

# 製造業ITマイスター指導者育成プログラム 研修テキスト 実習用教材(第3日) 高度IT実装技術の習得2 (IoTセンサー実装実習)



# 製造業ITマイスター研修教材一覧



日	テーマ		教材
1	製造業IT導入ワークショップ	午前	IoTとシステムの基礎
		午後	製造業IT導入ワークショップ
2	高度IT実装技術の習得 1	午前	IoTによるシステム開発入門
		午後	高度IT実装技術の習得 1 (ラズパイ+見える化実習)
3	高度IT実装技術の習得 2	午前	IoTによる生産管理入門
		午後	高度IT実装技術の習得 2 (IoTセンサー実装実習)
4	システム構築技術の習得 1	午前	IoTによる在庫管理入門
		午後	システム構築技術の習得 1 (業務システムの基本パターン)
5	システム構築技術の習得 2	午前	IoTによるデータ分析入門
		午後	システム構築技術の習得 2 (データ分析)
6	PBL 1 (事例企業調査)	午前	事例企業調査
		午前	事例企業の課題モデル化実習
7	PBL 2 (課題の設定と解決策の提案)	午後	システム構築の実際
		午後	システム構築実習 (1) 課題の設定と解決策の提案
8	高度IT実装技術の適用	午前	IT経営の実践方法
		午後	システム構築実習 (2) 高度IT実装技術の適用
9	システム構築技術の適用	午前	情報システムセキュリティ基礎 知財とオープン&クローズ戦略
		午後	システム構築実習 (3) システム構築技術の適用
10	筆記試験および成果発表会	午前	個人と組織の発展に繋がるキャリアデザイン講座 (筆記試験)
		午後	(成果発表会)

# ■ 前半5日間の進め方



## 午後の実習

- 1日目 実習のための環境設定 → 課題発見ワークショップ
- 2日目 デバイス信号のイン/アウト → センサデータの見える化
- 3日目 メールとWebサーバ利活用 → 人感センサとカメラの利用
- 4日目 業務システムの基本パターン → バーコードリーダとNFC
- 5日目 データ分析続き → 工程進捗管理ボード

## 1. メールとWebサーバ利活用

### ① Node-REDの基本

- ② 温度が異常だとブザーを鳴らす
- ③ 温度と湿度の履歴をCSVファイルに書出す
- ④ 定型メールの送信

## 2. 人感センサとカメラの利用

- ① 赤外線センサーに反応してLEDを点灯させる
- ② カメラで画像を撮影し保存する
- ③ Webサーバーを使う
- ④ 監視カメラを作る

# Node-REDエディタ画面





# ワークスペース

# フロー



フローは、ワークスペースの1ページに相当するひと塊を指します。

フローの追加

新しいフローを追加するには右上の+をクリックします。



フローの削除

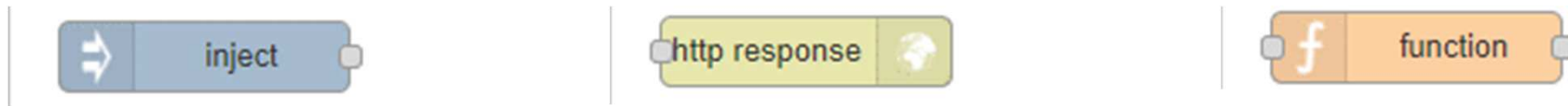
フローのタブをダブルクリックしたら表示されるフロープロパティで Delete ボタンをクリックします。

# ノード

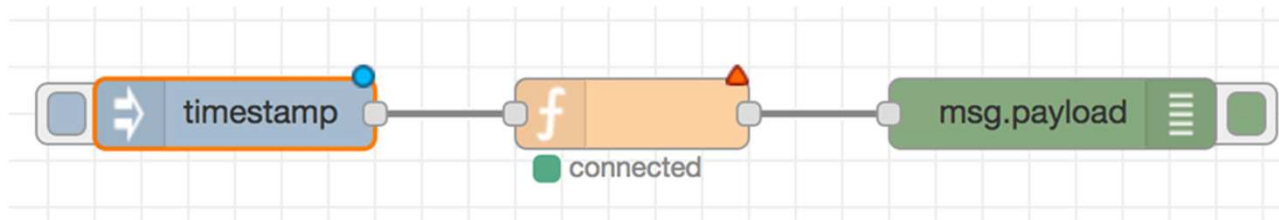


ノードは、アプリケーションを組み立てるためのソフトウェア部品です。  
ノードのポートを介してワイヤーで結んで、アプリケーションを組み立てます。

## ノードの例



ノードにデプロイ(実行できるようにする)されていない変更箇所があるとノードの上に青色の○が表示されます。  
ノードの設定にエラーがあった場合、赤い△が表示されます。





# ノードでjavascriptを実行する



ノードの上でダブルクリックするとノードの設定を編集することができます。  
Javascriptの処理プログラムを自前で記述することが可能です。

function ノードを編集

削除 中止 完了

プロパティ

名前

名前

コード

```
1  
2 return msg;
```

出力数 1

コードの記述方法はノードの「情報」を参照してください。

設定

function ノードを編集

削除 中止 完了

プロパティ

設定

入力

1. なし

出力

1. なし

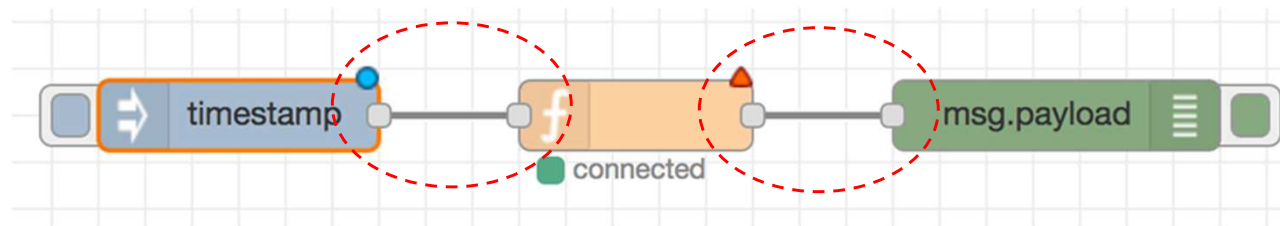
アイコン

f

# ワイヤー



ノード同士をつなぎ合わせるのがワイヤーです。  
入力から出力までをつなぎ合わせて、必要なアプリケーションを組み立てます。





# パレット

# パレット



パレットには、インストール済みで利用可能なすべてのノードが表示されています。ノードはカテゴリにまとめられています。



# パレットマネージャ



パレットマネージャは、パレットに新しいノードをインストールするために利用できます。

デプロイボタンの右側の三本線アイコンをクリックし、パレットの管理をクリックします。

パレットマネージャダイアログが開きます。現在のノードタブでは現在インストールされているノードが表示されます。

ノードの追加タブでは追加インストールしたいノードをキーワードで検索し、[ノードを追加]ボタンをクリックすると、インストールすることができます。





# サイドバー

# サイドバー



サイドバーには、エディタ内の便利ツールを提供するパネルが含まれています。

The screenshot shows a software interface with a sidebar on the right. The sidebar is titled "サイドバー" (Sidebar) in red. It contains a table with the following information:

情報	
フロー	"eede5d14.369428"
名前	Flow 1
状態	有効

Below the table is a section titled "フローの詳細" (Flow Details) which currently shows "なし" (None). At the bottom of the sidebar, there is a text instruction: "ノードを選択し、enter を押すとプロパティ設定画面が表示されます。" (Select a node and press enter to display the property setting screen.)

サイドバーの初期状態またはタブの横の“i”と書かれたアイコンボタンをクリックすると表示されます。

ノードを選択するとノードについてプロパティの概要、ノードのヘルプテキストを表示します。



i 情報	
▼ 情報	
フロー	"eede5d14.369428"
名前	Flow 1
状態	有効

▼ フローの詳細

なし

ノードを選択し、  を押すとプロパティ設定画面が表示されます。



# デバッグ



サイドバータブ右横の“虫の絵”が書かれたアイコンボタンをクリックすると表示されます。

右図の「全てのフロー」と書かれているボタンをクリックすると3つの選択肢を選ぶことができます。

- ・全てのフロー  
全てのメッセージを表示します。
- ・選択したノード  
全てのDebugノードの一覧から特定のノードを選択できます。
- ・現在のフロー  
ワークスペースの現在のフローのノードからのメッセージを表示します。





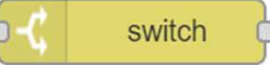






# Coreノート

# Coreノード

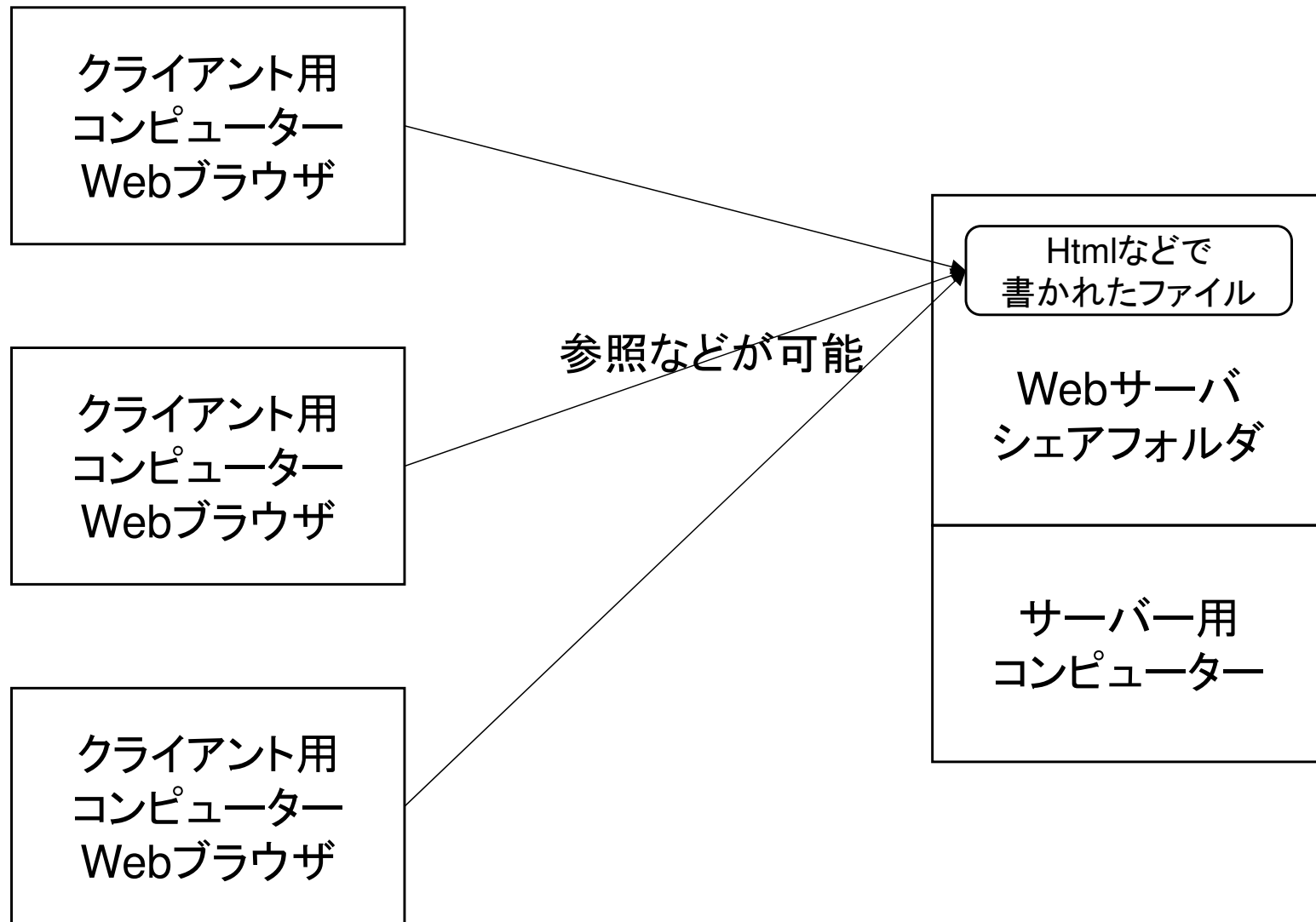


-  inject
  - インジェクト  
エディタ内でこのノードをクリックすることで、手動でフローを始動できる
-  debug
  - デバッグ  
エディタ内サイドバーにメッセージを表示することができる
-  function
  - ファンクション  
受け渡されるメッセージに対するJavaScriptコードの実行を可能にする
-  change
  - チェンジ  
受け渡されたメッセージプロパティを変更したり、コンテキストプロパティを設定したりすることができる  
値の代入、値の置換、値の移動、値の削除など
-  switch
  - スイッチ  
受け渡されたメッセージに対して、一連のルールで評価し、メッセージを異なるフローにルーティングすることができる  
Value rule: 設定されたプロパティに対して評価をおこなう  
Sequence rule: Splitノードによって生成されるようなメッセージシーケンスを利用できる  
JSONata式: メッセージ全体に対して評価を行い、trueの値を返した場合に一致したとみなすことができる  
その他: どのルールのどれにも一致しなかった場合、一致するように利用できる
-  template
  - テンプレート  
メッセージプロパティをテンプレートに設定することでテキストを生成することができる



# 簡易Webサーバーを作る

# Webサーバーとは



# ノードの配置



パレットの入力からhttpノードをワークスペースにドラッグ&ドロップ

Node-RED

q ノードを検索

フロー1

入力

- inject
- catch
- status
- link
- mqtt
- http
- websocket
- tcp
- udp
- Watson IoT

出力

- debug
- link
- mqtt
- http response
- websocket
- tcp
- udp
- Watson IoT

情報

情報

ノード "fd85735c-7a1f5"

型 http in

さらに表示

ノードのヘルプ

HTTPエンドポイントを作成し、Webサービスを構成します。

出力

payload

GETリクエストの場合、クエリパラメータからなるオブジェクト。それ以外の場合、HTTPリクエストの本体を指します。

req オブジェクト

HTTPリクエストオブジェクト。オブジェクトはリクエストの情報に関する複数のプロパティを含みます。

- body - リクエスト本体。形式はリクエストに依存します
- headers - HTTPリクエストヘッダを含むオブジェクト
- query - クエリパラメータを含むオブジェクト
- params - ルーティングパラメータを含むオブジェクト
- cookies - リクエストのクッキーを含むオブジェクト
- files - POSTリクエストでファイルのアップロードが設定で有効化されている場合、アップロード対象のフ

ノードを選択し、`enter`を押すとプロパティ設定画面が表示されます。

# http入力ノードプロパティ入力



4. 完了をクリック

2. プロパティが開く

1. ダブルクリック

Node-RED interface showing the configuration of an http node. The configuration panel is open, displaying the following properties:

- メソッド: GET
- URL: /function
- 名前: 名前

Buttons: 削除, 中止, 完了

3. プロパティを記入  
メソッド: GET  
URL: /function

# もう一つのノードも記入



The screenshot shows the Node-RED web interface. On the left, the 'Inputs' palette contains various nodes, with an 'http' node highlighted. A blue arrow points from the text '1. ダブルクリック' to this node. The main workspace shows a flow with two 'http' nodes. A second blue arrow points from the text '2. プロパティが開く' to the configuration panel on the right. The configuration panel, titled 'http in ノードを編集', has a '完了' button highlighted in red. A third blue arrow points from the text '3. プロパティを記入' to the 'メソッド' field, which is set to 'GET'. Below it, the 'URL' field is set to '/function'. A fourth blue arrow points from the text '4. 完了をクリック' to the '完了' button.

