Department of Computer Science

Week-9-10
Basic Wi-Fi
Simple web server

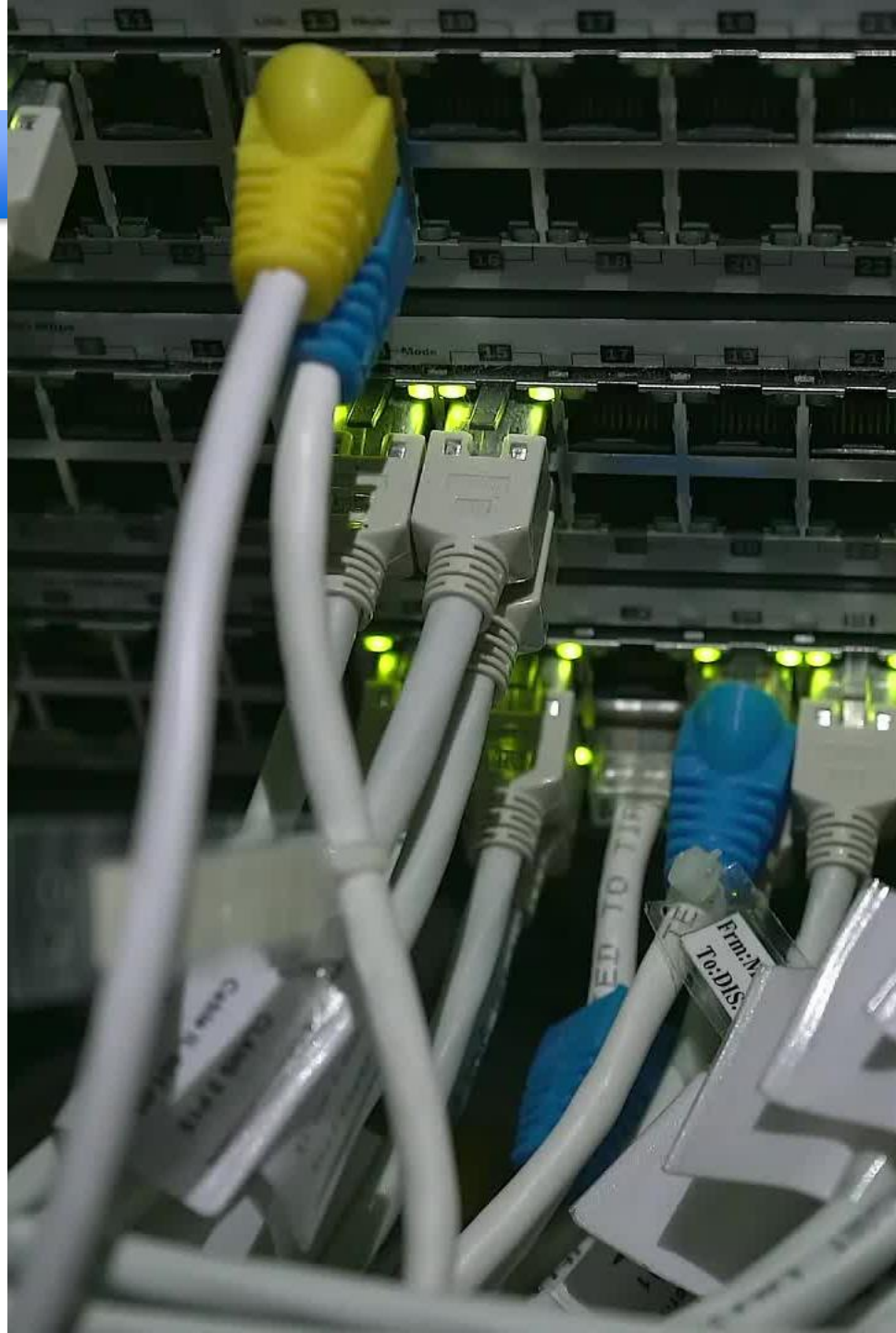
Embedded IoT Systems (CSE-3079)
Fall 2025

Instructor: Nasir Mahmood

ESP32 Wi-Fi

Overview

- Wi-Fi features in ESP32
- Station (STA), Access Point (AP), AP+STA
- Scanning networks & connecting to router
- Simple Webserver basics



Wi-Fi Basics

2.4 GHz IEEE 802.11 b/g/n

ESP32 supports multiple modes

Used for IoT projects, cloud access

Station Mode (WIFI_STA)

ESP32 connects to an existing router

Most common IoT use-case

Provides local IP for communication

Access Point Mode (WIFI_AP)

ESP32 creates its own Wi-Fi hotspot

Clients connect directly to ESP32

Useful for offline or local networks

AP + STA Mode (WIFI_AP_STA)

Both hotspot & router connection active

Allows local + cloud communication

Useful for configuration portals

WI-FI SCANNING

Wi-Fi Scanning

Use WiFi.scanNetworks()

