

# OpenBlocks IoT VX1向け Node-REDスターターガイド



Ver.2.0.0

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。

### 安全上のご注意

- ・ご使用の前に、この「安全上のご注意」をよくお読みのうえ、正しくお使いください。
- また、お読みになったあとは大切に保管してください。 ・ここに示した注意事項は、お使いになる人や、他の人への危害、財産への損害を未然に防ぐための内容を記載していま すので、必ずお守りください。
- ・本機の故障、誤動作または不具合などにより、通信などの機会を逸したために、お客様、または第三者が受けられた損 害につきましては、当社は責任を負いかねますのであらかじめご了承ください。

#### 表示の説明

次の表示の区分は、表示内容を守らず、誤った使用をした場 合に生じる危害や損害の程度を説明しています。内容をよく理 解したうえで本文をお読みください。

⚠危険	この表示は、取り扱いを誤った場合、「死亡また は重傷を負う危険が切迫して生じることが想定 される」内容です。
▲警告	この表示は、取り扱いを誤った場合、「死亡また は重傷を負う可能性が想定される」内容です。
⚠注意	この表示は、取り扱いを誤った場合、「軽傷を負う可能性が想定される場合および物的損害のみ の発生が想定される」内容です。

#### 絵表示の説明

次の絵表示の区分は、お守りいただく内容を説明しています。 内容をよく理解したうえで本文をお読みください。

❷禁止	禁止(してはいけないこと)を示します。
● 指示	指示に基づく行為の強制(必ず実行していただ くこと)を示します。

## 本機、SIM カード、AC アダプタ、 SD カードの取り扱いについて

	0	高温になる場所(火のそば、暖房器具のそば、直 射日光の当たる場所、炎天下の車内など)で使用・ 放置しないでください。 機器の変形・故障や内蔵電池の漏液・発熱・発火・ 破裂の原因となります。また、ケースの一部が熱 くなり、やけどなどの原因となることがあります。
⚠危険	0	分解・改造・ハンダ付けなどお客様による修理を しないでください。 火災・けが・感電などの事故または故障の原因と なります。また、内蔵電池の漏液・発熱・破裂・発 火などの原因となります。本機の改造は電波法 違反となり、罰則の対象となります。
	0	濡らさないでください。 水などの液体が入ったときに、濡れたまま放置す ると、発熱・感電・火災・けが・故障などの原因と なります。使用場所、取り扱いにご注意ください。
	0	添付された以外のACアダプタを本製品に使っ たり、本製品に添付のACアダプタを他の製品に 使ったりしないでください。 ACアダプタの発熱・発火・故障などの原因となり ます。

	0	本機・ACアダプタを、加熱調理機器(電子レンジ など)・高圧容器(圧力釜など)の中に入れたり、電 磁調理器(IH調理器)の上に置いたりしないでく ださい。 内蔵電池の漏液・発熱・破裂・発火や、本機・AC アダプタの発熱・発煙・発火・故障などの原因と なります。
	0	落としたり、投げたりして、強い衝撃を与えない でください。 内蔵電池の漏液・発熱・破裂・発火や火災・感電・ 故障などの原因となります。
	0	外部I/O端子やACアダプタ本体のプラグやUSB 給電コンソールケーブル、microUSBケーブルの プラグに水などの液体や導電性異物(鉛筆の芯 や金属片など)が触れないようにしてください。 また内部に入れないようにしてください。 ショートによる火災や故障などの原因となります。
⚠ 警告	0	プロパンガス、ガソリンなどの引火性ガスや粉 塵の発生する場所(ガソリンスタンドなど)では、 必ず事前に本機の電源をお切りください。 ガスに引火する恐れがあります。プロパンガス、 ガソリンなど引火性ガスや粉塵の発生する場所 で使用すると、爆発や火災などの原因となります。
	0	使用中、充電中、保管時に、異音・発煙・異臭な ど、今までと異なることに気づいたときは、次の 作業を行ってください。 1.本機の電源を切ってください。 2.給電用ケーブルを全て抜いて下さい。ACアダ ブタはアダプタ本体を持ってプラグを抜いてく ださい。異常な状態のまま使用すると、火災や感 電などの原因となります。
	•	電池を機器に入れる場合は、+(プラス)と- (マイナス)の向きに注意し、表示どおりに入れて ください。 間違えると電池の破裂、液もれ、発火の原因にな ります。
	0	ぐらついた台の上や傾いた所など、不安定な場 所に置かないでください。 落下して、けがや故障などの原因となります。
⚠ 注意	0	本機を給電機器から取り外す際は、コードを引っ 張らず、プラグを持って取り外してください。 コードを引っ張るとコードが傷ついたり、端子の 破損による火災や感電などの原因となります。
	0	ご使用環境によっては高温になる場合があります。 やけどのおそれがありますので、本体底面に手を 触れないようにしてください。

