ESP32-Relais-Modul – Steuerung AC Appliances (Web-Server)

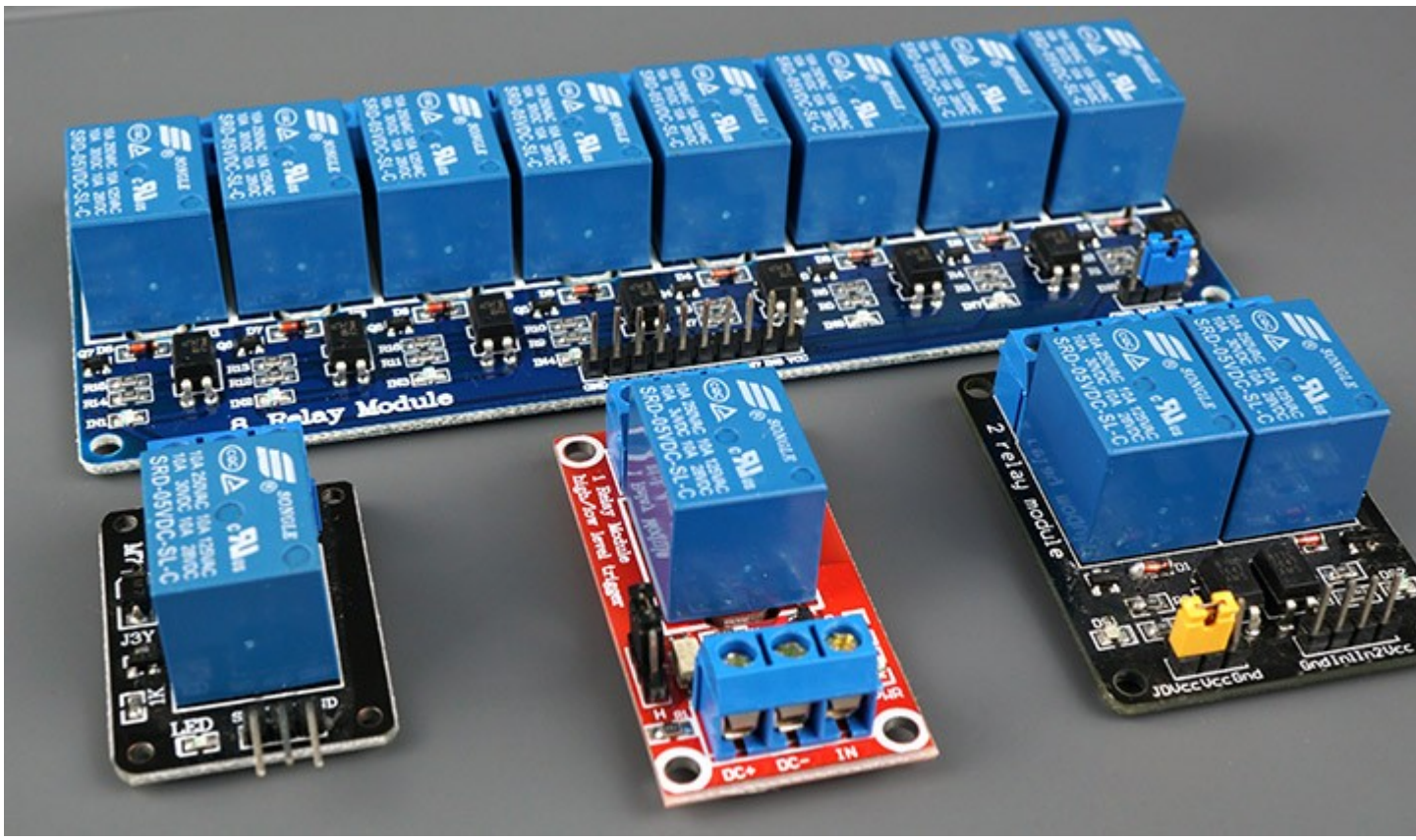
Ein relais mit dem ESP32 ist eine großartige Möglichkeit, um control AC Haushaltsgeräte aus der Ferne. In diesem tutorial wird erklärt, wie man eine relais-Modul mit dem ESP32. Wir werden einen Blick auf, wie ein relais-Modul arbeiten, so schließen Sie das relais an den ESP32 und bauen ein web-server zur Steuerung eines relais aus der Ferne (oder wie viele relais, wie Sie wollen).

Die Einführung Von Relais

Ein relais ist ein elektrisch betrieben Schalter, und wie jeder andere Schalter, es kann gedreht werden auf oder off, lassen Sie den Strom gehen Sie durch oder nicht. Gesteuert werden können mit niedrigen Spannungen, wie die 3,3 V zur Verfügung gestellt von der ESP32 GPIOs und ermöglicht es uns zu control hohe Spannungen wie 12 V, 24 V oder Netzspannung (230V in Europa und 120-V in den USA).

1, 2, 4, 8, 16 Kanäle, Relais-Module

Es gibt verschiedene relais-Module mit einer unterschiedlichen Anzahl von Kanälen. Finden Sie die relay-Module mit einem, zwei, vier, acht oder sogar sechzehn Kanäle. Die Anzahl der Kanäle bestimmt die Anzahl der Ausgänge wir werden in der Lage sein zu kontrollieren.



Es gibt relais-Module, deren Elektromagnet mit Strom versorgt werden kann durch die 5V und 3,3 V. Beide können verwendet werden mit die ESP32 – Sie können entweder verwenden Sie den VIN-pin (liefert 5 V) oder 3,3 V pin.

