TP2 - Surveillance de Température et Humidité

PROBLEMATIQUE : **On souhaite dans ce TP connaitre la température ET l'humidité d'une pièce particulière. La précision devra être de +- 0,5°C.**

Notre choix s'est donc tourné vers le microcontrôleur ESP32 bon marché et le capteur DHT22. Celui-ci est branché sur l'entrée 4 du microcontrôleur.

Dans un premier temps (PARTIE 1), on souhaite vérifier le fonctionnement de cet objet connecté par le moniteur série.

dans un second temps (PARTIE 2), on souhaite consulter les informations sur 2 lignes d'un afficheur LCD I2C graphique 0,96''.

Dans un troisième temps (PARTIE 3), vous devrez faire évoluer le système pour que la consultation se fasse sur une page web embarquée dans l'ESP32. La consultation de cette page pourra se faire grâce à votre téléphone portable connecté en WIFI.

PARTIE 1 : CABLAGE DHT22 / MONITEUR SERIE

a) en vous aidant du web, justifier l'utilisation du capteur DHT22.

La DHT22 est un capteur à bas cout permettant d'acquérir une température et une humidité ambiante d'une manière numérique.

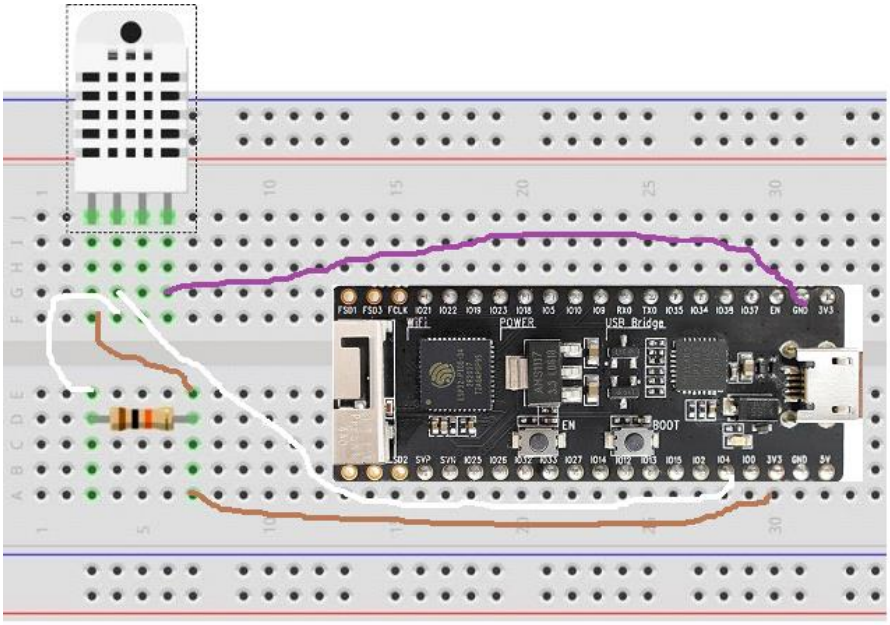
b) Donner le nom de la bibliothèque à intégrer dans le code pour pouvoir communiquer avec le DHT.

LA librairie qui doit être intégrer dans le code pour pouvoir communiquer avec le DHT est le « dht.h »

ATTENTION CETTE BIBLIOTHEQUE NECESSITE L'INSTALLATION D'UNE SECONDE BIBLIOTHEQUE (A rechercher….)

L'autre librairie est la librairie « DHT sensor library ».

c) Sur le schéma ci-dessous, compléter le schéma de câblage entre l'ESP32 et le capteur DHT22.



d) Effectuer le câblage hors tension. FAIRE VERIFIER A L'ENSEIGNANT avant de brancher l'alimentation c'est-à-dire avant de brancher le câble USB. n

e) Chercher le code permettant d'afficher par l'intermédiaire du moniteur série la température en degrés Celsius et l'humidité en %.

Voici le code qui nous a permis d'avoir l'affichage de la température et l'humidité via la moniteur série

