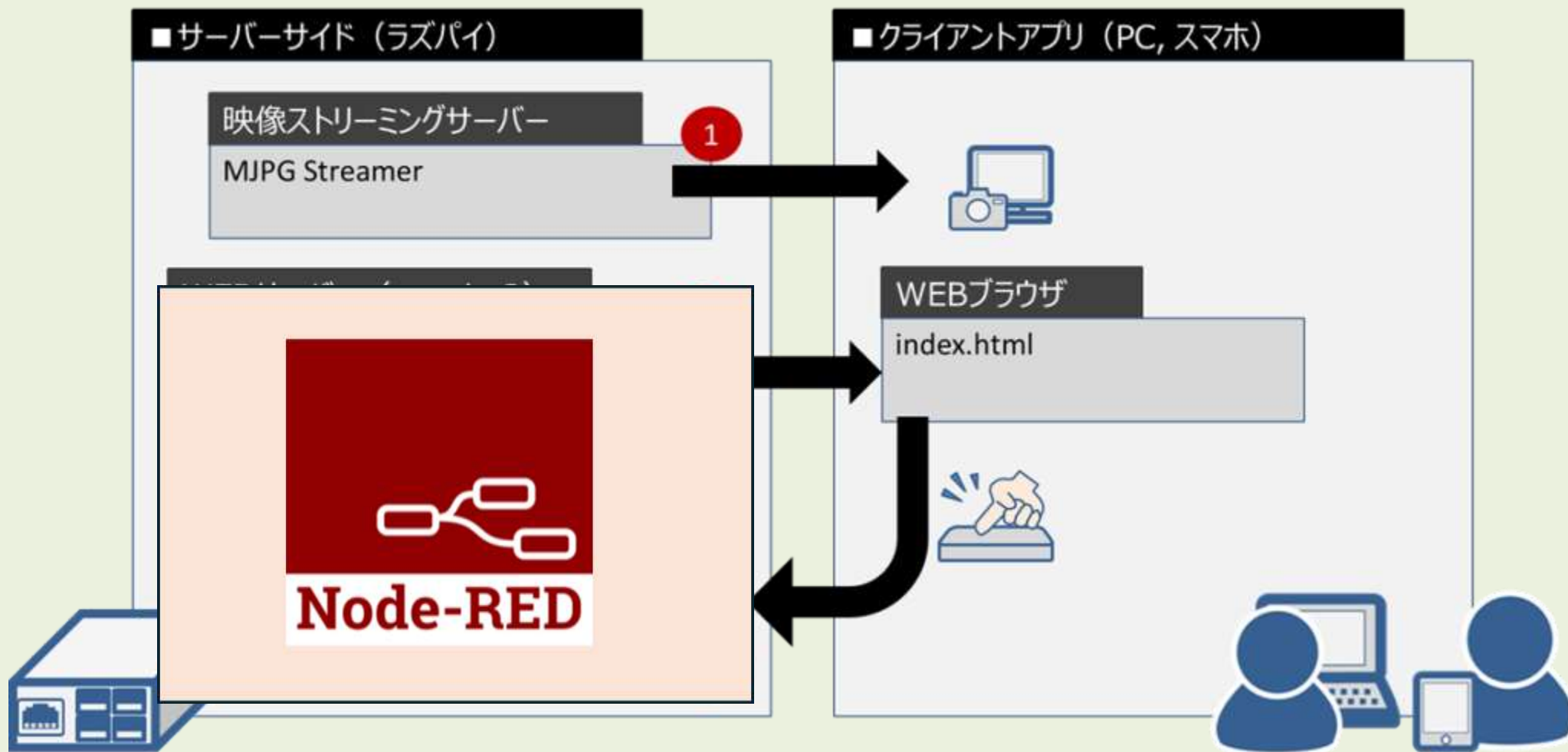
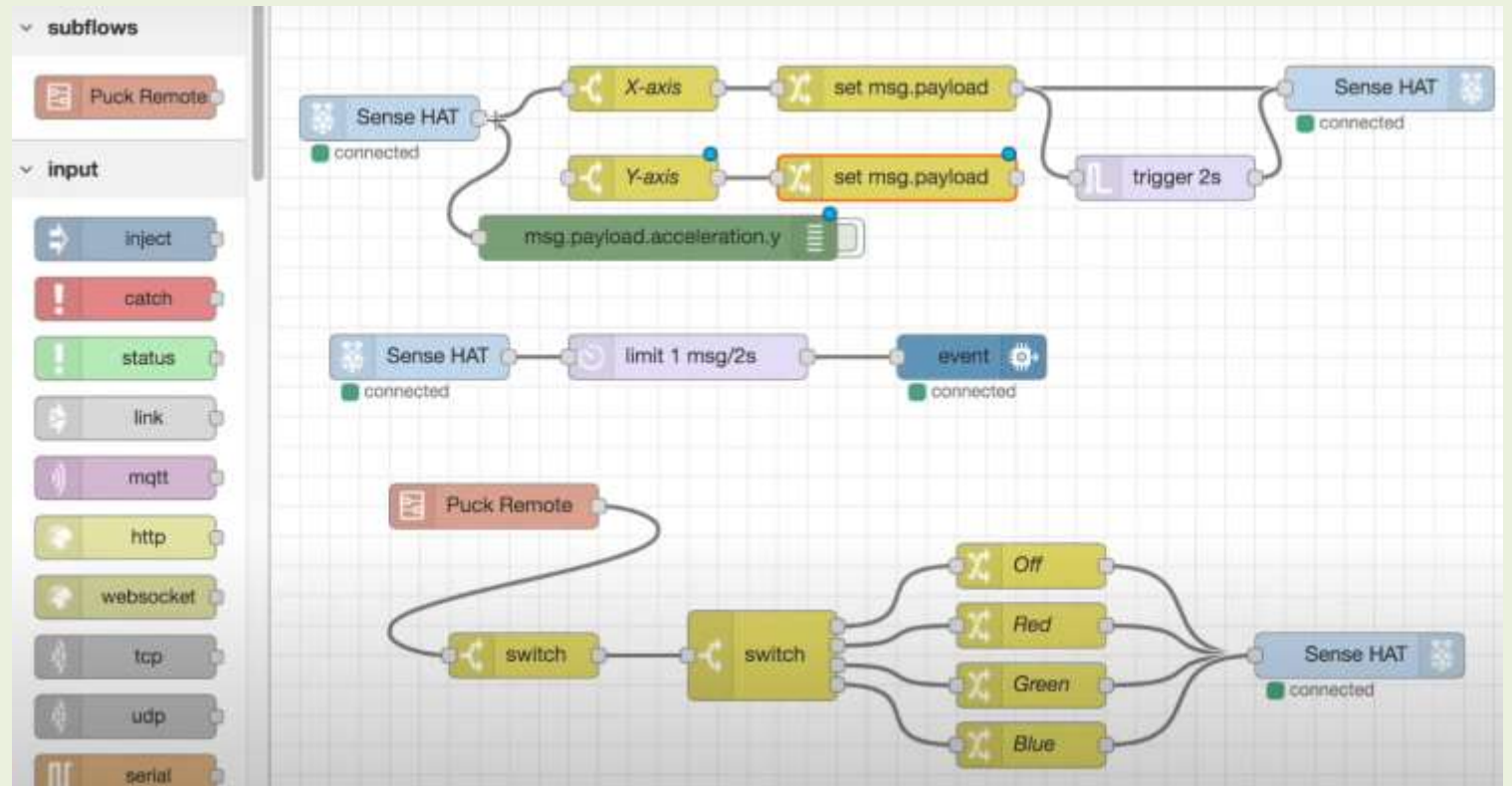
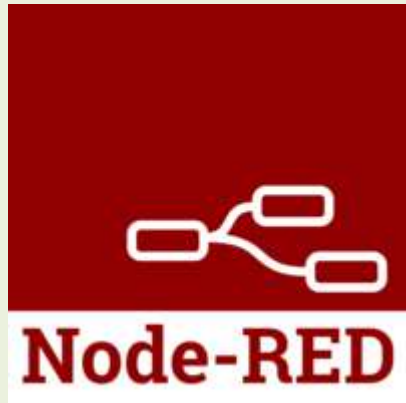


