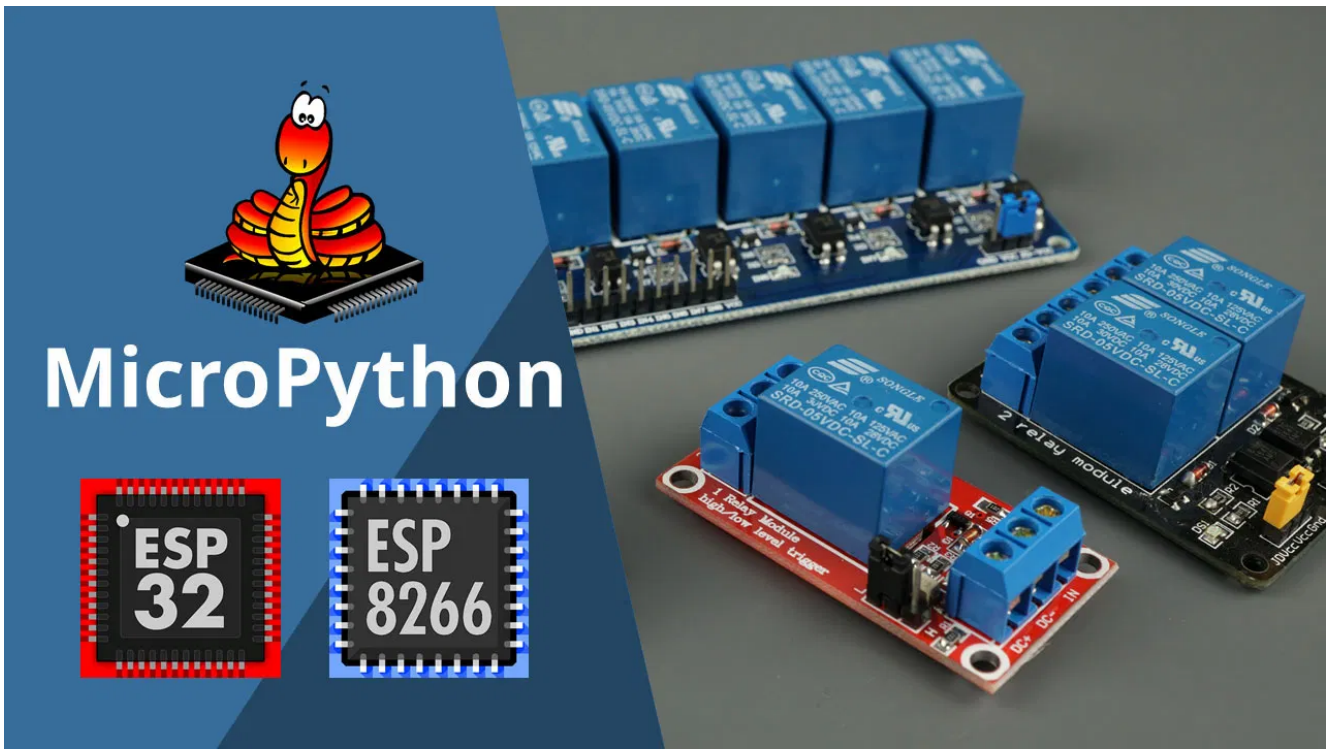
# MicroPython: Relay Module with ESP32/ESP8266 (Guide + Web Server)

Using a relay with the ESP32 or ESP8266 is a great way to control AC household appliances remotely. This tutorial explains how to control a relay module with the ESP32 or ESP8266 using MicroPython firmware.




We'll take a look at how a relay module works, how to connect the relay to the ESP32 or ESP8266 boards and build a web server to control a relay remotely.

We have similar guides using Arduino IDE:

- **Guide for ESP32 Relay Module with Arduino IDE** – Control AC Appliances + Web Server Example
- **Guide for ESP8266 Relay Module with Arduino IDE** – Control AC Appliances + Web Server Example

## Prerequisites

board. We suggest using Thonny IDE or uPyCraft IDE:

- Thonny IDE:
  - Installing and getting started with Thonny IDE
  - Flashing MicroPython Firmware with esptool.py
- uPyCraft IDE:
  - Getting Started with uPyCraft IDE
  - Install uPyCraft IDE (Windows, Mac OS X, Linux)
  - Flash/Upload MicroPython Firmware to ESP32 and ESP8266

