株式会社タブレイン 高本孝頼
改訂 2017/08

パート５： 3GIM V2 を使った GPS(位置情報取得)を学ぶ

１． 3GIM V2 の GPS 機能とは

３GIM V2（HL8548-G）の機能のひとつに GNSS（全球測位衛星システム）機能、つまり位置情報取得機能があります。この機能を使うことで、地図上の緯度・経度が分かり、それをインターネットと接続して、他の遠隔地に知らせることができるようになります。

単独の GPS 機能を持った製品とは違い、ここで紹介する３G 通信付き GNSS 機能持つ３GIM は、GPS や GLONASS（ロシア製）の衛星から取得した位置情報をどこに居ようがリアルタイムでインターネットに知らせることができることとなります。つまり、３GIM だからこそ、取得した位置情報を、即座にネット上のサーバにアップし、現時点での位置情報を知らせたり、把握したりすることが可能となります。特に**３GIM V2** では、**アシスト GPS** といって、３G 通信を利用することで短時間で誤差の少ない位置情報取得が可能となっています。さらに **V2.2** からは、**アクティブ GPS アンテナ**による測位が可能となり、より GPS 衛星からの電波を取得できるようになりました。

このことで、追跡システムと呼ばれる機能として利用でき、動物や人間、それに自動車・バイク・自転車などに取り付けて位置情報を把握することが可能となります。具体的な事例としては、子供やお年寄りの外出時の見守りや、動物の移動追跡、愛車やバイク・自転車などの盗難防止対策、その他移動追跡の履歴管理などにも利用できるようになります。

どう利用するかは、ユーザ様の目的に委ねるとして、ここでは、どう技術的に作り上げるかをご紹介いたします。

また、その前に簡単な GPS 機能や、GPS 独自の**フォーマット NMEA** 仕様についてもご紹介していきます。

最後に、今後期待されている**みちびき対応**の GPS についても触れています。

２． ３GIM による位置情報取得とは

位置情報取得の原理は、上空に飛んでいる３０以上の人工衛星の中で、最低でも３つから４つ以上の衛星から送られてくる正確な時刻を取得することで、人工衛星毎の時差をもって位置情報（緯度と経度）を計算する仕組みとなっています。（別途他の専門情報をご参考ください）

ただ、衛星から送られてくる時刻情報の前に、飛んでいる衛星そのものの固有の位置情報や移動情報などの情報（ここでは**衛星情報**と呼ぶ）を取得する必要があります。そのため GPS 立上げ時（電源 On の状態）には、どうしても衛星情報の読み取りに時間が掛る問題があります。

１）アシスト GPS 機能について

３GIM V2 では、この初期立上げ時の衛星情報（位置・移動情報）を読み取るのを、衛星自体からではなく、３G 通信基地局を経由しインターネット上から取得します。このことで、初期立上げ時間がとても短縮され、しかも正確な位置情報も取得できるようになります。

この仕組みを**アシスト GPS**（Assisted GPS）機能と呼び、単体の GPS モジュールよりも３GIM でのこの機能を使った方が、短時間かつより誤差が少なく立上げ時の位置情報取得が可能となります。

現在タブレインで GPS 単体のモジュールと３GIM の GPS 機能を調べたところでは、立上げ（コールドスタート）時の位置情報取得時間は、GPS 単体モジュールだと 2 分から 5 分ほど掛かりますが、アシスト GPS を使った３GIM だと 20 秒から 1 分ほどと短時間になります（利用環境・状況により取得時間は異なる）。また位置情報誤差も３GIM の方が 10m以下といい結果が出てきています。（ただ、３GIM の誤差が少ないのはアシスト GPS 立上げ時）

ただ、このアシスト GPS 機能は、先に述べたように 3G 通信の基地局とのネット接続が必須で、SIM カードが挿入されたネット接続ができることが前提となります。よってネット通信接続が発生し、何らか通信費が割高になる場合もあります。現時点での調査では、一度３G 通信基地局からの衛星情報を取得すると、その後衛星情報が途切れない状態では３G 通信での接続はないようです。

