株式会社タブレイン 高本孝頼
2016/04

パート４：３GIM を使ったクラウド連携

１． クラウド連携とは

IoT デバイスで取得したセンサ値は、ほとんどの場合、クラウド・サーバにアップすることとなります。このパート４でも、パート２で紹介した IoT デバイスを使って、温度センサ値をクラウド・サーバにアップする開発技術をご紹介します。

ここで利用するクラウドもフリーでサービスしている米国 AT＆T が提供する M2X（URL：m2x.att.com）を紹介します。

まず今回のクラウド連携における全体フローを図１に紹介いたします。

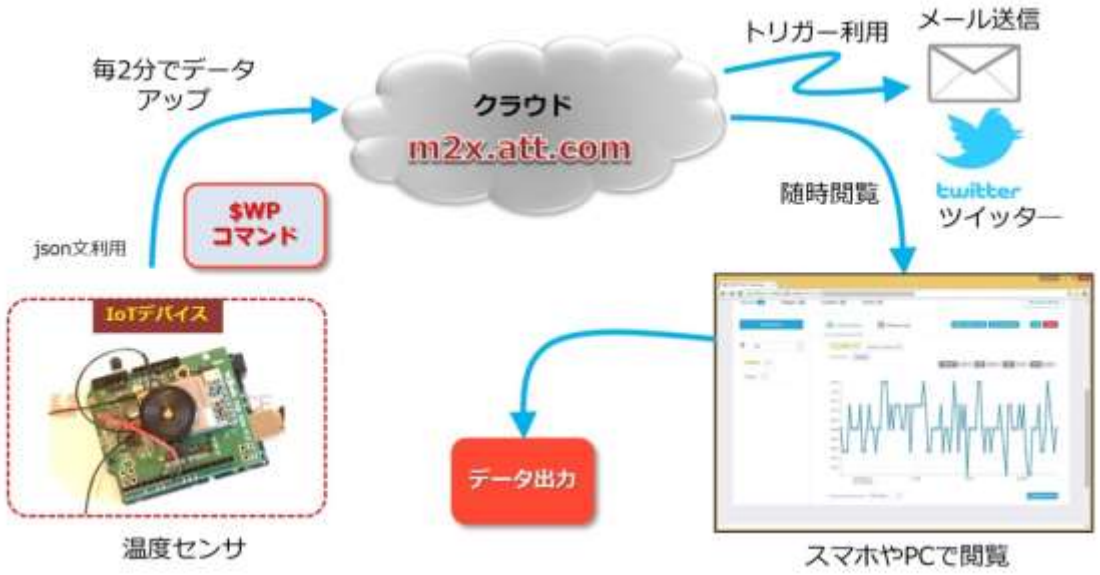

図１．クラウドを利用した IoT システム開発事例

ここでは先に、m2x の登録方法とその利用についてご紹介し、その後 IoT デバイスからデータをアップするプログラム（スケッチ）をご紹介し、さらにオプション機能として、トリガー利用によるメール送信・ツイッター送信や、蓄積したデータ出力について説明していきます。

また、ここでのデータアップについては、フォーマットとして json 形式を使っています。センサデータをアップする場合には、CSV や xml、それに json と３種類がほとんどですが、現時点では、データが構造体で、かつタグが付けられ、さらにコンパクトであることでは、json 形式が最も利用価値のあるものとなっています。単に１つのセンサだけだと、CSV でも構いませんが、今後多くの種類のセンサデータを蓄積していく場合には、json 形式をお薦め致します。

（注意：本 m2x サイトは、英文サイトのみです）

2． クラウド・サーバ m2x の登録

ここで紹介する m2x は、つぎの URL を Web ブラウザで接続し、あらかじめ図３および図４の画面でユーザ登録（SIGN UP）し、アカウントを取得しておいてください。

