

Beispiel : Grafische Datenauswertung mit **matplotlib** Temperaturverlauf am 13-JUL-2026 mit Sensor WS3

Dafür erforderlich ist die open source library **matplotlib**

Diese muss eventuell noch installiert werden mit:

sudo apt update und **sudo apt install python3-matplotlib**

Für eine bessere Archivierung habe ich noch den Ordner mit **mkdir -p /mnt/ssd/diagramme** angelegt.

Hier ein Auszug in welcher Form die Daten vorliegen in meiner

Datei: /mnt/ssd/data/2026_KW28/2026-07-13/daten.json

```
{"sensor": "Feinstaub", "pm25": 1.7, "pm10": 4.3, "station": "station1", "timestamp": "2026-07-13 00:00:48"} {"temp": 19, "status": 0, "station": "station1", "sensor": "co2", "co2": 507, "timestamp": "2026-07-13 00:00:58"} {"sensor": "Feinstaub", "pm25": 1.6, "pm10": 2.9, "station": "station1", "timestamp": "2026-07-13 00:00:59"} {"wind_speed": 0.0, "station": "station1", "sensor": "ws3", "pressure": 958.8, "wind_gust": 0.0, "rain_rate": 0, "rain_total": 0, "humidity": 53, "temperature": 20.0, "timestamp": "2026-07-13 00:01:06"}
```

Folgendes Programm:

```
import json
from datetime import datetime
import matplotlib.pyplot as plt

DATEI = "/mnt/ssd/data/2026_KW28/2026-07-13/daten.json"

zeiten = []
temperaturen = []

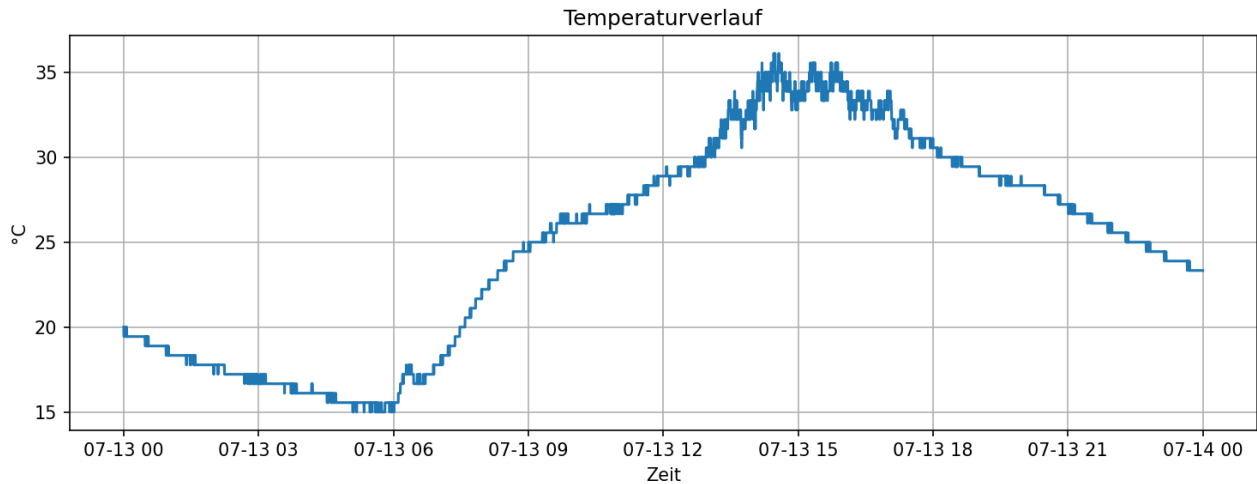
with open(DATEI, "r") as f:
    for zeile in f:
        try:
            daten = json.loads(zeile)
        except:
            continue
        if daten.get("sensor") != "ws3":
            continue

        zeit = datetime.strptime(
            daten["timestamp"],
            "%Y-%m-%d %H:%M:%S"
        )

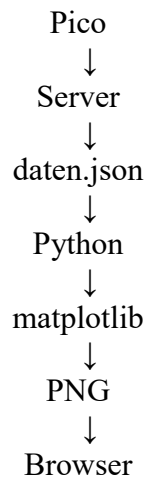
        temperaturen.append(daten["temperature"])
        zeiten.append(zeit)

plt.figure(figsize=(10,4))
plt.plot(zeiten, temperaturen)
plt.title("Temperaturverlauf")
plt.xlabel("Zeit")
plt.ylabel("°C")
plt.grid(True)
plt.tight_layout()
plt.savefig("/mnt/ssd/diagramme/temp_heute.png", dpi=150)
plt.close()
print("Diagramm gespeichert.")
```

