SP32-Relais-Modul – Steuerung AC Appliances (Web-Server)

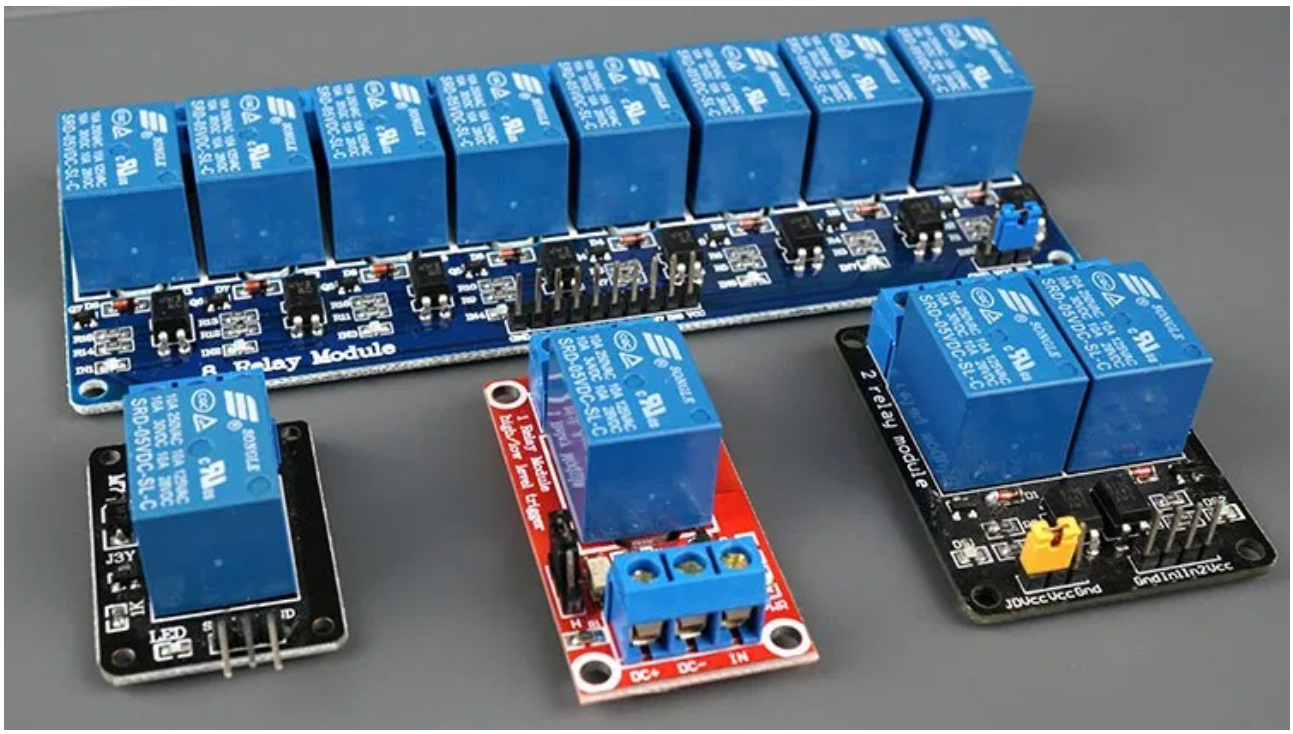
Ein relais mit dem ESP32 ist eine großartige Möglichkeit, um control AC Haushaltsgeräte aus der Ferne. In diesem tutorial wird erklärt, wie man eine relais-Modul mit dem ESP32. Wir werden einen Blick auf, wie ein relais-Modul arbeiten, so schließen Sie das relais an den ESP32 und bauen ein web-server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

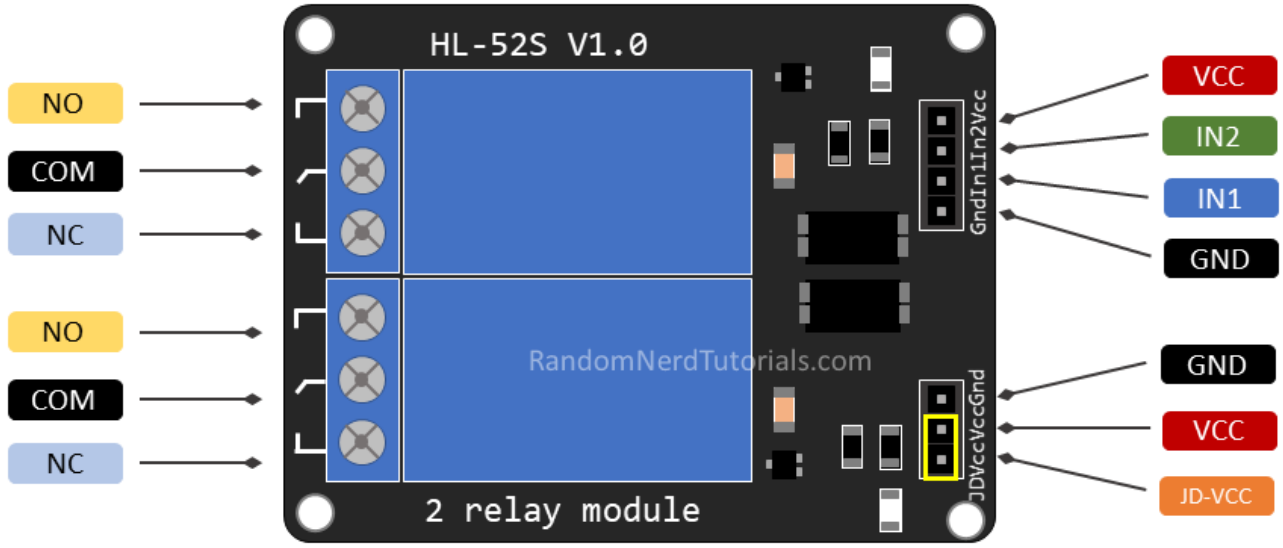
Zusätzlich, einige kommen mit built-in-Optokoppler, die fügen Sie eine zusätzliche "Schicht" der Schutz, optisch isoliert der ESP32 von der relais-Schaltung.

Holen Sie sich ein relais-Modul :

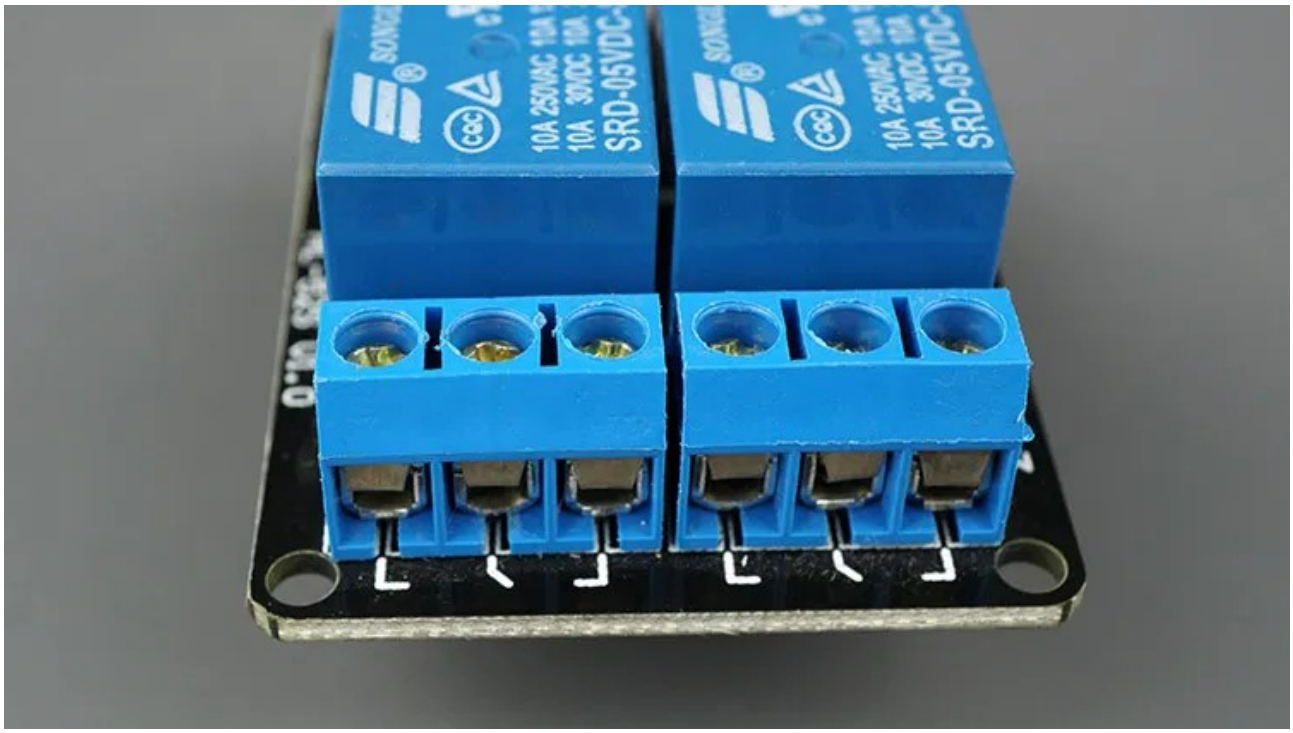
- 5V 2-Kanal relais-Modul (mit Optokoppler)
- 5V 1-Kanal-relais-Modul (mit Optokoppler)
- 5V 8-Kanal relais-Modul (mit Optokoppler)
- 5V 16-Kanal-relais-Modul (mit Optokoppler)
- 3.3 V 1 Kanal relay Modul (mit Optokoppler)

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.



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.

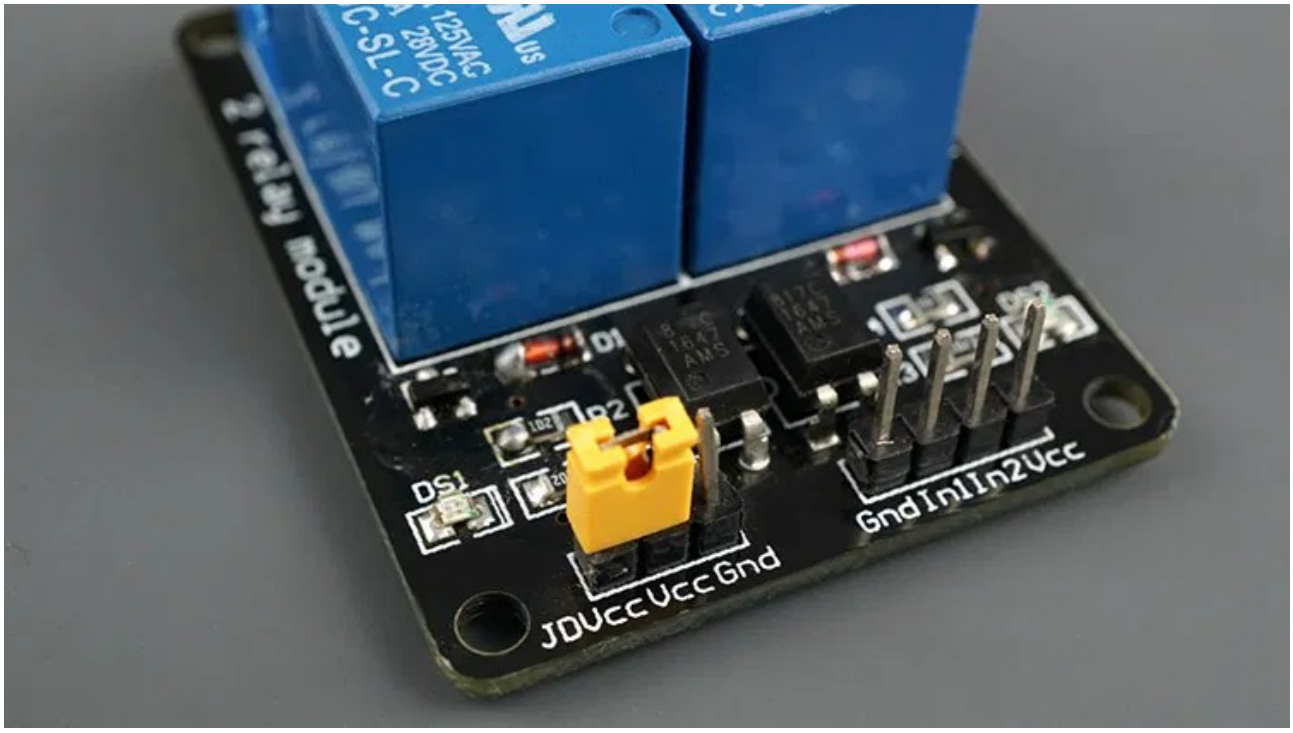
Das signal, das Sie senden an die IN der Stifte, der bestimmt, ob das relais aktiv ist oder nicht. Das relais wird ausgelöst, wenn der Eingang geht unterhalb von etwa 2V. Dies bedeutet, dass müssen Sie die folgenden Szenarien:

- **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.