https://m2x.att.com/signup

図２．m2x.att.com のメイン画面

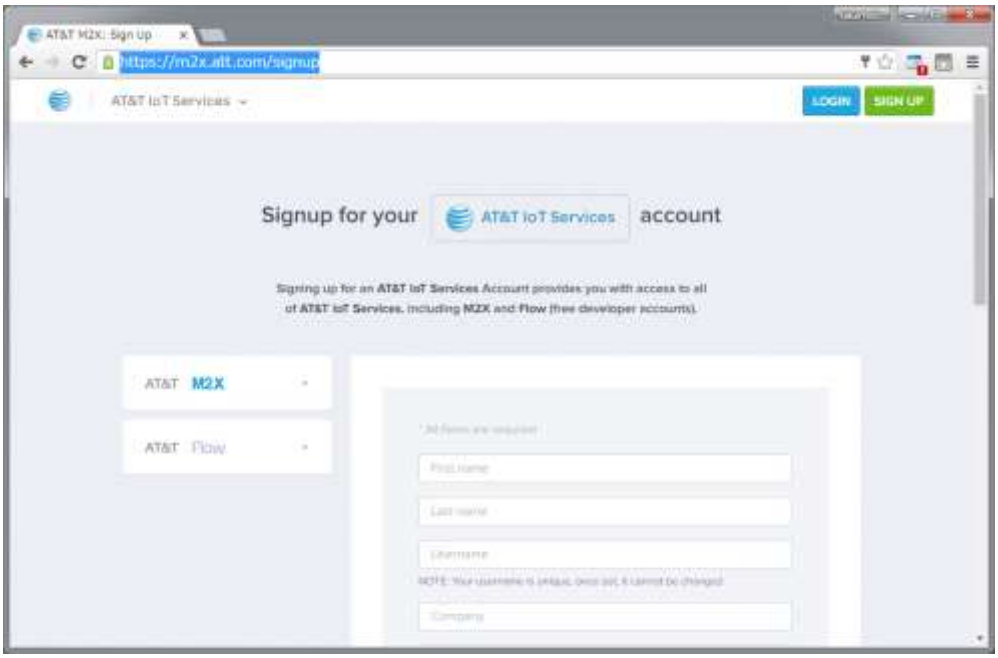

図３．m2x.att.com/signup のユーザ登録サイト

この図４でユーザ登録の詳細項目を記載しました。この入力欄では、全て入力は英数字で入力してください。

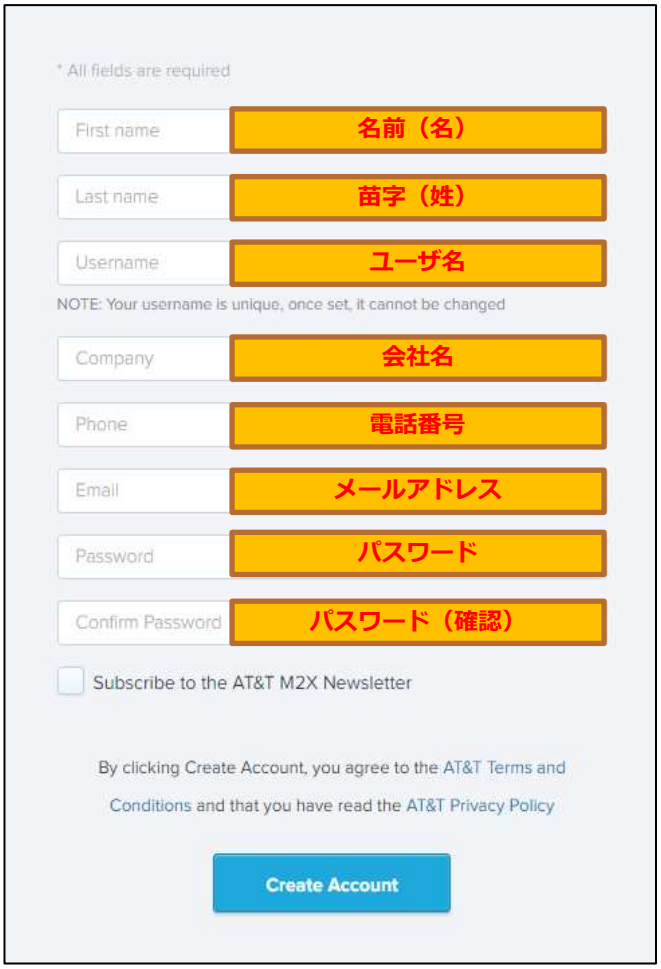

図４．m2x ユーザ登録入力欄

全ての入力欄の記入が終わったら、「Create Account」(アカウント作成)ボタンを押してください。直ぐにアカウントメールが入力したメールアドレスに送られ、図５の画面に切り替わります。この画面がアカウントによってログインされた画面となります。

