

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



Ver.2.1.0

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
第2章 Node-RED 事前準備 5
2-1. Node-RED の使用設定について 5
2-2. Node-RED のブラウザフィルタについて5
2-3. パケットフィルタについて
第3章 Node-REDの簡易説明7
3-1. Node-RED 画面構成
3-2. ノード種類
3-2. Input ノード
3-3. Output ✓ — F
3-4. function $\mathcal{I} - \mathcal{F}$
3-5. social $\mathcal{I} - \mathcal{F}$
3-6. storage $\prime - F$
3-7. analysis ノード
3-8. advanced $\checkmark - F$
3-9. cloud ノード
3-10. GatewayKit ノード13
3-11. location $\mathcal{I} - \mathcal{F}$
3-12. Google ノード
第4章 ノード操作サンプル15
4-1. 事前準備
4 <sup>-</sup> 2. 単純なデバッグ
4-3. 温度データをグラフに表示する

## 第1章 はじめに

本書は、OpenBlocks IoT Family に搭載されている Node-RED の使用方法を解説しています。

搭載している Node-RED はデータ収集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 の使用設定について

WEB UI の「拡張」→「Node-RED」タブから使用設定を実施してください。

репыоскъ	101	ログイン ID: admin (機果: スーパーユーザー) <u>マイページ</u> ロク
×±ポード サービス	システム ネットワーク メンテナ	2.2. 和34 技術機構
スタリナト編集 コマンド実行	SMS_72/F7RFF SMS/Eff PD	Stubseriber node red
ログ		
P9	(選択したものを表示します ▼	
node red (?) 使用¥©定	● 使用する ○ 使用しない	
ボート番号	(1880	
ログイン問証数定	● 使用する ◎ 使用しない	
ユーザー名		
パスワード		
操作		

Node-RED を使用する場合には、"使用する" に設定し保存ボタンを押してください。

また、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。

#### 2-2. Node-RED のブラウザフィルタについて

WEB UI の「システム」→「フィルター」タブにて Node-RED のフィルターを開放してください。

Open	Blocks®	loT		ログイン ID: admin (権)	限: スーパーユーザー	) <u> </u>
ッシュボード	サービス	システム	ネットワーク	メンテナンス	拡張	技術情報
001*#44	オスコーナ無外辺辺	金ー保存者が、水	博玉/ おまい			
33HZ#XIIC	a and a second second	あこ1米1チャランセ	IT I NEEVI			
基本	詳細	パスワー	-ド フィルター	- SSH関連	74*-	-92
WEB2-5-	- ファイル管理	512	SIN SIN			
フィルター	-関放設定 🗸 東橋	動がます、ファイルの開始を	とまた有効にする (2)			
	Distance - His	BUTE OF THE SHEAR	a			
SSH		(	🖲 有効 🔍 無効			
node red		(	● 有効 ● 無効			
操作						
(早721)						

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。

#### 2-3. 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 Input node 等、リモートからの接続を待ち受けるノードを使用する場合、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、WEB UI の「拡張」→「スクリプト編集」タブから、iptable を操作するスクリプトを設定してください。

## 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 WLAN 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)



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。

✓ input	∨ output	<ul> <li>function</li> </ul>	<ul> <li>social</li> </ul>	✓ analysis	~ cloud
📫 inject	debug	f function	e mail	sentiment	Azure_IoT_Hub
catch	link 🗦	🖣 { template 🏚	y twitter	~ advanced	azureiothub 🖗
status	(mqtt )	රට delay 🔶	🛛 e mail 🔛	Q watch	● 💦 azureiothubregi
🗧 link	http response 📀	trigger	o twitter 🍸	E feedparse	✓ GatewayKit
)) mqtt	websocket	comment	google plus	exec	iot
🔷 http	tcp	http request	google places	chart request	datasource
websocket	udp 🕴	tcp request	google	chart	~ location
)) tcp	hangouts	-< switch	calendar	response	google
)) udp	Watson IoT	φ 🗶 change 🖕	✓ storage		geocoding
hangouts	iot dev	oij range o	🕒 tail 👂		google directions
💮 Watson IoT 🕞	ibmiot 🔅	<b>●</b> ∎≣ split ●	file 🛉		u. Coordo
iot dev	serial 🔟	oten join o	a. amazon s3		♥ Google
N IPC		• 1.2 csv •	e file 🕒		calendar
ibmiot 💿		💽 html 👳	amazon s3 a.o		google
👖 serial 🖕		📢 json 🖸	o amazon s3 🧕		calendar
		🔹 🗤 📦			
		o 🕤 rbe 🖕			

