LABORATOR VIII – ESP32 PE POST DE CLIENT

1. Introducere

Lucrarea de laborator anterioară a prezentat cel mai folosit mod de lucru al ESP32, acela de punct de acces Wi-Fi. Alte dispozitive se pot conecta la ESP32, jucând rolul de client, putând solicita și primi date folosind protocolul HTTP.

În această lucrare de laborator vom explora un nou mod de lucru al plăcii, acela de Client, și vom studia modul în care două plăci pot schimba între ele date.

2. Moduri diferite de funcționare Wi-Fi

Plăcile de dezvoltare suportă trei standarde de comunicație wireless, și anume IEEE 802.11 *b* (Wi-Fi 1 - 11 Mbps) / *g* (Wi-Fi 3 - 54 Mbps) / *n* (Wi-Fi 4 - 600 Mbps). Luați în considerare că acestea sunt valori teoretice, dar în practică vitezele pot fi diferite. Plăcile pot fi configurate să accepte un singur standard, dar, implicit, ele folosesc toate cele trei standarde. În funcție de client, ESP32 va comuta la cea mai nouă versiune suportată, la fel ca un router Wi-Fi modern.

O placă ESP32 poate utiliza Wi-Fi în două moduri:

AP - modul **Access Point**, sau, în cazul nostru, **SoftAP**, ceea ce înseamnă **software enabled Access Point**, deoarece mecanismul este implementat de cod ce rulează pe microcontroller, nu de un hardware dedicat. Acest mod a fost utilizat în laboratorul de săptămâna trecută, și a permis altor dispozitive precum telefoane sau laptopuri să se conecteze la placă, pe post de clienți.

STA - modul **Station**, în care placa ESP32 funcționează ca client, și se va conecta la alte puncte de acces.

Lucrând în perechi, creați un punct de acces folosind codul exemplu de săptămâna trecută, și programați o altă placă să funcționeze pe post de client.

Nu uitați să schimbați numele SSID-ului (și poate parola) în ceva unic înainte de a încărca codul pe placă. Puteți adăuga o parolă la server. Asigurați-vă că SSID-ul și parola sunt setate identic la server și la client.

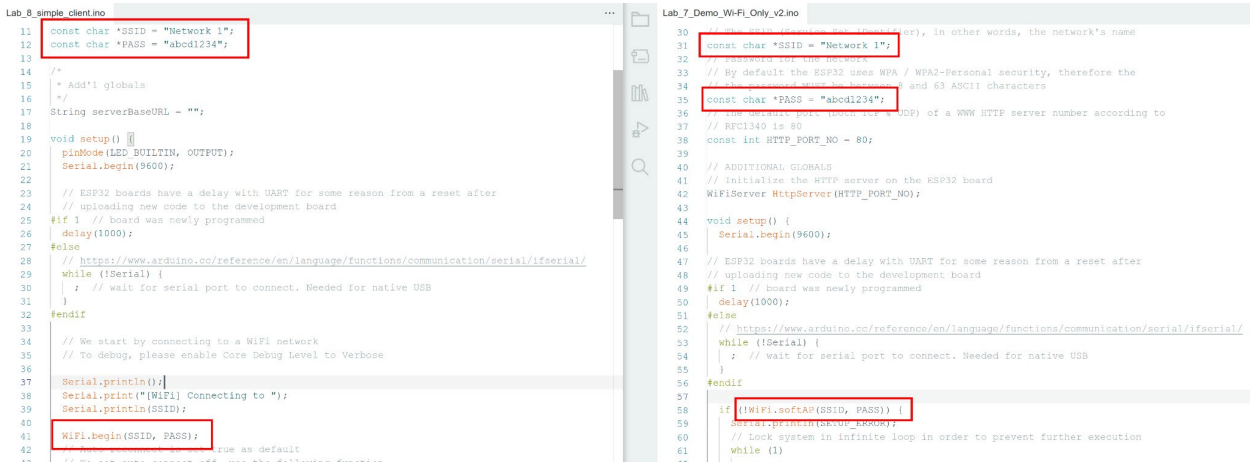

Figura 1. Asigurați-vă că SSID-ul și parola se potrivesc la server și la client.

