

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.

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.

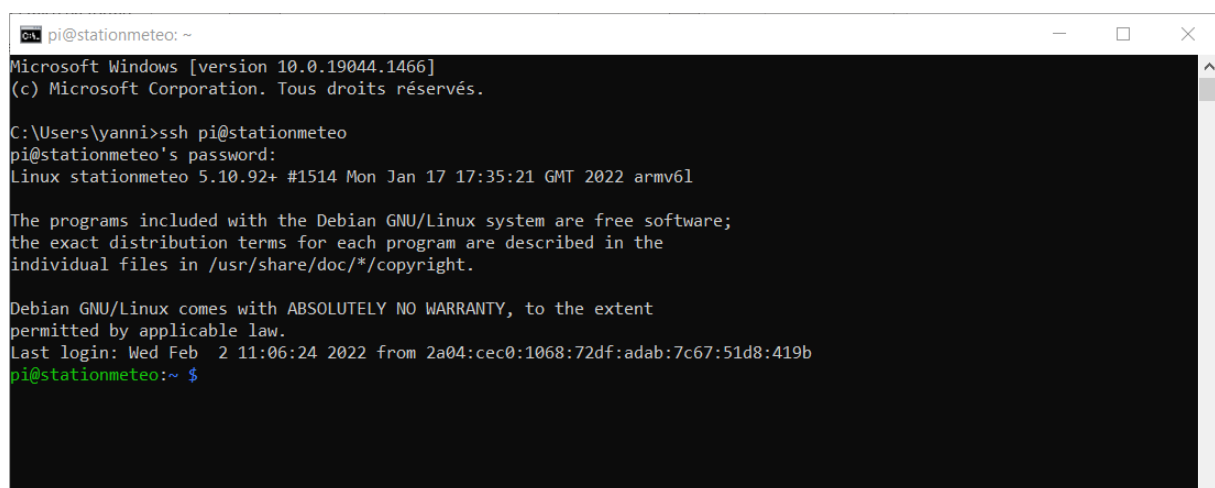
Une fois l'OS téléchargé et installé sur la carte, nous avons ajouté 2 fichiers de configuration à la racine de la carte SD. Le premier fichier crée est un fichier permettant d'activer la connexion SSH. Ce fichier se nomme simplement *ssh* et il ne contient aucune information spécifique. Le second fichier 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.

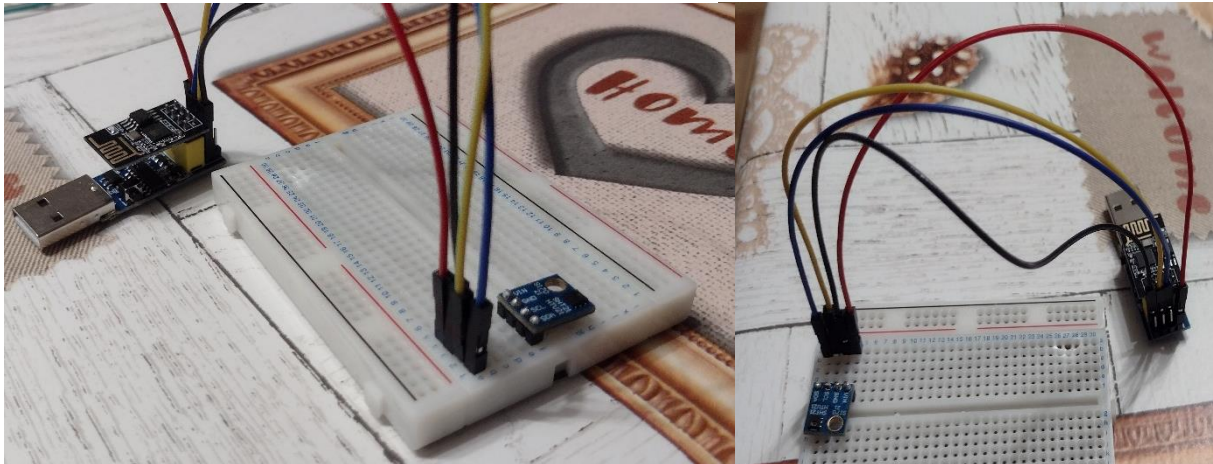
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

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

IV] Configuration du micro-contrôleur 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 de la figure 1. Il a été réalisé avec l'IDE Arduino et utilise la librairie *ESP8266WiFi.h* pour se connecter à Internet et récupérer des informations de connexion.

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.

Une fois les deux codes écrits et testés, nous les avons couplés afin que l'ESP se connecte à Internet et effectue des relevés 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.

