

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



Ver.2.1.1

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をご遠慮ください。
- ・ 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	6
第 3 章 Node-RED の簡易説明	7
3-1. Node-RED 画面構成	8
3-2. ノード種類	9
3-2. Input ノード	9
3-3. Output ノード	10
3-4. function ノード	11
3-5. social ノード	12
3-6. storage ノード	12
3-7. analysis ノード	13
3-8. advanced ノード	13
3-9. cloud ノード	13
3-10. GatewayKit ノード	13
3-11. location ノード	14
3-12. Google ノード	14
第 4 章 ノード操作サンプル	15
4-1. 事前準備	15
4-2. 単純なデバッグ	19
4-3. 温度データをグラフに表示する	22

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。

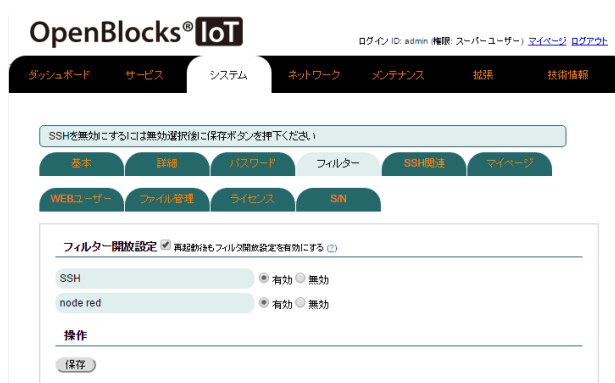


Node-RED を使用する場合には、“使用する”に設定し保存ボタンを押してください。

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。



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、`iptables` を操作するスクリプトを設定してください。

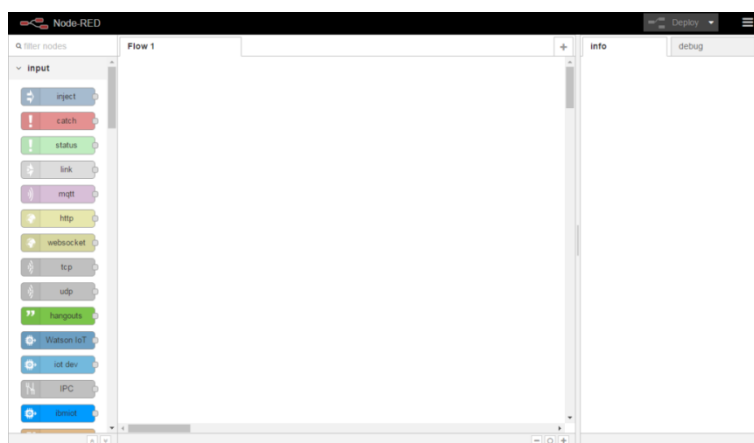
第 3 章 Node-RED の簡易説明

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。

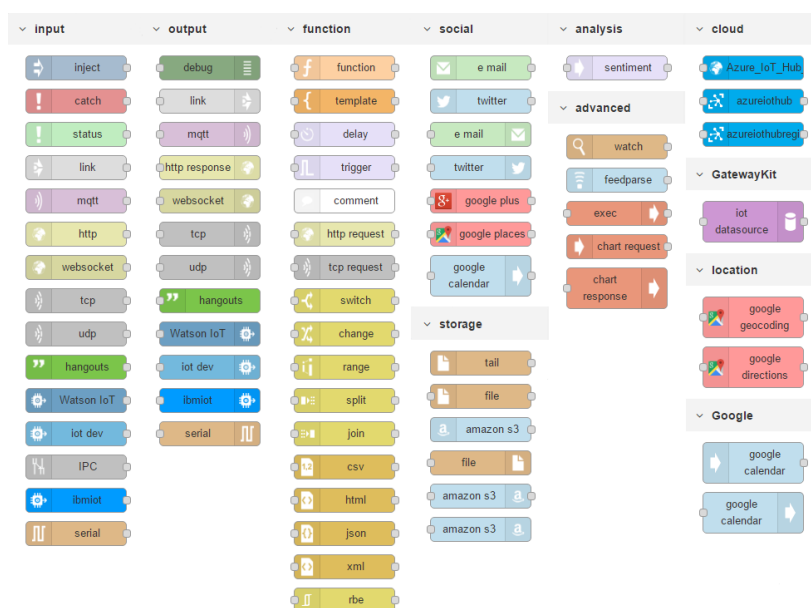
そのため、デフォルトでの WLAN 経由での Node-RED へのアクセスする為の URL は以下となります。

`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。



3-1. Node-RED 画面構成

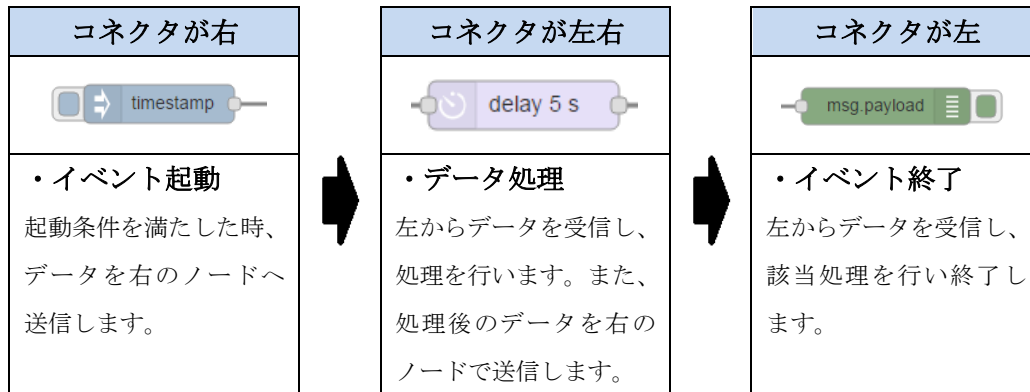
Node-RED の画面は以下のように構成されています。



#	項目	説明
1	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



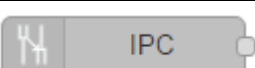
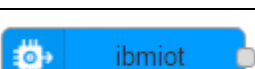
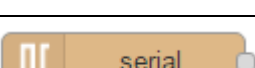
Node-RED では大きく分けて以下のようなコネクタ配置のノードがあります。



