



MoodBoxByFaBteam

Modular box: minimalist weather station and jukebox with Arduino

 Difficulté **Moyen**

 Durée **100 heure(s)**

 Catégories **Art, Décoration, Électronique, Musique & Sons**

 Coût **50 EUR (€)**

Sommaire

Introduction

Preamble

Overall specifications

Starting the box

Video d'introduction

Étape 1 - 2D design (new tier for the box)

Étape 2 - 3D design

Étape 3 - Laser engraving ad cutting of the new box tier

Étape 4 - 3D-printing

Étape 5 - Wiring diagram

Input

Output

Motor

Power supply

Étape 6 - Arduino programming

Libraries

Étape 7 - Assembling, testing and finishing off

Étape 8 - Now let's dance!

Notes et références

Commentaires

Introduction

This modular box is a 3-tier set. Once on, it provides basic weather station information (temperature, humidity, atmospheric pressure) but becomes a "Mood box", here a jukebox actually, when properly activated.

Preamble

This project is a group project (FaB team) carried out during a French hybrid training program called "Fabrication numérique" (CNC machining and fabrication) - class of 2018, July.

This final group project involves some of the know-how acquired during the training program and seals the deal. It is complementary with the "Bentolux" project which should be documented and referenced soon on wikifab.org.

Instructions for the project were:

- create a new tier to the box designed and developed during the course of the training program (3 available tiers: 1 base, 1 in Plexiglas, 1 for LCD screen);
- use newly acquired knowledge for this new tier: 3D printing, laser cutting, etc.;
- program with Arduino at least one user interaction (free choice).

Overall specifications

After some thought our group decided to create a musical tier - we called it "Mood box" - to complement the weather station using the base-tier and the LCD-tier.

Expected behavior is described here below.

Starting the box

When properly plugged in, the box can be started by using the switch on the front.

Starting the box triggers the following events:

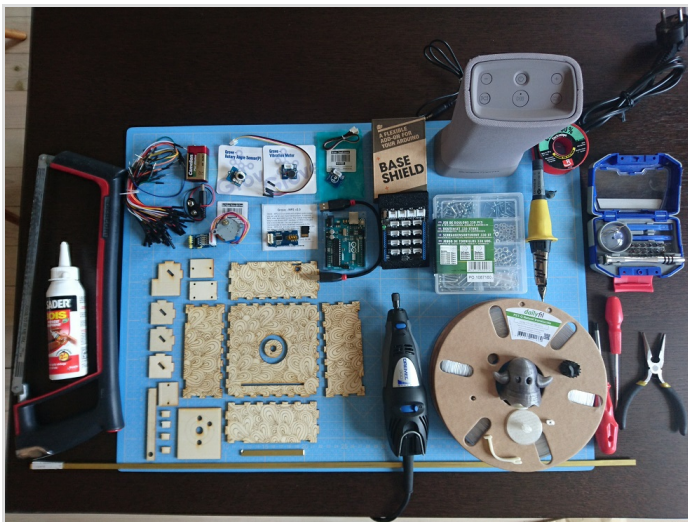
- figurine set on top spins on its stand;
- LED-ring light effect shows through the figurine's translucent stand;
- LCD screen displays a welcome message.

If no other action is made right away, the box goes in "sleep mode".

In "sleep mode", temperature and humidity are displayed with a custom message on LCD screen.

These custom messages as well as the color set for both the LED-ring and LCD screen depend on current temperature:

- LCD and LED-ring color varies from blue to red when temperature rises;
- LED-ring sparkles at a random frequency to add some animation;
- custom message is displayed on LCD screen according to the temperature range currently reading.



Matériaux

Necessary materials are listed here below by category.

For the box

- poplar plywood 3mm-thick for laser cutting

💡 Poplar is pretty cheap and allows precise and fast laser cutting

- brass bar 80x7x2mm - to change and increase the touching area with regards to the capacitive sensor

📌 A 1m-length brass bar can be found in DIY store for about 7€ but is easily replaceable by any conductive material whatever is shape.

- transparent PLA for 3D printing:
 - access door to base tier
 - figurine's stand and support - the stand will also diffuse the LED-ring light
 - button for volume control
 - lateral joints
- colored PLA (according to maker's preference) for the figurine and/or scenery set on top of the box

Electronic supplies

- 1 Arduino Uno Rev.3
- 1 MW power supply: 1500mA Rotary Switch Adaptor
- 1 reversing switch KNX-1, 3A, 250V AC, to be used as main switch
- 1 Seeed Studio potentiometer: Grove Rotary Angle Sensor (P)
- 1 Seeed Studio capacitive sensor: Grove Touch Sensor
- 1 Adafruit LED-ring: Neopixel Ring 12 x 5050 RGB
- 1 Seeed Studio MP3-player: Grove MP3 v2.0
- 1 micro SD-Card to store the playlist
- 1 Seeed Studio Gear Stepper Motor with Driver: Step Motor 28BYJ-48 5V DC



