9 Ein Temperatur-Server

In diesem Kapitel wollen wir einen Temperatur-Server herstellen. Die Idee ist: An den ESP32 schließen wir einen Temperatursensor an. Über einen Browser kann man dann die Temperatur anzeigen lassen.

Als Temperatursensor wollen wir den Baustein LM75A einsetzen. Diesen erhält man für wenige Euro schon fertig auf einer Platine montiert. Als I2C-Baustein besitzt er neben den Anschlüssen zur elektrischen Versorgung zwei Anschlüsse mit den Bezeichnungen **SDA** und **SCL** (s. u.). Über diese kann er mit unserem ESP32 Daten austauschen. Wie dies genau funktioniert, kann z. B. unter

Abb. 1

`www.g-heinrichs.de/attiny/I2C_Grundlagen_Attiny.pdf`

nachgelesen werden. Wir müssen hier nur wissen:

- Standardmäßig sind beim TTGO-Modul die Pins 21 und 22 für die I2C-Kommunikation vorgesehen. Wir schließen den LM75A folgendermaßen an:

LM75A	-	ESP32
SCL	-	22
SDA	-	21
GND	-	G
Vcc	-	3V

- An diese beiden Pins können mehrere I2C-Bausteine angeschlossen werden. Alle I2C-Bausteine besitzen eine so genannte **I2C-Adresse**. Diese besteht aus einer 7-Bit-Zahl.

- Der ESP32 steuert die I2C-Kommunikation: Er ist es, der über die I2C-Adresse den gewünschten I2C-Baustein aussucht; und er ist es auch, welcher bei dem adressierten Baustein Daten anfordert. In dieser Funktion bezeichnet man den ESP32 als **Master**. Der LM75A wird hier als so genannter **Slave** eingesetzt.

- Alle Slaves, welche an einen Master angeschlossen werden, müssen unterschiedliche Adressen haben; nur so können sie vom Master gezielt angesprochen werden. Bei dem LM75A-Baustein aus Abb. 1 kann man mit Hilfe von Lötbrücken die Adresse verändern; auf dieses Weise kann der Master auch mehrere I2C-Bausteine des gleichen Typs adressieren.

- Die Übertragung der Daten erfolgt seriell. Dabei steuert der Master mit einem **Taktsignal** an seinem SCL-Ausgang, wie rasch die einzelnen Daten-Bits über die SDA-Leitung wandern. Je höher die Taktfrequenz ist, desto schneller ist die Übertragung.

Micropython stellt im `machine`-Modul eine Klasse `I2C` zur Verfügung . Wir importieren sie und erzeugen damit ein I2C-Objekt und weisen es der Variablen `i2c` zu:

```
from machine import Pin, I2C
i2c = I2C(scl=Pin(22), sda=Pin(21), freq=100000)
addr = 0x48 # 7-Bit-Adresse im HEX-Format
```

Im Rahmen der Instanziierung haben wir auch die Pin-Zuordnung sowie die Taktfrequenz festgelegt. In der letzten Zeile haben wir die I2C-Adresse unseres I2C-Moduls in der Variablen `addr` gespeichert. In den Datenblättern von I2C-Bausteinen werden diese meist in hexadezimaler Form angegeben. Deswegen haben wir auch hier diese Form benutzt.

Nachdem der LM75 eine Temperaturmessung durchgeführt hat, hält er das Messergebnis in seinem Speicher in Form von 2 Bytes (beginnend mit der Speicheradresse 0) bereit. Wir können diese beiden Bytes mittels der Methode `readfrom_mem(addr, memaddr, nbytes)` von `i2c` auslesen:

```
rohwert = i2c.readfrom_mem(addr, 0, 2)
```

Hierbei stellt `rohwert` einen Byte-String dar, bestehend aus zwei Bytes. Das erste Byte (mit dem Index 0) gibt den ganzzahligen Anteil des Temperaturwertes an. Das zweite Byte (mit dem Index 1) gibt die "Nachkommastellen" des Temperaturwertes an, allerdings nicht in dezimaler Form. Vielmehr gibt es an, wie viele 256-tel noch zu dem ganzzahligen Wert hinzukommen. In dezimaler Form ist der Temperaturwert dann `rohwert[0] + rohwert[1]/256.`

Es bietet sich an, die Ermittlung des Temperaturwertes in eine Funktion `messwert()` auszulagern, wobei der Temperaturwert als String zurückgegeben wird:

```
def messwert():
  rohwert = i2c.readfrom_mem(adr, 0, 2)
  temp_wert = rohwert[0] + rohwert[1]/256
  return str(temp_wert)
```

An dieser Stelle müssen wir anmerken, dass bei unserer Betrachtung negative Temperaturwerte nicht berücksichtigt wurden. Hierzu sei auf das Datenblatt des LM75A verwiesen.

Kümmern wir uns nun um den HTML-Code, den der Server senden soll. Auch diesen wollen wir der Übersichtlichkeit halber in eine Funktion `html()` auslagern:

```
def html():
  return '''<html>
               <head>
                  <title>Temperaturmessung</title>
               </head>
             <body>
```

```
                <h1>Temperatur:
  '''                            + messwert() + '''
                      Grad Celsius</h1>
              </body>
            </html>
  '''
```

Es sei noch einmal darauf hingewiesen, dass das Einrücken nur der besseren Lesbarkeit dient; der Browser ignoriert es.

