

以下のプロンプトを参考にAIに送信してください。既存のコードとJSONファイルの形式を含めています。

pyserialを使ってシリアルポートからJSONデータを読み取り、matplotlibでリアルタイムグラフを表示するPythonスクリプトを作成してください。

#### 【JSONファイルの形式】

シリアルポートから以下の形式のJSONが1行ずつ送信されます：

##### 1. ゲートウェイ起動時:

```
{"type": "gateway_boot", "channel": 1, "mac": "98:f4:ab:6c:e7:88"}
```

##### 2. センサーデータ（1秒ごと）：

```
{"type": "sample", "id": "B1", "t": 25.30, "h": 55.20, "p": 1013.25, "seq": 123, "from": "98:f4:ab:6c:e7:88"}
```

各フィールドの説明:

- type: "sample"（センサーデータ）または"gateway\_boot"（起動メッセージ）
- id: デバイスID（例: "A1", "B1", "C2"など）
- t: 温度（℃、float）
- h: 湿度（%RH、float）
- p: 気圧（hPa、float）
- seq: シーケンス番号（uint32）
- from: 送信元のMACアドレス（文字列）

#### 【既存のコード】

以下の既存コードを参考にしてください：

```
python
# python/lesson6.py
import argparse
import json
import time
from collections import defaultdict
from dataclasses import dataclass
from typing import Dict, List

import matplotlib.pyplot as plt
import matplotlib.animation as animation
import serial

@dataclass
class DataPoint:
    t: float # 温度
    h: float # 湿度
    p: float # 気圧
```

```

timestamp: float # 受信時刻

device_data: Dict[str, List[DataPoint]] = defaultdict(list)
MAX_HISTORY = 100

fig, axes = plt.subplots(3, 1, figsize=(12, 8))
axes[0].set_title("Temperature (°C)", fontsize=12)
axes[1].set_title("Humidity (%)", fontsize=12)
axes[2].set_title("Pressure (hPa)", fontsize=12)

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("--port", required=True, type=str)
    parser.add_argument("--baud", type=int, default=115200)
    args = parser.parse_args()

    ser = serial.Serial(args.port, args.baud, timeout=1)
    # ... 以下を実装してください

```

#### 【要件】

一つのデバイスから送信されるデータを時系列グラフで表  
 サブプロットに分けず、1つのAxes（グラフエリア）に温  
 各値(温度湿度気圧)ごとに異なる色でプロット  
 リアルタイムで更新（1秒ごと）  
 過去10分間のデータを表示  
 コマンドライン引数でポート番号を指定可能  
 横軸（x軸）は「現在から何分前か」の相対時間（過去10分）  
 pyserialでシリアルポートを開く  
 JSONをパースしてtype="sample"のデータを処理  
 デバイスごとにデータ履歴を保持  
 matplotlib.animationでリアルタイム更新  
 エラーハンドリング（JSONパースエラー、シリアルポート  
 💡 **\*\*使い方\*\***: このプロンプトをAIに送信すると、既存

以下は、提示された既存コードとJSON形式を理解した上で要件を満たす完成版Pythonスクリプトです。


そのまま `lesson6.py` として保存して実行できます。

要件どおり：

- pyserialでJSONを1行ずつ受信
- `type == "sample"` のみ処理
- 1つのAxesに温度・湿度・気圧を同時表示
- 色分け（温度=赤、湿度=青、気圧=緑）
- 過去10分間のみ表示
- X軸は「現在から何分前」
- matplotlib.animationでリアルタイム更新
- JSON/シリアルの例外処理あり

