TP ELECTRONIQUE

Par VEREBELY Paul-loup & *HugoDUPRAT* & *NoéDIBLASI*

June 2025

Contents

List of Figures

1 Introduction

Le présent document retrace l'intégralité du déroulement du projet de fin d'année d'électronique AP3. Afin d'en assurer une lecture claire et structurée, le rapport est organisé en plusieurs sections distinctes.

Dans un premier temps, la section contexte présente les objectifs du projet, le cahier des charges initial ainsi que les principales contraintes et enjeux identifiés.

La section dimensionnement expose ensuite l'ensemble des calculs, hypothèses et choix techniques réalisés par l'équipe pour concevoir le système électronique répondant aux spécifications.

La section informatique décrit en détail le développement logiciel associé au projet, en précisant la logique de programmation, l'architecture du code et les principales fonctionnalités implémentées.

Enfin, la section tests et difficultés rencontrées présente les différentes étapes de validation du système, les problèmes techniques rencontrés au cours du projet, ainsi que les solutions apportées par l'équipe pour y remédier.

Pour ce qui est de la répartition de l'équipe, celle-ci est composée de 3 membres qui ont tous eux une participations sur les différents aspects du projet.

Ce rapport a pour but aussi de servir d'exemple en cas de reproduction du projet de la part du/des lecteurs.

2 Cahier des charges initiales

Ce projet permet de regrouper toutes les connaissances apprisent en électronique durant l'année. Pour pouvoir mettre tous ca dans un projet réalisable ne un minimum de temps, Antoine PIROG à écrit un cahier des charges complets et précis, voici sont contenus:

Le système à concevoir est un vu-mètre à LEDs RGBW destiné à visualiser le niveau sonore d'un signal audio en temps réel. L'architecture du projet s'articule en plusieurs blocs fonctionnels décrits ci-dessous.

2.1 Alimentation

- Tension unique d'alimentation : **+5V DC**
- Cette tension alimente l'ensemble de la carte électronique.
- Elle doit être respectée scrupuleusement pour éviter d'endommager les composants sensibles (notamment les LEDs et le microcontrôleur).

2.2 Acquisition et conditionnement du signal audio

- Le signal audio d'entrée est divisé en deux voies :
  - une vers les filtres analogiques,
  - une vers une sortie audio (vers des enceintes actives).
- Les signaux sont ensuite filtrés par 4 filtres passe-bande analogiques définissant les bandes de fréquence :
  - **Basses** : < 250 Hz
  - **Bas-médiums** : 250 Hz – 1 kHz
  - **Hauts-médiums** : 1 kHz – 4 kHz
  - **Aigus** : > 4 kHz
- Chaque filtre devra être correctement dimensionné à l'aide des AOP et composants passifs.

2.3 Traitement numérique

- Le microcontrôleur **PIC18F25K40** effectue :
  - l'acquisition analogique des signaux issus des filtres,
  - le traitement numérique du niveau dans chaque bande,
  - le formatage des données en vue de l'affichage.
- Le code est écrit en **langage C** pour la logique générale et en **assembleur** pour la commande des LEDs (bitbanging haute précision).

2.4 Affichage LED

- Affichage sur une matrice de **64 LEDs RGBW SK6812**, adressables individuellement.
- Chaque LED est contrôlée via un protocole série, selon une séquence de **2048 bits** (64 LEDs $\times$ 4 couleurs $\times$ 8 bits).
- L'ordre des couleurs est : **vert**, **rouge**, **bleu**, **blanc**.
- L'intensité de chaque couleur est codée sur 8 bits (valeurs recommandées : 16 à 32 pour éviter une intensité lumineuse excessive).

2.5 Interface utilisateur et tests

- **9 LEDs de test** et **2 boutons poussoirs** disponibles pour :
  - tester le programme indépendamment de la matrice LED,
  - implémenter des fonctionnalités supplémentaires (pause, modes d'affichage...).

2.6 Contraintes de développement

- **Durée du projet** : 7 séances
- **Livrables attendus** :
  - Prototype fonctionnel,
  - Code source complet (C + assembleur),
  - Rapport écrit incluant schémas, calculs, Bode, oscillogrammes, tableau de composants.
- **Tests à effectuer** :
  - Oscillogrammes sur les Test Points,
  - Validation des filtres avec générateur de fonctions,
  - Contrôle du timing assembleur avec oscilloscope.

3 Dimensionnement Electronique

Dans ce projet, le dimensionnement est une part importante qui ne doit pas être négliger. Plusieurs partie de la carte doivent être dimensionner correctement afin de pouvoir conserver un haut niveau de performance et un bonne intégrité du circuit. Pour décrire tous cela, le dimensionnement se fera en plusieurs blocs afin de facilité le travail et leurs compréhensions. Enfin pour des raisons de sécurité, tous les tests de dimensionnements se feront sur Falstad et avec la serie E6.

3.1 Tension de référence

Quand on regarde le schéma électrique, on peut remarquer que une bonne partie des AOPs sont alimenter avec Vref, cette tension fait office de tension de référence, elle permettera a notre signal audio d'osciller vers 2.5V car le PICKIT fonctionne entre 0 et 5 Volt. Pour cela on retrouve un pont diviseur de tension alimenter en 5V ainsi que un montage AOP suiveur.

