

ラズパイとDashで環境ダッシュボードを作ろう

PyCon JP 2021

2021/10/15 ~ 10/16

お前誰よ

- 佐野浩士 (Hiroshi Sano) [@hrs_sano645](#)
 - : 静岡県の富士市
- Job
 - 株式会社佐野設計事務所
 - 自動車向けプレス金型の機械設計事務所
 - 3Dデータのモデリングもやってます！
 - ITインフラや業務で使うアプリを作る担当です
 - 米農家
 - Community
 - Python駿河、PyCon mini Shizuokaスタッフ
 - Code for ふじのくに

目次

- 環境ダッシュボードを作った話
- PythonとIoT
- Dashでダッシュボードアプリを作る
- まとめ

この部分でツッコミや補足等を入れていきます。

このトークの趣旨、モチベーション

趣味プロジェクトを紹介しつつ、Pythonを使ったIoTとデータ可視化をデモを交えてお伝えします

- IoT: ラズパイ+CircuitPythonを合わせる選択肢
- データ可視化: Dashでダッシュボードアプリを作る

本日お伝えしたいこと

- 自分の欲しい物が無ければ作ろう
- 身の回りで見えない数字を可視化してみよう
 - 身近だけど見えないデータ

今日のテーマ

ラズパイとDashで環境ダッシュボードを作ろう

久しぶりに、長めのトークをやらせてもらっています。

- ラズパイ = Raspberry Pi
- Dash = Plotly Dash
(Webアプリフレームワークのこと)
- 環境ダッシュボード => 環境センサーの情報が見れるダッシュボードアプリ

目次

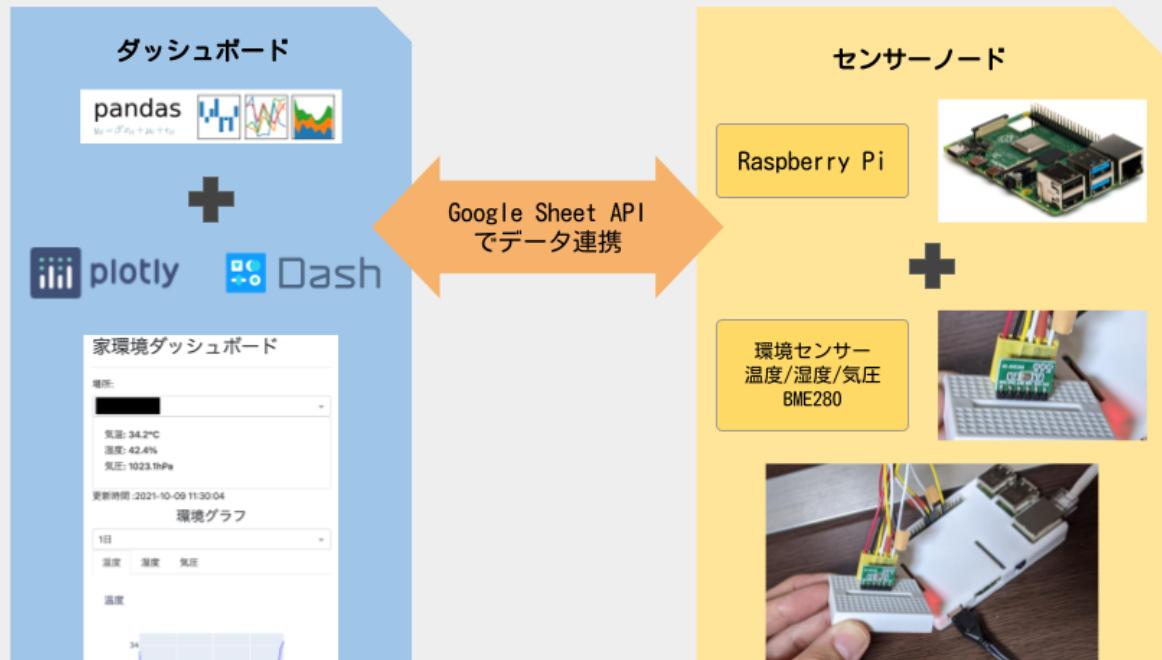
- 環境ダッシュボードを作った話
- PythonとIoT
- Dashでダッシュボードアプリを作る
- まとめ



homeenvdashプロジェクトの紹介

趣味プロジェクトです

- センサーの情報をダッシュボードで見るアプリ/システム
- 計測した時の最新の数値が見える
- グラフで1日分/1週間分の推移が見える



homeenvdashの全体構成

- センサーノード
 - ラズパイ + センサー
 - センサーノードは複数対応
- ダッシュボードはDash + Plotly
- センサーで取得した情報は Googleスプレッドシートで保存
 - 今後はローカルなDBを検討

現在は、温度湿度気圧のBME280に対応。今後はCO2や非接触温度センサなどに対応したい



なぜ作ったのか？

そもそもセンサーやダッシュボードは
すでに製品、サービスが多数ある



- 環境センサーは市販にも販売されている
 - 単体でクラウド対応とか
 - 高価な製品は精度がいい
- ダッシュボードサービスも多数ある
 - OSSなサーバーアプリ: Grafana, Zabbix, Home Assistant...
 - サービスとして提供: [Anbient](#), [Machinist](#), [MotionBoard](#), [Google データポータル](#)...