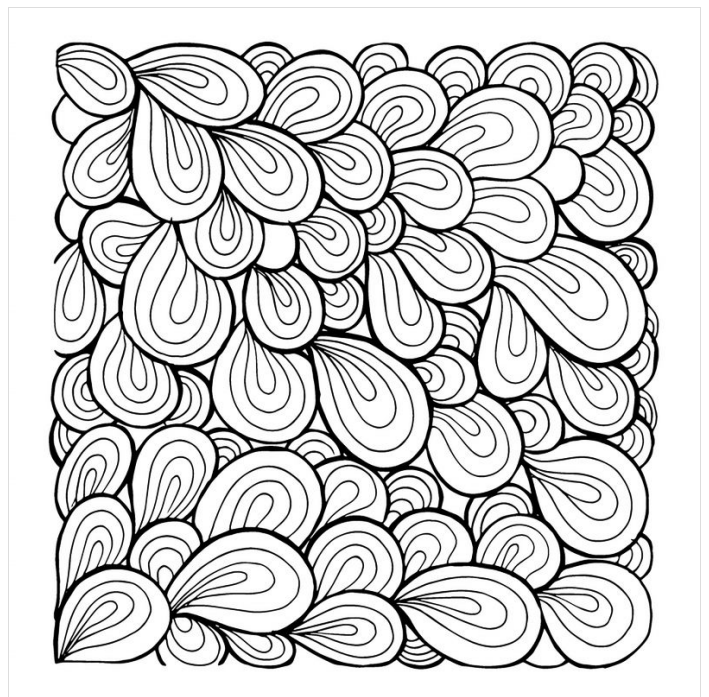
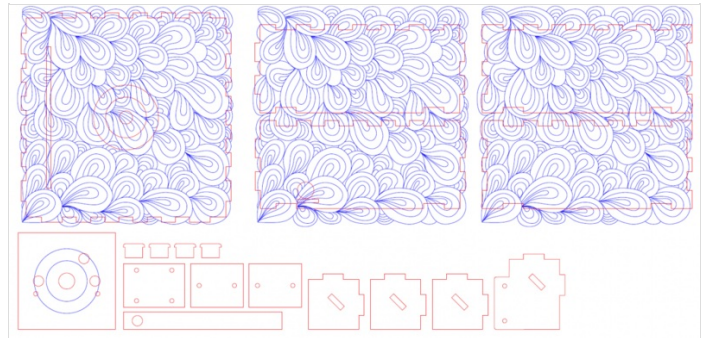
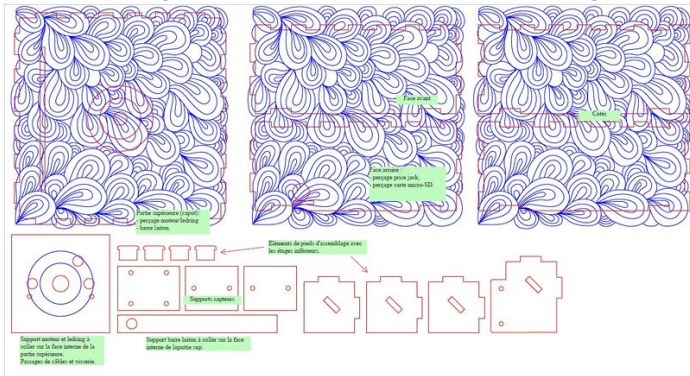
Étape 1 - 2D design (new tier for the box)

Goal was to create a new tier to the original box by using laser cutting of course but also laser engraving.

An engraving pattern was found.

This image file was reproduced using Inkscape and Beziers curves.

This newly designed pattern was then added to the cutting pattern.

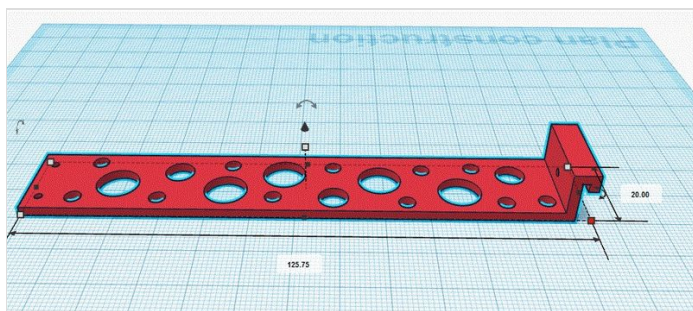
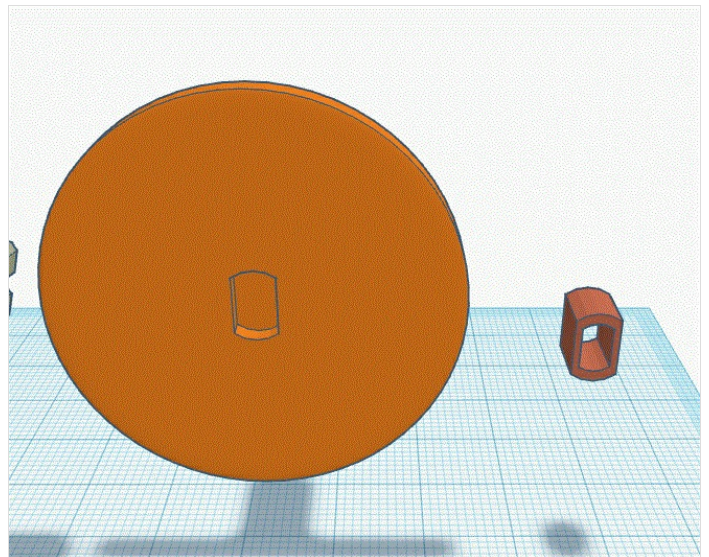
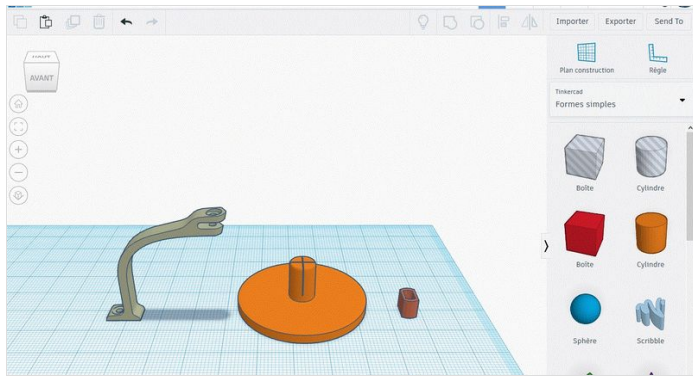
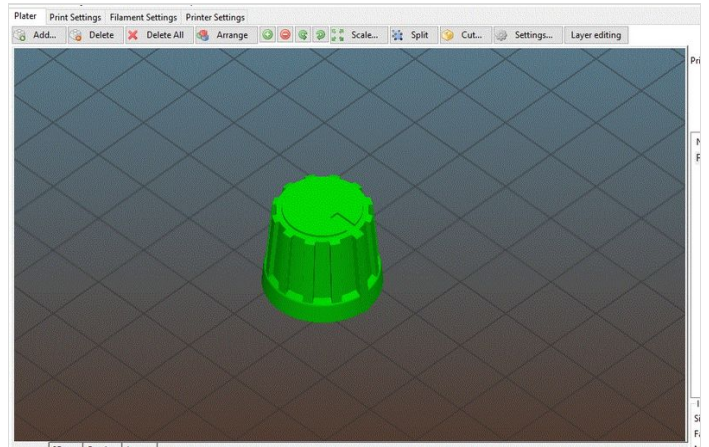
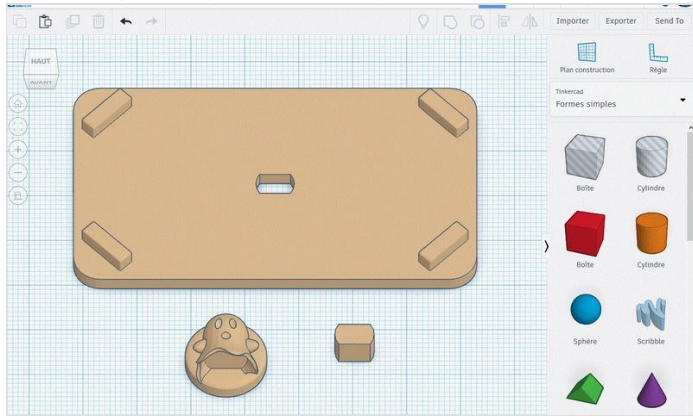