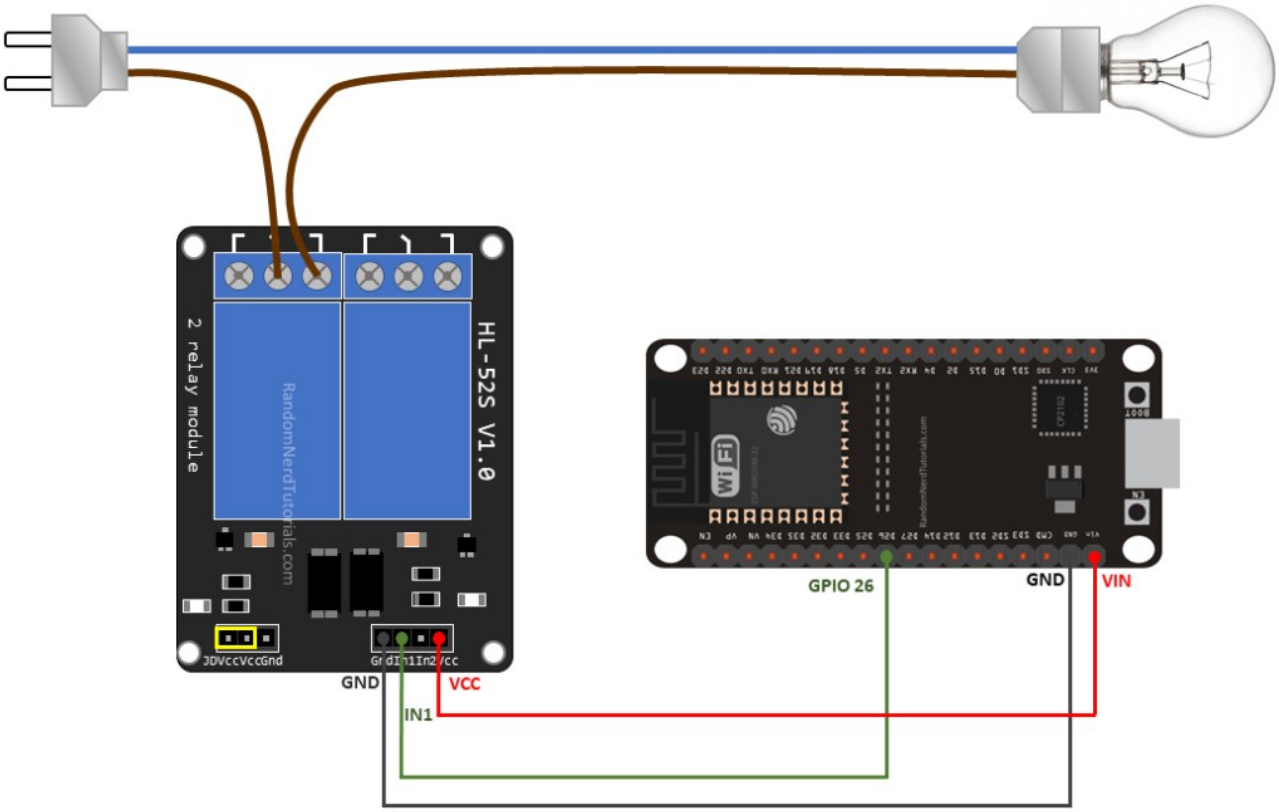
Ohne die jumper Kappe, die Sie brauchen, um bieten eine unabhängige Stromversorgung zur Stromversorgung der relais Elektromagnet durch das JD-VCC pin. Diese Konfiguration physisch trennt die relais von der ESP32 mit dem Modul integrierte Optokoppler, die verhindert, dass Schäden an den ESP32 in Fall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

Verdrahtung einer Relais Modul, um den ESP32

Schließen Sie das relais-Modul des ESP32, wie in der folgenden Abbildung gezeigt. Das Diagramm zeigt die Verdrahtung für eine 2-Kanal-relais-Modul, Verkabelung eine unterschiedliche Anzahl von Kanälen ähnlich ist.

Warnung: in diesem Beispiel befassen wir uns mit der Netzspannung. Missbrauch können Ergebnis in schweren Verletzungen. Wenn Sie nicht vertraut sind mit der Netzspannung bitten Sie jemanden, Ihnen zu helfen. Während der Programmierung des ESP oder Verdrahtung der Schaltung, stellen Sie sicher, alles ist getrennt von der Netzspannung.

Alternativ können Sie die Verwendung einer 12V Stromquelle zu kontrollieren, 12V Geräte.

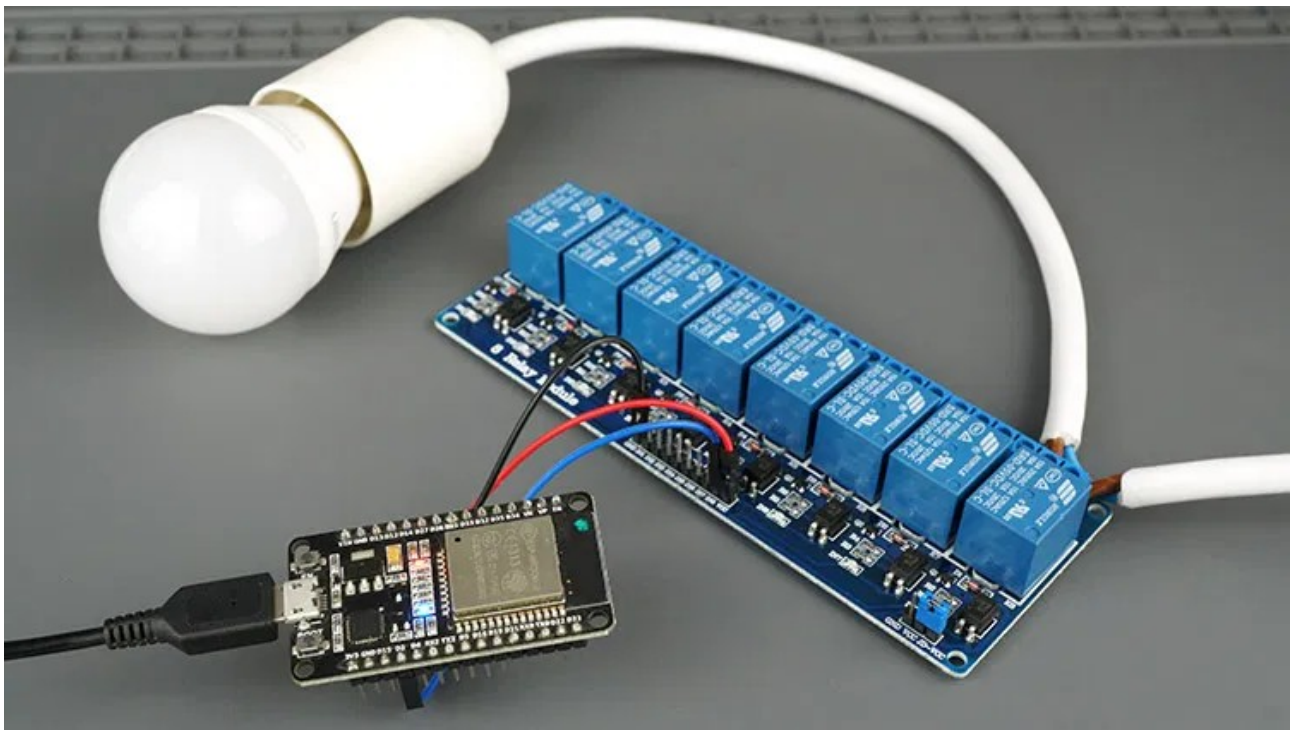


In diesem Beispiel sind wir Steuern eine Lampe. Wir wollen einfach nur, um Licht die Lampe gelegentlich, so ist es besser, verwenden Sie einen Schließer-Konfiguration.

Verbinden wir die IN1-pin GPIO 26 , können Sie mit jedem anderen geeigneten GPIO. Siehe ESP32 GPIO Referenz Guide .

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.

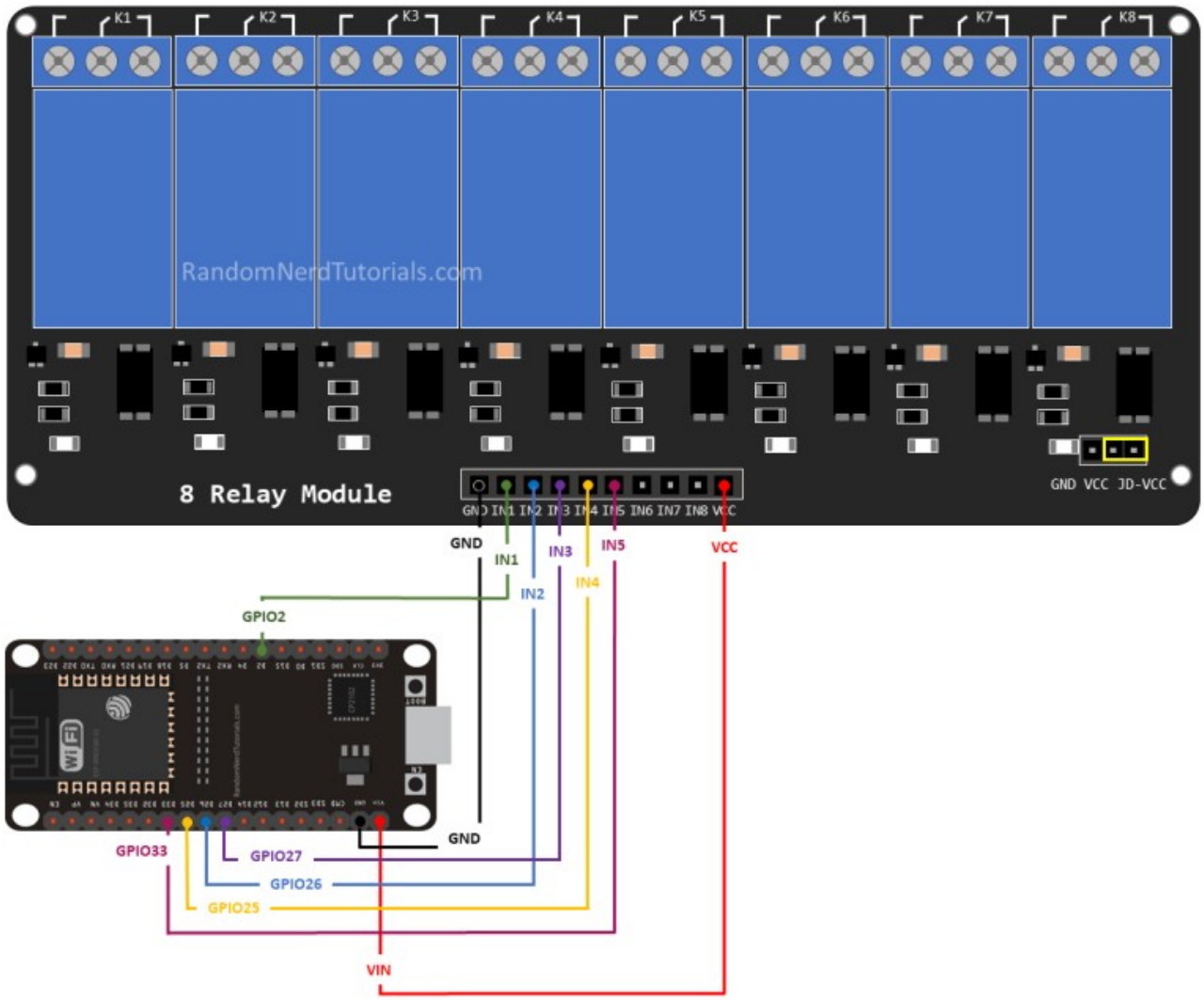
Netzwerk-Anmeldeinformationen

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.

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

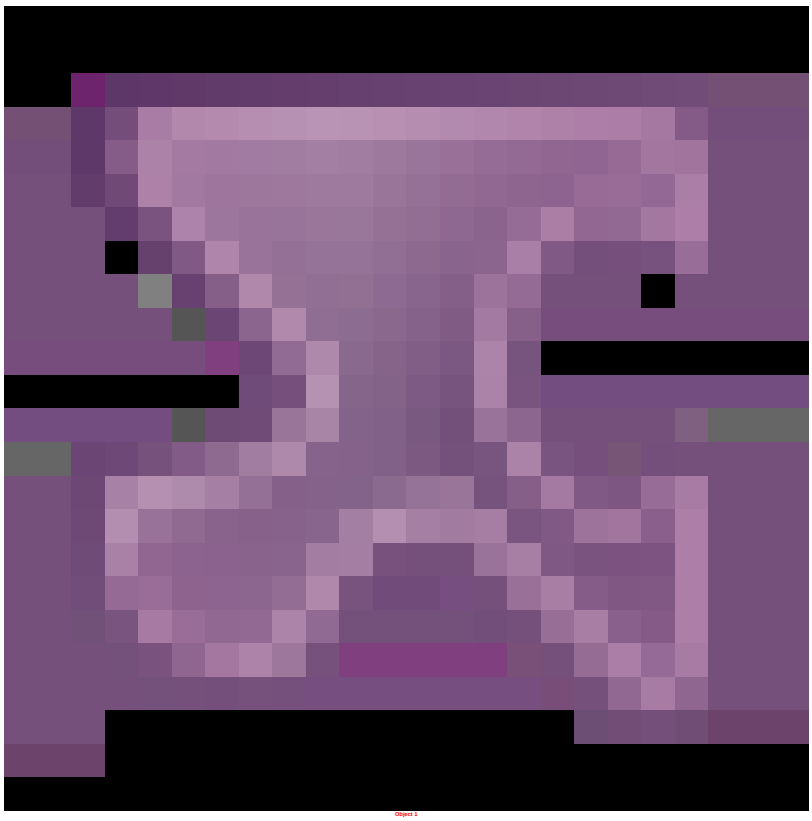
ESP32-Relais-Modul – Steuerung AC Appliances (Web-Server)

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).