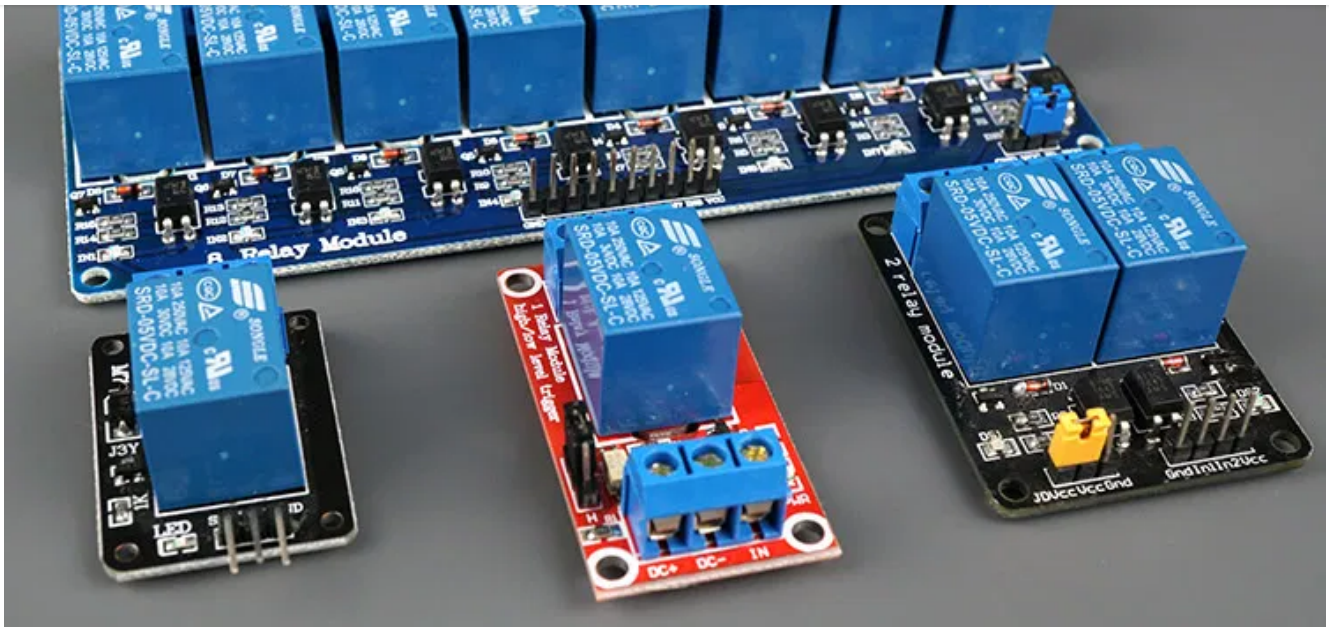
> Learn more about MicroPython: MicroPython Programming with ESP32 and ESP8266 eBook.

## Introducing Relays

A relay is an electrically operated switch and like any other switch, it that can be turned on or off, letting the current go through or not. It can be controlled with low voltages, like the 3.3V provided by the ESP32/ESP8266 GPIOs and allows us to control high voltages like 12V, 24V or mains voltage (230V in Europe and 120V in the US).

### 1, 2, 4, 8, 16 Channels Relay Modules

There are different relay modules with a different number of channels. You can find relay modules with one, two, four, eight and even sixteen channels. The number of channels determines the number of outputs we'll be able to control.




There are relay modules whose electromagnet can be powered by 5V and with 3.3V. Both can be used with the ESP32 or ESP8266 – you can either use the VIN pin (that provides 5V) or the 3.3V pin.

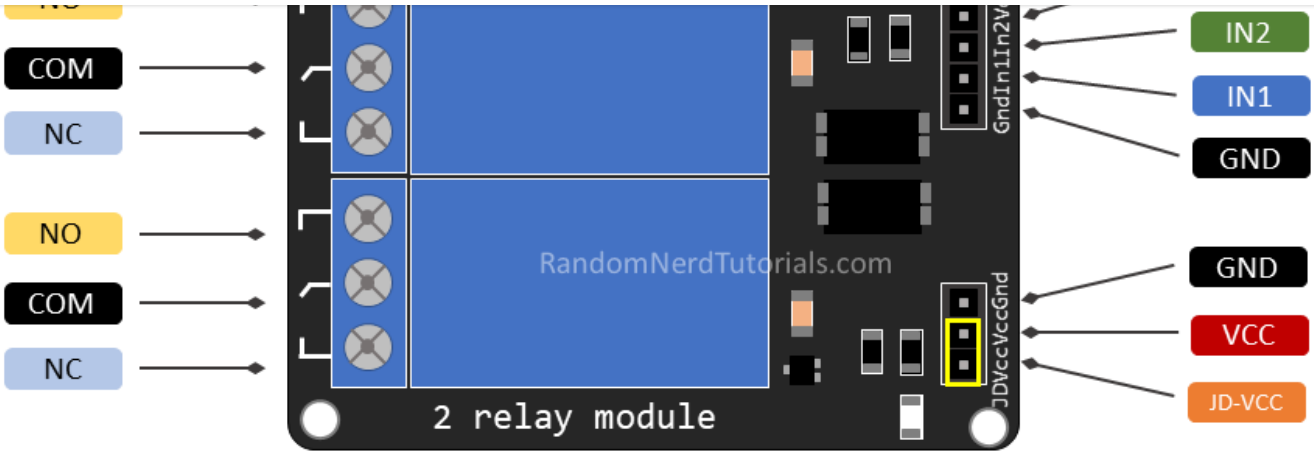
Additionally, some come with built-in optocoupler that add an extra "layer" of protection, optically isolating the ESP boards from the relay circuit.

