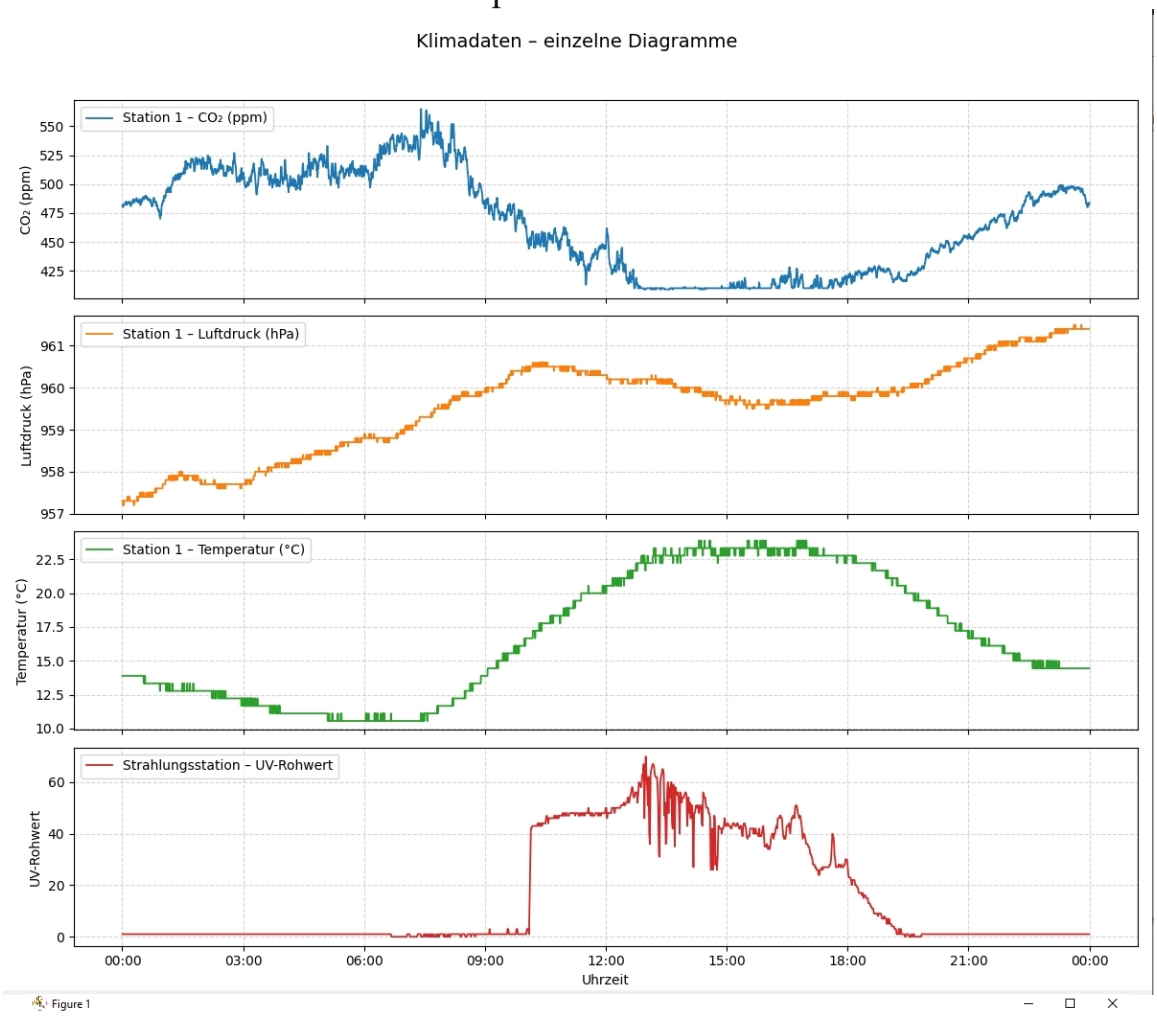
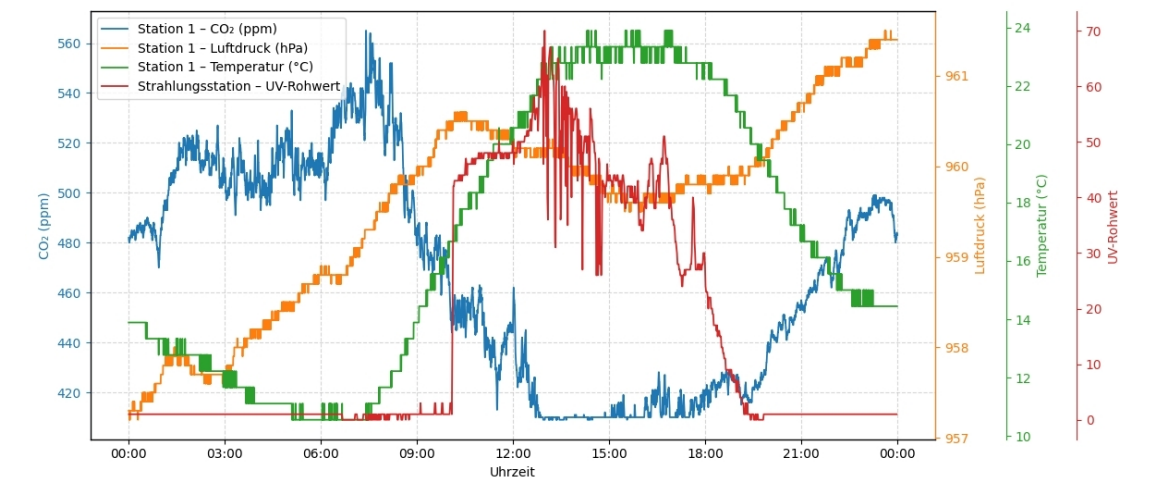
Interactive Climate Data Analysis and Visualization

Overview:

An interactive, Python-based tool for exploring environmental and climate data over time. Select a specific day or calendar week, choose up to four measurements, and visualize their time series with Matplotlib. Compare data from different stations to identify patterns, trends, and potential relationships between temperature, CO_2, particulate matter, UV radiation, and other environmental parameters.