#### 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。

	1. シート	2.	デプ	°ロイ	
Node-RED     A filter nodes	Flow 1	+	info	Deploy d	ebug
∽ input		<b>^</b>		~	
⇒ inject			5.	表示情報	報切替
status					
k link					
) mqtt	timestamp				
websocket			_		
) tcp					
) udp	3. 7				
Watson IoT					
iot dev					
	1	+			
		- 0 +			
4.	ノードパレット	6	;. /	ノード情報	報

#	項目	説明
1	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. Input 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
	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
Catch	くはイベントとします。
etatue	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
status	イベントとします。
	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 y	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
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。
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 による接続を待ち受け、入力データもしくはイ
http websocket	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 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。
websocket	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 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。 TCP による接続を待ち受け、入力データもしくはイベント
http websocket i) tcp	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 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。 TCP 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。
http websocket tcp	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 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。 TCP 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。 UDP による接続を待ち受け、入力データもしくはイベント
http       websocket       tcp       udp	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 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。 TCP 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。 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。
http       websocket       tcp       udp	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 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。 TCP 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。 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。 Google hangouts からのメッセージを待ち受け、入力デー

Watson IoT	Watoson IoT からの device command を待ち受け、入力
a Maison of P	データもしくはイベントとします。*1
the int day	Watoson IoT からの device command を待ち受け、入力
in the second	データもしくはイベントとします。*1
	UNIX ドメインソケット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。
the ibmist	Watoson IoT からの device command を待ち受け、入力
	データもしくはイベントとします。*1
	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。

\*1: Input ノード Watson IoT と iot dev, ibmiot は、同等の機能を提供します。

### 3-3. Output 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。
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。*1
iot dev	入力データを Watson IoT へ出力します。*1
ibmiot 🌼	入力データを Watson IoT へ出力します。*1



\*1: Output ノード Watson IoT と iot dev, ibmiot は、同等の機能を提供します。

### 3-4. function $\mathcal{I} - \mathcal{F}$

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。
<b>join</b>	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。
o {} json o	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。
o 🗾 rbe 🛛	入力データが変化した場合のみ出力します。

### 3-5. social 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. storage $\mathcal{I} - \mathcal{F}$

tail 🕴	設定されたファイルの末尾を入力データとします。
file 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。
amazon s3	Amazon S3 からデータを取り込みます。
file	設定されたファイルにデータを出力します。
amazon s3 a	Amazon S3 に対する入出力を提供します。



#### 3-7. analysis $\mathcal{I} - \mathcal{F}$

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

### 3-8. advanced $\checkmark - \checkmark$

२ 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

Azure_IoT_Hub	入力データを Azure IoT Hub へ出力します。
azureiothub	入力データを Azure IoT Hub へ出力します。
o <mark>_</mark>	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



入力データをダッシュボードアプリケーションへ出力しま	
す。	

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

google	Google geocording を用いて入力データをジオコーディン
geocoding	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. 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 calendar	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

本項では、BLE センサー (ALPS 電気社製 IoT Smart Module) のデータを受け Node-RED で処理します。

#### 4-1. 事前準備

BLE センサーのデータを受け Node-RED へ送る設定を行います。

1. WEB UI の「サービス」→「BT 関連」タブにて BLE デバイスの検出を行います。

	9770 4	ベットワーク メンテナンス	. 10.15	Parmanage
基本 BT開連	BLEメンテナンス	<b>VIII</b>		
BT & BLE(?)				
BTデバイス検出	検出			
BLEデバイス検出時間(秒)	(15			
BLEデバイス検出				
操作				
保存				
一覧				
デバイス番号	アドレス	ユーザーメモ	操作	