車輪の再開発になるけどどうなのか？

作った理由は、自分の欲しい物がなかったため



気圧の状況を見たい

- 低気圧に弱いので不調の前兆を調べたい
 - 予報サービスはあるけど現状を見たい
- 世の中にはIoT/センサー製品はあるものの
気圧対応がされてるものは**手を出しずらかった**
 - 選択肢が少ない
 - 独自のサービス
 - 高価な製品

教えていただきましたが、オムロンの環境センサーは高価ですが扱いやすそうです:
<https://www.omron.co.jp/ecb/product-detail?partNumber=2JCIE-BL>



見守りに使いたい

- 実家の祖母の部屋の状況確認に利用
- 高齢者は気温の変化を感じづらい
 - 体感より数値化されていた方が対応しやすい
- センシティブな環境なので扱ったことがない外部サービスだと不安
 - できればローカルのみで扱いたい

欲しいけどない -> 自分で作ろう！

欲しいものを自分で作る



- Maker（メイカー）という文化
メイカーとは「テクノロジー」という言葉を、できる限り開放的に解釈、自分で学び利用出来る技能全般のことを理解して、冒険と実験への招待状だと考えている人のことだ
オンラインリージャパン「私たちはみなメイカーだ」より引用
- 世の中に存在していなければ自分で作る精神！

趣味ってのもありますし、車輪の再開発も楽しいよ！

先人の例（教え）



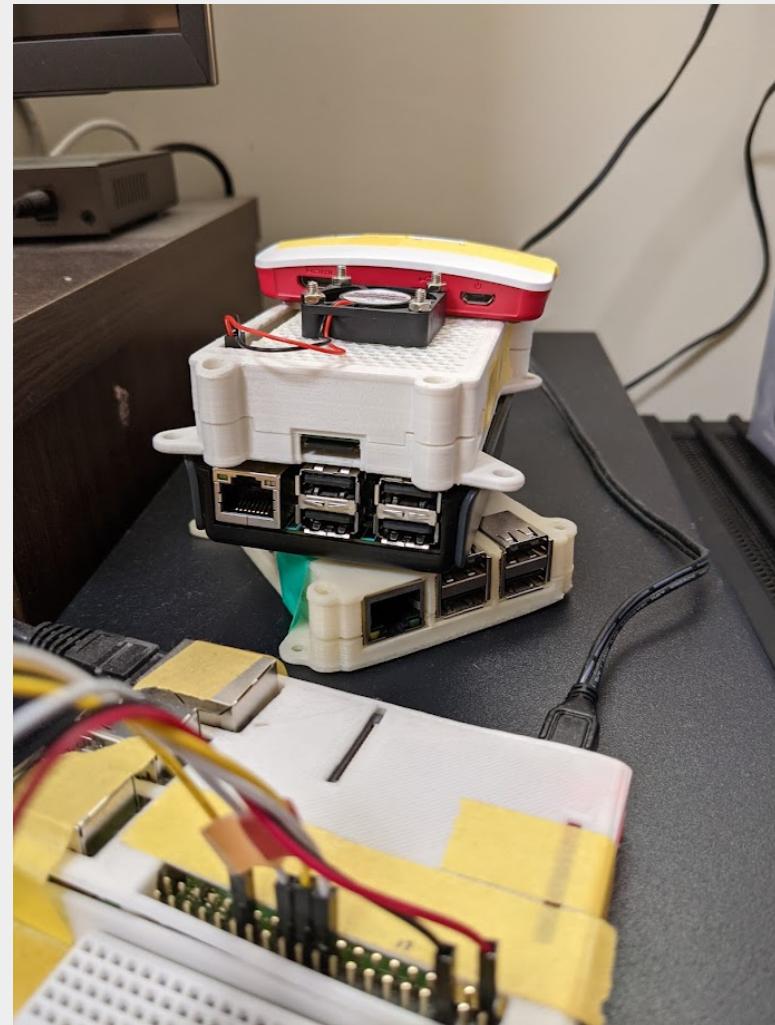
小池さん: きゅうり仕分け機



からあげさん: AIルンバ

- 小池さん: <http://workpiles.com/2016/08/ccb9-prototype2-complete/>
- からあげさん: <https://karaage.hatenadiary.jp/entry/2019/11/06/073000>

個人的に解決したい問題もある



ガジェット好きなもので...
ラズパイが大量に転がっている

みなさんも引き出しにありますよね？ 積みボード

しかし先人はこう言います



A screenshot of a Twitter post from user @karaage0703. The profile picture is a white cat wearing a small orange hat. The tweet text is "マイコンはとりあえず全部買って積みましょう". The timestamp is "午後1:00 · 2021年7月9日 · Twitter for iPhone". At the bottom, there are engagement metrics: "20 件のリツイート" (Retweets), "2 件の引用ツイート" (Quoted tweets), and "107 件のいいね" (Likes).