２）３GIM V2 搭載 HL8548-G モジュールの GPS 機能について

３GIM V2 搭載の通信モジュール **HL8548-G**（シエラワイヤレス製）は、**衛星測位システム**（GNSS：Global Navigation Satellite System）を持った３G 通信モジュールとなります。この GNSS 機能としては、米国の **GPS**(Global Positioning System) 機能とロシア政府の **GLONASS** 機能の２つを搭載していて、多くの人工衛星（約３０以上）から位置情報を取得できる仕組みとなっています。

この HL8548-G モジュールは、先に述べたアシスト GPS 機能を持つことで、３G 基地局を通じ、Google 社の **supl.google.com** から衛星情報を得るようにしています。現在、このサイトは無償で利用できるようになっています。

３）３GIM V2 でのアシスト GPS 機能設定について

３GIM V2 でアシスト GSP を使うようにするには、以下の AT コマンドで、あらかじめ SIM カードの APN 情報（ここではユーザ名とパスワード）を設定する必要があります。

```
at+wppp=2,4,”username“,”password”
```

ここでの「**username**」と「**password**」は、利用している SIM カードの APN 情報のユーザ名とパスワードとなります。

これを、Arduino などのスケッチで記述するには、一旦「**$YA**」（AT コマンドパススルーモード）を使って、AT コマンドモードに切り替えて処理を行う必要があります。この処理としては、以下のようなスケッチを入れておくことで、アシスト GPS 機能が設定されます。

```
Serial1.println("$YA 1");
delay(10);
Serial1.println(“at+wppp=2,4,¥“username¥”,¥”password¥"");
```

ここで、「Serial1」は、パート２で紹介した Arduino M0 Pro や Genuino101 のハードウェアシリアル通信の UART となります。（Serial1 は変更可能）

また、「$YA 1」に記述した「1」は、AT コマンドパススルーモードの時間を 100ms と設定したことになります。よって 100ms（0.1 秒）後には、＄コマンドモード（ファームェア・モード）に戻ってくることなります。

このことで上記の３行目の「**Serial1**.println」行の処理によって、アシスト GPS を設定したモードとなります。この「$YA 1」によって、＄コマンドモードに切り替わるとき「>>」が応答としてシリアル通信（Serial1）によって返ってきます。

４）GPS 専用出力フォーマット NMEA 仕様（プロトコル）について

つぎに GPS 機能を使う上での一般的な出力仕様（フォーマット）をご紹介しておきましょう。GPS 機能を使って位置情報を取得する場合、GPS モジュールは、1 秒間隔で位置情報を出力します。出力されるフォーマットは、一般に **NMEA 仕様**となっています。この NMEA 仕様は、ソナーや風速計、音波探査機などの仕様で、GPS でも使われているフォーマットでもあります。

この NMEA には、いくつもの出力仕様（**センテンス**）があり、＄GPRMC や$GPGGA、$GPGSA、$GPRMC などいろんな種類のセンテンスが存在します。この中で代表的なセンテンスとして、以下の 3 種類をご紹介しておきます。また内容の仕様概要についてもまとめておきます。

表１．NMEA フォーマットのセンテンス

センテンス	NMEA センテンス内容（概要：一部）	備考（AT コマンド）
＄GPGGA	UTC,緯度,N,経度,E,品質,使用衛星数,水平精度低下率,海抜高さ,M,ジオイド高さ,M,差動基準地点 ID,チェックサム	at+gpsnmea=1,,1
＄GPGSA	モード,特定タイプ,衛星番号,位置精度低下率,水平精度低下率,垂直精度低下率,チェックサム	at+gpsnmea=1,,2
＄GPRMC	UTC,A,緯度,N,経度,E,移動の速度,移動の真方位,日付,磁北と真北の間の角度の差,方向,モード,チェックサム	at+gpsnmea=1,,4

５）３GIM による GPS 出力「＄LG」コマンドについて

３GIM を使って GPS 機能を使うには、「$LG」コマンドを利用し、以下のように使います。

＄LG コマンドパラメータ

＄LG **method** ［ **option** [**repeatCount**]]

