année 2023

RAPPORT TECHNIQUE

CAMÉLÉON

INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech

Maëlle Bouhassane, Elena Rabadeux, Clémence Baudet

1 - Introduction

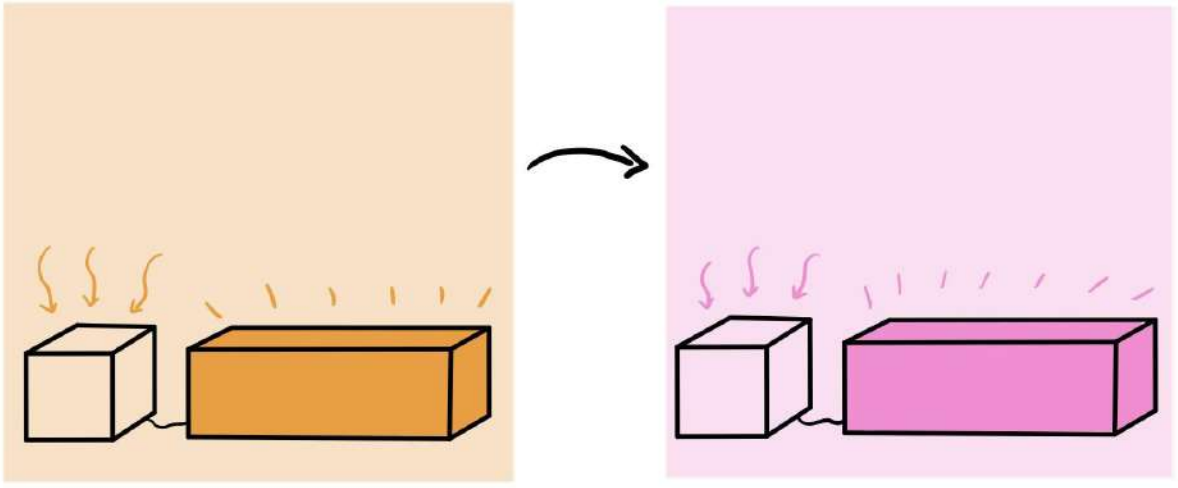
Le projet Caméléon consiste à réaliser un dispositif technologique d'ambiance qui sera capable de détecter la couleur d'un dispositif lumineux et de la restituer par l'intermédiaire d'un bandeau LED qui soit visible par l'œil humain. Un tel dispositif pourrait servir dans deux situations différentes. Dans un premier temps, pour les soirées (étudiants au sein de l'IOGS), pour pouvoir cacher les câbles des platines : le couleur changera en fonctions des lumières diffusées en soirée.

L'objectif est donc d'envoyer de la lumière sur un capteur RGB (ici, on utilisera un capteur de lumière colorée Color 14 Click de la marque Mikroe) qui va par la suite renvoyer son spectre RGB. À partir de ce spectre, il sera important de traiter le signal reçu et de récupérer les informations qui nous intéressent, telles que les composantes RGB, grâce à un programme réalisé sous MBED (Langage C). Ensuite, le signal numérique sera envoyé vers un bandeau de LEDs situé dans un pavé de verre diffusant et transmettra la même couleur que celle de la lumière de départ.

Cahier des charges

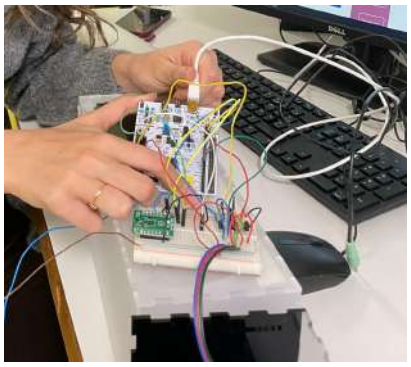
- pouvoir récupérer les composantes RGB de la lumière ambiante grâce à un capteur utilisant le protocole I2C.
- pouvoir restituer une couleur similaire à la couleur de la lumière ambiante envoyée sur le capteur dans un pavé de verre diffusant grâce à un bandeau de LEDs.

Schéma de principe

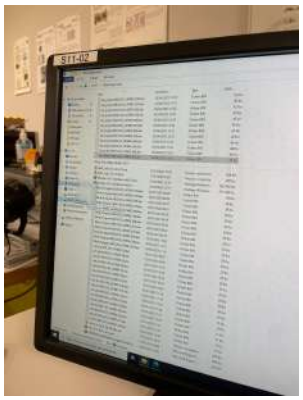


Notice d’utilisation

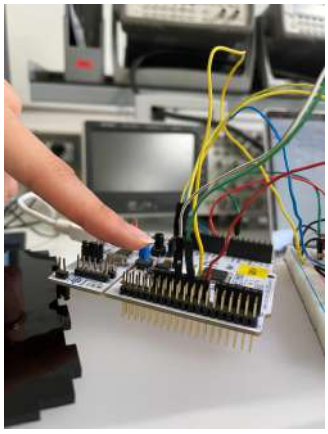
-Relier la carte Nucléo à l’ordinateur à l’aide du port USB.



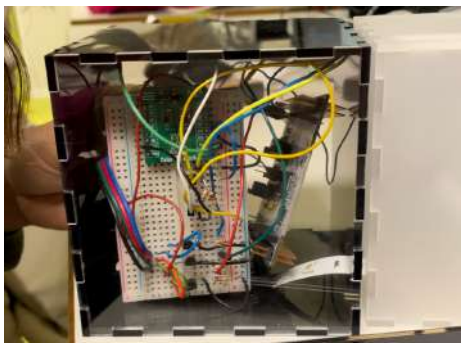
-Téléverser le code réalisé sous MBED sur le carte nucléo.