Project Description:

Using an interactive calendar to select individual days or calendar weeks, together with a dynamically generated sensor list, you can select up to four measurements from different stations. Their time series are visualized with Matplotlib through a Tkinter interface, making it possible to investigate potential interactions, correlations, and relationships between the sensor data.

This free application, **Climate_Data_Analysis_EN.py**, was developed in Python with assistance from OpenAI ChatGPT. Sensor data are collected by Raspberry Pi Zero 2 W stations and stored on an SSD. Graphical analysis is performed on a Windows PC; running the analysis on a Raspberry Pi may also be possible.The climate stations use an affordable Raspberry Pi Zero W, although this model has limited capacity for processing and visualizing large amounts of sensor data. Therefore:

- Raspberry Pi Zero 2 W: Collect sensor data and store it on the SSD.
- Windows PC: Copy the data via Samba to H:/data and analyze it there.

For detailed information about my climate stations, please visit my website:
https://www.pdp11gy.com/
https://pdp11gy.com/Climatestation.html

Download links for the software and additional project information are available here:
GitHub: **https://github.com/pdp11gy/Environmental-and-Climate-Data-Station**

Requirements and Installation

The folder Diagrams/Diagram-Example.pdf already describes the steps required to set up the prerequisites. The steps are summarized below:

The open-source **Matplotlib** library is required.
It may need to be installed first:
Windows Command Prompt: Check whether Python is installed: python --version
If it is not installed, install Python and Thonny (recommended).
Install Matplotlib: **python -m pip install matplotlib**
In Thonny, open Tools > Manage Packages and install Matplotlib if needed.

Raspberry Pi OS: run sudo apt update, followed by sudo apt install python3-matplotlib.

Transferring data from the Raspberry Pi server to Windows

This can be done using scp from the Windows Command Prompt, for example:
`scp pi@192.168.178.xx:/path/to/daten.json F:`

My preferred method is a Windows network drive, because a Raspberry Pi Zero can quickly reach its limits when performing graphical data analysis.
The computationally intensive analysis runs on my Windows PC, while the Raspberry Pi continues to perform its main task: reading sensors and storing data.
First, install **Samba** on the Raspberry Pi server by running sudo apt update and **sudo apt install samba**. Then add the following entry at the end of /etc/samba/smb.conf:

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

read only = yes, because the Raspberry Pi collects the data and Windows only reads it.
Restart Samba with: **sudo systemctl restart smbd**
In Windows File Explorer, map the network drive.

Configuration

The settings are stored in config.ini and can be edited with any text editor if necessary.

Windows	**Raspberry PI**
[Paths]	[Paths]
data_folder = H:/**data**	data_folder = /mnt/ssd/data
data_filename = daten.json	data_filename = daten.json
output_folder = H:/1save_klima	output_folder = /mnt/ssd/result
output_filename = **klimadaten**.png	output_filename = klimadaten.png

The config.ini file is read each time the program starts. These are the basic settings and the file must be present.

Important Information:

Climate_Data_Analysis uses the data folder specified in config.ini (for example, /data). This folder contains the JSON data files, organized by year, week, and day.

Example folder structure (15 June 2026): /data/2026_KW24/2026-06-15/daten.json

```
data/
└── 2026_KW24/
    └── 2026-06-15/
        └── daten.json
```

The supplied /data folder contains sample data from my weather stations for reference and testing, covering calendar weeks 28 and 37. Week 28 contains data only from Station 1. Week 37 contains data from Station 1, Station 2, and the Radiation Station. The examples below refer to these files._

(My historical data can be made available on request.)

output_filename:

To prevent existing results for the same date from being overwritten, the program checks whether a filename already exists before saving and automatically uses the next available version number. This also works after restarting the program because existing files on disk are checked. For repeated daily analyses, the files are named as follows:

V1-**klimadaten**_2026-10-16.png
V2-**klimadaten**_2026-10-16.png
V3-**klimadaten**_2026-10-16.png

The same versioning scheme is used for weekly analyses:

V1-**klimadaten**_week41_2026.png
V1-**klimadaten**_week41_2026_statistics.csv

Using the Program and Examples

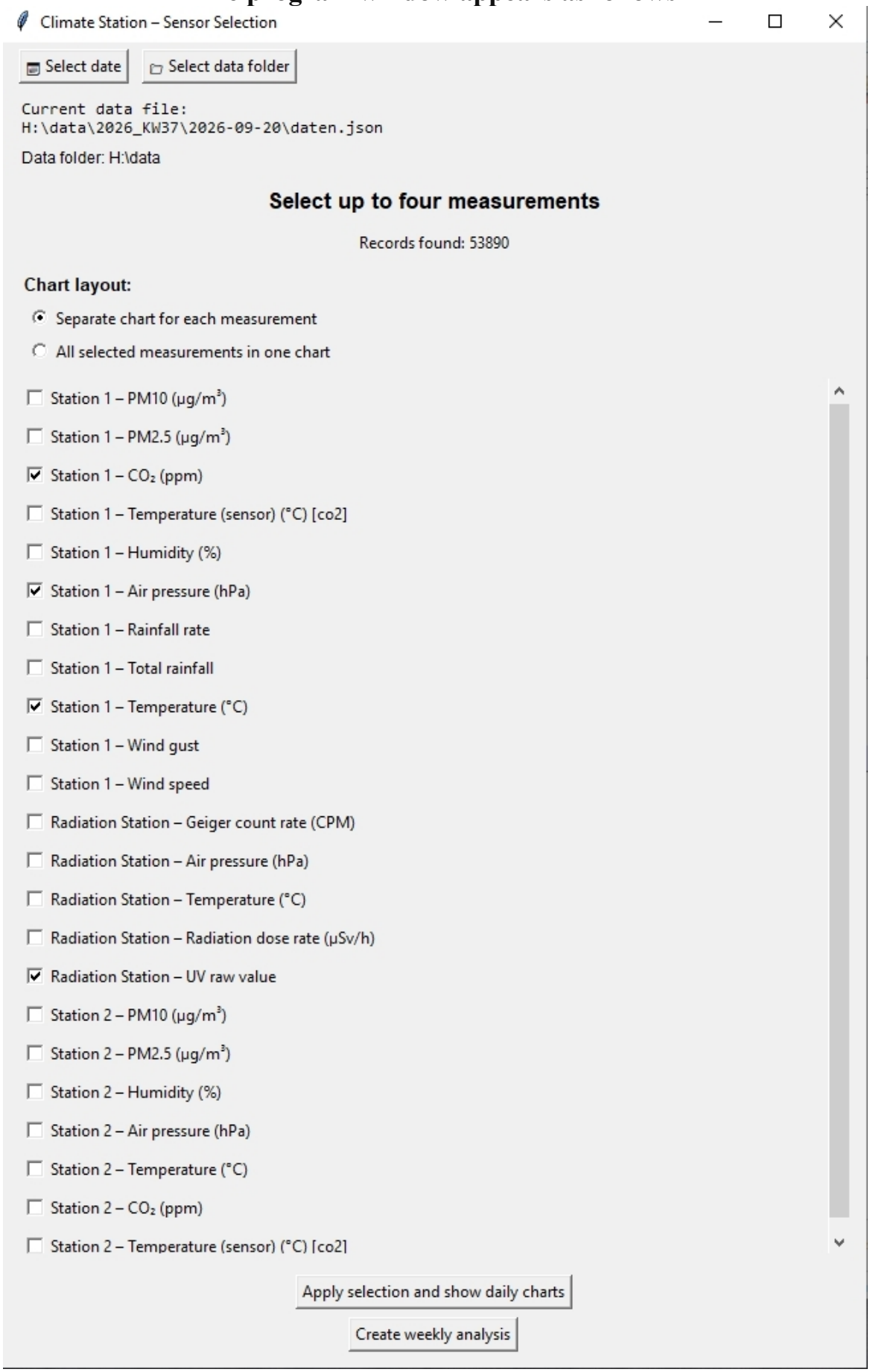
The program can be started in several ways:

1) Windows Command Prompt: python3 Climate_Data_Analysis_EN.py
2) In Thonny: File > Open > Climate_Data_Analysis_EN.py, then click Run.
3) Optional: PyInstaller can be used to package the program as a Windows executable. The Python environment must be configured correctly; this optional procedure has not yet been verified on every Python installation. Install PyInstaller as shown below:

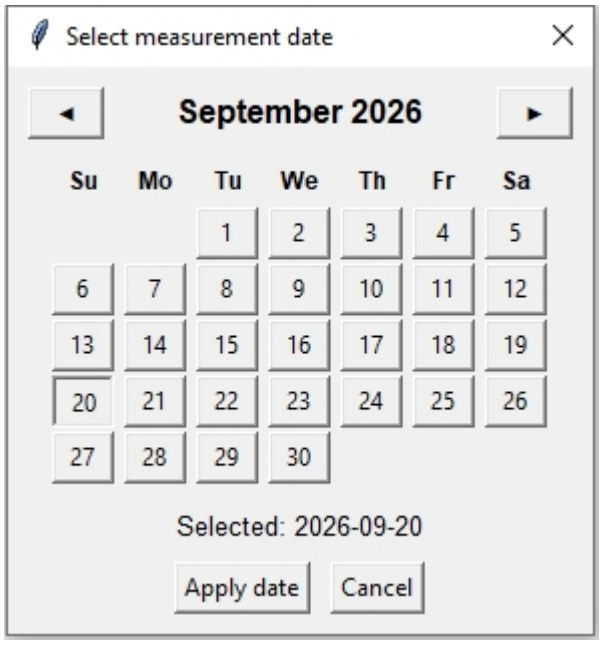
```
python -m pip install --upgrade pyinstaller
python -m PyInstaller --clean --onefile --name
Climate_Data_Analysis_EN Climate_Data_Analysis_EN.py
```

This creates Climate_Data_Analysis_EN.exe, which can be launched directly from the Windows desktop, for example using a desktop shortcut with an icon.

The program window appears as follows



The Select date button is located at the upper left. Clicking it opens the calendar:



Select the desired date in the calendar. The corresponding data file is then shown below, for example: H:\data**2026_KW37\2026-09-20**\daten.json.
The Select data folder button to the right opens a dialog where you can choose the folder containing the data. In this example, the folder is H:\data\.