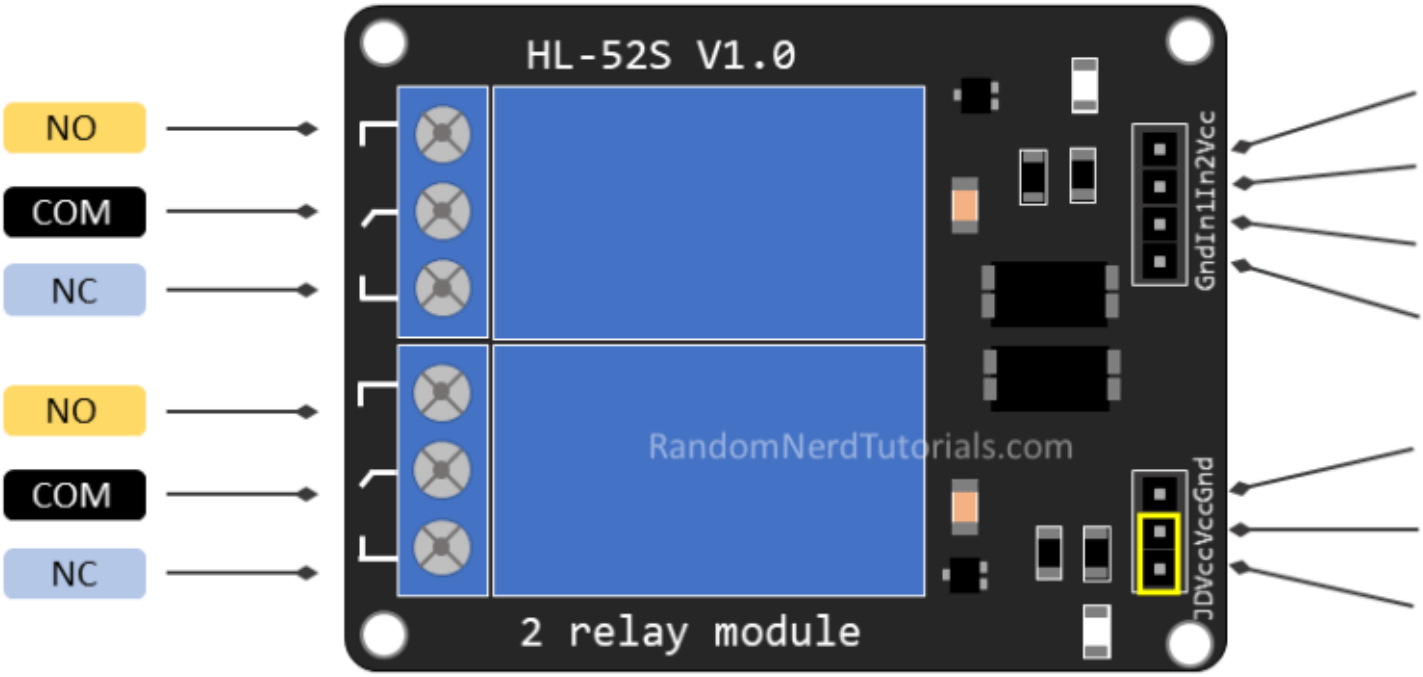
Zusätzlich, einige kommen mit built-in-Optokoppler, die fügen Sie eine zusätzliche "Schicht" der Schutz, optisch isoliert der ESP32 von der relais-Schaltung.

Holen Sie sich ein relais-Modul :

- 5V 2-Kanal relais-Modul (mit Optokoppler)
- 5V 1-Kanal-relais-Modul (mit Optokoppler)
- 5V 8-Kanal relais-Modul (mit Optokoppler)
- 5V 16-Kanal-relais-Modul (mit Optokoppler)
- 3.3 V 1 Kanal relay Modul (mit Optokoppler)

