

# 2022年度 テックキャンプ報告

ES1-1 菅野 洋史

# 今期テーマ

今期はマルチテーマ

- テーマ1: VR/ARプレゼンツール <停止>
- テーマ2: ブラウザIoT <作業中>
- テーマ3: ブラウザで機械学習

## 共通の問題設定

- 1.いつでもどこでも開発できて、
- 2.手軽に試せるような最新技術検証  
ブラウザで動く技術にこだわる

# テーマ1: VR/ARプレゼンツール

## 停止中

- 今プレゼンしてるMozilla Hubs で良いのではという気持ちが高まったため
- 調査していたaframeというフレームワークがMozillaHubsの実装に利用されています。

## テーマ2: ブラウザIoT

# モチベーション

IoT x クラウド に疑問をもっていた。

- 提供側サービス終了により照明のON/OFFができなくなる事例
- AWSのトラブルによりレンタル自転車のロックがはずれなくなる事例
- Goog IoT Coreがサービス終了

## 例: 目の前の機器の情報を読み出したい

AWS等を使うと

1. アプリがMQTTプロトコルでクラウド上のAWS IoTに情報プッシュして、
2. AWS kinesisでイベントを分類
3. Lambdaで処理をしDynamoDBに保存
4. アプリからAPI経由で情報を要求し
5. LambdaがDynamoDBから情報を取り出して返却

(もちろん本当に必要なユースケースもあるが、、)

# 手軽にIoTを試したい

- クラウドを否定するわけではないが、、、
- 手元のある機器と簡単につないでIoTプロトタイピングしたい
- 開発環境はクラウドで、動作環境は実機でアプリを作ってみる
- ブラウザで動く技術にこだわりたい
  - 特定の環境にロックインされたくない



# どんなアプリをつくった？

1. Webサイトの1ページにアクセスする
2. そのページでBluetoothペアリングを行うと近くに置いてあるM5Stackにつながる
3. M5Stackのセンサ情報がwebページにレンダリングする
4. ついでに、ページに入力したメッセージがM5Stackに表示される

# 構成要素

- Astro.js(静的サイトジェネレータ)を利用してWebサイト作成
- WebBluetooth でM5Stackにアクセスし,情報交換する
- M5StackのプログラミングはUIFlowで実装

# Webサイト上にアプリが乗っている

Astro.jsというCMSで作成したWebサイトに、このプレゼン資料自体もアプリもすべて突っ込んでます。



# 接続



# M5Stackのジャイロデータをリアルタイムに表示

手元でM5Stackを回転させると反映される

Google カレンダー

最新ツイート / Twitter

M5StackCore2...

接続

ピッチ

ロール

-3.616

11.992

要素

コンソール

Recorder

ソース

ネットワーク

Performance insights

1

1

デフォルト レベル

1 件の問題

2 件の非

Welcome, browser console!

DevTools がソースマップの読み込みに失敗しました (chrome-extension://cofdbpoegempjloogbagkncekinflcnj/build/content.j のコンテンツを読み込めませんでした (システムエラー: net::ERR\_BLOCKED\_BY\_CLIENT) )

Uncaught (in promise) DOMException: User cancelled the requestDevice() chooser.

Getting Service...

{rolling: 3.708, pitch: -0.201}

{rolling: 8.924, pitch: -3.713}

{rolling: 11.935, pitch: -3.838}

{rolling: 12.084, pitch: -3.411}

{rolling: 12.286, pitch: -3.385}

{rolling: 11.992, pitch: -3.616}

# UIFlowプログラム

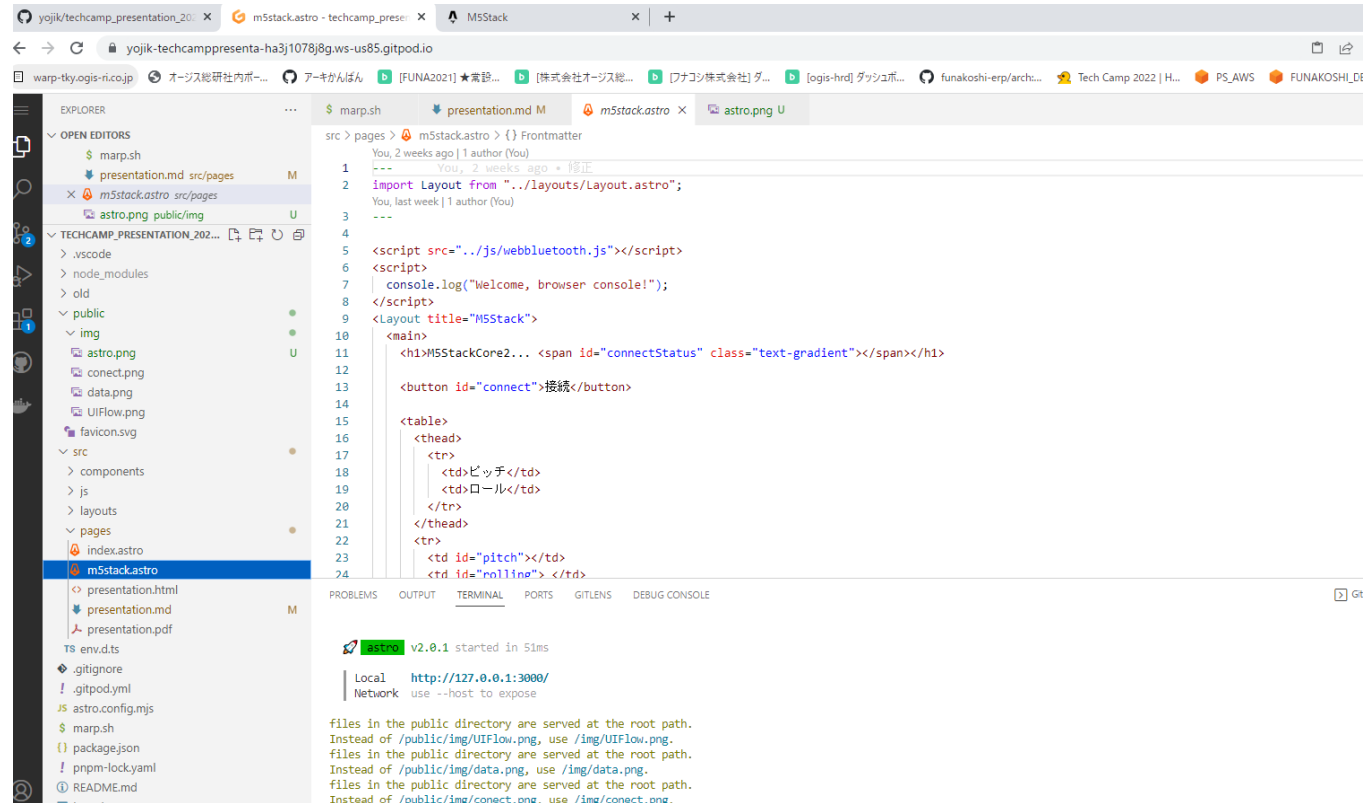
The image displays the UIFlow V1.11.1 interface for a project named "main". The interface is divided into several sections:

- Top Bar:** Shows the UIFlow logo, version "V1.11.1", a "プロジェクト" (Project) button, and the project name "main". It also includes tabs for "Blockly" and "Python", and various utility icons.
- Left Panel:** Contains a visual representation of a device screen with "start" and "stop" buttons. Below it are "Units" and "Stamps" indicators.
- Component Palette:** A list of components on the right side, including MQTT, Http, Socket, Modbus Master, Modbus Slave, CAN, Blynk, BLE (expanded to show BLE Peripheral and BLE Central), and Pin Servo.
- Scripting Area:** The main workspace for visual programming, showing a script for a BLE peripheral named "M5Core2".

**Scripting Details:**

- Setup:** "Init BLE Peripheral name" set to "M5Core2".
- start / wasPressed:** "タイマー timer1 を 900 [ミリ秒]間隔 PERIODIC モードで開始" (Start timer1 with 900ms interval in PERIODIC mode).
- stop / wasPressed:** "タイマー timer1 を停止" (Stop timer1).
- タイマー timer1 が呼び出されたとき (When timer1 is called):** A "BLE send data" block containing a "マップを作成" (Create Map) block. The map has two entries: "pitch" with value "ピッチ角" and "rolling" with value "ロール角". The map is then converted to JSON format ("をjson形式に変換したデータ").

# 開発環境もWeb上のコンテナで



# テーマ3: ブラウザで機械学習

## お遊び程度

- Colaboratory でStable Diffusion による画像生成は実施
- ChatGPT(というよりもOpenAI)クライアントをdenoで作成
- ChatGPTと外部リソースを組み合わせるLangChainを試してみる
-



# LangChain

- OpenAIは本質的に文字列のパターンから、ありそうな返答を作るだけなので意外と嘘つき
- ちょっとした計算でも「それっぽい数字」を出しているだけなので嘘をつく  
LangChainではOpenAIに質問した回答を他のシステムに流し込み、正しい結果を得たりすることができる
- また名前のとおり、質疑の流し込みを何段も積み重ねることができる

# サンプル

日本語の質問をOpenAIに流し込んで、Pythonに変換し、その処理結果を取得する。

```
# LLMの準備
llm = OpenAI(temperature=0.9)

# 数式eval
llm_math_chain = LLMMathChain(llm=llm, verbose=True)

# 計算をおこなうツール
# 計算を行うような文書パターンが出てきたらこちらで代行する
tools = [
    Tool(
        name="Calculator",
        func=llm_math_chain.run,
        description="useful for when you need to answer questions about math"
    )
]
llm = OpenAI(temperature=0)
agent = initialize_agent(tools, llm, agent="zero-shot-react-description", verbose=True)
```

## 普通のOpenAIなら

```
print(llm("1000円をもって買い物にでかけました。450円の本を買い、135円のジュースを2本買いました。おつりはいくらですか"))
```

## 自信たっぷり間違える

残りは425円です。

# LangChainなら

質問から数式を組み立てるところまではOpenAIの言語モデルを使い、その式をPythonで実行する

# 結局間違っている！！！！！！

計算しようとして間違えてる心意気は伝わる。。

```
agent.run("1000円をもって買い物にでかけました。450円の本を買い、135円のジュースを2本買いました。残りはいくらですか")
```

```
> Entering new AgentExecutor chain...  
I need to calculate how much money is left  
Action: Calculator  
Action Input: 1000 - 450 - (135 * 2)  
  
> Entering new LLMMathChain chain...  
1000 - 450 - (135 * 2) # ここまでやれてるのに。。  
Answer: 220 # 結局間違える。。  
> Finished chain.  
  
Observation: Answer: 220  
Thought: I now know the final answer  
Final Answer: 残りは220円です。  
  
> Finished chain.  
残りは220円です。
```

# まとめ

雑多な内容なのでまとめ辛いですが、、

## WebBluetooth

- 「開発をクラウド」「動作はエッジ」ですべてブラウザ上でアプリを作成した
- クラウドを利用しつつ、クラウドに利用されない気持ち！

## LangChain

- 機械学習本体にはもう手をつけられる時代ではない
- どうやって組み合わせてツールとして成立させるか