Контролирайте множество релета с ESP8266 NodeMCU уеб сървър

В този раздел създадохме пример за уеб сървър, който ви позволява да контролирате колкото искате релета чрез уеб сървър, независимо дали са конфигурирани като нормално отворени или като нормално затворени. Просто трябва да промените няколко реда код, за да определите броя на релетата, които искате да контролирате, и разпределението на щифтовете.

За да изградим този уеб сървър, използваме библиотеката ESPAsyncWebServer .

Инсталиране на библиотеката ESPAsyncWebServer

Следвайте следващите стъпки, за да инсталирате библиотеката ESPAsyncWebServer :

1. Щракнете тук, за да изтеглите библиотеката на ESPAsyncWebServer . Трябва да имате .zip папка във вашата папка за изтегляния
2. Разархивирайте папката .zip и трябва да получите *ESPAsyncWebServer - главна* папка

3. Преименувайте папката си от ~~ESPAsyncWebServer-master~~на *ESPAsyncWebServer*
4. Преместете *ESPAsyncWebServer* в папката на инсталационните библиотеки на Arduino IDE

Като алтернатива, във вашата Arduino IDE можете да отидете на **Sketch** > **Include Library** > **Add .ZIP библиотека...** и да изберете библиотеката, която току-що сте изтеглили.

Инсталиране на библиотеката ESPAsyncTCP за ESP8266

Библиотеката ESPAsyncWebServer изисква ESPAsyncTCP библиотека за работа. Следвайте следващите стъпки, за да инсталирате тази библиотека:

1. Щракнете тук, за да изтеглите библиотеката ESPAsyncTCP . Трябва да имате .zip папка във вашата папка за изтегляния
2. Разархивирайте папката .zip и трябва да получите *ESPAsyncTCP -главна* папка
3. Преименувайте папката си от ~~ESPAsyncTCP-master~~ на *ESPAsyncTCP*
4. Преместете *ESPAsyncTCP* в папката на инсталационните библиотеки на Arduino IDE
5. И накрая, отворете отново вашата Arduino IDE

Като алтернатива, във вашата Arduino IDE можете да отидете на **Sketch** > **Include Library** > **Add .ZIP библиотека...** и да изберете библиотеката, която току-що сте изтеглили.

След като инсталирате необходимите библиотеки, копирайте следния код във вашето Arduino IDE.

```
/*********
  Nikolay Manchev PhD
  Department of Communications Equipment and Technologies
   Technical University Gabrovo
   2025


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

```
*********/

// Import required libraries
#include "ESP8266WiFi.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] = {5, 4, 14, 12, 13};

// 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 ESP8266 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() {

