

# Découvrir l'IOT (Internet Of Things) et son univers

***Mise en place de la partie Infrastructure et Réseaux***

## I] Fonctionnement de la station météo

Afin que notre station météo fonctionne normalement, nous avons besoin d'un capteur de température et d'humidité ainsi qu'un serveur qui pourra stocker nos données et les partager à d'autres utilisateurs.

Nous allons donc mettre en place 2 technologies pour répondre à ce besoin. En effet, un Raspberry Pi servira de serveur tandis qu'un capteur d'humidité couplé à un microcontrôleur Wi-Fi jouera le rôle de la sonde.

**Remarque : Il est possible d'avoir plusieurs sondes effectuant des relevés. Il suffit d'ajouter un couple capteur/microcontrôleur pour avoir une nouvelle sonde.**

Le Raspberry Pi hébergera un site Internet permettant de consulter les derniers relevés de température d'une sonde. Afin que le site fonctionne, il possèdera également une base de données dans laquelle sera stockée les différents relevés de température et d'humidité pour une sonde, ainsi qu'une API REST permettant au site Internet ainsi qu'aux différentes sondes de communiquer avec la base de données.

Le microcontrôleur Wi-Fi sera quant à lui programmé pour se connecter au réseau Wi-Fi et effectuera des relevés de température et d'humidité toutes les 5 secondes. Une moyenne des relevés sera effectuée toutes les minutes et cette moyenne, composée donc de 12 relevés, sera ensuite envoyé à la base de données du Raspberry via l'API REST. Un écran OLED permet également aux utilisateurs de voir les derniers relevés effectués par la sonde, ainsi que l'adresse IP et l'ID de leur sonde.

## II] Installation du Système d'Exploitation (OS) et configuration du Raspberry Pi

Nous avons commencé par installer l'OS Raspberry Pi OS Lite sur notre Raspberry Pi Zero WH. Pour cela, nous avons utilisé l'application Raspberry Pi Imager et gravé l'image disque sur la carte SD.

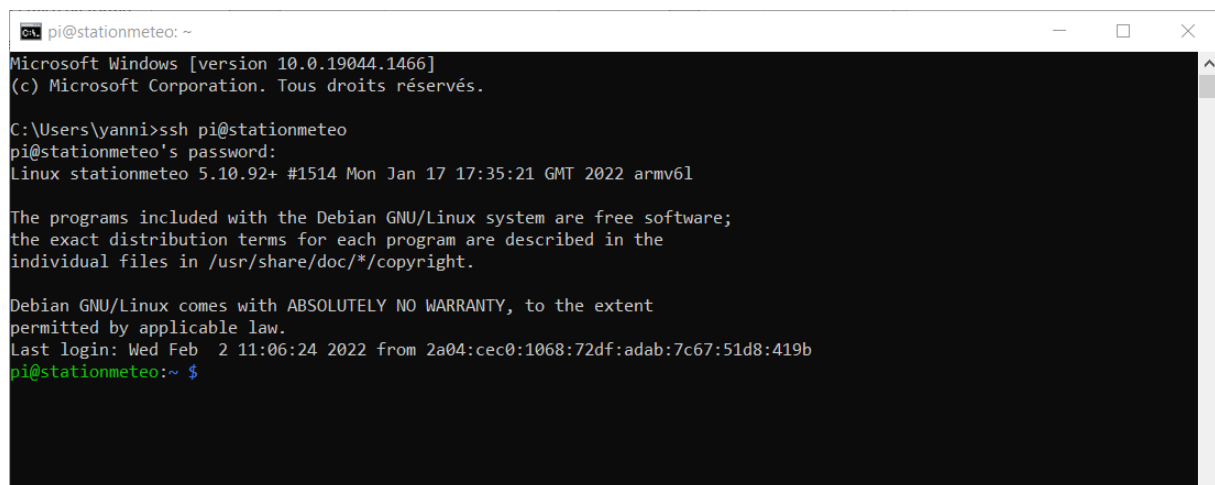
Une fois l'OS téléchargé et installé sur la carte, nous avons ajouté 2 fichiers de configuration à la racine de la carte SD. Le premier fichier créé est un fichier permettant d'activer la connexion SSH. Ce fichier se nomme simplement *ssh* et il ne contient aucune information spécifique. Le second fichier à créer permet d'activer et de configurer la connexion Wi-Fi du Raspberry Pi. Pour cela, il faut créer un second fichier à la racine de la carte SD nommé *wpa\_supplicant.conf*. Ce fichier contient les informations de connexion afin que le Raspberry Pi puisse accéder à Internet.

