

3 GIM 2.1

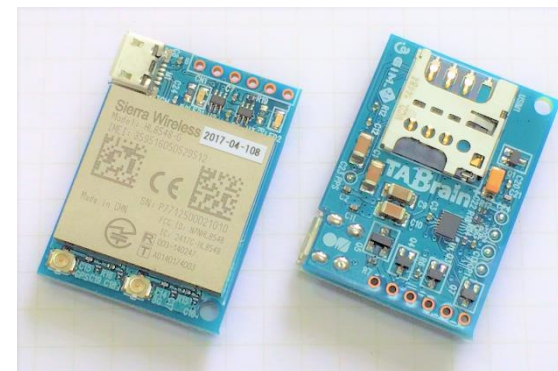
本マニュアルは、3G IoT Module V2.1 のご利用に当たっての説明資料です。
一部の機能詳細においては、既に販売しています3GIMシールドやIoTABシールドの資料
等もご参照ください。

従来品3GIM V1.0との相違点は、最終補足ページに記載しています。

以下の開発事例/技術資料もご参考ください。

- 1) 技術情報①：[開発事例](#)
- 2) 技術情報②：[遠隔制御・遠隔モニタリング（メール送信）](#)
- 3) 技術情報③：[ツイート連携](#)
- 4) 技術情報④：[クラウド連携](#)
- 5) 技術情報⑤：[アシストGPS](#)
- 6) [Arduinoライブラリ利用マニュアル](#)
- 7) [エラーコード一覧表](#)

世界最小クラス・省エネタイプ
3G通信モジュール（ボード）



3 GIM (3G IoT Module) V2.1 利用マニュアル

株式会社 タブレイン
3GIM-V2.1R01

注) V2.1ではファームウェアの一部をバージョンアップしています。
(有償にてバージョンアップしています：補足資料参照)

3 GIM V2.1 の優位性

3 GIM V2.1は、他の通信モジュールに比べ多くの優れた点を持っています。ここにご紹介いたします。

- 1) 3GIMのメインインタフェースはUARTで簡単（別途USB接続も利用可能）
- 2) 分かりやすいマニュアルと豊富なサンプル付き
- 3) 世界最小クラスでコンパクト
- 4) 省エネモードの取り入れが簡単
- 5) デバイ스에組み込むためのカスタマイズが容易
- 6) 超安価なSIMカードも利用可能（0円も可能）
- 7) 位置情報測位が速いアシストGPS利用可能
- 8) 豊富な開発事例及び安定稼働
- 9) 試作から実運用まで幅広く利用可能
- 10) Arduinoの開発環境や豊富な資産が利用可能
- 11) IoT試作環境として技術情報が豊富
- 12) ATコマンドを隠蔽した中学生でも分かる仕様実現

などなどまだ他にも挙げられます。

この他、3 GIM V2.1 の保守サポートは、以下のWikiページにて対応しています。

<http://a3gs.wiki.fc2.com/>

またFacebookにて、最新情報を掲載しています。

<https://www.facebook.com/tabrain>

もくじ

第1章 3GIM V2.1 の概要

p.3

1. はじめに
2. 3GIM V2.1 の外観
3. 3GIM V2.1 の機能概要
4. 3GIM V2.1 の仕様概要
5. 3GIM V2.1 のピンコネクタ配置
6. 3GIM を利用するソフトウェアについて
7. ご利用上の注意点

【ご参考】Arduinoで動かす配線・接続

第2章 \$コマンドインタフェース

p.12

1. UART送受信インタフェースの概要
2. 3GIM V2.1 コマンド一覧
3. 3GIMのインタフェース形式（共通事項）
4. 3GIMのインタフェース形式（補足事項）
5. \$コマンドの送信・受信の処理方法

第3章 3GIMコマンド・応答（レスポンス）

P.19

1. System関連
2. SMS関連
3. GPS関連
4. Web関連
5. TCP/IP関連
6. Profile関連

第4章 応用事例プログラミング

P.72

1. Arduinoでのシリアルモニタ操作
2. 3GIMでのツイッター連携使用例
3. 3GIMでのクラウド連携使用例（1）
4. 3GIMでのクラウド連携利用例（2）
5. 3GIMでのクラウド連携利用例（3）
6. GPS機能を使った応用例
7. Arduino関連ライブラリ（a3gim2）
8. サンプルツール群

第5章 Arduino用a3gimライブラリ群

P.121

1. a3gimライブラリとは
2. a3gimライブラリ関数群
3. コントロール関連関数
4. ショートメッセージ関連関数
5. Web関連関数
6. 現在位置取得（GPS）関連関数
7. 通信その他機能関数
8. TCP/IP関連関数
9. プロファイル関連ほか関数
10. サンプルスケッチ群の実行例

補足資料

P.195

- 【補足資料1】 3GIMコマンド・応答一覧表
- 【補足資料2】 5Vから3.3Vを作り出す回路例
- 【補足資料3】 トラブルシューティング
- 【補足資料4】 3GIM V2.1 外形寸法
- 【補足資料5】 3GIM サポートサイト
- 【補足資料6】 3GIM関連商品のご紹介
- 【補足資料7】 3GIM V1 との相違点
- 【補足資料8】 ラズベリーパイでの3GIM利用
- 【補足資料9】 ファームウェアのバージョンアップ対応

もくじ

1. はじめに
 2. 3GIM V2.1 の外観
 3. 3GIM V2.1 の機能概要
 4. 3GIM V2.1 の仕様概要
 5. 3GIM V2.1 のピンコネクタ配置
 6. 3GIM を利用するソフトウェアについて
 7. ご利用上の注意点
- 【ご参考】 Arduinoで動かす配線・接続