ここで、「**method**」は、旧３GIM V1.0 との整合性を保つためにつけた引数では、実際には任意の文字（列）でも構いません。

つぎの「**option**」は、測位のオプションで、以下の数値の何れかから選択できます。

0: 緯度・経度を表示（V1.0 と同じ）【省略値】

1: 緯度・経度の他に UTC、品質、捕捉衛星数、高度を出力

2:**＄GPGGA** センテンスの生データを出力（ダブルクォート付）

3: **NMEA フォーマット**（各 NMEA センテンス）出力※

9:ロケーションサービスを停止（SLEEP）させる（method は任意文字/'x'などでも可）

最後の「**repeaCount**」は、測位・出力行数（回数）となり、設定する場合には、１から最大 100 までの整数を指定します。上記 option= 0 ,1,2,3 の場合の出力行数となります。この応答値は、それぞれ以下のようになっています。

表２．＄LG の応答値

method＝０の場合：$LG=OK latitude longitude¥n
method＝１の場合：$LG=OK latitude longitude utc quality number height¥n
method＝２の場合：$LG=OK “gpgga”¥n
method＝３の場合：$LG=OK ¥nnmea_sen¥n･･･

この中の各項目の内容は表 3 ようになります。

表３．＄LG の応答の各値

latitude	緯度（北緯、9.99999 形式、ただし桁数は場合により可変）
longitude	経度（東経、9.99999 形式、ただし桁数は場合により可変）
utc	世界標準時間（協定世界時） 日本は 9 時間プラスする必要がある
quality	位置特定品質。０：位置特定できない、１：GPS（標準測位サービス）モード、２：differencetial GPS（干渉測位方式）モードの何れか
number	捕捉衛星数
height	GPS アンテナの海抜高さ（m）
gpgga	NMEA で規定されている**＄GPGGA** センテンスの生データ（ただし末尾の改行は削除）
nmea_sen	各 NMEA フォーマット出力

３． ３GIM による位置情報取得スケッチ

それでは、３GIM の位置情報取得の最も簡単なスケッチをご紹介しましょう。
ここでもパート２で紹介した Genuino101+３GIM シールド＋３GIM を使って行います。

１）アシスト GPS 機能による位置情報取得スケッチ

アシストGPS機能を使って位置情報を取得するスケッチをご紹介いたします。リスト①をIoTデバイスに書き込んで、USB ケーブルによる PC 側のシリアルモニタ画面で位置を表示してみてください。

リスト①：アシスト GPS による位置情報取得スケッチ

```
// リスト①：アシスト GPS を使った事例
const unsigned long baudrate = 38400;
void setup() {
  while(!Serial);
  Serial.begin(baudrate);
  Serial1.begin(baudrate);
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH);delay(100); digitalWrite(7,LOW);
  Serial.println("Ready...");
  delay(14000);
  Serial1.println("$YA 1");
  delay(5);
  Serial1.println("at+wppp=2,4,¥"username¥”,¥”password¥"");
  while(Serial1.readStringUntil('¥n').indexOf(">>")<0);
  Serial.println("start...");
```

アシスト GPS 設定

```
    Serial1.println("$LG x 0");
    while(!Serial1.available());
    Serial.println(Serial1.readStringUntil('¥n'));
    Serial.println("end");
}
void loop() { }
```

