Fiche technique du projet DARC

Louis Wieczorek, Guillaume Nicque, Mathilde Urbain et Fabien Traynard
Institut d'Optique Graduate School – 1A – 01/06/2022

Nous attestons que ce traval est original, que nous citons en références toutes les sources utilisées, et qu'il ne comporte pas de plagiat.

1 Introduction

Ce rapport technique porte sur le projet DARC (Détection Automatique et Résistution de Lumière) sur lequel nous avons travaillé tout au long du deuxième semestre de notre première année à l'IOGS. Notre système consiste en effet à capter un signal lumineux coloré et le retransmettre fidélement sur un bandeau de LEDs. Nous avons donc travaillé sur cette idée pendant une séance pour arriver à délimiter quelles sont les difficultés majeures et comment nous comptons les lever.
Tout d'abord, l'oeil n'a pas le même spectre de réception qu'une photodiode. C'est l'un des premiers problèmes important : si on n'applique pas un traitement adéquat aux données captées par la photodiode, la couleur restituée risque de ne pas ressembler du tout à la couleur incidente. Ensuite, d'autres difficultées annexes ont aussi été prises en compte : réussir à trouver un capteur RVB qui retranscrive au mieux la couleur incidente grâce à des filtres en fait partie. Enfin, nous avions également comme ambition de miniaturiser notre système en le plaçant dans une boîte en bois/plastique (figure 1). Toutes ces difficultés constituent la problématique que nous nous sommes efforcés de résoudre durant l'élaboration de DARC.

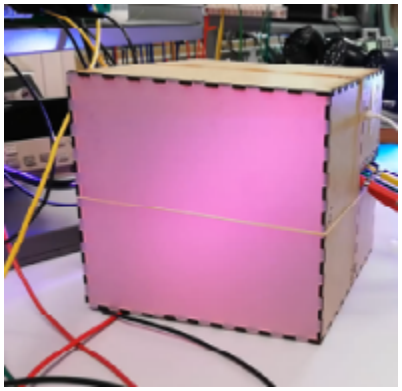

Figure 1: Photo de DARC miniaturisé

Nous traiterons ainsi par la suite plus en détail chaques difficultés rencontrées ainsi que les moyens mis en oeuvre pour les résoudre. On abordera également la prise en main d'un capteur photodiodes spécial qui utilise le protocole I2C et dont on s'est beaucoup servi lors de nos séances. Nous évoquerons les différents tests plus ou moins concluants effectués pour terminer sur une brève analyse de notre travail en équipe ainsi que des pistes d'amélioration pour les années futures.

2 Découpage fonctionnel

Après la définition des objectifs à atteindre durant ce semestres et des difficultés, nous avons réalisé un schéma fonctionnel global nous permettant de mieux visualiser les différentes étapes de notre projet et de nous répartir les tâches efficacement (figure 2).