BLE センサーデバイの電源を入れ、「BLE デバイス検出」の検出ボタン①を押してくだ さい。

2. 検出された BLE デバイスの一覧から使用するデバイスを選択します。

<del>ボード</del> サービ	ス システム ネット	フーク メンテナンス 拡張	AirManage
12			
nx.			
##		11 TE	
		Several Se	
DI & DLL(:)			
BTデバイス検出	横出		
BLEデバイス検出時	<b>祖</b> (秒) (15		
BLEデバイス検出	<b>福出)</b> 18日	總了	
使用設定		内容	_
borbibol.	Device Name	(unknown)	
	Device Address	46:80:80:E1:E1:2B	
	Memo		)
-	Device Name	WATTCHECKER	
	Device Address	8CIDE:52:F3:ATTT	~
	Device Neme	EM848BI 702	
(1)	Device Address	D7:F3:D2:59:EC:91	
	Memo	2 Sensor 001	
	Device Name	FWM8BLZ02	
	Device Address	DB:C5:F4:87:62:F9	
	Memo		
 操作			

「使用設定」のチェックボックス①をチェックし、必要に応じ「Memo」②を記載、保存ボタン③をクリックします。

3. 選択したデバイスが、一覧に表示されていることを確認します。

OpenBlocks	<sup>®</sup> [IoT]	ログ・	イン ID: admin (横限:ス	-バーユーザー) <u>マイイ</u>	<u>(一ジ ログアウト</u>
<b>ダッシュボード</b> サービス	۶ステム ۸۰۱	ワーク メンテナンス	拡張	AirManage	技術情報
設定を保存しました。					
最本 BT開連	BT編集 BL	Eメンテナンス 秋田			_
BT & BLE(?)					
BTデバイス検出	検出				
BLEデバイス検出時間(秒)	(15	$\supset$			
BLEデバイス検出	検出				
操作					
保存)					
一覧					
デバイス番号	アドレス	ユーザーメモ	操作		
dev_le_0000001	D7:F3:D2:59:EC:91	Sensor_001	削除		

ー覧に選択したデバイスが表示されてい ること①を確認します。

 WEB UI の「サービス」→「基本設定」タブにて「データ収集」と「PD Handler BLE」 を使用する設定を行います。

0	pen	Blocks	loT			ログイン IC	t: admin (權限: ス	-バーユーザー) <u>マイバ</u>	<u> ログアウト</u>
ダッミ	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
	基本	вт関連	BLEX	マテナンス 秋日	R				
	BT								
	使用設定			<ul> <li>使用する</li> <li>使用する</li> </ul>	目しない				
	データ	収集							
	データ収	策	(1)	) ® 使用する © 使用	目しない				
		PD Handler BLE	$\overline{2}$	) 🖲 使用する 🔍 使用	目しない				
		PD Handler PLC C	lient	◎ 使用する ® 使用	目しない				
		PD Handler PLC S	erver	◎ 使用する ® 使用	目しない				
		追加Unixドメインン	ノケット数	0 •					
		ユーザーHandler使	用設定	◎ 使用する ® 使用	目しない				
	PD自動車	再起動設定		◎ 使用する ® 使用	目しない				
	操作								
3	(保存)								