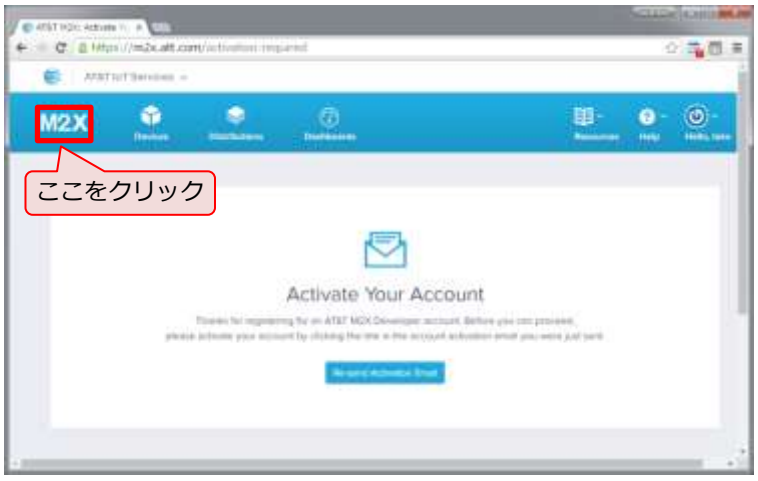

図５．m2x アカウント登録メール送信画面（初期画面）

それでは、このログインされた画面で、左上に表示されています「M2X」をクリックして、各種設定画面に切り替えてください。

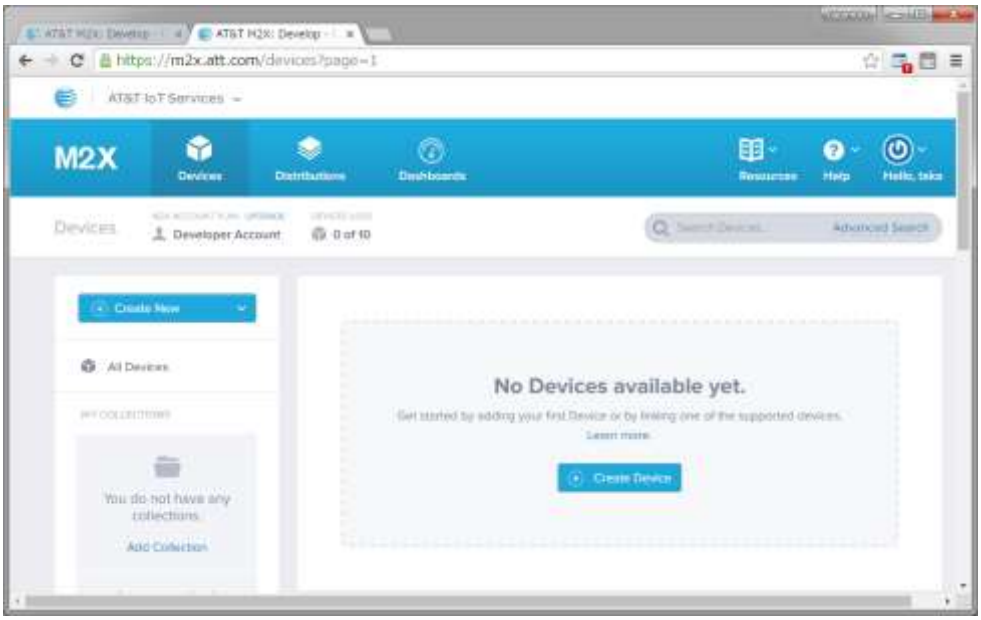

図６．各種設定画面（デバイス選択画面）

ここで一旦、m2x での IoT デバイス設定のための２つの設定情報について説明いたします。

デバイス（Device）：機器の設定（右画面の「Create Device」選択で設定：後述）

ストリーム（Stream）：デバイスに取りつくセンサ毎の設定

以下、この２つの設定を以下の画面で行っていきます。

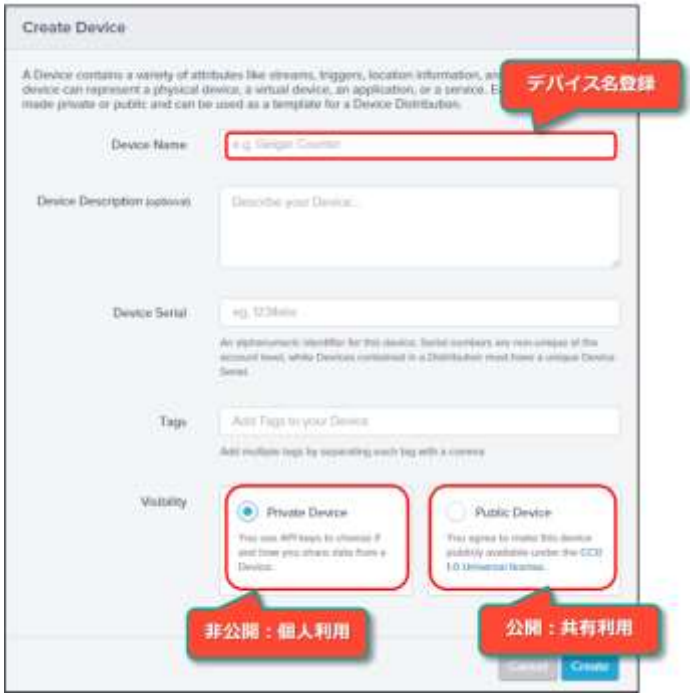

図７．デバイス登録画面

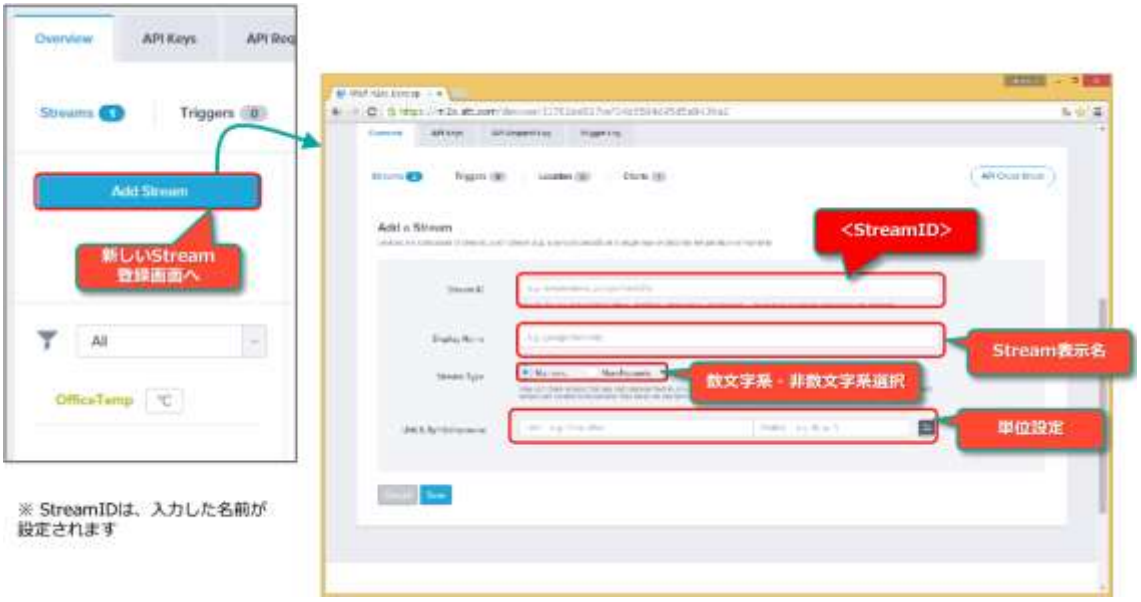

図８．ストリーム（各センサ値）の登録設定

上記の２つ（デバイスとストリーム）の設定によって、以下の２つの ID および Key が IoT デバイスとセンサとの関係でプログラム記述で引用する必要があります。

deviceID：デバイス機器 ID（システム側設定：m2x では 10 個まで無償登録可能）