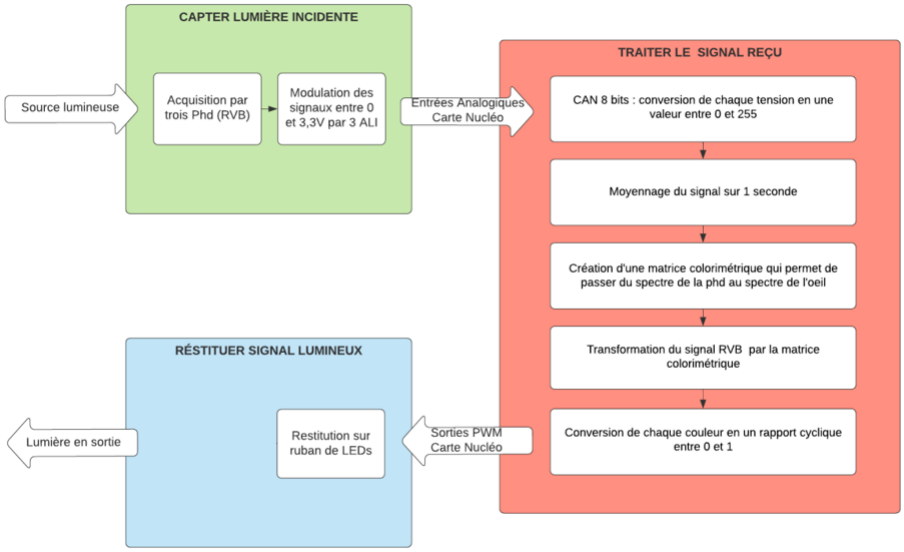

Figure 2: Schéma fonctionnel du système DARC

On peut diviser ce schéma en trois parties : la partie dans laquelle on capte la lumière colorée incidente, la partie dans laquelle on la traite sur Mbed et la partie dans laquelle on renvoie la lumière.

Tout d'abord, on capte le signal grâce à trois photodiodes placées derrière des filtres rouge, vert ou bleu (premier capteur étudié). On dispose ainsi de trois tensions qu'on doit maintenant amplifier pour qu'elles soient dans l'intervalle [0 V,3.3 V] afin que la carte nucléo puisse les lire correctement. On va donc avoir besoin d'un montage amplificateur (avec les résistances à déterminer suivant nos besoins et les caractéristiques de chaque composants).

Ensuite, on acquiert grâce à la carte nucléo les valeurs des tensions des trois photodiodes. On décide de moyenner ces tensions pour avoir des valeurs moins fluctantes et moins sensibles aux éclairages parasites par exemple. On crée ainsi un vecteur colonne (R V B) comprenant les trois valeurs de tensions. On mutliplie ce vecteur par un matrice colorimétrique calculée en amont qui transforme le vecteur correspondant au spectre de réception des photodiodes en un autre correspondant au spectre de l'oeil. On divise ensuite ce vecteur par la tension maximale délivrée par le circuit précédant pour trouver un pourcentage de couleurs RVB.

Enfin, on envoie simplement le pourcentage calculé dans le ruban de LEDs grâce aux ports PWD de la carte Nucléo pour délivrer la bonne couleur en sortie !

Ainsi, notre système sera capable de détection une couleur envoyée sur les photodiodes puis de la retransmettre fidèlement de l'autre côté grâce aux bandeaux de LEDs. Le système mettra aussi en évdience les nuances d'intensité entre les deux signaux. Surtout, il prendra en compte la sensibilité particulière de l'oeil et avec des temps de réponse relativement rapides pour pouvoir changer de couleur rapidement (le code et notamment le moyenne devra donc être particulièrement efficace).

3 Réalisation du prototype

Dans cette partie, nous allons traiter de la manière dont nous avons réalisé les différentes fonctionnalités demandées étapes par étapes

3.1 Le montage amplificateur

Tout d'abord, nous avons du capter le signal et amplifier les tensions trouvées pour les faire arriver dans l'intervalle [0 V,3.3 V]. Pour celà, on construit le schéma électrique suivant :

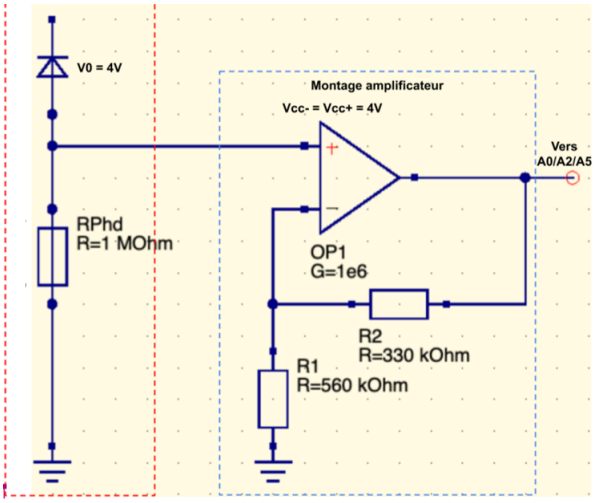

Figure 3: Schéma électrique d'acquisition de la couleur incidente

On a tout d'abord caractérisé notre photodiode à l'aide d'un montage simple de photodétéction. Ensuite, grâce à la documentation technique du composant on a pu estimer la tension maximale en sortie de la photodiode, cette tension étant d'environ 2V, on utilise alors un montage d'amplificateur non inverseur pour que notre plage de tension atteinte soit plutôt entre 0 et 3,3V. On peut rappeler la formule de transfert de l'amplificateur non inverseur :

$$Vs = Ve \times (1 + \frac{R2}{R1}) \tag{1}$$

Ce montage était le plus performant pour l'utilisation dont on avait besoin (ie une petite amplification sans contrainte très importante sur la fréquence). On l'a bien sûr construit en trois fois (pour la photodiode reponsable du rouge, celle du bleu et celle du vert).

