ESP8266Sketch7_ServeurWeb2

Ce programme utilise un module ESP8266 pour créer un serveur web permettant de contrôler une LED via une interface web.

Différence avec le code précédent :

- Contrairement au code précédent où le serveur traitait uniquement des requêtes simples en renvoyant du texte HTML basique, ici, une interface web plus complète est générée, avec des boutons pour allumer et éteindre la LED.
- Le nouveau code utilise la bibliothèque **ESP8266WebServer** pour simplifier la gestion des requêtes HTTP et permet de structurer l'interface utilisateur avec des boutons HTML. La LED peut être contrôlée par des requêtes HTTP vers des chemins définis ("/ledON" et "/ledOFF").
- Une page web plus dynamique est générée, affichant l'état de la LED en temps réel (allumée ou éteinte), ce qui rend l'interaction avec l'utilisateur plus intuitive et conviviale.

```
#include "Arduino.h"
#include "ESP8266WiFi.h"
#include <ESP8266WebServer.h>

const byte led = LED_BUILTIN;
const char *SSID = "xxxxxxx";
const char *PASSWORD = "123456";

String Page_Client(bool etat_led) {
    String etat = etat_led ? "allumée" : "éteinte";
    return R"=====(
    <!DOCTYPE html>
    <html>
      <head>
        <meta charset="UTF-8">
        <title>Contrôler la LED</title>
      </head>
      <body>
        <h1>Contrôler la LED</h1>
        <p>La LED est actuellement )=====" + etat + R"=====(</p> <!-- Affiche l'état de la LED -->
        <br><br>
        <a href="ledON"><button>ALLUMER</button></a>  <!-- Bouton pour allumer la LED -->
        <br><br><br>
        <a href="ledOFF"><button>ETEINDRE</button></a> <!-- Bouton pour éteindre la LED -->
      </body>
    </html>
    )=====";
}
```

```
ESP8266WebServer monWebServer(80);
void aff_page_web() {
    bool etat_led = digitalRead(led);
    monWebServer.send(200, "text/html", Page_Client(etat_led));
    Serial.println("-----> Page Web envoyée");
}

void allumer_led() {
    Serial.println("-----> led allumée");
    digitalWrite(led, HIGH);
    aff_page_web();
}

void eteindre_led() {
    Serial.println("-----> led éteinte");
    digitalWrite(led, LOW);
    aff_page_web();
}

void setup(){
  Serial.begin(115200);
  delay(10);
  Serial.println("\n");

  pinMode(led, OUTPUT);
  digitalWrite(led, LOW);

  Serial.print("Connexion à ");
  Serial.println(SSID);
  WiFi.begin(SSID, PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Connecté à ");
  Serial.println(SSID);
  Serial.print("Adresse IP: ");
  Serial.println(WiFi.localIP());

  monWebServer.on("/", aff_page_web);
  monWebServer.on("/ledON", allumer_led);
  monWebServer.on("/ledOFF", eteindre_led);

  monWebServer.begin();
  Serial.println("Serveur HTTP démarré");
}

void loop(){
  monWebServer.handleClient();
}
```

```
#include "Arduino.h"
#include "ESP8266WiFi.h"
#include <ESP8266WebServer.h>
```

- **Arduino.h** : Fournit les fonctions de base pour les opérations sur microcontrôleurs (entrées/sorties, délais, etc.).

- **ESP8266WiFi.h** : Bibliothèque qui permet de connecter le module ESP8266 à un réseau WiFi.

- **ESP8266WebServer.h** : Cette bibliothèque permet de créer un serveur web simple avec l'ESP8266.

```
const byte led = LED_BUILTIN;
const char *SSID = "Livebox-6280";
const char *PASSWORD = "hGwzonQpzUtjm4sRm6";
```

- **led** : Utilise la LED intégrée au module ESP8266 (variable associée à **LED_BUILTIN**).

- **SSID et PASSWORD** : Informations pour la connexion WiFi (nom du réseau et mot de passe).

```
String Page_Client(bool etat_led) {
    String etat = etat_led ? "allumée" : "éteinte";
    return R"=====(
    <!DOCTYPE html>
    <html>
      <head>
        <meta charset="UTF-8">
        <title>Contrôler la LED</title>
      </head>
      <body>
        <h1>Contrôler la LED</h1>
        <p>La LED est actuellement )=====" + etat + R"=====(</p> <!-- Affiche l'état de la LED -->
        <br><br>
        <a href="ledON"><button>ALLUMER</button></a>  <!-- Bouton pour allumer la LED -->
        <br><br><br>
        <a href="ledOFF"><button>ETEINDRE</button></a> <!-- Bouton pour éteindre la LED -->
      </body>
    </html>
    )=====";
}
```

Objectif de la fonction Page_Client

Page_Client : génère dynamiquement une page web en HTML qui permet de contrôler l'état d'une LED à distance via une interface simple.

Cette page contient :

- Un titre et un message indiquant l'état actuel de la LED (allumée ou éteinte).
- Deux boutons pour allumer ou éteindre la LED.

```
String Page_Client(bool etat_led)
```

- Utilisation de l'opérateur ternaire **? :** pour définir une chaîne de caractères **etat** en fonction de la valeur de **etat_led**.
  - Si **etat_led** est vrai (la LED est allumée), la variable **etat** prend la valeur "allumée".
  - Si **etat_led** est faux (la LED est éteinte), la variable **etat** prend la valeur "éteinte".