**Get a relay module**:

- 5V 2-channel relay module (with optocoupler)
- 5V 1-channel relay module (with optocoupler)
- 5V 8-channel relay module (with optocoupler)
- 5V 16-channel relay module (with optocoupler)
- 3.3V 1-channel relay module (with optocoupler)

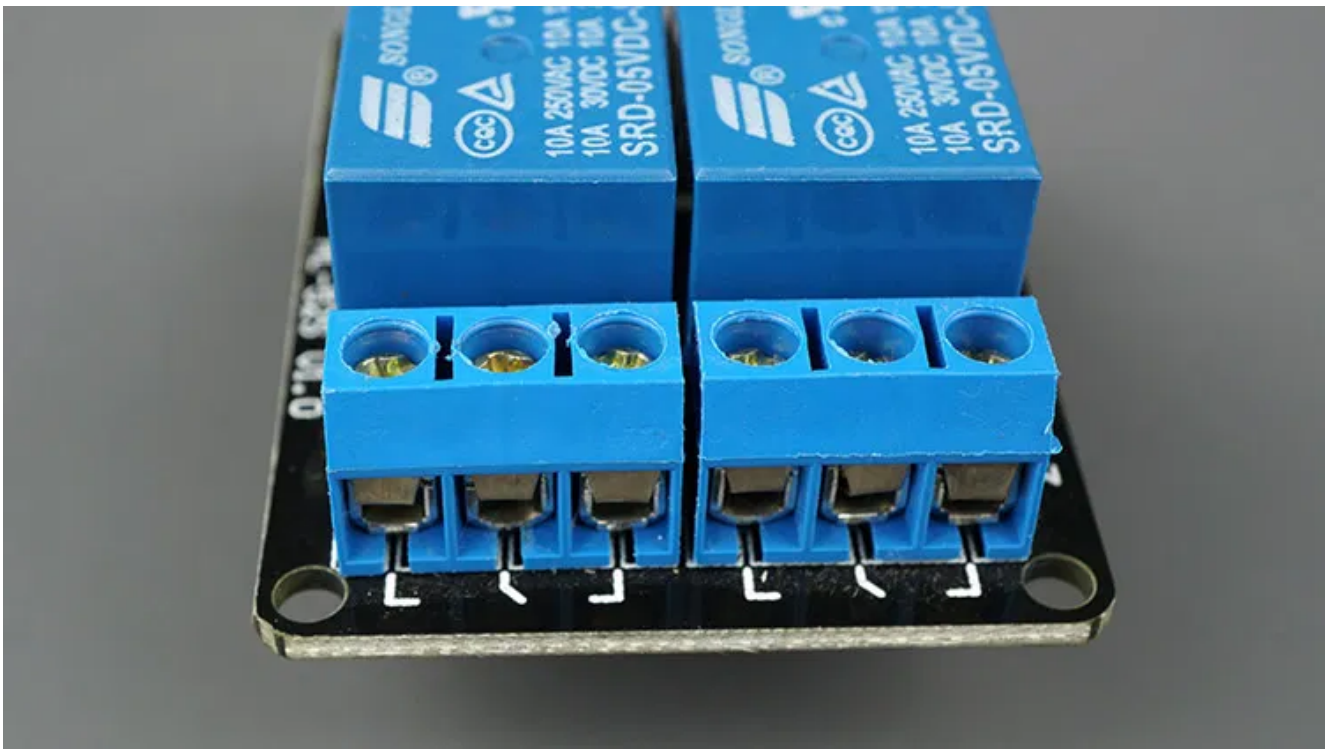
## Relay Pinout

For demonstration purposes, let's take a look at the pinout of a 2-channel relay module. Using a relay module with a different number of channels is similar.




On the left side, there are two sets of three sockets to connect high voltages, and the pins on the right side (low-voltage) connect to the ESP GPIOs.

## Mains Voltage Connections



The relay module shown in the previous photo has two connectors, each with three sockets: common ( COM ), Normally Closed ( NC ), and Normally Open ( NO ).

- **COM:** connect the current you want to control (mains voltage).
- **NC (Normally Closed):** the normally closed configuration is used when you want the relay to be closed by default. The NC are COM pins are connected, meaning the current is flowing unless you send a signal from

- **NO (Normally Open):** the normally open configuration works the other way around: there is no connection between the NO and COM pins, so the circuit is broken unless you send a signal from the ESP to close the circuit.

## Control Pins


The low-voltage side has a set of four pins and a set of three pins. The first set consists of VCC and GND to power up the module, and input 1 ( IN1 ) and input 2 ( IN2 ) to control the bottom and top relays, respectively.