「データ収集」①と「PD Handler BLE」
 ②を"使用する"に設定し、保存ボタン③を
 押してください。

5. WEB UI の「サービス」→「収集設定」タブの「送信先設定」にて Node-RED へ送る 設定を行います。

本体内(local)	● 使用する ● 使用しない	
デバイス一括設定	一括有効)一括無効)	
PD Exchange	◎ 使用する ⑧ 使用しない	
Amazon Kinesis	● 使用する ● 使用しない	
AWS IoT	● 使用する ● 使用しない	
Watson IoT(Device)	○ 使用する ⑧ 使用しない	
Watson IoT(Gateway)	◎ 使用する ® 使用しない	
MS Azure Event hubs	◎ 使用する ® 使用しない	
MS Azure IoT Hub	◎ 使用する ® 使用しない	
Toami for docomo(T4D)	◎ 使用する ® 使用しない	
MQTTサーバ	◎ 使用する ® 使用しない	
WEBサーバ(PLAIN)	● 使用する ● 使用しない	
node red(NRED)	● 使用する ─ 使用しない	
インターバル[sec]	30	
有効時間[sec]		
ソケットパスプレフィックス	(tmp/node-red/	
デバイス一括設定	一括有効 ) 一括無効 )	

「node red (NRED)」①を"使用する"に設 定します。

他のパラメータはデフォルト値のままとし てください。

保存ボタンは次のステップでクリックしま す。

6. WEB UI の「サービス」→「収集設定」タブの「デバイス情報送信設定」にて「デバ イス番号」dev\_le\_0000001 のデータを Node-RED へ送る設定を行います。

デバイス番号	dev_le_0000001
送信対象	① ● 送信する ● 送信しない
アドレス	34:C7:31:FF:E6:20
ユーザーメモ	Sensor_001
センサー信号強度[dbm]	0
取得時間間隔[ms]	(5000
送信先設定	Iocal PD KINESIS AWSIOT Watson IoT(Device)     Watson IoT(Gateway) EVENTHUB IoTHub T4D MQTT     PLAIN NRED
揭作	2

「送信対象」①を"送信する"に設定します。

「送信先設定」の"NRED"チェックボック ②をチェックします。

他のパラメータはデフォルト値のままと してください。

保存ボタン③をクリックします。

#### 7. WEB UI の「サービス」→「収集ログ」タブで動作を確認します。

収集ログ	
コグ選択	(pd-handler-stdout.log v ダウンロード) 🗌 自動更新(30秒間隔)
2016-10-03 17:28:44 4011 [INFO] pd-hand 2016-10-03 17:28:49,702 [INFO] pd-hand 2016-10-03 17:28:49,702 [INFO] pd-hand 2016-10-03 17:28:59.5141 [INFO] pd-hand 2016-10-03 17:28:51.28:11NFO] pd-hand 2016-10-03 17:28:51.28:31 [INFO] pd-hand 2016-10-03 17:28:51.28:31 [INFO] pd-hand 2016-10-03 17:28:54.7461 [INFO] pd-hand 2016-10-03 17:29:04.6950] [INFO] pd-hand 2016-10-03 17:29:04.6950] [INFO] pd-hand 2016-10-03 17:29:04.6950] [INFO] pd-hand 2016-10-03 17:29:04.6950] [INFO] pd-hand 2017:29:06-948-0900" "dataIndex": 0) "gets 54.999999999999999999999999999999999999	<pre>lier - hci0 down ▲ lier - hci0 up lier - colus startScanning lier - connectAndSetUp lier - disconnectAndSetUp lier -</pre>
'deviceld":"34c731ffe620","memo":"Senso 3117:29:08.892+0900","dataIndex":0,"pre mperature":27.46,"uv":0,"ambientLight":10	v_001","Ime":"2016-10- sesure":1008.967897589074,"humidith":57.28125,"1 01.12245929821012,"day":1,"month":10,"year":15}
'deviced'''34c731ffe20'',"memo"''Senso 3T17:29:08.892+0900","dataIndex'':0,"pre mperature"27:46,"uv":0,"ambientLight":10	r_001*,"ime*":2016-10- ssoure*1008.057897589074,"humidity*:57.28125,"t 01.12245929821012,"day*:1,"month*":10,"year*:15}
(device)(1***346620****memo***Senso 3171-29:08:29:0900**(datalndex**0);pre- mperature**27:46,*uv**0,*ambientLight**1( 収集ログ	r_001",10me";2016-10- sesure"1008.967897589074,"humidity":57.28125,1 01.12245929821012,"day":1,"month":10,"year::15)