<https://twitter.com/karaage0703/status/1413347181705105410?s=20>

積んでからが勝負のこと

やっぱり闇のエンジニアはちげーわ！

積みボードがある方は

贅沢に使って快適な日常を手に入れましょう！

もちろん、新しく購入して初めてトライしてもいいと思います！

トークで使うデモアプリ

今日は `homeenvdash-mini` というデモ用のアプリを使い扱う技術について
デモを交えて解説します。

<https://github.com/hrsano645/homeenvdash-mini>

homeenvdashとの違い

- ダッシュボードアプリとセンサー値取得を同時に実行します
- センサーの値を取得して現在情報とグラフ表示はほぼ同じ
- センサーの値はCSVファイルへ保存される
- (デモの都合上) ダッシュボード起動時にしかセンサー値の記録はされません

目次

- 環境ダッシュボードを作った話
- PythonとIoT
- Dashでダッシュボードアプリを作る
- まとめ

PythonとIoT

より手軽にPythonでIoTを扱う方法を紹介します

- ※IoTは広義だといろんな意味がありますが
ここではセンシングや電子機器操作をネットワーク経由で行うことを目指します

PythonでIoTを扱う選択肢

1. Raspberry Pi + CPython
2. MicroPython / CircuitPython
3. Raspberry Pi + CircuitPython

今日のデモでは 3番目を利用します

PythonでIoTを扱う選択肢

1. Raspberry Pi + CPython
2. MicroPython / CircuitPython
3. Raspberry Pi + CircuitPython

上記は私が知っている限りで一例です。ほかにも選択したがある場合は教えていただけます

Raspberry Piについて



- もともとは教育目的のLinuxが動作するシングルボードコンピューター(SBC)
 - 工場自動化やサイネージ、センサーノードとして業務利用も
- インターフェイスが豊富
 - GPIO, 無線LAN, Ethernet, USB, Bluetooth, HDMI出力
- GUI/CUIで利用可能
 - デスクトップ端末、ヘッドレスなサーバー

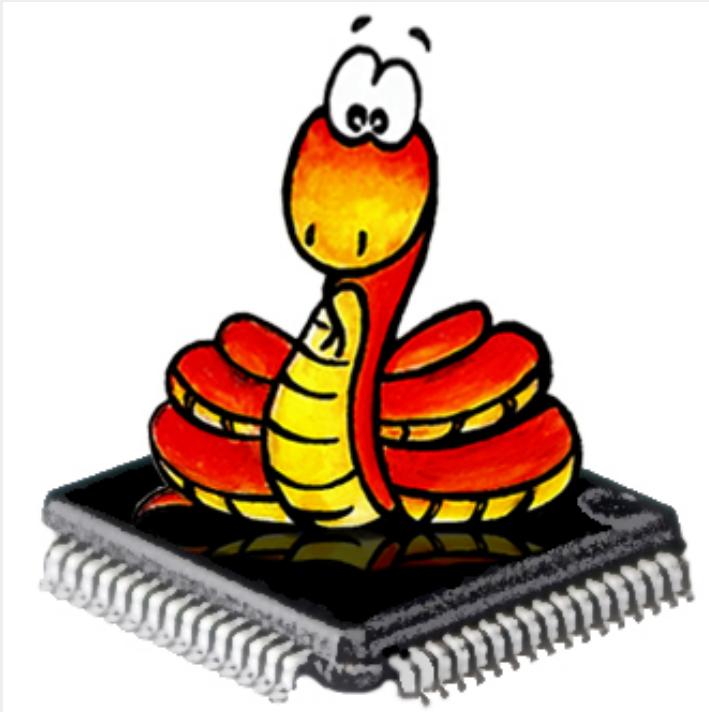
界隈ではK8sを使ってクラスター構成を作ったりもしてるそうです。楽しそう。

Raspberry PiとIoT

- Linuxが動きCPythonを扱える
 - GPIO（デジタル入出力）でセンサーと接続可能
 - シリアル通信規格: SPI, I2C
 - ディスプレイを繋ぐとサイネージ的なデバイスも作れる
- 安価ながら高機能なIoT端末として扱える
- ▲ ACアダプタなど給電環境が必要 電源がない環境では扱いづらい

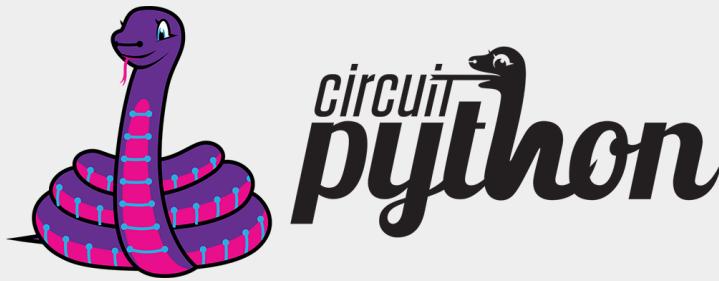
PythonでIoTを扱う選択肢

1. Raspberry Pi + CPython
2. **MicroPython / CircuitPython**
3. Raspberry Pi + CircuitPython



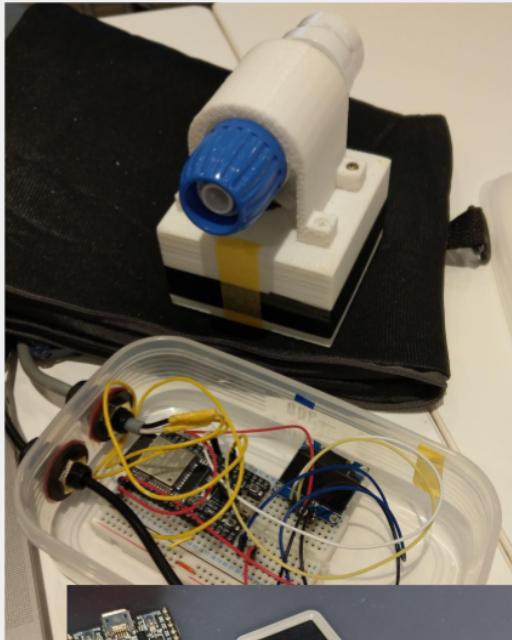
MicroPythonとは

- MicroPythonはマイコンボード向けの処理系
 - マイコン=マイクロコントローラー
 - 特定の機器制御に最適化された集積回路
- クラウドファンディングで生まれたpyboardの開発環境として作られた



