

Übung 6: Webbased LED Control

Ziele dieser Laborübung

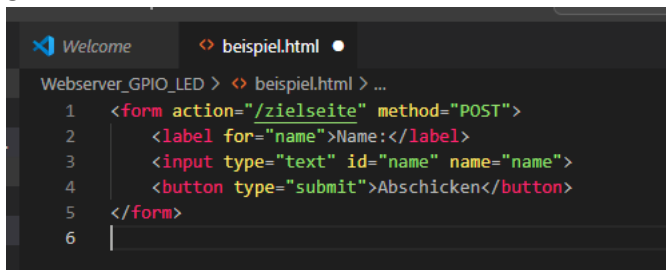
- Verständnis der Funktionsweise von **Flask** als Web-Framework zur Erstellung von Webanwendungen und Schnittstellen zur Hardwaresteuerung.
- Implementierung der **HTTP-POST-Methoden**, um Webdaten an einen Server zu senden und physische Komponenten (z.B. LEDs) über das Web zu steuern.
- Verknüpfung 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 sozusagen eine Library von Modulen, also eine Ansammlung von Libraries.
2. 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.
 - Um Daten von einer HTML-Seite an den Webserver zu senden, wird die HTTP-Methode POST genutzt.



```
Webserver_GPIO_LED > <> beispiel.html > ...
1 <form action="/zielseite" method="POST">
2   <label for="name">Name:</label>
3   <input type="text" id="name" name="name">
4   <button type="submit">Abschicken</button>
5 </form>
6 |
```



3. 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:

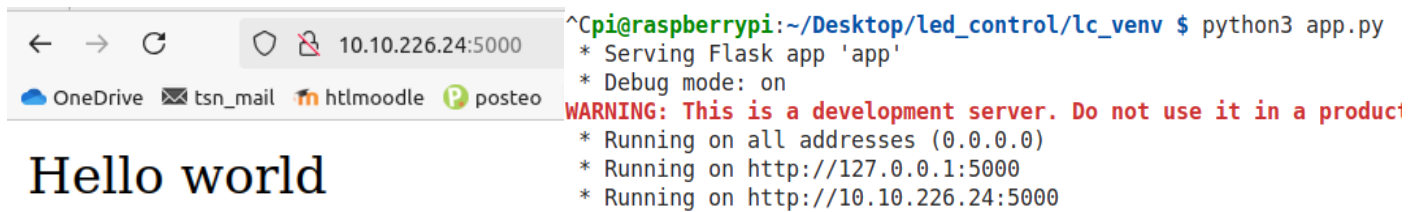
- --system-site-packages
- Kontrolliere, ob du die relevanten gpio-Pakete installiert hast:

pip list | grep gpio

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
pi@raspberrypi:~/Desktop $ cd Webserver_GPIO_LED/
pi@raspberrypi:~/Desktop/Webserver_GPIO_LED $ python3 -m venv led_control_venv --system-site-packages
pi@raspberrypi:~/Desktop/Webserver_GPIO_LED $ cd led_control_venv/
pi@raspberrypi:~/Desktop/Webserver_GPIO_LED/led_control_venv $ source bin/activate
(led_control_venv) pi@raspberrypi:~/Desktop/Webserver_GPIO_LED/led_control_venv $

