Raspberry Pi Pico W: WLAN-Stoppuhr mit Anzeige (TM1637)

Eine eigene Stoppuhr bauen? Wer braucht das schon? Aber, wie wäre es, wenn die Stoppuhr aus der Ferne über den Browser bedient wird. Das heißt, starten, stoppen und zurücksetzen der gezählten Zeit über WLAN. Einen Raspberry Pi Pico W könnte man mit einer Anzeige auf diese Weise aus der Ferne bedienen.

Folgende Probleme müssen wir lösen:

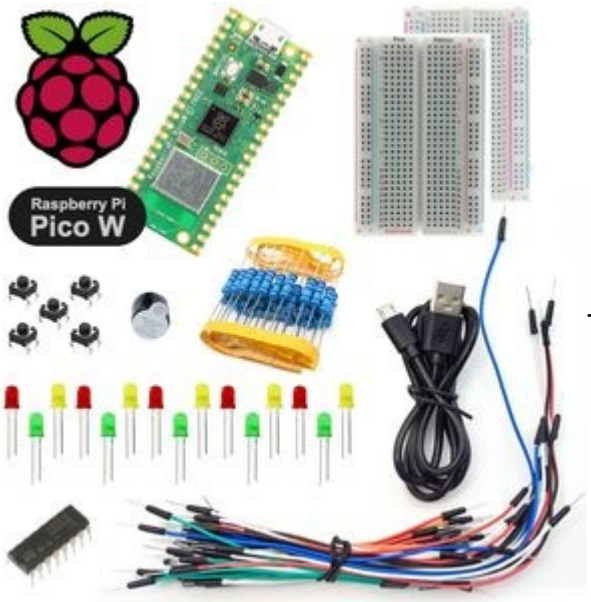
1. Anzeige der Zeit mit Minuten und Sekunden.
2. Bedienoberfläche zum Starten und Stoppen, sowie Zurückstellen auf „00:00“.
3. Zähler-Funktion mit Echtzeit-Fähigkeit.

Zur Anzeige der laufenden Zeit verwenden wir eine 4-fach 7-Segment-Anzeige vom Typ TM1637, die per I2C mit dem Raspberry Pi Pico verbunden ist. Für die Software-seitige Ansteuerung ist eine externe Bibliothek notwendig.

Zum Starten, Stoppen und Zurückstellen der Zeitanzeige lassen wir auf dem Raspberry Pi Pico W einen Webserver laufen, der es ermöglicht, die Stoppuhr zu bedienen.

Für die Echtzeit-Fähigkeit bedienen wir uns der Timer-Funktion von MicroPython, die vom Raspberry Pi Pico unterstützt wird.

NEU: Elektronik-Set Pico WLAN Edition



(/shop/elektronik-set/pico-wlan-edition)

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

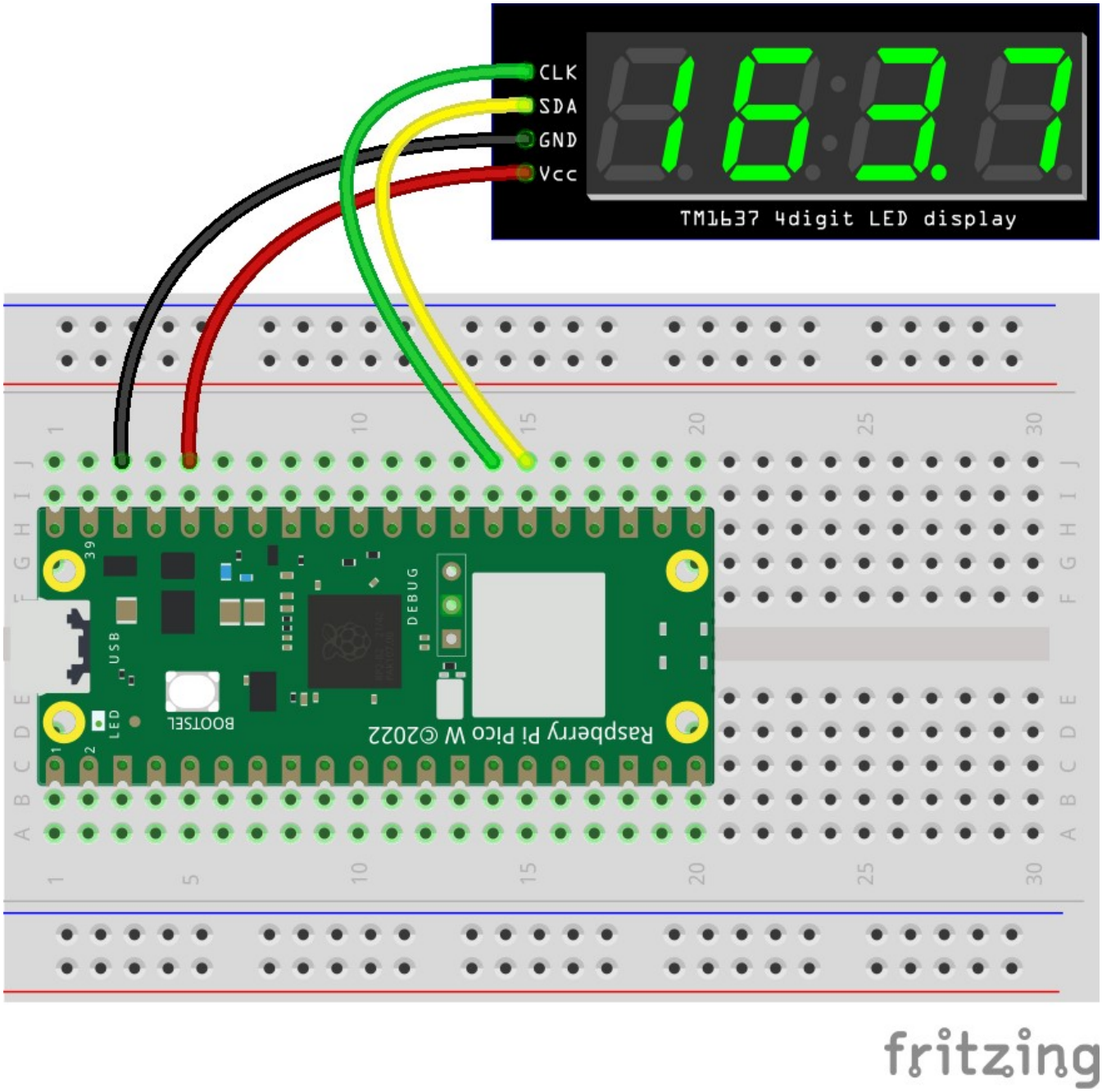
- Raspberry Pi Pico W mit gelöteten Stiftleisten
- Spezielles Steckbrett mit GPIO-Beschriftung
- Einführung ins Hardware-nahe Programmieren
- Schwerpunkte: WLAN, MQTT und Internet
- Deutschsprachige Anleitung als PDF-Datei zum Download

