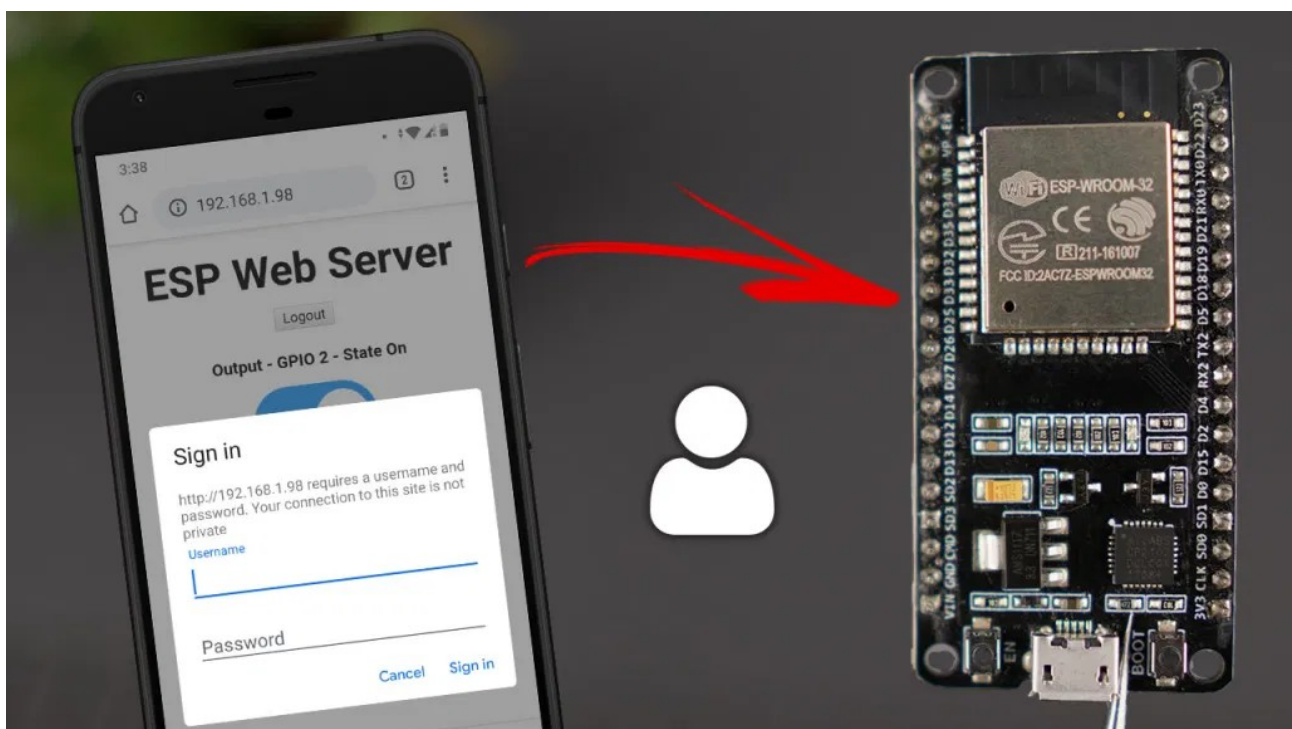


# ESP32/ESP8266 Web Server HTTP Authentication (Username and Password Protected)

Erfahren Sie, wie Sie die HTTP-Authentifizierung mit Benutzernamen und Passwort zu Ihrem ESP32 und ESP8266 NodeMCU web server Projekte mit der Arduino IDE. Sie haben nur Zugriff auf Ihren Webserver, wenn Sie geben Sie den richtigen Benutzernamen und pass. Wenn Sie sich Abmelden, können Sie erst wieder Zugriff, wenn Sie geben Sie die richtigen Anmeldeinformationen.

Die Methode, die wir verwenden werden, angewendet werden kann, um web-Server erstellt mit den ESPAsyncWebServer Bibliothek.



Der ESP32/ESP8266 boards werden mit Arduino IDE programmiert. So stellen Sie sicher, dass Sie diese boards installiert ist:

- [Installing ESP32 Board in Arduino IDE \(Windows, Mac OS X und Linux\)](#)
- [Installing ESP8266 Board in Arduino IDE \(Windows, Mac OS X, Linux\)](#)

## Sicherheits-Bedenken

Dieses Projekt ist gedacht, um verwendet werden in Ihrem lokalen Netzwerk zum Schutz von Personen nur die Eingabe der ESP-IP-Adresse und Zugriff auf die web-server (wie nicht autorisierte Familienmitglied oder Freund).