(led_control_venv) pi@raspberrypi:~/Desktop/Webserver_GPIO_LED/led_control_venv $ pip list | grep gpio
gpiozero 2.0
lgpio 0.2.2.0
pigpio 1.78
```

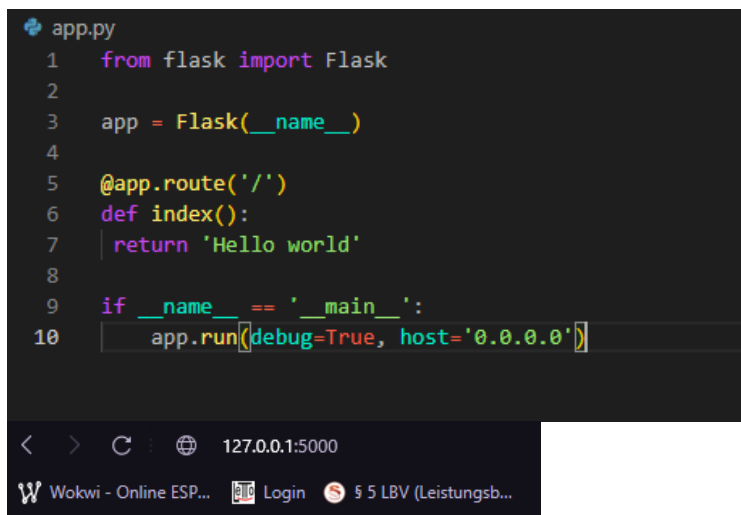
4. Erstelle eine „minimale Flask-Applikation“: Es soll „Hello World“ ausgegeben werden, wenn du die IP-Adresse deines Rpi gemeinsam mit dem im Terminal angegebenen Port in der Adressleiste deines Browsers eingibst:



- Füge folgende Bedingung in `app.py` ein und erkläre, warum dies notwendig ist, um von anderen Netzwerkgeräten auf deinen Server zugreifen zu können:
 - Es führt die App aus mit der Bedingung `host="0.0.0.0"`, sprich das alle im Netz darauf zugreifen können.

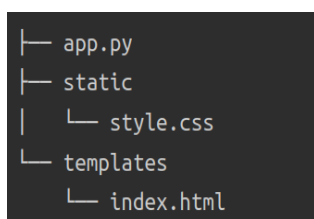
```
if __name__ == '__main__':
```

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



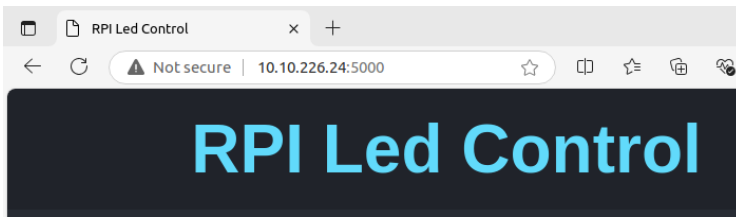
Hello world

5. 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 aussieht:

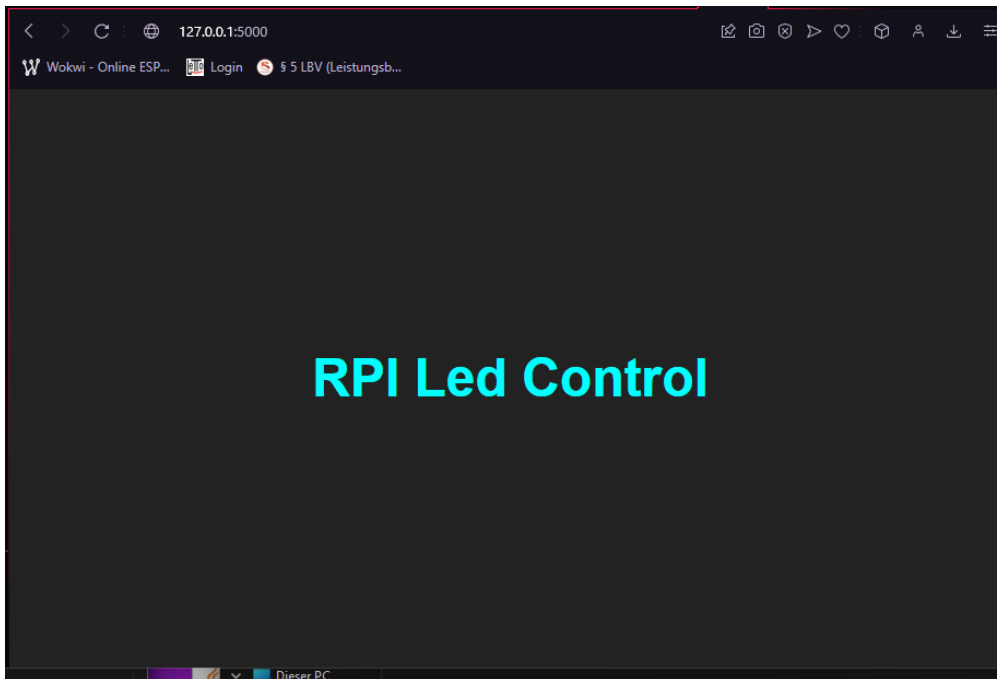




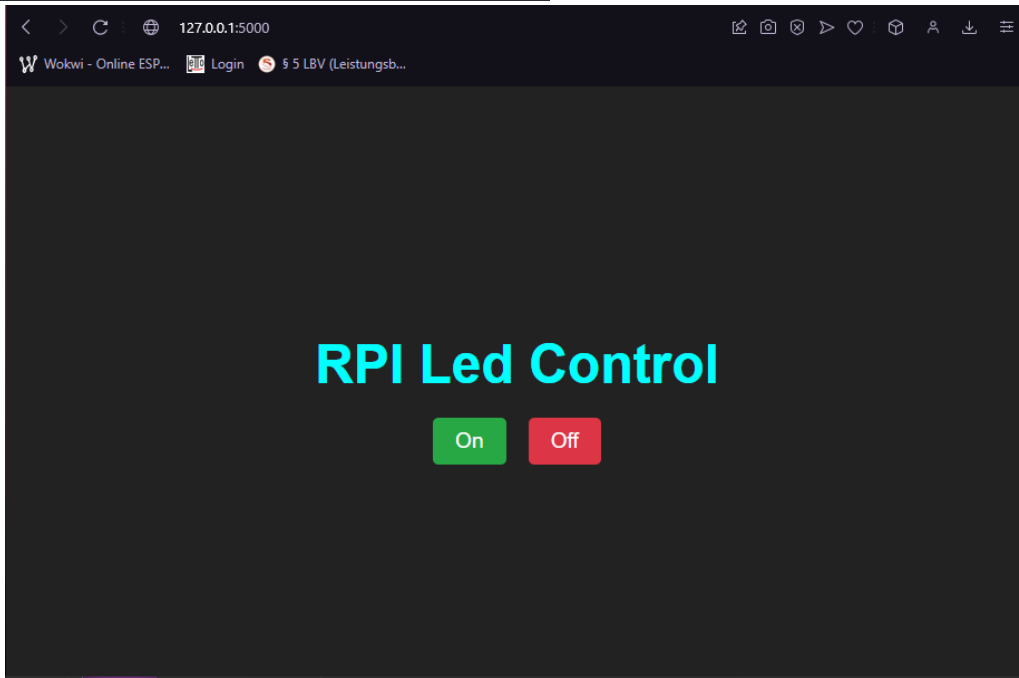
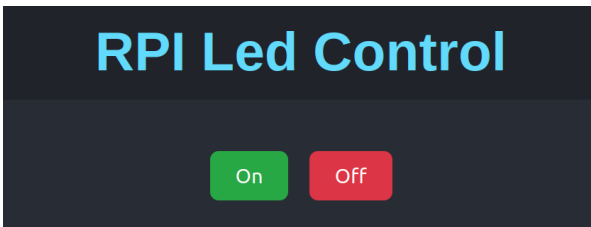
6. Erstelle den HTML- bzw. CSS Code, der (in etwa) folgende Website erzeugt:



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



7. 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:



8. 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.