位置情報取得

それでは、結果はつぎのようになりましたでしょうか？

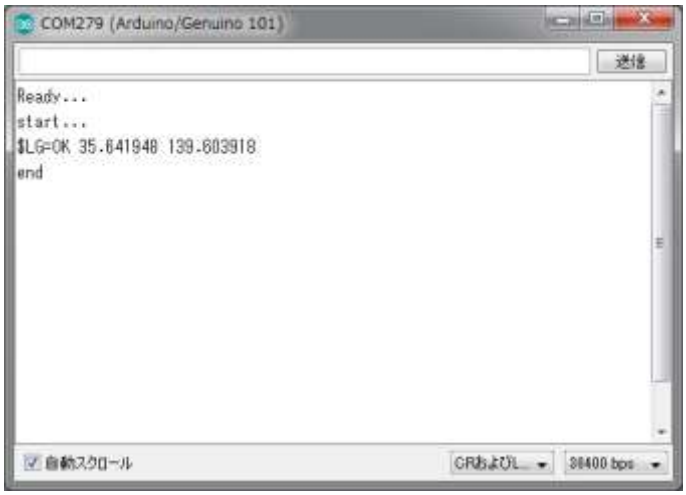

図１．出力結果例

結果は、わずか 1 分以内で出てきましたでしょうか。もし「＄LG=NG」などが出てきた場合には、SIM カードの設定ミスやアンテナの問題、その他電源不足、GPS 取得が難しい屋内とか、原因がいろいろと考えられますので、マニュアルや Wiki ページ（http://3gim.wiki）を参照しながら対応してみてください。

２）出力座標値について

それでは、出てきた結果の位置座標（緯度、経度）を Google マップ上で確認してみましょう。

具体的には、Google マップの検索欄に出力された座標値を入力して確認します。（ここでは図１で出てきた緯度・経度を使い「https://www.google.co.jp/maps/」で確認しています）

図２．Google マップ上に緯度と経度を指定して地図上の位置確認

3）$LG コマンドのオプションについて

つぎにリスト①の中の赤字の「**$LG x 0**」を先の$LG コマンドパラメータを見て、以下のように変更して実行してみてください。ここで表示される応答は、もちろん現在地が表示されます。

$LG x 1 →

```
$LG=OK 35.641991 139.604126 091254.000 1 8 37.
```

$LG x 2 →

```
$LG=OK "$GPGGA,091804.435,3538.5106,N,13936.2532,E,1,08,1.2,71.6,M,39.4,M,,*68"
```

$LG x 3 20 →

```
$GPGSA,A,3,01,07,08,10,11,16,26,27,,,,,,1.2,*38
$GPVTG,0.0,T,,M,0.4,N,0.8,K,A*01
$GPGSV,2,1,08,01,28,204,18,07,41,294,24,08,66,330,29,10,19,079,19*71
$GPGSV,2,2,08,11,45,227,17,16,42,104,20,26,16,118,25,27,52,039,31*71
$GLGSV,1,1,02,82,66,325,29,79,39,159,29*60
$GLGSA,A,3,,,,,,,,,,,,,,,1.2,*2D
$GNGNS,091802.219,3538.5088,N,13936.2503,E,AN,08,1.2,73.3,39.4,,*5F
$GNGSA,A,3,01,07,08,10,11,16,26,27,,,,,2.6,1.2,2.4*24
$GPGGA,091802.219,3538.5088,N,13936.2503,E,1,08,1.2,73.3,M,39.4,M,,*64
$GPGLL,3538.5088,N,13936.2503,E,091802.219,A,A*53
$GPRMC,091802.219,A,3538.5088,N,13936.2503,E,0.4,0.0,210416,,,A*60
$GPGSA,A,3,01,07,08,10,11,16,26,27,,,,,,1.2,*38
$GPVTG,0.0,T,,M,1.3,N,2.5,K,A*08
$GPGSV,2,1,08,01,28,204,18,07,41,294,24,08,66,330,29,10,19,079,19*71
$GPGSV,2,2,08,11,45,227,17,16,42,104,20,26,16,118,25,27,51,039,31*72
$GLGSV,1,1,02,82,66,325,29,79,39,159,29*60
$GLGSA,A,3,,,,,,,,,,,,,,,1.2,*2D
$GNGNS,091803.340,3538.5028,N,13936.2530,E,AN,08,1.2,90.8,39.4,,*5F
$GNGSA,A,3,01,07,08,10,11,16,26,27,,,,,2.6,1.2,2.4*24
$GPGGA,091803.340,3538.5028,N,13936.2530,E,1,08,1.2,90.8,M,39.4,M,,*64
```

4）NMEA 仕様による度・分の変換について

上記$LG コマンドパラメータの「**option**」が「0」と「1」の場合は、緯度と経度が単位「度」で出力されますが、「2」と「3」の場合は、NMEA 出力仕様の緯度と経度が「度」と「分」の単位でしかも 100 倍に桁上げした数値で出力されます。そこで、「2」と「3」の場合の出力を Google マップで表示させるには緯度と経度の「度」単位の変換計算が必要となります。このことを上記の出力例の最終行「**$GPGGA**」の緯度と経度の以下の数値例から見てみましょう。