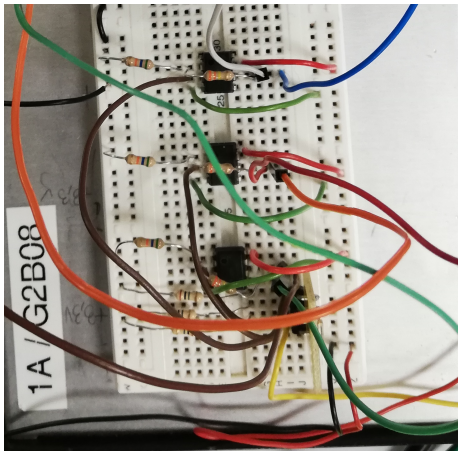

Figure 4: Montage des trois amplificateurs

3.2 Traitement du signal sur ordinateur

Dans un second temps, nous avons vu qu'on avait besoin de traiter le signal reçu en entrée de la carte nucléo. Nous allons nous baser ici sur un schéma des algorithmes utilisés les codes exacts étant en annexe.

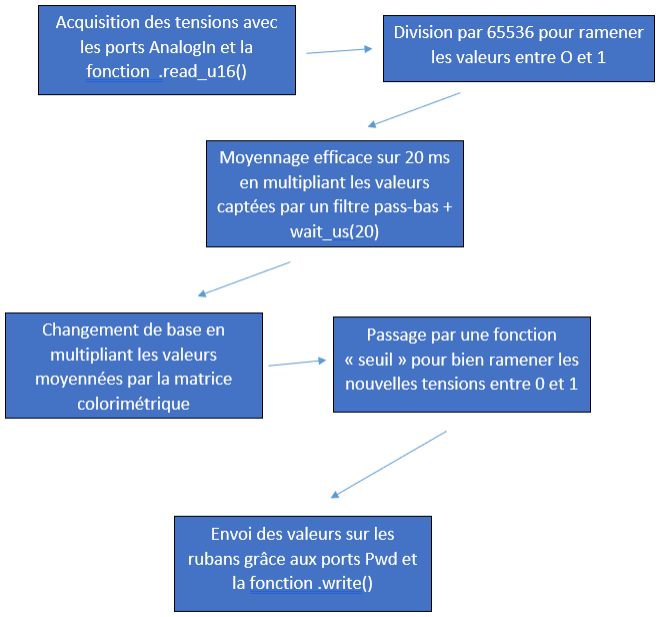

Figure 5: Schéma de l'algorithme utilisé

Nous pouvons nous arrêter un instant sur la construction de la matrice colorimétrique. En effet, cet aspect n'a pas vraiment été détaillé jusqu'à maintenant et constitue pourtant un point majeur. En réalité, on construit d'abord son inverse pour pouvoir calculer ensuite la matrice finale. Pour ce faire, on procède à des mesures manuelles. Dans un premier temps, on projette sur notre capteur une lumière que l'on voit parfaitement rouge puis on observe grâce à teraterm le pourcentage de lumière rouge, verte et bleue brut calculé grâce à la premiere partie de l'algorithme. On note ces valeurs respectivement dans les cases (1,1),(1,2) et (1,3) de notre matrice. On procède de même pour les couleurs bleues et vertes. On a ainsi la matrice qui relie les couleurs visualisées à l'oeil à celle captée par les photodiodes. Il suffit ensuite d'inverser la matrice en question pour avoir la matrice colorimétrique utilisée dans l'algorithme.
De manière pratique, nous avons simplement inversé la matrice à l'avance et noté comme variable globale (define ...) les valeurs de la matrice à utiliser par la suite.
0.5cm
L'autre partie intéressante de cet algorithme est la convolution par un passe-bas pour moyenner les tensions sur 20ms. C'est une méthode originale mais qui permet une très efficacité et donc un temps de réponse de notre système relativement court (ce qui constituait l'un des objectifs à atteindre).