Codul exemplu pentru client este prezentat mai jos. El poate fi descărcat și de la adresa https://github.com/UTCN-AC-CS-DMP/Lab-8-ESP32-Part-2/blob/main/Lab_8_simple_client.ino

```
#include <WiFi.h>

/*
* Variabile globale pentru punctul de acces WiFi
*/
const char *SSID = "Network 1";
const char *PASS = "abcd1234";

/*
* Variable globale suplimentare
*/
String serverBaseURL = "";

void setup() {
 pinMode(LED_BUILTIN, OUTPUT);
 Serial.begin(9600);

 // placile ESP32 au o intarziere la UART dupa un reset dupa
 // ce a fost pus un cod nou pe placa
#if 1  // placa a fost programată recent
 delay(1000);
#else
 // https://www.arduino.cc/reference/en/language/functions/communication/serial/ifserial/
 while (!Serial) {
   ;  // asteaptă ca portul serial să se conecteze. Necesar pentru USB-ul nativ
 }
#endif

// Incepem prin conectarea la un punct de acces WiFi

 Serial.println();
 Serial.print("[WiFi] Connecting to ");
 Serial.println(SSID);
```

```
 WiFi.begin(SSID, PASS);
 // Implicit este setata sa se reconecteze automat
 // Pentru a dezactiva reconectarea, decomentati functia de mai jos
 //    WiFi.setAutoReconnect(false);

 // Vom incerca pentru 10 secunde (20x 500ms)
 const int tryDelay = 500;
 int numberOfTries = 20;

 // Se asteapta pentru un eveniment WiFi
 while (true) {

   switch (WiFi.status()) {
     case WL_NO_SSID_AVAIL:
       Serial.println("[WiFi] SSID not found");
       break;
     case WL_CONNECT_FAILED:
       Serial.print("[WiFi] Failed - WiFi not connected! Reason: ");
       return;
       break;
     case WL_CONNECTION_LOST:
       Serial.println("[WiFi] Connection was lost");
       break;
     case WL_SCAN_COMPLETED:
       Serial.println("[WiFi] Scan is completed");
       break;
     case WL_DISCONNECTED:
       Serial.println("[WiFi] WiFi is disconnected");
       break;
     case WL_CONNECTED:
       Serial.println("[WiFi] WiFi is connected!");
       Serial.print("[WiFi] IP address: ");
       Serial.println(WiFi.localIP());
       serverBaseURL = String("http://") + WiFi.gatewayIP().toString();
       Serial.println(String("[WiFi] Current Server IP: ") + WiFi.gatewayIP().toString());
       digitalWrite(LED_BUILTIN, HIGH);
       return;
     default:
       Serial.print("[WiFi] WiFi Status: ");
       Serial.println(WiFi.status());
       break;
   }
   delay(tryDelay);

   if (numberOfTries <= 0) {
     Serial.print("[WiFi] Failed to connect to WiFi!");
     // Se foloseste functia disconnect pentru a opri incercarile de conectare
     WiFi.disconnect();
     return;
   } else {
     numberOfTries--;
   }
 }
}

void loop() {}
```

Rulând codul de mai sus, observați mesajele din Serial Monitor, atât din partea plăcii client (stație) cât și din partea server (access point).

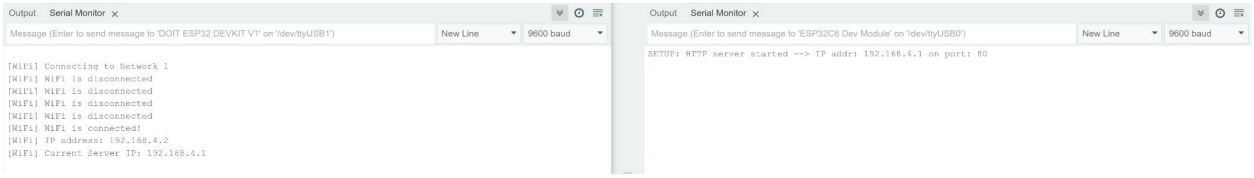

Figura 2. Mesaje din serial monitor, la client (stânga) și la server (dreapta).

La plăcuța client, LED-ul albastru se va aprinde dacă a fost realizată conexiunea cu serverul.