「**3538.5028**」と「**13936.2530**」

この場合、3538.5028 は、**35 度 38.5028 分**、13936.2530 は、**139 度 36.2530 分**となります。

それでは、これを度のみに変換する関数をご紹介します。
その前に、度と分の値を度のみにするには、

```
3538.5028 →  35+38.5028/60 =  35 + 0.641713333 = 35.641713333
13936.2530→139+36.2530/60 = 139 + 0.604216667 = 139.604216667
```

となります。(先の Google マップで同様にこの値を使ってみてください)
よって、NMEA フォーマットで出力される度・分を度に変換する上では、以下のような処理が必要となります。
取得した緯度・経度の値（度・分）を「val」とすると

```
      int(val /100)+(val -int(val /100)×100)/60
    = int(val /100)+ val /60 – int(val /100)×100/60
    =  val /60 + int(val /100)×40/60
    =  val /60 + int(val /100)×2/3
```

となります。これによって 度･分で出力された NMEA フォーマットを度に変換する関数は、以下のとおりとなります。

リスト②：NMEA フォーマット度換算式

```
float valcalc( float val ) {
  return (val/60.0 -((int)val/100)*2.0/3.0);
}
```

４．取得した位置情報をサーバにアップするスケッチ

今度は取得した位置情報をサーバにアップするスケッチに挑戦してみましょう。ここでは、位置情報の取得時刻も一緒にペアでサーバにアップするようにします。また、位置情報を取得する時間間隔を 10 秒毎にし、サーバへのデータアップを 1 分毎（60 秒毎）にして、処理を考えてみましょう。
ここでは、リスト③として 3 つに分けて説明しています。実際には、これらを連結してご利用ください。

１）IoT デバイス側のスケッチ

IoT デバイス側、つまり Arduino 側のスケッチの冒頭にシリアルモニタ画面および 3 GIM の両方の通信速度設定「BAUDRATE」を設定し、その後に、TGPS と TMGPS とで、GPS 取得間隔（ミリ秒）とネット接続しデータをアップする間隔（ミリ秒）を設定しています。

リスト③-1：位置情報取得サーバアップ・スケッチ

```
#define BAUDRATE 38400
#define TGPS   10000   // GPS 取得間隔（ミリ秒）
#define TMGPS 60000   // $WP アップロード間隔（ミリ秒）
String URL      = "http://サーバ URL/makefile.php?file=";
```

サーバ側 PHP プログラム（後述）

それと後で紹介しますサーバ側のプログラム（makefile.php）とファイル名の引数までを定義した URL を設定しておきます。
以下に関数群を紹介していきましょう。

リスト③-2：位置情報取得サーバアップ・スケッチ

```
void setup() {
  while (!Serial);//Genuino101 対応
  Serial.begin(BAUDRATE);
  Serial1.begin(BAUDRATE);
  Serial.println("start 3GIM setup()");
  pinMode(7, OUTPUT);
  digitalWrite(7, HIGH); delay(100);//待機重要
  digitalWrite(7, LOW); // 3GIM shield 対応
  delay(14000);//立ち上げ待機時間
  Serial1.println("$YA 1"); delay(20);
  Serial1.println("at+wppp=2,4,¥"username¥",¥"password¥""); delay(10);
  while (Serial1.readStringUntil('¥n').indexOf(">>") < 0);
  Serial.println("Connected");
}
// ----------------- loop --------------------
void loop () {
  unsigned long tim = millis();
  static String str;
  String dtime, fname, gpss;
  do {
    unsigned long tm = millis();
    LED_SW(false);
    //      Serial.println("GET_GPS start");
    String gps = GET_GPS();//位置情報取得
    if (gps.length() > 0) {
      LED_SW(true);
      dtime = DATETIME();
      Serial.println(dtime + " / GPS = " + gps);
      fname = dtime.substring(2, 10) + ".dat";//ファイル名
      dtime = dtime.substring(11);//位置情報取得時刻
      //       Serial.println("Fname = " + fname   + " / " + "dtime= " + dtime);
      gpss += dtime + "G" + gps ;//    Serial.println(gpss);
    };
    while ((millis() - tm < TGPS) && (millis() - tim < TMGPS - TGPS * 0.9));
    if (millis() - tim < TMGPS - TGPS * 0.9) gpss += "%0D%0A";//改行
  } while ( millis() - tim < TMGPS - TGPS * 0.9);
  //------------ GPS get & WP upload----------
  //  Serial.println("GPS = " + gpss);
  if (gpss != "") {
    WG_UPLOAD(URL + fname + "&data=" + gpss);
    while (millis() < (tim + TMGPS));//時間調整（待機）
    Serial.println(" ---------- end loop -------------");
  }
}
```