Autre capteur Nous nous permettons ici une courte digression sur l'autre capteur que nous avons essayé d'utiliser (le TCS34725). Ce capteur utilise le protocole I2C pour échanger avec l'ordinateur (à brancher sur les ports SCL et SDA de la carte nucléo). Celà implique quelques modifications du code même si le principe reste le même. Nous avons d'ailleurs eu beaucoup de mal à comprendre comment fonctionnait ce protocole et même si nous n'avons pas pu utiliser ce capteur finalement, les codes de ce capteur sont aussi en annexe.

3.3 Montage du bandeau de LEDs

On réalise ce montage à l'aide de transistor BS170. La résistance Rgs est choisie assez grande pour limiter le courant dans le bandeau et ainsi empecher les LEDs de griller.

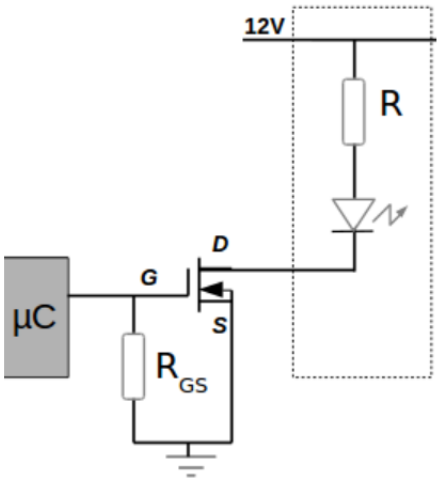

Figure 6: Schéma du montage du bandeau de leds (source : LEnsE)

Ce montage est inspiré du TP fait durant le semestre 5 dans lequel il était proposé d'utiliser des rubans de LEDs en annexe. C'est un montage très simple sur lequel on a décidé de brancher un unique ruban pour des raisons matérielles (nous n'en avions pas beaucoup) mais aussi car le plastique diffusant utilisé dans la construction de la boîte permettait une homogénéité de la lumière en sortie. Il n'était donc pas nécessaire d'en mettre un grand nombre pour avoir un résultat satisfaisant.

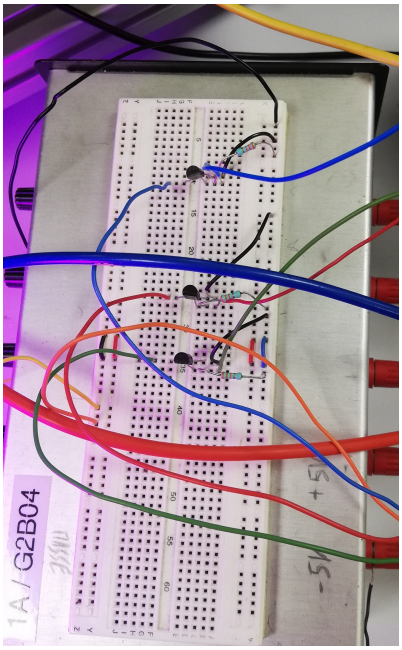

Figure 7: Montage du ruban de LEDs (ruban à gauche de la photo)

4 Validation et caractérisation du système final

4.1 L'amplification

Tout d'abord, il a fallu valider et caractériser le système amplificateur avant de le branche sur la carte nucléo pour être sûr que toutes les contraintes notament en tension étaient validées avant de brancher quoi que ce soit sur la carte nucléo. On a ainsi pu observer à l'oscilloscope les différentes tension tout d'abord en sortie du montage de photodétection pour évaluer la tension maximale qui rentrera dans le système. On a pour celà accolé à la photodiode un spot très lumineux qu'on a pris comme référence du maximum de puissance lumineuse qui serait amené à entrer dans le système. On a ensuite adapté la tension d'alimentation des AO et les valeurs des résistances en fonction de cette première observation.