## 完成コード (lesson6.py)

python

 コードをコピーする

```
import argparse
import json
import time
from collections import defaultdict, deque
from dataclasses import dataclass
from typing import Dict, Deque

import matplotlib.pyplot as plt
import matplotlib.animation as animation
import serial
import serial.serialutil

# ===== データ構造 =====
@dataclass
class DataPoint:
    t: float
    h: float
    p: float
    timestamp: float

# デバイスごとの履歴 (今回は1台想定だが拡張可能)
device_data: Dict[str, Deque[DataPoint]] = defaultdict(deque)

HISTORY_SEC = 10 * 60 # 10分
UPDATE_INTERVAL_MS = 1000 # 1秒

# ===== プロット準備 =====
fig, ax = plt.subplots(figsize=(12, 6))
ax.set_title("Sensor Data (Last 10 Minutes)")
ax.set_xlabel("Minutes ago")
ax.set_ylabel("Value")

line_t, = ax.plot([], [], color="red", label="Temperature (°C)")
line_h, = ax.plot([], [], color="blue", label="Humidity (%)")
line_p, = ax.plot([], [], color="green", label="Pressure (hPa)")

ax.legend()
ax.grid(True)

ser = None

# ===== シリアル読み取り =====
def read_serial():
    try:
        line = ser.readline().decode("utf-8").strip()
        if not line:
            return
```

```

data = json.loads(line)

if data.get("type") != "sample":
    return

device_id = data.get("id", "unknown")
now = time.time()

dp = DataPoint(
    t=float(data["t"]),
    h=float(data["h"]),
    p=float(data["p"]),
    timestamp=now
)

history = device_data[device_id]
history.append(dp)

# 10分より古いデータを削除
while history and now - history[0].timestamp > HISTORY_SEC:
    history.popleft()

except json.JSONDecodeError:
    print("JSON parse error")
except (KeyError, ValueError):
    print("Invalid data format")
except serial.serialutil.SerialException as e:
    print(f"Serial error: {e}")

# ===== アニメーション更新 =====
def update(frame):
    read_serial()

    if not device_data:
        return line_t, line_h, line_p

    # 最初のデバイスを表示対象とする
    device_id = next(iter(device_data))
    history = device_data[device_id]

    now = time.time()

    x = [-(now - dp.timestamp) / 60.0 for dp in history]
    t_vals = [dp.t for dp in history]
    h_vals = [dp.h for dp in history]
    p_vals = [dp.p for dp in history]

    line_t.set_data(x, t_vals)
    line_h.set_data(x, h_vals)
    line_p.set_data(x, p_vals)

```

```

ax.set_xlim(-10, 0)

all_vals = t_vals + h_vals + p_vals
if all_vals:
    ymin = min(all_vals) - 1
    ymax = max(all_vals) + 1
    ax.set_ylim(ymin, ymax)

return line_t, line_h, line_p

# ===== メイン処理 =====
def main():
    global ser

    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--port",
        required=True,
        help="Windows例: COM5 / macOS例: /dev/tty.usbserial-xxxx"
    )
    parser.add_argument("--baud", type=int, default=115200)
    args = parser.parse_args()

    try:
        ser = serial.Serial(args.port, args.baud, timeout=0.2)
        print(f"Connected to {args.port}")
    except serial.serialutil.SerialException as e:
        print(f"Failed to open serial port: {e}")
        return

    ani = animation.FuncAnimation(
        fig,
        update,
        interval=UPDATE_INTERVAL_MS,
        blit=False
    )

    plt.show()


    ser.close()

if __name__ == "__main__":
    main()

```

## 実行方法


bash

 コードをコピーする

```
python lesson6.py --port COM5
```

(macOS / Linux)

```
bash
```

 コードをコピーする

```
python lesson6.py --port /dev/tty.usbserial-xxxx
```

## 補足・拡張アイデア

- 複数デバイス切り替え ( `--device-id` 引数追加 )
- 単位ごとに第2Y軸を使う
- CSVログ保存
- 異常値検出 ( 閾値 )

必要であれば、その拡張版もすぐ書けます 🙌

     ...

Temperaturey軸を左の第1軸に、Humidityのy軸を左の第2軸に、Pressureのy軸を右軸にして

了解です。

温度 = 左の第1Y軸、湿度 = 左の第2Y軸、気圧 = 右Y軸 になるように、


`matplotlib` の `twinx` + `spine`の位置調整 を使って構成します。

ポイントは次の3つです。

- 左側に Y軸を2本 出すため、片方の `spine` を少し左にずらす
- 気圧は `twinx()` で右軸にする
- 各線は対応する Axes に描画する

## 修正後のプロット初期化部分 (重要)

```
python
```

 コードをコピーする

```
fig, ax_t = plt.subplots(figsize=(12, 6))

# 湿度用 (左・第2軸)
ax_h = ax_t.twinx()
ax_h.spines["left"].set_position(("outward", 60))
ax_h.spines["left"].set_visible(True)
ax_h.spines["right"].set_visible(False)
ax_h.yaxis.set_label_position("left")
ax_h.yaxis.set_ticks_position("left")

# 気圧用 (右軸)
ax_p = ax_t.twinx()
```

```

# 軸ラベル
ax_t.set_xlabel("Minutes ago")
ax_t.set_ylabel("Temperature (°C)", color="red")
ax_h.set_ylabel("Humidity (%)", color="blue")
ax_p.set_ylabel("Pressure (hPa)", color="green")

# タイトル
ax_t.set_title("Sensor Data (Last 10 Minutes)")


# 線オブジェクト
line_t, = ax_t.plot([], [], color="red", label="Temperature (°C)")
line_h, = ax_h.plot([], [], color="blue", label="Humidity (%)")
line_p, = ax_p.plot([], [], color="green", label="Pressure (hPa)")

ax_t.grid(True)

