FULLER Thaïs, HAMARD Hermine, HÉRAULT Eloïse, RAUDE Julie

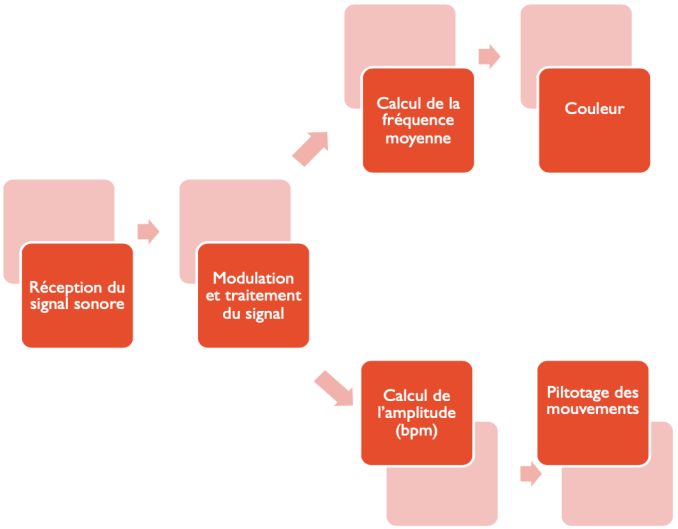
Rapport technique

Introduction

Beatbox & Light est un projet demandé par le LaserWave, association des lumières en soirée de l'école. Le but de ce projet est de rendre autonome le lighting d'une lyre.
Ainsi, nous avons cherché à produire un système qui prendrait directement la musique jouée en entrée et changerait en fonction du spectre sonore la couleur et le mouvement de la lumière. Nous avons aussi ajouté un bouton pour piloter l'option stroboscope.
Nous avons nommé notre projet Lightable, référence aux 'mixing table' pour la musique.

Découpage fonctionnel

Graphique synthétique des fonctions réalisées

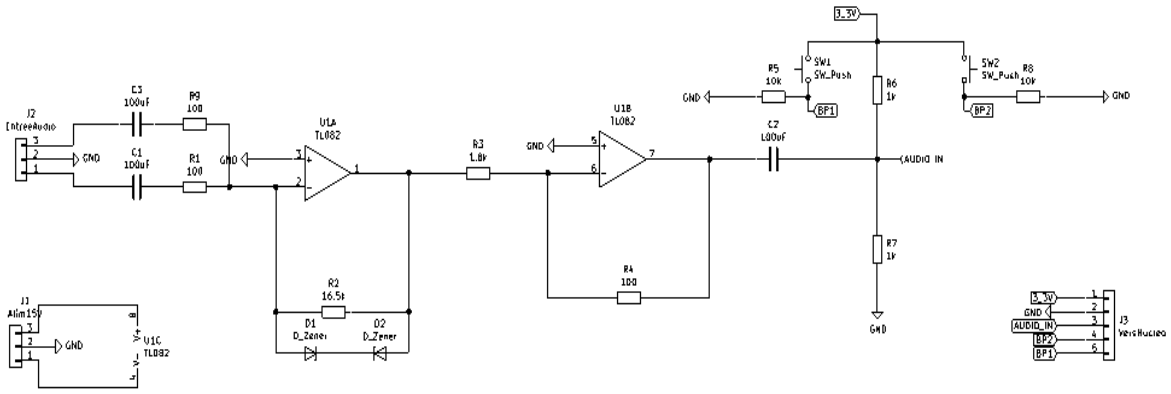


Descriptif des fonctionnalités

Acquisition du signal	Le système est directement branché à la source du signal (jack).
Modulation du signal	Modulation du signal par un circuit pour avoir une tension dans la plage [0, 3.3 V] en entrée de la carte nucleo dans laquelle se trouve le code.
Calcul de la FFT	La carte nucléo contient et exécute le code de calcul de la FFT et de pilotage de la lyre. On obtient en sortie le signal DMX à transmettre à la lyre.
Calcul de la fréquence moyenne de l'échantillon	A partir du spectre on calcule une fréquence moyenne pondérée par les amplitudes (cf code).
Calcul de la moyenne pondérée des amplitudes	S'il y a variation marquée de l'amplitude, la lyre arrête son tour et le recommence depuis là où elle en est et la couleur est actualisée (cf code).

