

# OpenBlocks IoT Family向け Node-REDスターターガイド



Ver.3.2.0

ぷらっとホーム株式会社

#### ■ 商標について

- ・ Linux は、Linus Torvalds 氏の米国およびその他の国における商標あるいは登録商標 です。
- 文中の社名、商品名等は各社の商標または登録商標である場合があります。
- その他記載されている製品名などの固有名詞は、各社の商標または登録商標です。

#### ■ 使用にあたって

- ・ 本書の内容の一部または全部を、無断で転載することはご遠慮ください。
- ・ 本書の内容は予告なしに変更することがあります。
- 本書の内容については正確を期するように努めていますが、記載の誤りなどにご指摘が ございましたら弊社サポート窓口へご連絡ください。
   また、弊社公開のWEBサイトにより本書の最新版をダウンロードすることが可能です。
- 本装置の使用にあたっては、生命に関わる危険性のある分野での利用を前提とされていないことを予めご了承ください。
- その他、本装置の運用結果における損害や逸失利益の請求につきましては、上記にかか わらずいかなる責任も負いかねますので予めご了承ください。

### 目次

第1章 はじめに	4
第2章 Node-RED 事前準備	5
2-1. Node-RED のインストール	5
2-2. Node-RED 起動設定	6
2-3. Node-RED のブラウザフィルタについて	7
2-4. パケットフィルタについて	7
第3章 Node-RED の簡易説明	8
3-1. Node-RED 画面構成	9
3-2. ノード種類	10
3-2. 入力ノード	10
3-3. 出力ノード	11
3-4. 機能ノード	
3-5. ソーシャルノード	
3 <sup>-</sup> 6. ストレージノード	
3-7. 分析ノード	
3-8. その他ノード	14
3-9. cloud ノード	
3-10. GatewayKit ノード	
3-11. location $\checkmark$ — F	
3-12. dashboard $\mathcal{I} - F$	
3-13. Google ノード	
第4章 ノード操作サンプル	
4-1. 位置情報の可視化	
4-2. ビーコンデータへの位置情報付与	
第5章 その他	
5-1. Node-RED ログについて	
5-2. Node-RED <b>へのノード追加</b>	30
5-3. Node-RED の Config 編集	
5-4. Node-RED のプロセス状況について	32
5-5. Node-RED の HTTPS 化	33

## 第1章 はじめに

本書は、OpenBlocks IoT Family に搭載可能な Node-RED の使用方法を解説しています。 搭載している Node-RED は IoT データ制御機能にて送信先の候補として用意しており、エ ッジコンピューティングの実装や対応していないクラウドへの対応を想定しております。

## 第2章 Node-RED 事前準備

#### 2-1. Node-RED のインストール

本製品出荷時では、Node-RED はインストールされておりません。そのため、WEB UI の 「メンテナンス」→「機能拡張」タブから Node-RED をインストールしてください。

(シュボード	サービス	システム	ネットワーク	メンテナンス	拡張	AirManage	技術情報
84	\$17710	<b>#5</b> (ā.t <b>#</b>	17.8h ++xt-1	機能拡張	1-1-	- 151714	
				10066310.00	2 1.0	1.478	
機能拡張							
機能拡張	- 11 жарт		19aa daa 🗢 💌				
機能拡張 インストー 地転用要	-ル機能		パッケージ・	,ケージを選択),アく	ださい。		