ラズタンク・操作ダッシュボード開発



ノード・レッドとは？



- IBMが開発
- ビジュアルプログラミング型の開発ツール
- WEBブラウザのエディタ
- IoT（ハードウェアデバイス + API + オンラインサービス を接続できる）
- JavaScript関数を作成できる

さっそく、みんなでやってみよう！

Node-RED を起動する

CoderDojo青梅のラズパイにはNode-REDがインストール済み

```
$ node-red-start
```

```
pi@razpi10:~ $ node-red-start

Start Node-RED

Once Node-RED has started, point a browser at http://192.168.1.63:1880
On Pi Node-RED works better with the Firefox or Chrome browser

Use node-red-stop to stop Node-RED
Use node-red-start to start Node-RED again
Use node-red-log to view the recent log output
Use sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use sudo systemctl disable nodered.service to disable autostart on boot

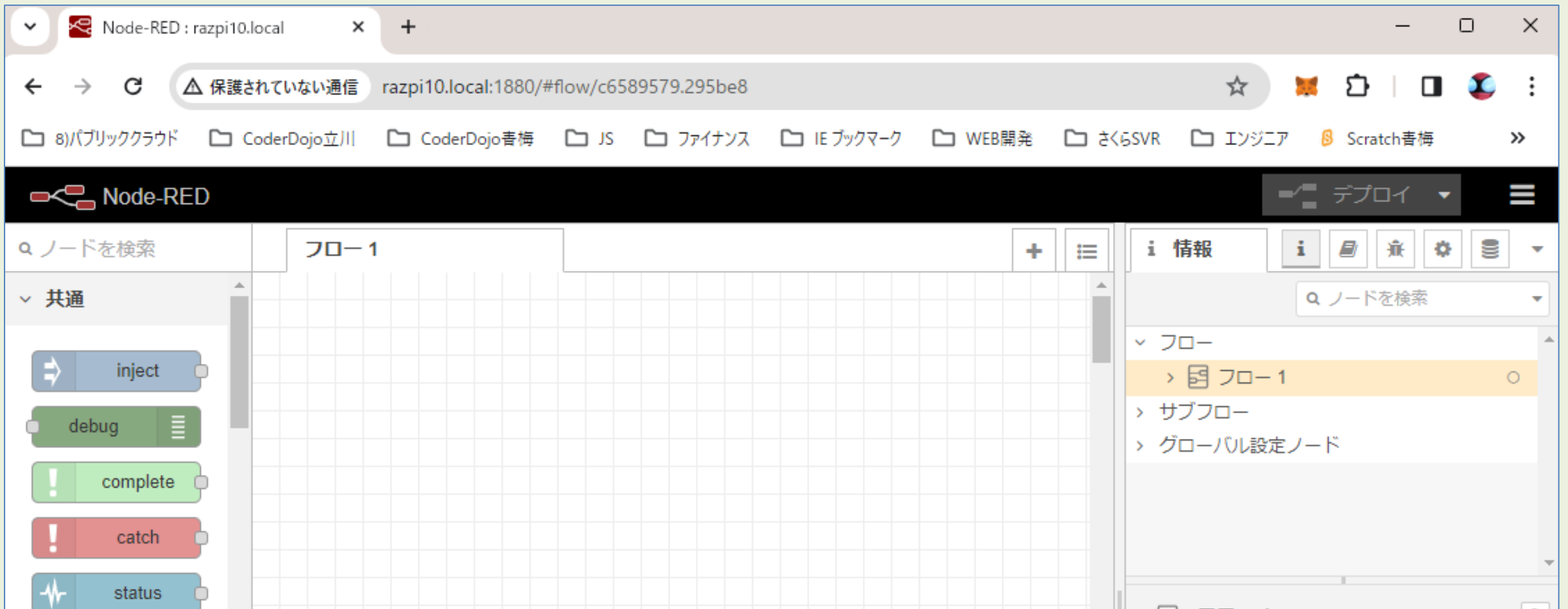
To find more nodes and example flows - go to http://flows.nodered.org

Starting as a systemd service.
16 Jan 22:25:29 - [info]
Welcome to Node-RED
=====
16 Jan 22:25:29 - [info] Node-RED バージョン: v1.3.4
```

```
11 Feb 19:52:04 - [info] サーバは http://127.0.0.1:1880/ で実行中です
```