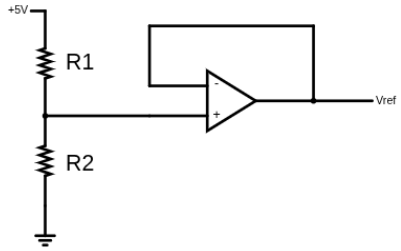

Figure 1: Schéma Pont diviseur de tension

Formule pont diviseur de tension: $Vref = U * \frac{R2}{R2+R1}$ Formule AOP Suiveur: $Vs = Ve$

Etant donné que notre tension d'entré est à 5V et que on veut 2.5V en sortie il nous suffit de diviser la tension par deux. Pour cela et grace au pont diviseur de tension, on va voir que les resistance Rref1 et Rref2 seront égal. Par simplicité la valeur de 1K Ohm sera garder pour les deux résistance.

$$Vref = 5 * \frac{1 * 10}{2 * 10}$$

$$Vref = 2.5V$$

3.2 Préamplification

Le circuit reçoit le signal audio via la première prise jack. Il faut savoir que la tension appliqué dans ce cable est variable entre les cables et les ordinateurs. La chose importante a retenir est que celle-ci est très faible. Dans le cas de l'équipe, où c'est le PC de Hugo qui est utiliser un observe une tension moyenne de 150 mV. On doit alors passer par une étape de préamplification afin d'augmenter le signal d'entrée, et pouvoir travailler dessus correctement içi 2.5V de moyenne. Dans le montage, on observe deux préamplificateurs, un droite et un gauche mais il sont identiques. Le dimensionnement de un seul sera donc nécéssaire.

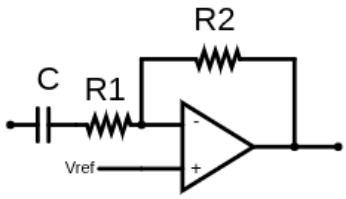

Figure 2: Schéma du Pré-amplificateur

Dans ce montage on observe un montage non inverseur précédé par un condensateur. Ici on sait que notre tension de sortie doit être maximum de 5V et de moyenne 2.5V, pour cela on va calculer le Gain nécéssaire et dimensionner nos résistance en fonction.
Quand au condensateur, il permet de faire plusieurs choses.Il permet de faire la barrière entre le signal d'entré avec une tension moyenne inconnue et la tension moyenne de l'AOP qui est a 2.5V depart sont alimentation avec Vref. Cela permet d'éviter les conflits entres ces deux zones car sans cela l'AOP qui applique sont gain au alentour de 2.5V va certe amplifié mais la partie jack sera pas amplifier correctement voir pas dutout. Pour le dimensionner, on peut dire que il fait office de filtre passe haut avec la résistance qui le suit. Cela permet en plus d'éliminer grâce a sa fréquence de coupure les fréquences trop basses. Son dimensionnement se fera de sorte a ce que la fréquence de coupure soit a 20hz la fréquence minimum écoutable pour un humain.
Formule AOP non inverseur : $Vs = (1 + \frac{Rpre1}{Rpre0}) * Ve$

Formule du filtre passe haut: $\frac{1}{2\pi * Rpre0 * Cpre0}$

Pour ce qui est du gain il faut faire le rapport de 150mv et 2.5V

$$\frac{Vs}{Ve} = \frac{5}{0.15} = 16.6$$

Avec la serie E6 et notre formule on trouve:

$$\frac{Vs}{Ve} = (1 + \frac{75 * 10}{4.7 * 10}) = 16.95$$

Grâce a cela on trouve pour Rpre0 et Rpre1 des valeurs de 75K ohm et 4.7K ohm, passons au dimensionnement du condensateur: $20 = \frac{1}{2\pi * 4.7 * 10 * C}$

$C = \frac{1}{2\pi * 4.7 * 10 * 16.95} = 1.7 * e - 6$

Avec la serie de composant, la valeur qui se rapproche le plus est 2.2uF ce qui nous donne un un fréquence de coupure a 15.40Hz

3.3 Mixage Gauche/Droite

La partie mixage sert a décider si on veux le côter gauche ou droite de notre signal audio. Dans notre circuit on retrouve un potentiomètre, if fonctionne comme 2 résistances variables reliés a Vref, sa valeur maximum est 10K Ohm.On retrouve aussi un montage sommateur un peu particulier.Sont dimensionnement se fera pas par pas afin de ne pas perdre le lecteur.
Comme pour les préamplificateur, on peut faire l'étude sur un coté du circuit car entre la gauche et la droite rien ne change. Pour simplifier les calculs le circuit ne sera pas réhausser avec Vref mais a la masse Voici notre circuit:

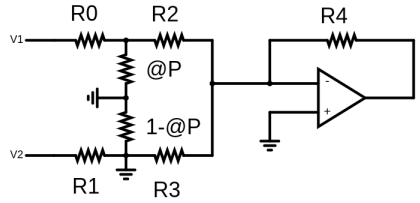

Figure 3: Schéma du Mixage

Et voici comment on va le l'interpréter:

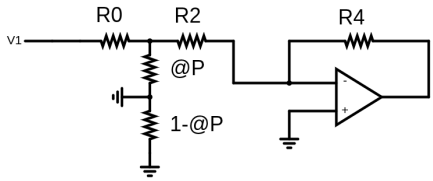

Figure 4: Schéma interprétatif du Mixage

On suppose que $V_2 = 0$, et que le potentiomètre délivre l'intégralité de sa bande passante sur V_1. Nous allons effectuer deux calculs selon la loi de Millman, un premier entre R_0, R_2 et αP, noté @P, et un second au niveau de la borne inverseuse du circuit.