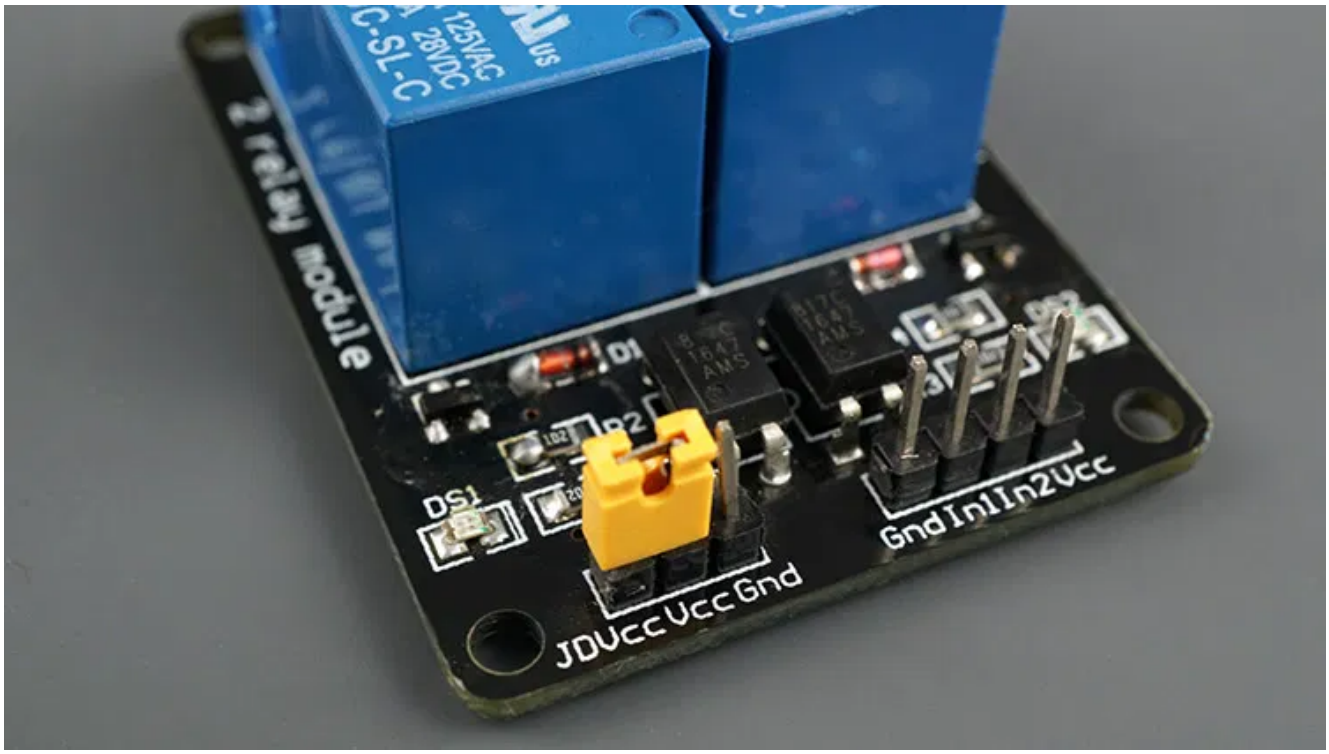
If your relay module only has one channel, you'll have just one IN pin. If you have four channels, you'll have four IN pins, and so on.

The signal you send to the IN pins, determines whether the relay is active or not. The relay is triggered when the input goes below about 2V. This means that you'll have the following scenarios:

- **Normally Closed configuration (NC)**:
  - HIGH signal – current is flowing
  - LOW signal – current is **not** flowing
- **Normally Open configuration (NO)**:
  - HIGH signal – current is **not** flowing
  - LOW signal – current in flowing

Use a normally open configuration when you want the current to flow occasionally (for example, turn on a lamp occasionally).

## Power Supply Selection




The second set of pins consists of GND, VCC, and JD-VCC pins. The JD-VCC pin powers the electromagnet of the relay. Notice that the module has a jumper cap connecting the VCC and JD-VCC pins; the one shown here is yellow, but yours may be a different color.

With the jumper cap on, the VCC and JD-VCC pins are connected. That means the relay electromagnet is directly powered from the ESP power pin, so the relay module and the ESP circuits are not physically isolated from each other.

Without the jumper cap, you need to provide an independent power source to power up the relay's electromagnet through the JD-VCC pin. That configuration physically isolates the relays from the ESP with the module's built-in optocoupler, which prevents damage to the ESP in case of electrical spikes.

