# Übung 6: Webbased LED Control

# Ziele dieser Laborübung

* Verständnis der Funktionsweise von Flask als Web-Framework zur Erstellung von Webanwendungen und Schnittstellen zur Hardwaresteuerung.
* Implementierung der HTTP-POST-Methoden, um Webdaten an einen Server zu senden und physische Komponenten (z.B. LEDs) über das Web zu steuern.
* Verknüpfung von GPIO-Pins des Raspberry Pi mit Webtechnologien, um eine direkte Steuerung über eine Webschnittstelle zu realisieren.

Tipps

* Offizielle FLASK Doku
  + <https://flask.palletsprojects.com/en/3.0.x/>
* Tutorial FLASK on Rpi
  + <https://projects.raspberrypi.org/en/projects/python-web-server-with-flask/1>
* HTTP „POST“ Method with Flask
  + <https://www.geeksforgeeks.org/flask-http-methods-handle-get-post-requests/>
  + <https://www.w3schools.com/tags/att_button_form.asp>

# Aufgaben

1. Erkläre in eigenen Worten, was ein „Framework“ inwiefern sich Flask von einem Python-Modul (wie gpiozero) unterscheidet.

Ein Framework ist eine Sammlung von Bibliotheken, sie stellen eine standardisierte Basis bereit. Ein Phyton Modul ist eine Sammlung von Funktionen und Klassen.

Flask ist ein Framework, das eine umfassende Basis für Webentwicklungsprojekte bietet, während GPIO Zero ein spezialisiertes Modul für die Interaktion mit Hardware ist.

1. Beschreibe, welche HTTP Methode verwendet werden kann, um Daten von einer HTML-Seite an den Webserver zu senden. Gib hierfür einen Beispiel-HTML-Code an.

POST oder GET

<form action="/submit-data" method="POST">

<label for="name">Name:</label>

<input type="text" id="name" name="name">

<br>

<label for="email">E-Mail:</label>

<input type="email" id="email" name="email">

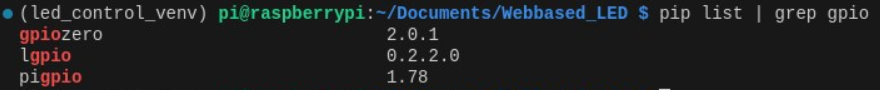
<br>

<button type="submit">Senden</button>

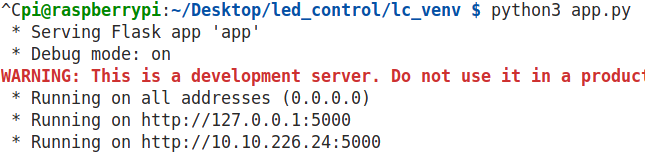
</form>

1. Erstelle eine venv „*led\_control\_venv“* am Rpi so, dass auf alle systemweit installierten Python-Pakete zugegriffen werden kann. Dadurch sind relevante Pakete (z.B. für die Verwendung der GPIO Pins) bereits funktionsfähig. Verwende folgendes Flag hierfür:
   * --system-site-packages
   * Kontrolliere, ob du die relevanten gpio-Pakete installiert hast:

pip list | grep gpio



1. Erstelle eine „minimale Flask-Applikation“: Es soll „Hello World“ ausgegeben werden, wenn du die IP-Adresse deines Rpi gemeinsam mit dem im Terminal angegebenen Port in der Adressleiste deines Browsers eingibst:

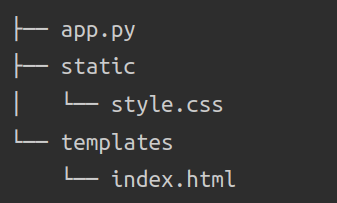


* Füge folgende Bedingung in *app.py* ein und erkläre, warum dies notwendig ist, um von anderen Netzwerkgeräten auf deinen Server zugreifen zu können:

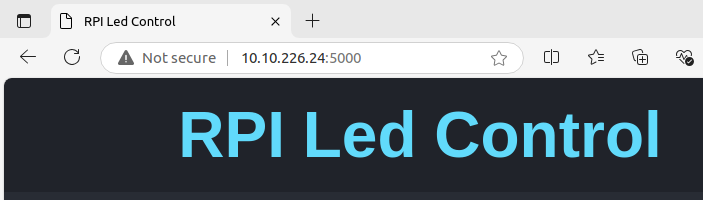
if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True, host='0.0.0.0')

