

Groupe B1 - Sonomètre portatif

Membres du groupe : Maïlys HERMANN-BOYER, Shirel FELLOUS, Lily-Rose GAVANON

Page de documentation d'étudiantes en L1 de CMI Physique dans le cadre du Projet Fablab (découverte du Fablab SU & réalisation d'un projet incluant un capteur environnemental) en 10 séances de 3h

Découverte du Fablab

Séance 1 : Introduction à l'UE

Aujourd'hui, ce 23 janvier 2023, nous avons eu la chance de participer à notre premier cours de l'UE Fablab. Cela a été organisé dans l'enceinte du Laboratoire Fablab qui se trouve dans le bâtiment Esclangon de l'université de la Sorbonne. Le Fablab est un atelier créé à l'origine par le professeur **Neil Gershenfeld** au MIT, dans l'optique de réunir différentes personnes passionnées par la recherche et le développement de créations quelconques. Un réseau mondial de laboratoires s'est donc formé afin de donner accès à des outils de fabrication numérique à de multiples individus.

Nous avons donc pu visiter ce laboratoire et suivre les explications des différentes machines telles que des imprimantes 3D, des fraiseuses numériques ou encore des découpeuses laser.

Notre but dans les séances à venir sera de réfléchir à un projet permettant la création d'un objet utilisable au quotidien. Ce projet devra inclure un ou plusieurs capteurs environnementaux.

Séance 2 : Circuits Arduino et capteurs

Ce lundi 30 janvier, la séance portait sur le **prototypage électronique**. Nous avons été initiés à l'histoire de l'électronique, du tube à vide en 1904 au M5Stack en 2017 ; nous avons pu voir la chronologie d'apparition des différentes techniques et découvertes électroniques.

Nous avons découvert **la carte Arduino**, prototypée en 2003 par un étudiant en Master et apparue réellement sous forme de carte en 2005. Ce composant est en fait un microcontrôleur, capable de capter des signaux analogiques (pouvant prendre toutes les valeurs entre 0 et 5 V). Pour programmer un Arduino, il faut utiliser un langage spécifique, écrire un code "IDE Arduino".

Ensuite, nous avons essayé de comprendre le fonctionnement de la carte Arduino, tout d'abord avec un programme permettant de faire clignoter la LED de l'Arduino. Nous avons ensuite essayé de récupérer les données recueillies par un **capteur de température et d'humidité**. Il fallait tout d'abord placé un Grove Shield sur la carte Arduino afin de pouvoir brancher différents composants. On retrouve sur ce Shield des **ports numériques** et **analogiques**, comme sur l'Arduino, mais aussi des ports **I2C**, que nous utilisons pour relier le capteur à l'Arduino. Le transfert de l'ordinateur à l'Arduino se fait **en série**.

Afin de pouvoir utiliser un composant particulier, il faut télécharger une **bibliothèque** associée qui contient toutes les fonctions liées à ce composant. Ces bibliothèques sont facilement trouvable sur [Seeed Studio](#) ou en cherchant le nom du composant sur Internet. Pour chaque capteur il peut y avoir une bibliothèque et un port différents. Il faut donc se renseigner sur le Wiki pour avoir les renseignements du capteur utilisé.

Par la suite, nous avons utilisé d'autres capteurs tel qu'un "**gas sensor**" qui permet de calculer la concentration de CO2 dans l'air afin d'estimer si l'on est dans un environnement nocif ou non. Celui se branche par la connecteur **A0**.

Nous avons enfin essayé de travailler avec plusieurs composants en cherchant à afficher les valeurs de température et d'humidité enregistrées par le capteur sur un petit écran (16x2 LCD). Pour cela, nous avons essayé de combiner les 2 codes des deux composants utilisés. Après quelques essais, on arrivait à lire les données du capteur en ayant téléversé le code suivant :

```
#include <Arduino.h>
```

```
#include <Wire.h>
```

```
#include "SHT31.h"
```

```
#include <Wire.h>
```

```
#include "rgb_lcd.h"
```

```
rgb_lcd lcd;
```

```
const int colorR = 209;
```

```
const int colorG = 141;
```

```

const int colorB = 247;

SHT31 sht31 = SHT31();

void setup() {
  Serial.begin(9600);
  while(!Serial);
  Serial.println("begin...");
  sht31.begin();
}

void loop() {
  float temp = sht31.getTemperature();
  float hum = sht31.getHumidity();

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  lcd.setRGB(colorR, colorG, colorB);

  // Print a message to the LCD.
  lcd.print("Temp = ");
  lcd.print(temp);
  lcd.println(" C"); // "\n" pour retour à la ligne
  lcd.print("Hum = ");
  lcd.print(hum);
  lcd.println("%");

  delay(1000);
}

```

Nous avons ensuite essayé de réaliser la même chose sur un écran un peu plus complexe, un **OLED Display 0.96 inch**. Le code était le suivant :

```

#include <Arduino.h>
#include <Wire.h>
#include "SHT31.h"

SHT31 sht31 = SHT31();

```

```

#include <Arduino.h>
#include <U8g2lib.h>

#ifdef U8X8_HAVE_HW_SPI
#include <SPI.h>
#else
#include <Wire.h>
#endif

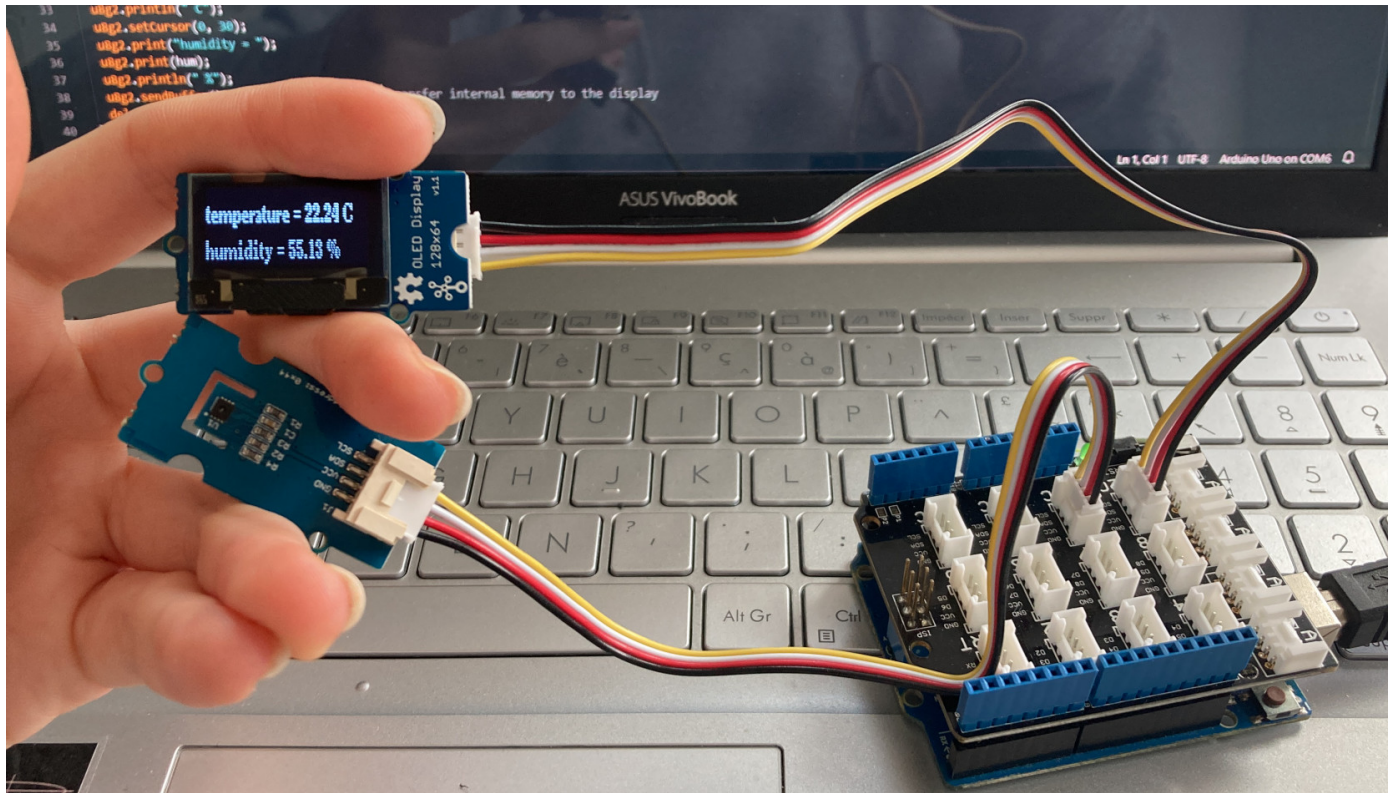
U8G2_SSD1306_128X64_ALT0_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);

void setup() {
  sht31.begin();
  u8g2.begin();
}

void loop() {
  float temp = sht31.getTemperature();
  float hum = sht31.getHumidity();

  u8g2.clearBuffer();           // clear the internal memory
  u8g2.setFont(u8g2_font_ncenB08_tr); // choose a suitable font
  u8g2.setCursor(0, 15);
  u8g2.print("temperature = ");
  u8g2.print(temp);
  u8g2.println(" C");
  u8g2.setCursor(0, 30);
  u8g2.print("humidity = ");
  u8g2.print(hum);
  u8g2.println(" %");
  u8g2.sendBuffer();           // transfer internal memory to the display
  delay(200);
}

```



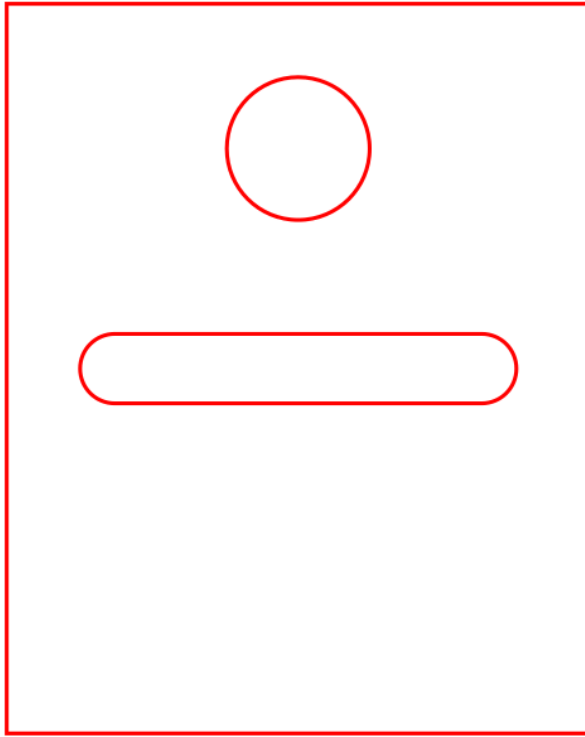
Pour conclure, c'était une séance très constructive qui nous a donné différentes idées pour notre projet final.