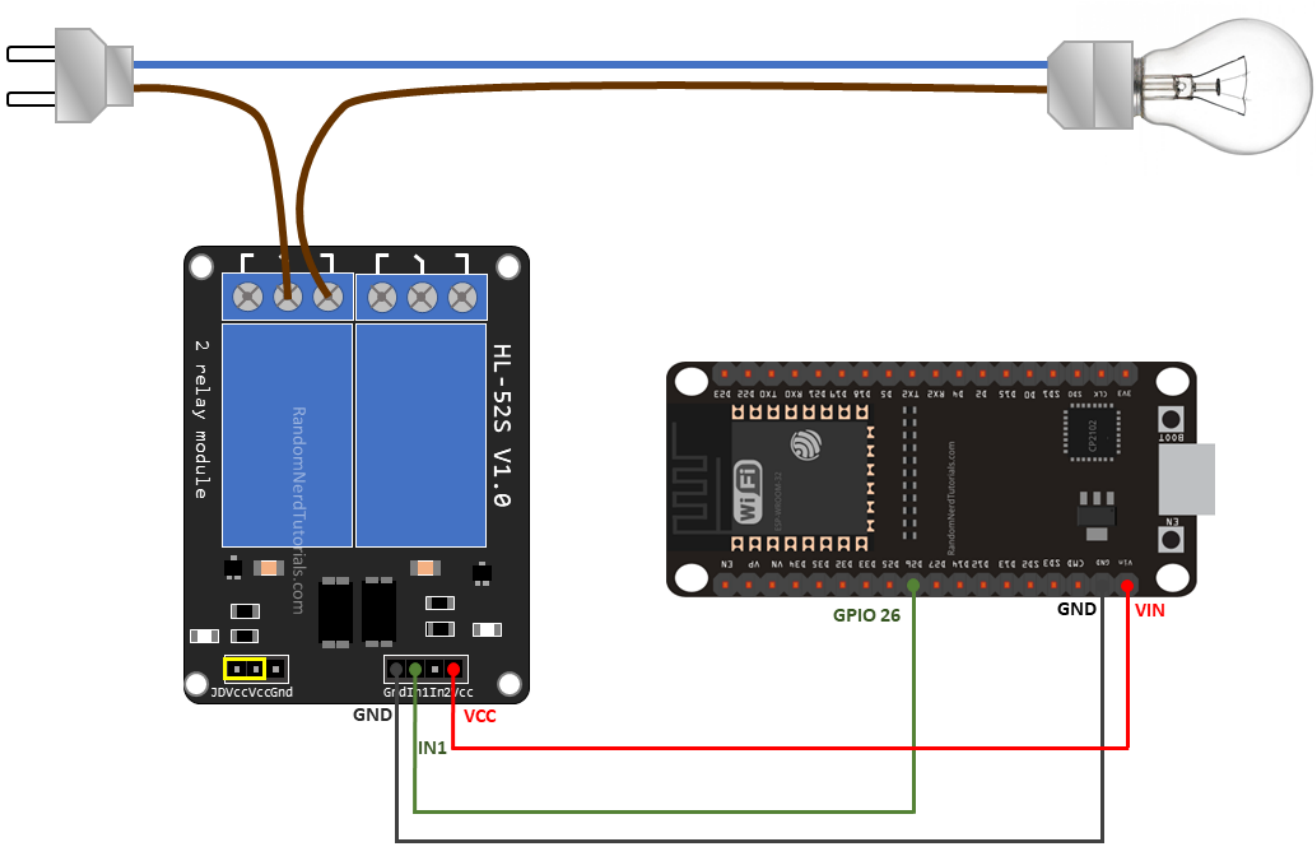
## Wiring a Relay Module to the ESP32/ESP8266

> someone who is to help you out. While programming the ESP or wiring your circuit make sure everything is disconnected from mains voltage.

Alternatively, you can use a 12V power source to control 12V appliances.

## ESP32 Schematic Diagram

Connect the relay module to the ESP32 as shown in the following diagram. The diagram shows wiring for a 2-channel relay module, wiring a different number of channels is similar.

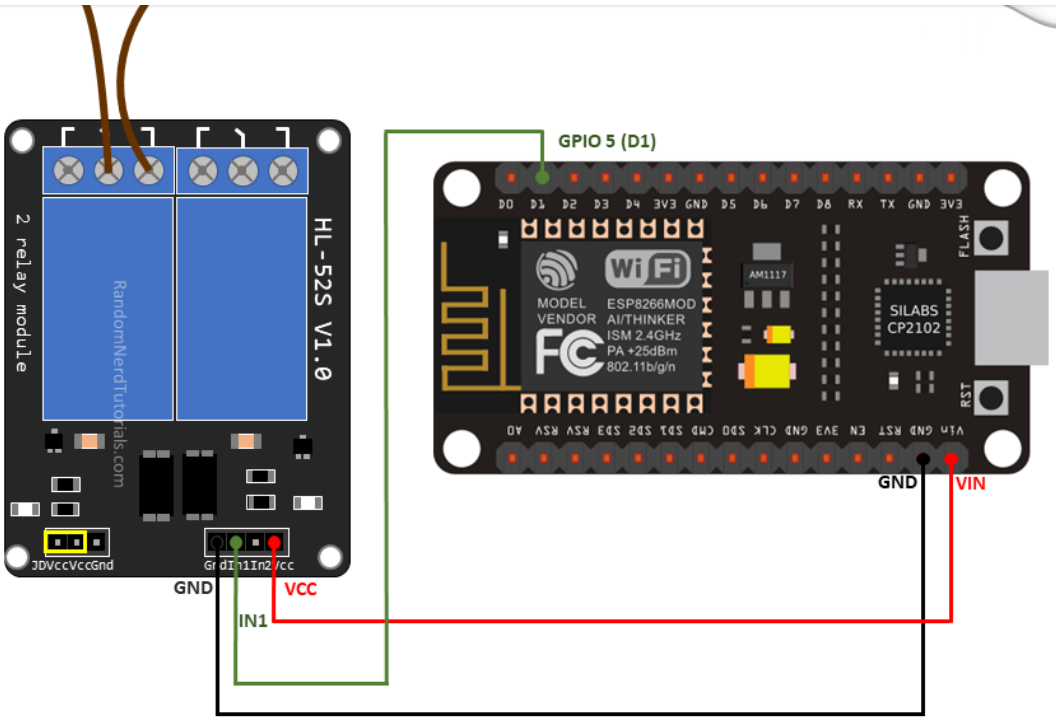



In this example, we're controlling a lamp. We just want to light up the lamp occasionally, so it is better to use a normally open configuration.