x-m2x-key：１つのデバイス ID と関係づけられた m2x のキー（システム側設定）

StreamID：センサ毎の ID（ユーザ側設定）

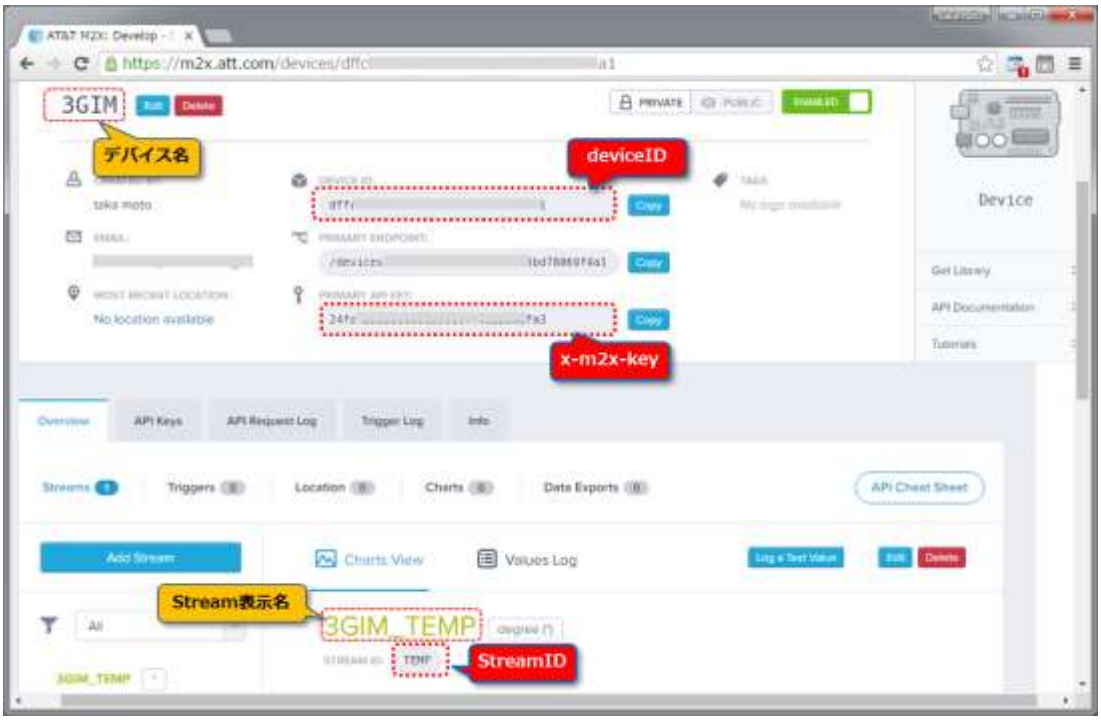

図９．デバイスとストリームの ID と Key

これらデバイスとストリームとの関係を再度、明確にしておきます。今回の IoT デバイス（Genuino101＋３GIM シールド＋３GIM）をデバイス ID として関係づけます。具体的

には、この IoT デバイスを m2x のデバイスと認識するだけです。図 9 でのデバイス名は「3GIM」としています。

この場合、自動的に **deviceID** と **x-m2x-key** が割り付けられます。つぎの **StreamID** は、IoT デバイスに取り付けた各センサを定義したもので、ストリーム（stream）表示名とストリーム ID（**StreamID**）をユーザ側で設定しますが、**StreamID** の方がプログラムの中で関係します。

ここでの例としては、図 8 での IoT デバイスの温度センサを定義するとき、スケッチと関連付け、しかも分かり易くするため、図 9 のようにストリーム表示名とストリーム ID を、「３GIM_TEMP」と「TEMP」としておきます。プログラム（スケッチ）では、**StreamID** 「TEMP」のみが必要となります。

