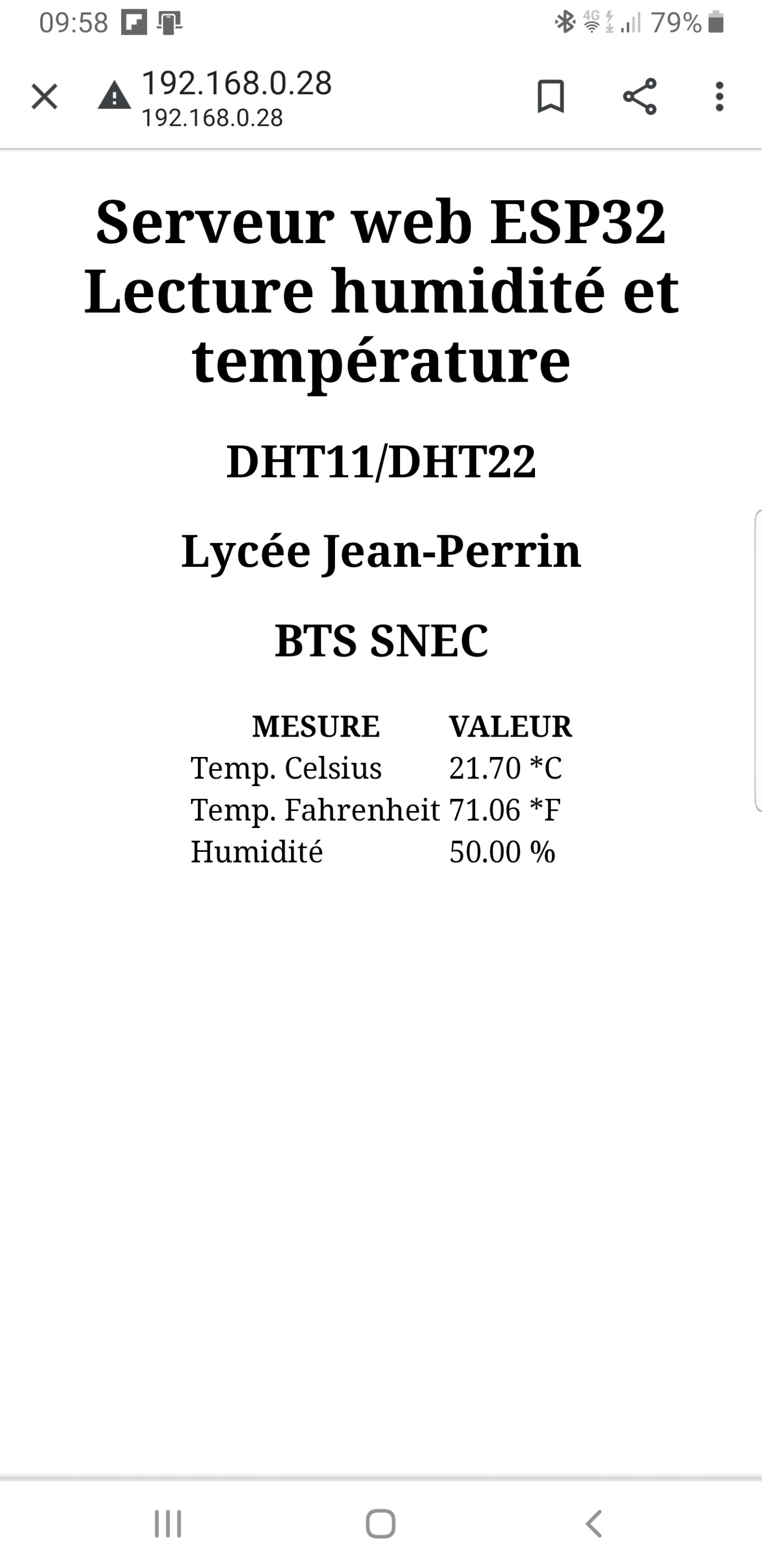
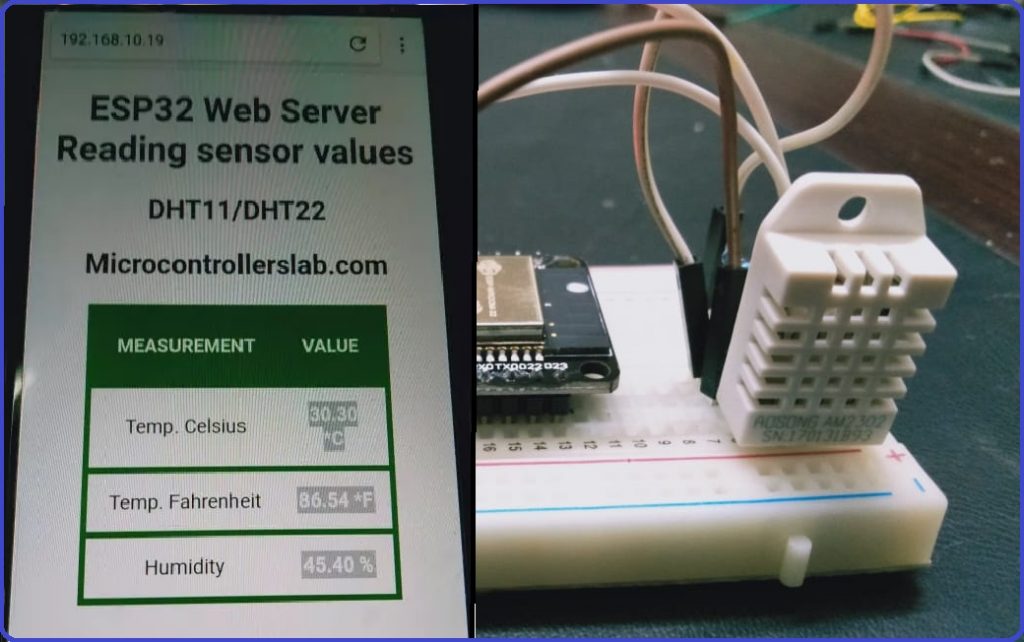
# Interface dht11/dht22 avec esp32 et affichage sur serveur Web