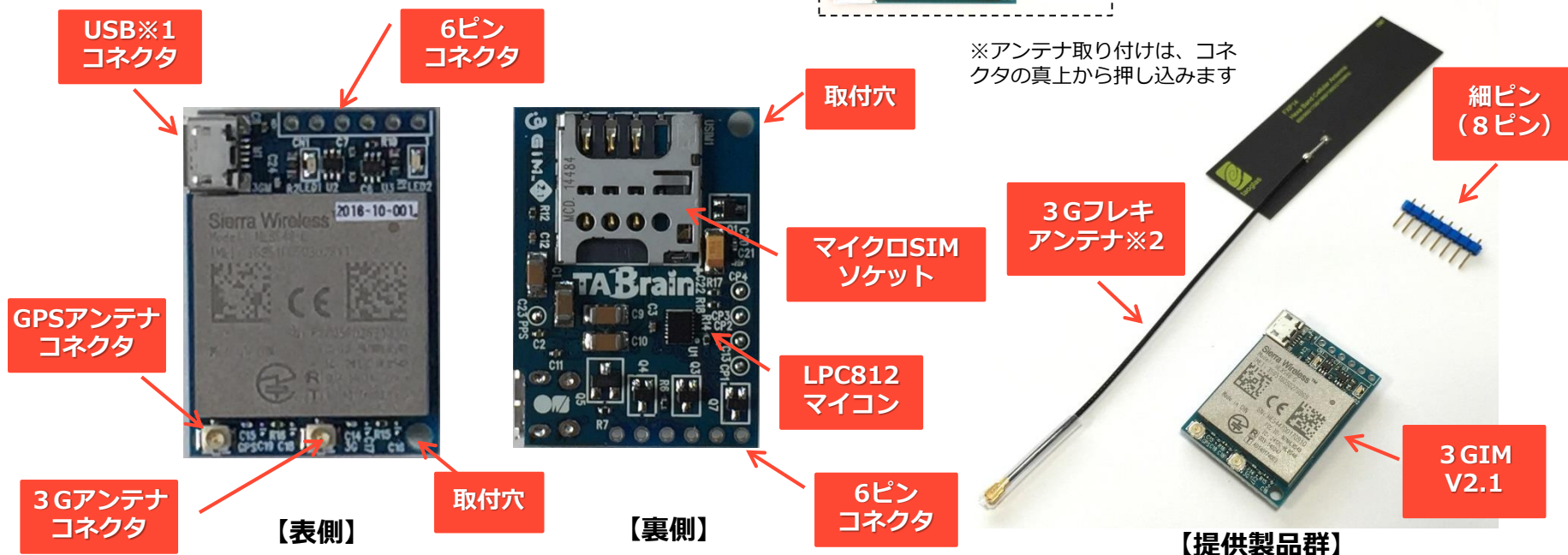
第1章 3 GIM V2.1 の概要

1. はじめに

- ▶ 3 GIMは、M2MやIoTシステム開発での試作やプロトタイプから、量産化に向けた3 G通信モジュールとなります。
- ▶ 今後の新しいモノづくりにおいて、広域ワイヤレス（3 G通信）を誰もが、高度な技術を、簡単に、短時間で、利用し、応用できるようにしたものが3 GIMとなります。
- ▶ 3GIM（3 G IoT Module）は、様々なマイコンを使って、簡単にインターネット接続することができるSDカードサイズの超小型3G通信モジュールです。
- ▶ 3GIMは、マイコンボード(mbed,GR-Sakura,PIC,Intel Edison等)やコンピュータボード（RaspberryPiなど）から **UART経由**または**USB経由**で簡単に利用することができます。
- ▶ Arduino関連互換機などでお使いの場合には、別途**3 Gシールド**関連ドキュメント類も参考にしながら開発を進められることをお薦め致します。
- ▶ 本製品の技術サポートおよび今後のマニュアル更新につきましては、以下のWikiページをご覧ください。
<http://a3gs.wiki.fc2.com/>
- ▶ 3 GIM V2.1のファームウェアは、V2.0と同じですので、特にソフトウェアの変更は不要です。また、従来の3 GIM V1.1とほぼ互換品としてもご利用頂けます。

2. 3GIM V2.1の外観

■ 3GIMの外観写真およびその説明



- ▶ 表側には、**通信モジュール (HL8548-G)**、**マイクロUSBコネクタ**、電源LED、シリアル番号シール等が配置されています。またシール上の脇には、**6ピンコネクタ**があります。
- ▶ 裏側には、**マイクロSIM(ミニSIM)ソケット**および**マイコン (LPC812)**があります。
(SIMカード挿し込む時、裏表・上下を間違わないようにしてください)
- ▶ 3GおよびGPS用のアンテナは、専用のアンテナをご利用ください。
(アンテナのコネクタ部分は、特に扱いに注意が必要となります)

※1：マイクロUSBコネクタは力を入れ過ぎると剥がれますので、ご注意ください。剥がれた場合には有償交換いたします。

※2：3Gフレキアンテナは、日本の技適を取得したものです。

3. 3 GIM 2.1 の機能概要

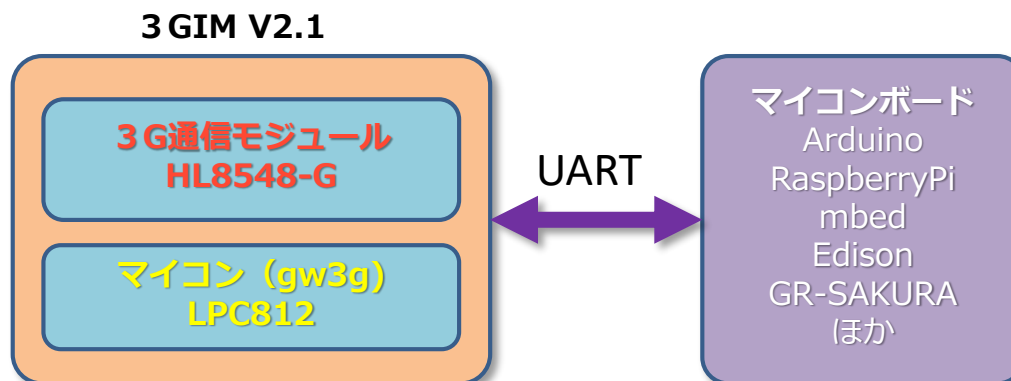
▶ **3GIM V2.1は、下記の機能が提供されます。** (gw3gアプリの機能)

1. 3Gを介したインターネット接続 (TCP/IPおよびHTTP、HTTPS※1)
2. 3Gネットワークからの時刻取得
3. GNSS (GPS/GLONASS) を使った位置情報の取得 (Assisted GPS機能あり)
4. SMSの送受信 (※2)
5. プレーン (機内) モード機能 (省エネモード利用)
6. その他機能 (電波強度の取得、日時情報取得、ボーレートの変更※3、APN※3の切り替え等)
7. HL8548-GのATコマンドパススルー機能

※1 : 対応できるSSL証明書には一部制約があります。すべてのサーバに対してHTTPS通信ができることを保証する訳ではありません。

※2 : ご利用頂けるSIMカードは、SMS対応のものとなります。

※3 : ボーレートやAPN情報は、一度設定すると不揮発性メモリに保存されます。



4. 3GIM V2.1 の仕様概要

| 項目 | 仕様 | 補足 |
|---------------|--|--|
| 外形寸法 | 幅25mm × 奥行35mm × 高さ7mm | 取付穴は ϕ 2.6(1ヶ所)、重さ7.5g |
| 電源電圧 | 電源コネクタ部 3.3~4.2V (注意 : 5Vは使用不可) | 安定したDC電源または3.7Vリチウムポリマ電池 ※1 を推奨 |
| 消費電流 | 10~900mA (最大) (HL8548-Gのエアプレーンモード時はMax1.5mA) | 利用状況や電波状態に依存 (エアプレーンモード時10mA 程度) |
| 通信規格・対応周波数 | HL8548 (シエラワイヤレス製) 対応 | 800/850/900/1900/2100MHz |
| マイコンとのインタフェース | UARTを介したコマンド・レスポンス方式 またはUSBモデム | 仕様書は別途公開予定(ただし、USBモデムは規格のみ公開 ※2) |
| 使用アンテナ | フレキアンテナやポールアンテナ多種 | 取付用コネクタおよび基板を標準で添付 |
| ロジック電圧 | 任意のロジック電圧で利用可能(3GIMにIO電圧を供給) | |
| UART | 9600~115200bps/8データビット/パリティなし/1ストップビット※3 | 初期設定 9600 bps |

※1 : USBコネクタ経由で充電は可能ですが、製品機能としてはサポート外・保証外とさせていただきます。

※2 : USBモデムとしてのご利用に関しては、技術サポートは行っていません。

※3 : ソフトシリアル通信では、9600、19200、38400bps (推奨) まで、ハードシリアル通信では、さらに57600、115200bpsまで可能

5. 3 GIM V2.1 のピンコネクタ配置

| ピン番号 | 名称 | 機能など |
|------|---------|--|
| #1 | PWR_ON | 電源のON/OFF制御(開放または0:LOWでON、1:HIGHでOFF) |
| #2 | RX | UARTインタフェース(RX) : 相手方のTxに接続 |
| #3 | TX | UARTインタフェース(TX) : 相手方のRxに接続 |
| #4 | IOREF | ロジック電圧(任意、通常は 1.8V~5V間 で利用) |
| #5 | VCC (+) | 電源電圧(3.3~4.2V) ※瞬間的に3.2V以下にならないこと |
| #6 | GND (-) | グラウンド |

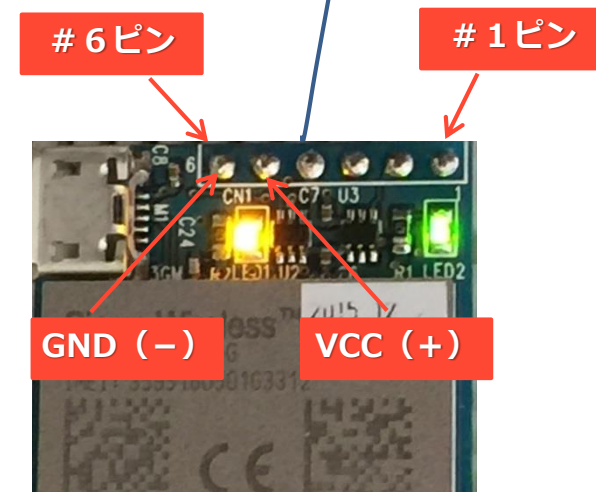
【補足説明】

- #1ピンは「1」とシルク印刷されている側のピンです。
- #1ピンの「HIGH」は、ロジック電圧(IOREF)とします。
- 「VCC(+)」ピンから電源を供給する場合は、PWR_ONの状態によらず、常にONとなります