1. ダブルクリック

2. プロパティが開く

3. プロパティを記入  
メソッド: GET  
URL: /template

4. 完了をクリック



# 機能ノードの配置



The screenshot shows the Node-RED web interface. On the left, a sidebar lists various nodes under the '機能' (Function) category. Two nodes are highlighted with blue arrows: 'function' and 'template'. In the main workspace, a flow named 'フロー 1' contains two nodes: '[get] /function' and '[get] /template'. The 'function' node is connected to the 'template' node. The text 'functionノードを配置' (Configure function node) is positioned above the function node, and 'templateノードを配置' (Configure template node) is positioned below the template node.

# functionノードを編集



function ノードを編集

名前 (任意) function

関数のプログラムを記述

```
var m = "<html>";
m = m + "<body>";
m = m + "Hello NodeRED Function";
m = m + "</body>";
m = m + "</html>";
msg.payload = m;
return msg;
```

var m = "<html>";  
m = m + "<body>";  
m = m + "Hello NodeRED Function";  
m = m + "</body>";  
m = m + "</html>";  
msg.payload = m;  
return msg;

メッセージの送信  
フロー内の次ノードにメッセージを渡すため  
には、メッセージを返却する  
か `node.send(messages)` を呼び出します。

ノードの端子に複数の接続がある  
時、 を押しなが  
ら `click` し  
ド  
ラッグすることで、複数の接続をま  
とめて他のノードの端子へ移動でき

# templateノードを編集



名前(任意) template

名前 (任意) template

設定先 msg.payload

形式 Mustacheテンプレート

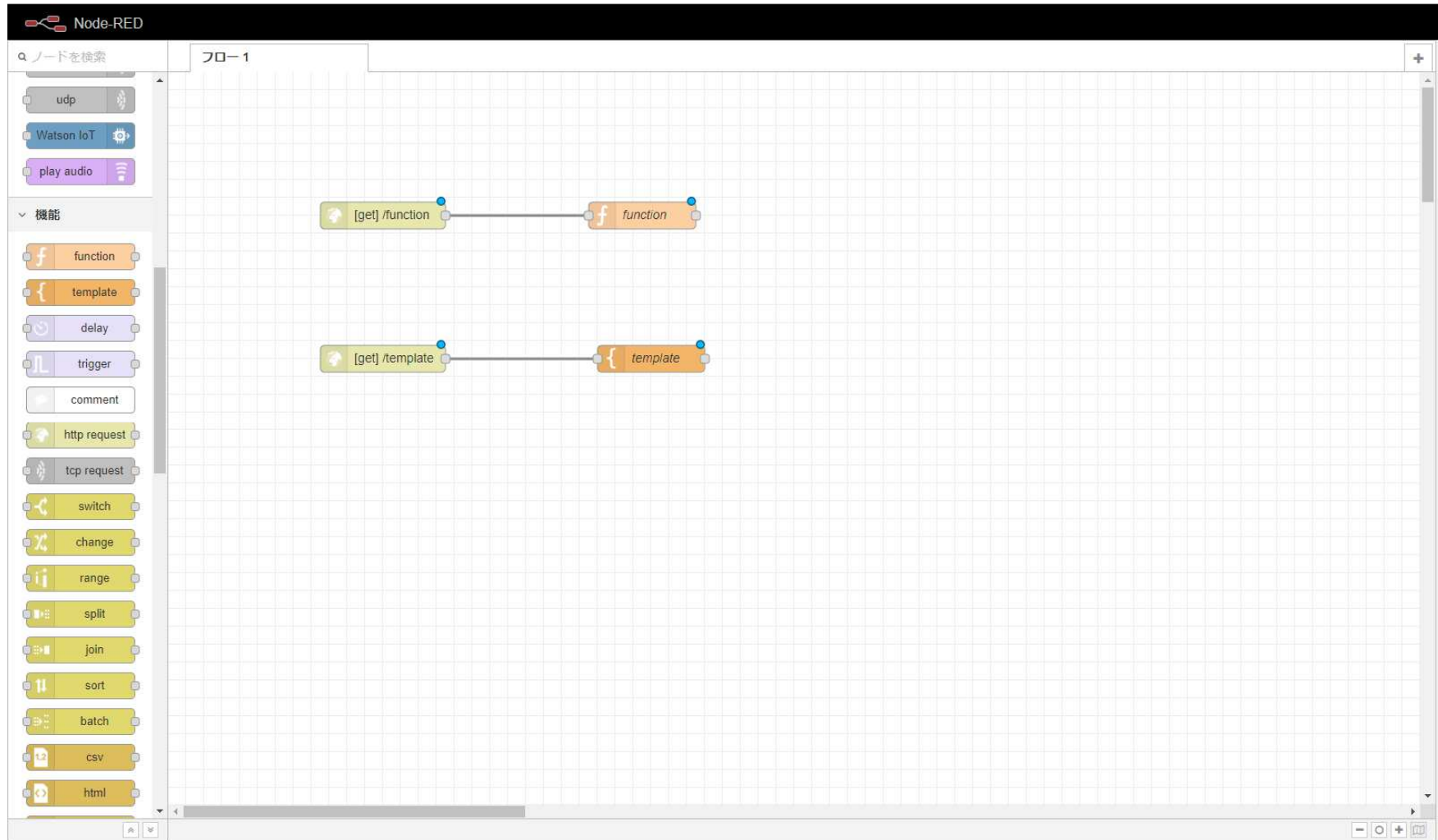
テンプレート

```
1- <html>
2-   <body>
3-     Hello NodeRED Template!
4-   </body>
5- </html>
```

HTMLを直接記述

```
<html>
<body>
Hello NodeRED Template
</body>
</html>
```

# 入力ノードと機能ノードをつなぐ



# 出力ノード「http response」を配置



デプロイボタンをクリック

機能ノードと出力ノードをつないでデプロイすれば完成。



Node-RED interface showing two flows. The top flow consists of a [get] /function node connected to a function node, which is connected to an http node. The bottom flow consists of a [get] /template node connected to a template node, which is connected to an http node. The left sidebar shows the '出力' (Output) section with 'http response' highlighted. The right sidebar shows the '情報' (Info) tab for the selected flow.

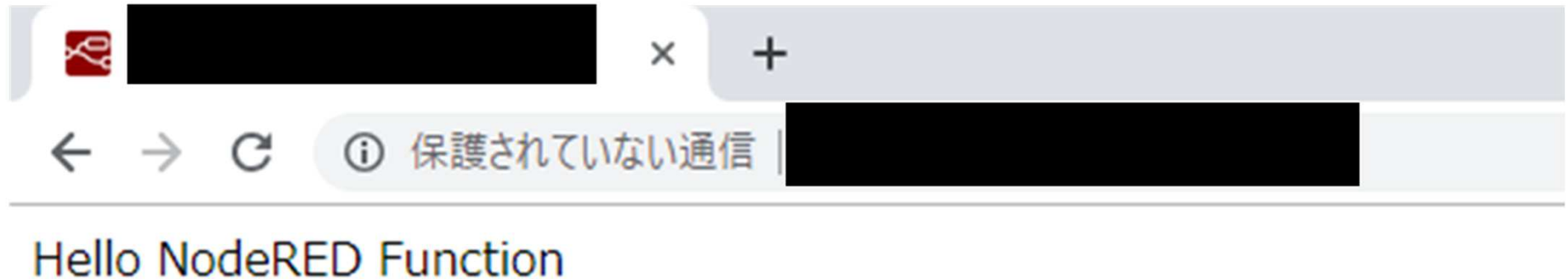
情報	
フロー	"1cbbd6ca.0abac9"
名前	フロー 1
状態	有効

ctrl-g | i | で「情報」タブを表示します。 ctrl-g | d | で「デバッグ」タブを表示します。

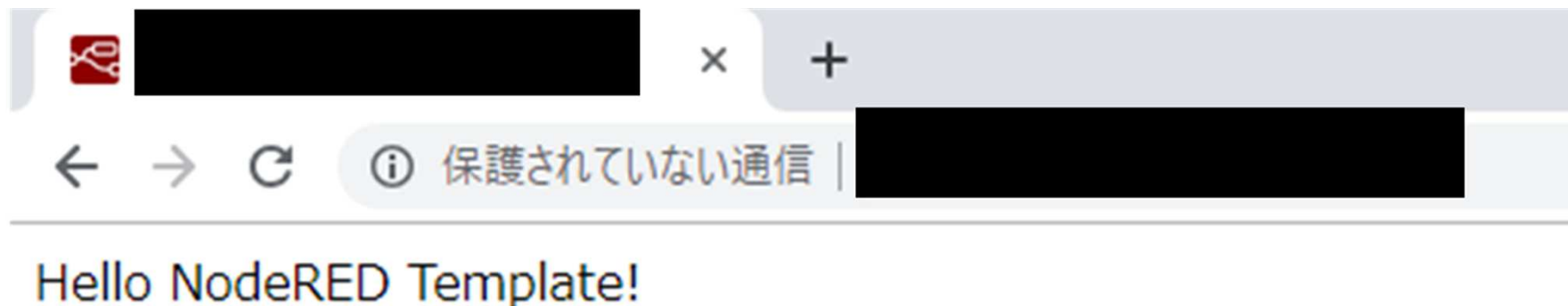
# PCからWebページを開く



/function の表示



/template の表示



Internet explore の場合は  
http://を必ず先頭に入れてください。

# functionとtemplateの違い



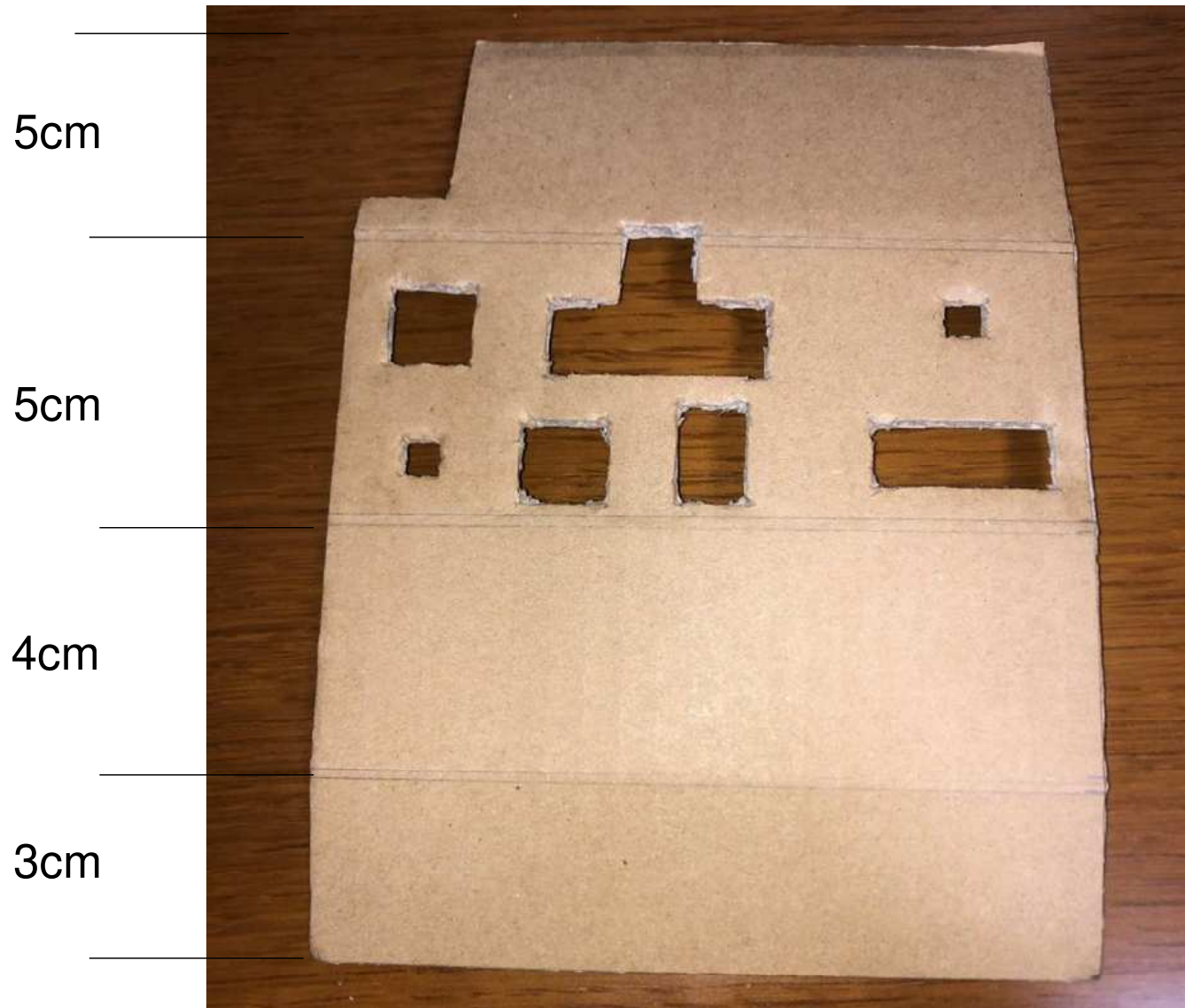
- function
  - javascriptのプログラムを動かすノード
  - 事例のプログラムはhtml文を作っている
- template
  - templateに書いた文をそのまま渡す