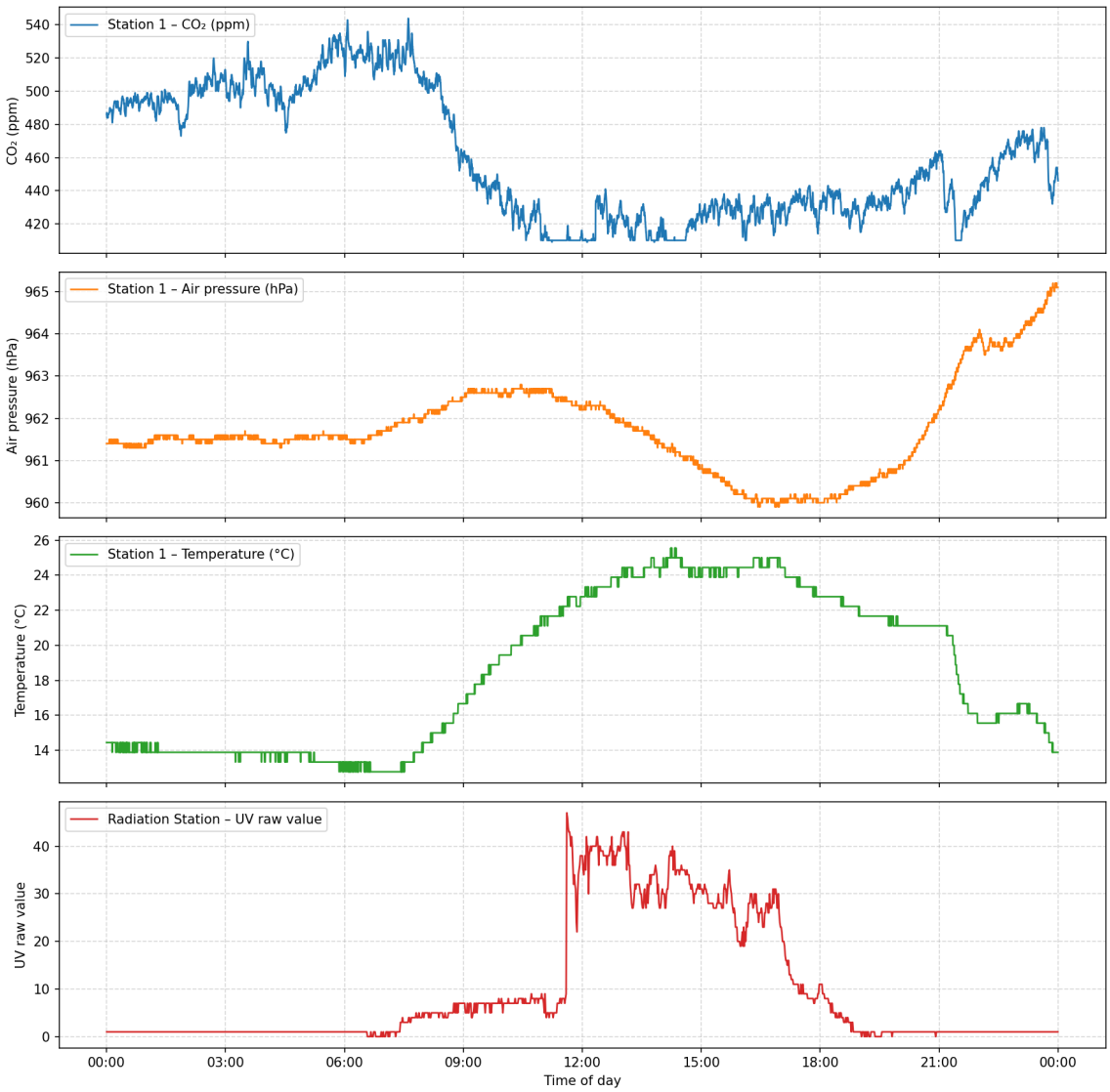
Below this, "**select the chart layout"**: either Separate chart for each measurement or All selected measurements in one chart.

The **sensor list** follows. In this example, four measurements are selected: CO_2 (ppm), air pressure (hPa), temperature (°C), and raw UV value.

In the first example, click "**Apply selection and show daily charts"**.
The following message appears, followed by the chart:

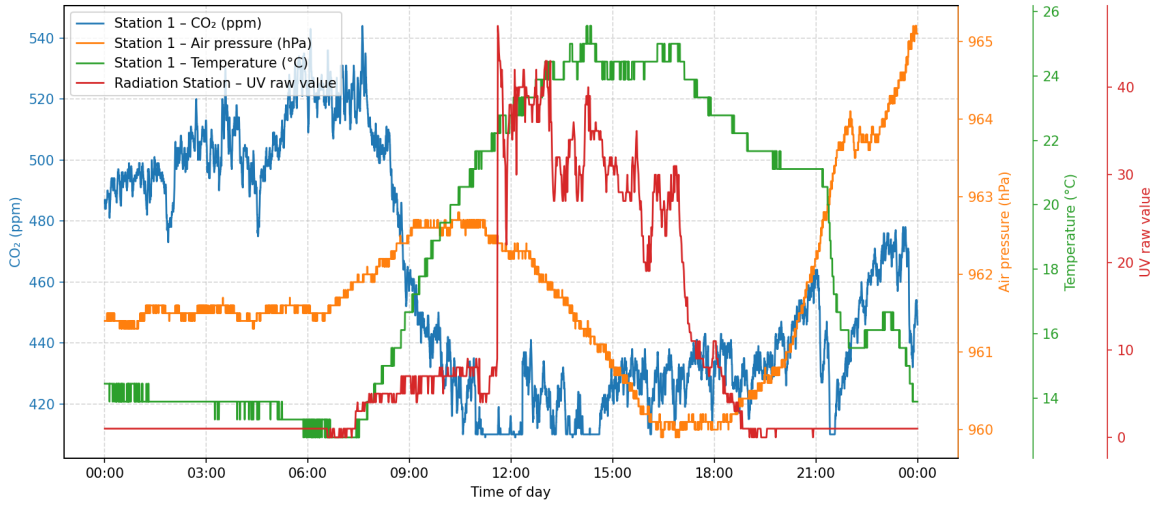
Selected measurements:
Station 1 – CO_2 (ppm) -> ('station1', 'co2', 'co2')
Station 1 – Air pressure (hPa) -> ('station1', 'ws3', 'pressure')
Station 1 – Temperature (°C) -> ('station1', 'ws3', 'temperature')
Radiation Station – UV raw value -> ('station1u2', 'geiger', 'uv_raw')
Data file: H:\data\2026_KW37\2026-09-20\daten.json
Chart saved (new version): H:\result\V2-klimadaten_2026-09-20.png
Number of measurements plotted: 4
Chart layout: separate charts

