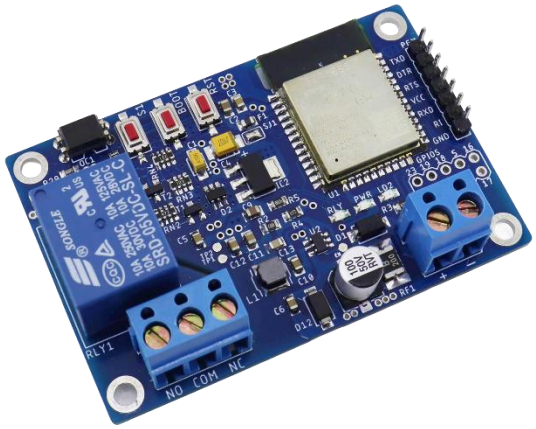

ESP32 One Channel Relay Module User Guide

Table of Contents

List of Figures

Introduction

The ESP32 One-Channel Relay Module is a compact, Wi-Fi and Bluetooth-enabled switching module designed to control a single high-voltage or high-current load using the ESP32 microcontroller. It uses a 5V DC SPDT relay, allowing the module to switch between Normally Open (NO) and Normally Closed (NC) contacts as required by the application.

To ensure maximum safety and noise immunity, the module features opto-isolated control using a PC817 optocoupler. This isolates the ESP32's 3.3V logic from the relay's 5V driving section, preventing electrical disturbances from affecting the microcontroller. The module also includes indicator LEDs for relay status, power status, and a user-configurable LED, along with 8 accessible GPIO pins and a UART interface for external device communication & programming.

The board integrates all required driver circuitry, including a transistor relay driver stage, flyback diode protection, and both 5V and 3.3V voltage regulation, along with convenient connector interfaces. This makes the module suitable for beginners and professionals seeking a reliable solution for automation, IoT projects, and remote switching applications.

Key Features

- Industrial-graded design
- On-board 5V and 3.3V voltage regulators for stable operation
- 5V DC SPDT relay for controlling high-voltage or high-current loads
- Indicator LEDs for power status, relay status, and a user-configurable LED
- Reset, Boot, and User-Configurable switches included
- Optocoupler isolation for safe ESP32–relay separation
- UART communication interface for programming
- Automatic Reset & Boot support via DTR and RTS signals or manual push buttons

Typical Applications

- Industrial and lab automation
- Home automation
- Wi-Fi/Bluetooth-based remote switching
- IoT and smart control projects
- Security and access control systems

Working principle

In the normal state, the relay is OFF and its contacts are in the Normally Closed (NC) position. When the relay is activated, the contacts switch to the Normally Open (NO) position. The relay is

connected to GPIO25 through optocoupler circuitry. To turn the relay ON, GPIO25 must be set LOW; to turn the relay OFF, GPIO25 must be set HIGH.

A user-configurable LED is connected to GPIO2. To turn the LED ON, GPIO2 should be set HIGH.

A user-configurable push button is connected to GPIO34, which is internally pulled up. When the button is pressed, GPIO34 reads LOW.