# センサーボックスを組み立てる



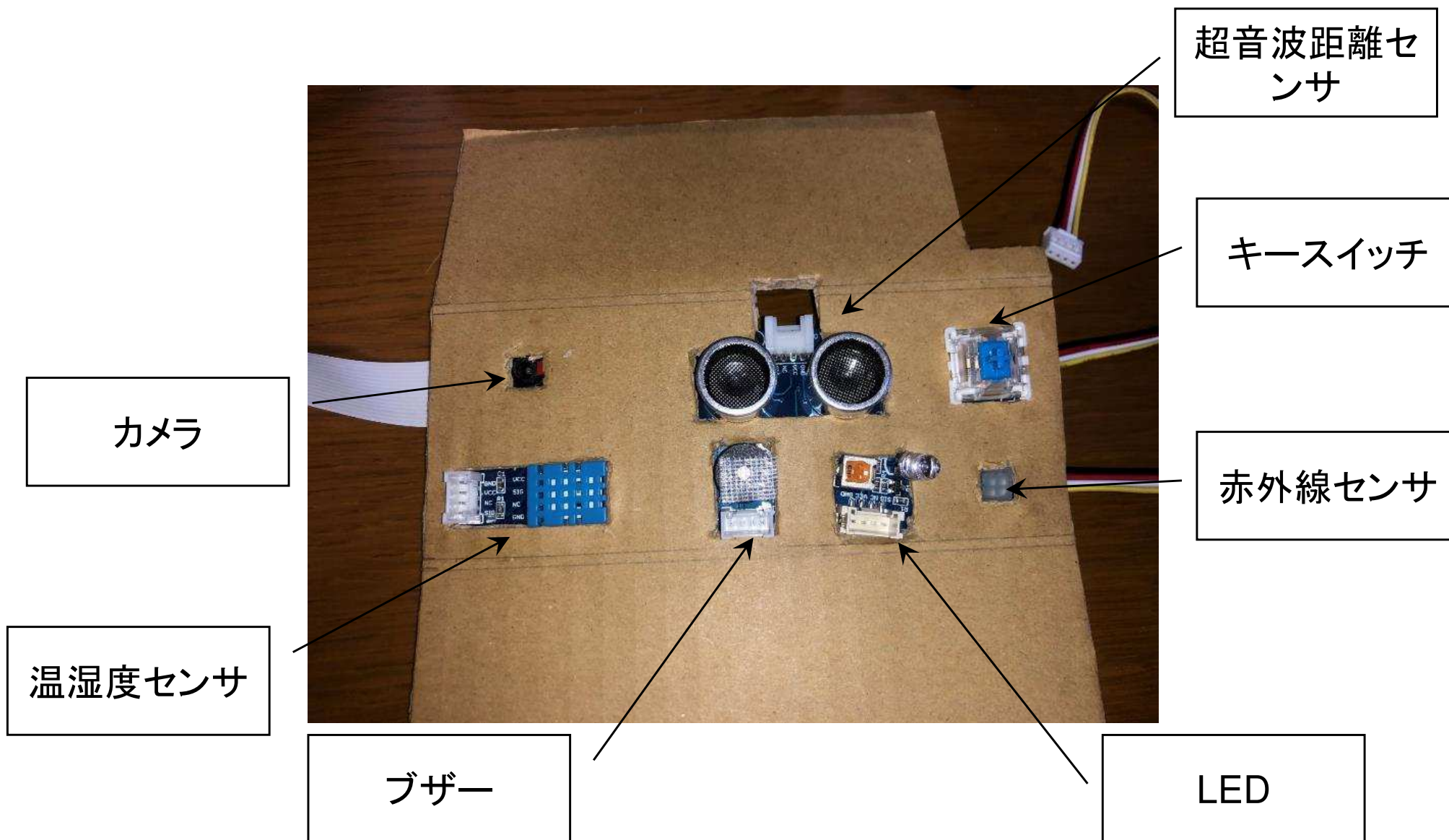
# 内側に丸まるように折り曲げる



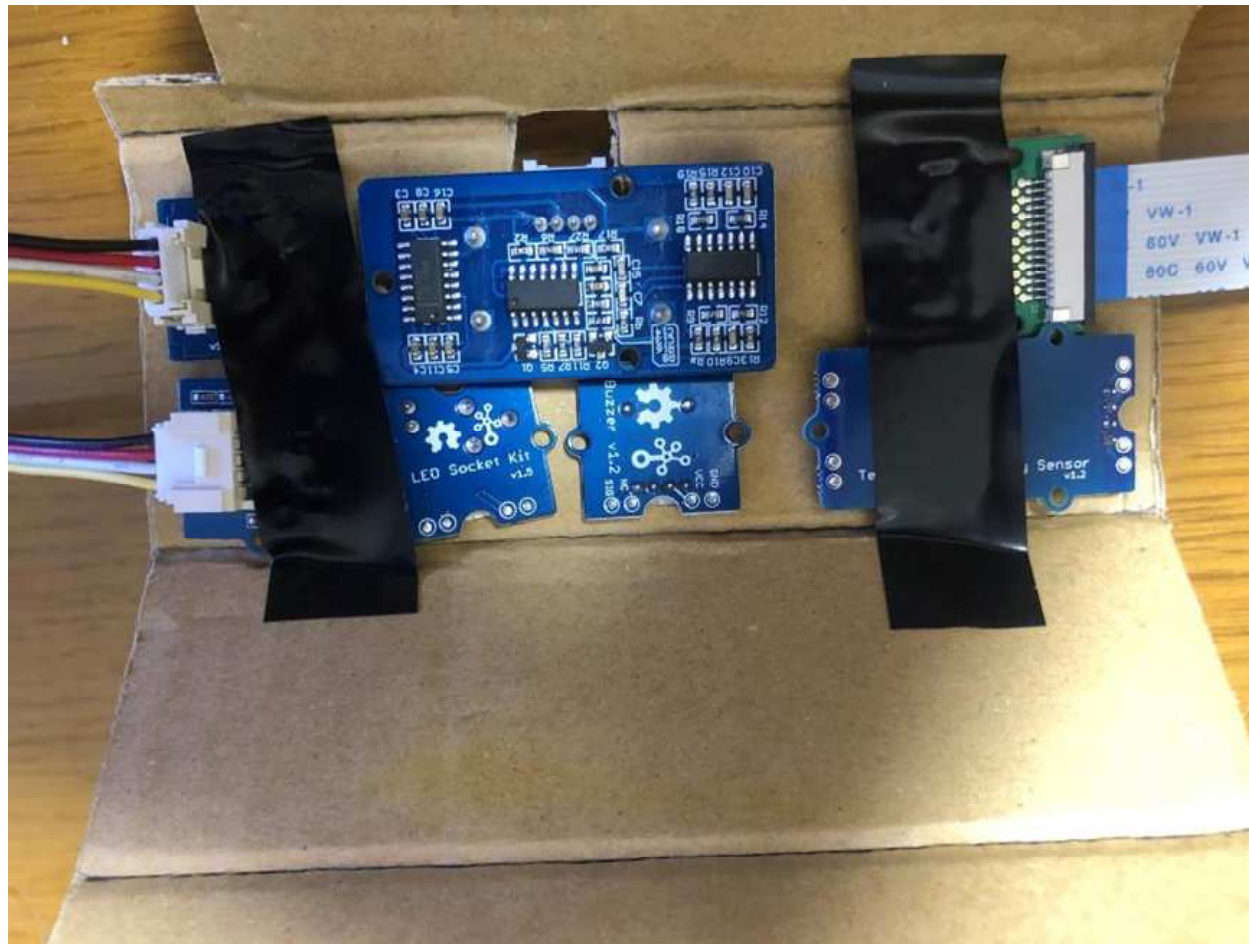
穴あき面

# 部品を穴に差し込む

丸まる内側から外側に向けて、穴あき面に部品を挿し込んでいきます。



# 部品をテープで固定する



両脇のセンサーやカメラをテープで固定します。



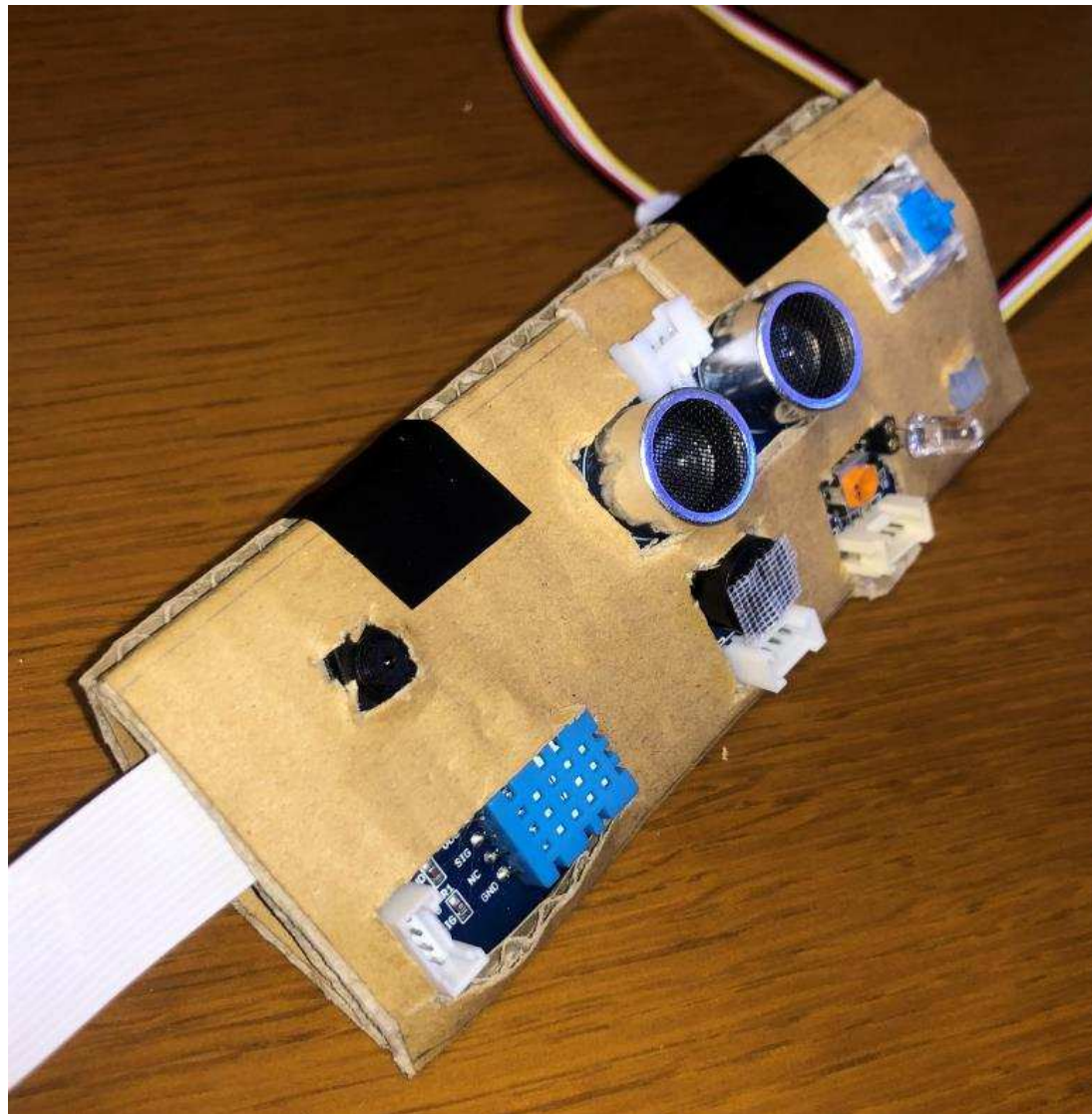
# 上面の受け側面を折り曲げて固定



ぎゅっと押し  
付けてテープ  
で固定します。

# センサーボックスの完成

横に倒した三角柱の様な形に折り曲げてテープで固定します。



# 接続するポート

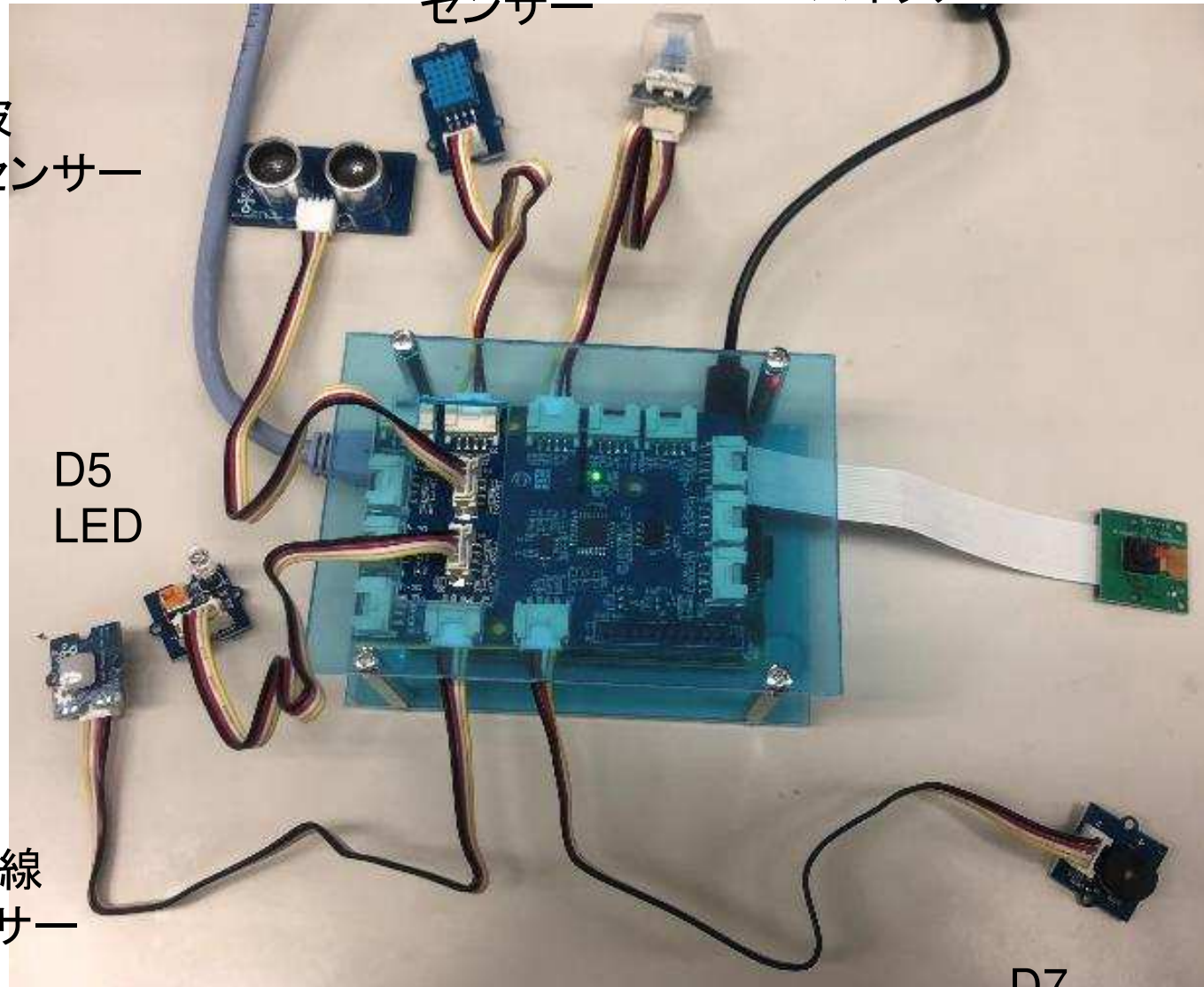
D6  
超音波  
距離センサー

D3  
温湿度  
センサー

D2  
押しボタン  
スイッチ

D5  
LED

D8  
赤外線  
センサー



D7  
ブザー



## 1. メールとWebサーバ利活用

① Node-REDの基本

② 温度が異常だとブザーを鳴らす

③ 温度と湿度の履歴をCSVファイルに書出す

④ 定型メールの送信

## 2. 人感センサとカメラの利用

① 赤外線センサーに反応してLEDを点灯させる

② カメラで画像を撮影し保存する

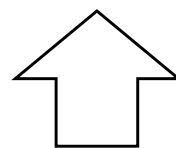
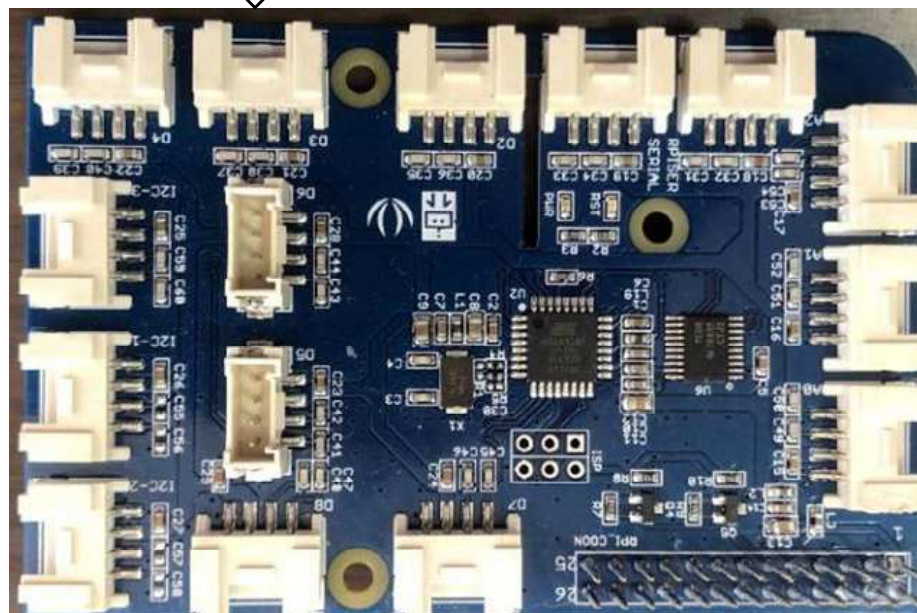
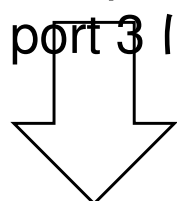
③ Webサーバーを使う