> We're connecting the IN1 pin to GPIO 26 , you can use any other suitable GPIO. See ESP32 GPIO Reference Guide.

## ESP8266 Schematic Diagram

Follow the next schematic diagram if you're using an ESP8266.

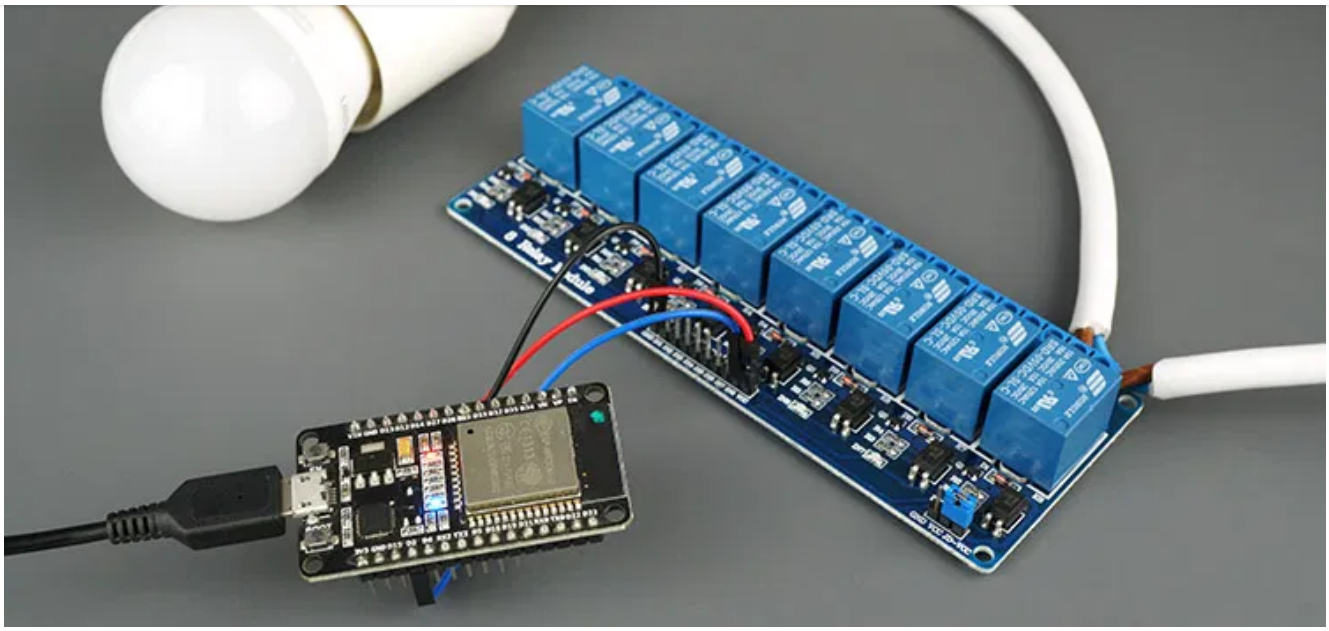



We're connecting the IN1 pin to GPIO 5, you can use any other suitable GPIO. See ESP8266 GPIO Reference Guide.

The best ESP8266 pins to use with relays are: GPIO 5, GPIO 4, GPIO 14, GPIO 12 and GPIO 13.

## Controlling a Relay Module – MicroPython Code (Script)

The code to control a relay with the ESP32 or ESP8266 is as simple as controlling an LED or any other output. In this example, as we're using a normally open configuration, we need to send a LOW signal to let the current flow, and a HIGH signal to stop the current flow.



Copy the following code to the *main.py* file and upload it to your board. It lights up your lamp for 10 seconds and turn it off for another 10 seconds.

```
# Complete project details at https://RandomNerdTutorials.com

from machine import Pin
from time import sleep

# ESP32 GPIO 26
relay = Pin(26, Pin.OUT)

# ESP8266 GPIO 5
#relay = Pin(5, Pin.OUT)

while True:
  # RELAY ON
  relay.value(0)
  sleep(10)
  # RELAY OFF
  relay.value(1)
  sleep(10)
```

View raw code

Import the `Pin` class from the `machine` module to interact with the GPIOs. We also import the `sleep()` method from the `time` module to add delays.

```
from machine import Pin
from time import sleep
```

Then, we define a `Pin` object called `relay` on `26` (if you're using an ESP32) and define it as an output.

```
# ESP32 GPIO 26
relay = Pin(26, Pin.OUT)
```

In case you're using an ESP8266, use `GPIO 5` instead. Comment the previous line and uncomment the following.

```
# ESP8266 GPIO 5
#relay = Pin(5, Pin.OUT)
```

In the while loop, send a LOW signal to light up the lamp for 10 seconds.