}
```

Дефиниране на конфигурация на реле

Променете следната променлива, за да посочите дали използвате вашите релета в нормално отворена (NO) или нормално затворена (NC) конфигурация. Задайте променливата RELAY_NO на true за нормално отворена операционна система, зададена на false за нормално затворена.

#define RELAY_NO вярно

Определете броя на релетата (канали)

Можете да определите броя на релетата, които искате да контролирате, в променливата NUM_RELAYS. За демонстрационни цели, ние го настройваме на 5.

#define NUM_RELAYS 5

Определете назначението на изводите на релетата

В следната променлива на масива можете да дефинирате ESP8266 GPIO, които ще управляват релетата.

int relayGPIOs [NUM_RELAYS] = {5, 4, 14, 12, 13};

Броят на релетата, зададен в променливата NUM_RELAYS, трябва да съответства на броя на GPIO, присвоени в масива relayGPIO .

Мрежови идентификационни данни

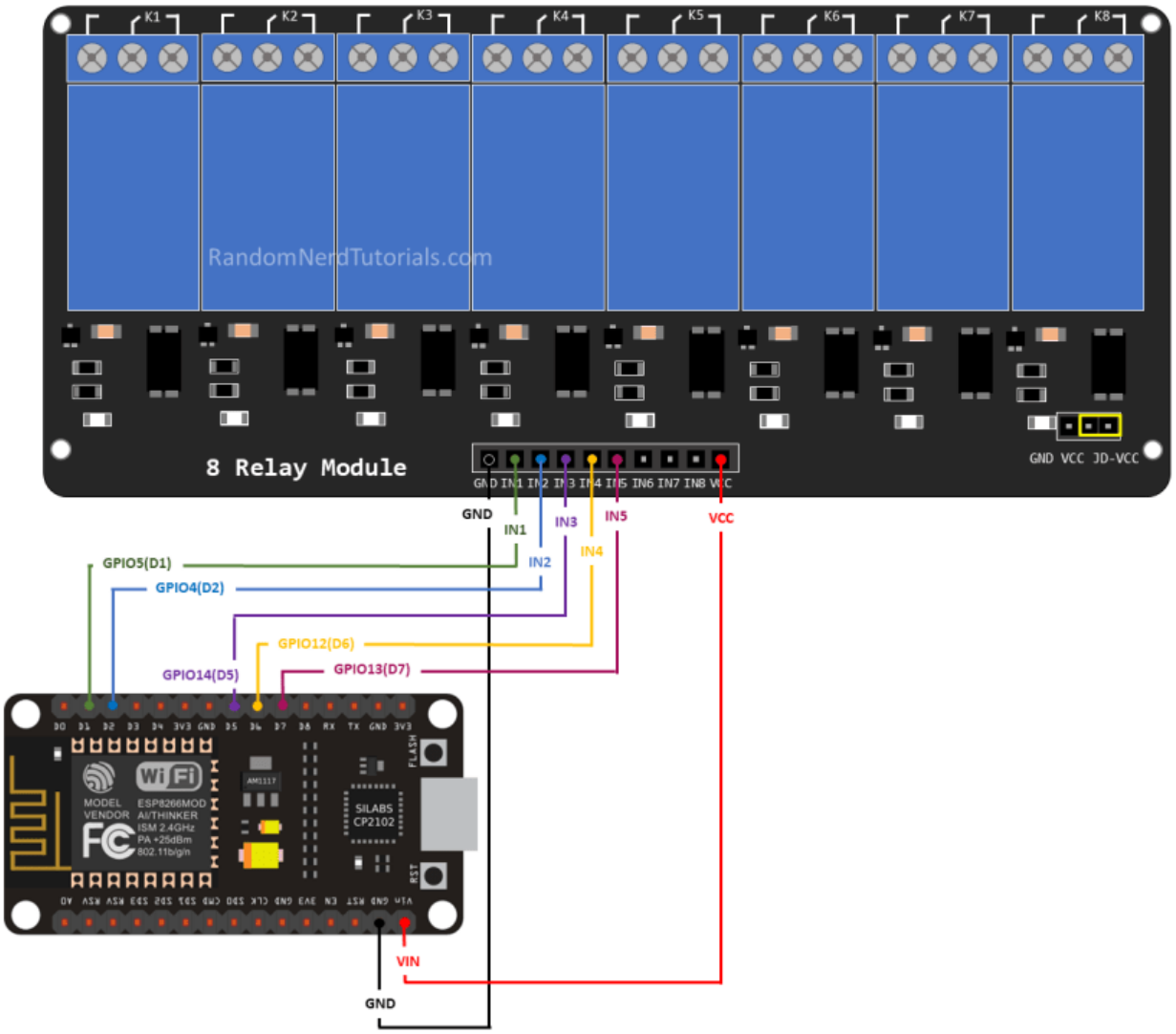
Въведете вашите мрежови идентификационни данни в следните променливи.

const char* ssid = "ЗАМЕНЕТЕ_С_ВАШИЯ_SSID ";