Climate data – separate charts



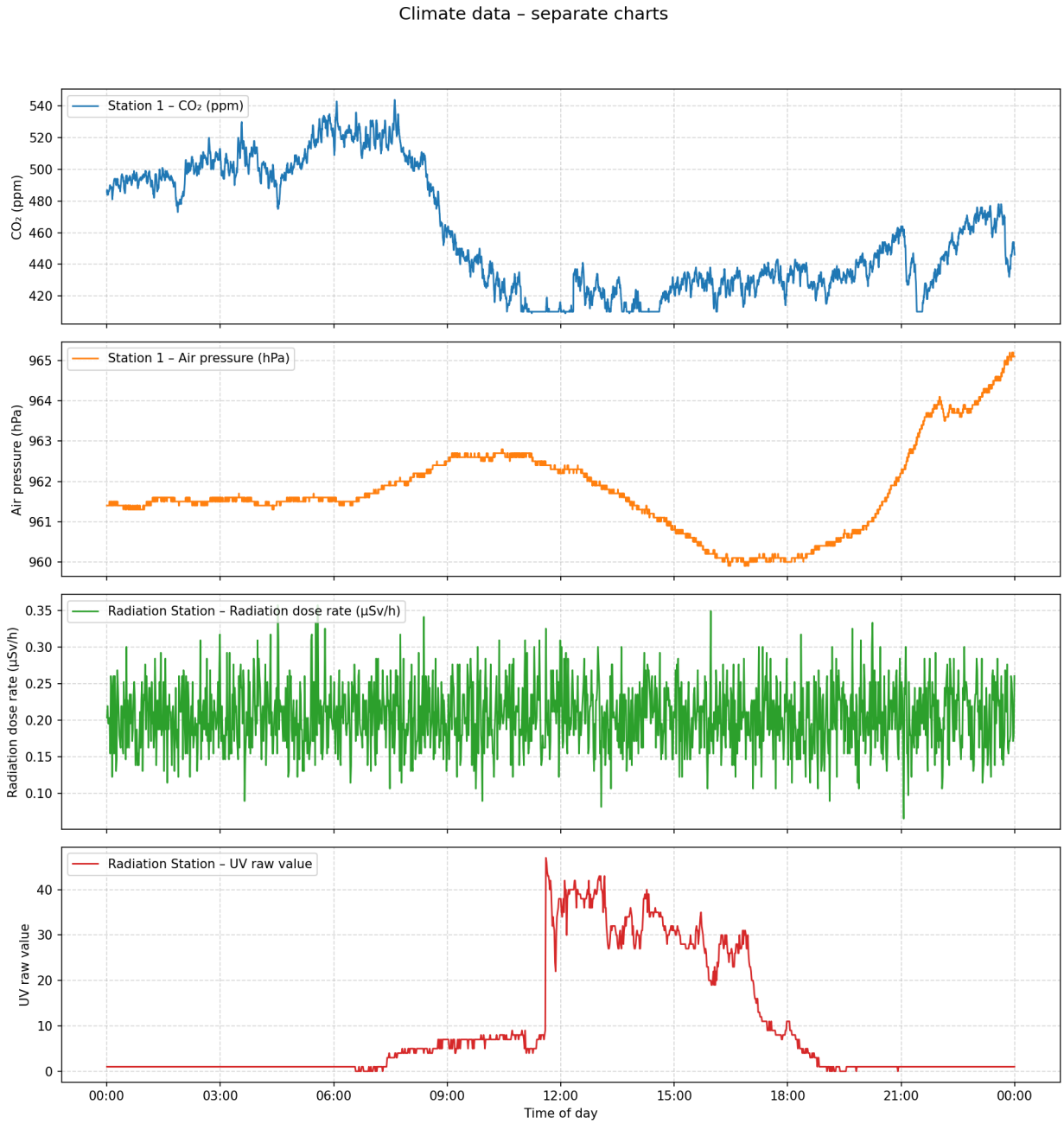
Now select the layout **All selected measurements in one chart**:

Climate data – combined chart



The Radiation Station makes it possible to examine radioactivity and UV radiation alongside air pressure and CO_2. The following message and chart are displayed:

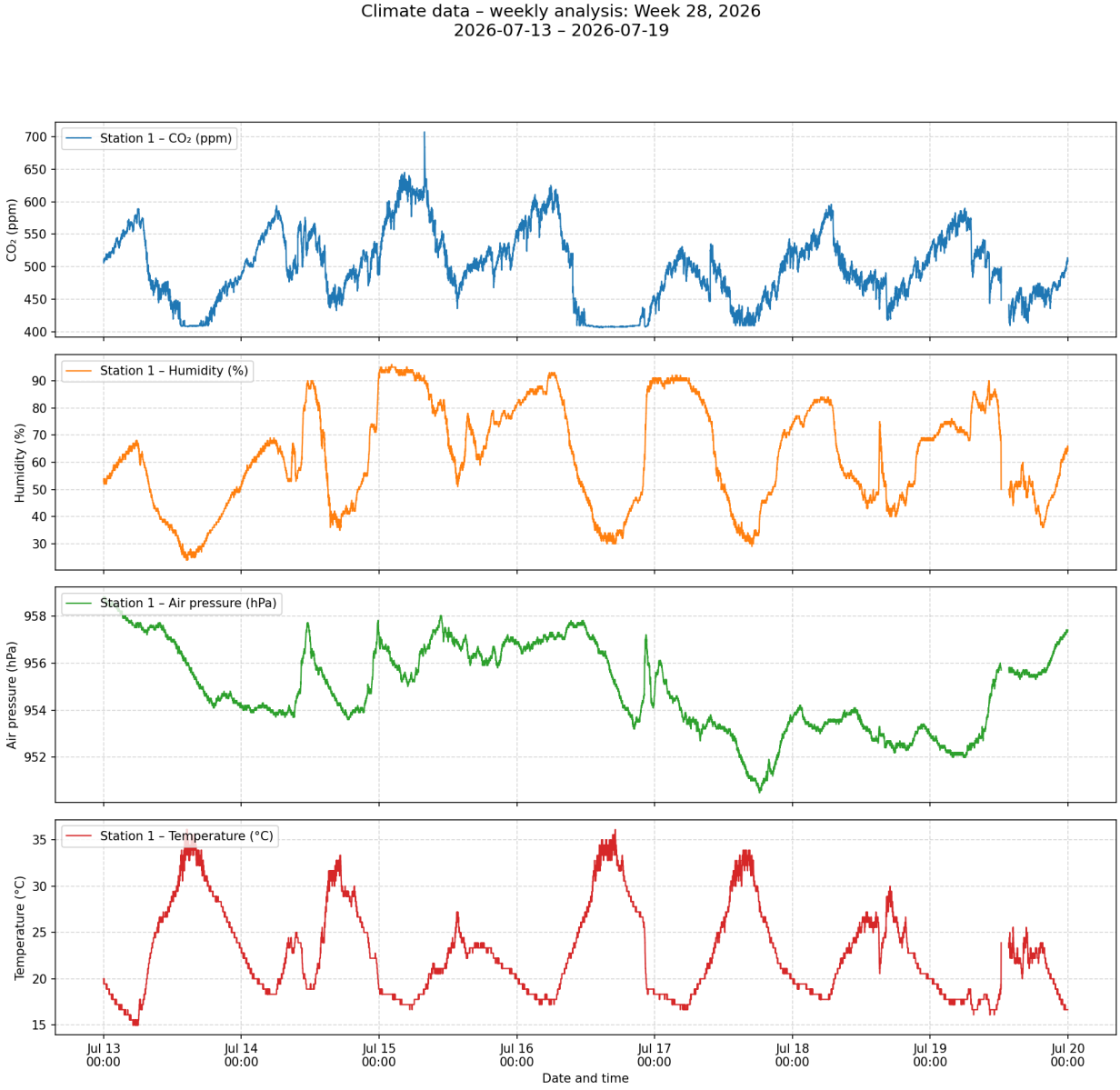
Selected measurements:
Station 1 – CO_2 (ppm) -> ('station1', 'co2', 'co2')
Station 1 – Air pressure (hPa) -> ('station1', 'ws3', 'pressure')
Radiation Station – Radiation dose rate (µSv/h) -> ('station1u2', 'geiger', 'usvh')
Radiation Station – UV raw value -> ('station1u2', 'geiger', 'uv_raw')
Data file: H:\data\2026_KW37\2026-09-20\daten.json
Chart saved (new version): H:\result\V2-klimadaten_2026-09-20.png
Number of measurements plotted: 4
Chart layout: separate charts



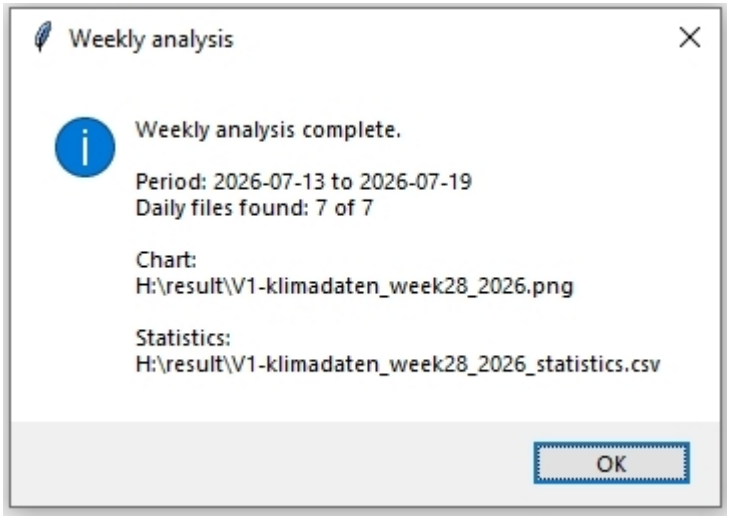
(No clear relationship with radioactivity is apparent.)

Here is an example of a weekly analysis. Select a date from the heatwave period in calendar week 28, for example, \2026_KW28\2026-07-13. When the weekly analysis is started, the following message and chart appear:

Selected measurements:
Station 1 – CO_2 (ppm) -> ('station1', 'co2', 'co2')
Station 1 – Air pressure (hPa) -> ('station1', 'ws3', 'pressure')
Radiation Station – Radiation dose rate (µSv/h) -> ('station1u2', 'geiger', 'usvh')
Radiation Station – UV raw value -> ('station1u2', 'geiger', 'uv_raw')
Data file: H:\data\2026_KW37\2026-09-20\daten.json
Chart saved (new version): H:\result\V2-klimadaten_2026-09-20.png
Number of measurements plotted: 4
Chart layout: separate charts
Weekly analysis: 2026-07-13 to 2026-07-19
Days loaded: 7 of 7
Weekly chart saved (new version): H:\result\V3-klimadaten_week28_2026.png
Weekly statistics saved (new version): H:\result\V1-klimadaten_week37_2026_statistics.csv



