# Beschreibung Python -Skripts

Diese Tests wurden mit der Micropython- Installation
esp8266-20230426-v1.20.0.bin
gemacht.

Für den ersten Test wird benötigt:

```
.
├── batterie.py
├── BME280_1.py
├── boot.py
├── client2.py
├── config.py
├── main.py
└── start_station.py
```

Diese Programme müssen sich im Speicher der ESP8266-Platine befinden. Im Modul ‚main.py' ist zunächst *import client2* auskommentiert.

```
import time
# hier ist zeit fuer unterbrechung mit cntrl C ...
time.sleep(5)
print('main...')
#import client2
```

Wenn nach erfolgreichem Test das Modul ‚client2.py' nach dem Booten selbst starten soll, muss das Zeichen ‚#' vor ‚#import client2' entfernt werden.

Wir starten mit ‚start_station.py'.
Hier wird der WLAN ( =WIFI) mit dem Router verbunden. Die Router-Daten SSID und PASSWORD befinden sich in der Datei ‚config.py'.

```
#config.py
wifi={
    'ssid_privat:'password_privat',
    'Schuelerlabor':'pw_schuelerlabor',
    # 'SSID' : 'Passwort'
}
```

Hier können mehrere Router-SSID / PW - Paare eingetragen werden; Die Verbindung erfolgt mit dem ersten in der Liste mit den passenden Zugangsdaten. Bei Erfolg wird im Terminal angezeigt:

```
 wlan "meine SSID "; "mein Password" gefunden
....
verbunden als :192.168.178.52 mit 192.168.178.1
```

oder bei Misserfolg:

```
AssertionError:  kein bekanntes wlan gefunden, bitte "config.py" anpassen
```

Das Programm ‚start_station.py' muss nur beim ersten Mal aufgerufen werden; das Wifi-Modul merkt sich die Zugangsdaten.

Jetzt starten wir ‚client2.py':
In diesem Modul müssen folgende Einstellungen zum RaspberryPi- Server passen:

```
SERVER    = '192.168.178.21'
CLIENT_ID = 'ESP8266'
PORT      = 1883
USER      = 'pi'
PASSWORD  = 'pi'
```

Falls der MQTT- Server(siehe Installation von Mosquitto) noch nicht gestartet ist, kommt die Fehlermeldung.

# Programm ‚client2.py‘

Dieses Programm wickelt die Verbindung mit dem Mqtt- Server des Raspi ab und schickt Messwerte zum Server; eventuell werden auch Daten vom Server gelesen.

#Importieren der benötigten Module:

```
from time import sleep
from umqtt.simple import MQTTClient # ist bei esp8266 micropython eingebaut
from BME280_1 import BME280
import batterie
from machine import I2C,Pin,RTC,DEEPSLEEP,deepsleep,reset_cause,DEEPSLEEP_RESET
```

#Diese Funktion wird beim Empfang aufgerufen ( subscribe Topic)

```
def sub_cb(topic, msg):
    topic=topic.decode()
    msg= msg.decode()
    print(f'empfangen: {topic} = {msg}')
```

Die Funktionen für die Kommunikation via MQTT sind in einer ‚class Mqtt()‘) verpackt.

Mqtt enthält folgende Funktionen:

#Diese Funktion wird beim Senden aufgerufen ( publish Topic)

```
def publish(self,topic,x):
    print(f'gesendet: {topic} ={x}')
    msg   = f'{x}'.encode()
    topic = topic.encode()
    self.client.publish(topic, msg)  # Publish sensor data to MQTT topi
```

#connect /disconnect

```
def connect(self):
    try:
        self.client.connect()
        self.connected=True
    except:
        self.connected=False

def disconnect(self):
    self.client.disconnect()
```

# Zugangsdaten des Raspberry Pi im Netz

```
SERVER    = '192.168.178.21'
CLIENT_ID = 'ESP8266'
PORT      = 1883
USER      = 'pi'
PASSWORD  = 'pi'
```

# ein Durchlauf...

```
def run():
    # construct an I2C bus für den Sensor BME
    i2c = I2C(scl=Pin(5), sda=Pin(4), freq=100000)
    bme = BME280(address=0x76, i2c=i2c)
    mqtt=Mqtt()
    mqtt.connect()
    if mqtt.connected:
        try:
            mqtt.publish('temp',round(bme.temperature,1))
            sleep(.1)
            mqtt.publish('humi',round(bme.humidity,1))
            sleep(.1)
            mqtt.publish('press',round(bme.pressure,1))
            sleep(.2)
            mqtt.publish('volt',volt.batterie.volt())
            sleep(.2)
        except Exception as e:
            print(e)
        mqtt.disconnect()
```

```
#Start von deepsleep in Sekunden
       def schlaf(*,seconds=10):
           t= int(seconds*1000)
           rtc = RTC()
           rtc.irq(trigger=rtc.ALARM0, wake=DEEPSLEEP)
           rtc.alarm(rtc.ALARM0, t)
           deepsleep()

########  Hauptprogramm #################
       run()   # ein durchlauf
       schlaf(seconds=60*60) # eine Stunde Schalfen
       # weiter durch Reset von GPIO16 an RST
       # nach einer Stunde bei boot.py und main.py
```

# Programm ‚batterie.py‘

```
       # batterie.py

       import machine
       from time import sleep

       ''' batteriespannung messen, loetbruecke 'MSG' muss geschlossen sein
           spannungsteiler 470k 100k an ADC0
           max spannung am adc 1 volt '''
       def volt():
               y=5.0*machine.ADC(0).read() /944
               return round (y,2) # 2 stellen hinter'm komma reichen
           # bei 5V 944 gemessen ,empirisch

       ''' test '''
       if __name__=='__main__':
           from time import sleep
           while True:
               print( f"Batteriespannung= {volt()} Volt" )
               sleep(1)
```