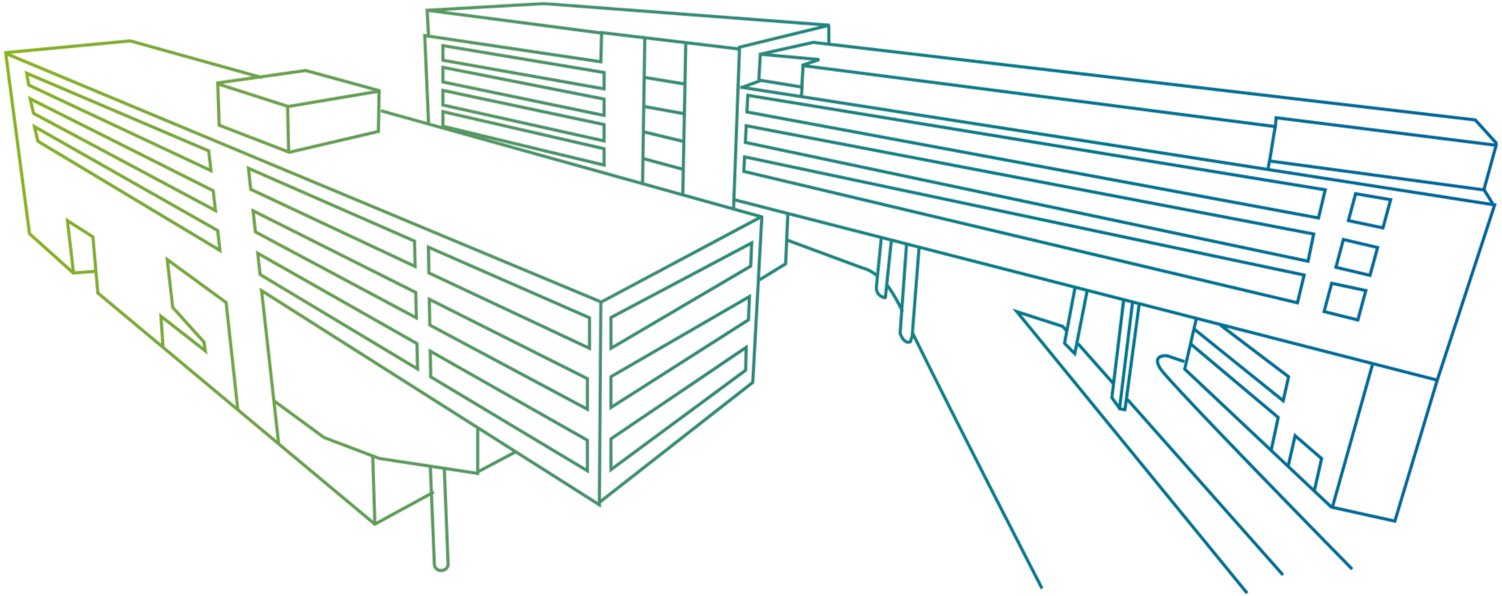


FH University of Applied Sciences TECHNIKUM WIEN

Smart Watering

Julia Koblmiller, Anton Shapovalov, Christoph Litschauer

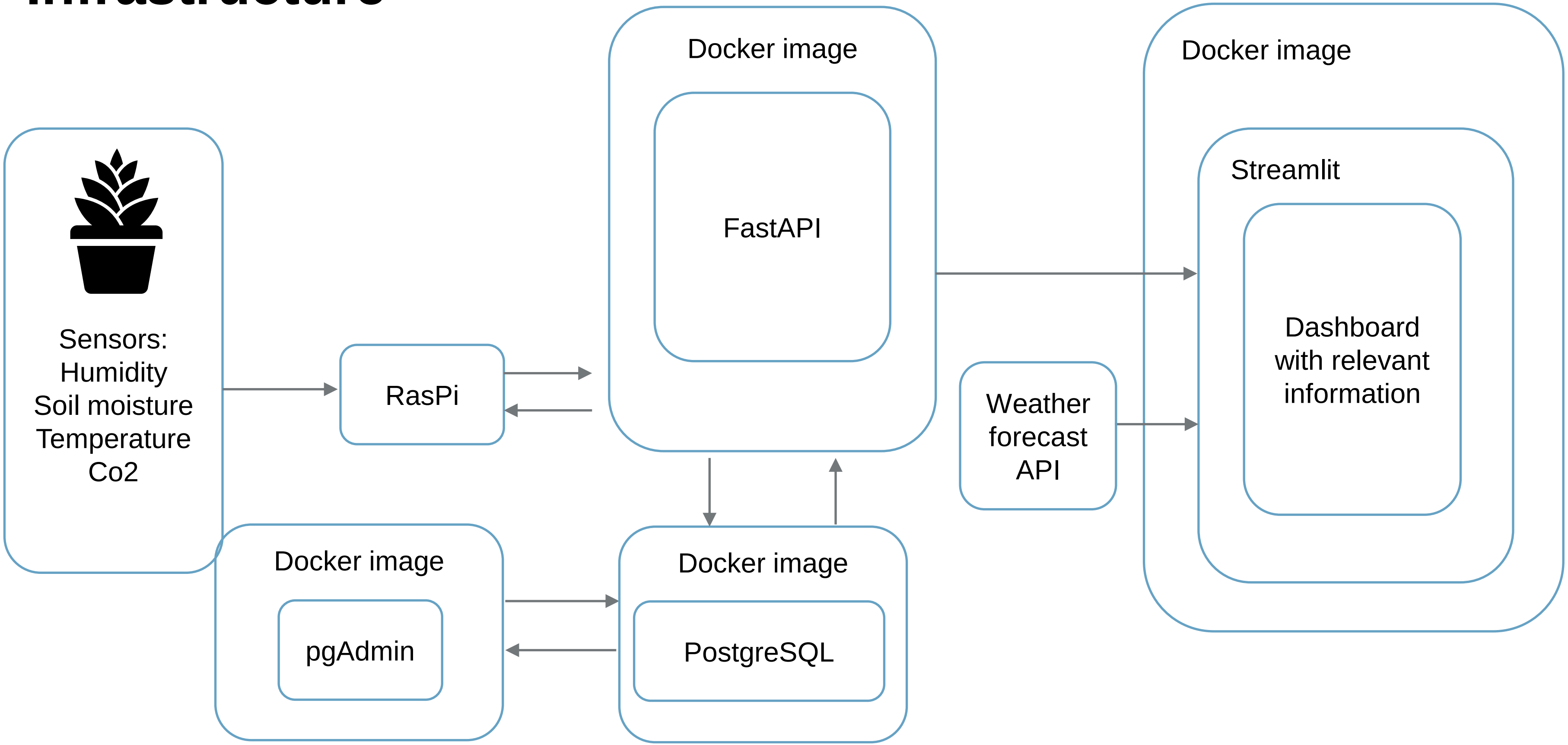
Contents

- Project Goal
- Infrastructure