Étape 2 - 3D design

The components to be 3D-printed are the following:

- access door for the base tier alongside with the main switch and potentiometer
- button to cover the potentiometer toothed wheel
- figurine stand which also serve to subdue LED-ring lighting effects
- figurine itself, in this instance the "Funkgeist" animated ghost found on Thingiverse (<https://www.thingiverse.com/thing:570654>) and its support elements
- lateral joints to ensure the box solidity

For this step, Inkscape, TinkerCAD and Blender were used.



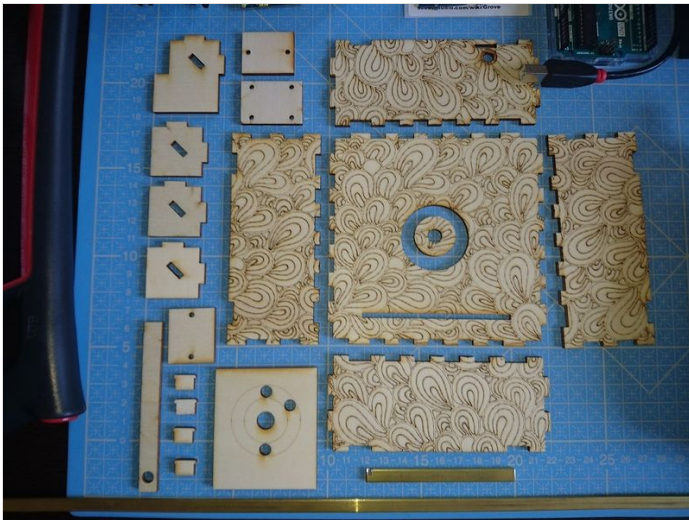
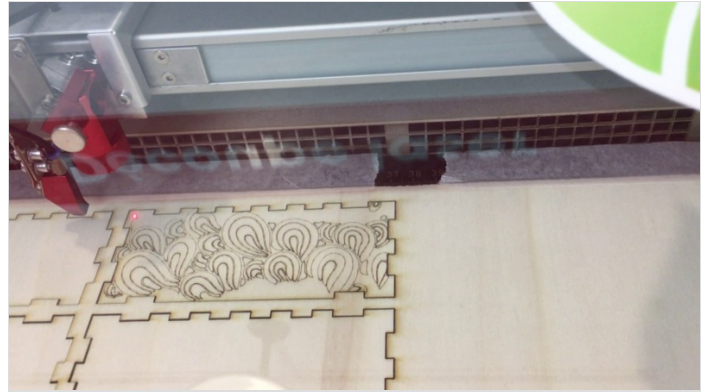
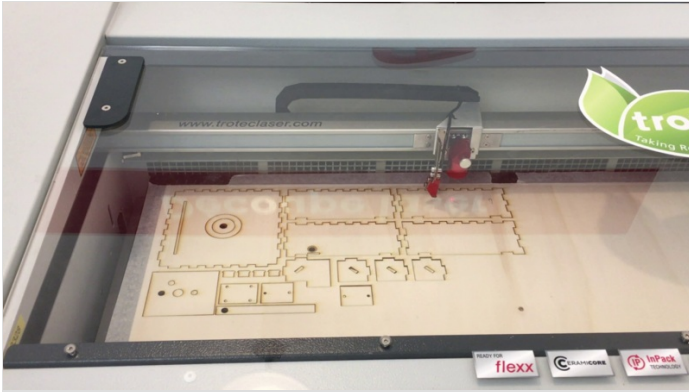
Étape 3 - Laser engraving and cutting of the new box tier

Laser cutting - and engraving - has been carried out within the EdFab in Saint-Denis, France, under the kindly direction of Lola and Arthur: big thanks to both of them!