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

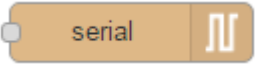
	ノードの左部のボタンを押すことでノードに設定された <code>timestamp</code> 等を入力データもしくはイベントとします。
	同じシート上のノードで発生したエラーを入力データもしくはイベントとします。
	同じシート上のノードのステータスを入力データもしくはイベントとします。
	いずれかの <code>link output node</code> の出力を入力データもしくはイベントとします。
	MQTT broker へ <code>subscrib</code> し、 <code>publish message</code> を待ち受け、入力データもしくはイベントとします。
	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、入力データもしくはイベントとします。

	Watson 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
	Watson 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
	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。
	Watson 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



	入力データをデバック情報として表示します。
	入力データをいずれかの Input link node へ出力します。
	入力データを MQTT broker へ publish します。
	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。
	入力データを Websocket サーバへ出力します。
	入力データを TCP サーバに出力します。
	入力データを UDP サーバに出力します。
	入力データを Google hangouts へ出力します。
	入力データを Watson IoT へ出力します。*1
	入力データを Watson IoT へ出力します。*1
	入力データを Watson IoT へ出力します。*1

	入力データをシリアルインタフェースへ出力します。
---	--------------------------








*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 ノード

	入力データを JavaScript で処理し出力します。
	入力データを整形して出力します。
	入力データを設定された時間後に出力します。
	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。
	フローにコメントを付けます。
	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。
	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。
	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。
	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。
	入力データのスケールを変更して出力します。
	入力データを設定される文字で区分して出力します。
	入力データを結合して出力します。
	CSV 書式のデータと JavaScript Object を相互変換します。
	HTML 書式のデータと JavaScript Object を相互変換します。
	JSON 書式のデータと JavaScript Object を相互変換します。

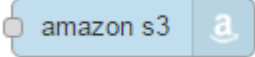
 xml	XML 書式のデータと JavaScript Object を相互変換します。
 rbe	入力データが変化した場合のみ出力します。

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 ノード

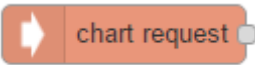
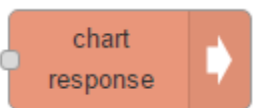
 tail	設定されたファイルの末尾を入力データとします。
 file	入力データに示されるファイルを開き、その内容を出力します。
 amazon s3	Amazon S3 からデータを取り込みます。
file 	設定されたファイルにデータを出力します。
amazon s3 	Amazon S3 に対する入出力を提供します。

	Amazon S3 にデータを出力します。
---	-----------------------


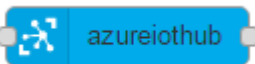

3-7. analysis ノード

	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。
---	--

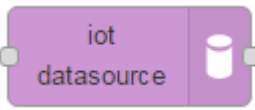
3-8. advanced ノード

	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。
	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。
	システムのコマンドを実行し、その出力を返します。
	Google chart にグラフ描画をリクエストします。
	Google chart のグラフ描画を出力します。

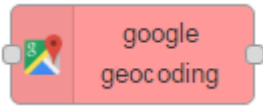
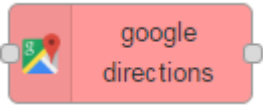
3-9. cloud ノード

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

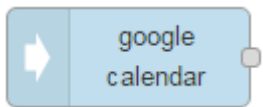
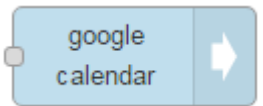
3-10. GatewayKit ノード

	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 ノード

 A red rounded rectangular node with a small icon of a location pin and a map on the left. The text "google geocoding" is centered in white.	Google geocoding 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。
 A red rounded rectangular node with a small icon of a location pin and a map on the left. The text "google directions" is centered in white.	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 ノード

 A light blue rounded rectangular node with a white arrow pointing right on the left. The text "google calendar" is centered in white.	Google calendar のイベント（予定の通知）の発生を待ち受けます。
 A light blue rounded rectangular node with the text "google calendar" on the left and a white arrow pointing right on the right.	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。



BLE センサーデバイスの電源を入れ、「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。



「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。



一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。

4. 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。



「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。

送信先設定

本体内(local) 使用する 使用しない
デバイス一括設定

PD Exchange 使用する 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 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]

有効時間[sec]

ソケットパスプレフィックス

デバイス一括設定

「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。

デバイス情報送信設定

デバイス番号 dev_le_0000001

送信対象 送信する 送信しない ^①

アドレス 34:C7:31:FF:E6:20

ユーザー名 Sensor_001

センサー信号強度[dbm]

取得時間間隔[ms]

送信先設定 local PD KINESIS AWSIOT Watson IoT(Device) Watson IoT(Gateway) EVENTHUB IoTHub T4D MQTT PLAIN NRED ^②