WEB UI の「メンテナンス」タブを選び、さらに「機能拡張」タブをクリックすると機能拡張」
 張用のパッケージを選択することができます。

インストール機能のリストから「Node-RED」 を選択します。

その後、インストールの「実行」ボタンを押し、 インストールを行ってください。

尚、インストール完了後には反映を行うため、 本体再起動が必要となります。そのため、「メ ンテナンス」→「停止・再起動」から本製品の 再起動を行ってください。

 CopenBlockのでいて
 CopeBlockのでいて
 <thCopeBlockのでいて</t

#### 2-2. Node-RED 起動設定

Node-RED のインストールが完了している場合、サービス WEB UI の「サービス」→「基本」タブに Node-RED の項目が表示されます。

この Node-RED のリンクを押して下さい。これにより、「Node-RED」タブへと遷移します。

ファイル
● 使用する ● 使用しない
(1880
◎ 使用する ◉ 使用しない
● 本システムコンフィグ ○ ユーザー定義コンフィグ

「Node-RED」→「Node-RED」タブから Node-RED 自体の設定を行えます。

#### $\underline{Node-RED}$

#### 使用設定:

Node-REDの使用設定を行います。Node-RED を使用する場合には、「使用する」を選択して ください。

#### ポート番号 :

Node-RED ヘアクセスするためのポート番号 を設定します。通常は、デフォルトの1880か ら変更する必要はありません。

#### ログイン認証設定:

ログイン認証を使用する場合には"使用する" を選択し、ユーザー名及びパスワードを設定し 保存ボタンを押してください。

#### 使用 Config 設定:

Node-RED 自体の起動設定 Config ファイルを 本システムまたはユーザー編集したものと切 り替えることができます。

「ユーザー定義コンフィグ」を選択した場合、

コンフィグ編集用タブが表示されます。

設定完了後、保存ボタンを押すことにより Node-RED が起動・停止します。

※リンクボタンを押すことによって起動中の Node-RED ヘアクセスすることができます。 ただし、AirManage 経由の場合にはアクセスは行えません。AirManage 経由の場合には URI プロキシ機能を用いてアクセスしてください。

#### 2-3. Node-RED のブラウザフィルタについて

Node-RED のインストールが完了している場合、Node-RED 自体への WEB アクセスを行 う場合、Node-RED のフィルターを開放する必要があります。そのため、WEB UI の「シ ステム」→「フィルター」タブにてフィルターを開放してください。

Io ento							ログイン ID: admin (植物: スーパーユーザ	-) <u>국국자</u> 일 대상
ユポード サービス	システム	ネットワーク	メンテナンス	45.95	AirManage	反抗情難		
地を保存しました。 SHを無効にするには無効道	択後に保存ボタンを	押下ください						
84 338	100-	יפטואר	- south	744-	9 WEB1-9	- 77-64	<b>部度 ライセンス</b>	-
VELOOKE SN								
フィルター開放設定 🖻 SSH	Rataba 6.2 < №988 ®	(設定を有効にする()) 有効 <sup>()</sup> 気効						_
		and () was						
Node-RED		HAU - MAN						
Node-RED								
Node-RED 推作 保持		nu – mu						_
Node-RED Reff (##) iptables		1. m - m v						_
Node-RED Heft (777) Iptables&x (ptables&x)		5	/211					_

デフォルトでは Node-RED のブラウザアクセ スはできないようにフィルターが適用されて います。

"有効"に設定し、保存ボタンを押してください。

尚、セキュリティの観点上、Node-RED の設 定完了後はフィルターを閉鎖してください。

#### 2-4. パケットフィルタについて

Openblocks IoT Familyの入力方向のパケットフィルタは、WebUI へのアクセスや時刻同 期等、システムの動作に必要となるポートを除き開放されておりません。

TCP 入力ノード 等、リモートからの接続を待ち受けるノードを使用し外部から受信する 場合、必要に応じて別途パケットフィルタを開放する必要があります。

パケットフィルタの操作は、/var/webui/local/bin/iptables-ext.sh ファイルを作成し WEB UI の「メンテナンス」→「フィルター」タブから、iptables 編集するスクリプトを作成し てください。

## 第3章 Node-RED の簡易説明

Node-RED はデフォルトで 1880 番ポートを用いてブラウザアクセス行います。 そのため、デフォルトでの Wi-Fi 経由での Node-RED へのアクセスする為の URL は以下 となります。

http://192.168.254.254:1880/

アクセスした場合、以下のような画面が初期状態では表示されます。(ログイン認証設定を していない場合)

Node-RED					=/= デブロイ	• =
q ノードを検索 フロー1			+	情報	デバッグ	dashboar %
~ 入力			^	Flow		*
iniart 0				Name	70-1	
- inject				ID	"90fcd19f.510a8"	
catch				Status	Enabled	
status				Information		
🗧 link 💿						
) mqtt						
Attp 💿						
💮 websocket						
🕴 tcp 💿						
🦂 udp 💿						
😩 ipc 😐						
) aws mqtt						
22 hangouts						
💽 Watson IoT 🔋						v C x
👖 serial 😐					P ctrl-0k 7	、タブの
~ 出力				t刀 t	替えができます	0
debug			+			
A V		- 0	+			