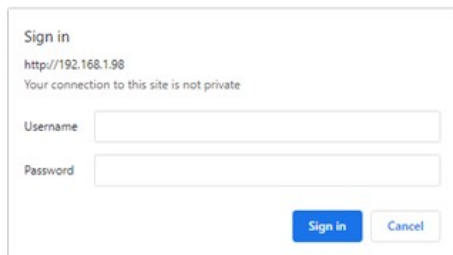
Wenn Ihr Netzwerk ordnungsgemäß gesichert ist, läuft ein HTTP-server mit basic authentication ist genug für die meisten Anwendungen. Wenn jemand hat es geschafft, der hack der Ihr Netzwerk, ist

es egal, ob Sie HTTP oder HTTPS. Die hacker umgehen können, HTTPS und Holen Sie sich Ihren Benutzer - /pass.

## Projekt-Übersicht

Werfen wir einen kurzen Blick auf die features des Projekt-wir bauen werde.

### 1) Login

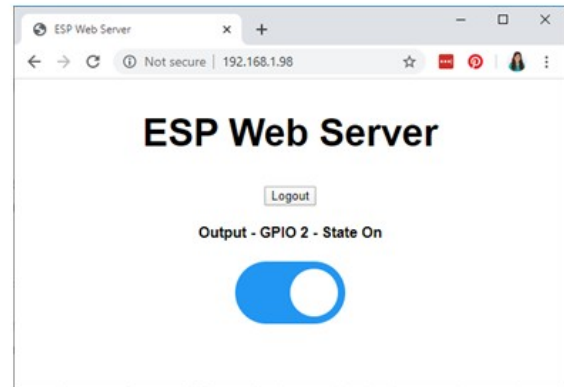


Sign in  
http://192.168.1.98  
Your connection to this site is not private

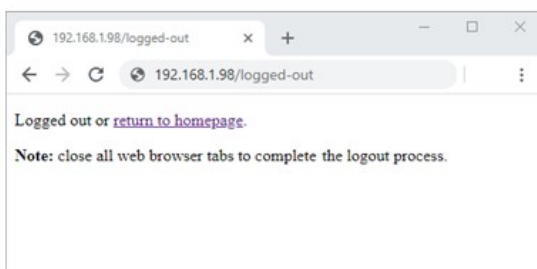
Username

Password

### 2) Access Web Server



### 4) Logged out



### 3) Logout



- In diesem tutorial erfahren Sie, wie Passwort schützen Sie Ihr web-server;
- Wenn Sie versuchen, Zugriff auf die web-server-Seite auf die ESP-IP-Adresse, ein Fenster erscheint und fragt nach einem Benutzernamen und Kennwort;
- Um Zugriff auf die web-server-Seite müssen Sie den richtigen Benutzernamen und das Kennwort ein (definiert in dem ESP32/ESP8266 sketch);
- Es gibt einen logout-button an das web-server. Wenn Sie klicken Sie auf die Schaltfläche Abmelden, werden Sie umgeleitet werden, um eine logout-Seite. Dann, schließen Sie alle browser-Registerkarten, um komplette den logout-Prozess;
- Sie können nur den Zugriff auf den web-server wieder, wenn Sie login mit den richtigen Anmeldeinformationen;
- Wenn Sie versuchen, Zugriff auf den web-server von einem anderen Gerät (im lokalen Netzwerk), müssen Sie auch login mit den richtigen Anmeldeinformationen (auch wenn Sie haben eine erfolgreiche Anmeldung auf einem anderen Gerät);
- Die Authentifizierung ist nicht verschlüsselt.

**Hinweis:** dieses Projekt wurde getestet unter Google Chrome und Mozilla Firefox web-Browser und Android-Geräte.

# Installation Von Bibliotheken – Async-Web-Server

Zum erstellen der web-server zu installieren, müssen Sie die folgenden Bibliotheken:

- **ESP32:** installieren Sie den [ESPAsyncWebServer](#) und die [AsyncTCP](#) Bibliotheken.
- **ESP8266:** installieren Sie den [ESPAsyncWebServer](#) und die [ESPAsyncTCP](#) Bibliotheken.

