Table des matières

1) Introduction.....1
2) Cahier des charges.....1
3) Objectif du TP.....1
4) Questions.....1

1) Introduction

Dans la suite des expérimentations sur l'utilisation des possibilités de la connexion wifi d'une carte de développement utilisant le SoC ESP32, nous souhaitons réaliser un objet connecté permettant la commande de l'allumage et de l'extinction d'un LED depuis tout appareil connecté en wifi à l'ESP32.

2) Cahier des charges

Le cahier des charges décrivant le fonctionnement du système est décrit ci-dessous

1-Le module wifi de l'ESP32 doit être configuré en point d'accès wifi autorisant ainsi des clients (smartphones, ordinateurs, etc.) à s'y connecter.

2-L'ESP32 sera équipé d'une LED et d'une résistance de limitation de courant.

3-L'ESP32 doit comporter un serveur TCP qui permettra de traiter les requêtes GET provenant des clients.

4-Lorsqu'on client se connecte à l'ESP32, il pourra accéder à une page web lui permettant d'allumer ou d'éteindre la LED.

3) Objectif du TP

Réaliser un objet connecté répondant au cahier des charges décrit précédemment.

On réalise donc le programme suivant en C++ qui répondra au cahier des charges :

```
#include <WiFi.h> //On inclut la librairie WiFi.h

const char *ssid = "ESP-SAM"; // Le nom choisi pour le SSID
const char *password = "maison1234"; // Le mot de passe choisi


// ********************************************************
// Serveur TCP sur le port 80 qui répondra aux requêtes HTTP
// ********************************************************
WiFiServer server(80);

void setup() {

  pinMode(25, OUTPUT); // LED comme une sortie sur le port 25

  Serial.begin(115200); // Moniteur à 115200 bauds
  WiFi.disconnect(); // Déconnexion du wifi pour éviter les bugs
  WiFi.mode(WIFI_AP); // seul le mode AP est activé
  WiFi.softAP(ssid, password,7,0,5); // Utilise le ssid et le mdp
SSID,password,channel,hidden ssid or not, max number of connexions
(les trois derniers sont optionnels)

  Serial.println();
  Serial.print("IP address: "); // Ecriture de "IP address"
  Serial.println(WiFi.softAPIP());

  server.begin(); // On démarre le serveur TCP
    Serial.println("SERVEUR TCP DEMARRE") ;
    Serial.print(WiFi.localIP()); // Ecriture de l'adresse IP
    Serial.println("");
}

// ******************
// Boucle principale
// ******************
void loop(void)
{
    // Vérifier si un client s'est connecté
    WiFiClient client = server.available();
    if (!client) {
        return;
    }
    Serial.println("");
    Serial.println("Nouveau client"); // Ecriture "Nouveau client"
```

```
    // Attendre que les données provenant du client soient
disponibles
    while(client.connected() && !client.available()){
        delay(1);
    }

    // Lecture de la première ligne de la requête HTTP
    String req = client.readStringUntil('\r');

    // La première ligne de la requête HTTP ressemble à "GET /path
HTTP/1.1"
    // Récupération de la partie "/path" en utilisant les espaces
    int addr_start = req.indexOf(' ');
    int addr_end = req.indexOf(' ', addr_start + 1);
    if (addr_start == -1 || addr_end == -1) {
        Serial.print("Invalid request: ");
        Serial.println(req);
        return;
    }
    req = req.substring(addr_start + 1, addr_end);
    Serial.print("Request: ");
    Serial.println(req);
    client.flush();
// ************************************************
// A présent, dans la variable req se trouve le /path
// ************************************************
    String s; // Cette variable s servira à construire la page
html à transmettre au client

    if (req == "/")
    {
        // Insérer ici les actions d'interaction si nécessaire.

        IPAddress ip = WiFi.softAPIP();
        String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' +
String(ip[2]) + '.' + String(ip[3]);
        s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!
DOCTYPE HTML>\r\n<html><meta charset=\"utf-8\">Hello from ESP32 at
";
        s += ipStr;
        s += "</html>\r\n\r\n";
        Serial.println("Sending 200");
    }
    else if (req == "/ON")
    {

        // Insérer ici les actions d'interaction si nécessaire.
```

```
          digitalWrite(25, HIGH);
          // en-tête de la page html
        s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!
DOCTYPE HTML>\r\n<html><meta charset=\"utf-8\">Vous avez envoyé OK
! ";
          // exemple d'un lien cliquable
        s += "<p>Cliquer ICI : <a href=\"ON\">ALLUMER</a></p>";
        s += "<p>Cliquer ICI : <a href=\"OFF\">ETEINDRE</a></p>";
           // ligne de fin
        s += "</html>\r\n\r\n";
        Serial.println("Sending 200");
    }
     else if (req == "/OFF")
    {

        // Insérer ici les actions d'interaction si nécessaire.
        digitalWrite(25, LOW);
          // en-tête de la page html
        s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!
DOCTYPE HTML>\r\n<html><meta charset=\"utf-8\">Vous avez envoyé
TRUC ! ";
          // exemple d'un lien cliquable
        s += "<p>Cliquer ICI : <a href=\"ON\">ALLUMER</a></p>";
        s += "<p>Cliquer ICI : <a href=\"OFF\">ETEINDRE</a></p>";
          // ligne de fin
        s += "</html>\r\n\r\n";
        Serial.println("Sending 200");
    }

    else // dans ce cas, aucune des commandes attendues n'a été
transmise
    {
        s = "HTTP/1.1 404 Not Found\r\n\r\n";
        Serial.println("Transmission de 404");
    }


//
*****************************************************************
*********
// On envoie la page au client (page construite selon l'un des cas
précédents)
//
*****************************************************************
*********
    client.print(s);
```

```
    Serial.println("Terminé avec le client");
}
```