Premier Millman sur V^- :
On note V_a la tension aux bornes de R_2.

$$\frac{V_a}{R_2} + \frac{V_s}{R_4} = \frac{V_a R_4 + V_s R_2}{R_2 + R_4}$$

Deuxième Millman entre R_0, R_2 et αP pour calculer V_a :

$$V_a = \frac{\frac{V_L}{R_0} + \frac{0}{\alpha P} + \frac{V^-}{R_2}}{\frac{1}{R_0} + \frac{1}{\alpha P} + \frac{1}{R_2}}$$
$$= \frac{V_L \alpha P R_2 + V^- R_0 \alpha P}{\alpha P R_2 + R_0 R_2 + R_0 \alpha P}$$

$$V^- = \frac{V_a R_4 + V_s R_2}{R_2 + R_4}$$
$$= \left(\frac{V_L \alpha P R_2 + V^- R_0 \alpha P}{\alpha P R_2 + R_0 R_2 + R_0 \alpha P} \right) \times \frac{R_4}{R_2 + R_4} + \frac{V_s R_2}{R_2 + R_4}$$
$$= \frac{V_L R_2}{R_0 + R_2 + \frac{R_0 R_2}{\alpha P}} \times \frac{R_4}{R_2 + R_4} + \frac{V_s R_2}{R_2 + R_4} = 0$$
$$V_L \times \frac{R2}{R_0 + R_2 + \frac{R_0 R_2}{\alpha P}} \times \frac{R_4}{R_2 + R_4} = -\frac{V_s R_2}{R_2 + R_4}$$
$$\leftrightarrow V_L \times \frac{R2}{R_0 + R_2 + \frac{R_0 R_2}{\alpha P}} \times \frac{R_4}{R_2 + R_4} \times \frac{R_2 + R_4}{R_2} = -V_s$$

$$V_L \times \frac{R2}{R_0 + R_2 + \frac{R_0 R_2}{\alpha P}} \times \frac{R_4}{R_2 + R_4} \times \frac{R_2 + R_4}{R_2} = -V_s$$
$$\leftrightarrow \frac{R_4}{R_0 + R_2 + \frac{R_0 R_2}{\alpha P}} = -\frac{V_s}{V_L}$$

On trouve alors que le rapport entre la sortie et l'entrée de l'AOP est

$$-\frac{V_s}{V_L} = \frac{R_4}{R_0 + R_2 + \frac{R_0 R_2}{\alpha P}}$$

Pour le dimensionnement il faut trouver des valeurs de R_0,R_2 et R_4 qui donnerons les valeurs :

$$1 \text{ pour } \alpha P = 0 \tag{1}$$
$$0.5 \text{ pour } \alpha P = 0.5 \tag{2}$$
$$1 \text{ pour } \alpha P = 0 \tag{3}$$

Après des tests sur la simulation de M.PIROG l'équipe a conclu que des valeurs de 47K, 68K et 470K pour respectivement R_0,R_2 et R_4 étaient des valeurs prométeuse. Ces valeurs s'appliquent donc aussi respectivement pour R_1,R_3.

3.4 Filtre Passe Haut / Bas

Le début de l'étude des filtres commencera par les deux filtres passe haut et passe bas. Pour leur dimmensionnement, l'élément rechercher est la fréquence de coupure a -3dB. Pour calculer cette fréquence on utilise la formule

$$F_{coupure} = \frac{1}{2\pi RC}$$

3.4.1 Filtre Passe Haut

Commençons par le filtre passe haut, son rôle est de ne laisser passer que les hautes fréquences aux dessus de 4KHz sa fréquence de coupure avec un gain de -3dB . Le montage se contruit de cette façon:

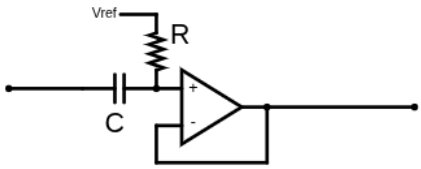

Figure 5: Filtres Passe Haut

Pour dimensionner ce filtre, il faut donc faire

$$\frac{1}{2\pi RC} = 4 \times 10^3 => RC = \frac{1}{2\pi \times 4 \times 10^3}$$

Toujour avec la serie E6, l'équipe choisit une valeur de résistance de 820 Ohm. On trouve alors une valeur de condensateur à 47n Farad, avec à 4KHz un gain de -3,16Db.

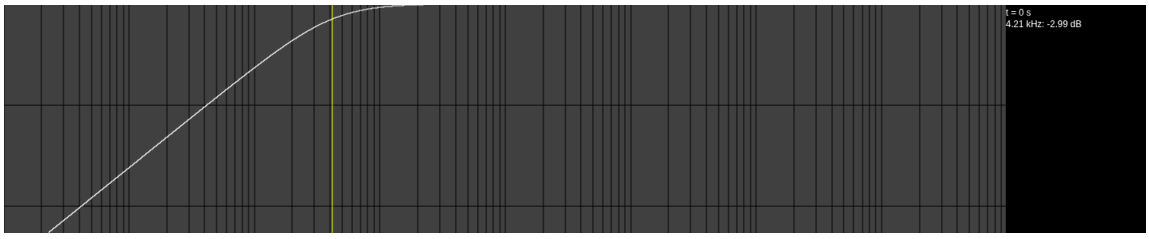

Figure 6: Fréquence de coupure PH

3.4.2 Filtre Passe Bas

Le filtre passe bas, son rôle est de ne laisser passer que les basses fréquences en dessous de 250Hz sa fréquence de coupure avec un gain de -3dB . Le montage s'effectue de cette façon:

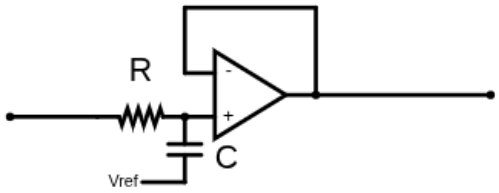

Figure 7: Filtre Passe Bas

En remaniant l'expréssion de la fréquence de coupure on trouve :

$$RC = \frac{1}{2\pi \times 250}$$

L'équipe a choisit de se basé sur une résistance de 33K Ohm, avec ceci on trouve une valeur de C a 22n Farad. En refaisant notre calcul:

$$\frac{1}{2\pi \times 33 \times 10^3 \times 22 \times 10^{-9}} = 219Hz$$

(La valeur qui se rapproche le plus de 250Hz)

En passant ces valeurs au simulateur, on observe effectivement vers 242Hz un gain de -3Db ce que l'on recherche exactement.

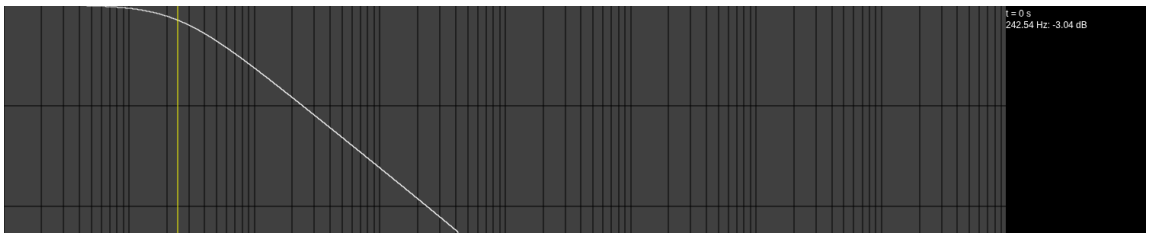

Figure 8: Fréquence de coupure PB

3.5 Filtres passes bandes

Après avoir dimensionner nos deux premier filtres qui s'occupe de laisser passer nos extrémums de fréquences, il faut s'occuper des valeurs intermédiare. On utilisera deux filtres passes bandes, un pour les bas médiums et un pour les hauts médiums.A savoir que, le cahier des charges demande à ce qu'il n'y ait aucun gain .Ils se contruisent de cette manière :

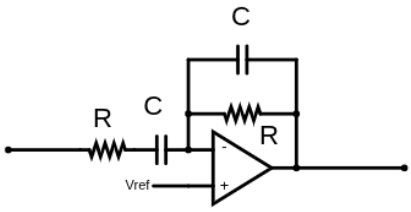

Figure 9: Schéma du Passe bande

Et pour les dimensionner on utilisera la formule:

$$f_0 = \frac{1}{2\pi RC}$$

3.5.1 Filtre Passe bande Bas

Ce filtre doit créer une bande entre 250Hz et 1KHz. Lorsque on applique la formule on trouve:

$$Pour f_0 = 250Hz et f_1 = 1KHz :$$

$$RC_1 = \frac{1}{2\pi f_0} = \frac{1}{500\pi} = 6.367 \times 10^{-4}$$
$$RC_2 = \frac{1}{2\pi f_1} = \frac{1}{2000\pi} = 1.59 \times 10^{-4}$$

On pose $R = 6.8K\Omega$:

$$6.8 \times 10^3 \times C_1 = 6,367 \times 10^{-4}$$
$$<=> C_1 = \frac{6,367 \times 10^{-4}}{6.8 \times 10^3} \approx 10 \times 10^{-8}$$

$$6.8 \times 10^3 \times C_2 = 1.59 \times 10^{-4}$$
$$<=> C_2 = \frac{1.59 \times 10^{-4}}{6.8 \times 10^3} \approx 2.52 \times 10^{-8}$$

