

ラズパイで始めるIoTハンズオン

～電気のキホンからデータ送受信、可視化まで～

Part.2 Node-REDを使う

今日やること

- Node-REDを使って、プログラマブルな動作をさせる
- Zabbix Serverにセンサーデータを送信する

目次

- Node-Redの基礎
- Lチカ (GPIO出力)
- 温湿度センサー
- HTTPサーバ (API)
- Zabbixへのメトリクス送信
- 注意事項

自己紹介



江草 陽太

さくらインターネット 執行役員
技術推進統括担当

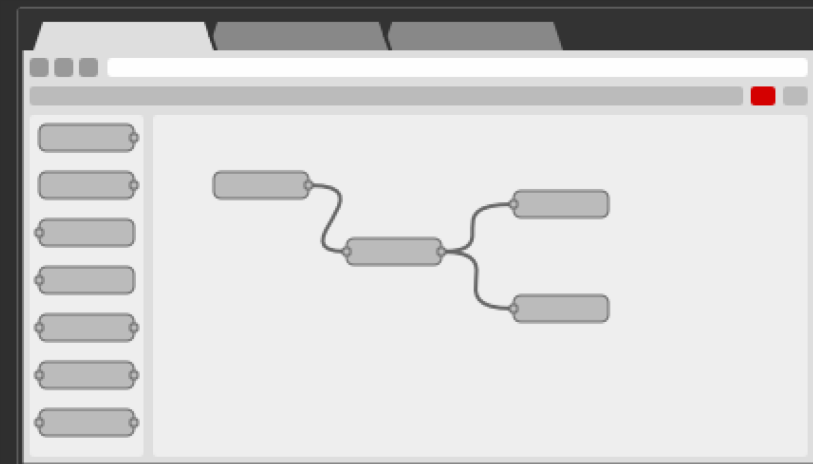
NW/DB/SC スペシャリスト
sakura.ioの設計・開発

Node-REDの基礎

Node-REDとは

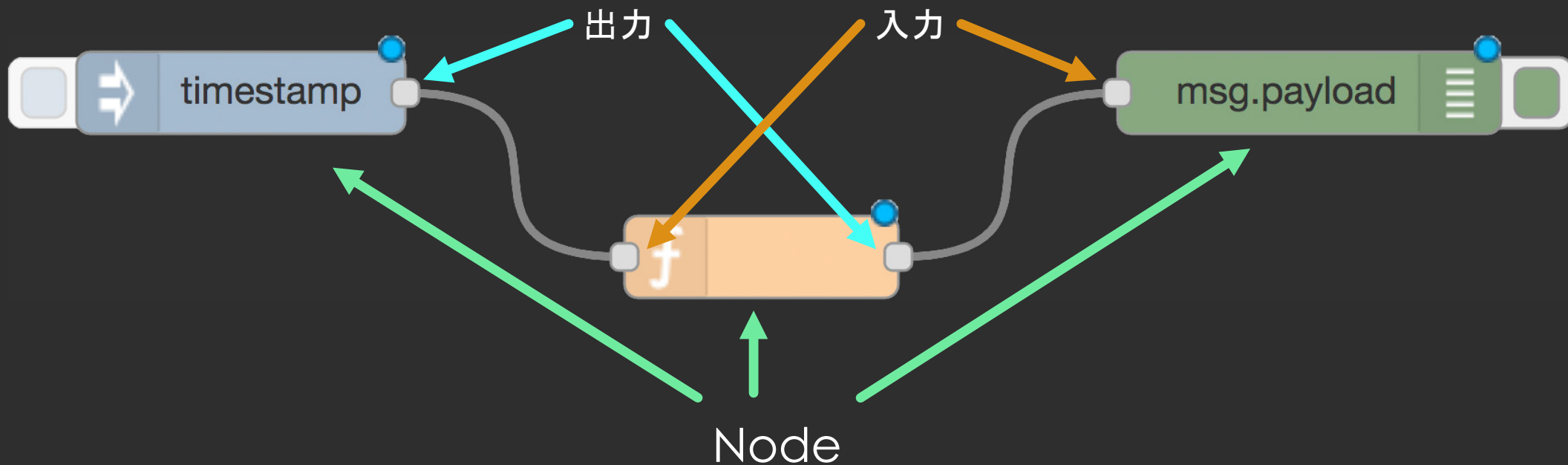
JavaScriptで開発された、ビジュアルプログラミングツール。入力と出力を持った「Node」を繋ぐことで動作を表現することができる。