Erfahren Sie, wie zu kontrollieren relais Modul mit ESP8266 board: **Leitfaden für ESP8266 Relais Modul** – Steuerung AC Appliances + Web-Server-Beispiel .

Sehen Sie sich das Video-Tutorial

Sehen Sie sich das folgende video-tutorial an oder Lesen Sie diese Seite für die schriftliche Anleitung und alle Ressourcen.

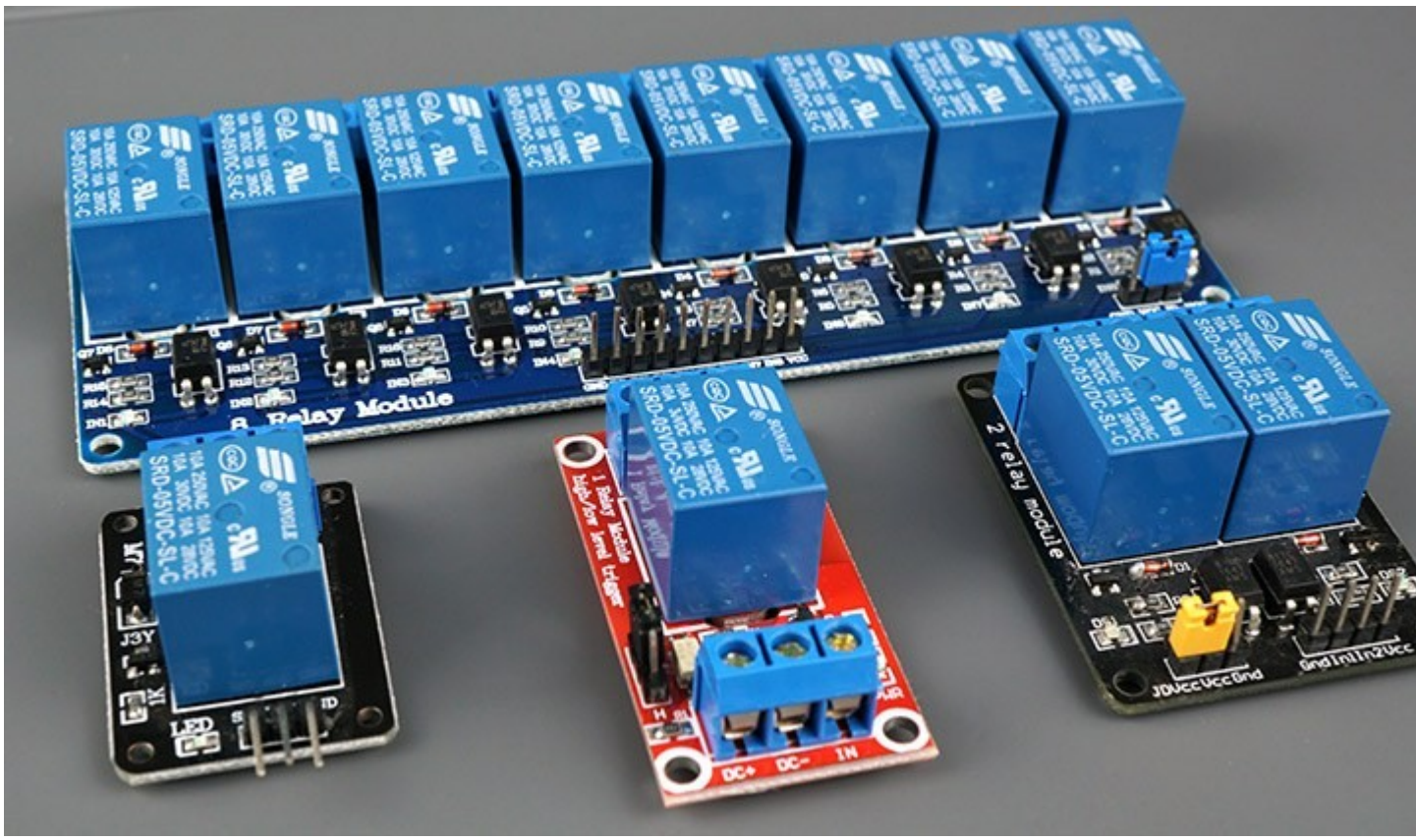


Die Einführung Von Relais

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.

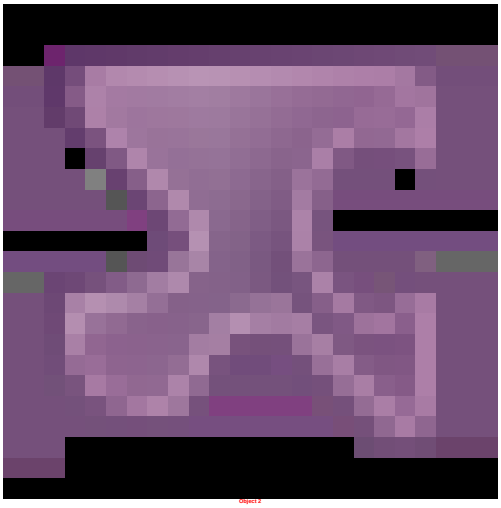


Es gibt relais-Module, deren Elektromagnet mit Strom versorgt werden kann durch die 5V und 3,3 V. Beide können verwendet werden mit die ESP32 – Sie können entweder verwenden Sie den VIN-pin (liefert 5 V) oder 3,3 V pin.

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

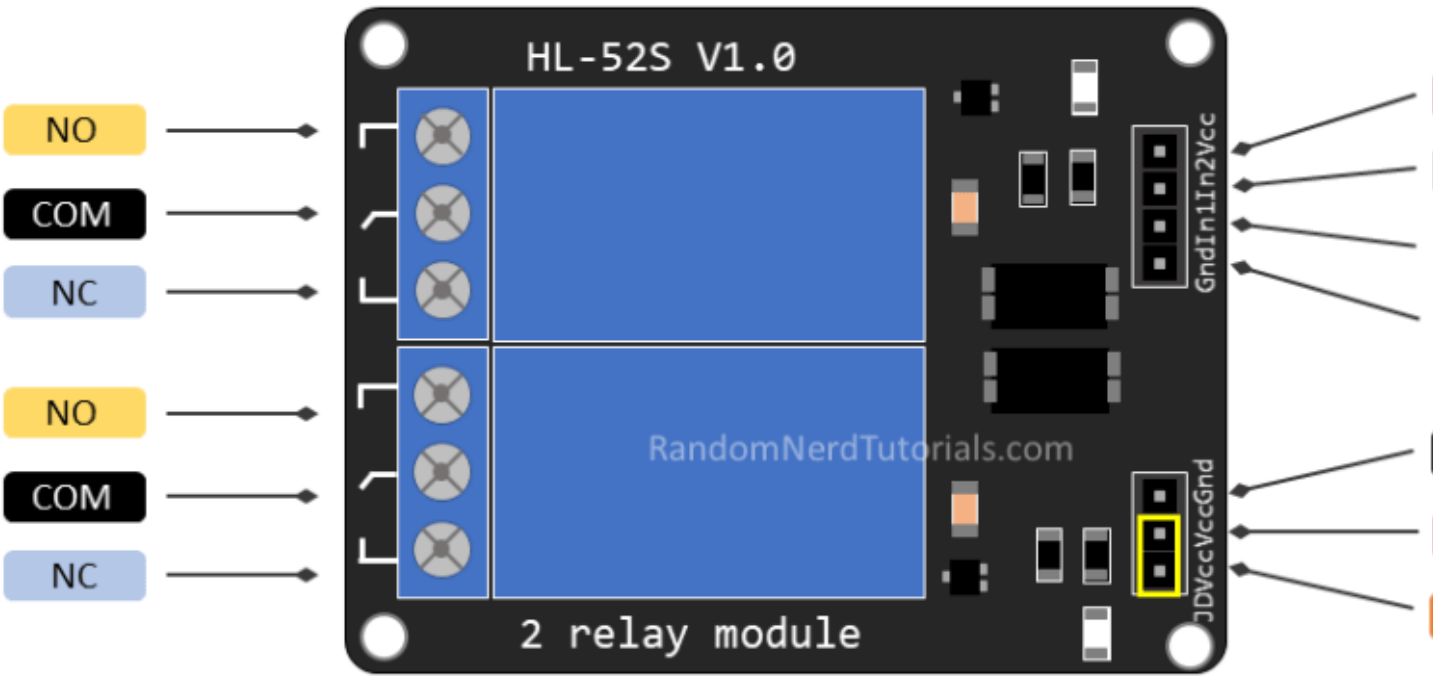
Holen Sie sich ein relais-Modul :

- 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)



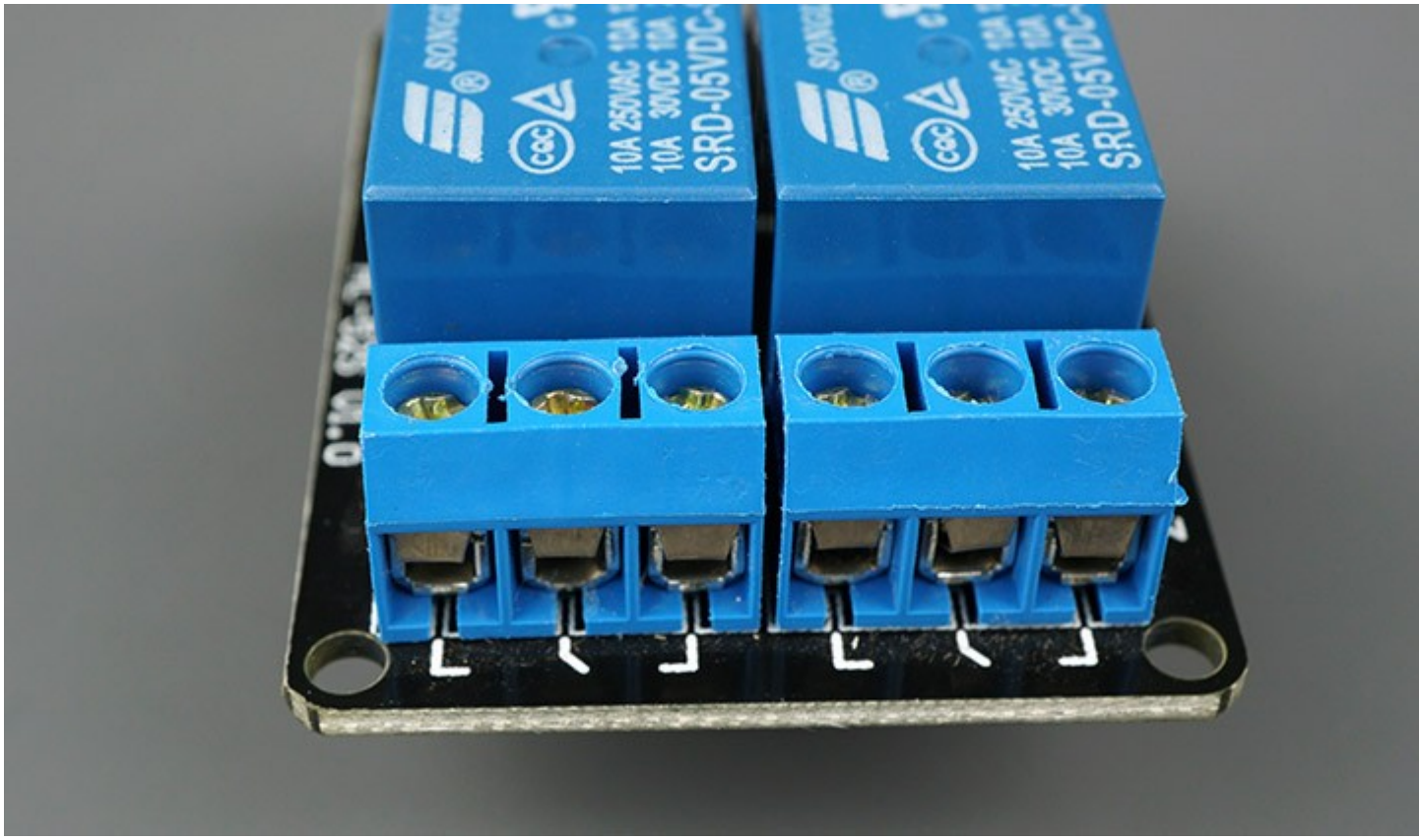
Relais Pinout

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.