注意：VCCを間違わないように
推奨：3.7Vリチウムイオン電池推奨

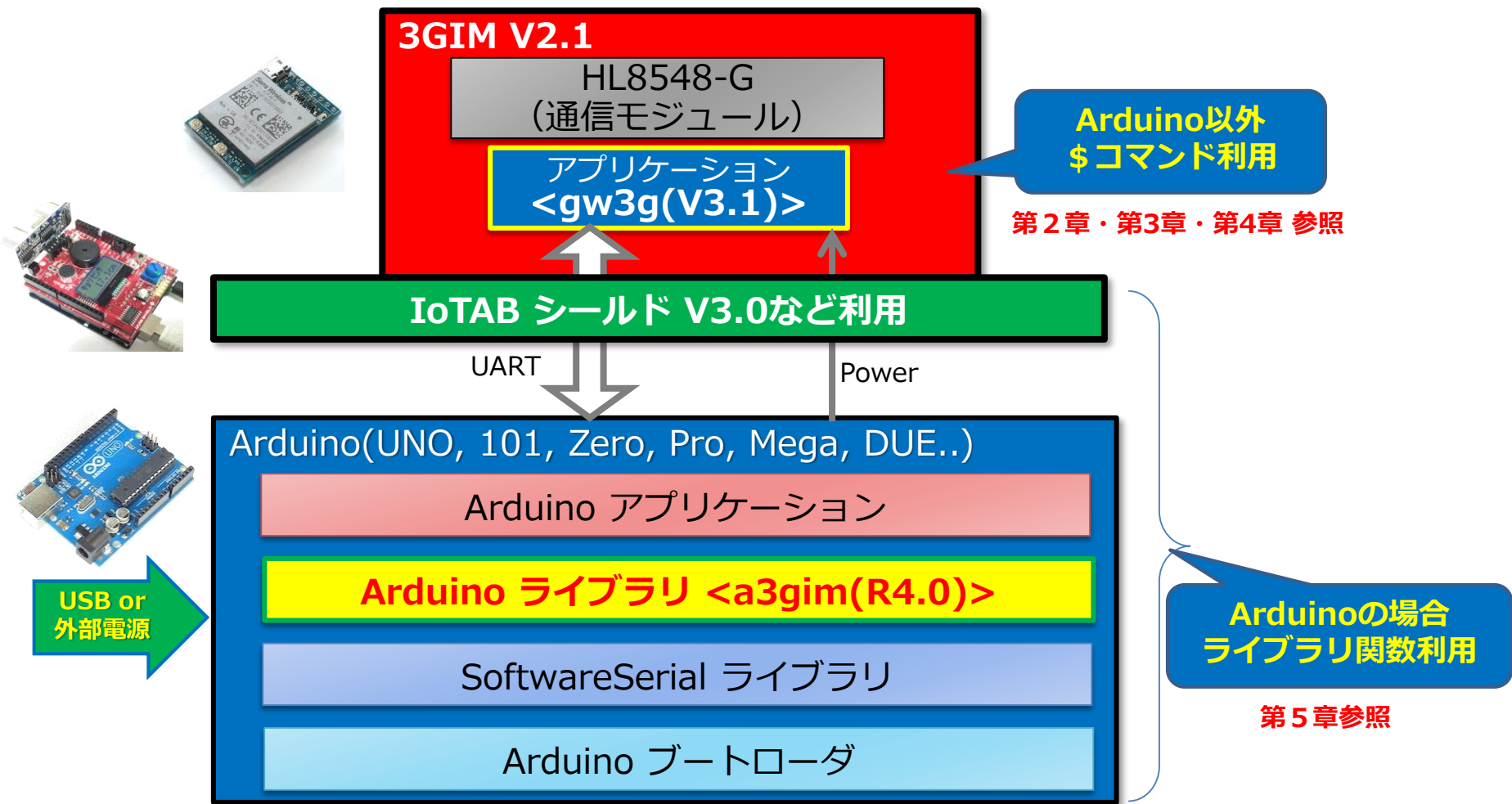
※ #5 (VCC) から電源供給必要
電源電圧 (3.3~4.2V : 200mA以上)
< #1で電源ON/OFF制御 >

【注意】Arduinoの外部出力電源の3.3Vでテストしましたが、一部のArduinoでは稼働しないことが分かりました。
供給電流が低いために通信できないものと思われます。
(補足：場合によってはPC側の電源供給不足問題もあります)



6. 3GIMを利用するソフトウェアについて

3G(W-CDMA)
Band 1/6/19



7. ご利用上の留意点

1. NTTドコモ様のFOMA回線を利用します。そのため、NTTドコモ様あるいはそのネットワークを利用するMVNO様が提供するマイクロSIMが利用できます（ただし、これらの条件を満たす全てのSIMカードでの利用を保証する訳ではありません。ご利用においては、SIMカードのAPN情報、ユーザ情報、パスワードが必要となります。）

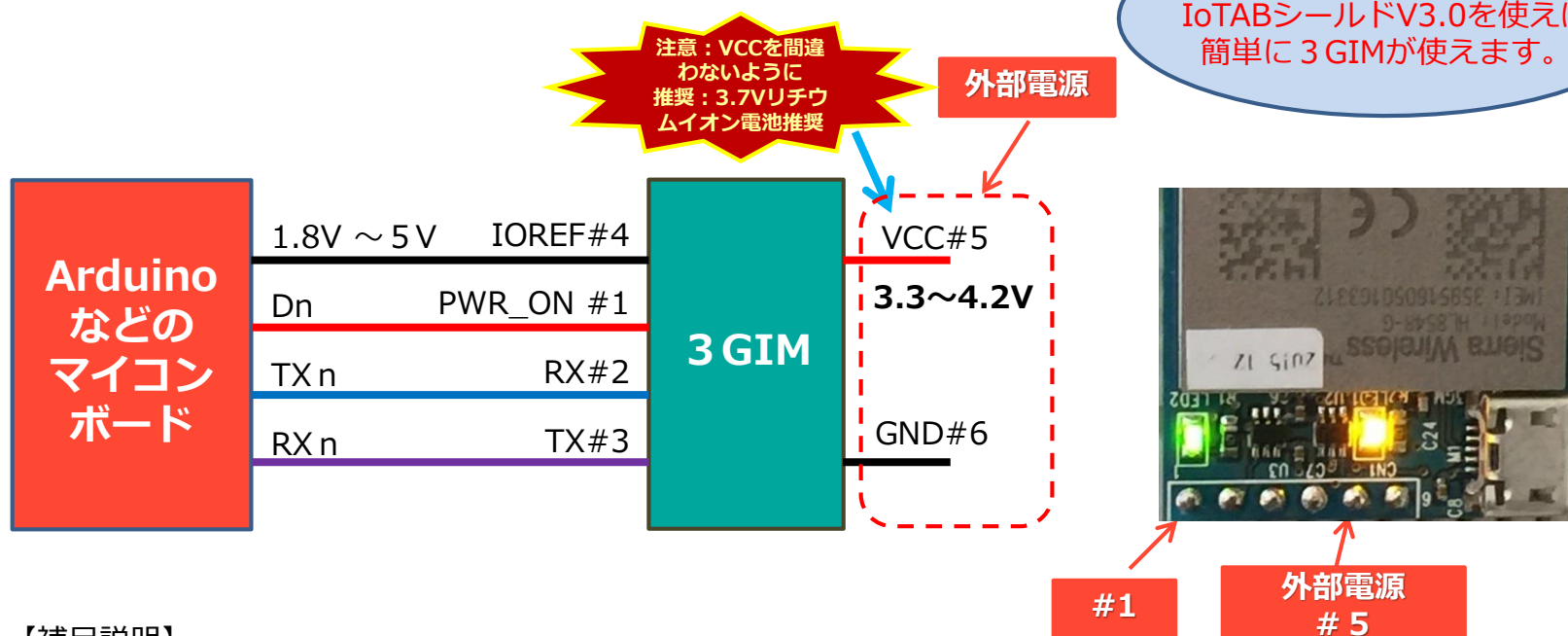
<http://a3gs.wiki.fc2.com/>

※提供されている以外のNTTドコモ製のSIMカードご利用されたい場合には、別途info@tabrain.jpへご連絡ください。

2. 日本国内のみでの利用をお願いします。海外では、各国の法律により現状ではご利用いただけません。詳細はタブレットまでご相談ください。
3. USBモデムとして利用する場合でも、電源供給（3.3~4.2V）は必要です。
4. 回路図は、オープンソースとして公開します。将来的にはプリインストールしているファームウェアも公開する予定でいます。
5. GPS取得は、電波障害が少ない屋外などで行ってください。また初回GPS取得時では、特にPCなどの電波障害を避けて、ご利用ください。（USBケーブルを長いものを使ってPC本体から離してご利用頂くなど）
（初回のGPS取得には、数分時間が掛かります。Assisted GPS機能を使うと野外で30秒前後で取得できます）

