

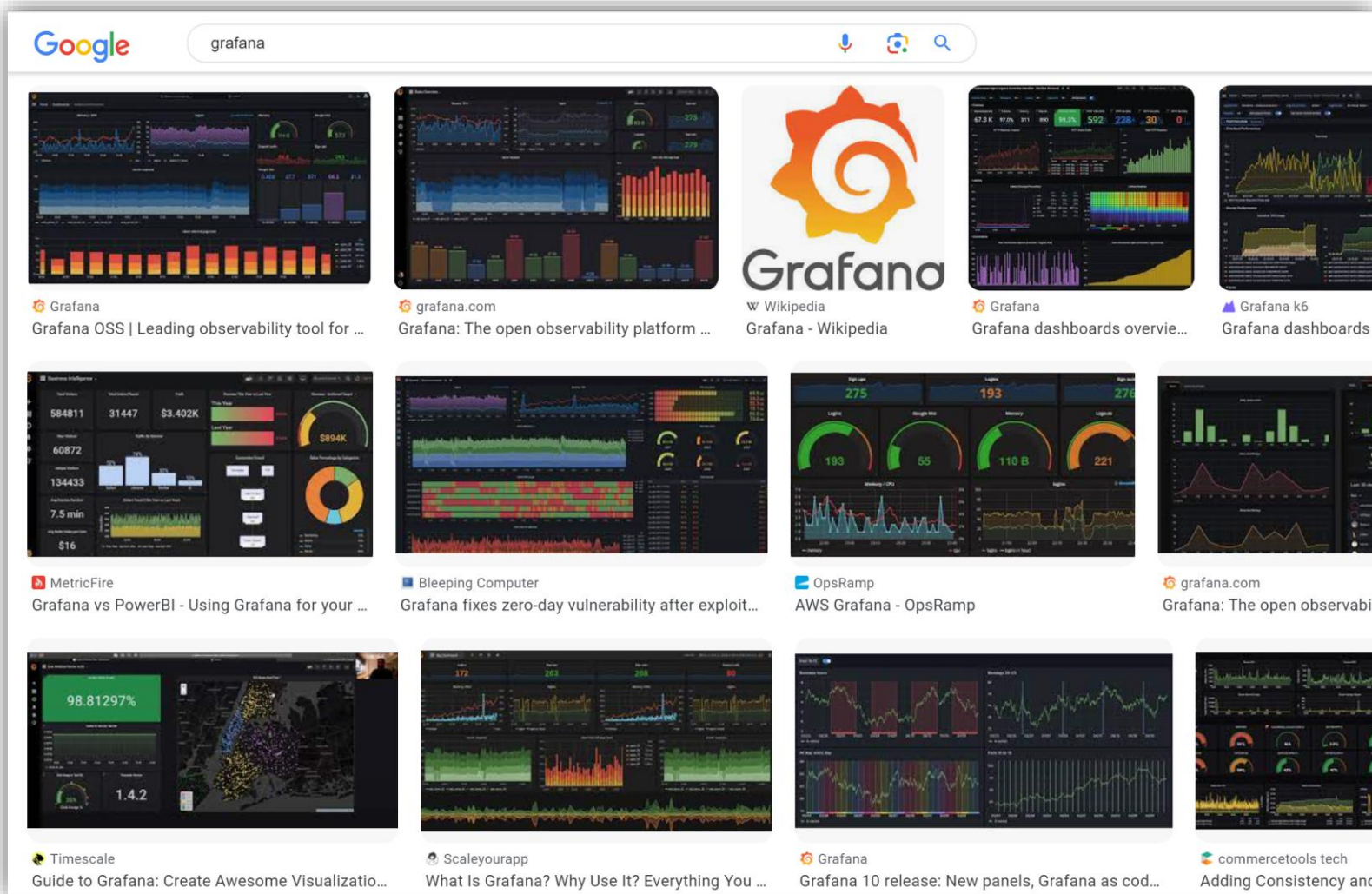
コントロールとモニタのウェブツール

榎本 三四郎
ワシントン大学

Grafana でかっこいいダッシュボードを作れ と言われた

Slow-Controls 用ダッシュボードを作る。

データを取る部分はすでに稼働していたので、UI だけあれば良かった



(ダークモードはかっこいいけど絵をスライドに貼れない...)

Grafana とは

- データベース上にあるデータ(主に時系列)を見える化する Web ベースのソフトウェア
- マウスクリックでダッシュボードを作成して共有できる

Google ×

Qiita
https://qiita.com > prometheus

10分で理解するGrafana #prometheus

2018/07/10 — Grafana とは Grafana は Grafana Labs が公開しているログ・データ可視化のためのツールです可視化ツールとしては kibana とほぼ同じようなもの ...

関連する質問 :

Grafanaは何ができますか？

Grafanaというのは、ざっくりと説明すると、色々なデータベースやデータソースから取得したデータをグラフにしたり、監視したりできる、モニタリングやデータ監視のためのOSS（無料で使えるソフトウェア）です。 2023/02/09



SIOS Tech. Lab
https://tech-lab.sios.jp > OSS

よくわかるGrafana入門【ダッシュボード編①】

検索: Grafanaは何ができますか？

Grafanaとは何ですか？

Grafanaとは Grafanaとは、 Grafana Labs社が開発したデータ可視化ツールです。 Grafanaを利用するためには元のデータが必要であるため、データを収集するツール（PrometheusやElasticsearch等）と組み合わせて使われます。



Grafana

Grafanaは、分析およびインタラクティブな視覚化を可能にする、マルチプラットフォームで動作するオープンソースのWebアプリケーションである。サポートされているデータソースに接続することで、Webブラウザ上でチャート、グラフ、アラートの機能を提供する。 [ウィキペディア](#)

プログラミング言語: Go (プログラミング言語); TypeScript

ライセンス: GNU AGPL

対応OS: Cross-platform

最新版: 7.5.3 / 2021年4月7日 (2年前)

開発元: Grafana Labs

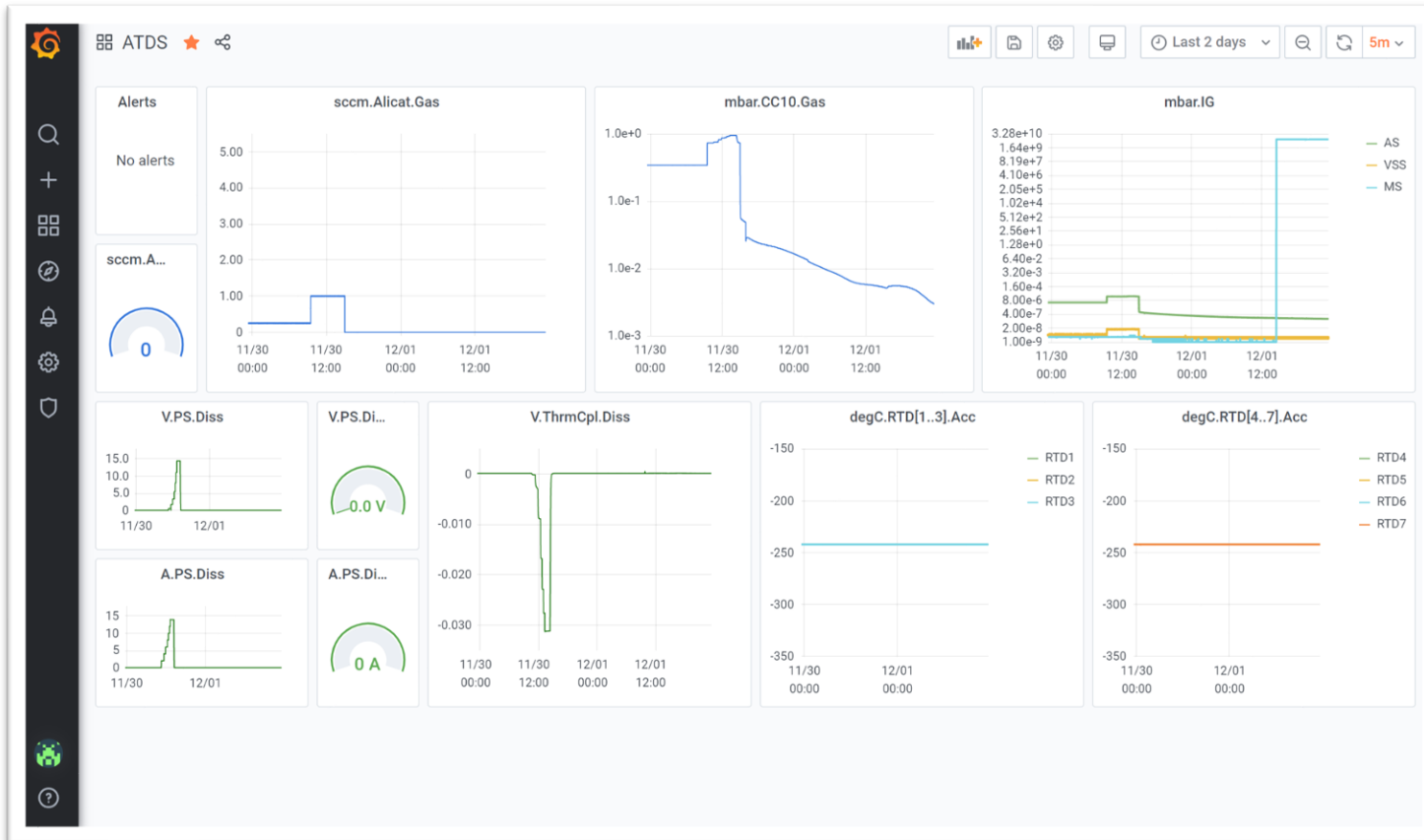
他の人はこちらを検索 他 10 件以上を表示

Prometheus
 Kibana
 InfluxDB
 Zabbix

[フィードバック](#)

3週間がんばってみた

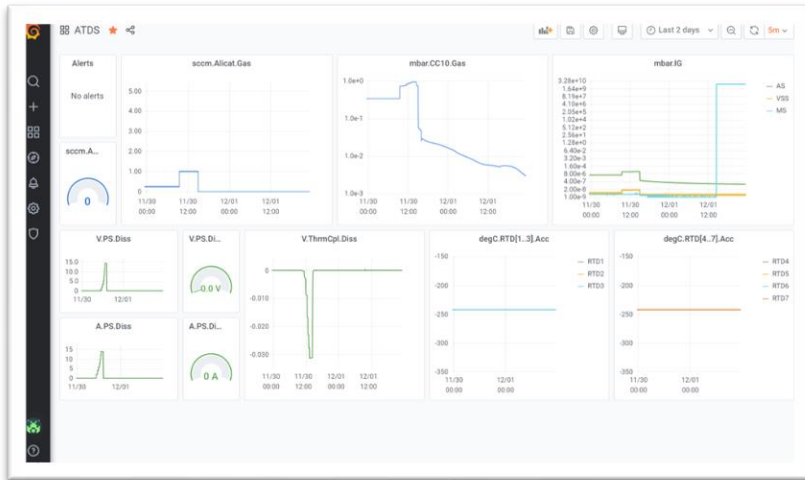
- 構成の決まった表示には良い. 対話的なデータブラウジングはほぼ無理
- 科学データ解析用途には作られていないと思う (データサイエンス用途 ☺)



- 表示範囲を簡単に換えられない. 対数軸に簡単に切り替えられない
- 一時的なプロットを作るのが大変
- データのダウンロードができない (深いところにある「開発モード」にいけばできるけど...)
- ヒストグラム, 誤差バー付きグラフ, 散布図, etc が素直でない

Grafana にもいいところはある

良いと思ったところ

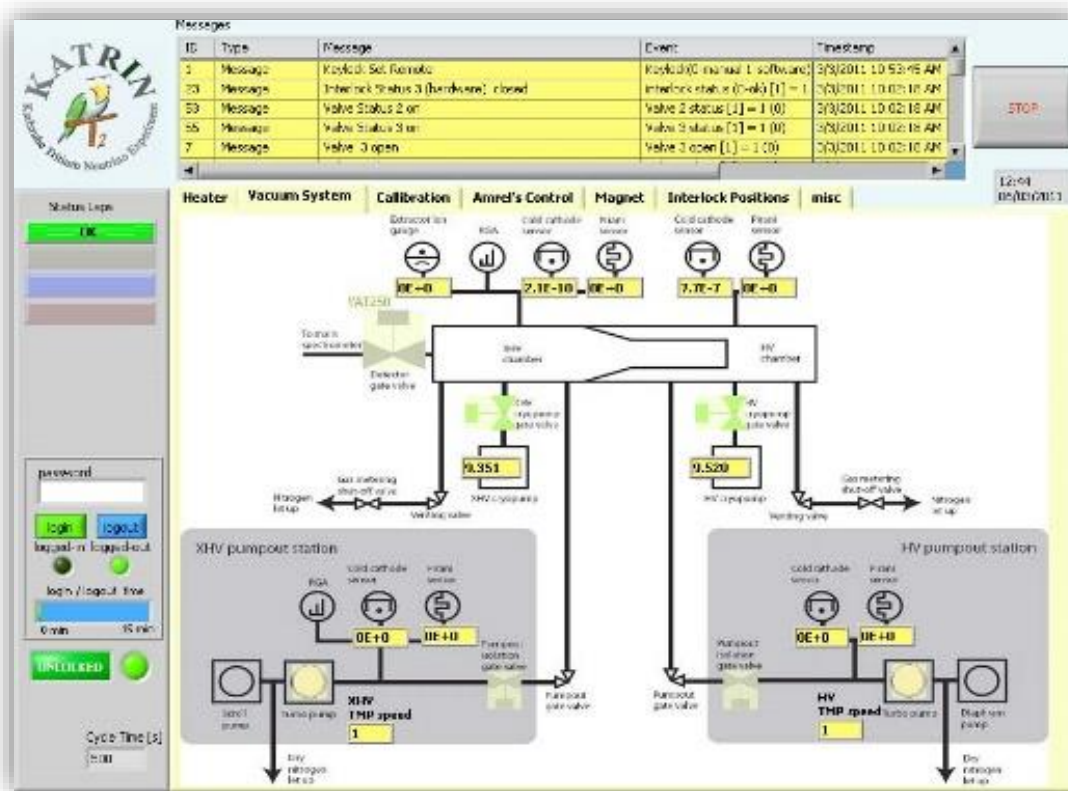


- ダッシュボードをマウス操作だけで作成できる（サーバーにログインなくて良い）
- 作ったダッシュボードを保存して共有できる
- ダークモードにするとかっこいい
- 基本的に全てプラグインで、それを画面上で組み合わせる感じ
- 表示時間範囲の操作感もいい感じ