Node-RED にアクセス

同じネットワーク上のPCのWEBブラウザを起動
<http://razpi01.local:1880/>



The screenshot shows a web browser window with the following details:

- Browser Tab:** Node-RED : razpi10.local
- Address Bar:** <http://razpi10.local:1880/#flow/c6589579.295be8>
- Bookmarks:** 8)パブリッククラウド, CoderDojo立川, CoderDojo香梅, JS, ファイナンス, IEブックマーク, WEB開発, さくらSVR, エンジニア, Scratch香梅
- Node-RED Interface:**
 - Header:** Node-RED logo, デプロイ button, and a hamburger menu icon.
 - Search Bar:** ノードを検索
 - Flow Editor:** A central workspace with a grid and a tab labeled "フロー 1".
 - Left Panel (Common Nodes):** A list of nodes under the "共通" category:
 - inject (blue)
 - debug (green)
 - complete (green)
 - catch (red)
 - status (blue)
 - Right Panel (Information):** Information panel with a search bar and a tree view:
 - 検索: ノードを検索
 - フロー
 - フロー 1 (selected)
 - サブフロー
 - グローバル設定ノード

はじめての Node-RED プログラミング

はじめてのプログラミング 1

1

inject

debug

complete

catch

status

link in

link out

comment

タイムスタンプ

msg.payload

ダブルクリック

ダブルクリック

プロパティ

名前

名前

msg.payload = 日時

msg.payload をデバッグウィンドウに出力

出力先 デバッグウィンドウ

2

デプロイ

情報

ノードを検索

フロー

- フロー 1
- サブフロー
- グローバル設定ノード

3

デバッグ

全てのフロー

実行 (クリック)

タイムスタンプ

4

5

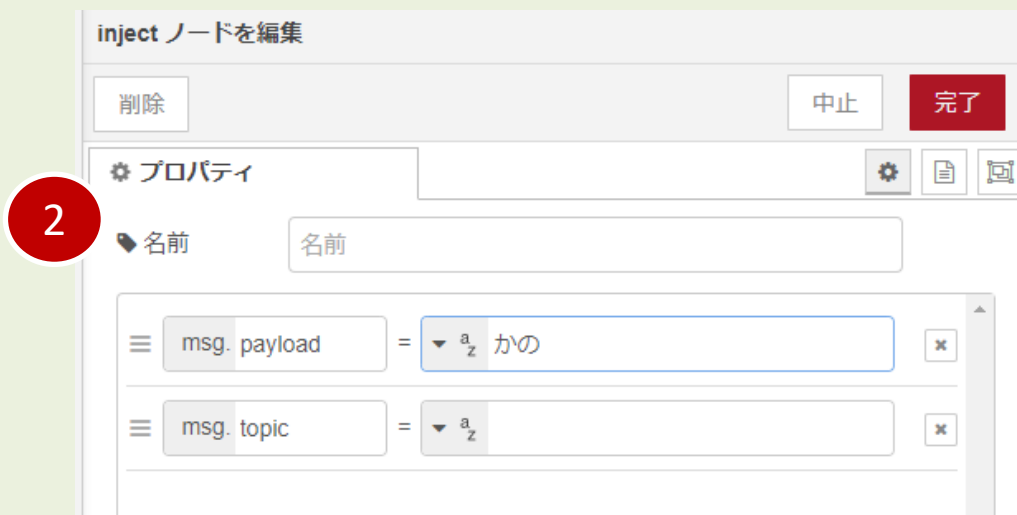
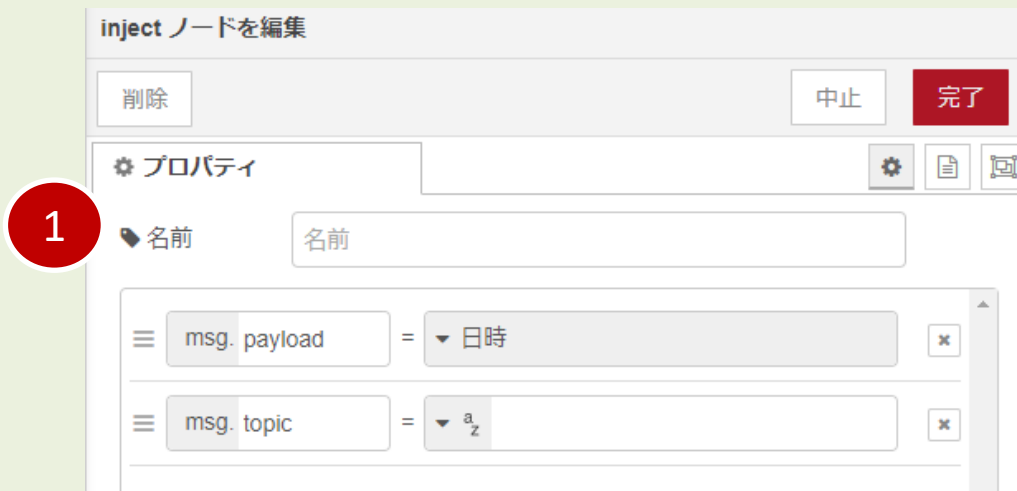
2024/1/16 22:38:38 node: 7321a995.078f68