Dans ce didacticiel de serveur Web ESP32, nous verrons comment **afficher les valeurs des capteurs sur le serveur Web**. dans ce didacticiel, vous apprendrez à **interfacer DHT11, DHT21 et DHT22 avec l’**[**ESP32**](https://microcontrollerslab.com/esp32-development-board/)et comment afficher ses valeurs sur un serveur Web. Les capteurs DHT11 et DHT22 sont utilisés pour mesurer la température et l’humidité. les valeurs de température et d’humidité comme ceci :

[](https://microcontrollerslab.com/wp-content/uploads/2019/03/interfacing-DHT11-DHT22-with-ESP32-and-display-values-on-web-server.jpg)

Cette page Web sera utilisée pour afficher la valeur de température à la fois dans Celcius et Fahrenheit et le pourcentage d’humidité.

Avant de commencer ce projet, il existe quelques concepts de base nécessaires pour comprendre ce projet. Ce sont les lectures suggérées.

Interface DHT11 , DHT22/ AM2302 à ESP32 – Diagramme de câblage

Les diagrammes de connexion pour les deux sont affichés dans les images ci-dessous. Comme vous pouvez le voir les connexions sont simples.

Diagramme de câblage de dht11 interfaçage avec ESP32Diagramme de câblage de dht22 interancing avec ESP32

Pinout des deux capteurs DHT11 et DHT22 sont les mêmes que vous pouvez le voir dans l’image.

* La première broche pour les deux capteurs est la broche Vcc. Vous devez le connecter avec la broche 3.3volt de ESP32.
* La deuxième broche est la broche de données à travers nous obtenons des données pour la température et l’humidité. Connectez ce terminal avec GPIO4 d’ESP32 et connectez également la goupille de données à une résistance de traction de 10 k comme le montrent les diagrammes de câblage. La résistance à la traction est utilisée pour maintenir la goupille de données haute pour une bonne communication entre le microcontrôleur et le capteur. Vous pouvez consulter la feuille de données de [DHT11](https://www.mouser.com/ds/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf) et [DHT22](https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf) pour obtenir plus d’informations à ce sujet. DHT22 est également connu avec un nom AM2302.
* La troisième broche n’est pas utilisée. Nous ne l’utiliserons donc pas.
* La quatrième broche est un terminal de terrain d’entente. Connectez-le avec la broche au sol de la planche ESP32.
* Vous pouvez placer à la fois le capteur et la planche sur la planche à pain et utiliser quelques fils de cavalier pour faire des connexions.