2 other identical tiers have been made using SplyLab facility in La Verrière, France, with the benevolent help of Francis in discovering yet a new source file pre-processing due to a different machine. An equal big thanks to him!



Once the new tier is engraved and cut, one can choose to paint, decorate and customize it at will.

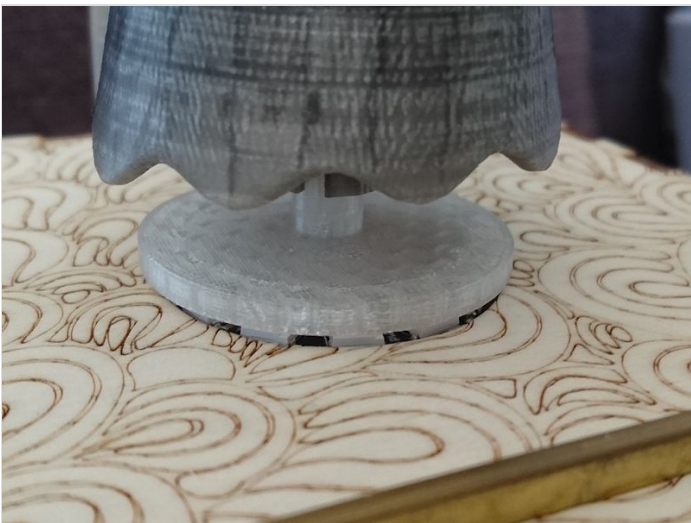


Étape 4 - 3D-printing

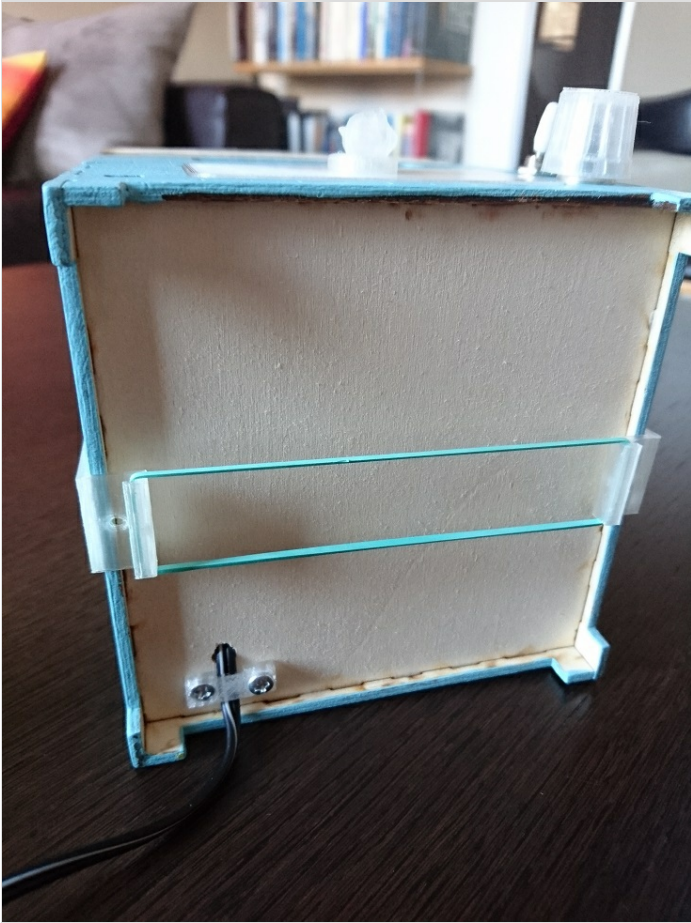
All printings needed for the project were made on a Prusa i3 MK3.

Those are shown as opposite:

- base tier
 - access door
 - button to put on the potentiometer toothed wheel
- MoodBox tier
 - figurine stand (spinning) and support (fixed)
 - figurine itself
 - lateral joints







Étape 5 - Wiring diagram

Sensors and other elements are connected to the base-shield set on top of the Arduino Uno as described here after with port numbers.

Input

- Potentiometer: analog input A0
- Capacitive sensor: analog input A1
- BME280 sensor:
 - Vin: 5V
 - GND: GND
 - CS: 10
 - SDI: 11
 - SDO: 12
 - SCK: 13