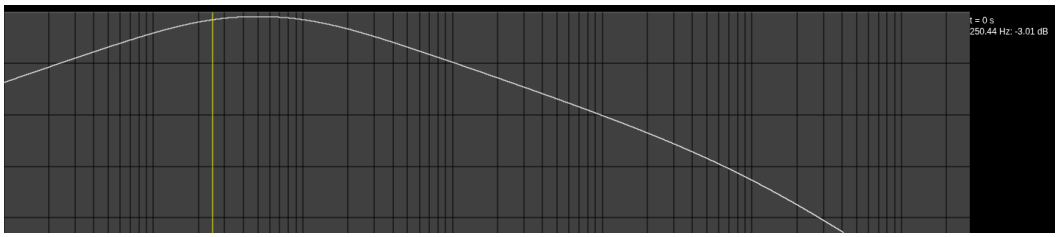

Figure 10: Diagramme de Bode avec fréquence de coupure à 250Hz

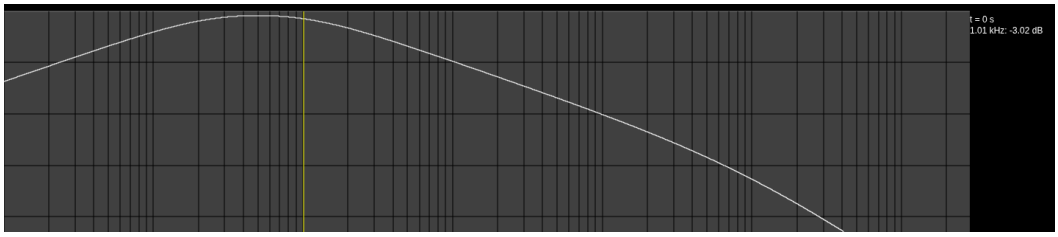

Figure 11: Diagramme de Bode avec fréquence de coupure à 1KHz

On trouve alors $C_1 = 100n$ Farad et $C_2 = 2.52n$ Farad qui deviendrons 22n Farad avec la serie E6 car elle est donne une valeur plus proche de f_1 plutôt que 32n Farad.

Ce qui sur le diagramme de Bode donne une première fréquence de coupure a 250 et une deuxième fréquence de coupure à 1KHz.

3.5.2 Filtre Passe bande Haut

Ce filtre doit créer une bande entre 1KHZ et 4KHz. Lorsque on applique la formule on trouve:

$$\text{Pour } f_0 = 1\text{KHz et } f_1 = 4KHz :$$

$$RC_1 = \frac{1}{2\pi f_0} = \frac{1}{2000\pi} = 1.59 \times 10^{-4}$$
$$RC_2 = \frac{1}{2\pi f_1} = \frac{1}{8000\pi} = 3.978 \times 10^{-5}$$

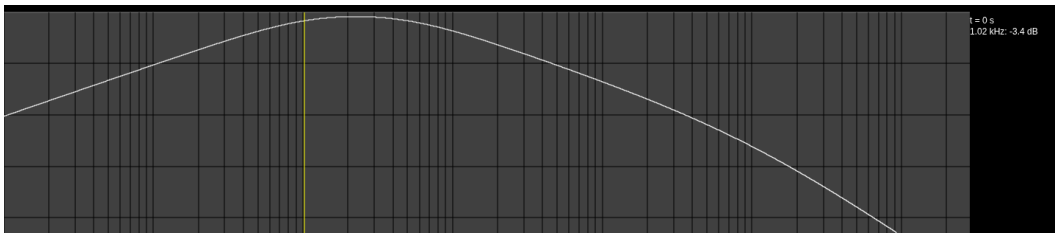

Figure 12: Diagramme de Bode avec fréquence de coupure à 1.1KHz

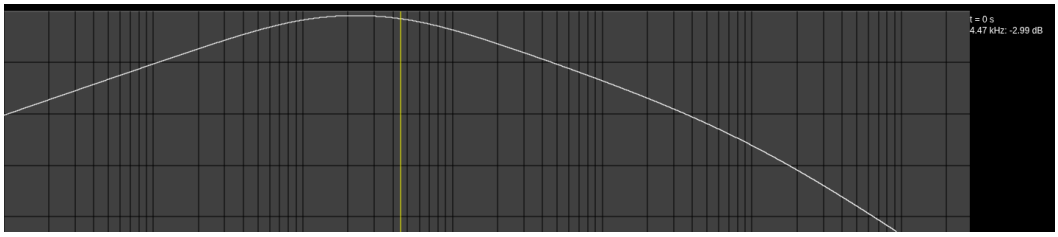

Figure 13: Diagramme de Bode avec fréquence de coupure à 4.5KHz

On pose alors $R = 1.5K\Omega$:

$$1.5 \times 10^3 \times C_1 = 1.59 \times 10^{-4}$$
$$<=> C_1 = \frac{1.59 \times 10^{-4}}{1.5 \times 10^3} \approx 10 \times 10^{-8}$$

$$1.5 \times 10^3 \times C_2 = 3.978 \times 10^{-5}$$
$$<=> C_2 = \frac{3.978 \times 10^{-5}}{1.5 \times 10^3} \approx 26 \times 10^{-9}$$

On trouve alors des valeurs similaire que sur le premier passe bande on mettra aussi le deuxième condensateur à 22nF.

Ce qui sur le diagramme de Bode donne une première fréquence de coupure a 1.1KHz et une deuxième fréquence de coupure à 4.5KHz ce qui n'est pas génant car on peut gérer l'exédant via le programme du PICKIT.

3.6 Enveloppe du signal

Un des enjeux du projet est que le microcontroleur doit afficher via la matrice LED les différents niveaux des 4 filtres. Pour lui faciliter le travail et surtout rendre la visualisation de ces niveaux plus lisible, il faut envellopper les signaux. C'est a dire lisser lorsque ceux-ci sont en phase déscendante. Pour cela on utilise un circuit d'envellope:

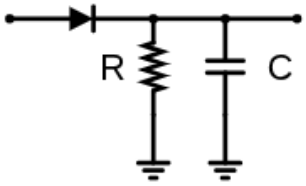

Figure 14: Schéma du circuit enveloppe
black

Pour dimensionner ces envellopes on peut dire qu'elles font office de filtres avec une petite particularité.Elles possède une diode qui permet de ne laisser passer que les alternances positives, en claire lorsque le signal est sur un front ascendant, le condensateur se charges puis lorsque le signal passe en front descendant la diode viens faire office de barrage. Le condensateur peut lui se décharger vers le microcontroleur ce qui lisse le signal jusqu'au prochain front ascendant dépassant la tension actuelle dans le circuit.Après consultation avec M.Pirog, il faut que ce temps de réponse soit au alentoure de 20ms. On vas alors utiliser et remanier la formule de la fréquence de coupure noté :

$$f_{coupure} = \frac{1}{2\pi RC}$$

En la remaniant on obtient:

$$RC = \frac{1}{2\pi \times f_{coupure}}$$

Il suffit de faire ensuite l'analyse dimensionnelle pour mieux comprendre d'ou viens ce fameux temps:

$$f_{coupure} = s^{-1}$$

$$2\pi = 1$$

donc:

$$RC = \frac{1}{2\pi \times f_{coupure}} = \frac{1}{1 \times s^{-1}} = s$$

En faisant cette analyse, on retrouve bien que RC donnera des secondes.