JavaScriptのパッケージマネージャ「npm」を使って、公開されている様々なNodeをインストールすることが可能。

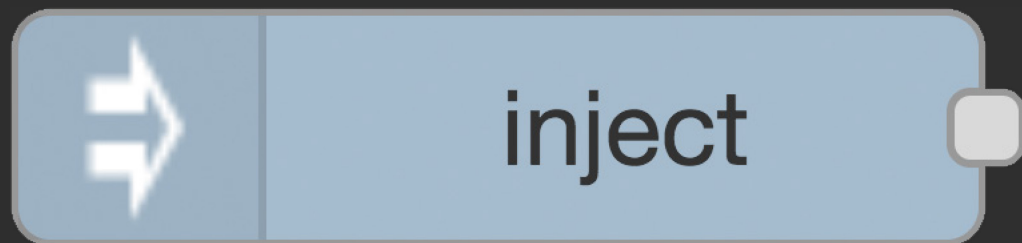


Node-REDとは

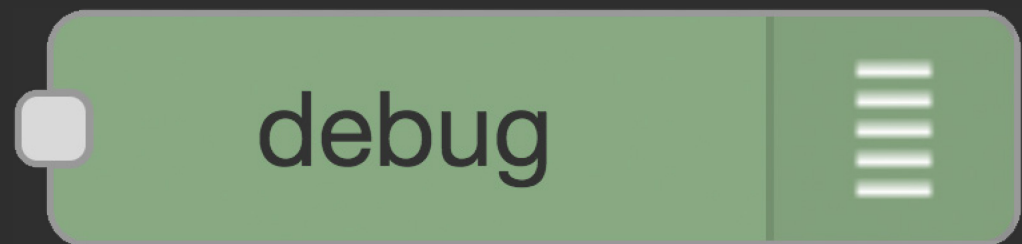
一連の動作の流れを Flow と呼ぶ



はじめてのFlow -利用するNode-



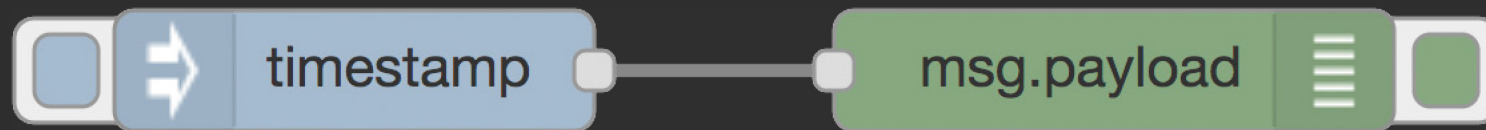
手動または定期的にメッセージをフローに送出します。メッセージのペイロードには、文字列、JavaScriptオブジェクト、現在の時刻など、さまざまな値を指定できます。



サイドバーの「デバッグ」タブに、選択したメッセージプロパティの値を表示します。デフォルトの表示対象は `msg.payload` です。

はじめてのFlow -Flow-

1. サイダーからノードをドラッグアンドドロップ
2. ノード間を接続
3. ノードの設定変更 (次ページ参照)
4. デプロイボタンをクリック



はじめてのFlow -Injectノードの設定-

1. Injectノードをダブルクリック
2. ペイロード「日時」
3. 繰り返し「指定した時間間隔」
4. 時間間隔「5秒」

inject ノードを編集

削除 中止 完了

▼ プロパティ

✉ ペイロード ▼ 日時

☰ トピック

Node-RED起動の 0.1 秒後、以下を行う

🔄 繰り返し 指定した時間間隔

時間間隔 5 秒

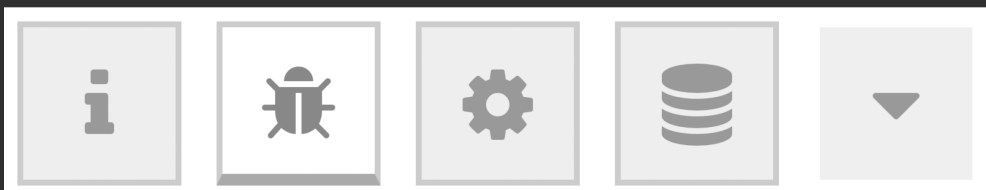
📌 名前 名前

注釈: 「指定した時間間隔、日時」と「指定した日時」はcronを使用します。詳細はノードの「情報」を確認してください。

はじめてのFlow -デバッグタブ-

デバッグメッセージは
「デバッグタブ」に表示されます

- サイドパネルの「デバッグタブ」をクリック



Timestamp	Node ID	Message
2018/11/12 11:06:31	node: e641101e.10fd	msg.payload : number 1541988391666
2018/11/12 11:06:36	node: e641101e.10fd	msg.payload : number 1541988396669
2018/11/12 11:06:41	node: e641101e.10fd	msg.payload : number 1541988401673
2018/11/12 11:06:46	node: e641101e.10fd	msg.payload : number 1541988406679
2018/11/12 11:06:51	node: e641101e.10fd	msg.payload : number 1541988411683

Lチ力 (LED点滅)

Lチカ -GPIOで使うNode-



rpi gpio

GPIOの入カピンの状態に応じて、0 または 1 の値を持つ msg.payload を生成します。

rpi gpio



GPIOの出カピンにデジタルモードまたはPWMモードで出力します。
デジタルモードの場合 0 または 1 の値を入力します。

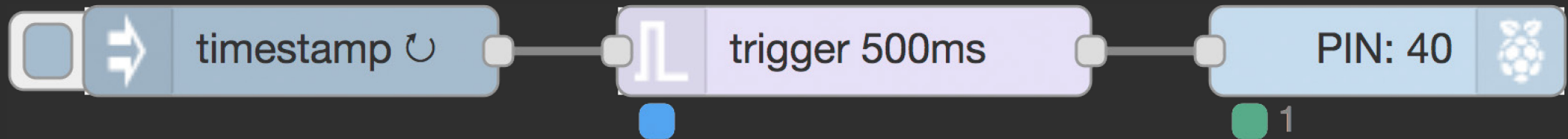
Lチカ -利用するNode-



メッセージを受信すると、別のメッセージを送信します。
遅延をかけて別のメッセージを送ることもできます。

Lチカ -Flow-

- 1秒ごとにメッセージを生成 (Injectノード)
- 1を送信し、0.5秒後に 0 を送信 (Triggerノード)
- これらをGPIOポートに出力 (GPIOノード)