#### 本機の取り扱いについて

本機の内蔵電池の種類は次のとおりです。

表示			電池の種類					
CR2032/K5GK		5GK	コイン型リチウム電池					
	0	火の 内蔵 なりま	ーに投下しないでください。 電池を漏液・破裂・発火させるなどの原因と 。す。					
▲警告	0	本機内のSIMカードスロットやmicroSDカー スロットに水などの液体や金属片、燃えやす ものなどの異物を入れないでください。 火災、やけど、けが、感電の原因となります。						
	0	航かのの航まなたのの航まなた	機へのご搭乗にあたり、本機の電源を切る 内モードに設定してください。航空機内で 用については制限があるため、各航空会社 示に従ってください。 機の電子機器に悪影響を及ぼす原因となり 航空機内での使用において禁止行為をし 合、法令により罰せられることがあります。					
	0	病院での使用については、各医療機関の指示に 従ってください。 使用を禁止されている場所では、本機の電源を 切ってください。 電子機器や医用電気機器に悪影響を及ぼす原 因となります。						
	0	高器子あ※補除自精の機切で聴細動	すな制御や微弱な信号を取り扱う電子機 丘くでは、本機の電源を切ってください。電 器が誤動作するなどの影響を与える場合が さす。 注意いただきたい電子機器の例 器・植込み型心臓ペースメーカ・植込み型 勤器・その他の医用電気機器・火災報知器・ ドア・その他の自動制御機器など。					
	0	車でくた。本様にする	電子機器に影響を与える場合は使用しない ざさい を自動車内で使用すると、車種によりまれ 両電子機器に影響を与え、安全走行を損な いがあります。					
▲ 注意	0	本機に磁気カードなどを近づけないでください。 キャッシュカード・クレジットカード・テレホンカ ード・フロッピーディスクなどの磁気データが消 えてしまうことがあります。						
	0	指定0 漏液・	D電池以外はご使用にならないでください。 破裂・発火の危険があります。					
	0	で使うる漏まで、	用後の電池は充電、分解、火の中に投下す なことはしないでください。 破裂・発火の危険があります。 電池を廃棄する場合は各自治体の指示に 「処分してください。					

#### AC アダプタの取り扱いについて 使用中は、布や布団でおおったり、包んだりしな 0 いでください 熱がこもって火災や故障などの原因となります。 指定以外の電源・電圧で使用しないでください。 指定以外の電源・電圧で使用すると、火災や故 障などの原因となります。 ACアダプタ:AC100V~240V(家庭用交流 ACコ 0 ンセント専用) また、海外旅行用として、市販されている「変圧 器」は使用しないでください。火災・感電・故障の 原因となります。 ACアダプタのコードが傷んだら使用しないでく Ø ださい。 火災、やけど、感電の原因となります。 雷が鳴り出したら、ACアダプタには触れないで 感電などの原因となります。 濡れた手でACアダプタのプラグや端子を抜き 差ししないでください。 感電や故障などの原因となります。 ▲ 警告 プラグにほこりがついたときは、ACアダプタを 持ってプラグをコンセントから抜き、乾いた布な 0 どで拭き取ってください。 火災の原因となります。 ACアダプタをコンセントに差し込むときは、AC アダプタのプラグや端子に導電性異物(鉛筆の 芯や金属片など)が触れないように注意して、 0 確実に差し込んでください。 感電やショートによる火災・やけど・故障などの 原因となります。 本機にACアダプタを抜き差しする場合は、無理 な力を加えず、水平に真っ直ぐ抜き差ししてくだ 0 さい 火災、やけど、けが、感電の原因となります。 長時間使用しない場合は、ACアダプタ本体を持 ってプラグをコンセントから抜いてください。 0 感電・火災・故障の原因となります。 万一、水などの液体が入った場合は、ただちに ACアダプタを持って、コンセントからブラグを抜いてください。 感電・発煙・火災の原因となります。 ACアダプタをコンセントに接続しているときは、 ❷ 引っ掛けるなど強い衝撃を与えないでください。 けがや故障の原因となります。 プラグに手や指など身体の一部が触れないよ Ø うにしてください。 ▲注意 やけど・感電・傷害・故障の原因となります。 ACアダプタをコンセントから抜くときは、コー ドを引っ張らず、必ずACアダプターを持ってプ ラグを抜いてください。 コードを引っ張るとコードが傷つき、感電や火