-Appuyer sur l’interrupteur bleu.



-Envoyer de la lumière colorée sur le capteur RGB à travers le cube en verre transparent



-Vous observez le dispositif changer de couleur !

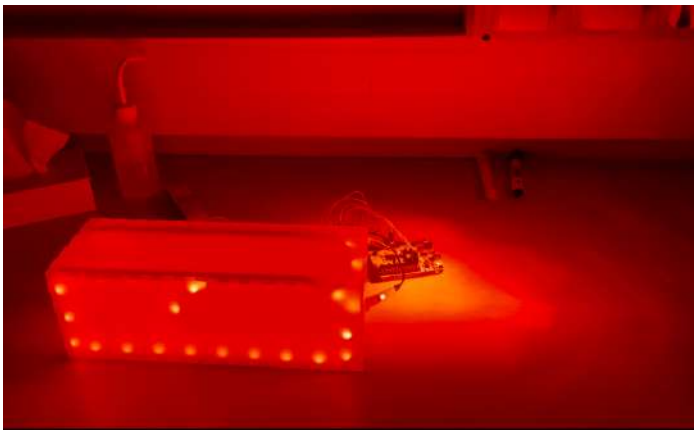
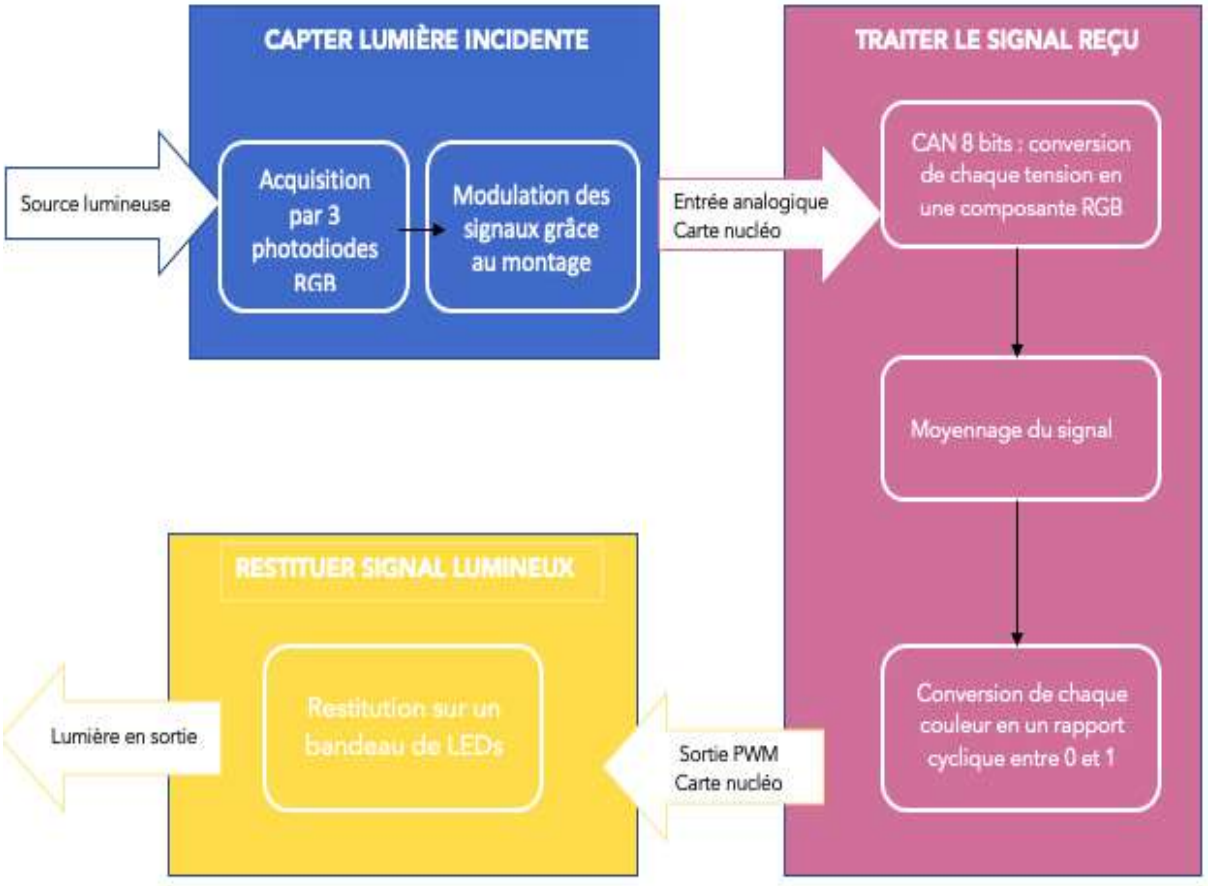


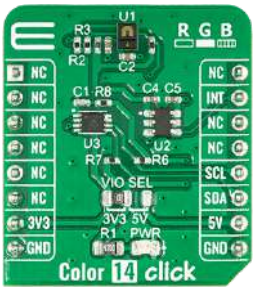
Schéma fonctionnel:



2 - Description des fonctions

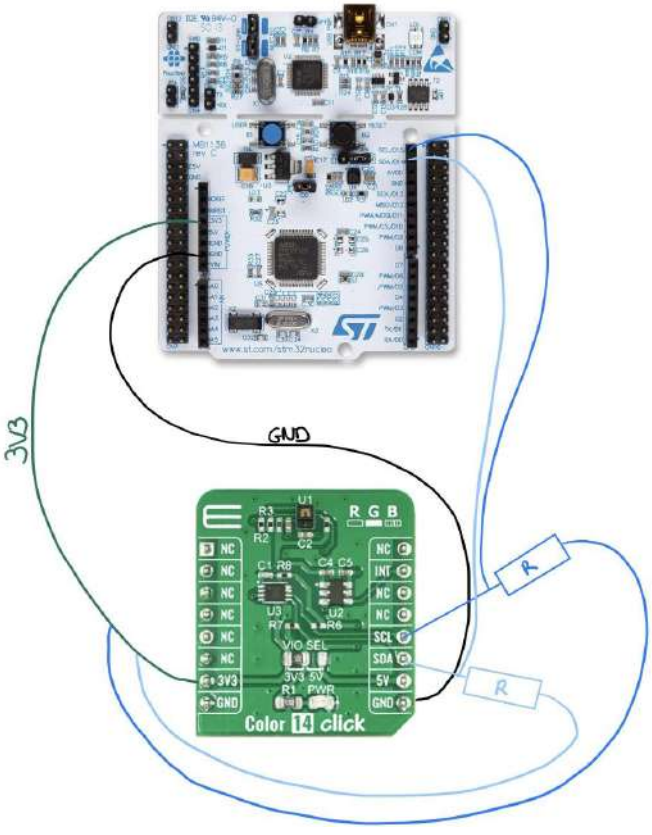
a - Détection

Utilisation d'un capteur de couleur Color 14 Click qui récupère les composantes RGB et IR de la lumière ambiante.
Les données sont stockées dans un vecteur de taille 4 (nommé RGBir).
Étant donné que la composante infrarouge ne nous intéresse pas dans le cadre de ce projet, nous n'utilisons que les 3 premières composantes (rouge, vert et bleu).



Color 14 Click

Câblage du capteur sur la carte Nucleo
Il faut tout d'abord alimenter le capteur en reliant les bornes 3V3 de chaque composant, ainsi que les masses GND. Il faut ensuite câbler les ports qui vont servir à envoyer les informations. Il s'agit des ports SCL et SDA. Pour les connecter à la carte Nucleo on ajoute des résistance de 1kΩ comme indiqué sur le schéma ci-dessous (résistances entre la sortie SCL/SDA) et l'alimentation à 3,3V).

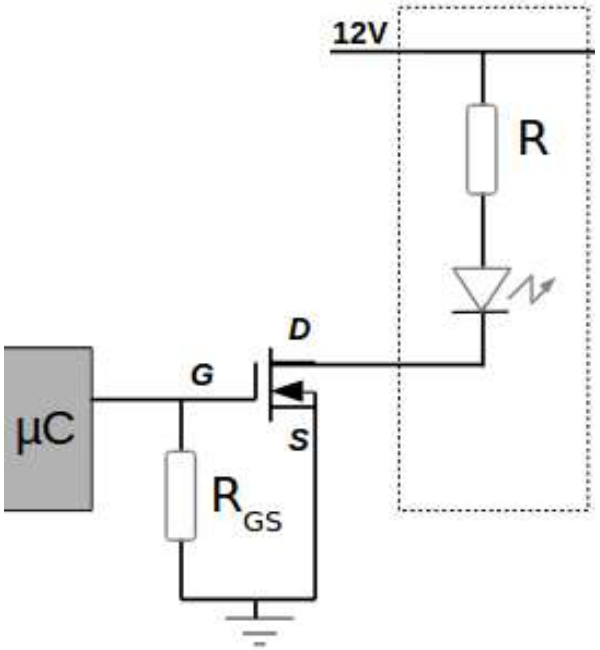


b - restitution

Pour pouvoir restituer la couleur de la lumière ambiante, on envoie les informations sur les composantes RGB de la carte nucleo vers le bandeau LED.
Pour que ces informations soient correctement assimilées par le bandeau LED on les normalise. On fixe la composante maximale à 1 et on ajuste les deux autres valeurs proportionnellement à ce maximum.
De plus, nous avons remarqué que dans la restitution des couleurs, le bleu étant souvent trop présent et le vert pas assez. Nous avons donc ajouté des coefficients aux valeurs envoyées au bandeau LED afin d'obtenir une couleur finale la plus proche de la couleur initiale (multiplié par 2 pour le vert et divisé par 1.5 pour le bleu).

Câblage du bandeau LED
Le bandeau LED doit être alimenté par une tension de 12 V.
Dans la partie détection, nous avons récupéré les 3 composantes RGB qui sont stockées dans la carte Nucléo. Chaque couleur va permettre d'alimenter les LED correspondantes du bandeau grâce à un circuit découpé par "étage". Un "étage" correspond à une couleur.
Celui-ci est constitué d'une résistance et d'un transistor, qui va permettre d'alimenter la lumière.

Voici un schéma d'un "étage" du câblage (photo prise sur le site du LEnsE dans "Comment alimenter un bandeau LED") :



Choix du transistor : BS170 MOSFET Pinout
Choix des résistances : il faut prendre une résistance de moins de 10kOhm. Dans notre cas, nous avons choisi une résistance de 8,2 kOhm.