３．m2x へのデータアップ方法

つぎに m2x にセンサ値を送信する方法を説明いたします。具体的には、前述した **deviceID** および **x-m2x-key**、**streamID** を使って、以下のようなフォーマット（形式）によって、M2X クラウドにデータをアップします。

リスト①：m2x 送信データ構造

```
$WP http://api-m2x.att.com/v2/devices/deviceID/updates/
"{$"values$" : {$"streamID$" : [{ $"timestamp$" : $"date-time$" , $"value$" : $"val$"}]}} "
"Host: api-m2x.att.com$r$nX-M2X-KEY:x-m2x-key:$r$nContent-Type:application/json$r$n"
```

ここで、「$WP」は、3GIM による **httpPOST コマンド**となります。（マニュアル参照）次の「http://api-m2x.att.com/v2/devices/**deviceID**/updates/」はクラウドにアップする **URL**、「"{$"values$" : {$"**streamID**$" : [{ $"timestamp$" : $"**date-time**$" , $"value$" : $"**val**$"}]}} "」 は **ボディ部**で、「"Host: api-m2x.att.com$r$nX-M2X-KEY:**x-m2x-key**:rnContent-Type:application/json$r$n"」 は **ヘッダー部**となります。

なお、この中の「**date-time**」はセンサ値の取得日時で、「**val**」は、取得したセンサ値を変換した値（摂氏温度など）となります。つまりいずれも可変のデータとなります。

この「**date-time**」は、以下のようなフォーマットとします。

```
2016-04-20T01:23:45+09:00
```

上記は、「年-月-日 T 時:分:秒+時:分」となっていて、「年」は 4 桁、以下「月」、「日」、「時」、「分」、「秒」は共に 2 桁の数字となります。また最後の「+時:分」は、国際時間の時差となります。日本時間だと、「+09:00」となります。ただ「+」がネット通信の際には特殊文字となるため、URL コードの「%2B」に置き換えて記述します。（マニュアル参考、後述）

それからリスト①で示した**ボディ部**は、**json 形式**とし、以下内容となります。

```
{"values" : {"streamID" : [{"timestamp" : "date-time" , "value" : "val" }]}}
```

先のリスト①で「$"」を使っているのは、3GIM での URL コード（特殊文字のみ）を指定しているためのもので、さらにこの内容をネット通信する場合には、「¥」を付けて、「¥$¥”」と変換したコードを付ける必要があります。（以下の json 形式およびリスト②参照）

```
{¥$¥"values¥$¥" : {¥$¥"streamID¥$¥" : [{¥$¥"timestamp¥$¥" : ¥$¥"date-time¥$¥" , ¥$¥ "value¥$¥" : ¥$¥"val¥$¥" }]}}
```

このように json 形式をネット通信で送信する場合、特殊文字の URL コード変換が必要で、具体的には＄コードや￥コードの挿入場所の理解が必要です。

さらに、これ全体を**ボディ部**として送るために、上記のリスト①のように前後に「"」（ダブルクォーテーション）で囲む必要があります。

４． IoT デバイスからの温度センサ値のクラウド・サーバアップ

それでは、既にパート２で紹介しました IoT デバイスを使って、温度センサの値を取得し、それを２分ごとに m2x クラウド・サーバにアップするスケッチを紹介しましょう。

以下のリスト②-1 の中の２箇所「**deviceID**」と「**x-m2x-key**」を先に登録した m2x サイトから読み込んできて書き替えてください（カットアンドペースト）。この２ヶ所の変更を行い、リスト②-1 とリスト②-2 からなるスケッチをコンパイルして、IoT デバイス（Geunino101）に書き込んで実行してください。これによって図＊のように温度センサ値がクラウドにアップされるようになります。

このリスト②-1 の中のストリーム名「**StreamID**」は、あらかじめ定義した「**TEMP**」に変更して記述しています。

リスト②-1：温度センサ値を３分ごとに m2x クラウドにアップするメインスケッチ

```
// リスト② : 温度センサ値を 3 分ごとに m2x クラウドにアップ
const unsigned long baudrate = 38400;

#define LIMITTIME 35000 // ms (3G module start time)
String URL      = "http://api-m2x.att.com/v2/devices/deviceID/updates/ ";
String HEADER = "¥"Host: api-m2x.att.com$r$nX-M2X-KEY: x-m2x-key$r$nContent-Type:application/json$r$n¥"";

void setup() {
  while(!Serial);
  Serial.begin(baudrate);
  Serial.println(">Ready. Initilaizing...");
  while( !_3Gsetup() ) {
      Serial.println(" Connect Error ... Stop");
      while(1);bodhi
  }
  Serial.println("Connected");
  pinMode(A0,OUTPUT);
  pinMode(A2,OUTPUT); digitalWrite(A2,HIGH);
}

void loop () {
  unsigned long tim = millis();
//-------------------- TEMP -----------------
  float   temp = analogRead(A1)*0.322-60.0;
  Serial.println("TEMP = " + String(temp));
  String body1 =  "¥"{¥$¥"values¥$¥" : {¥$¥"TEMP¥$¥" : [{ ¥$¥"timestamp¥$¥" : ¥$¥"";
  String body2 = "¥$¥" , ¥$¥"value¥$¥" : ¥$¥"" + String(temp) +"¥$¥"}]}}¥" ";
  if(_3G_WP("$WP " + URL + body1+ datetime()+  body2 + HEADER)){
    Serial.println(Serial1.readStringUntil('¥n')); }
  else Serial.println("Data Update false...");
  while(millis()-tim<120000); //waiting
}
```