CircuitPythonとは

- MicroPythonの派生版
 - adafruitが開発支援: オープンソースハードウェア企業で教育向け製品を扱う
- 当初はadafruitのマイコンボードに対応
 - その後他社製のマイコンボードにも広がる
- 専用ライブラリを使うとセンサーヤやLEDコントローラーが少ないコードで扱える



MicroPython/CircuitPythonの特徴

- CPythonの3系（3.4, 3.5の一部）の文法をベースにした独自の処理系
- マイクロコントローラー向けカスタムした標準ライブラリやサードパーティライブラリ
- シリアルポート（UART）から直接REPLを実行

この処理系は他のマイコンにも移植される

- Raspberry Piより安価。電源はバッテリーも
- ▲ CPythonライクだが完全互換ではない

写真はESP32+MicroPythonで水栓を開け閉めするものを作ってました

PythonでIoTを扱う選択肢

1. Raspberry Pi + CPython
2. MicroPython / CircuitPython
3. Raspberry Pi + CircuitPython

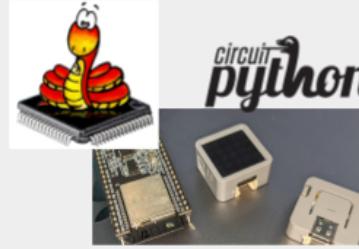
3. CircuitPythonのライブラリをRaspberry Piで扱う

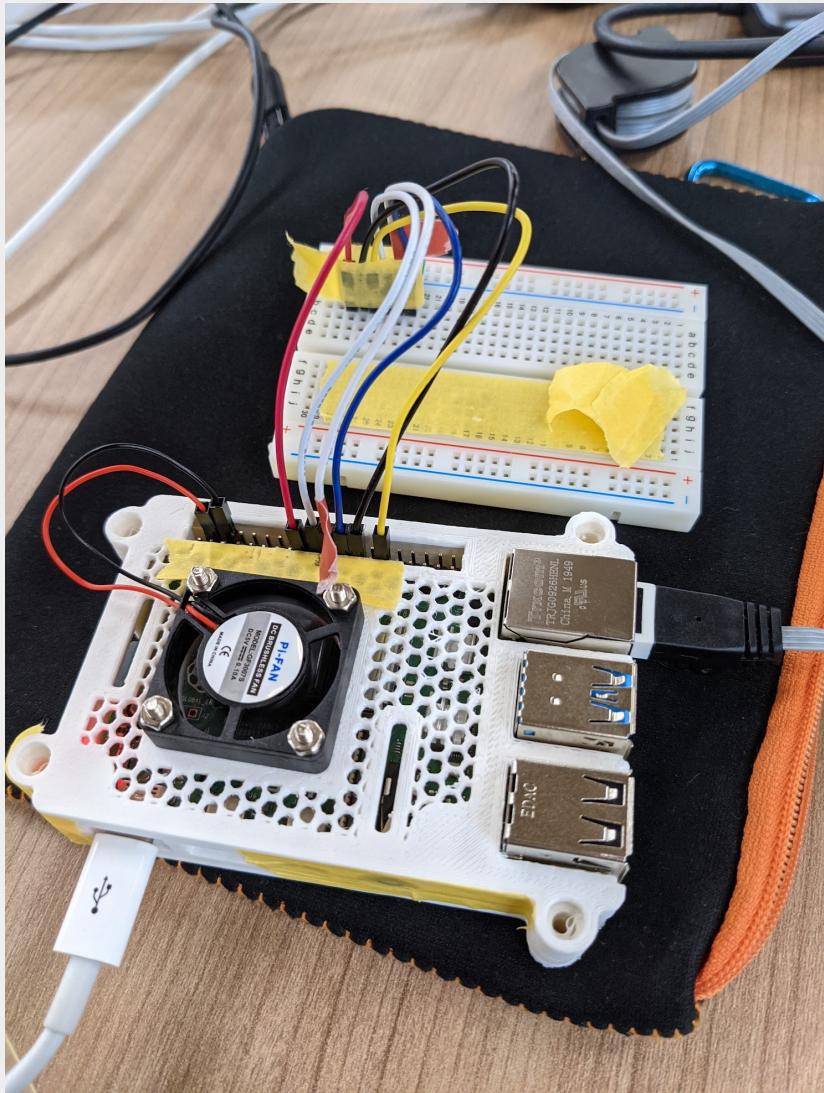
- CircuitPythonの豊富なライブラリをRaspberry Piで扱う選択肢
- Adafruit-Blinkaライブラリを使う
 - Raspberry PiのGPIOをCircuitPythonのハードウェア向けに変換するレイヤー
- Raspberry PiとCircityPython間でコードの相互利用がしやすい

CircuitPython features unified Python core APIs and a growing list of 300+ device libraries and drivers that work with it. These libraries also work on single board computers with regular Python via the Adafruit Blinka Library.