Il a ensuite suffit de brancher l'oscilloscope en sortie du montage électrique pour tester si la saturation des AO fonctionnait bien et si l'amplification en fonction de la quantité de lumière incidente (ou encore de la distance du spot par rapport aux photodiodes) était cohérente. C'est d'ailleurs pendant ces tests qu'on a remarqué en premier l'instabilité du signal et donc la nécessité de le moyenner ensuite de manière informatique.

On a donc passé une séance et demie à examiner cette partie du montage qui était l'une des plus importantes pour la suite. L'amplification a d'ailleurs posé quelques problèmes nous obligeant à conduire de nombreux tests avec différents rapports de résistance (surtout effectués par Fabien et Mathilde).

4.2 Codage sur Mbed

Nous avions en parallèle de cette validation et caractérisation de l'amplificateur commencé à coder un algorithme naif (efficacité médiocre) pour moyenner le signal et afficher les valeurs sur TeraTerm. Il était important de commencer de cette manière pour pouvoir brancher le capteur et les amplis sur la carte nucléo et vérifier que les valeurs de signal RVB moyennées et en pourcentage étaient cohérentes. Après quelques debugages du code, il apparaissait des problèmes : tous les pourcentages étaient à O et il a fallu revenir à la caractérisation du système amplificateur pour comprendre que des fils se touchaient dans le montage. Une fois ces difficultés passées, on a pu rapidement obtenir des valeurs cohérentes. L'essentiel du travail suivant a été de rendre l'algorithme plus efficace, de commenter le code et de créer la matrice colorimétrique à utiliser pour rendre les couleurs cohérentes (surtout fait par Louis et Guillaume). Enfin, le débugage du code utilisant le protocole I2C a quant à lui été beaucoup

plus long puisqu'il était plus difficile pour nous de comprendre exactement toutes les fonctions du protocole. On avait donc fait un certain nombre d'erreur notament sur le codage des nombres en binaires qu'on a mis beaucoup de temps à comprendre. Mais au bout de quelques séances, les deux codes étaient valides et plutôt efficaces.

4.3 Ruban de LEDs

Cette partie a été de loin la plus simple de la conception de notre système puisqu'on avait déjà abordé les rubans en TP du premier semestre. On a donc très vite fait le montage, on l'a testé à l'oscilloscope pour voir si le pourcentage envoyé par la carte nucléo était bien retransmis en tension dans les trois branches RVB du montage. On a donc très vite pu brancher le ruban alimenté ici avec une tension de 11 V.

5 Planning de conception et difficultés rencontrées

Enfin, le planning de conception de DARC que nous avions imaginé au début a été bien sûr modifié par les aléas des réussites et échecs dans notre avancée. Nous aviosn au départ prévu qu'à la séance 2, on aurait terminé la partie amplification, qu'à la séance 3 on aurait reproduit le système avec le nouveau capteur TCS34725 commandé. Cependant tout ne s'est pas vraiment passé comme prévu et le planning final ressemble plutôt au tableau ci-dessous.

Séance 1	Conception du cahier des charges
Séance 2	Début de la partie amplification
Séance 3	Fin de la partie amplification
Séance 4	Prise en main du nouveau capteur/Matrice colorimétrique
Séance 5	Prise en main du protocole I2C/Création de la première boîte
Séance 6	Prise en main du protocole I2C/Premier système fonctionnel dans la boîte/Commande d'une nouvelle boîte pour le deuxième capteur
Séance 7	Essai d'implémentation du deuxième capteur avec batterie/Défaut de fonctionnement du deuxième capteur
Séance 8	Présentation de la première boîte

Nous avons en effet passé beaucoup de temps à comprendre et prendre en main le protocole I2C même si nous n'avons pas pu utiliser ce nouveau capteur avec sa nouvelle boîte dont l'avantage aurait été la petite taille et la portabilité grâce à l'implémantation d'une batterie 12V.

