

Example : Graphical data analysis with matplotlib Temperature profile on 13-JUL-2026 with sensor WS3

This requires the open source library matplotlib.

This may need to be installed with

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

For better archiving, I also created the folder with

mkdir -p /mnt/ssd/diagramme.

Here is an excerpt showing the form in which the data is available in my

File: /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"}
```

The following program:

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

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

zeiten = []
temperaturen = []

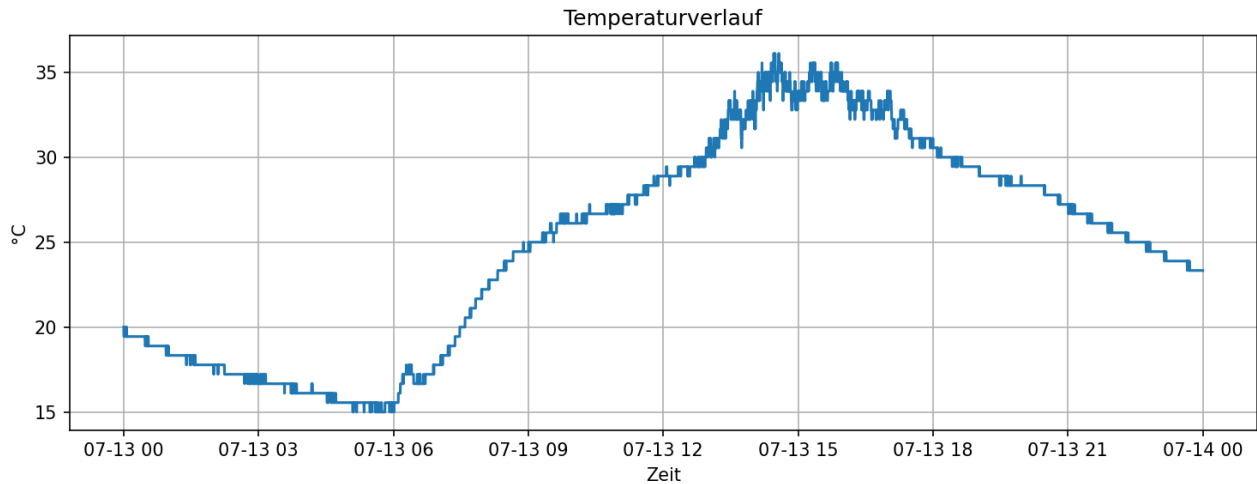
with open(DATEI, "r") as f:
    for zeile in f:
        try:
            daten = json.loads(zeile)
        except:
            continue
        if daten.get("sensor") != "ws3":
            continue

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

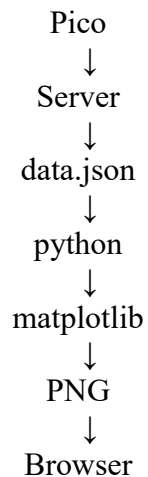
        temperaturen.append(daten["temperature"])
        zeiten.append(zeit)

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.")
```

generates this graphic: temp_heute.png :



Der Ablauf ist wie folgt:



Ways to represent the data graphically:

1) Copy the file, e.g. to Windows(drive F:) using scp:
scp pi@192.168.178.xx:/mnt/ssd/diagramme/temp_heute.png F:

2) or then in the Flask server. Then, a single click on "Diagrams" in the browser is all it takes , and the image will be loaded directly from the Raspberry Pi, e.g

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

3) However, my **favorite** is a Windows network drive, as a Raspberry Pi Zero quickly reaches its limits when it comes to data analysis with graphical representation. The **computationally intensive analysis** runs on my Windows PC, while the Raspberry Pi... Pi continues to perform only its primary task – reading sensors and storing data.

First, **samba** must be installed on the Raspberry Pi server using **sudo apt update** and **sudo apt install samba** . Then add the following entry to the end of the file `/etc/samba/smb.conf`

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

`read only = yes` , because: Raspberry Pi collects the data and Windows only reads it.

Then restart Samba: **sudo systemctl restart smbd**

In Windows, enter the following in Explorer: `\\192.168.178.xx\Klimadaten`.

If necessary, **matplotlib** for Windows must also be installed:

install using the Windows command prompt: ``py -m pip install matplotlib``. I prefer to use Thonny. Add matplotlib to the list under Tools → Manage Packages.

