3 Ein Client für den Zeit-Dienst der FU Berlin (Terminal)

Unser ESP32 soll nun eine Verbindung zum Server der FU Berlin aufbauen und über den Port 13 das aktuelle Datum sowie die Zeit abfragen. Die erhaltenen Daten soll er dann im Terminal von Thonny ausgeben.

Um ins Internet zu gelangen, müssen wir zunächst den ESP32 als Station konfigurieren:

```
import network

station = network.WLAN(network.STA_IF) # network interface configuration
station.active(True)
station.connect('ssid', 'password') # hier Ihre Wlan-Daten eintragen!
while station.isconnected() == False:
  pass
print('WLAN connected')
print(station.ifconfig()) # nur zur Information
```

Diese Station wird dann durch die Funktion **`connect`** des Wlan-Objekts `station` mit Ihrem Access Point verbunden. Es kann etwas Zeit in Anspruch nehmen, bis diese Verbindung zustande gekommen ist. Die anstehende Abfrage beim Server in Berlin kann aber nur dann erfolgreich durchgeführt werden, sobald diese Verbindung zum Access Point hergestellt ist. Die Schleife

```
while station.isconnected() == False:
  pass
```

wartet deswegen so lange, bis die Funktion **`isconnected`** des Wlan-Objekts `station` den Wert `True` liefert. Anschließend werden im Terminal-Fenster die Meldung 'Wlan connected' sowie einige Details zur Konfiguration der Station ausgegeben.

Nun müssen wir auf dem ESP32 ein Socket-Objekt erzeugen, welches über das TCP-Protokoll eine Verbindung mit dem Server der FU Berlin aufnehmen kann. Dies geschieht folgendermaßen:

```
import socket

server_addr = ('time.fu-berlin.de', 13)
client=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(server_addr)
print('Socket connected')
```

Bei der Erzeugung des Objekts `client` verweisen der Parameter **`AF_INET`** auf die Adress-Familie IPv4 (s. Kapitel 4) und der Parameter **`SOCK_STREAM`** auf das TCP-Protokoll. Durch die Funktion `connect` des Socket-Objekts `client` wird der Socket des Clients mit dem Socket des Servers (mit der angegebenen Adresse inkl. Portnummer) verbunden. Dabei sendet der Client auch seine IP-Adresse mitsamt einer beliebigen freien Portnummer an den Server.

Weil an dem Port 13 des Servers nur Zeitabfragen bearbeitet werden, braucht der Server keine (weiteren) Informationen vom Client. Dies entspricht bei unserem Firmen-Modell einer Anfrage über die Dienstleistungsnummer 555. Hier benötigt die entsprechende Abteilung neben der Adresse auch keine weiteren Informationen vom Kunden. Wie die Abteilung "Geschäftsbedingungen" daraufhin die gewünschten Geschäftsbedingungen automatisch an den Kunden schickt, so sendet nun der Server die Datum-Zeit-Information an den Client. Dieser empfängt die Daten mit Hilfe der Funktion `recv` des Objekts `client`:

```
data = client.recv(1024) # max. Anz. der Bytes
print(str(data,'UTF-8'))
client.close()
print('Socket closed')
station.active(False)
print('Wlan deactivated')
```

Die recv-Methode gibt die empfangenen Daten als Byte-String zurück. Diese müssen wir vor der Ausgabe durch die `print`-Funktion mit der `str`-Funktion zunächst noch in eine Zeichenkette umwandeln (vgl. Kapitel E.5).

Zu guter Letzt werden die Socket-Verbindung durch den Client mit der `close`-Funktion abgebaut und das Wlan getrennt und deaktiviert.

Wir geben das Programm in unsere Entwicklungsumgebung Thonny mit dem eigenen(!) SSID und dem zugehörigen Passwort ein, schließen das ESP32-Modul an und starten das Programm (`sta_client_time_0.py`). Wenn eine Verbindung hergestellt werden konnte, erhalten wir im Terminal-Fenster die folgenden Meldungen:

```
WLAN connected
('192.168.2.117', '255.255.255.0', '192.168.2.1', '192.168.2.1')
Socket connected
20 APR 2020 12:45:43 CEST

Socket closed
Wlan deactivated
```

