## Das ESP-Modul ohne WLAN

Das ESP-Modul wurde mit der MicroPython Firmware neu geflasht.
Bei früheren WLAN-Experimenten hatte es die IP-Nummer 182.168.2.115.
Ein Scan mit dem „Advanced IP-Scanner“ zeigte damals das Modul wie folgt an.

| Status | Name | IP | Hersteller | MAC-Adresse | Kommentare |
|---|---|---|---|---|---|
| | ESP_FDEDCB | 192.168.2.115 | | 50:02:91:FD:ED:CB | |

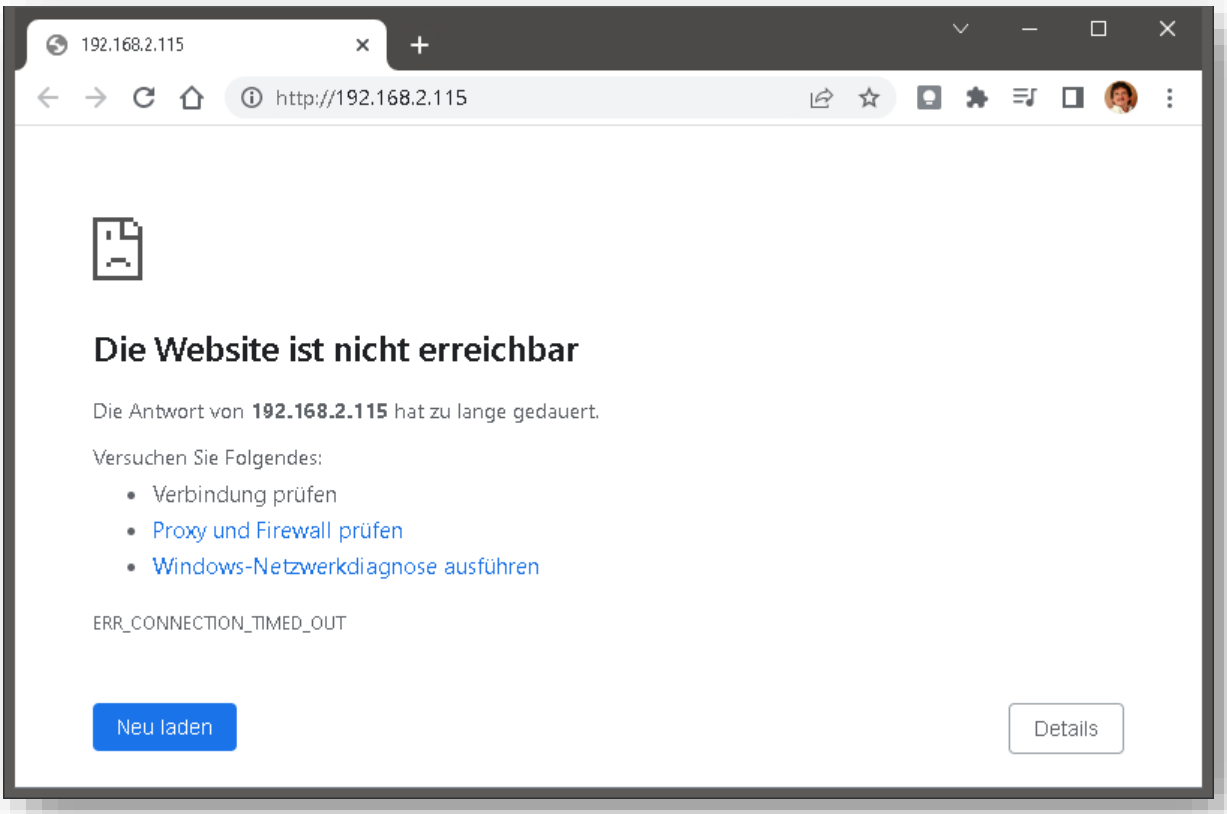
Ein neuer Scan mit dem dem „Advanced IP-Scanner“ und natürlich auch der Test per „ping“ können das ESP-Modul nicht erreichen.

```
C:\Users\l5889>ping 192.168.2.115

Ping wird ausgeführt für 192.168.2.115 mit 32 Bytes Daten:
Zeitüberschreitung der Anforderung.
Zeitüberschreitung der Anforderung.
Zeitüberschreitung der Anforderung.
Zeitüberschreitung der Anforderung.

Ping-Statistik für 192.168.2.115:
    Pakete: Gesendet = 4, Empfangen = 0, Verloren = 4
    (100% Verlust),
```

Der Aufruf im Webbrowser zeigt ebenfalls, dass sich unter dieser Adresse nicht tut.




Niemand will unter dieser Adresse mit dem Webbrowser „reden“.

Mit dem Start des folgenden Programmes wird das ESP-Modul mit dem WLAN Access Point vebunden.