Pin Assignments

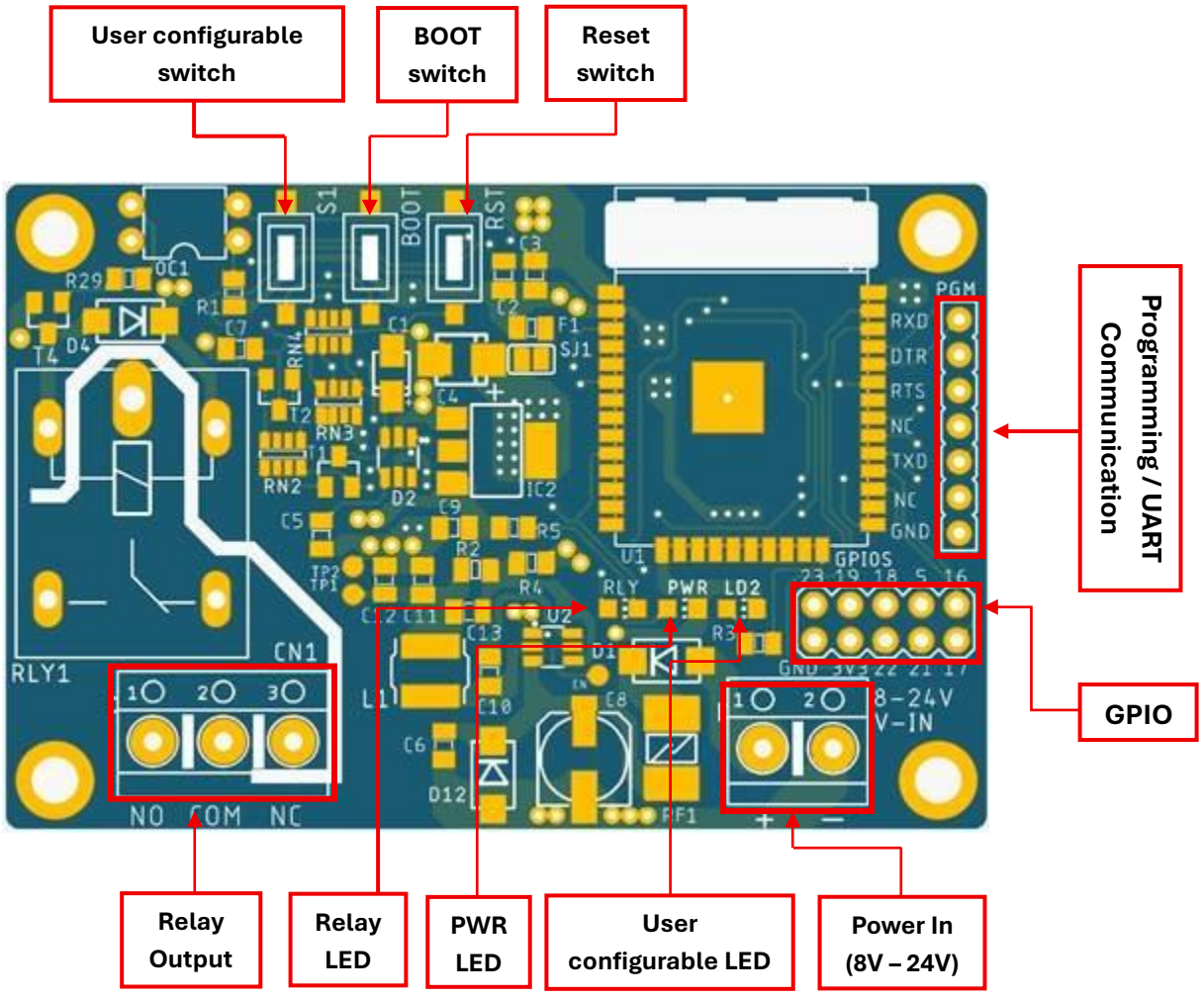

Figure 1 : Connector identification diagram

- If UART communication does not work, try swapping the RXD and TXD connections.
- For programming, a UART-to-USB connector is required. The recommended device is the FT232 USB UART converter.
    - **Ensure that the UART interface and its control signals operate at the 3.3V logic level.**

Board Specifications

Specification	Description
Power supply	Nominal voltage: 8V – 24V Maximum voltage: 24V Minimum voltage: 8V Maximum current: 2A
Microcontroller	Model: ESP32-WROOM-32D Processor: Dual-core 32-bit CPU, up to 240MHz RAM: 520 KB SRAM Flash: 4 MB Wi-Fi: 802.11 b/g/n, up to 150 Mbps Bluetooth: v4.2 (classic & BLE) Antenna: Onboard PCB antenna
User configurable features	Switch: User configurable tactile switch connected to GPIO34 Available Pins: GPIO 5, 16, 17, 18, 19, 21, 22, 23 LED: User configurable LED connected to GPIO2

Item	Contact Rating
Contact rating	10A 250VAC 10A 125VAC 10A 30VDC 10A 28VDC
Max switching current	10A
Max switching voltage	250AVC / 30VDC
Contact type	SPDT (Form C)

Programming the ESP32 Car Controller Board

- Pin connection for programming: The ESP32 automatically enters programming mode.
    - Once you upload the program, it will start running automatically.

PGM header on Relay Module	UART USB Converter
TXD	RXD
RXD	TXD
RTS	RTS
DTR	DTR
GND	GND

- If you use only TXD, RXD & GND pins:
    - Before uploading the code ESP32 needs to enter download mode for flashing (bootloader mode). For that you may need to:
        - Hold BOOT button while pressing RST
        - Release RST, then release BOOT
        - Now upload the code

- To run the code, switch the power off and back on.

Sample Codes

Programs can be uploaded via the Arduino IDE.

Relay Toggle Control with Push Button

The code is provided at the end of the document. This program shows how to control a one-channel relay module with the ESP32 using a push button. The button is connected to GPIO34 as an input, and the relay is connected to GPIO25 as an output. When the ESP32 starts, the relay is set to OFF. In the main loop, the ESP32 keeps checking the button. Each time the button is pressed (changing from HIGH to LOW), the relay switches its state: if it was OFF, it turns ON; if it was ON, it turns OFF. The variable relayState remembers whether the relay is ON or OFF, and lastButtonState makes sure the relay only changes once per press rather than repeatedly while the button is held down. This way, the button works like a simple power switch that toggles the relay ON and OFF with each press.

