AWS 接続環境構築ガイド

CiP-1 CC3200：IoT 向けソリューション

2017/05/24

目次

1.はじめに

1.1 本マニュアルについて

本マニュアルでは、CC3200とCip-1を組み合わせ、CiP-1のセンサ情報を、AWSのEC2インスタンス上のサーバに接続して送信する環境の構築方法をご紹介いたします。AWSのご利用にあたっては、AWSにて定められている規約等をご確認の上、お客様の責任の下でご利用されるものとします。
また、本マニュアル内にて記載しているソフトウェアやサービス（mosquitto、paho、nodejs、dweet.io）についても、各ソフトウェアやサービスのライセンス等をご確認のうえ、お客様の責任の下でご利用されるものとします。

1.2 リファレンス環境

本環境では、センサ情報の送受信に MQTT プロトコルを利用します。
AWS の EC2 インスタンス上に MQTT Broker を立ち上げていただき、CC3200 が MQTT Publisher としてセンサ情報を MQTT メッセージで送信する構成です。
また、AWS EC2 インスタンス上の MQTT Broker で受信したセンサ情報は、各種のソフトウェアやサービスを利用することで分析や可視化を行うことが可能です。本マニュアルの Appendix では、可視化サービスの一例として dweet.io の利用例をご紹介いたします。

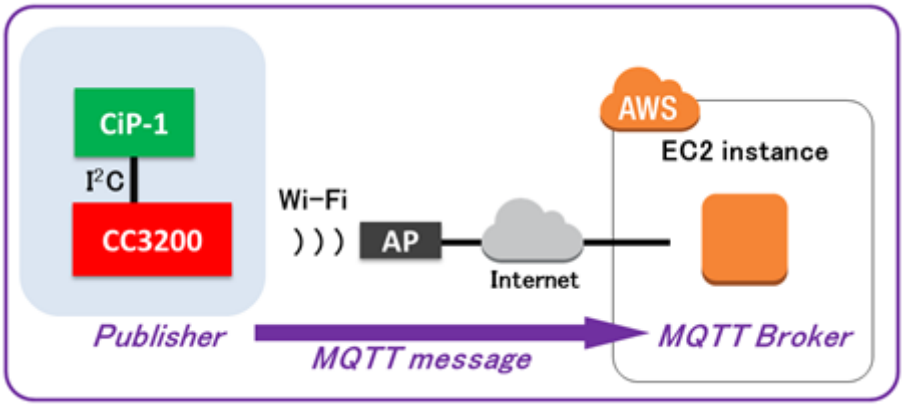


図 1. AWS 接続環境構成図

1.3 事前準備

本マニュアルの事前準備として、『CiP-1 User's Guide』に記載された"CiP-1 CC3200 向け OOB デモおよびデバック"を完了しているものとします。また、本ガイドで使用する『差分 Patch ファイル』および『Debug 環境構築ガイド』は Web では公開しておりませんので、下記よりお問合せください。

Mpression ブランド Web サイト内　お問い合わせページ：　https://service.macnica.co.jp/contact

また、AWS環境の事前準備として、AWS の EC２インスタンスを立ち上げて頂く必要があります。

本マニュアルでは、Ubuntu Server 14.04 LTS (64bit) の Amazonマシンイメージ（AMI）を使用しました。
異なるLinuxディストリビューションのマシンイメージを利用することも可能ですので、その際は使用するディストリビューションに沿った操作方法をご確認ください。

表 1．使用 AMI

AMI
Ubuntu Server 14.04 LTS (64bit)

また、以下のプロトコルを使用しますので、使用するセキュリティグループに、適切なルールの設定をお願いします。

表 2．使用するプロトコル一覧

プロトコル	ポート番号
MQTT (TCP)	１８８３

2.セットアップ

2.1 AWS EC2 インスタンスでの設定

この項では、AWS EC2 インスタンス上にて実施していただくアプリケーションのインストール、プログラムの実行について解説します。

2.1.1 MQTT Broker（Mosquitto）のインストール

MQTT Broker として動作するアプリケーション【Mosquitto】をインストールします。

```
$ sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
$ sudo apt-get update
$ sudo apt-get install mosquito mosquitto-clients
```

