Elektronik Kompendium (/)

Suchen

# Raspberry Pi Pico W: Onboard-LED über Webserver einschalten und ausschalten

Wir haben einen Raspberry Pi Pico W, können ihn vielleicht mit einem WLAN verbinden und dabei drängt sich die Frage auf, ob man jetzt nicht einfach die Onboard-LED aus der Ferne einschalten und ausschalten könnte?

Ja, das wollen wir hier tun. Die Idee besteht darin, die Onboard-LED von einem beliebigen Client im selben WLAN ein- und auszuschalten. Als Bedienoberfläche verwenden wir einfach eine Webseite, die vom Raspberry Pi Pico W bereitgestellt wird, die über einen Button zum Ein- und Ausschalten der Onboard-LED verfügt.
Der implementierte Webserver verfügt über eine Logik, um aus der übermittelten URL die Information zum Ein- und Ausschalten zu erkennen.

Der Programmcode ist so gestaltet, dass weitere GPIO-Ausgänge hinzugefügt und gesteuert werden können.

Hinweis: Diese Lösung entspricht keinen professionellen Ansprüchen. Es handelt sich um eine Demo-Implementierung zu Versuchs- und Experimentierzwecken.

---

Hardware-nahes Programmieren mit dem Mikrocontroller Raspberry Pi Pico W und MicroPython.

- Raspberry Pi Pico W mit gelöteten Stiftleisten
- Spezielles Steckbrett mit GPIO-Beschriftung
- Einführung ins Hardware-nahe Programmieren
- Schwerpunkte: WLAN, MQTT und Internet
- Deutschsprachige Anleitung als PDF-Datei zum Download

In unseren Online-Workshops bieten wir intensiven Erfahrungsaustausch in kleinen Gruppen und Unterstützung bei individuellen Problemen.



---

## Programmcode

Der Programmcode ist vergleichsweise umfangreich und dafür auch vollständig und sofort nutzbar. Es gibt nur 2 Parameter, die individuell konfiguriert werden MÜSSEN:

- **wlanSSID**: Das ist der WLAN-Name, mit dem sich der Raspberry Pi Pico W verbinden soll.
- **wlanPW**: Das ist das WLAN-Passwort für die Authentifizierung an dem zu verbindenden WLAN.

```
# Bibliotheken laden
import machine
import socket
import rp2
import utime as time

# Onboard-LED initialisieren
led = machine.Pin('LED', machine.Pin.OUT)

# WLAN-Konfiguration
wlanSSID = 'WLAN_NAME'
wlanPW = 'WLAN_PASSWORD'
rp2.country('DE')

# HTML-Datei
html = """<!doctype html><html lang="en"><head><meta charset="utf-8"><meta name="viewport" c

# Funktion: WLAN-Verbindung
def wlanConnect():
    import network
    wlan = network.WLAN(network.STA_IF)
    if not wlan.isconnected():
        print('WLAN-Verbindung herstellen')
        wlan.active(True)
        wlan.connect(wlanSSID, wlanPW)
        for i in range(10):
            if wlan.status() < 0 or wlan.status() >= 3:
                break
            print('.')
            time.sleep(1)
    if wlan.isconnected():
        print('WLAN-Verbindung hergestellt')
        netConfig = wlan.ifconfig()
        print('IPv4-Adresse:', netConfig[0])
        print()
        return netConfig[0]
    else:
        print('Keine WLAN-Verbindung')
        print('WLAN-Status:', wlan.status())
        print()
        return ''

# WLAN-Verbindung herstellen
ipv4 = wlanConnect()

# HTTP-Server starten
if ipv4 != '':
    print('Server starten')
    addr = socket.getaddrinfo(ipv4, 80)[0][-1]
    server = socket.socket()
    server.bind(addr)
    server.listen(1)
    print('Server hört auf', addr)
    print()
    print('Beenden mit STRG + C')
    print()

# Auf eingehende Verbindungen hören
while True:
    try:
        conn, addr = server.accept()
        print('HTTP-Request von Client', addr)
        request = conn.recv(1024)
        #print('Request:', request)
        request = str(request)
        request = request.split()
        print('URL:', request[1])
        # URL auswerten
        if request[1] == '/light/on':
            print('LED einschalten')
            led.value(1)
```

```
        elif request[1] == '/light/off':
            print('LED ausschalten')
            led.value(0)
        elif request[1] == '/light/toggle':
            print('LED umschalten')
            led.toggle()
        # LED-Status auswerten
        #print('LED-Status:', led.value())
        state_is = ''
        if led.value() == 1:
            state_is += '<p align="center"><b>LED ist AN</b> <a href="/light/off"><button>AU
        if led.value() == 0:
            state_is += '<p align="center"><b>LED ist AUS</b> <a href="/light/on"><button>AN
        # HTTP-Response erzeugen und senden
        response = html.replace('TEXT', state_is)
        conn.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
        conn.send(response)
        conn.close()
        print('HTTP-Response gesendet')
        print()
    except OSError as e:
        break
    except (KeyboardInterrupt):
        break

try: conn.close()
except NameError: pass
try: server.close()
except NameError: pass
print('Server beendet')
```

Nach der Ausführung des Programmcodes wird eine Verbindung zum WLAN aufgebaut und nach erfolgreicher Verbindung die IPv4-Konfiguration des Picos angezeigt. Die IPv4-Adresse ist die Adresse, die man in die Browser-Adresszeile eingeben muss, um die Onboard-LED des Picos über seinen gestarteten Webserver bedienen zu können.

Hinweis: In der Variable „html“ befindet sich die Vorlage für die HTML-Datei, die der Browser per HTTP-Response zugeschickt bekommt und darstellt. Bei Bedarf kann der Inhalt der Variable angepasst werden. Dabei ist zu beachten, dass bei Fehlern der Browser nichts oder etwas falsches darstellt.

## Troubleshooting

Folgende oder eine ähnliche Fehlermeldung erscheint in der Kommandozeile:

```
Traceback (most recent call last):
File "<stdin>", line 51, in <module>
OSError: [Errno 98] EADDRINUSE
```

Was ist das Problem? Wenn sich in der genannten Zeile (line) das Kommando „server.bind(addr)“ befindet, dann könnte das daran liegen, dass man diesen oder einen ähnlichen Programmcode schon einmal ausgeführt hat, der irgendwie noch aktiv im Speicher ist und die Ausführung dieses Programmcodes blockiert.
Man kann das dadurch lösen, in dem man die WLAN-Verbindung trennt, neu herstellt und anschließend den Programmcode erneut ausführt.