操作 ^③

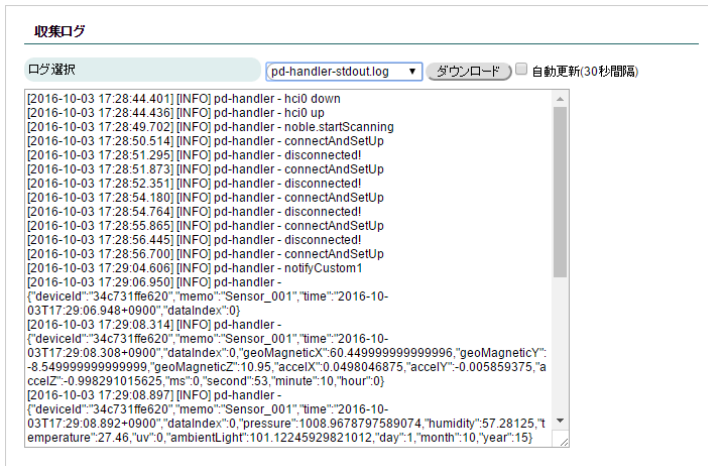
「送信対象」^①を”送信する”に設定します。

「送信先設定」の”NRED”チェックボックス^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。

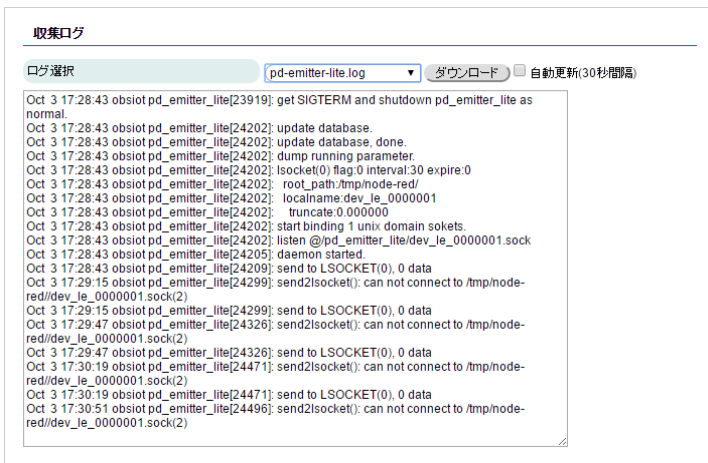
保存ボタン^③をクリックします。

7. WEB UI の「サービス」→「収集ログ」タブで動作を確認します。



「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。

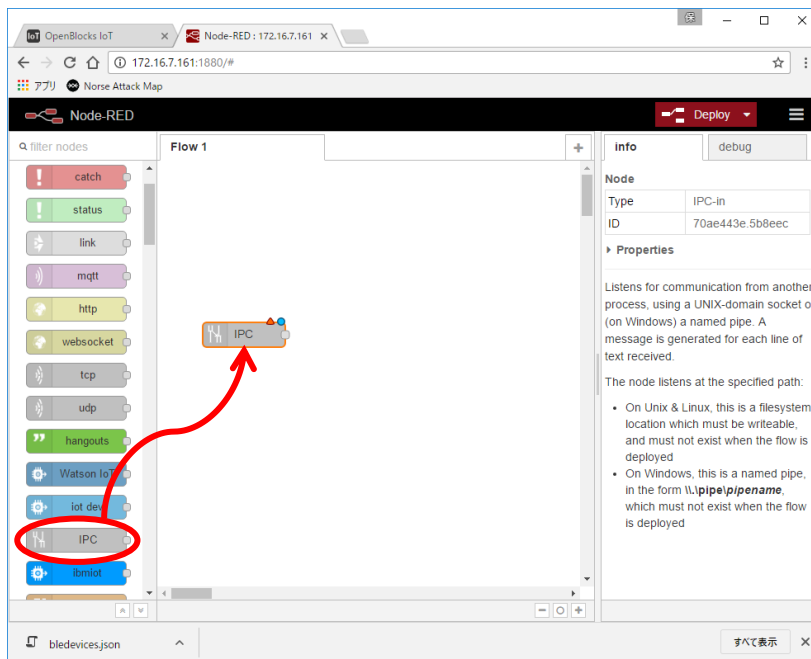
この時点では、Node-RED 側の UNIX ドメインソケットが用意されていないため、接続エラーが発生しています。

4-2. 単純なデバッグ

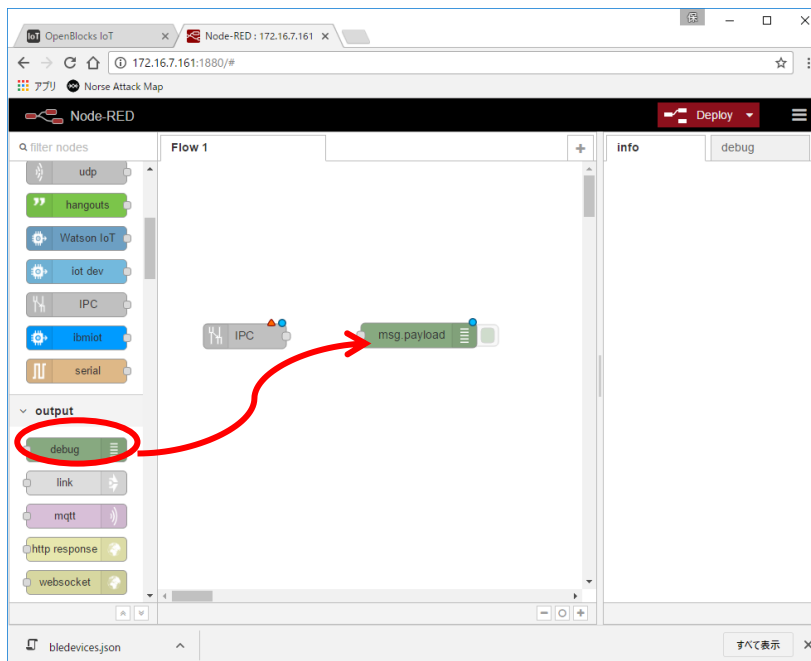
1. Node-RED にアクセスします。(第 3 章参照)

<http://192.168.254.254:1880/>

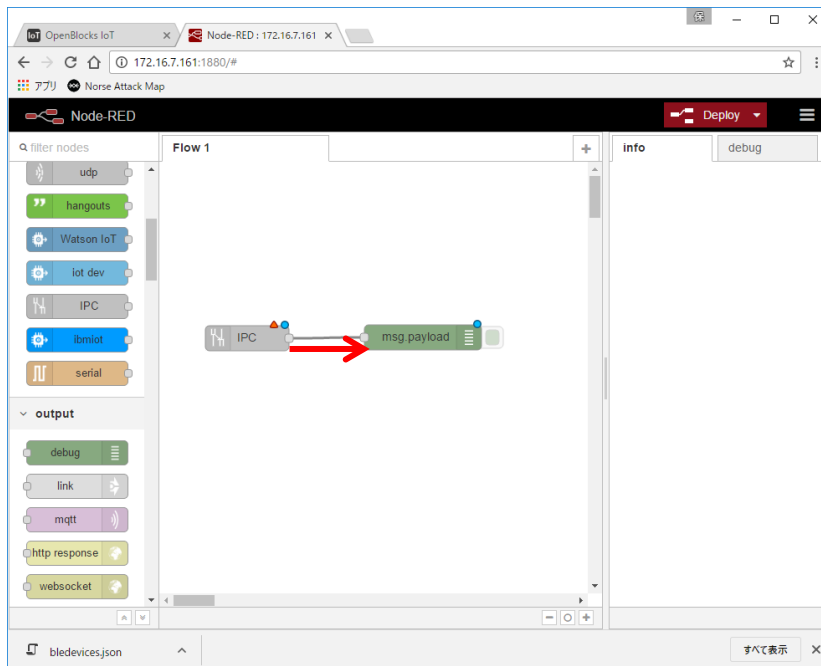
2. Input ノードパレットから IPC-in node をドラッグしてシートにドロップします。