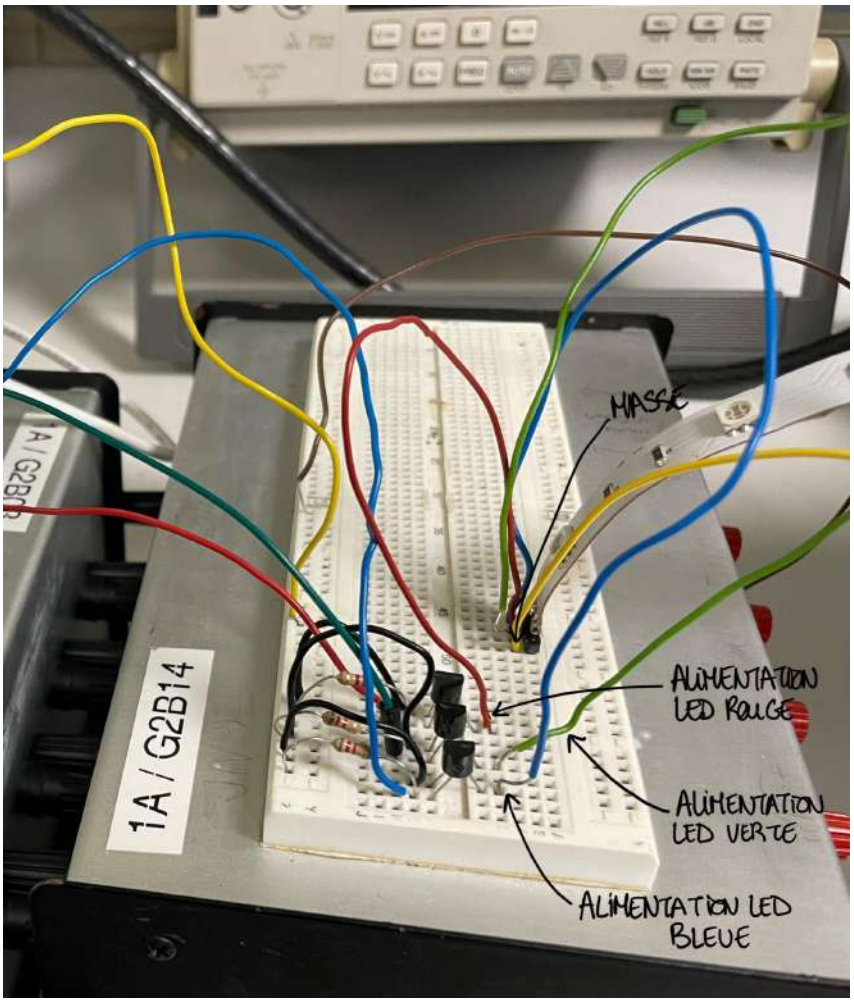
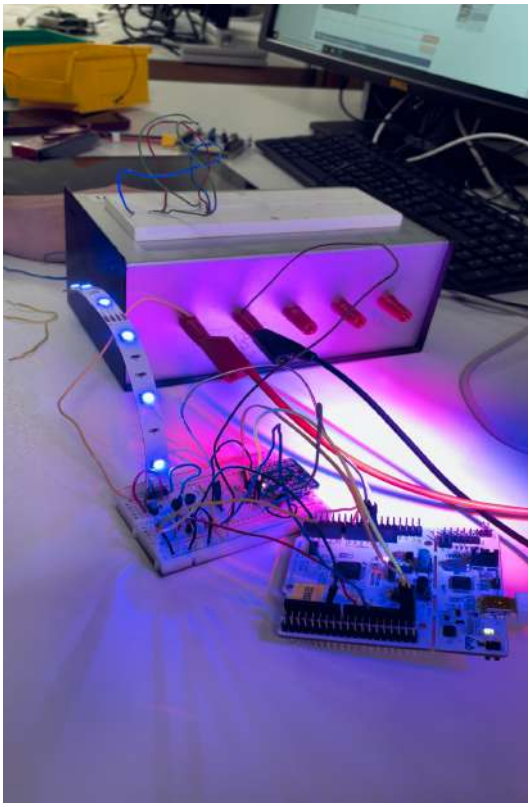


Photo du câblage du bandeau LED (circuit non miniaturisé)

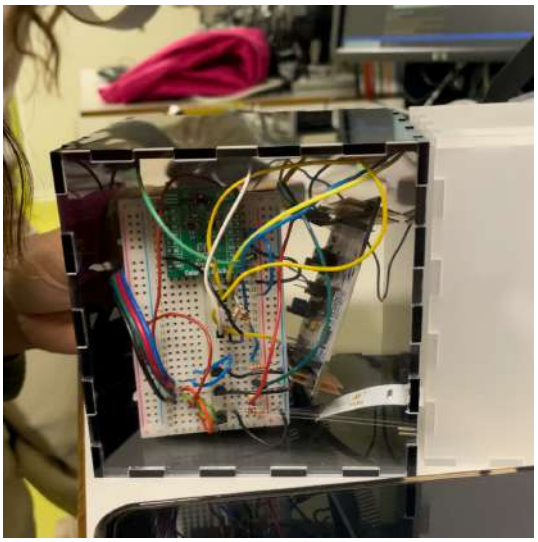
De plus, la masse du bandeau LED est reliée à la masse de tout notre système.

Boîtes de diffusion de la lumière
Finalement, le bandeau LED est placé dans une boîte de 50x10x10 cm composé d'un verre diffusant. Ces dimensions ont été choisies dans le but de cacher les platines pour les musiques en soirée. Nous avons également réalisé une deuxième boîte de dimension 10x10x10 cm pour y mettre notre capteur et le circuit miniaturisé. Cette dernière possède quatre faces noircies et deux transparentes afin que le capteur puisse récupérer la matrice des couleurs, tout en gardant un certain esthétisme.