また、本製品向けにデフォルトで用意している入力・出力・処理等のノードは以下となり ます。



### 3-1. Node-RED 画面構成

Node-RED の画面は以下のように構成されています。



#	項目	説明
1	フローシート	処理フローを記述するワークスペースです。
2	デプロイ	デプロイボタンをクリックすることでシートに記述した
		処理フローを有効化します。
3	フロー	ノードを配置し結線することでデータの流れ(処理フロ
		ー)を定義します。
4	ノードパレット	処理フローの構成に用いられるノードの一覧です。
5	表示切替	ノード情報・デバック情報の表示を切り替えます。
6	ノード情報	ノード情報、又はデバック情報が表示されます。

#### 3-2. ノード種類

Node-RED では大きく分けて以下のようなコネクタ配置のノードがあります。



上記のように処理が行われるため、データは左から右へと処理が行われます。

#### 3-2. 入力ノード

- inject	ノードの左部のボタンを押すことでノードに設定された
inject )	timestamp 等を入力データもしくはイベントとします。
eatch	同じシート上のノードで発生したエラーを入力データもし
Cattin	くはイベントとします。
status	同じシート上のノードのステータスを入力データもしくは
	イベントとします。
> link	いずれかの link output node の出力を入力データもしくは
	イベントとします。
) matt	MQTT broker へ subscrib し、publish message を待ち受
inqu )	け、入力データもしくはイベントとします。
http	HTTP リクエストを待ち受け、入力データもしくはイベン
	トとします。
websocket	Websocket による接続を待ち受け、入力データもしくはイ
Webbook Ref	ベントとします。
A ten	TCP による接続を待ち受け、入力データもしくはイベント
W L	とします。
dbu (fr	UDP による接続を待ち受け、入力データもしくはイベント
	とします。

inc	Unix ドメインソケットによる接続を待ち受け、入力データ
	もしくはイベントとします。
)) aws mqtt	AWS IoT からのメッセージを待ち受け、入力データもしく
	はイベントとします。
22 handouts	Google hangouts からのメッセージを待ち受け、入力デー
hangouts	タもしくはイベントとします。
- Tol orego	Watoson IoT からの device command を待ち受け、入力
Watsonio	データもしくはイベントとします。
	シリアルインタフェースからの入力を待ち受け、入力デー
	タもしくはイベントとします。

## 3-3. 出**力ノー**ド

debug	入力データをデバック情報として表示します。
link	入力データをいずれかの Input link node へ出力します。
(mqtt )	入力データを MQTT broker へ publish します。
http response	入力データを http input node への入力に対する応答とし て出力します。
websocket	入力データを Websocket サーバへ出力します。
tcp	入力データを TCP サーバに出力します。
udp 🚯	入力データを UDP サーバに出力します。
o ipc 🕏	入力データを Unix ドメインソケットに出力します。
aws mqtt 🔊	入力データを AWS IoT に出力します。
hangouts	入力データを Google hangouts へ出力します。
Watson IoT	入力データを Watson IoT へ出力します。



### 3-4. 機能ノード

f function	入力データを JavaScript で処理し出力します。
• { template	入力データを整形して出力します。
<del>ර</del> ා delay	入力データを設定された時間後に出力します。
trigger	入力データに対し、タイムアウトを設け2つのメッセージ を出力します。
comment	フローにコメントを付けます。
http request	入力データに対し、設定された URL に対し http リクエス トを行い、その応答を出力します。
tcp request	入力データに対し、設定されたサーバに対し TCP 接続を行 い、その応答を出力します。
switch	入力データを設定された分岐条件に応じて異なるノードに 出力します。
οχ change	入力データの属性を設定・変更・削除もしくは移動して出 力します。
otij range o	入力データのスケールを変更して出力します。
o∎≡ split o	入力データを設定される文字で区分して出力します。
o ∎•∎ join o	入力データを結合して出力します。
1,2 CSV	CSV 書式のデータと JavaScript Object を相互変換します。
html	HTML 書式のデータと JavaScript Object を相互変換します。
<mark>(}</mark> json □	JSON 書式のデータと JavaScript Object を相互変換します。