以上で AWS EC2 インスタンス上での設定は完了です。
MQTT Broker は自動的に起動しています。

2.2 CC3200 ソースコードの変更点

この項では、CC3200 がセンサ情報を MQTT メッセージで送信するためのソースコードの変更点を解説します。

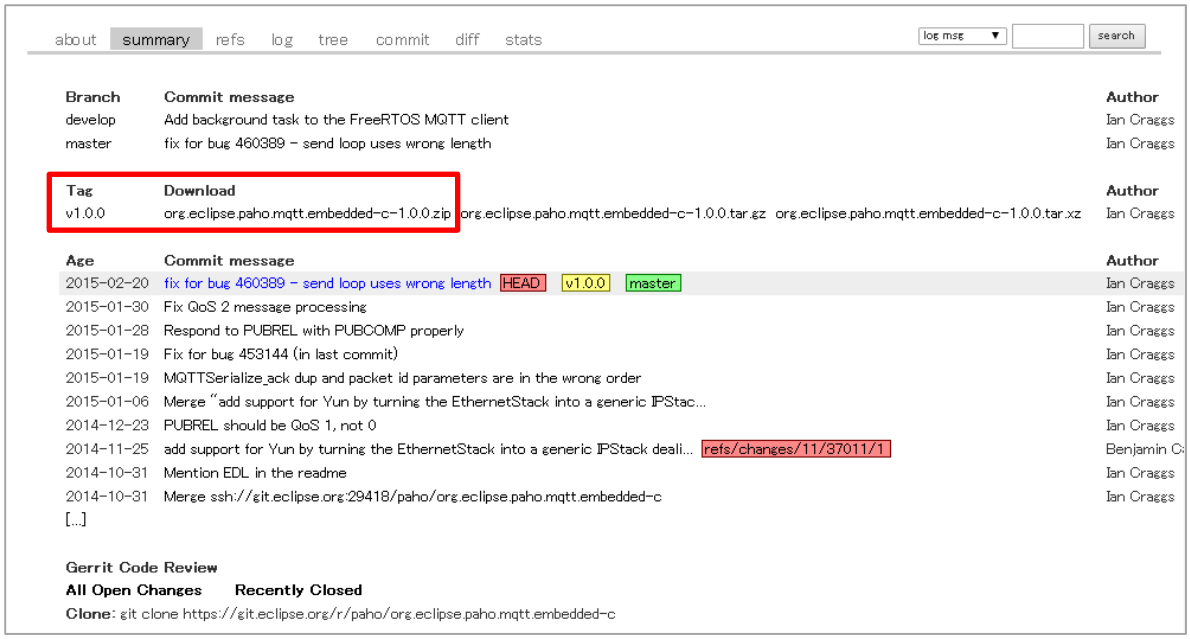
2.2.1 AWS 接続用差分 Patch の適用

『Debug 環境構築ガイド』の【2.2　差分 Patch の適用】を参考に 、差分 Patch ファイル【0002-MqttPublisher.patch】を適用してください。

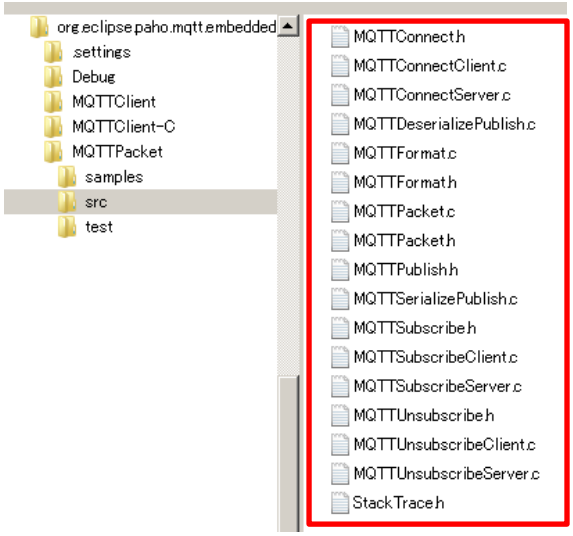
2.2.2 MQTT(Paho)ソースコードの取得

この項では、CC3200 からセンサデータを出力するために使用しているプロトコル：MQTT のために必要なソースコード Paho Embedded MQTT C/C++ Client Libraries（以下、Paho と記載）の取得方法を解説します。