- Hardware Setup / Sensors
- Application
- Outlook & Lessons Learned

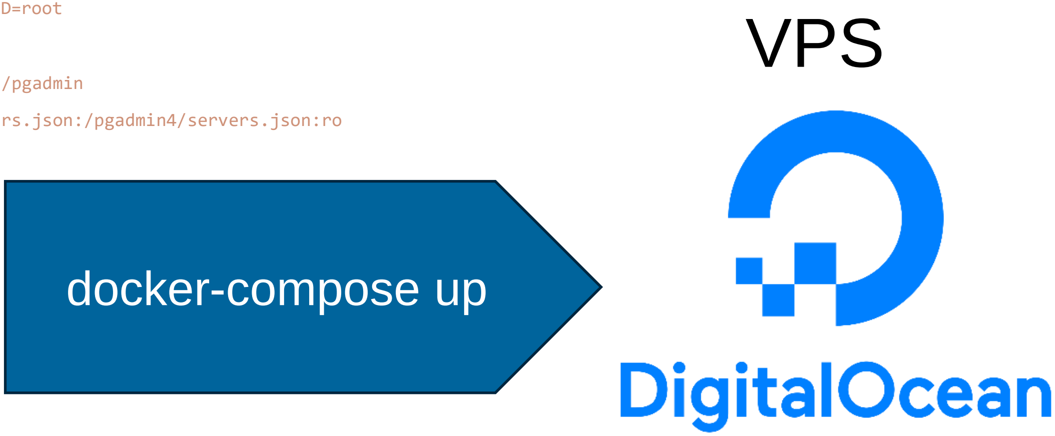
Project Goal

- Optimizing plant growth
- Predicting watering times
- Use local weather information
- Gathering plant-specific data