msg.payload : number

1705412319464

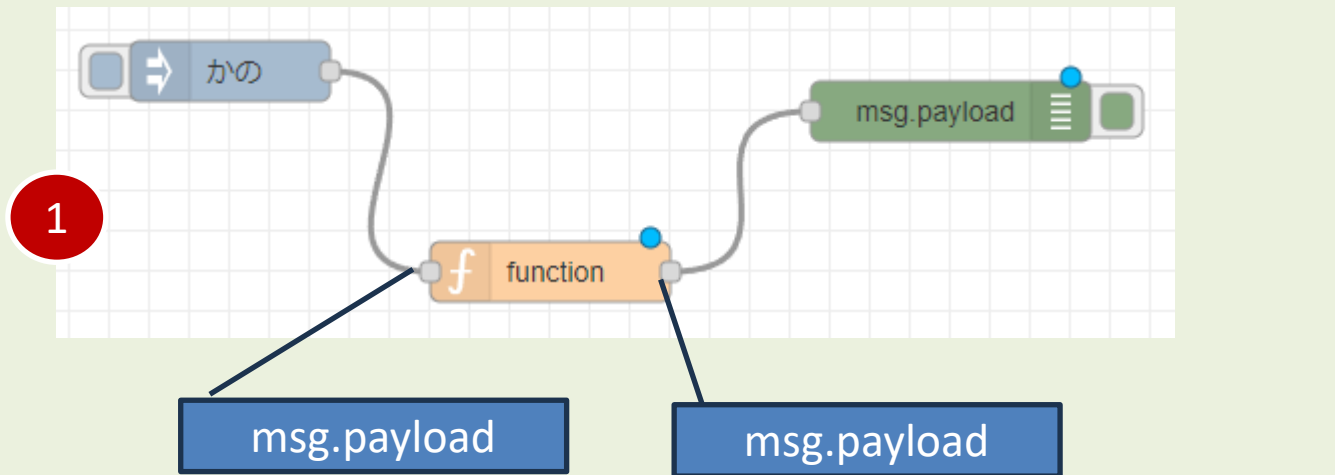
日時の数字が表示される

はじめてのプログラミング 2 (改造)



はじめてのプログラミング 3 (改造)

Functionノードに JavaScript コードを書く



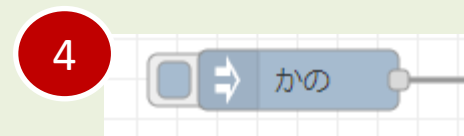
2

The screenshot shows the configuration panel for the function node. It has tabs for '設定' (Settings), '初期化処理' (Initialization), 'コード' (Code), and '終了処理' (Finalization). The 'コード' tab is selected, showing the following JavaScript code:

```
1 var strMsg = msg.payload;
2 msg.payload = strMsg + "いちろう"
3 return msg;
```

3

The screenshot shows the top right corner of the interface with a red 'デプロイ' (Deploy) button. Below it is a '情報' (Info) section with icons for information, copy, refresh, settings, and list. A search bar contains the text 'ノードを検索' (Search nodes).



5

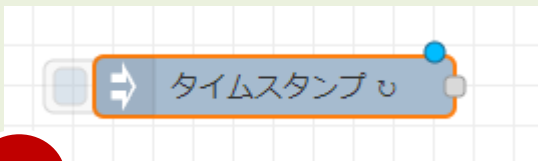
The screenshot shows the 'デバッグ' (Debug) console with a list of log entries:

- 2024/1/16 22:38:38 node: 7321a995.078f68
msg.payload : number
1705412319464
- 2024/1/16 22:40:06 node: 7321a995.078f68
msg.payload : string[2]
"かの"
- 2024/1/16 22:46:09 node: e80a623.793ada
msg.payload : string[6]
"かのいちろう"

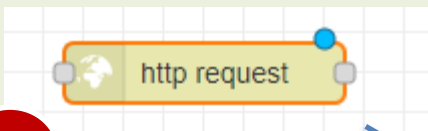
はじめての アプリ開発 (防災アプリ)

はじめてのアプリ開発（防災アプリ）

Inject トリガー（ボタンを押す）



1



2

http request でWEB
ページのファイル「地
震データ」を取得する



https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/significant_month.csv

はじめてのアプリ開発（防災アプリ）

CSVノードを追加



csv ノードを編集

削除 中止 完了

プロパティ

列名 コンマ区切りで列名を入力

区切り文字 コンマ

名前 名前

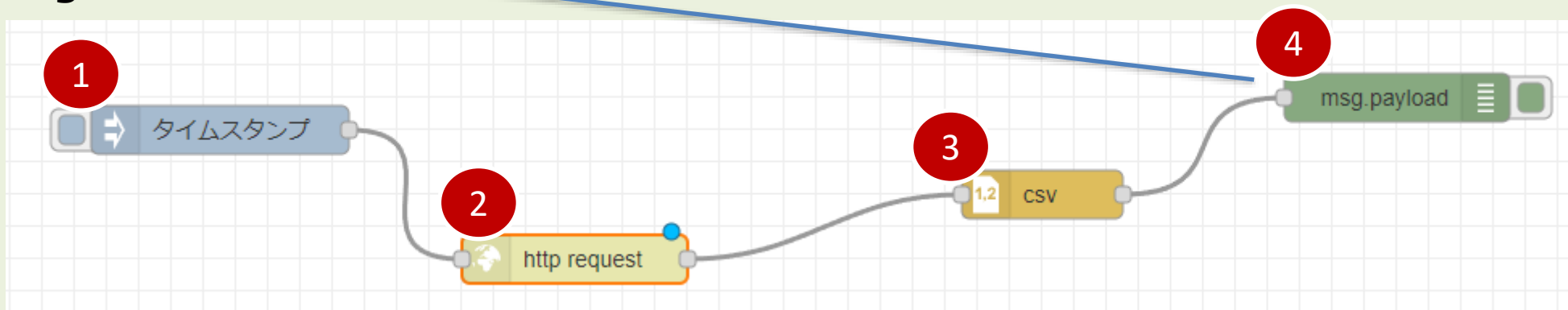
CSVからオブジェクトへ変換

入力 最初の 0 行を無視する

- 1行目に列名を含む
- 数値を変換する
- 空の文字を含む
- null値を含む

はじめてのアプリ開発（防災アプリ）

debugノードを追加



1. Inject プログラムを実行する
2. WEBページからCSVファイルをダウンロード
3. CSVデータを整形
4. デバッグ画面に表示



```
2024/1/16 22:57:42 node: e0b1c9e4.004538
msg.payload : Object
  ▶ { time: "2024-01-11T09:20:26.809Z",
    latitude: 36.5061, longitude:
    70.5994, depth: 206.561, mag: 6.4 ... }

2024/1/16 22:57:42 node: e0b1c9e4.004538
msg.payload : Object
  ▶ { time: "2024-01-08T20:48:42.916Z",
    latitude: 4.8606, longitude:
    126.1861, depth: 68.017, mag: 6.7 ... }

2024/1/16 22:57:42 node: e0b1c9e4.004538
```