Output

- LCD: I2C port
- MP3 player: D2
- LED-ring: D3

Motor

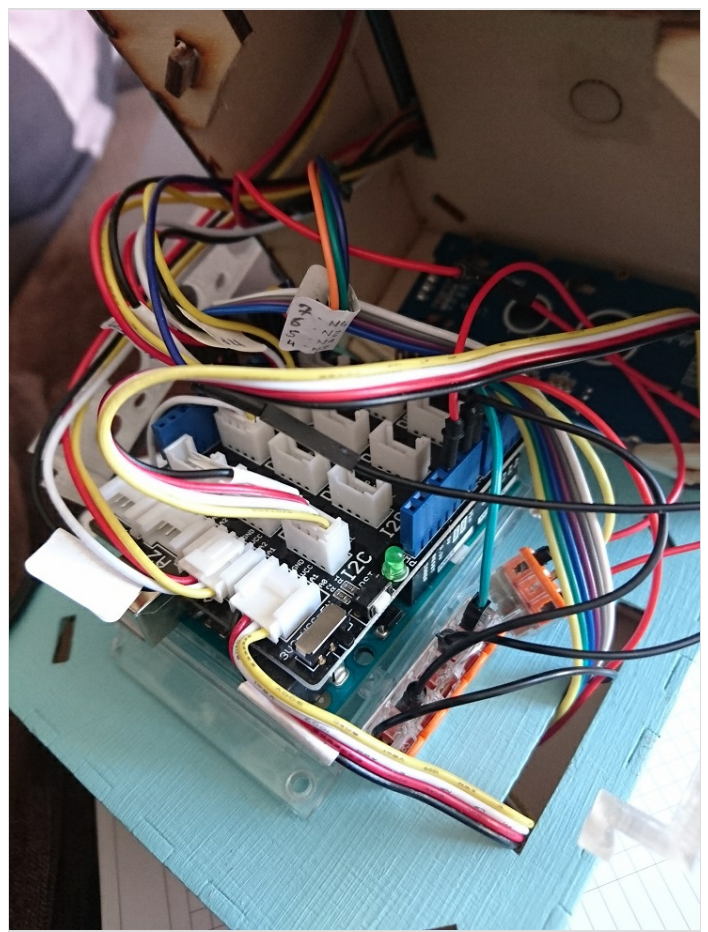
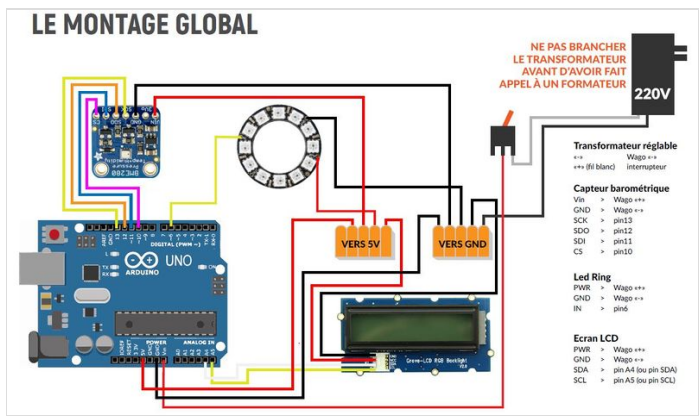
- Step motor
 - IN 1: 10
 - IN 2: 9
 - IN 3: 11
 - IN 4: 8

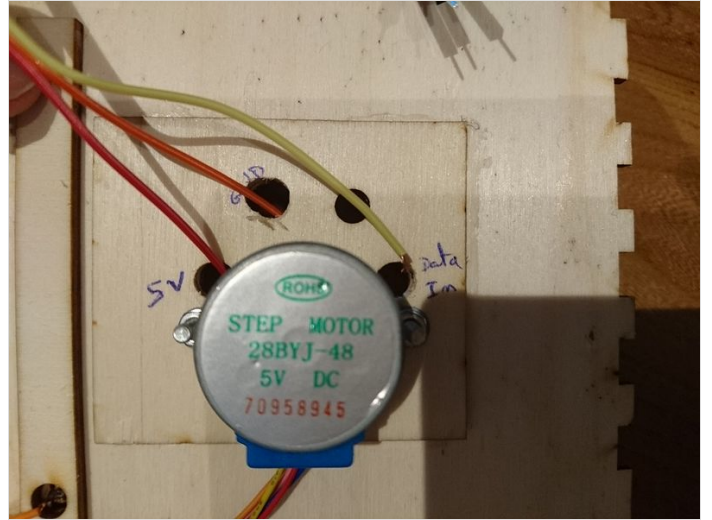
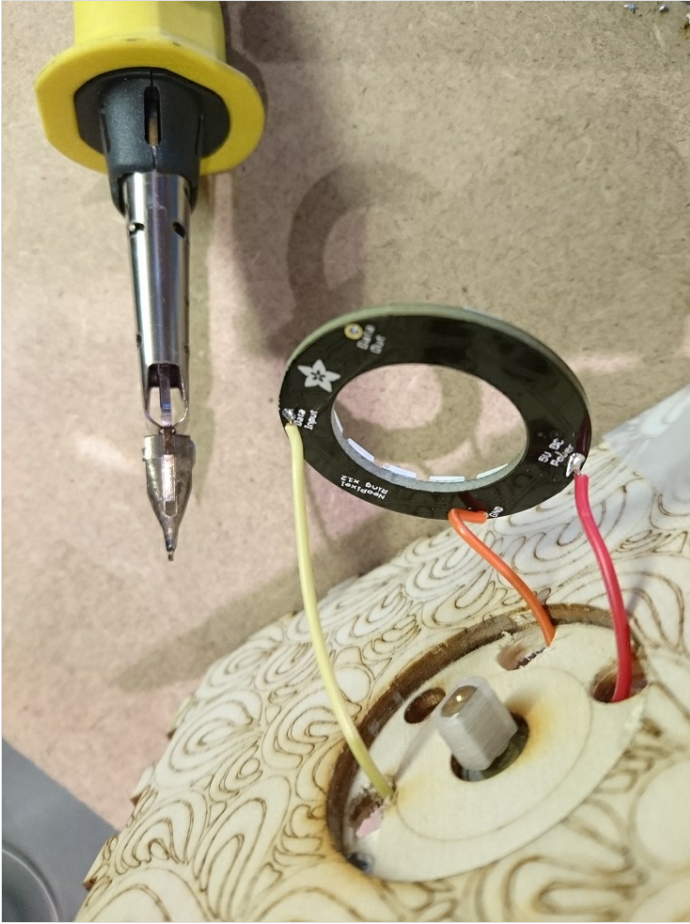
Power supply

It's important to notice that all in all, components need a lot of power to work properly. There is not enough power available through USB connection (about 5V) and a 9V battery will be depleted faster than you'd like.