1. Erstelle zwei Ordner „templates“ sowie „static“ in deinem Projektordner. Erstelle eine Datei „index.html“ im Ordner „*templates*“ sowie „styles.css“ im Ordner „*static*“. Achte darauf, dass die Ordnerstruktur folgendermaßen ausschaut:



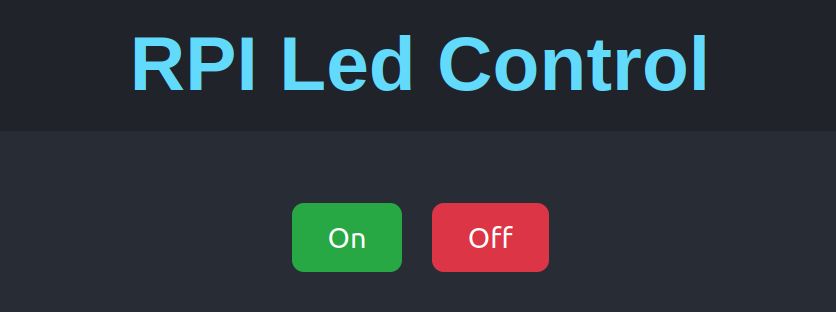
1. Erstelle den HTML- bzw. CSS Code, der (in etwa) folgende Website erzeugt:



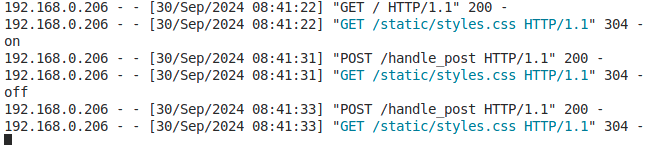
* Tipp: Du benötigst die **render\_template** Funktion aus dem flask-Modul, damit du auf deine HTML-Seite in app.py verweisen kannst (siehe Doku).

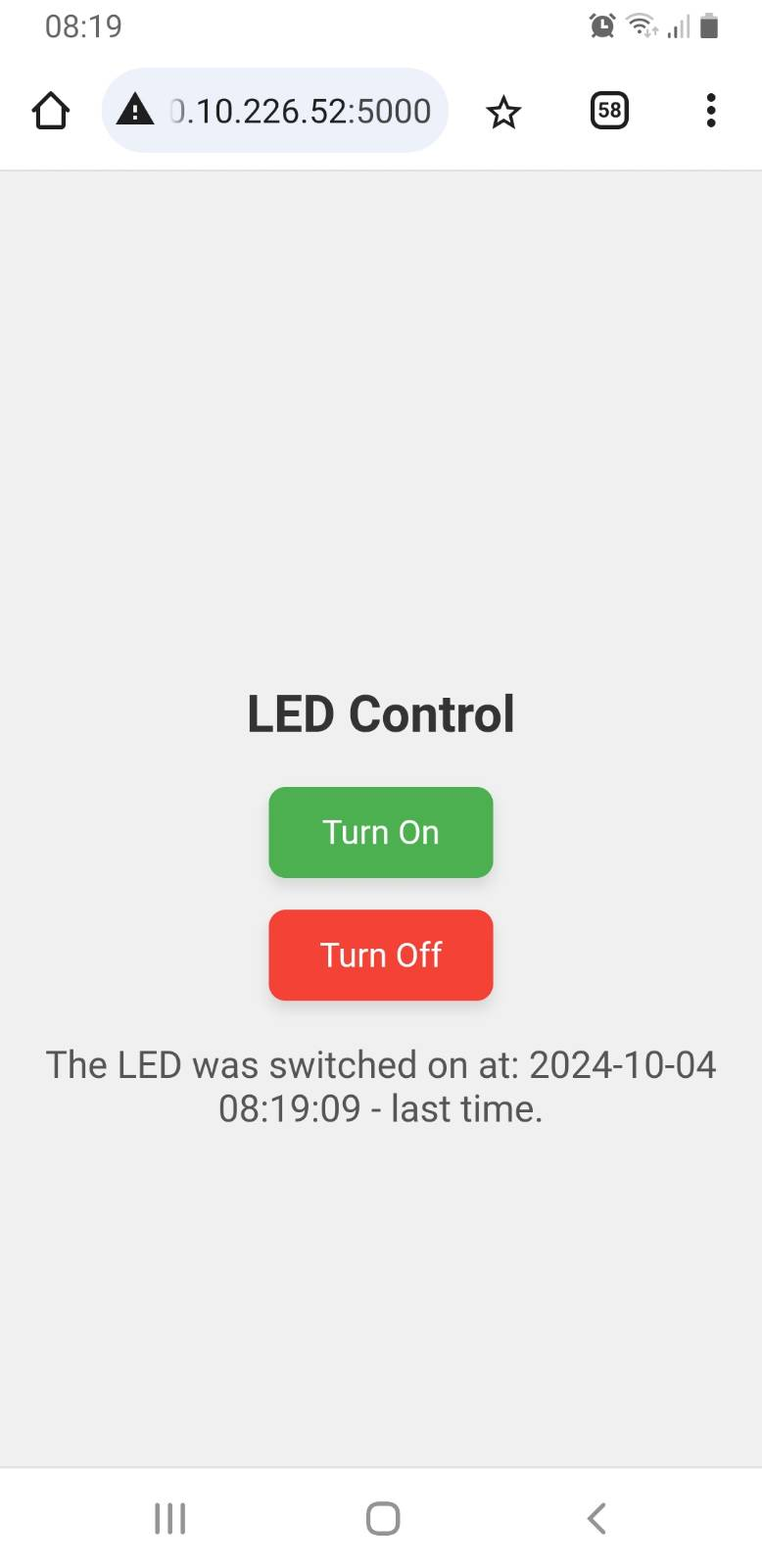
Ersetzt variablen in der template mit einem Wert