Diese Bibliotheken sind nicht verfügbar, um die Installation über den Arduino Bibliotheksmanager, so müssen Sie kopieren Sie die library-Dateien, um die Arduino-Installation Ordner "Libraries". Alternativ dazu können Sie in Ihrem Arduino-IDE, Sie können gehen Sie auf **Sketch > Include Library > Add .zip-Bibliothek** und wählen Sie die Bibliotheken haben Sie gerade heruntergeladen haben.

## Web-Server mit Authentifizierung Code

Kopieren Sie den folgenden code in Ihre Arduino IDE.

```
/*  
  Rui Santos  
  Complete project details at https://RandomNerdTutorials.com/esp32-esp8266-web-  
  server-http-authentication/  
  
  The above copyright notice and this permission notice shall be included in all  
  copies or substantial portions of the Software.  
  */  
  
// Import required libraries  
#ifdef ESP32  
  #include <WiFi.h>  
  #include <AsyncTCP.h>  
#else  
  #include <ESP8266WiFi.h>  
  #include <ESPAsyncTCP.h>  
#endif  
#include <ESPAsyncWebServer.h>  
  
// Replace with your network credentials  
const char* ssid = "REPLACE_WITH_YOUR_SSID";  
const char* password = "REPLACE_WITH_YOUR_PASSWORD";  
  
const char* http_username = "admin";  
const char* http_password = "admin";  
  
const char* PARAM_INPUT_1 = "state";  
  
const int output = 2;  
  
// Create AsyncWebServer object on port 80  
AsyncWebServer server(80);  
  
const char index_html[] PROGMEM = R"rawliteral(  
<!DOCTYPE HTML><html>  
<head>  
  <title>ESP Web Server</title>  
  <meta name="viewport" content="width=device-width, initial-scale=1">  
  <style>  
    html {font-family: Arial; display: inline-block; text-align: center;}  
    h2 {font-size: 2.6rem;}  
    body {max-width: 600px; margin:0px auto; padding-bottom: 10px;}  
  </style>  
)"
```

```

        .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>
    <button onclick="logoutButton()">Logout</button>
    <p>Output - GPIO 2 - State <span id="state">%STATE%</span></p>
    %BUTTONPLACEHOLDER%
<script>function toggleCheckbox(element) {
    var xhr = new XMLHttpRequest();
    if(element.checked){
        xhr.open("GET", "/update?state=1", true);
        document.getElementById("state").innerHTML = "ON";
    }
    else {
        xhr.open("GET", "/update?state=0", true);
        document.getElementById("state").innerHTML = "OFF";
    }
    xhr.send();
}
function logoutButton() {
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "/logout", true);
    xhr.send();
    setTimeout(function(){ window.open("/logged-out","_self"); }, 1000);
}
</script>
</body>
</html>
)rawliteral";

```

```

const char logout_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
    <p>Logged out or <a href="/">return to homepage</a>.</p>
    <p><strong>Note:</strong> close all web browser tabs to complete the logout
process.</p>
</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 = "";
        String outputStateValue = outputState();
        buttons+= " <p><label class=\"switch\"><input type=\"checkbox\"
onchange=\"toggleCheckbox(this)\" id=\"output\" " + outputStateValue + "><span
class=\"slider\"></span></label></p>";
        return buttons;
    }
}

```

```

    }
    if (var == "STATE"){
        if(digitalRead(output)){
            return "ON";
        }
        else {
            return "OFF";
        }
    }
    return String();
}

String outputState(){
    if(digitalRead(output)){
        return "checked";
    }
    else {
        return "";
    }
    return "";
}

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

    pinMode(output, OUTPUT);
    digitalWrite(output, LOW);

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

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

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

    server.on("/logout", HTTP_GET, [](AsyncWebServerRequest *request){
        request->send(401);
    });

    server.on("/logged-out", HTTP_GET, [](AsyncWebServerRequest *request){
        request->send_P(200, "text/html", logout_html, processor);
    });

    // Send a GET request to <ESP_IP>/update?state=<inputMessage>
    server.on("/update", HTTP_GET, [](AsyncWebServerRequest *request) {
        if(!request->authenticate(http_username, http_password))
            return request->requestAuthentication();
        String inputMessage;
        String inputParam;
        // GET input1 value on <ESP_IP>/update?state=<inputMessage>
        if (request->hasParam(PARAM_INPUT_1)) {
            inputMessage = request->getParam(PARAM_INPUT_1)->value();
            inputParam = PARAM_INPUT_1;
        }
    });
}

```

```

        digitalWrite(output, inputMessage.toInt());
    }
    else {
        inputMessage = "No message sent";
        inputParam = "none";
    }
    Serial.println(inputMessage);
    request->send(200, "text/plain", "OK");
});

// Start server
server.begin();
}

void loop() {
}

```

Sie müssen nur geben Sie Ihre Netzwerk-Anmeldeinformationen (SSID und Kennwort) und der web-server wird sofort mit der Arbeit. Der code ist kompatibel mit dem [ESP32](#) und [ESP8266](#) - boards.