An dieser Stelle wollen wir uns auch gleich noch um den HTTP-Header vom Response kümmern; diesen hatten wir ja bislang einfach weggelassen. Auch diese Meldung wollen wir in eine Funktion auslagern:

```
def http_header():
  return '''HTTP/1.1 200 OK\r
Content-Type: text/html\r
Connection: close\r
\r
'''
```

Für unseren Temperatur-Server gehen wir von unserem ersten Webserver aus und fügen ein:

- die zur Konfiguration des i2c-Objektes benötigten Zeilen (samt Angabe der Adresse des LM75A)
- die Definition der Funktion `messwert()`
- die Definition der Funktion `html()`
- die Definition der Funktion `http_header()`

HTTP-Header und HTML-Dokument werden nun folgendermaßen an den Client gesendet:

```
connection.send(http_header())
connection.send(html())
```

Anstatt all diese Änderungen und Ergänzungen selbst vorzunehmen, können Sie auch direkt die Datei `sta_temp_server_0.py laden`. Vergessen Sie dann bitte nicht, Ihre eigenen Wlan-Daten einzutragen!

Zur besseren Übersicht hier noch einmal das vollständige Programm:

```
import network
import socket
from machine import Pin, I2C
from time import time
```

```
# Wlan-Zugangsdaten; hier Ihre Wlan-Daten eintragen:
ssid = 'ssid'
password = 'password'

# Konfiguration von I2C/LM75A
i2c = I2C(scl=Pin(22), sda=Pin(21), freq= 100000)
adr = 0x48  # 7-Bit-Adresse im HEX-Format

def messwert():
  rohwert = i2c.readfrom_mem(adr, 0, 2)
  temp_wert = rohwert[0] + rohwert[1]/256
  return str(temp_wert)

def html():
  return '''  <html>
              <head>
                <title>Temperaturmessung</title>
              </head>
                 <body>
                   <h1>Temperatur:
  '''                              + messwert() + '''
                       Grad Celsius</h1>
                 </body>
               </html>
  '''

def http_header():
  return '''HTTP/1.1 200 OK\r
Content-Type: text/html\r
Connection: close\r
\r
'''

# Konfiguration des Wlans:
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)
time0 = time()
while (not station.isconnected()) and (time() - time0 < 5):
  pass

if station.isconnected():
  # Konfiguration des Sockets
  print('Wlan connected')
  print(station.ifconfig()) # zur Information
  s=socket.socket() # Socket des Servers
  s.bind(('', 80))
  s.listen(5)
  print('Server listening')
  print('---------------------------------------')

  # Daten holen
  while True:
    connection, addr = s.accept()
    print("Client connected: ", addr)
```

```
    connection.send(http_header())
    connection.send(html())
    connection.close()

station.active(False)
print('Wlan deactivated')
print('Please reset!')
```

Wir starten nun das Programm; es zeigt wie immer zunächst im Terminal die IP-Adresse unseres Servers an (erstes Element im IP-Konfigurationstupel). Diese Adresse geben wir in unserem Browser ein. Wir erhalten dann eine Temperatur-Anzeige wie in Abb. 2. Hier sehen wir übrigens auch, dass der im Kopfteil mit dem **<title>-Tag** festgelegte **Titel** nicht im Browser-Fenster erscheint, sondern in der **Titelleiste (**hier oberhalb der Schaltflächenleiste**)**.

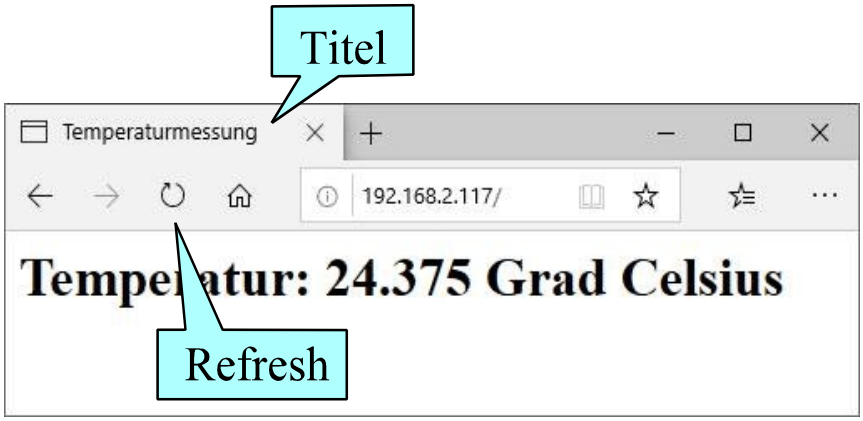

Abb. 2

Nun erwärmen wir für einige Sekunden den LM75A mit dem Finger und führen dann beim Browser eine Aktualisierung (Refresh) durch. (Bei Handy-Bowsern versteckt sich die entsprechende Schaltfäche oft hinter dem ⋮ -Icon.) Der Browser zeigt jetzt einen um ein paar Grad erhöhten Temperaturwert an. Weiteres Refreshen zeigt, dass das Abkühlen deutlich langsamer vonstatten geht.

Um die Raumtemperatur möglichst genau zu messen, sollte man darauf achten, dass der LM75A nicht zu nahe am PC steht; denn dieser gibt permanent Wärme ab und könnte den Messwert verfälschen. Zur Kontrolle kann man die Angabe des LM75A mit einem möglichst genauen Raumthermometer vergleichen. Bei meinem LM75A habe ich eine geringe (allerdings auch reproduzierbare) Abweichung festgestellt; diese habe ich dann durch einen Korrekturterm in der Temperatur-Funktion berücksichtigt. Im Datenblatt findet man Hinweise zur Genauigkeit des LM75A.