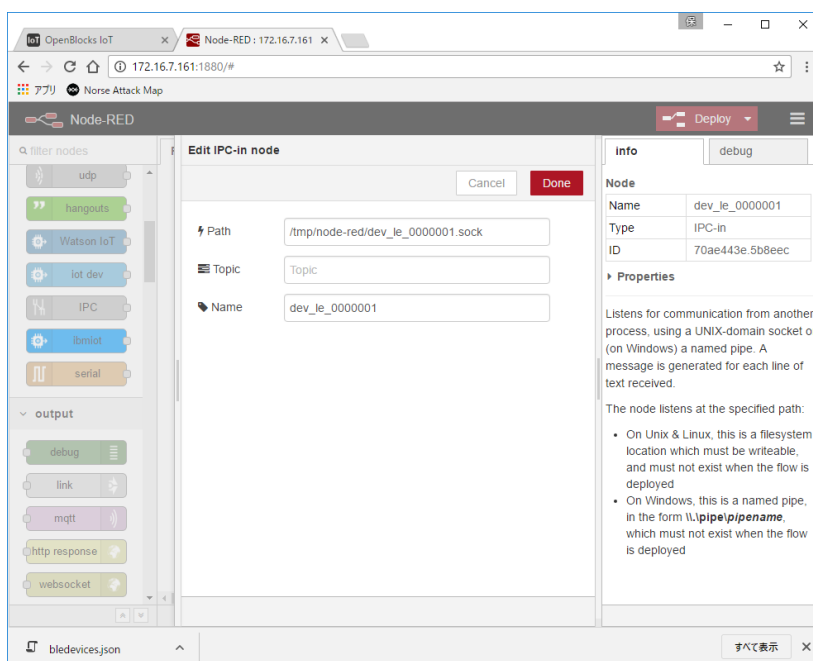
3. Output ノードパレットから debug node をドラッグしてシートにドロップします。



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。



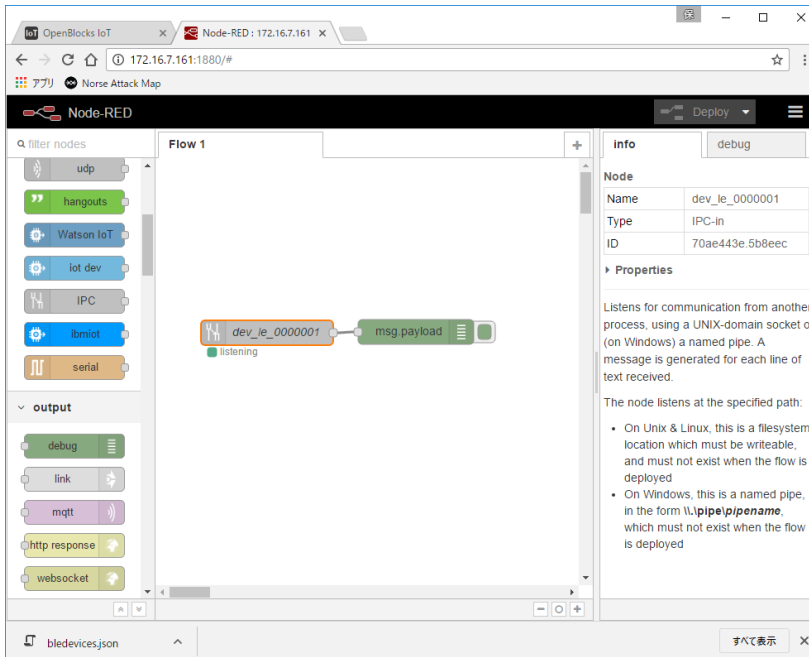
ここで、Path は、4-1 事前設定の 1 に図示される「ソケットパスプレフィックス」と同 2 に図示される「デバイス番号」に .sock を付加したものとします。