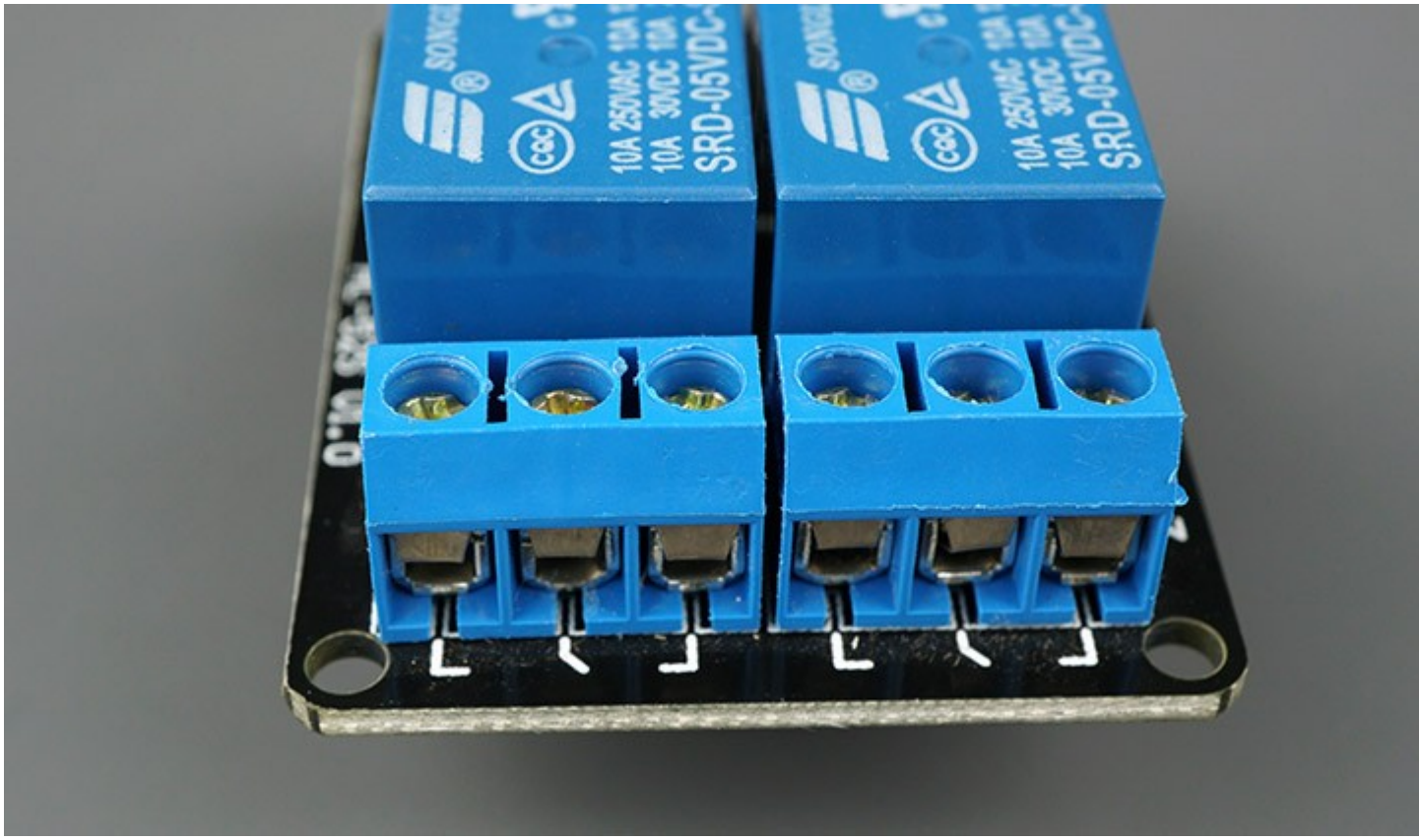
Relais Pinout

Für die demonstration, lassen Sie uns nehmen einen Blick auf die Pinbelegung des 2-Kanal-relais-Modul. Ein relais-Modul mit einer unterschiedlichen Anzahl von Kanälen ähnlich ist.



Auf der linken Seite, es sind zwei Sätze von drei Steckdosen zu verbinden hohen Spannungen, und die Stifte auf der rechten Seite (Niederspannung) Anschluss an den ESP32 GPIOs.

Netzspannung Verbindungen



Das relais-Modul in den obigen Fotos hat zwei Anschlüsse, die jeweils mit drei Steckdosen: common (COM), Normalerweise Geschlossen (NC) und Normal Offen (NO).

- **COM:** schließen Sie den Strom, den Sie Steuern möchten (Netzspannung).

- **NC (Normalerweise Geschlossen):** normalerweise geschlossene Konfiguration wird verwendet, wenn Sie möchten, dass die relais zu werden standardmäßig geschlossen. Die NC-COM-pins angeschlossen sind, d.h. der Strom fließt, es sei denn, Sie senden ein signal von der ESP32 an die relais-Modul, um die Schaltung zu öffnen und zu stoppen, um den Stromfluss.
- **NO (Normal Open):** normalerweise offene Konfiguration funktioniert anders herum: es gibt keinen Zusammenhang zwischen der NO-und COM-pins, so dass die Schaltung ist gebrochen, es sei denn, Sie senden ein signal von der ESP32 an den Stromkreis schließen.

Control Pins

Die low-Spannung Seite einen Satz von vier Stifte und einen Satz von drei pins. Die erste Gruppe besteht aus VCC und GND zu schalten Sie das Modul ein, und der Eingang 1 (IN1) und Eingang 2 (IN2) zur Steuerung der unteren und oberen relais, beziehungsweise.

Wenn Sie Ihre relais-Modul hat nur einen Kanal haben, müssen Sie nur einen IN-pin. Wenn Sie über vier Kanäle, die Sie haben vier pins, und so auf.

Das signal, das Sie senden an die IN der Stifte, der bestimmt, ob das relais aktiv ist oder nicht. Das relais wird ausgelöst, wenn der Eingang geht unterhalb von etwa 2V. Dies bedeutet, dass müssen Sie die folgenden Szenarien:

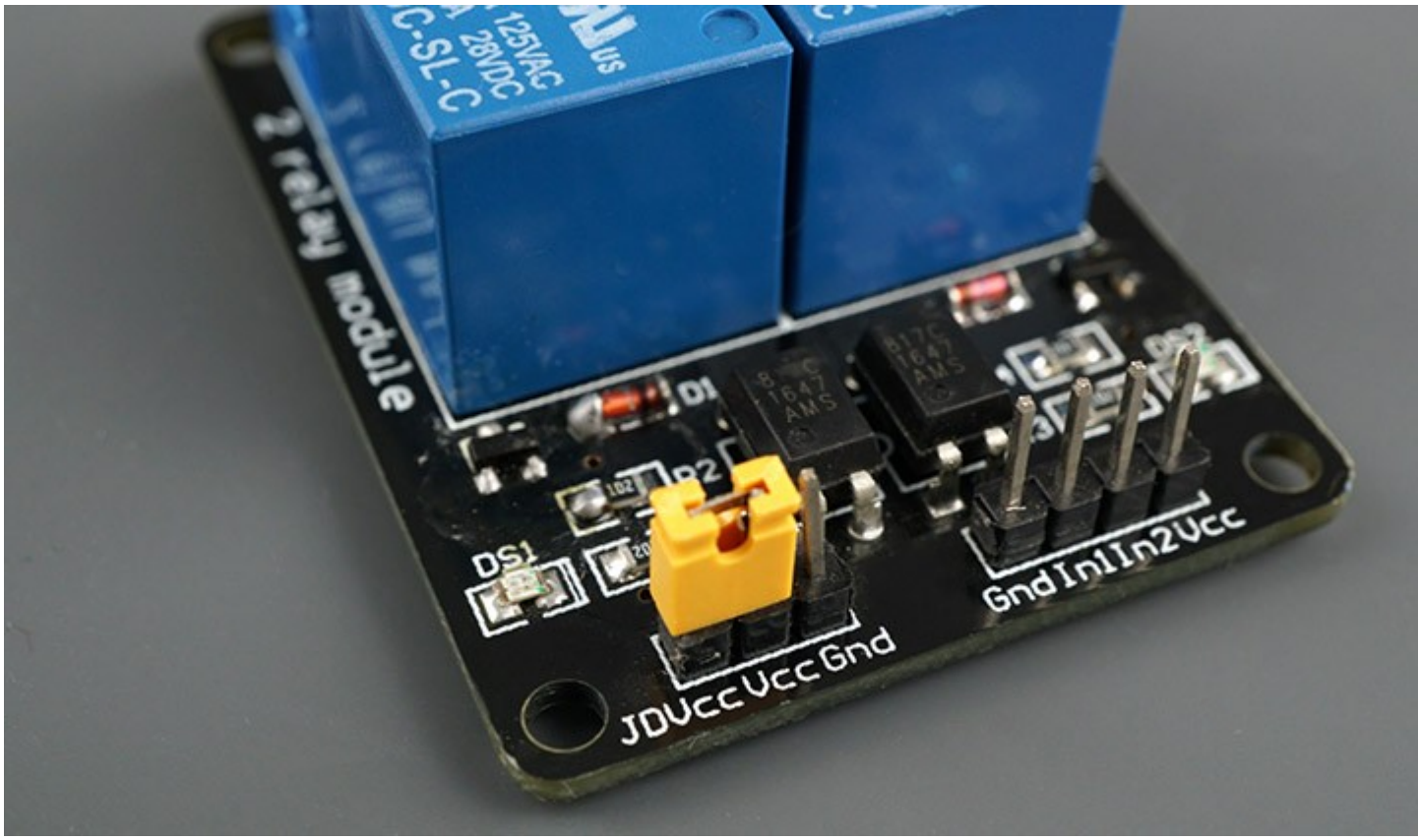
- **Normalerweise Geschlossene Konfiguration (NC)** :
    - HIGH-signal – Strom fließt
    - LOW-signal – Strom **nicht** fließt
- **Normalerweise Offene Konfiguration (KEINE)** :
    - HIGH-signal – Strom **nicht** fließt

- LOW-signal – Strom im fließt

Sollten Sie eine normalerweise geschlossene Konfiguration, wenn der Strom zu fließen die meisten der Zeit, und Sie möchten nur aufhören, es gelegentlich.

Verwenden Sie einen Schließer-Konfiguration, wenn Sie möchten, dass der Strom fließen, der gelegentlich (für Beispiel, schalten Sie die Lampe gelegentlich).

Stromversorgung Auswahl



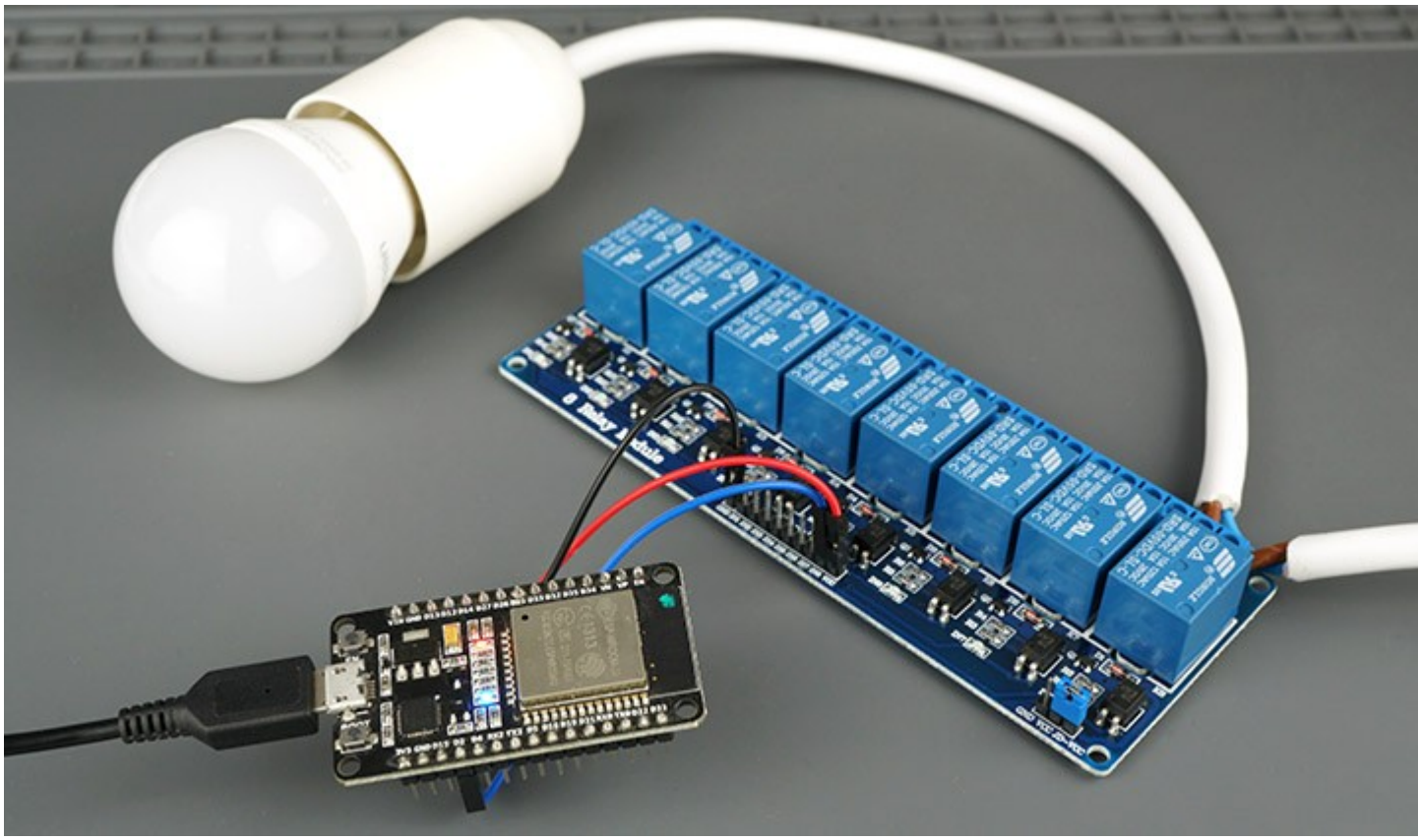
Die zweite Reihe von Stiften besteht, GND , VCC und JD-VCC - pins. Der JD-VCC pin Befugnisse der Elektromagnet des relais. Beachten Sie, dass das Modul verfügt über eine jumper-Kappe Anschluss VCC und JD-VCC-pins; der eine, der hier gezeigt wird, ist gelb, aber Sie können eine andere Farbe haben.