Lチカ -Injectノードの設定-

1. Injectノードをダブルクリック
2. 繰り返し「指定した時間間隔」
3. 時間間隔「1秒」

2秒間隔でメッセージを送出

inject ノードを編集

削除 中止 完了

▼ プロパティ

✉️ ペイロード ▼ 日時

☰ トピック

Node-RED起動の 0.1 秒後、以下を行う

🔄 繰り返し 指定した時間間隔

時間間隔 1 秒

🏷️ 名前 名前

注釈: 「指定した時間間隔、日時」と「指定した日時」はcronを使用します。詳細はノードの「情報」を確認してください。

Lチカ -Triggerノードの設定-

1. Triggerノードをダブルクリック
2. 送信データ 「1」
3. 送信後の処理 「指定した時間待機」 「500ミリ秒」
4. 再送信データ 「数値」 「0」

メッセージが入ってきたら「1」を送出
0.5秒後に「0」を送出

trigger ノードを編集

削除 中止 完了

▼ プロパティ

送信データ

送信後の処理 指定した時間待機

500 ミリ秒

新たなメッセージを受け取った時に遅延を延長

再送信データ

初期化条件:

- msg.resetを設定
- msg.payloadが次の値

処理対象

名前

Lチカ -GPIOノードの設定-

1. GPIOノードをダブルクリック
2. LEDを接続した端子をクリック
3. 出力形式「デジタル出力」
4. 再送信データ「数値」「0」

rpi-gpio out ノードを編集

削除 中止 完了

▼ プロパティ

● 端子

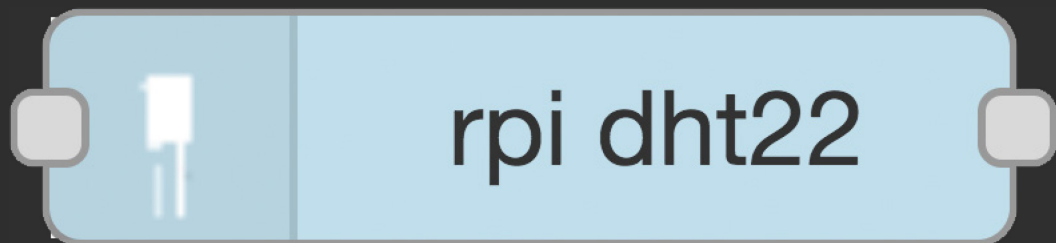
3.3V Power - 1	2 - 5V Power
SDA1 - GPIO02 - 3	4 - 5V Power
SCL1 - GPIO03 - 5	6 - Ground
GPIO04 - 7	8 - GPIO14 - TxD
Ground - 9	10 - GPIO15 - RxD
GPIO17 - 11	12 - GPIO18
GPIO27 - 13	14 - Ground
GPIO22 - 15	16 - GPIO23
3.3V Power - 17	18 - GPIO24
MOSI - GPIO10 - 19	20 - Ground
MISO - GPIO09 - 21	22 - GPIO25
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

出力形式 デジタル出力

端子の状態を初期化

温湿度センサー

温湿度センサー -使うNode-

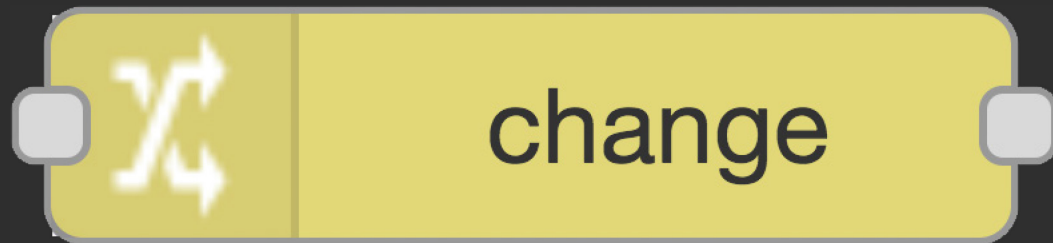


メッセージが入力されたタイミングでDHT11またはDHT22から温度と湿度を読み取ります。

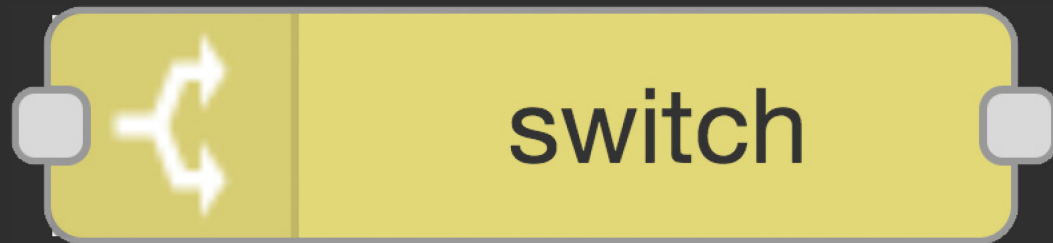
送出されるメッセージの `msg.payload` に温度、`msg.humidity` に湿度が含まれます

