Internet Of Things

Title: WebSocket | ESP32

---

1. Why Do We Need WebSocket?

❓ Problem with HTTP

HTTP works like this:

- Client (e.g., browser) sends a request
- Server (e.g., ESP32) sends a response
- Connection is closed

Drawback:

The server can never send data first (no real-time updates).

Imagine you're monitoring a temperature sensor.
With HTTP, you'd have to keep refreshing to check changes — wasteful!

Solution: WebSocket

WebSocket is a full-duplex communication protocol.

- Once connected, both client and server can send data anytime
- Connection stays open
- Ideal for real-time control and monitoring

---

2. WebSocket vs HTTP: Core Differences

Feature	HTTP	WebSocket
Type	Half-Duplex (Request/Response)	Full-Duplex (Real-Time)
Persistent	❌ Closes after response	✅ Stays open
Who can send first	Client only	Both Client and Server
Ideal for	Page requests, REST APIs	Live control, Chat, IoT
Protocol Prefix	http:// or https://	ws:// or wss://

---

3. How WebSocket Works – Step by Step

Step 1: Client Sends WebSocket Handshake

- Client sends a special HTTP upgrade request:

```
GET / HTTP/1.1
Host: esp32.local
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: XYZ==
```

Step 2: Server Accepts and Upgrades

- Server replies:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: abc123==
```

From here, WebSocket connection is open!

Step 3: Real-Time Communication

- Client ➡ Server: "LED ON"
- Server ⬅ Client: "LED is now ON"

✅ Both can push data any time — no request needed!

---

4. WebSocket in IoT (ESP32)

Use Cases:

- LED control
- Sensor dashboard updates
- Chat apps
- Real-time logs from ESP32
- Motor/PWM status updates

---

5. WebSocket Message Flow in IoT

```
Browser     <-- ws -->     ESP32
  ↑                    ↑
 Button pressed        Sensor update
  ↓                    ↓
 "Turn ON LED"        "Temp: 32.1°C"
```

Both sides actively exchange messages without delay.

---

6. WebSocket Libraries for ESP32

You can use:

Library	Async?	Easy?	Note
WebSocketsServer.h	❌	✅ Beginner-friendly	Synchronous
AsyncWebSocket	✅	❌ More complex	Non-blocking

You've chosen synchronous version = simpler for learning ✅

---

7. Protocol Prefixes

Protocol	Used When	Example
`ws://`	Unencrypted	`ws://192.168.1.42:81`
`wss://`	Secure (TLS/SSL)	`wss://iot.myserver.com`

---

8. Advantages of WebSocket in ESP32

✅ Real-time
✅ Two-way communication
✅ No constant polling
✅ Lightweight and fast

---

9. WebSocket Ports

By default:

- HTTP uses `80`
- WebSocket often uses `81` (or custom)
- HTTPS uses `443`
- Secure WebSocket uses `443` (with `wss://`)

You can configure ESP32 WebSocket to run on port 81, like:

```
WebSocketsServer webSocket = WebSocketsServer(81);
```

Behind the Scenes of Code:

When you write:

```
webSocket.begin();
webSocket.onEvent(webSocketEvent);
```

You're telling ESP32 to:

- Accept WebSocket connections
- Wait for incoming messages
- Trigger a callback (like turn on LED) when a message is received

---

Summary

WebSocket is	Meaning
Full-Duplex	Send & receive data both ways in real-time
Persistent	One-time handshake, then keeps alive
Low-latency	Much faster than HTTP polling
Lightweight	Excellent for microcontrollers like ESP32