[ Station+WebServer.py ] *

```
1  import network, time
2
3  wlan_sta = network.WLAN(network.STA_IF)
4  wlan_sta.active(True)
5
6  wlan_sta.connect("WLAN-Name","WLAN-Passwort")
7
8  print("Verbinde mit WLAN ..",end="")
9  while not wlan_sta.isconnected():
10   print(". ",end="")
11   time.sleep(0.2)
12 print("\nMit WLAN verbunden")
13
14 print(wlan_sta.ifconfig())
15 print(wlan_sta.ifconfig()[0])
16 print("\nhttp://"+wlan_sta.ifconfig()[0]+"\n\n")
```

Nach erfolgreichem Start des Programms meldet sich das ESP-Modul in der Kommandozeile:

```
Verbinde mit WLAN ..
Mit WLAN verbunden
('192.168.2.115', '255.255.255.0', '192.168.2.1', '192.168.2.1')
192.168.2.115

http://192.168.2.115
```

und informiert über die aktuell zugeteilte IP-Nummer, sowie Subnetzmaske, Gateway und DNS-Server.
Der Test per „ping“ in der Befehlszeile deines PCs und im Webbrowser zeigen folgende Reaktionen:

```
C:\Users\l5889>ping 192.168.2.115

Ping wird ausgeführt für 192.168.2.115 mit 32 Bytes Daten:
Antwort von 192.168.2.115: Bytes=32 Zeit=3ms TTL=255
Antwort von 192.168.2.115: Bytes=32 Zeit=3ms TTL=255
Antwort von 192.168.2.115: Bytes=32 Zeit=3ms TTL=255
Antwort von 192.168.2.115: Bytes=32 Zeit=3ms TTL=255

Ping-Statistik für 192.168.2.115:
    Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0
    (0% Verlust),
Ca. Zeitangaben in Millisek.:
    Minimum = 3ms, Maximum = 3ms, Mittelwert = 3ms
```


Das ESP-Modul meldet sich nun beim „Anpingen“ und auch beim Aufruf im Webbrowser.
Auch ein **Ablehnung i**st ein Antwort und macht Hoffung auf einen erfolgreicheren Versuch.

Wir fügen an unser bisheriges Programm die Zeilen 18 bis 22 an.

```
17
18 import socket
19 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
20 s.bind(('', 80))
21 s.listen(5)
22
```

Der Socket auf dem ESP-Modul ermöglicht diesem eine Antwort auf eine Anfrage vom Webbrowser.

*Ein **Socket** (engl. Sockel) ist eine bidirektionale Netzwerk-Kommunikationsschnittstelle, deren Verwaltung das Betriebssystem übernimmt. Die Kommunikation findet zwischen einem Server und einem Client über einen definierten Port statt. ...* https://moviecultists.com/was-ist-ein-socket

Ein Socket ermöglicht eine Verbindung zwischen zwei Programmen.
In diesem Fall sind es der Webbrowser auf deinem PC und das aktuellen Programm auf dem ESP-Modul.

Mit `s.bind('',80)` wird der Socket an den Port 80 gebunden.
Es gibt 65536 verschiedene Ports.

Standartmäßig arbeiten die Browser immer über den Port 80. `http://192.168.2.115`

Gibt man den Port explizit an, funktioniert das logischerweise auch. `http://192.168.2.115:80/`

Versucht man den Webserver auf dem http://192.168.2.115:81/ zu erreiche, kommt keine Verbindung zum Webserver auf dem ESP-Modul zu zustande.

Außerdem ist das Programm `Station+WebServer.py` auf dem ESP-Modul noch nicht vollständig.
Es fehlt der Programmteil, welches den Webserver realisiert.

Wir fügen an unser Programm die Zeilen 23 bis 30 an, um den Webserver bereitzustellen.

```
22
23 while True:
24   conn, addr = s.accept()
25   print('Got a connection from %s' % str(addr))
26   request = conn.recv(1024)
27   print('Content = %s' % str(request))
28   response = ""
29   conn.send(response)
30   conn.close()
```

Diese Zeilen ermöglichen dem ESP-Modul,
- eine Anfrage eines Webbrowsers entgegen zu nehmen,
- diese Anfrage in der Kommandozeile der IDE auszugeben,
- ein Antwort an den Webbrowser zu senden.

Das schauen wir uns auf der nächsten Seite genauer an.

Unser Pogramm auf dem ESP-Modul nimmt eine eingehende Verbindung
- über den Socket an,
- lies diese ein,
- gibt deren Inhalt in der Kommandozeile der IDE aus und
- sendet eine Antwort an die Gegenstelle.

