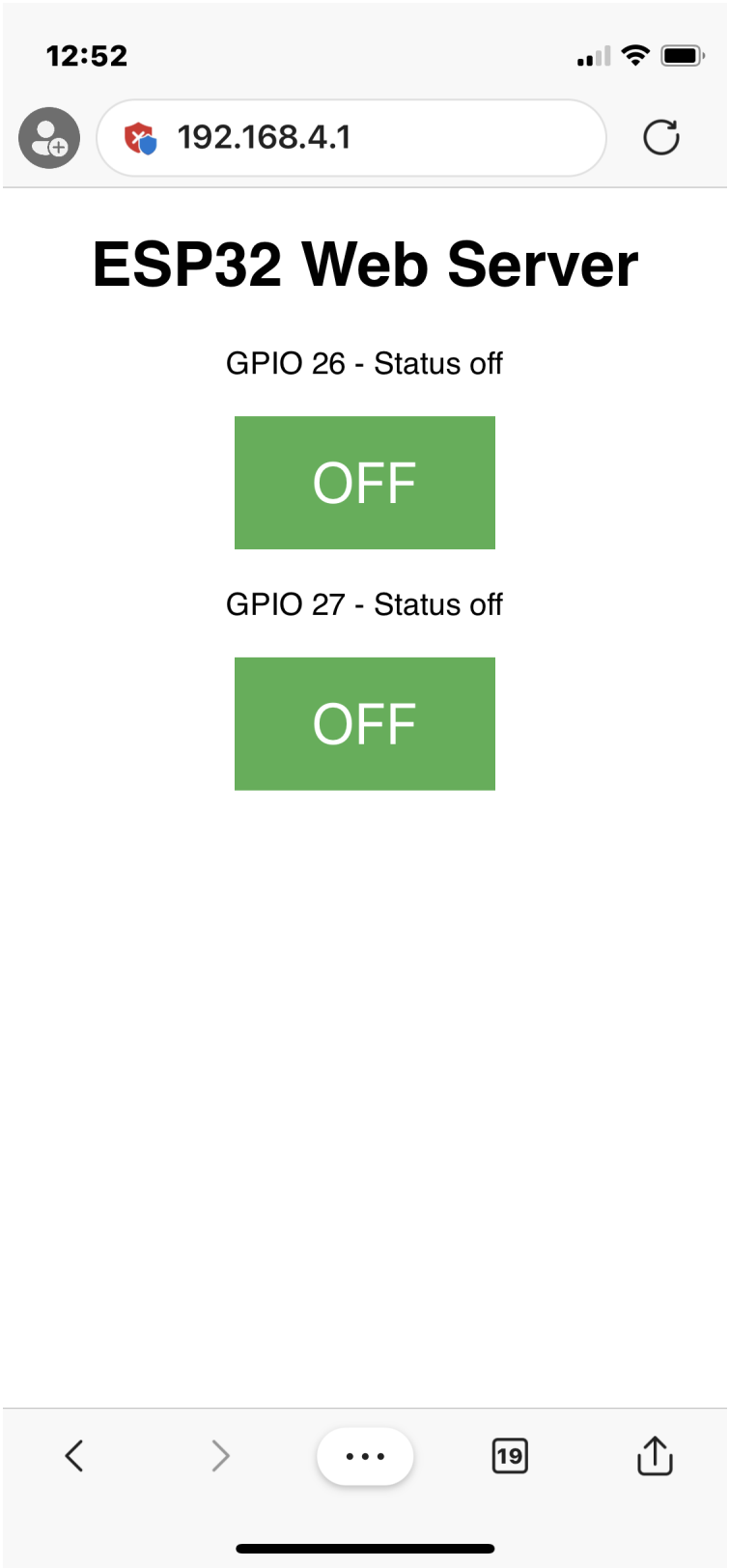
AP Webserver - Programmier Anleitung

Einleitung

Diese Anleitung beschreibt den Aufbau eines WEB-Servers mit einem ESP32. Dabei nutzen wir die WLAN-Funktion des ESP32 und erstellen einen Access-Point. Nachdem wir uns mit dem Access-Point verbunden haben, rufen wir den WEB-Server mit seiner IP-Adresse auf und steuern damit zwei Ausgänge des ESP32.