災などの原因となります。

## loT機器を安全に利用するために

従来の人が介在するインターネット利用とは違い、IoT機器では機械同士が情報を自動でやり取りをするため、通信のセキュリティにおいて見落としがちになります。

ここではIoT機器を安全に利用するために、必要最小限考慮すべき事柄について述べます。

- 1. IoT機器のログイン設定においで、製品出荷時のデフォルトパスワードを必ず変更する。
- 2. インターネットに接続される機器は定期的にセキュリティアップデートを行う。
- 3. 長期停止後のIoT機器の運用開始前には、必ず始動点検を行う。
- 4. 通信における暗号化技術を積極的に導入する。
- 5. ハードウェアが本来接続された本物かを判断できる認証技術をなるべく導入する。
- 6. その他、総務省が発行する「IoTセキュリティガイドライン」を参考にする。

### Bluetooth<sup>®</sup>/Wi-Fi(無線LAN)ご使用上の注意

● 本機の Bluetooth<sup>®</sup> 機能および Wi-Fi (無線 LAN) 機能は、2.4GHz 帯の周波数を使用します。

#### [現品表示] Bluetooth<sup>®</sup>機能:2.4 FH8

本機は 2.4GHz 帯を使用します。FH8 は、変調方式として FH-SS 変調方式を採用し、与干渉距離は約 80m 以下です。 Wi-Fi (無線 LAN) 機能: 2.4DS/OF4 本機は 2.4GHz 帯を使用します。変調方式として DS-SS 方式および OFDM 方式を採用しています。与干渉距離は約 40m 以下です。

2400MHz ~ 2483.5MHz の全帯域を使用し、かつ移動体識別装置の帯域を回避可能です。

- 本製品の使用周波数帯では、電子レンジ等の産業・科学・医療用機器のほか工場の製造ライン等で使用されている移動体識別用の構内無線局(免許を要する無線局)および特定小電力無線局(免許を要しない無線局)が運用されています。
- (1).本製品を使用する前に、近くで移動体識別用の構内無線局及び特定小電力無線局が運用されていないことを確認してください。
   (2).万一、本製品から移動体識別用の構内無線局に対して電波干渉の事例が発生した場合には、速やかに電波の発射を停止した上、下記の連絡先にご連絡頂き、混信回避のための処置等(例えば、パーティションの設置など)についてご相談してください。
- (3). その他、本製品から移動体識別用の特定小電力無線局に対して電波干渉の事例が発生した場合など何かお困りのことが 起きたときは、次の連絡先へお問い合わせください。
  - 連絡先:ぷらっとホーム株式会社 TEL:03-5213-4372 E-Mail:support@plathome.co.jp

本機は5GHzの周波数帯においてW52のチャンネルを使用できます。W52は、電波法により屋外での使用が禁じられています。

本機の Bluetooth<sup>®</sup> / Wi-Fi (無線 LAN) 機能は日本国内規格に準拠し、認定を取得しています。一部の国/地域では Bluetooth<sup>®</sup> / Wi-Fi (無線 LAN) 機能の使用が制限されることがあります。海外でご利用になる場合は、その国/地域の 法規制などの条件をご確認ください。