On peut donc ensuite faire l'équation $RC = 20 * 10^{-3}$.Il faut une résistance

assez haute pour que le condensateur soit petit et mettent peut de temps a se charger. Une Valeur de 100K Ohm fera parfaitement l'affaire. Avec les valeurs E6 on trouve un condensateur à 220n Farad ce qui donne un temps de réponse à 22ms ce qui est largement acceptable.

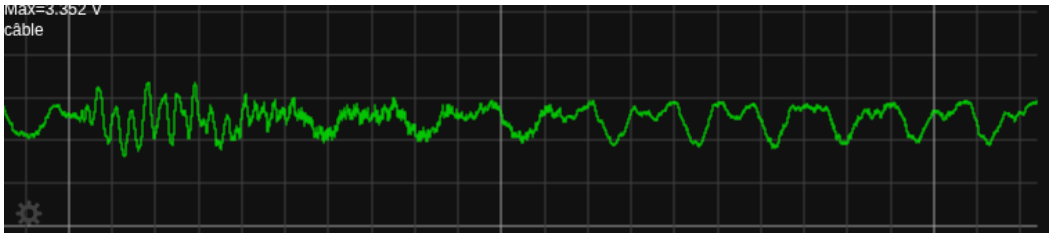

Figure 15: Exemple avant enveloppe

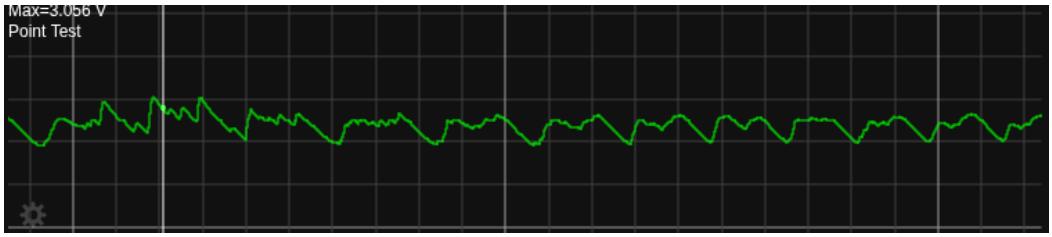

Figure 16: Exemple après enveloppe

4 Algorythmie et programmation

Pour programmer le pickit, il faut faire deux code, un en language C pour la conversion analogique vers numérique et le traitement des données et un en assembleur pour la partie commande des leds.
Voici les deux codes:

4.1 Main.c

```
    /* ------------------------------------------------------------
 * Fichier     :   main.c
 * Auteur(s)   :
 * Description :
 * ------------------------------------------------------------ */

#include <xc.h>

// Configuration materielle du PIC :
#pragma config FEXTOSC = OFF           // Pas de source d'horloge externe
#pragma config RSTOSC = HFINTOSC_64MHZ // Horloge interne de 64 MHz
#pragma config WDTE = OFF              // D?sactiver le watchdog

#define _XTAL_FREQ 64000000 // Frequence d'horloge - necessaire aux macros de delay (_delay(

// D?finition des masques, macros, etc. :
// TODO


// D?claration de fonctions et variables globales permettant au code C et ? l'asm de les par
// Une m?me fonction ou variable c?t? asm est pr?fix?e par un underscore, et ne l'est pas c?
// Avec ce formalisme, elles sont utilisables de fa?on intercangeable et transparente :
// | ---- asm ----- | ------------- C ----------------- |
// | _TX_64LEDS  <--|--> void TX_64LEDS(void)           |
// | _pC         <--|--> volatile const char * pC       |
// | _LED_MATRIX <--|--> volatile char LED_MATRIX [256] |

// D?finition des fonctions relatives ? la matrice de LEDs:
extern void TX_64LEDS(void); // Fonction d?finie dans tx.asm ; Fonction permettant d'envoyer

// D?finition des constantes / variables relatives ? la matrice de LEDs :
volatile char LED_MATRIX [256] ; // Definition d'une matrice de 64 x 4 octets contenant les
volatile const char * pC = LED_MATRIX; // Pointeur vers LED_MATRIX


char ad_read() {
    ADCON0bits.ADGO = 1;             // Démarre la conversion
```

```
    while(ADCON0bits.ADGO == 1) {}  // Attend la fin
    return ADRESH;                  // Retourne les 8 bits MSB (valeur entre 0 et 255)
}


void get_color(unsigned char n, unsigned char color[4]) {
    // Renvoie une couleur différente selon la valeur de `n`
    if (n <= 3) {
        color[0] = 0;
        color[1] = 16;   // Vert
        color[2] = 0;
        color[3] = 0;
    } else if (n <= 5) {
        color[0] = 16;   // Orange (rouge + un peu de vert)
        color[1] = 8;
        color[2] = 0;
        color[3] = 0;
    } else {
        color[0] = 16;   // Rouge
        color[1] = 0;
        color[2] = 0;
        color[3] = 0;
    }
}


void FILL_LED_MATRIX(unsigned char levels[4]) {
    for (int i = 0; i < 256; i++) {
        LED_MATRIX[i] = 0; // Réinitialise toute la matrice
    }

    for (int level = 0; level < 4; level++) {
        unsigned char l = levels[level];
        unsigned char color[4];
        get_color(l, color); // Détermine la couleur pour ce niveau

        for (int led_index = 0; led_index < l + 1; led_index++) {
            int led_index_adjacent = led_index + 8;
            for(int j = 0; j < 4; j++) {
                LED_MATRIX[led_index + j * level] = color[j];          // LED principale
                LED_MATRIX[led_index_adjacent + j * level] = color[j]; // LED adjacente
            }
        }
    }
}
```

```
// - Fonction main ------------------------------------------------------------------
void main(void) {
    // --- Configuration des entrées analogiques (RA2 à RA5) ---
    TRISAbits.TRISA2 = 1; // Entrée
    TRISAbits.TRISA3 = 1;
    TRISAbits.TRISA4 = 1;
    TRISAbits.TRISA5 = 1;

    ANSELAbits.ANSELA2 = 1; // Entrée analogique
    ANSELAbits.ANSELA3 = 1;
    ANSELAbits.ANSELA4 = 1;
    ANSELAbits.ANSELA5 = 1;

    // --- Configuration ADC ---
    ADPCH = 0b000010;        // Sélection de la première entrée (RA2)
    ADCON0bits.ADON  = 1;    // Activation du module ADC
    ADCON0bits.ADCONT = 0;   // Conversion unique
    ADCON0bits.ADCS = 1;     // Horloge ADC
    ADCON0bits.ADFM = 0;     // Résultat justifié à gauche
    ADCON0bits.ADGO = 0;     // Conversion inactive

    // --- Configuration des ports de sortie ---
    TRISC &= 0x00;           // PORTC en sortie
    LATC = 0x00;

    TRISBbits.TRISB4 = 0;    // Sortie pour déclencher TX
    TRISBbits.TRISB5 = 0;

    // --- Variables de mesure ---
    char bas, bande_bas, bande_haut, haut;

    // --- Boucle principale ---
    while(1) {
        LATBbits.LATB4 = 1; // Signalisation ou déclenchement
        //(à documenter selon schéma matériel)

        // Lecture des 4 capteurs analogiques
        bande_bas = ad_read();
        ADPCH = 0b000011;
        bande_haut = ad_read();
        ADPCH = 0b000100;
        haut = ad_read();
        ADPCH = 0b000101;
        bas = ad_read();
        ADPCH = 0b000010;
```

```
        // Calcul des niveaux sur 4 bandes (0 à 3)
        unsigned char levels[4];
        levels[0] = (bas - 128 < 0 ? 0 : bas - 128) / 32;
        levels[1] = (bande_bas - 128 < 0 ? 0 : bande_bas - 128) / 32;
        levels[2] = (haut - 128 < 0 ? 0 : haut - 128) / 32;
        levels[3] = (bande_haut - 128 < 0 ? 0 : bande_haut - 128) / 32;

        // Appel à la fonction propre
        FILL_LED_MATRIX(levels);

        // Envoi des données vers les LEDs
        TX_64LEDS();

        __delay_ms(50); // Petite pause pour éviter un
        //rafraîchissement trop rapide
    }
}
```