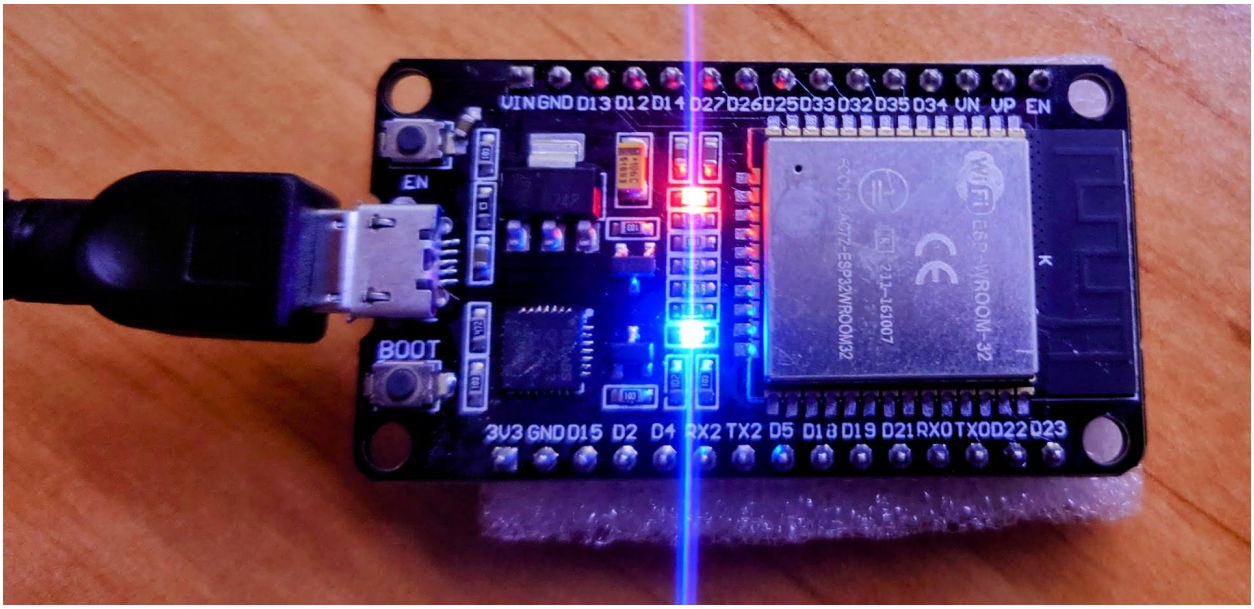

Figura 3. LED-ul albastru al plăcuței client se va aprinde dacă aceasta s-a conectat cu succes la un server.

3. Comunicare Wi-Fi între două plăci cu ESP32

Punctul de acces Wi-Fi oferă nivelul MAC (Medium Access Control) pentru comunicarea între două dispozitive. Pe acest nivel, nivelul rețea pune la dispoziție adresarea **IP (Internet Protocol)**. În domeniul rețelelor TCP/IP, comunicarea dintre procese este facilitată de o adresă IP, de un număr de port. Adresa IP servește drept identificator unic al unui dispozitiv (gazdă) din rețea, și permite rutarea pachetelor de date de la o sursă la o destinație. În contextul mai larg al Internetului, fiecare dispozitiv are o adresă IP unică.

Un **număr de port** desemnează un anume proces sau serviciu ce rulează pe un dispozitiv gazdă. Porturile permit ca mai multe procese (programe, servicii) de pe același dispozitiv să folosească în mod concurent resursele rețelei. Numerele de port pot fi din intervalul 0 … 65535. Serviciile standard, precum HTTP (web), au numere de port cunoscute (de exemplu, 80), iar alte aplicații pot folosi alte numere de port.

Când un client încearcă să comunice cu un server, folosește adresa IP a serverului împreună cu numărul de port asociat cu serviciul pe care dorește să-l acceseze pe server. Combinația dintre o adresă IP și un număr de port crează un punct de conexiune numit și **socket**.

Deși majoritatea aplicațiilor Wi-Fi pe ESP32 folosesc protocolul HTTP, acest protocol de nivel aplicație nu este necesar pentru schimbul elementar de date între plăci. Clasele WiFiClient și WiFiServer operează la nivelul transport (TCP), și pot fi utilizate pentru a trimite și a primi orice fel de date.

În continuare veți găsi exemple de cod pentru transmiterea de mesaje între două plăci ESP32, una fiind server și cealaltă client.

