1 Wlan-Scan

Wenn wir zuhause mit dem Handy (Notepad oder Notebook) ins Internet gehen, benutzen wir dazu meist ein Wlan. **Wlan** steht für **W**ireless **L**ocal **A**rea **N**etwork (drahtloses lokales Netzwerk). Über ein solches Netzwerk nimmt unser Handy Kontakt mit unserem Router auf, über den es dann Informationen mit anderen Computern im Internet austauschen kann.

Ein Wlan-Router stellt einen so genannten **Access Point** (kurz: **AP**) bereit. Dieser Access Point sendet gewöhnlich in regelmäßigen Zeitabständen kleine Datenpakete, so genannte **Beacons** (Leuchtfeuer), aus, welches die charakteristischen Parameter seines Funknetzes beinhalten. Zu diesen Parametern gehört z. B. der sogenannte **SSID** (**S**ervice **S**et **Id**entifier), also der Name des Funknetzes.

Damit unser Handy den Weg ins Internet findet, muss es mit "unserem Wlan", d. h. dem Wlan, welches durch den Access Point unseres Routers gebildet wird, verbunden werden. Im Gegensatz zum Access Point bildet das Handy normalerweise nicht selbst ein Funknetz mit eigener Kennung; es kann aber über seinen eingebauten Wlan-Adapter die Beacons von Access Points empfangen und deren Funknetzwerke identifizieren.

In Abb. 1 sieht man einen Teil der Funknetze, welche mein Handy zuhause anzeigt. Es kann eine gewisse Zeit dauern, bis die Liste vollständig ist.

Wollen wir unser Handy mit einem dieser Netzwerke verbinden, dann brauchen wir meist nur mit dem Finger auf den Namen tippen und ggf. noch das zugehörige Passwort eingeben.

Geräte, welche wie unser Handy bestehende Funknetze erkennen und sich mit ihnen verbinden können, bezeichnen wir in diesem Skript als eine **Station**.

Übrigens kann unser Handy ausnahmsweise auch als Access Point dienen; dazu muss es allerdings entsprechend eingestellt werden (Hot Spot).

Unser ESP32 kann nun in beiden Funktionen, als eine Station oder als ein Access Point oder auch als beides gleichzeitig, eingesetzt werden. Je nach gewünschter Funktion muss der ESP32 unterschiedlich konfiguriert werden.

Abb. 1

Kommen wir nun zur ersten praktischen Anwendung: Unser ESP32 soll die Umgebung nach allen (feststellbaren) Netzwerken durchsuchen und sie im Terminal auflisten; ein solcher Suchvorgang wird auch als **Scan** bezeichnet. Die Programmierung ist denkbar einfach, weil Python uns in Form des **network-Moduls** ein mächtiges Werkzeug an die Hand gibt:

```
import network

w = network.WLAN(network.STA_IF)
w.active(True)

print('------------gefundene Netzwerke-------------')
nets=w.scan()
for net in nets:
  print(net)
```

Mit

```
w = network.WLAN(network.STA_IF)
```

wird ein Wlan-Interface-Objekt vom Typ "**Sta**tion" erzeugt. (Wollten wir hingegen ein Wlan-Objekt vom Typ "**AP**" erzeugen, müsste hier als Argument `network.AP_IF` stehen.) Dieses Objekt `w` stellt nun eine Reihe von Funktionen bzw. Methoden zur Verfügung; hier eine Auswahl:

Funktion	Bedeutung
`active(p)`	Ist der Parameter `p True` bzw. `False`, wird das Interface ein- bzw. ausgeschaltet. Gibt es keinen Parameter, dann liefert `active()` den Rückgabewert `True` bzw. `False`, je nachdem ob das Interface aktiv ist oder nicht.
`connect(ssid,pw)`	verbindet die Station mit dem angegebenen Access Point
`scan()`	liefert als Rückgabewert eine Liste der gefundenen Wlans. Jedes Element der Liste besteht seinerseits aus einem Tupel, welches neben dem Namen des Netzwerks weitere Eigenschaften angibt (s. u.)

Im nächsten Schritt wird das Wlan-Interface mit der Anweisung

```
w.active(True)
```

eingeschaltet. Anschließend wird dann mit der Zeile

```
nets=w.scan()
```

die Liste mit den Netzwerke-Informationen in der Variablen `nets` gespeichert.

Durch die Schleife

```
for net in nets:
  print(net)
```

werden alle Elemente dieser Liste nun zeilenweise im Terminal-Bereich ausgegeben.

Das Ergebnis des Scans kann z. B. so aussehen:

```
>>> exec(open('wlan_scan_2.py').read())
-------------gefundene Netzwerke--------------------
(b'mipa_devolo', b'\xf4\x06\x8d\xc6\x98y', 3, -51, 3, False)
(b'WLAN-100684', b'd\xcc"[r\xbc', 6, -69, 3, False)
(b'EasyBox-68CB52', b'L\t\xd4h\xcb\xca', 1, -72, 4, False)
(b'WLAN-100684', b'x\xdd\x12\xb6\x01\n', 6, -76, 3, False)
(b'Telekom_FON', b'x\xdd\x12\xb6\x01\x0c', 6, -77, 0, False)
(b'mipamipa', b'L\t\xd4.\xd6b', 7, -77, 3, False)
(b'unten', b'\xf4\x06\x8d\xc6\x92\xd1', 10, -78, 4, False)
(b'DIRECT-c0-HP M277 LaserJet', b'BI\x0ft\xc1\xc0', 6, -81, 3, False)
(b'DIRECT-07-HP OfficeJet 5200', b'\x12\xe7\xc6\xd4\x8a\x07', 1, -83, 3, False)
(b'Telekom_FON', b'D\xfe;\xc6\x8b\x1f', 6, -86, 0, False)
(b'WLAN-186984', b'D\xfe;\xc6\x8b\x1d', 6, -91, 3, False)
>>>
```

Abb. 2

Übrigens: Vor der Ausgabe der Liste gibt das Python-System selbstständig zusätzliche Informationen aus; diese kann man mit den Zeilen

```
import esp
esp.osdebug(None)
```

unterdrücken.