アシスト GPS 設定

位置情報と時刻取得

サーバにデータアップ

初期設定の setup 関数では、 3 GIM シールドによる 3 GIM の電源 On と AT コマンドによるアシスト GPS 設定（at+wppp コマンド利用）を行っています。

次のループ（loop 関数）では、位置情報取得と時間（日時）取得を行い、日時からファイル名作成（月日をファイル名）それにサーバアップの時分秒に変換し、位置情報取得間隔ごとに行い、位置情報データがまとまった段階でサーバにアップします。

リスト③-３：位置情報取得サーバアップ・スケッチ

```
//=============== get Date Time   ==================
String DATETIME() {
  String dtm;
  do {
    Serial1.println("$YT"); delay(10);
    do {
      dtm = Serial1.readStringUntil('¥n');
    } while (dtm.indexOf("$YT") < 0);
  } while ( !(dtm.indexOf("201") == 7) );//201*年を確認
  dtm.replace(" ", "T"); dtm.replace("/", "-"); dtm.replace(":", "");
  return (dtm.substring(7));
}
//===============    LED on/off    ==================
void LED_SW(boolean sw) {
  Serial1.println("$YL " + String((int)sw));
  while (Serial1.readStringUntil('¥n').indexOf("$YL") < 0);
}
//===============    GET GPS    ==================
String GET_GPS() {
  Serial1.println("$LG x 0");
  String gps;
  do {
    gps = Serial1.readStringUntil('¥n');
    //      Serial.println("GPS = " + gps);
  } while (gps.indexOf("$LG") < 0);
  if (gps.indexOf("OK") > 0) {
    gps = gps.substring(7);
    gps.replace(" ", ",");
  } else gps = "";
  //   Serial.println("GPS = " + gps);
  return gps;
}
//===============    httpGET    ==================
void WG_UPLOAD(String str) {
  do {
    Serial1.println( "$WG " + str);
    do {
      str = Serial1.readStringUntil('¥n');
    } while (str.indexOf("$WG") < 0);
  } while (str.indexOf("OK") < 0);
}
```

日時取得関数

LED On/Off 関数

位置情報取得関数

httpGET 関数

ここでは、日時取得関数（**DATETIME**）、LED On/Off 関数（**LED_SW**）、位置情報取得関数（**GET_GPS**）、それに httpGET 関数の４つのサブ関数（**WG_UPLOAD**）を設定しています。これらは、いずれも＄コマンドの書き出しと応答の読み込みを以下のようにペアで行っていることに注意してください。

```
Serial1.println( "$WG " + str);
      do {
        str = Serial1.readStringUntil('¥n');
      } while (str.indexOf("$WG") < 0);
```

応答値を使う場合

または

```
Serial1.println("$YL " + String((int)sw));
  while (Serial1.readStringUntil('¥n').indexOf("$YL") < 0);
```

応答値を使わない場合

ここで注意してほしいのは、応答値を使う場合と使わない場合で処理が異なることです。特に上段の場合、文字列変数「str」を使って一旦応答値を取り出し、この後利用しています。

２）サーバ側位置情報蓄積プログラム

それでは、つぎにサーバ側の位置情報取得蓄積プログラム（php による makefile.php）を掲載しておきます。引数を、「file」と「data」とし、それぞれ IoT デバイスから取得したデータを先の WG_UPLOAD 関数を使って通信します。