Erzeugt diese Grafik, temp_heute.png :



Der Ablauf ist wie folgt:



Möglichkeiten um die Daten grafisch darzustellen:

1) Die Datei kopieren , z.B. von windows mit scp :

scp pi@192.168.178.xx:/mnt/ssd/diagramme/temp_heute.png F:

2) oder dann im Flask-Server. Dann reicht im Browser ein Klick auf "**Diagramme**" und das Bild wird direkt vom Raspberry Pi geladen, z.B.

```
@app.route("/diagramme/temp")
def temp():
    return send_file("/mnt/ssd/diagramme/temp_heute.png", mimetype="image/png")
```

3) Mein **Favorit** ist allerdings ein Windows Netzlaufwerk, da ein Raspberry PI Zero bei der Datenauswertung mit grafischer Darstellung schnell an seine Grenzen kommt. Die **rechenintensive Auswertung läuft auf meinem Windows-PC**, während der Raspberry Pi weiterhin nur seine eigentliche Aufgabe erledigt – Sensoren auslesen und Daten speichern.

Dazu muss zuerst auf den Raspberry Pi Server **samba** installiert werden mit **sudo apt update** und **sudo apt install samba** . Danach die Datei /etc/samba/smb.conf am Ende mit folgenden Eintrag ergänzen:

```
[Klimadaten]
  path = /mnt/ssd
  browseable = yes
  read only = yes
  guest ok = yes
```

read only = yes , weil :Raspberry Pi sammelt die Daten und Windows liest sie nur aus.

Danach Samba neu starten: **sudo systemctl restart smbd**

Unter Windows dann im Explorer eingeben: \\192.168.178.xx\Klimadaten.

Falls erforderlich muss noch **matplotlib** für Windows installiert werden:

Mit Windows-Eingabeaufforderung installieren: `py -m pip install matplotlib`

Ich arbeite vorzugsweise mit Thonny. Bei Tools → manage Packages noch matplotlib eintragen.

Das obige Programm kann wie folgt geändert/erweitert werden:

```
DATEI = r"\\192.168.178.xx\Klimadaten\data\2026_KW28\2026-07-13\daten.json"
```

```
AUSGABE = r"F:\save_klimadiagramme\temp_heute.png"
```

```
plt.savefig(AUSGABE, dpi=150)
```

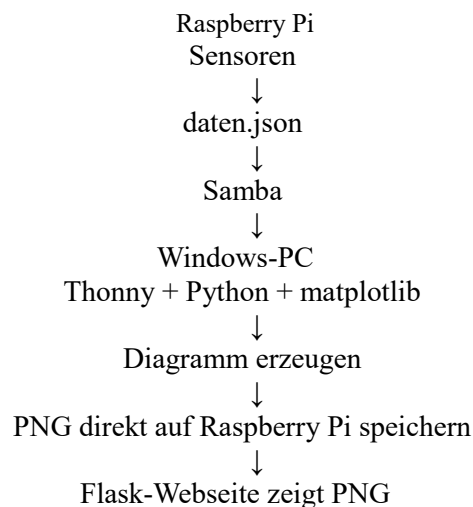
Vorteil: Direkte Anzeige

```
plt.show()
```

```
plt.close()
```

Will man dann eventuell die Daten auf das Raspberry Pi kopieren muss der Eintrag in der Datei /etc/samba/smb.conf von " read only = yes" nach "read only = no" ändern und dann mit `sudo systemctl restart smbd` aktivieren:

Der Ablauf ist dann wie folgt::