Als Beispiel bauen wir eine web-server steuert GPIO-2 . Sie können mithilfe der HTTP-Authentifizierung mit einem beliebigen web-server aufgebaut mit den [ESPAsyncWebServer Bibliothek](#) .

## Wie der Code Funktioniert

Wir haben bereits erklärt, in großen details wie web-Server, so wie diese Arbeit in vorherigen tutorials ( [DHT Temperatur Web-Server](#) oder [- Relay-Web-Server](#) ), so dass wir nur einen Blick auf die relevanten Teile add Benutzername-und Passwort-Authentifizierung auf dem web-server.

### Netzwerk-Anmeldeinformationen

Wie bereits erwähnt, müssen Sie Ihre Netzwerk-Anmeldeinformationen in den folgenden Zeilen:

```

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

```

### Einstellung Ihren Benutzernamen und Ihr Passwort

In den folgenden Variablen legen Sie den Benutzernamen und das Kennwort für Ihr web-server. Standardmäßig ist der Benutzername **admin** und das Kennwort ist ebenfalls **admin** . Wir empfehlen auf jeden Fall, Sie zu ändern.

```

const char* http_username = "admin";
const char* http_password = "admin";

```

### Logout-Button

In der index\_html variable sollten Sie einige HTML-text zu fügen Sie einen logout-button. In diesem Beispiel ist es eine einfache logout-button ohne styling Dinge einfacher machen.

```

<button onclick="logoutButton()">Logout</button>

```

Beim klicken auf die Schaltfläche ruft die `logoutButton()` JavaScript-Funktion. Diese Funktion macht eine HTTP-GET-Anfrage an den ESP32/ESP8266 auf die **an** - **/Abmelden** URL. Dann, in der ESP-code, sollten Sie behandeln, was geschieht nach Erhalt dieser Anfrage.

```
function logoutButton() {  
  var xhr = new XMLHttpRequest();  
  xhr.open("GET", "logout", true);  
  xhr.send();  
}
```

Eine Sekunde, nachdem Sie auf die Schaltfläche Abmelden, die Sie umgeleitet werden auf der logout-Seite, auf der **/logged-out** - URL.

```
  setTimeout(function(){ window.open("/logged-out","_self"); }, 1000);  
}
```

## Die Behandlung von Anfragen mit Authentifizierung

Jedes mal, wenn Sie machen eine Anfrage an den ESP32 oder ESP8266 für den Zugriff auf die web-server, es wird geprüft, ob Sie bereits eingegeben, Benutzername und Passwort zu authentifizieren.

Grundsätzlich Authentifizierung hinzufügen, um Ihre web-server, müssen Sie nur fügen Sie die folgenden Zeilen nach jeder Anfrage an:

```
if(!request->authenticate(http_username, http_password))  
  return request->requestAuthentication();
```

Diese Linien kontinuierlich pop-up-Authentifizierung Fenster, bis Sie legen Sie die richtigen Zugangsdaten.

Sie müssen dies tun, für alle Anforderungen. Auf diese Weise stellen Sie sicher, dass Sie bekommen nur Antworten, wenn Sie angemeldet sind.

Zum Beispiel, wenn Sie versuchen, den Zugriff auf die root-URL (ESP IP-Adresse), fügen Sie die vorherigen zwei Zeilen vor dem senden der Seite. Wenn Sie die falschen Anmeldeinformationen, die browser halten für Sie Fragen.

```
server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request){  
  if(!request->authenticate(http_username, http_password))  
    return request->requestAuthentication();  
  request->send_P(200, "text/html", index_html, processor);  
});
```

Hier ist ein weiteres Beispiel, wenn der ESP empfängt eine Anforderung an den **/state** - URL.

```
server.on("/state", HTTP_GET, [] (AsyncWebServerRequest *request) {  
  if(!request->authenticate(http_username, http_password))  
    return request->requestAuthentication();  
  request->send(200, "text/plain", String(digitalRead(output)).c_str());  
});
```