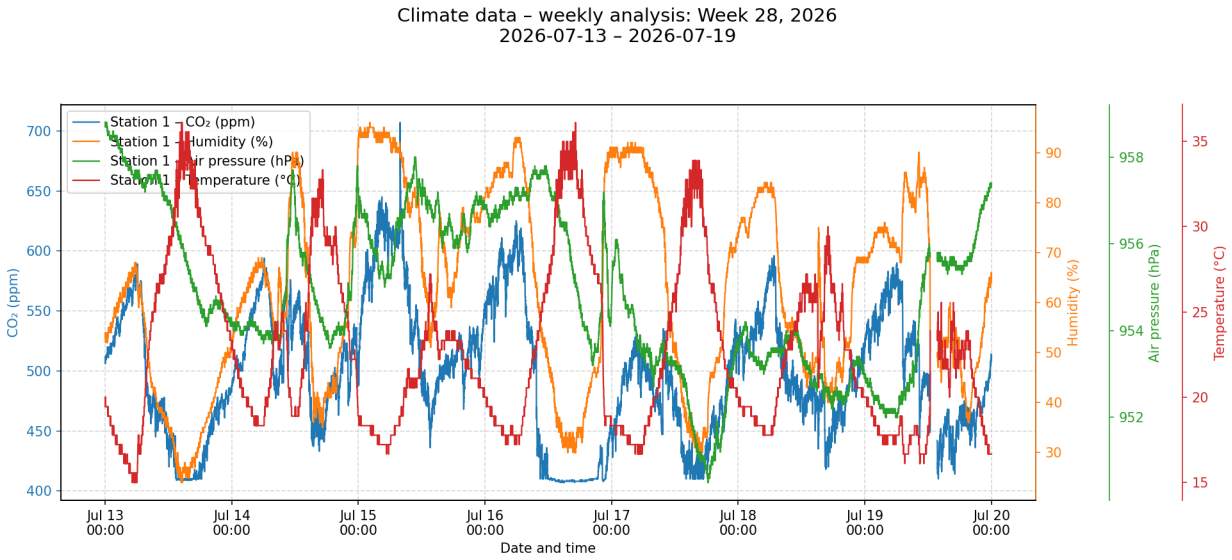
After the analysis is complete, the following message appears:



The program also creates a CSV file containing the sample count, minimum, mean, and maximum for each selected measurement.
The statistics file is saved in CSV format with the following columns:

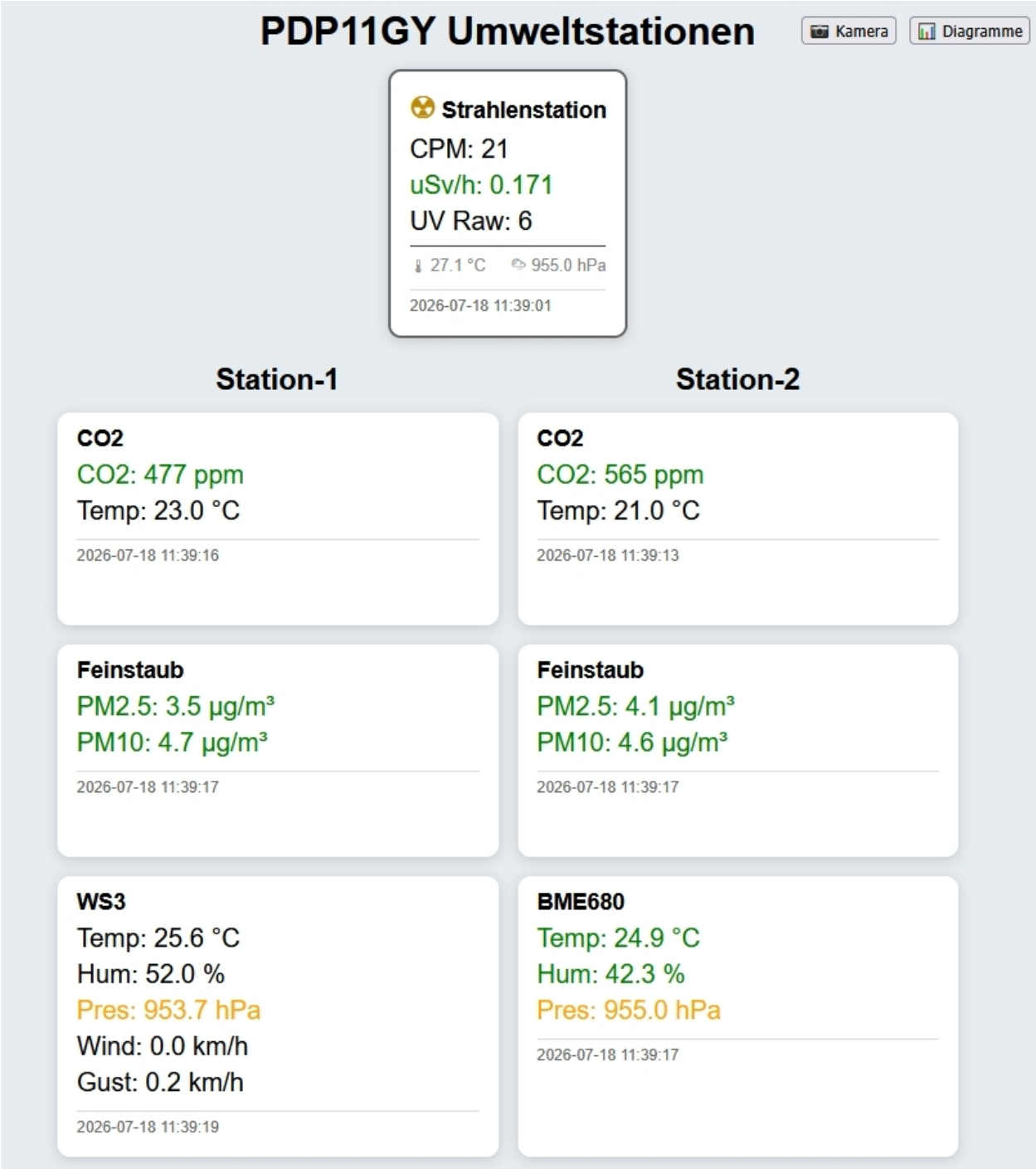
```
Station,Sensor,Measurement,Unit,Sample count,Minimum,Mean,Maximum
Station 1,co2,CO₂,ppm,54373,407.000,497.818,707.000
Station 1,ws3,Humidity,%,46420,24.000,62.082,96.000
Station 1,ws3,Air pressure,hPa,46420,950.500,954.904,958.800
Station 1,ws3,Temperature,°C,46420,15.000,22.835,36.111
```