実際の **deviceID** に変更

実際の **x-m2x-key** に変更

StreamID で「**TEMP**」設定

３GIM での httpPOST 送信

つぎにリスト②-2 について、紹介します。

リスト②-2：関数群のスケッチ

```
// ================== datetime ==============
// Get Date & Time (3GIM command)
//   return --> string "2015-12-23T01:23:45%2B09:00"
//   2015-09-20T14:05:12.345Z¥$¥" , ¥$¥"value¥$¥" : ¥$¥"32.0¥$¥"}]}}¥" ";
// + "2015-09-02T11:59:00%2B09:00&csv=TEMP01,2015-09-02T10:00:00%2B09:00LIGHT01,24.2"
String datetime() {
  String dtime;
  do {
     Serial1.println("$YT");
     while(!Serial1.available());
     dtime = Serial1.readStringUntil('¥n');
     Serial.println(dtime);
     delay(1000);
  } while ( !(dtime.indexOf("201") >0) );
  dtime.replace(" ","T"); dtime.replace("/","-");
  return(dtime.substring(7) + "$+09:00");
}
//============== 3G setup =================
boolean _3Gsetup() {
  Serial.println("start _3Gsetup()");
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH); delay(100);
  digitalWrite(7,LOW); // 3G shield --> digitalWrite(7,HIGH);
  Serial1.begin(baudrate);
 //------- 3G module begin & connect ---------
  String str;
  unsigned long tim = millis();
  do{   while(!Serial1.available());
        str=Serial1.readStringUntil('¥n');
  }while(!(str.indexOf("3GIM")>0) && (millis() - tim) <LIMITTIME);
  if( millis() -tim >= LIMITTIME) {
     return false;
  } else return true;
}
//============== $WP command =================
boolean _3G_WP(String command) {
 delay(10);
  Serial.println(command); // debug
  delay(10);
  Serial1.println(command);
  String rstr;
  unsigned long tim = millis();   // time set(ms)
  do{ while(!Serial1.available());
        rstr=Serial1.readStringUntil('¥n');
        Serial.println(rstr);   //debug print....
  }while(!(rstr.indexOf("$WP=")==0) && (millis() - tim) <LIMITTIME);// $WP return check
  return (rstr.indexOf("$WP=")==0);
}
```