Schémas électriques & Algorithmes

Schémas électriques



Nous avons utilisé le montage du thème 3 des TP de Céti pour avoir un signal qui puisse entrer dans la nucleo sans l'abîmer. En utilisant aussi le code de ce même TP mais de manière aménagée, nous obtenons le spectre du signal sonore sur lequel nous nous basons pour le pilotage des couleurs et avoir un signal.

La carte nucleo qui suit le montage est équipée du code détaillé ci-dessous.
Nous avons ajouté deux boutons poussoir protégés par des résistances de 10 kOhm, le premier sert à activer l'option stroboscope, le deuxième à l'éteindre.

Algorithmes

Comme précisé ci-dessus, nous avons utilisé et aménagé le code sur mbed qui a servi à obtenir le spectre sonore. Pour piloter les couleurs, nous sommes parties sur l'idée d'une fréquence moyenne pondérée par les amplitudes. Nous sommes restées sur ce choix. Nous avons ajouté un traitement du bruit, en coupant le traitement des amplitudes en dessous de 2% de l'amplitude maximale.
Nous voulions un autre paramètre pour piloter les mouvements et le changement de couleurs car lors des premiers tests, nous avions des tours complets et changements de couleurs qui n'étaient pas très dynamiques. Nous avons donc décidé d'utiliser la variation d'amplitude pour décider des changements de couleurs et au niveau du mouvement : arrêter un tour et en démarrer un autre, donnant l'illusion d'un mouvement aléatoire mais en rythme avec la musique.

La FFT calculée par le code est stockée sous forme de tableau qui fait correspondre les amplitudes aux fréquences du spectre, comme schématisé ci-dessous.

Fréquence	liste de fréquences
Amplitude	liste d'amplitudes

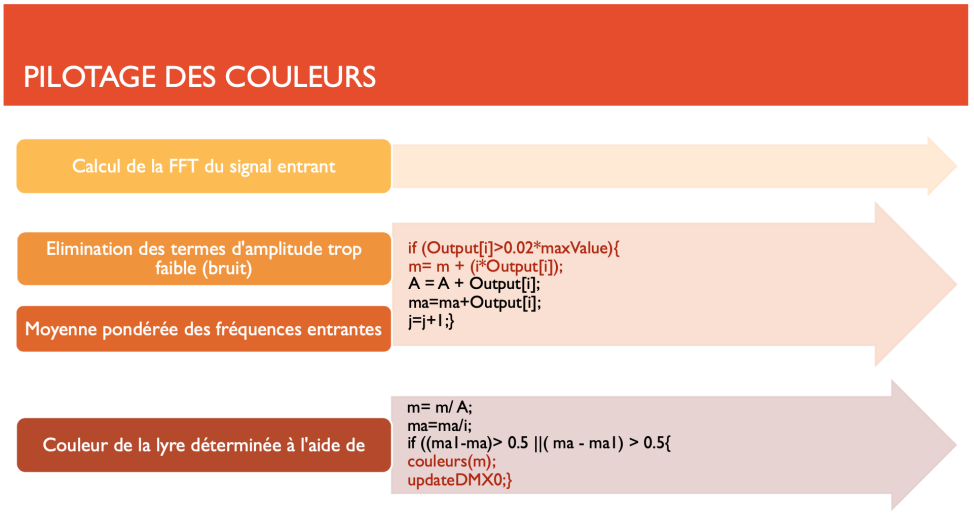
Le calcul de l'amplitude moyenne et de la fréquence moyenne pondérée par les amplitudes pouvait s'effectuer directement à partir du calcul de la FFT.

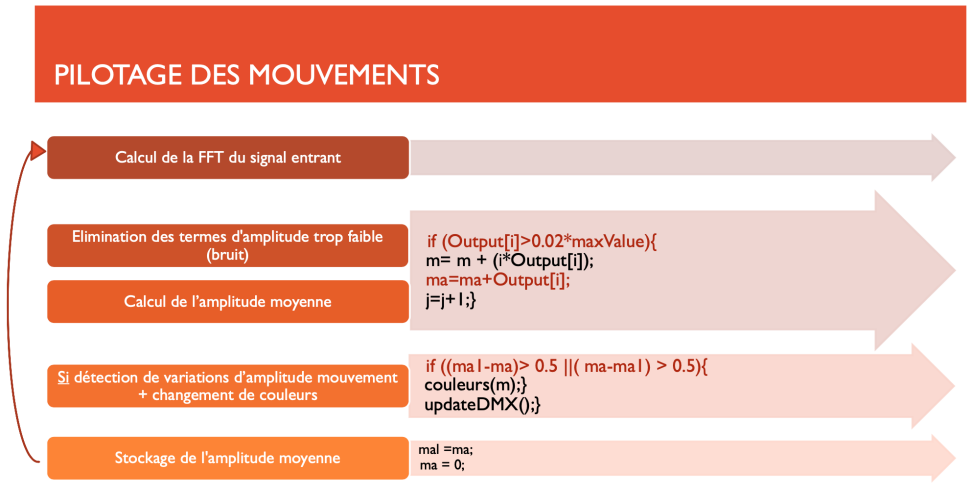
Le signal DMX qui permet de piloter la lyre est stocké sous forme de tableau à une ligne et 256 colonnes dont les correspondances sont détaillées dans l'annexe 1.