「 ロ グ 選 択 」 に て 「pd-handler-stdout.log」を選択し

ます。

「{"deviceId": 」で始まる JSON 文 字列があることを確認します。

「 ロ グ 選 択 」 に て 「pd-emitter-lite.log」を選択しま す。

この時点では、Node-RED 側の UNIX ドメインソケットが用意され ていないため、接続エラーが発生し ています。

#### 4-2. 単純なデバッグ

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



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



OpenBlocks IoT	× Node-RED : 172.16.7.161 >	<		ß	- 0	×
← → C ☆ ③ 172.1	<b>6.7.161</b> :1880/#				\$	:
🔢 アブリ 🐵 Norse Attack Map	p					_
RED				-/ Dep	oloy 👻	
Q filter nodes	Flow 1		+ 1	nfo	debug	
) udp 🔶 🔦			-			
hangouts						
🐡 Watson IoT						
The lot day						
TH IPC P	<b></b>	9				
ibmiot 😐		msg.payload				
👖 serial 🖡		•				
∽ output						
debug						
link						
(mqtt )						
http response						
websocket			*			
	•		- 0 +			
bledevices.json     bledevices.json	^				すべて表示	×

4. IPC-in node と debug node を結線します。

5. IPC-in node をダブルクリックし、Path と Name を設定し、Done ボタンをクリック します。

OpenBlocks IoT ×	Rode-RED : 172.	16.7.161 ×		@ – 🗆 ×
← → C ☆ ③ 172.16.7.1	<b>61</b> :1880/#			☆ :
-C Node-RED			-	🖌 🗧 Deploy 👻 📃
Q filter nodes	Edit IPC-in node	•	info	debug
udp 🔶 🔶		Cancel Done	Node	<u> </u>
P hangouts			Name	dev_le_0000001
Watson InT	🕈 Path	/tmp/node-red/dev_le_0000001.sock	Туре	IPC-in
UT WARSON TO T	Tania		ID	70ae443e.5b8eec
iot dev	E TOPIC	Торіс	Propertie	s
₩ IPC ibmiot serial	Name Name	dev_le_0000001	Listens for c process, usi (on Windows message is text received	communication from another ng a UNIX-domain socket or s) a named pipe. A generated for each line of 1.
∽ output			The node lis	tens at the specified path:
debug			On Unix location and mus deployer     On Winc in the fo which m is deploy	& Linux, this is a filesystem which must be writeable, st not exist when the flow is d duous, this is a named pipe, rm \Lybpe\pipe\pipename, ust not exist when the flow yed
I bledevices.json				すべて表示 X

ここで、Path は、4-1 事前設定の1 に図示される「ソケットパスプレフィックス」と 同2に図示される「デバイス番号」に .sock を付加したものとします。

```
/tmp/node-red/dev_le_0000001.sock
```

Name は、何でも構いませんが本例では dev\_le\_0000001 とします。 Topic は、空欄のままとします。 6. Deploy ボタンをクリックします。

OpenBlocks IoT	× Node-RED : 172.16.7.161	×			₿ - □ ×
← → C ☆ ③ 172.1	16.7.161:1880/#				☆ :
🔢 アブリ 🐵 Norse Attack Ma	ι <b>ρ</b>				
Node-RED					- Deploy -
Q filter nodes	Flow 1		+	info	debug
) udp 🔶 🍝			-	Node	
nangouts				Name	dev_le_0000001
Watson IoT				Туре	IPC-in
Watson for				ID	70ae443e.5b8eec
iot dev				Propertie	s
ibmiot	H dev_ie_0000001	msg.payload		Listens for o process, usi (on Window message is text receive	communication from another ing a UNIX-domain socket or s) a named pipe. A generated for each line of d.
output     debug     dink     mqtt     mqtt				<ul> <li>On Unix location and mu- deploye</li> <li>On Wini in the for which m is deploy</li> </ul>	k.s. a line specifiely pain. k. & Linux, this is a filesystem which must be writeable, st not exist when the flow is d dows, this is a named pipe, rm \\.pipe\pipename, nust not exist when the flow yed
websocket	•		• •		
× ×			- 0 +		
I bledevices.json	^				すべて表示 ×