はじめてのアプリ開発（防災アプリ）

```
2024/1/16 23:11:39 node: e0b1c9e4.004538
msg.payload : Object
  ▾ object
    time: "2023-12-18T15:59:30.033Z"
    latitude: 35.7377
    longitude: 102.8059
    depth: 10
    mag: 5.9
    magType: "mww"
    nst: 157
    gap: 12
    dmin: 0.755
    rms: 0.79
    net: "us"
    id: "us7000ljvg"
    updated: "2024-01-14T15:06:24.649Z"
    place: "39 km WNW of Linxia
    Chengguanzhen, China"
    type: "earthquake"
    horizontalError: 5.43
    depthError: 1.757
```

2023年12月18日 15時59分

緯度 latitude

経度 longitude

深さ depth

マグニチュード mag

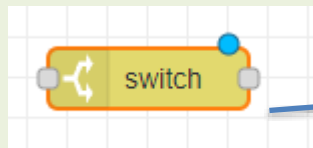
google map

検索: 35.7377, 102.8059



はじめてのアプリ開発（防災アプリ）

switchノードを追加



switch ノードを編集

削除 中止 完了

プロパティ

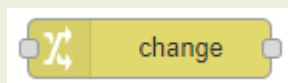
名前 名前

プロパティ msg.payload.mag

>= a_z 6 → 1 x

The screenshot shows the configuration for a switch node. The 'msg.payload.mag' property is selected. A rule is defined with the operator '>=' and the value 'a_z 6'. The switch is set to '→ 1'.

changeノードを追加



change ノードを編集

削除 中止 完了

プロパティ

名前 名前

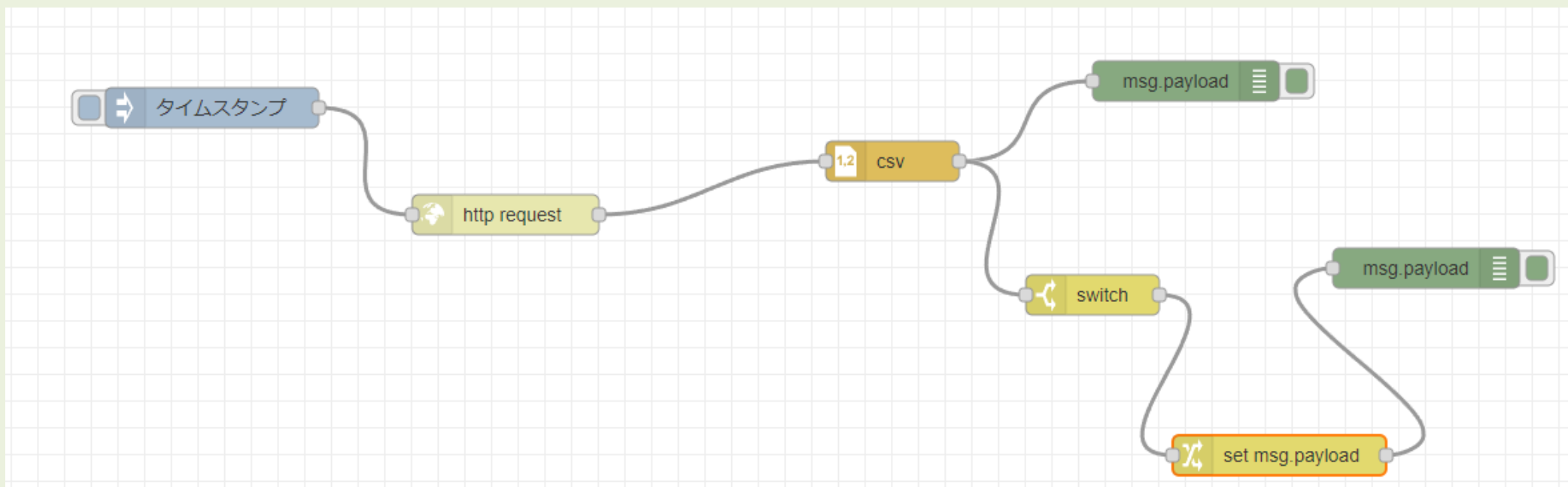
ルール

値の代入 msg.payload

対象の値 a_z パニック!

The screenshot shows the configuration for a change node. The 'msg.payload' property is selected. A rule is defined with the operator '値の代入' and the value 'a_z パニック!'.

はじめてのアプリ開発（防災アプリ）

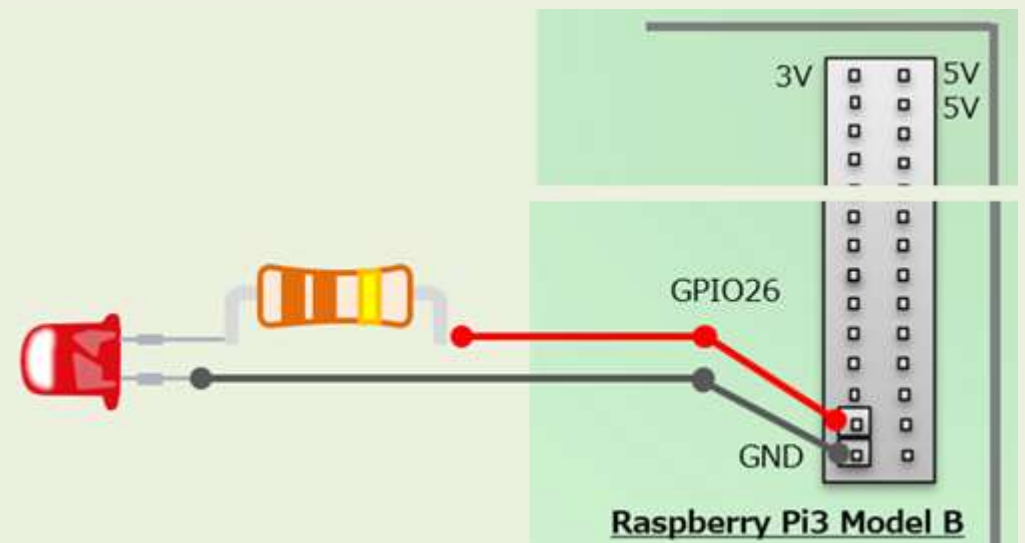


```
2024/1/16 23:11:39 node: e0b1c9e4.004538
msg.payload : Object
  ▶ { time: "2023-12-20T12:11:22.247Z",
    latitude: -15.8941, longitude:
    -72.508, depth: 101.325, mag: 6.2 ... }
```