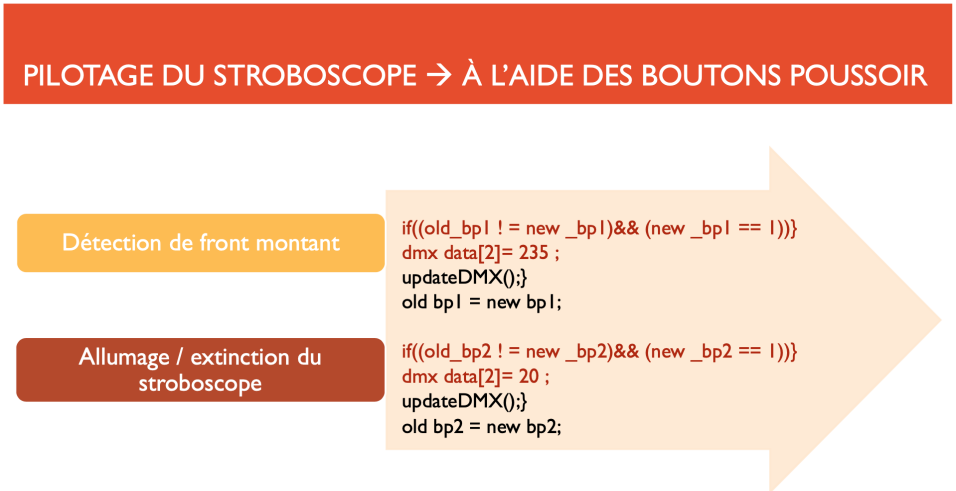
Nous avons choisi des plages de couleurs comme décrites dans le tableau ci-dessous.

Plages de fréquences en Hz et couleurs (code RVB)

< 5	[6, 10]	[11, 15]	[16, 20]	[21, 25]	[26, 27]	[28, 29]	[30, 32]	[33, 35]	> 36
violet	bleu	vert	cyan	rouge	orange	vert tropi	bleu	ambre	rose
253,63,146	0,0,255	52,170,6	0, 255, 255	255, 73,1	255,127,0	0,86,27	15,5,107	240,195,0	255,70,58







Tests de validation

Nous avons commencé par prendre possession du code en DMX avec des petits tests sans sons pour voir comment modifier les lumières, comment effectuer des mouvements, les couper pour avoir un rendu plus dynamique. Puis nous avons travaillé sur le code.

Nous avons obtenu des résultats très satisfaisants avec une lyre dynamique dont les couleurs changeaient harmonieusement sur un spectre plutôt large de musiques (en partant du classique et en arrivant sur de la techno).

La première chose que nous avons codée était le pilotage des couleurs, nous avons donc mis en entrée de la musique depuis Youtube pour les couleurs. Nous avons aménagé ensuite les plages de couleurs puisque certaines étaient mal choisies, des plages trop grandes ou trop petites, un RVB avec parfois des couleurs trop intenses.

Après des réglages et une fois satisfaites, nous sommes passées au traitement des amplitudes pour les mouvements. Nous avons procédé de la même manière.

Nous avons eu quelques soucis parfois avec une lyre éteinte : nous avons utilisé des print dans les boucles pour vérifier que nous avions des valeurs et non pas une suite de 0 ou de NaN et régler ensuite les soucis puisque nous savions où ils étaient.