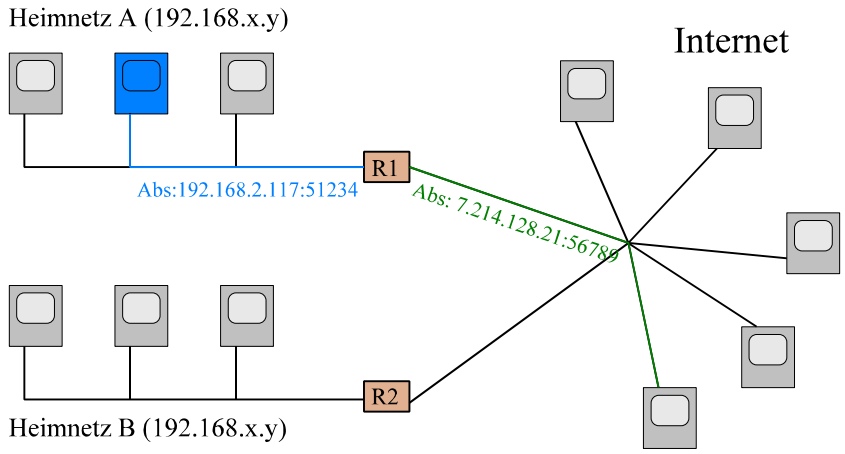
In der 2. Zeile finden wir an erster Stelle die IP-Adresse unseres ESP32. Für Insider: Bei den restlichen Angaben handelt es sich um die Subnetz-Maske sowie die Adressen des Gateways und des **DNS**-Server (**D**omain **N**ame **S**ystem). Die Angaben in dieser Zeile können bei Ihnen anders lauten.

In der 4. Zeile stehen schließlich die empfangenen Daten: das aktuelle Datum und die Uhrzeit.

Ein Blick hinter die Kulissen: **Lokale Netz-Adressen und Internet-Adressen (NAT)**

Die Struktur der IPv4-Adressen bringt es mit sich, dass höchstens 256^4 = 4.294.967.296 verschiedene Adressen zur Verfügung stehen. Deswegen können auch nicht allen Geräten, die einem Netzwerk angeschlossen sind, eine **weltweit einzigartige** Adresse gegeben werden. Aus diesem Grund benutzt man neben dem weltweiten Internet eine große Anzahl von kleineren lokalen Netzen. Im Heim-Bereich spricht man dann auch von Heimnetzen.

Die verschiedenen Heimnetze sind nicht (direkt) miteinander verbunden. Deswegen können die IP-Adressen in diesen Heimnetzen auch alle vom gleichen Typ sein, z. B. (192.168.2.x). In meinem Heimnetzwerk hat mein ESP32 die Adresse (192.168.2.117).



Wie kann dann ein Kontakt zwischen dem Client im Heimnetz und dem Server im Internet zustande kommen? Nun, der Client sendet seine Daten an den Router. Dieser besitzt eine Verbindung zum Internet. Von Seiten des Heimnetzes hat er die IP-Adresse 192.168.2.1. Von Seiten des Internets besitzt er – sagen wir einmal – die Adresse (7.214.128.21). Diese IP-Adresse wurde meinem Router von meinem Provider zugewiesen.

Wenn nun der Client eine Anfrage tätigt, gibt er dabei als Absender seine ***Heimnetz***-IP-Adresse 192.168.2.117 zusammen mit einer Port-Nummer, sagen wir 51234, an. Die Daten gelangen zunächst zum Router; der schickt diese weiter ins Internet. Dabei benutzt er als Absender nun aber seine ***Internet***-IP-Adresse und eine gerade freie Portnummer, sagen wir 56789. Entscheidend ist nun: Der Router speichert die Zuordnung

192.168.2.117:51234 (Heimnetz) — 7.214.128.21:56789 (Internet)

Der Server sendet nun seine Antwort an die Socket-Adresse 7.214.128.21:56789; die Daten gelangen so zu meinem Router. Und der kann sie dann dank der gespeicherten Zuordnung im Heimnetz an den Client 192.168.2.117 weiterleiten. Das hier beschriebene Verfahren nennt sich **N**etwork **A**ddress **T**ranslation (**NAT**).

Es mag nun sein, dass unser Programm gar keine Meldung im Terminal-Fenster anzeigt. In diesem Fall gelingt es wohl nicht, eine Verbindung mit dem AP aufzubauen; der ESP32 gelangt so in eine Endlos-Schleife. Möglicherweise kann aber auch gar keine Verbindung zum Server aufgebaut werden, obschon unsere Station mit dem AP verbunden worden ist. Dies ist z. B. der Fall, wenn die Adresse des Servers nicht korrekt angegeben wurde. Dann kann unser Socket sich nicht mit dem Server verbinden und wir erhalten im Terminal eine Fehlermeldung. Was auch möglich ist: Die beiden Sockets werden verbunden, aber diese Verbindung reißt ab, bevor oder während die Daten an den Client gesendet werden. Wünschenswert wäre es vielleicht auch, wenn die Datum-Zeit-Angaben automatisch in bestimmten Zeitabständen angefordert und angezeigt würden.

Wie unser Programm in Hinblick auf all diese Probleme ergänzt werden kann, davon handelt das nächste Kapitel.