1. Füge nun zwei Buttons „ON“ sowie „OFF“ zu deiner HTML-Seite hinzu und verändere das Design in styles.css so, dass (in etwa) folgende Website erzeugt wird:



1. Verwende HTTP-POST, um den Wert „on“ (für den Button on) bzw. „off“ (für den Button off) in einer Variable „led\_status“ an deine Flask-Applikation zu senden, sobald die Benutzerin den jeweiligen Button im Browser klickt. Gib den Wert von „led\_status“ im Terminal immer dann aus, wenn ein Button geklickt wird.

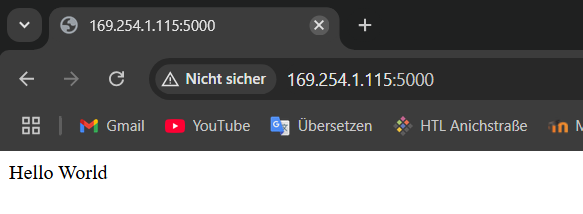


1. Baue eine Schaltung mit einer LED und einem Vorwiderstand auf und verbinde die Pins mit einem freien GPIO-Pin bzw. mit GND. Verändere den Code in app.py so, dass die LED bei entsprechendem Klick (im Browser am Client) auf den Button ein- bzw. ausgeschaltet wird.
2. Füge einen Timestamp zur Website hinzu: Jedes Mal, wenn die LED eingeschaltet wird, soll die aktuelle Zeit auf deiner Website erscheinen: 

**Optionale Aufgaben:**

* Verbinde zusätzlich den Sensor TSL2591 und verändere dein Framework so, dass die LED automatisch ausgeschaltet wird, wenn ein von dir festgelegter Lux-Wert überschritten wird. Unter diesem Schwellenwert soll das Ein- und Ausschalten im Browser funktionieren.

# Dokumentation



Ein Bild, das Text, Screenshot, Schrift, Design enthält.

Automatisch generierte Beschreibung

from flask import Flask, render\_template, redirect

from gpiozero import LED

app = Flask(\_\_name\_\_)

led = LED(17) # PIN ÃNDERNNNNNNNNNNNNNNNNNNNN

@app.route('/led\_on', methods=["POST"])

def led\_on():

print("turning led on")

# LED anschalten

led.on()

return redirect("/")

@app.route('/led\_off', methods=["POST"])

def led\_off():

print("turning led off")

# LED ausschalten

led.off()

return redirect("/")

@app.route('/')

def home():

return render\_template("index.html")

if \_\_name\_\_ == '\_\_main\_\_':

# Flask-App auf der IP-Adresse und einem Port starten

app.run(host='0.0.0.0', port=5000)

Ein Bild, das Text, Schrift, Logo, Screenshot enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift, Design enthält.

Automatisch generierte Beschreibung