Mit die jumper Kappe auf die VCC und JD-VCC - pins verbunden sind. Das bedeutet, dass das relais Elektromagnet ist direkt mit dem ESP32 power pin, so dass das relais-Modul und der ESP32-schaltungen, die nicht physisch voneinander isoliert.

Ohne die jumper Kappe, die Sie brauchen, um bieten eine unabhängige Stromversorgung zur Stromversorgung der relais Elektromagnet durch das JD-VCC pin. Diese Konfiguration physisch trennt die relais von der ESP32 mit dem Modul integrierte Optokoppler, die verhindert, dass Schäden an den ESP32 in Fall von elektrische spikes.

Ansteuerung eines Relais Modul mit die ESP32 – Arduino-Skizze

Der code zum Steuern eines relais mit der ESP32 ist so einfach wie die Steuerung eine LED oder eine andere Ausgabe. In diesem Beispiel verwenden wir eine normalerweise offene Konfiguration, die wir brauchen, um senden Sie ein LOW-signal, um lassen Sie den Stromfluss, und ein HIGH-signal zum stoppen der aktuellen flow.



Der folgende code wird Licht up Ihre Lampe für 10 Sekunden und schalten Sie ihn für weitere 10 Sekunden.

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-relay-
module-ac-web-server/

  The above copyright notice and this permission notice shall be included in all
  copies or substantial portions of the Software.
*********/

const int relay = 26;

void setup() {
  Serial.begin(115200);
  pinMode(relay, OUTPUT);
}

void loop() {
  // Normally Open configuration, send LOW signal to let current flow
  // (if you're usong Normally Closed configuration send HIGH signal)
  digitalWrite(relay, LOW);
  Serial.println("Current Flowing");
```

```
  delay(5000);

  // Normally Open configuration, send HIGH signal stop current flow
  // (if you're usong Normally Closed configuration send LOW signal)
  digitalWrite(relay, HIGH);
  Serial.println("Current not Flowing");
  delay(5000);
}
```

Wie der Code Funktioniert

Definieren Sie die pin der relais pin verbunden ist.

```
const int relay = 26;
```

In der setup() , definieren Sie das relais als Ausgang.

```
pinMode(relay, OUTPUT);
```

In der loop() , senden Sie ein LOW - signal lassen Sie die Strom fließen und Licht bis die Lampe.

```
digitalWrite(relay, LOW);
```

Wenn Sie eine normalerweise geschlossene Konfiguration, senden Sie eine HIGH - signal leuchtet die Lampe. Warten Sie dann 5 Sekunden.

```
delay(5000);
```

Stoppen Sie den Stromfluss durch senden einer HIGH - signal an den relais-pin. Wenn Sie eine normalerweise geschlossene Konfiguration, senden Sie ein LOW - signal zu stoppen, um den Stromfluss.