ここでの以下の３つの関数群について説明します。

_3Gsetup 関数： 3 GIM の初期接続設定を行う関数

_3G_WP 関数：httpPOST によるインターネット接続関数

引数は、「$WP URL BODY HEADER」となります。（マニュアル参考）

datetime 関数：先に紹介したセンサ値取得日時で、3 GIM による日時よるフォーマットから、m2x クラウド用に変換する関数

それでは、実際に実行した事例の結果を表示させてみましょう。

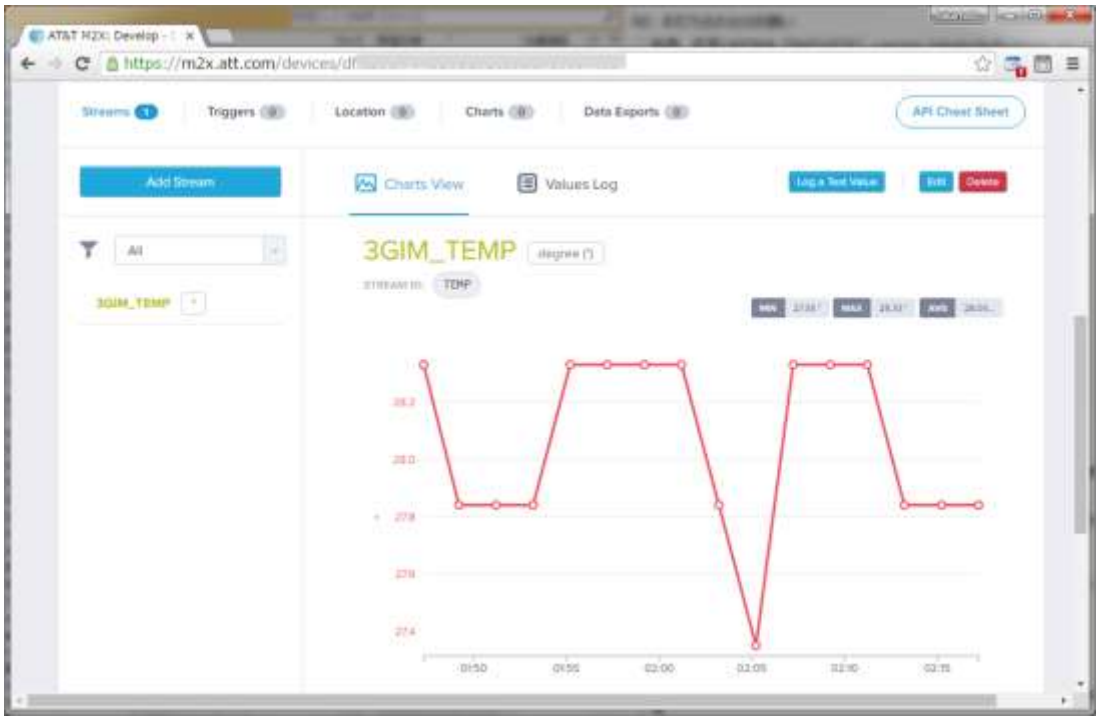

図 10. 出力グラフ事例

前に述べましたように、m2x での時間表示は、国際標準時（UTC）を用いて表示されます。日本時間から比べると、9 時間遅れていることから、表示された時間に 9 時間足した時間が日本時間となります。

5． センサ値によるトリガー（メール送信・ツイート送信）の処理

m2x では、センサ値の動きを監視することができる仕組みがあり、その仕組みを使うことで、異常事態を知らせたりすることができるようになっています。

ここではセンサの値がある値（閾値）を超えた場合に、メールを送信する、またはツイート送信するものをご紹介しましょう。

例えば、ここでの温度センサの値が 30 度を超えた場合には、自分にメールを送信するようにしましょう。メール送信については、すでにパート 2 のリスト③「メール送信プログラム（sendmail.php）」をご紹介しました。このプログラムを使い、トリガー（温度が 30 度を超えた場合）によって、メールを自分に送信するものをご紹介しましょう。

このことで人手によっていつもチェック確認することなく、機械的に・自動的に、異常を知らせるシステムとして利用できるようになります。

それでは、先の図 9 の中にある「Triggers」を選択します。（図 11 参照）

その選択によって表示されたトリガー設定画面での内容を紹介いたします。

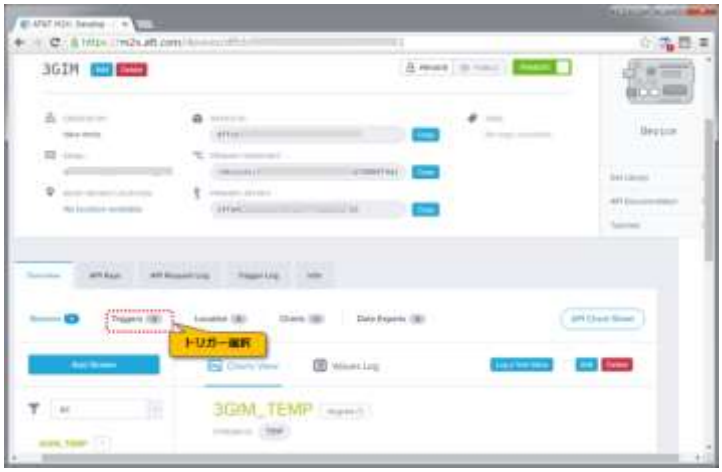

図１１．トリガー（Triggers）設定選択

図１２．トリガー（Triggers）詳細設定画面

ここで、「トリガー名」は任意に付けられますので、ここでは「temp over 30 degree」としておきます。次のストリームは、既に定義したストリーム表示名から選択できるようになっていますので、ここでは温度センサ値の「３GIM_TEMP」を選択します。
また比較演算は「＞」を選択し、トリガー値は「30」を入力してください。
このことで、30 を超えた場合に、以下の URL を起動するようになります。
この URL は上記の「メール送信 URL」というところに記入してください。

http://******.webcrow.jp/sendmail.php?email=**送信先アドレス**&temp=over%2030C

ここでの sendmail.php の説明は、パート 2 をご覧ください。赤文字の「over%2030C」は、メール送信される内容となります。
また、この部分を「ツイッター」によるものしたければ、以下の表記 URL を代わりに入れてみてください。(トークンについては、パート３で紹介しています)

http://arduino-tweet.appspot.com/update?token=**トークン**&status=**ツイート文**

この場合には、温度が 30 度を超えた場合に、ツイートされます。ただ、同じ文章は 2 度以上アップされない状況となります。

その他、トリガーの条件によっては、1 度だけの URL 起動や、閾値を超えたりした場合とか、同じ状態でもある時間間隔ごとに URL 起動する場合があります。
ここでは、詳しく説明しませんが、条件等を変えて、起動してみてください。

６． クラウド蓄積データの CSV ファイル出力

m2x 上に溜めたデータは、任意に CSV ファイルで出力することができます。つまり指定したデータを図 12 および図 13 のメニュー選択で、一時の CSV ファイルに出力し、ダウンロードすることができます。