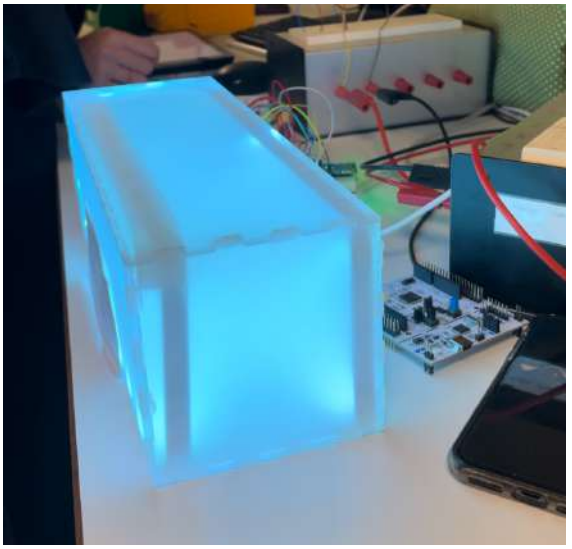
Nous avons également pensé à découper des trous de 2 cm de diamètre afin de faire passer les fils.



Principe de fonctionnement de notre projet



Boîtier contenant le capteur
et l'alimentation du bandeau LED



Boîte diffusante

3 - Bilan

Bilan technique et critiques

Finalement, notre dispositif marche comme nous le souhaitions et nous avons même réussi à aller plus loin en développant un cube portatif qui renvoie d'autres couleurs que du RGB (comme du rose, du violet ou du cyan par exemple).
Néanmoins, notre dispositif aurait pu être amélioré sur différents aspects. Par exemple, les fréquences de changement de la lumière sont assez limitées. Ceci est dû au temps d'exécution de la boucle du code. Une solution possible aurait été d'avoir une boucle plus courte ou de réduire le nombre d'opérations. De plus, le boîtier lumineux ne marche pas tant que le capteur n'est pas proche de la source lumineuse, ce qui risque d'être contraignant en pratique (surtout dans le contexte des soirées). Finalement, nous avons réalisé un boîtier trop petit pour y mettre tout le circuit miniaturisé et une batterie.

Pour autant, des difficultés ont été rencontrées au cours de notre projet. Par exemple, la compréhension et la réalisation du protocole I2C a pris plus de temps que prévu (cf planning et répartition des tâches). Sans l'aide des professeurs qui nous ont envoyé les codes par mail, nous n'aurions pas pu avancer.
Ensuite la batterie nous a posé beaucoup de problèmes. Avant les vacances de la Toussaint, le boîtier lumineux marchait très bien avec cette dernière. A notre retour, rien ne voulait fonctionner. Nous n'avions pas touché au code, ni aux circuits. Il semblait que la batterie fournie était beaucoup trop puissante (nous avons fait cramer deux cartes Nucléo, quelques fils et vu des étincelles) pour ce type de projet.

Planning et répartition des tâches

A la deuxième séance IéTI, nous avons établi un planning dans notre cahier des charges afin de bien organiser notre projet et d'obtenir un dispositif qui marche à la fin des séances. Voici le planning prévu :

Numéro séance	Étapes du projet	Qui ?
1	- Description du système - Lecture fiche technique et choix du capteur - Schéma de principe	- Clémence - Elena - Maëlle
2	- Rédaction cahier des charges - Planning	- Elena - Maëlle
3	- Lire et protocole I2C - Réaliser le code afin de récupérer le spectre et convertir les données pour récupérer une composante en RGB - Faire le prototype de la boîte(schéma, câblage,etc)	- Toutes - Clémence et Maëlle - Elena
4	- Lire fiche technique du spot + schéma de câblage du système - Construction de la boîte (découper verre diffusant (laser), coller bandeau led,etc)	- Maëlle - Clémence et Elena
5	- Tester le prototype	-Toutes
6	- Schéma, photo prototype - Tracer le spectre (python, langage c)	- Clémence - Elena et Maëlle
7	- Amélioration du système et miniaturisation possible ? (plus de couleur)	- Toutes

Seulement, nous avons rencontré des difficultés qui nous ont obligé à changer quelques aspects de notre planning. Par exemple, la rédaction du protocole I2C a pris plus de temps que prévu.
De plus, pour mieux améliorer notre avancement, nous nous sommes séparées les tâches, mais nous nous concertions régulièrement afin de suivre la progression de chacune. Par exemple, Maëlle et Clémence ont beaucoup travaillé sur le protocole I2C, alors qu'Elena a fait des recherches bibliographiques sur les différents composants (capteur Color 14 Click, transistors, bandeau LED) et s'est concentrée sur le câblage du bandeau LED.
Néanmoins, une fois ces étapes réalisées, nous avons toutes réfléchi à la conception des boîtiers.
Pour suivre nos avancements, nous mettions tous nos suivis sur une plateforme de travail, qui est Google Drive.

Suivi de la progression :

DATE	PLANNING	QU'EST CE QU'ON A FAIT	À FAIRE
17/01		Clem: description du système + questions Elena : lecture des descriptions des deux capteurs à disposition (Color 10 click and Color 14 click) + choix du capteur à utiliser - Color 14 Click Maëlle: schéma de principe du système	
24/01	Cahier des charges / Planning et répartition des rôles / Compétences que l'on cherche à acquérir Ne pas mélanger fonctions et fonctionnalités	Maëlle : réalisation du planning pour les prochaines séances Elena : rédaction du cahier des charges et des compétences que l'on cherche à acquérir Toutes : discussion avec les chargés de TD Clémence : absente	
7/02	- Bien comprendre protocole I2C - Rédaction de première partie du code - Réflexion design de la boîte	Maëlle et Clémence : réalisation et compréhension des premiers blocs de codes (protocole I2C) Elena : recherches bibliographiques sur comment piloter des LEDs... + réalisation du câblage du bandeau LED + conception d'une première idée de prototype	Maëlle et Clémence : comprendre le code envoyé par les profs Elena : bien mettre sur papier différences entre les deux capteurs + réflexion boîte pour prototype
14/02	- Continuer à comprendre le protocole I2C envoyé par les profs par mail - Aller voir Thierry Avignon pour construire le prototype - Lecture fiche tech LED + début schéma	Maëlle et Clémence réflexion sur le code et protocole I2C Elena : branchement du bandeau LED et circuit qui va avec	Tout le monde : commencer à réfléchir à la rédaction du livrable En lien avec notre projet : http://lense.institutoptique.fr/livrables/leTI/2021-2022/Gpe2/G2_4/Poster%20DARC.pdf http://lense.institutoptique.fr/livrables/leTI/