リスト④：サーバ側位置情報蓄積プログラム（makefile.php）

```
<?php
   $file=$_GET["file"];
   $data=$_GET["data"] . "¥r¥n";
   $fp=fopen("./dat/" . $file,'a');
   fwrite($fp, $data);
   fclose($fp);
   ?>
```

ここで、＄WG コマンドでは、リスト③では、以下のように設定しています。

```
WG_UPLOAD(URL + fname + "&data=" + gpss);
```

この「URL+fname+”&data=”+gpss」は、実際には、以下のようになっています。

「http://**サーバ URL**/makefile.php?file=16-04-22.dat&data=182402G35.641919,139.604251」

つまり、

「fname=16-04-22.dat」

「data=182402G35.641919,139.604251」

となって、引き渡されます。

３）実行事例と出力ファイル内容

それでは実際に実行した場合のシリアルモニタ画面と、サーバ側に出力されたファイルの内容を掲載しておきます。

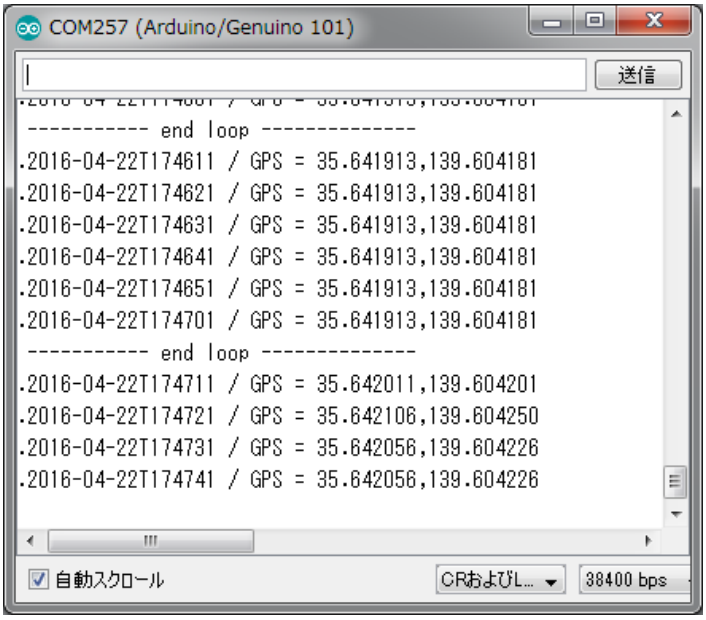

図３：リスト③によるシリアルモニタ画面出力例

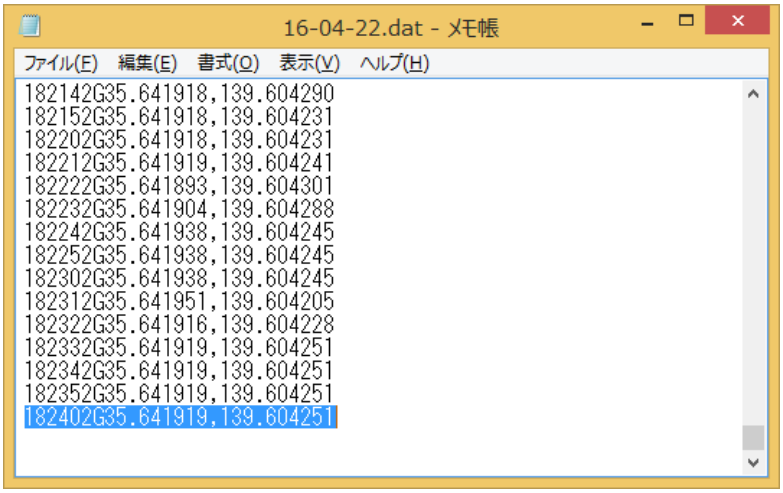

図４：リスト④よるサーバ側の出力ファイル内容例

５．出力された位置情報を使った位置情報の確認

上記図４で出力されたファイルを実際に Google マップ上に表示させてみたのが、図５のように時系列に取れた値を地図上にライン（直線）でプロットし、取得した位置にマークをプロットしています。