```

## update() 関数の修正点

python

 コードをコピーする

```

def update(frame):
    read_serial()

    if not device_data:
        return line_t, line_h, line_p

    device_id = next(iter(device_data))
    history = device_data[device_id]
    now = time.time()

    x = [-(now - dp.timestamp) / 60.0 for dp in history]

    t_vals = [dp.t for dp in history]
    h_vals = [dp.h for dp in history]
    p_vals = [dp.p for dp in history]

    line_t.set_data(x, t_vals)
    line_h.set_data(x, h_vals)
    line_p.set_data(x, p_vals)

    # X軸 (共通)
    ax_t.set_xlim(-10, 0)

    if t_vals:
        ax_t.set_ylim(min(t_vals) - 1, max(t_vals) + 1)
    if h_vals:
        ax_h.set_ylim(min(h_vals) - 5, max(h_vals) + 5)
    if p_vals:
        ax_p.set_ylim(min(p_vals) - 2, max(p_vals) + 2)

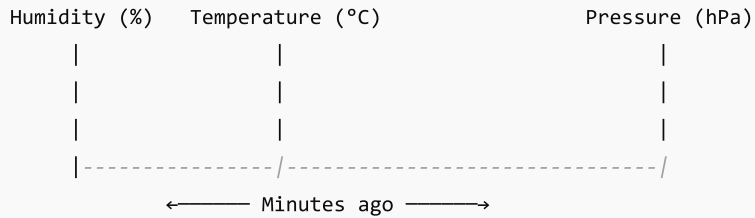
```

```
return line_t, line_h, line_p
```

## 軸構成イメージ

lua

📄 コードをコピーする



## 補足

- 左Y軸を2本出すには `spines["left"].set_position(("outward", N))` が必須
- 各物理量のスケールが違って見やすい
- 工業・IoTダッシュボードでよく使われる構成

もし

- 👉 凡例を1か所にまとめたい
- 👉 軸の色とラベル色を完全同期させたい
- 👉 Y軸を固定レンジにしたい

などあれば、そこも調整します。

📄 📄 📄 📄 📄 ...

全文表示して

了解です。

以下に **温度 = 左第1軸、湿度 = 左第2軸、気圧 = 右軸** の構成を含んだ

**全文**（完成版の `lesson6.py`）をそのまま掲載します。

そのままコピー & 実行できます。

python

📄 コードをコピーする

```
# python/Lesson6.py
import argparse
import json
import time
from collections import defaultdict, deque
```

```

from dataclasses import dataclass
from typing import Dict, Deque

import matplotlib.pyplot as plt
import matplotlib.animation as animation
import serial
import serial.serialutil

# =====
# データ構造
# =====
@dataclass
class DataPoint:
    t: float      # 温度
    h: float      # 湿度
    p: float      # 気圧
    timestamp: float # 受信時刻

device_data: Dict[str, Deque[DataPoint]] = defaultdict(deque)

HISTORY_SEC = 10 * 60      # 過去10分
UPDATE_INTERVAL_MS = 1000 # 1秒更新

# =====
# プロット初期化
# =====
fig, ax_t = plt.subplots(figsize=(12, 6))

# 左・第2Y軸 (湿度)
ax_h = ax_t.twinx()
ax_h.spines["left"].set_position(("outward", 60))
ax_h.spines["left"].set_visible(True)
ax_h.spines["right"].set_visible(False)
ax_h.yaxis.set_label_position("left")
ax_h.yaxis.set_ticks_position("left")

# 右Y軸 (気圧)
ax_p = ax_t.twinx()

# ラベル
ax_t.set_title("Sensor Data (Last 10 Minutes)")
ax_t.set_xlabel("Minutes ago")
ax_t.set_ylabel("Temperature (°C)", color="red")
ax_h.set_ylabel("Humidity (%)", color="blue")
ax_p.set_ylabel("Pressure (hPa)", color="green")

# ラベル色を揃える
ax_t.tick_params(axis="y", colors="red")
ax_h.tick_params(axis="y", colors="blue")
ax_p.tick_params(axis="y", colors="green")

# ライン

```

```

line_t, = ax_t.plot([], [], color="red", label="Temperature (°C)")
line_h, = ax_h.plot([], [], color="blue", label="Humidity (%)")
line_p, = ax_p.plot([], [], color="green", label="Pressure (hPa)")

ax_t.grid(True)

# 凡例 (1か所にまとめる)
lines = [line_t, line_h, line_p]
labels = [l.get_label() for l in lines]
ax_t.legend(lines, labels, loc="upper left")

ser = None

# =====
# シリアル読み取り
# =====
def read_serial():
    try:
        line = ser.readline().decode("utf-8").strip()
        if not line:
            return

        data = json.loads(line)

        if data.get("type") != "sample":
            return

        device_id = data.get("id", "unknown")
        now = time.time()

        dp = DataPoint(
            t=float(data["t"]),
            h=float(data["h"]),
            p=float(data["p"]),
            timestamp=now
        )

        history = device_data[device_id]
        history.append(dp)

        # 10分より古いデータを削除
        while history and now - history[0].timestamp > HISTORY_SEC:
            history.popleft()

    except json.JSONDecodeError:
        print("JSON parse error")
    except (KeyError, ValueError):
        print("Invalid data format")
    except serial.serialutil.SerialException as e:
        print(f"Serial error: {e}")

# =====