4.2 Tx.asm

```
    #include <xc.inc>

; When assembly code is placed in a psect,
//it can be manipulated as a
; whole by the linker and placed in memory.
psect   txfunc,local,class=CODE,reloc=2 ;
//PIC18's should have a reloc (alignment) flag of
;2 for any psect which contains executable code.

; -----------------------------------------------------------------
; GLOBALS
;
; DÃ©claration de fonctions et variables globales
;permettant au code C et Ã  l'asm de les partager
; Une mÃªme fonction ou variable cÃŽtÃ© asm est prÃ©fixÃ©e
;par un underscore, et ne l'est pas cÃŽtÃ© C
; Avec ce formalisme, elles sont utilisables de faÃ§on intercangeable
;et transparente :
; | ---- asm ----- | ------------- C ----------------- |
; | _TX_64LEDS  <--|--> void TX_64LEDS(void)           |
; | _pC         <--|--> volatile const char * pC       |
; | _LED_MATRIX <--|--> volatile char LED_MATRIX [256] |

; Fonction globales
global _TX_64LEDS ; Fonction dÃ©finie dans tx.asm ; Fonction permettant
```

```
;d'envoyer la commande pour piloter les 64 LEDs,
;telle que dÃ©crite dans LED_MATRIX

; Constantes/variables globales
global _pC         ; Constante dÃ©finie dans main.c
; Pointeur vers LED_MATRIX
global _LED_MATRIX ; Variable  dÃ©finie dans main.c
; Tableau (256 octets = 64 x 4)
;des composantes RGBW de la matrice LED (1 octet/couleur/LED)


; ------------------------------------------------------------------------------
; Macros d'envoi de bits pour la communication avec des LEDs adressables
; ------------------------------------------------------------------------------
; Envoie un bit '1' :
; Temps à l'état haut ? 750 ns (12 NOPs)
; Temps à l'état bas  ? 250 ns (4 NOPs)
SEND_1 MACRO
    BSF LATB, 5        ; Met la broche RB5 à 1 (HIGH)
    NOP                ; Attente (~62.5 ns à 64 MHz)
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    BCF LATB, 5        ; Met la broche RB5 à 0 (LOW)
    NOP
    NOP
    NOP
    NOP
ENDM

; Envoie un bit '0' :
; Temps à l'état haut ? 250 ns (4 NOPs)
; Temps à l'état bas  ? 560 ns (9 NOPs)
SEND_0 MACRO
    BSF LATB, 5        ; Met la broche RB5 à 1 (HIGH)
    NOP
    NOP
    NOP
```

```
    NOP
    BCF LATB, 5        ; Met la broche RB5 à 0 (LOW)
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
ENDM

; ------------------------------------------------------------------------------
; Routine SEND_8 : Envoie 8 bits (1 octet) lus depuis LED_MATRIX via FSR0
; ------------------------------------------------------------------------------
SEND_8:
    MOVF POSTINC0, 0, 0   ; Récupère l?octet pointé par FSR0, puis incrémente
    ;le pointeur

    ; Bit 7 (MSB)
    BTFSC WREG, 7         ; Teste si le bit 7 est à 1
    GOTO SEND_BIT_7
    SEND_0
    GOTO END_B7
    SEND_BIT_7:
 SEND_1
    END_B7:

    ; Bit 6
    BTFSC WREG, 6
    GOTO SEND_BIT_6
    SEND_0
    GOTO END_B6
    SEND_BIT_6:
 SEND_1
    END_B6:

    ; Bit 5
    BTFSC WREG, 5
    GOTO SEND_BIT_5
    SEND_0
    GOTO END_B5
    SEND_BIT_5:
 SEND_1
    END_B5:
```

```
   ; Bit 4
   BTFSC WREG, 4
   GOTO SEND_BIT_4
   SEND_0
   GOTO END_B4
   SEND_BIT_4:
SEND_1
   END_B4:

   ; Bit 3
   BTFSC WREG, 3
   GOTO SEND_BIT_3
   SEND_0
   GOTO END_B3
   SEND_BIT_3:
SEND_1
   END_B3:

   ; Bit 2
   BTFSC WREG, 2
   GOTO SEND_BIT_2
   SEND_0
   GOTO END_B2
   SEND_BIT_2:
SEND_1
   END_B2:

   ; Bit 1
   BTFSC WREG, 1
   GOTO SEND_BIT_1
   SEND_0
   GOTO END_B1
   SEND_BIT_1:
SEND_1
   END_B1:

   ; Bit 0 (LSB)
   BTFSC WREG, 0
   GOTO SEND_BIT_0
   SEND_0
   GOTO END_B0
   SEND_BIT_0:
SEND_1
   END_B0:
```

```
    RETURN              ; Retour à l'appelant

; ------------------------------------------------------------------------------
; Routine _TX_64LEDS : envoie les 256 octets de LED_MATRIX vers les LEDs
; ------------------------------------------------------------------------------
_TX_64LEDS:
    ; Chargement du pointeur vers LED_MATRIX dans FSR0
    MOVFF _pC + 0, WREG  ; Charge l'adresse basse (LSB) de LED_MATRIX
    MOVWF FSR0L, 0       ; Affecte à FSR0L (adresse indirecte)
    MOVFF _pC + 1, WREG  ; Charge l'adresse haute (MSB)
    MOVWF FSR0H, 0       ; Affecte à FSR0H

    BANKSEL LATB         ; Sélection du registre LATB (banque correcte)

    ; Boucle pour envoyer les 256 octets de LED_MATRIX (64 LEDs × 4 octets)
    REPT 256
        CALL SEND_8      ; Envoie 1 octet via la routine SEND_8
    ENDM
```

5 Conclusion et tests

Par manque de temps et d'organisation, aucunes photos des oscillogrammes n'ont été prises, une partie des tests se base donc sur de la déduction et du théorique.

Après que le dimensionnement et la programmation aient été fait, il était temps de tester, malheureusement aucun test concluant à été effectuer. Ce que l'on peut dire, le dimensionnement à été réaliser avec succès, l'équipe a pus sur les leds de la carte afficher le niveau sonore d'une des 4 composantes. Cependant lors du passage a la matrice leds cela ne fonctionnais pas, il semblerai que cela soit un problème d'au niveau du code assembleur.

Ce projet à permis àl'équipe de mieux comprendre les enjeux d'un bon dimensionnement, auquel cas le projet finis pourrait être incertain voir ne pas fonctionner complètement. Avec un minimum plus de temps ou un accès possible à la carte electronique, le groupe serait ravi de continuer de travailler dessus pour voir les possibilité que leur offres le projet.

5.1 Bonus

En petit bonus et pour vérifier via simulation la véracité de nos résultats, voici la simulation du circuit complet. Il suffit juste de double click sur No File et mettre un fichier MP3. Simulation du circuit

Et voici le circuit découper avec les valeurs que l'on peut copier/coller dans falstad pour vérifier les diagrammes de Bode: Circuit découper

5.2 Crédits

PAUL-LOUP VEREBELY :RAPPORTDIMENSIONNEMENT/SOUDURE

HUGO DUPRAT :PROGRAMMATION

NOE DIBLASI :DIMENSIONNEMENT/PROGRAMMATION