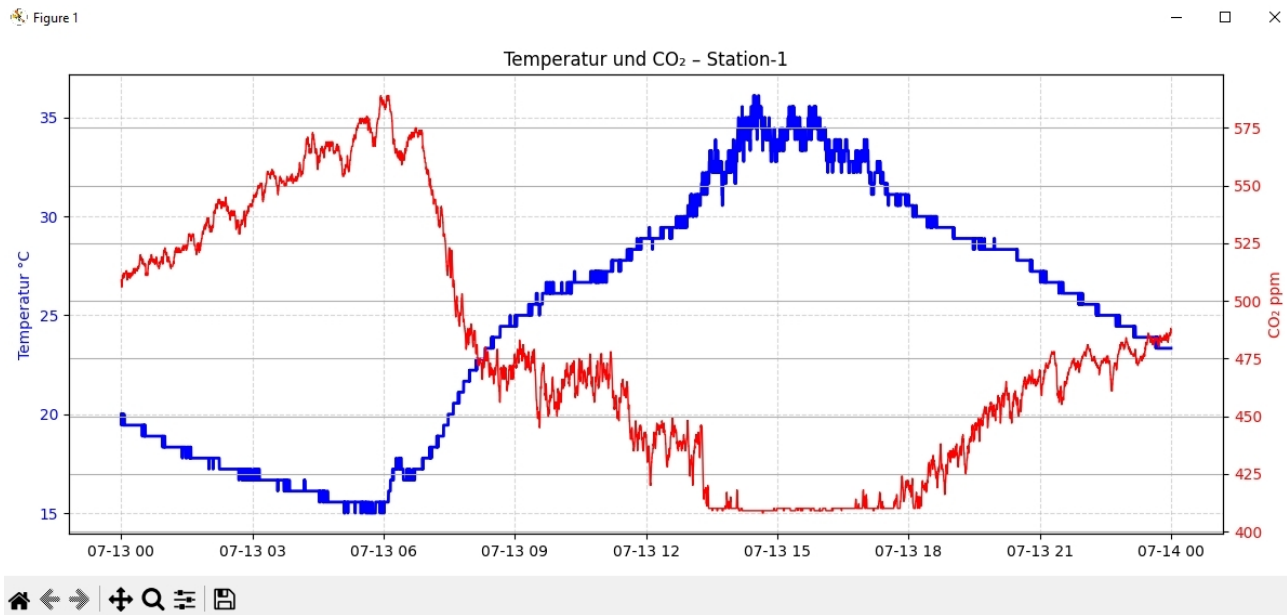
Anregungen/Referenzen/Meinungsaustausch bitte an info@pdp11gy.com

Hier noch ein weiteres Beispiel, Temperatur und CO2 mit meinen .json Daten :

```
import json
from datetime import datetime
import matplotlib.pyplot as plt

DATEI = "/mnt/ssd/data/2026_KW28/2026-07-13/daten.json"
zeiten_temp = []
temperaturen = []
zeiten_co2 = []
co2_werte = []
with open(DATEI, "r") as f:
    for zeile in f:
        try:
            daten = json.loads(zeile)
        except:
            continue
        zeit = datetime.strptime(
            daten["timestamp"],
            "%Y-%m-%d %H:%M:%S")
        sensor = daten.get("sensor")
        if sensor == "ws3":
            zeiten_temp.append(zeit)
            temperaturen.append(daten["temperature"])
        elif sensor == "co2":
            zeiten_co2.append(zeit)
            co2_werte.append(daten["co2"])
fig, ax1 = plt.subplots(figsize=(12, 5))
ax1.grid(axis="x", linestyle="--", alpha=0.5)
ax1.plot(
    zeiten_temp,
    temperaturen,
    label="Temperatur",
    color="blue",
    linewidth=2
)
ax1.tick_params(axis="y", labelcolor="blue")
ax1.set_ylabel("Temperatur °C", color="blue")
ax2 = ax1.twinx()
ax2.plot(
    zeiten_co2,
    co2_werte,
    label="CO2",
    color="red",
    linewidth=1
)
ax2.tick_params(axis="y", labelcolor="red")
ax2.set_ylabel("CO2 ppm", color="red")
plt.title("Temperatur und CO2 - Station-1")
plt.grid(True)
plt.tight_layout()
Ausgabe = "/mnt/ssd/diagramme/temp_co2.png"
plt.savefig(AUSGABE, dpi=150)
plt.show()
print("Diagramm gespeichert.")
```

Ergibt dann folgende grafische Darstellung:



Siehe Programm **temperatur_co2-13-JUL-2026.py**

Es gibt sehr viele weitere Möglichkeiten für eine grafische Auswertung.
Wäre schön wenn hier ein Meinungsaustausch stattfinden würde.

Anregungen/Referenzen/Meinungsaustausch bitte an **info@pdp11gy.com**