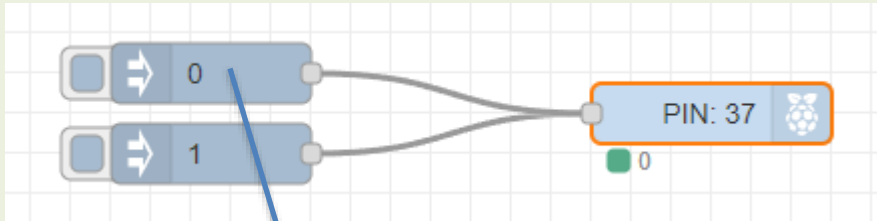
```
2024/1/16 23:11:39 node: e0b1c9e4.004538
msg.payload : Object
  ▶ { time: "2023-12-18T15:59:30.033Z",
    latitude: 35.7377, longitude:
    102.8059, depth: 10, mag: 5.9 ... }
```

```
2024/1/16 23:11:39 node: dcf5042.bc15df8
msg.payload : string[5]
"パニック!"
```


はじめての Node-RED 電子工作



はじめての Node-RED 電子工作



inject ノードを編集

削除 中止 完了

プロパティ

名前 名前

msg. payload = 0

msg. topic =

文字列

数値

GPIO17 - 11	GPIO18
GPIO27 - 13	14 - Ground
GPIO22 - 15	16 - GPIO23
3.3V Power - 17	18 - GPIO24
MOSI - GPIO10 - 19	20 - Ground
MISO - GPIO09 - 21	22 - GPIO25
SCLK - GPIO11 - 23	24 - GPIO8 - CE0
Ground - 25	26 - GPIO7 - CE1
SD - 27	28 - SC
GPIO05 - 29	30 - Ground
GPIO06 - 31	32 - GPIO12
GPIO13 - 33	34 - Ground
GPIO19 - 35	36 - GPIO16
GPIO26 - 37	38 - GPIO20
Ground - 39	40 - GPIO21

37

出力形式

デジタル出力

端子の状態を初期化

名前

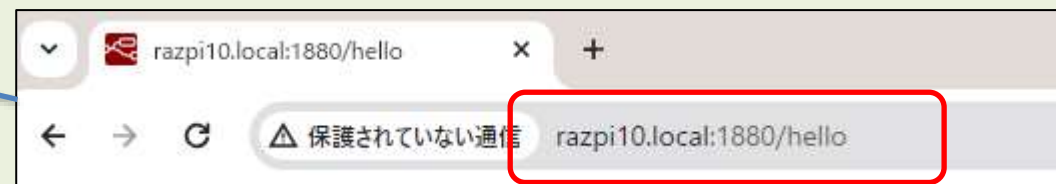
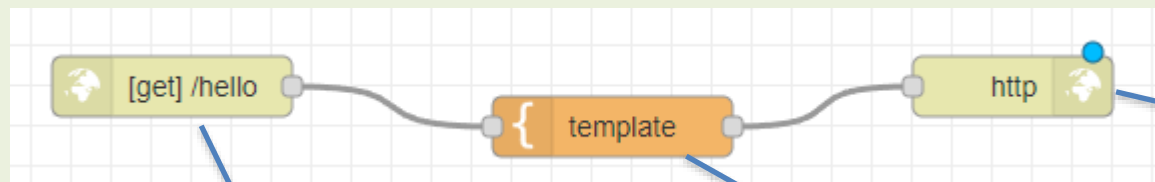
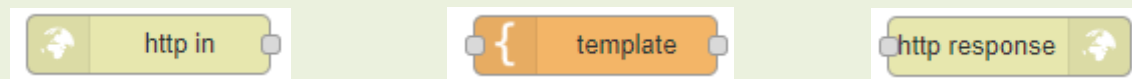
名前

使用中の端子: 37

注釈: 「出力形式」として「デジタル出力」を用いる場合、入力値は0または1の数値である必要があります。

WEBページ（HTTPエンドポイント作成）

WEBページ (HTTPエンドポイント作成)



http in ノードを編集

削除 中止 完了

プロパティ

メソッド GET

URL hello

名前 名前

template ノードを編集

削除 中止 完了

プロパティ

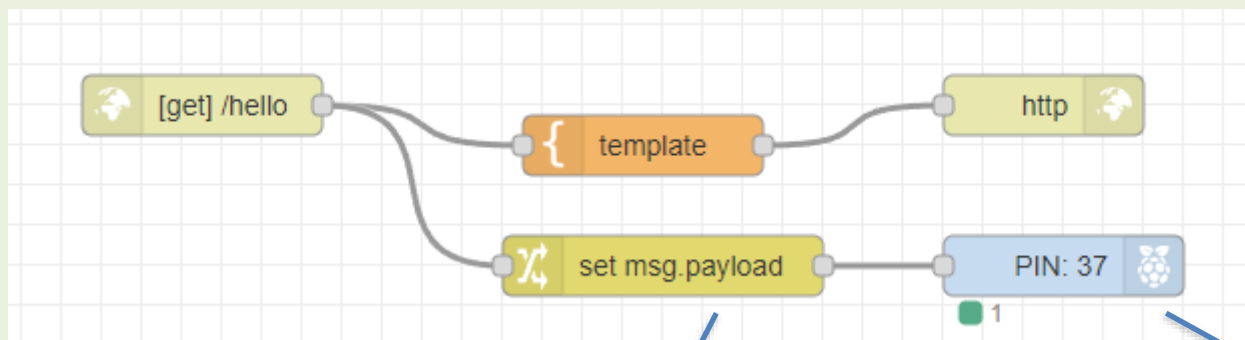
名前 名前

プロパティ msg.payload

テンプレート 構文: mustache

```
1 <html>
2   <head></head>
3   <body>
4     <h1>Hello World!</h1>
5   </body>
6 </html>
```