Codul pentru server

Codul poate fi descărcat și de la adresa:
https://github.com/UTCN-AC-CS-DMP/Lab-8-ESP32-Part-2/blob/main/Lab_8_tcp_server.ino

```
#include <WiFi.h>
#include <WiFiAP.h>
#include <WiFiClient.h>

// Mesaje predefinite
const String SETUP_INIT = "SETUP: Initializing ESP32 dev board";
const String SETUP_ERROR = "!!ERROR!! SETUP: Unable to start SoftAP mode";
const String INFO_NEW_CLIENT = "New client connected";
const String INFO_DISCONNECT_CLIENT = "Client disconnected";

// Constante pentru configurarea WiFI
// SSID - Numele retelei
// Schimbati acest nume in ceva unic inainte de a programa placa

const char *SSID = "DMP Server";
// Parola pentru retea
// Implicit ESP32 foloseste WPA / WPA2-Personal, deci parola trebuie sa aiba
// intre 8 si 63 caractere ASCII
const char *PASS = "123456789";
// Acest port (80) este utilizat implicit pentru serverele HTTP
// Conform RFC1340
// Puteti schimba acest numar, dar asigurati-va ca si clientul stie asta!
const int port = 80;

// Initializare server pe placa ESP32
WiFiServer server(port);

void setup() {
 Serial.begin(9600);
  // placile ESP32 au o intarziere la UART dupa un reset dupa
 // ce a fost pus un cod nou pe placa
#if 1  // placa a fost programată recent
 delay(1000);
#else
 while (!Serial) {
   ;  // asteaptă ca portul serial să se conecteze. Necesar pentru USB-ul nativ
 }
```

```
#endif

if (!WiFi.softAP(SSID, PASS)) {
   Serial.println(SETUP_ERROR);
   // Blocheaza in ciclu infinit daca nu se poate porni punctul de acces
   while (1)
     ;
 }

 // Citeste adresa IP a serverului pentru mesaj de informare
 const IPAddress accessPointIP = WiFi.softAPIP();
 const String serverInfoMessage = "Server started " + accessPointIP.toString()
                                  + " on port " + port;

 // Pornire server
 server.begin();
 Serial.println(serverInfoMessage);
}

void loop() {
 WiFiClient client = server.available();

 if (client) {  // Daca un client este conectat
   Serial.println(INFO_NEW_CLIENT);
   String currentLine = "";
   while (client.connected()) {  // Cât timp clientul e conectat
     if (client.available()) {  // Daca exista date trimise de client in buffer
       const char c = client.read(); // sunt citite
       currentLine += c;             // si adaugate la un string
       if (c == '\n') { // se citesc caractere pana cand se primeste NEWLINE
         // Nu folosim println deoarece mesajul are deja new line la final
         Serial.print(currentLine);
         // se trimite un mesaj la client, ce include si mesajul primit
         client.print("Hello client, I received the following message: " + currentLine);
         currentLine = "";  // se sterge stringul de receptie
         client.stop();  // deconectează clientul
         Serial.println(INFO_DISCONNECT_CLIENT);
         Serial.println();
       }
     }
   }
 }
}
```

Cod client

Codul poate fi descărcat și de la adresa:
https://github.com/UTCN-AC-CS-DMP/Lab-8-ESP32-Part-2/blob/main/Lab_8_tcp_client.ino

```
#include <WiFi.h>

const char* ssid = "DMP Server";      // Change this to your WiFi SSID
const char* password = "123456789";  // Change this to your WiFi password

const char* host = "192.168.4.1";  // Adresa IP a serverului
const int port = 80;               // Portul implicit. Se poate schimba valoarea, dar
```

```
                                    // trebuie sa fie cunoscuta si de client si de server
void setup() {
 Serial.begin(9600);
 // placile ESP32 au o intarziere la UART dupa un reset dupa
 // ce a fost pus un cod nou pe placa
#if 1  // placa a fost programată recent
 delay(1000);
#else
 while (!Serial) {
   ;  // asteaptă ca portul serial să se conecteze. Necesar pentru USB-ul nativ
 }
#endif

 // Vom incepe prin conectarea la o retea WiFi
 Serial.println("*******************************************************");
 Serial.print("Connecting to ");
 Serial.println(ssid);

 WiFi.begin(ssid, password);

 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }

 Serial.println("");
 Serial.println("WiFi connected");
 Serial.print("IP address: ");
 Serial.println(WiFi.localIP());
}

void loop() {
 WiFiClient client;  // TCP client

 // Se incearca conectarea la un server cu adresa "host" si portul "port"
 if (!client.connect(host, port)) {
   Serial.println("Cannot connect");
   delay(2000);
   return;
 }

 String message = "Hello Server!\n";

 client.print(message);  // Trimite mesaj la server

 // se asteapta raspunsul
 unsigned long timeout = millis();
 while (client.available() == 0) {
   if (millis() - timeout > 5000) {
     Serial.println(">>> Client Timeout !");
     client.stop();
     return;
   }
 }

 String line = "";
 while (client.available()) {
   // se citeste raspunsul serverului, caracter cu caracter
   char c = client.read();
   line = line + c;

   if (c == '\n')
```

```
    Serial.print(line);  // final de linie, se tipareste raspunsul in Serial
  }

  delay(2000);  // se asteapta 2 secunde si apoi se repeta pasii
}
```