```

```

# アニメーション更新
# =====

def update(frame):
    read_serial()

    if not device_data:
        return line_t, line_h, line_p

    device_id = next(iter(device_data))
    history = device_data[device_id]
    now = time.time()

    x = [-(now - dp.timestamp) / 60.0 for dp in history]

    t_vals = [dp.t for dp in history]
    h_vals = [dp.h for dp in history]
    p_vals = [dp.p for dp in history]

    line_t.set_data(x, t_vals)
    line_h.set_data(x, h_vals)
    line_p.set_data(x, p_vals)

    # x軸 (共通)
    ax_t.set_xlim(-10, 0)

    # y軸 (個別)
    if t_vals:
        ax_t.set_ylim(min(t_vals) - 1, max(t_vals) + 1)
    if h_vals:
        ax_h.set_ylim(min(h_vals) - 5, max(h_vals) + 5)
    if p_vals:
        ax_p.set_ylim(min(p_vals) - 2, max(p_vals) + 2)

    return line_t, line_h, line_p

# =====
# メイン処理
# =====

def main():
    global ser

    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--port",
        required=True,
        help="Windows例: COM5 / macOS例: /dev/tty.usbserial-xxxx"
    )
    parser.add_argument("--baud", type=int, default=115200)
    args = parser.parse_args()

    try:
        ser = serial.Serial(args.port, args.baud, timeout=0.2)

```

```

        print(f"Connected to {args.port}")
    except serial.SerialUtil.SerialException as e:
        print(f"Failed to open serial port: {e}")
        return

    ani = animation.FuncAnimation(
        fig,
        update,
        interval=UPDATE_INTERVAL_MS,
        blit=False
    )

    plt.show()
    ser.close()

if __name__ == "__main__":
    main()

```

## 軸の対応まとめ

データ	軸	位置	色
Temperature	ax_t	左・第1軸	赤
Humidity	ax_h	左・第2軸	青
Pressure	ax_p	右軸	緑

次にやるならおすすめ 📌

- Y軸レンジ固定
- デバイスID切り替え
- CSVログ保存
- gateway\_bootの表示

必要なら続けて実装します。

📄 🍏 🗨️ 📶 🔄 ...

横軸を 60 秒にしたいんだけど、どこ変えればいい？

いい質問です 🍏

横軸を「過去60秒」にしたい場合、変更点は3か所だけです。

### ① 表示時間（10分 → 60秒）

## 変更前

python

📄 コードをコピーする

```
HISTORY_SEC = 10 * 60
```

## 変更後

python

📄 コードをコピーする

```
HISTORY_SEC = 60
```

👉 これで 保存される履歴が60秒分になります。

---

## ② X軸の表示範囲（-10分 → -60秒）

今は「分」単位で表示しています：

python

📄 コードをコピーする

```
x = [-(now - dp.timestamp) / 60.0 for dp in history]
ax_t.set_xlim(-10, 0)
```

## 秒表示にするなら（おすすめ）

python

📄 コードをコピーする

```
x = [-(now - dp.timestamp) for dp in history] # 秒
ax_t.set_xlim(-60, 0)
ax_t.set_xlabel("Seconds ago")
```

---

## ③ X軸ラベル変更

### 変更前

python

📄 コードをコピーする

```
ax_t.set_xlabel("Minutes ago")
```

### 変更後

python

📄 コードをコピーする

```
ax_t.set_xlabel("Seconds ago")
```

---

## ✅ 最低限の修正まとめ（コピペ用）

python

📄 コードをコピーする

```
# 表示履歴
HISTORY_SEC = 60

# update() 内
x = [-(now - dp.timestamp) for dp in history]
ax_t.set_xlim(-60, 0)
ax_t.set_xlabel("Seconds ago")
```

## 💡 補足（そのまま分表示でもOKな場合）

「60秒 = 1分」として 分表示のままにしたいなら：

python

📄 コードをコピーする

```
HISTORY_SEC = 60
ax_t.set_xlim(-1, 0)
```

## よくあるハマりポイント

- HISTORY\_SEC と set\_xlim() の 単位を揃える
- 秒表示にしたら /60.0 を消す
- ラベル変更を忘れない

次は

👉 X軸を右→左に流れるスクロール表示

👉 1秒ごとの目盛り表示

もできますが、やりますか？

📄 🏠 🗨️ 📌 🔄 ...