This is why we used an adaptor that can be plugged into a wall-socket and deliver enough power steadily.

⚠ Don't forget to set the power supply rotary switch on 9V in order to get the right power for your box.





Étape 6 - Arduino programming

Programming was a bit tricky due to Arduino native sequential working.

Workarounds must have been found in order to animate several parts of the box not only sequentially but also simultaneously.


The first 3 screenshots as shown opposite exhibit declarations for each component of the box and the MoodBox tier.

Main program is a loop which executes the sequence as requested with regards to user interaction. Each sequence is identified with a number:

- 0: sequence where nothing happens
- 1: start sequence
- 2: awakening sequence, triggered by at least one touch on the brass bar
- 3: music sequence

Waiting time management, especially for counting how many times the bar was touched, is using the millis() function to avoid using the delay() function where the whole program is put on hold.

To bypass the sequential functioning, the AccelStepper.h library was used as well as combined commands programmed within routines called from the main loop. This allows for example simultaneousness for both LED effects and step motor spinning.

 This is the trick used in code to prevent stopping the motor from spinning while waiting for a signal from the brass bar.

Libraries

Main program calls libraries functions specific to sensors and components used in the box. Those are listed below:

- LCD: "rgb_lcd.h"
 - https://github.com/Seeed-Studio/Grove_LCD_RGB_Backlight
- LED-ring: "FastLED.h"
 - <https://github.com/FastLED/FastLED>
- MP3 player: "SoftwareSerial.h" and "MP3Player_KT403A.h"
 - https://github.com/Seeed-Studio/Grove_Serial_MP3_Player_V2.0/archive/master.zip
- Step motor: "AccelStepper.h"
 - <https://github.com/adafruit/AccelStepper>
- BME280 sensor: "SPI.h", "Adafruit_Sensor.h" and "Adafruit_BME280.h"
 - https://github.com/adafruit/Adafruit_BME280_Library
 - https://github.com/adafruit/Adafruit_Sensor

```
// 1- LCD
// -----
// Bibliothèques nécessaires à l'afficheur LCD :
// - I2C : pour "Inter-Integrated Circuit", gère le lien entre un microprocesseur et des circuits, module "Wire.h" du package Arduino.
// - LCD : on utilise la bibliothèque fournie par le fabricant Seeed "Grove_LCD_RGB_Backlight", module "rgb_lcd.h".
// cf. https://github.com/Seeed-Studio/Grove\_LCD\_RGB\_Backlight
#include <Wire.h>
#include "rgb_lcd.h"

// Déclaration du nom de l'afficheur LCD : la construction de la bibliothèque Grove impose "lcd" comme nom par défaut.
rgb_lcd lcd;

// Rem : par définition une sortie I2C ne nécessite pas de définition de brochage, ceci est autogéré par le protocole.

// Définition de caractères spéciaux :
byte heart[8] = {0b00000,0b01010,0b11111,0b11111,0b11111,0b01110,0b00100,0b00000};
byte smiley[8] = {0b00000,0b00000,0b01010,0b00000,0b00000,0b10001,0b01110,0b00000};
byte frownie[8] = {0b00000,0b00000,0b01010,0b00000,0b00000,0b00000,0b01110,0b00001};
byte armsDown[8] = {0b00100,0b01010,0b00100,0b01010,0b01110,0b01010,0b00100,0b01010};
byte armsUp[8] = {0b00100,0b01010,0b00100,0b01010,0b01110,0b00100,0b00100,0b01010};

// Définition de la variable qui va contenir le message à afficher :
String Message = "Message de base";

// Définition de la luminosité de l'écran [0 255]
const int LCD_Seuil = 255;

// 2- LED-RING
// -----
// Bibliothèques nécessaires à la gestion de la led-ring :
// - FastLED : bibliothèque dédiée aux led-rings "Mopixels" du fabricant Adafruit.
#include "FastLED.h"

// Définition du nombre de leds du led-ring et de la broche de sortie correspondante
#define NBR_LEDS 12 // 12 leds sur l'anneau
#define LEDR_PIN 3 // Broche du led-ring (D3 sur le Base Shield V2 Seeed)

// Déclaration du tableau de valeurs de couleurs des leds
CRGB leds[NBR_LEDS];

// Variable de stockage de la dernière couleur utilisée
byte Latest_Color[3] = {0,20,50};

// Variable et valeur de brillance de base
float BaseBrightness = 20;

// 3- TOUCHE CAPACITIVE
// -----
// Broche d'entrée de la touche/ie de la barre laiton
#define TCAP_PIN A1

// Variable d'entrée associée à la touche
int TCAP_Value = 0;
```

```
// 4- LECTEUR MP3
// -----
// Le module MP3 Grove de la marque Seeed (http://wiki.seeedstudio.com/Grove-MP3\_V2.0/)
// lit une carte micro-SD et renvoie vers une sortie jack.
// Ce module utilise la bibliothèque "MP3Player_KT403A.h" contenue dans le paquet "Grove_Serial_MP3_V2.0"
// (https://github.com/Seeed-Studio/Grove\_Serial\_MP3\_Player\_V2.0/archive/master.zip).
#include <SoftwareSerial.h>
#include <MP3Player_KT403A.h>

// Comme pour le moteur (ci-dessous) il faut définir un nom à "l'objet" lecteur et son nom est
// obligatoirement "mp3" et une broche de connexion :
#define MP3_PIN 2
SoftwareSerial mp3(MP3_PIN,3); // Rem : le 2nd argument définit la broche de sortie sur la carte MP3
// et "3" est la prise jack.

// Commandes de pilotage du lecteur :
// 1- Play (P)
// 2- Pause/Resume (p)
// 3- Next (N)
// 4- Previous (L)
// On définit le statut du lecteur au départ :
int PlayStatus = 0;

// 5- POTENTIOMETRE
// -----
// Broche d'entrée du potentiomètre
#define POT_PIN A0

// Variables d'entrée associée au potentiomètre
int PotValue;
int PotValueRef = 14;

// 6- MOTEUR
// -----
// Le moteur pas-à-pas bipolaire (https://www.motronic.fr/art-moteur-24kv18-3628.htm)
// est commandé via la bibliothèque "Stepper.h" qui est pré-installée sous Arduino.
// Déclaration de la bibliothèque :
#include <Stepper.h>
#include <AccelStepper.h>

// Déclaration du nom du moteur ("Ghost_Stepper") et passage des arguments (type et numéros des broches) à la bibliothèque.
// ATTENTION : le brochage n'est pas "linéaire" sur le driver !!!
// - nom_stepper(nbr tours, IN4, IN2, IN1, IN3)
AccelStepper Ghost_Stepper(AccelStepper::FULL4WIRE, 4, 5, 6, 7);

// Définition des vitesses typiques (en RPMs) :
const int LOW_SPEED_Stepper = 4096/8/4;
const int NORMAL_SPEED_Stepper = 4096/8/2;
const int HIGH_SPEED_Stepper = 4096/8;
const int ACC_Stepper = 4096/8/4;
int SPEED_Stepper = HIGH_SPEED_Stepper;
```



```

// 7- CAPTEUR "METEO" BME280
// -----
// Ce multio capteur BME280 (https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout?view=all)
// est monté ici en mode SPI ("Serial Peripheral Interface").
// Il utilise 3 bibliothèques. Une commune à tous les capteurs Adafruit, une spécifique au capteur en question et une dédiée
// à la communication SPI (native Arduino). On les déclare donc :
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

// Brochage : - CS (Slave Select)      : broche 10
//             - MISO (Master Out Slave In) : broche 11
//             - MOSI (Master In Slave Out) : broche 12
//             - SCK (Serial Clock)       : broche 13

// Définition de la pression de référence locale (ie du "sea level pressure") nécessaire à l'étalonnage du capteur :
#define BME_SEALEVELPRESSURE_HPA (1024.6)

// Déclaration du nom du capteur et passage des arguments (nécessaire au module SPI) :
Adafruit_BME280 bme(10, 11, 12, 13);

// N- Autres variables
// -----
// Variable des différentes séquences, en fonction de sa valeur :
// - 0 : il ne se passe rien;
// - 1 : séquence de démarrage;
// - 2 : séquence de "réveil";
// - 3 : séquence musicale.
// Au démarrage on a de l'affichage et une animation led-ring :
int Seq_Box = 1;

// Création d'un compteur de temps :
unsigned long TimeCounter;
unsigned long ElapsedTime;