Ainsi, nous avons bien sûr rencontrés des difficultés. Le protocole I2C est l'une des plus grandes même si nous avions fini par réussir à comprendre son fonctionnement mais nous avons passé beaucoup de temps sur cette partie du système. Ensuite, c'est la prise en main d'un capteur inconnu qui a été le plus long pour nous. Même si la documentation technique associée au capteur était plutôt complète, nous avons dû souder des fils dessus, comprendre le fonctionnement des ports SDA et SCL, faire attention aux tensions et courants supportés par le capteur... Bien sûr notre erreur majeure aura été d'aller trop vite pour mettre le deuxième système (pourtant fonctionnel lors des tests) dans sa boîte. Notre précipitation nous a conduit à omettre un branchement à la masse qui aura fait, avec la batterie 12V, griller le capteur.

Ce projet nous aura aussi permis de mieux gérer le travail d'équipe en se repartissant les tâches efficacement. La création d'un Drive commun nous permettait de transmettre plutôt correctement les informations entre les différents mini-groupes de travail même si il manquait encore de la communication sur des sujets précis que seuls un ou deux d'entre nous comprenait correctement par manque de patience pour l'expliquer aux autres. C'est surtout sur ce point que se trouvent les pistes d'améliorations à notre sens.

6 Annexes

6.1 Code fonctionnel pour le système numéro 1

```
 /* mbed Microcontroller Library
 * Copyright (c) 2019 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */

#include "mbed.h"
#include "platform/mbed_thread.h"
#include "math.h"



// Blinking rate in milliseconds
#define BLINKING_RATE_MS        100

#define N 100
#define _N 0.01
//matrice inverse (colonne par colonne)
#define iM11 5.53e-3*255
#define iM21 -2.37e-3*255
#define iM31 -3.16e-4*255

#define iM12 -7.97e-4*255
#define iM22 1.62e-2*255
#define iM32 -3.76e-3*255

#define iM13 -1.14e-3*255
#define iM23 -5.86e-3*255
#define iM33 9.59e-3*255



InterruptIn bouton(USER_BUTTON);    //interruption par le bouton de la carte
AnalogIn e_B(A0);    //Entrée Bleu
AnalogIn e_R(A2);    //Entrée Rouge
AnalogIn e_V(A5);    //Entrée Vert
DigitalOut led_indic(LED1);    //led de la carte
PwmOut s_B(D6);    //Sortie Bleu
PwmOut s_V(D5);    //Sortie Vert
PwmOut s_R(D3);    //Sortie Rouge
Serial pc(USBTX, USBRX);

int n_cal=0;
double M[9]={1,0,0 ,0,1,0 ,0,0,1};
double _M[9]={iM11,iM12,iM13,iM21,iM22,iM23,iM31,iM32,iM33};
double R_ee,V_ee,B_ee;
double R_es,V_es,B_es;



void calibrage(void);
/*Enregistre dans la matrice M les valeurs brutes de la couleur captée
à l'appui du bouton intégré à la carte, puis met l'inverse de M dans _M (la matrice de passage
 de l'espace de couleur d'entrée à celui de sortie):
```

```
-Le premier appui est pour enregistrer la référence du rouge (la LED intégrée à la carte
 s'allume une fois pour 1/3 seconde).
-Le second est pour le vert (la LED s'allume 2 fois).
-Le troisième est pour le bleu (la LED s'allume 3 fois). La matrice inverse est ensuite calculée
 et la fonction se réinitialise. */

double seuil_lf(double x,double min,double max);