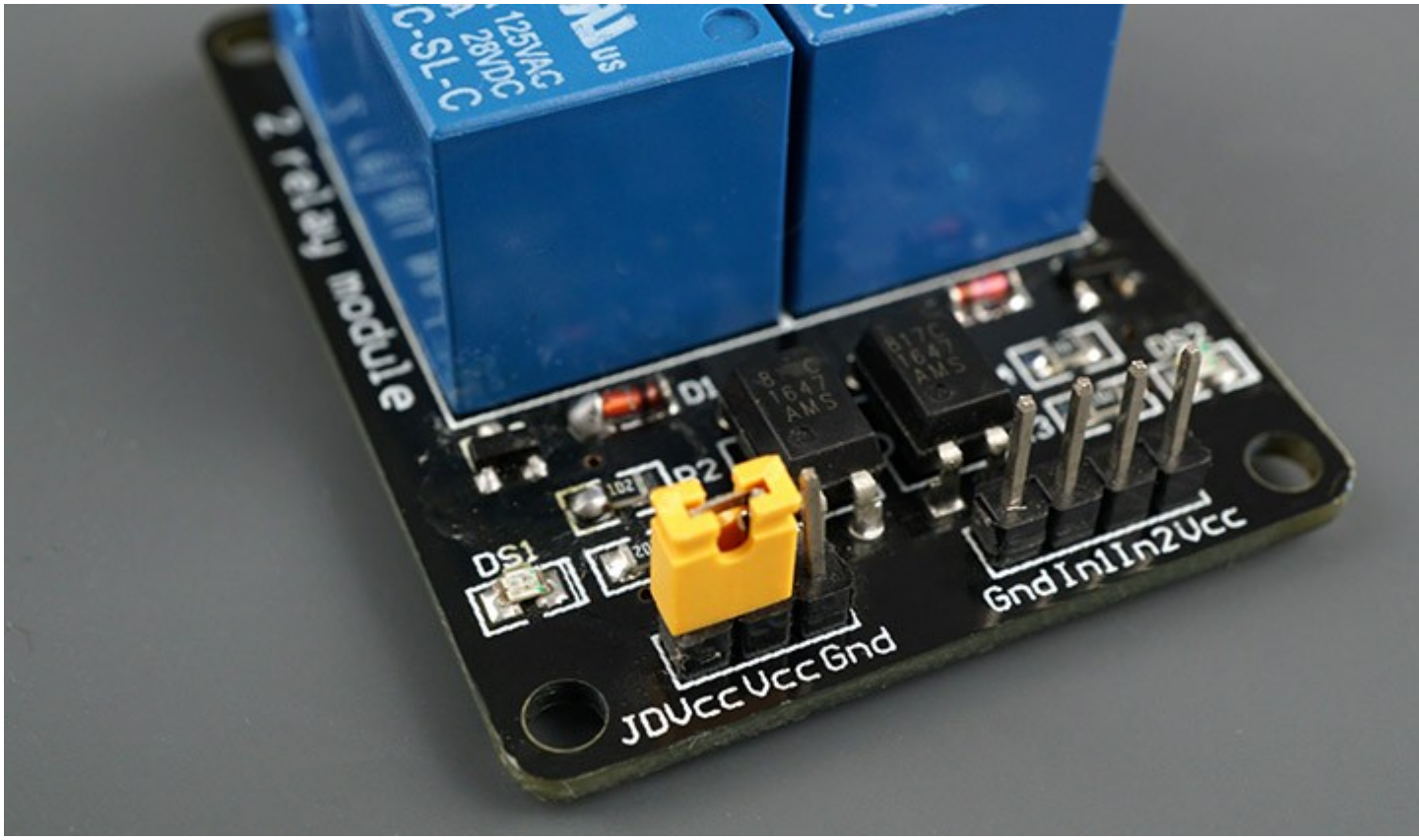
Das signal, das Sie senden an die IN der Stifte, der bestimmt, ob das relais aktiv ist oder nicht. Das relais wird ausgelöst, wenn der Eingang geht unterhalb von etwa 2V. Dies bedeutet, dass müssen Sie die folgenden Szenarien:

- **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.

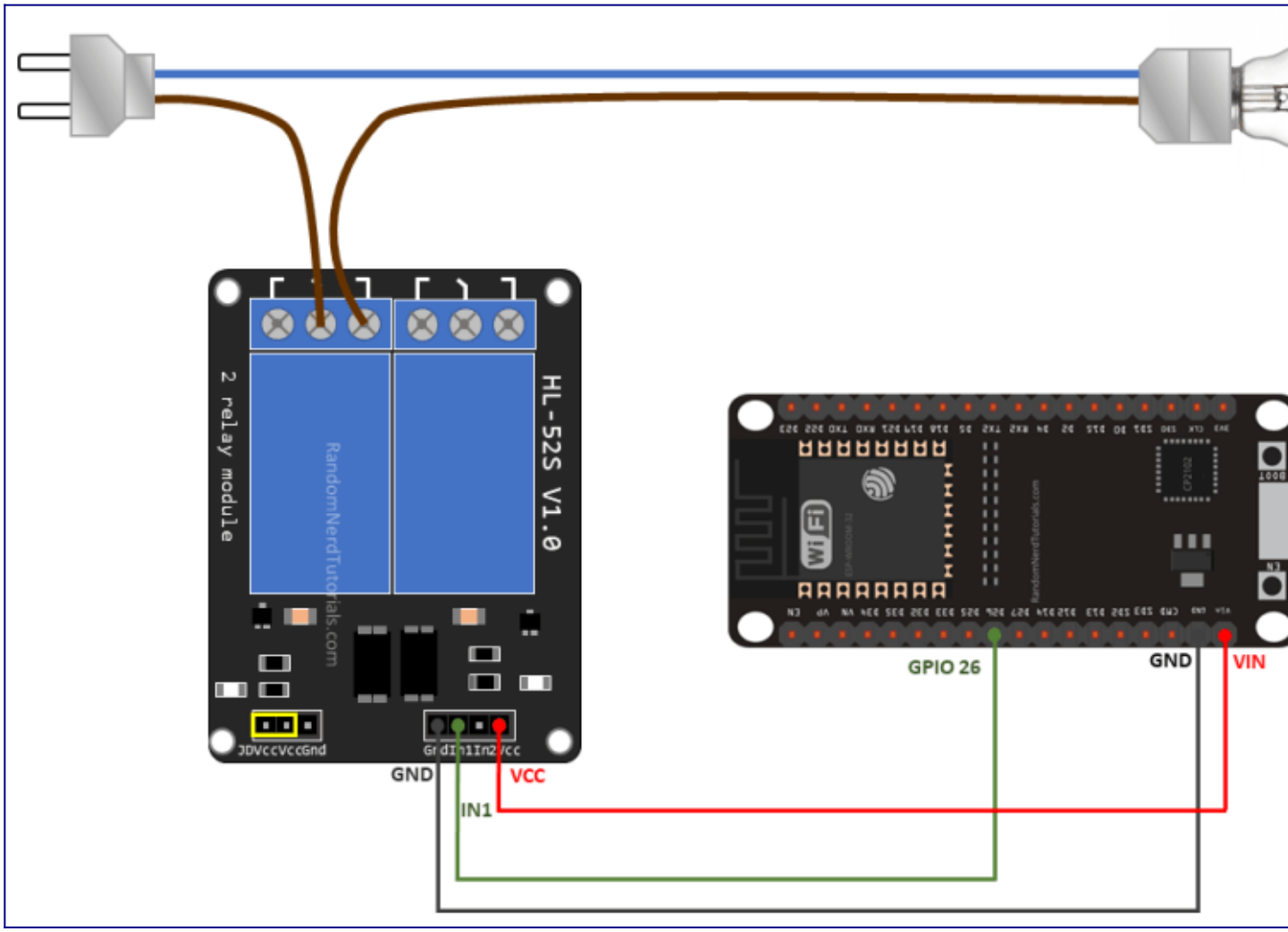
Verdrahtung einer Relais Modul, um den ESP32

Schließen Sie das relais-Modul des ESP32, wie in der folgenden Abbildung gezeigt. Das Diagramm zeigt die Verdrahtung für eine 2-Kanal-relais-Modul, Verkabelung eine unterschiedliche Anzahl von Kanälen ähnlich ist.

Warnung: in diesem Beispiel befassen wir uns mit der Netzspannung. Missbrauch können Ergebnis in schweren Verletzungen. Wenn Sie nicht vertraut sind mit der Netzspannung bitten Sie

jemanden, Ihnen zu helfen. Während der Programmierung des ESP oder Verdrahtung der Schaltung, stellen Sie sicher, alles ist getrennt von der Netzspannung.

Alternativ können Sie die Verwendung einer 12V Stromquelle zu kontrollieren, 12V Geräte.

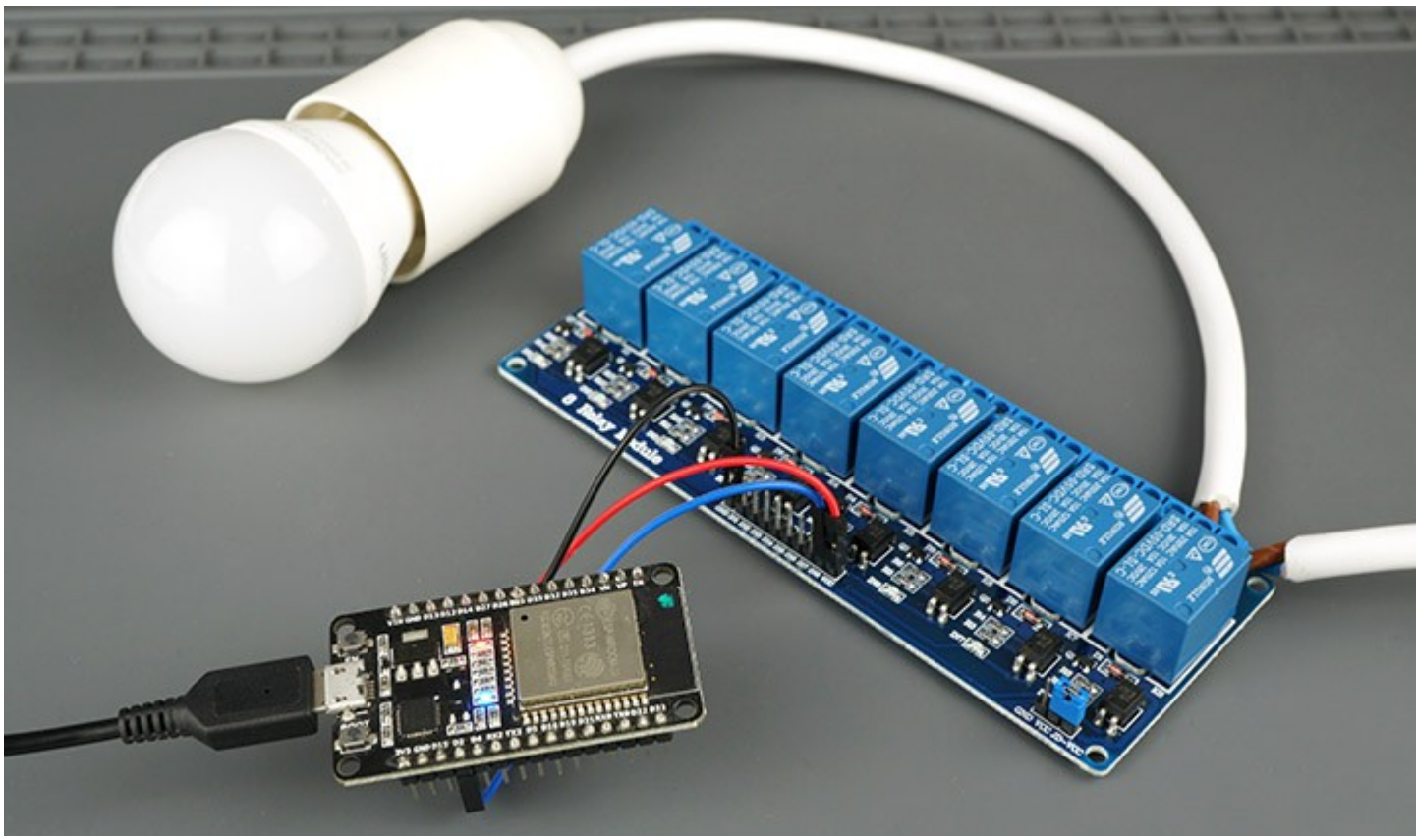


In diesem Beispiel sind wir Steuern eine Lampe. Wir wollen einfach nur, um Licht die Lampe gelegentlich, so ist es besser, verwenden Sie einen Schließer-Konfiguration.

Verbinden wir die IN1-pin GPIO 26 , können Sie mit jedem anderen geeigneten GPIO. Siehe ESP32 GPIO Referenz Guide .