足りないと思ったところ

- ダッシュボードなら装置全体の状態を視覚化したい
- 表示した装置は操作したい
- 表示したデータは解析したい

LabVIEW: 伝統的な装置状態の見える化と制御



良いかもしれないところ

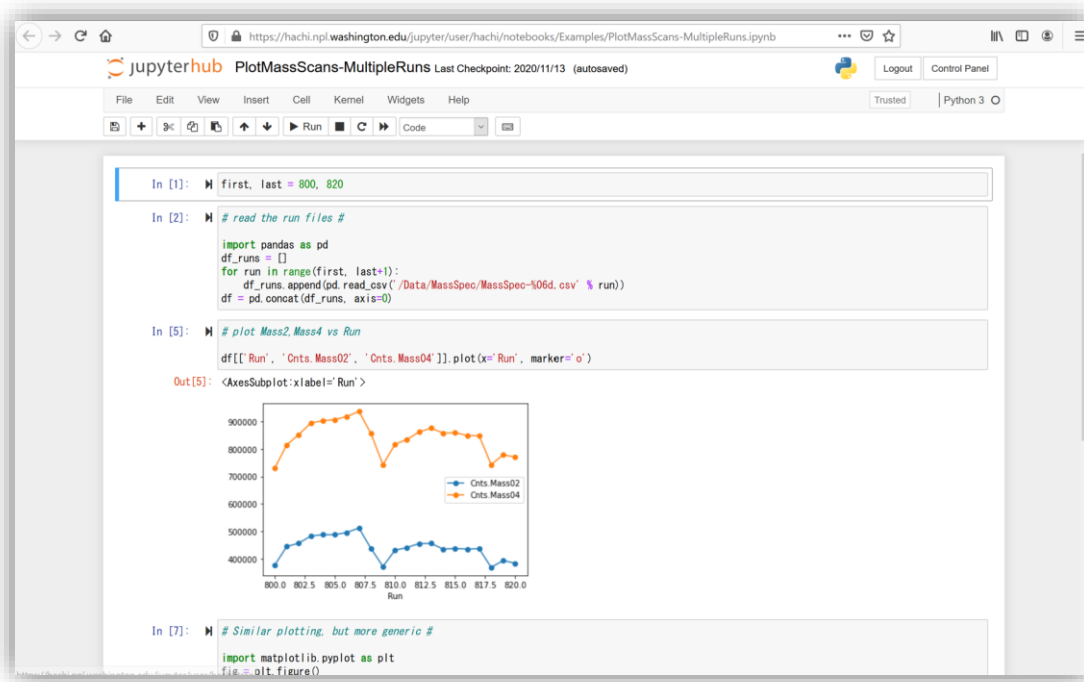
- 好きな人も多い操作パネル
- 全体の状態を俯瞰できる表示

好きでなかったところ

- いろいろ
- データのビジュアライズは基本的でない (プロットを配置できることは知っています)
- 対話的なデータ解析もたぶん想定されていない

EPICS は使用機会がなかったので
分かりません. すいません.

対話的なデータ解析とえば Jupyter-Notebook



文句なく良いところ

- プログラミングを知らなくても思いつきで発見的にコードを書ける
- ウェブ上で使える（サーバーにログインしなくて良い）
- みんな大好きなパイソン

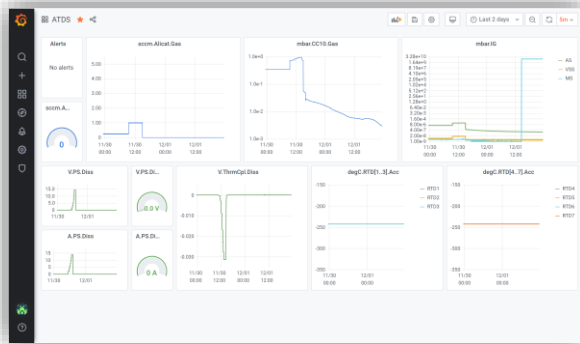
向いてないこと

- 実時間処理はほぼ不可能（定期的にセルを連打するロボットを作れば可）
- ユーザ操作の GUI を作るのにも向いていない（シーケンスを記述するのにはむしろ向いているかも）

欲しいものリスト

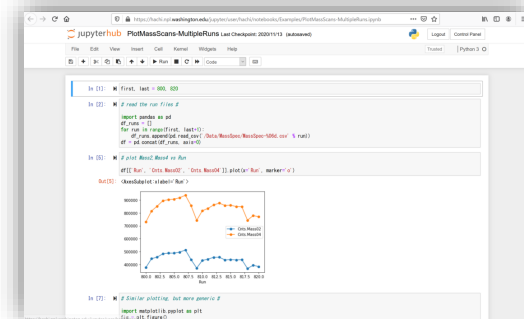
- 基本: 実時間で蓄積するデータを簡単にビジュアライズしたい (Grafana 風)
 - 全体を俯瞰する表示と, 装置のコントロールも欲しい (LabVIEW 風)
 - データは解析をしたい. マウスでもやりたいし, パイソンも使いたい (Jupyter 風)
 - 実時間データ解析からアラーム発行や自動的な装置制御もしたい

Grafana

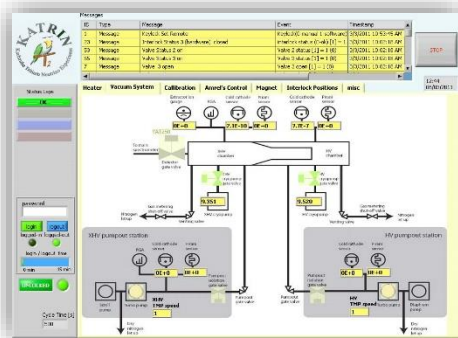


- コンテナに入ってるプロット
- Web ユーザによる作成と共有

Jupyter-Notebook



LabVIEW

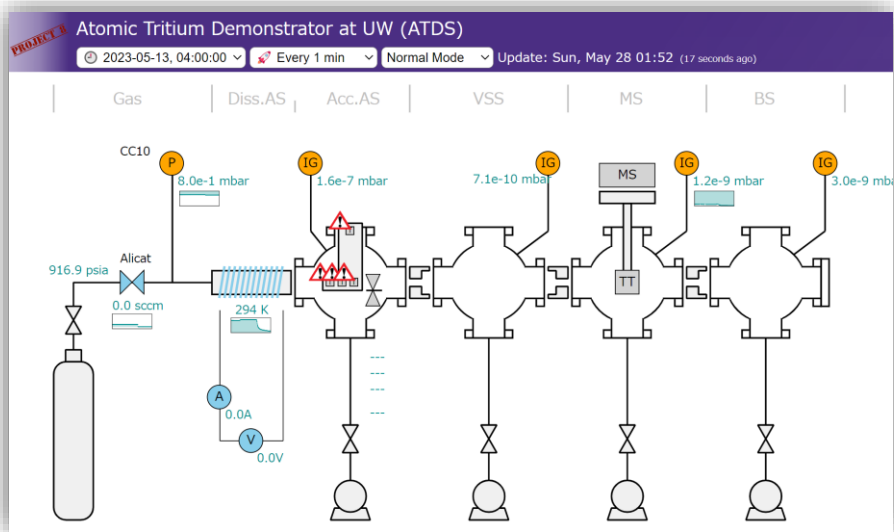


- 図式的状態表示
- コントロール機能

- 思いつき Python スクリプティング
- 豊富なライブラリ

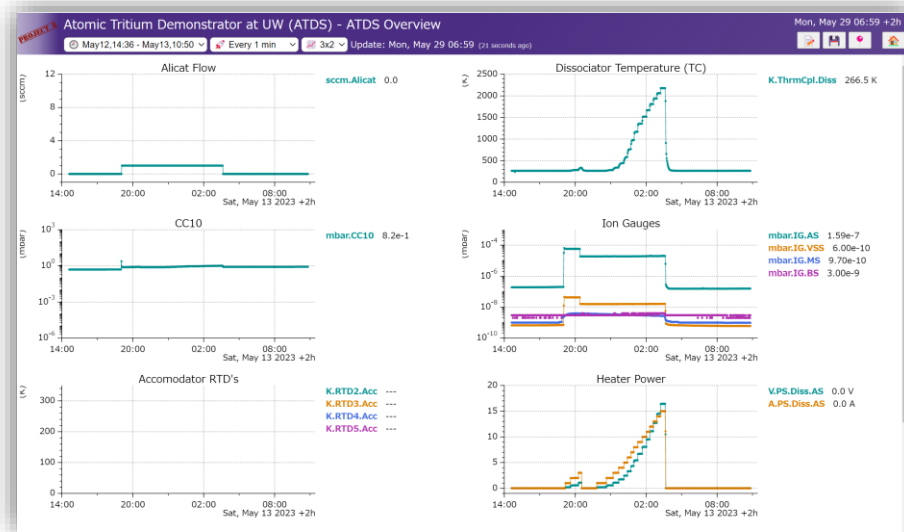
設計：時間断面と空間断面のデュアルビュー

LabVIEW 風のダッシュボード（固定時間）



- 色で運転状態表示, アイコンで異常状態表示
- クリックすると時系列データを表示(右画面)

Grafana 風のプロット(時系列)



- インタラクティブプロット(ズーム, 対数軸, ...)
- レイアウトをマウスで作成
- 作成したページを保存, 共有可
- データダウンロード

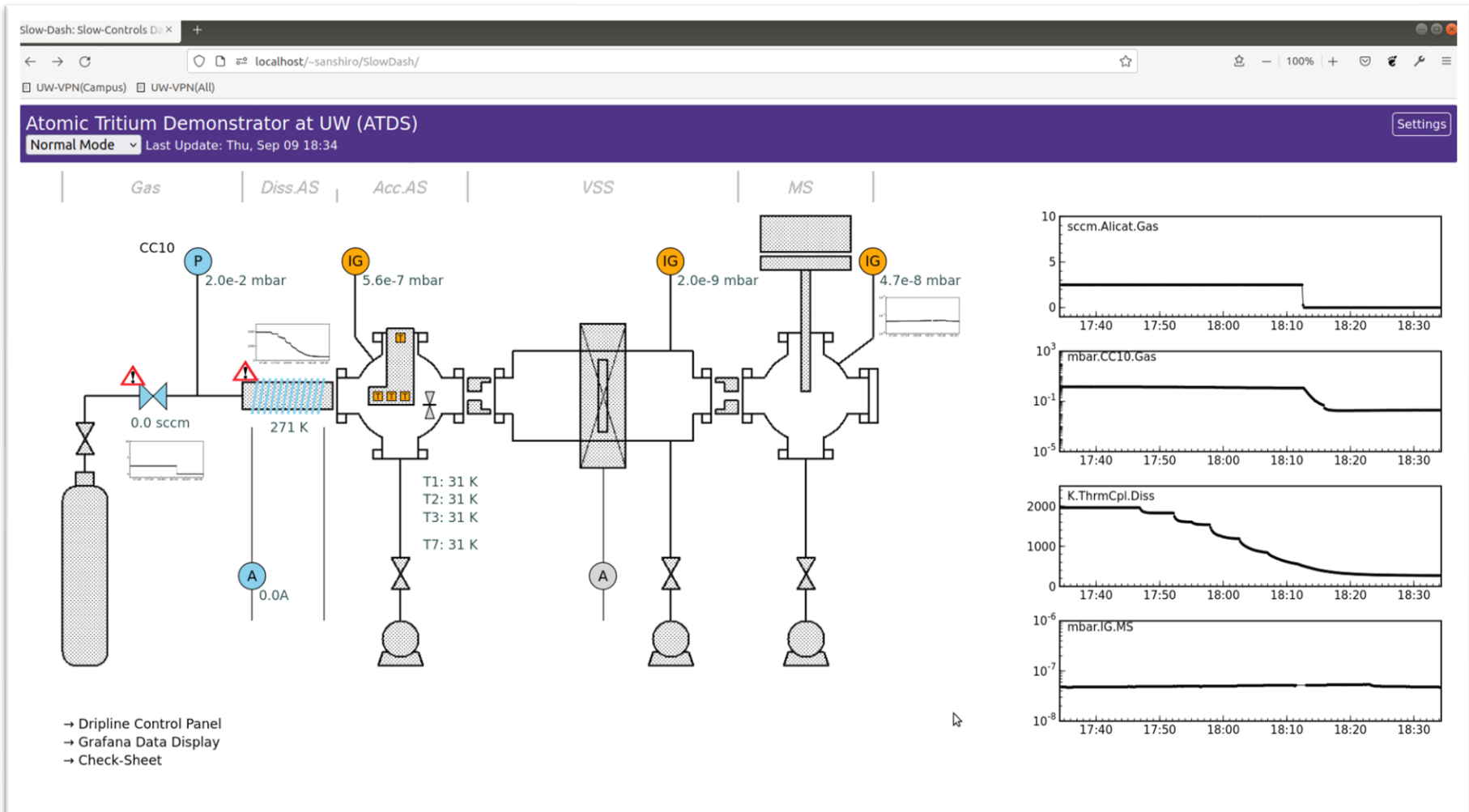
プロットの構成は URL からでもできる. 作成した複雑なページを URL にエンコードすることもできる.

hachi.npl.washington.edu/~hachi/SlowDash/ATDS/slowplot.html?channel=V.ThrmCpl&time=2023-05-13,13:00