Die Programmierung

Das Programm in den ESP32 laden:

Starte die Arduino IDE

Kopiere aus der Nextcloud das Programm "ESP32_AP.txt" in die Arduino IDE

Ändere in der Zeile const char* ssid = "Oberlab01"; die Nummer 01 in deine Platznummer

Verbinde den ESP32 über die USB-Schnittstelle mit dem Notebook

Lade das Programm in den ESP32

Die Verbindung mit dem ESP32 herstellen und den WEB-Server starten:

Öffne die WLAN-Einstellungen deines Handys/Tablet

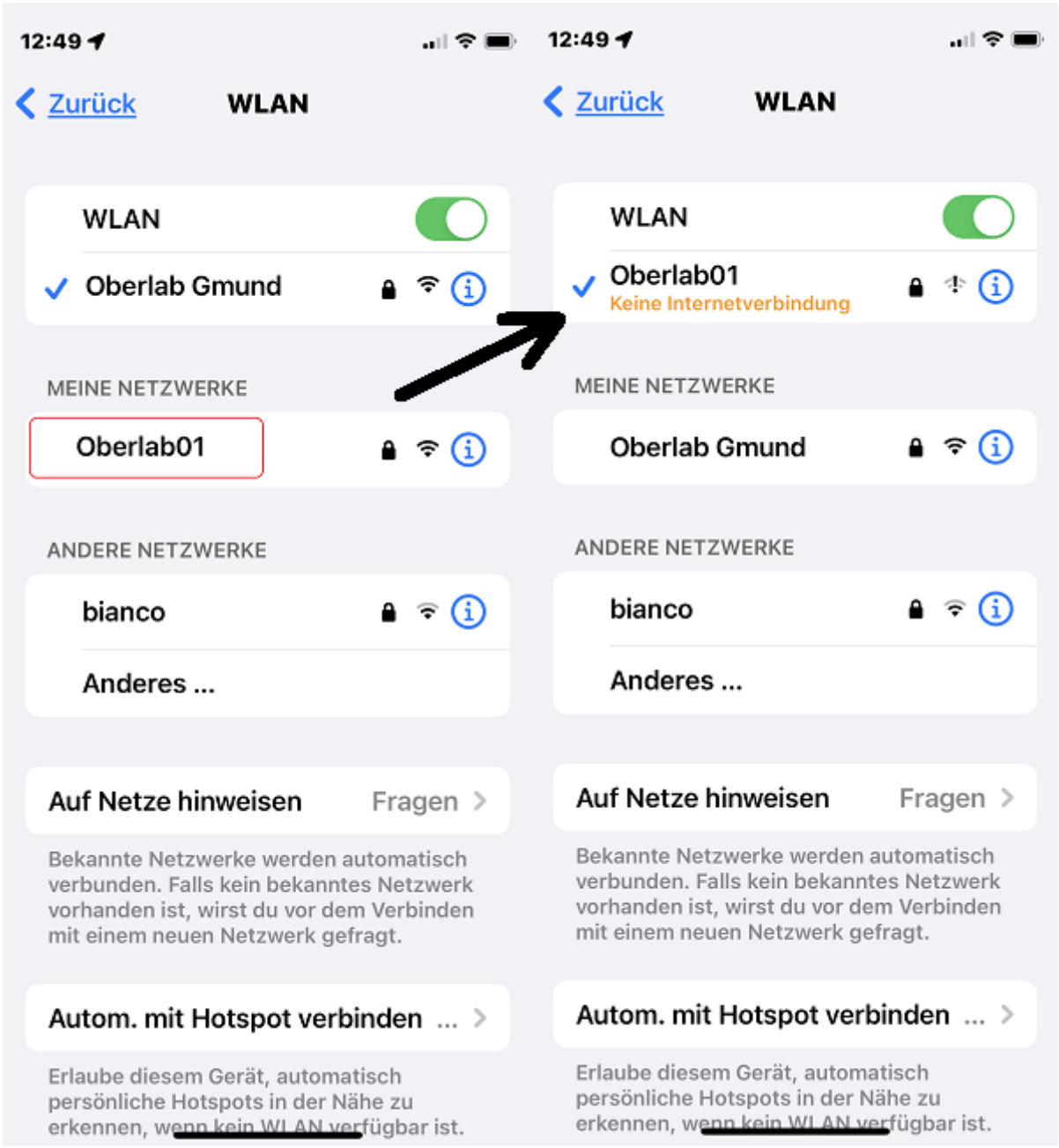
Wähle das Netzwerk Oberlabxx aus