💽 xml	XML 書式のデータと JavaScript Object を相互変換します。
yaml 🛛	YAML 書式のデータと JavaScript Object を相互変換します。
d t4d keymapper	入力データを Toami for DOCOMO 用の入力データへと変換します。
aws thing	AWS IoT の Thing Shadow 制御設定を行います。
o 🗊 rbe 🛛	入力データが変化した場合のみ出力します。

### 3-5. ソーシャルノード

e mail	電子メールを待ち受け、入力データもしくはイベントとし ます。
twitter	Twitter からのメッセージを待ち受け、入力データもしく はイベントとします。
e mail 🔽	入力データを設定された電子メールアドレスに送付しま す。
twitter	入力データを Twitter へ出力します。
google plus	Google plus に対する入出力を提供します。
google places	Google places に対する入出力を提供します。
google calendar	Google calendar に登録されている次のイベントを返します。

### 3-6. ストレージノード

tail	設定されたファイルの末尾を入力データとします。
file	入力データに示されるファイルを開き、その内容を出力し ます。



設定されたファイルにデータを出力します。

### 3-7. 分析ノード

sentiment	入力データをAFINN-111 単語リストを用いて感情分析(肯
	定的/否定的/中立)を行い出力します。

### 3-8. その他ノード

Q watch	ディレクトリまたはファイルの更新を監視し、入力イベン トとします。
feedparse	RSS/Atom を監視し、Web コンテンツの更新を入力イベン トとします。
exec	システムのコマンドを実行し、その出力を返します。
chart request	Google chart にグラフ描画をリクエストします。
chart response	Google chart のグラフ描画を出力します。

### 3-9. cloud ノード

azureiothub	入力データを Azure IoT Hub へ出力します。
orthe azureiothubregio	入力データに示されるデバイスを Azure IoT Hub へ登録します。

### 3-10. GatewayKit ノード

datasource T.
---------------

## 3-11. location $\mathcal{I} - \mathcal{F}$

worldmap	入力データを WEB 上にプロットします。(インターネット 接続が必要です。)
worldmap	Worldmap の出力 Web ページからイベントを受信します。
tracks	指定した数の前の場所に基づいてトラックを作成します。
google geocoding	Google geocording を用いて入力データをジオコーディン グ(住所情報と地理的座標の相互変換)し、出力します。
google directions	Google directions を用いて入力された出発地点と目的地点の経路を出力します。

## 3-12. dashboard $\checkmark-\texttt{F}$

button	ユーザーインターフェースにボタンを追加します。
dropdown	ユーザーインターフェースにドロップダウン選択ボックス を追加します。
switch	ユーザーインターフェースにスイッチを追加します。
slider	ユーザーインターフェースにスライダーウィジェットを追 加します。
123 numeric	ユーザーインターフェースに数値入力ウィジェットを追加 します。
ebe text input	ユーザーインターフェースにテキスト入力ウィジェットを 追加します。
date picker	ユーザーインターフェースに日付選択ウィジェットを追加 します。
colour picker	ユーザーインターフェースに色選択ウィジェットを追加し ます。
form	ユーザーインターフェースにフォームを追加します。

text abc	ユーザーインターフェースに編集不可能なテキストフィー ルドを表示します。
gauge 🕥	ユーザーインターフェースにゲージタイプのウィジェット を追加します。
chart 🗾 🗠 🗖	ユーザーインターフェースに入力値をグラフにプロットし ます。
audio out	ダッシュボードの音声またはテキストを音声で再生しま す。
notification	msg.payload をポップアップ通知または OK / Cancel ダイ アログメッセージとしてユーザーインターフェイスに表示 します。
ui control	ダッシュボードを動的に制御できます。
	テンプレートウィジェットには、任意の有効な html およ び角度/角度 ・材料指示を含めることができます。

## 3-13. Google ✓ – ド

google	Google calendar のイベント(予定の通知)の発生を待ち
calendar	受けます。
google c alendar	Google calendar に新しい予定(イベント)を登録します。