`/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。

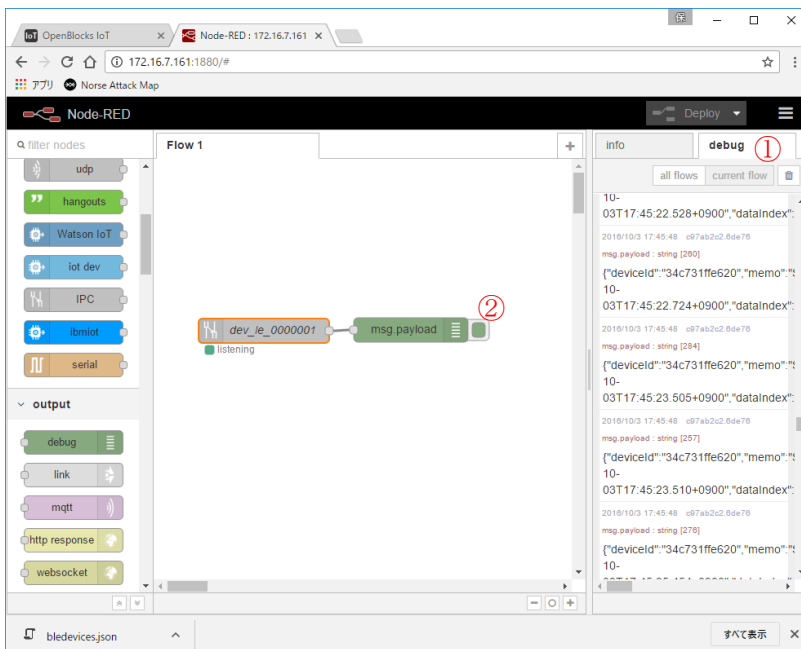
Topic は、空欄のままとします。

6. Deploy ボタンをクリックします。



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。

7. 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)。



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。

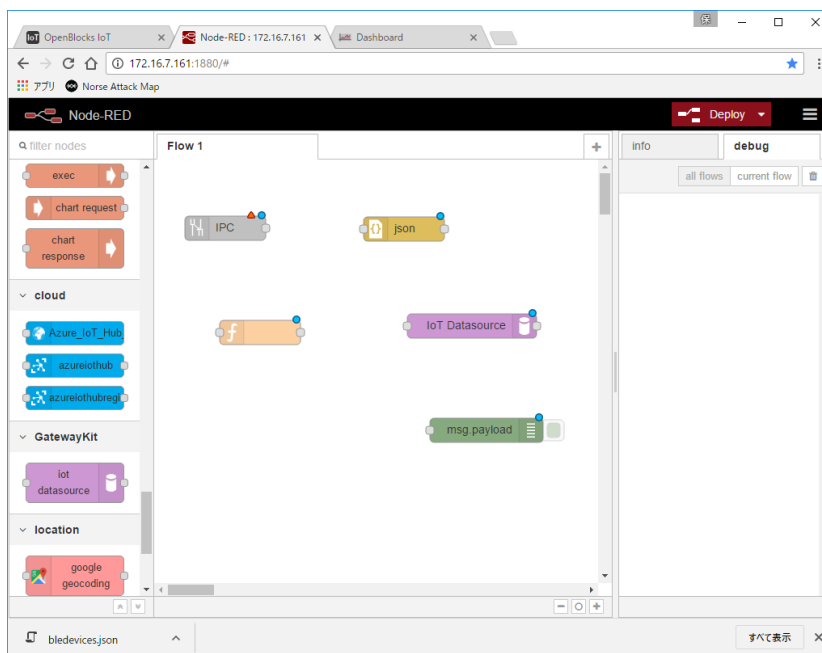
4-3. 温度データをグラフに表示する

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。

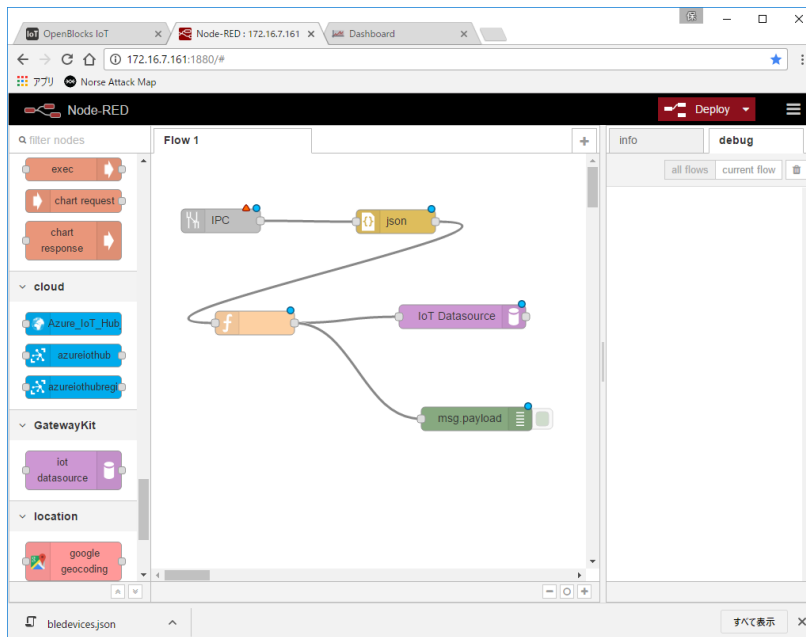
1. Node-RED にアクセスします。(第 3 章参照)

<http://192.168.254.254:1880/>

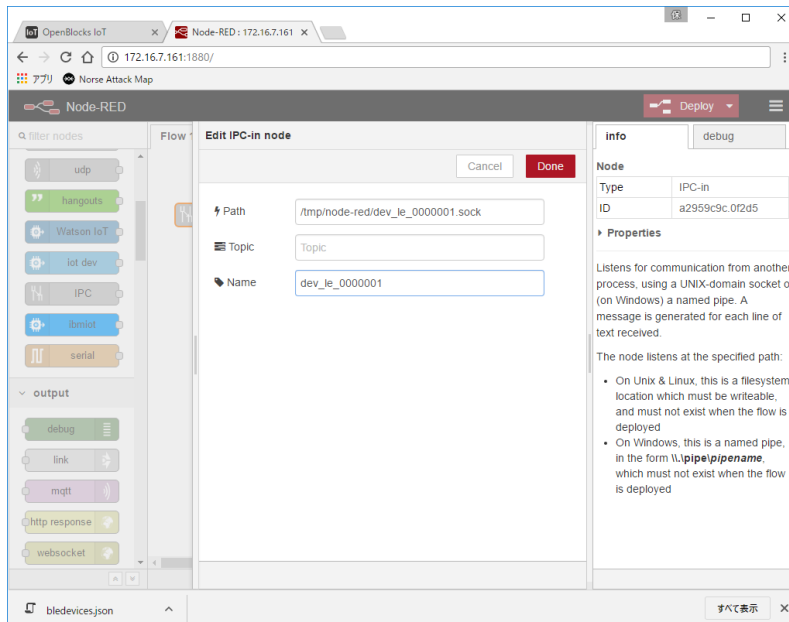
2. 下図に示す通り IPC-in node と json node, function node, iot datasource node, debug node をそれぞれドラッグしシートにドロップします。



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。



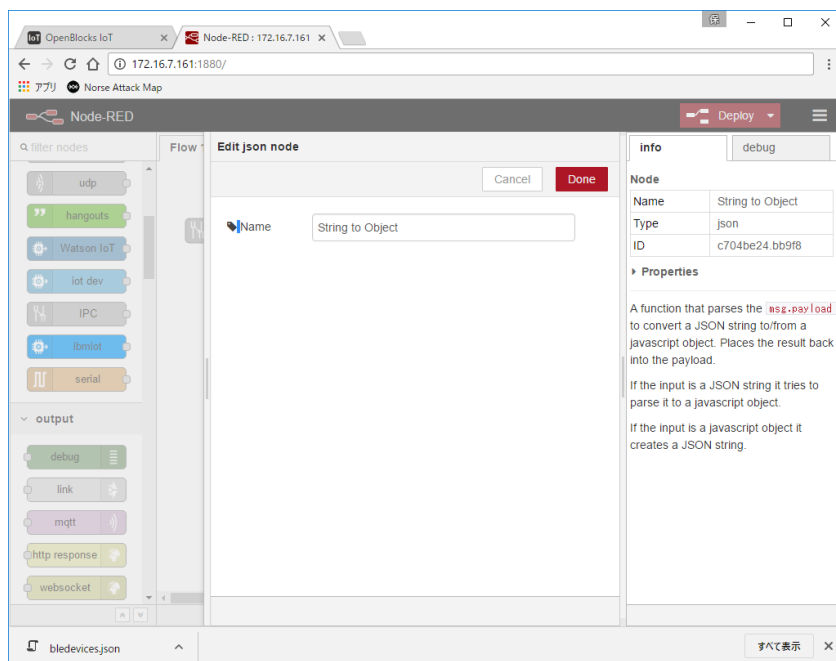
ここで、Path は、4-1 事前設定の 1 に図示される「ソケットパスプレフィックス」と同 2 に図示される「デバイス番号」に .sock を付加したものとします。