④ 監視カメラを作る

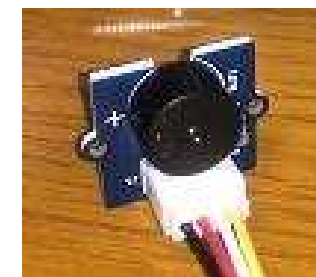
# Grove Temp&Humiの取付



Grove Temp&Humi  
Digital port 3 に接続する



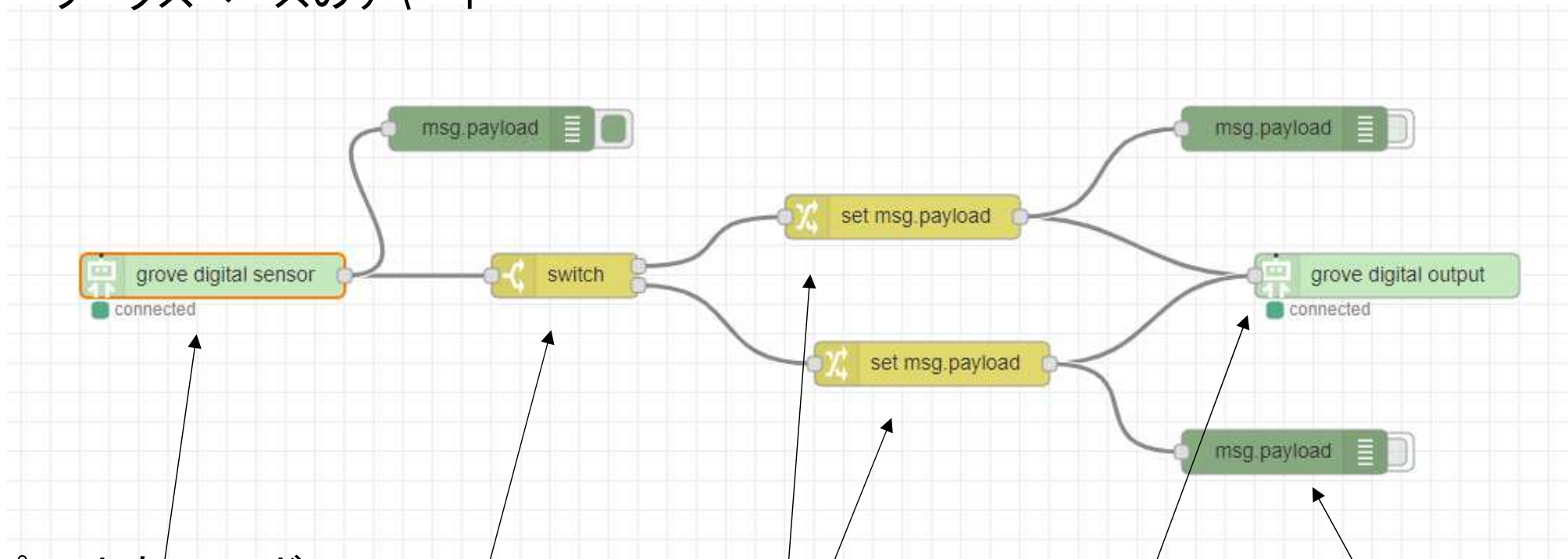
Grove Buzzer  
Digital port 7に  
接続する



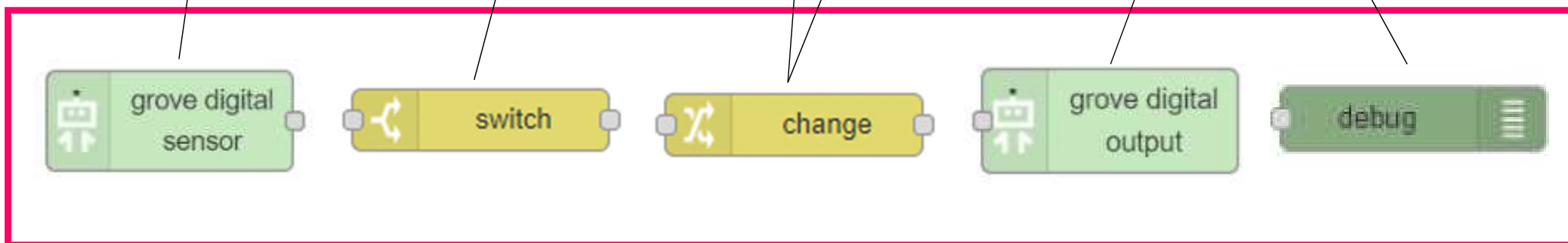


# フローを作ってみる

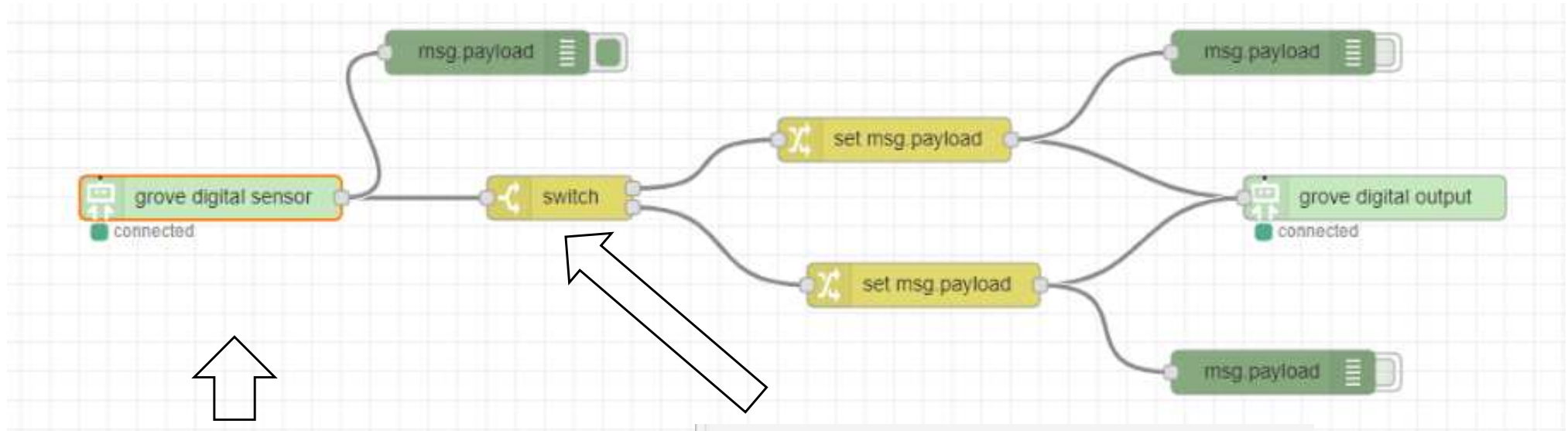
ワークスペースのチャート



パレット内のノード



# プロパティを設定する



grove digital sensor ノードを編集

削除 中止 完了

▼ プロパティ

Board: GrovePi

Sensor Type: Temperature / Humidity DHT1

Digital Pin: Digital 3

Interval: 1 Seconds

Name: Name

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.temperature

>= 0 40 → 1

その他 → 2

プロパティ  
msg.payload.temperature  
条件  
>= 数値:40 →1  
その他 →2

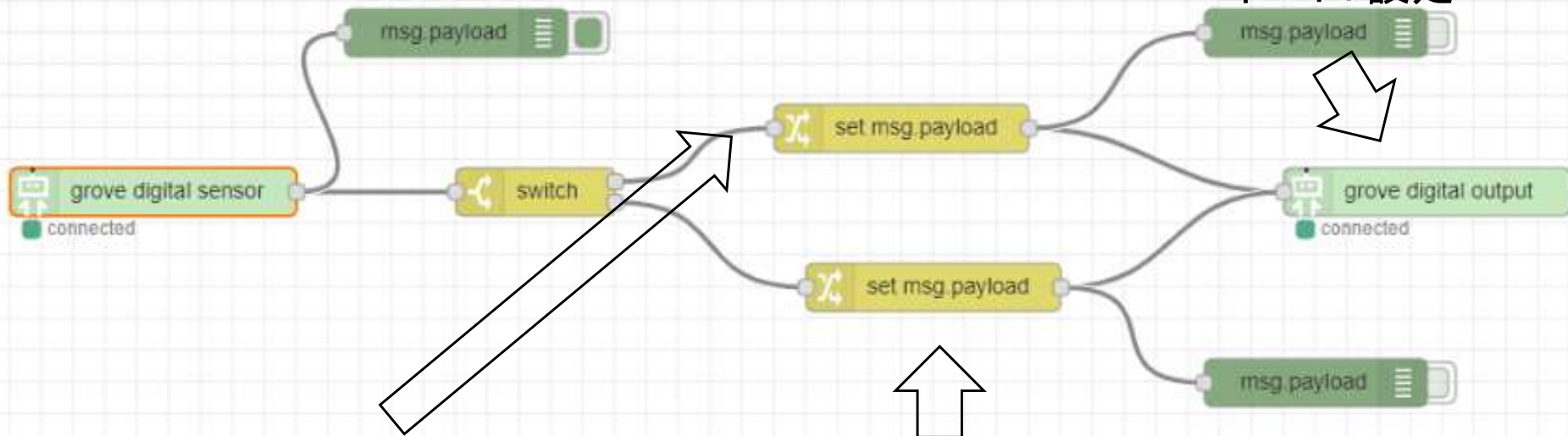
センサ選定しポート番号3選択

# プロパティを設定する

Board: GrovePi  
Digital Pin: Digital 7  
Name: Name



ポート7設定



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: true

ルール  
値の代入 msg.payload  
対象の値 真疑 true

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: false

ルール  
値の代入 msg.payload  
対象の値 真疑 false

# Node-RED+GrovePi使用での注意

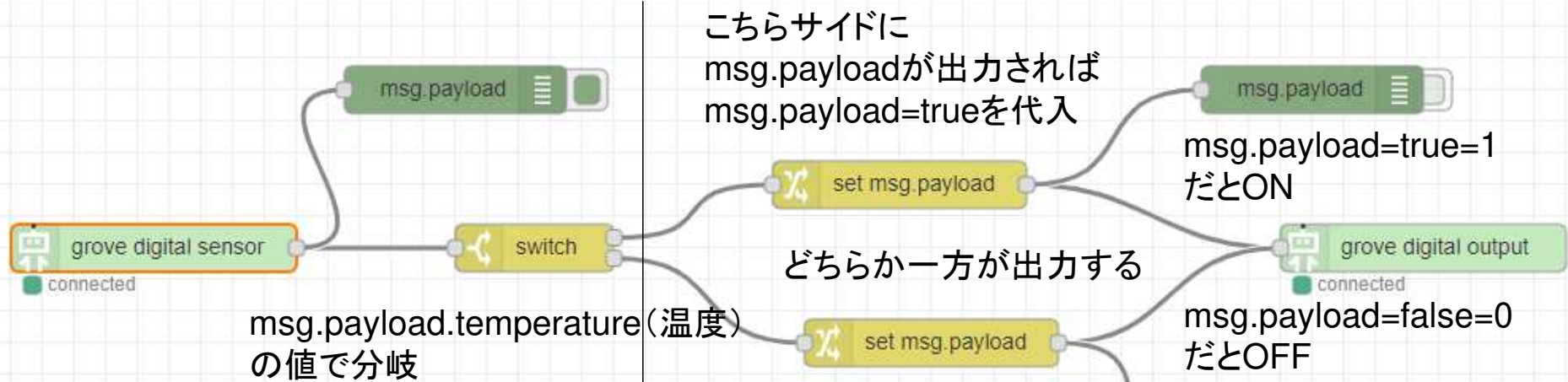
GrovePiのポートにセンサーを沢山付けた場合、動作異常を起こす可能性があります。

今回の実習で使ったセンサーを全て接続して、この操作を行ったところ、測定値が異常になり、動作異常をおこしました。

必ず起きるわけではありませんが、実際に使用する場合はその様な異常がない様にしておく必要があります。  
以下の様な対策を実施するよう心掛けてください。

- ・センサーを接続しすぎない
- ・測定値が一瞬異常になっても異常反応させない
- ・測定異常値の形式が分かった場合、その様な形式をフィルタリングする

# ノード間のデータの流れ

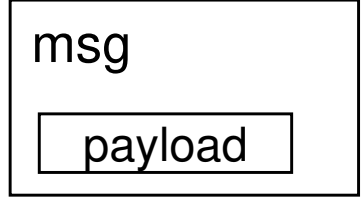
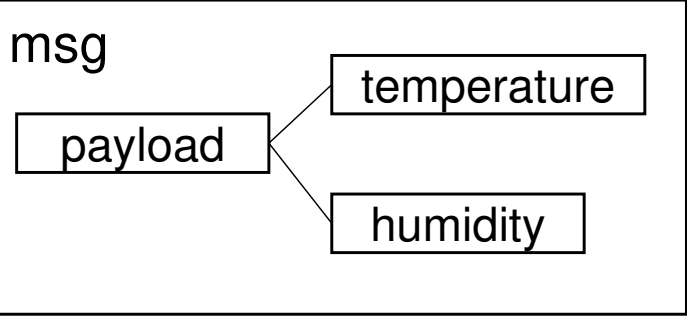