/*Prend en entrée un double x et renvoie sa valeur tronquée aux seuils min et max.
  x: double, égal à la sortie si min < x < max
  min: double, égal à la sortie si min >= x
  max: double, égal à la sortie si max <= x */

void inverser_M3lf(double A[9],double _A[9]);
/*Inverse la matrice A et stocke l'inverse dans _A.
A et _A sont des tableaux de doubles de taille 9. */

int main()
{
    bouton.fall(&calibrage);
    led_indic=0;
    pc.baud(9600);
    int meas_int;

    double R_capt,V_capt,B_capt;
    int i;

    s_R.period_ms(10);
    s_V.period_ms(10);
    s_B.period_ms(10);

    R_ee = 0;
    V_ee = 0;
    B_ee = 0;

    while(1)
    {
        for(i = 0; i<N;i++){
            meas_int = e_R.read_u16();    // sur 12 bits MSB
            R_capt = meas_int / 65536.0 ;    //Valeur captée rouge entre 0 et 1

            meas_int = e_V.read_u16();    // sur 12 bits MSB
            V_capt = meas_int / 65536.0 ;    //Valeur captée vert entre 0 et 1

            meas_int = e_B.read_u16();    // sur 12 bits MSB
            B_capt = meas_int / 65536.0 ;    //Valeur captée bleu entre 0 et 1

            //Valeurs dans l'espace d'entrée (valeur captée * filtre passe-bas):
            R_ee = R_capt*_N + R_ee*(1-_N);
            V_ee = V_capt*_N + V_ee*(1-_N);
            B_ee = B_capt*_N + B_ee*(1-_N);
            wait_us(20);}    //avec N=100, ça fait comme une moyenne sur 20ms


        // changement de base de l'espace d'entrée vers celui de sortie <==> multiplication par M^-1
        R_es = seuil_lf(_M[0] * R_ee + _M[1] * V_ee + _M[2] * B_ee , 0,1);
        V_es = seuil_lf(_M[3] * R_ee + _M[4] * V_ee + _M[5] * B_ee , 0,1);
```

```
        B_es = seuil_lf(_M[6] * R_ee + _M[7] * V_ee + _M[8] * B_ee , 0,1);


        pc.printf("R_ee = %lf, V_ee = %lf , B_ee = %lf \r\n",R_es,V_es,B_es);
        pc.printf("R_es = %lf, V_es = %lf , B_es = %lf \r\n",R_es,V_es,B_es);

        s_R.write(R_es);
        s_V.write(V_es);
        s_B.write(B_es);
        wait(0.1);
    }
}

double seuil_lf(double x,double min,double max)
{
    double y=x;
    if (x<min) y=min;
    if (x>max) y=max;
    return(y);
}

void calibrage(void)
{
    int i;
    M[n_cal*3+0]=R_ee;    //on met les valeurs captées brute la n_cal -ème colonne de M
    M[n_cal*3+1]=V_ee;
    M[n_cal*3+2]=B_ee;
    for (i=0;i<=n_cal;i++){    //la led clignote n_cal+1 fois
        led_indic=1;
        wait_ms(333);
        led_indic=0;}
    n_cal++;    //au prochain tour, on remplira la colonne suivante
    if (n_cal>=3){
        n_cal=0;
        inverser_M3lf(M,_M);}
}

void inverser_M3lf(double A[9],double _A[9])
{
    double det;
    double _det=1;
    det=( A[0]*A[4]*A[8] + A[1]*A[5]*A[6] + A[2]*A[3]*A[7]
        - A[0]*A[5]*A[7] - A[1]*A[3]*A[8] - A[2]*A[4]*A[6] );
    if (abs(det) > 1e-8){ _det=1.0/det;}
    _A[0] = +(A[4]*A[8]-A[7]*A[5])*_det;
    _A[3] = -(A[3]*A[8]-A[6]*A[5])*_det;
    _A[6] = +(A[3]*A[7]-A[6]*A[4])*_det;
    _A[1] = -(A[1]*A[8]-A[7]*A[2])*_det;
    _A[4] = +(A[0]*A[8]-A[6]*A[2])*_det;
    _A[7] = -(A[0]*A[7]-A[6]*A[1])*_det;
    _A[2] = +(A[1]*A[5]-A[4]*A[2])*_det;
    _A[5] = -(A[0]*A[5]-A[3]*A[2])*_det;
    _A[8] = +(A[0]*A[4]-A[3]*A[1])*_det;
}
```

