4 Upgrade für den Zeit-Dienst-Client

Kümmern wir uns zunächst darum, wie man das Problem mit der Verbindung zum AP beheben kann (vgl. Kap. 3). Dazu sorgen wir dafür, dass die Warte-Schleife nach dem Befehl

```
station.connect('ssid', 'password')
```

nicht mehr beliebig lange andauern kann. Wir ändern die Schleife so ab, dass sie maximal 5 Sekunden dauert:

```
time0 = time()
while (not station.isconnected()) and (time() - time0 < 5):
  pass
```

Die Funktion `time()` entnehmen wir zuvor mit

```
from time import time
```

dem Modul `time`. Sie ermittelt die Zeit (in Sekunden) seit dem letzten Boot-Vorgang des ESP32. Die Schleife wird nur so lange durchlaufen, wie die Verbindung mit dem AP noch nicht zustande gekommen ist ***und*** seit dem Start der Schleife noch keine 5 Sekunden vergangen sind.

Der Aufbau der Verbindung zum Server und der Empfang der Daten finden natürlich nur statt, wenn die Station zuvor erfolgreich mit dem AP verbunden werden konnte:

```
if station.isconnected():
  print('Wlan connected')
  server_addr = ('time.fu-berlin.de', 13)
  client=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  client.connect(server_addr)
  print('Socket connected')
  data = client.recv(1024) # max. Anz. der Zeichen
  print(str(data,'UTF-8'))
  client.close()
  print('Socket closed')
```

In jedem Fall werden die beiden letzten Zeilen ausgeführt:

```
station.active(False)
print('Wlan deactivated')
```

Testen Sie dieses Programm (`sta_client_time_1.py`) aus. Geben Sie dabei auch einmal ein falsches Passwort für Ihren AP ein.

Wie kann man nun dafür sorgen, dass die Requests in regelmäßigen Zeitabständen erfolgen? Dazu setzen wir alle für den Request nötigen Befehle in eine Endlos-Schleife:

```
  while True:
    client=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(server_addr)
    print('----------------------------')
    print('Socket connected')
    data = client.recv(1024) # max. Anz. der Zeichen
    print(str(data,'UTF-8'))
    client.close()
    print('Socket closed')
    sleep(20)
```

Dabei muss die Funktion `sleep` ebenfalls aus dem `time`-Modul importiert werden; `sleep(20)` lässt das ESP32-Modul 20 Sekunden warten. Kleinere Zeitabstände sollte man nicht wählen, weil sonst der Server unnötig belastet würde. Wenn Sie dieses Programm (`sta_client_time_2.py`) starten, sieht die Ausgabe im Terminal nach 1 Minute so aus:

```
WLAN connected
----------------------------
Socket connected
20 APR 2020 16:27:43 CEST

Socket closed
----------------------------
Socket connected
20 APR 2020 16:28:03 CEST

Socket closed
----------------------------
Socket connected
20 APR 2020 16:28:23 CEST

Socket closed
```

Nehmen wir jetzt das nächste Problem in Angriff: Die Verbindung der Sockets kommt nicht zustande oder wird gestört. Für diese Situation benutzen wir die `try - except - Konstruktion`:

```
try:
 tue_was_1()
 tue_was_2()
 ...
except:
 tue_was_anderes_1()
 tue_was_anderes_2()
 ...
...
```

Der ESP32 versucht, die Befehle `tue_was_1(), tue_was_2(), ...` in dem try-Block auszuführen. Tauchen dabei keine Fehler auf, werden die Befehle im except-Block ignoriert und das Programm fährt mit den Befehlen fort, die hinter dem except-Block stehen.

Gibt es aber einen Fehler (eine Exception) im `try`-Block, z. B. beim Befehl `tue_was_2()`, dann springt das Programm sofort in den `except`-Block und führt dessen Befehle aus.

```
while True:
  try:
    client=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(server_addr)
    print('----------------------------')
    print('Socket connected')
    data = client.recv(1024) # max. Anz. der Zeichen
    print(str(data,'UTF-8'))
    client.close()
    del client
    print('Socket closed')
    sleep(20)
  except:
    print('----------------------------')
    print('Exception')
    break
```

Mit Hilfe des Schlüsselworts `break` wird außerdem die Endlos-Schleife abgebrochen, wenn es zu einem Fehler im try-Block kommt.

Dieses Programm (`sta_client_time_3.py`) können Sie austesten, indem Sie für den Server eine falsche Adresse eingeben. Bei dem folgenden Terminal-Protokoll wurde gegen 17:16 Uhr der AP vom Stromnetz getrennt.

```
Socket connected
20 APR 2020 17:15:15 CEST

Socket closed
----------------------------
Socket connected
20 APR 2020 17:15:35 CEST

Socket closed
----------------------------
Socket connected
20 APR 2020 17:15:55 CEST

Socket closed
----------------------------
Exception
Wlan deactivated
```

Manch einen stört vielleicht noch die Leerzeile unter der Datums- und Zeitangabe. Um diese Leerzeile zu verhindern, ersetzen wir

```
print(str(data,'UTF-8'))
```

durch:

```
print(str(data,'UTF-8'), end = '')
```

Auf die Bedeutung dieses zweiten Parameters der `print`-Funktion werden wir in Kapitel 6 noch näher eingehen.