```
digitalWrite(relay, HIGH);
```

Steuerung Mehrerer Relais mit ESP32 Web Server

In diesem Abschnitt haben wir ein web-server-Beispiel, können Sie Steuern, wie viele relais, wie Sie wollen, die via web-server, ob Sie so konfiguriert sind, als normal geöffnet oder normal geschlossen. Sie müssen nur ein paar Zeilen code zum definieren der Anzahl der relais Sie Steuern möchten, und der pin-Belegung.

Bauen Sie diese web-server, verwenden wir den ESPAsyncWebServer Bibliothek .

Die Installation des ESPAsyncWebServer Bibliothek

Befolgen Sie die nächsten Schritte zum installieren des ESPAsyncWebServer Bibliothek:

1. Klicken Sie hier, um den download des ESPAsyncWebServer Bibliothek . Sie sollten eine .zip-Ordner in Ihrem Downloads-Ordner
2. Entpacken Sie die .zip-Ordner und Sie sollten *ESPAsyncWebServer-master* - Ordner
3. Benennen Sie Ihren Ordner aus ESPAsyncWebServer-master zu *ESPAsyncWebServer*
4. Bewegen Sie den *ESPAsyncWebServer* Ordner, um Ihre Arduino IDE-installation Ordner Bibliotheken

Alternativ dazu können Sie in Ihrem Arduino-IDE, Sie können gehen Sie auf **Sketch** > **Include Library** > **Add .ZIP library...** und wählen Sie in der Bibliothek haben Sie gerade heruntergeladen haben.

Installieren Sie die Async-TCP-Bibliothek für ESP32

Die ESPAsyncWebServer Bibliothek erfordert die AsyncTCP Bibliothek zu arbeiten. Befolgen Sie die nächsten Schritte zum installieren der Bibliothek:

1. Klicken Sie hier zum herunterladen der AsyncTCP Bibliothek . Sie sollten eine .zip-Ordner in Ihrem Downloads-Ordner
2. Entpacken Sie die .zip-Ordner und Sie sollten *AsyncTCP-master* - Ordner
3. Benennen Sie Ihren Ordner aus AsyncTCP-master zu *AsyncTCP*
4. Bewegen Sie den *AsyncTCP* Ordner, um Ihre Arduino IDE-installation Ordner Bibliotheken
5. Schließlich, re-öffnen Sie Ihre Arduino IDE

Alternativ dazu können Sie in Ihrem Arduino-IDE, Sie können gehen Sie auf **Sketch** > **Include Library** > **Add .ZIP library...** und wählen Sie in der Bibliothek haben Sie gerade heruntergeladen haben.

Nach der Installation der erforderlichen Bibliotheken, kopieren Sie den folgenden code in Ihre Arduino IDE.

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-relay-
module-ac-web-server/

  The above copyright notice and this permission notice shall be included in all
  copies or substantial portions of the Software.
*********/

// Import required libraries
#include "WiFi.h"
#include "ESPAsyncWebServer.h"

// Set to true to define Relay as Normally Open (NO)
#define RELAY_NO    true

// Set number of relays
#define NUM_RELAYS  5

// Assign each GPIO to a relay
int relayGPIOs[NUM_RELAYS] = {2, 26, 27, 25, 33};

// Replace with your network credentials
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

const char* PARAM_INPUT_1 = "relay";
const char* PARAM_INPUT_2 = "state";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
    html {font-family: Arial; display: inline-block; text-align: center;}
    h2 {font-size: 3.0rem;}
    p {font-size: 3.0rem;}
    body {max-width: 600px; margin:0px auto; padding-bottom: 25px;}
    .switch {position: relative; display: inline-block; width: 120px; height:
68px}
    .switch input {display: none}
    .slider {position: absolute; top: 0; left: 0; right: 0; bottom: 0;
background-color: #ccc; border-radius: 34px}
```

```
    .slider:before {position: absolute; content: ""; height: 52px; width: 52px;
left: 8px; bottom: 8px; background-color: #fff; -webkit-transition: .4s;
transition: .4s; border-radius: 68px}
    input:checked+.slider {background-color: #2196F3}
    input:checked+.slider:before {-webkit-transform: translateX(52px); -ms-
transform: translateX(52px); transform: translateX(52px)}
  </style>
</head>
<body>
  <h2>ESP Web Server</h2>
  %BUTTONPLACEHOLDER%
<script>function toggleCheckbox(element) {
  var xhr = new XMLHttpRequest();
  if(element.checked){ xhr.open("GET", "/update?relay="+element.id+"&state=1",
true); }
  else { xhr.open("GET", "/update?relay="+element.id+"&state=0", true); }
  xhr.send();
}</script>
</body>
</html>
)rawliteral";

// Replaces placeholder with button section in your web page
String processor(const String& var){
  //Serial.println(var);
  if(var == "BUTTONPLACEHOLDER"){
    String buttons ="";
    for(int i=1; i<=NUM_RELAYS; i++){
      String relayStateValue = relayState(i);
      buttons+= "<h4>Relay #" + String(i) + " - GPIO " + relayGPIOs[i-1] +
"</h4><label class=\"switch\"><input type=\"checkbox\"
onchange=\"toggleCheckbox(this)\" id=\"" + String(i) + "\" "+ relayStateValue
+"><span class=\"slider\"></span></label>";
    }
    return buttons;
  }
  return String();
}

String relayState(int numRelay){
  if(RELAY_NO){
    if(digitalRead(relayGPIOs[numRelay-1])){
      return "";
    }
    else {
      return "checked";
    }
  }
  else {
    if(digitalRead(relayGPIOs[numRelay-1])){
      return "checked";
    }
    else {
      return "";
    }
  }
  return "";
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);
```

```
  // Set all relays to off when the program starts - if set to Normally Open
(NO), the relay is off when you set the relay to HIGH
  for(int i=1; i<=NUM_RELAYS; i++){
    pinMode(relayGPIOs[i-1], OUTPUT);
    if(RELAY_NO){
      digitalWrite(relayGPIOs[i-1], HIGH);
    }
    else{
      digitalWrite(relayGPIOs[i-1], LOW);
    }
  }

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

  // Print ESP32 Local IP Address
  Serial.println(WiFi.localIP());

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });

  // Send a GET request to <ESP_IP>/update?
relay=<inputMessage>&state=<inputMessage2>
  server.on("/update", HTTP_GET, [] (AsyncWebServerRequest *request) {
    String inputMessage;
    String inputParam;
    String inputMessage2;
    String inputParam2;
    // GET input1 value on <ESP_IP>/update?relay=<inputMessage>
    if (request->hasParam(PARAM_INPUT_1) & request->hasParam(PARAM_INPUT_2)) {
      inputMessage = request->getParam(PARAM_INPUT_1)->value();
      inputParam = PARAM_INPUT_1;
      inputMessage2 = request->getParam(PARAM_INPUT_2)->value();
      inputParam2 = PARAM_INPUT_2;
      if(RELAY_NO){
        Serial.print("NO ");
        digitalWrite(relayGPIOs[inputMessage.toInt()-1], !
inputMessage2.toInt());
      }
      else{
        Serial.print("NC ");
        digitalWrite(relayGPIOs[inputMessage.toInt()-1], inputMessage2.toInt());
      }
    }
    else {
      inputMessage = "No message sent";
      inputParam = "none";
    }
    Serial.println(inputMessage + inputMessage2);
    request->send(200, "text/plain", "OK");
  });
  // Start server
  server.begin();
}

void loop() {

}
```