Dans ce programme à son exécution il crée un serveur TCP et il le démarre. Ensuite il attend de voir si un client se connecte puis il attend tout le temps qu'il est disponible. L'ESP sert d'AP et quand le client est connecté dessus il peut passer des requêtes http. Le programme crée une variable qui va stocker le chemin de la requête. Ici dans le programme une fois le client connecté on ne peut passer que 2 types de requêtes :

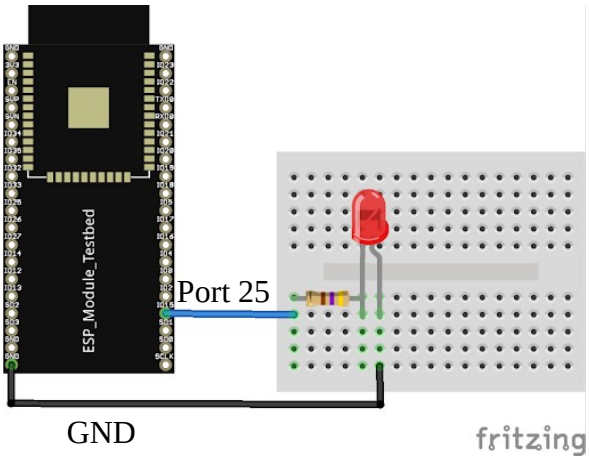
1) IP_ESP/ON (Cette requête va allumer la led relié à l'ESP puis afficher une page web avec écrit "Vous avez envoyé OK" et marquer les exemples des deux types de requêtes que nous pouvons passer : /ON ou /OFF)

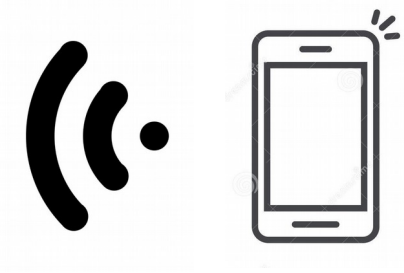
2) IP_ESP/OFF (Cette requête va éteindre la led relié à l'ESP puis afficher une page web avec écrit "Vous avez envoyé TRUC" et marquer les exemples des deux types de requêtes que nous pouvons passer : /ON ou /OFF)

Toutes les autres requêtes qui seront passées ne seront pas prises en compte par le serveur TCP et nous retournera une erreur 404 comme quand on ne trouve pas une page internet

Pour connecter le client il suffit simplement sur son téléphone d'aller dans les paramètres wifi et de trouver l'AP avec le nom qu'on a donné dans le programme. Ensuite une fois connecté on peut ouvrir le navigateur avec le client et taper l'un des deux types de requêtes

Voici donc un schéma de ce qui se passe :