```
192.168.0.206 - - [30/Sep/2024 08:41:22] "GET / HTTP/1.1" 200 -
192.168.0.206 - - [30/Sep/2024 08:41:22] "GET /static/styles.css HTTP/1.1" 304 -
on
192.168.0.206 - - [30/Sep/2024 08:41:31] "POST /handle_post HTTP/1.1" 200 -
192.168.0.206 - - [30/Sep/2024 08:41:31] "GET /static/styles.css HTTP/1.1" 304 -
off
192.168.0.206 - - [30/Sep/2024 08:41:33] "POST /handle_post HTTP/1.1" 200 -
192.168.0.206 - - [30/Sep/2024 08:41:33] "GET /static/styles.css HTTP/1.1" 304 -
█
```



```
Webserver_GPIO > app.py
1 from flask import Flask, request, render_template
2
3 app = Flask(__name__)
4
5 @app.route('/', methods=['GET', 'POST'])
6 def control_led():
7     led_status = None
8     if request.method == 'POST':
9         led_status = request.form.get('action') # "on" oder "off"
10        print(f"LED Status: {led_status}") # Ausgabe im Termin
11        return render_template('index.html')
12
13 if __name__ == '__main__':
14     app.run(debug=True, host='0.0.0.0')
15
```

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.10.218.131:5000

Press CTRL+C to quit

* Restarting with stat
* Debugger is active!
* Debugger PIN: 127-332-477

127.0.0.1 - - [29/Nov/2024 11:23:05] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [29/Nov/2024 11:23:05] "GET /static/style.css HTTP/1.1" 200 -
LED Status: on
127.0.0.1 - - [29/Nov/2024 11:23:07] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [29/Nov/2024 11:23:07] "GET /static/style.css HTTP/1.1" 304 -
LED Status: on
127.0.0.1 - - [29/Nov/2024 11:23:13] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [29/Nov/2024 11:23:13] "GET /static/style.css HTTP/1.1" 304 -
LED Status: off
127.0.0.1 - - [29/Nov/2024 11:23:16] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [29/Nov/2024 11:23:16] "GET /static/style.css HTTP/1.1" 304 -

9. 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.