Création du système embarqué

Une fois que le système avec le circuit électronique et la carte nucleo fonctionnait correctement, nous avons décidé de créer une carte électronique pour remplacer le circuit sur plaquette, très encombrant.
Pour imprimer la carte, nous avons d'abord dû tracer le circuit électronique sur le logiciel KiCad. Nous avons dû chercher tous les composants que nous avions utilisés (ALI double, résistance, diodes zener, etc.) sur le logiciel, puis les disposer de façon à ne pas avoir de croisement de pistes.
La carte a pu être imprimée au LEnsE à partir de ce document.
Nous avons ensuite soudé les composants dessus. Nous avons malheureusement fait des erreurs de débutantes en soudant la plupart des composants sur la masse,et n'avons jamais réussi à faire fonctionner cette carte malgré plusieurs tentatives de dessoudage/soudage.

Nous avons aussi produit une boîte pour ranger les cartes Nucleo et électronique, pour que le système soit facilement déplaçable.
Nous avons conçu une boîte en bois avec couvercle en plexiglas qui pouvait s'ouvrir afin d'accéder au système en cas de soucis.
Nous avons réalisé un schéma vectoriel de la carte en prévoyant les dimensions et les trous pour faire passer les câbles d'alimentation, la prise jack et les boutons poussoirs. Puis nous avons produit cette boîte à la découpe laser.

Planning & Analyse du travail d'équipe

Séances	Objectif de la séance	Bilan	Compétences
1	Reprise du code de la FFT sur la plaquette du thème 3. Ajout d'un moyennage en fréquence. Test avec en entrée un GBF puis un signal sonore de fréquence connue et enfin sur une musique. Découverte du pilotage des couleurs de la lyre.	Tout a été réalisé Problèmes lors de la mise en commun des 2 codes	Créer un projet sous MBEP Tester ma première application sur nucléo Récupérer des infos dans la documentation Piloter une LED Récupérer un signal analogique (pour faire la FFT) Faire un action à intervalle régulier (calcul de FFT)
2	Mise en commun des codes et débogage Changement du code, moyennage des fréquences de la FFT pondérées par leur amplitude, et suppression du bruit avec un pourcentage de la valeur max. Passage de deux cartes nucléo à une	Problèmes de mise en commun résolus. Début de la prise en main du pilotage en couleur et en mouvement. La fréquence moyenne calculée est stable. Réussite : la lyre change de couleur selon la fréquence moyenne calculée	Déboguer son programme Interface Série. Récupérer une information numérique (la FFT sert à définir les plages de fréquences) Connecter une source sonore Faire des actions à intervalle régulier (calcul de moyenne des fréquences et de l'amplitude) Caractériser un traitement numérique Supprimer une fréquence parasite (le bruit parasite) Mettre en place un asservissement numérique (gérer les couleurs et la direction de la lyre) Corriger un asservissement numérique
3	Synchroniser le changement de couleurs avec le changement de rythme de la musique. Pilotage des mouvements de la lyre.	Calcul et moyennage de l'amplitude des pics de la FFT. Changements des couleurs déterminé par les variations d'amplitude.	Faire une action après un événement (changement de couleur et direction après un changement de rythme et de fréquences) Régler la luminosité d'une LED
4	Définition des mouvements que la lyre devra effectuer selon les fréquences ou le rythme de la musique. Travail du code sur les mouvements de la lyre, et mise en commun avec le code précédent. Ajout d'un mode stroboscope	Ajout des commandes de direction dans la fonction couleur, pour chaque plage de fréquences. Ajustement des plages de couleurs pour avoir plus de changements. Couleur et direction bien synchronisées sur les	Contrôler un mouvement angulaire

	pour les passages rapides de musique.	musiques.	
5	Revoir le code et l'optimiser. Tout passer sur la même carte nucléo. Réaliser un bouton pour passer la lyre en mode stroboscope.	Refait fonctionner le système après de nombreuses difficultés de carte et de fils défectueux. On a pu tout passer sur la même carte nucléo. On a câblé deux boutons poussoirs permettant de passer en mode stroboscope, puis de revenir en mode normal..	Câbler un bouton poussoir Faire une action après un évènement
6	Faire un système embarqué pour le circuit, afin de pouvoir le transporter avec la carte nucléo . Tout mettre dans une boîte qu'on a juste à brancher, et une sortie jack.	Schéma du circuit sur KiCad et empreinte des composantes. Schéma de la boîte et découpe laser des différentes parties de la boîte.	
7	Souder les composants sur la nucléo, refaire les branchements, et finaliser la boîte. Faire les tests finaux.	Soudure des composants sur la carte électronique et derniers tests	
			- niveau 0 - niveau 1 - niveau 2 - niveau 3 - niveau 4