Séance 3 : Dessin 2D, 3D, impression 3D et découpe laser

Lors de cette séance, nous avons découvert certains logiciels libres et gratuits de dessin et modélisation 3D. Des tutoriels pour chacun de ces logiciels sont disponibles sur le Wiki du FABLAB ou sur Internet.

La logiciel principalement utilisé pour la découpe laser est le logiciel **Inkscape**. Afin que le dessin réalisé soit exploitable, il faut s'assurer que l'épaisseur du trait soit toujours de 1px. En ce qui concerne la couleur des traits, le rouge correspond à la découpe alors que le noir fait référence à la gravure. Le plus important lors de l'utilisation de ce logiciel est de contrôler les dimensions et le positionnement des différents objets. Le format lu par les découpes laser est le **format SVG**.

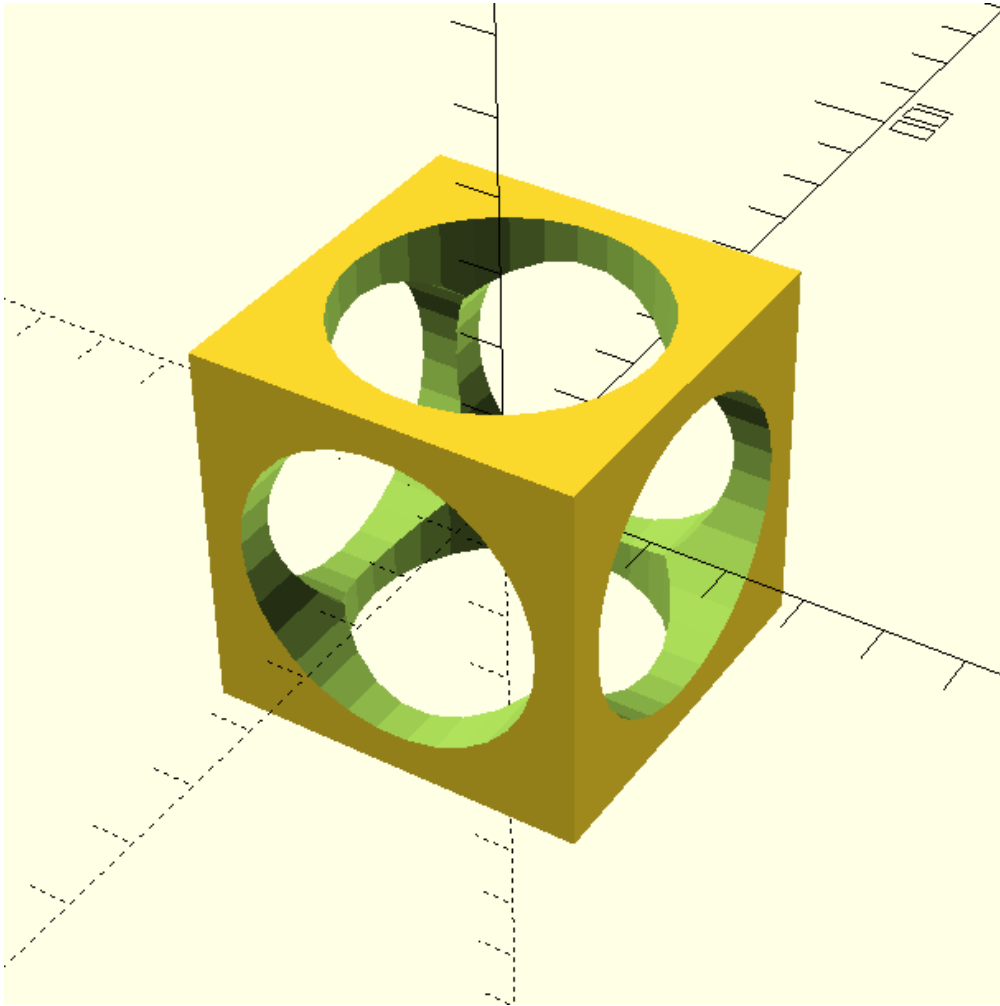
Nous avons réalisé ce dessin lors de notre découverte du logiciel.



Nous avons ensuite été initiées à 2 logiciels de modélisation 3D : **OpenSCAD** et **FreeCAD**. Ces logiciels sont utilisés dans le cadre d'impressions 3D. Pour cela, il est important de savoir que le format lu par les imprimantes est le **format STL**. Nous nous limiterons à l'utilisation du workbench **Part** dans FreeCAD. Afin de se familiariser avec les outils des deux logiciels, nous avons essayé de réaliser un cube de 50mm de côté troué par 3 cylindres de 20mm de diamètre.

Sur OpenSCAD :

```
difference() {  
  cube(50,center=true);  
  translate([0,0,-25]) cylinder(h=50,r=20);  
  rotate([90,0,0]) translate([0,0,-25]) cylinder(h=50,r=20);  
  rotate([0,90,0]) translate([0,0,-25]) cylinder(h=50,r=20);  
}
```



Nous avons ensuite pu voir un peu plus en détails comment faire fonctionner les différentes machines du FABLAB.

Pour les imprimantes 3D, le logiciel utilisé est **IdeaMaker**. Il est toujours important de réfléchir au remplissage d'un objet lors de son impression. Il faut aussi penser à l'épaisseur du fil utilisé (+ c'est fin + c'est précis mais + c'est long), à l'orientation de l'objet sur le support.

Séance 4 : Prototypage électronique

Aujourd'hui, nous avons exploré un peu plus en profondeur le prototypage électronique avec un **M5Stack**.

Le M5Stack est un boîtier programmable avec **Arduino IDE**, contenant de la mémoire (4Mo), une carte SD et des haut-parleurs. Une sortie I2C permet d'y brancher des capteurs et un port USB-C pour le relier à un ordinateur.

Nous avons commencé par tenter de faire fonctionner le M5Stack avec des programmes exemples trouvés sur GitHub.

En téléchargeant [ce fichier ZIP](#), on accède à la bibliothèque du M5Stack. Nous avons tout d'abord ouvert le programme "Hello World" (exemples > basics). Il ne faut pas oublier de sélectionner la board M5Stack Core ESP32 dans Arduino IDE ainsi que le bon port.

Nous avons ensuite essayé d'afficher les données d'un capteur de température et d'humidité sur l'écran du M5Stack.

on a téléchargé le fichier ZIP du M5stack puis dans exemples > advanced > display > free font demo > on ouvre direct le fichier

```
#include <Arduino.h>
#include <M5Stack.h>
#include <Wire.h>
#include "SHT31.h"
#include "Free_Fonts.h"

SHT31 sht31 = SHT31();

void setup() {
  Serial.begin(9600);
  while(!Serial);
  Serial.println("begin...");
  sht31.begin();
  M5.begin();      // Init M5Core.  [ ] [ ] M5Core
  M5.Power.begin(); // Init Power module. [ ] [ ] [ ] [ ] [ ] [ ]
}

void loop() {
  float temp = sht31.getTemperature();
  float hum = sht31.getHumidity();
  M5.Lcd.setFreeFont(FSB18);
  M5.Lcd.print("Temp = ");
  M5.Lcd.print(temp);
  M5.Lcd.println(" C");
  M5.Lcd.print("Hum = ");
  M5.Lcd.print(hum);
  M5.Lcd.println(" %");
  M5.Lcd.println();
  delay(1000);
}
```


}

Projet final

Séance 5 : Première idée de projet final

Nous avons commencé à réfléchir au projet que nous souhaiterions réaliser. Après être tombées sur [ce Tiktok](#), nous avons convenu d'inclure des capteurs de son et des LEDs dans notre projet. Nous voulions à l'origine tenter de reproduire l'objet de la vidéo, avant de vite nous rendre compte que c'était un objet encore bien trop complexe pour nos capacités. Nous avons alors cherché sur Internet des projets utilisant quand même des micros et des LEDs, tout en restant à notre niveau novice. En voici quelques-uns :

- [Music / hand controlled led strip with arduino](#)
- [Neopixel Ws2812 Rainbow LED Glow With M5stick-C](#)

À partir de là, nous avons décidé que le but de notre projet serait de contrôler la couleur et la luminosité de LEDs en fonction de la fréquence et l'intensité sonore (donc par exemple avec de la musique).

Nous avons commencé à travailler avec 3 composants :

- un [capteur de son](#) (micro constitué d'une membrane qui vibre en fonction de la pression acoustique et qui convertit les oscillations en signal électrique)
- un [capteur de sonie](#) (micro qui isole et amplifie les signaux à haute fréquence)
- une [barre de LED](#) (10 LEDs RGB indépendantes les unes des autres)

Dans un premier temps, nous avons testé un à un chacun des composants pour comprendre leur fonctionnement.

Nous avons réussi à faire fonctionner les LEDs grâce au [code de "Seedstudio"](#) :

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// released under the GPLv3 license to match the rest of the AdaFruit NeoPixel library

#include "Adafruit_NeoPixel.h"
```

```

#ifdef __AVR__
    #include <avr/power.h>
#endif

// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1
#define PIN        6

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS  10

// When we setup the NeoPixel library, we tell it how many pixels, and which pin to use to send signals.
// Note that for older NeoPixel strips you might need to change the third parameter--see the strandtest
// example for more information on possible values.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

int delayval = 500; // delay for half a second

void setup() {
    // This is for Trinket 5V 16MHz, you can remove these three lines if you are not using a Trinket
    #if defined (__AVR_ATtiny85__)
        if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
    #endif
    // End of trinket special code
    pixels.setBrightness(255);
    pixels.begin(); // This initializes the NeoPixel library.
}

void loop() {

    // For a set of NeoPixels the first NeoPixel is 0, second is 1, all the way up to the count of pixels minus one.