## 第4章 ノード操作サンプル

#### 4-1. 位置情報の可視化

- Node-RED にアクセスします。(第3章参照) http://192.168.254.254:1880/
- 2. 入力ノードパレットから Inject ノードをドラッグしてシートにドロップします。



3. 配置した Inject ノードをダブルクリックし、ノードの編集を行います。

RED		
<b>Q</b> ノードを検索	70-1	inject ノードを編集
~ 入力		削除 中止 完了
⇒ inject	timesta	✓ プロパティ
catch		✓ペイロード {}
status		
link		
) mqtt		€繰り返し なし ▼
http		<ul> <li>Node-RED起動時に実行</li> </ul>
websocket		◆名前 名前
) tcp		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
y udp		
ipc b		

名前欄に GPS と入力します。ペイロードの種類に JSON を選択します。また、ペイロ ード部の『…』を選択し、表示されるウィンドウに以下を設定します。



設定後、「完了」ボタンを押します。

4. 機能ノードから function ノードをドラッグしてシートにドロップします。

Node-RED	
Q ノードを検索	70-1
serial 👖	
~ 機能	
of function	
e { templete	of the second
delay delay	
comment	
http request	
tcp request	
-C switch	
A shares	

5. 配置した function ノードをダブルクリックし、名前欄に"modify\_location"を設定しま す。また、コード部に以下を設定します。設定完了後、『完了』ボタンを押します。

```
//var res = JSON.parse(msg.payload);
var res = msg.payload;
msg.payload = {
    name: "Tokyo sta.",
    lat: res.latitude,
    lon: res.longitude
};
return msg;
```

- Node-RED **q** ノードを検索 70-1 azureiothub 🛛 . azureiothubreg GPS ✓ GatewayKit iot datasource modify\_location location worldmap world map 🏼 📀 worldmap tracks google geocoding
- 6. location ノードから worldmap ノードをドラッグしてシートにドロップします。

配置した Inject ノード - function ノード、function ノード - worldmap ノード間を接続します。

RED	
<b>Q</b> ノードを検索	フロー1
azureiothub	
azureiothubreg	GPS GPS
✓ GatewayKit	
iot datasource	f modify_location
<ul> <li>location</li> </ul>	)
worldmap	
📀 worldmap 🖣	word map
🛛 🛜 🛛 tracks 😐	
google geocoding	

8. Deploy ボタンをクリックします。



Deploy が完了すると、Deploy ボタンの背景が赤から黒に変わります。

- ブラウザの別タブにて以下にアクセスを行います。 http://192.168.254.254:1880/worldmap/
- 10. Inject ノードの左部のボタンを押します。

Rode-RED	
Q ノードを検索	70-1
~ 入力	
😫 inject	GPS
catch	
status	(
ink 🖯	modify_location
mqtt	
http 🕒	
websocket	Connected 1

11. 別タブにて表示した地図の日本の東京駅にプロットされます。



#### 4-2. ビーコンデータへの位置情報付与

本項では IoT データ制御機能にて収集したビーコンデータに対して、LTE モジュール(NTT ドコモ/KDDI)または BWA モジュールを用いて取得した GPS 情報を付与するサンプルとなります。

1. 入力ノードパレットから ipc ノードをドラッグしてシートにドロップします。



配置した ipc ノードをダブルクリックし、ノードの編集を行います。
 Name に"device\_beacon"と入力し、Path に"/node-red/device\_beacon.sock"と入力します。
 す。尚、Abstract Flag をチェックしたままとします。
 入力完了後、完了ボタンを押します。

■< Sode-RED						-7 รีวีอา
<b>Q</b> ノードを検索	70-1 7	ipc-in ノードを編	集		◎ コンテキ:	ストデータ i
~ 入力		削除		中止 完了	V Node	
⇒ inject		> プロパティ				込みのため更新してくだ
catch	device_beacor	Name	device_beacon		~ Flow	
status		Path	/node-red/device_beacon.sock			2
🗦 link		Abstract Flag				データが存在しません
) mqtt		r iag	•		<ul> <li>Global</li> </ul>	
http						2 データが存在しません
websocket						
)) tcp						
🧳 udp						