Wenn das Handy/Tablet mit dem ESP32 verbunden ist, dann öffne deinen Browser

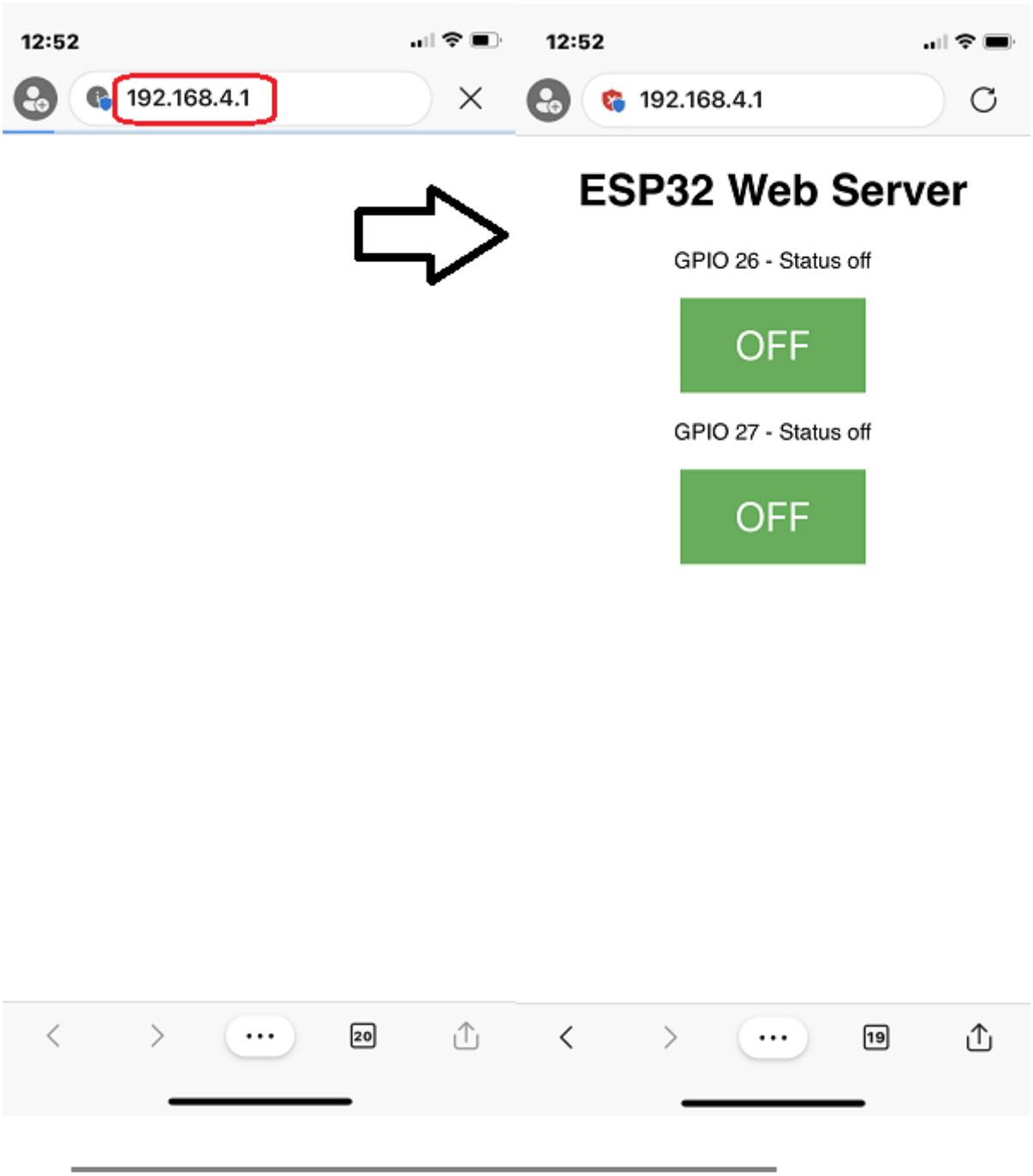
gibt im Webadressfeld die IP-Adresse 192.168.4.1 ein

Wenn der WEB-Server geladen ist, kannst du die Ausgänge des ESP32 steuern

Das Oberlabxx Netzwerk auswählen



IP-Adresse 192.168.4.1 eingeben



ESP32 Test-Programm (interne LED blinkt)

```
void setup() {
  pinMode(2, OUTPUT);
```

```
}

void loop() {
    digitalWrite(2, HIGH);
    delay(1000);
    digitalWrite(2, LOW);
    delay(1000);

}
```