Ansteuerung eines Relais Modul mit die ESP32 – Arduino-Skizze

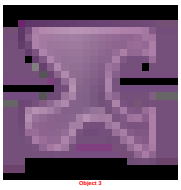
Der code zum Steuern eines relais mit der ESP32 ist so einfach wie die Steuerung eine LED oder eine andere Ausgabe. In diesem Beispiel verwenden wir eine normalerweise offene Konfiguration, die wir brauchen, um senden Sie ein LOW-signal, um lassen Sie den Stromfluss, und ein HIGH-signal zum stoppen der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



Anzeigen von raw-code

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}
```

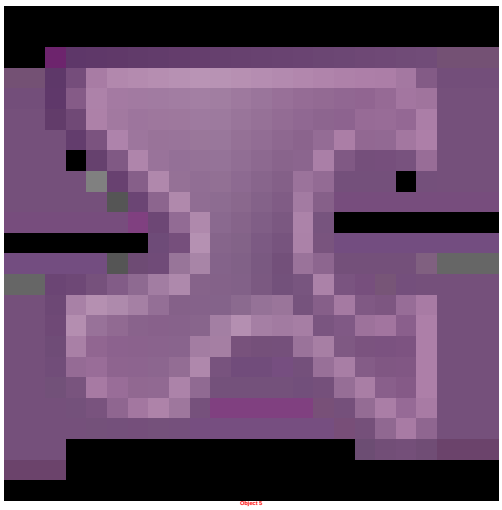
```
    .switch input {display: none}

    .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() {

}
```



[Anzeigen von raw-code]

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.

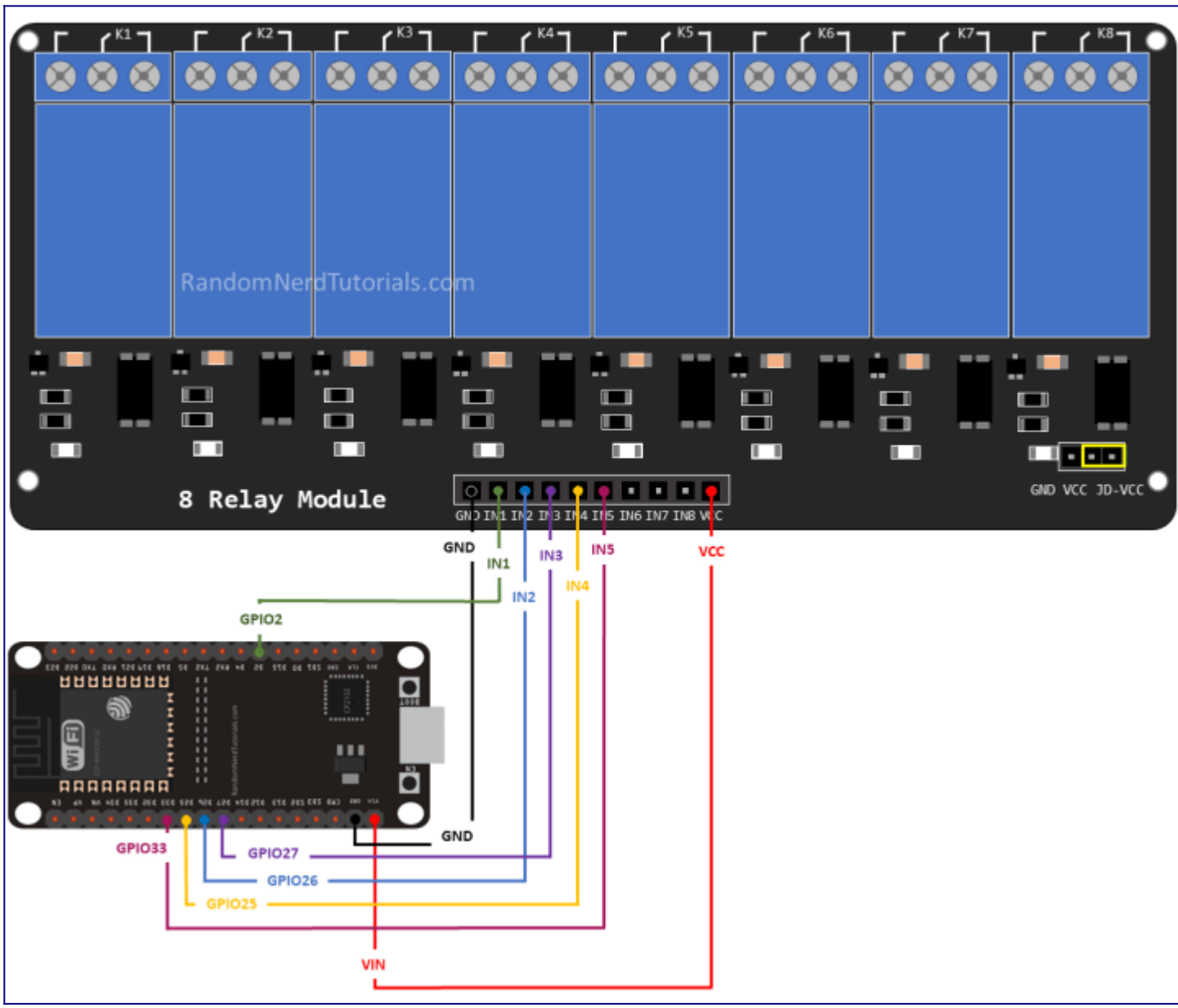
Netzwerk-Anmeldeinformationen

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.



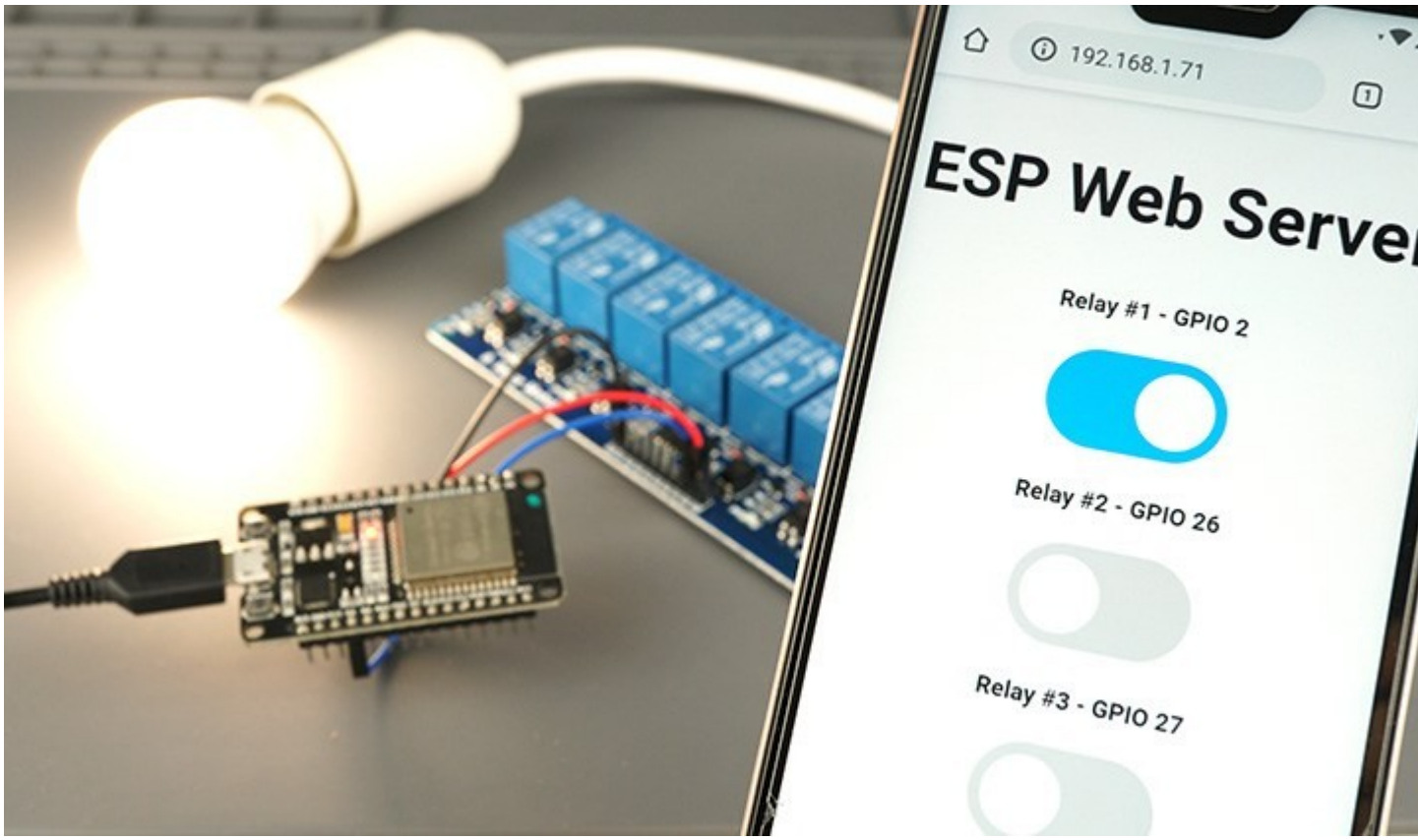
Demonstration

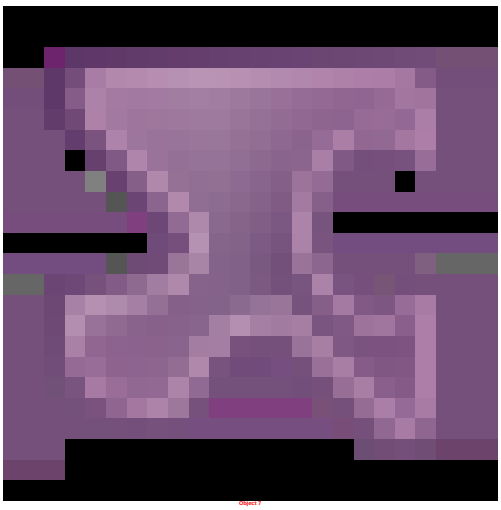
Nachdem die notwendigen änderungen vornehmen, laden Sie den code auf Ihrer ESP32.

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.

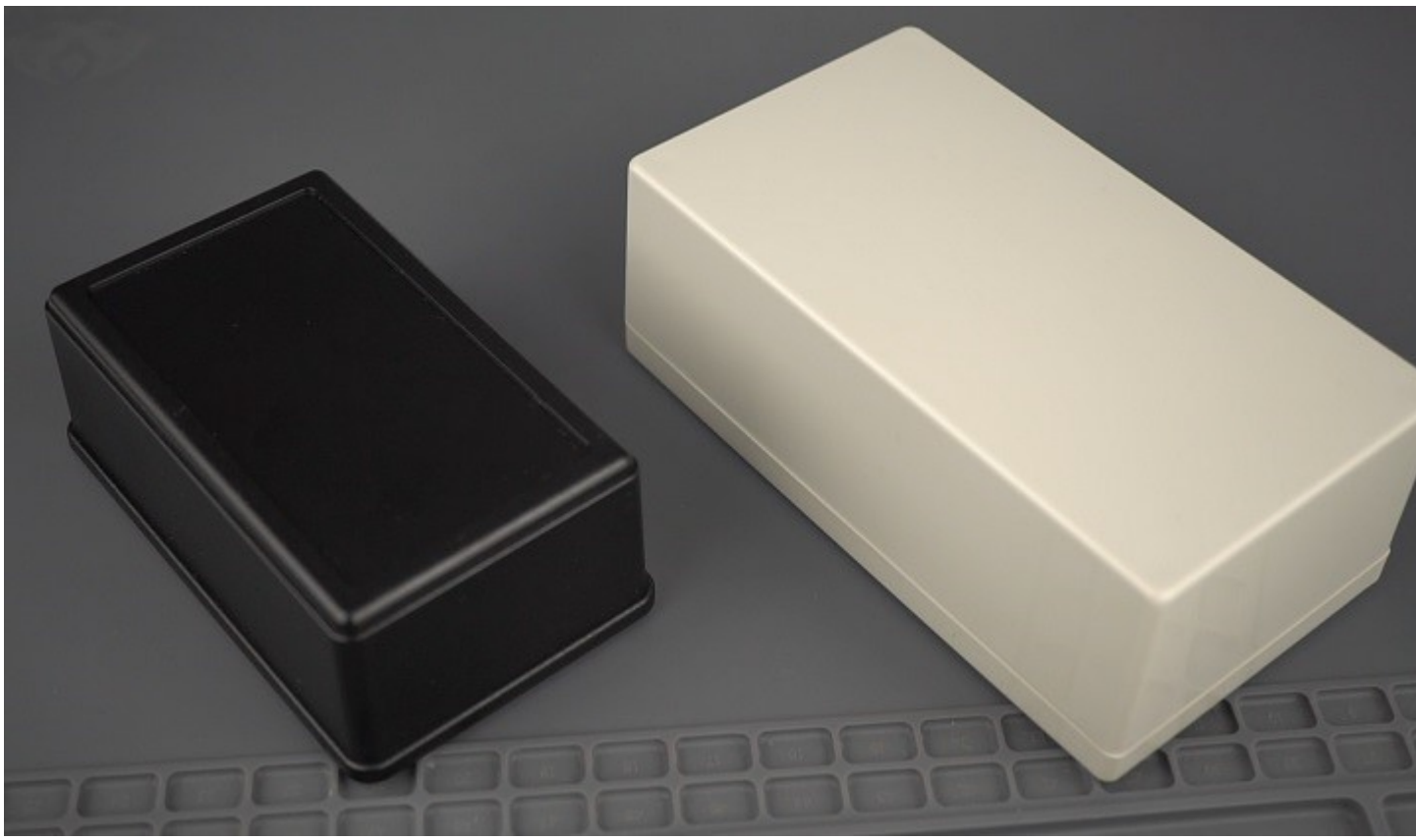




Jetzt können Sie über die Schaltflächen Steuern Sie Ihre relais aus der Ferne mit Ihrem smartphone.

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

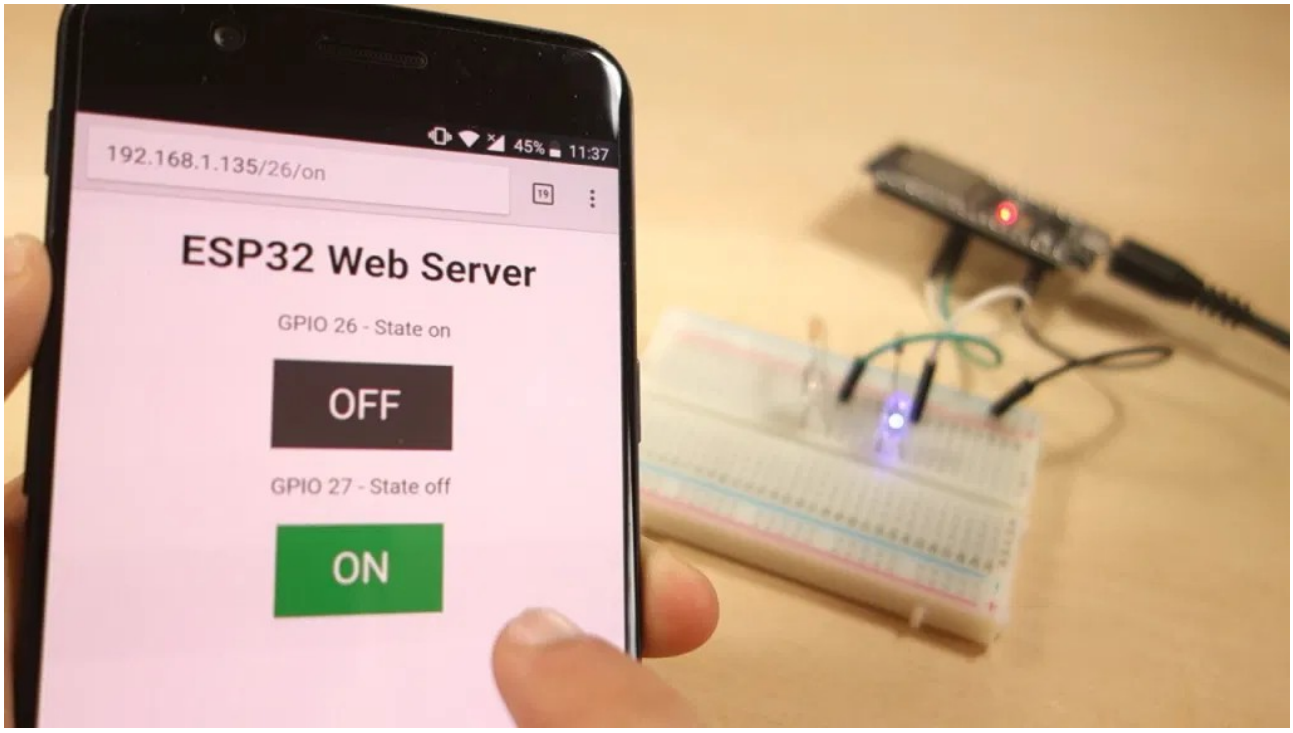
Erfahren Sie mehr über die ESP32 mit unseren Ressourcen:

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

Vielen Dank für das Lesen.

ESP32 Web Server – Arduino IDE

In diesem Projekt erstellen Sie eine eigenständige web-server mit einem ESP32 steuert outputs (two LEDs) using the Arduino IDE programming environment. Der web-server ist mobile responsive und zugegriffen werden kann mit jedem Gerät, das wie ein browser auf dem lokalen Netzwerk. Wir zeigen dir, wie du zum erstellen der web-server und wie der code funktioniert, Schritt-für-Schritt.



Wenn Sie wollen zu erfahren Sie mehr über die ESP32, Lesen Sie Erste Schritte mit ESP32 .

Projekt-Übersicht