A ce stade, la carte SD peut être insérée dans le Raspberry Pi et ce dernier peut être branché. Lors de sa première mise sous tension, le Raspberry charge le Système d'Exploitation. Une fois qu'il a fini son installation, la LED verte située sur le côté du Raspberry ne clignote plus.

Nous devons alors récupérer l'adresse IP du Raspberry afin de s'y connecter. Une fois l'adresse IP récupérée, nous allons nous connecter au Raspberry via une connexion SSH (que nous avons configurée à l'étape précédente). Pour cela, nous devons taper la commande `ssh pi@raspberrypi` dans l'invite de commande.

**Remarque :** Lors de la première connexion au Raspberry Pi, un mot de passe générique est utilisé, ce mot de passe est *raspberrypi*. Ce mot de passe par défaut représente une des failles majeures des Raspberry.

Une fois connectés en SSH au Raspberry, nous avons modifié le nom de la machine en modifiant les fichiers */etc/hosts* et */etc/hostname*. Nous avons nommé notre Raspberry Pi : *stationmeteo*. Nous avons également modifié le mot de passe par défaut pour l'utilisateur *pi* grâce à la commande *passwd*.



```
pi@stationmeteo: ~
Microsoft Windows [version 10.0.19044.1466]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\yanni>ssh pi@stationmeteo
pi@stationmeteo's password:
Linux stationmeteo 5.10.92+ #1514 Mon Jan 17 17:35:21 GMT 2022 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Feb  2 11:06:24 2022 from 2a04:cec0:1068:72df:adab:7c67:51d8:419b
pi@stationmeteo:~ $
```

### III] Mise en place du serveur sur Raspberry

Afin de mettre en place un serveur web, nous avons installé Apache, PHP et MariaDB sur le Raspberry Pi.

Nous avons commencé par installer Apache sur le Raspberry Pi. Pour cela, nous avons exécuté la commande `sudo apt install apache2 -y` dans le terminal de commande. Une fois les paquets chargés, nous avons vérifié le fonctionnement d'Apache en rentrant l'adresse IP du Raspberry dans un moteur de recherche. La page d'accueil d'Apache s'est affichée, ce qui signifie que notre serveur web est en place.

Nous avons ensuite installé les paquets PHP afin de pouvoir coder des sites dynamiques et notamment d'utiliser l'API REST qui communiquera entre notre site Internet et notre base de données. Pour ce faire, nous avons exécuté la commande `sudo apt install php php-mbstring`.

Enfin, pour terminer la mise en place du serveur Web, nous avons installé le SGBDR (Système de Gestion de Base de Données Relationnelles) MariaDB. Pour cela, nous avons exécuté la commande `sudo apt install mariadb-server`. Une fois le téléchargement terminé, nous avons configuré MariaDB afin de sécuriser la connexion à nos bases de données. Nous avons alors exécuté la commande `sudo mysql_secure_installation` afin de :

- + définir un nouveau mot de passe pour l'utilisateur root ;
- + supprimer les utilisateurs anonymes ;
- + interdire la connexion à distance à la racine ;
- + supprimer la base de données des tests.

Une fois cette sécurisation effectuée, nous pouvons nous connecter à MariaDB via la commande `mysql -u root -p`.

```
pi@stationmeteo:/ $ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.5.12-MariaDB-0+deb11u1 Raspbian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

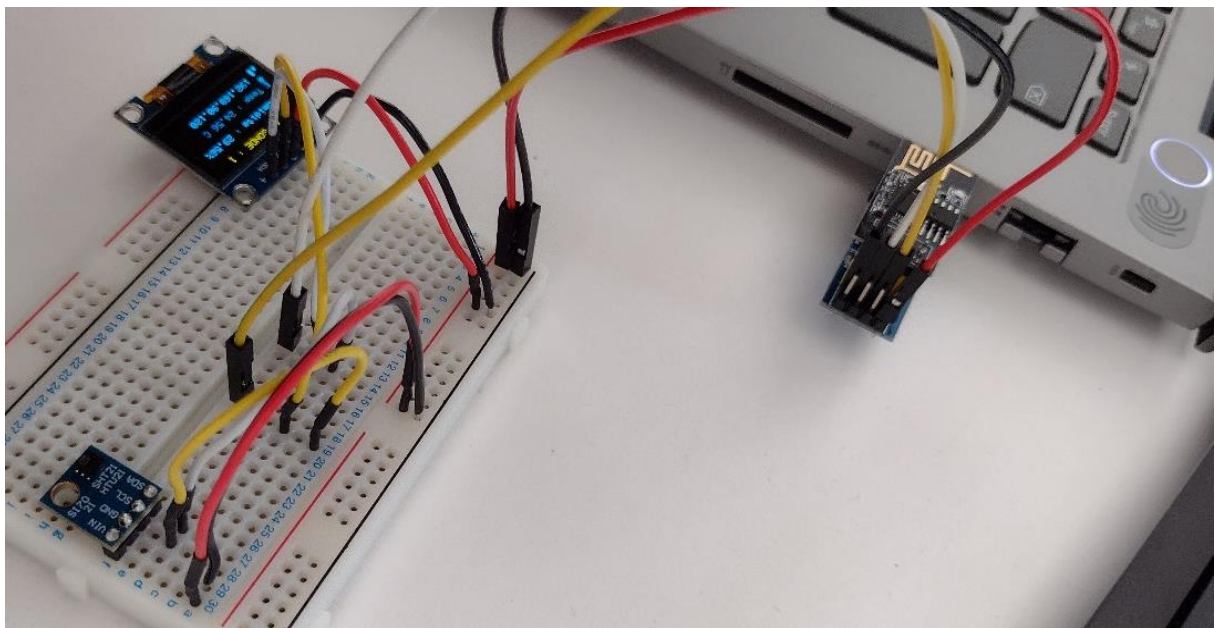
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
+-----+
3 rows in set (0.036 sec)
```