installation de la bibliothèque DHT11 et DHT22 à Arduino

DHT11 et DHT22 fournit la sortie de température et d’humidité dans le format de sortie numérique complexe qui ne peut pas être lu directement avec des broches GPIO sans écrire aucune technique qui peut lire ces signaux de sortie. Ces capteurs fournissent des données par le biais d’un seul protocole de communication bidirectionnelle. Une communication à processus unique se compose de 40 bits. Mais nous n’avons pas besoin de nous soucier du fonctionnement de ces capteurs et du protocole sur lequel nous pouvons recevoir ces données. Nous avons une bibliothèque Arduino pour les capteurs DHT qui peuvent être facilement utilisés pour obtenir des valeurs de température et d’humidité uniquement en appelant deux lignes de fonctions. Nous verrons plus tard comment le faire. Voyons maintenant comment installer la bibliothèque DHT à Arduino. Cette bibliothèque est fournie par Adafruit. Procédez comme suit pour installer la bibliothèque de capteurs DHT :

* Accédez au menu Croquis>> inclure le fichier et cliquer sur l’option Gérer les bibliothèques.
* Lorsque vous cliquez sur l’option Gérer les bibliothèques, vous obtiendrez cette fenêtre. Dans cette fenêtre écrire '**capteur DHT**' dans la barre de recherche et appuyez sur entrer.
* Vous verrez de nombreuses options disponibles pour les bibliothèques DHT11 et DHT22. Sélectionnez Bibliothèque Adafruit et cliquez sur le bouton Installer. Vous pouvez sélectionner la dernière version dans la fenêtre de version.
* La même bibliothèque peut être utilisée pour le capteur DHT11 et DHT22/AM2302.
* Après cela, vous verrez le message de la bibliothèque correctement installé.
* Maintenant, cliquez sur le bouton fermer. Cette bibliothèque est maintenant incluse avec succès dans le gestionnaire de bibliothèque d’Arduino IDE.
* C’est une bibliothèque très puissante. Parce que nous pouvons utiliser la même bibliothèque pour différents types de microcontrôleurs comme Arduino et stm32 microcontrôleurs.

Adafruit fournit également des bibliothèques pour d’autres capteurs. Ils ont donc fourni un package de support qui est utilisé pour gérer toutes les bibliothèques de capteurs. Nous devons également installer la bibliothèque **de capteurs unifiés Adafruit.**Pour l’installer, collez la barre de recherche **du capteur unifié Adafruit**et sélectionnez cette option et cliquez sur le bouton installer.

Vous pouvez vérifier d’autres tutoriels qui ont utilisé ce capteur:

Créer un serveur Web ESP32 en mode station pour afficher les valeurs du capteur

Jusqu’à présent, vous avez appris sur le diagramme de câblage et comment installer des bibliothèques pour ces capteurs. Passons maintenant à la partie la plus intéressante de cet article. Nous allons créer un serveur Web ESP32 en mode station. Ce serveur Web servira les pages Web aux clients qui tentent d’accéder à ce serveur Web via un réseau existant à l’aide d’une adresse IP d’ESP32. Nous enregistrerons la page Web à l’intérieur d’ESP32 et chaque fois qu’un client Web fera la demande via HTTP pour cette page Web via l’adresse IP, le serveur Web ESP32 enverra ces pages Web au client via le protocole HTTP.

Maintenant, copiez le code ci-dessous dans Arduino IDE et téléchargez ce code sur la carte ESP32. Vous devez apporter des modifications au nom du code ssid et du mot de passe comme indiqué ci-dessous. En ssid, écrivez le nom de votre réseau WiFi et dans mot de passe écrire mot de passe de votre WiFi.

Code pour le serveur Web ESP32

Maintenant, téléchargez ce code en cliquant sur le bouton de téléchargement.

#include <WiFi.h>

#include <Wire.h>

#include "DHT.h"

// Uncomment one of the lines below for whatever DHT sensor type you're using!