Bevor Sie direkt zu dem Projekt, es ist wichtig, um zu skizzieren, was unsere web-server, so dass es einfacher ist, befolgen Sie die Schritte, die später auf.

- Der web-server Sie bauen steuert zwei LEDs angeschlossen, um den ESP32 GPIO 26 und GPIO 27 ;
- Sie können auf den ESP32 web server durch Eingabe des ESP32 IP-Adresse auf einem browser im lokalen Netzwerk;
- Durch klicken auf die Schaltflächen auf Ihrem web-server können Sie sofort ändern Sie den Status der jeweiligen LED.

Dies ist nur ein einfaches Beispiel, um zu illustrieren, wie man einen web-server steuert Ausgänge, die Idee ist, solche zu ersetzen, die LED mit einem relais oder andere elektronische Komponenten, die Sie wollen.

Installing the ESP32 board in Arduino IDE

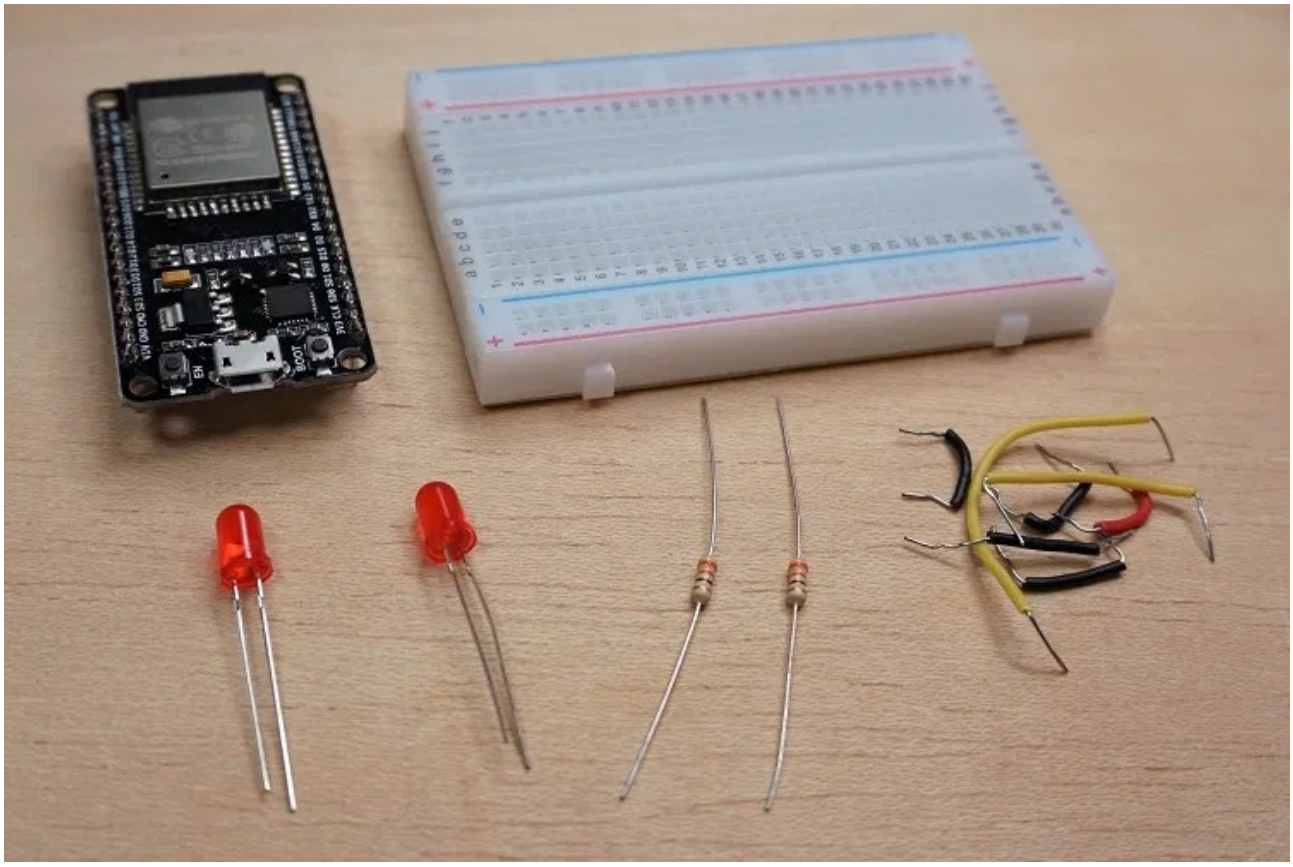
Es gibt ein add-on für die Arduino IDE für die Programmierung des ESP32 mit der Arduino IDE und der Programmiersprache. Folgen Sie einem der folgenden tutorials bereiten Sie Ihre Arduino IDE:

- **Anleitung für Windows** – Installation der ESP32 Board in Arduino IDE

- **Mac-und Linux-Anweisungen** – Installation der ESP32 Board in Arduino IDE

Benötigte Teile

Für dieses tutorial benötigen Sie folgende Teile:



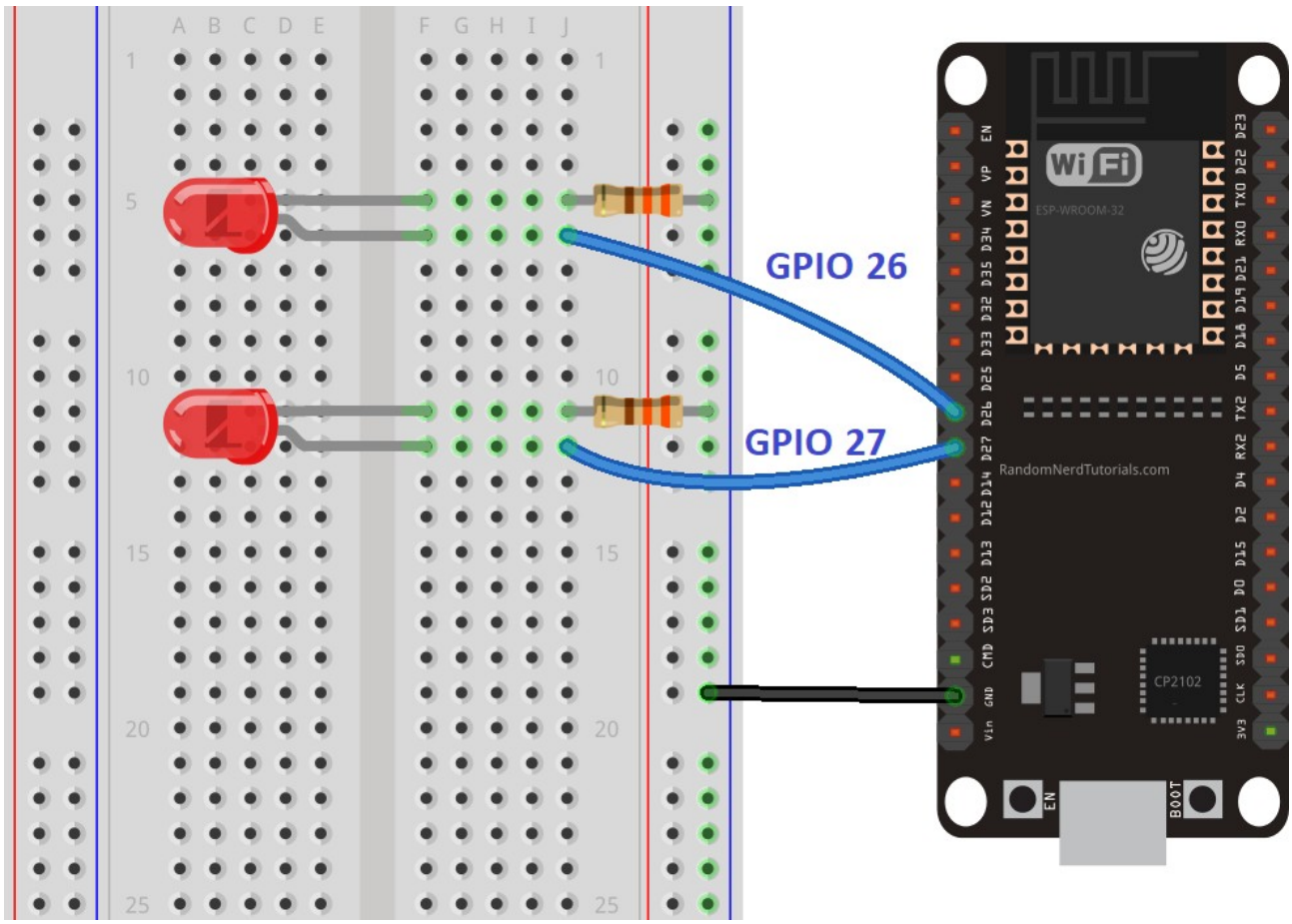
- ESP32 development board – Lesen ESP32 Entwicklung Boards-Review und Vergleich
- 2x 5mm LED
- 2x 330 Ohm Widerstand
- Steckbrett
- Jumper Drähte

Sie können die vorherigen links oder gehen Sie direkt zu MakerAdvisor.com/tools finden Sie alle Teile für Ihre Projekte zum besten Preis!

Schaltplan

Starten Sie durch den Aufbau der Schaltung. Verbinden Sie zwei LEDs, die den ESP32 wie in der folgenden schematischen Abbildung – eine LED angeschlossen an GPIO 26 , und der andere an GPIO 27.

Hinweis : Wir verwenden den ESP32 DEVKIT DOIT-board mit 36 pins. Vor der Montage der Schaltung, stellen Sie sicher, dass Sie überprüfen die pinout für das board, das Sie verwenden.



ESP32 Web Server-Code

Hier bieten wir Ihnen den code, schafft der ESP32 web server. Kopieren Sie den folgenden code in Ihre Arduino IDE, aber nicht hochladen es noch nicht. Sie müssen machen einige änderungen, um es für Sie arbeiten.

