

Host-Device Interaction via Wi-Fi (HTTP Control on Windows)

This tutorial walks you through how to control your robot driver board over Wi-Fi using **HTTP requests**.

By connecting a Windows PC (as the *host controller*) with the robot driver board (as the *robot controller*), you can easily build an upper-lower control architecture for scripting automation and advanced robot control.

1. Getting the Project

GitHub repository:

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

Clone or download the repository to your local Windows environment before starting.

2. Why Use HTTP for Robot Control?

HTTP (HyperText Transfer Protocol) is a **request-response, stateless** communication protocol — perfect for structured command delivery and data exchange between your PC and the robot.

Advantages	Description
Low Development Cost	No need to design custom communication protocols. Every major programming language (Python, C#, Java, etc.) provides mature HTTP libraries such as Python's <code>requests</code> .
Cross-Platform Compatibility	Works seamlessly across devices — PC, mobile app, or cloud server — via LAN or the Internet.

Advantages	Description
Easy to Extend	You can enhance functionality with HTTPS for encryption, or build RESTful APIs (POST for control, GET for data retrieval).

Limitations	Description
Lower Real-Time Performance	Each HTTP request includes headers and connection overhead. Typical latency ranges from tens to hundreds of milliseconds — not suitable for microsecond-level control loops.
No Native Push Mechanism	The robot only responds to requests. To receive async alerts (e.g., collision warnings), polling or WebSocket integration is required.
Unencrypted by Default	Plain HTTP is insecure on public networks. Use HTTPS or authentication tokens for safety.

In short: HTTP offers a **universal, flexible, and easy-to-use** way to control and monitor your robot — ideal for most automation and research applications.

3. How It Works

The driver board includes a built-in **lightweight HTTP server** running on port **80**. Once powered on, it connects (or creates) a Wi-Fi network and exposes an HTTP endpoint:

```
http://<DEVICE_IP>:80
```

Any network-enabled device (PC, smartphone, Raspberry Pi, etc.) can send HTTP requests to this address to interact with the board.

Commands are formatted as **JSON** objects.

For example:

```
{"T":202, "line":1, "text":"http cmd test", "update":1}
```

You can send these JSON payloads via **HTTP POST** or **GET**.

The driver parses each command, executes the corresponding action, and sends back a JSON response.

Key advantages of this approach:

- No driver or special software required — any browser or script can talk to the board.
- Human-readable JSON makes debugging straightforward.
- Works across Python, JavaScript, C++, ESP-IDF, and more.
- Ideal for automation scripts or cloud-based control systems.

Essentially, the driver acts as a **mini web server** inside your robot — you just send HTTP requests to move motors, run actions, or query system status.

4. Environment Setup (Windows)

4.1 Install Python

1. Download and install the latest version of Python (≥ 3.10) from <https://www.python.org/downloads/>
 2. During installation, **check** the option “Add Python to PATH”.
-

4.2 Locate Example Scripts

Inside the cloned project, navigate to:

```
Example for Robot Driver Lite/python_example/http
```

This folder contains the HTTP control examples.

Power on your driver board before proceeding.

4.3 Create a Python Virtual Environment

This step isolates dependencies and avoids version conflicts between projects.

You only need to do it once per folder.

1. In the example folder, **right-click** → “Open in Terminal”.
2. Run:

```
python -m venv venv
```

3. Activate the virtual environment:

```
.\venv\Scripts\activate
```

4. Install required packages:

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

4.4 Edit the Example Script

Open `http_example.py` with your preferred editor (Notepad, VSCode, or SublimeText).

Locate and update the `ESP32_IP` variable:

```
ESP32_IP = "192.168.4.1"
```

- If your PC connects directly to the board's Wi-Fi hotspot, use `192.168.4.1`.
- If both are on the same LAN, use the IP address shown on the OLED screen after boot.

Example modification:

```
data = {"T":202, "line":1, "text":"Hello, world!", "update":1}  
send_json_command(data)
```

Save the file when done.

4.5 Run the Script

In the terminal, execute:

```
python .\http_example.py
```

You should see feedback printed in the terminal as the robot board processes your HTTP command.

5. Example Overview

The provided example `http_example.py` demonstrates how to:

- Send JSON commands to the robot board via HTTP
- Display text on the onboard OLED
- Easily extend your own automation scripts

You can modify or expand the example to control servos, execute motion sequences, or integrate it into a larger application.

6. Summary

Using HTTP control, your robot driver board effectively becomes a **network-accessible robot node**.

Whether you're prototyping, teaching, or developing an advanced automation system, this approach offers a simple yet powerful way to build high-level control logic — entirely over Wi-Fi.