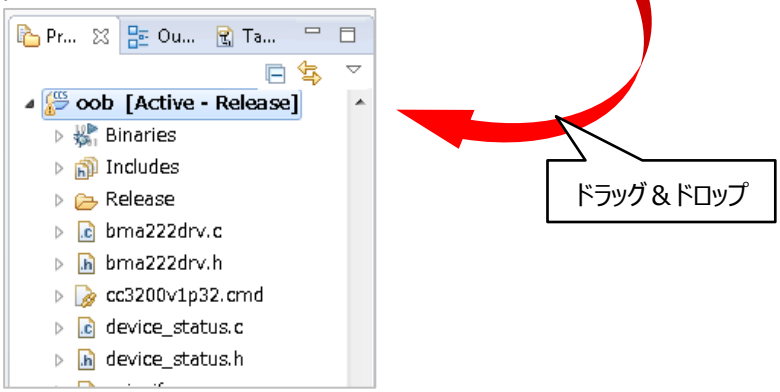
① 以下の URL にアクセスし、
http://git.eclipse.org/c/paho/org.eclipse.paho.mqtt.embedded-c.git/
【org.eclipse.paho.mqtt.embedded-c-1.0.0.zip】をダウンロードします。



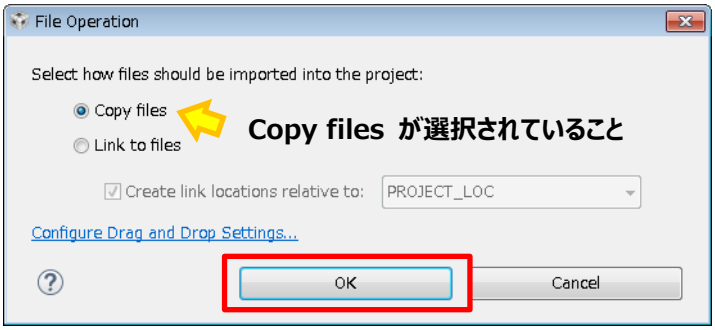
② ダウンロードしたファイルを解凍し、【MQTTPacket/src】配下のすべてのソースコードを選択します。



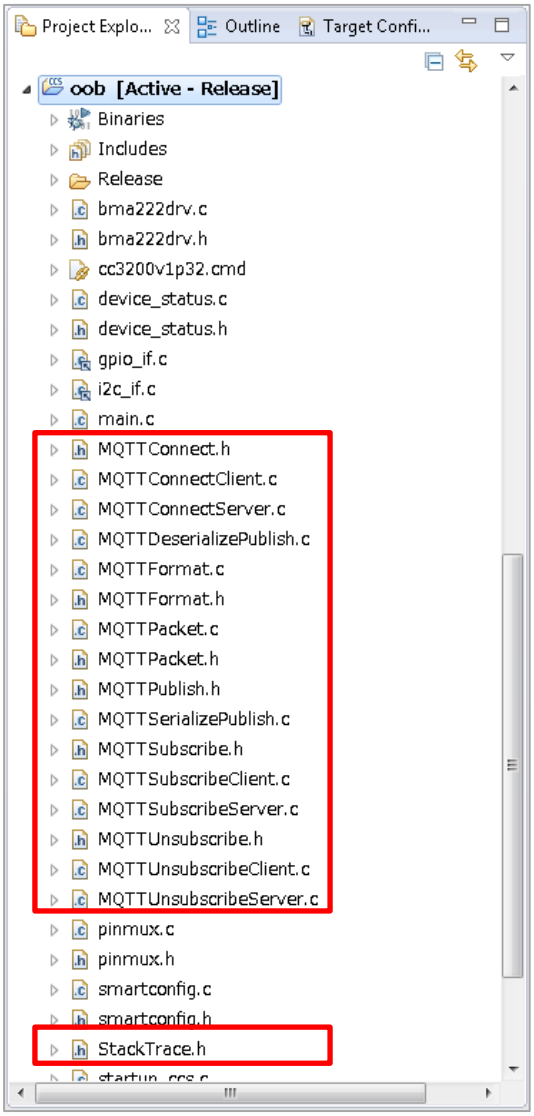
③ 選択したファイルを、CCS Project Explorer の【oob プロジェクト】へドラッグドロップします。