※ `node-dht-sensor` と `node-red-contrib-dht-sensor` のインストールが必要です

API -HTTPサーバで使うNode-



ルールに基づいてメッセージを変更したり、変数の操作を行います。
温度と湿度それぞれを文字列から数値に変換するのに利用します。



ルールに基づいて条件分岐を行います。

完成したフローを読み出し

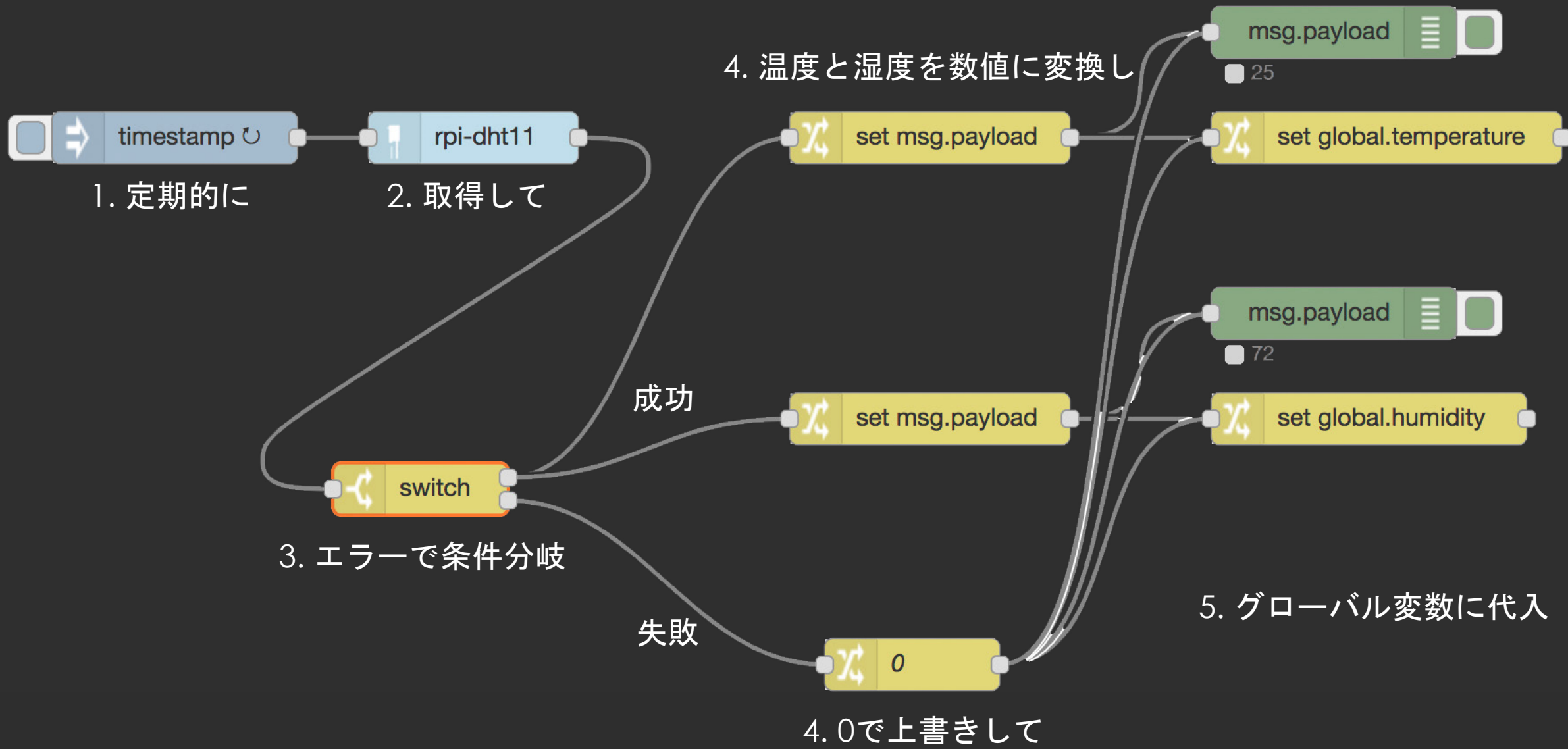
3分クッキング

1. メニューをクリック
2. 「読み込み」
3. 「ライブラリ」
4. 「DHT11」

DHT11ノードの番号を修正してください



温湿度センサー



1. 定期的に

2. 取得して

3. エラーで条件分岐

4. 温度と湿度を数値に変換し

成功

失敗

4. 0で上書きして

5. グローバル変数に代入

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. isValid



is true



→ 1



is false



→ 2



+ 追加

最初に合致した条件で終了



メッセージ列の補正

HTTPサーバ (API)