Returns SSID, RSSI, Encryption type

Used for network availability detection

Scan Code

```
WiFi.mode(WIFI_STA)
```

```
WiFi.disconnect()
```

```
WiFi.scanNetworks()
```

```ino
#include <WiFi.h>

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);   //
  WiFi.disconnect();     //
  delay(1000);
}

void loop() {
  Serial.println("Scanning WiFi networks...");

  int count = WiFi.scanNetworks();  // Scan request

  if (count <= 0) {
    Serial.println("No networks found");
  } else {
    Serial.print("Found ");
    Serial.print(count);
    Serial.println(" networks:");
  }

  for (int i = 0; i < count; i++) {
    Serial.print(i + 1);
    Serial.print(": ");
    Serial.print(WiFi.SSID(i));
    Serial.print(" (");
    Serial.print(WiFi.RSSI(i));
    Serial.println(" dBm)");
  }

  Serial.println();
  delay(5000);   // Scan every 5 seconds
}
```

SSID Encoding

- SSID = Service Set Identifier (WiFi network name)
- SSID encoding means how an SSID is represented internally in bytes.
- ✔ Key Points
- ESP32 reads SSIDs as UTF-8 encoded strings
- WiFi routers may use:
  - ASCII
  - UTF-8
  - UTF-16 (rare)

Important Notes

- Hidden SSIDs appear as "" (empty string)
- Special characters (Ø, ä, ₹, etc.) may show incorrectly on older routers
- Some routers broadcast SSID in *hexadecimal form* for special purposes
- ESP32 supports:
  - emojis (UTF-8)
  - spaces
  - long SSIDs (up to 32 bytes)

RSSI Thresholds Interpretation

RSSI (dBm)	Signal Quality	Meaning
–30 to –50	Excellent	Very strong signal, close to router
–50 to –60	Good	Stable for IoT devices
–60 to –70	Fair	Works but may drop packets
–70 to –80	Weak	Slow or unstable connection
–80 to –90	Very Poor	Hard to connect, frequent disconnections
< –90	Unusable	Almost no link

- RSSI = Received Signal Strength Indicator
  It tells how strong the WiFi signal is (in dBm).
  Values are always negative — closer to 0 means stronger.

WiFi.encryptionType(i)

ESP32 Constant	Meaning
WIFI_AUTH_OPEN	No security
WIFI_AUTH_WEP	WEP
WIFI_AUTH_WPA_PSK	WPA
WIFI_AUTH_WPA2_PSK	WPA2
WIFI_AUTH_WPA_WPA2_PSK	WPA/WPA2 Mixed
WIFI_AUTH_WPA3_PSK	WPA3
WIFI_AUTH_WPA2_WPA3_PSK	WPA2/WPA3 Mixed

WI-FI CONNECTION

Wi-Fi Connection

Use WiFi.begin(ssid, password)

Auto-sets ESP32 to station mode

Wait until WL_CONNECTED

Department of Computer Science

WiFi.begin(ssid, password)

while(WiFi.status()!=WL_CONNECTED)

Connecti on Code

localIP() for IP address

Department of Computer Science

```cpp
#include <WiFi.h>

const char* ssid     = "YOUR_WIFI_NAME";
const char* password = "YOUR_WIFI_PASSWORD";

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

  Serial.println("Connecting to WiFi...");
  WiFi.begin(ssid, password);   // Start connection

  // Wait until connected
  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() {
  // Nothing here – stays connected
}
```

STATIC IP SETUP (BASIC VERSION)

```ino
#include <WiFi.h>

const char* ssid     = "YOUR_WIFI";
const char* password = "YOUR_PASS";

// Static IP configuration
IPAddress local_IP(192, 168, 1, 200);     // ESP32 IP
IPAddress gateway(192, 168, 1, 1);        // Router IP
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(8, 8, 8, 8);         // Optional
IPAddress secondaryDNS(8, 8, 4, 4);       // Optional

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

  // Set Static IP
  if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
    Serial.println("Static IP Failed!");
  }

  WiFi.begin(ssid, password);

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

  Serial.println("\nConnected!");
  Serial.print("ESP32 IP: ");
  Serial.println(WiFi.localIP());
}

void loop() {
}
```

Important Notes

- ✓ WiFi.config() must be **before** WiFi.begin().

✓ Make sure the chosen IP (e.g., 192.168.1.200) is **not used by another device**.
✓ Use the correct IP range for your router:

- PTCL / TP-Link: usually 192.168.10.x or 192.168.0.x
- Zong / Jazz 4G device: often 192.168.8.x
-

Static IP Advantages

- Webserver always opens at same address
- No need to search IP again
- Best for home automation
- Useful for MQTT dashboards, Node-RED, etc.

