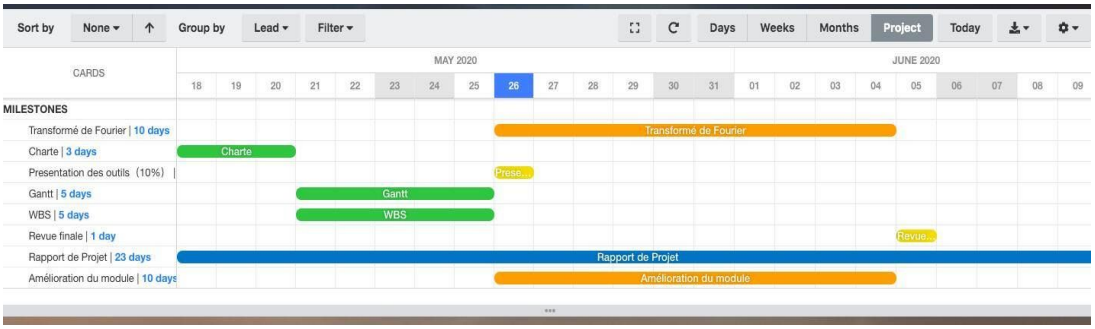
Pour répartir les rôles, on le détermine à l'aide le plateforme Trello, le test de Belbin-Rôles et l’autoévaluation des compétences. Les deux membres font une partie, chacun ses deux membres occupent son tâche soit faire la simulation, soit faire le recherche.

Pour gérer les tâches, on utilise Trello, WBS et Gantt



<WBS>

On peut confirmer les dates et les processus du projet et observer les délais des programmes.



<Gantt>

Pour organiser les réunions, malgré de la fin du semestre, chacun occupe beaucoup de travail, plus la situation actuelle, les réunions ne sont pas bien organisées. Donc la communication entre chaque membre est manquée un peu.

En général, on a bien réparti les rôles, mais la manque des réunions qui cause les tâches n’ont pas bien effectuées, plus la confinement on ne peut pas bien consulter.

VI. Conclusion

Ce projet nous a permis de fusionner nos acquis en différents modules tels que le traitement de signal, management,électronique analogique et numérique etc. Ainsi d'améliorer nos compétences en échangeant entre membres d'équipe les informations nécessaires pour réaliser ce projet. Malgré la situation actuelle et les conditions du travail, les difficultés existent toujours.

VII. Bibliographie

- https://files.seeedstudio.com/wiki/Grove_Sound_Sensor/res/LM386.pdf
- https://www.ti.com.cn/cn/lit/ds/symlink/lm386.pdf?ts=1590421170723
- https://en.wikipedia.org//wiki/LM386
- https://wiki.seeedstudio.com/Grove-Sound_Sensor/
- https://wiki.seeedstudio.com/Grove-Speaker/

- https://fr.wikipedia.org/wiki/Transformation_de_Fourier_rapide
- http://nicolas.thiery.name/DESS/Notes/Cours4.pdf
- https://www.researchgate.net/publication/340448622_FFT_Algorithm
- https://rosettacode.org/wiki/Fast_Fourier_transform
- https://en.wikipedia.org/wiki/Cooley–Tukey_FFT_algorithm
- https://os.mbed.com/docs/mbed-os/v6.0/mbed-os-api-doxy/_ticker_8h_source.html

-LM386: http://www.ti.com/lit/ds/symlink/lm386.pdf
-PGA207: https://www.ti.com/lit/ds/sbos033/sbos033.pdf?ts=1591360317062
-ADG408/409:https://www.analog.com/media/en/technical-documentation/data-sheets/ADG408_409.pdf
-PAM8403:https://www.diodes.com/assets/Datasheets/PAM8403.pdf