## Griff-Logout-Button

Wenn Sie auf die Schaltfläche Abmelden, die ESP empfängt eine Anforderung an den **/logout** - URL ein. Wenn das passiert, senden Sie den response-code 401.

```
server.on("/logout", HTTP_GET, [] (AsyncWebServerRequest *request){
```

```
request->send(401);  
});
```

Die Antwort-code 401 ist ein nicht autorisierter Fehler HTTP-Antwort-Statuscode, der angibt, dass die Anfrage gesendet durch der client konnte nicht authentifiziert werden. Also, es wird haben die gleiche Wirkung wie ein logout – es wird Sie bitten, den Benutzernamen und das Kennwort ein, und lassen Sie nicht den Zugriff auf den web-server wieder, bis Sie sich anmelden.

Wenn Sie auf den web-server logout-button, nach einer Sekunde, die ESP erhält eine weitere Anforderung an das **/logged-out** - URL. Wenn das passiert, senden Sie den HTML-text zu bauen, die der logout-Seite ( logout\_html variable).

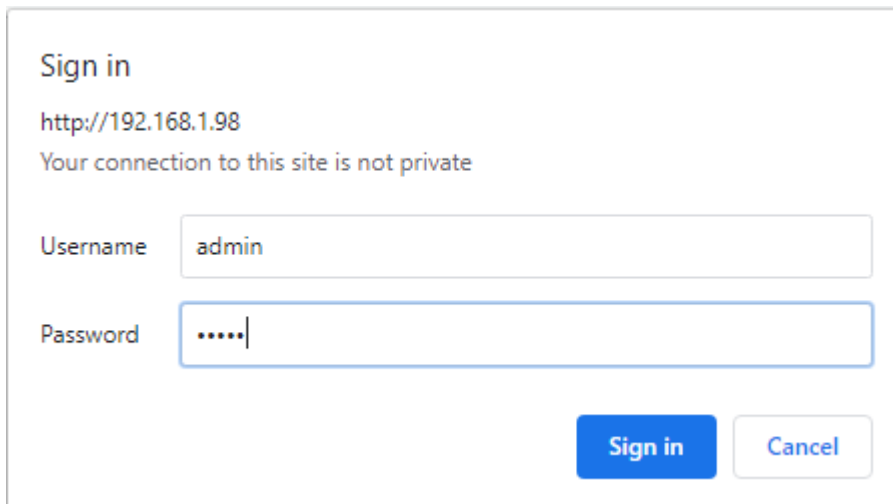
```
server.on("/logged-out", HTTP_GET, [](AsyncWebServerRequest *request){  
    request->send_P(200, "text/html", logout_html, processor);  
});
```

## Demonstration

Laden Sie den code auf Ihrer ESP32 oder ESP8266 board. Öffnen Sie dann den Seriellen Monitor und drücken Sie den on-board-RST/EN-Taste wird die IP-Adresse.