#### IV] Configuration du microcontrôleur ESP8266 01S et du capteur HTU21D-GY21

Nous allons maintenant mettre en place le couple microcontrôleur Wi-Fi et capteur d'humidité et de température. Ce couple correspond à une sonde dans notre projet de station météo.

Nous allons utiliser le capteur d'humidité et de température HTU21D-GY21 ainsi que le microcontrôleur Wi-Fi ESP8266-01S. Afin de programmer l'ESP8266, nous allons utiliser un convertisseur série USB ainsi que l'IDE Arduino.



Nous allons commencer par écrire un script permettant à l'ESP8266-01S de se connecter à Internet. Afin de faciliter la mise en place du projet, nous allons connecter l'ESP8266-01S au même réseau Wi-Fi que notre Raspberry Pi. Pour cela, nous utilisons le script suivant, qui a été réalisé avec l'IDE Arduino et utilise la librairie *ESP8266WiFi.h*, pour se connecter à Internet.

```

//-----RECUPERER LES BIBLIOTHEQUES NECESSAIRES-----
#include <Arduino.h>
//Bibliothèques pour la connexion wifi et les échanges HTTP
#include <ESP8266WiFi.h>

//-----VARIABLES POUR LE RESEAU-----
//Informations de connexion au point d'accès
const char * SSID = "moto g(10)_4713";
const char * PASSWORD = "cxurp5yghfit3cz";

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

//*****
//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);
}

void loop() {
  // put your main code here, to run repeatedly:
}

```

Le code suivant permet quant à lui de récupérer les informations de connexion et de les afficher dans le terminal de retour.

```

//*****
//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){
  //Affiche 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();
}

```

Une fois la connexion Internet configurée et fonctionnelle sur l'ESP8266-01S, nous avons écrit un script permettant de récupérer les relevés de température et d'humidité effectués par le capteur.

Afin que le capteur fonctionne, nous avons relié sa broche VIN avec une alimentation de 3.3 Volts ainsi que sa broche GND avec une prise de terre. Ces 2 branchements permettent au capteur d'être alimenté. Il faut également brancher la broche SCL du capteur avec la broche IO0 de l'ESP8266 ainsi que sa broche SDA avec la broche IO2 du microcontrôleur. Ces 2 branchements permettent de mettre en place le bus I2C, qui nous servira à récupérer les relevés de la sonde.

Lorsque la mise en place des branchements a été effectuée, nous avons ensuite téléversé le script de la figure 2 dans l'ESP8266-01S. Ce script utilise la librairie *Wire.h* afin de récupérer les relevés. Il effectue des relevés toutes les 5 secondes et au bout de 12 relevés (soit 1 minute), il nous affiche la température et l'humidité moyenne relevé lors de la minute écoulée. Nous avons ensuite mis en place un code permettant à l'ESP d'envoyer les données des relevés à notre serveur (ici le Raspberry Pi).

Le script utilisé est le suivant :

```
#include <Arduino.h>
//Bibliothèques pour la connexion wifi et les échanges HTTP
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

//Informations de connexion au point d'accès
const char * SSID = "moto g(10)_4713";
const char * PASSWORD = "cxurp5yghfit3cz";
//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;

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);
}

void loop() {
  //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
      client.println("GET /assets/api-station/v1/add-release.php?sensor=1&temperature=" + String(moyenne_temperature) +
        "&humidity=" + String(moyenne_humidite) + " HTTP/1.0");
      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");
}
}
```

Ce script commence par vérifier que notre microcontrôleur dispose d'une connexion wifi. Si cette connexion est présente, il envoie alors une requête HTTP au serveur précisé dans la variable *server*.