【参考】Arduinoで動かす配線・接続

▶ Arduino（マイコンボード）などとの接続方法例



【補足説明】

- デジタル出力Dn（#1接続）をLOWにすることで、3GIMの電源をオンにします。なお、3GIMの電源供給後の立ち上げ時間は約15秒となります。立ち上げ時に、緑LEDが点滅します。またDn（#1接続）を解放していると、常に通信モジュールに電源が供給された状態となります。
※一度電源を入れると、初期立ち上げ以降、コマンド操作での待機時間は不要となります。
- IOREFピンには、1.8V~5Vの範囲ですので、Arduinoのロジック電圧(3.3V or 5V)が接続できます。
(Arduinoの3.3Vから直接電源#5に取る事で稼働できますが、一部電流不足で動かない場合もあります)
- UARTはクロスで接続します（TX/RXを交差させて接続します）
<Genuino101では、ハードウェアシリアル通信のSerial1での通信を推奨します。高速な通信が可能です>

もくじ

1. UART送受信インタフェースの概要
2. 3GIM V2.1 コマンド一覧
3. 3GIMのインタフェース形式 (共通事項)
4. 3GIMのインタフェース形式 (補足事項)
5. \$コマンドの送信・受信の処理方法

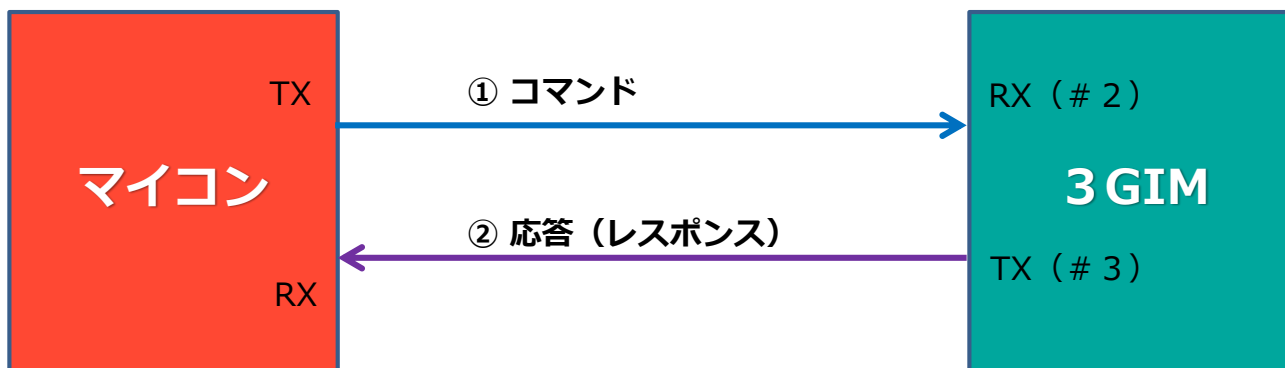


第2章 \$コマンドインタフェース

1. UART送受信インタフェースの概要

■ UARTによる送信（\$ コマンド）と受信（応答：レスポンス）との関係

- ▶ 外部(マイコン側)と3 GIMとの通信は、UARTを通じて行います。
- ▶ マイコン側から**\$ コマンド**を送信し、3 GIM側で受信します。
- ▶ つぎに3 GIM側から**応答（レスポンス）**を送信し、マイコン側で受信して、コマンド制御を終了します。
- ▶ つまりUART送受信の一連の処理は、マイコン側から3 GIM側へのコマンド送信と、3 GIM側からマイコン側への応答（レスポンス）送信で、1つの**シーケンス**として完結します。
 - ▶ コマンドおよび応答（レスポンス）は、改行コード('¥n')で終端します。
(Arduino IDEのシリアルモニタ画面で直接やりとりされる場合には、改行選択メニューで「CRおよびLF」を選択してください)



Arduino
RaspberryPi
mbed
Intel Edison など

通信ボーレートは、
9600bps、19200pbs、38400bps、57600bps、115200pbsから選択できます。
ただし、115200pbsは、ハードウェアシリアル通信時のみ正常に通信可能です。
ソフトウェアシリアル通信は、57600bpsでは文字化けが起きる場合もあり、
高速利用では38400bpsを推奨します。

2. 3 GIM V2.1 コマンド一覧

3GIM V2.1 のUART経由で利用できるコマンドを下表に示します

| No | 分類 | 機能 | コマンド | 頁 | 機能概要 | 補足 (V1.0との比較) |
|----|---------|---------------|------|----------------------|---------------------------|-------------------|
| 1 | System | Version | \$YV | P.22 | gw3gアプリのバージョン情報の取得 | |
| 2 | | RSSI | \$YR | P.24 | 電波受信強度(RSSI)の取得 | |
| 3 | | Service | \$YS | P.26 | 利用可能サービスの取得 | |
| 4 | | IMEI | \$YI | P.29 | IMEIの取得 | |
| 5 | | LED | \$YL | P.30 | LED (RUN) の状態の取得、設定 | |
| 6 | | Baudrate | \$YB | P.32 | UARTの通信速度の変更 | 【拡張】リセット後に有効 |
| 7 | | Reset | \$YE | P.34 | リセット (初期化) | |
| 8 | | Time | \$YT | P.36 | 日時の取得 | |
| 9 | | Airplane mode | \$YP | P.38 | エアプレーン (機内) モードの切り替え | |
| 10 | | ATcommand | \$YA | P.40 | ATコマンドパススルーモード切換え | 【新規】 |
| 11 | SMS | Send | \$SS | P.43 | SMSの送信 | |
| 12 | | Receive | \$SR | P.45 | SMSの受信 | 【拡張】 |
| 13 | | Check | \$SC | P.47 | SMS着信の有無チェック | 【拡張】 |
| 14 | GPS | get Location | \$LG | P.50 | 位置情報の取得、ロケーションサービスの停止 | 【拡張】GPS/GLONASS利用 |
| 15 | Web | Get | \$WG | P.55 | GETリクエストの送付、レスポンスの取得 | ヘッダ指定可(R2.0から) |
| 16 | | Post | \$WP | P.57 | POSTリクエストの送付、レスポンスの取得 | |
| 17 | TCP/IP | Read | \$TR | P.60 | TCP/IPコネクションからのデータからの読み出し | バイナリデータも取扱可 |
| 18 | | Write | \$TW | P.61 | TCP/IPコネクションへのデータの書き込み | 同上 |
| 19 | | Connect | \$TC | P.62 | TCP/IPコネクションの接続 | |
| 20 | | Disconnect | \$TD | P.63 | TCP/IPコネクションの切断 | |
| 21 | | Status | \$TS | P.64 | TCP/IPコネクションの状態の取得、設定 | 【変更】 |
| 22 | | Get sockname | \$TN | P.65 | ソケットのIPアドレスとポート番号を取得 | 接続時のみ有効 |
| 23 | | Write2 | \$TT | P.66 | 現在のコネクションヘダイレクトにデータ書出し | 【新規】 |
| 24 | Profile | Set | \$PS | P.71 | デフォルトプロファイル番号の設定 | 【拡張】 |