- いろいろなプロットへのリンクをあちこちに作れる
- Slack や ELOG に投稿できる

ダッシュボード

- 静止画の上にデータ要素を並べる. YAML ファイルで記述.
- 色で On/Off 状態を表示 (正常/異常ではなく)
- 最新値の数値表示と、いくつかのトレンドグラフ(表示範囲は固定)



ダッシュボード: YAML 設定ファイル例

```
viewBox: 0 0 1600 700
```

```
widgets:
```

- type: image
attr: { x: 50, y: 49, height: 503, width: 933, href: "ATDS.png" } Base image

- type: circle
attr: { x: 180, y: 95, width: 40, height: 40, label: P } Pressure Gauge 1
data: { channel: "mbar.Gas", "active-below": 500, format: "%.1e mbar" }
Sensor name

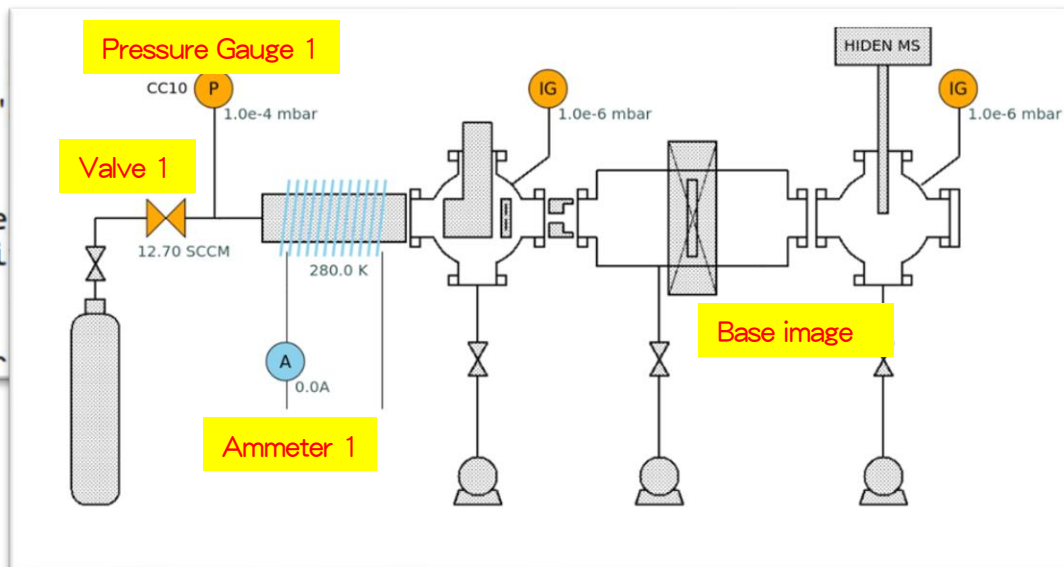
- type: valve
attr: { x: 130, y: 230, width: 40, height: 40, orientation: h, "data-dx": -10, "data-dy": 60 }
data: { channel: "SCCM.Gas", "active-above": 0.1, format: "%.1f SCCM" } Valve 1

- type: circle
attr: { x: 255, y: 430, width: 40, height: 40, label: A } Ammeter 1
data: { channel: "A.PS.Diss", "active-above": 0.1, format: "%.1fA" }

- type: solenoid
attr: { x: 270, y: 210, width: 110, height: 40, href: "Solenoid.png" }
data: { channel: "K.ThermCo.Diss", "active-above": 0.1, format: "%.1f" }

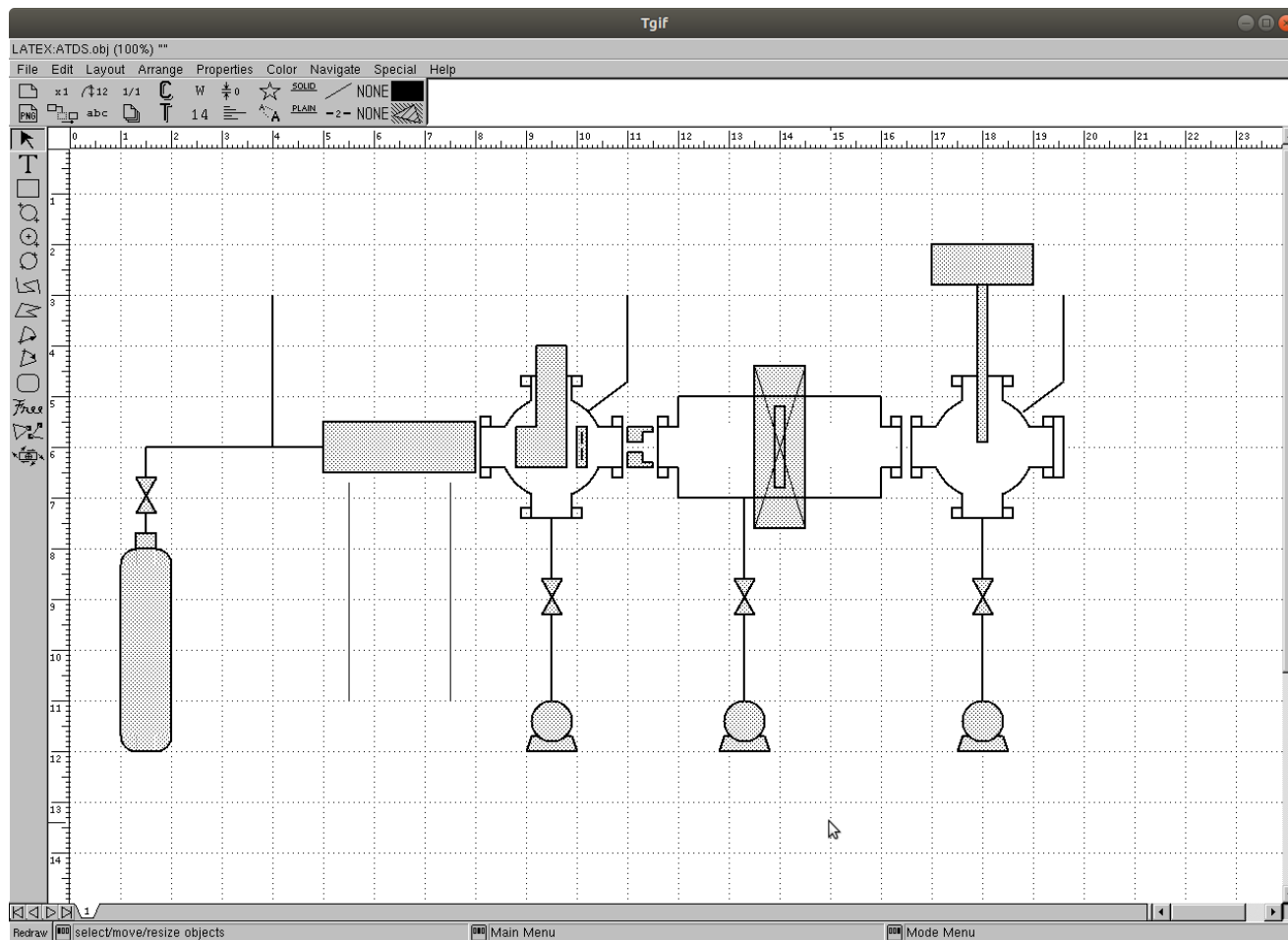
- type: circle
attr: { x: 530, y: 95, width: 40, height: 40, label: IG }
data: { channel: "mbar.IG.AS", "active-below": 500, format: "%.1e mbar" }

- type: circle
attr: { x: 900, y: 95, width: 40, height: 40, label: IG }
data: { channel: "mbar.IG.BS", "active-below": 500, format: "%.1e mbar" }



ダッシュボード: 静止画のベースイメージ

- フォーマットやサイズは自由 (設定ファイルでスケールできる)
- データ要素を並べるためには, 荒いグリッドを使うのがよい



- ここでは tgif を使った. 編集可能な図がテキストなので Git での管理が簡単

ダッシュボード:データ要素図形

- Javascript 10 行くらいで簡単に図形要素を追加できる

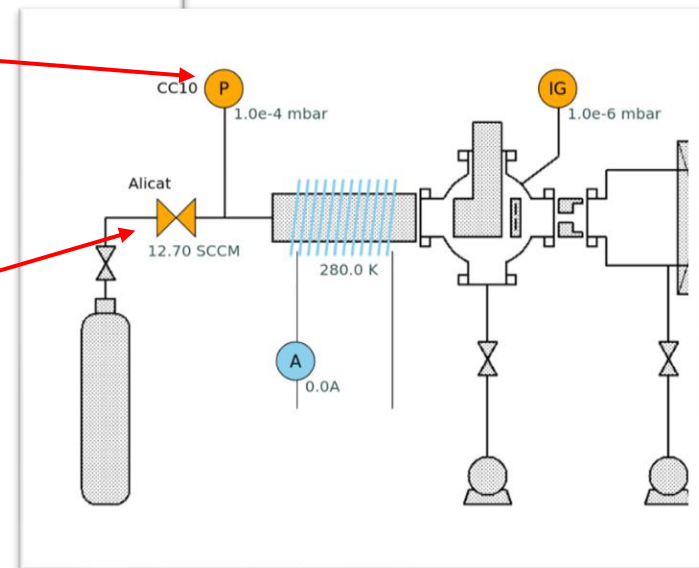
```

class SCCircleWidget extends SCShapeWidget {
  get_defaults() {
    return $.extend({}, super.get_defaults(), {
      width: 20, height: 20
    });
  }
  create_path() {
    let circleAttr = $.extend({}, this.attr, {
      cx: this.attr.x + this.attr.width/2,
      cy: this.attr.y + this.attr.height/2,
      rx: this.attr.width/2,
      ry: this.attr.height/2
    });
    return $('<ellipse>', 'svg').attr(circleAttr);
  }
};

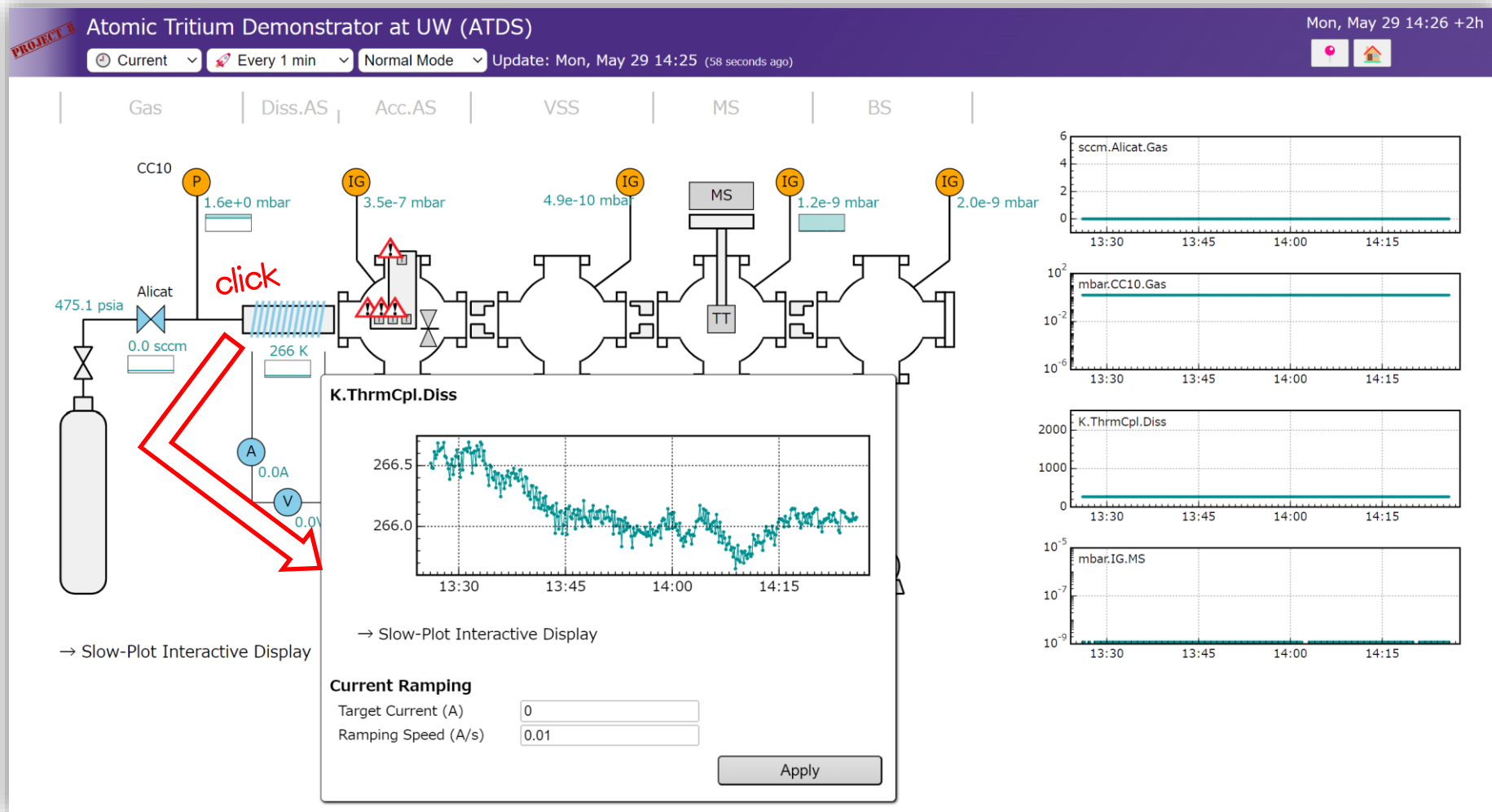
class SCValveWidget extends SCShapeWidget {
  get_defaults() {
    return $.extend({}, super.get_defaults(), {
      width: 20, height: 20, orientation: 'horizontal'
    });
  }
  create_path() {
    let [x0, y0] = [this.attr.x, this.attr.y];
    let [x1, y1] = [x0 + this.attr.width, y0 + this.attr.height];
    let points = (
      (this.attr.orientation[0] == 'v') ?
      `${x0},${y0} ${x1},${y0} ${x0},${y1} ${x1},${y1} ${x0},${y0}` :
      `${x0},${y0} ${x0},${y1} ${x1},${y0} ${x1},${y1} ${x0},${y0}`
    );
    return $('<polyline>', 'svg').attr(this.attr).attr({'points': points});
  }
};

class SCSolenoidWidget extends SCShapeWidget {
  get_defaults() {