	câblage		
29/03	- Tester le prototype - Se refamiliariser avec le code - Apporter plus de couleurs dans le code - Aller au LenSE pour demander la fabrication de la boîte - Réflexion sur la miniaturisation du système	Tout le monde : découpage laser des deux boîtes (le pavé de 30x10x10 cm - changement de taille et un cube de 10x10x10 cm avec 4 faces noires et deux faces transparentes pour que le capteur puisse capter les couleurs Clémence : améliorer le code pour que le capteur capte des couleurs autres que RGB Maëlle et Elena : Réflexion sur comment placer et brancher le bandeau LED dans la boîte	
5/04	- Miniaturiser le prototype - Coller les LEDs avec de la super glue - Tester le prototype (vérifier son fonctionnement) - Trouver une alimentation externe - Comment charger le programme sur la carte Nucléo sans qu'elle ne soit reliée par l'ordinateur - Réflexion sur comment ajouter plus de couleurs dans le code	Problèmes que l'on peut rencontrer : - Comment charger le programme sur la carte Nucléo ? - Comment alimenter à la fois la carte Nucléo et le bandeau LED ? - Fonctionnement continu : instauration d'un bouton poussoir et le programmer ? Mise en place d'un scotch double face dans la boîte diffusive Code du bouton poussoir Miniaturisation du prototype Branchement avec une batterie Programmation pour avoir plus de couleurs que du RGB	Réflexion sur les livrables, poster et vidéo (à filmer au SonopOptik) Mieux scotcher le bandeau LED (esthétisme)
10/05	- Retester le prototype - Tournage vidéo au	Problème rencontré aujourd'hui : plus d'espace de stockage carte Nucléo - possibilité qu'elle ait cramée	
	Sonop - Faire affiche livrable - Faire schémas de câblage (circuits 1 et 2) - Répartition du temps de paroles - Réalisation des diapos ? - Mettre le code dans un dossier zip - Commentaire code ?		

Retour sur l'expérience

Nous avons pris du plaisir à réaliser le Caméléon. En effet, le format du projet IéTI est une manière beaucoup plus ludique de découvrir les applications de l'optique dans l'électronique, de comprendre le codage et les différentes interfaces nécessaires pour convertir des signaux analytiques en numériques. Nous avons pu voir une application plus concrète de ce que l'on a appris en électronique au semestre 5.
Nous avons également appris à travailler ensemble et la gestion de projet sur plusieurs semaines.

En outre, voici les différentes compétences acquises pendant les semaines IéTI :

- Prototyper avec Nucléo et MBED
- Piloter des LEDs
- Traiter un signal numérique
- Faire de la découpe laser
- Coder sous MBED (Langage C)
- Faire de la documentation
- Apprendre la gestion d'un travail en équipe