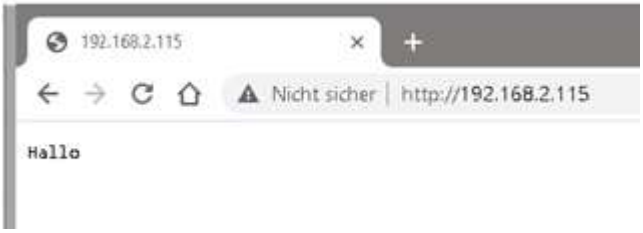
```
Habe ein Verbindung:  ('192.168.2.108', 54322)

Anfrage lautet:
 GET /leds_blink HTTP/1.1
Host: 192.168.2.115
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.489
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,appl
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7,no;q=0.6
```

'192.168.2.108‘ ist die IP-Nummer des Gerätes (PC, Handy,…), welches 'http://192.168.2.15‘ gesendet hat.

Unser Webserver reagiert mit einer Antwort


Leider waren keien Daten in der Anwort enthalten `allo"` ab und starten das Programm neu, bekommt der Webbrowser eine Antwort und stellt diese auf dem Bildschirm dar.




Ändere die Zeile 28 schrittweise

```
responce = "<h1>Hallo</h1>"

responce = "<h1>Hallo</h1><p>Es hat geklappt!<p>"

responce = "<h1>Hallo</h1><p>Es hat <b>endlich</b> geklappt!<p>"
```

und starte den Webserver jeweils neu und aktualisiere die Ansicht der Webseite im Browser deines PCs..

Besser ist es, die Webseite wie folgt zu definieren:

```
23  def web_page():
24    html = """<html>
25      <head>
26        <title>Mein ESP-Webserver</title>
27      </head>
28      <body>
29        <h1>"Hello World" from ESP-Webserver</h1>
30        <p>Es hat <b>endlich</b> geklappt!</p>
31      </body>
32    </html> """
33    return html
34  
35  while True:
36    conn, addr = s.accept()
37    print('Got a connection from %s' % str(addr))
38    request = conn.recv(1024)
39    print('Content = %s' % str(request))
40    response = web_page()
41    conn.send(response)
42    conn.close()
```

[ Station+WebServer.py ] [ web_pages.py ]

```
13
14 print(wlan_sta.ifconfig())
15 print(wlan_sta.ifconfig()[0])
16 print("\nhttp://"+wlan_sta.ifconfig()[0]+"\n\n")
17
18 import socket
19 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
20 s.bind(('', 80))
21 s.listen(5)
22
23 import web_pages
24
25 while True:
26   conn, addr = s.accept()
27   print('Got a connection from %s' % str(addr))
28   request = conn.recv(1024)
29   print('Content = %s' % str(request))
30   response = web_pages.startseite()
31   conn.send(response)
32   conn.close()
33
```

[ Station+WebServer.py ] [ web_pages.py ]

```
1  def startseite():
2    html = """<html>
3      <head>
4        <title>Mein ESP-Webserver</title>
5      </head>
6      <body>
7        <h1>"Hello World" from ESP-Webserver</h1>
8        <p>Es hat <u>endlich</u> geklappt!</p>
9      </body>
10   </html> """
11   return html
```

Beachte das neue Verhalten des ESP-Moduls.
Denn wird

- das Programm auf dem ESP-Modul gestoppt per IDE gestoppt,
- das Modul per Reset-Knopf neu gestartet
- oder es nach dem Ausschalten wieder neu gestartet,

das Modul zeigt weiterhin die gleichen Reaktionen.
Es ist weiter mit dem WLAN verbunden.

Gib in der Kommandozeile der IDE folgendes ein:

```
>>> import network
>>> wlan_sta = network.WLAN(network.STA_IF)
>>> wlan_sta.isconnected()
True
>>> wlan_sta.ifconfig()
('192.168.2.115', '255.255.255.0', '192.168.2.1', '192.168.2.1')
```

Erst nach

```
>>> wlan_sta.active(False)
```

wird das Modul bezüglich derWLAN-Verbindung verstummen.

```
>>> wlan_sta.ifconfig()

('0.0.0.0', '0.0.0.0', '0.0.0.0', '192.168.2.1')
```

Kann das WLAN wieder reaktiviert werden?

```
>>> wlan_sta.active(True)
 #7 ets_task(4020f560, 28, 3fff9448, 10)
```

Nein, das Interface ist nicht mehr ereichbar.

```
>>> wlan_sta.ifconfig()
('0.0.0.0', '0.0.0.0', '0.0.0.0', '192.168.2.1')
```

und müßte wieder neu verbunden werden.

```
>>>wlan_sta.connect("deineSSID","deinPasswort")
>>> wlan_sta.ifconfig()

('192.168.2.115', '255.255.255.0', '192.168.2.1', '192.168.2.1')
```

Wie sich das ESP-Modul verhält, wenn der WLAN Access Point zwischenzeitlich ausfällt, wurde noch nicht untersucht.