```

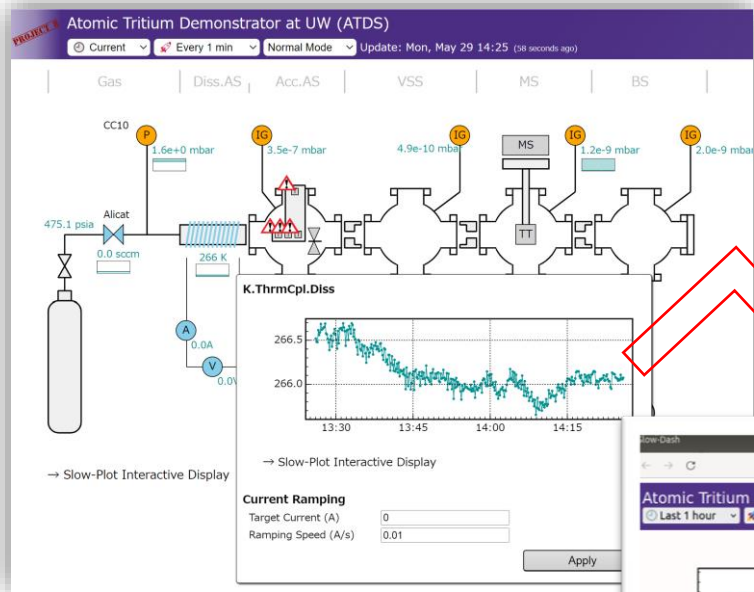


ダッシュボード: チャンネル詳細と操作ダイアログ



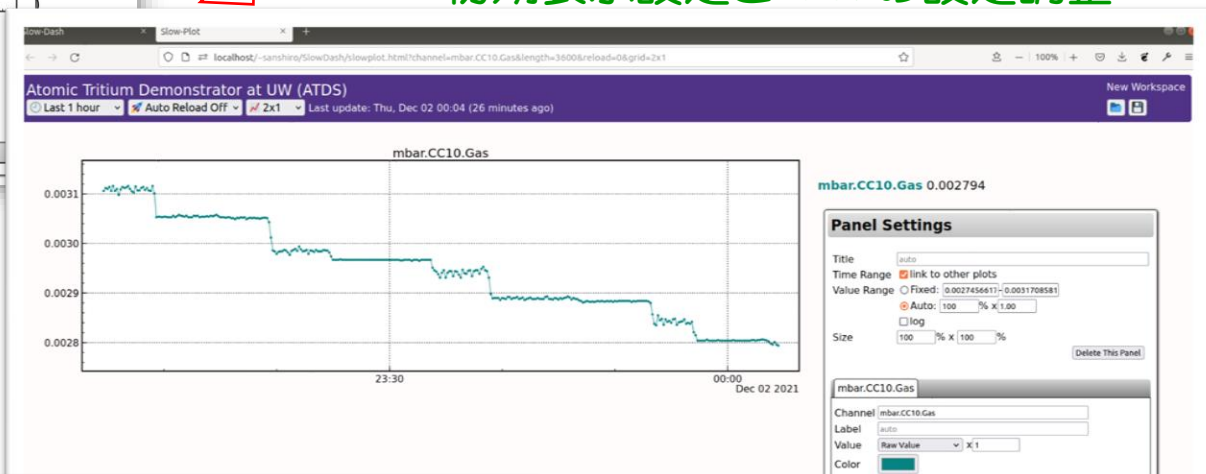
- 直近データ値推移
- デバイス操作
- プロパティ

対話的データブラウザ



クリックで時系列プロットに飛ぶ

初期表示設定と GUI の設定調整



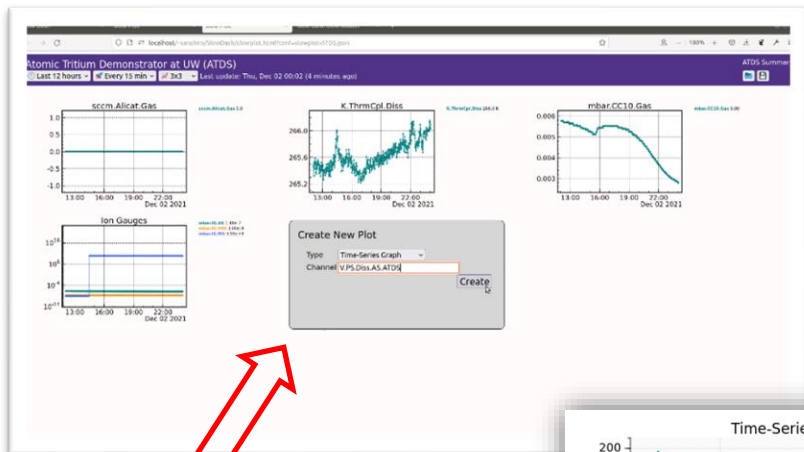
URL によるプロット構成

ishiro/SlowDash/slowplot.html?channel=mbar.CC10.Gas&length=3600&reload=

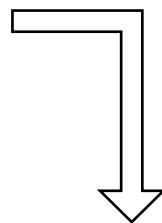
- 設定ファイルなし. すぐに始められる
- いろいろな場所からリンクを貼って, それぞれ違った構成で開始できる
- 最初のレイアウトを出発点として対話的に構成できる

対話的データブラウザ

新しいプロットの追加も簡単



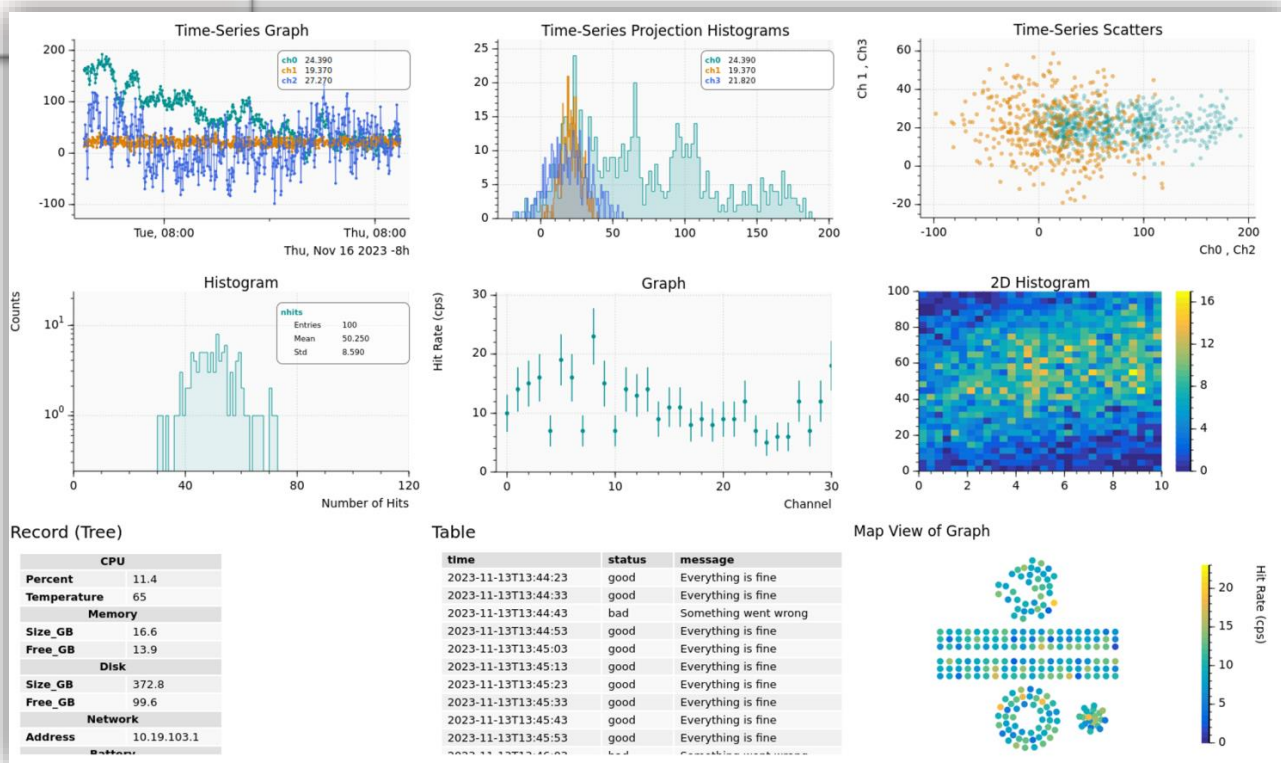
マウスでいろいろ追加していく



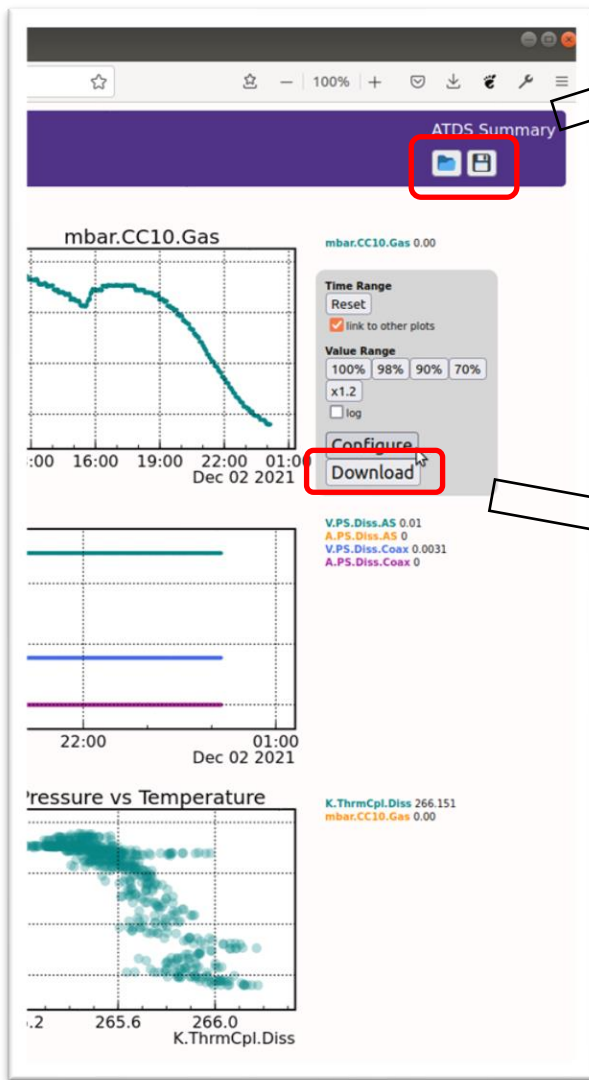
「パネル追加」

いろいろなプロットタイプ

- 時系列グラフ
- 時系列値のヒストグラム
- 複数時系列値の散布図
- ヒストグラム
- エラーバー付きグラフ
- 2次元ヒストグラム
- 表
- ツリー
- マップビュー
- ...



対話的データブラウザ



作ったレイアウトの保存と共有

Atomic Tritium Demonstrator at UW (ATDS)
Slow-Dash Catalog

Settings

Dashboards

- [ATD](#)
- [Coaxial Cracker](#)

Plots

- [Blank](#)
- [Blank 2x2](#)
- [ATDS](#)
- [Coax](#)

プロット表示データのダウンロード

Atomic Tritium Demonstrator at UW (ATDS)
Slow-Dash Downloader

Channels

- V.PS.Diss.AS
- A.PS.Diss.AS
- V.PS.Diss.Coax
- A.PS.Diss.Coax

Add Channel: Add
(can use wildcards; examples: *.PS.*.AS, *.Coax)

Data Time Range

from 2021 / 12 / 01 17 : 55 to 2021 / 12 / 01 18 : 10

Resample at s intervals

Note: dates above are local time, dates in data files are (currently) UTC.

