```
//--------------------RECUPERE LES BIBLIOTHEQUES
NECESSAIRES--------------------
#include <Arduino.h>
//Bibliothèques pour la connexion wifi et les échanges HTTP
#include <ESP8266WiFi.h>
#include<ESP8266HTTPClient.h>
#include <WiFiClient.h>
//Bibliothèques pour le capteur et l'écran OLED
#include <Wire.h>
#include <SPI.h>
#include<Adafruit_GFX.h>
#include<Adafruit_SSD1306.h>
#include<Adafruit_I2CDevice.h>

//--------------------VARIABLES POUR ECRAN--------------------
//La largeur de l'écran en pixels
#define SCREEN_WIDTH 128
//La hauteur de l'écran en pixels
#define SCREEN_HEIGHT 64

#define OLED_RESET     4 // Reset pin # (or -1 if sharing
Arduino reset pin)
//L'adresse I2C de notre écran OLED
#define SCREEN_ADDRESS 0x3C
//Instancie la connexion à l'écran I2C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

//La largeur du logo pour l'humidité en pixels
#define LOGO_HUM_WIDTH    13
//La hauteur du logo pour l'humidité en pixels
#define LOGO_HUM_HEIGHT   13
//La largeur du logo pour la température en pixels
#define LOGO_TEMP_WIDTH    15
//La hauteur du logo pour la température en pixels
#define LOGO_TEMP_HEIGHT   15
//La largeur du logo pour le réseau en pixels
#define LOGO_RES_WIDTH    12
//La hauteur du logo pour le réseau en pixels
#define LOGO_RES_HEIGHT   12
```

```
//Logo représentant une goute d'eau (pour le taux d'humidité)
static const unsigned char PROGMEM logo_humidity[] =
 { 0b00000000, 0b01000000,
   0b00000001, 0b10000000,
   0b00000011, 0b10000000,
   0b00000111, 0b11000000,
   0b00001111, 0b11000000,
   0b00001111, 0b11100000,
   0b00011111, 0b11110000,
   0b00011111, 0b11110000,
   0b00011111, 0b11110000,
   0b00011101, 0b11110000,
   0b00001001, 0b11110000,
   0b00000100, 0b11100000,
   0b00000011, 0b11000000,
 };
//Logo représentant un thermomètre (pour la température)
static const unsigned char PROGMEM logo_temperature[] =
 {
   0b00000001, 0b10000000,
   0b00000001, 0b10110000,
   0b00000001, 0b10000000,
   0b00000001, 0b10100000,
   0b00000001, 0b10000000,
   0b00000001, 0b10110000,
   0b00000001, 0b10000000,
   0b00000001, 0b10100000,
   0b00000001, 0b10000000,
   0b00000001, 0b10110000,
   0b00000011, 0b11000000,
   0b00000111, 0b11100000,
   0b00000111, 0b11100000,
   0b00000011, 0b11000000,
   0b00000001, 0b10000000,
 };
//Logo représentant les barres réseau
static const unsigned char PROGMEM logo_reseau[] =
 {
   0b00000000, 0b00000111,
   0b00000000, 0b00000111,
   0b00000000, 0b00000111,
```

```
  0b00000000, 0b00000111,
  0b00000001, 0b11000111,
  0b00000001, 0b11000111,
  0b00000001, 0b11000111,
  0b00000001, 0b11000111,
  0b01110001, 0b11000111,
  0b01110001, 0b11000111,
  0b01110001, 0b11000111,
  0b01110001, 0b11000111,
 };

//--------------------VARIABLES POUR CAPTEUR--------------------
//Définit l'adresse I2C du capteur
const int ADDRESS = 0x40;

//Variables pour les relevés et la moyenne de ces relevés
double temperature, humidity;
double moyenne_temperature, moyenne_humidite;
double somme_temperature = 0.0;
double somme_humidite = 0.0;
int nombre_releves = 0;

//--------------------VARIABLES POUR LE
RESEAU--------------------
//Informations de connexion au point d'accès
const char * SSID = "moto g(10)_4713";
const char * PASSWORD = "cxurp5yghfit3cz";
//Variables contenant l'adresse IP lorsqu'elle est attribuée
String adressIP = "inconnu";
String requete;

//--------------------VARIABLES POUR ECHANGES
HTTP--------------------
//Définit l'adresse IP du serveur (ici de notre Raspberry Pi)
byte server[] = {192, 168, 90, 251};
//Client pour effectuer les requêtes HTTP
WiFiClient client;

//--------------------DECLARATION DES
FONCTIONS--------------------
void onConnected(const WiFiEventStationModeConnected& event);
```

```
void onGotIP(const WiFiEventStationModeGotIP& event);
void sensor_init(const int addr);
double read_temperature(const int addr);
double read_humidity(const int addr);
void  ecranOLED(double temp, double hum);


//************************************************************
******************
//FONCTION    : setup
//UTILISATION : Permet d'initialiser l'ESP8266-01
//************************************************************
******************

void setup() {
  //Mise en place d'une liaison série
  Serial.begin(115200);
  Serial.println("");
  //Mode de connexion
  WiFi.mode(WIFI_STA);
  //Démarrer la connexion
  WiFi.begin(SSID, PASSWORD);

  //Afficher les informations de connexion
  static WiFiEventHandler onConnectedHandler = WiFi.
onStationModeConnected(onConnected);
  static WiFiEventHandler onGotIPHandler = WiFi.
onStationModeGotIP(onGotIP);

  //Lance la connexion au bus I2C sur les broches 2 (SDA) et
0 (SCL)
  Wire.begin(2,0);
  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.
