Laboratorio ESP32: Client, Server e Access Point

Questo laboratorio si focalizza sulla programmazione degli ESP32 in modalità:

- **Access Point & Web Server:** L'ESP32 crea una rete Wi-Fi e ospita un server web.
- **Client:** L'ESP32 si connette a una rete Wi-Fi e svolge richieste HTTP verso un server esterno.

---

1. ESP32 in modalità Access Point & Web Server

Obiettivo

Creare un Access Point in cui l'ESP32 ospita un web server che risponde con una pagina HTML.

Codice di esempio

```
#include <WiFi.h>
#include <WebServer.h>

// Credenziali per l'Access Point
const char* ssid = "ESP32-AP";
const char* password = "12345678";

// Istanza del web server sulla porta 80
WebServer server(80);

// Gestione della richiesta sulla root ("/")
void handleRoot() {
  String html = "<!DOCTYPE html><html lang='it'><head><meta charset='UTF-8'>
<title>ESP32 AP</title></head>"
                "<body><h1>Benvenuto sull'ESP32 Access Point!</h1>"
                "<p>Accedi a /toggle per cambiare lo stato del LED.</p>"
                "</body></html>";
  server.send(200, "text/html", html);
}

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Avvio dell'Access Point
  WiFi.softAP(ssid, password);
```

```
  Serial.println("Access Point avviato");
  Serial.print("IP dell'AP: ");
  Serial.println(WiFi.softAPIP());

  // Configurazione delle rotte del server
  server.on("/", handleRoot);
  // Inizialmente, /toggle sarà implementato nell'esercizio 1.

  server.begin();
  Serial.println("Web server avviato");
}

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

---

2. ESP32 in modalità Client

Obiettivo

Far sì che l'ESP32 si connetta a una rete Wi-Fi esistente e invii una richiesta HTTP GET a un server (es. example.com).

Codice di esempio

```
#include <WiFi.h>
#include <HTTPClient.h>

// Credenziali della rete Wi-Fi a cui connettersi
const char* ssid     = "NomeRete";
const char* password = "PasswordRete";

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Connessione alla rete Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connessione in corso");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnesso!");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
```

```
  // Inizializzazione della richiesta HTTP
  HTTPClient http;
  http.begin("http://example.com"); // Sostituire con il server target
  int httpCode = http.GET();

  if (httpCode > 0) {
    String payload = http.getString();
    Serial.println("Risposta HTTP:");
    Serial.println(payload);
  } else {
    Serial.print("Errore nella richiesta: ");
    Serial.println(http.errorToString(httpCode));
  }

  http.end();
}

void loop() {
  // Nessuna operazione nel loop
}
```

Esercizi di Laboratorio

Esercizio 1: Estensione del Server Web per il Controllo di un LED

Consegna

Modifica il codice del server (sezione 1) per:

- Aggiungere una nuova rotta `/toggle` che inverte lo stato di un LED collegato al pin LED_BUILTIN (o un pin definito).
- Visualizzare nella risposta HTML lo stato attuale del LED (ON/OFF).

Soluzione

```
#include <WiFi.h>
#include <WebServer.h>

const char* ssid = "ESP32-AP";
const char* password = "12345678";

WebServer server(80);
```

```
const int ledPin = LED_BUILTIN;  // Utilizza il pin LED_BUILTIN o definisci un altro pin
bool ledState = false;

void handleRoot() {
  String html = "<!DOCTYPE html><html lang='it'><head><meta charset='UTF-8'><title>ESP32 AP</title></head>"
                "<body><h1>ESP32 Access Point</h1>"
                "<p>Stato LED: " + String(ledState ? "ON" : "OFF") + "</p>"
                "<p><a href='/toggle'>Toggle LED</a></p>"
                "</body></html>";
  server.send(200, "text/html", html);
}

void handleToggle() {
  ledState = !ledState;
  digitalWrite(ledPin, ledState ? HIGH : LOW);
  String html = "<!DOCTYPE html><html lang='it'><head><meta charset='UTF-8'><title>ESP32 AP</title></head>"
                "<body><h1>LED Toggled!</h1>"
                "<p>Nuovo stato LED: " + String(ledState ? "ON" : "OFF") + "</p>"
                "<p><a href='/'>Torna alla Home</a></p>"
                "</body></html>";
  server.send(200, "text/html", html);
}

void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);

  WiFi.softAP(ssid, password);
  Serial.println("Access Point avviato");
  Serial.print("IP dell'AP: ");
  Serial.println(WiFi.softAPIP());

  server.on("/", handleRoot);
  server.on("/toggle", handleToggle);
  server.begin();
  Serial.println("Web server avviato");
}

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

Esercizio 2: ESP32 Client - Richiesta a un Server che Restituisce Dati JSON

Consegna

Scrivi un codice per l'ESP32 in modalità Client che:

- Si connette a una rete Wi-Fi.
- Esegue una richiesta HTTP GET a un endpoint che restituisce dati in formato JSON (es. un'API di test come `https://jsonplaceholder.typicode.com/todos/1`).
- Stampa in Serial i campi principali del JSON (ad es. `userId`, `id`, `title`, `completed`).

Note: Puoi utilizzare la libreria ArduinoJson per il parsing del JSON.

Soluzione

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

const char* ssid = "NomeRete";
const char* password = "PasswordRete";

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Connessione alla rete Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connessione in corso");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnesso!");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());

  HTTPClient http;
  // Endpoint di test che restituisce un JSON
  http.begin("https://jsonplaceholder.typicode.com/todos/1");
  int httpCode = http.GET();

  if (httpCode > 0) {
    String payload = http.getString();
    Serial.println("Risposta HTTP:");
    Serial.println(payload);
```

```
    // Parsing del JSON
    const size_t capacity = 256;
    DynamicJsonDocument doc(capacity);
    DeserializationError error = deserializeJson(doc, payload);
    if (!error) {
      int userId = doc["userId"];
      int id = doc["id"];
      const char* title = doc["title"];
      bool completed = doc["completed"];

      Serial.println("Dati estratti:");
      Serial.print("userId: "); Serial.println(userId);
      Serial.print("id: "); Serial.println(id);
      Serial.print("title: "); Serial.println(title);
      Serial.print("completed: "); Serial.println(completed ? "true" :
"false");
    } else {
      Serial.print("Errore di parsing: ");
      Serial.println(error.c_str());
    }
  } else {
    Serial.print("Errore nella richiesta: ");
    Serial.println(http.errorToString(httpCode));
  }

  http.end();
}

void loop() {
  // Nulla da fare nel loop
}
```

Riferimenti Utili

- **ArduinoJson:** https://arduinojson.org/
- **ESP32 WiFi e WebServer Library:** Documentazione ESP32 Arduino

---