Download CSV

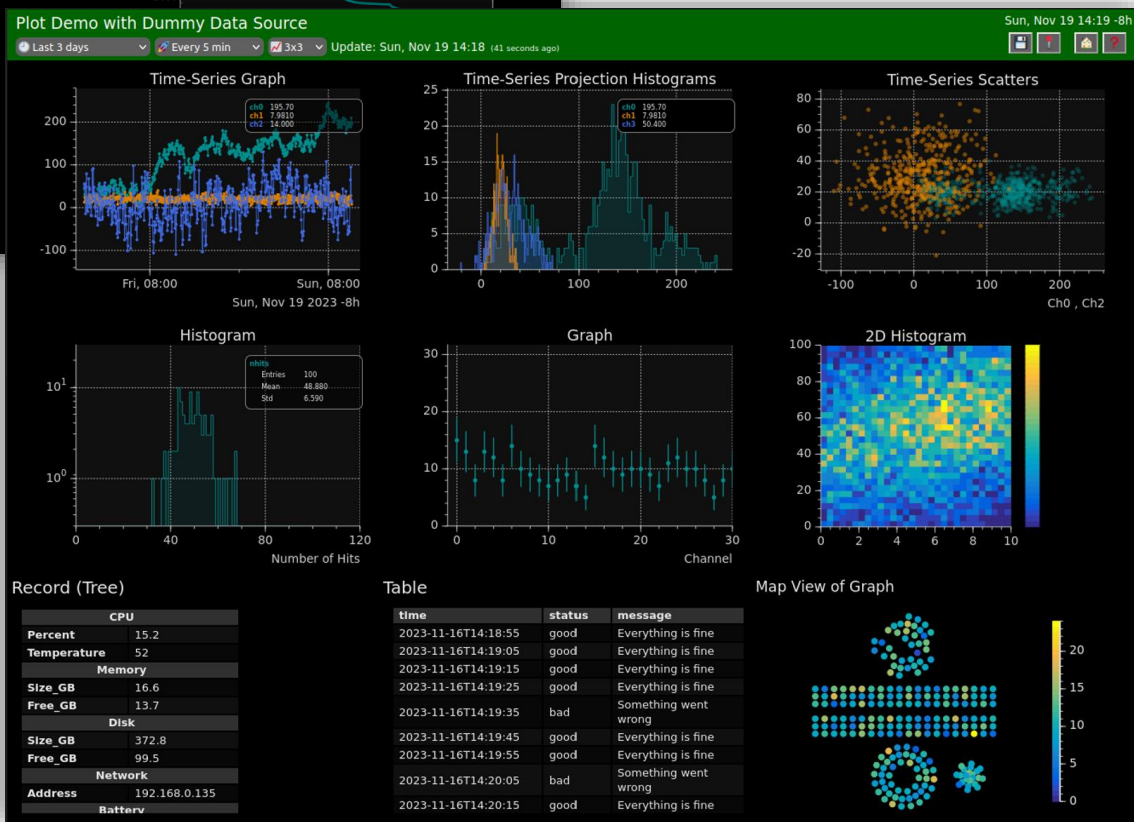
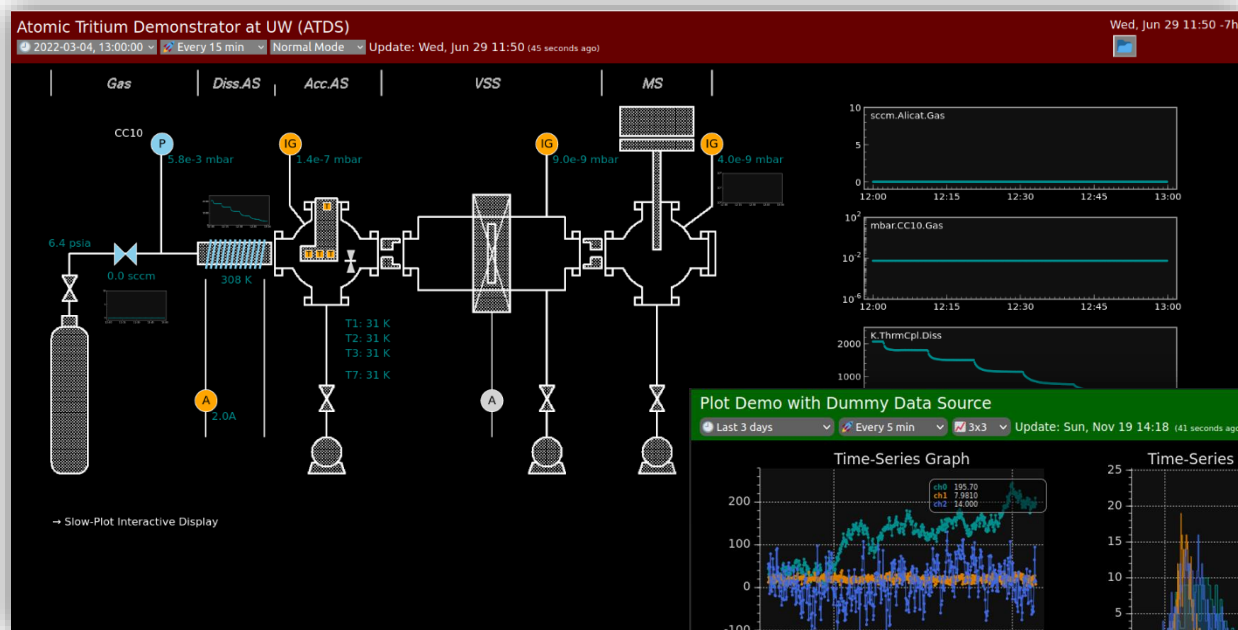
チャンネルはあらかじめ選ばれてる追加もできる

フィルドカード使用可

時間範囲もプロットから設定

テーブル形式へのリサンプリング

「Grafana みたいに作れ」と言われたときのための ダークモード



コマンド受け取りインターフェース

設定値の変更

コマンド送信

ユーザ作成 HTML フォーム

Slow-Plot: WaferTestDAQ

Atomic Tritium Demonstrator at UW (ATDS)

2022-03-05, 03:00:00

Every 15 min

Normal Mode

Update: Tue, Sep 27 07:07 (21 seconds ago)

Gas | Diss.AS | Acc.AS | VSS | MS | BS

CC10 3.8e-3 mbar

IG 2.9e-7 mbar

IG 9.0e-9 mbar

MS

IG 4.0e-9 mbar

6.4 psia

Alicat

0.0

sccm.Alicat.Gas

→ Slow-Plot Interactive Display

Injection Flow

Set-point (sccm) 15

Apply Set-point

Close Valve

→ Slow-Plot Interactive Display

Software Restarting (for Sanshiro)

IG Dripline

K&S ext. port

Tunnel for I

Slow-Plot: WaferTestDAQ

FPD Wafer Test 2022 DAQ

Last 15 minutes

Every 2 s

Update: Tue, Sep 27 06:12 (2 seconds ago)

ADC

Sampled Trace

Trace

Shaped ADC Peak Height

ADC

Event Rate

EventRate

Rate (cps)

Time from Start (s)

Run Settings & Status

Run Info	
Run Number	86
Start Time	Tue, Sep 27 2022, 06:09:14
Status	Running (Offline)
Lapse	00:02:55
Data File Size	0
DAQ Config	
Channel Mask	0b0000001
Trace Length	4096
Post Trigger Fraction	0.3
Trapezoidal Shaping	
Shaping Length	512
Gap Length	128
Threshold	0
Trace Recording Interval	5

Log Messages

Level	Time	Message
Info	Tue, Sep 27 2022, 06:09:14	start data taking
Info	Tue, Sep 27 2022, 06:09:14	output file: /dev/null

Run Settings

Shaping Length 512

Gap Length 128

Trace Length 4096

Post-Trigger Fraction 0.3

Histogram Bins

nbins: 300 min: 0 max: 300 rebin

Run Control

Stop after 3600 5

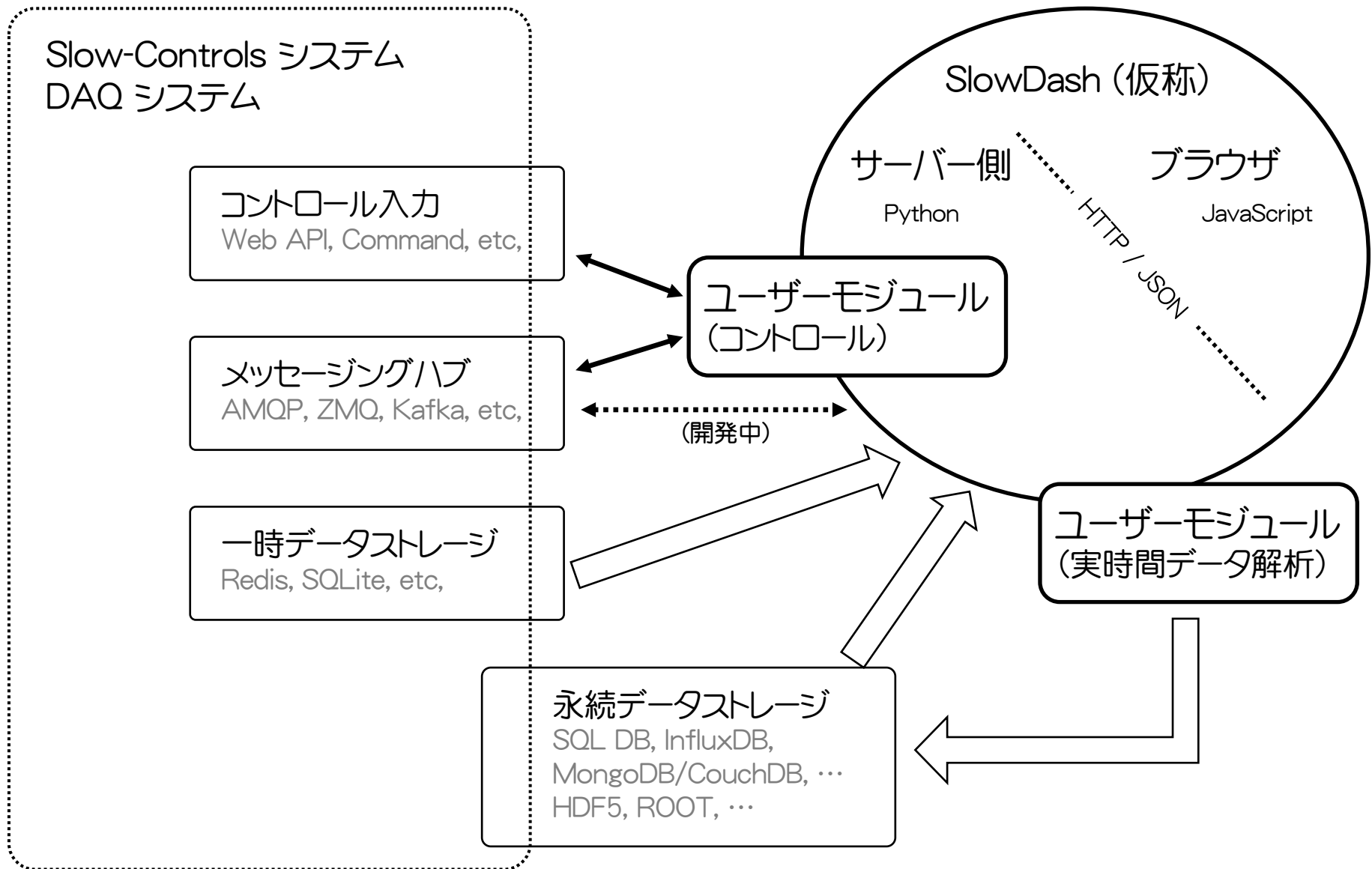
Then repeat

Offline

Start / Stop

- サーバー側のユーザディレクトリにユーザーの Python スクリプトを置いておく
- ボタンをクリックすると入力フィールド値の JSON ドキュメントがユーザコードに渡される

SlowDash: コントロールとモニタの Web ツール



ユーザーモジュール

ごく普通の Python スクリプト

```
import sys, os, logging
import psycopg2 as db

db_conn = None

def initialize(params):
    global db_conn

    db_url = params.get('db_url', None)
    if db_url is None:
        return
    if db_url[0:13] != 'postgresql://':
        db_url = 'postgresql://' + db_url

    try:
        db_conn = db.connect(db_url)
    except Exception as e:
        logging.error(e)

def finalize():
    if db_conn is not None:
        db_conn.close()

def process_command(doc):
    endpoint_id = doc.get('endpoint_id', '')
    endpoint_name = doc.get('endpoint_name', '')
    value_type = doc.get('type', 'numeric')
    if not (endpoint_id == '' or endpoint_id.isdigit()):
        return { "status": "error", "message": 'bad Endpoint ID' }
    if endpoint_name == '' or not endpoint_name.replace('_', '').isalnum():
        return { "status": "error", "message": 'bad Endpoint Name' }
    if value_type not in ['numeric', 'json']:
        return { "status": "error", "message": 'bad Endpoint type' }

    cursor = db_conn.cursor()
```

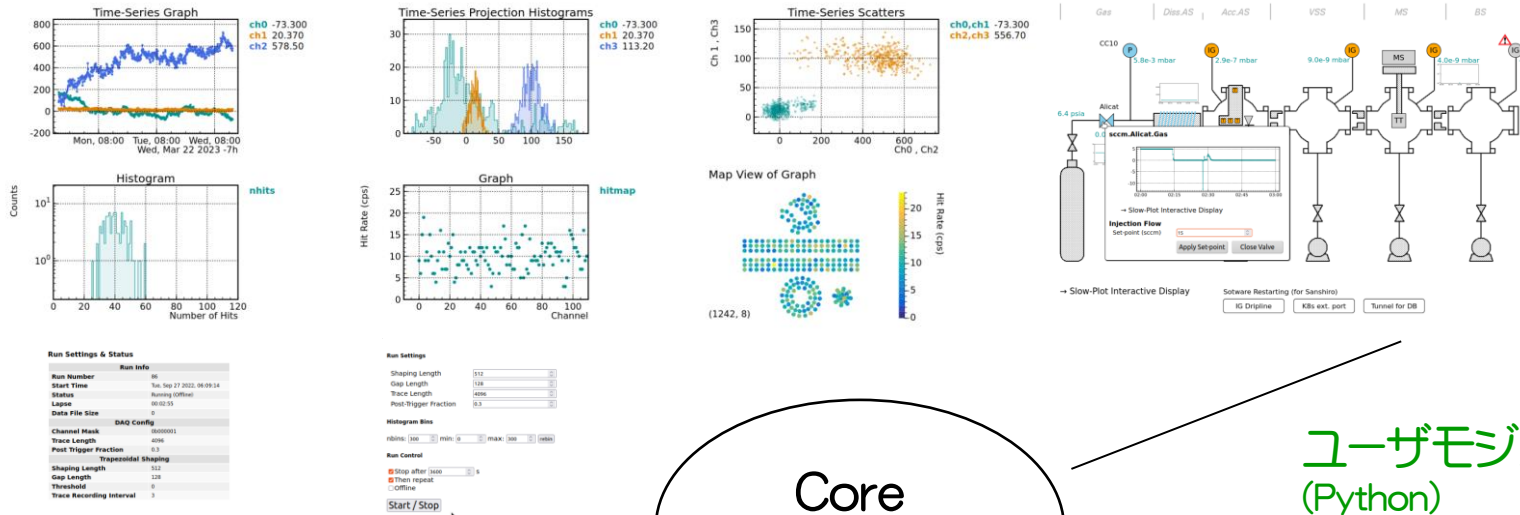
この関数があると最初に呼ばれる

これは最後に呼ばれる

ブラウザでボタンがクリックされたりすると呼ばれる。パラメータが JSON で渡される

SlowDash: ほぼ全てがプラグイン

UI 部品 (JavaScript)



Core Logic

ユーザモジュール
(Python)