const char* password = "ЗАМЕНЕТЕ_С_ВАШАТА_ПАРОЛА ";

Свързване на 8-канално реле към ESP8266 NodeMCU

За демонстрационни цели контролираме 5 релейни канала. Свържете платката ESP8266 NodeMCU към релейния модул, както е показано на следващата схематична диаграма.



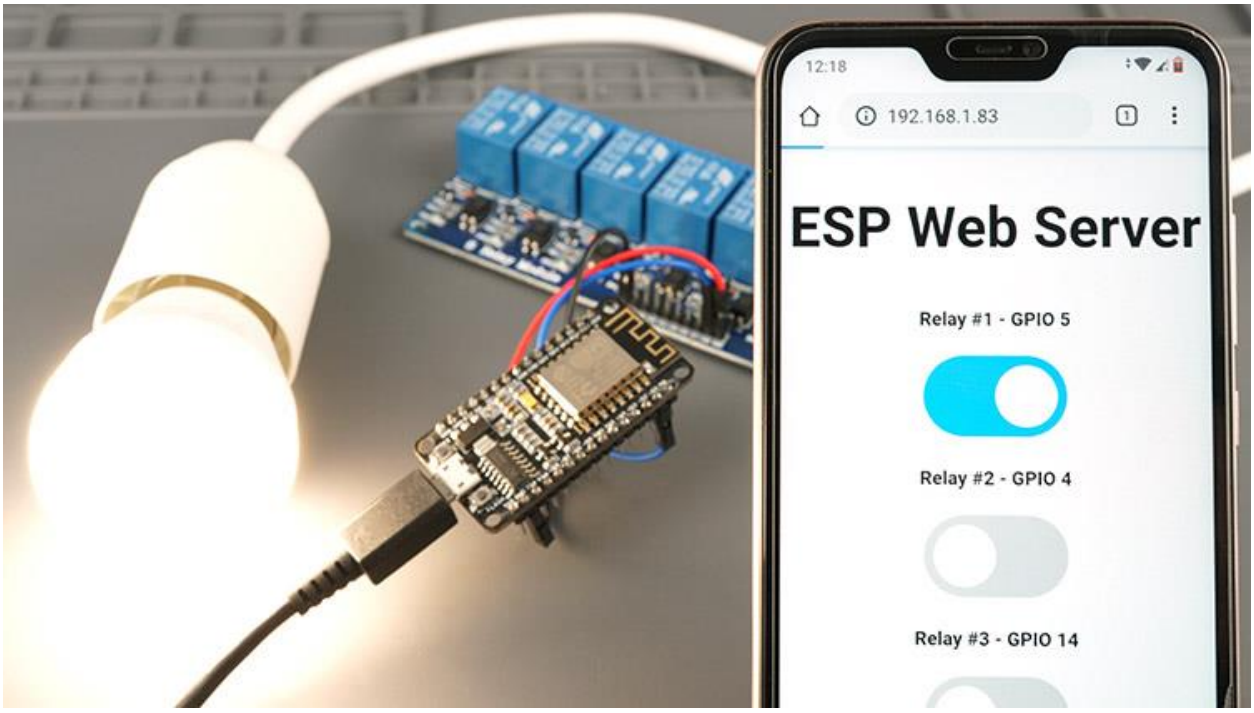
Демонстрация

След като направите необходимите промени, качете кода на вашия ESP8266.

Отворете серийния монитор при скорост на предаване от 115200 бода и натиснете бутона ESP8266 RST, за да получите неговия IP адрес.

След това отворете браузър във вашата локална мрежа и въведете IP адреса на ESP8266, за да получите достъп до уеб сървъра.

Трябва да получите нещо както следва с толкова бутони, колкото броя на релетата сте дефинирали във вашия код.



Сега можете да използвате бутоните, за да контролирате вашите релета от разстояние с помощта на вашия смартфон.

Корпус за безопасност

За окончателен проект се уверете, че сте поставили релейния си модул и ESP вътре в кутия, за да избегнете излагане на AC щифтове.