ここでのこのためのプログラムは割愛しますが、この後の処理はいろいろと展開ができ、さまざまな可能性もあるでしょうから、挑戦してみたらいかがでしょうか？

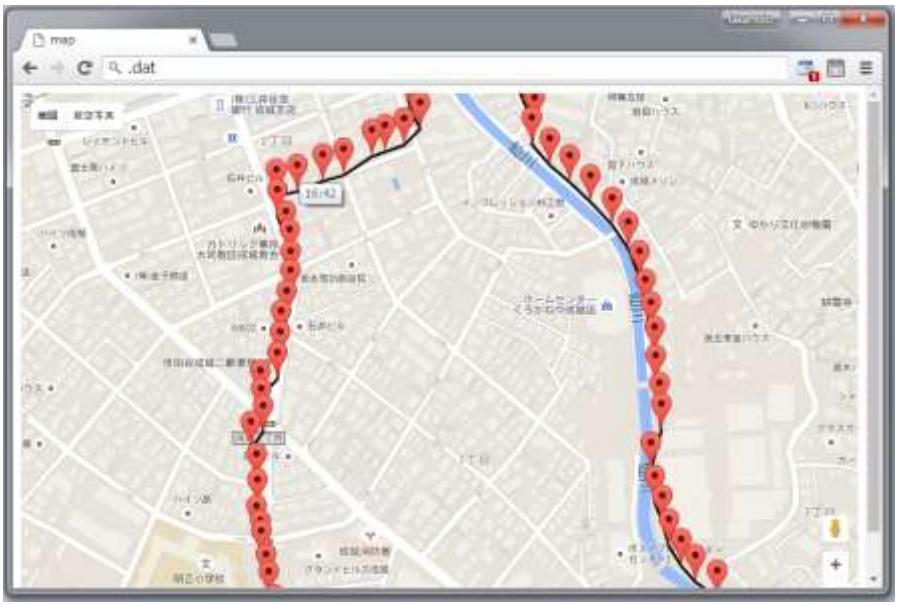

図５．IoT デバイスを使って位置情報取得した座表値の Google マップ表示例

6．アクティブ GPS アンテナについて

３GIM V2.2 からアクティブ GPS アンテナが利用できるようになりました。アクティブ GPS アンテナとは、数メータの長いケーブルの GPS アンテナなどで、車両などに取り付けてあるようなものとなります。

ここでは、このアクティブ GPS アンテナの設定について紹介します。

このアクティブ GPS アンテナは、HL8548-G の AT コマンド（at+gpsconf）を使って、切り替えることができます。ただし出荷時は、アクティブ GPS アンテナ利用モードとなっています。この切り替えは、不揮発性メモリーに保存されますので、一度設定すると電源が切れても保存されたままとなります。

at+gpsconf=1,1	← アクティブ GPS アンテナ利用モード
at+gpsconf=1,2	← アクティブ GPS アンテナ未使用モード

アクティブ GPS アンテナ利用モード時は、アンテナへの電源供給を行うため、わずかですが消費電流がアップします。

7．みちびき対応 GPS について

すでに日本の上空には「みちびき」対応の GPS 衛星が打ちあがりました。（2018 年春からみちびき対応になるとの情報が入っています）

３GIM でも、この「みちびき」対応にならないかとの問い合わせを多く頂きますが、現状 HL8548－G のファームウェアの書き換えで対応できるとの情報が入ってきています。これについては、新たな情報が入りましたら、公開していきますので、今しばらくお待ちお願いいたします。

これまでの参考資料
パート１：http://tabrain.jp/3GIM_V2.2/Part1%20Arduino%20and%203GIM.pdf
パート２：http://tabrain.jp/3GIM_V2.2/Part2%20Arduino%20and%203GIM.pdf
パート３：http://tabrain.jp/3GIM_V2.2/Part3%20Arduino%20and%203GIM.pdf
パート４：http://tabrain.jp/3GIM_V2.2/Part4%20Arduino%20and%203GIM.pdf

以　上