11 Ein Server zum Steuern

In diesem Kapitel wollen wir eine am ESP32 angeschlossene LED per Wlan ein- und ausschalten. Der Client, sei es ein PC oder ein Handy, kann dann als Fernsteuerung für unsere LED angesehen werden. Wenn man die LED durch ein Relais ersetzt, kann man damit auch ein Gerät steuern, welches größere Ströme bzw. Spannungen benötigt als der ESP32 liefern kann. Auf diese Weise könnte man so z. B. eine 12V-Halogenlampe aus der Ferne mit dem Handy ein- und ausschalten. **Bitte beachten Sie bei der Auswahl des Verbrauchers: Das Arbeiten mit Spannungen, die höher als 24 V sind, sind lebensgefährlich! Eine übliche 230V-Glühbirne sollte mit unserem Server daher auf keinen Fall gesteuert werden.**

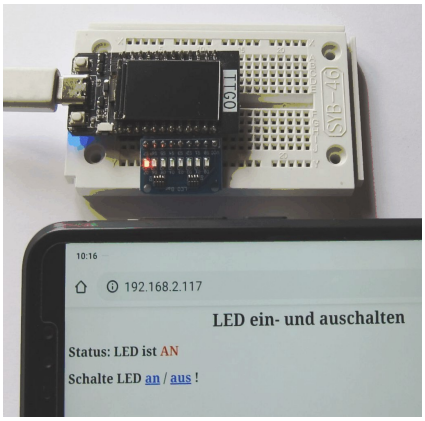

Abb. 1

Nach dem, was wir bislang gelernt haben, ist folgende Idee naheliegend: Wie schon in den vorherigen Kapiteln wollen wir einen Webserver implementieren. Dieser soll jetzt - und das ist neu - über den Browser gesteuert werden. Genauer: Bei einem Request sendet der Server wie bislang eine HTML-Datei an den Client. Diese wird im Browser angezeigt und informiert den Nutzer über den Zustand der angeschlossenen LED: Ist sie eingeschaltet oder nicht? Die Vorgehensweise ist in dieser Hinsicht ähnlich wie bei unserem Temperatursensor. Das Server-Programm soll nun zusätzlich dem Nutzer die Möglichkeit bieten, vom Client (Browser) aus die LED ein- und auszuschalten. Dazu kann der Nutzer in der Adresszeile hinter die IP-Adresse des Servers Zeichenketten `wie '/LED_AN'` bzw. `'/LED_AUS'` schreiben. Diese werden dann als GET-Parameter übertragen. Das Server-Programm muss diese Zeichenketten dann aus dem Request isolieren - ganz ähnlich wie wir es bei dem Namen-Server des letzten Kapitels gemacht haben. Und je nach empfangener Zeichenkette wird schließlich der Server die LED ein- bzw. ausschalten.

Diese manuelle Eingabe der GET-Parameter ist zwar möglich, aus der Sicht des Nutzers ist sie aber recht umständlich und auch wenig intuitiv. Wir verbessern sie, indem wir den Nutzer die GET-Parameter nicht über die Adresszeile, sondern mit Hilfe von Hyper-Links (kurz: Links) eingeben lassen: Klickt er nämlich auf einen solchen Link, fordert der Browser beim Server eine neue Webseite an; wir können nun dafür sorgen, dass hierbei als GET-Parameter einer der beiden erwähnten Zeichenketten `'/LED_AN'` bzw. `'/LED_AUS'` verwendet werden. Wie das geht, schauen wir uns jetzt genauer an.

Zunächst ist es erforderlich, die Zusammenhänge zwischen Links, URL und GET-Parameter deutlich zu machen. Dazu betrachten wir die Webseite in Abb. 2. Wenn wir mit dem Mauszeiger (ohne zu klicken) über den Link fahren, zeigen die meisten Browser am unteren Fensterrand die Ziel-URL an, hier: `http://www.g-heinrichs.de/beispiel.htm`. Klickt man auf diesen Link, dann wechselt der Browser zu der neuen Webseite mit genau dieser URL. Hierzu wird ein neuer HTTP-Request an den Server `www.g-heinrichs.de` gesendet. Der Get-Parameter ist `'/beispiel23.htm'`.

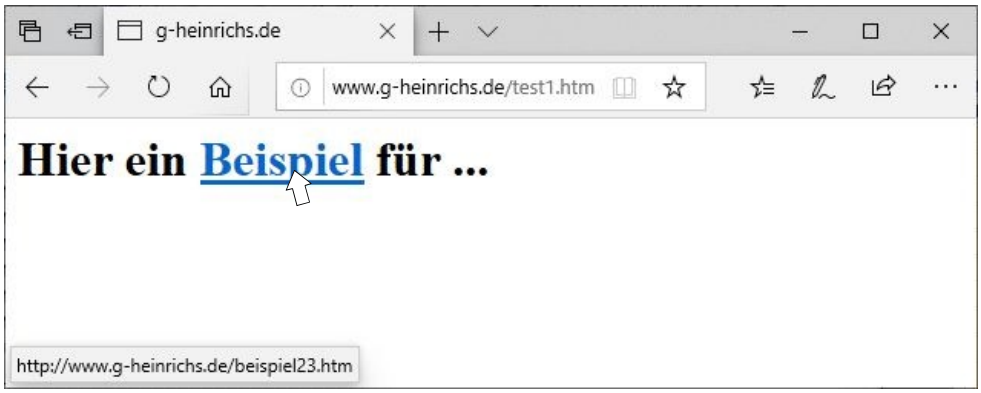

Abb. 2

Damit der Nutzer auf diese Weise die LEDs steuern kann, soll die HTML-Datei zwei Links bereitstellen: Mit dem einen Link soll die LED eingeschaltet werden; mit dem anderen Link soll sie ausgeschaltet werden.