Tâches	Thaïs	Julie	Eloïse	Hermine
familiarisation avec le code DMX (la matrice)	x	x		
le circuit et sur le calcul de la fréquence moyenne			x	x
problèmes de mise en commun des codes		x	x	
livrables intermédiaires	x			x
conception de la boîte	x		x	
travail sur KiCad		x		
soudures en relais	x	x	x	x
dessoudage des composants et les courts-circuits				x
livrables finaux	x	x	x	x

Conclusion

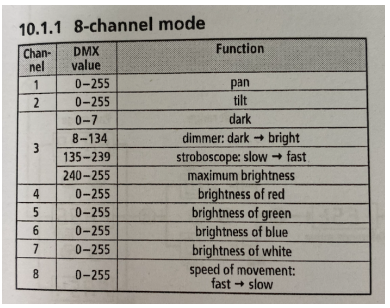
Nous avons passé un projet très agréable avec une bonne répartition des tâches. Malgré sa finalité. En effet, le projet fonctionnait très bien avant d'être passé sur système embarqué, nous pensons avoir fait trop de courts-circuits à la masse en soudant mal les composants, l'ALI a explosé et nous pensons que d'autres composants ont été abîmés dans ce processus mais nous n'avons pas eu le temps de tous les tester et les remplacer après avoir refait les soudures correctement.

Ce projet a mis à rude épreuve notre résilience et motivation et nous a appris à bien répartir les tâches en fonction de l'efficacité et des préférences de chacune. Niveau technique, il nous a bien entraînées à coder en C, à manipuler une carte nucléo, différentes entrées et sorties, aussi à trouver des paramètres dans le signal en entrée pour piloter la lyre. Nous avons aussi mis un pied dans la production d'un système embarqué avec la conception de la boîte découpée ensuite au laser et la conception de la plaque de cuivre, modélisée sur KiCad.

La fierté résultante de ce projet : avoir (presque) obtenu une light qui aurait pu être accrochée à la structure

Annexes

Annexe 1 : code DMX utilisé



10.1.1 8-channel mode

Channel	DMX value	Function
1	0–255	pan
2	0–255	tilt
3	0–7	dark
	8–134	dimmer: dark → bright
	135–239	stroboscope: slow → fast
	240–255	maximum brightness
4	0–255	brightness of red
5	0–255	brightness of green
6	0–255	brightness of blue
7	0–255	brightness of white
8	0–255	speed of movement: fast → slow