<https://circuitpython.readthedocs.io/en/latest/docs/index.html>

- CircityPythonのライブラリが使えセンサーモジュールの扱いが楽
- ▲ どちらかの環境依存のコードを書く場合は扱いが難しい

	Raspberry Pi	MicroPython CircuitPython	Raspberry Pi + CircuitPython
イメージ			 
Python処理系	CPython	マイコン向け	CPython Blinkaライブラリ
扱いやすさ	通常のPythonとして	専用のライブラリが 豊富	CPython CircuitPython いいところ
電源	ACアダプタ	バッテリー利用も	ACアダプタ



デモ: 温度湿度気圧センサーから情報を取得する

Raspberry Pi + Blinkaライブラリを使って、
デモしながら様子を見せていきます

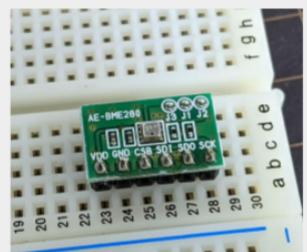
- ラズパイ4BとBME280を接続
- BME280を搭載したモジュール製品
 - 今回は[AE-BME280](#)（秋月電子通商）
 - ラズパイとの通信はSPI接続を利用



ブレットボード



ジャンプワイヤー



センサー:BME280
(AE-BME280)



- はんだごて
- はんだ



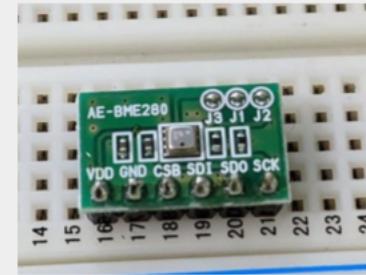
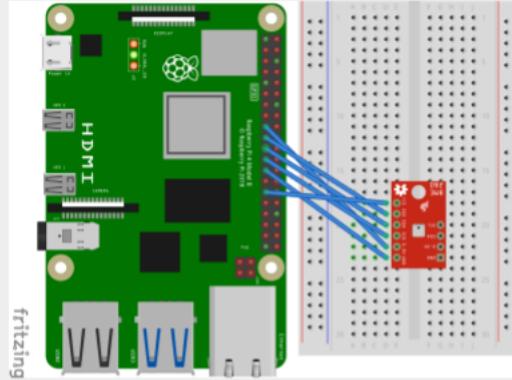
はんだこて台

デモを試す時に必要なもの

- 利用する部品: ブレットボード、ワイヤー、BME280 (利用するセンサー)
- 道具: はんだごて、はんだ、はんだこて台 (センサーにピンが実装されていない場合は必要です)
- ラズパイも買ってください
- 購入先:
Amazonとかでも集まる。
[秋月電子通商](#)、[スイッチサイエンス](#)、
[aitendo](#)、[マルツオンライン](#)、[せんごくネット通販](#) がおすすめ

上記を用意できれば作れます。そのうちに工具も部材も増えていきます。沼ですね

配線の様子を見せます



ラズパイGPIO	ラズパイ ピン番号(例)	SPI	AE-BME280 ピン名
3V3	17	VDD	VDD
GND	25	GND	GND
GPIO 5	29	CS	CSB
GPIO 10 (SPI0 MOSI)	19	MOSI	SDI
GPIO 9 (SPI0 MISO)	21	MISO	SDO
GPIO 11 (SPI0 SCLK)	23	SCLK	SCK

配線方法まとめ

- 配線は一例です。
- 画像で利用しているBME280はAE-BME280ではないので、表を元に配線してください

このピン配置の作業をやるとUSBが偉大に見えますね

動作させたテストコード

https://github.com/hrsano645/homeenvdash-mini/blob/main/test_bme280.py

Tips: Raspberry Pi上で開発をしやすくする

VS Codeのリモート開発が便利です -> Remote-SSH

- ssh経由で扱うといい。ただpi3あたりからでないと、リモート開発できない
(vscodeのリモートサーバーが対応するCPUアーキテクチャの問題)
- <https://www.raspberrypi.org/blog/coding-on-raspberry-pi-remotely-with-visual-studio-code/>

Remote SSH needs a Raspberry Pi 3 or 4. It is not supported on older Raspberry Pis, or on Raspberry Pi Zero.

目次

- 環境ダッシュボードを作った話
- PythonとIoT
- Dashでダッシュボードアプリを作る
- まとめ

Dashでダッシュボードアプリを作る

Dashライブラリを使ってセンサー情報を表示する
ダッシュボードを作ります

Dashの紹介

- Plotlyが作成しているWebアプリフレームワーク
 - Plotlyはグラフライブラリの名称でもある: `Plotly.js`、`Plotly.py`
- データ分析向けのプロトタイピングがしやすい
- サンプル（有料機能を使ったものもあるので注意）<https://dash.gallery/Portal/>

Dashの特徴

- ほぼPythonのみでWebアプリのレイアウトが作れる
- コールバック機能を使ってインタラクティブ操作
- データセットやDBを扱いたい場合は自前で用意
 - PlotlyはPandasのDataFrameを使うのでPandas経由がやりやすいはず

ちなみにDashはFlask + Reactで作られている。

Built on top of Plotly.js, React and Flask, Dash ties modern UI elements like dropdowns, sliders, and graphs directly to your analytical Python code.