Jetzt müssen wir noch wissen, wie ein Link mit HTML kodiert wird. Dies geschieht mit dem `<a>`-Tag, oft auch als Link-Tag bezeichnet. Für den Link aus Abb. 2 sieht dieser Tag z. B. so aus:

```
<a href = '/beispiel23.htm'>Beispiel</a>
```

Was zwischen dem Start-Tag `<a>` und dem End-Tag `</a>` steht, bezeichnen wir als **Link-Text**; was sich hinter `href =` zwischen den Anführungszeichen befindet, nennen wir die **Link-Referenz**.

Für den Link-Tag zum Einschalten benutzen wir jetzt `/LED_AN` als Link-Referenz und das Wort `'an'` als Link-Text. In unserem HTML-String wird dieser Tag demnach so aussehen:

```
<a href = '/LED_AN'>an</a>
```

Im Browser wird man diesen Link lediglich als an sehen (vgl. Abb. 3). Wenn man ihn anklickt, sendet der Browser wie gewünscht einen Request mit dem GET-Parameter `'/LED_AN'` an den Server. Entsprechendes gilt für den Link zum Ausschalten.

Die Bildung des HTML-Strings lagern wir wie gehabt in eine Funktion `html()` aus; das hierbei benutzte `led`-Objekt muss natürlich zuvor erzeugt und konfiguriert worden sein (vgl. Kap. E.2):

```
def html():
  if led.value() == 1:
    led_state = 'AN'
  else:
    led_state = 'AUS'
```

```
  return '''<html>
              <body>
                 <p> <h1> <center> LED ein- und auschalten </center> </h1> </p>
                 <p> </p>
                 <p> <h2> Status: LED ist <font color='red'>
  '''                          + led_state +   '''
                                                 </font> </h2> </p>
                 <p> <h2>
                   Schalte LED <a href='/LED_AN'> <b>an</b> </a> /
                   <a href='/LED_AUS'> <b>aus</b> </a> !
                 </h2> </p>
              </body>
            </html>
  '''
```

Zu Beginn der Funktion ermitteln wir den Zustand der LED und fügen ihn mit dem String `led_zustand` in den HTML-String ein.

Drei weitere Tags wurden noch benutzt: Mit dem <p>-Tag wird ein Absatz gekennzeichnet; am Ende eines solchen Absatz geht der Browser zu einer neuen Zeile über. <h2> sorgt für eine Schriftgröße, die etwas geringer als die bei <h1> ist. Und das -Tag dient zum Zentrieren. Das Browser-Fenster sieht dann so aus wie in Abb. 3.

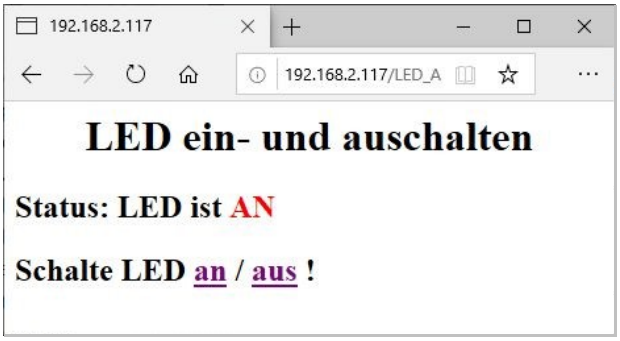

Abb. 3

Der Rest des Programms ist ähnlich aufgebaut wie beim Namen-Server des letzten Kapitels. Den GET-Parameter erhalten wir aus dem `request`-String wieder mit der Anweisung

```
getparam = request.split(' ')[1]
```

Damit kann die LED nun folgendermaßen gesteuert werden:

```
if getparam == '/LED_AN':
  led.value(1)

if getparam == '/LED_AUS':
  led.value(0)
```

Das vollständige Programm finden Sie unter dem Namen `sta_led_server_0.py`. Zur besseren Übersicht geben wir hier noch eine (verkürzte) Gliederung des Programms an:

```
1.    Importieren der benötigten Module und Klassen
2.    Wlan-Zugangsdaten
3.    Konfigurieren des Pins, an den die LED angeschlossen wird
```

```
4.    Definition der Funktionen getrequest, http_header und html
5.    Konfiguration des Wlans
6.    Erstellen des Server-Sockets
7.    Wiederholen:
      a.    Akzeptieren einer Client-Verbindung und Empfang des Requests
      b.    Steuern der LEDs (s. o.)
      c.    Senden des Response (HTTP-Header und HTML-String)
      d.    Schließen der Verbindung
```

Beachten Sie: Sobald Sie im Browser einen der beiden Links anklicken, wird das LED-Steuer-Kommando in Form des GET-Parameters zum Server gesendet. Dieser schaltet die LED befehlsgemäß an bzw. aus und sendet anschließend sofort die Webseite mit dem aktualisierten LED-Zustand an den Client (Browser). Der Anwender bemerkt nur Folgendes: Nach dem Anklicken einer der beiden Links wird die LED ein- bzw. ausgeschaltet und der neue LED-Zustand im Browser angezeigt.

Zwei Anregungen möchte ich noch geben:

1. Probieren Sie aus, ob sich die LED mit unserem Programm auch über die Adresszeile des Browsers (d. h. ohne die beiden Links) steuern lässt.

2. Erweitern Sie das Server-Programm so, dass damit zwei LEDs getrennt ein- und ausgeschaltet werden können. Hinweis: Dazu sind jetzt vier Links erforderlich.