Infrastructure

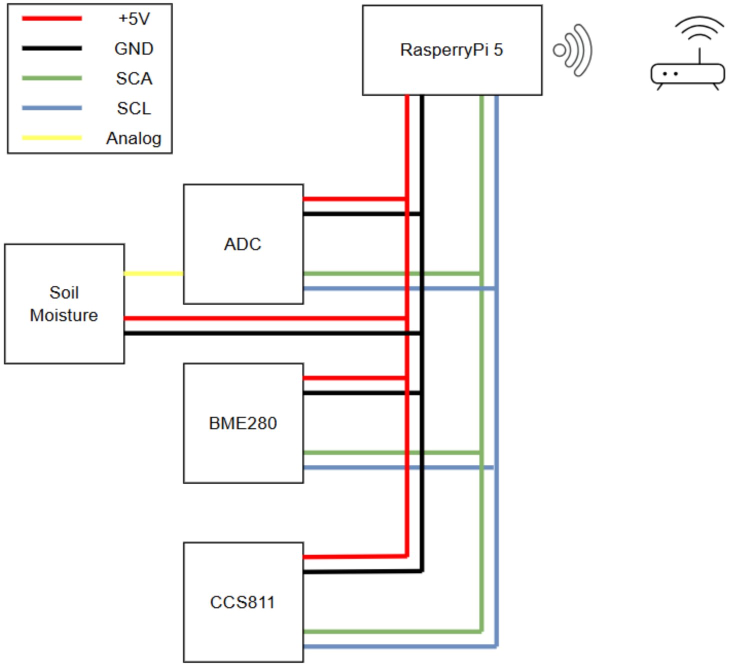


```yaml
services:
  pgdatabase:
    image: postgres:13
    container_name: pgdatabase
    environment:
      - POSTGRES_USER=root
      - POSTGRES_PASSWORD=root
      - POSTGRES_DB=sensors_data
    volumes:
      - ./db/data:/var/lib/postgresql/data:rw
    ports:
      - "5432:5432"
    networks:
      - smart_watering

  pgadmin:
    image: dpage/pgadmin4
    container_name: pgadmin
    environment:
      - PGADMIN_DEFAULT_EMAIL=admin@admin.com
      - PGADMIN_DEFAULT_PASSWORD=root
    volumes:
      - pgadmin_config:/var/lib/pgadmin
      - ./db/pgadmin_data/servers.json:/pgadmin4/servers.json:ro
    ports:
      - "8080:80"
    networks:
      - smart_watering

  backend:
    build:
      context: ./backend
    container_name: backend
    depends_on:
      - pgdatabase
    ports:
      - "8000:8000"
    networks:
      - smart_watering

  frontend:
    build:
      context: ./frontend
    container_name: frontend
    ports:
      - "8501:8501"
    networks:
      - smart_watering

networks:
  smart_watering:
    name: smart_watering
volumes:
  pgadmin_config:
```

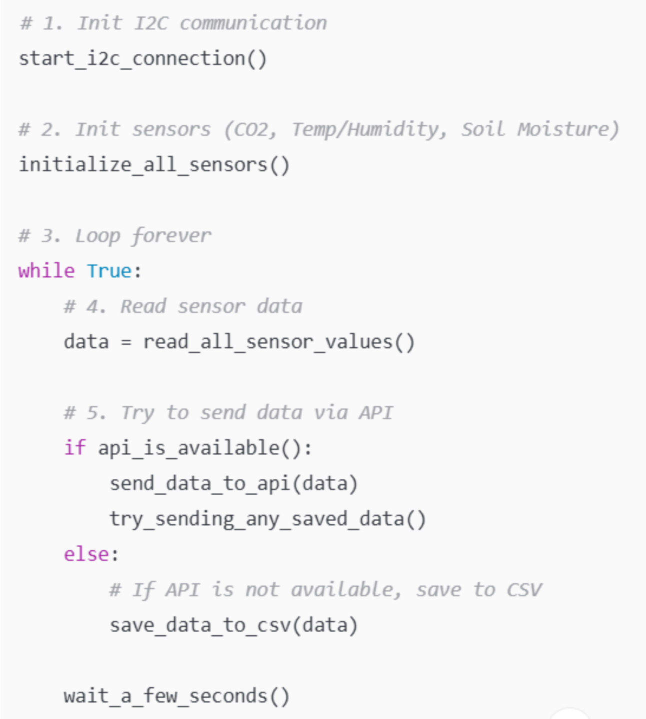


Hardware Setup / Sensors

- Wi-Fi Router
- RaspberryPi 5
- BME280 (Temperature, Humidity)
- CCS811 (CO2)
- ADS1115 (ADC)
- Soil Moisture (Capacitor)