API -HTTPサーバで使うNode-



http



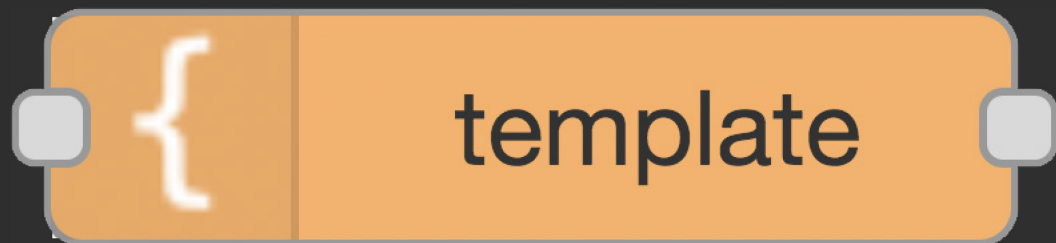
HTTPエンドポイントを作成し、HTTPリクエストを受け付けます。

http response

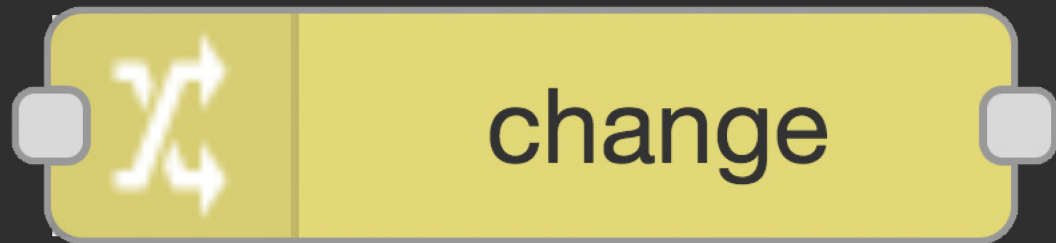


HTTPノードで受け付けたリクエストに対し、レスポンスを返します。

API - レスポンスの作成に使うノード-



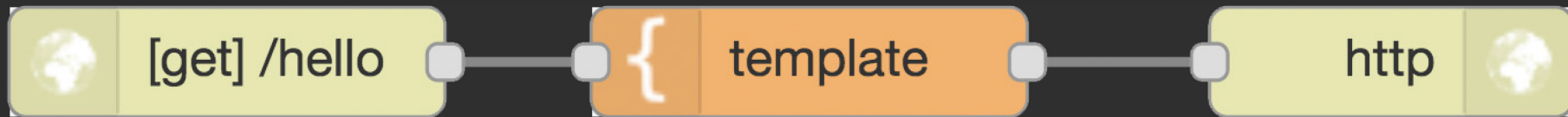
テンプレートに基づいて文字列を生成します。



ルールに基づいてメッセージを変更したり、変数の操作を行います。

Hello World!

- HTTPリクエストを /hello エンドポイントで受け付ける
- 文字列を含むJSONを生成する
- HTTPレスポンスとして返す



Hello World! -HTTP Inノードの設定-

1. HTTP inノードをダブルクリック
2. メソッド「GET」
3. URL「/hello」

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

Hello World! -Templateノードの設定-

1. Templateノードをダブルクリック
2. 設定先「msg.payload」
3. 形式「Mustacheテンプレート」
4. テンプレート構文「JSON」
5. テンプレート内容を入力
6. 出力形式「JSON」

template ノードを編集

削除 中止 完了

▼ プロパティ

📌 名前
名前

🔗 設定先

</> 形式

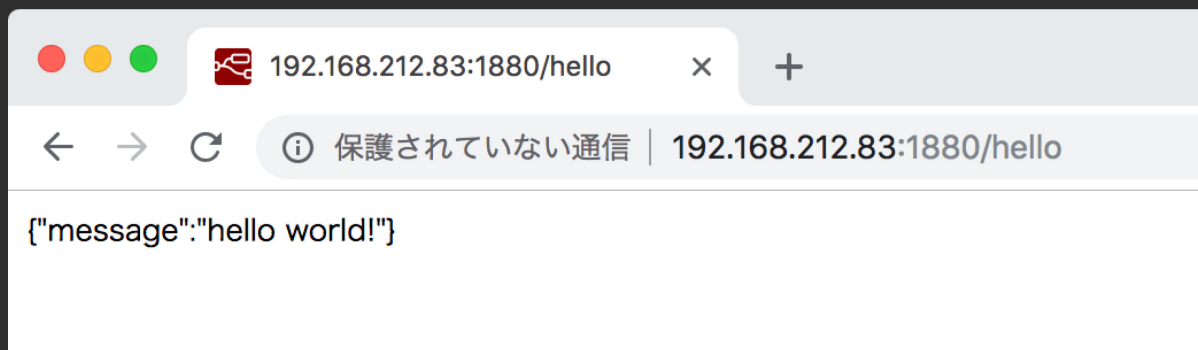
📄 テンプレート 構文:

```
1 {  
2   "message": "hello world!"  
3 }
```

→ 出力形式

Hello World! -動作確認-

- ブラウザから作成したエンドポイントにアクセスする
 - `http://(raspberrypiのIPアドレス):1880/hello`
 - `http://127.0.0.1:1880/hello`