基本のデータの塊  
`msg.payload`  
 温湿度センサーの出力データ  
`msg.payload.temperature` (温度)  
`msg.payload.humidity` (湿度)

こちらサイドに  
`msg.payload`が出力されれば  
`msg.payload=true`を代入

どちらか一方が出力する

こちらサイドに  
`msg.payload`が出力されれば  
`msg.payload=false`を代入



## 1. メールとWebサーバ利活用

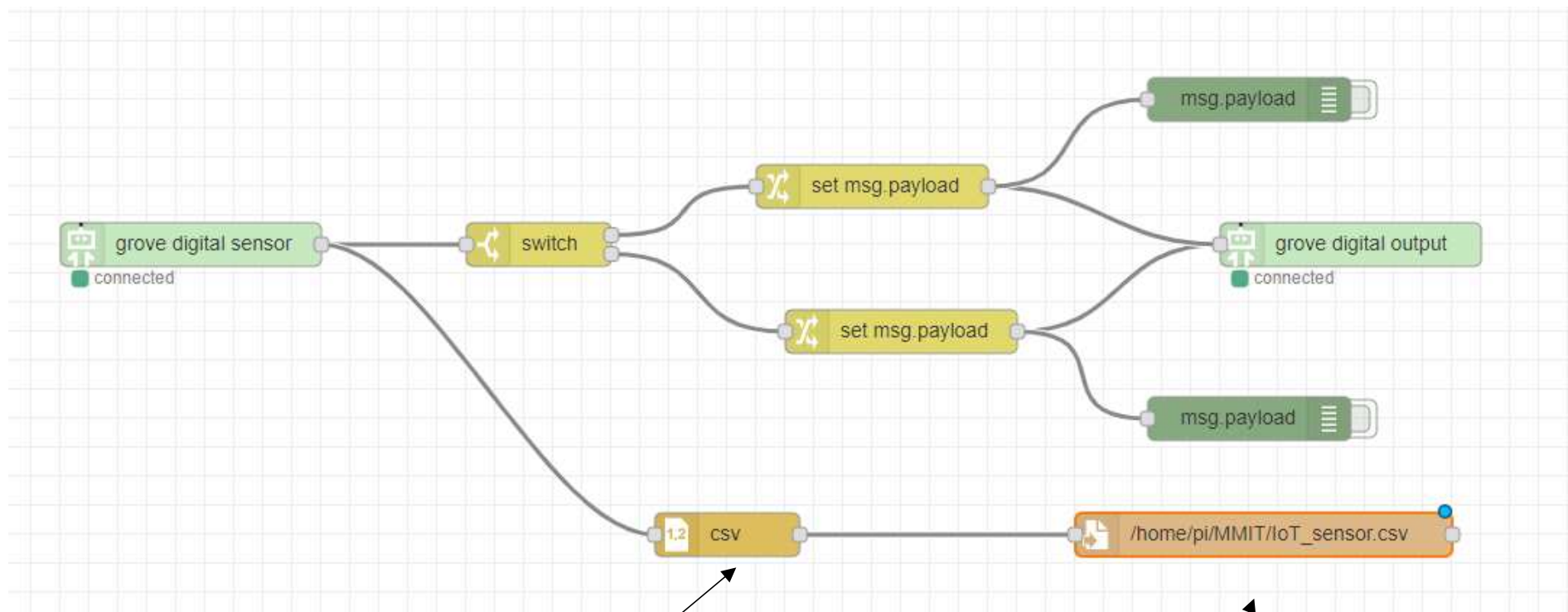
- ① Node-REDの基本
- ② 温度が異常だとブザーを鳴らす
- ③ 温度と湿度の履歴をCSVファイルに書出す
- ④ 定型メールの送信

## 2. 人感センサとカメラの利用

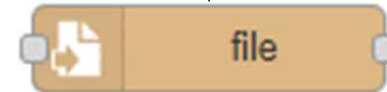
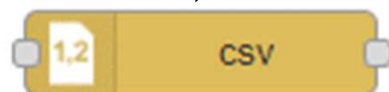
- ① 赤外線センサーに反応してLEDを点灯させる
- ② カメラで画像を撮影し保存する
- ③ Webサーバーを使う
- ④ 監視カメラを作る



# CSV出力用フローを追加する



パレット内のノード



# プロパティを設定する



特に修正なし。

出力したいファイル名を  
ディレクトリから入力する。

csv ノードを編集

削除 中止 完了

▼ プロパティ

列名

区切り文字

名前

---

CSVからオブジェクトへ変換

入力 最初の  行を無視する  
 1行目に列名を含む

出力

---

オブジェクトからCSVへ変換

出力  1行目を列名とする  
改行コード

file ノードを編集

削除 中止 完了

▼ プロパティ

ファイル名

動作

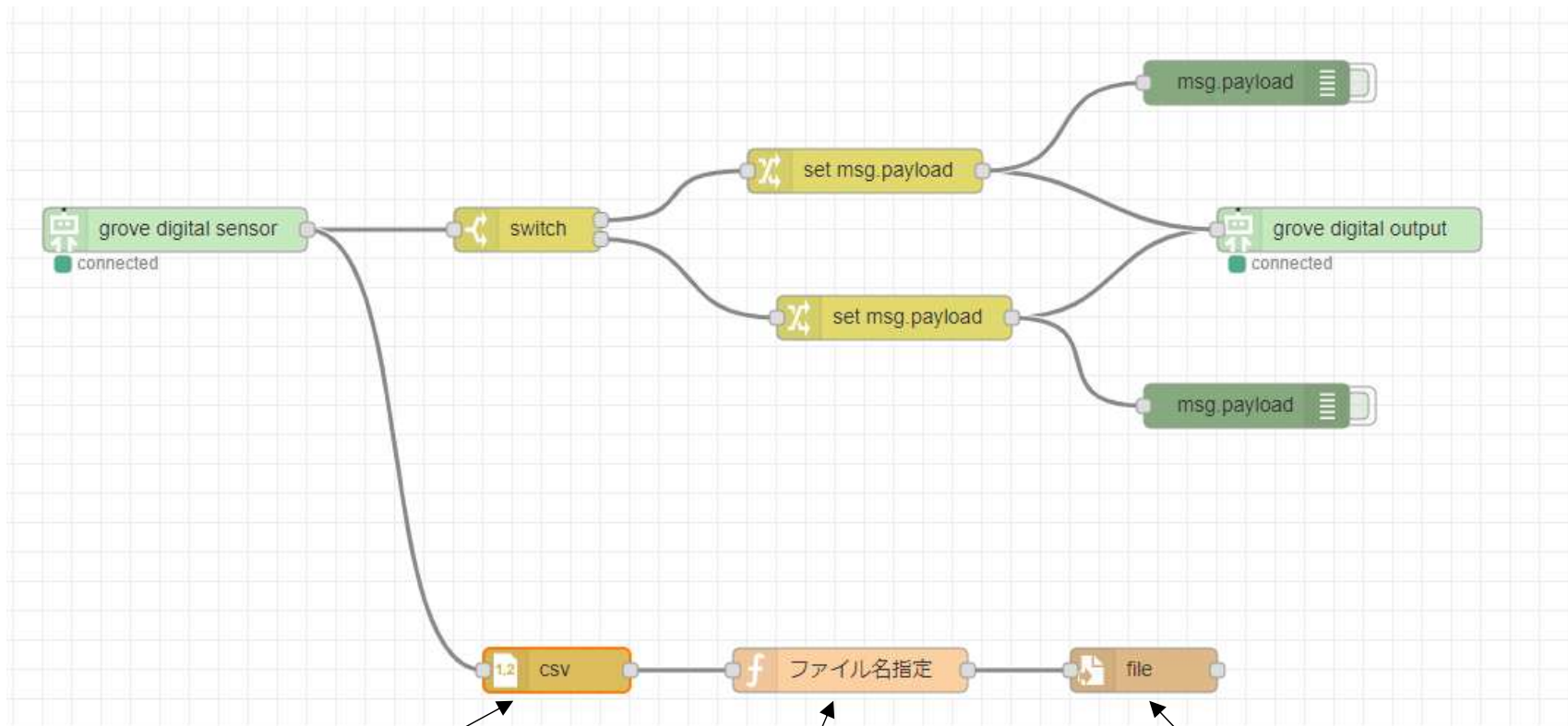
メッセージの入力のたびに改行を追加  
 ディレクトリが存在しない場合は作成

名前

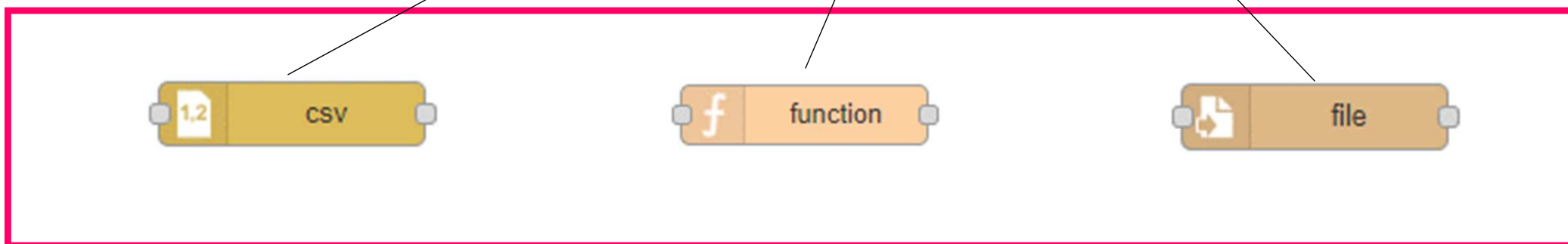
注釈: 「ファイル名」はフルパスを設定する必要があります。



# ファイル名を動的につけたい場合



パレット内のノード



# プロパティを設定する



特に修正なし。

ファイル名を動的につけたい場合はfunctionノードで指定しなければならない。

# ファイル名指定プログラム



```
1 var today = new Date();
2 var y = today.getFullYear();
3 var m = ("00" + String(today.getMonth() + 1)).slice(-2);
4 var d = ("00" + String(today.getDate())).slice(-2);
5 var filedate = "/home/pi/mmit/" + y + m + d + ".csv";
6 msg.filename = filedate;
7 return msg;
```

# 注意



- Node-REDには2千に近いノードが公開されています
- オープンソースのため全てが正しく動作する保証はありません
- 想定通りに動かない場合は、functionノードなどでjavascriptを使って一から作成する場合があります
- 今回のcsvノードも一例で、一行目にタイトル行を設定すると正常に動きません

# CSVで1行目にタイトル行を入れる方法

タイトル行を出力するフローとデータを出力するフローを別にする

※ファイル出力の1行目処理時

[csv]ノード

- 「列名」にカンマ区切りで列名を指定。
- 「オブジェクトからCSVへ変換」の「出力」にて
- 「1行目を列名とする」にチェックを入れる。
- Windowsでcsvを使用する前提なら「改行コード」は”Windows”にした方が良いでしょう。

[file]ノード

- 「動作」にて”ファイルを上書き”を指定。

-----  
※ファイル出力の2行目以降処理時

[csv]ノード

- 「列名」(1行目時と同様)
- 「オブジェクトからCSVへ変換」の「出力」にて
- 「1行目を列名とする」にチェックを入れない。
- 「改行コード」(1行目時と同様)