※ 3 GIMからの応答 (レスポンス) は、各コマンドの機能紹介にて説明していきます。

\$コマンドのエラー一覧は、[こちら](#)からダウンロードできます。

3. 3 GIMのインタフェース形式（共通事項）

■ コマンドの指定表示形式

\$XX 引数1 引数2 …¥n ※ここでの「XX」はコマンド名です。

引数は1つ以上の半角スペースで区切る。引数には制御コードは含まないでください。

（制御文字を含む引数の指定では、\$文字エスケープシーケンスを使用してダブルクォートで囲む）

¥n は 改行コードとなります。

注意： []（カギ括弧）表記は、オプション（省略可）のもので、実際には記述は不要です。

■ 応答（レスポンス）結果表示形式

\$XX=OK 【結果】 ¥n ※ここでの「XX」はコマンド名です。

【結果】が複数行になる場合は結果部分全体を"で囲みます。

\$XX=NG エラーコード 【付加情報】 ¥n

エラーコードは別途定義する1~3桁の数字となります。

※ 【結果】と【付加情報】はオプションです。

■ コマンドパラメータの特殊文字の表現形式

'\$'文字に引き続く文字を使って、特殊な文字（コード）を表現します。具体的には下記の通りとなります。

| |
|--------------------------|
| \$t : TAB(0x09) |
| \$r : CR(0x0d) |
| \$n : NL(0x0a) |
| \$" : "そのもの" |
| \$ \$: \$そのもの |
| \$xhh または \$Xhh : 16進数hh |

例えば、下記のように使用する：

HTTPヘッダの例 "Content-Type: text/csv\$r\$rn"

【補足1】 3 GIMに電源供給して約15秒ほど経たないとファームウェアが立ち上がりません。立ち上げ時の最初には、以下の応答（レスポンス）が返信されます。これらは読み飛ば（無視）して処理してください。

Welcome to 3GIM(v2)

【補足2】 コマンドが間違った場合の応答（レスポンス）は、以下のようになります。

\$=NG Unknown

4. 3 GIMのインタフェース形式（補足事項①）

■ 初期起動時について

3 GIM の 1 番ピン（#1）を電源OFFの状態か、一度電源ONにした後OFFにすることで、ファームウェアが起動しはじめ、約 13 秒後に 3 GIM 上の **緑LEDが点滅** します。その後、シリアル通信（UART）の 3 番ピン（# 3）から以下のメッセージが送信されます。この状態で 3 GIM と接続できたことになります。

```
Welcome to 3GIM(v2.1)
```

※下線部は、3GIMのファームウェアのバージョンを示す

■ コマンドの応答（レスポンス）の出力について

「\$」で始まるコマンド群では、ほとんどのものが 1 行で応答が返ってきます。しかしながらHTTP/GETやHTTP/POSTそれにTCP/IP機能群では、ヘッダー部とボディ部が返ってきますので、そのための処理がプログラミングで必要となります。

```
$WG http://tabrain.jp/  
$WG=OK 1023
```

※応答（レスポンス）結果だけを知りたい場合は、途中ステータス行を無視するプログラムが必要です。

つまり、結果表示「\$」で始まる行以外は読み捨てるなどが必要です。

※特に応答までの時間が掛る「\$LG」による初回のGPS取得の場合には、何度かコマンドを実行してみる必要もあります。

4. 3 GIMのインタフェース形式（補足事項②）

■ 3 GIMの接続サンプル・スケッチ（起動プログラム）について

3GIMに電源を供給した約13秒後、ボード上の**緑LEDが点滅**し、その後の応答（レスポンス）メッセージが3 GIMから返ってくることを前頁で紹介しました。

3 GIMの起動プログラムについてご紹介します。以下の関数がtrueで返ってくる場合は、正常に接続できたこととなります。falseの場合には、異常があったことで、主に通信速度の設定が間違ったことによります。

```
boolean _3G_SETUP() {  
  String str;  
  uint32_t tim = millis();  
  do {  
    while (!Serial3g.available()) {  
      if (millis() - tim > 15000) return false;  
    }  
    str = Serial3g.readStringUntil('\n');  
  } while (str.indexOf("3GIM") < 0);  
  return true;  
}
```

補足・注意：

緑LEDは点滅するが、応答が無い場合
もしくは、応答した文字が化けている場合
→ シリアル通信速度が、間違っていることによります。

工場出荷時のシリアル通信速度は、すべて9600bpsとなっています。
もし、変更された場合には、メモしておく必要があります。

ここでは、Arduino IDEを使った参考事例となります。

■ 応答（レスポンス）がない場合についての処理

正常な実行中に、応答（レスポンス）がない場合は、異常時（例えば、ハングアップした等）となります。通常は起こりませんが、ハングアップした時などの異常時への対応として、電源のOFFやタイムアウト処理などが必要となります。

■ エラーコードについて

\$ コマンドのエラーコード一覧は、[こちら](#)からダウンロードできます。

5. \$コマンドの送信・受信の処理方法

\$コマンドを発行（シリアル通信から送信）した場合、その処理受付は全てLPC812側で行い、LPC812側のファームウェアから、HL8538-Gへ「ATコマンド」を使って処理が移ります。

この場合（正しく通信HL8548-Gが通信できている場合）

- 1) 一つの\$コマンドに対して、必ず応答があります
- 2) 応答は、1行もしくは複数行（文字列）となります
- 3) 応答の1行目は、コマンド実行の真偽を返します
- 4) \$コマンドの連続発行は、応答を待つことなくできます

以上のことを理解し、\$コマンドを連続して発行した場合、どの順で応答が返ってくるかを認識したプログラミングが必要となります。

一般的には、\$コマンドを発行した場合には、必ず応答を待って、次の処理に移ることが分かり易いプログラミングとなります。

しかし、応答が待てない場合には、連続し発行した\$コマンドを把握しておく必要があります。

右の図の場合、②の応答が複数行ある場合には、③の応答があった直前の応答が最終応答行となります。

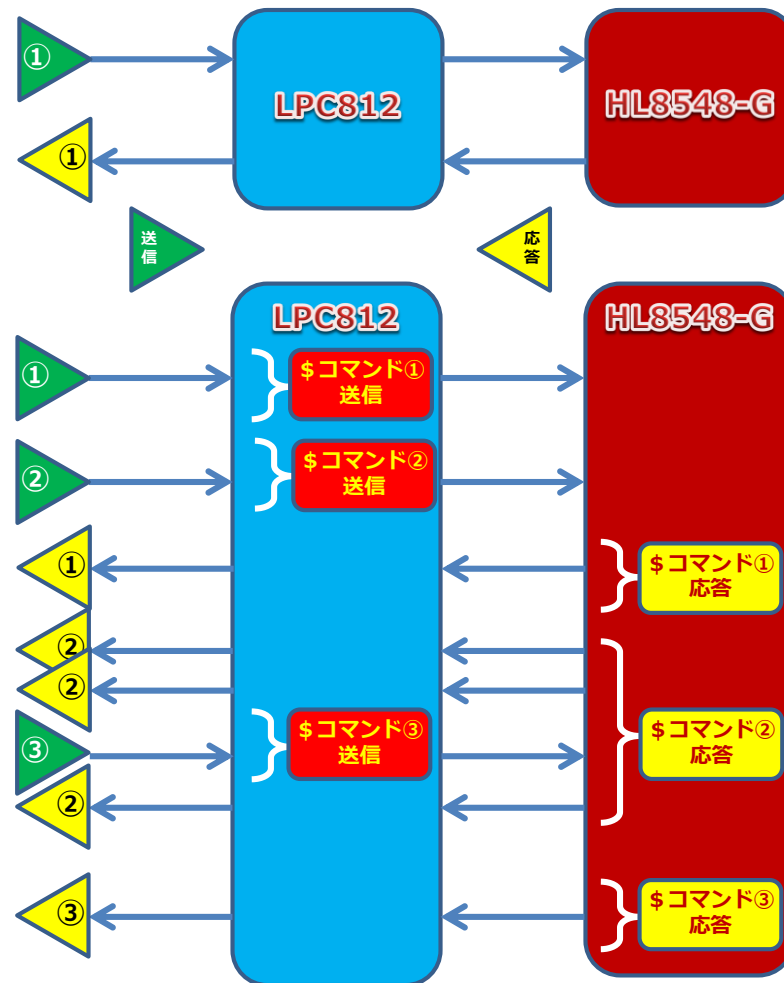
以下のサンプルスケッチは、送信後、応答を待つ2つのケースです。

■ 応答値を使う場合

```
Serial3g.println("$WG" + str);
do { str = Serial3g.readStringUntil('\n'); }
while (str.indexOf("$WG") < 0);
```