<https://github.com/plotly/dash>

ほぼPythonのみでWebアプリのレイアウトが作れる

- HTMLタグを書く必要がない
 - HTMLを扱うコンポーネント
=> Pythonの操作のみでアプリが作れる
 - `dash-core-components` (`dcc`) を使うと高度なフォームや操作インターフェイスが扱える
- Plotlyと連携して豊富なグラフを扱うことができる

※ HTMLを書く必要がない≠HTMLの知識が必要ない

```
from dash import Dash, callback, html, dcc, Input, Output

# dashアプリの初期化
app = Dash(
    __name__,
    meta_tags=[{"name": "viewport", "content": "width=device-width, initial-scale=1"}],
)
app.title = "Hello Dash App"

def _layout():
    """アプリの全体のレイアウト"""
    return html.Div(
        [
            html.H2(app.title),
            html.Label("PythonのみでWEBアプリを作ります")
        ],
    )

if __name__ == "__main__":
    app.layout = _layout
    app.run_server(debug=True, host="0.0.0.0")
```

```
from dash import Dash, callback, html, dcc, Input, Output

# dashアプリの初期化
app = Dash(
    __name__,
    meta_tags=[{"name": "viewport", "content": "width=device-width, initial-scale=1"}],
)
app.title = "Hello Dash App"

def _layout():
    """アプリの全体のレイアウト"""
    return html.Div(
        [
            html.H2(app.title),
            html.Label("PythonのみでWEBアプリを作ります")
        ],
    )

if __name__ == "__main__":
    app.layout = _layout
    app.run_server(debug=True, host="0.0.0.0")
```



コールバック機能:動的な操作

- 動的な操作を可能にする
- たとえば
 - HTMLのフォーム操作
 - グラフの描写結果を更新する（日付別とか）
 - 定期的な表示の更新を行う（dcc.Interval）
- コールバック関数を用意してデコレーターで設定
 - アプリ上の操作を入力としてトリガーさせる
 - 入力された情報を元にデコレーター対象の関数が処理
 - 処理した結果をアプリ上へ出力する

高度なコールバックとして、パターンマッチングやロングコールバック（v2.0から）があります

```
from dash import Dash, callback, html, dcc, Input, Output

# dashアプリの初期化
app = Dash(
    __name__,
    meta_tags=[{"name": "viewport", "content": "width=device-width, initial-scale=1"}],
)
app.title = "Hello Dash Callback"

def _layout():
    return html.Div(
        [
            html.H1(app.title),
            html.Hr(),
            html.P("文字を入力すると、出力の部分が更新されます"),
            html.Div(
                [
                    html.Span("入力: "),
                    dcc.Input(id="input-form", value="Callbackを試しています", type="text"),
                ]
            ),
            html.P(id="output-p"),
        ]
    )
# 次へ続く
```

```
# 続き
@callback(Output("output-p", "children"), Input("input-form", "value"))
def update_output_text(input_value):
    # 引数がInputのvalueの値を取得
    # return側に更新したいコンポーネントを指定する。childrenは指定コンポーネントの子要素の事
    return "出力: {}".format(input_value)

if __name__ == "__main__":
    app.layout = _layout
    app.run_server(debug=True, host="0.0.0.0")
```

Hello Dash Callback

文字を入力すると、出力の部分が更新されます

入力:

出力: Callbackを試しています



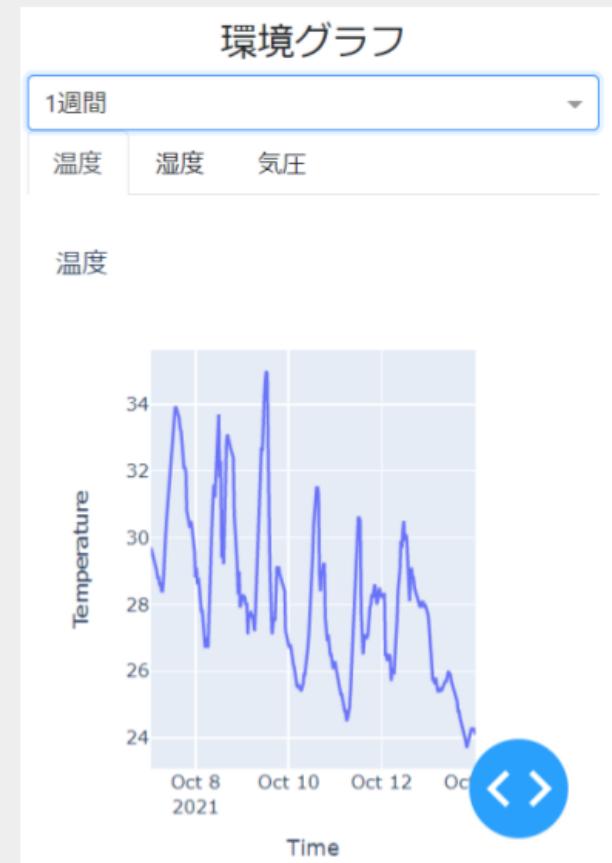
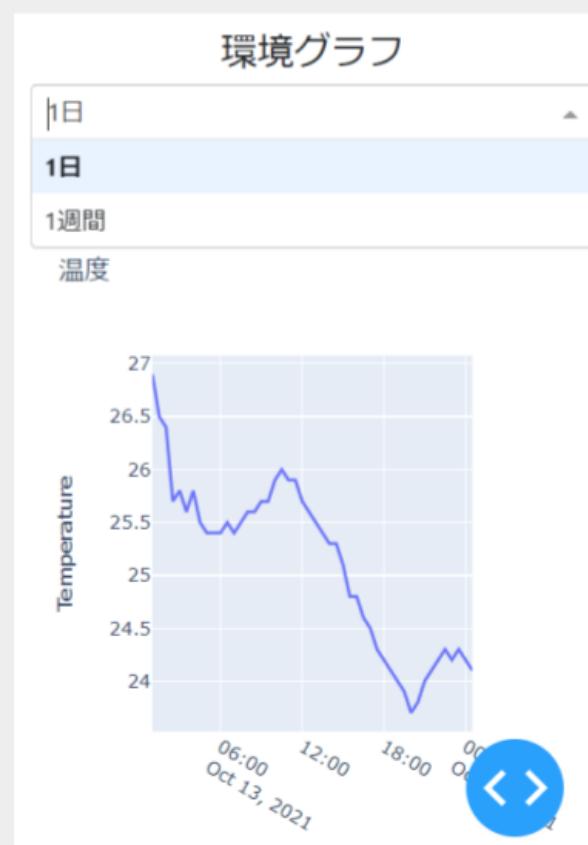
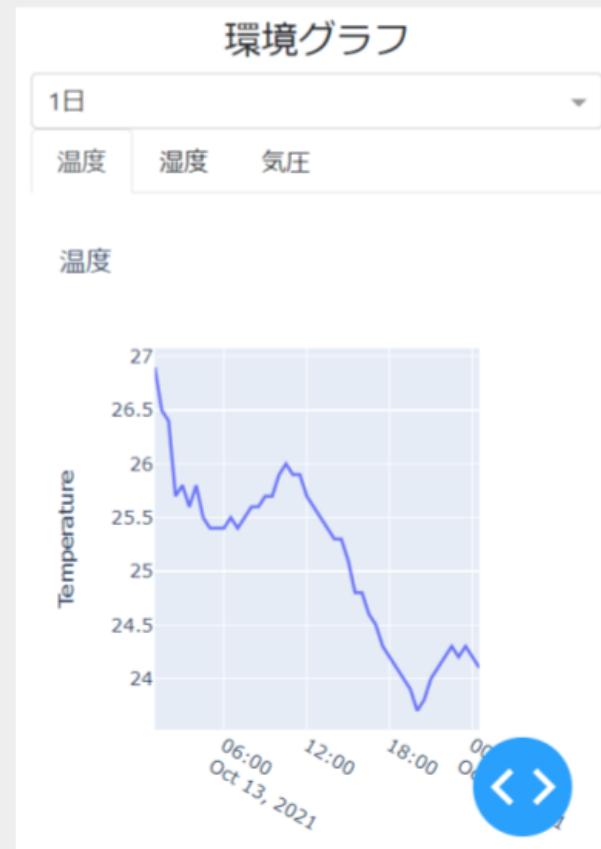
Inputフォームに文字を入れると
「出力:」の先が更新される

Hello Dash Callback

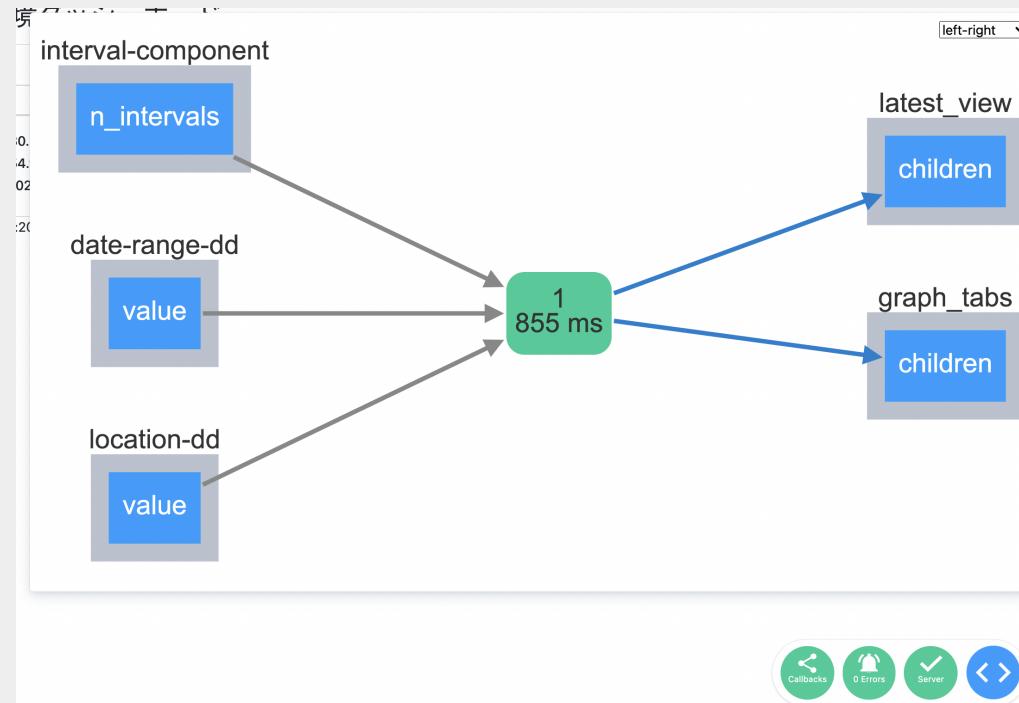
文字を入力すると、出力の部分が更新されます

入力:

出力: 変更します😊



ドロップダウンリストを切り替えることで、グラフの範囲を切り替える



Tips: ホットリロードによる開発

- 自動的にリロードしてくれる。
デバッグモードを有効にしておくと開発しやすい

```

if __name__ == "__main__":
    app.layout = _layout
    # debug=Trueでデバッグモード
    app.run_server(debug=True, host="0.0.0.0")
    
```

host="0.0.0.0"は外部公開として扱います。この辺もFlask踏襲ですかね

デモ: センサー情報を可視化する

センサー情報の取得方法は、直接センサーの値を取りに行く

- 時系列グラフを作るなら、データの記録は必須になる。
 - 実際のところはファイルかDB, 外部のデータソースへ保存して扱う方がわかりやすい
 - 今回はCSVファイルに1分ごと、30回分の測定結果を保存

センサーの最新の値を見る

デモ用アプリの該当箇所:

<https://github.com/hrsano645/homeenvdash-mini/blob/main/homeenvdash-mini.py#L83>

記録したセンサーの値をグラフ化する

デモ用アプリの該当箇所:

<https://github.com/hrsano645/homeenvdash-mini/blob/main/homeenvdash-mini.py#L106>

Tips: 複雑化したらまとめてコンポーネント化する

複雑なアプリやレイアウトを作ると、構造も複雑になりがち

- 複雑になることはしょうがない
- 複雑になるので関数などで部品（カプセル）化していく
- コールバックで更新したいコンポーネント箇所をカプセル化すると、コールバック側の更新処理を作る時に呼び出しやすい

慣れてくるとwebアプリを書いている様な扱いに

- 説明箇所コードのリンク: <https://github.com/hrsano645/homeenvdash-mini/blob/main/homeenvdash-mini.py#L103>

Tips: Bootstrapを使ってデザインを良くする

Dashの便利なライブラリ: dash-bootstrap-components (dbc) によるデザインの整え方

- Dashのデメリットは、CSSを扱ったデザインがしづらい
 - レスポンシブ対応とか
- CSSフレームワークのBootstrapを扱いやすくコンポーネント化されている
- 扱いやすいもののBootstrapを知っている必要あり

※Dashのv2バージョンアップに伴って、dash-bootstrap-componentsも追従したバージョンアップが行われます。

デモ中は最新バージョンのリリース候補版を利用してます。

デモ: デザインを整えてみる

- 実際にコードを見せつつ解説します
- レスポンシブ対応を例にします
- コードはこちらから見れます:

<https://github.com/hrsano645/homeenvdash-mini/blob/main/homeenvdash-mini-dbc.py>

目次

- 環境ダッシュボードを作った話
- PythonとIoT
- Dashでダッシュボードアプリを作る
- まとめ

まとめ

趣味プロジェクトを紹介しつつ、Pythonを使ったIoTとデータ可視化をデモを交えてお伝えしました

- IoT: ラズパイ+CircuitPythonを合わせる選択肢
- データ可視化: Dashでダッシュボードアプリを作る

本日お伝えしたこと

- 自分の欲しい物が無ければ作ろう
- 身の回りで見えない数字を可視化してみよう
 - 身近だけど見えないデータ

終わりに

今日のトークで、

- IoTやダッシュボードを作ることが楽しいと思ってもらえたうれしいです
- また作るきっかけにもなつたらさらにうれしいです
- 手間暇はかかるけど、そこはモノづくりの大変な所だと思います

存分に手間暇かけてテクノロジーを楽しみましょう！

最後にお知らせ

PyCon mini Shizuoka 2021 やります🎉

是非来てください👋

- 2021/11/20 土曜日
- 詳しくは公式サイトをチェック
 - <https://shizuoka.pycon.jp/2021>
 - Twitterアカウント: @PyconShizu
- LTと参加者募集をします。詳細は近日
- 同時にイベントの詳しい内容は近日公開します！

ぜひこちらにも遊びに来てください！

質問想定

- Dashを使うメリット、デメリットは？
 - メリットはHTMLを書く必要なくWEBアプリ作成ができる
 - デメリットはDB接続や高度な動的処理（ログイン処理やセッション管理）、API連携はそれほど得意ではない
 - それを解決する有償のサービスが提供されている
- 気圧センサーいろいろあるよ！
 - 知らなかつたのでありがとうございます！
- 精度はどう？
 - 数字というより変化を見たかったので、精度については今回は扱っていない
 - 実際にキャリブレーションする必要はあると思うし、高度なセンサーもあるので、用途によると思います。