[file]ノード

- 「動作」にて”ファイルへ追記”を指定。

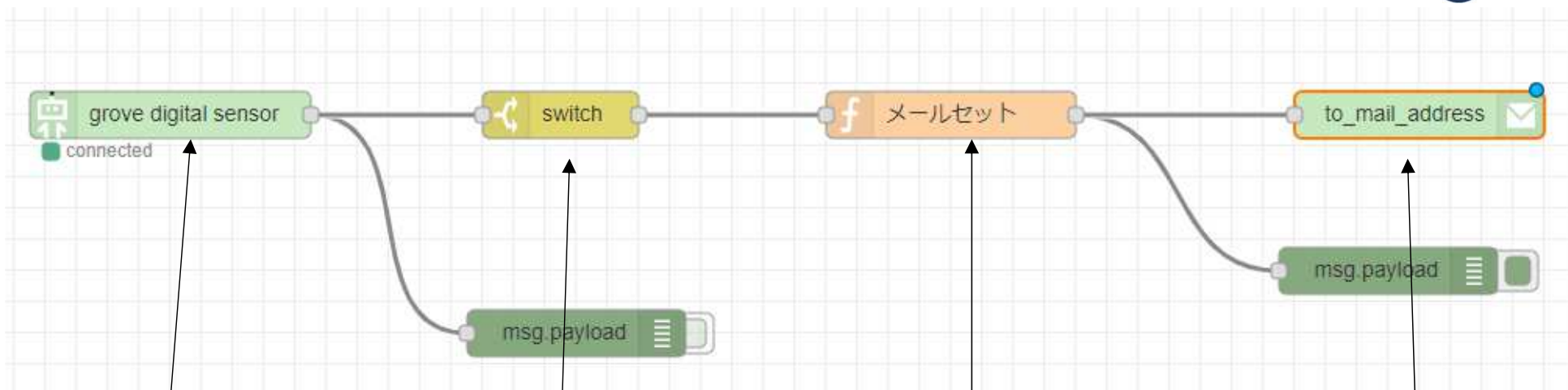
## 1. メールとWebサーバ利活用

- ① Node-REDの基本
- ② 温度が異常だとブザーを鳴らす
- ③ 温度と湿度の履歴をCSVファイルに書出す
- ④ 定型メールの送信

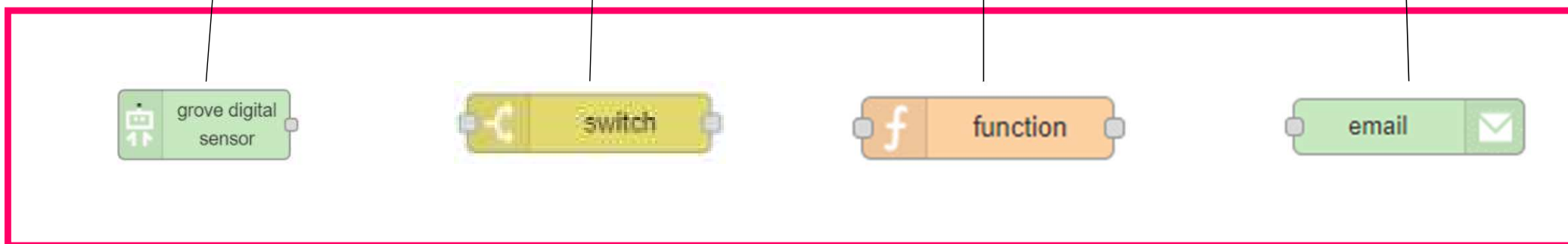
## 2. 人感センサとカメラの利用

- ① 赤外線センサーに反応してLEDを点灯させる
- ② カメラで画像を撮影し保存する
- ③ Webサーバーを使う
- ④ 監視カメラを作る

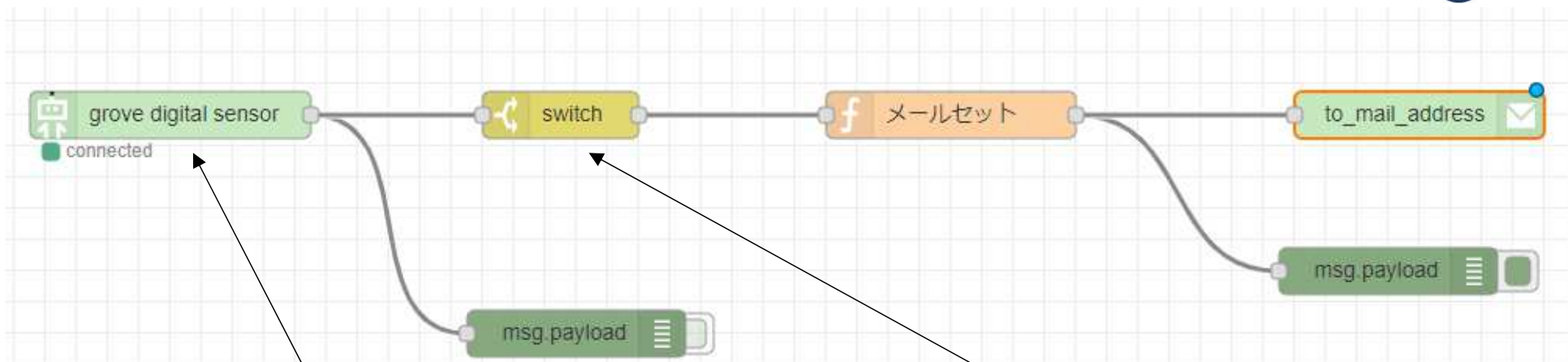
# ボタンを押したらメール送信



パレット内のノード



# プロパティを設定する



ポート2の押しボタンスイッチに設定。

Trueが送られてきた時だけメッセージを送る様に設定。

grove digital sensor ノードを編集

削除 中止 完了

▼ プロパティ

Board GrovePi

Sensor Type Button

Digital Pin Digital 2

Interval 1 Seconds

Name Name

switch ノードを編集

削除 中止 完了

▼ プロパティ

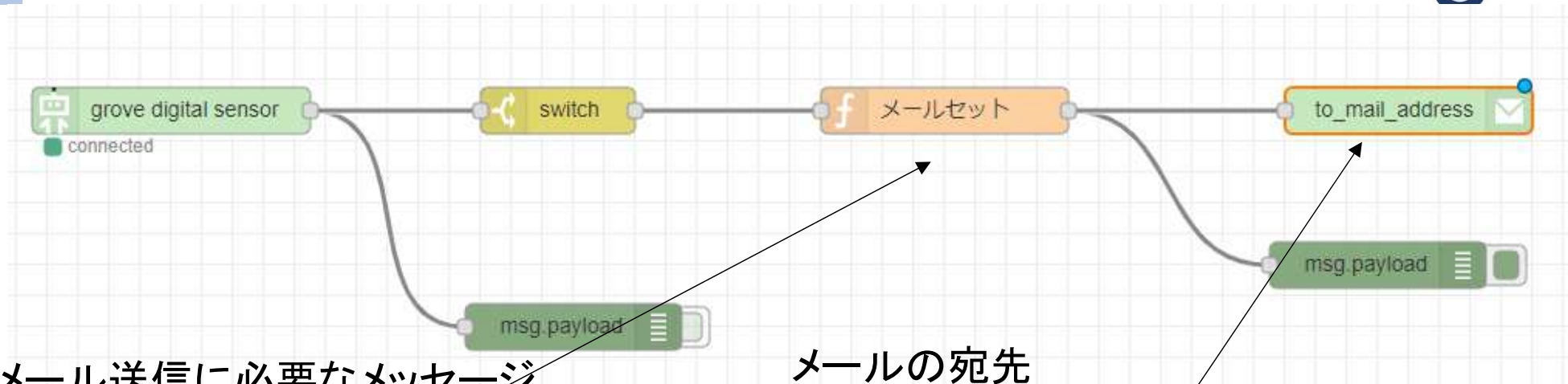
名前 名前

プロパティ ▼ msg. payload

is true → 1 x



# プロパティを設定する



メール送信に必要なメッセージ  
データを設定する。

メールの宛先  
Gmail用ユーザID  
パスワードを設定。

function ノードを編集

msg.payload  
msg.topic

削除 中止 完了

プロパティ

名前

メールセット

コード

```

1 msg.payload = "Node-REDからこんにちは。Googleアカウン
2 msg.topic = "テスト連絡";
3 return msg;

```

email ノードを編集

削除 中止 完了

プロパティ

宛先 to\_mail\_address

サーバ smtp.gmail.com

ポート 465  安全な接続を使用

ユーザID user\_ID

パスワード .....

名前 名前

# 前半5日間の進め方



## 午後の実習

- 1日目 実習のための環境設定 → 課題発見ワークショップ
- 2日目 デバイス信号のイン/アウト → センサデータの見える化
- 3日目 メールとWebサーバ利活用 → 人感センサとカメラの利用
- 4日目 業務システムの基本パターン → バーコードリーダとNFC
- 5日目 データ分析続き → 工程進捗管理ボード

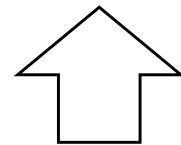
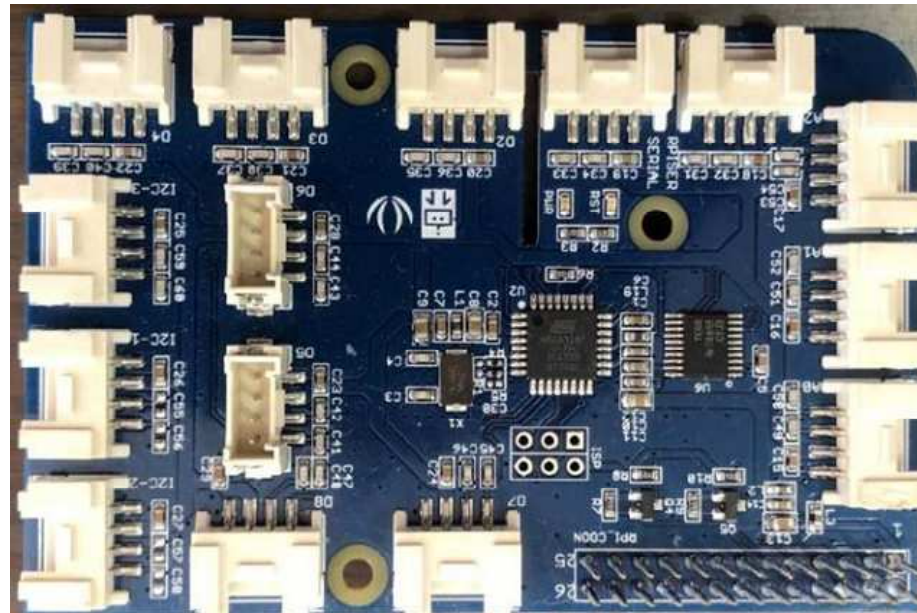
## 1. メールとWebサーバ利活用

- ① Node-REDの基本
- ② 温度が異常だとブザーを鳴らす
- ③ 温度と湿度の履歴をCSVファイルに書出す
- ④ 定型メールの送信

## 2. 人感センサとカメラの利用

- ① 赤外線センサーに反応してLEDを点灯させる
- ② カメラで画像を撮影し保存する
- ③ Webサーバーを使う
- ④ 監視カメラを作る

# Grove Mini PIR motion の取付

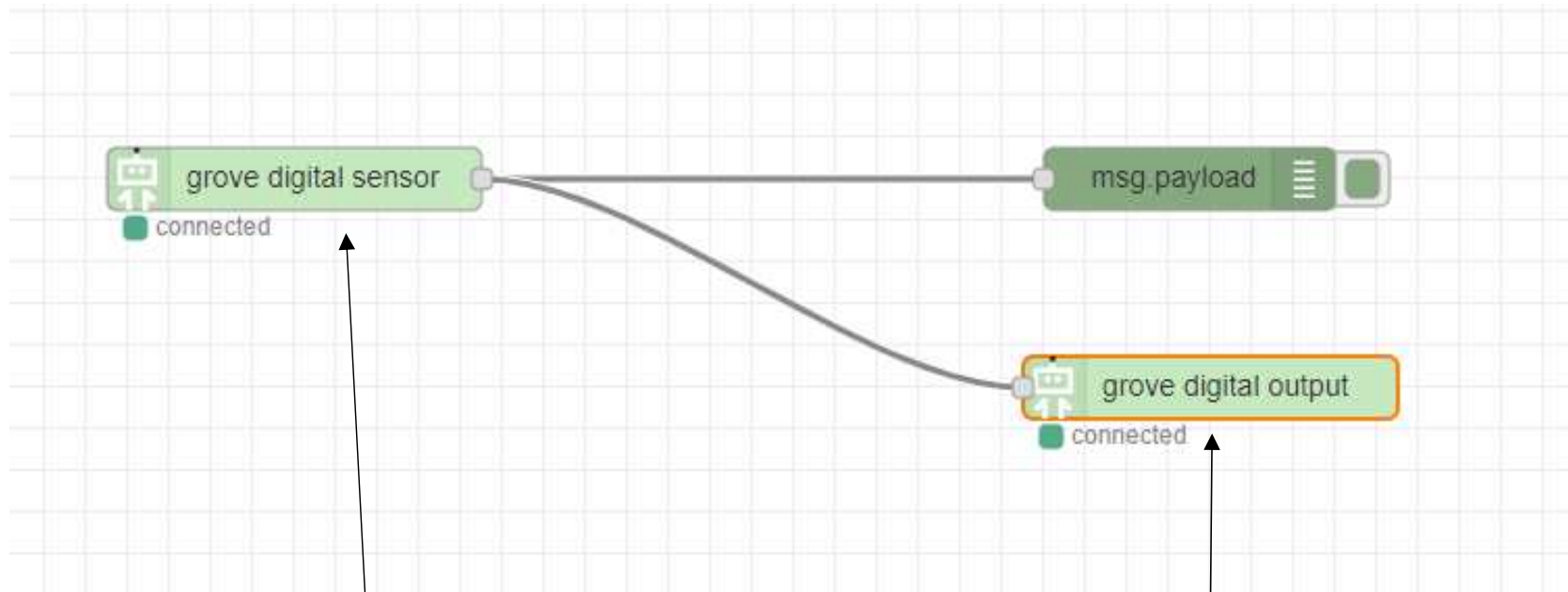


Digital port 8 に接続する



Buttonスイッチと同じように使えます。  
センサーの前の人が動くとONします。

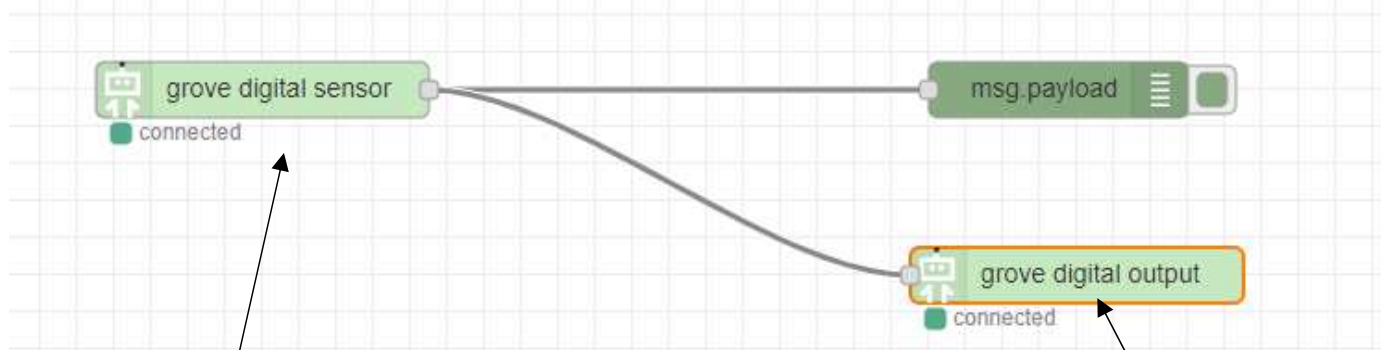
# 赤外線センサーが反応しLED点灯



パレット内のノード



# 赤外線センサーが反応しLED点灯



赤外線センサーの接続されたポート8に設定。

LEDの接続されたポート5に設定。

grove digital sensor ノードを編集

削除 中止 完了

▼ プロパティ

Board GrovePi

Sensor Type Button

Digital Pin Digital 8

Interval 1 Seconds

Name Name

grove digital output ノードを編集

削除 中止 完了

▼ プロパティ

Board GrovePi

Digital Pin Digital 5

Name Name

## 1. メールとWebサーバ利活用

- ① Node-REDの基本
- ② 温度が異常だとブザーを鳴らす
- ③ 温度と湿度の履歴をCSVファイルに書出す
- ④ 定型メールの送信

## 2. 人感センサとカメラの利用

- ① 赤外線センサーに反応してLEDを点灯させる
- ② カメラで画像を撮影し保存する
- ③ Webサーバーを使う
- ④ 監視カメラを作る



# camerapiノードのインストール



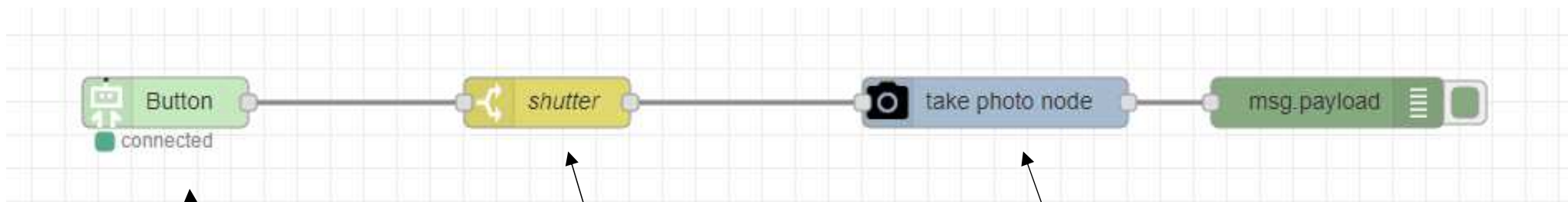
サイドバーメニュー>パレットの管理>ノードを追加>camerapiで検索>インストール>追加

The image illustrates the process of installing the camerapi node in Node-RED through three sequential screenshots:

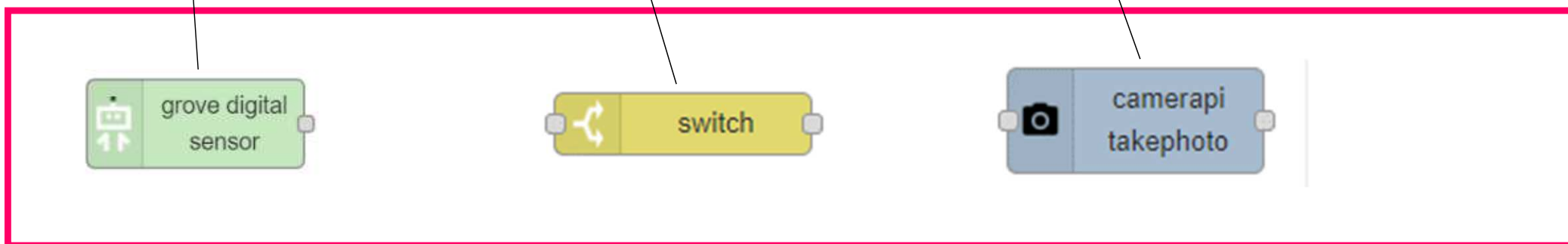
- Side Menu:** The left sidebar menu is open, and the 'パレットの管理' (Manage Palettes) option is highlighted with a red box.
- Node Addition Dialog:** The 'ノードを追加' (Add Node) dialog is displayed. The search bar contains 'camerapi', and the 'node-red-contrib-camerapi' node is selected in the palette. The 'ノードを追加' button is also highlighted.
- Node Palette:** The node palette is shown with the 'camerapi takephoto' node highlighted in the 'Raspberry Pi' category.