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。

 ノード表示をデバック表示に切り替え(①をクリック)、debug node を active にしま す(②をクリック)。

□ OpenBlocks IoT × Set Node-RED : 172.16.7.161 ×		保 — □ ×
← → C △ ① 172.16.7.161:1880/#		☆ :
デブリ 🐵 Norse Attack Map		
■ <p node-red<="" th=""><th></th><th>Teploy 👻</th></p>		Teploy 👻
Q filter nodes Flow 1	+	info debug ()
dbu (db	Ê	all flows current flow
22 hangouts		10- 03T17:45:22.528+0900","dataIndex":
💮 Watson IoT 🏮		2016/10/3 17:45:48 c97ab2c2.6de76
🔅 iot dev o		{"deviceId":"34c731ffe620","memo":"
		10- 03T17:45:22.724+0900","dataIndex":
🔅 ibmiot		2016/10/3 17:45:48 c97ab2c2.6de76
If serial		<pre>"msg.payload : string [284] {"deviceId":"34c731ffe620","memo":"\$</pre>
		10-
<ul> <li>output</li> </ul>		2018/10/3 17:45:48 _c07eb2c2 8de78
debug 🗐		msg.payload : string [257]
- link 3		{"deviceId":"34c731ffe620","memo":"
		03T17:45:23.510+0900","dataIndex":
( mqtt ))		2018/10/3 17:45:48 c97ab2c2.6de76
http response		"sg.payload : string [270] {"deviceId":"34c731ffe620","memo":"
websocket	-	10-
	•	•
I bledevicesjson ^		すべて表示 X

デバック表示にビーコンから得られたデータが表示されます。

### 4-3. 温度データをグラフに表示する

IoT datasource ノードを用いて、BLE センサーから取り込まれる温度データをグラフ表示 します。

- Node-RED にアクセスします。(第3章参照) <u>http://192.168.254.254:1880/</u>
- 2. 下図に示す通り IPC-in node と json node, function node, iot datasource node, debug node をそれぞれドラッグしシートにドロップします。



3. 下図に示す通り IPC-in node と json node, function node, iot datasource node, debug node の間を結線します。



4. IPC-in node をダブルクリックし、Path と Name を設定し、Done ボタンをクリック します。

					ಔ – □	×
OpenBlocks loT X	Node-RED : 172.16.7.16	51 × \				٦.
← → C ∩ 0 172.16.7.16	61:1880/					_ :
Norse Attack Map						
■< Node-RED					Deploy 🔻	
Q filter nodes Flo	ew 1 Edit IPC-in node	e		info	debug	
udp			Cancel Done	Node		
				Туре	IPC-in	
nangouts	👖 🕈 Path	/tmp/node-red/dev_le_000000	1.sock	ID	a2959c9c.0f2d5	
Watson IoT	Topic	Tapia		Properties		
iot dev	as topic	торіс		Listens for cor	mmunication from a	nother
	Name	dev_le_0000001		process, using	a UNIX-domain so	cket or
the line of the li				(on Windows) message is ge	a named pipe. A enerated for each lir	ne of
				text received.		
II serial				The node liste	ns at the specified	path:
✓ output				On Unix & location with the second sec	Linux, this is a files hich must be writea	ystem
				and must	not exist when the f	1ow is
depuĝ				On Windo	ws, this is a named	pipe,
link				in the form	1 \\.\pipe\pipename t not exist when the	≢, e flow
e mqtt ()				is deploye	d	
http response						
websocket	_					
🗊 bledevices.json \land					すべて表示	×

ここで、Path は、4・1 事前設定の1 に図示される「ソケットパスプレフィックス」と同2に図示される「デバイス番号」に .sock を付加したものとします。

```
/tmp/node-red/dev_le_0000001.sock
```