    for(int i=0;i<NUMPIXELS;i++){

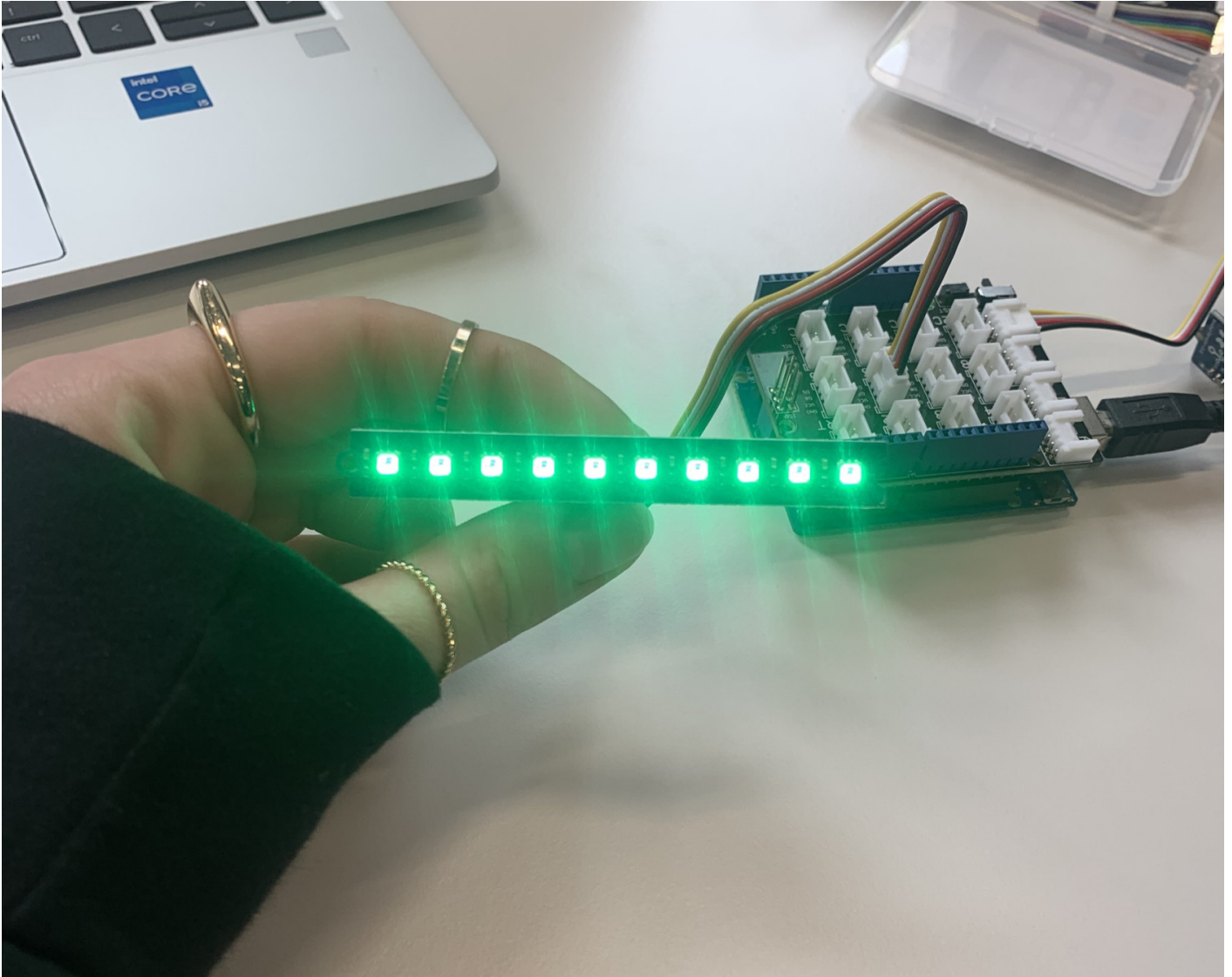
        // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
        pixels.setPixelColor(i, pixels.Color(0,150,0)); // Moderately bright green color.

        pixels.show(); // This sends the updated pixel color to the hardware.

        delay(delayval); // Delay for a period of time (in milliseconds).
    }
}

```

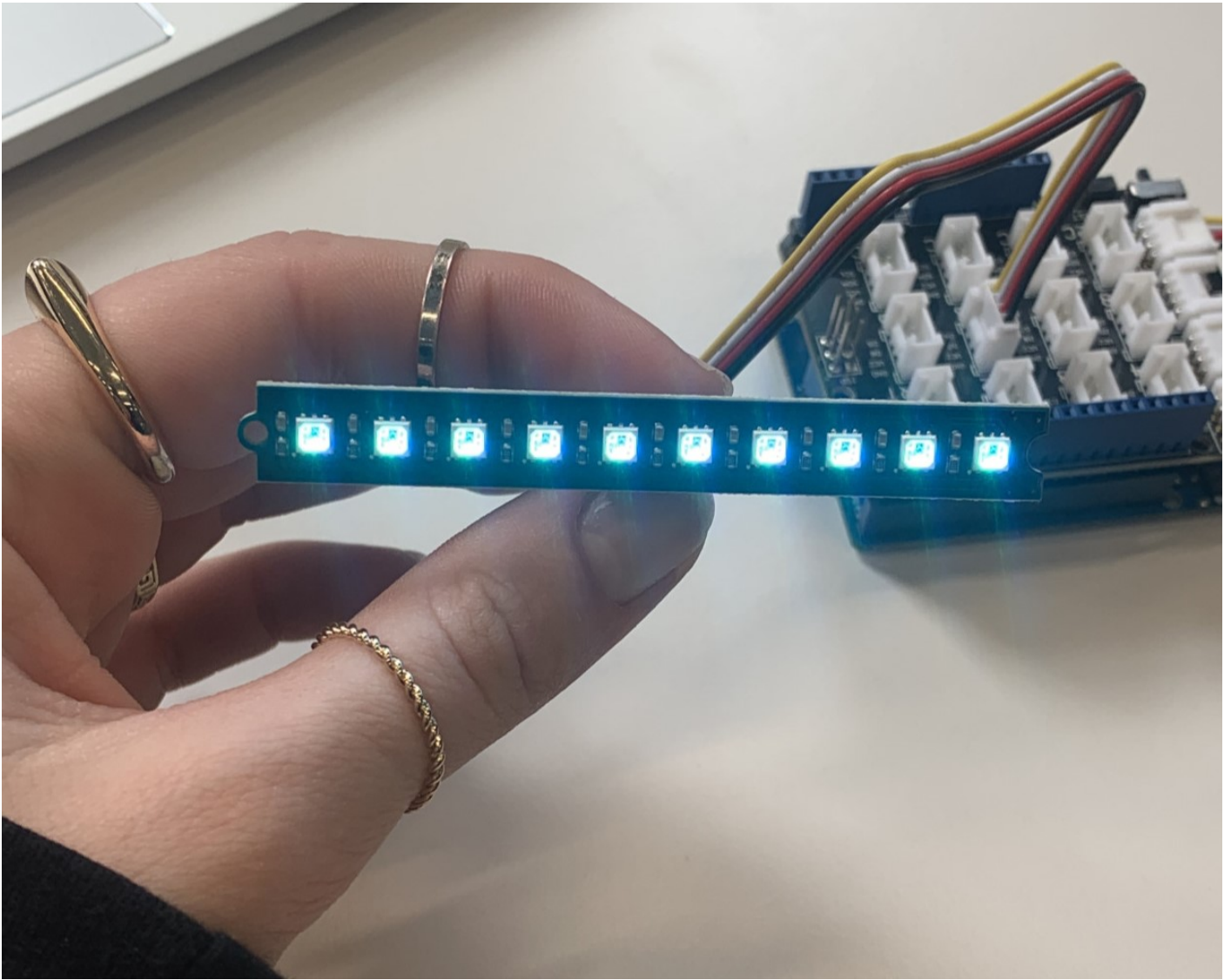
```
}  
}
```



En effectuant différentes modifications sur le code, nous avons pu baisser la luminosité des LED ainsi que changer leur couleur.

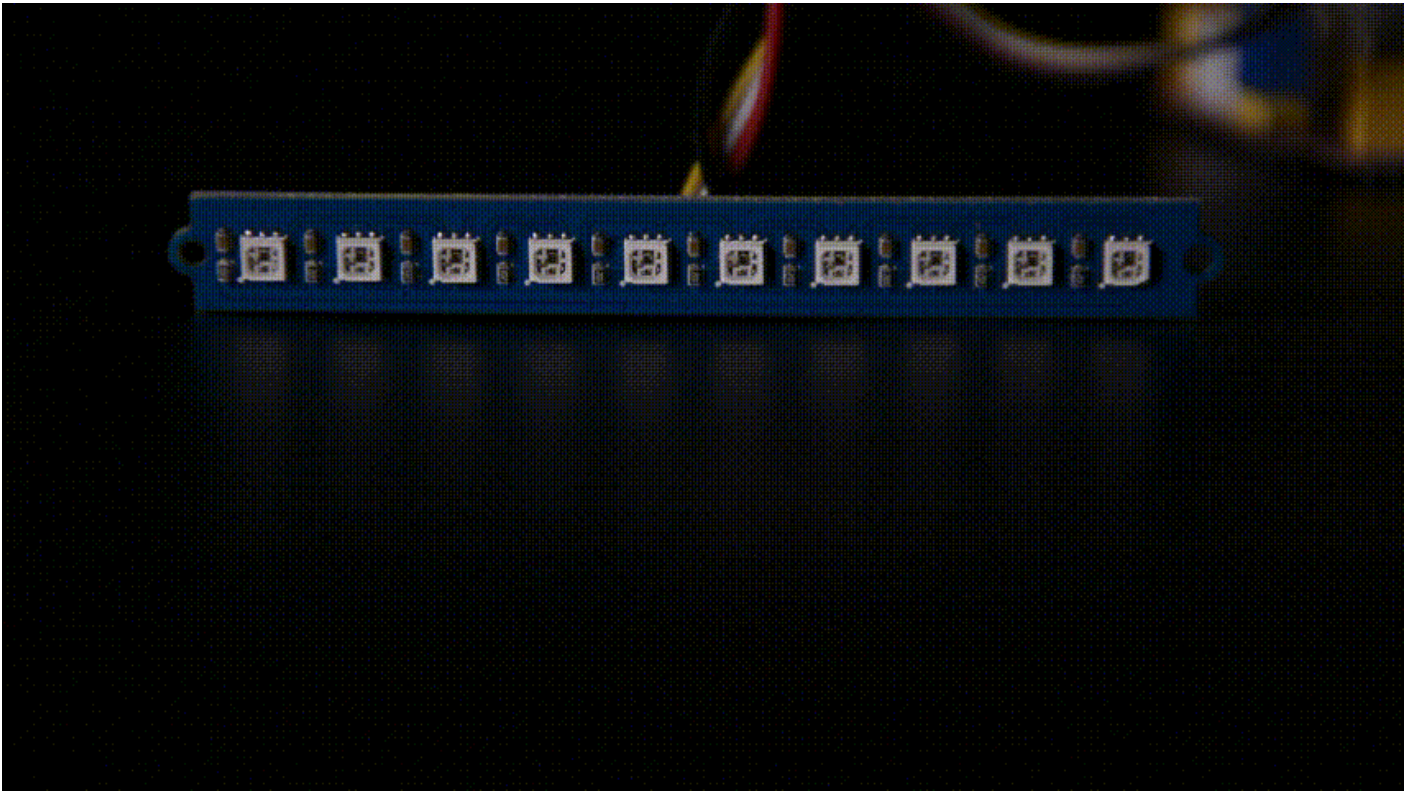
Pour changer la luminosité : remplacer 255 par une valeur entre 0 et 255 (ligne 29) (ici nous avons choisi 100)

Pour changer la couleur : remplacer 0,150,0 par 3 valeurs entre 0 et 255 (RGB) (ligne 40) (ici nous avons choisi (34,117,68))



Ces multiples manipulations nous permettent de mieux appréhender les fonctionnalités d'Arduino et renforcent nos compétences en informatique.

Par la suite, nous avons utilisé le code "[RGBWstrandtest](#)" afin d'afficher différentes couleurs sur les LED et obtenir un résultat similaire à celui-ci :



À force de chercher des projets qui ressemblent au nôtre afin de nous baser sur l'un d'entre eux comme point de départ, nous nous sommes rendues compte que cela n'allait pas être facile. En effet, même si l'idée de contrôler des LEDs avec un capteur sonore a été visitée et revisitée, les projets que l'on trouve sur Internet utilisent tous des composants différents des nôtres. Il est très certainement possible de créer son propre programme à partir de tous les projets proposés mais notre niveau et le temps qui nous était donné ne nous permettaient pas de réaliser cela. Ainsi, nous ne savions pas de quel point de départ partir afin de lancer notre projet.

Néanmoins, voici une tentative de faire réagir les LEDs aux données recueillies par le capteur de son :