Hardware Setup / Sensors

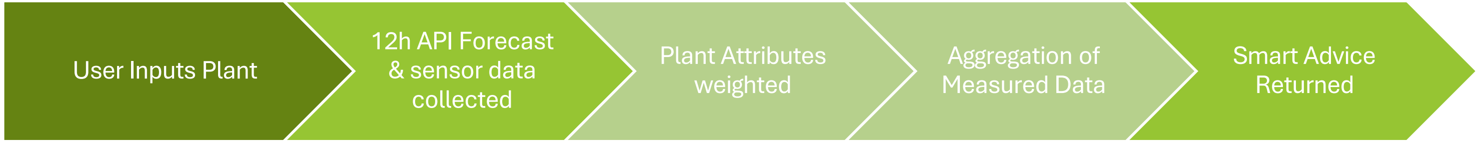


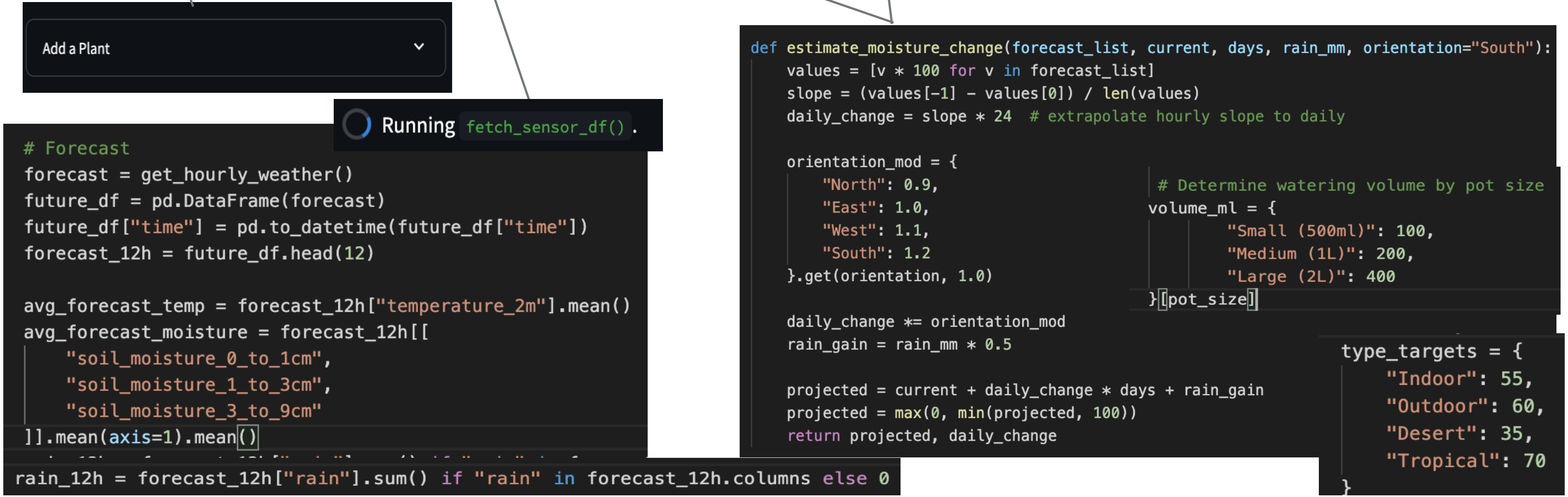
```
# 1. Init I2C communication
start_i2c_connection()

# 2. Init sensors (CO2, Temp/Humidity, Soil Moisture)
initialize_all_sensors()

# 3. Loop forever
while True:
    # 4. Read sensor data
    data = read_all_sensor_values()

    # 5. Try to send data via API
    if api_is_available():
        send_data_to_api(data)
        try_sending_any_saved_data()
    else:
        # If API is not available, save to CSV
        save_data_to_csv(data)

    wait_a_few_seconds()
```