■ 応答値を使わない場合

```
Serial3g.println("$YL" + String((int)sw));
while (Serial3g.readStringUntil('\n').indexOf("$YL") < 0);
```



\$コマンド送信と応答の関係（事例）

もくじ

1. システム関連
2. SMS（ショートメッセージ）関連
3. 位置情報取得（GPS）関連
4. Web関連
5. TCP/IP関連
6. プロファイル関連



第3章 3 GIMコマンド・応答（レスポンス）



1. システム関連

1. SYSTEM VERSION ①

■ 3 GIM V2.1（通信）モジュールに設定されたファームウェアのバージョン取得

通信モジュールに設定されているファームウェア（gw3g）のバージョンを取得する

| 項目 | 値など | 説明 | 補足 |
|--------|----------------|--------------------------------------|----|
| 機能分類 | System | | |
| 機能名 | VERSION | ファームウェア（gw3g）のバージョンを取得する | |
| コマンド形式 | | \$YV¥n | |
| 引数 | - | | |
| 応答値 | 【正常時】 | \$YV=OK version¥n | |
| | version | "9.9"形式のバージョン（整数桁がメジャ番号、小数以下がマイナ番号）※ | |
| | 【エラー時】 | \$YV=NG errno ¥n | |
| | errno | 141：コマンドの形式に誤りがある | |
| 前提条件 | | | |
| 補足事項 | | | |

※バージョンは、2016年4月時点の出荷時点「3.1」となる。

1. SYSTEM VERSION ②

■ 事例：バージョン取得サンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup() {
  Serial.begin(9600);
  Serial3g.begin(9600);
  Serial.println("begin System version");
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH);delay(5);
  digitalWrite(7,LOW);
  delay(14000);
  while(Serial3g.available()) Serial3g.read();
  Serial3g.println("$YV");
  while(!Serial3g.available());
  Serial.println(Serial3g.readStringUntil('\n'));
  Serial.println("end");
  pinMode(7,OUTPUT);digitalWrite(7,HIGH);
}

void loop() {}
```

通信速度設定

【補足】ここでの呼出し関数群について

- SoftwareSerial は、ソフトシリアル通信利用宣言
- Serial3g.begin(通信速度)は、ボーレート宣言
- delay(num)は、numミリ秒の待機時間
- Serial3g.listen()は、受信状態占有関数
- Serial3g.available()は、受信バイト数を返す
- Serial3g.readStringUntil('\n')は、リターン値までの文字列読み関数

■ 実行モニタ画面（サンプル）

```
begin System Version
$YV=OK 3.2
end
```

応答受信

返信が無い場合は、以下のようなことが考えられる

- 1) 通信モジュールに電源（橙色LED点灯）が入っていない
- 2) 通信ボーレートが間違っている
- 3) 配線（特にUARTのTxとRxの接続）が間違っている
- 4) 電源電力が不足している

2. SYSTEM RSSI ①

■ 電波受信強度（RSSI）の取得

| 項目 | 値など | 説明 | 補足 |
|--------|-----------------|---|--------------|
| 機能分類 | System | | |
| 機能名 | RSSI | 現在のRSSI値を取得する | |
| コマンド形式 | | \$YR¥n | |
| 引数 | - | | |
| 応答値 | 【正常時】 rssi | \$YR=OK rssi¥n 電波強度（-51~-113）[dBm] | rssiは常にマイナス値 |
| | 【エラー時】 errno | \$YR=NG errno¥n 102：コマンド形式に誤りがある 101：電波強度が取得できない | |
| | | | |
| 前提条件 | ① | あらかじめ3G用のアンテナが正しく装着されていること | |
| 補足事項 | | | |

RSSIとは、無線通信機器が受信する信号の強度を測定するための回路または信号のこと。Received Signal Strength Indication, Received Signal Strength Indicator 別名：受信信号強度のこと。

RSSI値は常にマイナスで、目安は下記の通りである：
 -113の場合には、アンテナが接続されていないとき
 -112~-90 の場合は、電波受信状態が悪い状況
 -89~-51 の場合は、電波受信状態が良い状況

2. SYSTEM RSSI ②

■ 事例：電波受信強度（RSSI）を取得サンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup() {
  Serial.begin(9600);
  Serial3g.begin(9600);
  Serial.println("begin System version");
  delay(14000);
  while (Serial3g.available()) Serial3g.read();
  Serial3g.println("$YR");
  while (!Serial3g.available());
  Serial.println(Serial3g.readStringUntil('\n'));
  Serial.println("end");
}

void loop() {}
```

■ 実行モニタ画面（アンテナ正常接続）

```
begin SYSTEM RSSI
$YR=OK -86
end
```

応答受信

■ 実行モニタ画面（アンテナ未接続）

```
begin SYSTEM RSSI
$YR=OK -113
end
```

応答受信

RSSIの感度が悪い場合

- 1) 通信モジュールとアンテナのコネクタが正しく接続されていない
- 2) アンテナとケーブル・コネクタのネジ部が緩んでいる
- 3) 電波状態が悪い屋内の壁・天井・床などで閉ざされたところにある
- 4) SIMカードが正しく挿入されていないか、APN情報が正しく設定されていない

3. SYSTEM SERVICE ①

■ SIMカードによる通信サービス状況を取得

| 項目 | 値など | 説明 | 補足 |
|--------|---------|--|------------------------|
| 機能分類 | System | | |
| 機能名 | SERVICE | 現在利用できる通信サービスを取得する | |
| コマンド形式 | | \$YS¥n | |
| 引数 | - | | |
| 応答値 | 【正常時】 | \$YS=OK serice¥n | |
| | service | 0 : サービス利用不可 1 : パケット通信(PS)のみ利用可 | |
| | 【エラー時】 | \$YS=NG errono¥n | SIMカードやアンテナが無い時 |
| | error | 131 : コマンドの形式に誤りがある 132 : 内部エラー (サービス種別を取得できない) | |
| 前提条件 | ① | あらかじめSIMカードが装着されていること | SIMカードがないと常に結果としてNGが返る |
| 補足事項 | ① | 3GIM(Ver2)では音声通信の利用可否を取得できないため、音声通信SIMとしての値は返却しない。 | |

3 GIM V2.1で利用できるSIMカードは、NTTドコモおよびドコモFOMA回線を使ったMVNO提供のSIMに限ります。

3. SYSTEM SERVICE ②

■ 事例：SIMカードによる通信サービス状況を取得サンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup() {
  Serial.begin(9600);
  Serial.println("begin System service");
  Serial3g.begin(9600);
  delay(14000);
  while(Serial3g.available()) Serial3g.read();
  Serial3g.println("$YS");
  while(!Serial3g.available());
  Serial.println(Serial3g.readStringUntil('\n'));
  Serial.println("end");
}

void loop() {}
```

通信速度設定

コマンド送信

■ 実行モニタ画面（パケット通信のみの利用の場合）

```
begin SYSTEM Service
$YS=OK 1
end
```

応答受信

4. SYSTEM IMEI ①

■通信モジュールのID（固有）番号（IMEI）を取得

| 項目 | 値など | 説明 | 補足 |
|--------|-----------------|--|----|
| 機能分類 | System | | |
| 機能名 | IMEI | IMEIを取得する | |
| コマンド形式 | | \$YI¥n | |
| 引数 | - | | |
| 応答値 | 【正常時】 imei | \$YI=OK imei¥n 15桁の数字 | |
| | 【エラー時】 errno | \$YI=NG errno¥n 143：IMEIを取得できない 146：コマンドの形式に誤りがある | |
| | | | |
| 前提条件 | | | |
| 補足事項 | | | |

IMEIは「国際移動体装置識別番号（端末識別番号）」を意味する英語"International Mobile Equipment Identifier"の略称



4. SYSTEM IMEI ②

■ 事例：通信モジュールのID（固有）番号（IMEI）を取得サンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup() {
  Serial.begin(9600);
  Serial.println("begin System IMEI");
  Serial3g.begin(9600);
  delay(14000);
  while (Serial3g.available()) Serial3g.read();
  Serial3g.println("$YI");
  while (!Serial3g.available());
  Serial.println(Serial3g.readStringUntil('\n'));
  Serial.println("end");
}

void loop() {}
```