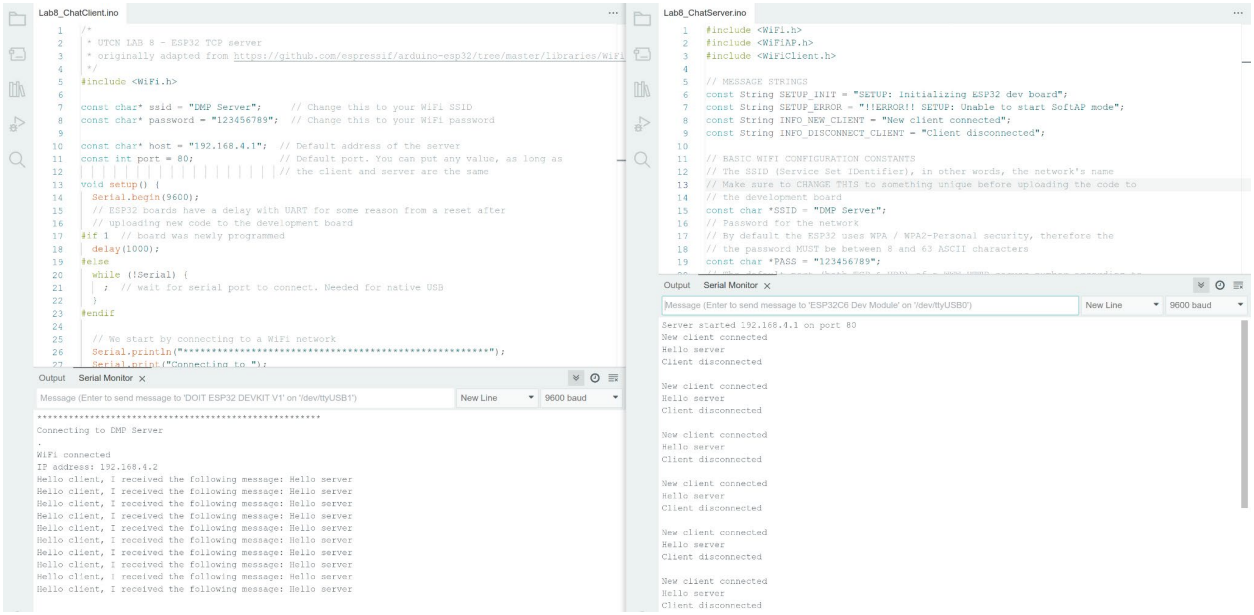

Figura 4. Mesajele trimise între client și server

4. Lucru individual

1. Rulați exemplele, analizați codul și explicațiile.
2. Creați un mecanism pe server (AP) care va permite comutarea LED-ului de pe placa ESP32 între starea aprins/stins, pe baza unui caracter primit de la client. Pe placa client (STA), citiți starea butonului BOOT (pinul digital D0 folosit ca intrare) și folosiți această stare pentru a trimite comanda pentru schimbarea stării LED-ului serverului.
3. Implementați un program chat între două ESP32. Fiecare placă este conectată la un PC, iar când utilizatorul va tasta un mesaj în Serial Monitor, mesajul va apărea în Serial Monitor la calculatorul conectat la a doua placă ESP32. Chat-ul trebuie să fie bidirecțional.

Bibliografie

1. Descrierea microcontrollerului ESP32 https://www.espressif.com/en/products/socs/esp32

2. API-ul WiFi ESP32 https://docs.espressif.com/projects/arduino-esp32/en/latest/api/wifi.html

3. Adrese și porturi TCP/IP https://www.ibm.com/docs/en/cics-ts/5.4?topic=protocols-tcpip-internet-addresses-ports