- Adafruit libraries
- I^{2}C
- HTTP requests
- API
- JSON
- CSV files

My Plants

Combines sensor data, weather forecasts, and plant-specific traits


User Inputs Plant
12h API Forecast & sensor data collected
Plant Attributes weighted
Aggregation of Measured Data
Smart Advice Returned



Running `fetch_sensor_df()`.

```python
# Forecast
forecast = get_hourly_weather()
future_df = pd.DataFrame(forecast)
future_df["time"] = pd.to_datetime(future_df["time"])
forecast_12h = future_df.head(12)

avg_forecast_temp = forecast_12h["temperature_2m"].mean()
avg_forecast_moisture = forecast_12h[[
    "soil_moisture_0_to_1cm",
    "soil_moisture_1_to_3cm",
    "soil_moisture_3_to_9cm"
]].mean(axis=1).mean()
```

```python
rain_12h = forecast_12h["rain"].sum() if "rain" in forecast_12h.columns else 0
```

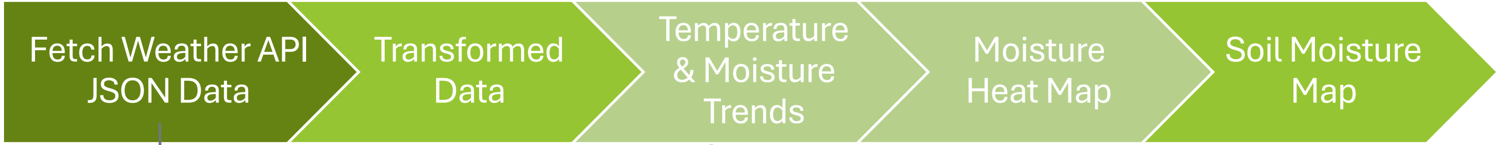
```python
def estimate_moisture_change(forecast_list, current, days, rain_mm, orientation="South"):
    values = [v * 100 for v in forecast_list]
    slope = (values[-1] - values[0]) / len(values)
    daily_change = slope * 24  # extrapolate hourly slope to daily

    orientation_mod = {
        "North": 0.9,
        "East": 1.0,
        "West": 1.1,
        "South": 1.2
    }.get(orientation, 1.0)

    daily_change *= orientation_mod
    rain_gain = rain_mm * 0.5

    projected = current + daily_change * days + rain_gain
    projected = max(0, min(projected, 100))
    return projected, daily_change
```

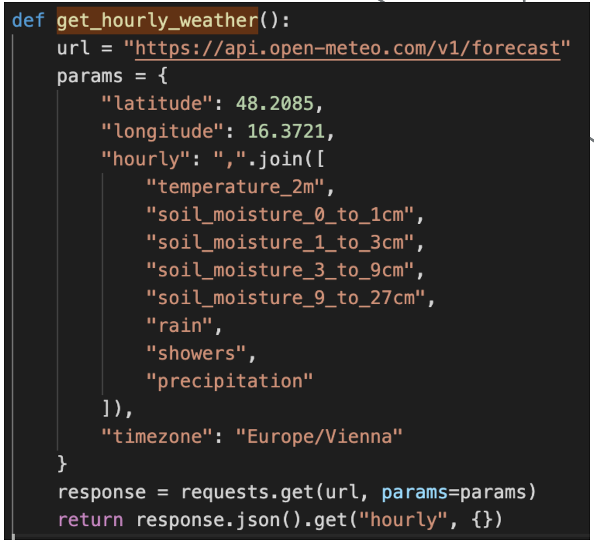
```python
# Determine watering volume by pot size
volume_ml = {
        "Small (500ml)": 100,
        "Medium (1L)": 200,
        "Large (2L)": 400
}[pot_size]
```

```python
type_targets = {
    "Indoor": 55,
    "Outdoor": 60,
    "Desert": 35,
    "Tropical": 70
}
```

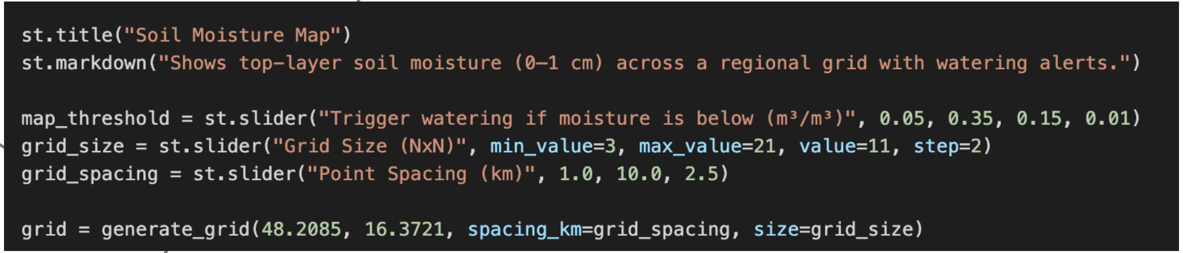
Vienna

Uses Weather Data from Open-Meteo to generate Interactive Soil Moisture, Temperature Visuals

