24EC504 – Unit III – Part 3: Wi-Fi, BLE, Cloud, Edge, IoT Architecture and Web Servers

1. Wi-Fi and Bluetooth Low Energy (BLE)

1.1 Wi-Fi

Wi-Fi is a family of wireless standards (IEEE 802.11) giving high-bandwidth connectivity, typically in the 2.4 GHz and 5 GHz ISM bands.

- **Topology:** star, with devices connected to a central access point (AP).
- **Range:** roughly 30–100 m indoors, depending on environment and AP power.
- **Data rate:** several Mbps to over 1 Gbps depending on the standard (a/b/g/n/ac/ax).
- **Power:** relatively high, which limits use in battery/coin-cell sensors.
- **Typical embedded use:** video streaming, smart cameras, mains-powered or larger-battery devices, and gateways bridging local sensor networks to the internet (for example ESP32/ESP8266).

1.2 Bluetooth Low Energy (BLE)

BLE is a wireless personal-area technology (Bluetooth 4.0+) designed for low power, ideal for small battery-operated IoT devices.

- **Topology:** central–peripheral (master–slave) star; newer versions also support mesh.
- **Range:** roughly 10–100 m depending on power class and environment.
- **Data rate:** low, but enough for periodic sensor readings and control commands.
- **Power:** very low; months to years on a coin cell.
- **GATT (Generic Attribute Profile):** organizes data into **Services** (logical groups) and **Characteristics** (individual values, for example heart rate).
- **Device states:** Advertising (broadcasting presence) and Connected (exchanging data with a central device).
- **Typical use:** wearables, fitness trackers, medical sensors, beacons, smart locks.

1.3 Wi-Fi vs BLE

Parameter	Wi-Fi	BLE
Power consumption	High	Very low
Data rate	High (Mbps–Gbps)	Low (kbps range)
Range	~30–100 m	~10–100 m
Topology	Star (via access point)	Star / mesh (central–peripheral)
Best suited to	High-bandwidth, mains or large-	Small battery-powered sensors and

Parameter	Wi-Fi	BLE
	battery devices	wearables

2. Cloud Connectivity

Cloud connectivity links embedded/IoT devices to remote cloud platforms so data can be stored, processed, analysed and visualised, and devices monitored or controlled remotely.

2.1 Connectivity Protocols

Protocol	Notes
MQTT	Lightweight publish/subscribe; most common for IoT telemetry
HTTP/HTTPS	Request/response; REST APIs and one-off uploads
CoAP	Lightweight, UDP-based protocol for constrained devices and networks
WebSockets	Persistent, full-duplex connection; useful for live dashboards

2.2 Popular IoT Cloud Platforms

AWS IoT Core, Microsoft Azure IoT Hub, Google Cloud IoT, ThingSpeak (popular for academic prototypes, MATLAB-integrated) and Blynk (hobbyist mobile dashboards).

2.3 Key Cloud Services

- **Device management and provisioning:** registering devices, assigning identities and certificates.
- **Data ingestion and storage:** collecting telemetry into time-series or general databases.
- **Rules engine / analytics:** processing streams to trigger alerts or actions.
- **Dashboards and visualisation:** web/mobile interfaces for live monitoring.
- **OTA (over-the-air) updates:** remote firmware updates.
- **Security services:** device identity, certificates and encrypted channels.

2.4 Typical Architecture

Sensor/Device → (local processing) → Gateway → Internet → Cloud broker / ingestion service → Data processing and storage → Application / dashboard / mobile app.

Benefits: scalability, remote monitoring and control, powerful analytics, less local processing and storage, easier integration with other services.

3. Edge Computing Concepts

Edge computing processes data close to where it is generated, on or near the device, instead of sending all raw data to a distant cloud.

3.1 Why Edge Computing?

- **Reduced latency:** real-time decisions without a cloud round trip.
- **Reduced bandwidth and cost:** only meaningful processed data goes to the cloud.
- **Improved privacy and security:** sensitive raw data can stay local.
- **Offline operation:** keeps working when connectivity is lost.
- **Faster decision-making:** critical for safety systems.

3.2 Edge vs Fog vs Cloud

Aspect	Edge	Fog	Cloud
Processing location	On the device itself	Local network / gateway	Remote data centre
Latency	Lowest	Low	Higher
Compute power	Limited	Moderate	Very high
Typical role	Immediate local decisions	Aggregation across nearby devices	Large-scale storage and analytics

3.3 Applications and Hardware

- **Applications:** autonomous vehicles, industrial IoT and predictive maintenance, smart security cameras with on-device AI, healthcare wearables needing instant alerts.
- **Edge hardware:** microcontrollers, edge gateways, single-board computers such as Raspberry Pi or NVIDIA Jetson.
- **TinyML:** lightweight ML inference directly on microcontrollers (keyword spotting, anomaly detection), entirely at the edge.

4. IoT Architecture and Security

4.1 Layered IoT Architecture

1. **Perception / sensing layer:** sensors and actuators that sense and/or act on the environment.
2. **Network / transport layer:** connectivity between devices and higher layers using Wi-Fi, BLE, Zigbee, LoRa/LPWAN or cellular.
3. **Middleware / processing layer:** aggregates, filters and processes data at the edge, in a fog node or in the cloud.
4. **Application layer:** delivers information through dashboards, mobile apps or automated services.

4.2 Security Challenges

- Weak or default authentication credentials.
- Unencrypted communication channels.
- Firmware vulnerabilities and missing security patches.

- Physical tampering, since devices are often in unsecured locations.
- Large-scale botnets exploiting insecure devices (the Mirai botnet used default IoT credentials for DDoS attacks).
- Limited CPU/RAM restricts strong cryptography on very small devices.
- No single universally adopted IoT security standard.

4.3 Security Measures

- Device authentication with unique identities, certificates and secure boot.
- Encryption in transit (TLS/DTLS) and at rest where possible.
- Secure, signed OTA firmware updates.
- Network segmentation, isolating IoT devices from critical infrastructure.
- Strict access control and least privilege.
- Regular audits, patching, intrusion detection and monitoring.

5. Embedded Web Servers

An embedded web server is a lightweight HTTP server running inside an embedded device (usually a networked microcontroller) so it can serve web pages, APIs or configuration interfaces to a browser without an external server.

5.1 Characteristics

- Runs within the limited RAM/flash of a microcontroller.
- Serves static content (HTML/CSS/JS) and dynamic content built from sensor readings.
- Commonly built on Wi-Fi/Ethernet modules, for example the WebServer / ESPAsyncWebServer libraries on ESP32/ESP8266, or lwIP-based servers on other MCUs.

5.2 Typical Uses

- Local device configuration pages (Wi-Fi credentials, parameters).
- Real-time monitoring dashboards on the local network.
- REST APIs (GET/POST) to query or control the device.
- Simple web-based OTA firmware update pages.

5.3 Example Workflow (ESP32-style)

1. Connect the microcontroller to a Wi-Fi network.
2. Initialise the web server object and start listening (`server.begin()`).
3. Register route handlers, for example `server.on("/status", handler)`.
4. Call `server.handleClient()` repeatedly in the main loop to process requests.
5. The handler builds an HTML/JSON response from current sensor data and sends it to the browser or client app.

5.4 Advantages and Limitations

Advantages	Limitations
No external server hardware needed	Only a small number of simultaneous connections
Direct access from any browser on the same network	Limited processing power and memory restrict complex web apps
Simple setup for local monitoring and control	Plain HTTP (no HTTPS) on low-cost boards can be a security concern on untrusted networks

Quick Revision

- **Wi-Fi vs BLE:** Wi-Fi = high bandwidth, high power; BLE = low bandwidth, very low power, ideal for wearables and sensors.
- **Cloud path:** device → gateway → cloud broker → storage/analytics → dashboard, using MQTT, HTTP or CoAP.
- **Edge computing:** local processing for lower latency, lower bandwidth and offline capability; contrast with fog and cloud.
- **IoT model:** four layers (Perception → Network → Middleware → Application); security rests on authentication, encryption, secure OTA and access control.