```
#include "Adafruit_NeoPixel.h"
#ifdef __AVR__
    #include <avr/power.h>
#endif

#define PIN 6
#define NUMPIXELS 10
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(10, 6, NEO_GRB + NEO_KHZ800);
int loudness;

void setup()
{
    pixels.setBrightness(100);
```

```
pixels.begin();
pixels.show();
Serial.begin(9600);
}

void loop()
{
  loudness = analogRead(0);
  Serial.println(loudness);
  delay(200);
  for (int i = 0; i < NUMPIXELS; i++) {
    pixels.setPixelColor(i, pixels.Color(0, 150, 0));
    pixels.show(); // This sends the updated pixel color to the hardware.
    delay(500); // Delay for a period of time (in milliseconds).
  }
}
```

Séances 6 à 9 : Le sonomètre portatif

Nous nous sommes rendues compte que notre projet de base, qui était de faire réagir des LEDs à du son, nous paraissait au-delà de nos capacités. Nous avons donc décidé de partir sur un projet plus simple, en restant dans le thème de l'intensité sonore.

Notre projet est donc de réaliser un sonomètre portatif destiné aux étudiants qui ont tendance à passer beaucoup de temps dans les bars, boîtes de nuit, concerts, etc...

Programmation

Matériel :

- [M5Stack Core](#)
- [module PLUS](#)
- [capteur de son](#)

Le module PLUS permet de connecter des capteurs qui se branchent en analogie ou en numérique. Cela correspond au port noir (GPIO ou port B).

La toute première étape est de réussir à afficher les données recueillies par le capteur de son sur le M5Stack. Nous nous sommes donc servies d'un code permettant d'[afficher un spectre audio](#) trouvé sur le site Hackster.io. Il a fallu modifier la valeur de "micpin" (ligne 28) de 34 à 36 pour que les

données soient lues. Le code utilisé est le suivant :

```
/* ESP8266/32 Audio Spectrum Analyser on an SSD1306/SH1106 Display
 * The MIT License (MIT) Copyright (c) 2017 by David Bird.
 * The formulation and display of an Audio Spectrum using an ESP8266 or ESP32 and SSD1306 or SH1106 OLED
Display using a Fast Fourier Transform
 * Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated
documentation files
 * (the "Software"), to deal in the Software without restriction, including without limitation the rights to use,
copy, modify, merge,
 * publish, distribute, but not to use it commercially for profit making or to sub-license and/or to sell copies of the
Software or to
 * permit persons to whom the Software is furnished to do so, subject to the following conditions:
 * The above copyright notice and this permission notice shall be included in all copies or substantial portions of
the Software.
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING
BUT NOT LIMITED TO THE WARRANTIES
 * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE AUTHORS OR COPYRIGHT HOLDERS BE
 * LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 * See more at http://dsbird.org.uk
 */
// https://github.com/tobozo/ESP32-8-Octave-Audio-Spectrum-Display/tree/wrover-kit
// https://github.com/G6EJD/ESP32-8266-Audio-Spectrum-Display
// https://github.com/kosme/arduinoFFT

#include "arduinoFFT.h"      // Standard Arduino FFT library
arduinoFFT FFT = arduinoFFT();
#include <M5Stack.h>
#define SAMPLES 512          // Must be a power of 2
#define SAMPLING_FREQUENCY 40000
// Hz, must be 40000 or less due to ADC conversion time.
// Determines maximum frequency that can be analysed by the FFT  $F_{max} = \text{sampleF}/2$ .

int micpin = 36; //change this to 36 if you are using the fc-04

struct eqBand {
```

```

const char *freqname;
uint16_t amplitude;
int peak;
int lastpeak;
uint16_t lastval;
unsigned long lastmeasured;
};

eqBand audiospectrum[8] = {
    //Adjust the amplitude values to fit your microphone
    { "125Hz", 500, 0, 0, 0, 0},
    { "250Hz", 200, 0, 0, 0, 0},
    { "500Hz", 200, 0, 0, 0, 0},
    { "1KHz", 200, 0, 0, 0, 0},
    { "2KHz", 200, 0, 0, 0, 0},
    { "4KHz", 100, 0, 0, 0, 0},
    { "8KHz", 100, 0, 0, 0, 0},
    { "16KHz", 50, 0, 0, 0, 0}
};

unsigned int sampling_period_us;
unsigned long microseconds;
double vReal[SAMPLES];
double vImag[SAMPLES];
unsigned long newTime, oldTime;
uint16_t tft_width = 320; // ILI9341_TFTWIDTH;
uint16_t tft_height = 240; // ILI9341_TFTHEIGHT;
uint8_t bands = 8;
uint8_t bands_width = floor( tft_width / bands );
uint8_t bands_pad = bands_width - 10;
uint16_t colormap[255]; // color palette for the band meter (pre-fill in setup)

void setup() {
    M5.begin();
    dacWrite(25, 0); // Speaker OFF
    M5.Lcd.fillScreen(TFT_BLACK);
    M5.Lcd.setTextColor(YELLOW, BLACK);
    M5.Lcd.setTextSize(1);
    M5.Lcd.setRotation(1);
    sampling_period_us = round(1000000 * (1.0 / SAMPLING_FREQUENCY));

```



```

delay(2000);
for(uint8_t i=0;i<tft_height;i++) {
    colormap[i] = M5.Lcd.color565(tft_height-i*.5, i*1.1, 0);
}
for (byte band = 0; band <= 7; band++) {
    M5.Lcd.setCursor(bands_width*band + 2, 0);
    M5.Lcd.print(audiospectrum[band].freqname);
}
}

void loop() {
    for (int i = 0; i < SAMPLES; i++) {
        newTime = micros()-oldTime;
        oldTime = newTime;
        vReal[i] = analogRead(micpin); // A conversion takes about 1uS on an ESP32
        vImag[i] = 0;
        while (micros() < (newTime + sampling_period_us)) {
            // do nothing to wait
        }
    }
    FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
    FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);
    FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);

    for (int i = 2; i < (SAMPLES/2); i++){
        // Don't use sample 0 and only first SAMPLES/2 are usable.
        // Each array element represents a frequency and its value the amplitude.
        if (vReal[i] > 1500) { // Add a crude noise filter, 10 x amplitude or more
            byte bandNum = getBand(i);
            if(bandNum!=8) {
                displayBand(bandNum, (int)vReal[i]/audiospectrum[bandNum].amplitude);
            }
        }
    }
}

long vnow = millis();
for (byte band = 0; band <= 7; band++) {
    // auto decay every 50ms on low activity bands
    if(vnow - audiospectrum[band].lastmeasured > 50) {
        displayBand(band, audiospectrum[band].lastval>4 ? audiospectrum[band].lastval-4 : 0);
    }
}

```

```

}
if (audiospectrum[band].peak > 0) {
    audiospectrum[band].peak -= 2;
    if(audiospectrum[band].peak<=0) {
        audiospectrum[band].peak = 0;
    }
}

// only draw if peak changed
if(audiospectrum[band].lastpeak != audiospectrum[band].peak) {
    // delete last peak
    M5.Lcd.drawFastHLine(bands_width*band,tft_height-audiospectrum[band].lastpeak,bands_pad,BLACK);
    audiospectrum[band].lastpeak = audiospectrum[band].peak;
    M5.Lcd.drawFastHLine(bands_width*band, tft_height-audiospectrum[band].peak,
        bands_pad, colormap[tft_height-audiospectrum[band].peak]);
}
}
}

void displayBand(int band, int dsize){
    uint16_t hpos = bands_width*band;
    int dmax = 200;
    if(dsize>tft_height-10) {
        dsize = tft_height-10; // leave some hspace for text
    }
    if(dsize < audiospectrum[band].lastval) {
        // lower value, delete some lines
        M5.Lcd.fillRect(hpos, tft_height-audiospectrum[band].lastval,
            bands_pad, audiospectrum[band].lastval - dsize, BLACK);
    }
    if (dsize > dmax) dsize = dmax;
    for (int s = 0; s <= dsize; s=s+4){
        M5.Lcd.drawFastHLine(hpos, tft_height-s, bands_pad, colormap[tft_height-s]);
    }
    if (dsize > audiospectrum[band].peak) {
        audiospectrum[band].peak = dsize;
    }
    audiospectrum[band].lastval = dsize;
    audiospectrum[band].lastmeasured = millis();
}

```

```

byte getBand(int i) {
  if (i<=2 )          return 0; // 125Hz
  if (i >3  && i<=5 ) return 1; // 250Hz
  if (i >5  && i<=7 ) return 2; // 500Hz
  if (i >7  && i<=15 ) return 3; // 1000Hz
  if (i >15 && i<=30 ) return 4; // 2000Hz
  if (i >30 && i<=53 ) return 5; // 4000Hz
  if (i >53 && i<=200 ) return 6; // 8000Hz
  if (i >200      ) return 7; // 16000Hz
  return 8;
}

```

Ce programme permet donc de visualiser un spectre audio en fonction de la fréquence, or nous cherchons à visualiser l'intensité sonore. Même si nous n'avons pas utilisé ce programme par la suite, il nous a permis de comprendre comment définir le capteur dans Arduino (**pin 36**).

L'étape suivante est de configurer plusieurs réactions du M5Stack en fonction des données sonores. Le but est d'afficher différents messages de prévention pour informer l'utilisateur de son exposition au bruit.

Le code suivant permet d'afficher le message "attention" lorsque la valeur récoltée par le capteur est supérieure à 1000. Cette valeur est choisie arbitrairement pour l'instant.

Ce programme suivant est construit à partir du code de test du [capteur de son](#), ainsi que de quelques recherches qui nous ont permis d'inclure une condition "if". La valeur de la condition choisie est ici totalement arbitraire vu qu'on cherche uniquement à ce que le programme marche.