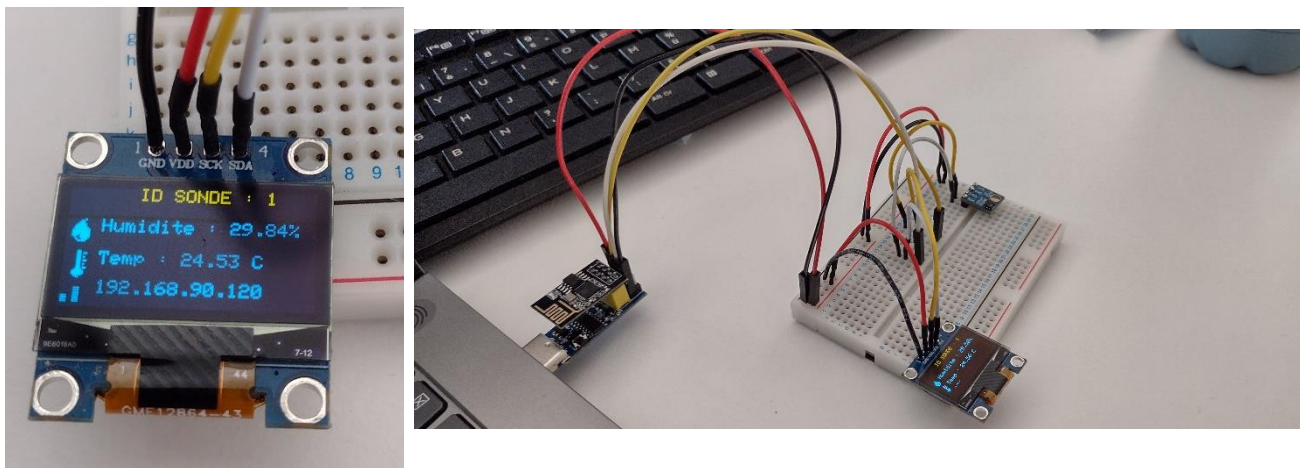
Une fois ces codes écrits et testés, nous les avons couplés afin que l'ESP se connecte à Internet, effectue des relevés et les envoie à notre serveur lors de son fonctionnement. Une fois ces actions effectuées, le couple que nous venons de configurer est opérationnel et peut servir de sonde à notre station météo.

L'ensemble du code est disponible à la fin du document ou dans le fichier *EP8266.ino*.

## V] Mise en place de l'écran OLED sur le microcontrôleur wifi

Dans le contexte du projet, notre sonde doit disposer d'un écran OLED lui permettant d'afficher à l'utilisateur les derniers relevés de température et d'humidité effectués, ainsi que l'adresse IP de la sonde.

Pour cela, nous avons mis en place un écran de type I2C couplé avec l'ESP8266 que nous avons programmé. Nous avons relié la broche VDD à la broche 3.3V du microcontrôleur et la broche GND de l'écran à la broche GND de l'ESP. Ces deux branchements permettent l'alimentation de l'écran afin qu'il puisse fonctionner.



Afin d'afficher les informations demandées à l'écran, nous avons utilisé le bus I2C. Nous avons relié la broche SDA (transportant la Data) de l'écran OLED à la broche IO2 (3) de l'ESP et la broche SCK (transportant les informations horaires) au pin IO0 du microcontrôleur wifi.

Nous avons ensuite installé les bibliothèques nécessaires au fonctionnement de l'écran avec l'IDE Arduino, puis nous avons développé la fonction suivante permettant d'afficher sur l'écran OLED les informations qui nous intéressent.

## VI] Le script complet de l'ESP8266



```

//-----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 goutte 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, 0b11000000,
    0b00011111, 0b11100000,
    0b00011111, 0b11100000,
    0b00011111, 0b11100000,
    0b00011101, 0b11100000,
    0b00001001, 0b11100000,
    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_relevés = 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"));
        //Bloquent l'exécution du programme
        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_relevés != 11){
            somme_temperature = somme_temperature + temperature;
            somme_humidite = somme_humidite + humidity;
            nombre_relevés = nombre_relevés + 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_relevés;
        moyenne_humidite = somme_humidite / nombre_relevés;

        //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_relevés = 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) {
    //Affiche 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 variable 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 initialiser le relevé 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 pourcentage 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 initialiser le relevé 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();
}
```