Internet Of Things

Bidirectional WebSocket Communication Between Two ESP32s

1. What is WebSocket?

WebSocket is a **bi-directional, full-duplex communication protocol** over a single TCP connection. It allows two devices to **send and receive data in real-time** without repeatedly opening new HTTP connections.

- **Key features:**
  - Low latency
  - Persistent connection
  - Bi-directional communication
- **Use case in ESP32:**
  - Even though WebSocket was designed for web browsers, it can be used between **two ESP32 devices** directly.

2. Why WebSocket?

- Real-time data exchange without delays.
- Avoids HTTP overhead of multiple requests.
- Works on any TCP network, including WiFi, without the need for an actual browser.

3. Is WebSocket an IoT Protocol?

- **No**, WebSocket is **not an IoT protocol** by design.
- IoT protocols are usually MQTT, CoAP, or LwM2M.
- WebSocket is a **web communication protocol** (works for browsers) but can also be adapted for IoT devices like ESP32.

4. Libraries Needed

1. **WiFi.h** → Connect ESP32 to WiFi.
2. **WebSocketsServer.h** → Create a WebSocket server on ESP32.
3. **WebSocketsClient.h** → Connect ESP32 to another WebSocket server.

Installation:

- Open Arduino IDE → Tools → Manage Libraries → Search and install WebSockets by Markus Sattler.

---

5. Key Library Functions

Server Side (WebSocketsServer):

- begin() → Starts the server.
- onEvent() → Sets the callback function to handle events (connect, disconnect, message).
- loop() → Must be called in loop() to process events.
- broadcastTXT() → Sends a text message to all connected clients.

Client Side (WebSocketsClient):

- begin(host, port, url) → Connects to server.
- onEvent() → Handles incoming messages.
- loop() → Must be called in loop().
- sendTXT() → Sends a text message to the server.

---

6. ESP32-to-ESP32 Communication Setup

- **ESP32-A:** WebSocket server on port 81 → sends **odd numbers**
- **ESP32-B:** WebSocket server on port 82 → sends **even numbers**
- Each ESP32 acts as a **server and client simultaneously**.

---

7. Step-by-Step Implementation

Step 1: Connect Both ESP32s to WiFi

```
WiFi.begin(ssid, password);
while(WiFi.status() != WL_CONNECTED) {
  delay(500);
}
```

Step 2: Setup WebSocket Server

```
webSocketServer.begin();
webSocketServer.onEvent(serverEvent);
```

Step 3: Setup WebSocket Client

```
webSocketClient.begin(peer_ip, peer_port, "/");
webSocketClient.onEvent(clientEvent);
```

Step 4: Loop Handling

```
webSocketServer.loop();
webSocketClient.loop();
```

Step 5: Send Data Every 2 Seconds

- Use millis() to send data periodically.
- Odd numbers from ESP32-A, Even numbers from ESP32-B.

```
if(millis() - lastTime > 2000) {
 lastTime = millis();
 webSocketServer.broadcastTXT(String(counter));
 webSocketClient.sendTXT(String(counter));
 counter += 2; // odd or even
}
```

---

8. Complete Code

ESP32-A (Odd Numbers)

```
#include <WiFi.h>
#include <WebSocketsServer.h>
#include <WebSocketsClient.h>

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

WebSocketsServer webSocketServer(81);
WebSocketsClient webSocketClient;

const char* esp32B_ip = "ESP32_B_IP"; // Replace with ESP32-B IP
const uint16_t esp32B_port = 82;

int counter = 1;
unsigned long lastTime = 0;

void serverEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {
 if(type == WStype_TEXT) {
  Serial.printf("Received from ESP32-B: %s\n", payload);
 }
}

void clientEvent(WStype_t type, uint8_t * payload, size_t length) {
 if(type == WStype_TEXT) {
  Serial.printf("Message from ESP32-B server: %s\n", payload);
 }
}

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

 WiFi.begin(ssid, password);
 while(WiFi.status() != WL_CONNECTED) delay(500);

 webSocketServer.begin();
 webSocketServer.onEvent(serverEvent);

 webSocketClient.begin(esp32B_ip, esp32B_port, "/");
 webSocketClient.onEvent(clientEvent);
}

void loop() {
 webSocketServer.loop();
 webSocketClient.loop();

 if(millis() - lastTime > 2000) {
  lastTime = millis();
  webSocketServer.broadcastTXT(String(counter));
  webSocketClient.sendTXT(String(counter));
  counter += 2; // Send odd numbers
 }
}
```

ESP32-B (Even Numbers)

```
#include <WiFi.h>
#include <WebSocketsServer.h>
#include <WebSocketsClient.h>

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

WebSocketsServer webSocketServer(82);
WebSocketsClient webSocketClient;

const char* esp32A_ip = "ESP32_A_IP"; // Replace with ESP32-A IP
const uint16_t esp32A_port = 81;

int counter = 2;
unsigned long lastTime = 0;

void serverEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {
 if(type == WStype_TEXT) {
  Serial.printf("Received from ESP32-A: %s\n", payload);
 }
}

void clientEvent(WStype_t type, uint8_t * payload, size_t length) {
 if(type == WStype_TEXT) {
  Serial.printf("Message from ESP32-A server: %s\n", payload);
 }
}
```

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

  WiFi.begin(ssid, password);
  while(WiFi.status() != WL_CONNECTED) delay(500);

  webSocketServer.begin();
  webSocketServer.onEvent(serverEvent);

  webSocketClient.begin(esp32A_ip, esp32A_port, "/");
  webSocketClient.onEvent(clientEvent);
}

void loop() {
  webSocketServer.loop();
  webSocketClient.loop();

  if(millis() - lastTime > 2000) {
    lastTime = millis();
    webSocketServer.broadcastTXT(String(counter));
    webSocketClient.sendTXT(String(counter));
    counter += 2; // Send even numbers
  }
}
```

---

9. Testing

1. Upload ESP32-A code and note its IP from Serial Monitor.
2. Upload ESP32-B code and note its IP from Serial Monitor.
3. Replace `ESP32_A_IP` and `ESP32_B_IP` in the code with actual IPs.
4. Open Serial Monitor on both devices → you will see **odd and even numbers exchanged** every 2 seconds.

This setup works **without any browser** and demonstrates **ESP32-to-ESP32 real-time communication using WebSockets**.