Annexe 2 : le code (en double colonne)

```
/************************************************************
******************/
/*  Test DMX512                                 */
/************************************************************
******************/
/*  LEnsE / Julien VILLEMEJANE     /   Institut
d'Optique Graduate School */
/************************************************************
******************/
/*  Brochage
     */
/*    TO COMPLETE
     */
/************************************************************
******************/
/*  Test réalisé sur Nucléo-L476RG
     */
/************************************************************
******************/

#include "mbed.h"
#include "arm_math.h"
#include "dsp.h"
#include "arm_common_tables.h"
#include "arm_const_structs.h"

#define SAMPLES   512
#define FFT_SIZE            SAMPLES / 2

Serial        debug_pc(USBTX, USBRX);

Serial        dmx(A0, A1);
DigitalOut  out_tx(D5);
DigitalOut  start(D4);          //envoie des données
DigitalOut  enableDMX(D6);
AnalogIn     CV_volume(PC_1);
AnalogIn     myADC(PB_0);

///DigitalIn my_bp(USER_BUTTON);
DigitalIn my_bp1(D7);
DigitalIn my_bp2(D5);

AnalogIn     variationR(PC_0);   //potentiomètres sur
la carte nucléo
AnalogIn     variationG(PC_2);
AnalogIn     variationB(PC_3);

float32_t Input[SAMPLES];
float32_t Output[FFT_SIZE];
bool          trig=0;
int           indice = 0;
double m;
double ma;
double ma1;
double A;
int j;

float maxValue;                 // Max FFT value is
stored here
uint32_t maxIndex;              // Index in Output array
where max value is

DigitalOut myled(LED1);
AnalogOut  myDAC(A2);
Serial        pc(USBTX, USBRX);
Ticker        timer;

// DMX
char dmx_data[SAMPLES] = {0};
char nb = 0;

void initDMX();
void updateDMX();

void couleurs(double m);
int moy();
void sample ();
```

```
// MOSI, MISO, SCK, CSN, CE, IRQ
// InterruptIn          my_bp(USER_BUTTON);


// Main

int main() {
    debug_pc.baud(9600);
    debug_pc.printf("Essai DMX512\r\n");
    initDMX();
    //dmx_data[2] = 20 ;
    int old_bp1, new_bp1=0;
    int old_bp2, new_bp2=0;

    while(1) {

    new_bp1 = my_bp1;
    new_bp2 = my_bp2;
    //debug_pc.printf("new_bp=%d\r\n",new_bp);
    //debug_pc.printf("old_bp=%d\r\n",old_bp);
            if((old_bp1 != new_bp1) && (new_bp1
== 1)){      // front montant
            dmx_data[2]= 235 ;
            updateDMX();
            debug_pc.printf("strobo\r\n");

            }
            old_bp1 = new_bp1;
            if((old_bp2 != new_bp2) && (new_bp2
== 1)){      // front montant
            dmx_data[2]= 20 ;
            updateDMX();
            debug_pc.printf("non\r\n");
            }
            old_bp2 = new_bp2;



    if(trig == 0){
            timer.detach();
            // Init the Complex FFT module, intFlag
= 0, doBitReverse = 1
            // NB using predefined
arm_cfft_sR_f32_lenXXX, in this case XXX is 256
            arm_cfft_f32(&arm_cfft_sR_f32_len256,
Input, 0, 1);

            // Complex Magniture Module put
results into Output(Half size of the Input)
            arm_cmplx_mag_f32(Input, Output,
FFT_SIZE);
            // Calculates maxValue and returns
corresponding value
            arm_max_f32(Output, FFT_SIZE/2,
&maxValue, &maxIndex);
            Output[0] = 0;
            m = 0;
            A=0;
            ma1 =ma;
            ma = 0;
            j=0;
            for(int i=0; i < FFT_SIZE / 2; i++){
            //myDAC=(Output[i]) * 0.9 ;   // Scale to
Max Value and scale to 90 / 100
            //wait_us(10);              //Each pulse of
10us is 25KHz/256 = 97.7Hz resolution


/*debug_pc.printf("Output=%lf\r\n",Output[i]);
            debug_pc.printf("maxValue =
%lf\r\n",maxValue);
            debug_pc.printf("maxValue =
%lf\r\n",0.02*maxValue);
            */

            if (Output[i]>0.02*maxValue){
            pc.printf("ok");
            m = m + (i*Output[i]);
            A = A + Output[i];
            ma=ma+Output[i];
            j=j+1;
            }
            }


            m = m / A ;
            debug_pc.printf("m=%lf\r\n",m);
            ma=ma/j ;
            if ((ma1-ma)> 0.5 ||( ma-ma1) > 0.5){
            debug_pc.printf("ok1\r\n");
            couleurs(m);
            updateDMX();
            }
            /*luminosite(ma);*/

            trig = 1;
            indice = 0;
            timer.attach_us(&sample,40);     //40us
25KHz sampling rate

        }

    }
}

void sample(){