  Cette chaîne sera ensuite insérée dans le code HTML pour indiquer visuellement l'état de la LED sur la page web.

```
====" + etat + R"===
```

- **" + etat + "** permet d'insérer la variable **etat** (qui contient "allumée" ou "éteinte") dans le code HTML.

```
<a href="ledON"><button>ALLUMER</button></a>
<br><br><br>
<a href="ledOFF"><button>ETEINDRE</button></a>
```

- **<button>ALLUMER</button>** : Ce lien hypertexte avec un bouton permet à l'utilisateur d'envoyer une requête pour allumer la LED. Le lien pointe vers **"ledON"**, qui est associé à la fonction **allumer_led** dans le programme.

- **<button>ETEINDRE</button>** : De même, ce bouton permet d'envoyer une requête pour éteindre la LED en pointant vers **"ledOFF"**, qui appelle la fonction **eteindre_led**.

```
ESP8266WebServer monWebServer(80);
```

- **monWebServer(80)** : Déclare un serveur web qui écoute sur le port 80 (port par défaut pour HTTP).

```
void aff_page_web() {
    bool etat_led = digitalRead(led);
    monWebServer.send(200, "text/html", Page_Client(etat_led));
    Serial.println("-----> Page Web envoyée");
}
```

- **aff_page_web()** : Envoie la page web HTML générée par **Page_Client()** au client. Elle vérifie l'état actuel de la LED et affiche cette information dans la page.

- **monWebServer.send(200, "text/html", ...)** : Envoie une réponse HTTP avec un code **200 OK** et le type de contenu **HTML**.

```
void allumer_led() {
    Serial.println("-----> led allumée");
    digitalWrite(led, HIGH);
    aff_page_web();
}
```

- **allumer_led()** : Cette fonction est appelée lorsqu'un utilisateur clique sur le bouton "ALLUMER" sur la page web. Elle allume la LED et renvoie ensuite la page web mise à jour.

```
void eteindre_led() {
    Serial.println("-----> led éteinte");
    digitalWrite(led, LOW);
    aff_page_web();
}
```

- **eteindre_led()** : Similaire à **allumer_led()**, mais cette fonction éteint la LED lorsque le bouton "ETEINDRE" est cliqué.

```
void setup(){
  Serial.begin(115200);
  delay(10);
  Serial.println("\n");
```

- **Serial.begin(115200)** : Démarre la communication série avec une vitesse de transmission de **115200 bauds**. Cette communication permet d'envoyer des messages au moniteur série pour déboguer et suivre les actions du programme.

- **delay(10)** : Attend 10 millisecondes pour permettre à l'initialisation de se stabiliser.

- **Serial.println("\n")** : Ajoute une nouvelle ligne vide dans le moniteur série pour organiser la présentation des messages suivants.

```
pinMode(led, OUTPUT);
digitalWrite(led, LOW);
```

- **pinMode(led, OUTPUT)** : Définit la broche associée à la LED comme une sortie numérique, afin de contrôler la LED.

- **digitalWrite(led, LOW)** : Assure que la LED est initialement éteinte en définissant l'état de la broche à **LOW**.

```
Serial.print("Connexion à ");
Serial.println(SSID);
WiFi.begin(SSID, PASSWORD);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
```

- **WiFi.begin(SSID, PASSWORD)** : Démarre la connexion Wi-Fi à un réseau en utilisant le **SSID** et le **mot de passe** spécifiés.

- **while (WiFi.status() != WL_CONNECTED)** : Tant que l'ESP8266 n'est pas connecté au réseau Wi-Fi, le programme attend. Pendant ce temps, des points . sont imprimés dans le moniteur série chaque demi-seconde pour indiquer que la tentative de connexion est en cours.

```
Serial.println();
Serial.print("Connecté à ");
Serial.println(SSID);
Serial.print("Adresse IP: ");
Serial.println(WiFi.localIP());
```

- **WiFi.localIP()** : Affiche l'adresse IP locale attribuée à l'ESP8266 par le routeur.

```
monWebServer.on("/", aff_page_web);
monWebServer.on("/ledON", allumer_led);
monWebServer.on("/ledOFF", eteindre_led);
```

- **monWebServer.on("/")** : Spécifie la fonction à exécuter lorsque la racine du serveur web ("/") est demandée par un client. Ici, la fonction **aff_page_web** est appelée pour envoyer la page de contrôle de la LED.

- **monWebServer.on("/ledON")** et **monWebServer.on("/ledOFF")** : Spécifient les actions à effectuer lorsque l'utilisateur clique sur les boutons "ALLUMER" ou "ÉTEINDRE" la LED. Les fonctions **allumer_led** et **eteindre_led** sont respectivement appelées pour allumer ou éteindre la LED.

```
monWebServer.begin();
Serial.println("Serveur HTTP démarré");
```

- **monWebServer.begin()** : Démarre le serveur web sur le port 80 (par défaut pour HTTP).

- **Serial.println("Serveur HTTP démarré")** : Affiche dans le moniteur série que le serveur HTTP est en cours d'exécution.

```
void loop(){
  monWebServer.handleClient();
}
```

- **monWebServer.handleClient()** : Cette fonction est essentielle au fonctionnement du serveur web. Elle vérifie en permanence si un client (comme un navigateur web) tente d'accéder au serveur. Si une requête est détectée, elle est traitée (par exemple, l'envoi de la page web ou l'exécution d'une action comme allumer ou éteindre la LED).

  Sans cette ligne, le serveur ne pourrait pas répondre aux requêtes des utilisateurs.