//#define DHTTYPE DHT11 // DHT 11

//#define DHTTYPE DHT21 // DHT 21 (AM2301)

#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

//DHT Sensor;

uint8\_t DHTPin = 4;

DHT dht(DHTPin, DHTTYPE);

float Temperature;

float Humidity;

const char\* ssid = "Enter SSID here";

const char\* password = "Enter Password here";

WiFiServer server(80);

String header;

void setup() {

Serial.begin(115200);

pinMode(DHTPin, INPUT);

dht.begin();

Serial.print("Connecting to Wifi Network");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("Successfully connected to WiFi.");

Serial.println("IP address of ESP32 is : ");

Serial.println(WiFi.localIP());

server.begin();

Serial.println("Server started");

}

void loop()

{

Temperature = dht.readTemperature();

Humidity = dht.readHumidity();

WiFiClient client = server.available();

if (client)

{

Serial.println("Web Client connected ");

String request = client.readStringUntil('\r');

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

client.println("<!DOCTYPE html><html>");

client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

client.println("</style></head><body><h1>ESP32 Web Server Reading sensor values</h1>");

client.println("<h2>DHT11/DHT22</h2>");

client.println("<h2>Microcontrollerslab.com</h2>");

client.println("<table><tr><th>MEASUREMENT</th><th>VALUE</th></tr>");

client.println("<tr><td>Temp. Celsius</td><td><span class=\"sensor\">");

client.println(dht.readTemperature());

client.println(" \*C</span></td></tr>");

client.println("<tr><td>Temp. Fahrenheit</td><td><span class=\"sensor\">");

client.println(1.8 \* dht.readTemperature() + 32);

client.println(" \*F</span></td></tr>");

client.println("<tr><td>Humidity</td><td><span class=\"sensor\">");

client.println(dht.readHumidity());

client.println(" %</span></td></tr>");

client.println("</body></html>");

client.stop();

client.println();

Serial.println("Client disconnected.");

Serial.println("");

}

}

Obtention de l’adresse IP

Après le téléchargement du codage, ouvrez le moniteur de série d’Arduino IDE. vous verrez ces messages sur le moniteur de série.

Tout d’abord, vous verrez le message « Connexion à un réseau WiFi ». Après la connexion avec succès d’ESP32 au WiFi, vous verrez le message de connexion avec succès et l’adresse IP d’ESP32. Cette adresse IP sera utilisée par les clients Web pour accéder aux pages Web sur n’importe quel navigateur Web. Notez cette adresse IP, nous en aurons besoin pour accéder au serveur Web.

Accès au serveur Web

Maintenant, ouvrez n’importe quel navigateur Web et entrez l’adresse IP que nous avons noté dans la dernière étape. Après avoir appuyez sur Entrée, vous obtiendrez cette sortie dans le navigateur Web au format de table. Sa température montrant dans Celcius et Fahrenheit et aussi l’humidité en pourcentage.

Travail du code

Maintenant, je vais vous expliquer le travail de croquis ligne par ligne. Au début de l’esquisse, nous avons ajouté trois bibliothèques pour le pilote WiFi, capteurs DHT et bibliothèque de fil. Ces bibliothèques sont nécessaires pour inclure pour utiliser ses fonctions associées dans le code.

#include <WiFi.h> // Include library for WiFi

#include <Wire.h> // Add library for Wire

#include "DHT.h" //Add library for DHT11 and DHT11

Ceci est utilisé pour définir quel type de capteur DHT nous voulons utiliser. Vous pouvez l’utiliser avec le capteur DHT11, DHT21 et DHT22. Vous devez décomplexer la ligne en fonction du capteur que vous utilisez. Par exemple, j’ai utilisé DHT22 dans cet exemple, j’ai non commenté cela et d’autres sont restent commentés. La variable DHTTYPE stocke le nom du capteur que nous utilisons.

// Uncomment the line according to your selection of sensing device

//#define DHTTYPE DHT11 // DHT 11

//#define DHTTYPE DHT21 // DHT 21 (AM2301)

#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321

Nous avons défini un nom de goupille GPIO avec lequel nous connecterons le capteur DHT11/DHT22. Nous utilisons la broche GPIO numéro quatre. Par conséquent, nous devons définir la broche numéro quatre.

uint8\_t DHTPin = 4;

Cette ligne créera l’objet de DHT en fonction de notre numéro d’épingle défini et du type DHT. Nous avons défini ces deux paramètres dans les dernières étapes.

DHT dht(DHTPin, DHTTYPE);

Ces deux variables de type flotteur seront utilisées pour stocker les valeurs de température et d’humidité.

float Temperature;

float Humidity;

Ces variables stockent le nom et le mot de passe du réseau WiFi.

const char\* ssid = "Enter the name of your network";

const char\* password = "Enter the password here";

Cette ligne créera l’instance de la bibliothèque de serveurs Web sur le port numéro 80. Pour le protocole HTTP, 80 est un numéro de port par défaut. Nous allons donc utiliser ce numéro de port par défaut. L’avantage de l’utilisation de ce numéro de port par défaut est que nous n’aurons pas besoin de spécifier le numéro de port lors de l’accès au serveur Web via une adresse IP.

WiFiServer server(80);

String header; //Variable used to store requests from web clients

Fonction d’installation intérieure

Maintenant à l’intérieur de cette fonction, la première ligne va initialiser la communication en série avec le taux de baud de 115200 et la fonction pinMode fera de la broche GPIO quatre comme une broche d’entrée numérique. La fonction dht.begin() commence à lire les valeurs du capteur DHT11/DHT22.

Serial.begin(115200);

pinMode(DHTPin, INPUT);

dht.begin();

Maintenant, cette partie de l’esquisse va essayer de se connecter avec un réseau WiFi et après avoir réussi à se connecter avec le réseau, il affichera le message et l’adresse IP de la carte ESP32.

Serial.print("Connecting to Wifi Network"); // Display this string on serial monitor

Serial.println(ssid); // Print name of ssid on the serial monitor

WiFi.begin(ssid, password); // It will start connecting to the network

while (WiFi.status() != WL\_CONNECTED) // This condiction checks either connected to

{ // to a network or not

delay(500); // it will keep try after every half second

Serial.print("."); // It will exist this loop upon connection

}

Serial.println("");

Serial.println("Successfully connected to WiFi."); // Display this message on Serial monitor

Serial.println("IP address of ESP32 is : ");

Serial.println(WiFi.localIP()); // This function will display IP address of ESP32 or station mode ESP32

server.begin(); // This statement will start the server. Now web client can connect to ESP32 web server

Serial.println("Server started"

À l’intérieur de la fonction de boucle

Ces lignes obtiennent les valeurs de température et d’humidité et les enregistrent dans ces deux variables.

Temperature = dht.readTemperature();

Humidity = dht.readHumidity();

Cette fonction server.available vérifie si un client Web est disponible ou non. S’il est disponible logique valeur élevée sera stocké à l’intérieur de cette variable client.

WiFiClient client = server.available();

si la valeur de la variable client est logiquement élevée, cette condition « » devient vraie. À l’intérieur de cette condition « » nous affichons la page Web et les valeurs des données du capteur sur la page Web. Chaque demande HTTP d’un client Web se termine par '\r', de sorte que cette fonction client.readStringTling() écoute la demande du client Web jusqu’à ce que \r ne soit pas trouvé.

if (client)

{

Serial.println("Web Client connected ");

String request = client.readStringUntil('\r');

Après cela, la fonction client.println() envoie des fichiers HTML à la page Web. Donc, pour comprendre cette partie, me permet de comprendre certaines bases de ce HTML et comment faire des tables en HTML parce que nous affichons des données au format de table.

Pour créer une table en HTML, nous utilisons <table> et </table> tags.

* Comme vous le savez, chaque table se compose de lignes et de colonnes. Ainsi ,lt;tr> et </tr> les balises sont utilisées pour ajouter une ligne à une table.
* Chaque table a également un titre. Pour ajouter un titre à une table, nous utilisons <th> et </th> tags.
* Pour définir le nombre de colonnes ou de cellules dans chaque ligne, nous utilisons ces balises <td> et </td> .

Dans cet exemple, nous mesurons trois valeurs. Par conséquent, nous aurons besoin de trois lignes et deux cellules et un titre avec mesure et message de valeur.

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" href="data:,">

</head>

<body>

<table>

<tr>

<th>MEASUREMENT</th>

<th>VALUE</th>

</tr>

<tr>

<td>Temperature in Celsius</td>

<td>--- \*C</td></tr>

<tr>

<td>Temperature in Fahrenheit</td>

<td>--- \*F</td>

</tr>

<tr>

<td>Humidity</td>

<td>--- %</td>

</tr>

</body>

</html>

Donc, ce code va créer une table qui ressemblera à ceci:

Ainsi, ce code crée d’abord la tête avec le message de « esurc » dans une cellule et le message de « a valeu » dans une autre cellule. Après cela, nous avons créé trois lignes pour afficher les valeurs de température et d’humidité. Maintenant permet de venir à Arduino programmation partie de ce projet d’affichage des valeurs des capteurs sur le serveur Web à l’aide ESP32.

Après les demandes de client Web pour une page Web, le serveur Web ESP32 répond au client Web avec un fichier HTML qui s’affichera sous forme de page Web dans le navigateur Web. La fonction client.println() est utilisée pour envoyer les fichiers HTML au client Web. Ce code enverra les fichiers HTML au client Web ainsi que les valeurs de température et d’humidité sous forme tabulaire.

client.println("<!DOCTYPE html><html>");

client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

client.println("</style></head><body><h1>ESP32 Web Server Reading sensor values</h1>");

client.println("<h2>DHT11/DHT22</h2>");

client.println("<h2>Microcontrollerslab.com</h2>");

client.println("<table><tr><th>MEASUREMENT</th><th>VALUE</th></tr>");

client.println("<tr><td>Temp. Celsius</td><td><span class=\"sensor\">");

client.println(dht.readTemperature());

client.println(" \*C</span></td></tr>");

client.println("<tr><td>Temp. Fahrenheit</td><td><span class=\"sensor\">");

client.println(1.8 \* dht.readTemperature() + 32);

client.println(" \*F</span></td></tr>");

client.println("<tr><td>Humidity</td><td><span class=\"sensor\">");

client.println(dht.readHumidity());

client.println(" %</span></td></tr>");

client.println("</body></html>");

Pour envoyer la valeur de la température, vous envoyez cette valeur via la fonction **dht.readtemperature()** comme indiqué ci-dessous. Tout d’abord, il enverra une nouvelle entrée de ligne de table et enverra la valeur entre ces balises.

client.println("<tr><td>Temp. Celsius</td><td><span class=\"sensor\">");

client.println(dht.readTemperature());

client.println(" \*C</span></td></tr>");

Cette ligne arrêtera la connexion client Web.

client.stop();

Ajout de styles à une page Web pour paraître plus professionnel

Comme vous le voyez le tableau qui affiche des valeurs n’est pas attrayant sans couleurs et bordures de table. Alors maintenant, nous allons ajouter un peu de CSS à ce code et faire à plus professionnellement à la recherche de bonnes comme montré ci-dessous.

Ajoutez ce code à l’intérieur de la partie HTML avant la balise d’en-tête de fermeture </head>. vous verrez la page web comme indiqué ci-dessus. Vérifiez cette image pour savoir où vous devez ajouter ce code qui ajoutera du style à la page Web.

client.println("<style>body { text-align: center; font-family: \"Arial\", Arial;}");

client.println("table { border-collapse: collapse; width:40%; margin-left:auto; margin-right:auto;border-spacing: 2px;background-color: white;border: 4px solid green; }");

client.println("th { padding: 20px; background-color: #008000; color: white; }");

client.println("tr { border: 5px solid green; padding: 2px; }");

client.println("tr:hover { background-color:yellow; }");

client.println("td { border:4px; padding: 12px; }");

client.println(".sensor { color:white; font-weight: bold; background-color: #bcbcbc; padding: 1px; }");

CSS signifie Feuilles de style en cascade et il est utilisé pour ajouter des couleurs et des styles aux pages Web. Dans le code ci-dessus, CSS ajoute du style et des couleurs à chaque élément de la table comme les lignes, les cellules et le titre. vous pouvez vérifier plus d’informations sur la création de CSS et HTML [ici](https://www.w3schools.com/html/html_tables.asp).

C’est ainsi que nous pouvons interfacer dht11/dht22 avec esp32 et afficher les valeurs sur le serveur Web.

https://microcontrollerslab.com/esp32-dht11-dht22-web-server/