The above program can be modified/extended as follows:

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

```
OUTPUT = r"F:\save_climate_diagrams\temp_today.png"
plt.savefig(OUTPUT, dpi=150)
```

```
# Advantage: Direct display
```

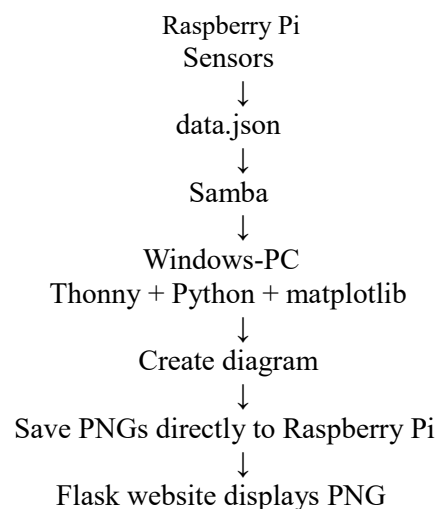
```
plt.show()
```

```
plt.close()
```

Then if you want to copy the data to the Raspberry Pi, you need to change the entry in the file `/etc/samba/smb.conf` from `"read only = yes"` to `"read only = no"` and then..

```
sudo systemctl restart smbd
```

The process is then as follows:



Please send suggestions/references/exchange of opinions to info@pdp11gy.com

Here's another example, temperature and CO2 with my .json data:

```

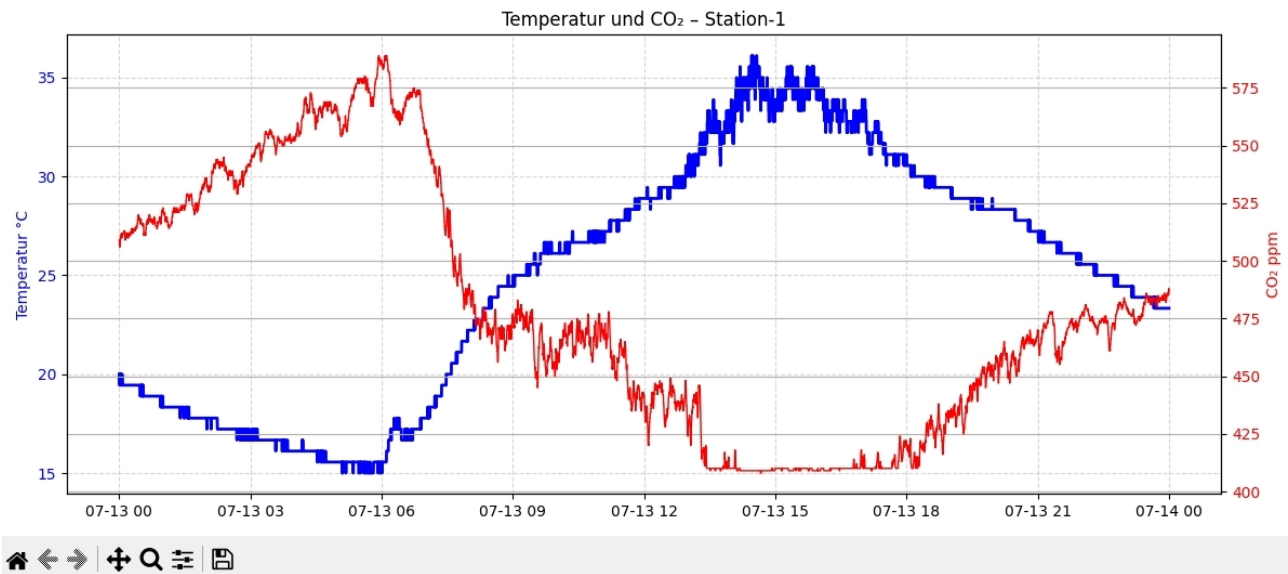
import json
from datetime import datetime
import matplotlib.pyplot as plt

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.")

```

This results in the following graphical representation

Figure 1



See program **temperatur_co2-13-JUL-2026.py**

There are many other possibilities for graphical analysis.
It would be nice if an exchange of opinions could take place here.

Please send suggestions/references/exchange of opinions to **info@pdp11gy.com**