==========

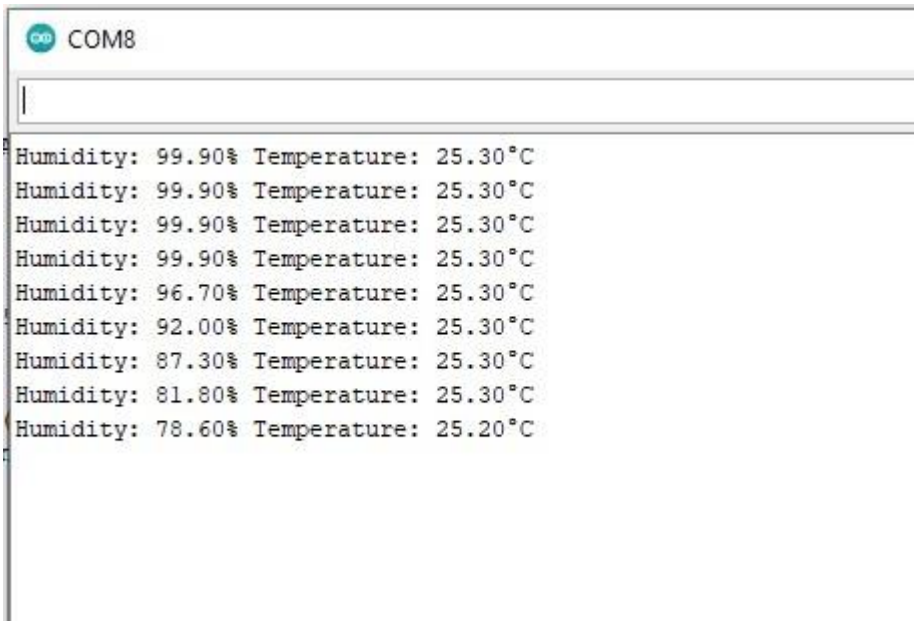
```
#include "DHT.h"
#define DHTPIN 4
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println(F("DHTxx test!"));

  dht.begin();
}
```

```
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);

  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  Serial.print(hif);
  Serial.println(F("°F"));
}
```

==========

f) . Téléverser le programme. Tester le bon fonctionnement

Voici l'affichage via moniteur série :



g) **FAIRE VERIFIER LE BON FONCTIONNEMENT A L'ENSEIGNANT.**

PARTIE 2 : AFFICHEUR LCD

En mode de fonctionnement normal, le moniteur série ne sera pas présent.
On souhaite donc afficher la température sur un afficheur graphique 0,96'' LCD I2C fourni.

Vous aurez à intégrer une bibliothèque logicielle nécessaires pour que l'ESP32 puisse envoyer des données à l'afficheur par le bus I2C.

Ces bibliothèques communiquent par le bus I2C avec l'afficheur : la patte SDA doit être connectée à la patte 21 et la patte SCL doit être connectée à la patte 22.

a) Importer les bibliothèques **ESP8266 AND ESP32 OLED DRIVER FOR SSD1306 DISPLAYS** dans l'éditeur ARDUINO.
b) Effectuer le câblage hors tension. FAIRE VERIFIER A L'ENSEIGNANT
c) Dans les exemples Arduino, ouvrir dans le répertoire « ESP8266 and ESP32 OLED DRIVER for SSD1306 display » le fichier SSD1306SIMPLEDEMO.ino
d) Téléverser ce code dans l'ESP32 et observer le résultat du code.

Ce code nous a permis d'avoir l'affichage de la températures et l'humidité via l'afficheur LCD.

==========

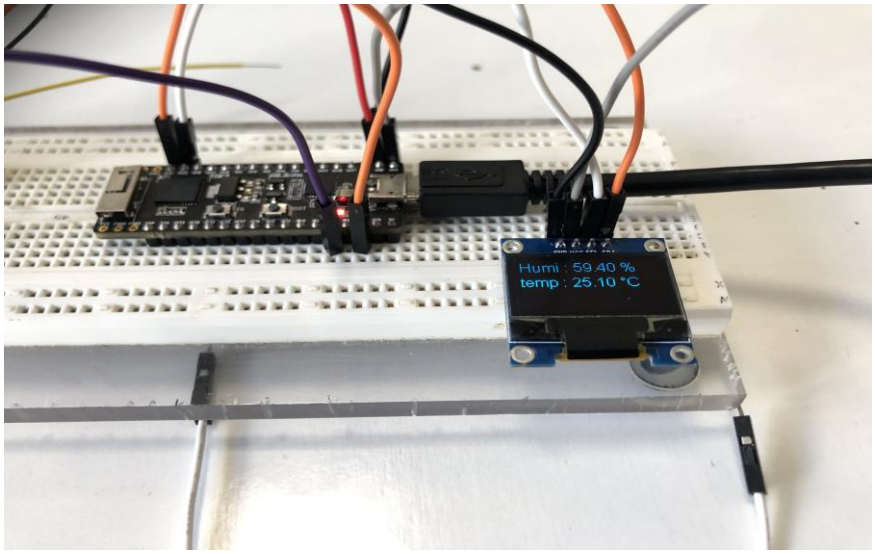
```
#include "DHT.h"
#include <Wire.h>
#include "SSD1306Wire.h"
#define DHTPIN 4
#define DHTTYPE DHT22

SSD1306Wire display(0x3c, SDA, SCL);
DHT dht(DHTPIN, DHTTYPE);

void setup() {
Serial.begin(9600);
Serial.println(F("DHTxx test!"));
dht.begin();
display.init();
display.flipScreenVertically();
display.setFont(ArialMT_Plain_10);

}

void loop() {
delay(2000);
```

```
float h = dht.readHumidity();
float t = dht.readTemperature();
float f = dht.readTemperature(true);

if (isnan(h) || isnan(t) || isnan(f)) {
Serial.println(F("Failed to read from DHT sensor!"));
return;

}

float hif = dht.computeHeatIndex(f, h);
float hic = dht.computeHeatIndex(t, h, false);

Serial.print(F("Humidity: "));
Serial.print(h);
Serial.print(F("% Temperature: "));
Serial.print(t);
Serial.print(F("°C \n"));

char humi[255];

sprintf(humi, "Humi : %.2f %%", h);

char temp[255];

sprintf(temp, "temp : %.2f °C", t);

display.clear();
display.setTextAlignment(TEXT_ALIGN_LEFT);
display.setFont(ArialMT_Plain_16);
display.drawString(0, 0, humi );
display.setFont(ArialMT_Plain_16);
display.drawString(0, 20, temp);
display.display();

            }
```