④ プロジェクトへのインポート形式を問われますので、【Copy files】を選択し、【OK】をクリックしてください。



⑤　以上で、Paho のインポート作業は完了です。



2.2.3 MQTT Broker のホスト名変更

CC3200 がセンサ情報を送信する MQTT Broker のホスト名が、main.c 内に HOST_NAME として定義されています。この HOST_NAME を 2.1 章にて作成していただいたお客様サーバのホスト名（パブリック DNS またはパブリック IP）に変更してください。

以上で、CC3200 ソースコード変更点についての作業は完了となります。
ソースコード変更後、問題なくビルドできることを確認ください。
ビルドしたイメージは、『CiP-1 User's Guide』の【4.3.3　ビルドされたプロジェクトを書き込む】を参考に CC3200 に書き込んでください。

3.接続確認

3.1 CC3200 の Wi-Fi AP への接続

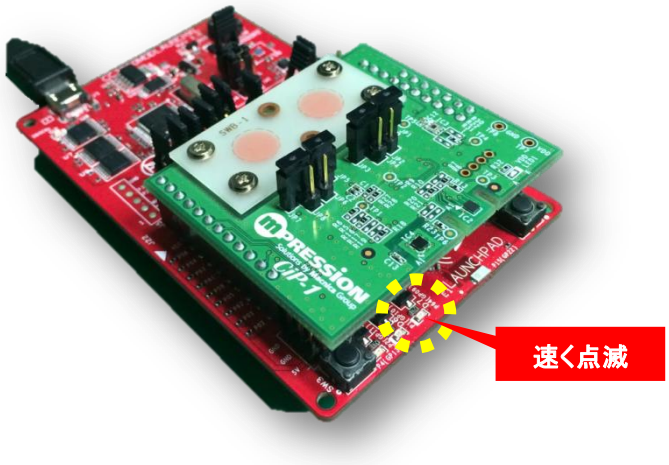
この項では、iOS/Android アプリを使用して、CC3200 を Wi-Fi AP へ接続する方法について解説します。

3.1.1 SmartConfig アプリの実行

① ご使用の環境に合わせて、アプリをダウンロードします。

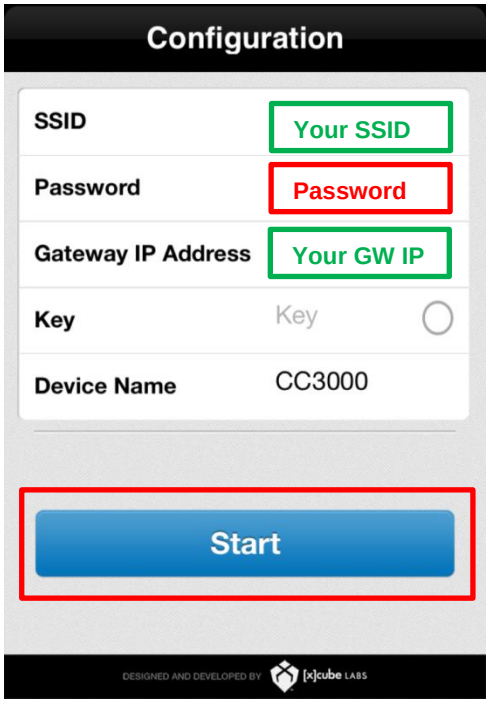
- iOS：SmartConfig iOS Application
  https://itunes.apple.com/jp/app/ti-wifi-smartconfig/id580969322?mt=8
- Android：SimpleLink™ Wi-Fi® Starter
  https://play.google.com/store/apps/details?id=com.pandaos.smartconfig&hl=ja