```

#include <M5Stack.h>
const int sensor = 36;

void setup()
{
  Serial.begin(115200);
  M5.begin();
  M5.Power.begin();
}

void loop()
{
  long sum = 0;
  for(int i=0; i<32; i++)

```

```

{
    sum += analogRead(sensor); // permet de lire la valeur relevée par capteur
    sum >>= 2; // divise les valeurs par un certain facteur pour ne pas avoir de trop grandes valeurs
}
Serial.println(sum); // écrit la valeur dans le plotter

if (sum > 1000)
{
    M5.Lcd.clear();
    M5.Lcd.print("attention");
}

delay(10);

}

```

Le but de notre objet est d'afficher différents messages de prévention en fonction de l'intensité sonore. Il faut donc imposer plusieurs conditions.

Les niveaux d'intensité sonore sont indiqués en décibels et correspondent à des temps d'exposition hebdomadaires. Dans une boîte de nuit, l'intensité sonore varie **entre 85 et 115 dB**. Voici les messages affichés par le M5Stack en fonction de l'intensité sonore :

- **96 dB (5h d'exposition) :** "Tu peux encore chiller pendant 5h"
- **103 dB (1h) :** "Plus qu'une heure avant l'extinction des feux"
- **105 dB (10 min) :** "Le temps d'une petite clope et au revoir"
- **115 dB (5 min) :** "Prends tes affaires et sors ;)"

Pour imposer différentes conditions au M5Stack, il suffit de reprendre le programme ci-dessus et d'y ajouter de nouveaux opérateurs "if".

```

#include <M5Stack.h>
#include "Free_Fonts.h"
const int sensor = 36;

void setup()
{
    Serial.begin(115200);
    M5.begin();
    M5.Power.begin();
}

void loop()

```

```
{
while(1)
{
    long sum = 0;
    for(int i=0; i<32; i++)
    {
        sum += analogRead(sensor); // permet de lire la valeur relevée par le capteur
        sum >>= 2; // divise la valeur par un certain facteur pour ne pas avoir des valeurs trop grandes
    }
    Serial.println(sum); // écrit la valeur dans le plotter

    if (sum < 100) //en dessous de 100 dB
    {
        M5.Lcd.clear();
        M5.Lcd.print("Tu peux encore chiller pendant 5h");
        delay(3000);
    }

    if (sum >= 100 && sum < 500) //entre 100 et 105 dB
    {
        M5.Lcd.clear();
        M5.Lcd.print("Plus qu'une heure avant l'extinction des feux");
        delay(3000);
    }

    if (sum >= 500 && sum < 1000) //entre 105 et 115 dB
    {
        M5.Lcd.clear();
        M5.Lcd.print("Le temps d'une petite clope");
        delay(3000);
    }

    if (sum >= 1000) //au dessus de 115 dB
    {
        M5.Lcd.clear();
        M5.Lcd.print("Prends tes affaires et sors ;)");
        delay(3000);
    }
}
```

```
    delay(10);  
  }  
  
}
```

Ici, les valeurs des conditions "if" ont encore une fois été choisies arbitrairement. Pour que l'objet soit réellement fonctionnel, il aurait fallu déterminer quelle plage de valeurs lues sur l'ordinateur correspond à quelle intensité sonore. On aurait pu utiliser une source de son de référence, dont on connaît l'intensité sonore, afin de "calibrer" notre capteur. Par souci de temps, nous n'avons pas pu réaliser cela.

Il faut maintenant s'occuper de la mise en forme du message affiché. Nous avons décidé de changer la couleur de l'écran en fonction de la dangerosité du niveau sonore. Nous avons aussi changer la police ainsi que la taille de la police. Toutes les modifications liées à la mise en forme sont inspirées des programmes que l'on peut trouver dans la bibliothèque du M5Stack, comme "[Free Fonts](#)" ou "[Display](#)".

Voici le programme final que nous avons utilisé :

```
#include <M5Stack.h>  
#include "Free_Fonts.h"  
const int sensor = 36;  
  
void setup()  
{  
  Serial.begin(115200);  
  M5.begin();  
  M5.Power.begin();  
  M5.Lcd.setBrightness(100);  
}  
  
void loop()  
{  
  while(1)  
  {  
    long sum = 0;  
    for(int i=0; i<32; i++)  
    {  
      sum += analogRead(sensor);  
      sum >>= 2;  
    }  
  }  
}
```

```

Serial.println(sum);

if (sum < 100) //en dessous de 100 dB
{
    M5.Lcd.clear();
    //uint16_t colorvalue = 0;
    //colorvalue = color565(0, 153, 0);
    M5.Lcd.fillScreen(GREEN); //colorvalue(0,153,0)
    M5.Lcd.drawRect(20, 20, 280, 200, WHITE); //cadre
    M5.Lcd.setTextColor(TFT_WHITE); //couleur du texte
    M5.Lcd.setTextSize(2);
    M5.Lcd.drawString("Tu peux encore chiller", 70, 108, 2);
    M5.Lcd.drawString("pendant 5h", 94, 132, 2);
    delay(3000);
}

if (sum >= 100 && sum < 500) //entre 100 et 105 dB
{
    M5.Lcd.clear();
    M5.Lcd.fillScreen(YELLOW); //color565(255,213,0)
    M5.Lcd.drawRect(20, 20, 280, 200, BLACK); //cadre
    M5.Lcd.setTextColor(TFT_BLACK); //couleur du texte
    M5.Lcd.setTextSize(2);
    M5.Lcd.drawString("Plus qu'une heure avant", 70, 110, 2);
    M5.Lcd.drawString("l'extinction des feux", 90, 130, 2);
    delay(3000);
}

if (sum >= 500 && sum < 1000) //entre 105 et 115
{
    M5.Lcd.clear();
    M5.Lcd.fillScreen(ORANGE); //color565(255,162,0)
    M5.Lcd.drawRect(20, 20, 280, 200, WHITE); //cadre
    M5.Lcd.setTextColor(TFT_WHITE); //couleur du texte
    M5.Lcd.setTextSize(2);
    M5.Lcd.drawString("Le temps d'une petite clope", 160, 112, 2);
    M5.Lcd.drawString("et au revoir", 160, 128, 2);
    delay(3000);
}

```

```
if (sum >= 1000) //au dessus de 115 dB
{
    M5.Speaker.tone(1000, 500);
    M5.Lcd.clear();
    M5.Lcd.fillScreen(RED);
    M5.Lcd.drawRect(20, 20, 280, 200, WHITE); //cadre
    M5.Lcd.setTextDatum(MC_DATUM); //texte au centre
    M5.Lcd.setTextColor(TFT_WHITE); //couleur du texte
    M5.Lcd.setTextSize(2);
    M5.Lcd.drawString("Prends tes affaires et sors ;)", 160, 120, 2);
    M5.Speaker.end();
    delay(3000);
}