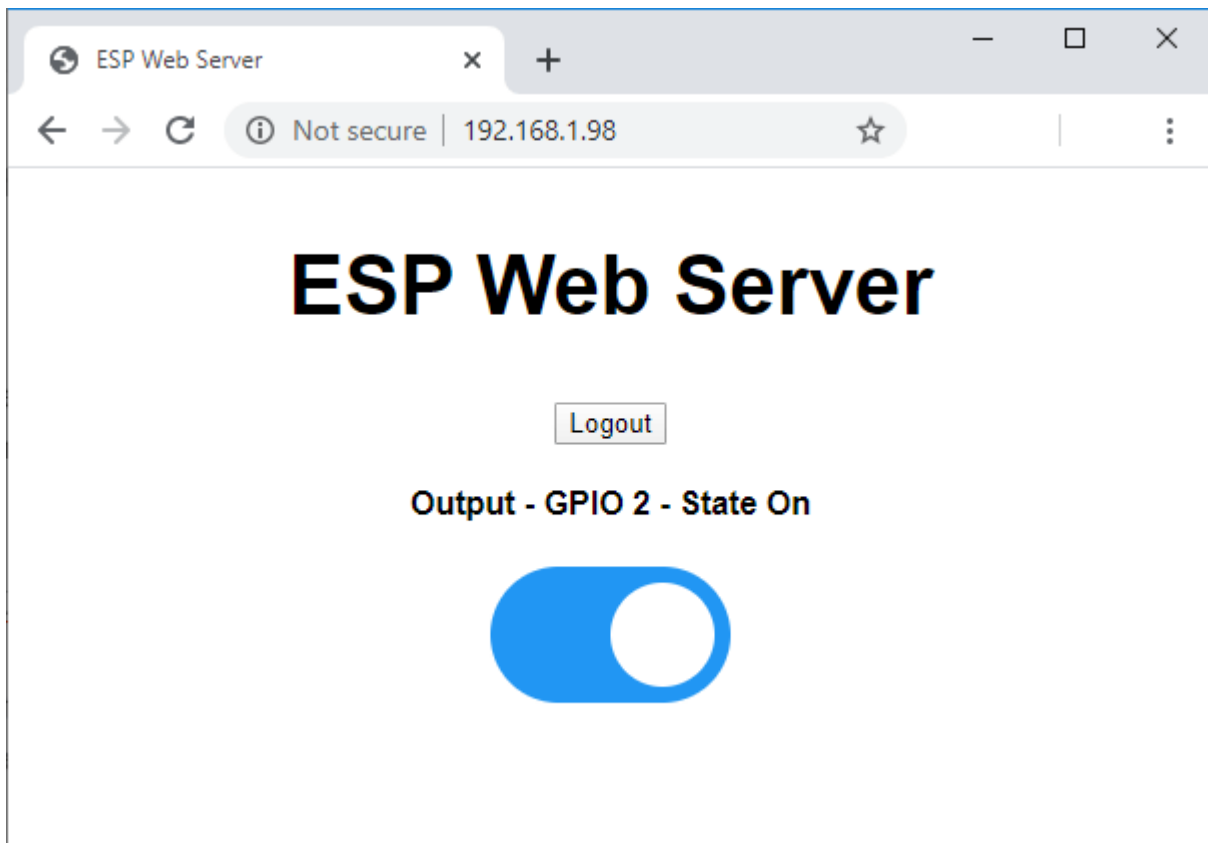
Öffnen Sie einen browser in Ihrem lokalen Netzwerk und geben Sie das ESP-IP-Adresse.

Auf der folgenden Seite laden soll gefragt werden, den Benutzernamen und das Passwort ein. Geben Sie den Benutzernamen und Passwort ein und Sie sollten Zugriff auf den web-server. Wenn Sie noch nicht geändert, die code, wird der Benutzername ist **admin** und das Kennwort ist **admin** .