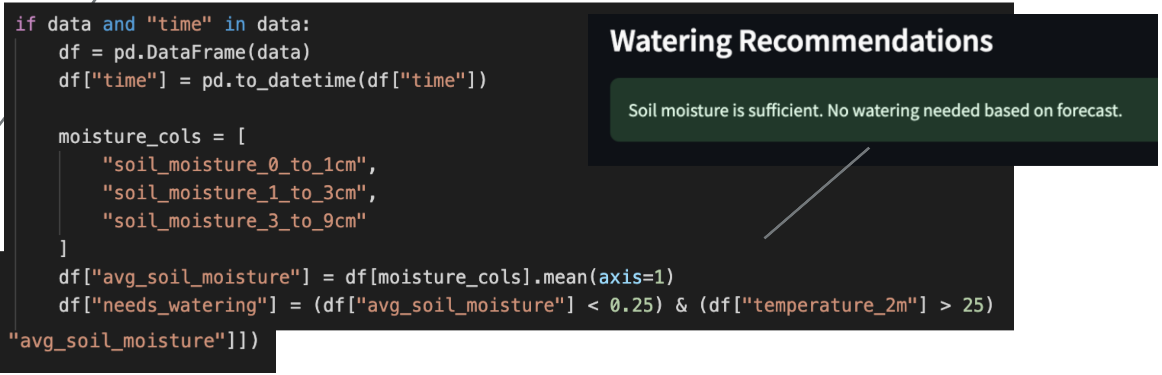




```
def get_hourly_weather():
    url = "https://api.open-meteo.com/v1/forecast"
    params = {
        "latitude": 48.2085,
        "longitude": 16.3721,
        "hourly": ",".join([
            "temperature_2m",
            "soil_moisture_0_to_1cm",
            "soil_moisture_1_to_3cm",
            "soil_moisture_3_to_9cm",
            "soil_moisture_9_to_27cm",
            "rain",
            "showers",
            "precipitation"
        ]),
        "timezone": "Europe/Vienna"
    }
    response = requests.get(url, params=params)
    return response.json().get("hourly", {})
```



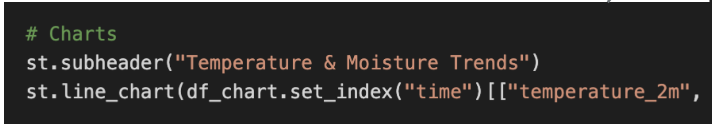
```
st.title("Soil Moisture Map")
st.markdown("Shows top-layer soil moisture (0–1 cm) across a regional grid with watering alerts.")

map_threshold = st.slider("Trigger watering if moisture is below (m³/m³)", 0.05, 0.35, 0.15, 0.01)
grid_size = st.slider("Grid Size (NxN)", min_value=3, max_value=21, value=11, step=2)
grid_spacing = st.slider("Point Spacing (km)", 1.0, 10.0, 2.5)

grid = generate_grid(48.2085, 16.3721, spacing_km=grid_spacing, size=grid_size)
```



```
if data and "time" in data:
    df = pd.DataFrame(data)
    df["time"] = pd.to_datetime(df["time"])

    moisture_cols = [
        "soil_moisture_0_to_1cm",
        "soil_moisture_1_to_3cm",
        "soil_moisture_3_to_9cm"
    ]
    df["avg_soil_moisture"] = df[moisture_cols].mean(axis=1)
    df["needs_watering"] = (df["avg_soil_moisture"] < 0.25) & (df["temperature_2m"] > 25)
```