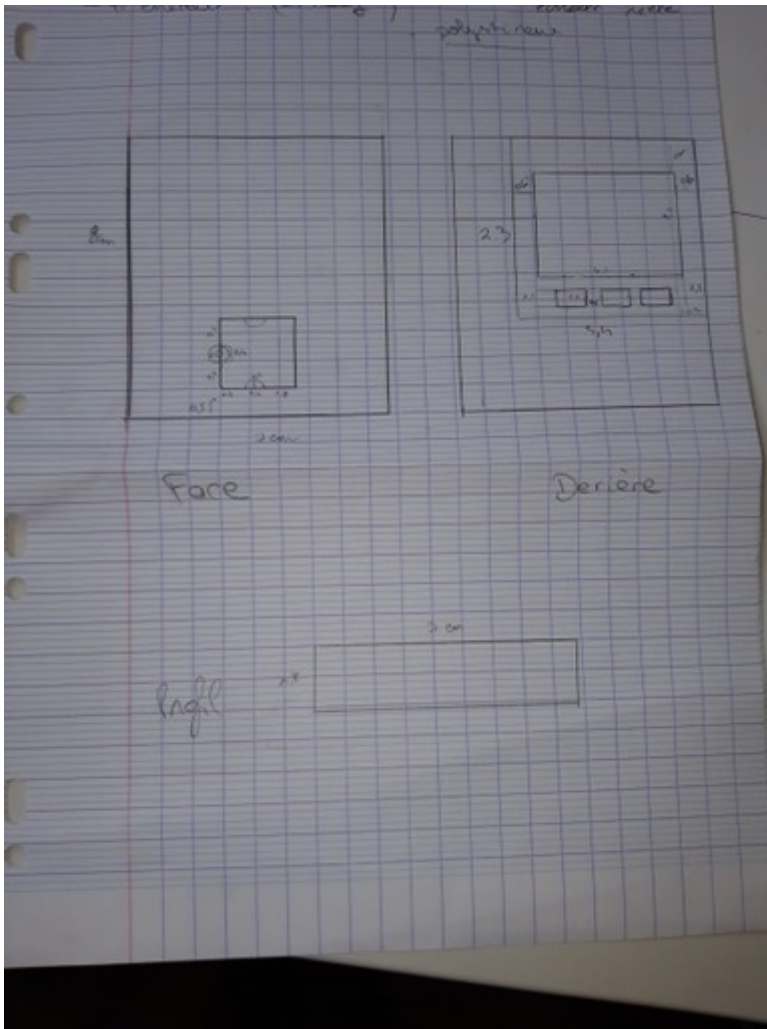
delay(10);
}

}
```

Modélisation 3D

Shirel s'est chargée de concevoir et dessiner le packaging de notre objet. Elle a mesuré toutes les longueurs sur un M5stack de référence et les a reportées. On a donc choisi de créer un boîtier imprimé grâce à une imprimante 3D. Le choix d'une impression 3D s'est imposé de lui-même. L'objet semble plus design, plus adapté à des jeunes. En découpe laser, il semblerait plus rustique (ce qui n'est pas forcément mauvais, mais ce n'est pas le but recherché).

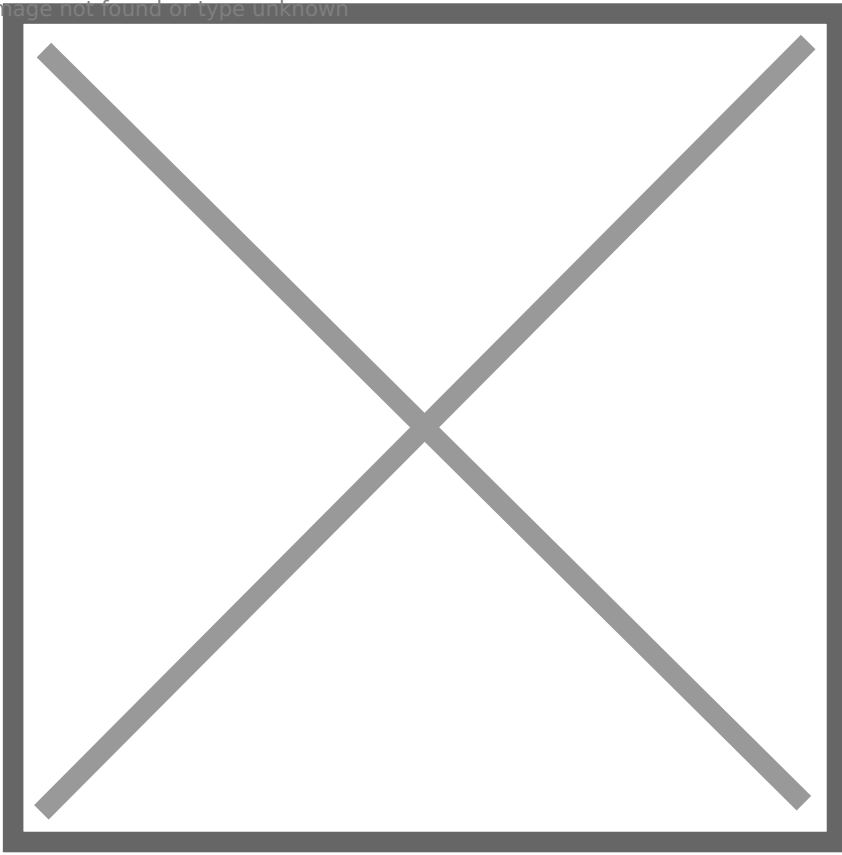
On ferait une sorte d'ouverture permettant de laisser visible l'écran du M5 afin d'afficher les messages d'alerte.



Lily a commencé à modéliser sur FreeCAD la boîte qui servira à contenir le M5stack et le capteur.

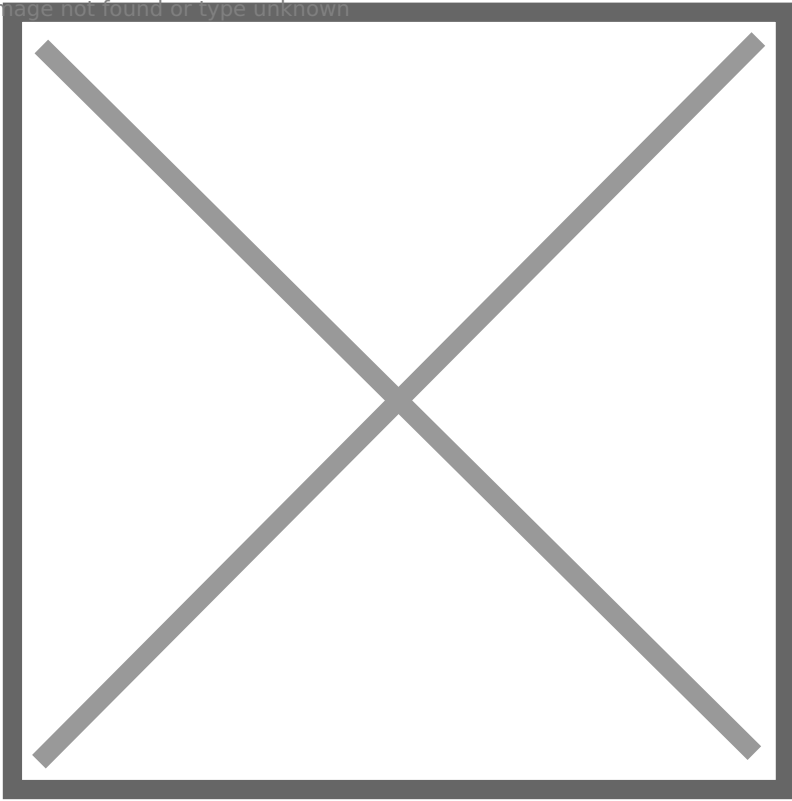
Donc dans un premier temps, j'ai simplement créé un nouveau sketch dans la partie Part design. J'ai appliqué à ce sketch différentes contraintes de longueur et de symétrie pour qu'il soit dans les proportions que nous souhaitions. Par la suite, j'ai entré une protrusion afin d'obtenir une rectangle en 3D. Afin de créer une boîte, il a fallu utiliser l'option "générer une coque solide" en sélectionnant la surface du dessus. Évidemment on modifie l'épaisseur de la protrusion afin que le boîte corresponde à nos mesures.

Image not found or type unknown



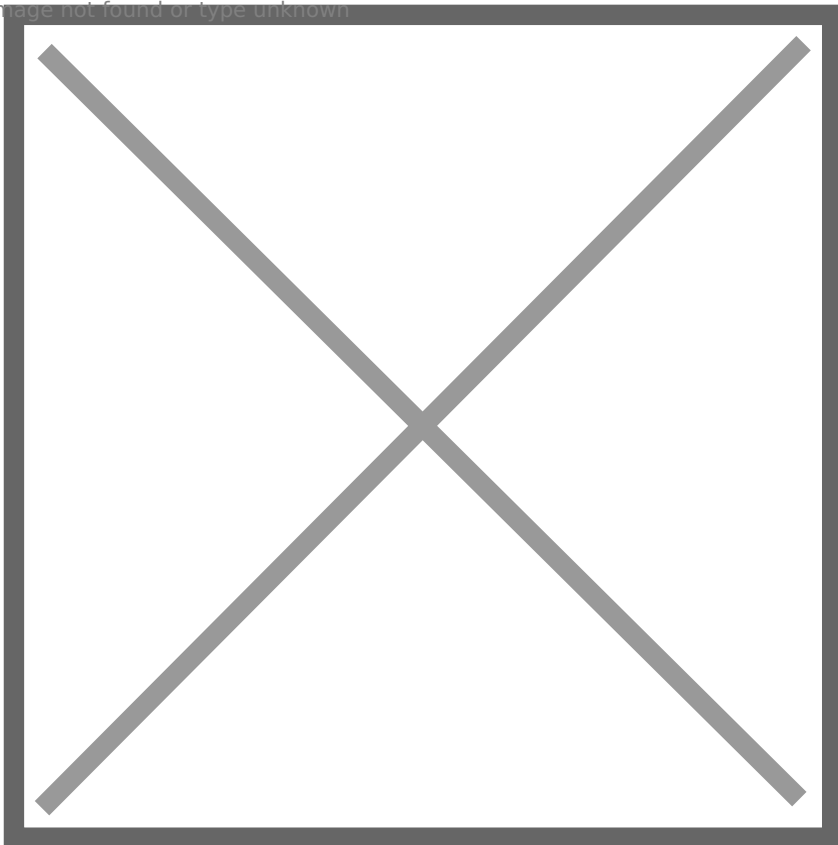
Par la suite, il a été question de dessiner le profil de l'ergo. Pour se faire, il faut créer un nouveau sketch en sélectionnant le plan XZ pour avoir la face souhaitée de profil. On fait donc "créer une arête liée" afin d'avoir un point fixé au centre de l'arête de la boîte. Puis on peut tracer notre ergo en poly ligne. Il suffit simplement d'effectuer un triangle rectangle, auquel encore une fois, on applique des contraintes de longueur. Afin de mieux voir les arêtes et mieux me repérer, je suis passée en représentation filaire.

Image not found or type unknown



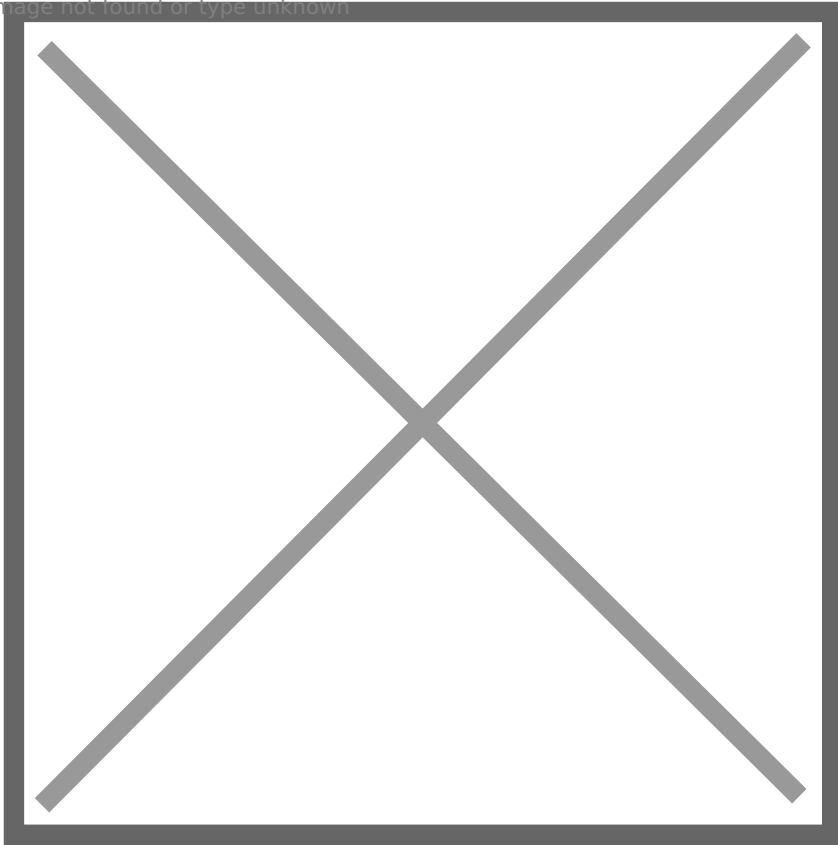
À partir de ce sketch, on va donc à nouveau créer une protrusion symétrique au plan, c'est-à-dire centrée sur l'arête. Par ailleurs, il est toujours important de centrer les choses par rapport aux axes de coordonnées, le wiki du fablab le conseil notamment.

Image not found or type unknown

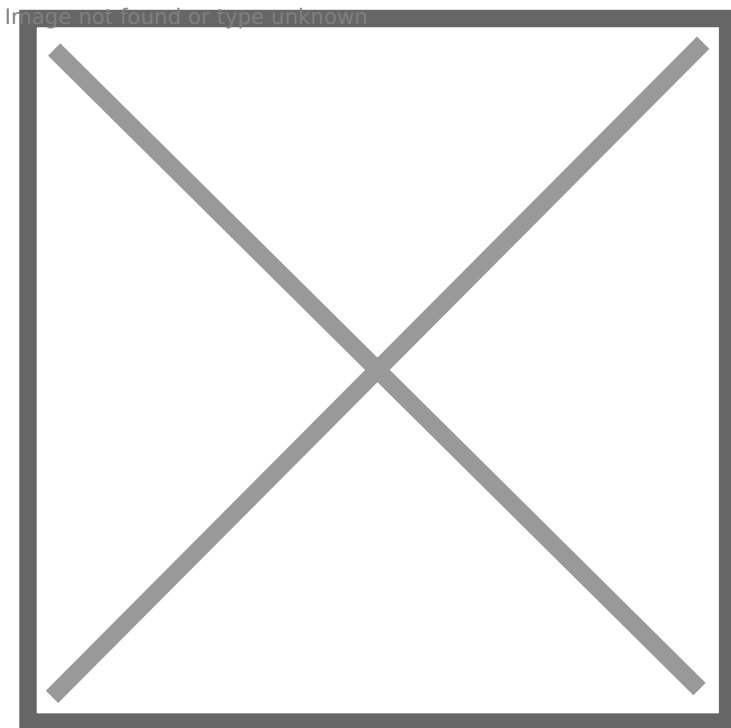
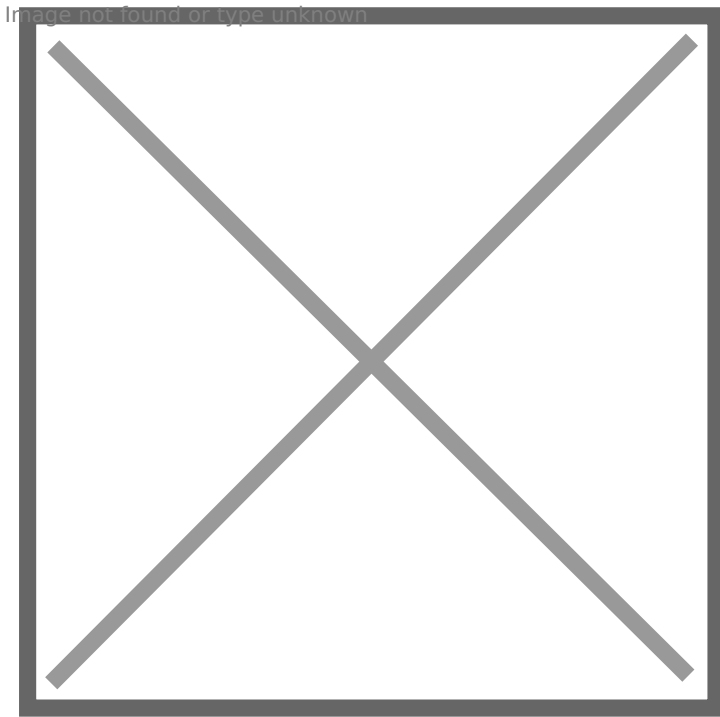


Par la suite, pour me faciliter la tâche, j'ai simplement "créé une fonction de symétrie" afin de dupliquer mon ergo de l'autre côté.

Image not found or type unknown

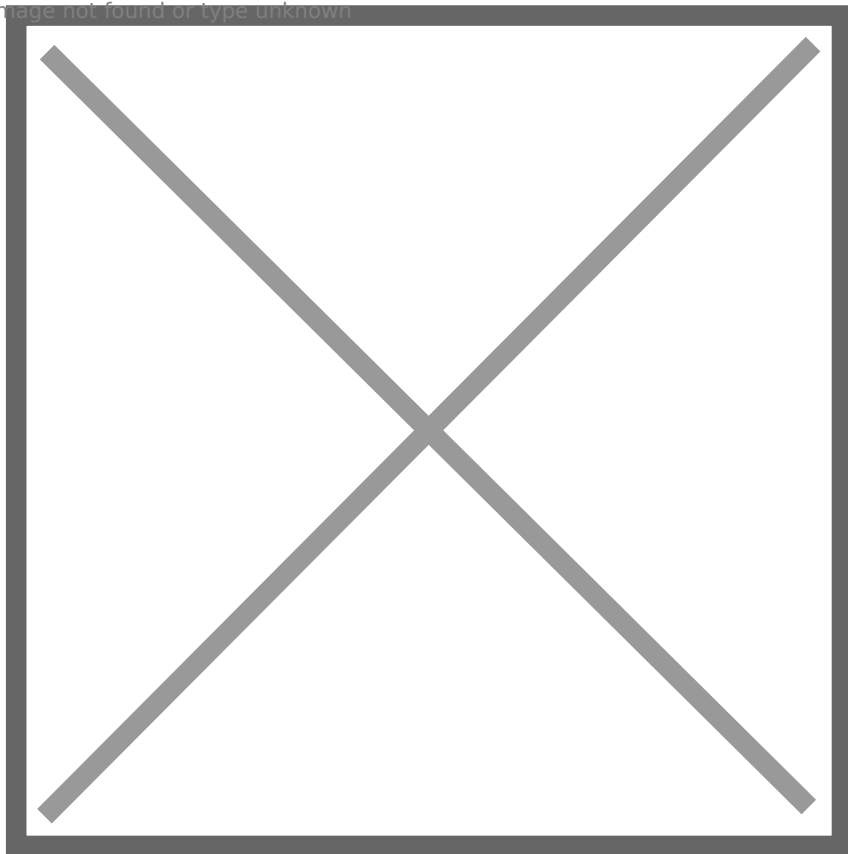