完成したフローを読み出し

3分クッキング

1. メニューをクリック
2. 「読み込み」
3. 「ライブラリ」
4. 「API」



完成したフローを読み出し

3分クッキング

1. メニューをクリック
2. 「読み込み」
3. 「ライブラリ」
4. 「GPIO」

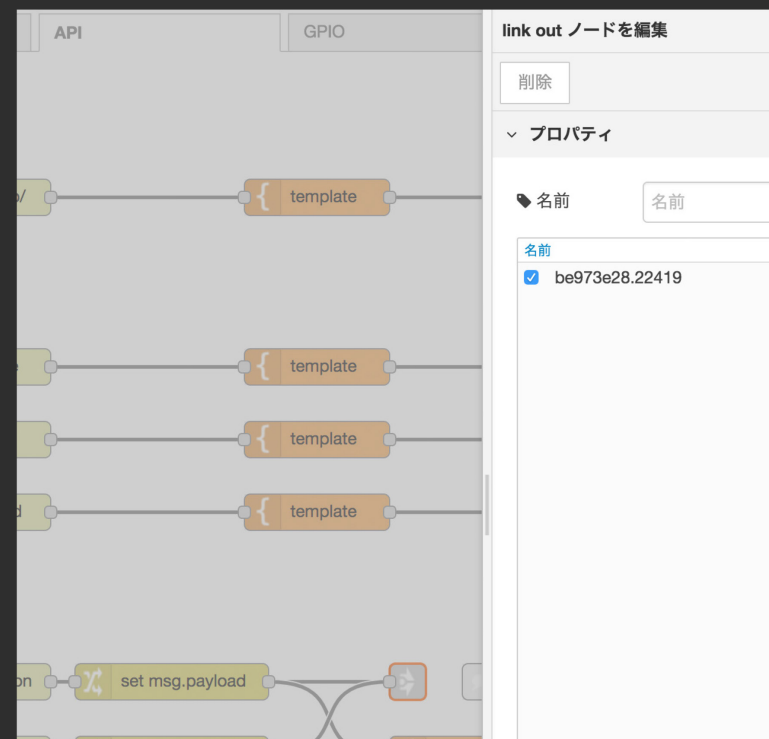
GPIOノードの番号を修正してください



フロー間の接続

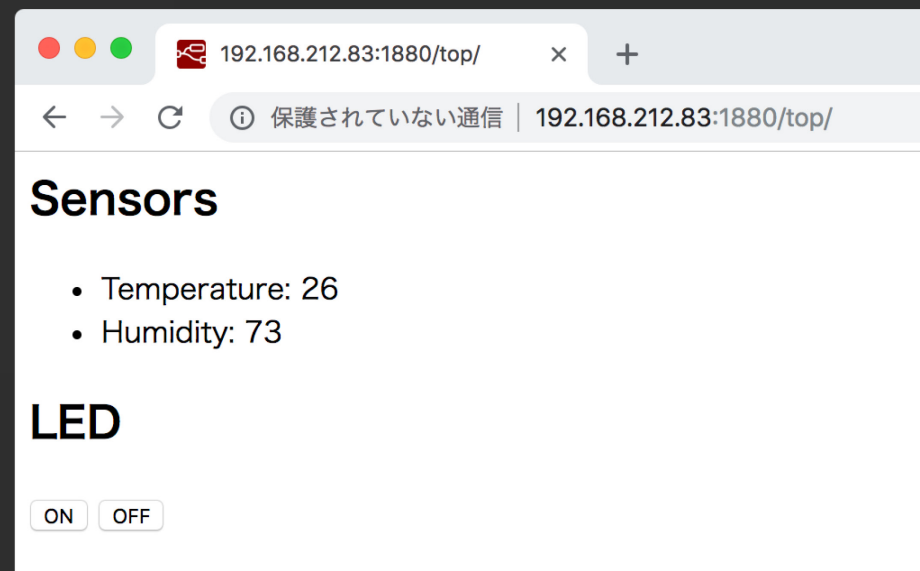
- APIフローの「link out」ノードをダブルクリック
- 表示された「link in」ノードにチェックをつける
- 「完了」をクリック

「API」フローと「GPIO」フローが接続される



API - 動作確認 -

- ブラウザからHTMLを返すエンドポイントにアクセスする
 - `http://(raspberrypiのIPアドレス):1880/top/`
 - `http://127.0.0.1:1880/top/`
- 定期的にセンサーの値がAjaxで更新される
- ボタンを押すとLEDが操作できる