```
/*********
  Rui Santos
  Complete project details at http://randomnerdtutorials.com
*********/

// Load Wi-Fi library
#include <WiFi.h>

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

// Set web server port number to 80
WiFiServer server(80);

// Variable to store the HTTP request
String header;

// Auxiliar variables to store the current output state
String output26State = "off";
String output27State = "off";

// Assign output variables to GPIO pins
const int output26 = 26;
const int output27 = 27;
```

```
// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(115200);
  // Initialize the output variables as outputs
  pinMode(output26, OUTPUT);
  pinMode(output27, OUTPUT);
  // Set outputs to LOW
  digitalWrite(output26, LOW);
  digitalWrite(output27, LOW);

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop(){
  WiFiClient client = server.available();   // Listen for incoming clients

  if (client) {                             // If a new client connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");          // print a message out in the serial
port
    String currentLine = "";                // make a String to hold incoming
data from the client
    while (client.connected() && currentTime - previousTime <= timeoutTime)
{  // loop while the client's connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read from the
client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline
character
          // if the current line is blank, you got two newline characters in a
row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200
OK)
            // and a content-type so the client knows what's coming, then a
blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();
```

```
            // turns the GPIOs on and off
            if (header.indexOf("GET /26/on") >= 0) {
              Serial.println("GPIO 26 on");
              output26State = "on";
              digitalWrite(output26, HIGH);
            } else if (header.indexOf("GET /26/off") >= 0) {
              Serial.println("GPIO 26 off");
              output26State = "off";
              digitalWrite(output26, LOW);
            } else if (header.indexOf("GET /27/on") >= 0) {
              Serial.println("GPIO 27 on");
              output27State = "on";
              digitalWrite(output27, HIGH);
            } else if (header.indexOf("GET /27/off") >= 0) {
              Serial.println("GPIO 27 off");
              output27State = "off";
              digitalWrite(output27, LOW);
            }

            // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the on/off buttons
            // Feel free to change the background-color and font-size attributes
to fit your preferences
            client.println("<style>html { font-family: Helvetica; display:
inline-block; margin: 0px auto; text-align: center;}");
            client.println(".button { background-color: #4CAF50; border: none;
color: white; padding: 16px 40px;");
            client.println("text-decoration: none; font-size: 30px; margin: 2px;
cursor: pointer;}");
            client.println(".button2 {background-color:
#555555;}</style></head>");

            // Web Page Heading
            client.println("<body><h1>ESP32 Web Server</h1>");

            // Display current state, and ON/OFF buttons for GPIO 26
            client.println("<p>GPIO 26 - State " + output26State + "</p>");
            // If the output26State is off, it displays the ON button
            if (output26State=="off") {
              client.println("<p><a href=\"/26/on\"><button
class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/26/off\"><button class=\"button
button2\">OFF</button></a></p>");
            }

            // Display current state, and ON/OFF buttons for GPIO 27
            client.println("<p>GPIO 27 - State " + output27State + "</p>");
            // If the output27State is off, it displays the ON button
            if (output27State=="off") {
              client.println("<p><a href=\"/27/on\"><button
class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/27/off\"><button class=\"button
button2\">OFF</button></a></p>");
            }
            client.println("</body></html>");

            // The HTTP response ends with another blank line
```

```
            client.println();
            // Break out of the while loop
            break;
          } else { // if you got a newline, then clear currentLine
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a carriage
return character,
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

Einstellung Ihre Netzwerk-Anmeldeinformationen

Sie müssen ändern Sie die folgenden Zeilen mit Ihren Netzwerk-Anmeldeinformationen: SSID und das Passwort. Der code ist gut kommentiert, wo Sie die änderungen vornehmen.

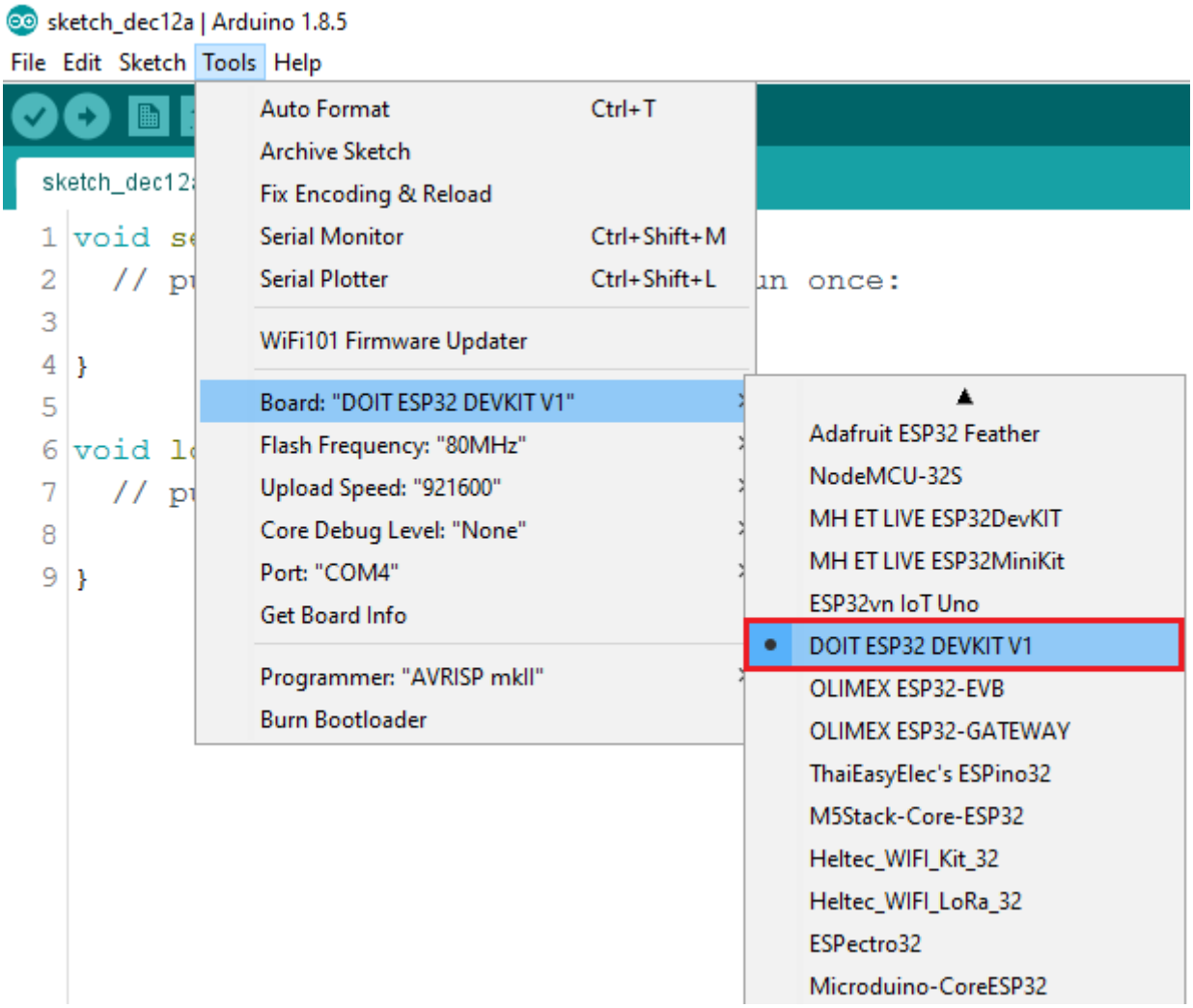
```
// Replace with your network credentials
const char* ssid     = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

Hochladen der Code

Jetzt können Sie laden Sie die code und der web-server wird sofort mit der Arbeit. Folgen Sie den nächsten Schritten, um das hochladen von code auf das ESP32:

1) Stecken Sie Ihre ESP32 board in Ihrem computer;

2) In der Arduino IDE wählen Sie Ihren board-in **Tools** > **Board** (in unserem Fall haben wir mit dem ESP32 DEVKIT DOIT-board);



3) Wählen Sie die COM-port-in **Tools** > **Port** .
4) Drücken Sie die **Upload** - Taste in der Arduino-IDE und ein paar Sekunden warten, während der code kompiliert und uploads auf Ihrem board.
5) Warten, bis die " **Done uploading** " angezeigt.

Finden Sie die ESP-IP-Adresse

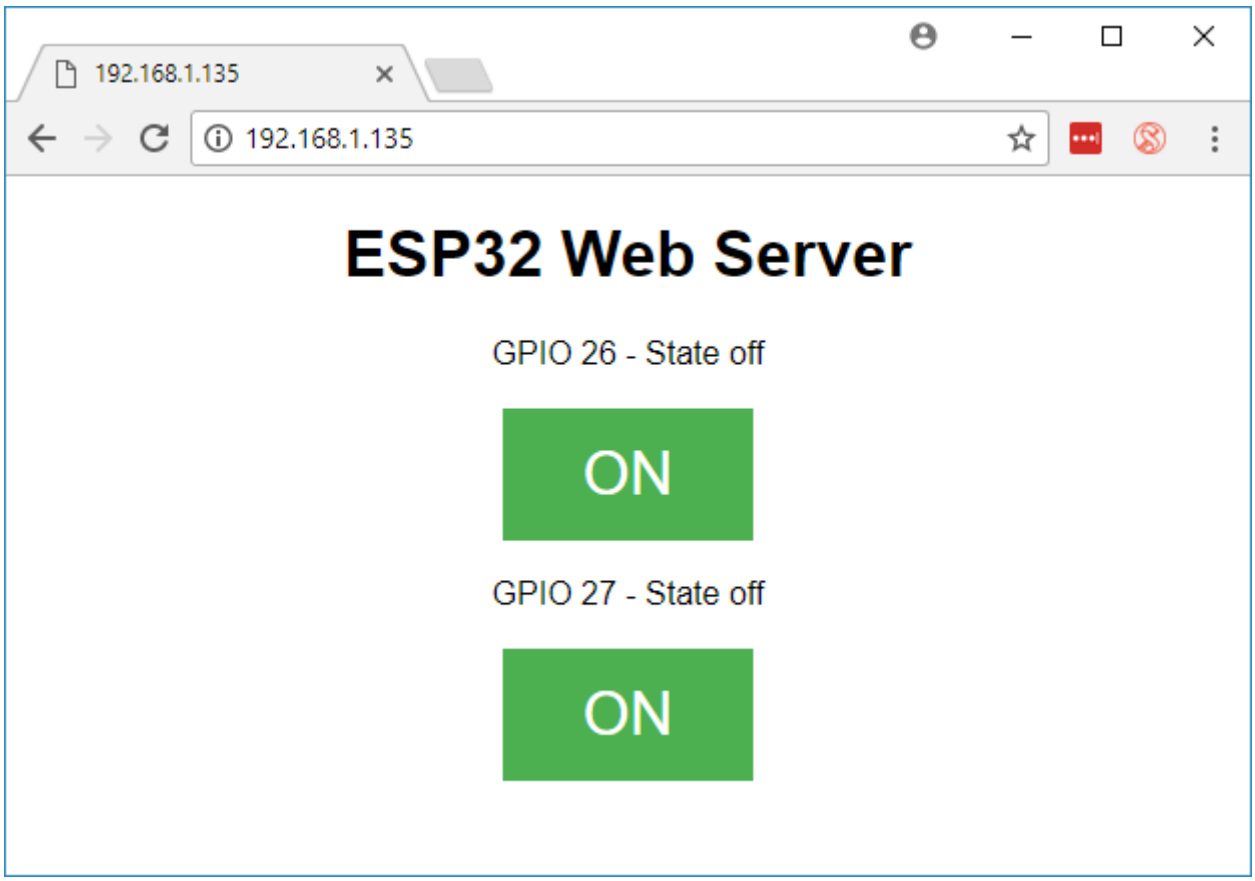
Nach dem hochladen der code, öffnen Sie den Seriellen Monitor mit einer Baudrate von 115200.

Drücken Sie die ESP32 EN-Taste (reset). Der ESP32 verbindet zu Wi-Fi-und Ausgänge des ESP-IP-Adresse über den Serial Monitor. Kopieren Sie die IP-Adresse, weil Sie es brauchen, um auf die ESP32 web server.

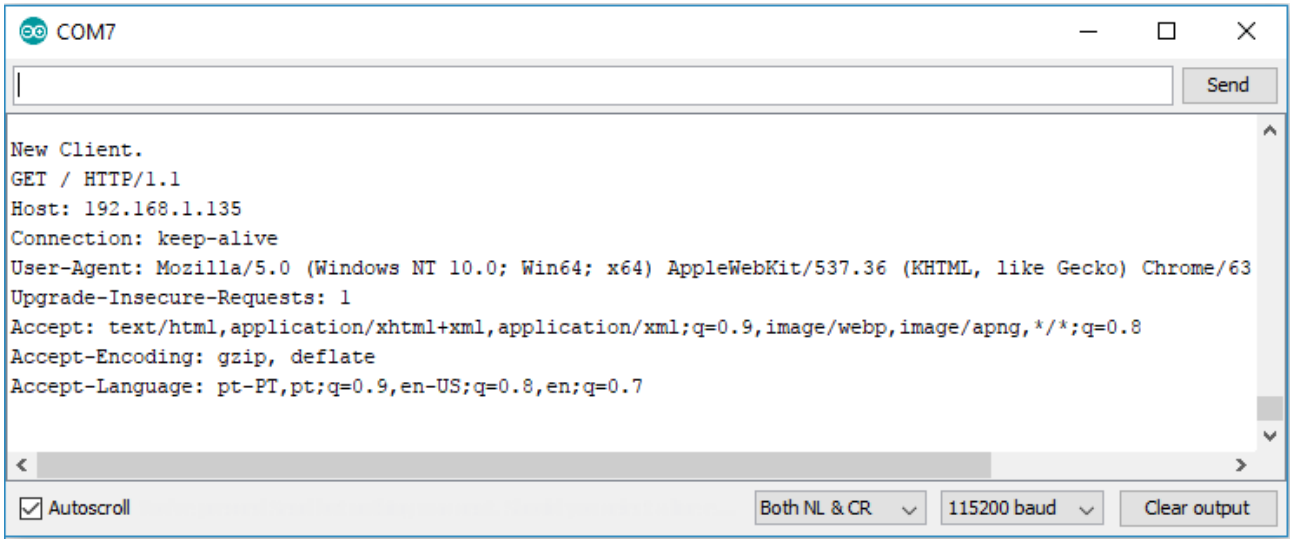
Oder serieller Monitor.

Zugriff auf die Web-Server

Zugriff auf den web-server, öffnen Sie Ihren browser, fügen Sie den ESP32 IP-Adresse, und du wirst sehen, auf der folgenden Seite. In unserem Fall ist es **192.168.1.135** .



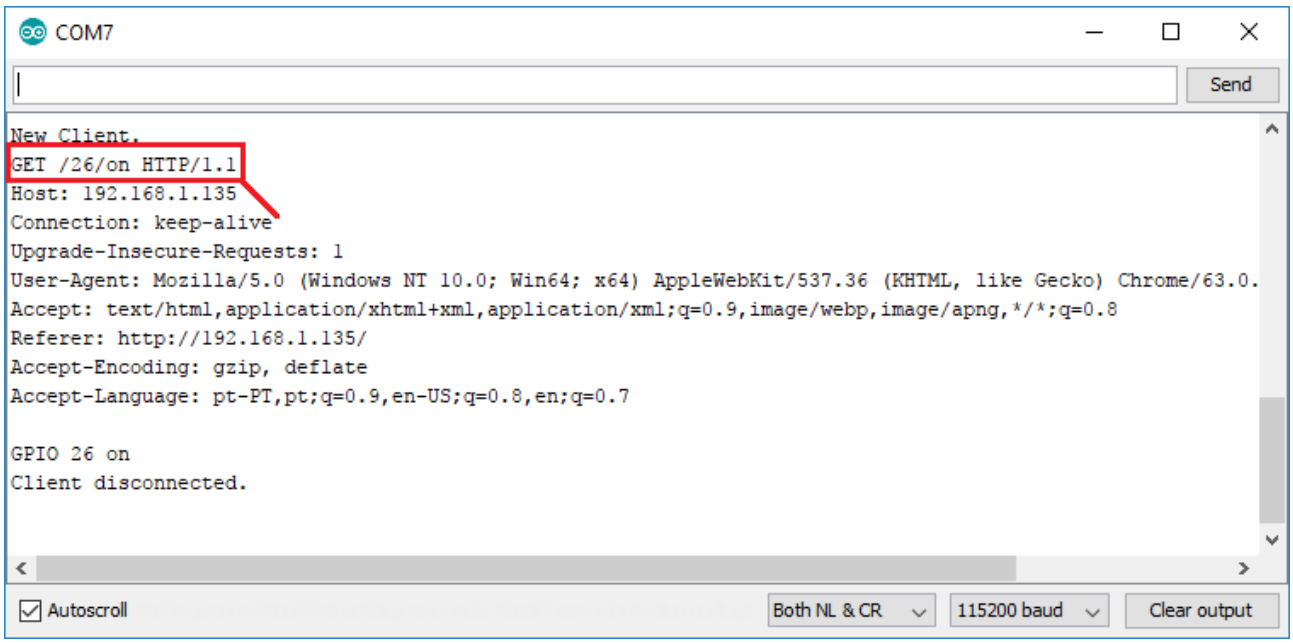
Wenn man einen Blick auf den Serial Monitor, Sie können sehen, was geschieht auf dem hintergrund. Der ESP empfängt eine HTTP-Anfrage von einem neuen Kunden (in diesem Fall wird in Ihrem browser.



Testen der Web-Server

Sie können jetzt testen, ob Ihr web-server ordnungsgemäß funktioniert. Klicken Sie auf die Schaltflächen zur Steuerung der LEDs.

Zur gleichen Zeit, können Sie nehmen Sie einen Blick auf den Seriellen Monitor, um zu sehen, was Los ist in den hintergrund. Zum Beispiel, wenn Sie auf die Taste, um GPIO 26 AUF, ESP32 erhält eine Anfrage an den **/26/auf** - URL.



Wenn der ESP32 erhält die Anforderung, es stellt die LED am GPIO 26 und seinen Status aktualisiert auf der Webseite.

Die Schaltfläche für die GPIO 27 auf eine ähnliche Weise funktioniert. Testen Sie, ob es ordnungsgemäß funktioniert.

Wie der Code Funktioniert

In diesem Abschnitt wird näher auf den code, um zu sehen, wie es funktioniert.

Das erste, was Sie tun müssen, ist zu schließen die WiFi-Bibliothek.