Name は、何でも構いませんが本例では dev\_le\_0000001 とします。 Topic は、空欄のままとします。 5. JSON node をダブルクリックし、適当な Name を設定し、Done ボタンをクリックし ます。



特に指定が無い限り OpenBlocks IoT ファミリーに標準搭載されているデータ取り込 みアプリケーション (PD-Handler) の出力フォーマットは JSON 形式です。

従って IPC-in node を用いて PD-Handler のデータを取り込み Node-RED でデータを 処理する場合、JSON node は必須となります。  function node をダブルクリックし、適当な Name を設定、Function として次の JavaScrip を記述し、Done ボタンをクリックします。

```
/* ① */
if(msg.payload.temperature == null) {
    return null;
}
/* ② */
var date = new Date(msg.payload.time);
```

msg.payload.tstamp = date.getTime();

return msg;

OpenBlocks IoT ×	x 🔀 Node-RED : 172.16.7.161 X 🔎 Zashboard X	63 – 🗆 X
← → C ① ① 172.16.	5.7.161:1880/#	☆ :
🎹 アブリ 💿 Norse Attack Map		
RED		🚽 🗧 Deploy 👻 🚍
Q filter nodes	Flow 1 Edit function node	info debug
✓ input	Cancel Done	all flows current flow
inject catch status ink ink ink ink ink ink	Name Time to Timestamp	2016/108 9-45.24 debug mg.psyled: Objett ("devicedi" 34c73116620", "memor" "Sensor_001", "time", "2016-10-05T08 44:54.288-09900", "dataIndec": 243, "pressure", 1016-1995198748761, "humidity", 54.71875, "temperature" 27.76, "uv": 0, "ambientLight", 126.40307412276265, "day", 1, "month", 10, "year", 15 }
<ul> <li>websocket</li> <li>tcp</li> <li>udp</li> <li>hangouts</li> </ul>		2010/105.8-44-4-bebg msg.psylad: Object ("devicel" 34C731ff6520", "memo" "Sensor_001", "time", "2016:1-0-0570:8-415.46;77-0500", "dataIndex": 244, "geoMagneticX"; 44.55, "geoMagneticX", 116.5, "geoMagneticZ", -110.24; "accetX",
<ul> <li>Watson IoT</li> <li>iot dev</li> </ul>	See the Info tab for help writing functions.	0.041015625, "accelY": -0.008056640625, "accelZ": -1.00048828125, "ms": 0, "second": 1 "minute": 30 "hourd": 0
		3
I bledevices.json	^	すべて表示 X

- ALPS 電気社製 IoT Smart Module は、温湿度等の環境データと加速度等のモーションデータを別々のタイミングで送信するため、温度情報を含まないデータ (msg)をブロックします。
- ② データに含まれる RFC3339 形式のタイムスタンプ(msg.payload.time)から、iot Datasource node で使用する UNIX 形式のタイムスタンプに変換します。

 Iot Dataresource node をダブルクリックし、Name として dev\_le\_0000001を設定、 Timestamp Field として msg.payload.tstamp、Data Field として msg.payload.temperature を設定し、Done ボタンをクリックします。

OpenBlocks IoT ×	Se Node-RED : 172.16.7.161 ×		₿ - □ ×
← → C ☆ ③ 172.16.7.	I61:1880/		:
🎹 アプリ 🐵 Norse Attack Map			
-C Node-RED		-1	Deploy 👻 🗮
Q filter nodes	ow Edit iot-datasource node	info	debug
) udp 🔶	Cancel Done	Node	· · · · · · · · · · · · · · · · · · ·
22 hangouts		Туре	iot-datasource
	Name         dev_le_0000001	ID	57da9e4d.25713
Watson IoT	<ul> <li>Disable subcomponent discovery</li> </ul>	Properties	
iot dev	Timestamp Field	IoT Gateway	Kit Datasource node.
	msg.payload. tstamp	Configuratio	n
ibmiot	Data Field	Disable subc	e datasource will not
∬ serial	msg.payload. temperature	and split it into	subfields. For
∽ output		example, if yo something like	our data format looks e this
debug		msg.payload tstamp: 1	= { 438637044000.
link 🔶		data: { x: 3.14	
mqtt ()		y: 1.41	,
http response		}	
websocket		harden it	
		having disc	overv enabled will allow
bledevices.json	×		すべて表示 X