通信速度設定

コマンド送信

■ 実行モニタ画面（正常時）

```
begin SYSTEM IMEI
$YI=OK:359516050164625
end
```

応答値

※このID番号の数値はサンプルです

■ 実行モニタ画面（エラー時）

```
begin SYSTEM IMEI
$=NG 10
end
```

※コマンドが正しく読み込めなかった場合

IMEI番号



5. SYSTEM LED ①

■ 3 GIM上の緑LEDの点滅設定および状態取得

| 項目 | 値など | 説明 | 補足 |
|--------|------------------------------|---|---------------|
| 機能分類 | System | | |
| 機能名 | LED | 緑LED (RUN) ピンの状態取得・設定を行う | |
| コマンド形式 | 取得 設定 | \$YL¥n \$YL status¥n | 状態取得 |
| 引数 | status | ONにするか(1の時)、OFFにするか(0の時) | 1 : 点灯、0 : 消灯 |
| 応答値 | 【正常時】 | \$YL=OK status¥n | |
| | status | 本コマンド実行後のLED状態 (0:OFF/1:ON) | |
| | 【エラー時】 | \$YL=NG errno¥n | |
| errno | 191 : コマンド形式または引数statusがおかしい | | |
| 前提条件 | | | |
| 補足事項 | ① | 本機能で扱うLEDは、1番ピン脇に配置されている緑色LEDである(基板上に「LED2」と記載) | |



このLEDが点灯

5. SYSTEM LED ②

■ 事例：LED状態取得とLEDの点滅10回繰り返しのサンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup(){
  Serial.begin(9600);
  Serial3g.begin(9600);
  delay(35000);
  Serial.println("begin SYSTEM LED");
  Serial3g.println("$YL");
  while (!Serial3g.available() );
  String rt = Serial3g.readStringUntil('\n');
  Serial.print(rt);
  for (int i=0; i<10; i++) {
    Serial3g.println("$YL 1");
    delay(500);
    Serial3g.println("$YL 0");
    delay(500);
  }
  Serial.print("\r\nend");
}
void loop() {}
```

点滅を10回繰り返す

■ 実行モニタ画面（正常時）

```
begin SYSTEM LED
$YL=OK 0
end
```

6. SYSTEM BAUDRATE ①

■ 3 GIMのUART（通信ポート）の通信速度（ボーレート）取得確認と設定

| 項目 | 値など | 説明 | 補足 |
|--------|----------|--|----------------|
| 機能分類 | System | | |
| 機能名 | BAUDRATE | UARTの通信速度(ボーレート)の取得・設定を行う | |
| コマンド形式 | 取得 設定 | \$YB¥n \$YB baudrate¥n | |
| 引数 | baudrate | 設定するボーレート(9600/19200/38400/57600/115200) | ご購入時は9600bps |
| 応答値 | 【正常時】 | \$YB=OK baudrate¥n | |
| | baudrate | 本コマンド実行後のボーレート | |
| | 【エラー時】 | \$YB=NG errno¥n | |
| | errno | 111：コマンド形式または引数baudrateがおかしい | |
| 前提条件 | ① | 指定するボーレートで正しく動作することを確認しておくこと | ボーレートの目安を参照のこと |
| 補足事項 | ① | 本コマンドで設定した新しいボーレートは直ちに反映される | |
| | ② | 本コマンドで設定したボーレートは電源の再投入やリセットしても維持される。 | |

【注意事項】

- ① 現状のボーレートと、変更後のボーレートは、常に把握した上でこのコマンドを使うこと
(現在のボーレートが分からなくなる/通信できなくなると、一つ一つボーレートを試して探り当てる必要がある)
- ② Arduinoのソフトウェアシリアルを利用する場合は、ハードウェアシリアルを使った場合よりも低いボーレートでしか利用できない
(ソフトウェアシリアルの場合は38400bps以下での利用を推奨)

6. SYSTEM BAUDRATE ②

■ 事例：3 GIMのUARTのボーレート取得確認サンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
#define NOWBAUDRATE 9600
#define NEWBAUDRATE 38400

void setup() {
  Serial.begin(115200);
  pinMode(7,OUTPUT);
  Serial.println("begin System Baudrate");
  Serial3g.begin(NOWBAUDRATE);
  delay(14000);
  while (Serial3g.available()) Serial3g.read();
  Serial3g.println("$YB " + String(NEWBAUDRATE));
  delay(5);
  Serial3g.begin(NEWBAUDRATE);
  String str;
  do{
    while (!Serial3g.available()) ;
    str=Serial3g.readStringUntil('\n');
  } while (str.indexOf("$YB") < 0);
  Serial.println(str);
  Serial.println("end");
}

void loop() {}
```

この通信速度にする

ボーレートを9600bpsから38400bpsに変更するサンプルです。

■ 実行モニタ画面（正常時）

```
begin SYSTEM Baudrate
$YB=OK 38400
end
```

ボーレート設定での注意点

- 1) ボーレート設定を変更すると、改めてシリアル通信速度も変更が必要となります。
- 2) ボーレート変更設定した場合には、その情報は控えておいてください。

ボーレート変更によって起こる問題

- 1) 文字化けが起きることがあります。特にスクリーンモニタ画面に出す場合には、なるべく、通信速度をシリアルモニタ画面の通信速度と合わせるようにお願いいたします。
- 2) 処理の待機時間が関係してきますので、同じソフトウェアでも調整が必要となります。

7. SYSTEM RESET ①

■ 3 GIMをリセットするコマンド

| 項目 | 値など | 説明 | 補足 |
|--------|---------|---|---------------------|
| 機能分類 | System | | |
| 機能名 | RESET | 3 GIMをリセットする | |
| コマンド形式 | ソフトリセット | \$YE¥n | |
| | 指定リセット | \$YE level¥n | |
| 引数 | level | リセットのレベル (0: ソフトリセット、1: ハードリセット) | levelの内容に拘らず常に再起動する |
| 応答値 | 【正常時】 | \$YE=OK level¥n | |
| | level | 引数と同じ | |
| | 【エラー時】 | \$YE=OK errno¥n | |
| | errno | 191: コマンドの形式に誤りがある | |
| 補足事項 | ① | リセット後、復帰までには14秒程度の時間が掛かる。再起動するまで3 GIMは利用できない。 | |
| | ② | 実装上の理由で、ハードリセットはソフトリセットと同じ動作となっている。 | |

ハードウェアリセットは、#1ピンをHIGHにすることで電源OFFにできます。
この場合3 GIM上の橙LEDが消灯します。

7. SYSTEM RESET ②

■ 事例：リセットのサンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup(){
  Serial.begin(38400);
  Serial3g.begin(38400);
  delay(14000);
  Serial.println("SYSTEM Reset");
  Serial3g.println("$YE");
  while (!Serial3g.available() );
  String rt = Serial3g.readStringUntil('\n');
  Serial.print(rt);
  Serial.print("¥r¥nend");
}
void loop() {}
```

■ 実行モニタ画面（正常時）