WEB SERVER SIMPLE

Simple Webserver

```
Use WiFiServer server(80)

server.begin()

client = server.available()
```

```
36  void loop() {
37    WiFiClient client = server.available();
38    if (!client) return;  // No client, exit
39
40    Serial.println("New Client connected");
41    String request = client.readStringUntil('\r');
42    Serial.println(request);
43
44    // ----- LED CONTROL -----
45    if (request.indexOf("/LED=ON") != -1) {
46      digitalWrite(LED_PIN, HIGH);
47    }
48    if (request.indexOf("/LED=OFF") != -1) {
49      digitalWrite(LED_PIN, LOW);
50    }
51
52    // ----- RESPONSE PAGE -----
53    String htmlPage =
54      "<!DOCTYPE html><html>"
55      "<h1>ESP32 LED Control</h1>"
56      "<p><a href=\"/LED=ON\"><button>LED ON</button></a>
57      "<p><a href=\"/LED=OFF\"><button>LED OFF</button></
58      "</html>";
59
60    client.println("HTTP/1.1 200 OK");
61    client.println("Content-Type: text/html");
62    client.println("Connection: close");
63    client.println();
64    client.println(htmlPage);
65
66    delay(1);
67    client.stop();
68    Serial.println("Client disconnected");
69  }
```

```
6   #include <WiFi.h>
7
8   const char* ssid = "YOUR_WIFI";
9   const char* password = "YOUR_PASS";
10
11  WiFiServer server(80);
12  const int LED_PIN = 2;    // Built-in LED
13
14  void setup() {
15    Serial.begin(115200);
16    pinMode(LED_PIN, OUTPUT);
17    digitalWrite(LED_PIN, LOW);  // LED off at s
18
19    // Connect WiFi
20    Serial.print("Connecting to ");
21    Serial.println(ssid);
22    WiFi.begin(ssid, password);
23
24    while (WiFi.status() != WL_CONNECTED) {
25      delay(500);
26      Serial.print(".");
27    }
28    Serial.println("\nWiFi connected!");
29
30    Serial.print("ESP32 IP Address: ");
31    Serial.println(WiFi.localIP());
32
33    server.begin();
34  }
```

ESP32 LED Control

LED ON

LED OFF

Summary

ESP32 supports STA, AP, AP+STA

Scanning useful for choosing networks

Easy Wi-Fi connection with WiFi.begin()

Webserver enables IoT dashboards

DHT11 WITH SIMPLE WEB SERVER

```ino
#include <WiFi.h>
#include "DHT.h"

// ---- WiFi settings ----
const char* ssid     = "YOUR_WIFI_NAME";
const char* password = "YOUR_WIFI_PASSWORD";

// ---- DHT settings ----
#define DHTPIN 4        // DHT data pin to GPIO 4
#define DHTTYPE DHT11   // DHT11 or DHT22
DHT dht(DHTPIN, DHTTYPE);

// ---- Web server on port 80 ----
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  dht.begin();

  // Connect to 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("\nWiFi connected!");
  Serial.print("ESP32 IP: ");
  Serial.println(WiFi.localIP());

  server.begin();
}

void loop() {
  // Wait for a client (browser)
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  Serial.println("New Client connected");
  // wait until client sends data
  while (!client.available()) {
    delay(1);
  }

  // Read the first line of HTTP request
  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();

  // ---- Read DHT sensor values ----
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // Celsius

  // Check if any reads failed
  bool error = isnan(h) || isnan(t);

  // ---- Build simple HTML page ----
  String html = "<!DOCTYPE html><html><head>"
                "<meta charset='UTF-8'>"
                "<title>ESP32 DHT11 Webserver</title>"
                "<meta http-equiv='refresh' content='5' />"
                "</head><body>"
                "<h1>ESP32 DHT11 Sensor</h1>";

  if (error) {
    html += "<p><b>Error reading DHT11 sensor!</b></p>";
  } else {
    html += "<p>Temperature: <b>" + String(t, 1) + " &deg;C</b></p>";
    html += "<p>Humidity: <b>" + String(h, 1) + " %</b></p>";
  }

  html += "<p>Page refreshes every 5 seconds.</p>";
  html += "</body></html>";

  // ---- Send HTTP response ----
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close");
  client.println();
  client.print(html);

  delay(1);
  Serial.println("Client disconnected");
}
```

What this sketch does

1.Connects ESP32 to your WiFi using SSID & password.

2.Starts a **web server on port 80**.

3.On each browser request:

1. Reads **temperature & humidity** from DHT11.
2. Generates a simple **HTML page** with the values.
3. Sends it back to the browser.

4.Page **auto-refreshes every 5 seconds** (`<meta http-equiv='refresh' content='5'>`).

How to view it

1.Upload code → open Serial Monitor (115200).

2.Note the printed IP, e.g. `ESP32 IP: 192.168.1.50`

3.On your laptop/mobile (same WiFi), open browser and go to:

`4.http://192.168.1.50`

You'll see a simple page:

•ESP32 DHT11 Sensor

•Temperature in °C

•Humidity in %