`/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。

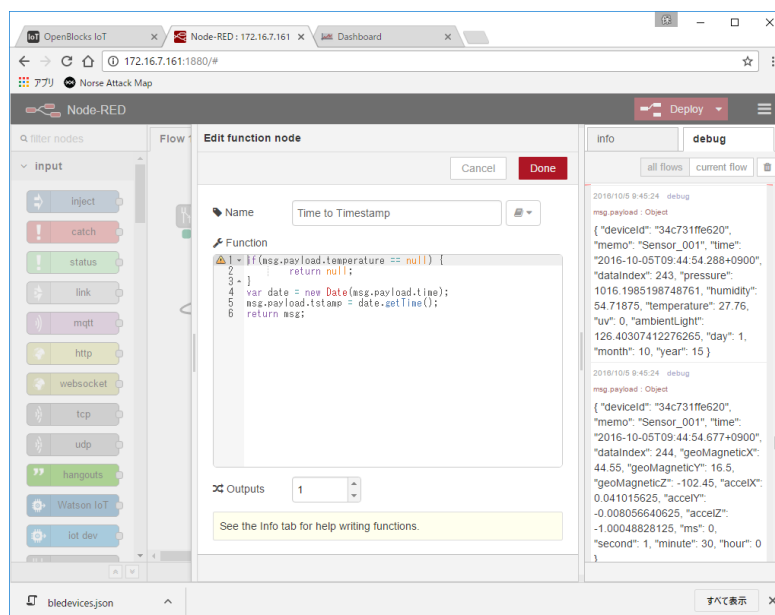


特に指定が無い限り OpenBlocks IoT ファミリーに標準搭載されているデータ取り込みアプリケーション (PD-Handler) の出力フォーマットは JSON 形式です。

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。

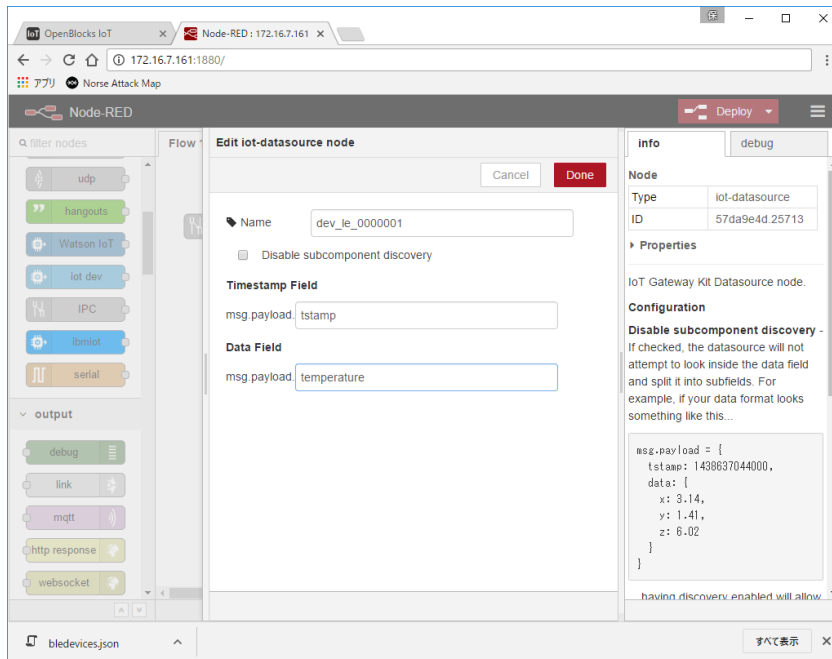
6. function node をダブルクリックし、適当な Name を設定、Function として次の JavaScript を記述し、Done ボタンをクリックします。

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

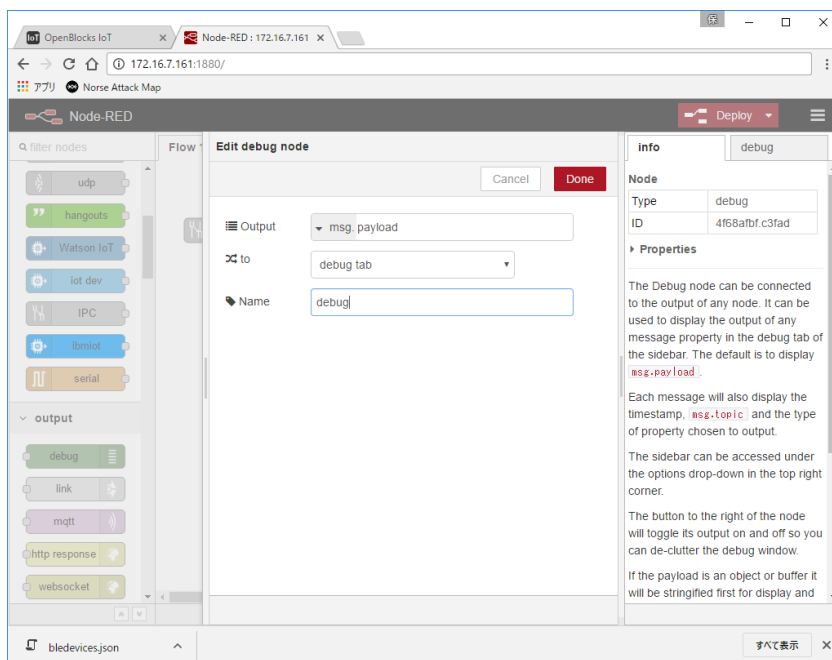


- ① ALPS 電気社製 IoT Smart Module は、温湿度等の環境データと加速度等のモーションデータを別々のタイミングで送信するため、温度情報を含まないデータ (msg) をブロックします。
- ② データに含まれる RFC3339 形式のタイムスタンプ (msg.payload.time) から、iot Datasource node で使用する UNIX 形式のタイムスタンプに変換します。