デバイス
コントロール

外部システム
連携

実時間
データ解析

異常検知
と対処

Mini-DAQ

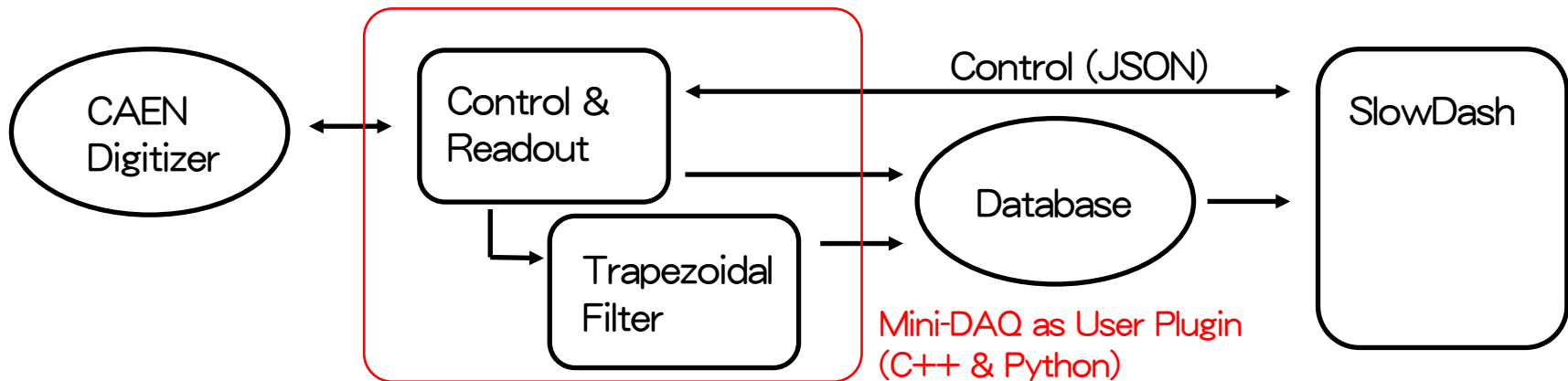
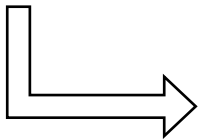
データベースインターフェース



データベースプラグイン以外では外部ライブラリは使用していない (外部依存なし → 設計寿命25年)

Mini-DAQ 例: CAEN のデジタイザと波形解析

読み出しと波形解析をユーザモジュールとして実装してみた



Mini-DAQ 例: オシロによる継続的ノイズモニタ

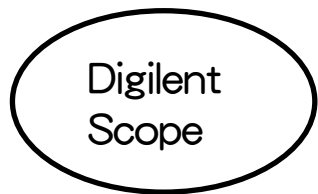
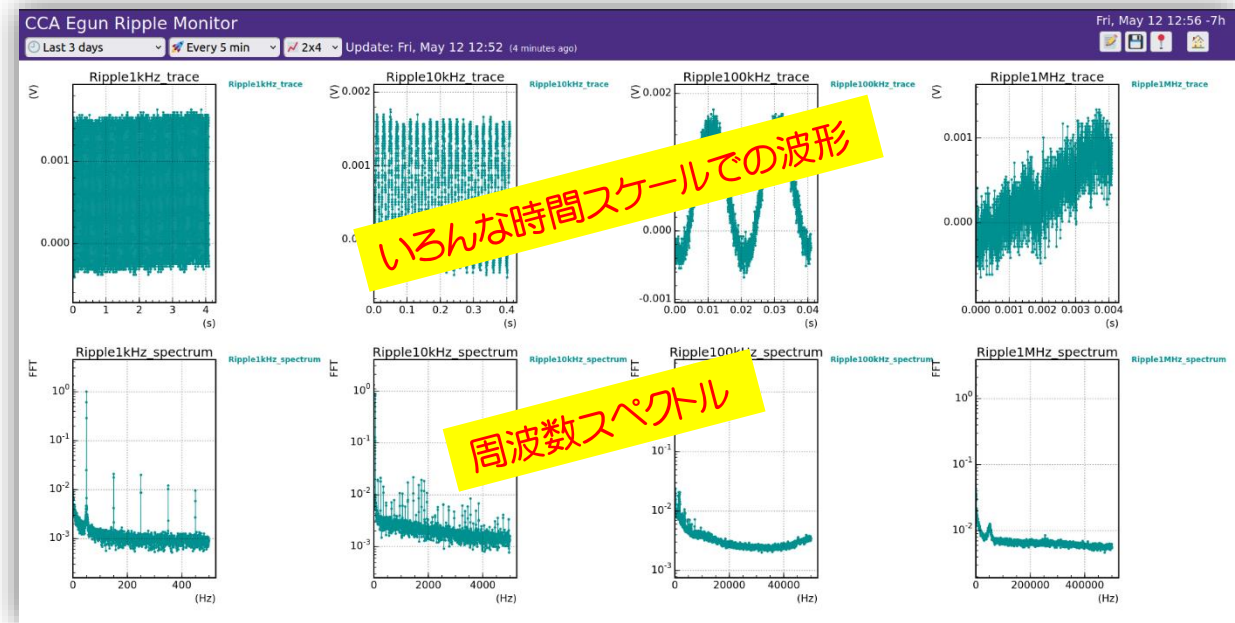
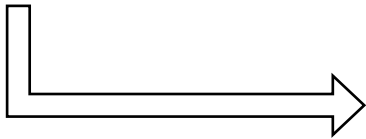
1300ドルの小型オシロスコープが便利に使える

(小型モデルはアカデミックで300ドル)



Analog Discovery Pro 3000 Series:
Portable High Resolution Mixed Signal
Oscilloscopes

\$1,295.00



Control &
Readout

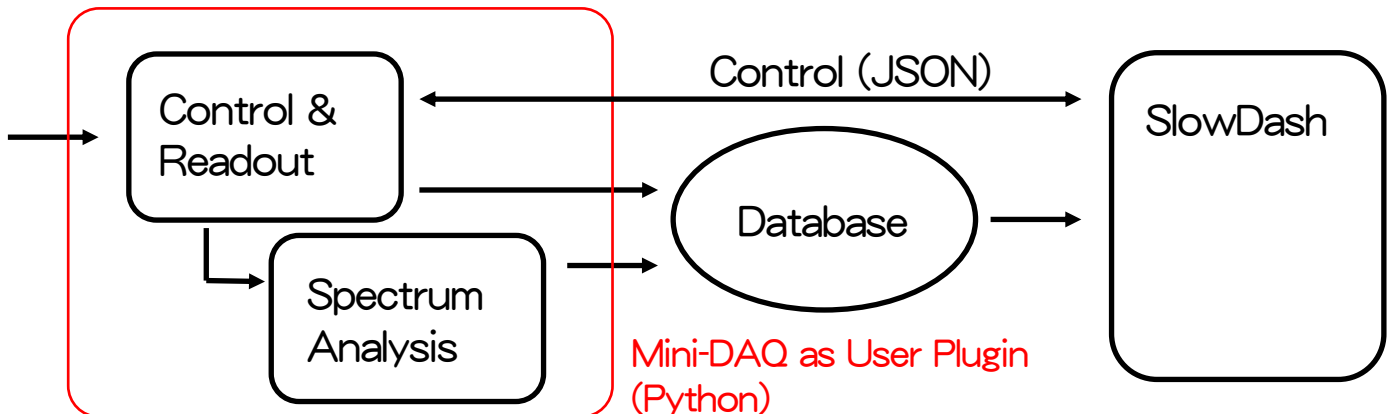
Spectrum
Analysis

Control (JSON)

Database

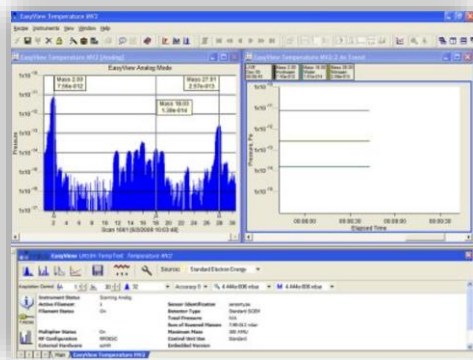
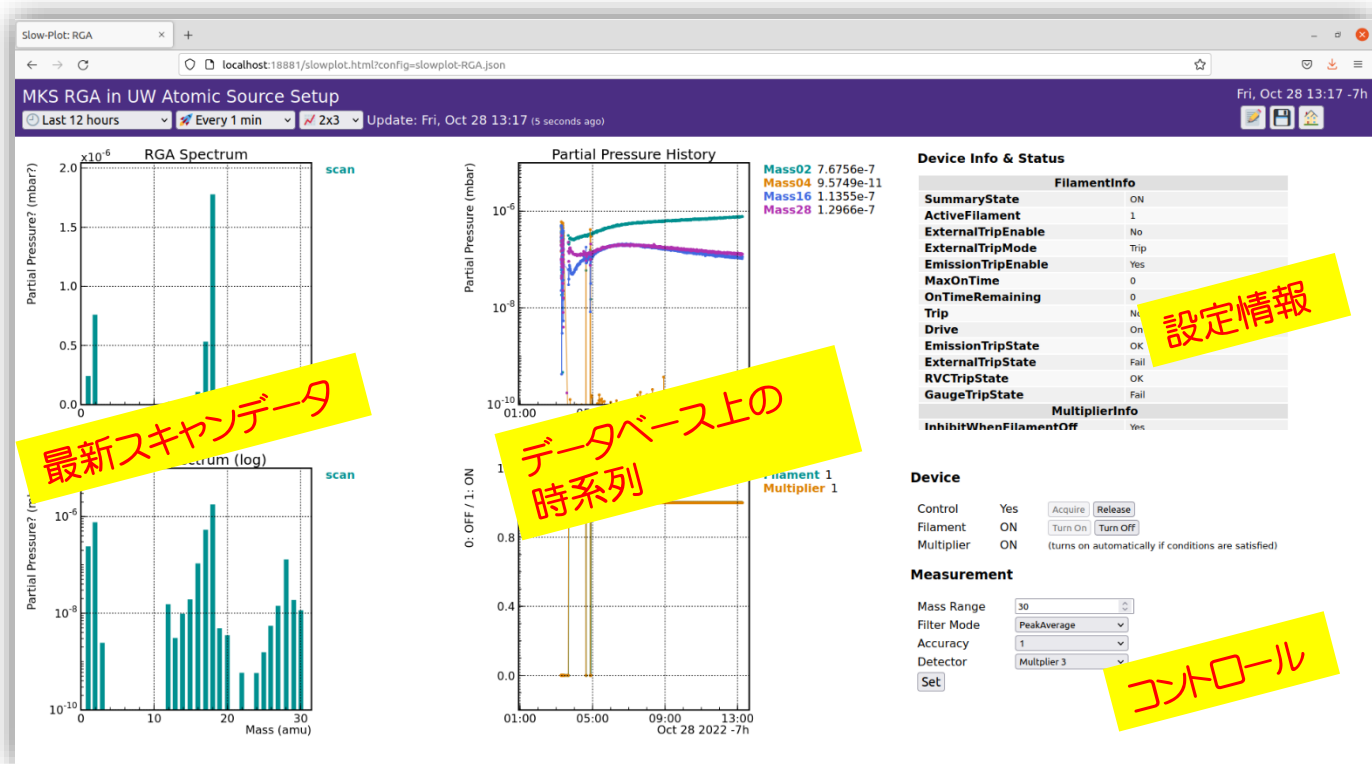
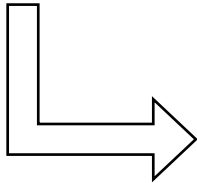
SlowDash

Mini-DAQ as User Plugin
(Python)



Mini-DAQ 例: 残留ガス分析器

付属の Windows のソフトウェアが使われていて、使い勝手が悪かった



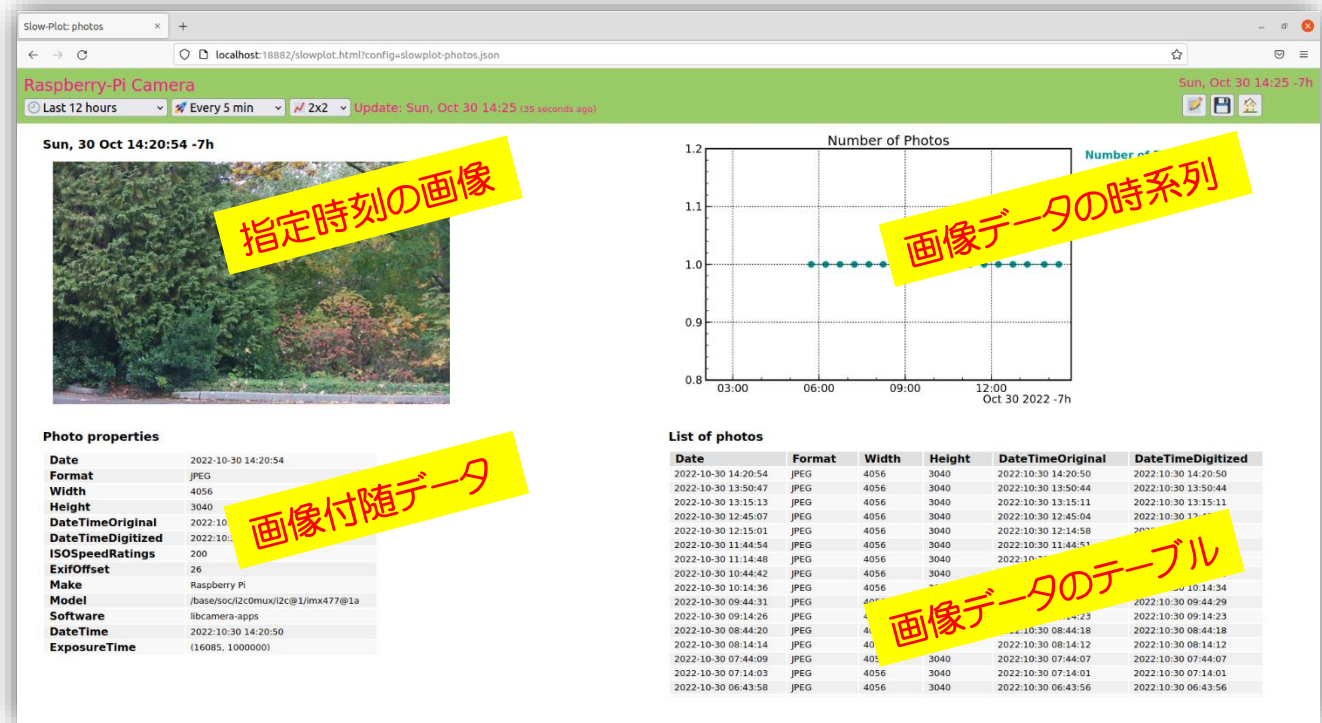
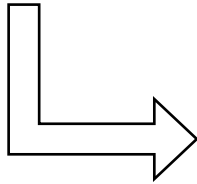
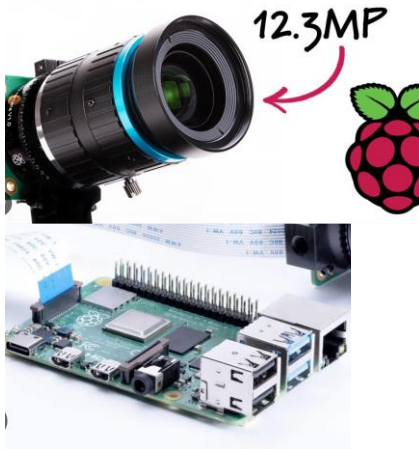
スペクトル分析をして時系列データとしてデータベースに書き込める