3V internally
  //Si l'écran n'est pas trouvé
 if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    //Bloque l'exécution du porgramme
    for(;;);
  }
```

```
  //Affiche l'image d'accueil d'Adafruit
  display.display();
  delay(2000);
  //Nettoie l'écran
  display.clearDisplay();
  //Dessine un pixel en blanc
  display.drawPixel(10, 10, SSD1306_WHITE);
  //Affiche à l'écran ce que nous lui avons demandé de faire
  display.display();
  delay(2000);

  //Affiche l'écran d'accueil
  ecranOLED(0, 0);

  //Initialise la connexion au capteur
  sensor_init(ADDRESS);
}


//************************************************************
******************
//FONCTION    : loop
//UTILISATION : Permet d'effectuer les actions de la fonction
indéfiniment lors
//    du fonctionnement de l'ESP8266-01S.
//************************************************************
******************

void loop() {
  //Récupère les relevés de température et d'humidité
effectués par le capteur
  temperature = read_temperature(ADDRESS);
  humidity = read_humidity(ADDRESS);
  //Si les valeurs renvoyées par le capteur sont cohérentes
  if (temperature > -45 and humidity > 0){
    //Effectue les 11 premiers relevés pour faire une moyenne
par minute
    if (nombre_releves != 11){
      somme_temperature = somme_temperature + temperature;
      somme_humidite = somme_humidite + humidity;
      nombre_releves = nombre_releves + 1;
```

```
    }
    //Effectue le 12 relevé et fait la moyenne des relevés
sur la minute passée
    else {
      //Récupère le 12 relevé pour faire la moyenne
      somme_temperature = somme_temperature + temperature;
      somme_humidite = somme_humidite + humidity;

      //Fait la moyenne des relevés récupérés
      moyenne_temperature = somme_temperature /
nombre_releves;
      moyenne_humidite = somme_humidite / nombre_releves;

      //Vérifie le statut de la connexion
      if(WiFi.status()== WL_CONNECTED){
          //Si le client arrive à se connecter au serveur
          if(client.connect(server, 80)){
            Serial.println("Requête envoyée");
            //Envoie la requête au serveur
            requete = "GET /assets/api-station/v1/add-release.
php?sensor=1&temperature=";
            requete += String(moyenne_temperature);
            requete += "&humidity=" +
String(moyenne_humidite);
            requete += " HTTP/1.0";
            client.println(requete);
            client.println();
            //Ferme la connexion
            client.stop();
          }
          //Sinon, un message d'erreur est affiché
          else{
            Serial.println("ERREUR: Connexion au serveur
(Raspberry Pi) impossible");
          }
      }
      //Si la WiFi n'est pas active
      else {
        Serial.println("WiFi non connecté");
      }
```

```
      //Affiche sur l'écran les moyennes des relevés
     ecranOLED(moyenne_temperature, moyenne_humidite);

      //Affiche dans la console la température en °C
      Serial.print("Temperature: ");
     Serial.print(moyenne_temperature);
      Serial.println("*C");
      //Affiche dans la console le pourcentage d'humidité
      Serial.print("Humidity: ");
     Serial.print(moyenne_humidite);
      Serial.println("%");

      //Remet les variables à 0
      somme_temperature = 0.0;
      somme_humidite = 0.0;
      nombre_releves = 0;
    }
  }else{
    Serial.println("Le capteur ne fonctionne pas");
  }

  //Effectue des relevés toutes les 5 secondes
  delay(5000);
}


//************************************************************
******************
//FONCTION    : onConnected
//UTILISATION : Permet d'afficher dans le moniteur série que
le WiFi est
//    connecté.
//PARAMETRES  :
//    + WiFiEventStationModeConnected : Correspond à
l'événement WiFi connecté.
//************************************************************
******************

void onConnected(const WiFiEventStationModeConnected& event){
  Serial.println("Wifi connecté");
}
```

```
//************************************************************
******************
//FONCTION    : onGotIP
//UTILISATION : Permet d'afficher dans le moniteur série les
informations de
//    connexion WiFi de notre ESP8266-01S
//PARAMETRES  :
//    + WiFiEventStationModeGotIP : Correspond à l'événement
WiFi connecté et
//        données de connexion récupérées.