Definieren Relay Konfiguration

Ändern Sie die folgenden Variablen, um anzugeben, ob Sie das relais normal offen (NO) oder normal geschlossen (NC) - Konfiguration. Legen Sie die RELAY_NO variable auf true für die Schließer-os set to false for normal geschlossen.

```
#define RELAY_NO true
```

Definieren Sie die Anzahl der Relais (Kanäle)

Sie können die Anzahl der relais Sie Steuern möchten, auf der NUM_RELAYS variable. Zu Demonstrationszwecken setzen wir es auf 5.

```
#define NUM_RELAYS 5
```

Definieren Relais Pin-Belegung

In den folgenden array-Variablen können Sie festlegen, ESP32 GPIOs zu Steuern die relais:

```
int relayGPIOs[NUM_RELAYS] = {2, 26, 27, 25, 33};
```

Die Anzahl der relais-Satz auf die NUM_RELAYS variable muss mit der Anzahl der GPIOs zugewiesen, in der relayGPIOs array.

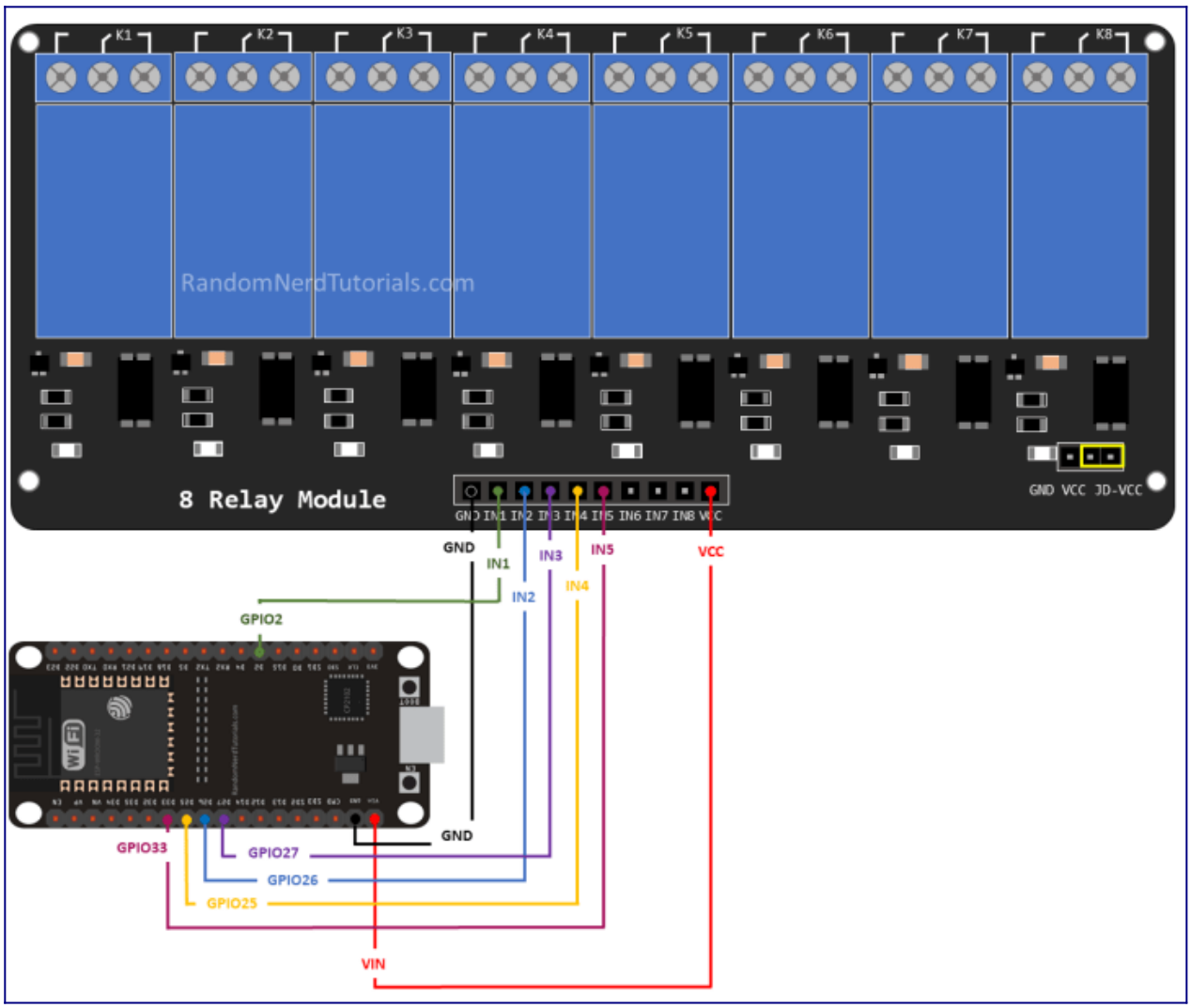
Netzwerk-Anmeldeinformationen

Legen Sie Ihre Netzwerk-Anmeldeinformationen in die folgenden Variablen.

```
const char* ssid     = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

Verkabelung 8-Kanal-Relay, ESP32

Für Testzwecke, wir sind controlling-5-relais-Kanäle. Draht-der ESP32 an die relais-Modul wie gezeigt in den nächsten schematische Diagramm.



Demonstration

Nachdem die notwendigen änderungen vornehmen, laden Sie den code auf Ihrer ESP32.

Öffnen Sie den Seriellen Monitor mit einer Baudrate von 115200 und drücken Sie die ESP32 EN " - button, um Ihre IP-Adresse.

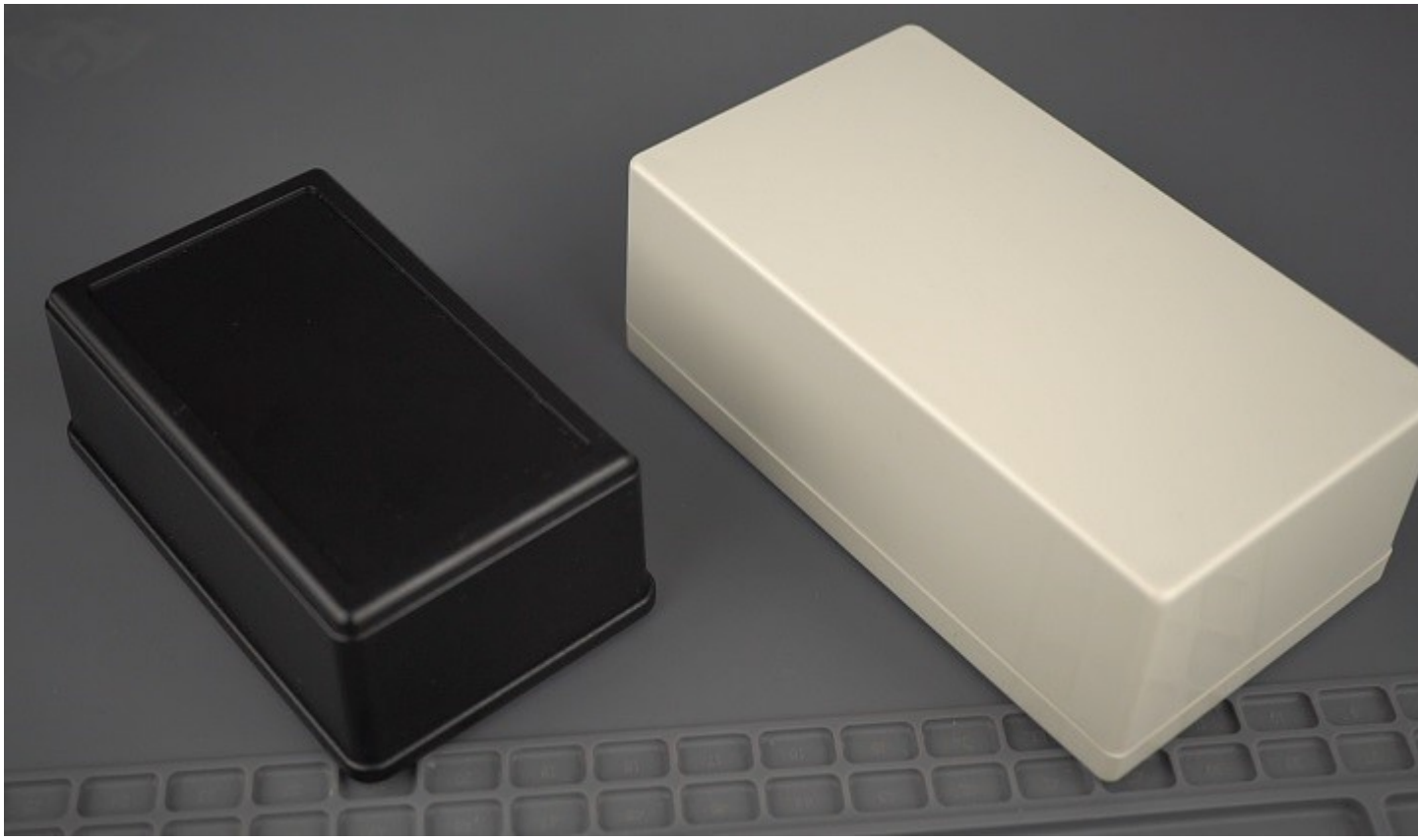
Öffnen Sie dann einen browser in Ihrem lokalen Netzwerk und geben Sie den ESP32 IP-Adresse, um Zugriff auf den web-server.