⇐ 製品付属 Windows ソフトウェア: 悪くはないけれど、システム統合が難しい

TCP/IP コマンドが公開されているけれどそうすると UI が使えなくなる
Java アプレットも内蔵されているが話にならない

Mini-DAQ 例: Raspberry-Pi カメラと画像解析

画像データ(の時系列)も扱える



- 装置監視写真の時系列, 歪監視, ...
- ML 画像解析
- 高温温度計(パイロメータ)
- ...
- 画像データは Couch DB に保存
- もちろんクエリもできる

構想: Mini-DAQ のパッケージ化と配布

ユーザモジュールと SlowDash 設定ファイルをパッケージにして簡単に組み込めるようにする



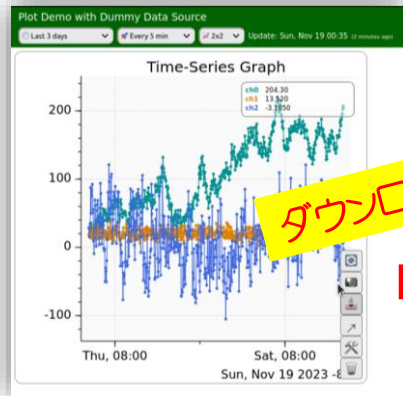
データ解析モジュール(試行錯誤中)

夢: ユーザが Jupyter-Notebook で適当に書いたものを(ほぼ)そのまま走らせたい

データ解析モジュール(試行錯誤中)

夢: ユーザが Jupyter-Notebook で適当に書いたものを(ほぼ)そのまま走らせたい

Step 1: データに Python から簡単にアクセスできるようにする



ダウンロードボタン

同じプロットを作る
Python スクリプト

SlowPy Plotting Script

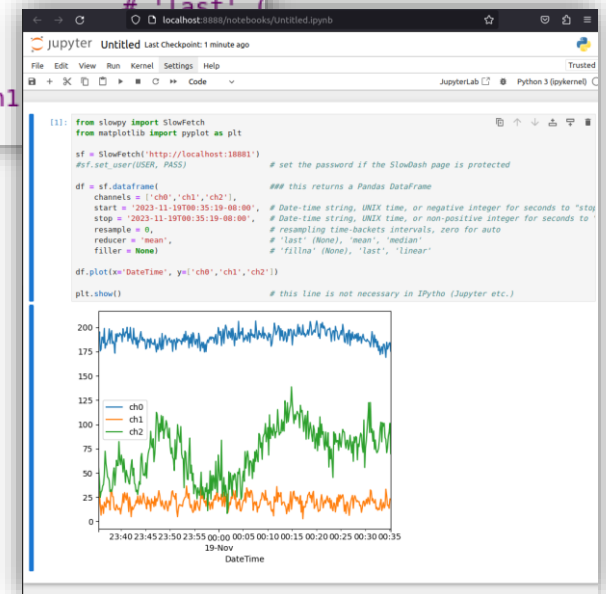
```
from slowpy import SlowFetch
from matplotlib import pyplot as plt

sf = SlowFetch('http://localhost:18881')
#sf.set_user(USER, PASS) # set the

df = sf.dataframe(      ### this r
    channels = ['ch0', 'ch1', 'ch2'],
    start = '2023-11-19T00:35:19-08:00', # Date-time
    stop = '2023-11-19T00:35:19-08:00', # Date-time
    resample = 0, # resampling
    reducer = 'mean', # resampling
    filler = None) # 'last', 'linear'

df.plot(x='DateTime', y=['ch0', 'ch1', 'ch2'])
```

Pandas の
DataFrame
を返す

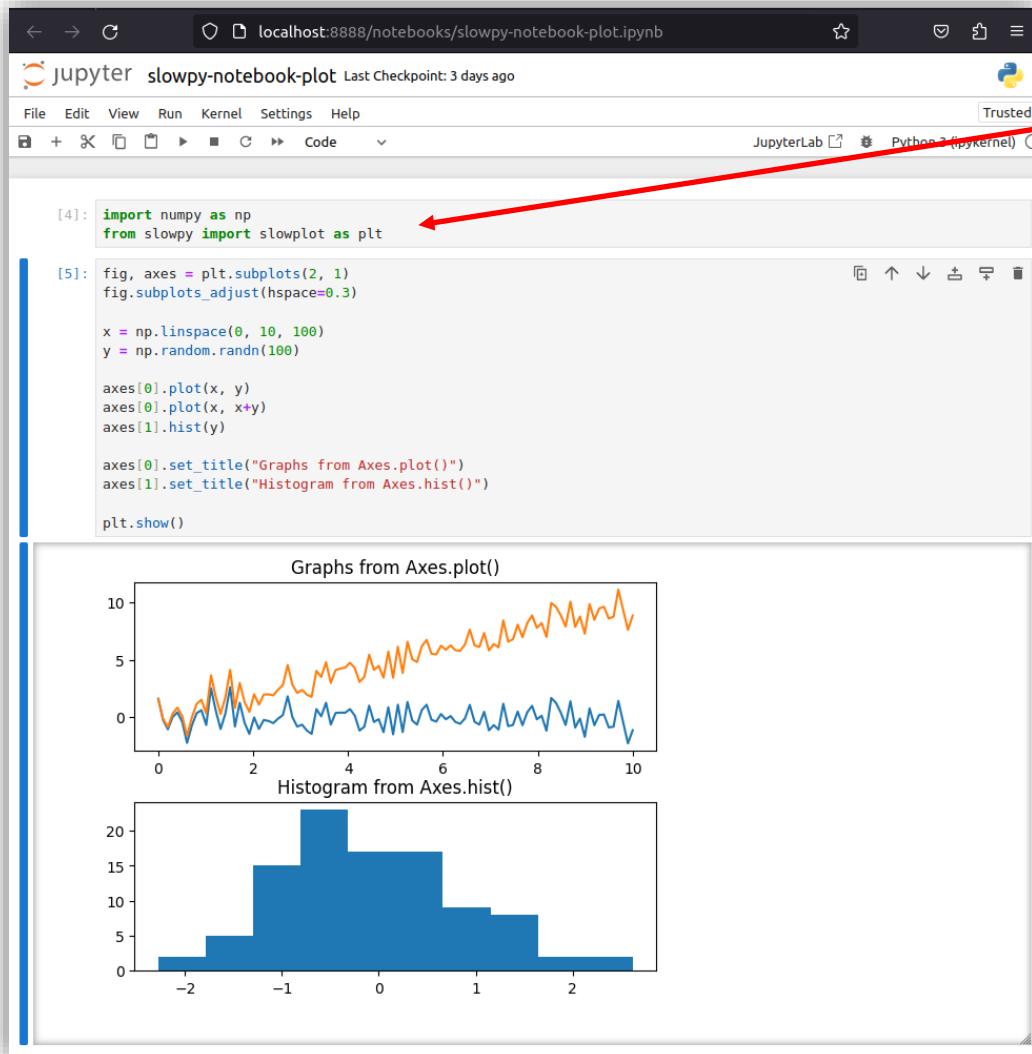


Jupyter にコピー

データ解析モジュール(試行錯誤中)

夢: ユーザが Jupyter-Notebook で適当に書いたものを(ほぼ)そのまま走らせたい

Step 2: ユーザに好きなように解析をしてもらう。 NumPy とか SciPy とか Matplotlib とか scikit-learn でも Stan でも



ただし、この行だけちょっと違う

from matplotlib import pyplot as plt



from slowpy import slowplot as plt

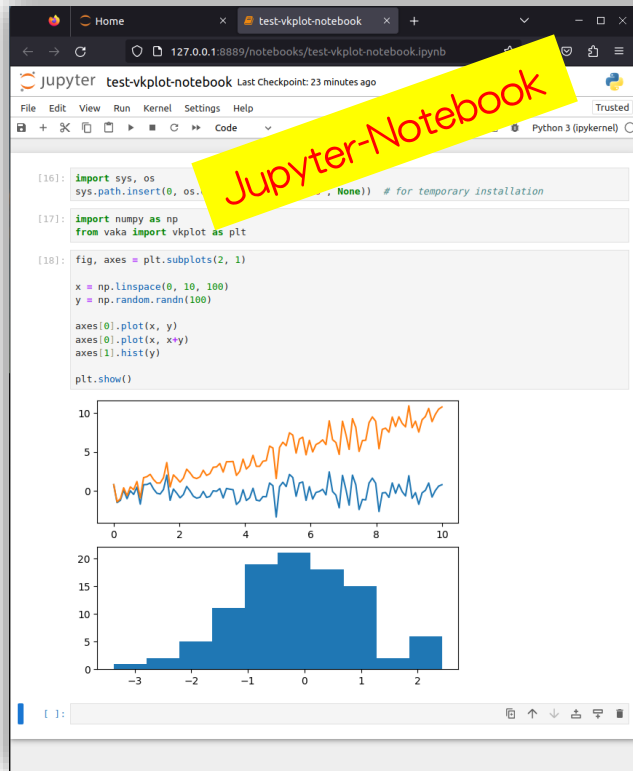
そうすると SlowPy が操作を横取り

- ① Matplotlib のプロットと同時にデータをデータベースにも書き込む
- ② Matplotlib と同じレイアウトの SlowDash 設定ファイルを生成

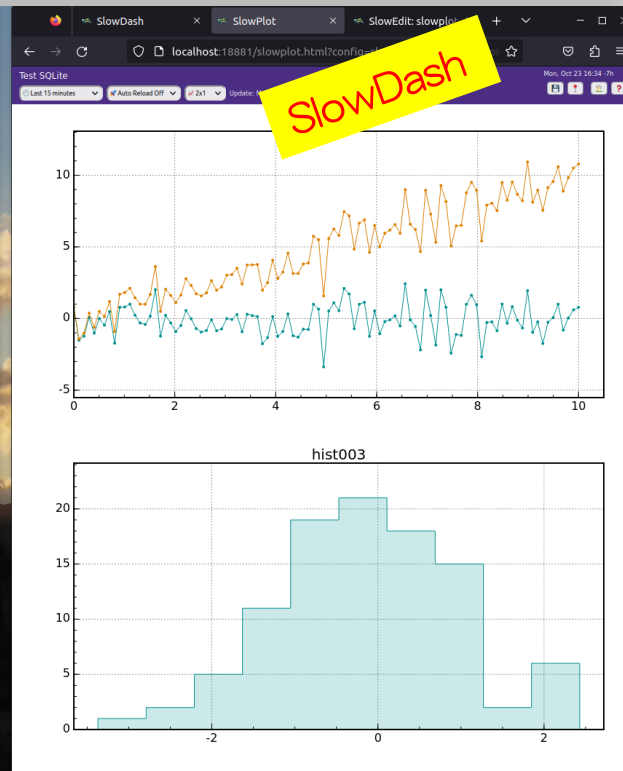
データ解析モジュール(試行錯誤中)

夢: ユーザが Jupyter-Notebook で適当に書いたものを(ほぼ)そのまま走らせたい

Step 3: ユーザ解析プログラムを SlowDash モジュールとして(ほぼ)そのまま登録



こっちを変えると
(ファイル保存が必要)