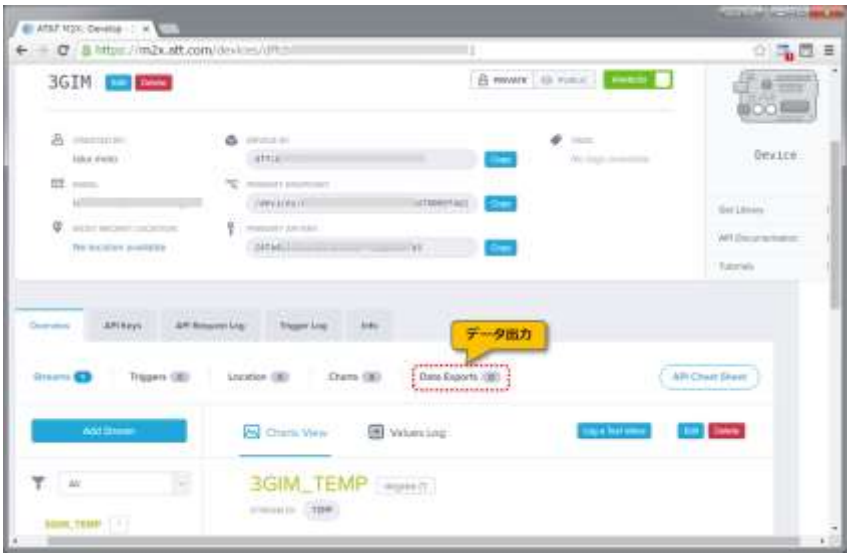

図１３．データ出力（Data Exports）選択

図 14．データ出力（Data Exports）設定画面

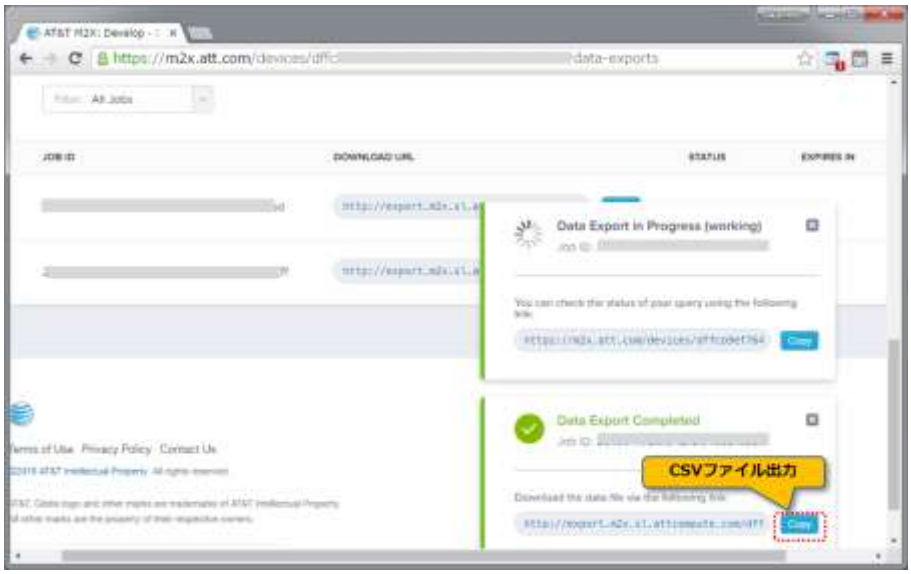

図 15．データ出力（Data Exports）設定画面

上記の図 13、図 14、図 15 に沿って操作を行うことで、一時の CSV ファイルをダウンロードすることができます。また、その中身は、図 16 の内容で、時間とセンサ値のペアで出力することができます。

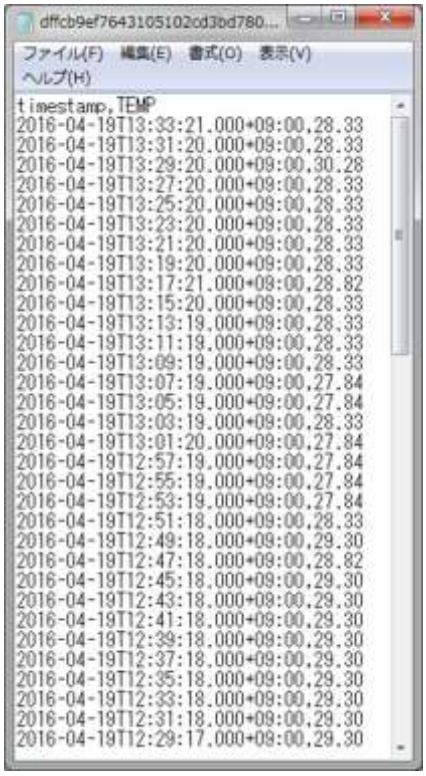

図 16. m2x データ出力事例（CSV ファイル）

パート１ ： 3 GIM を使った開発事例
http://tabrain.jp/3GIM_V2.0/Part1%20Arduino%20and%203GIM.pdf
パート２ ： 3 GIM を使った技術資料（遠隔制御・メール送信など）
http://tabrain.jp/3GIM_V2.0/Part2%20Arduino%20and%203GIM.pdf
パート３ ： 3 GIM を使ったツイート連携
http://tabrain.jp/3GIM_V2.0/Part3%20Arduino%20and%203GIM.pdf

以　上