Bluetooth

- The code is provided at the end of the document.
- Before compiling the code `BluetoothSerial.h` library should be installed via the **Tools → Manage Libraries** in the Arduino IDE.

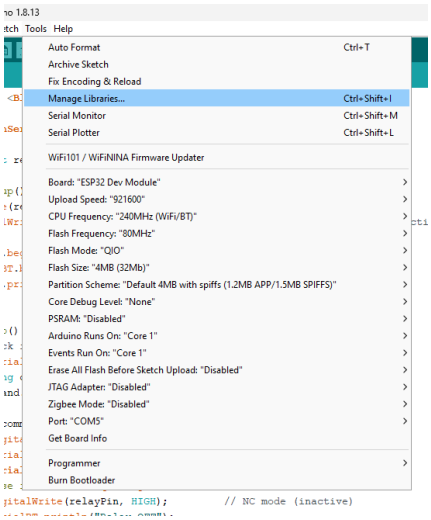

Figure 2 : Installing the required library

- Compile the code and upload it to the relay module using a USB-UART Converter.
- After uploading the code, the Serial Monitor will display the output. Make sure to set the baud rate of serial monitor to 115200.

```
Bluetooth started. Pair and send 'ON' or 'OFF'.
```

Figure 3 : Serial monitor

- Turn on Bluetooth on your phone and pair with ESP32_Relay.
- Bluetooth commands can be sent from your phone using the **Serial Bluetooth Terminal** app.
- When the ‘ON’ command is sent, the relay switches to the NO (Normally Open) mode.
- When the ‘OFF’ command is sent, the relay switches to the NC (Normally Closed) mode.
- Ex: When the ON and OFF commands (blue-colored ones) are sent via the app, the relay will switch on and off accordingly. Both the app and the Serial Monitor will display the relay status as “Relay ON” or “Relay OFF.” In the app, these messages are shown in green. If any command other than ON or OFF is sent, the message “Unknown command” will be displayed.