```

```

// Fonction d'attente avec, ou sans, la rotation du moteur
void Task_Wait(int waiting_time, boolean Moteur){
// =====
// La clef ici est de pouvoir faire tourner le moteur en simultané //
// avec la fonction d'attente. C'est l'ASTUCE du code, mettre la //
// commande dans la boucle while, avec un check booléen. //
// =====

  unsigned long previousMillis;
  unsigned long currentMillis;

  previousMillis = millis();
  currentMillis = millis();
  while (currentMillis - previousMillis < waiting_time) {
    currentMillis = millis();

    // Moteur en simultané en fonction de la variable booléenne
    // *****
    if (Moteur) Ghost_Stepper.runSpeed();
  }
}

```

Étape 7 - Assembling, testing and finishing off

Arduino Uno, potentiometer and main switch are connected to the power supply inside and on the base tier of the box (lower tier). LCD screen is installed in side the LCD tier (middle tier).

MoodBox tier (upper tier) is composed as described here below.

On box walls are fixed:

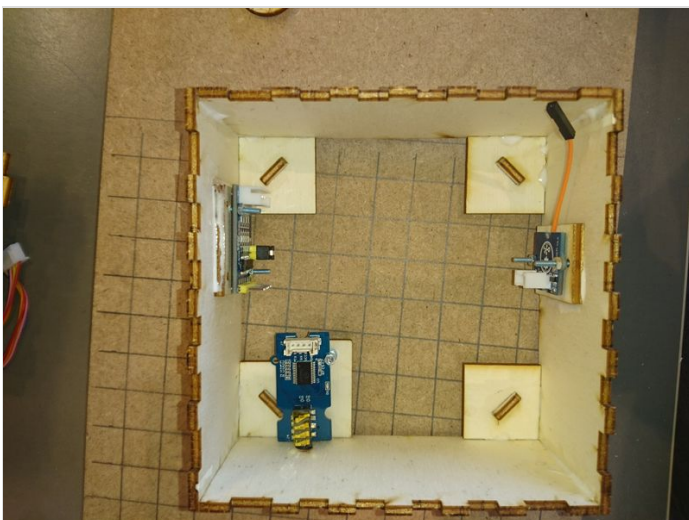
- the step motor used to animate the figurine - its controller card can be seen on the left on the overview picture as shown opposite
- the MP3 player with both SD-card slot and AudioIn port aligned with MoodBox rear-side wall openings - at bottom on overview picture
- the capacitive sensor to be connected to the brass bar - see the orange wire on the right on the overview picture