<b>Q</b> ノードを検索	70- 1	フロー 2	フロー 3	70-4
χ change ij range φ	device_beacon			
tl sort	at Op	n		
e∋≘ batch o				
C html				

3. 機能ノードパレットから json ノードをドラッグしてシートにドロップします。

4. 機能ノードパレットから function ノードをドラッグしてシートにドロップします。



配置した function ノードをダブルクリックし、ノードの編集を行います。
 名前はビーコンデータと GPS 情報の2種類を扱う為、"並列設定"とします。
 後述のコードを入力後、完了ボタンを押します。

Node-RED					
<b>q</b> ノードを検索	70-1	フロー 2	70-	function ノードを編集	
serial 1				削除	中止 完了
~ 機能	device_beacon			> プロパティ	
f function				▶名前	
<pre>{ template }</pre>	() jsc	n		並列設定	-
රට delay p				F⊐−F	
				<pre>1 context.global.n = 2; 2 context.global.data = new Array(2); 3 return msg;</pre>	
comment		- (f - )			
http request					

また、コードは以下を設定します。

context.global.n = 2;
context.global.data = new Array(2);
return msg;

6. ストレージノードパレットから file ノードをドラッグしてシートにドロップします。



 配置した file ノードをダブルクリックし、ノードの編集を行います。 ファイル名は"/tmp/.gps\_posi.json"とします。また、名前は"READ GPD JSON"としま す。尚、出力形式は文字列からの変更はありません。 各項目入力後、完了ボタンを押します。

Q ノードを検索	70-1	フロー 2	70-	file in ノードを編	i集
e mail 🔶 🔺				削除	中止 完了
twitter	⇒ device_beacon				
e mail 🖂					
8 google plus				■ファイル名	/tmp/.gps_posi.json
google places	S jsoi	, <b>P</b>		➡ 出力形式	
google					- エラーイッセージを洋信(万換エード)
calendar				▶ 么前	
~ ストレージ		f 並列設定		<ul> <li>▲ 1 = 100</li> </ul>	READ GFS JSON
				注釈:「ファイ	(ル名」はフルバスを設定する必要があります。
La tail			_		
E file			() ile		
file file					

8. 機能ノードパレットから json ノードをドラッグしてシートにドロップします。



9. 機能ノードパレットから function ノードをドラッグしてシートにドロップします。



配置した function ノードをダブルクリックし、ノードの編集を行います。
 名前はビーコンデータと GPS 情報の2種類のデータをマージする為、"データマージ"とします。

後述のコードを入力後、完了ボタンを押します。

Rode-RED				
Q ノードを検索	70-1	フロー 2	70-	function ノードを編集
websocket				削除 中止 完了
tcp	evice_beacon			✓ プロパティ
udp 🛞				▶ 名前
o ipc 🔅	js	on		データマージ
aws mqtt ))				<pre>♪ □-ド 1 context.global.n;</pre>
hangouts				<pre>2 context.global.data[context.global.n] = msg.payload; 3</pre>
Watson IoT		● ƒ 並列設定 ●		<pre>5 var array = {}; 6* for (var i = 0; i &lt; context.global.data.length; i</pre>
serial N				<pre>7 array = Object.assign(array, context.global.c 8  }</pre>
~ 機能				9 msg.payload = array; 10 return msg; 11 ^ }
of function				
e { template				
elay p				
trigger				
comment				
http request				

また、コードは以下を設定します。

```
context.global.n--;
context.global.data[context.global.n] = msg.payload;
if (context.global.n === 0){
    var array = {};
    for (var i = 0; i < context.global.data.length: i++) {
        array = Object.assign(array, context.global.data[i]);
    }
    msg.payload = array;
    return msg;
```

11. 出力結果を確認する為、出力ノードパレットから debug ノードをドラッグしてシート にドロップします。



12. 以下の図のように各ノード間を接続します。接続完了後、デプロイボタンを押します。 これにより、ビーコンデータを Node-RED 側で受信した際に GPS 情報ファイルのデ ータをマージし、debug ノードにマージした情報が出力されます。