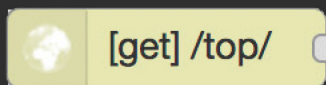
グローバル変数初期化



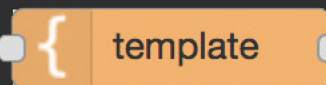
1. 起動時に

2. グローバル変数を初期化する

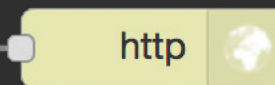
フロントエンド



1. HTTPリクエストを受けて

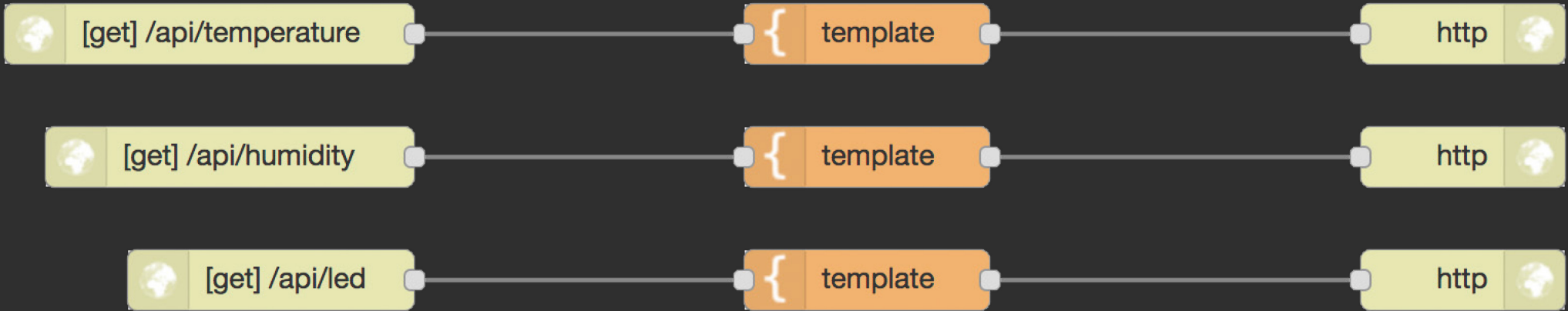


2. HTMLを生成し



3. レスポンスとして返す

GETリクエスト

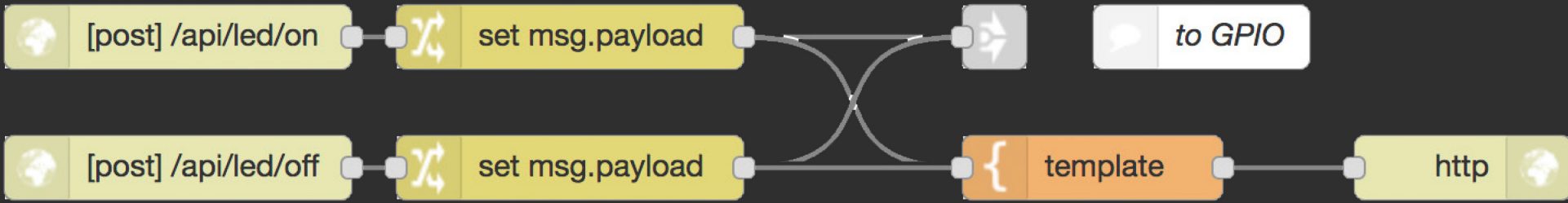


1. HTTPリクエストを受けて

2. 変数からJSONを生成し

3. レスポンスとして返す

POSTリクエスト



3. GPIO操作に渡す

1. HTTPリクエストを受けて

2. Bool値を生成し

3. JSONを生成

4. レスポンスとして返す

LED

from API



msg.payload

set global.led

PIN: 38

1. APIからのBool値

2. グローバル変数へ

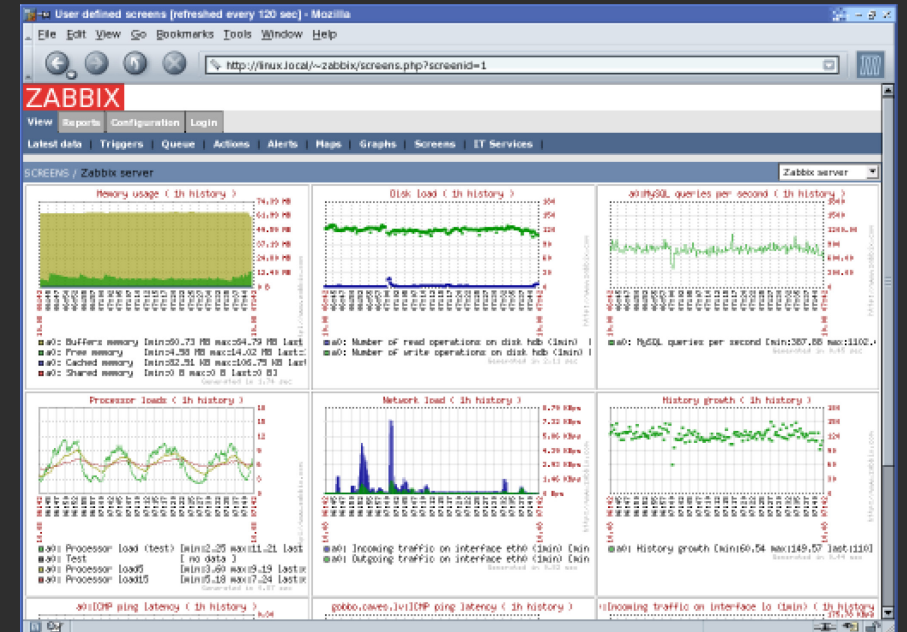
2. GPIOへ

Zabbixへのメトリクス送信

Zabbixとは

サーバー、ネットワーク、アプリケーションを監視するためのソフトウェアです。Zabbixは主に以下の3つの機能を有しています。

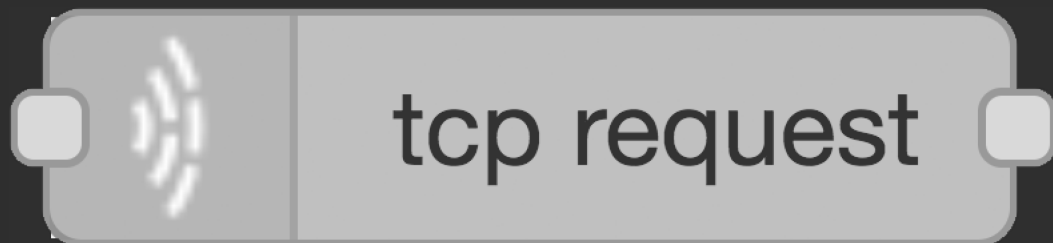
- サーバやネットワークに接続されたデバイスを監視する監視機能
- 収集したデータをもとにグラフ化、ネットワークマップの作成を行うグラフィカル表示機能
- 収集したデータに閾値を設定し、障害/復旧時に管理者に通知を行う障害検知/通知機能