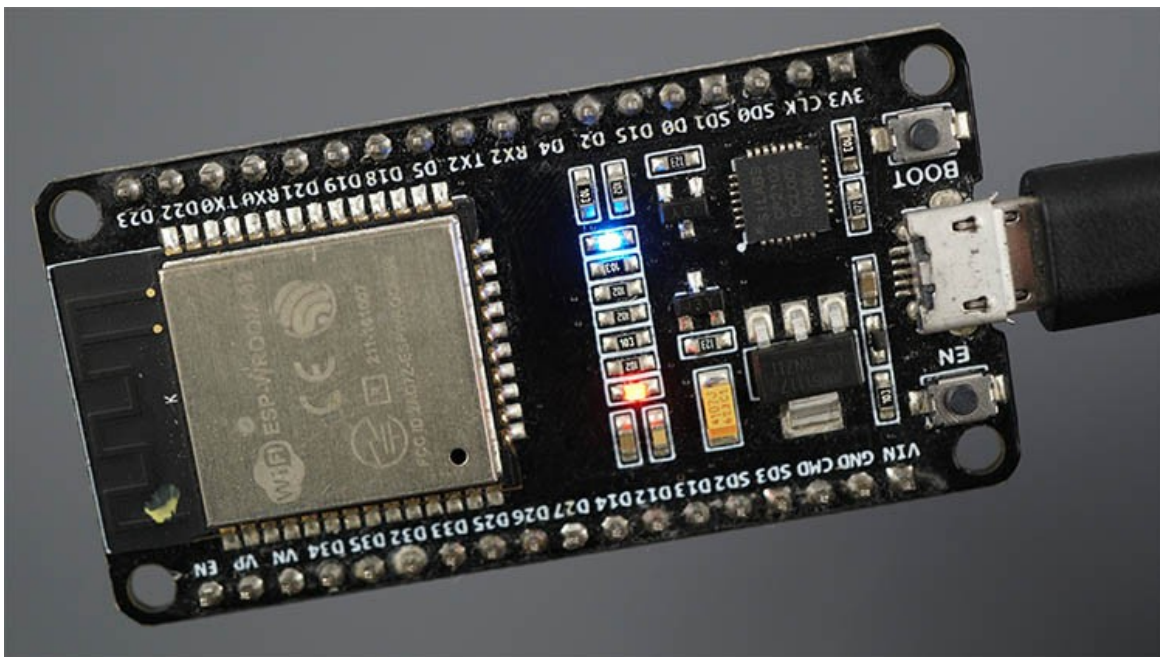


Nach der Eingabe des richtigen Benutzernamens und Passworts, sollten Sie sich den Zugriff auf den web-server.

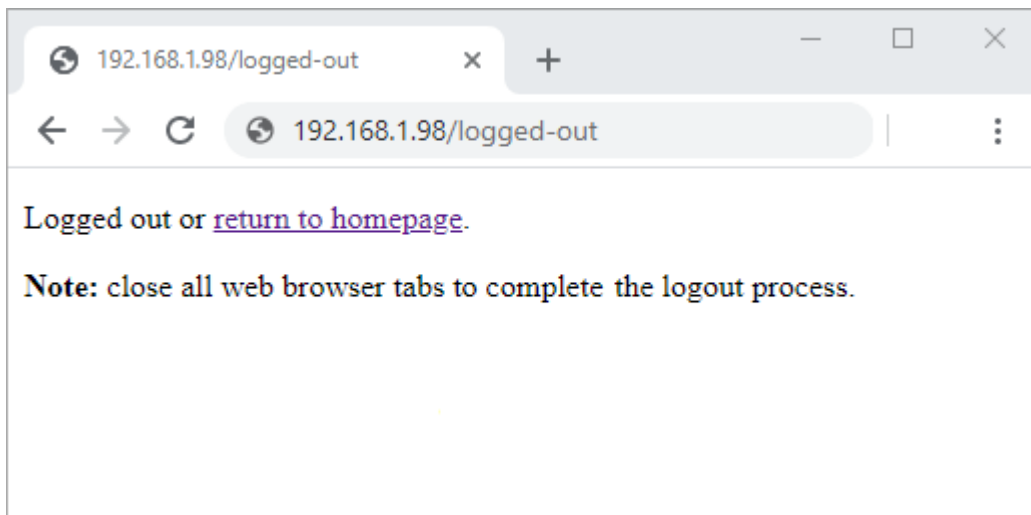




Sie können mit dem web-server, und sehen, dass es tatsächlich steuert den ESP32 oder ESP8266 on-board-LED.



In der web server-Seite gibt es einen logout-button. Wenn Sie auf diesen button klicken, werden Sie umgeleitet werden, um eine logout-Seite, wie unten dargestellt.