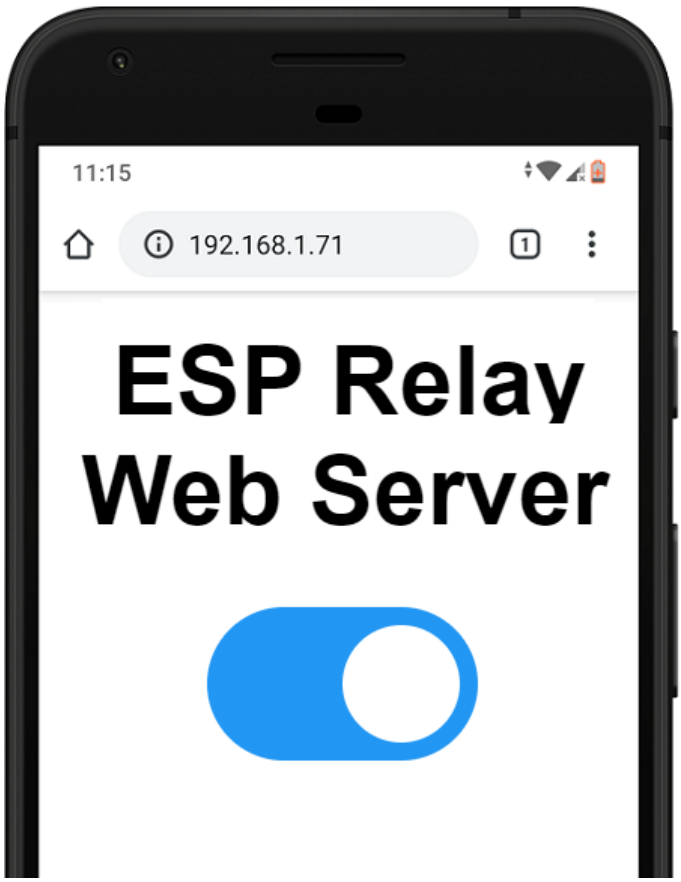
```
# RELAY ON
relay.value(0)
sleep(10)
```

If you're using a normally closed configuration, send a HIGH signal to light up the lamp.

Stop the current flow by sending a HIGH signal to the relay pin. If you're using a normally closed configuration, send a LOW signal to stop the current flow.

```
# RELAY OFF
relay.value(1)
```

# Control Relay Module with MicroPython Web Server




In this section, we've created a web server example that allows you to control a relay remotely via web server.

## boot.py

Copy the following code to your *boot.py* file.

```
# Complete project details at https://RandomNerdTutorials.com

try:
  import usocket as socket
except:
  import socket

from machine import Pin
import network
```

```
esp.osdebug(None)

import gc
gc.collect()

ssid = 'REPLACE_WITH_YOUR_SSID'
password = 'REPLACE_WITH_YOUR_PASSWORD'

station = network.WLAN(network.STA_IF)

station.active(True)
station.connect(ssid, password)

while station.isconnected() == False:
  pass
```

View raw code

Insert your network credentials in the following variables:

```
ssid = 'REPLACE_WITH_YOUR_SSID'
password = 'REPLACE_WITH_YOUR_PASSWORD'
```

Uncomment one of the following lines accordingly to the board you're using. By default, it's set to use the ESP32 GPIO.

```
# ESP32 GPIO 26
relay = Pin(26, Pin.OUT)

# ESP8266 GPIO 5
#relay = Pin(5, Pin.OUT)
```

## main.py

```
# Complete project details at https://RandomNerdTutorials.com


def web_page():
  if relay.value() == 1:
    relay_state = ''
  else:
    relay_state = 'checked'
  html = """<html><head><meta name="viewport" content="width=device-wid
  body{font-family:Arial; text-align: center; margin: 0px auto; padding
  .switch{position:relative;display:inline-block;width:120px;height:68p
  .slider{position:absolute;top:0;left:0;right:0;bottom:0;background-co
  .slider:before{position:absolute;content:"";height:52px;width:52px;le
  input:checked+.slider{background-color:#2196F3}
  input:checked+.slider:before{-webkit-transform:translateX(52px);-ms-t
  </style><script>function toggleCheckbox(element) { var xhr = new XMLH
  else { xhr.open("GET", "/?relay=off", true); } xhr.send(); }</script>
  <h1>ESP Relay Web Server</h1><label class="switch"><input type="checkb
  </span></label></body></html>""" % (relay_state)
  return html

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)

while True:
  try:
    if gc.mem_free() < 102000:
```

View raw code

We won't explain how this code works because we already have a very similar tutorial with detailed explanation of each line of code. Read the next project:

- ESP32/ESP8266 MicroPython Web Server – Control Outputs

## Demonstration

address.

Then, open a browser in your local network and type the ESP IP address to get access to the web server.

You should get a web page with a toggle button that allows you to control your relay remotely using your smartphone or your computer.

## Enclosure for Safety

For a final project, make sure you place your relay module and ESP inside an enclosure to avoid any AC pins exposed.

## Wrapping Up

In this tutorial you've learned how to control relays with the ESP32 or ESP8266 using MicroPython. We have similar guides using Arduino IDE:

- [Arduino IDE] Guide to control a Relay Module with the **ESP32**
- [Arduino IDE] Guide to control a Relay Module with **ESP8266**

Controlling a relay with the ESP32 or ESP8266 is as easy controlling any other output, you just need to send HIGH and LOW signals as you would do to control an LED.