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

Elektronik-Set jetzt bestellen (/shop/elektronik-set/pico-wlan-edition)

Online-Workshop buchen (/service/events/)

Aufbau und Bauteile



- TM1637 mit 4-fach 7-Segment-Anzeige (../praxis/bauteil_tm1637-display.htm)
- GPIO-Belegung (Pinout) (2611051.htm)

Raspberry Pi Pico W		**TM1637**
Pin 38	GND	GND
Pin 36	VCC +3,3V	VCC
Pin 27	GPIO 21 (I2C SCL)	CLK
Pin 26	GPIO 20 (I2C SDA)	DIO

MicroPython-Bibliothek für TM1637

Zur Ansteuerung der 7-Segment-Anzeige TM1637 ist eine externe Bibliothek erforderlich, die heruntergeladen und mit dem Dateinamen „tm1637.py“ auf dem Raspberry Pi Pico

gespeichert werden muss.

- Dokumentation (https://github.com/mcauser/micropython-tm1637)
- Download (https://raw.githubusercontent.com/mcauser/micropython-tm1637/master/tm1637.py)

Programmcode

Es gibt 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.

Am Anfang des Programmcodes werden die Bibliotheken geladen und die Anzeige initialisiert. Danach folgen verschiedene Funktionen, die im Anschluss nacheinander aufgerufen werden. Im Hauptteil des Programms wird zuerst die WLAN-Verbindung hergestellt. Danach wird der Webserver initialisiert und gestartet.

In dieser Lösung wird nur die Anzahl der Sekunden vergangenen Sekunden gezählt. Erst bei der Darstellung der Zeit, wird aus der Anzahl der Sekunden die Minuten und Sekunden berechnet. MicroPython hat dafür eine eigene Funktion.