### その他のご注意

- この装置は、クラスA情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この 場合には使用者が適切な対策を講ずるよう要求されることがあります。VCCI-A
- 本製品は、国内での使用を前提に作られています。
- 海外での使用につきましては、お客様の責任で行っていただくようお願いいたします。
- ・ DC Wide 入力をご使用いただく場合、下記及び同等の外付けノイズフィルターの接続が必要です。
- 推奨ノイズフィルタ:NAC-04-472(COSEL)
- ・本製品に搭載されている記憶媒体は eMMC で、書き込み回数に制限が設けられた有寿命部品です。修理の際、書き込み上限に達していることが確認された場合には保証期間内であっても有償修理となります。
- ・周囲温度が 40℃を超える環境に本製品を設置する場合は、添付の放熱・設置ブラケットを取り付けてご使用ください。

### 目次

第1章 はじめに
第2章 Node-RED 事前準備
2-1. Node-RED の使用設定について
2-2. Node-RED のブラウザフィルタについて
2-3. パケットフィルタについて
第3章 Node-REDの簡易説明10
3-1. Node-RED 画面構成11
3-2. ノード種類12
3-2. Input ノード
3-3. Output ノード
3-4. function $\mathcal{I} - \mathcal{F}$
3-5. social $\mathcal{I} - \mathcal{F}$
3-6. storage $\prime - F$
3-7. analysis $\mathcal{I} - \mathcal{F}$
3-8. advanced $\checkmark - F$
3-9. cloud $\prime - F$
3-10. GatewayKit ノード17
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18
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 VX1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。

OpenBlocks®	loT		ログイン ID: admin 横線: スーパーユーザー) <u>マイページ ログア</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	8136 <b>81.67</b> 1	14E
スクリプト編集 コマンド教行	) SMSコマンド東南 SMS送信 PD Subscrib	node red	
ログ			
89	(選択したものを表示します ▼		
node red (?) 使用缺定	使用する ○ 使用.たい		
术小者号	(1880		
ロジイン問題設定	<ul> <li>使用する (使用しない)</li> </ul>		
ユーザー名			
パスワード			
操作			
(課存) リンク)			

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。

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
SSHを無効し	するには無効選択	剣に保存ボタンを	押下ください			
基本	I¥18	1127	ード フィルター	- SSH関連	71-1-	2
WEB2-tf-	- ファイル管理	5/17	Z SN			
フィルター	-開放設定 🗹 再起	動揺もフィルタ開放	設定を有効にする (?)			
0011			● 有効 ◎ 無効			
SSH			◉ 有効 ◎ 無効			
node red						
SSH node red 操作						

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 ファミリー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 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)



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

	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。
antah	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
status	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。
a 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
intep 9	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
Webbooker	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
	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。

hangouts	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	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
iot dev	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
ГЧ ІРС	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。
ibmiot	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
N serial	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 ✓ – 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
serial 👖	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 $\mathcal{I} - \mathcal{F}$

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。
රා 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。
oti range	入力データのスケールを変更して出力します。
split	入力データを設定される文字で区分して出力します。
join	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 <mark>{}</mark> 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。
f I 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。
amazon s3 a.	Amazon S3 にデータを出力します。

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

continent	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 $\mathcal{I} - \mathcal{F}$

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 ✓ — ド

Azure_loT_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。
azureiothubregi	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	Google directions を用いて入力された出発地点と目的地点
directions	の経路を出力します。

### 3-12. Google ノード

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」→「Bluetooth 関連」タブにて Bluetooth LE デバイスの検出 を行います。

Open	Blocks	<sup>®</sup> IoT		п <i>9</i> 4	(ンID: admin (権限: J	(ーパーユーザー) <u>マイベ</u>	<u>-9 87795</u>
ダッシュボード	サービス	システム	ネットワーク	メンテナンス	拉張	技術情報	
基本	Bluetooth	関連 BLEメンテ	サンス 状態				
Bluetoo	th(?)						_
Bluetooth	デバイス検出		検出				
Bluetooth	LEデバイス検出	寺間(秒)	(15				
Bluetooth	LEデバイス検出		機出				
操作							_
保存							
一覧							
デバイス番	큑	アドレス	고-!	ゲーメモ	操作		
						Version 2.0.0de	1

Bluetooth LE センサーデバイの電源を入 れ、「Bluetooth LE 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 Bluetooth LE デバイスの一覧から使用するデバイスを選択します。