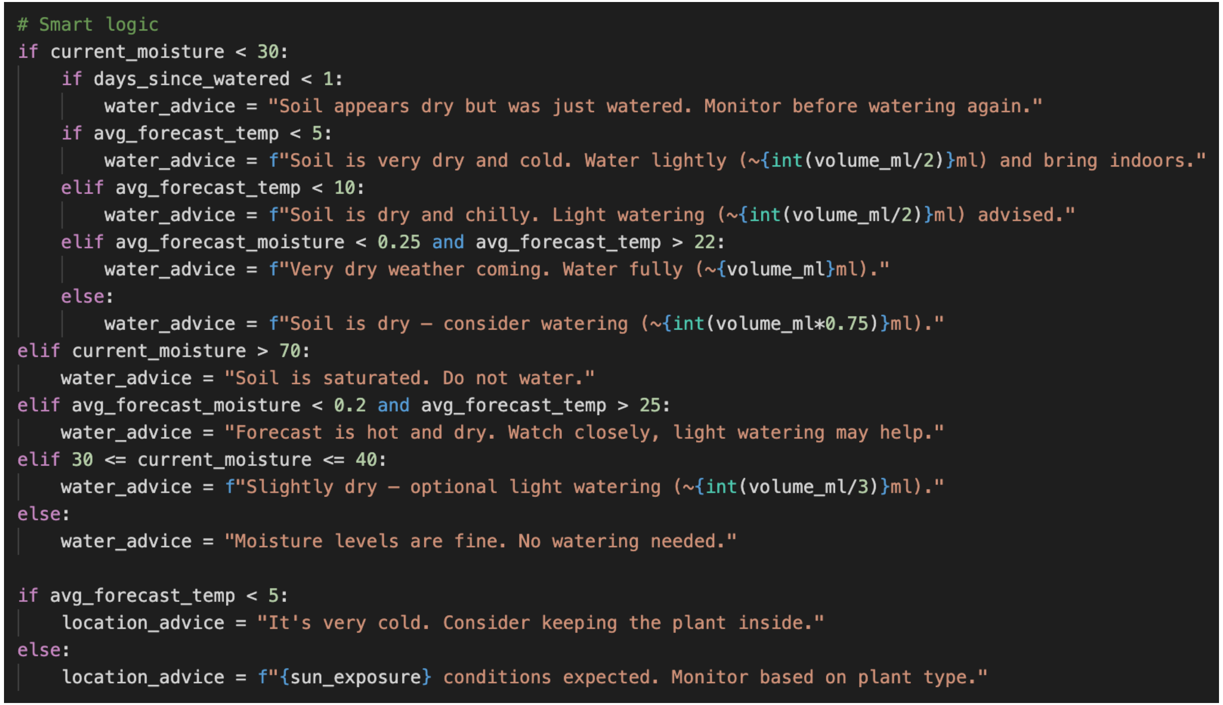
Watering Recommendations

Soil moisture is sufficient. No watering needed based on forecast.



```
# Charts
st.subheader("Temperature & Moisture Trends")
st.line_chart(df_chart.set_index("time")[["temperature_2m", "avg_soil_moisture"]])
```

Smart Advice



```python
# Smart logic
if current_moisture < 30:
    if days_since_watered < 1:
        water_advice = "Soil appears dry but was just watered. Monitor before watering again."
    if avg_forecast_temp < 5:
        water_advice = f"Soil is very dry and cold. Water lightly (~{int(volume_ml/2)}ml) and bring indoors."
    elif avg_forecast_temp < 10:
        water_advice = f"Soil is dry and chilly. Light watering (~{int(volume_ml/2)}ml) advised."
    elif avg_forecast_moisture < 0.25 and avg_forecast_temp > 22:
        water_advice = f"Very dry weather coming. Water fully (~{volume_ml}ml)."
    else:
        water_advice = f"Soil is dry – consider watering (~{int(volume_ml*0.75)}ml)."
elif current_moisture > 70:
    water_advice = "Soil is saturated. Do not water."
elif avg_forecast_moisture < 0.2 and avg_forecast_temp > 25:
    water_advice = "Forecast is hot and dry. Watch closely, light watering may help."
elif 30 <= current_moisture <= 40:
    water_advice = f"Slightly dry – optional light watering (~{int(volume_ml/3)}ml)."
else:
    water_advice = "Moisture levels are fine. No watering needed."

if avg_forecast_temp < 5:
    location_advice = "It's very cold. Consider keeping the plant inside."
else:
    location_advice = f"{sun_exposure} conditions expected. Monitor based on plant type."
```

Outlook

- Extend with rain and light sensor
- Extend with pump and water tank

Live Demo and Handson