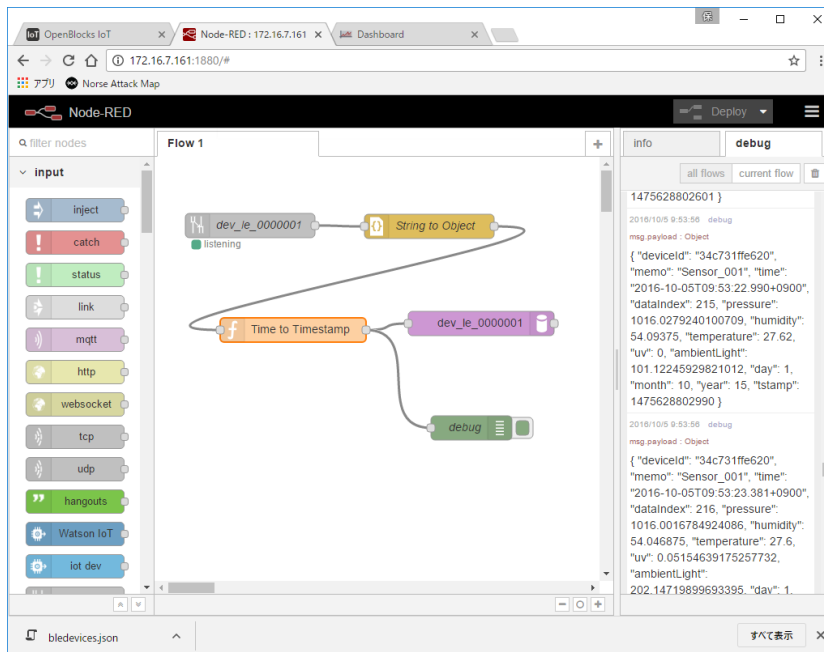
7. Iot Datasource node をダブルクリックし、Name として dev_le_0000001 を設定、Timestamp Field として msg.payload.tstamp 、 Data Field として msg.payload.temperature を設定し、Done ボタンをクリックします。



