

Host-Device Communication — Wi-Fi Wireless Control via WebSocket (Linux / macOS)

This guide explains how to use **WebSocket** to establish real-time, two-way communication between your computer and the robot driver board on **Linux or macOS**.

Unlike HTTP, WebSocket enables **full-duplex communication** with **millisecond-level latency**, making it ideal for real-time robot control, sensor monitoring, or telemetry visualization.

 GitHub Repository

```
https://github.com/EffectsMachine/robot_driver_with_esp32s3_lite
```

1. How It Works

The robot driver board includes a lightweight **WebSocket server** that runs on port **80**, sharing the same port as its HTTP interface.

The WebSocket endpoint path is fixed at `/ws`.

When your PC connects to `ws://<ESP32_IP>:80/ws`, a persistent data channel is created:

1. **Handshake:** The PC initiates a TCP connection and upgrades to WebSocket.
2. **Command Transmission:** The PC can continuously send JSON-based control commands.
3. **Real-Time Feedback:** The board actively pushes back telemetry, such as servo status, sensor readings, or warning messages.

Example message:

```
{"T":202,"line":1,"text":"webSocket msg","update":1}
```

This message updates line 1 of the OLED with the text “webSocket msg”.

2. Setup Environment

2.1 Install Python

Make sure Python **3.10 or newer** is installed.

Check your version:

```
python3 --version
```

If not installed:

- **Linux:** Use your package manager (`apt` , `dnf` , etc.)
- **macOS:** Install via [Python.org](https://python.org) or simply use **Homebrew**:

```
brew install python
```

2.2 Clone the Project

Open a terminal and clone the example repository:

```
git clone https://github.com/EffectsMachine/robot_driver_with_esp32s3_lite.git
```

```
cd robot_driver_with_esp32s3_lite/Example\ for\ Robot\ Driver\ Lite/python_example/webSocket
```

3. Running the Example

3.1 Create and Activate a Virtual Environment

To keep dependencies clean, create a Python virtual environment:

```
python3 -m venv venv  
source venv/bin/activate
```

When you see `(venv)` in your terminal prompt, the environment is active.

Install the required packages:

```
pip install -r requirements.txt
```

3.2 Configure Connection Settings

Open the example file:

```
webSocket_example.py
```

Find this line:

```
ESP32_IP = "192.168.0.104"
```

Change it according to your connection type:

- **Direct Wi-Fi Hotspot:** `ESP32_IP = "192.168.4.1"`
- **Local Network:** Use the IP address displayed on the robot's OLED screen.

3.3 Run the Example Script

Make sure the driver board is powered on and connected to Wi-Fi. Then run:

```
python3 webSocket_example.py
```

If successful, you should see:

```
connected to ESP32 WebSocket  
send: {"T":202,"line":1,"text":"webSocket msg","update":1}
```

To stop the program:

Ctrl + C

4. Troubleshooting

Symptom	Possible Cause	Solution
<code>connection refused</code>	Wrong IP address or WebSocket service not running	Check the OLED display for the correct IP and ensure port 80 is open
No data received	Device not sending feedback	Verify the robot firmware has feedback enabled
<code>ModuleNotFoundError: websocket</code>	Dependency not installed	Run <code>pip install websocket-client</code>

5. Code Overview

The example file `websocket_example.py` demonstrates how to establish a WebSocket connection, send JSON commands, and receive feedback.

Core Settings

```
ESP32_IP = "192.168.0.104"
PORT = 80
PATH = "/ws"
WS_URL = f"ws://{ESP32_IP}:{PORT}{PATH}"
```

Key Functions

Function	Description
<code>on_open(ws)</code>	Triggered when connection is established; sends an initial JSON command
<code>on_message(ws, message)</code>	Handles incoming data from the board

Function	Description
<code>on_error(ws, error)</code>	Prints error details for debugging
<code>on_close(ws, code, msg)</code>	Called when the connection closes or drops

These callbacks can easily be extended for custom behaviors such as auto-reconnect or continuous telemetry logging.

6. Full Example

```
import json
import websocket

ESP32_IP = "192.168.0.104"
PORT = 80
PATH = "/ws"
WS_URL = f"ws://{ESP32_IP}:{PORT}{PATH}"

def on_message(ws, message):
    print(f"recv: {message}")

def on_error(ws, error):
    print(f"WebSocket error: {error}")

def on_close(ws, close_status_code, close_msg):
    print("connection closed")

def on_open(ws):
    print("connected to ESP32 WebSocket")
    command = {
        "T": 202,
        "line": 1,
        "text": "websocket msg",
        "update": 1
    }
    ws.send(json.dumps(command))
```

```
print(f"send: {command}")

if __name__ == "__main__":
    websocket.enableTrace(True)
    ws = websocket.WebSocketApp(
        WS_URL,
        on_message=on_message,
        on_error=on_error,
        on_close=on_close,
        on_open=on_open
    )
    ws.run_forever()
```

7. Summary

- WebSocket enables **real-time, bidirectional, low-latency** communication.
- Commands and feedback are exchanged in simple **JSON format**.
- Compatible with **Windows, Linux, and macOS** environments.
- The code is fully open source and easy to extend for advanced applications.