Diese Liste soll nun eingehend betrachtet werden: Zunächst stellen wir fest, dass die Elemente dieser Liste aus Tupeln besteht: diese bestehen jeweils aus zwei Byte-Strings, einigen Zahlen und einem Wahrheitswert. Jedes dieser Tupel beschreibt offensichtlich ein Funknetz.

Schauen wir uns das Tupel des ersten Funknetzes einmal genauer an: An erster Stelle steht der Name des Funknetzes, der SSID, und zwar als Byte-String (d. h. als Objekt vom Typ `bytes`). In diesem Fall handelt es sich also um ein Funknetz mit dem Namen “mipa_devolo”.

Es folgt der **Basic Service Set Identifier (BSSID)**; hierbei handelt es sich um einen 48-Bit-Identifikator des Access Points. Dieser BSSID wird manchmal auch als MAC-Adresse des Access Points bezeichnet. Der BSSID ist hier als Byte-String angegeben; hexadezimal lautet er f4:06:8d:c6:98:79 (vgl. Abb. 3).

Das dritte Element des Tupels ist die Zahl 3. Diese bezeichnet die **Kanalnummer** des Funknetzes. Ähnlich wie beim Fernseher ist auch der Frequenz-Bereich beim Funknetz in Kanäle unterteilt (vgl. Abb. 3).

Die nächste Zahl im Tupel gibt den **RSSI**-Wert (**R**eceived **S**ignal **S**trength **I**ndicator) an; dieser beschreibt die **Signalstärke** in der Einheit dBm (Dezibel Milliwatt). (Für Fachleute: Es handelt sich hier um ein logarithmisches Dämpfungsmaß, deswegen das Minuszeichen.) Der Betrag dieses Wertes ist um so kleiner, je stärker das Signal ist. In der Abb. 3 sieht man, dass für das Netz "mipa-devolo" vom Handy die Signalstärke -36 dBm gemessen wurde. Hier war das Signal offensichtlich noch etwas stärker als bei der Messung in Abb. 2. In der Tat befand sich bei der Messung in Abb. 3 das Handy näher am AP als der ESP32 bei der Messung von Abb. 2. Man beachte, dass die gemessene Signalstärke neben der Sendeleistung und dem Abstand auch von anderen Parametern wie z. B. der Empfindlichkeit des Empfängers und der Ausrichtung seiner Antenne abhängt.

Nebenbei: Die vom ESP32 ausgegebene Wlan-Liste ist nach der Signalstärke geordnet: das Netz mit dem stärksten Signal wird als erstes aufgelistet, das mit dem schwächsten als letztes.

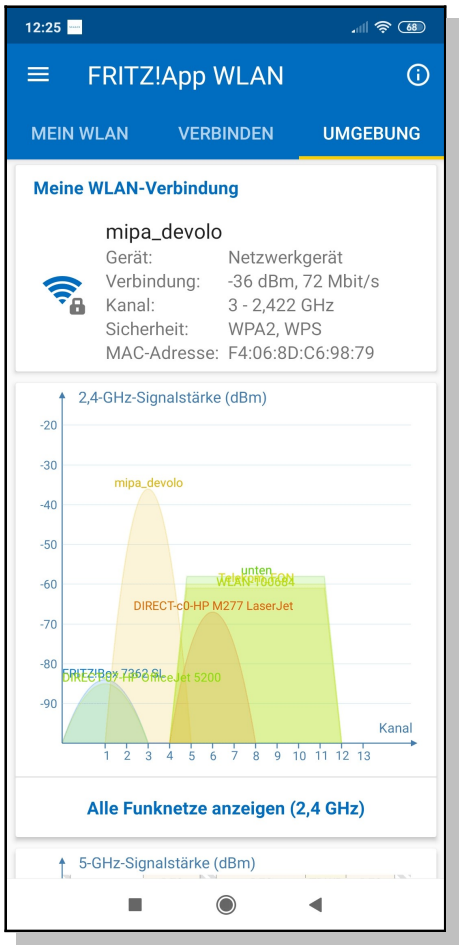

Abb. 3

Das nächste Element in unserer Liste ist eine Zahl, welche den **Authentifizierungs-Typ** angibt:

- 0 – open
- 1 – WEP
- 2 – WPA-PSK
- 3 – WPA2-PSK
- 4 – WPA/WPA2-PSK

Erscheint hier die Zahl 0, handelt es sich um ein offenes Netz (keine Verschlüsselung). In unserem Fall erfolgt eine Verschlüsselung gemäß WPA2-PSK. (WPA2 kennzeichnet das benutzte Verschlüsselungsverfahren. Die Ergänzung PSK weist darauf hin, dass hier für die Verschlüsselung ein **P**re **S**hared **K**ey, also ein geteilter Geheimcode (das Wlan-Passwort) eingesetzt wird, welchen die Nutzer und der AP kennen.

Das letzte Element in unserem Tupel beschreibt, ob das Netzwerk verborgen ("hidden") ist; das ist hier nicht der Fall. Verborgene Netzwerke werden durch unsere `scan`-Funktion nicht erfasst.