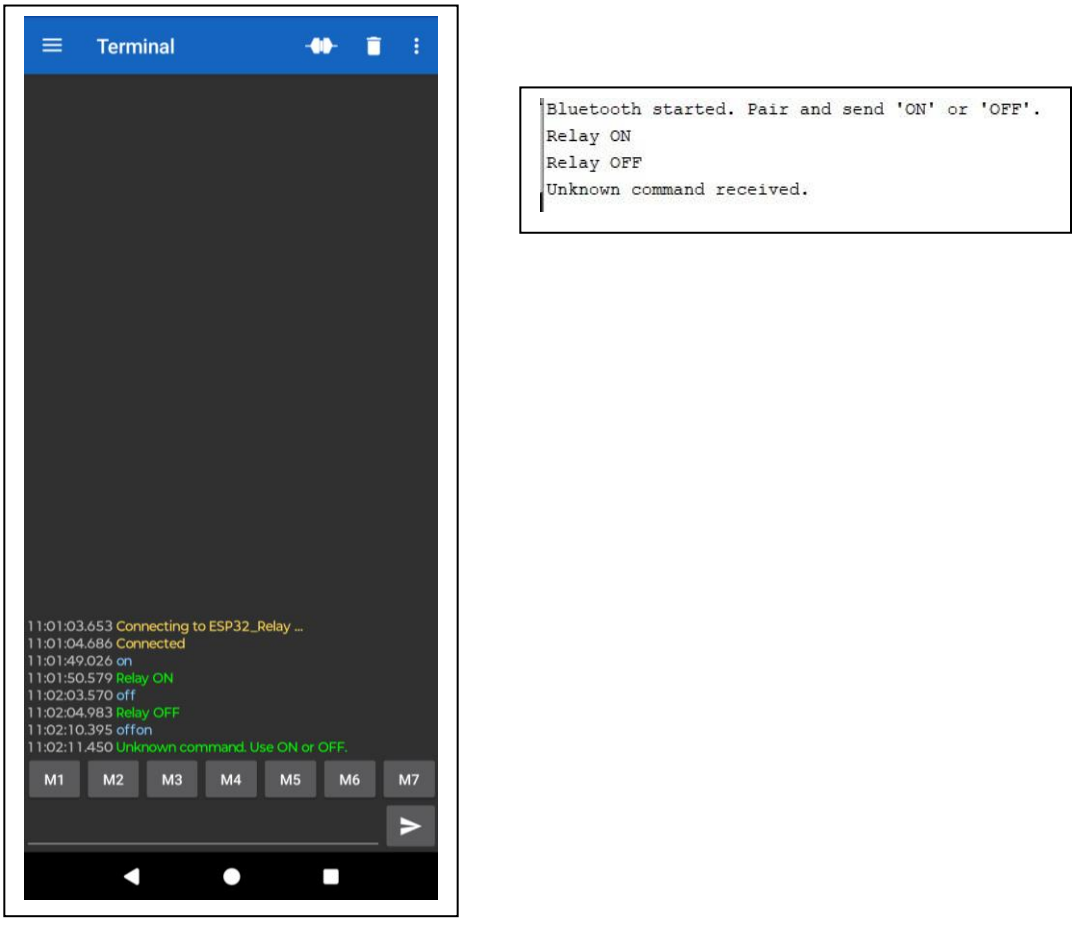

Figure 4 : Serial Bluetooth Terminal and Serial monitor Output

WiFi

The code is provided at the end of the document. This code demonstrates how to use an ESP32 in Access Point (AP) mode to host a simple web server that controls a relay. The relay can be turned ON or OFF through a web page served by the ESP32.

- Upload the code, open the Serial Monitor, and set the baud rate to 115200. It will then display the default IP address as follows,

```
ESP32 AP started
IP address: 192.168.4.1
```

Figure 5 : Serial monitor output

- Connect your phone/laptop to the WiFi network specified in the code:
  - SSID: ESP32_AP
  - Password: 12345678
- Open a browser and go to: http://192.168.4.1/ or type the IP address into the web browser, and you will see an interface like the one below.

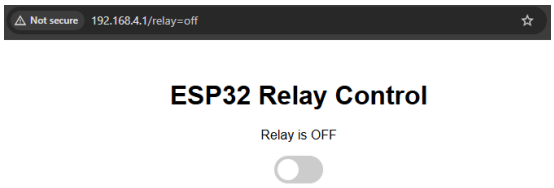

Figure 6 : Webpage interface for relay control via Wi-Fi

- Use the slider button to turn the relay on and off.

Relay toggle control with push button

```
//Relay Toggle Control with Push Button

// Define pin numbers
const int buttonPin = 34;   // GPIO34 as button input pin
const int relayPin  = 25;   // GPIO25 as relay output pin

// Variables to track states
int relayState = HIGH;      // Relay OFF at startup (active LOW relay module)
int lastButtonState = HIGH; // Previous button state

void setup() {
  pinMode(buttonPin, INPUT);    // Button pin as input
  pinMode(relayPin, OUTPUT);    // Relay pin as output

  digitalWrite(relayPin, relayState); // Relay OFF at startup
}

void loop() {
  // Read current button state
  int buttonState = digitalRead(buttonPin);

  // Detect a press event: button goes from HIGH → LOW
  if (buttonState == LOW && lastButtonState == HIGH) {
    // Toggle relay state
    relayState = (relayState == HIGH) ? LOW : HIGH;
    digitalWrite(relayPin, relayState);
  }

  // Save current state for next loop
  lastButtonState = buttonState;
}
```

Relay toggle control using Bluetooth

```
//Relay Toggle Control usign Bluetooth

#include <BluetoothSerial.h>   // Include Bluetooth Serial library

BluetoothSerial SerialBT;      // Create Bluetooth Serial object

const int relayPin = 25;       // GPIO25 controls relay (active-LOW)

void setup() {
  pinMode(relayPin, OUTPUT);   // Set relay pin as output
  digitalWrite(relayPin, HIGH); // Relay starts OFF (HIGH = inactive for active-LOW relay)

  Serial.begin(115200);        // Start Serial Monitor
  delay(1000);
  SerialBT.begin("ESP32_Relay"); // Start Bluetooth with device name
  Serial.println("Bluetooth started. Pair and send 'ON' or 'OFF'.");
}

void loop() {
  // Check if Bluetooth has received data
  if (SerialBT.available()) {
    String command = SerialBT.readString(); // Read incoming string
    command.trim();                         // Remove whitespace/newline

    if (command.equalsIgnoreCase("ON")) {
      digitalWrite(relayPin, LOW);          // NO mode (active-LOW)
      SerialBT.println("Relay ON");
      Serial.println("Relay ON");
    } else if (command.equalsIgnoreCase("OFF")) {
      digitalWrite(relayPin, HIGH);         // NC mode (inactive)
      SerialBT.println("Relay OFF");
      Serial.println("Relay OFF");
    } else {
      SerialBT.println("Unknown command. Use ON or OFF.");
      Serial.println("Unknown command received.");
    }
  }
}
```