V] Mise en place de l'écran OLED sur le Raspberry Pi

Dans le contexte du projet, notre serveur devait également posséder un écran qui affiche son adresse IP, la date et l'heure ainsi que les derniers relevés effectués par la sonde.

Pour cela, nous avons mis en place un écran de type I2C sur le Raspberry Pi. Nous avons relié la broche VVD à la broche 3.3V du Raspberry et la broche GND de l'écran à la broche 7 de notre serveur. 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 du Raspberry Pi. Nous avons relié la broche SDA de l'écran OLED à la broche GPIO2 (3) du Raspberry et la broche SCK au pin GPIO3 (broche 5).

Nous avons ensuite activé l'interface I2C avec l'outil *raspi-config* et installé les bibliothèques python afin de pouvoir développer des programmes utilisant notre écran OLED. Une fois les bibliothèques téléchargées, nous avons vérifié que le module d'affichage OLED était bien présent sur le bus I2C du Raspberry avec la commande *i2cdetect -y 1*. Nous avons trouvé notre écran à l'adresse *0x3c*.

Nous avons ensuite cloné le dépôt Git *Adafruit_Python_SSD1306* et installé les bibliothèques pour Python 3 en exécutant la commande *sudo python3 setup.py install* (le fichier *setup.py* étant situé dans le répertoire *Adafruit_Python_SSD1306*).

Nous avons ensuite copié le script *stats.py* et l'avons adapté afin qu'il affiche l'adresse IP du Raspberry Pi, la date et l'heure ainsi que les derniers relevés de température. Une tâche planifiée réalisée avec l'outil *crontab* permet de lancer ce script modifié au démarrage du Raspberry Pi.

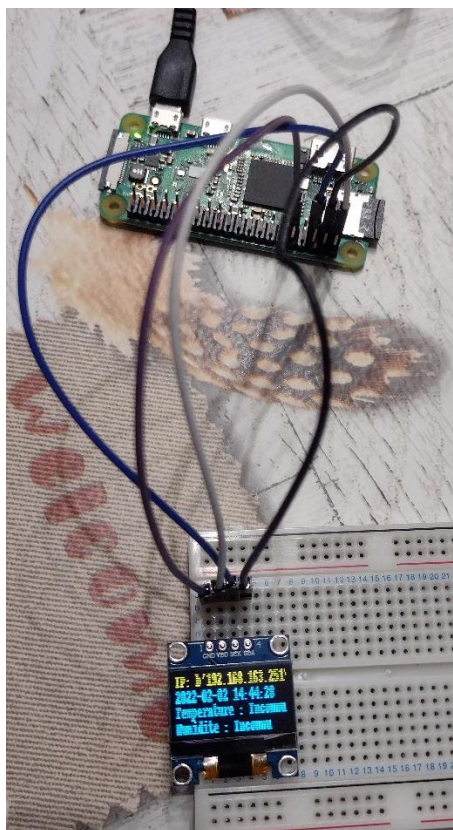


Figure 1 : Connexion Wi-Fi avec l'ESP8266-01S

```
#include <ESP8266WiFi.h>

//Informations de connexion
const char * SSID = "moto g(10)_4713";
const char * PASSWORD = "cxurp5yghfit3cz";

//Déclaration 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);
}

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

void loop() {
}

//*****
//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){
    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());
}
```


Figure 2: Récupérer relevés avec l'ESP8266-01S (Partie1)

```
#include <Wire.h>

//Variables pour le capteur
const int ADDRESS = 0x40;
double temperature, humidity;

//Variables pour la moyenne des relevés
double somme_temperature = 0.0;
double somme_humidite = 0.0;
int nombre_relevés = 0;
double moyenne_temperature, moyenne_humidite;

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

void setup() {
    //Mise en place d'une liaison série
    Serial.begin(115200);
    Serial.println("");

    //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);
    //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 {
        somme_temperature = somme_temperature + temperature;
        somme_humidite = somme_humidite + humidity;
        nombre_relevés = nombre_relevés + 1;

        moyenne_temperature = somme_temperature / nombre_relevés;
        moyenne_humidite = somme_humidite / nombre_relevés;

        //Affiche la température en °C
        Serial.print("Temperature: ");
        Serial.print(temperature);
        Serial.println("°C");
        //Affiche le pourcentage d'humidité
        Serial.print("Humidity: ");
        Serial.print(humidity);
        Serial.println("%");

        //Remet les variables à 0
        somme_temperature = 0.0;
        somme_humidite = 0.0;
        nombre_relevés = 0;
    }

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


Figure 3 : Récupérer relevés avec l'ESP8266-01S (Partie2)

```
//*****
//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 (SDA) et 0 (SCL)
    Wire.begin(0, 2);
    //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;
}
```