② CC3200 を起動してください。正常に起動すると D7 LED(赤)が速く点滅します。



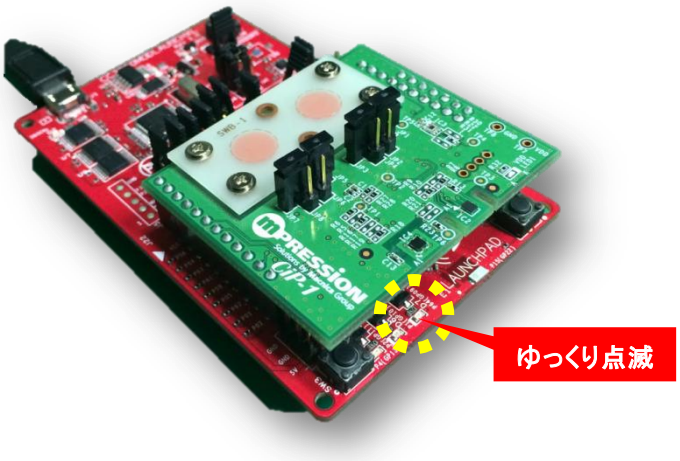
③ iOS/Android 端末を Wi-Fi AP に接続してください。

④ ダウンロードした【TI WiFi SMARTCONFIG】アプリを実行し、利用する Wi-Fi AP の【Password】を設定した後に、【Start】をタップしてください。※【SSID】【Gateway IP Address】は自動的に設定されます。



（※iOS アプリの画面を例に記載しています。）

⑤ 接続が成功すると、D7 LED(赤)がゆっくり点滅します。



以上で、CC3200 の Wi-Fi AP への接続作業は完了となります。
MQTT メッセージは約 1 秒周期で送信されている状態になっています。

3.2 MQTT メッセージの確認

この項では、AWS EC2 インスタンス上で MQTT メッセージを確認する方法について解説します。
MQTT ブローカーが起動している AWS EC2 インスタンス上で、MQTT Subscriber を実行することで、MQTT メッセージを確認します。

3.2.1 MQTT Subscriber の実行

MQTT Subscriber を実行します。

```
$ mosquitto_sub -d -t '#'
```

【実行例】
約 1 秒周期で、MQTT メッセージを受信していることがわかります。

```
$ mosquitto_sub -d -t '#'
Client mosqsub/1234-ip-10-0-0- sending CONNECT
Client mosqsub/1234-ip-10-0-0- received CONNACK
Client mosqsub/1234-ip-10-0-0- sending SUBSCRIBE (Mid: 1, Topic: #, QoS: 0)
Client mosqsub/1234-ip-10-0-0- received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/1234-ip-10-0-0- received PUBLISH (d0, q0, r0, m0, 'Example/cc3200', ... (211 bytes))
{"macAddr":"aa:bb:cc:dd:ee:ff","d":{"opt3000_ambientLight":480.0,"tmp007_irTemperature":29.1,"hdc1000_temperature":29.0,"hdc1000_humidity":69.1,"ldc1612_inductive_ch0":9.413402,"ldc1612_inductive_ch1":9.471347}}
Client mosqsub/1234-ip-10-0-0- received PUBLISH (d0, q0, r0, m0, 'Example/cc3200', ... (211 bytes))
{"macAddr":"aa:bb:cc:dd:ee:ff","d":{"opt3000_ambientLight":479.7,"tmp007_irTemperature":29.3,"hdc1000_temperature":29.0,"hdc1000_humidity":69.0,"ldc1612_inductive_ch0":9.413878,"ldc1612_inductive_ch1":9.471347}}
```