WEBページ (HTTPエンドポイント作成)



change ノードを編集

削除 中止 完了

プロパティ

名前 名前

ルール

値の代入 ▼ msg. payload

対象の値 ▼ % 1

SCLK - GPIO11 - 23	<input type="radio"/>	24 - GPIO8 - CE0	<input type="radio"/>
Ground - 25	<input type="radio"/>	26 - GPIO7 - CE1	<input type="radio"/>
SD - 27	<input type="radio"/>	28 - SC	<input type="radio"/>
GPIO05 - 29	<input type="radio"/>	30 - Ground	<input type="radio"/>
GPIO06 - 31	<input type="radio"/>	32 - GPIO12	<input type="radio"/>
GPIO13 - 33	<input type="radio"/>	34 - Ground	<input type="radio"/>
GPIO19 - 35	<input type="radio"/>	36 - GPIO16	<input type="radio"/>
GPIO26 - 37	<input checked="" type="radio"/>	38 - GPIO20	<input type="radio"/>
Ground - 39	<input type="radio"/>	40 - GPIO21	<input type="radio"/>

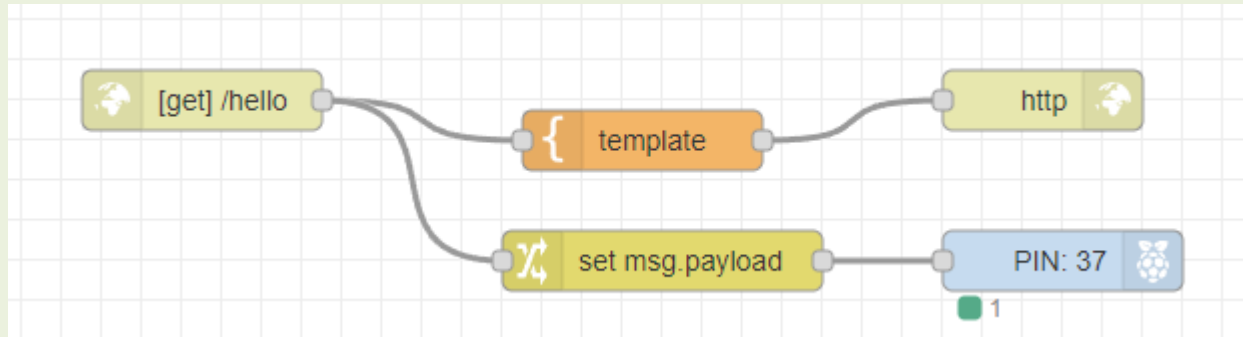
37

出力形式 デジタル出力 ▼

端子の状態を初期化

名前 名前

WEBページ（HTTPエンドポイント作成）



△ 保護されていない通信 razpi10.local:1880/hello

Hello World!

WEBブラウザに「Hello World!」が表示される。
同時にGPIO37に接続したLEDが点灯する。

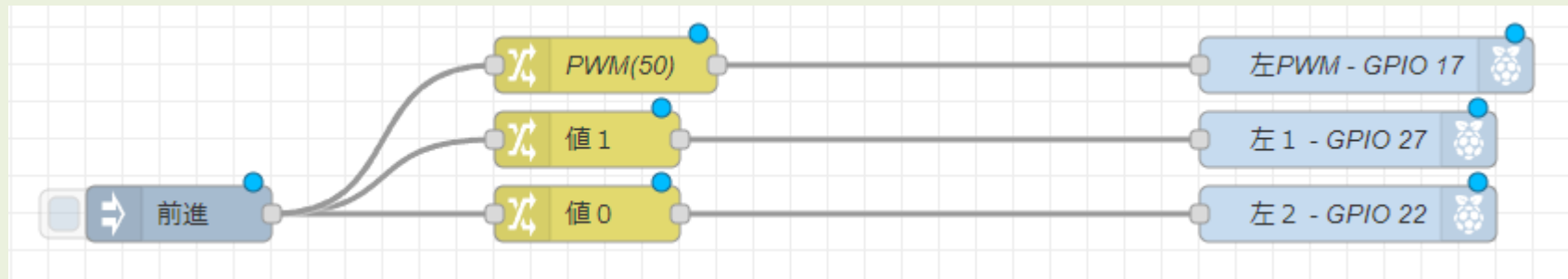
ラズタンクのWEB-APIを作る！

信 razpi10.local:1880/forward

信 razpi10.local:1880/back

信 razpi10.local:1880/stop

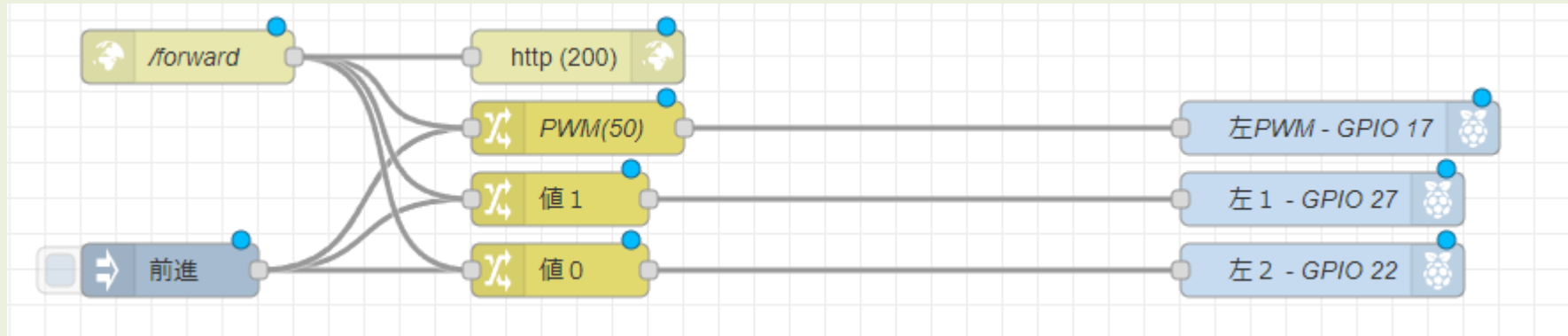
左側キャタピラー：前進駆動



```
# 初期設定(GPIO番号)  
LeftPWM = 17  
LeftIN1 = 27  
LeftIN2 = 22  
  
# 前進コマンド  
def moveForward()  
    GPIO.PWM( LeftPWM, 50 )  
    GPIO.output( LeftIN1, 1 )  
    GPIO.output( LeftIN2, 0 )
```