ボード	サービ	ג א <del>זד</del> ע	ネットワー	ク メンテナンス	拡張	技術情報
基本	Blue	tooth関連BLE火	ヮテナンス	状態		
		_				
Bluetoot	th( <u>?</u> )					
Bluetooth <sup>=</sup>	デバイス検片	К	拾出)			
		5	1200			
Pluotooth	1日ボバイフ	1全山185月月(長小)	65			
Bluetootii	LED 7 MIX	1800/181(12)	(1)			
Bluetooth	LEデバイス	検出	(検出)検出約	17		
6	中田時会会	10	H	_	大廠	
5	ATLEAAC	Device Name		unknown	1.10	
(1)	<b></b>	Device Address		34:C7:31:FF:E6:20		
<u>ب</u>		Memo	(2)	Sensor_001		
		Device Name		FWM8BLZ02		
		Device Address		D4:D4:64:B2:8F:6A		
		Memo		(		2
		Device Name		FCLBeacon1		
		Device Address		D7:C8:30:DA:D2:48		
		Memo		C		
		Device Name		FWM8BLZ02		
		Device Address		D7:F3:D2:59:EC:91		
		Memo				
		Device Name		FCLWirelessModule		
		Device Address		EF:FD:BF:57:85:46		
		Memo		(		
				1		
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 (権限: スーパーユーザー) <u>マイページ ログアウト</u> ダッシュボード サービス システム ネットワーク 設定を保存しました。 基本 Bluetooth関連 Bluetooth编集 BLE火 Bluetooth(?) Bluetoothデバイス検出 検出 Bluetooth LEデバイス検出時間(秒) (15 Bluetooth LEデバイス検出 検出 操作 保存 一覧 ユーザーメモ 操作 アドレス dev\_le\_0000001 34:C7:31:FF:E6:20 Sensor\_001 削除

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。

д 70 — Г.	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
基本	収集設定	403	集ログ データ	表示 Bluetooth费	連 Bluetooth編集	
LEメンテ	ナンス 状態					
データ	収集					
データル	限集	(1)	● 使用する ● 使用	用しない		
	PD Handler BLE	2	<ul> <li>使用する</li> <li>使用する</li> </ul>	用しない		
	追加Unixドメインソク	アット数	0 •			
	ユーザーHandler使	用設定	◎ 使用する ® 使用	用しない		
PD自動	再起動設定		● 使用する ● 使用	用しない		

「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
デバイスー括設定	一括有効 ) 一括無効 )
PD Exchange	○ 使用する ● 使用しない
Amazon Kinesis	<ul> <li>使用する</li> <li>使用しない</li> </ul>
AWS IoT	<ul> <li>● 使用する ● 使用しない</li> </ul>
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 ⑧ 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]	0
ソケットパスプレフィックス	(tmp/node-red/
デバイスー括設定	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。

ドバイス番号	dev_le_0000001
送信対象	<ol> <li>送信する <sup>(1)</sup> 送信しない</li> </ol>
アドレス	34:C7:31:FF:E6:20
ユーザーメモ	Sensor_001
2ンサー信号強度[dbm]	0
败得時間間隔[ms]	(5000
送信先設定	Ilocal PD KINESIS AWSIOT Watson IoT(Device)     Watson IoT(Gateway) EVENTHUB IoTHub T4D MQTT     PLAIN RD
<b>揭存</b>	(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。

ログ選択	(pd-handler-stdout.log ▼ ダウンロード) □ 自動更新(30秒間隔)
2016-10-03 17-28:44.401] [INFO] pd-hand 2016-10-03 17-28:44.401] [INFO] pd-hand 2016-10-03 17-28:49.702] [INFO] pd-hand 2016-10-03 17-28:50.514] [INFO] pd-hand 2016-10-03 17-28:50.514] [INFO] pd-hand 2016-10-03 17-28:50.873] [INFO] pd-hand 2016-10-03 17-28:55.4180] [INFO] pd-hand 2016-10-03 17-28:55.4180] [INFO] pd-hand 2016-10-03 17-28:55.4865] [INFO] pd-hand 2016-10-03 17-29:06.950] [INFO] pd-hand 2016-10-03 317-29:08.397] [INFO] pd-hand 20171-29:08.390+900":dataIndex".0; pc 20171-29:08.390+9000";dataIndex".0; pc 20172-29:08.390+9000;dataIndex".0; pc 20172-29:08.392+1015625", mmom~Senso 303177-29:08.392+1015625", mmom~Senso 303177-29:08.392+1015625", mmom~Senso 303172-29:08.392+1015625", mmom~Senso 303172-29:08.392+1015625", mmom~Senso 303172-29:08.392+1015625", mmom~Senso 303172-29:08.392+10000", dataIndex".0; pc emperature"-27.46; ww0.70 ambient.ll;hr11	<pre>lier - hci0 down lier - hci0 up lier - could startScanning lier - connectAndSetUp lier - disconnectAndSetUp lier - di</pre>
収集ログ	
<b>収集ログ</b> ログ選択	(pd-emitter-lite.log ▼ ダウンロード) □ 自動更新(30秒間隔)