## 第5章 その他

## 5-1. Node-RED ログについて

Node-RED 自体のログについては、「Node-RED」→「ログ」タブから確認が行えます。

OpenBlocks <sup>®</sup> loT	閲覧したいログファイルをログ欄から選択し
ダッシュポード 基本 Node-RED ログ ファイル	ます。
GI 2015 - 2017 PlatHome Co., Lid. All rights reserved.	ログファイルの選択を行うと、対象ログファイ ルの末尾が表示されます。 また、ダウンロードボタンを押すことによりロ
DP         rodered log         Ø700-FK         B&E#5(1005/8/m)           RNNer 164.44 - Infol Starting fore         (Nor 16.44.4 - Infol Starting fore)         (Nor 16.44.4 - Infol Starting fore)           RNNer 164.44 - Infol Starting fore         (Nor 16.44.2 - Infol Starting fore)         (Nor 16.44.2 - Infol Starting fore)           RNNer 164.41 - Infol Starting fore         (Nor 16.47.1 - Infol Starting fore)         (Nor 16.47.1 - Infol Starting fore)           RNNer 164.7 1.1 - Infol Starting fore         (Nor 16.47.2 - Infol Stopper fore)         (Nor 16.47.2 - Infol Stopper fore)           RNNer 164.7 1.1 - Infol Starting fore         (Nor 16.47.2 - Infol Stopper fore)         (Nor 16.47.2 - Infol Stopper fore)           RNN 164.7 1.1 - Infol Starting fore         (Nor 16.47.2 - Infol Stopper fore)         (Nor 16.47.2 - Infol Stopper fore)           RNN 164.7 1.1 - Infol Starting fore)         (Nor 16.47.2 - Infol Stopper fore)         (Nor 16.47.2 - Infol Stopper fore)           RNN 164.7 1.1 - Infol Stopper fore)         (Nor 16.47.2 - Infol Stopper fore)         (Nor 16.47.2 - Infol Stopper fore)           RNN 164.7 1.1 - Infol Stopper fore)         (Nor 14.47.2 - Infol Stopper fore)         (Nor 14.47.2 - Infol Stopper fore)           RNN 164.7 1.1 - Infol Stopper fore)         (Nor 14.42.2 - Infol Stopper fore)         (Nor 14.42.2 - Infol Stopper fore)           RNN 164.7 1.1 - Infol Stopper fore)         (Nor 14.42.2 - Infol Stoper fore)         (Nor 14.42.2 - Infol Stop	グファイルのダウンロードが行えます。

#### 5-2. Node-RED **へのノー**ド追加

Node-RED に対して対応するファイルをアップロードすることにより、Node-RED に対し てノードを追加することができます。

ノード用のファイルアップロードは「Node-RED」→「ファイル」から行えます。

penBlocks®	Tol		
осполоско ая-к ал	Node-RED	ログイン (D: admin (相称: スーパーユーザー) <u>マイベーシ ログアウト</u>	N
Node RED 07	ファイル		ル
nodeファイル			P
icons icons icons ico-in.html ico-in.html ico-in.js ico-in.html		*	P
ipc-out.js t4d-keymapper.html t4d-keymapper.js			
			Ν
( 夏新)(思味)(奥行権付	5)		ル
アップロード アップロード	ファイルを運訳 選択されていません アップロード		を
一話アップロード			

node ファイル アップロード: Node-RED のアップロード用に単体のファイ ルをアップロードが行えます。 アップロードするファイルを選択後、対応する アップロードボタンを押してください。 一括アップロードボタンを押してください。 一括アップロードが行えます。複数のファイ ルをアップロードが行えます。複数のファイル をアップロードする場合、tar 形式に圧縮した ファイルを指定してください。 アップロードするファイルを選択後、対応する アップロードボタンを押してください。

アップロードしたノードファイルを Node-RED に反映させる場合には、Node-RED 自体の 再起動が必要となります。後述のダッシュボードからプロセスの再起動を推奨します。