WEB-Server Programm

```
/*****************************************************************************************
****
                                        PROGRAMMINFO
*****************************************************************************************
****
Funktion: ESP32 Access Point mit WEB Server fuer zwei GPIOs
Mobile Browser:   192.168.4.1
*****************************************************************************************
****
Version: khf 23.05.2022
*****************************************************************************************
****
Board: ESP32vn IoT UNO Boardverwalter V1.0.4 (2.0.0)
*****************************************************************************************
****
C++ Arduino IDE V1.8.19
*****************************************************************************************
****
Einstellungen:
https://dl.espressif.com/dl/package_esp32_index.json
http://dan.drown.org/stm32duino/package_STM32duino_index.json
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-
pages/package_esp32_dev_index.json
*****************************************************************************************
****
```

```
**********************************************************************************************
****/
//Wi-Fi Library https://github.com/arduino-libraries/WiFi
#include <WiFi.h>

const char* ssid     = "Dein Name"; //Mit diesen Namen erscheint der ESP32 in deiner WLAN-
Liste
const char* password = "12345678";

WiFiServer server(80);

String header;

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

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

void setup() {
  Serial.begin(115200);
  // Definiere Ausgänge
  pinMode(output26, OUTPUT);
  pinMode(output27, OUTPUT);
  // Setzt Ausgänge auf LOW
  digitalWrite(output26, LOW);
  digitalWrite(output27, LOW);

  // Wi-Fi verbinden
  Serial.print("Service Set Identifier (Access Point)â€¦");

  WiFi.softAP(ssid, password);

  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  server.begin();
}
```

```
void loop(){
  WiFiClient client = server.available();   // eingehende Clients

  if (client) {                             // Wenn neue Client-Verbindung,
    Serial.println("New Client.");          // sende Info an seriellen Monitor
    String currentLine = "";                // Zeichenfolge, um eingehende Daten vom Client zu
speichern
    while (client.connected()) {            // Schleife, während der Client verbunden ist
      if (client.available()) {             // wenn es Bytes gibt, die vom Client gelesen
werden müssen,
        char c = client.read();             // Byte lesen und dann
        Serial.write(c);                    // auf den seriellen Monitor ausgeben
        header += c;
        if (c == '\n') {                    // Wenn es sich bei dem Byte um ein
Zeilenumbruchzeichen handelt
          // Wenn die aktuelle Zeile leer ist, dann schreibe zwei Zeilenumbruchzeichen in
einer Zeile.
          // Das ist das Ende der Client-HTTP-Anforderung, also sende eine Antwort:
          if (currentLine.length() == 0) {
            // HTTP-Header beginnen immer mit einem Antwortcode (z.B. HTTP/1.1 200 OK)
            // und ein Content-Typ, damit der Client weiß, was kommt, dann eine Leerzeile:)
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // Aktiviert und deaktiviert die GPIOs
            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);
            }

            // Anzeigen der HTML-Webseite
            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 Style on/off Buttons
            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: #FF0D0D;}</style></head>");

            // Webseite Kopf
            client.println("<body><h1>ESP32 Web Server</h1>");

            // Anzeigen des aktuellen Zustands und der Ein-/Aus-Tasten für GPIO 26
            client.println("<p>GPIO 26 - Status " + output26State + "</p>");
            // Wenn der output26State deaktiviert ist, wird die ON-Taste angezeigt
            if (output26State=="off") {
              client.println("<p><a href=\"/26/on\"><button
class=\"button\">OFF</button></a></p>");
            } else {
              client.println("<p><a href=\"/26/off\"><button class=\"button
button2\">ON</button></a></p>");
            }

            // Anzeigen des aktuellen Zustands und der Ein-/Aus-Tasten für GPIO 27
            client.println("<p>GPIO 27 - Status " + output27State + "</p>");
            // Wenn der output27State deaktiviert ist, wird die ON-Taste angezeigt
            if (output27State=="off") {
              client.println("<p><a href=\"/27/on\"><button
class=\"button\">OFF</button></a></p>");
            } else {
```

```
              client.println("<p><a href=\"/27/off\"><button class=\"button
button2\">ON</button></a></p>");
            }
            client.println("</body></html>");

            // Die HTTP-Antwort endet mit einer weiteren Leerzeile
            client.println();
             break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {  // Wenn etwas anderes als ein Return-Zeichen dann,
          currentLine += c;      // füge es am Ende der currentLine hinzu
        }
      }
    }
    // Lösche header variable
    header = "";
    // Schließe Verbindung
    client.stop();
    Serial.println("Client getrennt");
    Serial.println("");
  }
}
```

Fertig, Glückwunsch!

Aufgabe

Erweitere den ESP-WEB-Server auf vier Ausgänge.

Version #1
Erstellt: 2025-03-31 09:05:56 UTC von Joel Hatsch
Zuletzt aktualisiert: 2025-03-31 09:12:08 UTC von Joel Hatsch