「 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					_
Reference Node-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 ×	🥰 Node-RED : 172.	16.7.161 ×		₿ – □ ×
← → C ☆ ③ 172.16.7.16	51:1880/#			☆ :
🎹 アブリ 🐵 Norse Attack Map				
■ Node-RED			-	🖌 🗧 Deploy 👻 🚍
Q filter nodes	Edit IPC-in node	3	info	debug
) udp 🔶 📤		Cancel Done	Node	
P hangouts			Name	dev_le_0000001
Wataon InT	🕈 Path	/tmp/node-red/dev_le_0000001.sock	Туре	IPC-in
watson for			ID	70ae443e.5b8eec
iot dev	E Topic	Торіс	Propertie	s
N IPC di> Ibmiot ∬ serial	Name	dev_le_0000001	Listens for o process, usi (on Window 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	itens at the specified path:
debug			On Unix location and mus deploye     On Wind in the fo which m is deploy	& 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, ust not exist when the flow yed
🗊 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。 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 × Node-RED : 172.16.7.161 ×		₿ – □ ×
← → C ① 172.16.7.161:1880/#		☆ :
III アブリ 🐵 Norse Attack Map		
Node-RED		=/ Deploy -
Q filter nodes Flow 1	+	info debug 1
i) udp	Â	all flows current flow
hangouts		10- 03T17:45:22.528+0900","dataIndex":
💮 Watson IoT 🎐		2016/10/3 17:45:48 c97ab2c2.6de76
🔅+ iot dev 💿		{"deviceId":"34c731ffe620","memo":"
		10- 03T17:45:22.724+0900","dataIndex":
🔅 ibmiot 🛉 dev_ie_0000001 🚽 🖉 msg.payload 🗐 🔲		2016/10/3 17:45:48 c97ab2c2.6de76
Istening		{"deviceId":"34c731ffe620","memo":"
v output		10- 03T17:45:23.505+0900","dataIndex":
		2016/10/3 17:45:48 c97ab2c2.6de76
debug		{"deviceId":"34c731ffe620","memo":"
e link		10- 03T17:45:23.510+0900","dataIndex":
(mqtt s)		2016/10/3 17:45:48 c97ab2c2.6de76
http response		"sg.payload : string [278] {"deviceId":"34c731ffe620","memo":"
websocket		10-
	0+	
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。

				ß – □ ×
OpenBlocks Iol X	Node-RED : 1/2.16.	(161 ×		
₹ → C 1 0 1/2.16. 1 770 S Norse Attack Map.	7.101:1000/			
■< <sup>■</sup> Node-RED				Deploy - =
Q filter nodes	Flow 1 Edit IPC-in n	ode	info	debug
udp		Cancel	Done Node	
			Туре	IPC-in
nangouts	Path	/tmp/node-red/dev_le_0000001.sock	ID	a2959c9c.0f2d5
Watson IoT	E Topic	Торіс	Properties	
iot dev	Name Name	dev_le_0000001	Listens for com process, using (on Windows) message is ge	nmunication from another a UNIX-domain socket or a named pipe. A merated for each line of
∭ serial			The node liste	ns at the specified path:
output     debug     ink     mqt     mtpresponse     websocket			<ul> <li>On Window</li> <li>In the deployed</li> <li>On Window</li> <li>In the form which must is deployed</li> </ul>	Linux, this is a mession ich must be writeable, not exist when the flow is ws, this is a named pipe, it ( <b>\\pipe\pipename</b> , it not exist when the flow d
D bledevices.json	^			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 V 🛲 Dashboard x	₿ – □ ×
← → C ☆ ① 172.16	<b>6.7.161</b> :1880/#	☆ :
11 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  inik  mqt  thtp	Name       Time to Timestamp	2016/105 9.4524 debug mg.paylos - Objet (* 'device'), "34.c731ffe520', "memo": "Sensor, 001", "time", "2016-10-05709:44.54.288-0900", "dataIndex": 243, "pressure"; 1016.19851987.48761, "humidity", 54.71875, "temperature" 27.76, "uv": 0, "ambientLight", 126.40307412278265, "day": 1, "month": 10, year": 15)
websocket		2010/105 9-45.24 debug mg.paylask : Objet ("device!d": "34c731ffe620", "memo": "Sensor_001", "time": "2016-10-05705:44:54.677+0900", "dataIndex": 244, "geoMagneticX", 44.55, "geoMagneticX": 16.5,
Watson IoT	X Outputs	"geoMagneticZ": -102.45, "accelX": 0.041015625, "accelY": -0.008056640625, "accelZ":
iot dev	See the Info tab for help writing functions.	-1.00048828125, "ms": 0, "second": 1, "minute": 30, "hour": 0 3
I bledevices.json	^	すべて表示 ×