//************************************************************
******************

void onGotIP(const WiFiEventStationModeGotIP& event){
  //Affcihe les informations de connexion dans la console
  Serial.println("Adresse IP " + WiFi.localIP().toString());
  Serial.println("Passerelle IP " + WiFi.gatewayIP().
toString());
  Serial.println("DNS IP " + WiFi.dnsIP().toString());
  Serial.print("Puissance signal ");
 Serial.println(WiFi.RSSI());
  //Attribue l'adresse IP à la variables globale adressIP
  adressIP = WiFi.localIP().toString();
}


//************************************************************
******************
//FONCTION    : sensor_init
//UTILISATION : Permet d'initialiser la connexion au capteur
de température et
//    d'humidité.
//PARAMETRES  :
//    + addr : Correspond à l'adresse I2C de notre capteur.
//************************************************************
******************

void sensor_init(const int addr) {
  //Se connecte au capteur sur les broches 2 (SCL) et 0 (SDA)
```

```
  Wire.begin(2, 0);
  //Effectue une première connexion au bout de 100
millisecondes
  delay(100);
 Wire.beginTransmission(addr);
 Wire.endTransmission();
}


//************************************************************
******************
//FONCTION    : read_temperature
//UTILISATION : Récupère la température relevé par le capteur
et la renvoie sous
//    la forme d'un nombre décimal
//PARAMETRES  :
//    + addr : Correspond à l'adresse I2C de notre capteur.
//************************************************************
******************

double read_temperature(const int addr) {
  double temperature;
  int low_byte, high_byte, raw_data;
  //Envoie une commande pour intialiser le relévé de
température
 Wire.beginTransmission(addr);
 Wire.write(0xE3);
 Wire.endTransmission();
  //Récupère et lit le relevé de température
 Wire.requestFrom(addr, 2);
  if (Wire.available() <= 2) {
    high_byte = Wire.read();
    low_byte = Wire.read();
    high_byte = high_byte << 8;
    raw_data = high_byte + low_byte;
  }
  temperature = (175.72 * raw_data) / 65536;
  temperature = temperature - 46.85;
  return temperature;
}
```

```
//************************************************************
******************
//FONCTION    : read_humidity
//UTILISATION : Récupère le porcentage d'humidité relevé par
le capteur et le
//    renvoie sous la forme d'un nombre décimal
//PARAMETRES  :
//    + addr : Correspond à l'adresse I2C de notre capteur.
//************************************************************
******************

double read_humidity(const int addr) {
  double humidity, raw_data_1, raw_data_2;
  int low_byte, high_byte, container;
  //Envoie une commande pour intialiser le relévé d'humidité
 Wire.beginTransmission(addr);
  Wire.write(0xE5);
  Wire.endTransmission();
  //Récupère et lit le relevé d'humidité
  Wire.requestFrom(addr, 2);
  if(Wire.available() <= 2) {
    high_byte = Wire.read();
    container = high_byte / 100;
    high_byte = high_byte % 100;
    low_byte = Wire.read();
    raw_data_1 = container * 25600;
    raw_data_2 = high_byte * 256 + low_byte;
  }
  raw_data_1 = (125 * raw_data_1) / 65536;
  raw_data_2 = (125 * raw_data_2) / 65536;
  humidity = raw_data_1 + raw_data_2;
  humidity = humidity - 6;
  return humidity;
}

//************************************************************
******************
//FONCTION    : ecranOLED
//UTILISATION : Affiche sur l'écran OLED relié à l'ESP8266,
les derniers relevés
//  effectués ainsi que l'adresse IP et le numéro du capteur
```

```
//PARAMETRES  :
//    + temp : Correspond à la température qui va être
affichée à l'écran.
//    + hum  : Correspond à l'humidité qui va être affichée à
l'écran.
//************************************************************
******************


void ecranOLED(double temp, double hum) {
  //Nettoie l'écran OLED
 display.clearDisplay();

  //Définit la taille et la couleur d'écriture
 display.setTextSize(1);
 display.setTextColor(SSD1306_WHITE);

  //Dessine les informations relatives à la sonde
  display.setCursor(36, 0);
  display.println(F("ID SONDE : 1"));

  //Dessine les logos de température, d'humidité et de réseau
que nous allons utiliser
  display.drawBitmap(0, 16, logo_humidity, LOGO_HUM_WIDTH,
LOGO_HUM_HEIGHT, SSD1306_WHITE);
  display.drawBitmap(0, 34, logo_temperature,
LOGO_TEMP_WIDTH,LOGO_TEMP_HEIGHT,SSD1306_WHITE);
  display.drawBitmap(0, 50, logo_reseau, LOGO_RES_WIDTH,
LOGO_RES_HEIGHT, SSD1306_WHITE);

  //Dessine les informations relatives à l'humidité
  display.setCursor(18, 16);
  display.print(F("Humidite : "));
  display.print(hum);
  display.println(F("%"));

  //Dessine les informations relatives à la température
  display.setCursor(18, 34);
  display.print(F("Temp : "));
  display.print(temp);
  display.println(F(" C"));
```

```
  //Dessine les informations relatives au réseau
  display.setCursor(18, 50);
 display.println(adressIP);

  //Affiche à l'écran ce que nous lui avons fait dessiner
  display.display();
}
```