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

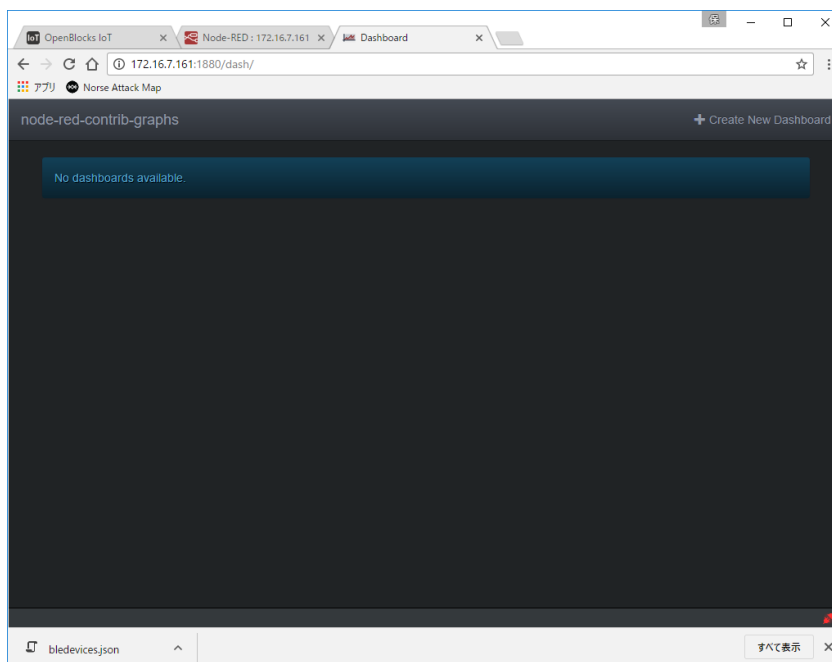


9. Deploy ボタンをクリックします。

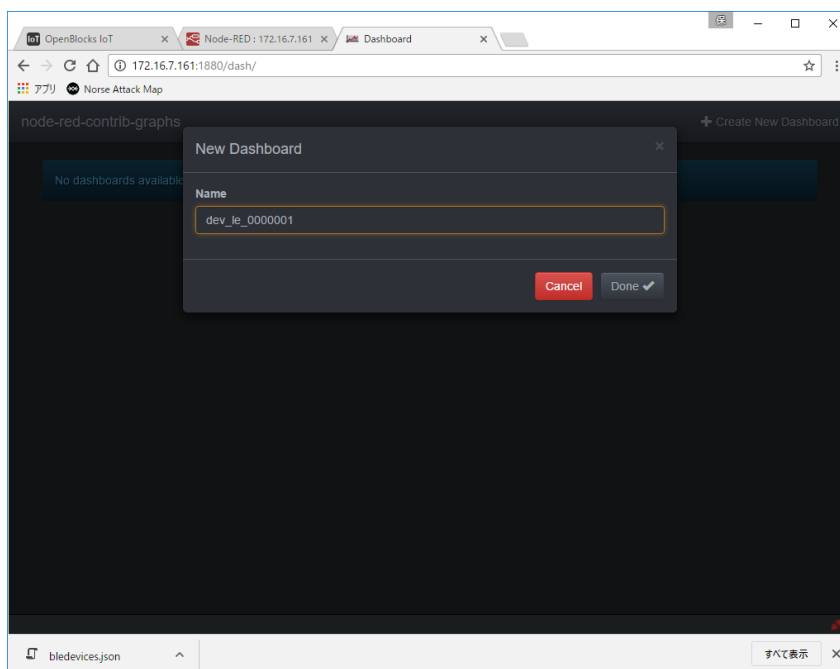


10. 新しいブラウザタブを開き Iot Datasource node のダッシュボードにアクセスします。

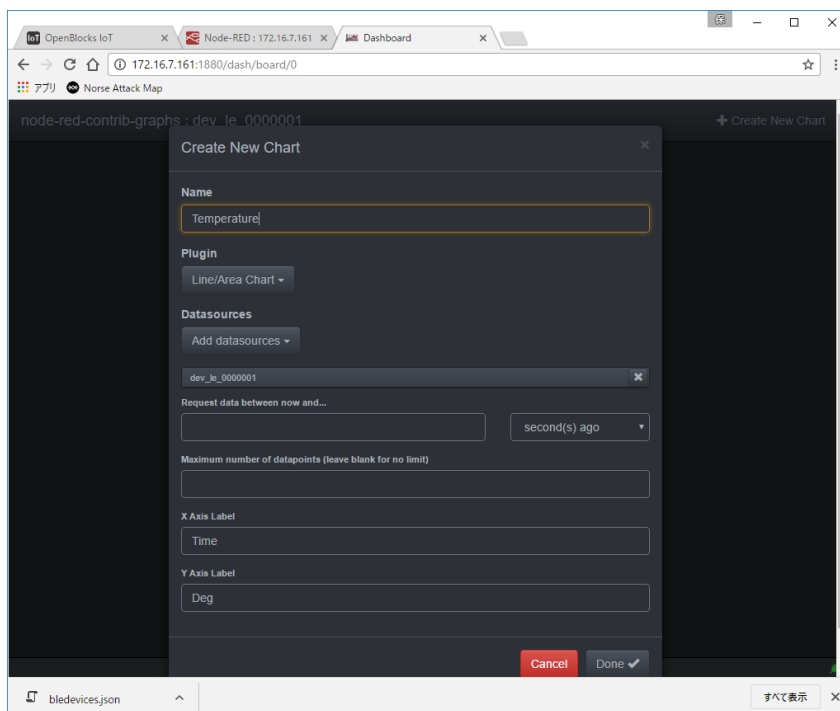
<http://192.168.254.254:1880/dash/>



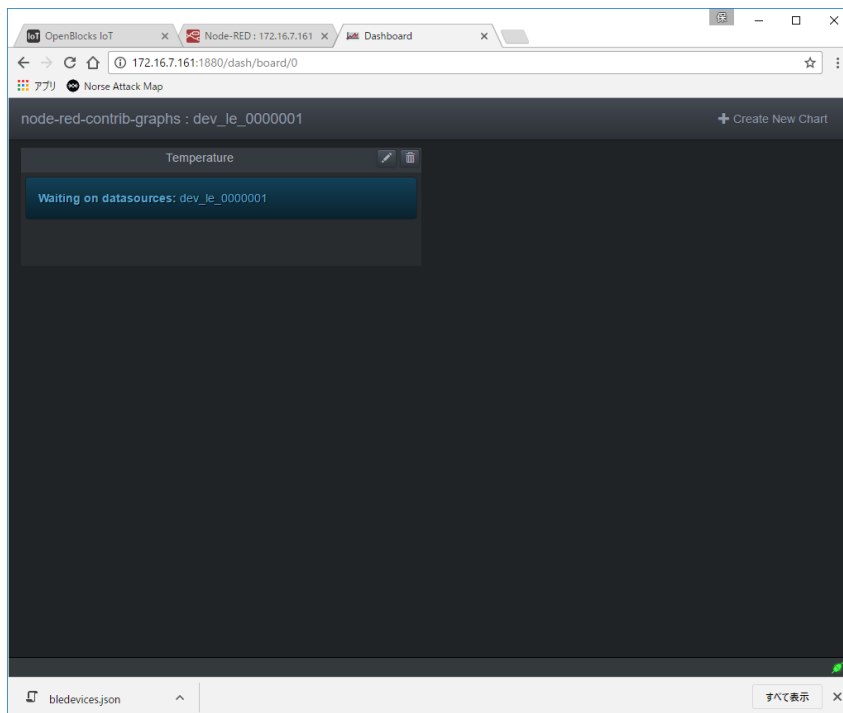
11. +Create New Dashboard をクリックし、Name として dev_le_0000001 を設定し、Done をクリックします。



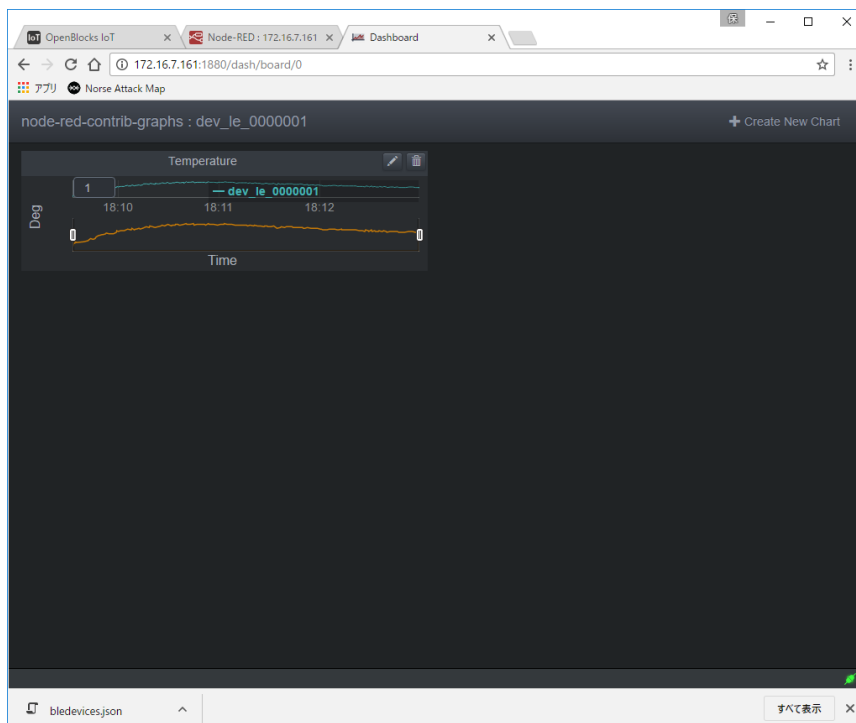
12. +Create New Chart をクリックし、適当な Name を設定し、Plugin として Line/Area Char を選択、Add datasource として dev_le_0000001 を選択、X Axis Label として”Time”、Y Axis Label として”Deg”を設定し、Done をクリックします。



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
(2017/04/20 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