Pour ce qui est du couvercle, j'avais préalablement dupliqué le fond de ma boîte pour éviter de refaire un rectangle, puis une protrusion etc... J'ai donc simplement modifié les dimensions en changeant les contraintes de longueur et largeur. En effet, afin que le fond et le couvercle de la boîte se clipsent entre eux à la fin, il faut que mon couvercle soit légèrement plus grand que le fond. J'ai donc laissé 0,5 mm de chaque côté afin d'avoir de la marge.



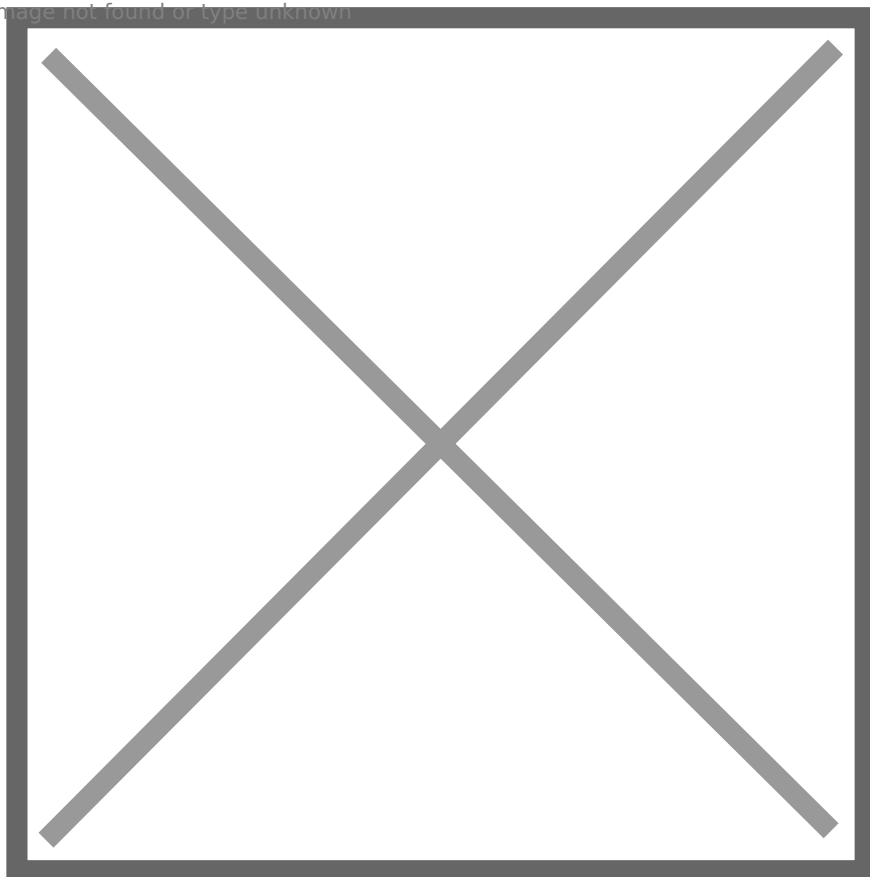
Afin de créer l'ergo du couvercle, j'ai suivi exactement les mêmes étapes que précédemment en créant un nouveau sketch, créer une arête liée, faire un triangle rectangle en poly ligne etc... La seule différence est, qu'afin que les deux parties de la boîte se clipsent, il est nécessaire que l'ergo du couvercle soit dirigé vers l'intérieur au lieu de l'extérieur. Donc au lieu de créer une protrusion, je crée une cavité. De plus, ses dimensions doivent être légèrement plus grandes (0,5mm de plus) afin d'avoir de la marge.

Image not found or type unknown



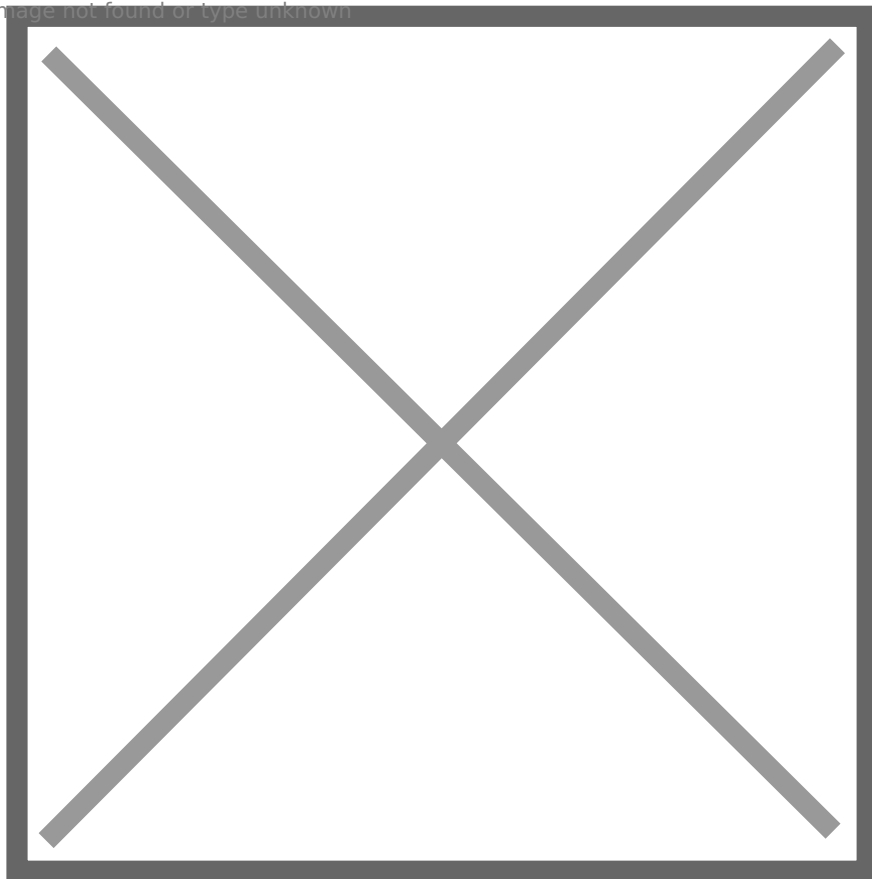
Pareil que pour le fond, je crée une fonction de symétrie pour avoir cet ergo des deux côtés.

Image not found or type unknown



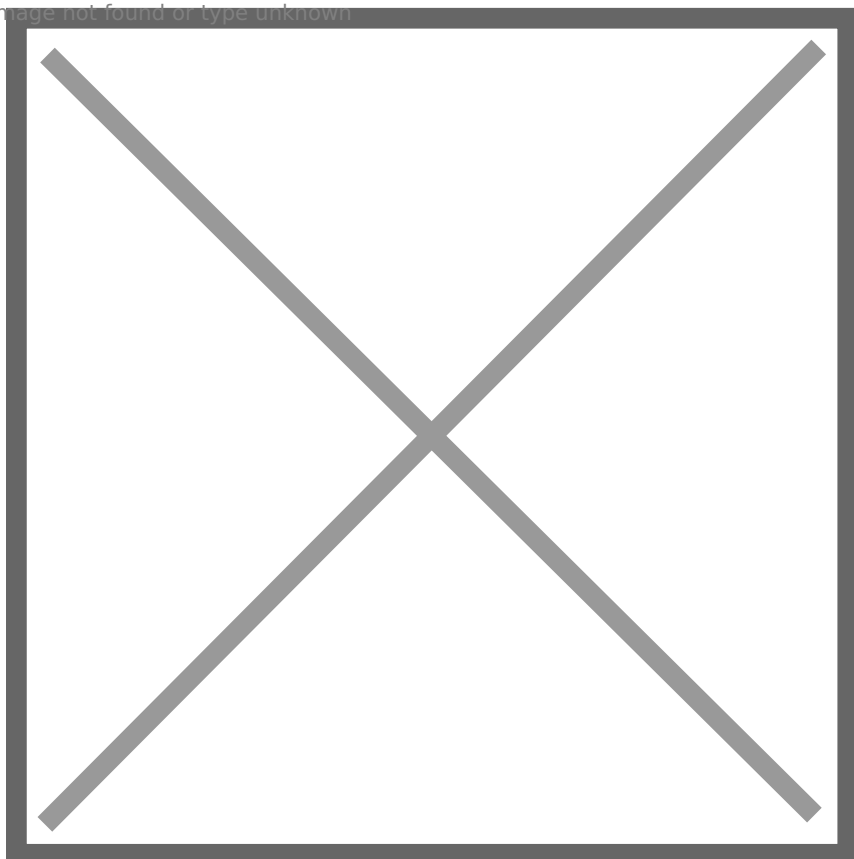
Voici donc le rendu final de la boîte.

Image not found or type unknown

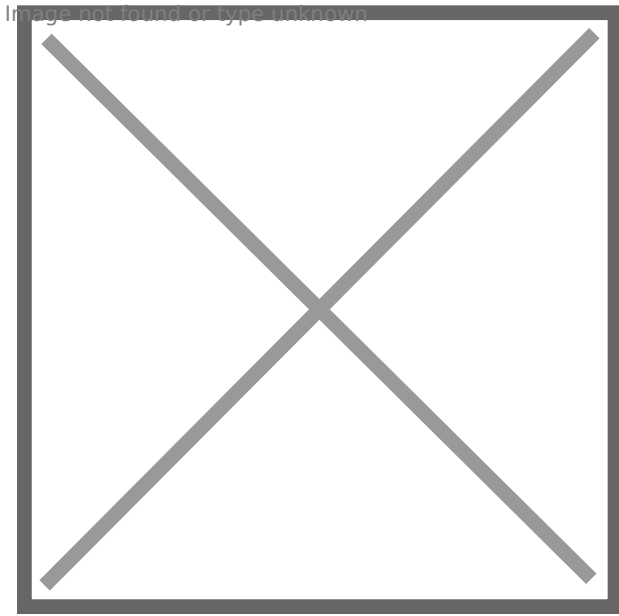


Afin de ne pas avoir beaucoup de support lors de l'impression, j'ai superposé les deux parties sur le même plan.

Image not found or type unknown



J'ai donc pu passer à l'impression 3D en enregistrant mon fichier dans le modèle qui convient pour FreeCAD. Une première impression a été faite mais malheureusement, on a eu un problème avec la machine.



L'impression finale a prit 6 heures au total et on a pu obtenir un rendu convaincant. Malheureusement, je pense que je n'avais pas pris assez de marge au niveau des dimensions du couvercle car la boîte avait du mal à se fermer, notamment avec le M5Stack dedans.

On a subi un autre échec quant à la volonté de creuser un trou dans le couvercle pour que l'on puisse voir l'écran du M5Stack à travers la boîte. L'application FreeCAD ne me permet pas d'appliquer une contrainte de longueur entre le haut de la boîte et le haut de mon rectangle. Je n'ai donc pas réussi à placer mon rectangle où je le souhaitais sur le couvercle. Cette idée a donc été abandonnée.

Image not found or type unknown

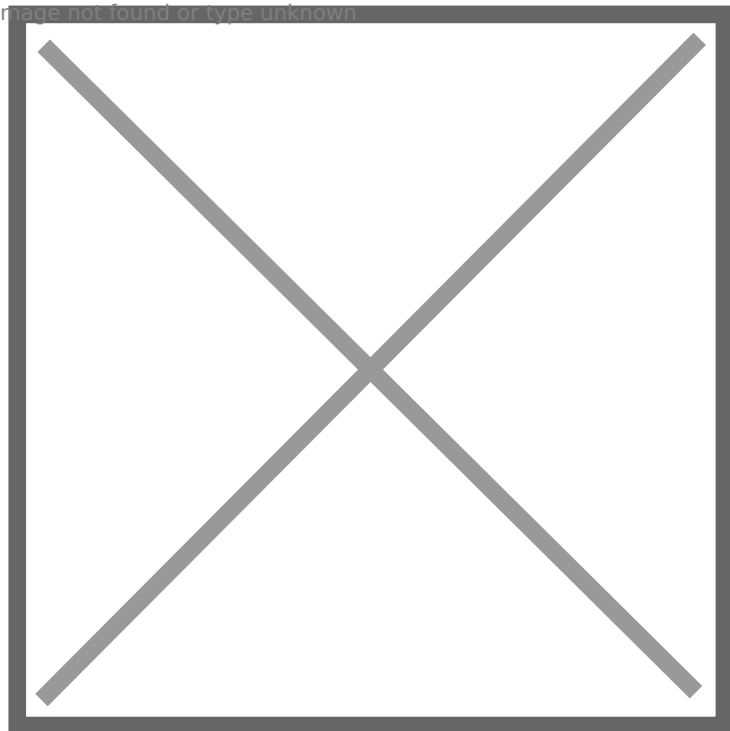
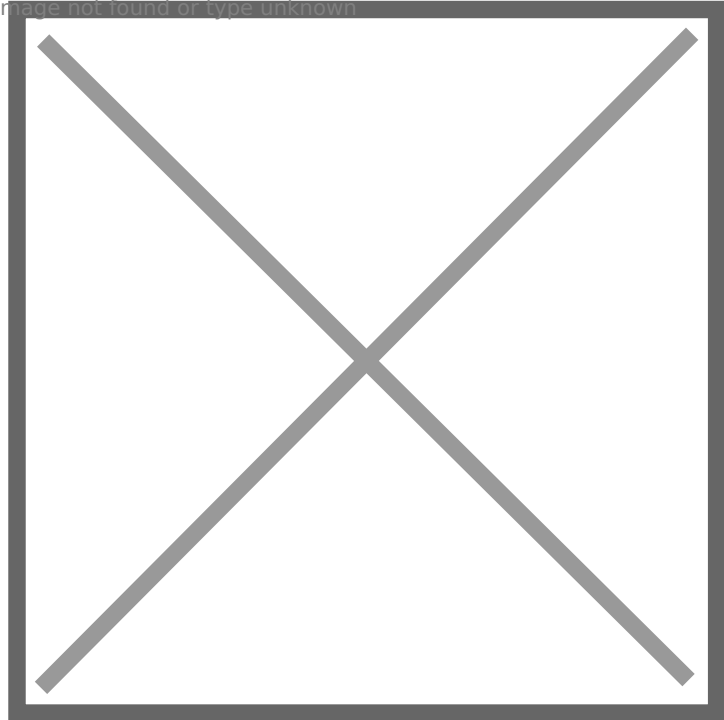


Image not found or type unknown



Fonctionnement de l'objet et possibles améliorations

Pour s'assurer de son bon fonctionnement, nous avons effectué différents tests sur notre objet. Nous avons essayé en parlant et en mettant de la musique à différents volumes. Les messages affichés par le M5Stack sont bien cohérents avec les tests effectués mais on remarque tout de même que l'objet ne semble parfois pas capter correctement les ondes sonores. Par exemple, même en collant un téléphone avec du son très fort au sonomètre, le message affiché reste celui