The combined chart is shown below:



Data Format and Architecture

The following provides an overview of the climate station setup:
Detailed information: **https://pdp11gy.com/Climatestation.html**



The system currently consists of three stations: Station 1, Station 2, and a dedicated Radiation Station. The low-cost hardware architecture uses Raspberry Pi Pico W boards to transmit sensor data and Raspberry Pi Zero W devices as server systems.

Sensor data are transferred to the server system at intervals of approximately 15

seconds and stored daily in daten.json format. Each file contains records such as the following:

{"temp": 13, "status": 0, "station": "station1", "sensor": "co2", "co2": 484, "timestamp": "2026-09-20 00:02:20"}
{"usvh": 0.195, "station": "station1u2", "cpm": 24, "uv_raw": 1, "temperature": 23.4, "pressure": 964, "sensor": "geiger", "timestamp": "2026-09-20 00:02:22"}
{"wind_speed": 0.0, "station": "station1", "sensor": "ws3", "pressure": 961.4, "wind_gust": 0.0, "rain_rate": 0, "rain_total": 0, "humidity": 76, "temperature": 14.444445, "timestamp": "2026-09-20 00:02:23"}
{"sensor": "Feinstaub", "pm25": 2.2, "pm10": 4.4, "station": "station2", "timestamp": "2026-09-20 00:02:23"}
{"sensor": "Feinstaub", "pm25": 2.2, "pm10": 4.6, "station": "station1", "timestamp": "2026-09-20 00:02:24"}
{"humidity": 38.7, "pressure": 963.1, "station": "station2", "sensor": "bme680", "temperature": 22.5, "timestamp": "2026-09-20 00:02:25"}
{"temp": 18, "status": 0, "station": "station2", "sensor": "co2", "co2": 565, "timestamp": "2026-09-20 00:02:29"}

The data are displayed in the software as follows:

```
STATION_NAMEN = {
    "station1": "Station 1",
    "station2": "Station 2",
    "station1u2": "Strahlungsstation",
}

FELD_INFO = {
    "co2": ("CO₂", "ppm"),
    "pm25": ("PM2.5", "µg/m³"),
    "pm10": ("PM10", "µg/m³"),
    "temperature": ("Temperatur", "°C"),
    "temp": ("Temperatur (Sensor)", "°C"),
    "humidity": ("Luftfeuchtigkeit", "%"),
    "pressure": ("Luftdruck", "hPa"),
    "wind_speed": ("Windgeschwindigkeit", ""),
    "wind_gust": ("Windböen", ""),
    "rain_rate": ("Niederschlagsrate", ""),
    "rain_total": ("Niederschlag gesamt", ""),
    "usvh": ("Strahlungsdosisleistung", "µSv/h"),
    "cpm": ("Geiger-Zählrate", "CPM"),
    "uv_raw": ("UV-Rohwert", ""),
}
```

The architecture is flexible and can be extended or adapted as needed.

Contributors and Collaboration

Project Concept, Hardware, and Data Acquisition

The Environmental and Climate Data Station project was conceived and developed by Reinhard Heuberger. The project concept, selection and integration of sensor hardware, programming of the data acquisition, and construction and operation of the climate stations are his responsibility.

The goal is to collect environmental and climate data over the long term using low-cost hardware and to make the data accessible and understandable to interested users.

Support from OpenAI ChatGPT

OpenAI ChatGPT was used as an AI-assisted tool during development of the Python-based analysis software. The collaboration covered the program structure, the Tkinter graphical user interface, processing of JSON measurement data, Matplotlib visualization, and development and debugging of the calendar and folder selection.

Additional support included configuration through config.ini, automatic version numbering for saved charts and statistics, and implementation of the weekly analysis.

Joint Development and Responsibility

The analysis software was developed step by step. Reinhard Heuberger provided requirements based on practical operation of his climate stations, tested proposed changes in his own environment, and provided feedback on functionality and usability. ChatGPT then assisted with programming, revisions, and troubleshooting.

The final selection, integration, and practical testing of the software were carried out by Reinhard Heuberger. ChatGPT is explicitly acknowledged here to make the use of AI-assisted tools in software development transparent.

Project concept and data acquisition: Reinhard Heuberger
AI-assisted development support: OpenAI ChatGPT

Outlook

Development of the analysis software is an ongoing process. Experience from practical operation, new requirements, and user feedback can lead to future improvements. The aim is to keep the software understandable, reliable, and useful to other interested users.

The project demonstrates how low-cost sensor hardware, practical experience, and AI-assisted programming support can work together. Development will continue step by step by Reinhard Heuberger, with OpenAI ChatGPT supporting programming, analysis, and troubleshooting.

For comments and suggestions:
Email: **info@pdp11gy.com**