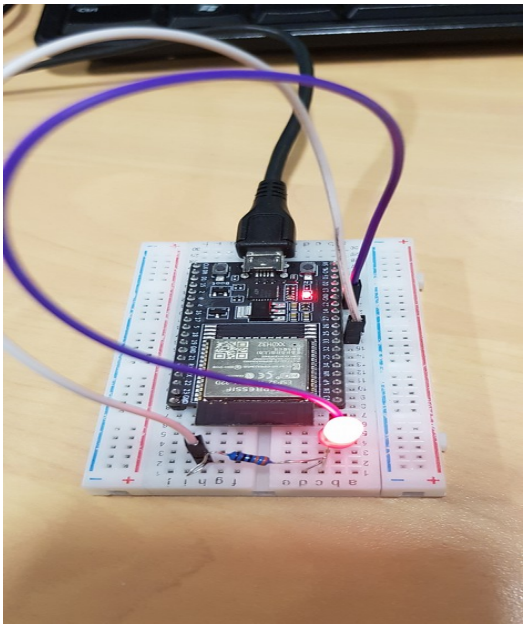
Requêtes passées sur le client dans un navigateur

4) Questions

1) Le fonctionnement entre le client et le serveur :

Un serveur TCP est crée sur l'ESP32 et une fois le programme lancé il attend qu'un client se connecte sur l'AP. Une fois connecté tant que le client est disponible le serveur attend. Sur le client on a plus qu'à passer des requêtes http qui seront envoyés sur le serveur TCP et exécuté sur l'ESP

2) Schéma de branchement de la led et de la résistance sur l'ESP :



Le formule pour trouver la résistance à utiliser est la suivante :

R = (U – Uled) / Iled

Ici la résistance pour notre led sera de 340 ohms

3) Le programme suivant effectue :

- La configuration de l'ESP32 (création d'un AP, serveur TCP, broche de sortie de la LED, application des librairies)

4) Le programme suivant détecte si un client est connecté et disponible sur l'AP. Tant que le client est connecté on attend. On crée une variable req qui va contenir le chemin de la requête http

5) Voici ce qu'il faut ajouter à la suite du programme pour répondre aux requêtes GET et afficher une page web qui affiche des exemples de requêtes à passer :

```
    String s; // Cette variable s servira à construire la page
html à transmettre au client

    if (req == "/")
    {
        // Insérer ici les actions d'interaction si nécessaire.

        IPAddress ip = WiFi.softAPIP();
        String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' +
String(ip[2]) + '.' + String(ip[3]);
        s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!
DOCTYPE HTML>\r\n<html><meta charset=\"utf-8\">Hello from ESP32 at
";
        s += ipStr;
        s += "</html>\r\n\r\n";
        Serial.println("Sending 200");
    }
    else if (req == "/ON")
    {

        // Insérer ici les actions d'interaction si nécessaire.
          digitalWrite(25, HIGH);
          // en-tête de la page html
        s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!
DOCTYPE HTML>\r\n<html><meta charset=\"utf-8\">Vous avez envoyé ON
! ";
          // exemple d'un lien cliquable
        s += "<p>Cliquer ICI : <a href=\"ON\">ALLUMER</a></p>";
        s += "<p>Cliquer ICI : <a href=\"OFF\">ETEINDRE</a></p>";
           // ligne de fin
        s += "</html>\r\n\r\n";
        Serial.println("Sending 200");
    }
     else if (req == "/OFF")
    {

        // Insérer ici les actions d'interaction si nécessaire.
        digitalWrite(25, LOW);
          // en-tête de la page html
        s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!
DOCTYPE HTML>\r\n<html><meta charset=\"utf-8\">Vous avez envoyé
OFF ! ";
          // exemple d'un lien cliquable
        s += "<p>Cliquer ICI : <a href=\"ON\">ALLUMER</a></p>";
        s += "<p>Cliquer ICI : <a href=\"OFF\">ETEINDRE</a></p>";
          // ligne de fin
        s += "</html>\r\n\r\n";
```

```
        Serial.println("Sending 200");
    }

    else // dans ce cas, aucune des commandes attendues n'a été
transmise
    {
        s = "HTTP/1.1 404 Not Found\r\n\r\n";
        Serial.println("Transmission de 404");
    }


//
*************************************************************
*********
// On envoie la page au client (page construite selon l'un des cas
précédents)
//
*************************************************************
*********
    client.print(s);

    Serial.println("Terminé avec le client");
}
```

6)

5) Pour aller plus loin

6)

7) On peut placer une pile entre la résistance et la led et dans le programme on met que si on détecte un courant alors c’est que la led est allumé et donc on peut envoyer une notification au client.