Bentolux - Module qualité de l'air ambiant

L'objectif de ce projet est de fabriquer et programmer un module de contrôle de qualité de l'air ambiant (TVOC et eCO2) qui viendra se greffer en tant que 3eme étage sur une station météo Bentolux, et qui permettra l'affichage des valeurs sur deux demi-cadrans faussement analogiques.

Difficulté **Moyen** | Durée **4 jour(s)** | Catégories **Électronique, Bien-être & Santé**

Coût **150 EUR (€)**

Sommaire

Introduction

La Bentolux est une station météo construite dans le cadre des cours de fabrication numérique dispensés par l'IMT d'Albi. Elle est d'abord constitué de deux premiers étages, qui sont le socle commun à tout les apprenants. Le premier étage contient la carte Arduino, une Led actionnée par potentiomètre, ainsi qu'un écran d'affichage LCD. Le second étage comporte le capteur météo dont les valeurs (Temperature, hygrometrie relative et pression atmosphériques) sont envoyés vers l'écran LCD., ainsi qu'un anneau de Led permettant des animations visuelles en fonction des valeurs retournées. Le 3eme étage, objet de ce tuto, vise a afficher grâce a des servomoteurs les valeurs de eCO2 et de TVOC sous la forme de cadran à aiguilles.

Matériaux

- Bentolux
- Bentolux_-_Module_qualit_de_l_air_ambiant_Plan_d_coupes_bois_-_Explication.pdf
- Bentolux_-_Module_qualit__de_l_air_ambiant_Manuel_montage_1_.pdf
- Bentolux_-_Module_qualit__de_l_air_ambiant_Plan_de_cablage_POA8.pdf
- Bentolux_-_Module_qualit__de_l_air_ambiant_Bentolux_-_station_m_t_o.ino

Outils

Étape 1 - Matériaux et outils

Outils
- Fer à souder
- Pince coupante
- Pince à dénuder
- Multimètre
- Colle cyanoacrylate
- Cintreuse
- Scie japonaise
- Ciseaux

Matériaux:
- Etain
- Un carrelet de méranti blanc
- Un carrelet d'acajou
- Une plaque d'isorel

Composants:
- arduino UNO
- 20 cables M/M
- 10 cables M/F
- 2 servomoteur DF9GMS
- 1 plaque de prototypage
- 1 potentionmetre
- 1 Led Blanche
- 1 Afficheur OLED 0,96'' I2C TF052
- 1 Anneau NeoPixel 12 leds RGB ADA1643
- 1 capteur pression/temp/hum BME280
- 1 Capteur de qualité d'air CCS811 SEN0339
- 4 Wago 5 entrées
- 2 Wago 3 entrées

Étape 2 - Prérequis - Bentolux Station Météo

- Decoupe au laser des parois de la boite (DOC1)
- Assemblage de la boite (DOC2)
- Branchement des composants (DOC3)
- Programmation du code pour faire interagir les élements (ecran LCD, capteur Temp/Hum, anneau OLED) (DOC4)

Étape 3 - Cablage

A partir du plan de cablage fourni à l'étape 1, il nous faut rajouter
- relier 1 Wago 5 entrée au premier Wago ou se rejoignent les alimentations des composants
- relier 1 Wago 5 entrées au premier Wago ou se rejoignent les GND des differents composants.
- relier 1 Wago 3 entrées à la broche SDL de la carte Arduino
- relier 1 Wago 3 entrées à la broche SDA de la carte Arduino

- Sur les Wago 5 entrées, brancher les VCC et GND des servomoteurs et du CSS811.
- Sur les bornes restantes des Wago 3 entrées, brancher les SDA et SCL du CSS811 et du BME 280.