You can use our web server examples that control outputs to control relays. You just need to pay attention to the configuration you're using. In case you're using a normally open configuration, the relay works with inverted logic. You can use the following web server examples to control your relay:

- ESP32 Web Server – Arduino IDE
- ESP32 Web Server using SPIFFS (control outputs)
- ESP32/ESP8266 MicroPython Web Server – Control Outputs

Learn more about MicroPython with the ESP32 and ESP8266 with our resources:

- **MicroPython Programming with the ESP32 and ESP8266 (eBook)**

Thanks for reading.




**[eBook] Build Web Servers with ESP32 and ESP8266 (2nd Edition)**

Build Web Server projects with the ESP32 and ESP8266 boards to control outputs and monitor sensors remotely. Learn HTML, CSS, JavaScript and client-server communication protocols **DOWNLOAD »**

## Recommended Resources

**Build a Home Automation System from Scratch »** With Raspberry Pi, ESP8266, Arduino, and Node-RED.

## What to Read Next...

Control ESP32 and ESP8266 GPIOs from Anywhere in the World

ESP32 Capacitive Touch Sensor Pins with Arduino IDE

ESP32 with MPU-6050 Accelerometer, Gyroscope and Temperature Sensor (Arduino)

ESP8266 NodeMCU Web Server with Slider: Control LED Brightness (PWM)

ESP8266 NodeMCU MQTT – Publish BME680 Temperature, Humidity, Pressure, and Gas Readings (Arduino IDE)

ESP8266 Publishing DHT22 Readings to SQLite Database

**Enjoyed this project? Stay updated by subscribing our weekly newsletter!**

## 3 thoughts on “MicroPython: Relay Module with ESP32/ESP8266 (Guide + Web Server)”

**Brian Hambleton**
February 26, 2020 at 7:34 pm

I am planning a similar system but with 8 relays. I have an 8 relay card physically interfaced to an ESP32, via Port ‘A’ on a MCP23017 I/O Expander. I would like to use Port ‘B’ of the MCP23017 for local control of the relays. Suggestions on how best to mod the code here to accommodate my setup? Thanks.

Reply

**Mihael**
April 7, 2020 at 6:40 pm

Hi, I made a similar project with ESP-01 and 1-channel relay. The relay is connected to GPIO0, have boot.py and main.py programs. It works ok when I connect the relay after the ESP was booted. But by finished and connected device, every time after connecting to the power network, the relay is high, and main.py was avoided.
It is probably hardware problem, does it help if I put resistor or capacitor to GPIO0? Or is ESP8266 bad choice for such a device?

Reply

October 13, 2020 at 5:38 pm

This is great, but even better would be a project using those boards that include the relays plus an ESP-01S board that attaches directly. Those are trickier because they have a serial connection to the relays but it is a lot nicer to have that single board than wiring the two boards together as you have here.

Reply

## Leave a Comment

Name *

Email *

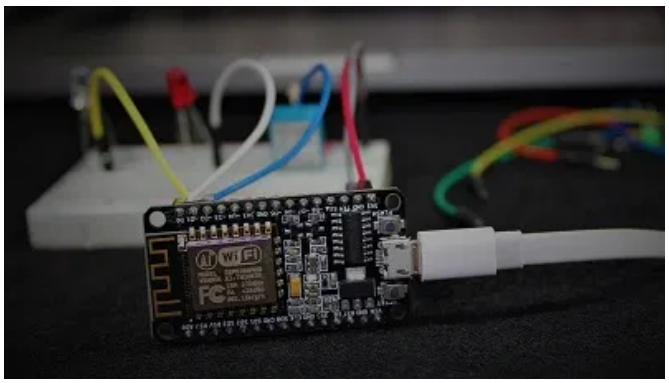
Website

Notify me of follow-up comments by email.

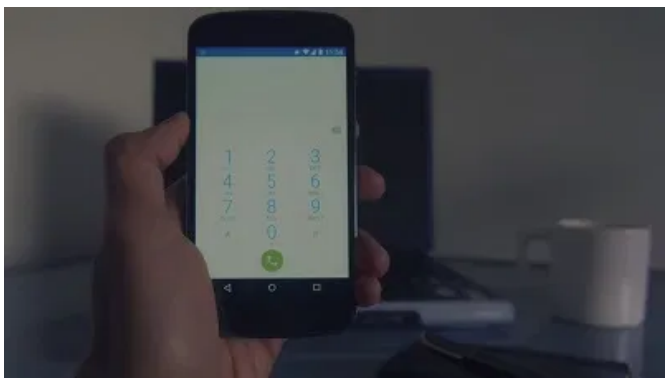
Notify me of new posts by email.

Post Comment

**Visit Maker Advisor – Tools and Gear for makers, hobbyists and DIYers »**



**Home Automation using ESP8266 eBook and video course »** Build IoT and home automation projects.



**Build a Home Automation System from Scratch »** With Raspberry Pi, ESP8266, Arduino, and Node-RED.