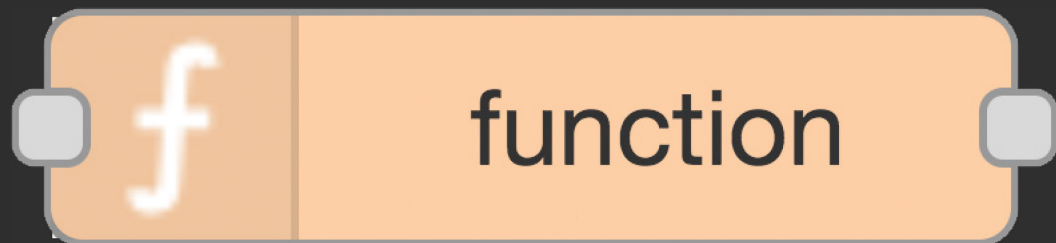
やること

本来はサーバにZabbix Agentをインストールするが
今回はNode-REDからZabbix Serverに温湿度をメトリクスとして送信します

Zabbixへの送信 -使うNode-



TCPサーバに接続し、入力されたmsg.payloadを送信。
応答をメッセージとして送出します。



任意のJavaScriptのコードを実行します。

完成したフローを読み出し

3分クッキング

1. メニューをクリック
2. 「読み込み」
3. 「ライブラリ」
4. 「Zabbix」



Zabbixへの送信 -ホスト名の設定-

1. Injectノードをダブルクリック
2. ペイロードに各々のホスト名を入力

inject ノードを編集

削除 中止 完了

▼ プロパティ

✉ ペイロード

☰ トピック

Node-RED起動の 秒後、以下を行う

🔄 繰り返し

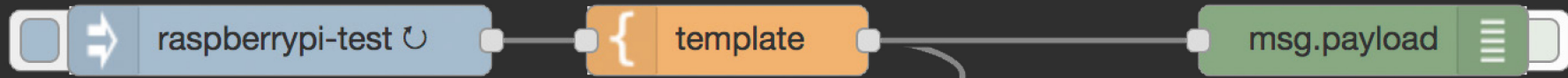
時間間隔

📌 名前

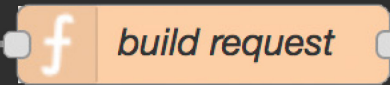
注釈: 「指定した時間間隔、日時」と「指定した日時」はcronを使用します。詳細はノードの「情報」を確認してください。

1. 定期的に

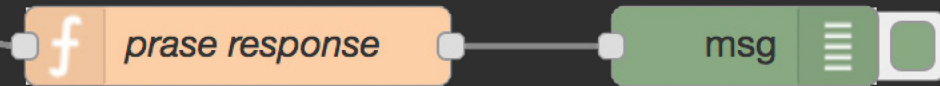
2. 変数からリクエストを生成



3. Zabbix Agent Protocol へ変換



4. Zabbix Server へ送信



5. 応答をパース

Zabbixへの送信 - 動作確認 -

1. 「デバッグ」タブを表示
2. “processed :2” の応答を確認

```
デバッグ
▼ 全てのフロー
2018/11/15 1:05:50 node: e3b2fd5.5d899
msg : Object
▶{ topic: "", payload: "processed: 2; failed: 0; total...", _msgid: "19a955d7.c9cdea", response: "success" }
2018/11/15 1:05:55 node: e3b2fd5.5d899
msg : Object
▶{ topic: "", payload: "processed: 2; failed: 0; total...", _msgid: "ac94719.1643e9", response: "success" }
2018/11/15 1:06:00 node: e3b2fd5.5d899
msg : Object
▶{ topic: "", payload: "processed: 2; failed: 0; total...", _msgid: "b8b6b5b2.01bf78", response: "success" }
2018/11/15 1:06:05 node: e3b2fd5.5d899
msg : Object
▶{ topic: "", payload: "processed: 2; failed: 0; total...", _msgid: "20ff43b6.20962c", response: "success" }
2018/11/15 1:06:10 node: e3b2fd5.5d899
msg : Object
▶{ topic: "", payload: "processed: 2; failed: 0; total...", _msgid: "a87daef5.22cf9", response: "success" }
```


Zabbixへの送信 - 動作確認 -

1. <http://zabbix.iw.ishikari-dc.jp/>
2. ゲストとしてログイン
3. 最新データから自ホストの値を確認する

The screenshot shows the 'Latest data' interface in Zabbix. It includes a search area with filters for Host groups (set to 'Raspberry Pi'), Hosts, and Application. There are also checkboxes for 'Show items without data' (checked) and 'Show details'. Below the filters are 'Apply' and 'Reset' buttons. The main data table has columns for Host, Name, Last check, Last value, Change, and Graph. The data is filtered to show items for the 'raspberrypi-test' host group.

<input type="checkbox"/> Host	Name ▲	Last check	Last value	Change	
<input type="checkbox"/>	raspberrypi-test	- other - (2 Items)			
<input type="checkbox"/>	Humidity	2018-11-15 01:18:09	63	+1	Graph
<input type="checkbox"/>	Temperature	2018-11-15 01:18:09	25		Graph

注意事項

注意事項

- 管理画面へのアクセスを制限してください
 - 公開するエンドポイントと管理画面を別ポートにする
 - パスワードを設定する
 - etc.
- SDカードはできるだけRead Onlyにしてください
 - aufsやoverlayfsなどを利用し、Read Onlyにする
- しっかりとした電源を供給してください
 - microUSBではなくピンヘッダから供給する
- Linuxに対する一般的なセキュリティ対策をしてください