```
#include <WiFi.h>
```

Wie bereits erwähnt, müssen Sie Ihre ssid und das Passwort in die folgenden Zeilen innerhalb der Anführungszeichen.

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

Dann legen Sie Ihr web-server auf port 80.

```
WiFiServer server(80);
```

Die folgende Zeile erstellt eine variable zum speichern der header der HTTP-Anfrage:

```
String header;
```

Als Nächstes erstellen Sie auxiliar Variablen zum speichern des aktuellen Status der Ausgänge. Wenn Sie hinzufügen möchten mehr Ausgänge und Ihren Zustand speichern, müssen Sie zum erstellen mehrerer Variablen.

```
String output26State = "off";
String output27State = "off";
```

Sie müssen auch weisen Sie einen GPIO zu jeder Ihrer Ausgaben. Hier sind wir mit GPIO 26 und GPIO 27 . Sie kann auch jede andere geeignete GPIOs.

```
const int output26 = 26;
const int output27 = 27;
```

setup()

Nun, lassen Sie uns gehen Sie in die setup() . Als erstes starten wir eine serielle Kommunikation mit einer Baudrate von 115200 für debugging-Zwecke.

```
Serial.begin(115200);
```

Definieren Sie auch Ihre GPIOs als Ausgänge und auf LOW.

```
// Initialize the output variables as outputs
pinMode(output26, OUTPUT);
pinMode(output27, OUTPUT);

// Set outputs to LOW
digitalWrite(output26, LOW);
digitalWrite(output27, LOW);
```

Die folgenden Zeilen beginnen, das Wi-Fi-Verbindung mit WiFi.begin(ssid, password) , warten Sie, bis eine erfolgreiche Verbindung-und drucken Sie das ESP-IP-Adresse im Seriellen Monitor.

```
// Connect to Wi-Fi network with SSID and password
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
// Print local IP address and start web server
Serial.println("");
Serial.println("WiFi connected.");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
server.begin();
```

loop()

In der loop() Programmieren wir, was passiert, wenn ein neuer client stellt eine Verbindung mit dem web-server.

Der ESP32 ist immer empfangsbereit für eingehende Kunden mit der folgenden Zeile:

```
WiFiClient client = server.available(); // Listen for incoming clients
```

Beim empfangen einer Anforderung von einem client, wir werden speichern die eingehenden Daten. Die while-Schleife, die folgendermaßen ausgeführt wird, solange der client verbunden bleibt. Wir empfehlen nicht, ändern Sie den folgenden Teil des Codes, es sei denn, Sie wissen genau, was Sie tun.

```
if (client) { // If a new client connects,
  Serial.println("New Client."); // print a message out in the serial port
```

```
  String currentLine = ""; // make a String to hold incoming data from the
client
  while (client.connected()) { // loop while the client's connected
    if (client.available()) { // if there's bytes to read from the client,
      char c = client.read(); // read a byte, then
      Serial.write(c); // print it out the serial monitor
      header += c;
      if (c == '\n') { // if the byte is a newline character
      // if the current line is blank, you got two newline characters in a row.
      / that's the end of the client HTTP request, so send a response:
        if (currentLine.length() == 0) {
        // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
        // and a content-type so the client knows what's coming, then a blank
line:
          client.println("HTTP/1.1 200 OK");
          client.println("Content-type:text/html");
          client.println("Connection: close");
          client.println();
```

Der nächste Abschnitt von if-und else-Anweisungen überprüft, welche Taste gedrückt wurde, in Ihrem web-Seite, und steuert die Ausgänge entsprechend. Wie wir bereits gesehen haben, machen wir eine Anfrage über verschiedene URLs je nach der gedrückten Taste.

```
// turns the GPIOs on and off
if (header.indexOf("GET /26/on") >= 0) {
  Serial.println("GPIO 26 on");
  output26State = "on";
  digitalWrite(output26, HIGH);
} else if (header.indexOf("GET /26/off") >= 0) {
  Serial.println("GPIO 26 off");
  output26State = "off";
  digitalWrite(output26, LOW);
} else if (header.indexOf("GET /27/on") >= 0) {
  Serial.println("GPIO 27 on");
  output27State = "on";
  digitalWrite(output27, HIGH);
} else if (header.indexOf("GET /27/off") >= 0) {
  Serial.println("GPIO 27 off");
  output27State = "off";
  digitalWrite(output27, LOW);
}
```

Zum Beispiel, wenn Sie haben drücken Sie die GPIO-26 ON-Taste, der ESP32 eine Anforderung empfängt, auf dem **/26/AUF-URL** (die wir sehen können, dass Informationen über die HTTP-header auf dem Serial Monitor). So können wir prüfen, ob der header enthält den Ausdruck **BEKOMMEN /26/auf** . Wenn es enthält, wir ändern die output26state variable AUF, und der ESP32 schaltet die LED auf.

Dies funktioniert Analog für die anderen buttons. Also, wenn Sie hinzufügen möchten mehr Ausgänge, sollten Sie ändern Sie diesen Teil des Codes sind.

Die Anzeige des HTML-web-Seite

Das nächste, was Sie tun müssen, ist die Erstellung der web-Seite. Der ESP32 wird das senden einer Antwort an den browser mit ein paar HTML-code zum erstellen der web-Seite.

Die web-Seite an den client gesendet werden mit diesem Ausdruck - client.println() . Sie sollten geben, was Sie wollen zu senden an den client als argument.

Das erste, was wir senden sollen, ist immer die folgende Zeile, das anzeigt, dass wir versenden von HTML.

```
<!DOCTYPE HTML><html>
```

Dann die folgende Zeile macht die web-Seite reagieren in jedem web-browser.

```
client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");
```

Und das folgende wird verwendet, um zu verhindern, dass Anforderungen an die favicon. – Sie brauchen nicht zu sorgen über diese Linie.

```
client.println("<link rel=\"icon\" href=\"data:,\">");
```

Styling der Web-Seite

Neben, wir haben einige CSS text-Stil, die Tasten und die web-Seite Aussehen. Wählen wir die Helvetica-Schrift, definieren Sie den Inhalt als block ausgerichtet und in der Mitte.

```
client.println("<style>html { font-family: Helvetica; display: inline-block;
margin: 0px auto; text-align: center;}");
```

Wir Stil unsere Knöpfe mit der #4CAF50 Farbe, ohne Rand, text in weiß Farbe, und mit dieser padding: 16px 40px. Wir setzen auch den text-decoration none, definieren Sie die schriftart, - Größe, - Rand, und der Mauszeiger zu einem Zeiger.

```
client.println(".button { background-color: #4CAF50; border: none; color: white;
padding: 16px 40px;");
client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor:
pointer;}");
```

Wir definieren auch den Stil für eine zweite Schaltfläche, mit der alle Eigenschaften der Schaltfläche, die wir definiert haben früher, aber mit einer anderen Farbe. Wird dies der Stil für Sie die aus-Taste.

```
client.println(".button2 {background-color: #555555;}</style></head>");
```

Die Einstellung der Web-Seite der Ersten Überschrift

In der nächsten Zeile können Sie die ersten Titel Ihrer web-Seite. Hier haben wir " **ESP32 Web Server** ", aber Sie können diesen text ändern, um was auch immer Sie mögen.

```
// Web Page Heading
client.println("<h1>ESP32 Web Server</h1>");
```

Die Anzeige der Tasten und Entsprechende Staatliche

Dann schreiben Sie einen Absatz zur Anzeige der GPIO 26 aktuellen Stand. Wie Sie sehen können, wir verwenden die output26State variable, so dass die Status-updates sofort, wenn diese variable sich ändert.

```
client.println("<p>GPIO 26 - State " + output26State + "</p>");
```

Dann haben wir die auf-oder ab-Taste, je nach dem aktuellen Status der GPIO. Wenn der aktuelle Zustand des GPIO deaktiviert ist, zeigen wir die AUF-Taste, wenn nicht, wir zeigen Sie die AUS-Taste.

```
if (output26State=="off") {
  client.println("<p><a href=\"/26/on\"><button
class=\"button\">ON</button></a></p>");
} else {
  client.println("<p><a href=\"/26/off\"><button class=\"button
button2\">OFF</button></a></p>");
}
```

Wir verwenden das gleiche Verfahren für die GPIO 27 .

Schließen der Verbindung

Schließlich, wenn die Reaktion endet, werden wir die header - variable, und beenden Sie die Verbindung mit dem client mit dem client.stop() .

```
// Clear the header variable
header = "";
// Close the connection
client.stop();
```

Zusammenfassung

In diesem tutorial haben wir Ihnen gezeigt, wie Sie eine web server mit die ESP32. Wir haben gezeigt, dass Sie ein einfaches Beispiel, das steuert zwei LEDs, aber die Idee ist, solche zu ersetzen, die LED mit einem relais, oder jede andere Ausgabe, die Sie Steuern möchten. Weitere Projekte mit ESP32, überprüfen Sie die folgenden tutorials:

- Bauen Sie ein All-in-One ESP32 Weather Station Shield
- ESP32-Servo-Motor-Web-Server
- Erste Schritte mit ESP32 Bluetooth Low Energy (BLE)
- Mehr ESP32 tutorials

Dies ist ein Auszug aus unserem Kurs: Lernen ESP32 mit der Arduino IDE . Wenn Sie wie ESP32 und Sie mehr erfahren möchten, empfehlen wir die Einschreibung in Lernen ESP32 mit der Arduino IDE natürlich .