左側キャタピラー：前進駆動（WEB-API）



http response ノードを編集

削除

プロパティ

名前: 名前

ステータスコード: 200

ヘッダ

http in ノードを編集

削除

プロパティ

メソッド: GET

URL: forward

名前: /forward

```
# 初期設定(GPIO番号)
```

```
LeftPWM = 17
```

```
LeftIN1 = 27
```

```
LeftIN2 = 22
```

```
# 前進コマンド
```

```
def moveForward()
```

```
    GPIO.PWM( LeftPWM, 50 )
```

```
    GPIO.output( LeftIN1, 1 )
```

```
    GPIO.output( LeftIN2, 0 )
```

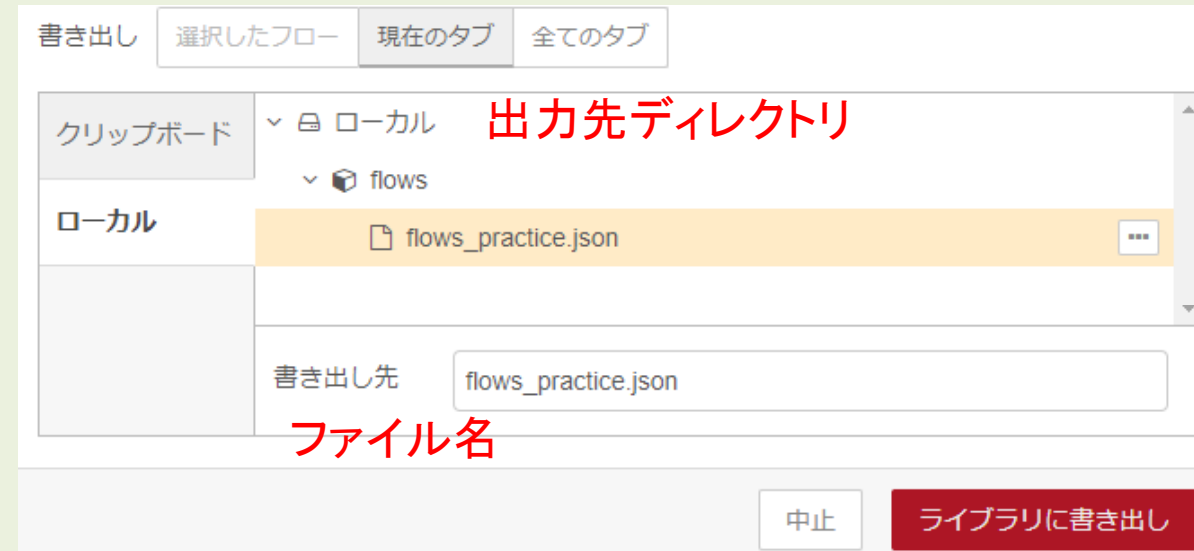
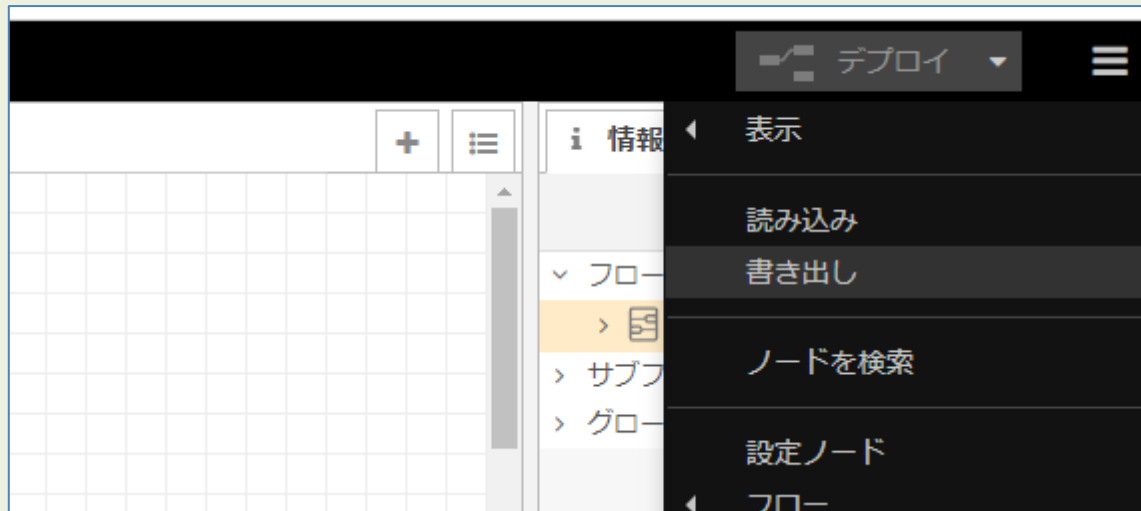
フローの保存先

ラズパイ側にファイルを保存するには

デプロイすると作成されるファイル

```
$ cd .node-red  
$ ls -l  
-rw-r--r-- 1 pi pi 8539 1月 17 23:22 flows_razpi10.json
```

ファイルに出力する(書き出し)



```
[{"id":"3227a95.bfdda56","type":"tab","label":"フロー 1","disabled":false,"info":""},  
{"id":"438bd2d8.1b960c","type":"inject","z":"3227a95.bfdda56","name":"","props":[{"p":"payload"},  
{"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"前進","p  
{"id":"4e59432c.56ec4c","type":"inject","z":"3227a95.bfdda56","name":"","props":[{"p":"payload"}, {"p":"topic",
```

ここから先