4-Annexes

```
#define __COLOR_14_CLICK_HEADER_H__

#include <mbed.h>

/** Constant definition */
#define     DEBUG_MODE              1
#define     COLOR_14_CLICK_ADD          0x52
#define     COLOR_14_CLICK_MAIN_CTRL    0x00
#define     COLOR_14_CLICK_PART_ID      0x06
#define     COLOR_14_CLICK_MAIN_STAT    0x07
#define     COLOR_14_CLICK_RED_CHAN     0x13
#define     COLOR_14_CLICK_GREEN_CHAN   0x0D
#define     COLOR_14_CLICK_BLUE_CHAN    0x10
#define     COLOR_14_CLICK_IR_CHAN      0x0A
#define     COLOR_14_CLICK_LS_GAIN      0x05

#define     COLOR_14_CLICK_LS_GAIN_1X   0x00
#define     COLOR_14_CLICK_LS_GAIN_3X   0x01
#define     COLOR_14_CLICK_LS_GAIN_6X   0x02
#define     COLOR_14_CLICK_LS_GAIN_9X   0x03
#define     COLOR_14_CLICK_LS_GAIN_18X  0x04



/**
 * @class Color_14_Click
 * @brief Access to the Color14Click module from MikroE
 * @details     Color14Click module allows to measure RGB light intensities
 *  with a APDS-9151 sensor from BroadCom.
 */
class Color_14_Click{
    private:
      /// Intensity of the Red component
      int     Red_color;
      /// Intensity of the Green component
      int     Green_color;
       /// Intensity of the Blue component
      int     Blue_color;
      /// Intensity of the InfraRed component
      int     IR_color;
      /// Command to send
      char    cmd[2];
      /// Received Data
      char    data[3];
      /// Acknowledgement variables
      char    ack1, ack2;

      /// I2C interface pins
      I2C           *__i2c = NULL;
       /// interrupt Input pin
```

```
    InterruptIn     *__int = NULL;

public:
    /**
    * @brief Simple constructor of the Color_14_Click class.
    * @details Create a Color_14_Click object with
    *    an I2C interface
    *    I2C communication will be initialized at 400kHz
    * @param _i2c SPI interface not initialized
    * @param _int interrupt Input
    */
      Color_14_Click(I2C *_i2c, InterruptIn *_int);

    /**
    * @brief Initiatlization of the sensor
    * @details Initialize the sensor
    */
    void powerUp(void);

    /**
    * @brief Initiatlization in RGB Mode only
    * @details Initialize the sensor in RGB Mode only
    */
    void initRGBSensor(void);

    /**
    * @brief Return the ID of the module
    * @return the part ID of the module - default 0xC2 = 194d
    */
    int getPartID(void);

    /**
    * @brief Return the status of the module
    * @return the part status of the module - default 0x20 = 32d or 0x00
    */
    int getMainStatus(void);

    /**
    * @brief Set the analog gain of the sensor
    * @details Gain in range : 1x, 3x (default), 6x, 9x, 18x
    * @param val gain value - COLOR_14_CLICK_LS_GAIN_1X
    */
    void setGainRGB(int val);

    /**
    * @brief Read the red data from the Color_14_Click module
    * @details Read the red data from the Color_14_Click module
    *   and update the member value of the object -
    *
    * @return the Red component of the light.
    */
    int readRedValue(void);

    /**
    * @brief Read the green data from the Color_14_Click module
```

```
    * @details Read the green data from the Color_14_Click module
    *   and update the member value of the object -
    *
    * @return the Green component of the light.
    */
    int readGreenValue(void);

    /**
    * @brief Read the blue data from the Color_14_Click module
    * @details Read the blue data from the Color_14_Click module
    *   and update the member value of the object -
    *
      * @return the Blue component of the light.
    */
    int readBlueValue(void);

    /**
    * @brief Read the InfraRed data from the Color_14_Click module
    * @details Read the infrared data from the Color_14_Click module
    *   and update the member value of the object -
    *
    * @return the IR component of the light.
    */
    int readIRValue(void);

    /**
    * @brief Collect all the RGBIR data from the Color_14_Click module
    * @details Read all the RGBIR data from the Color_14_Click module
    *   and update the members value of the object -
    *
    * @param rgbIR first cell of a 4 int arrays
    * @return R G B IR value in a 4 int arrays
    */
    void readRGBIRValue(float rgbIR[]);
};
```

```
#include "mbed.h"


#define WAIT_TIME_MS 500
DigitalOut led1(LED1);
PwmOut rouge(D9); //Déclaration de la sortie pour la composante rouge
PwmOut vert(D6); //Déclaration de la sortie pour la composante vert
PwmOut bleu(D11);//Déclaration de la sortie pour la composante bleu
DigitalIn bouton(PC_13); // Déclaration du bouton comme entrée numérique
/// Create an I2C interface with specific pins : SDA / SCL
I2C               my_color_i2c(D14, D15); // SDA / SCL
/// Create an interrupt input for the Color_14_Click Module INT pin
InterruptIn       my_color_int(D10);
/// Create a Color_14_Click module connection with an I2C interface and an Interrupt Input
Color_14_Click    my_sensor(&my_color_i2c, &my_color_int);
```

```
/// Create a 4 integers array to collect R, G, B and IR data from the sensor
float           RGBir[4] = {0};

int main()
{
   while(1){
   int a=1;
   rouge.write(0); //on met la composante rouge à 0
   vert.write(0);//on met la composante verte à 0
   bleu.write(0);//on met la composante bleue à 0
   wait_us(100000);
   while(a==1){
      a = bouton;
   }
   while(a==0){ // si on appuie sur le bouton
   int k = 0;
   printf("Test.\n");
   /// PowerUp the module
   my_sensor.powerUp();