6.2 Code fonctionnel pour le système numéro 2

```
 #include "mbed.h"
 #define N 100
 #define _N 0.01
//matrice inverse (colonne par colonne)
#define iM11 1.54e-5
#define iM21 -5.30e-7
#define iM31 -1.15e-6

#define iM12 -2.70e-6
#define iM22 2.64e-5
#define iM32 -9.046e-6

#define iM13 2.23e-7
#define iM23 -1.13e-5
#define iM33 1.92e-5


double readRed();

double  readGreen();

double readBlue();

double seuil_lf(double x,double min,double max);
/*Prend en entrée un double x et renvoie sa valeur tronquée aux seuils min et max.
  x: double, égal à la sortie si min < x < max
  min: double, égal à la sortie si min >= x
  max: double, égal à la sortie si max <= x */


I2C i2c(D0,D1); //pins for I2C communication (SDA, SCL)
PwmOut s_B(D6);//Bleu
PwmOut s_V(D5);//Vert
PwmOut s_R(D3);//Rouge
Serial pc(USBTX, USBRX);    //Used to view the colors that are read in
int sensor_addr = 41 << 1;
int n_cal=0;
double M[9]={1,0,0 ,0,1,0 ,0,0,1};
double _M[9]={iM11,iM12,iM13,iM21,iM22,iM23,iM31,iM32,iM33};
double R_ee,V_ee,B_ee;
double R_es,V_es,B_es;

int main() {
    pc.baud(9600);
    pc.printf("toto");
    // Connect to the Color sensor and verify

    i2c.frequency(400000);

    char id_regval[1] = {146};
    char data[1] = {0};
    i2c.write(sensor_addr,id_regval,1, true);
    i2c.read(sensor_addr,data,1,false);

    // Initialize color sensor
```

```
    char timing_register[2] = {129,0};
    i2c.write(sensor_addr,timing_register,2,false);

    char control_register[2] = {143,0};
    i2c.write(sensor_addr,control_register,2,false);

    char enable_register[2] = {128,3};
    i2c.write(sensor_addr,enable_register,2,false);

    double R_capt,V_capt,B_capt;
    int i;
    R_ee = 0;
    V_ee = 0;
    B_ee = 0;

    while (true) {
        for (i = 0; i<N;i++){
            R_capt =  readRed();
            V_capt =  readGreen();
            B_capt =  readBlue();

            //Valeurs dans l'espace d'entrée (valeur captée * filtre passe-bas):
            R_ee = R_capt*_N + R_ee*(1-_N);
            V_ee = V_capt*_N + V_ee*(1-_N);
            B_ee = B_capt*_N + B_ee*(1-_N);
            wait_us(20);    //avec N=100, ça fait comme une moyenne sur 20ms
        }


    // changement de base de l'espace d'entrée vers celui de sortie <==> multiplication par M^-1
    R_es = seuil_lf(_M[0] * R_ee + _M[1] * V_ee + _M[2] * B_ee , 0,1);
    V_es = seuil_lf(_M[3] * R_ee + _M[4] * V_ee + _M[5] * B_ee , 0,1);
    B_es = seuil_lf(_M[6] * R_ee + _M[7] * V_ee + _M[8] * B_ee , 0,1);

    pc.printf("R_ee = %lf, V_ee = %lf , B_ee = %lf \r\n",R_es,V_es,B_es);
    pc.printf("R_es = %lf, V_es = %lf , B_es = %lf \r\n",R_es,V_es,B_es);

    s_R.write(R_es);
    s_V.write(V_es);
    s_B.write(B_es);
    wait(0.1);
    }
}

double readRed(){
    char red_reg[1] = {150};
    char red_data[2] = {0,0};
    i2c.write(sensor_addr,red_reg,1, true);
    i2c.read(sensor_addr,red_data,2, false);
    int red_value = ((int)red_data[1] << 8) | red_data[0];
    return red_value/65535.0;
    }

double readGreen(){
    char green_reg[1] = {152};
```

```
    char green_data[2] = {0,0};
    i2c.write(sensor_addr,green_reg,1, true);
    i2c.read(sensor_addr,green_data,2, false);
    int green_value = ((int)green_data[1] << 8) | green_data[0];
    return green_value/65535.0;
    }

double readBlue(){
    char blue_reg[1] = {154};
    char blue_data[2] = {0,0};
    i2c.write(sensor_addr,blue_reg,1, true);
    i2c.read(sensor_addr,blue_data,2, false);
    int blue_value = ((int)blue_data[1] << 8) | blue_data[0];
    return blue_value/65535.0;
    }

double seuil_lf(double x,double min,double max)
{
    double y=x;
    if (x<min) y=min;
    if (x>max) y=max;
    return(y);
}
```