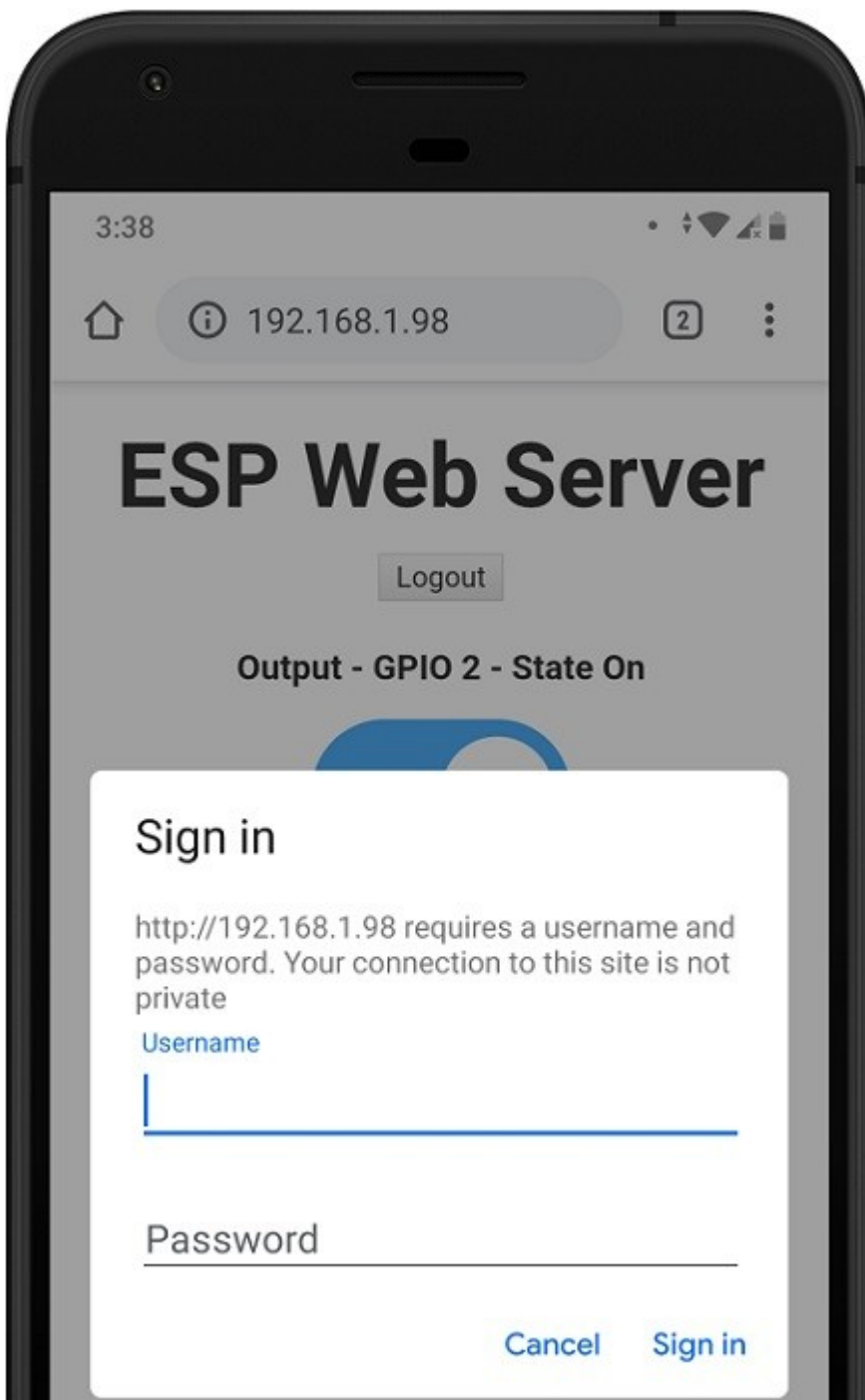
Wenn Sie auf "zurück zur Startseite" - link, Sie werden weitergeleitet auf die web-server-Seite.

Wenn Sie mit Google Chrome, geben Sie den Benutzernamen und das Kennwort zum Zugriff auf den Webserver erneut.

Falls Sie Firefox verwenden, müssen Sie zum schließen aller web-browser-Registerkarten, um sich vollständig Abmelden. Andernfalls, wenn Sie gehen zurück zu die wichtigsten web-server-Seite, Sie werde noch Zugriff haben.

So, empfehlen wir Ihnen, schließen Sie alle web-browser-tabs nach einem Klick auf den logout-button.

Müssen Sie auch geben Sie den Benutzernamen und das Kennwort, wenn Sie versuchen, den Zugriff mit einem anderen Gerät im lokalen Netzwerk, auch wenn Sie Zugriff auf ein anderes Gerät.



## Zusammenfassung

In diesem tutorial haben Sie gelernt, wie Sie auf Authentifizierung hinzufügen zu Ihrem ESP32 und ESP8266 web-Server (Passwort-geschützten web-server). Sie können anwenden, was Sie gelernt haben, in dieser tutorial zu einem beliebigen web-server aufgebaut mit den ESPAsyncWebServer Bibliothek.

Wir hoffen, Sie haben gefunden Sie dieses tutorial nützlich. Andere web-server-Projekten, die Sie möglicherweise mögen:

- [ESP32/ESP8266: Steuerausgänge mit Web-Server und eine Physische Taste Gleichzeitig](#)
- [ESP32/ESP8266 Web Server: Steuerausgänge mit Momentary Switch](#)
- [ESP32/ESP8266 Relais Modul Web Server using Arduino IDE](#)

Erfahren Sie mehr über die ESP32 und ESP8266-boards mit unseren Ressourcen:

- [Lernen ESP32 mit Arduino-IDE](#)
- [Home-Automation mit ESP8266](#)
- [Kostenlose ESP32 Projekte, Tutorials und Guides](#)
- [Free ESP8266 NodeMCU Projekte, Tutorials und Guides](#)

Vielen Dank für das Lesen.