   /// Collect informations about the sensor
   printf("Part ID = %d\r\n", my_sensor.getPartID());
   printf("Status = %d\r\n", my_sensor.getMainStatus());

   /// Initialize the sensor in RGB Mode
   my_sensor.initRGBSensor();
   /// Modify the analog gain of the sensor - 3x by default
   my_sensor.setGainRGB(COLOR_14_CLICK_LS_GAIN_6X);

   //while (true)
   //{
      k++;
      led1 = !led1;
      /// Collect R, G, B and IR data from the sensor
      my_sensor.readRGBIRValue(RGBir);

        int max=RGBir[0]; //on recherche le maximum parmi les composantes RGB collectées par
le capteur
      if (RGBir[1]>max){
         max=RGBir[1];
      }
      if (RGBir[2]>max){
         max=RGBir[2];
      }
      int i;
      for (i=0; i<3; i++){
         if (RGBir[i] == max){
            RGBir[i] = 1.0; // on fixe la composante maximum à 1 afin de normaliser
         }
         else{
            RGBir[i] = RGBir[i]*1.0/max; // on normalise les deux autres composantes à l'aide du
max
            if(RGBir[i]<0.4){
               RGBir[i]=0; // on fixe à zéro les composantes en dessous du seuil
            }
         }
```

```
        } printf("[%4d] R=%d / G=%d / B=%d / IR = %d \r\n\n ", k, RGBir[0], RGBir[1], RGBir[2],
RGBir[3]);
         /// Every 0.5s
        thread_sleep_for(WAIT_TIME_MS);
        rouge.period_us(100);
        vert.period_us(100);
        bleu.period_us(100);
        rouge.write(RGBir[0]); // on envoie la valeur colléctée et normalisée pour la composante
rouge en sortie
        vert.write(2*RGBir[1]);// on envoie la valeur colléctée,normalisée et ajustée pour la
composante verte en sortie
        bleu.write(RGBir[2]/1.5);// on envoie la valeur colléctée,normalisée et ajustée pour la
composante bleue en sortie
      // }
       a = 1-bouton;
       }
       }
}

Color_14_Click::Color_14_Click(I2C *_i2c, InterruptIn *_int){
    /* Initialisation of interrupt input */
    if (_int){ delete __int; }
    __int=_int;
    /* Initialisation of i2c module */
    if (_i2c){ delete __i2c; }
    __i2c=_i2c;
    __i2c->frequency(400000);   // Frequency of 400kHz
    thread_sleep_for(10);      // 10 ms
}

void Color_14_Click::powerUp(void){
    cmd[0] = COLOR_14_CLICK_MAIN_CTRL;
    cmd[1] = 0b00000100;
    ack1 = __i2c->write(COLOR_14_CLICK_ADD << 1, cmd, 2);
    if(DEBUG_MODE) printf("Init Acq = %d\r\n", ack1);
    wait_us(1000);
}

void Color_14_Click::initRGBSensor(void){
    cmd[0] = COLOR_14_CLICK_MAIN_CTRL;
    cmd[1] = 0b00000110;
    ack1 = __i2c->write(COLOR_14_CLICK_ADD << 1, cmd, 2);
    if(DEBUG_MODE) printf("Init Acq = %d\r\n", ack1);
    wait_us(1000);
}

int Color_14_Click::getPartID(void){
    // Part ID Status
    cmd[0] = COLOR_14_CLICK_PART_ID;
    ack1 = __i2c->write(COLOR_14_CLICK_ADD << 1, cmd, 1, true);
    ack2 = __i2c->read(COLOR_14_CLICK_ADD << 1, data, 1);
    if(DEBUG_MODE)  printf("Part ID Acq (W) = %d\r\n", ack1);
    if(DEBUG_MODE)  printf("Part ID Acq (R) = %d\r\n", ack2);
```

```
    return data[0];
}

int Color_14_Click::getMainStatus(void){
    cmd[0] = COLOR_14_CLICK_MAIN_STAT;
     ack1 = __i2c->write(COLOR_14_CLICK_ADD << 1, cmd, 1, true);
    ack2 = __i2c->read(COLOR_14_CLICK_ADD << 1, data, 1);
    if(DEBUG_MODE)  printf("Main Status Acq (W) = %d\r\n", ack1);
    if(DEBUG_MODE)  printf("Main Status Acq (R) = %d\r\n", ack2);
    return data[0];
}

void Color_14_Click::setGainRGB(int val){
    cmd[0] = COLOR_14_CLICK_LS_GAIN;
    cmd[1] = val;
    ack1 = __i2c->write(COLOR_14_CLICK_ADD << 1, cmd, 2);
    if(DEBUG_MODE)  printf("Gain Acq (W) = %d\r\n", ack1);
}

int Color_14_Click::readRedValue(void){
    cmd[0] = COLOR_14_CLICK_RED_CHAN;
    ack1 = __i2c->write(COLOR_14_CLICK_ADD << 1, cmd, 1, true);
    ack2 = __i2c->read(COLOR_14_CLICK_ADD << 1, data, 3);
    if(DEBUG_MODE)  printf("Red Chan Acq (W) = %d\r\n", ack1);
    if(DEBUG_MODE)  printf("Red Chan Acq (R) = %d\r\n", ack2);
     Red_color = (data[2] << 16) + (data[1] << 8) + data[0];
    return Red_color;
}

int Color_14_Click::readGreenValue(void){
    cmd[0] = COLOR_14_CLICK_GREEN_CHAN;
    ack1 = __i2c->write(COLOR_14_CLICK_ADD << 1, cmd, 1, true);
    ack2 = __i2c->read(COLOR_14_CLICK_ADD << 1, data, 3);
    if(DEBUG_MODE)  printf("Green Chan Acq (W) = %d\r\n", ack1);
    if(DEBUG_MODE)  printf("Green Chan Acq (R) = %d\r\n", ack2);
    Green_color = (data[2] << 16) + (data[1] << 8) + data[0];
    return Green_color;
}

int Color_14_Click::readBlueValue(void){
    cmd[0] = COLOR_14_CLICK_BLUE_CHAN;
    ack1 = __i2c->write(COLOR_14_CLICK_ADD << 1, cmd, 1, true);
    ack2 = __i2c->read(COLOR_14_CLICK_ADD << 1, data, 3);
    if(DEBUG_MODE)  printf("Blue Chan Acq (W) = %d\r\n", ack1);
    if(DEBUG_MODE)  printf("Blue Chan Acq (R) = %d\r\n", ack2);
    Blue_color = (data[2] << 16) + (data[1] << 8) + data[0];
    return Blue_color;
}



int Color_14_Click::readIRValue(void){
    cmd[0] = COLOR_14_CLICK_IR_CHAN;
    ack1 = __i2c->write(COLOR_14_CLICK_ADD << 1, cmd, 1, true);
    ack2 = __i2c->read(COLOR_14_CLICK_ADD << 1, data, 3);
```

```
    if(DEBUG_MODE)  printf("IR Chan Acq (W) = %d\r\n", ack1);
    if(DEBUG_MODE)  printf("IR Chan Acq (R) = %d\r\n", ack2);
    IR_color = (data[2] << 16) + (data[1] << 8) + data[0];
    return IR_color;
}

void Color_14_Click::readRGBIRValue(float rgbIR[]){
    rgbIR[0] = readRedValue();
    rgbIR[1] = readGreenValue();
    rgbIR[2] = readBlueValue();
    rgbIR[3] = readIRValue();
}
```