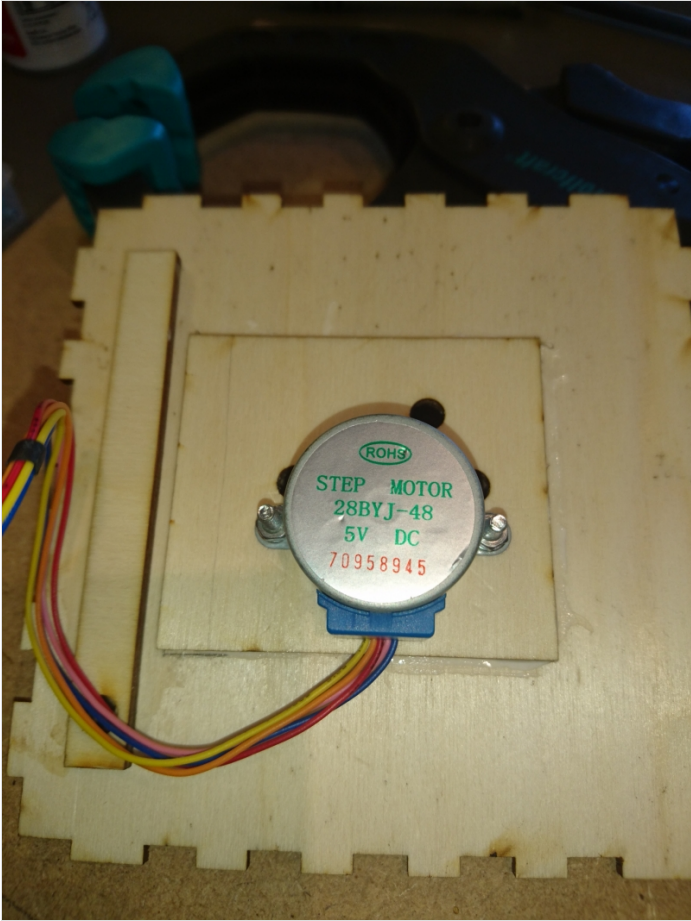
Right underneath and throughout the MoodBox top cover are fixed:

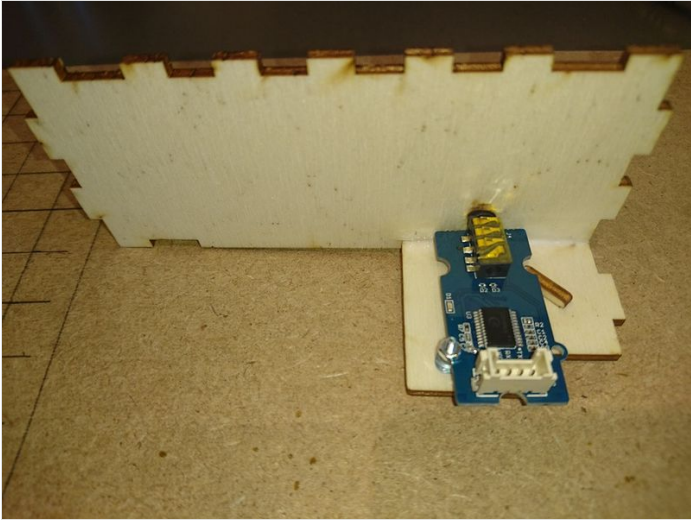
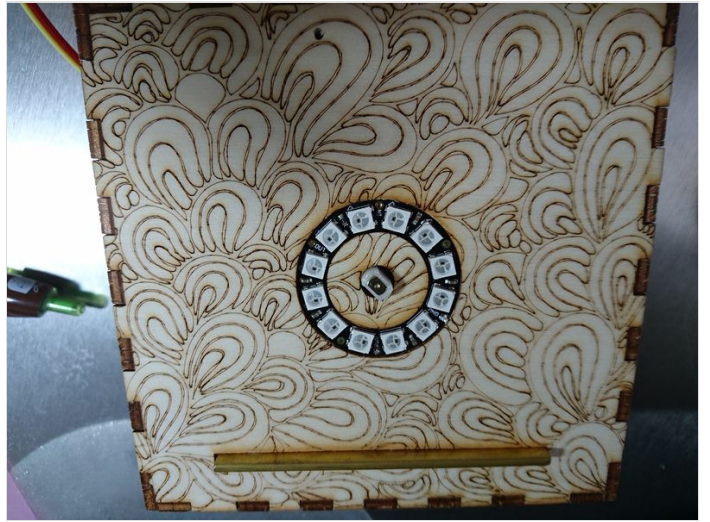
- the LED-ring
- the step motor with only its axis pulling through, centered with the LED-ring
- the brass bar

On top of the MoodBox tier are fixed:

- the figurine stand, just above the LED-ring in order to subdue its bright light
- the figurine itself, set on the motor axis to be able to spin with it







Étape 8 - Now let's dance!

MoodBox tier is now up and able to run.

Please refer to Introduction and overall specifications to properly use it!

Thank you for your interest :)



Notes et références

Figurine used for final presentation comes from Thingiverse (<https://www.thingiverse.com/thing:570654>).

It was chosen by default. First idea was to recreate in 3D the fabulous Fantasia dancing couple: Hyacinth and Ben Ali Gator.

"Bentolux" project - main box design - to be available soon on WikiFab (around July 2018).

French page describing this project can be found here: <https://wikifab.org/wiki/MoodBox>