Étape 4 - Code

```
1 // librairies pour le BME
2 #include <BME280I2C.h>
3 #include "DFRobot_CCS811.h"
4 #include <Wire.h>
```

```
#include <Wire.h>
#include "ssd1306.h"
#include "FastLED.h"
#include "DFRobot_CCS811.h"
#include <Servo.h>
//ici le nombre total de leds
#define NUM_LEDS 12
// ici la pin pour les leds
#define DATA_PIN 6
// déclaration d'un tableau pour les leds
CRGB leds[NUM_LEDS];
DFRobot_CCS811 CCS811;
Servo ServoCO2;
Servo ServoTVOC;

const int POTAR = A0; // broche du potar
const int LED = 9; // broche de la LED
int valPOTAR = 0; //RAZ valeur

int angle=120;
int angle2=0;

BME280I2C::Settings settings(
   BME280::OSR_X1,
   BME280::OSR_X1,
   BME280::OSR_X1,
   BME280::Mode_Forced,
   BME280::StandbyTime_1000ms,
   BME280::Filter_Off,
   BME280::SpiEnable_False,
   0x77 // I2C address. I2C specific.
);

BME280I2C bme(settings);

/* Ces tableaux de caractères serviront UNIQUEMENT pour l'affichage à l'écran.
 *  Il faudra afficher ces variables et non les "floats" déclarées pour le capteur,
 *  l'écran ne reconnaissant que des chaînes de caractère
 */

  char tempC[9];
  char humC[10];
  char presC[11];

void setup()
{
     Serial.begin(9600);
     ServoCO2.attach(5);
     ServoTVOC.attach(3);

// on vérifie que le capteur COE2/TVOC est branché
   while(CCS811.begin() != 0){
       Serial.println("failed to init chip, please check if the chip connection is fine");
       delay(1000);
    }

// déclaration entrée et sortie de Potar et LED. Lecture en enregistrement dans la variable valPOTAR de la valeur analog du Potar

     pinMode(POTAR, INPUT);
     pinMode (LED, OUTPUT);
     valPOTAR = analogRead (POTAR);

// tout ce blabla sert uniquement à vérifier si un capteur est présent et bien branché

  while(!Serial) {} // Wait

  Wire.begin();

  while(!bme.begin())
  {
    Serial.println("Could not find BME280 sensor!");
    delay(1000);
  }

  // bme.chipID(); // Deprecated. See chipModel().
```

```
// bme.chipID(); // Deprecated. See chipModel().
switch(bme.chipModel())
{
  case BME280::ChipModel_BME280:
    Serial.println("Found BME280 sensor! Success.");
    break;
  case BME280::ChipModel_BMP280:
    Serial.println("Found BMP280 sensor! No Humidity available.");
    break;
  default:
    Serial.println("Found UNKNOWN sensor! Error!");
}

  // on lance l'écran et on le colore de noir

 ssd1306_128x64_i2c_init();
 ssd1306_fillScreen(0x00);
 // on lance la typo
 ssd1306_setFixedFont(ssd1306xled_font6x8);

 FastLED.addLeds<WS2811, DATA_PIN, GRB>(leds, NUM_LEDS);
   // on peut régler ici la luminosité : 0-255
    LEDS.setBrightness(50);

}

void loop()

 {

  analogRead (A0);
  valPOTAR = analogRead (POTAR);
  analogWrite (9, valPOTAR/4);
  // On déclare 3 variables : température, humidité, pression
   float temp(NAN), hum(NAN), pres(NAN);

     if(CCS811.checkDataReady() == true){
      Serial.print("CO2: ");
      Serial.print(CCS811.getCO2PPM());
      Serial.print("ppm, TVOC: ");
      Serial.print(CCS811.getTVOCPPB());
      Serial.println("ppb");
      delay(1000);

  } else {
      Serial.println("Data is not ready!");
  }


  // Déclenchement du capteur
  BME280::TempUnit tempUnit(BME280::TempUnit_Celsius);
  BME280::PresUnit presUnit(BME280::PresUnit_hPa);
  bme.read(pres, temp, hum, tempUnit, presUnit);

// ces lignes servent à convertir les valeurs "float" du capteur en "char" destinées à l'écran
// (utile uniquement pour votre code final)
 dtostrf (temp,5,1,tempC);
 dtostrf (hum,5,1,humC);
 dtostrf (pres,5,2,presC);

// on imprime les valeurs sur le moniteur série

 if(CCS811.checkDataReady() == true){
      Serial.print("CO2: ");
      Serial.print(CCS811.getCO2PPM());
      Serial.print("ppm, TVOC: ");
      Serial.print(CCS811.getTVOCPPB());
      Serial.println("ppb");

  } else {
      Serial.println("Data is not ready!");
  }
  /*!
   * @brief Set baseline
```

```
    * @param get from getBaseline.ino
    */
   CCS811.writeBaseLine(0x447B);
   //delay cannot be less than measurement cycle
   delay(1000);

   // Première ligne, normal

    ssd1306_printFixed (0, 8, "Temp:", STYLE_NORMAL);
   ssd1306_printFixed(56, 8, tempC , STYLE_BOLD);
   ssd1306_printFixed (96, 8, "C", STYLE_NORMAL);
   ssd1306_printFixed(0, 16,"RH: ", STYLE_NORMAL);
   ssd1306_printFixed(56, 16, humC , STYLE_BOLD);
   ssd1306_printFixed(96, 16,"%", STYLE_NORMAL);
   ssd1306_printFixed(0, 24, "Pres: ", STYLE_NORMAL);
   ssd1306_printFixed(56, 24, presC , STYLE_BOLD);
   ssd1306_printFixed(96, 24, "hPa" , STYLE_NORMAL);


int CO2;
CO2 = map (CCS811.getCO2PPM(),400,6000,0, 120);

    for (int position=angle; position < CO2 ; position++) {
    ServoCO2.write(position);
     Serial.println(position);
      angle = position;
    delay(20);
  }
  for (int position=angle ; position > CO2 ; position--) {
    ServoCO2.write(position);
     Serial.println(position);
     angle = position;
    delay(20);
  }

int TVOC;
TVOC = map (CCS811.getTVOCPPB(),8000,0,0, 120);

     for (int position=angle2; position < TVOC ; position++) {
    ServoTVOC.write(position);
     Serial.println(position);
      angle2 = position;
    delay(20);
  }
  for (int position=angle2 ; position > TVOC ; position--) {
    ServoTVOC.write(position);
     Serial.println(position);
     angle2 = position;
    delay(20);
  }


  if (hum > 40 && hum <50)
  {
   //Allumage rotatif des Leds 0 à 5 en bleu suivant valeur de délais

 for( int i = 0; i < 12; i++){
      leds[i] = CRGB::Green;
       FastLED.show();
        delay (60);
       }
       for( int i = 0; i < 12; i++){
     leds[i] = CRGB::Black;
      FastLED.show();
       delay (20);
       }
 }
   else {
  //Allumage rotatif des Leds 6 à 11 en rouge suivant valeur de délais
 for( int i = 0; i < 12; i++){
      leds[i] = CRGB::Red;
       FastLED.show();
```

```
226         delay (20);
227         }
228  }
229  // Extinction rotative des 12 Led suivant valeur de délais
230  for( int i = 0; i < 12; i++){
231      leds[i] = CRGB::Black;
232       FastLED.show();
233         delay (10);
234         }
235   }
```

Étape 5 - Support décoratif en bois

Étape 6 - Impression 3D

-Impression 3D de supports de servomoteur. STL récupéré sur Thingiverse.
- Impression 3D d'un dessin perso de bouton de potentiometre