==========

e) Chercher le code associé en réutilisant la bonne fonction d'affichage du texte. Vous devrez respecter l'affichage sous la forme :

Ainsi le rendue suite au téléversements du code :



Conseil : On utilisera les fonctions **sprintf(x,y,z);** pour transformer les données ici des capteurs de type float en type string avant de les envoyer à l'afficheur. On utilisera aussi la fonction **strcat(x,y);** pour rajouter les unités (% et °C)

f) Téléverser le programme. Tester le bon fonctionnement.

g) **FAIRE VERIFIER LE BON FONCTIONNEMENT A L'ENSEIGNANT.**

PARTIE 3 : WIFI / PAGE WEB EMBARQUEE

On souhaite également que l'ESP32 devienne un serveur WEB connecté en WIFI à un point d'accès. La page web embarquée devra afficher la température et l'Humidité et la page devra se recharger dans un navigateur toutes les secondes.

En vous aidant du site https://randomnerdtutorials.com

a) Chercher le code associé.

Voici le code qui a permis de faire un point d'accès pour l'esp32

==========

```
#include <WiFi.h>


const char* ssid     = "MA WIFI";
const char* password = "azertyuiop";

void setup()
{
   Serial.begin(115200);
   Serial.println("\n[*] Creating AP");
   WiFi.mode(WIFI_AP);
   WiFi.softAP(ssid, password);
   Serial.print("[+] AP Created with IP Gateway ");
   Serial.println(WiFi.softAPIP());
}
```

==========

```
void loop(){}
```

b) Téléverser le programme. Tester le bon fonctionnement avec votre téléphone portable connecté en WIFI

Pour commencer on a utiliser comme base ce programme qui nous a permis d'avoir le rendu d'image depuis un serveur web via a l'adresse ip de l'esp32 qui est 172.20.10.2 .

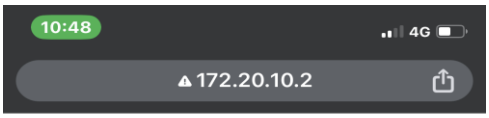
==========

```
#include <WiFi.h>   // Utilisation de la librairie WiFi.h
#include <WebServer.h>   // Utilisation de la librairie WebServer.h
```

```
const char* ssid = "test";  // Mettre le ssid de votre réseau Wifi
const char* password = "azertyuiop";   // Mettre le mot de passe de votre réseau Wifi
WebServer server(80);  // Permet l'affichage de la page d'accueil

void handleRoot(){  // Page d'accueil La page HTML est mise dans le String page

//  Syntaxe d'écriture pour être compatible avec le C++ / Arduino

// String page = " xxxxxxxx ";

// page += " xxxxx ";

// etc ...
  String page = "<!DOCTYPE html>";  // Début page HTML
    page += "<head>";
    page += "    <title>Serveur ESP32</title>";
    page += "    <meta http-equiv='refresh' content='60' name='viewport'
content='width=device-width, initial-scale=1' charset='UTF-8'/>";
    page += "</head>";
    page += "<body lang='fr'>";
    page += "    <h1>Serveur</h1>";
    page += "    <p>Ce serveur est hébergé sur un ESP32</p>";
    page += "    <i>Créé par Ali et Mdahoma</i>";
    page += "</body>";
    page += "</html>";  // Fin page HTML

    server.send(200, "text/html", page);  // Envoie de la page HTML
}

void handleNotFound(){  // Page Not found
  server.send(404, "text/plain","404: Not found");
}

void setup() {
  Serial.begin(115200);
  delay(1000);
  Serial.println("\n");
  WiFi.begin(ssid, password);  // Initialisation du Wifi
  Serial.print("Attente de connexion ...");

  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
```

```
  delay(100);
 }

 Serial.println("\n");  // Affichage "Connexion établie" et de l'adresse IP
 Serial.println("Connexion etablie !");
 Serial.print("Adresse IP: ");
 Serial.println(WiFi.localIP());

 server.on("/", handleRoot);  // Chargement de la page d'accueil
 server.onNotFound(handleNotFound);  // Chargement de la page "Not found"
 server.begin();  // Initialisation du serveur web
 Serial.println("Serveur web actif");
}

void loop() {
 server.handleClient();  // Attente de demande du client
}
```