3.2.2 MQTT メッセージ内容

MQTT メッセージは、JSON 形式です。

表 3. MQTT メッセージ内容

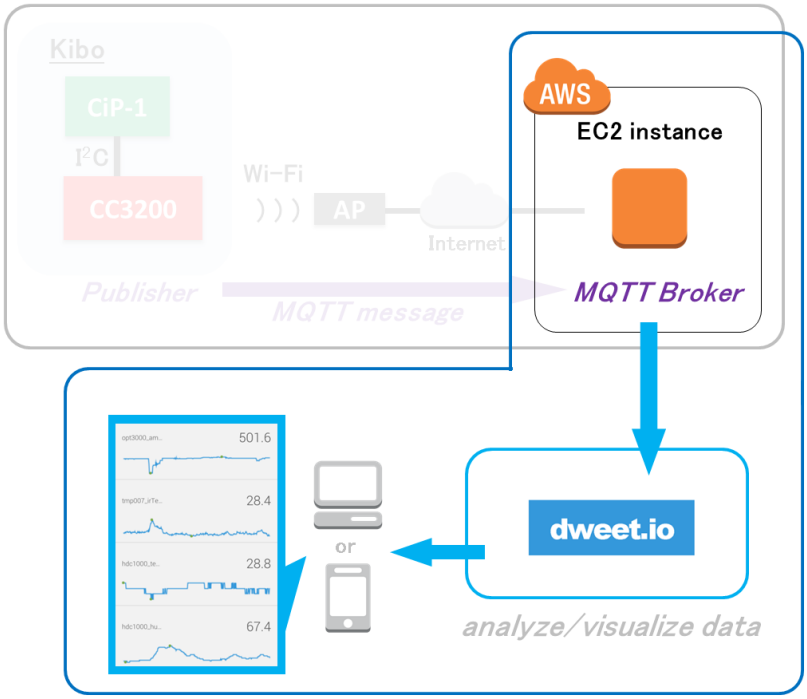
トピック名	Example/ CC3200		
メッセージ内容	macAddr	CC3200 の WLAN デバイス MAC アドレス	
	d	opt3000_ambientLight	照度センサ：OPT3001
		tmp007_irTemperature	赤外線温度センサ：TMP007
		hdc1000_temperature	温湿度センサ：HDC1000（温度）
		hdc1000_humidity	温湿度センサ：HDC1000（湿度）
		ldc1612_inductive_ch0	インダクティブセンサ：LDC1612（CH0）
		ldc1612_inductive_ch1	インダクティブセンサ：LDC1612（CH1）

以上で、MQTT メッセージの確認は完了です。

4.Appendix

4.1 dweet.io の利用例

Appendix では、可視化サービスの一例として dweet.io の利用例をご紹介いたします。
https://dweet.io/ を利用することで、センサ情報の可視化を簡易的に実現します。



4.1.1 Node.js client for dweet の実行例

MQTT Broker と同じ EC2 インスタンス上で実行します。

① Node.js のインストール

```
$ sudo apt-get install nodejs
$ sudo update-alternatives --install /usr/bin/node node /usr/bin/nodejs 10
```

② 必要ライブラリのインストール

```
$ mkdir mqtt2dweet
$ cd mqtt2dweet
$ npm install node-dweetio mqtt
```

③ mqtt2dweet.js スクリプト例

vi などのテキストエディタで、以下の内容のファイルを【mqtt2dweet.js】というファイル名で作成します。
このファイルは、必要ライブラリをインストールしたディレクトリと同じディレクトリに作成してください。

```
var dweetClient = require("node-dweetio");
var dweetio = new dweetClient();

var mqttBroker = 'localhost';
var mqtt  = require('mqtt');
var client = mqtt.connect('mqtt://' + mqttBroker);

client.on('connect', function () {
  console.log('MQTT connected: ' + mqttBroker);
  client.subscribe("#");
});

client.on('message', function (topic, message) {
  if ( topic === 'Example/cc3200' ) {
    var msg = JSON.parse(message);
    var thing = "cc3200_" + msg.macAddr.replace(/:/g,"");
    var content = msg.d;

    dweetio.dweet_for(thing, content, function(err, dweet){
    });
    console.log("https://dweet.io/follow/" + thing);
  }
});
```