Sie sollten etwas bekommen, das wie folgt, mit so vielen Tasten wie der Anzahl der relais, die Sie haben, die im code definiert.

Jetzt können Sie über die Schaltflächen Steuern Sie Ihre relais aus der Ferne mit Ihrem smartphone.

Gehäuse für Sicherheit

Für ein abschließendes Projekt, stellen Sie sicher, dass Sie Ihre relay-Modul und ESP in einem Gehäuse zu vermeiden jede AC-pins ausgesetzt.



Zusammenfassung

Mit einem relais mit der ESP32 ist eine großartige Möglichkeit, um control AC Haushaltsgeräte aus der Ferne. Lesen Sie auch unsere anderen Führer zur Steuerung eines Relais Modul mit **ESP8266** .

Ansteuerung eines relais mit der ESP32 ist so einfach, die Steuerung jeder anderen Ausgang, Sie müssen nur senden Sie HIGH-und LOW-Signale, wie Sie tun würden, um die Kontrolle einer LED.

Sie können unsere web-server Beispiele, die control-Ausgänge zur Steuerung von relais. Sie brauchen nur zu zahlen Aufmerksamkeit auf die Konfiguration, die Sie gerade verwenden. In Fall Sie sind mit einem Schließer-Konfiguration, die relais arbeitet mit invertierter Logik. Sie verwenden können die folgenden web server-Beispiele zur Steuerung Ihres relais:

- ESP32 Web Server – Arduino IDE
- ESP32 Webserver mit SPIFFS (control outputs)
- ESP32/ESP8266 MicroPython Web-Server – Control-Ausgänge

Erfahren Sie mehr über die ESP32 mit unseren Ressourcen:

- Lernen ESP32 mit der Arduino IDE (Videokurs + eBook)
- MicroPython Programmierung mit dem ESP32 und ESP8266
- Mehr ESP32 Ressourcen...

Vielen Dank für das Lesen.