Node-RED のノードの作成方法については以下を参照してください。

https://nodered.jp/docs/creating-nodes/first-node

また、本項によるノード追加の場合、ノードの公開は行われないため package.json の作成 等は不要です。

尚、フォーマットエラー等のファイル(html 及び js ファイル)をアップロードした場合、読 み込みがスキップされ、対象ノードは表示されません。

#### 5-3. Node-RED の Config 編集

「Node-RED」タブにて使用 Config 設定にてユーザー定義コンフィグを選択している場合、「編集」タブが表示されます。

この「編集」タブから Node-RED 自体のコンフィグ情報を編集できます。



#### <u>編集</u>

#### 編集対象ファイル:

"setting.js"を選択してください。
ファイルを選択した段階で、現状使用するコン
フィグファイルの内容がロードされます。
最終保存時のユーザーコンフィグを読み込み
たい場合は、「Load(ユーザーコンフィグ)」ボ
タンを押してください。
また、システム作成のコンフィグを読み込みた
い場合は、「Load(システムコンフィグ)」ボタンを押してください。

※本項目の編集については十二分に理解があ る方を前提としています。

コンフィグファイルの編集後、保存ボタンを押すことにより反映されます。 また、編集中のコンフィグファイルの内容をエクスポートボタンを押すことで保存するこ とが可能です。

### 5-4. Node-RED のプロセス状況について

Node-RED を使用設定にしている場合、ダッシュボードにて Node-RED 自体のプロセス状況を確認することができます。

プロセス状況に応じて、「起動」・「停止」・「初期化」ボタン等が表示されます。

「初期化」ボタンは Node-RED のフロー状況等が削除されますので、ご注意ください。

ュボード サービス システム	ネットワーク	メンテナンス	拡張	AirMana
システム全体の概要更新				
ハードウェアリソース				
メインメモリ : 370 MB / 880 MB ストレージ : 1093 MB / 5337 MB				
ネットワーク (設定)				
FQDN : obsiot.example.org ゲートウェイ : 172.16.7.1 IPアドレス (eth0) : 172.16.7.135 IPアドレス (wlan0) : 192.168.254.254				
プロセス状況 (Node-RED) 停止)				

#### 5-5. Node-RED の HTTPS 化

Node-RED のダッシュボードへのアクセスは通常では HTTP 通信となります。

セキュリティ強化のため HTTPS 通信へと変更を行いたい場合には、以下の方法を実施する ことで切り替え可能となります。

※HTTPS 通信に切り替えた場合、WEB UI の Node-RED タブ内のリンクボタンからのリ ンクは HTTP 前提となっている為、リンク切れが発生します。

1. 証明書の作成

本項では/var/tmp ディレクトリ配下に自己署名証明書を作成しています。正式なもの として使用するには推奨いたしません。



2. WEB UI の Node-RED タブ内の使用 Config 設定をユーザー定義コンフィグに設定 し保存します。

Ор	oenBlocks®	ToT	
ダッシュ	レポード 基本	Node-RED IOTデータ	
Ν	Node-RED ログ	ファイル	
_	Node-RED (?)		
1	使用設定	● 使用する ◎ 使用しない	
,	ポート番号	(1880	
ſ	ログイン認証設定	○ 使用する ● 使用しない	
1	使用Config設定	○ 本システムコンフィグ ● ユーザー定義コンフィグ	
-	<b>操作</b> 保存 リンク エクス	( -+	

 Node-RED カテゴリの編集タブにてコンフィグを修正します。 編集対象ファイルを setting.js を選択し、テキストボックス内を変更します。 変更内容は"var fs"部のコメント解除及び https ディレクティブのコメント解除・ パス修正となります。 ※変更前

// The `https` setting requires the `fs` module. Uncomment the following
// to make it available.
//var fs = require("fs");
// The following property can be used to enable HTTPS
// See http://nodejs.org/api/https.html#https_https_createserver_options_requestlistener
// for details on its contents.
// See the comment at the top of this file on how to load the `fs` module used by
// this setting.
11
//https: {
// key: fs.readFileSync('privatekey.pem'),
// cert: fs.readFileSync('certificate.pem')
/},

#### 4. ※変更後



5. 変更完了後、保存ボタンを押すことで HTTPS 化は完了となります。

OpenBlocks IoT Family向け Node-RED スターターガイド (2018/08/30 第2版)

ぷらっとホーム株式会社

〒102-0073 東京都千代田区九段北 4-1-3 日本ビルディング九段別館 3F