Relay toggle control using WiFi

```
//Relay Toggle Control usign WiFi

#include <WiFi.h>  // Include WiFi library for ESP32

// ---------------- Pin Definitions ----------------
const int relayPin = 25;       // GPIO25 is connected to the relay module (active-LOW relay assumed)

// ---------------- Access Point Settings ----------
const char* ssid     = "ESP32_AP";     // Name of the WiFi Access Point (SSID)
const char* password = "12345678";     // Password for the WiFi AP (must be at least 8 characters)

// Create a web server object that listens on port 80 (HTTP default)
WiFiServer server(80);

// ---------------- State Variables ----------------
bool relayState = false; // Tracks relay state: false = OFF, true = ON

void setup() {
  // Configure relay pin as output
  pinMode(relayPin, OUTPUT);

  // Ensure relay is OFF at startup
  // Active-LOW relay: HIGH = OFF, LOW = ON
  digitalWrite(relayPin, HIGH);

  // Start serial communication for debugging
  Serial.begin(115200);
  delay(1000); // Small delay to stabilize serial output

  // Start ESP32 in Access Point (AP) mode
  WiFi.softAP(ssid, password);

  // Print AP status and IP address to Serial Monitor
  Serial.println("ESP32 AP started");
  Serial.print("IP address: ");
  Serial.println(WiFi.softAPIP());

  // Start the web server
  server.begin();
}

void loop() {
  // Check if a client (browser) has connected
  WiFiClient client = server.available();
  if (client) {
    // Read the HTTP request from the client until carriage return
    String request = client.readStringUntil('\r');
    client.flush(); // Clear remaining data

    // ---------------- Parse Request ----------------
    // If request contains "/relay=on", turn relay ON (NO closes, NC opens)
    if (request.indexOf("/relay=on") != -1) {
      relayState = true;
      Serial.println("Relay ON");
    }
    // If request contains "/relay=off", turn relay OFF (NC closes, NO opens)
    else if (request.indexOf("/relay=off") != -1) {
      relayState = false;
      Serial.println("Relay OFF");
    }

    // ---------------- Update Relay Output ----------------
    // Active-LOW relay: LOW = ON, HIGH = OFF
    digitalWrite(relayPin, relayState ? LOW : HIGH);

    // ---------------- Send HTML Response ----------------
    // Send standard HTTP response headers
    client.println("HTTP/1.1 200 OK");
    client.println("Content-type:text/html");
    client.println();

    // Begin HTML page
    client.println("<!DOCTYPE html><html>");
    client.println("<head>");
    client.println("<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

    // Inline CSS styling for toggle switch
    client.println("<style>");
    client.println("body { font-family: Arial; text-align: center; margin-top: 50px; }");
    client.println(".switch { position: relative; display: inline-block; width: 60px; height: 34px; }");
    client.println(".switch input { display: none; }");
    client.println(".slider { position: absolute; cursor: pointer; top: 0; left: 0; right: 0; bottom: 0;");
    client.println("background-color: #ccc; transition: .4s; border-radius: 34px; }");
```

```
    client.println(".slider:before { position: absolute; content: \"\"; height: 26px; width: 26px; left: 4px; bottom: 4px;");
    client.println("background-color: white; transition: .4s; border-radius: 50%; }");
    client.println("input:checked + .slider { background-color: #4CAF50; }");
    client.println("input:checked + .slider:before { transform: translateX(26px); }");
    client.println("</style>");
    client.println("</head>");

    // Body content
    client.println("<body>");
    client.println("<h1>ESP32 Relay Control</h1>");

    // Show current relay state
    client.print("<p>Relay is ");
    client.print(relayState ? "ON" : "OFF");
    client.println("</p>");

    // Toggle button (styled checkbox)
    client.println("<label class=\"switch\">");
    client.print("<input type=\"checkbox\" onchange=\"toggleRelay(this)\" ");
    if (relayState) client.print("checked"); // Keep checkbox synced with relay state
    client.println(">");
    client.println("<span class=\"slider\"></span>");
    client.println("</label>");

    // JavaScript function to send requests when toggle changes
    client.println("<script>");
    client.println("function toggleRelay(element) {");
    client.println("  if(element.checked) { window.location.href = '/relay=on'; }");
    client.println("  else { window.location.href = '/relay=off'; }");
    client.println("}");
    client.println("</script>");

    // End HTML page
    client.println("</body></html>");
    client.println();
  }
}
```

Doc Version: 1.0.0
Product Version: 1.0.0
Report issues to: support@tesla.lk