où il n'y a pas de danger.

De plus, notre sonomètre n'est pas très efficace car la réelle dangerosité du son est lorsque l'on reste exposé.e pendant une certaine durée. Or ici, le message de prévention s'affiche lorsque le capteur recueille une seule valeur supérieure à la limite à un instant t précis, et non sur une certaine durée. Il est possible de visualiser le message "Prends tes affaires et sors ;)" puis quelques secondes plus tard, "Plus qu'une heure avant l'extinction des feux", ce qui n'a aucun sens pour l'utilisateur.

Pour régler ce problème, il faudrait ajouter à chaque condition "if" une condition de temps, où le message s'afficherait après que le capteur ait enregistré un certain nombre de valeurs dépassant la limite imposée à des intervalles de temps petits.

En ce qui concerne la boîte, il aurait aussi fallu fixer le capteur dedans et créer un trou pour que la paroi ne fasse pas office de barrière du son, qui viendrait diminuer et fausser les valeurs.

Conclusion générale du projet

Cette UE est construite de manière à ce que chaque séance soit indispensable pour la création de notre objet final. Nous avons réussi à faire en sorte que le sonomètre capte un son à une intensité sonore donnée. Cependant avec plus de temps, nous aurions pu trouver le bon code, le code qui permettrait par exemple d'alerter au 110 dB exactement atteints.

De plus, nous pouvions également améliorer toute la partie modélisation. En outre, notre boîte ne se ferme pas correctement. Si on continuait l'UE, on recommencerait la modélisation avec plus de marge dans les impressions.

Finalement, le choix d'une impression 3D, ne fut pas très judicieux. Si on avait réalisé la boîte à la découpe laser, la fabrication aurait été plus rapide. Des ajustements seraient encore possibles, si nous détectons des erreurs, bien que le design recherché est au rendez-vous.

Rendu final

Messages

Tu peux encore
chiller pendant 5h

Plus qu'une heure
avant l'extinction
des feux

Le temps d'une petite
clope et au revoir



Séance 10 : Présentation des projets

Lors de cette séance, nous avons assisté à la présentation des projets de chaque groupe.

Voici un lien vers le support de notre présentation : [projet fablab - sonomètre portatif](#)

Projets personnels

Mailys Hermann--Boyer

Réalisation d'une étoile à l'imprimante 3D & de jaeminbun à la découpe-laser

page de documentation : [Étoile & jaeminbun](#)

Shirel Fellous

[2D et 3D Shirel](#)

Lily-Rose Gavanon

[Modélisations](#)

Revision #104

Created 22 January 2023 14:25:45 by Vincent Dupuis

Updated 22 April 2023 08:39:53 by Hermann--Boyer Mailys