8. Debug node をダブルクリックし、適当な Name を設定し、Done ボタンをクリックします。

OpenBlocks IoT	× 🖉 🛚	Node-RED : 172.16.7.16	1 x			☞ - □	×
← → C ☆ ③ 172.1	6.7.161:18	380/					:
🎹 アプリ 🐵 Norse Attack Mag	р						
- Node-RED					-	Deploy 👻	
Q filter nodes	Flow 1	Edit debug nod	e		info	debug	
udp 👌				Cancel Dor	Node		<b>^</b>
					Туре	debug	
nangouts	۲.,	i≣ Output	👻 msg. payload		ID	4f68afbf.c3fad	
Watson IoT		<b>&gt;⊄</b> to	debug tab	•	Properties		
iot dev     ibmiot     serial     output     debug     ink		<b>∿</b> Name	debug		The Debug not to the output u used to displa message prop the sidebar. T msg.payload Each message timestamp, m of property ch The sidebar c the options of corner.	orde can be connected of any node. It can be y the output of any perty in the debug tab he default is to display le e will also display the sstopic and the type iosen to output. can be accessed unde cop-down in the top rig	of y e r
mqtt b) http response *	4				The button to will toggle its o can de-clutter If the payload will be stringiff	the right of the node output on and off so yo the debug window. is an object or buffer ied first for display and	ou it d <sub>▼</sub>
bledevices.json     bledevices.json	^					すべて表示	×

9. Deploy ボタンをクリックします。



10. 新しいブラウザタブを開き Iot Datasource node のダッシュボードにアクセスします。

http://192.168.254.254:1880/dash/

🖸 OpenBlocks IoT x 🧟 Node-RED : 172.16.7.161 X 🚾 Dashboard x	儤	-		×
← → C û 172.16.7.161:1880/dash/			☆	:
📅 アプリ 🐵 Norse Attack Map				
node-red-contrib-graphs				
No dashboards available.				
I bledevices.json ^		<b>ৰ</b> ^	て表示	×

11. +Create New Dashboard をクリックし、Name として dev\_le\_0000001 を設定し、 Done をクリックします。

📷 OpenBlocks IoT X 😒 Node-RED : 172.16.7.161 X 🕍 Dashboard X	儤	-		×
← → C ① 172.16.7.161:1880/dash/			☆	:
🏥 アブリ 🐵 Norse Attack Map				
				ard
New Dashboard ×				
No dashboards available Name				
dev_ie_0000001				
Cancel Done 🗸				
				1
I bledevicesjson ^		すべて	表示	×

 +Create New Chart をクリックし、適当な Name を設定し、Plugin として Line/Area Char を選択、Add datasource として dev\_le\_0000001 を選択、X Axis Label とし て"Time"、Y Axis Label として"Deg"を設定し、Done をクリックします。

🔟 OpenBlocks IoT x 😪 Node-RED : 172.16.7.161 x k Dashboard x	æ	-		×
← → C ① 172.16.7.161:1880/dash/board/0			☆	:
III アブリ 🕲 Norse Attack Map				, 
node-red-contrib-graphs : dev. le_0000001				rt
Create New Chart ×				
Name				
Temperature				
Plugin				
Line/Area Chart +				
Datasources				
Add datasources +				
dev_le_0000001 X				
Request data between now and				
second(s) ago				
Maximum number of datapoints (leave blank for no limit)				
X Axis Label				
Time				
Y Axis Label				
Deg				
Cancel Done 🗸				<i>,</i> , -
J bledevices.json		đv	て表示	×

13. データ待ち状態の画面が表示されます。



14. データが取り込まれるとグラフが表示されます。



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

(2017/02/21 第1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