- 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 loT 🛛 🗙 🧟	Node-RED : 172.16.7.161 ×		₿ – □ X	
← → C ☆ ③ 172.16.7.161:1	1880/		:	
🎹 アプリ 🐵 Norse Attack Map				
-C Node-RED		-	Deploy 👻 🗮	
Q filter nodes Flow	Edit iot-datasource node	info	debug	
) udp	Cancel Done	Node	A	
22 handouts		Туре	iot-datasource	
hangoard	► Name dev_le_0000001	ID	57da9e4d.25713	
Watson IoT	Disable subcomponent discovery	Properties		
iot dev	Timestamp Field	IoT Gateway K	Kit Datasource node.	
M IPC	msg.payload. tstamp	Configuration		
🔹 ibmiot 👂	Data Field	Disable subcomponent discovery - If checked, the datasource will not		
II serial	msg.payload. temperature	attempt to look inside the data field and split it into subfields. For		
✓ output		example, if you something like	ur data format looks this	
debug		msg.payload = {		
e link		data: {	+30037044000,	
mqtt		y: 1.41,		
http response		}		
websocket		having discr	wany anablad will allow	
			WELX EDALIEU WILLAUUW	
🗸 bledevices.json \land			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。

D OpenBlocks IoT	× / 🔜 N	Vode-RED : 172.16.7.1	61 X			₿ – □ ×
← → C ☆ 0 172.1	16.7.161:18	380/				:
🏭 アブリ 🐵 Norse Attack Ma	p					
-C Node-RED					-	🗲 Deploy 🔸 📃
Q filter nodes	Flow 1	Edit debug noo	de		info	debug
udp b				Cancel Done	Node	
					Туре	debug
hangouts P	۳.,	i≣ Output	👻 msg. payload		ID	4f68afbf.c3fad
Watson IoT		⊅⊄ to	debug tab	<b>.</b>	Properties	s
iot dev     int dev		Name	debug		The Debug r to the output used to disp message pro- the sidebar. <b>msg.payloar</b> Each messa timestamp, of property of The sidebar	node can be connected of any node. It can be lay the output of any operty in the debug tab of The default is to display <b>j</b> . ge will also display the nsg.topic and the type chosen to output.
ink 🔶					the options of corner.	drop-down in the top right
http response					The button to will toggle its can de-clutte	o the right of the node s output on and off so you er the debug window.
websocket	4				If the payloa will be string	d is an object or buffer it ,ified first for display and 🗸
× ¥						
D bledevices.json	^					すべて表示 X

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 ☆ ③ 172.16.7.161:1880/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 🔟 🕊 Dashboard X	æ	-		×
← → C ① 172.16.7.161:1880/dash/board/0			☆	:
III アプリ 🕲 Norse Attack Map				
				rt 🌔
Create New Chart ×				
Name				
Temperature				
Plugin				
Datasources				
Request data between now and				
second(s) ago 🔻				
Maximum number of datapoints (leave blank for no limit)				
X Axis Label				
Y Axis Label				
Deg				
Cancel Done 🗸				<i>j</i> , -
I bledevices.json		₹^	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 VX1 向け Node-RED スターターガイド

(2016/10/17 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