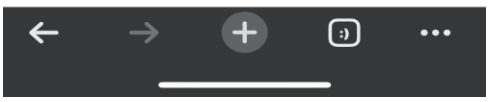
==========



Serveur

Ce serveur est hébergé sur un ESP32

Créé par Ali et Mdahoma



Puis on a modifié pour qu'on obtient pour qu'on obtient les températures et l'humidité en temps réel :

```
==========================================================================
#include <WiFi.h>
#include <WebServer.h>
#include "DHT.h"
#include <Wire.h>
#include "SSD1306Wire.h"

const char* ssid = "test";      // Mettez le SSID de votre réseau WiFi
const char* password = "azertyuiop";  // Mettez le mot de passe de votre réseau WiFi

#define DHTPIN 4
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
float temperature = 0.0;
float humidity = 0.0;

WebServer server(80);

SSD1306Wire display(0x3c, SDA, SCL);

void handleRoot() {
  String page = "<!DOCTYPE html>";
  page += "<html>";
  page += "<head>";
  page += "    <title>Serveur ESP32</title>";
  page += "    <meta http-equiv='refresh' content='60' name='viewport' content='width=device-width,
initial-scale=1' charset='UTF-8'/>";
  page += "    <script>";
  page += "      function updateData() {";
  page += "        var xhttp = new XMLHttpRequest();";
  page += "        xhttp.onreadystatechange = function() {";
  page += "          if (this.readyState == 4 && this.status == 200) {";
  page += "            var data = JSON.parse(this.responseText);";
  page += "            document.getElementById('temperature').innerHTML = 'Température : ' +
data.temperature + ' °C';";
  page += "            document.getElementById('humidity').innerHTML = 'Humidité : ' + data.humidity + '
%';";
  page += "          }";
  page += "        };";
  page += "        xhttp.open('GET', '/data', true);";
```

```
  page += "       xhttp.send();";
  page += "     }";
  page += "     setInterval(updateData, 5000);";  // Met à jour les données toutes les 5 secondes
  page += "    </script>";
  page += "</head>";
  page += "<body lang='fr'>";
  page += "    <h1>Serveur</h1>";
  page += "    <p>Ce serveur est hébergé sur un ESP32</p>";
  page += "    <i>Créé par Ali et Mdahoma</i>";
  page += "    <p id='temperature'>Température : en attente...</p>";
  page += "    <p id='humidity'>Humidité : en attente...</p>";
  page += "</body>";
  page += "</html>";

  server.send(200, "text/html", page);
}

void handleData() {
  String jsonData = "{\"temperature\":" + String(temperature, 2) + ",\"humidity\":" + String(humidity,
2) + "}";
  server.send(200, "application/json", jsonData);
}

void setup() {
  Serial.begin(115200);
  delay(1000);
  Serial.println("\n");

  WiFi.begin(ssid, password);
  Serial.print("Attente de connexion ...");

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(100);
  }

  Serial.println("\n");
  Serial.println("Connexion établie !");
  Serial.print("Adresse IP: ");
  Serial.println(WiFi.localIP());

  dht.begin();

  server.on("/", handleRoot);
  server.on("/data", handleData);  // Ajout d'une nouvelle route pour les données

  server.begin();
```

```
  Serial.println("Serveur web actif");

  display.init();
  display.flipScreenVertically();
  display.setFont(ArialMT_Plain_10);
}

void loop() {
  server.handleClient();

  delay(2000);

  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  temperature = t;
  humidity = h;

  char humi[255];
  sprintf(humi, "Humi : %.2f %%", h);

  char temp[255];
  sprintf(temp, "temp : %.2f °C", t);

  display.clear();
  display.setTextAlignment(TEXT_ALIGN_LEFT);
  display.setFont(ArialMT_Plain_16);
  display.drawString(0, 0, humi);
  display.setFont(ArialMT_Plain_16);
  display.drawString(0, 20, temp);
  display.display();
}
```

==========

Grace a ce code QR on a une courte vidéo qui nous montre l’actualisation,

en temps réel de la température et de l'humidité.

c) **FAIRE VERIFIER LE BON FONCTIONNEMENT A L'ENSEIGNANT.**