Wie können wir die Echtzeit-Fähigkeit des Zählers gewährleisten? Echtzeit bedeutet, dass eine Sekunde auch wirklich jeweils eine Sekunde dauert oder vergeht. Wenn wir also Sekunden zählen wollen, dann bedarf es einer Zeitlösung, die auch wirklich sekundengenau ist. Die einzige zuverlässige Lösung ist ein Timer, der jede Sekunde eine Funktion auslöst, in der der Zähler hochgezählt wird, die Anzeige ändert und zeitabhängige Funktionen ausführt.

```
# Bibliotheken laden
from machine import Pin, Timer
import network
import socket
import rp2
import utime as time
import tm1637

# Konfiguration: WLAN-Zugang
wlanSSID = 'WLANNAME'
wlanPW   = 'WLANPASSWORD'
rp2.country('DE')

# Display TM1637 initialisieren
display = tm1637.TM1637(clk=Pin(21), dio=Pin(20))

# Funktion: Ausgabe auf dem Display
def outputDisplay(counter, points):
    # Zähler in Minuten und Sekunden umrechnen und anzeigen
    timeValue = time.localtime(counter)
    secs = timeValue[5]
    if counter >= 3600: mins = int(counter/60)
    else: mins = timeValue[4]
    display.numbers(mins, secs, points)

# Funktion: Herunterzählen und zeitabhängige Bedingungen
def count(value):
    global counter
    global points
    global run
    # Blinkender Doppelpunkt im Wechsel
    if points == 0: points = 1
    else: points = 0
    # Zähler erhöhen
    counter += 1
    # höchster darstellbarer Wert erreicht
    if counter == 5940:
        clock.deinit()
        run = 0
        points = 1
        print('Ende')
    # Ausgabe auf dem Display
    outputDisplay(counter, points)

# Funktion: WLAN-Verbindung
def connectWLAN():
    wlan = network.WLAN(network.STA_IF)
    if not wlan.isconnected():
        print('WLAN-Verbindung herstellen')
        wlan.config(pm = 0xa11140)
        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)
    # WLAN-Verbindung prüfen
    if wlan.isconnected():
        print('WLAN-Verbindung hergestellt / WLAN-Status:', wlan.status())
        ipconfig = wlan.ifconfig()
        #print('IPv4-Adresse:', ipconfig[0])
    else:
        print('WLAN-Status:', wlan.status())
        raise RuntimeError('Keine WLAN-Verbindung')
    print()
    # IP-Adresse zurückgeben
    return ipconfig[0]

# Funktion: Webserver
def openSocket(ip):
    print('Server starten')
```

```
    address = (ip, 80)
    server = socket.socket()
    server.bind(address)
    server.listen(1)
    print('Server hört auf', ip)
    print('Beenden mit STRG + C')
    print()
    return server

# Funktion: HTML-Seite erzeugen
def getHTML(counter, run):
    html = """<!doctype html><html lang="en"><head><meta charset="utf-8"><meta name="viewpor
    if run == 0:
        html += """<p><a href="/toggle"><button>STARTEN</button></a></p>"""
        if counter > 0:
            html += """<p><a href="/reset"><button>Zurücksetzen</button></a></p>"""
    else:
        html += """<p><a href="/toggle"><button>STOPPEN</button></a></p>"""
    html += """</body></html>"""
    return str(html)

# Funktion: Webserver-Verbindungen
def webserver(server):
    global counter
    global run
    while True:
        try:
            client, addr = server.accept()
            #print('HTTP-Request von Client', addr)
            request = client.recv(1024)
            #print('Request:', request)
            request = str(request)
            try: url = request.split()[1]
            except IndexError: url = '/'
            #print('URL:', url)
            if url == '/toggle':
                if run == 1:
                    clock.deinit()
                    run = 0
                    outputDisplay(counter, 1)
                    print('Stop')
                else:
                    clock.init(freq=1, mode=Timer.PERIODIC, callback=count)
                    run = 1
                    print('Start')
            elif url == '/reset':
                counter = 0
                outputDisplay(counter, 1)
                print('Reset')
            elif url == '/':
                print('Startseite')
            else:
                print('Unbekannte URL:', url)
            response = getHTML(counter, run)
            client.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
            client.send(response)
            client.close()
            #print('HTTP-Response gesendet')
            print()
        except OSError as e:
            break
        except (KeyboardInterrupt):
            break
    # Client-Verbindung und Server beenden
    try: client.close()
    except NameError: pass
    try: server.close()
    except NameError: pass
    print('Server beendet')

# Hauptprogramm
```

```
# Startwert der Stoppuhr
counter = 0

# Countdown läuft (1) / läuft nicht (0)
run = 0

# Doppelpunkt
points = 1

# Initialisierung Timer für Countdown
clock = Timer()

# Ausgabe auf dem Display
outputDisplay(counter, points)

try:
    ip = connectWLAN()
    server = openSocket(ip)
    webserver(server)
except KeyboardInterrupt:
    machine.reset()
```

Stoppuhr testen und benutzen

Die Steuerung der Stoppuhr muss über den Browser erfolgen. Dazu kopiert man die IP-Adresse unter der der Webserver erreichbar ist aus der Kommandozeile in die Adresszeile des Webbrowsers eines PCs oder Smartphones, dass sich im selben WLAN befinden muss, wie der Raspberry Pi Pico W.

Auf der dargestellten Webseite wird ein Button angezeigt, der mit „STARTEN“ beschriftet ist, wenn die Stoppuhr nicht läuft. Wenn die Stoppuhr läuft, ist der Button mit „STOPPEN“ beschriftet. Mit dem Button „Zurücksetzen“ wird der interne Zähler auf 0 gesetzt und die Anzeige auf „00:00“ gestellt. Der Button „Zurücksetzen“ wird nur dann angezeigt, wenn die Stoppuhr nicht läuft und die Zeit nicht „00:00“ ist.

Erweiterung

Die Bedienung über den Browser ist eine interessante Lösung. Allerdings möchtest Du vielleicht lieber einen Taster zum Starten, Stoppen und zurücksetzen.

- Raspberry Pi Pico: Stoppuhr mit Anzeige (TM1637) (2711131.htm)

Weitere verwandte Themen:

- Raspberry Pi Pico: 4-fach 7-Segment-Anzeige programmieren (TM1637) (2711111.htm)
- Raspberry Pi Pico: Digitale Uhr mit Anzeige (TM1637) (2711121.htm)
- Raspberry Pi Pico: Countdown mit Zeitanzeige (TM1637) (2711151.htm)

Teilen: