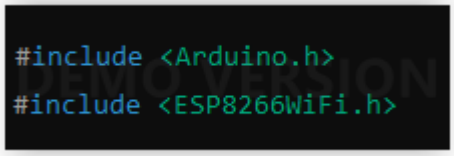
# ESP8266Sketch8_PageWebDynamique

Ce programme permet de contrôler une LED connectée à un ESP8266 via une interface web dynamique. L'ESP8266 se connecte à un réseau Wi-Fi et crée un serveur web, sur lequel une page web dynamique est servie. Cette page permet à l'utilisateur de contrôler l'état de la LED en temps réel, grâce à des requêtes asynchrones (AJAX) qui permettent de modifier l'état de la LED sans recharger la page.

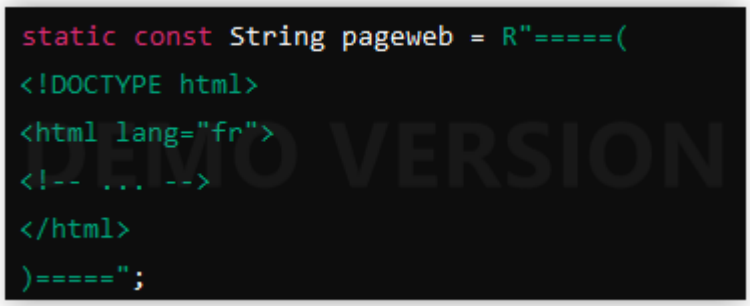
1. **Bibliothèques utilisées :**




Ces lignes importent les bibliothèques nécessaires.

- **Arduino.h** : C'est la bibliothèque standard pour les fonctions Arduino.
- **ESP8266WiFi.h** : Cette bibliothèque permet à l'ESP8266 de se connecter à un réseau Wi-Fi et d'agir en tant que serveur web.

**2. Page Web stockée en mémoire**




Ici, le contenu de la page web est écrit directement dans le programme sous forme de chaîne de caractères (String). Cette page contient :

- Un titre **"Commande LED"**.
- Deux boutons, un pour allumer et un pour éteindre la LED.
- Un petit script JavaScript pour envoyer des requêtes au serveur lorsqu'un bouton est cliqué, afin d'envoyer l'ordre **"allumer"** ou **"éteindre"** la LED.

**3. Connexion au Wi-Fi**

```
const char* SSID = "XXXXXX";       // Nom du réseau Wi-Fi
const char* password = "1234566"; // Mot de passe du réseau Wi-Fi
```

Ces lignes contiennent le **SSID** (nom) et le **mot de passe** du réseau Wi-Fi auquel l'ESP8266 va se connecter. Il faudra adapter ces valeurs en fonction du réseau local.

**4. Création du serveur**

```
WiFiServer MonServeur(80); // créer un serveur web sur le port 80
```

Le serveur web est configuré pour écouter sur le **port 80**, le port standard pour les communications HTTP. Ce serveur permettra à n'importe quel appareil sur le même réseau Wi-Fi de se connecter à l'ESP8266 et de contrôler la LED.

**5. Définition de la broche de la LED**

```
#define LED_PIN  2
```

La broche **2** est utilisée pour la LED. Si vous utilisez une carte comme le **Wemos D1 Mini**, la LED intégrée est généralement connectée à la broche 2. Si nécessaire, cela peut être changé selon la carte utilisée.

**6. Fonction pour envoyer la page web au client**

```
void envoyer_pageweb(){
    MonClient.println("HTTP/1.1 200 OK");
    MonClient.println("Content-Type: text/html");
    MonClient.println("Content-Length: " + String(pageweb.length()));
    MonClient.println("Connection: close");
    MonClient.println("");
    MonClient.println(pageweb);
}
```

Cette fonction envoie la page web au client (navigateur web) lorsqu'il accède à l'adresse IP de l'ESP8266. Elle prépare une réponse HTTP en envoyant les en-têtes et le contenu HTML.

**7. Connexion au réseau Wi-Fi**

```
void setup() {
  // ... Connexion au Wi-Fi ...
}
```

Dans cette partie :

- **WiFi.disconnect()** : L'ESP8266 déconnecte tout réseau Wi-Fi précédent.
- **WiFi.mode(WIFI_STA)** : Met l'ESP8266 en mode station, c'est-à-dire qu'il se connecte à un réseau Wi-Fi comme un client normal.
- **WiFi.begin(SSID, password)** : Démarre la connexion au réseau Wi-Fi.
- **WiFi.localIP()** : Cette fonction renvoie l'adresse IP locale de l'ESP8266, qui sera utilisée pour accéder au serveur web via un navigateur.

**8. Fonction loop()**

La fonction **loop** s'exécute en continu et contient la logique principale pour traiter les requêtes envoyées au serveur web :

**a) Attendre une connexion client**

```
MonClient = MonServeur.accept();
if (!MonClient) {
    return;
}
```

L'ESP8266 attend qu'un client (comme un navigateur) se connecte au serveur web. Si aucun client ne se connecte, il n'exécute pas le reste du code.

**b) Lire la requête HTTP**

```
String request = MonClient.readStringUntil('\r');
```

Lorsque le client se connecte, il envoie une **requête HTTP**. Ici, l'ESP8266 lit cette requête jusqu'à la fin de la ligne.

**c) Analyser et traiter la requête**

La requête reçue peut être :

- **GET /** : Le client demande la page web principale. Le programme envoie la page web.
- **GET /LEDON** : Le client demande d'allumer la LED. Le programme allume la LED et renvoie un message indiquant que la LED est allumée.
- **GET /LEDOFF** : Le client demande d'éteindre la LED. Le programme éteint la LED et renvoie un message indiquant que la LED est éteinte.

**d) Envoyer des réponses**

```
void envoyer_reponsevide(const String& etat) {
    MonClient.println("HTTP/1.1 200 OK");
    MonClient.println("Connection: close");
    MonClient.println("");
    MonClient.println(etat);
}
```

Cette fonction envoie une réponse HTTP avec l'état actuel de la LED (allumée ou éteinte) au client, permettant à la page web de mettre à jour le texte affichant l'état de la LED.