④ スクリプトの実行

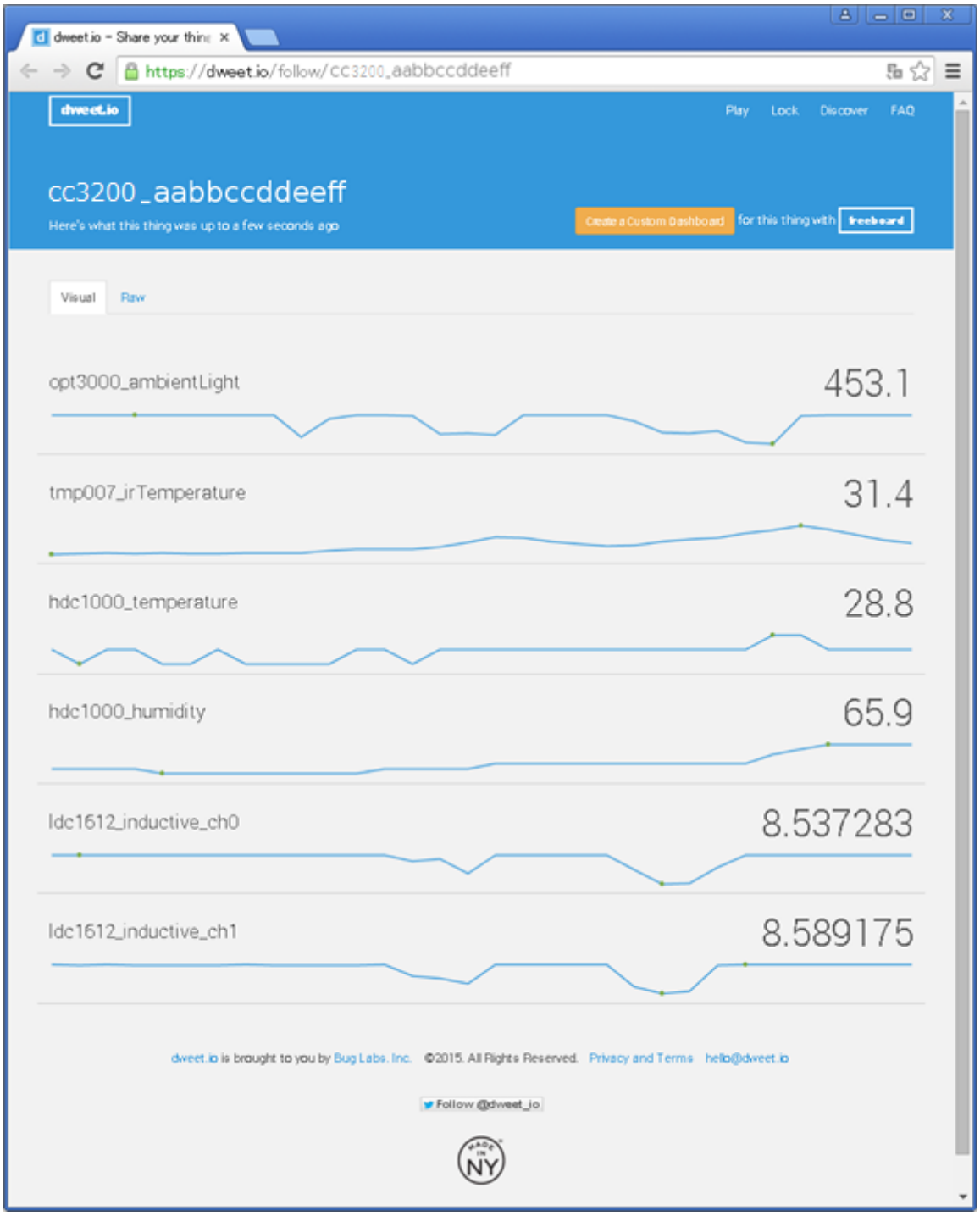
```
$ node mqtt2dweet.js
```

実行例 ：以下のように macAddr の情報から URL を生成して出力します。
https://dweet.io/follow/cc3200_{macAddr}

```
$ node mqtt2dweet.js
MQTT connected: localhost
https://dweet.io/follow/cc3200_aabbccddeeff
https://dweet.io/follow/cc3200_aabbccddeeff
https://dweet.io/follow/cc3200_aabbccddeeff
   :
   :
```

4.1.2 ブラウザでの表示例

出力された URL をブラウザで表示すると、MQTT メッセージの内容がグラフ表示されます。



dweet.io の利用例についてのご紹介は以上です。

5.更新履歴

日付	版	更新概要
2015 年 8 月 4 日	1.0	• 初版リリース
2015 年 5 月 24 日	1.1	• マクニカオンラインストアで販売していた際の名称「Kibo」を使用した記載を修正
		•