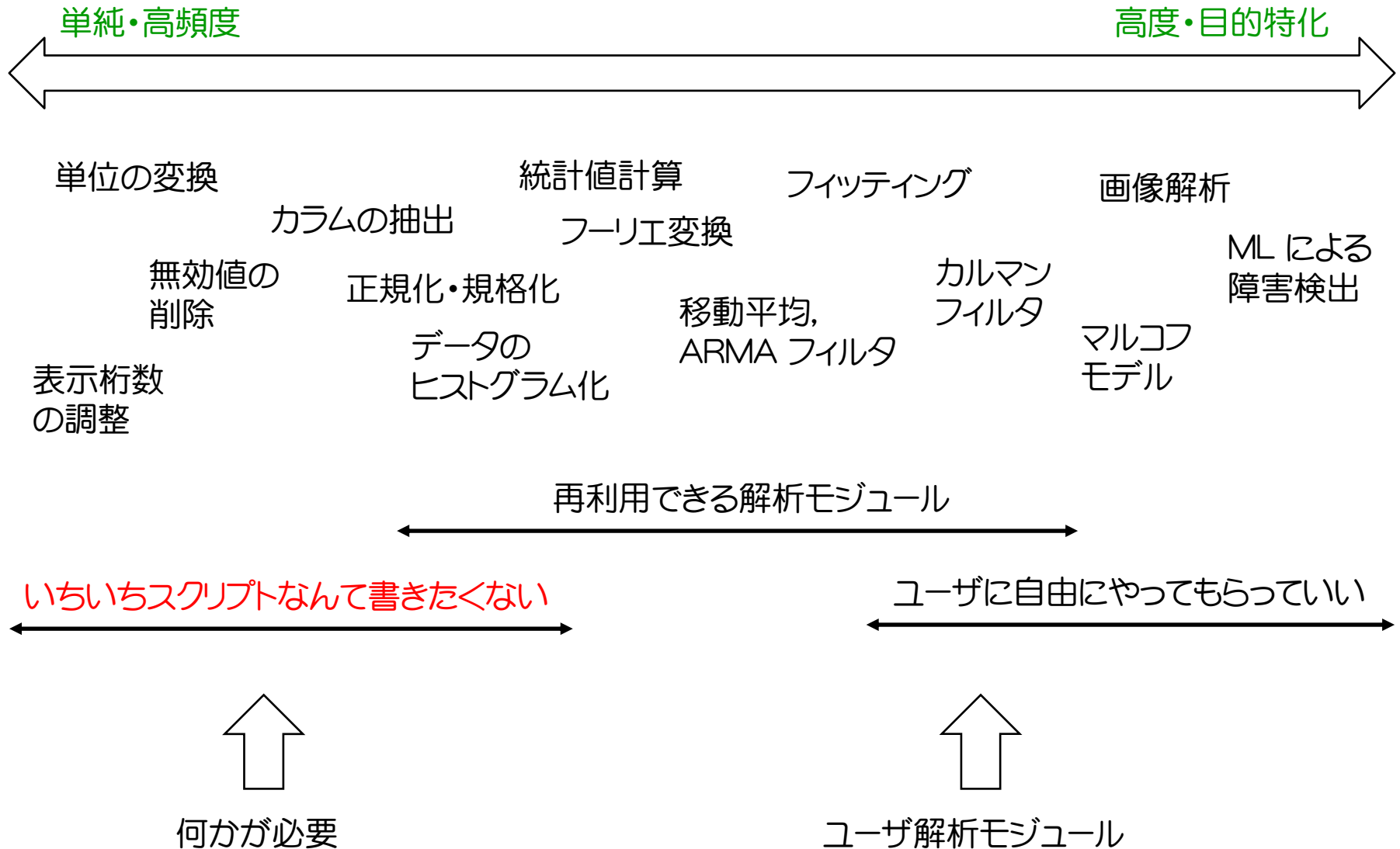


こっちも変わる
(リロードが必要)

現時点では、繰り返し実行のセルを
関数にしてもらう必要がある
(セルを連打する代わりに SlowDash が関数を呼び)

ひとつ問題が：簡単な解析が簡単にできない

解析といってもいろいろある。データストア上のデータはそのまま使えないことが多い



難しい簡単な解析の例

Slow-Plot: RGA

localhost:18881/slowplot.html?config=slowplot-RGA.json

MKS RGA in UW Atomic Source Setup

Last 15 minutes Every 5 s 2x2 Update: Wed, Oct 26 17:26 (4 seconds ago)

Wed, Oct 26 17:26 -7h

RGA Spectrum

Partial Pressure? (mbar?) $\times 10^{-7}$

Mass (amu)

RGA Spectrum (log)

Partial Pressure? (mbar?)

Mass (amu)

Device Info & Status

FilamentInfo	
SummaryState	ON
ActiveFilament	1
ExternalTripEnable	No
ExternalTripMode	Trip
EmissionTripEnable	Yes
MaxOnTime	0
OnTimeRemaining	
Trip	None
Drive	On
EmissionTripState	OK
ExternalTripState	Fail
RVCTripState	OK

Device

Control	Yes	Acquire	Release
Filament	ON	Turn On	Turn Off
Multiplier	ON	(turns on automatically if conditions are satisfied)	

Measurement

Mass Range	30
Filter Mode	PeakAverage
Accuracy	1
Detector	Multiplier 3
Set	

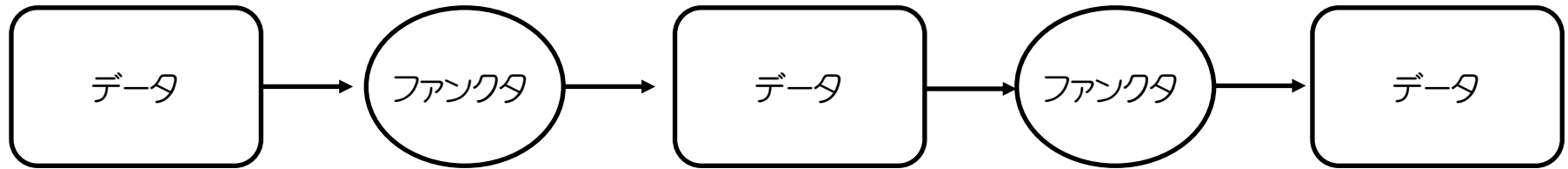
データストア上のツリーデータ (JSON)

ユーザー作成のHTMLフォーム

- データ値を抽出してHTMLに表示したい
- データ値を抽出して入力フィールドの初期値に使いたい
- データ値によってボタンの Enable / Disable をコントロールしたい

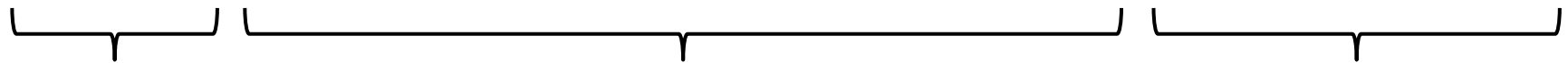
データ解析ファンクタ (試行錯誤中)

汎用ファンクタをチェーンにして必要な機能を実装する



```

    = "status['FilamentInfo/SummaryState']->match('OFF')
  
```



データストア上の
JSON オブジェクト
(Tree 型データ)

->get() ファンクタの別名

正規表現ファンクタ

Tree 型 ⇒ Scalar<string> 型

Scalar<string> 型 ⇒ Scalar<bool> 型

- 抽出ファンクタでいろいろなデータ構造を配列やスカラに変換
- 主に配列データにアルゴリズムを実装する (統計値, フーリエ変換, 時系列フィルタ, 等)
- スカラファンクタはデータの整形にも使える (->format('%.3f MB/s') とか)

今後追加予定

- 操作対象選別フィルタ
- 複数データ入力

レンドリング時データ加工 (Vue.js 風)

Device Info & Status

FilamentInfo	
SummaryState	ON
ActiveFilament	1
ExternalTripEnable	No
ExternalTripMode	Trip
EmissionTripEnable	Yes
MaxOnTime	0
OnTimeRemaining	0
Trip	None
Drive	On
EmissionTripState	OK
ExternalTripState	Fail
RVCTripState	OK

Device

Control Yes

Filament ON ^① ^②

Multiplier ON ^③ (turns on automatically if conditions are satisfied)

Measurement

Mass Range

Filter Mode

Accuracy

Detector

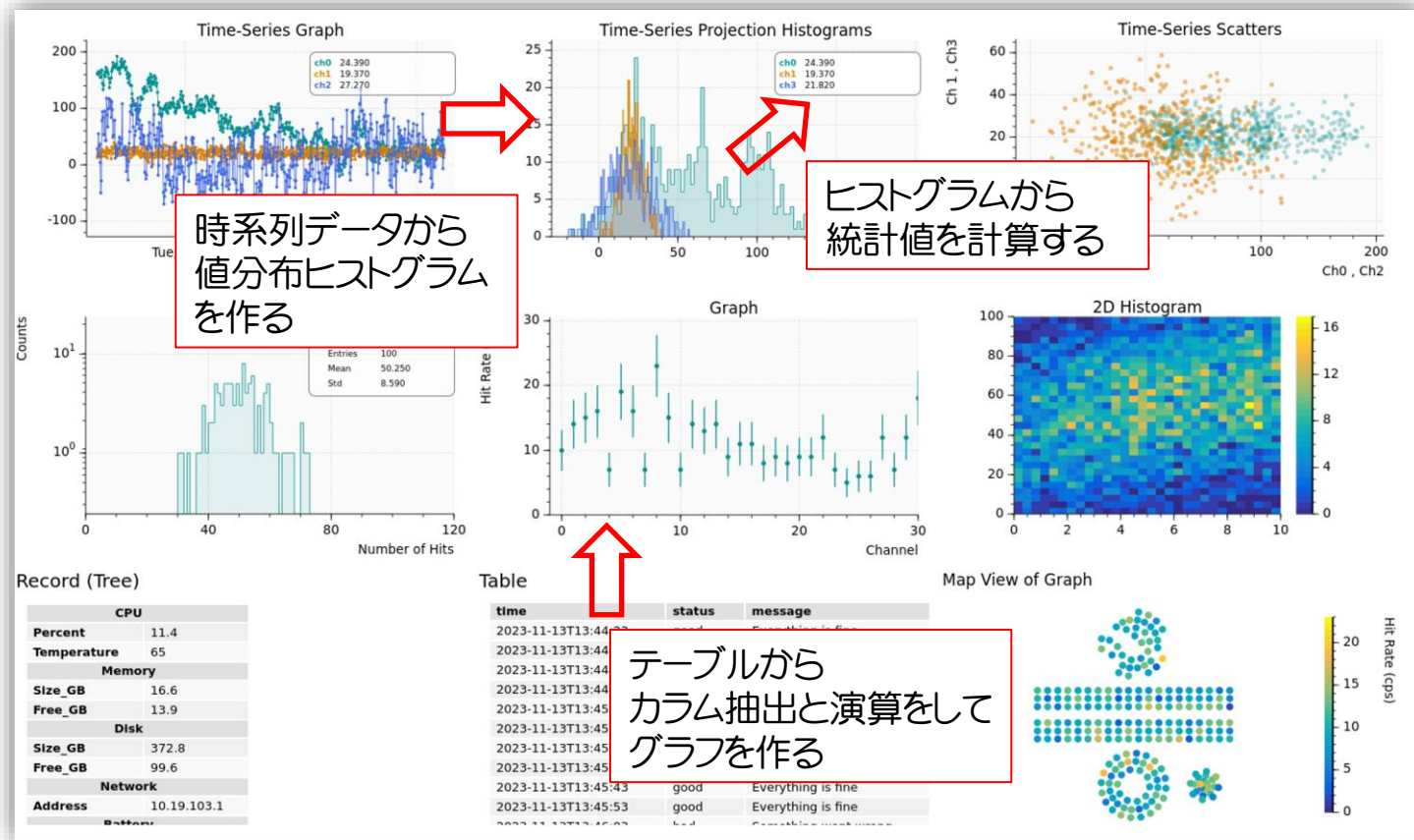
```

<h3>Device</h3>
<table>
  <tr>
    <td>Control</td>
    <td><span sd-value="status['Control/Controlling']">--</span></td>
    <td>
      <input type="submit" name="AcquireControl" value="Acquire" sd-enabled="status['Control/Controlling']->match('No')">
      <input type="submit" name="ReleaseControl" value="Release" sd-enabled="status['Control/Controlling']->match('Yes')">
    </td>
  </tr>
  <tr>
    <td>Filament</td>
    <td><span sd-value="status['FilamentInfo/SummaryState']">--</span></td> ①
    <td>
      <input type="submit" name="FilamentOn" value="Turn On" sd-enabled="status['FilamentInfo/SummaryState']->match('OFF')">
      <input type="submit" name="FilamentOff" value="Turn Off" sd-enabled="status['FilamentInfo/SummaryState']->match('ON')">
    </td>
  </tr>
  <tr>
    <td>Multiplier</td>
    <td><span sd-value="status['MultiplierInfo/MultiplierOn']">replace('Yes','ON')->replace('No','OFF')>--</span></td> ②
    <td><span style="font-size:small">(turns on automatically if conditions are satisfied)</span></td> ③
  </tr>
</table>

<h3>Measurement</h3>
<table>

```

データ解析ファンクタの使い道(構想)



他にも

- 無効値除去・変換
- ビットフラグ値の文字列変換
- 時系列フィルタ(スムージングとか), フーリエスペクトル
- 異常値検出時のアラームアイコン表示
- ヒストグラムフィッティング?

スタック記憶域を持ったデータ解析ファンクタ（構想）

単純な例（セミコロンがスタックプッシュ演算）

二つの時系列データ Ch1 と Ch2 の差の時系列を作る

```
Ch1; Ch2; -> diff()
```

Ch1 と Ch2 をリサンプリングしてデータ点を揃えたうえで散布図を作る

```
Ch1 -> resample(bucket=10); Ch2 -> resample(bucket=10); -> graph()
```

要検討な例

ヒストグラムを作り、ピークをフィットして、相対ピーク幅 (rms/mean) を求める (スタックマシン実装)

```
Ch1 -> hist() -> peakfit() -> get("stats") -> dup() -> get("rms"); -> excl() -> get("mean"); -> ratio()
```

↑

JSON オブジェクトを返す

↑

スタックの先頭データを複製

↑

スタックの上2つを入れ替える

別案:

スタックマシンに外部記憶域 (レジスタファイル) を追加し、出力を新しいデータチャネルのように見せる

```
Ch1 -> hist() -> peakfit() -> get("stats") -> Peak; Peak -> get("rms"); Peak -> get("mean"); -> ratio()
```

↑

スタック外の名前付き記憶域 (動的に作成された仮想データチャネル)

開発状況

先週: Version 231116, “Nisqually”

- Grafana 風のデータビジュアライゼーションはほぼ完成
- 単純なコントロールインターフェース (将来変更の可能性あり)
- 試験的なデータ解析インターフェース (毎日変化中)

<https://hachi.npl.washington.edu/~sanshiro/SlowdashDownload-JP/> (仮置き場)でダウンロード可能,
(日本語ドキュメントあり)

SPADI WG3 に居候中.

半年~1年くらい: Version “Snoqualmie”

- 名前を決めて GitHub に置く
- コントロールと解析を完成させたい
- アラーム通知と対応履歴管理
- コンテナ化

その次: Version “Nooksack”

- メッセージングシステム (AMQP / ZMQ / Kafka etc) への直接接続
- モジュールと設定ファイルのパッケージング
- カッコいいレイアウト