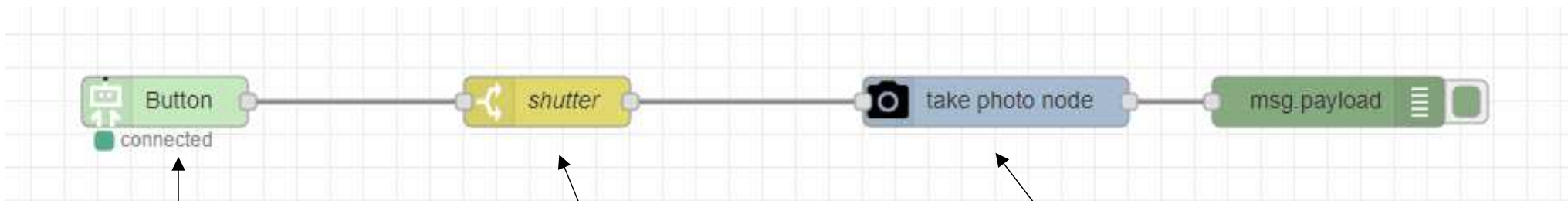
# ボタンを押したら写真を撮る



パレット内のノード



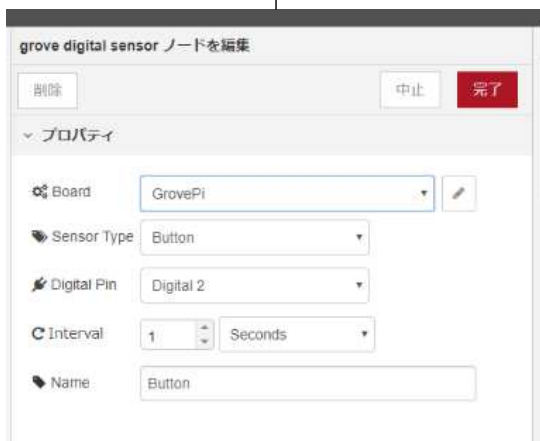
# ボタンを押したら写真を撮る



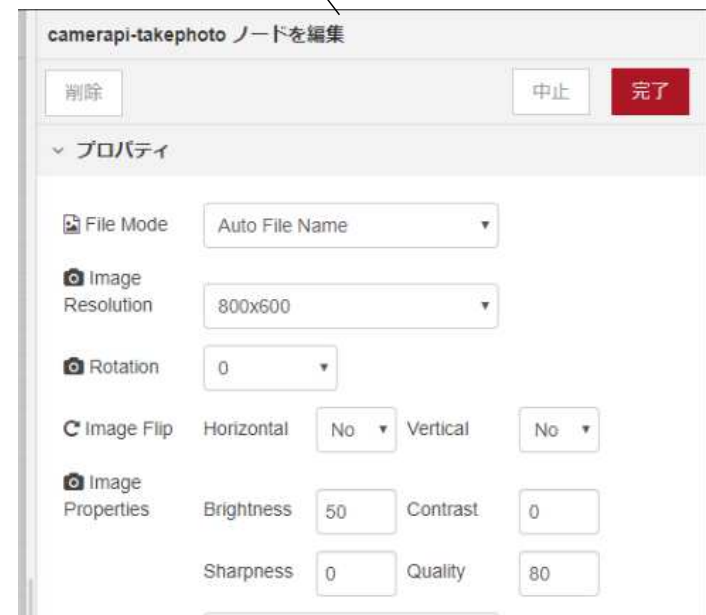
押しボタンスイッチが接続されたポート2に設定。

Buttonノードからtrueデータが渡された時にのみ出力するように設定。

写真データが重くならないように800x600に設定。デフォルトのままだと/home/pi/Picturesにファイル名自動設定で写真を保存。



msg.payload is true.



## 1. メールとWebサーバ利活用

- ① Node-REDの基本
- ② 温度が異常だとブザーを鳴らす
- ③ 温度と湿度の履歴をCSVファイルに書出す
- ④ 定型メールの送信

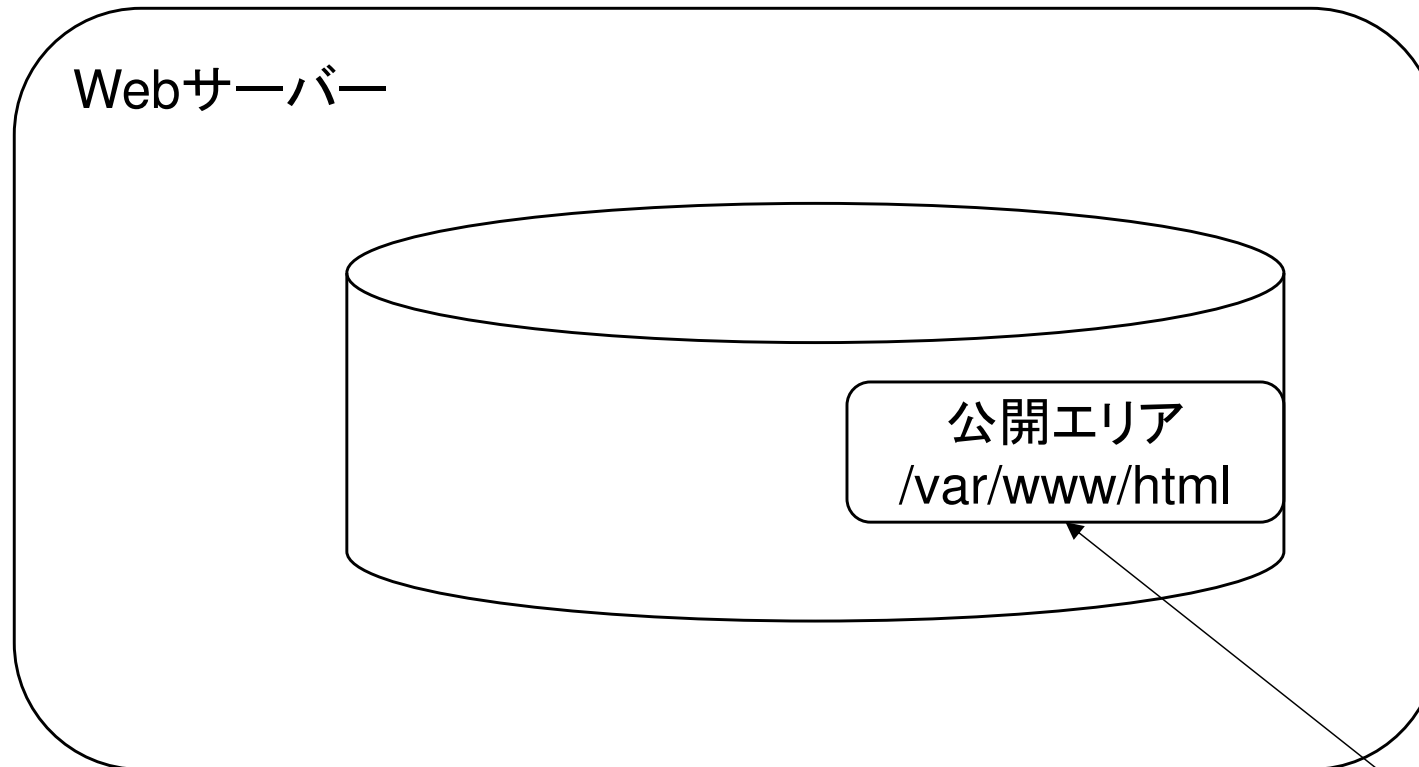
## 2. 人感センサとカメラの利用

- ① 赤外線センサーに反応してLEDを点灯させる
- ② カメラで画像を撮影し保存する
- ③ Webサーバーを使う
- ④ 監視カメラを作る

# Webサーバーのしくみ(超簡略)



Raspbian(ラズパイのOS)にはApache2というWebサーバーがインストールされています



htmlファイル:  
Webコンテンツの  
基本となるファイル

cssファイル:  
htmlファイルの表  
示形式を共通で設  
定することができる  
ファイル

この中のファイルをブラウザで開く

# シェアディレクトリの所有権を変える

- 以下のコマンドをコンソールから入力する  
/var/www/htmlの所有者をpi ユーザーに変更する。  
`$ sudo chown -R pi /var/www/html`

# photo.html

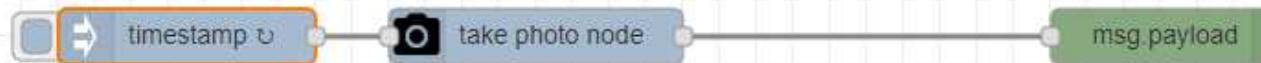


```
<html>
<head>
<meta http-equiv="refresh" content="5">
<meta http-equiv="Pragma" content="no-
cache">
<meta http-equiv="Cache-Control" content="no-
cache">
</head>
<body>
hello world!<br>

</body>
</html>
```

撮影した写真ファイルがこれで表示されます。

# 10秒間隔でカメラで撮像し更新する



タイムスタンプで  
10秒間隔に  
msg.payloadを出力

msg.payloadの出力を  
受けて、写真を撮影

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 0.1 秒後、以下を行う

繰り返し

時間間隔  秒

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。

File Mode: Custom File Name  
File Name: photo.jpg  
File Path: /var/www/html/  
Image Resolution: 800x600

camerapi-takephoto ノードを編集

削除 中止 完了

プロパティ

File Mode: Custom File Name

File Name: photo.jpg

File default path: No

File Path: /var/www/html/

File Format: JPEG

Image Resolution: 800x600

Rotation: 0

Image Flip: Horizontal No Vertical No

Image Properties: Brightness 50 Contrast 0 Sharpness 0 Quality 80

Image Effect: none

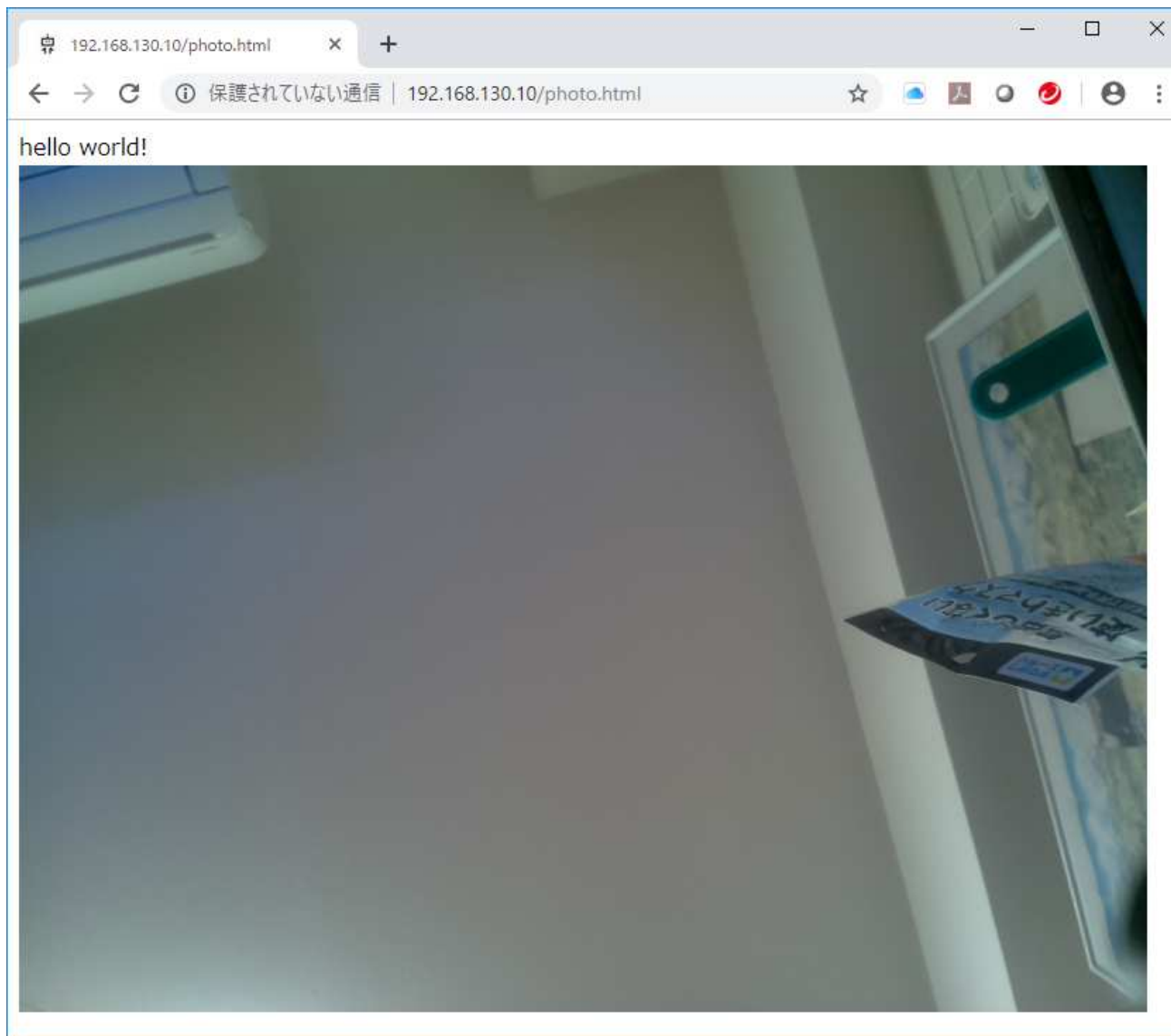
ISO: auto

Preset modes: Exposure auto AWB auto

# PCのブラウザで確認



URL





## 1. メールとWebサーバ利活用

- ① Node-REDの基本
- ② 温度が異常だとブザーを鳴らす
- ③ 温度と湿度の履歴をCSVファイルに書出す
- ④ 定型メールの送信

## 2. 人感センサとカメラの利用

- ① 赤外線センサーに反応してLEDを点灯させる
- ② カメラで画像を撮影し保存する
- ③ Webサーバーを使う
- ④ 監視カメラを作る

赤外線センサーで人の動きを検出したら自動的に写真を撮影し、PCのWebブラウザで監視できるシステムを作成してください。

ヒント:

ボタンの代わりに

Grove-Mini PIR Motion Sensorを使う。

写真の保存先を/var/www/html/にする。

# 自宅や職場で 新たにラズパイやGrovePiを 使ってみるときの注意点

# ■ 今入手できる最新の環境



- ラズパイ

- 最新のRaspbian Stretch with desktop and recommended softwareをダウンロード・インストール  
<https://www.raspberrypi.org/downloads/raspbian/>

- GrovePi

- ファームウェアバージョンv.1.3.0  
Node-REDのgrovepi digital sensor でbutton入力を検出できない(回避方法あり)

# GrovePiライブラリインストール方法

- piユーザーホームディレクトリ (/home/pi) で次のコマンドを実行する  
`$ curl -kL dexterindustries.com/update_grovepi | bash`
- 念のためGrovePiのファームウェアバージョンを確認するために以下のコマンドを実行する  
`$ cd /home/pi/Dexter/GrovePi/Software/Python`  
`$ python grove_firmware_version_check.py`
- GrovePi has firmware version 1.3.0と表示されればOK

# Button入力への対応例



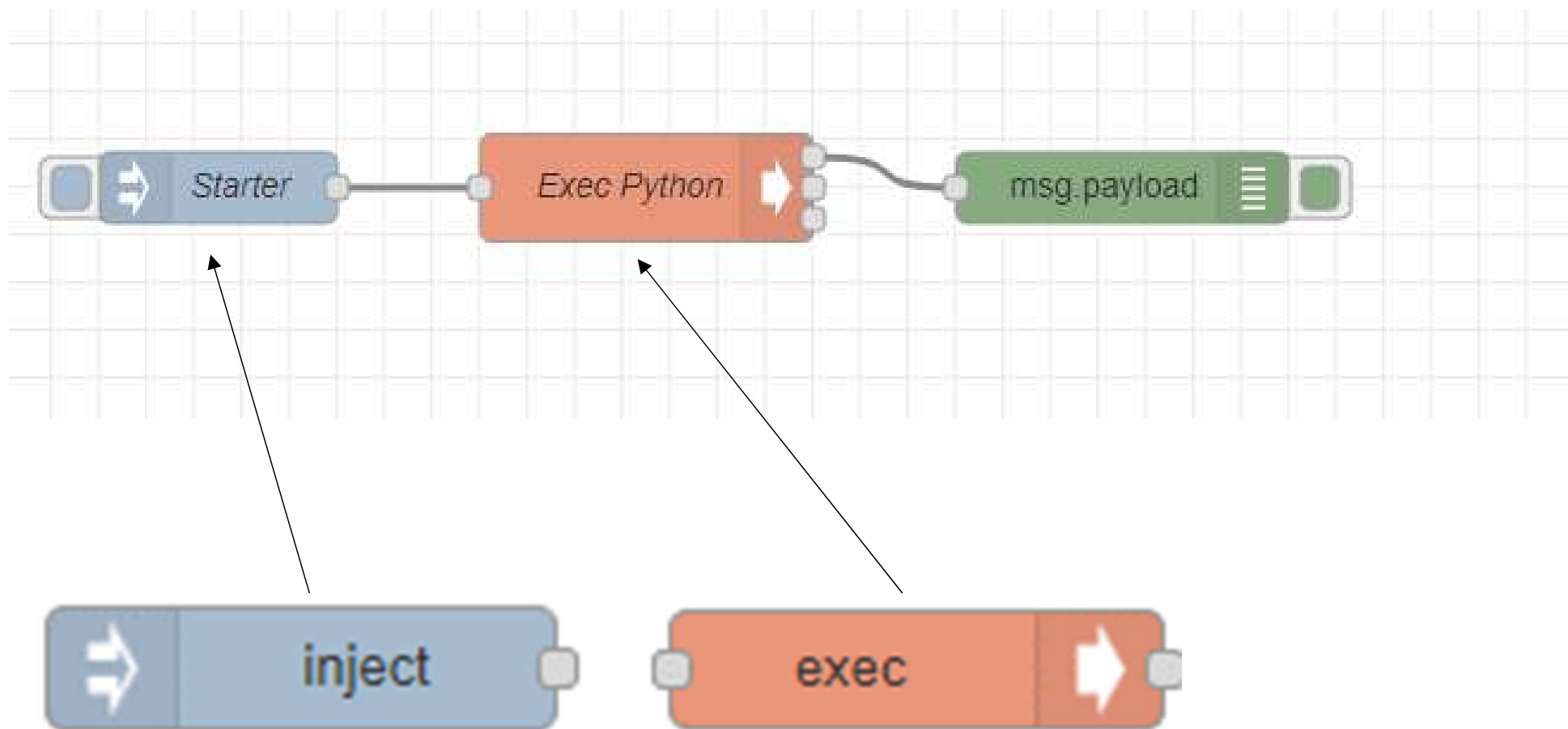
The screenshot shows the Node-RED web interface. The main workspace contains a flow named "gp\_button" with three nodes connected in sequence: a blue "Starter" node, an orange "Exec Python" node, and a green "msg.payload" node. The left sidebar shows the "Inputs" and "Outputs" categories with various nodes like inject, catch, status, link, mqtt, http, websocket, tcp, udp, Watson IoT, debug, link, mqtt, http response, websocket, tcp, udp, and play audio. The right sidebar displays the "Information" panel for the selected flow, showing details such as the flow ID, name, and status.

情報	
フロー	"25d39070.8f3eb"
名前	gp_button
状態	有効

フローの詳細  
なし

フローデータが入ったJSONファイルをエディタへドラッグ、または `ctrl+i` により、フローを読み込むことができます。

# ノードを配置する



# プロパティを設定



起動時にpythonプログラムを走らせる

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 なし

📌 名前 Starter

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。

python3でtest\_grove\_button1.pyを実行

exec ノードを編集

削除 中止 完了

▼ プロパティ

📄 コマンド python3 /home/pi/mmit/test\_grove\_button1.py

+ 引数  msg.payload

追加引数

👉 出力 コマンド実行中 - spawnモード

旧形式の出力を使用(互換モード)

🕒 タイムアウト 30 秒

📌 名前 Exec Python



# フローの出力

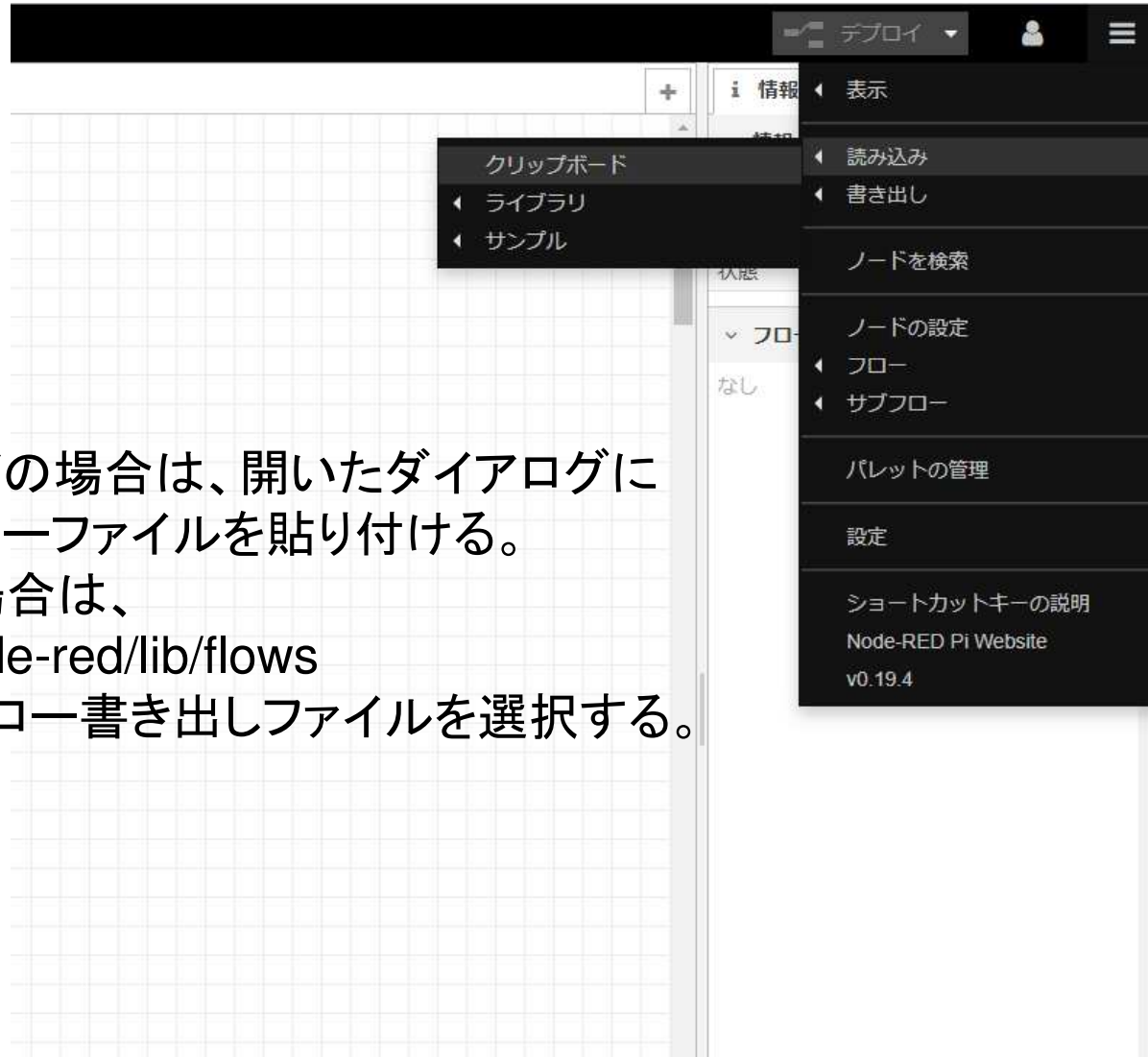


右上の三本線アイコンをクリック>書き出し>ライブラリかクリップボード

クリップボードでは、ここから  
右クリック>コピーで  
Windowsの.txtにもできる。

The screenshot shows the Node-RED interface. In the top right corner, a red box highlights the hamburger menu icon (three horizontal lines). A dropdown menu is open, showing options like '表示', '読み込み', '書き出し', 'ノードを検索', 'ノードの設定', 'フロー', 'サブフロー', 'パレットの管理', '設定', 'ショートカットキーの説明', 'Node-RED Pi Website', and 'v0.19.4'. The '書き出し' (Export) option is highlighted. Below the menu, a dialog box titled 'フローをクリップボードへ書き出し' (Export flow to clipboard) is open. It has buttons for '書き出し' (Export), '選択したフロー' (Selected flow), '現在のタブ' (Current tab), and '全てのタブ' (All tabs). A text area contains JSON code for a flow node. Below the text area are radio buttons for 'インデントのないJSONフォーマット' (JSON format without indentation) and 'インデント付きのJSONフォーマット' (JSON format with indentation). At the bottom of the dialog are '中止' (Cancel) and '書き出し' (Export) buttons.

# 作ったフローの読み込み



ダッシュボードの場合は、開いたダイアログにコピーしたフローファイルを貼り付ける。  
ライブラリの場合は、  
`/home/pi/.node-red/lib/flows`  
に置かれたフロー書き出しファイルを選択する。

# 最新バージョンでの機能



- v0.20.5 ではJSONファイル単位でダウンロード（エクスポート）や 読み込み（インポート）する機能が追加されたので、貼り付け以外に選択肢ができました
- JSONでのダウンロードは、大規模なフローのエクスポートにおいて圧倒的に効率が良くなりました。（処理が早い）

# Node-REDをwebサーバにする方法

- シェアディレクトリを新規作成(例)  
`$ mkdir /home/pi/mmit/html`
- photo.htmlとstyle.cssを上記ディレクトリにコピー
- /home/pi/.node-red/setting.jsを変更

# settings.jsを開く



The screenshot shows a VNC viewer window titled "192.130.10 (mmi190xx) - VNC Viewer". The main window displays a file manager interface for the "/home/pi/node-red" directory. The "node-red" folder is selected, and its contents are shown, including "lib", "node\_modules", "package.json", "package-lock.json", "settings.js", "flows\_mmit\_190xx.json", "config.json", "config.json.backup", and "sessions.json". The "settings.js" file is highlighted, and a preview window is open, displaying its contents:

```
/**
 * Copyright JS Foundation and other contributors, http://js.foundation
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

// The 'https' setting requires the 'fs' module. Uncomment the following
// to make it available:
//var fs = require("fs");

module.exports = {
  // the tcp port that the Node-RED web server is listening on
  uiPort: process.env.PORT || 1880,

  // By default, the Node-RED UI accepts connections on all IPv4 interfaces.
  // To listen on all IPv6 addresses, set uiHost to "::".
  // The following property can be used to listen on a specific interface. For
  // example, the following would only allow connections from the local machine
```

# シェアディレクトリの設定

反転部分を下図のように修正する。

Node-REDのWebサーバーのシェアフォルダが/home/pi/mmit/htmlになる。  
ラズパイを再起動する。



```
*settings.js
ファイル(F) 編集(E) 検索(S) オプション(O) ヘルプ(H)

// The following property can be used in place of 'httpAdminRoot' and 'httpNodeRoot',
// to apply the same root to both parts.
//httpRoot: '/red',

// When httpAdminRoot is used to move the UI to a different root path, the
// following property can be used to identify a directory of static content
// that should be served at http://localhost:1880/.
httpStatic: '/home/pi/mmit/html/',

// The maximum size of HTTP request that will be accepted by the runtime api.
// Default: 5mb
//apiMaxLength: '5mb',

// If you installed the optional node-red-dashboard you can set it's path
// relative to httpRoot
//ui: { path: "ui" },

// Securing Node-RED
// -----
// To password protect the Node-RED editor and admin API, the following
// property can be used. See http://nodered.org/docs/security.html for details.
adminAuth: {
  type: "credentials",
  users: [{
    username: "admin",
    password: "$2a$08$2lWsDyQ3Tc1m1QuAU36/f0SteBDivHlQTrxl.WJoUkXbKirR.z8GK",
    permissions: "*"
  }
  ]
}
```

## 本教材利用上の注意事項

本教材の著作権は、厚生労働省に帰属します。  
詳細については、下記の利用規約をご確認ください。  
<https://www.mhlw.go.jp/chosakuken/index.html>