    if(indice < SAMPLES){
```

```
    Input[indice] = myADC.read() - 0.5f;        //Real
part NB removing DC offset
    Input[indice + 1] = 0;                  //Imaginary
Part set to zero
    indice += 2;
    }
    else{ trig = 0; }
}
/*
int main() {
    float maxValue;             // Max FFT value is
stored here
    uint32_t maxIndex;        // Index in Output array
where max value is



    while(1) {
    if(trig == 0){
            timer.detach();
            // Init the Complex FFT module, intFlag
= 0, doBitReverse = 1
            //NB using predefined
arm_cfft_sR_f32_lenXXX, in this case XXX is 256
            arm_cfft_f32(&arm_cfft_sR_f32_len256,
Input, 0, 1);

            // Complex Magniture Module put
results into Output(Half size of the Input)
            arm_cmplx_mag_f32(Input, Output,
FFT_SIZE);
            Output[0] = 0;



            for(int i=0; i < FFT_SIZE / 2; i++){
            //myDAC=(Output[i]) * 0.9 ;   // Scale to
Max Value and scale to 90 / 100
            wait_us(10);                //Each pulse of
10us is 25KHz/256 = 97.7Hz resolution
            m = m + (Output[i]) ;
            debug_pc.printf("1-%d",m);
            }

            m = m/ (FFT_SIZE/2) ;
            debug_pc.printf("2-%d",m);
            trig = 1;
            indice = 0;
            timer.attach_us(&sample,40);     //40us
25KHz sampling rate

    }
    }
}
*/
```

```
/*
int moy() {

    while(1) {
            m = m / (FFT_SIZE/2) ;
            printf("3-%d",m);
            /*myDAC= moyenne ;
            wait(0.2);
            myDAC=0.0;
            pc.printf("MAX = %lf, %d \r\n",
maxValue, maxIndex);
            wait(0.2);
            trig = 1;
            indice = 0;
            timer.attach_us(&sample,40);     //40us
25KHz sampling rate
            couleurs(m);
    updateDMX();
    return m ;
}
}

*/

void initDMX(){
    // Initialisation DMX
    dmx.baud(250000);
    dmx.format (8, SerialBase::None, 2);
    enableDMX = 0;
    // Initialisation canaux DMX
    for(int k = 0; k < SAMPLES; k++){
    dmx_data[k] = 0;
    }
    updateDMX();
}


void couleurs(double m){

    if (  m<5) {        //violet
    dmx_data[3] = 253;
    dmx_data[4] = 63;
    dmx_data[5] = 146;
    dmx_data[0] = 255 ; // fait un tour complet sur
lui même
    dmx_data[1] = 255 ;
    }
    if (6<m && m<10) {  //bleu
    dmx_data[3] = 0;
    dmx_data[4] = 0;
    dmx_data[5] = 255;
    dmx_data[0] = 0 ; // fait un tour complet sur lui
même
    dmx_data[1] = 0 ;
```

```
    }
    if (11<m && m<15) {  //vert
    dmx_data[3] = 52;
    dmx_data[4] = 170;
    dmx_data[5] = 6;
    dmx_data[0] = 255 ; // fait un tour complet sur
lui même
    dmx_data[1] = 255 ;
    }
  if (16<m && m<20) {  //cyan
    dmx_data[3] = 0;
    dmx_data[4] = 255;
    dmx_data[5] = 255;
    dmx_data[0] = 0; // fait un tour complet sur lui
même
    dmx_data[1] = 0 ;
    }
    if (21<m && 25>m) {  //rouge
    dmx_data[3] = 255;
    dmx_data[4] = 73;
    dmx_data[5] = 1;
    dmx_data[0] = 255 ; // fait un tour complet sur
lui même
    dmx_data[1] = 255 ;
    }
    if (26<m && m<27) {  //orange
    dmx_data[3] = 255;
    dmx_data[4] = 127;
    dmx_data[5] = 0;
    dmx_data[0] = 0 ; // fait un tour complet sur lui
même
    dmx_data[1] = 0 ;
    }
    if (27<m && m<29) {  //vert tropi
    dmx_data[3] = 0;
    dmx_data[4] = 86;
    dmx_data[5] = 27;
    dmx_data[0] = 255 ; // fait un tour complet sur
lui même
    dmx_data[1] = 255 ;
    }
    if (30<m && m<32) {  //bleu
    dmx_data[3] = 15;
    dmx_data[4] = 5;
    dmx_data[5] = 107;
    dmx_data[0] = 0 ; // fait un tour complet sur lui
même
    dmx_data[1] = 0 ;
    }
    if ( 33<m && m<35) {  //ambre
    dmx_data[3] = 240;
    dmx_data[4] = 195;
    dmx_data[5] = 0;
    dmx_data[0] = 255 ; // fait un tour complet sur
lui même
    dmx_data[1] =255 ;
    }
    if (36<m ) {  //rose
    dmx_data[3] = 255;
    dmx_data[4] = 70;
    dmx_data[5] = 58;
    dmx_data[0] = 0 ; // fait un tour complet sur lui
même
    dmx_data[1] = 0 ;
    }



    }

/*
void luminosite(ma){
    if (ma<){
    dmx_data[2] = 5;
    }
    }*/

void updateDMX(){
    enableDMX = 1;
    start = 1;          // /start
    out_tx = 0;         // break
    wait_us(88);
    out_tx = 1;         // mb
    wait_us(8);
    out_tx = 0;         // break
    start = 0;
    dmx.putc(0);        // Start
    for(int i = 0; i < SAMPLES; i++){
            dmx.putc(dmx_data[i]);  // data
    }
    wait_us(23000); // time between frame
}
```