```
SYSTEM Reset
$YE=OK
end
```

8. SYSTEM TIME ①

■ 通信モジュールの取得した時間取得

| 項目 | 値など | 説明 | 補足 |
|--------|-------------|--|------------------------|
| 機能分類 | System | | |
| 機能名 | System Time | 日時の取得 | |
| コマンド形式 | 時刻取得 | \$YT¥n | |
| | 時刻取得(2) | \$YT 1¥n | 時刻サーバから日時を取得しなおす |
| 引数 | | | |
| 応答値 | 【正常時】 | \$YT=OK datetime¥n | |
| | datetime | 出力例として \$YT=OK 2016/05/14 15:52:23 などのように「年 (4バイト) '/'月 (2バイト) '/'日 (2バイト) '' (スペース1バイト) 時 (2バイト) ':'分 (2バイト) ':'秒 (2バイト)」でリターン値が返ってくる。 | 日時はJST |
| | 【エラー時】 | \$YT=NG errno¥n | |
| | errno | 121:コマンドの形式に誤りがある 122:内部エラー (日時の取得に失敗した) | errnoの後にエラー情報を出力する場合あり |
| 前提条件 | ① | アンテナ接続と正しいSIMカードの設定で正確な時刻が取得できる | |
| 補足事項 | ① | 本コマンドの最初の実行時にインターネット上の時刻サーバから正しい日時を取得して、HL8548-Gのリアルタイムクロック(RTC)に設定して保持する。2回目以降の実行では、RTCから時刻を読み出す。 | |

ファームウェア起動後に「\$YT」を起動すると初回のみ5秒ほど掛って時間を取得する。その後2回目以降は、瞬時に取得できる。

8. SYSTEM TIME ②

■ 事例：日時取得表示のサンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup() {
  Serial.begin(38400);
  Serial3g.begin(38400);
  delay(30000);
  Serial.println("begin System Time");
  Serial3g.println("$YT");
  while (!Serial3g.available());
  String rt = Serial3g.readStringUntil('\n');
  Serial.print(rt);
  Serial.print("\r\nend");
}
void loop() {}
```

■ 実行モニタ画面（正常時）

```
begin System Time
$YT=OK 2016/05/13 15:50:53
end
```

■ 時間取得関数（例）

```
String datetime() {
  String dtime;
  uint32_t tim=millis();
  do {
    Serial3g.println("$YT");
    while (!Serial3g.available());
    dtime = Serial3g.readStringUntil('\n');
    if (millis() - tim > 60000) return "";
  } while (dtime.indexOf("20") < 0);
  return dtime.substring(7);
}
```

※注意：正しいSIMカードやアンテナ接続がされていないと時間取得ができません。

【補足】取得した時間が間違っている場合

- 1) 正しいSIMカードが設定されているかを確認
- 2) アンテナ接続が正しく接続されているかを確認
- 3) アンテナ感度が良い環境かどうかを確認（参照：\$YR）
- 4) 正しい時間取得までに5秒ほど掛る
※2回目以降の取得は瞬時となる

9. AIRPLANE MODE ①

■ エアプレーンモード（機内モード：省エネモード）の取得・設定

| 項目 | 値など | 説明 | 補足 |
|--------|---------------|--|-----------------|
| 機能分類 | System | | |
| 機能名 | Airplane mode | 3 GIMのエアプレーン（機内）モードを切り替える | |
| コマンド形式 | モード取得 | \$YP¥n | |
| | モード切替 | \$YP mode¥n | |
| 引数 | mode | 設定するモード（0: 通常モード、1:エアプレーンモード） | |
| 応答値 | 【正常時】 | \$YP=OK mode¥n | |
| | mode | 設定後のモードを返す 0 : 通常モード 1 : エアプレーン（機内）モード | |
| | 【エラー時】 | \$YP=NG errno¥n | |
| | errno | 181 : コマンド形式またはモードの値がおかしい 182 : 内部エラー（エアプレーンモードの変更ができない） | |
| 前提条件 | ① | gw3gのバージョンがR1.3以降のみで利用できる | |
| 補足事項 | ① | エアプレーンモードでは、 \$YB および \$YP 、 \$YE コマンド以外のコマンドは利用できない。 | |
| | ② | エアプレーンモード時の消費電流は実測値で約10mAである。 | |
| | ③ | エアプレーンモード中でも、SMSの受信は可能である。 | SMSが利用できるSIMの場合 |

補足：本エアプレーンモード時は、**3 GIMの消費電流を数ミリアほどに抑えることができます。**
完全に消費電力をゼロにするには、3 GIMの# 1ピンをHIGHにすることで、電源をOFFにできます。

9. AIRPLANE MODE ②

■ 事例：エアプレーンモード値の取得サンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup(){
  Serial.begin(38400);
  Serial3g.begin(38400);
  delay(30000);
  Serial.println("begin AirPlane Mode");
  Serial3g.print("$YP¥n");
  while(!Serial3g.available());
  String rt = Serial3g.readStringUntil('¥n');
  Serial.print(rt);
  Serial.print("¥r¥nend");
}
void loop() {}
```

■ 実行モニタ画面（正常時）

```
begin AirPlane Mode
$YP=OK 0
end
```

10. AT COMMAND ①

■ ATコマンドモードへの切り替え

| 項目 | 値など | 説明 | 補足 |
|--------|-----------------|---|--|
| 機能分類 | System | | |
| 機能名 | AT Pass Through | ATコマンドパススルーモードに入る | |
| コマンド形式 | モード取得 | \$YA [time]¥n | |
| 引数 | time | 待機時間：単位100ミリ秒；（例：time=300 は 30秒間だけモード切替え） | |
| 応答値 | 【正常時】 | \$YA=OK¥n | |
| 前提条件 | ① | ATコマンドの機能を理解した上で使用のこと。 | |
| 補足事項 | ① | 利用できるATコマンドの詳細は「AT Commands Interface Guide - AirPrime HL6 and HL8 Series」を参照のこと。 | Sierra Wireless社のサイトで公開 |
| | ② | ATコマンドの使用に制限はないため、HL8548-Gの設定を任意に変更することができる。しかし、変更した設定等によっては、gw3gファームウェアの動作に支障を来す場合があるので、十分の留意すること。 | 例えば、ATコマンドでプロファイルの設定等を変えてしまうと、3G通信ができなくなる可能性がある。 |
| | ③ | ATコマンドパススルーモードから抜けるには、以下のいずれかの方法を使用する： ①PWR_ONピンを制御して、3GIMの電源を入れなおす。 ②「 AT+CPOF 」コマンドを実行して3GIMをリセットする。 | ①・②共に初期起動時に戻る。「Welcome to 3GIM(v*)」応答 |

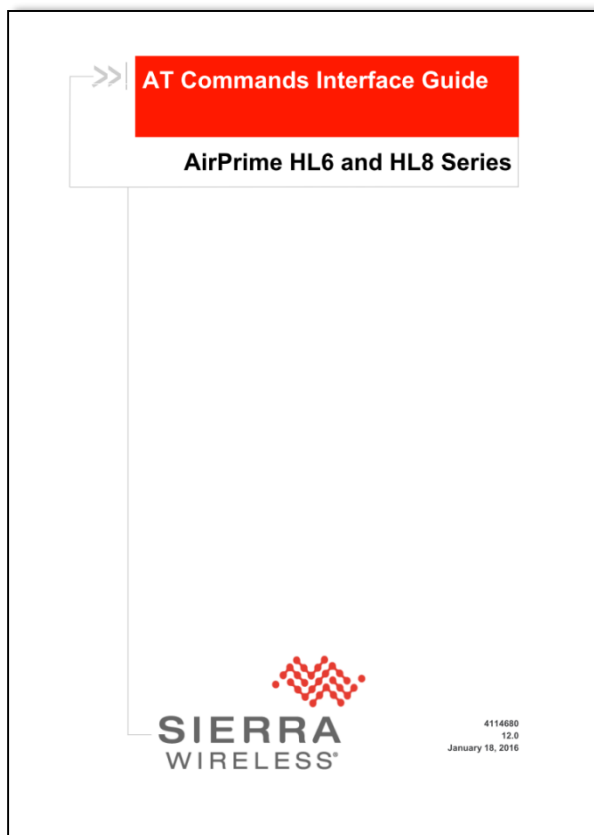
事例：アシストGPSを使う場合、以下のようなATコマンド設定を行います。
 (以下の「\$YA 1」で100ミリ秒間だけATコマンドパススルーモード)

```
Serial3g.println("$YA 1");
delay(10);
Serial3g.println("at+wppp=2,4,¥"username¥",¥"password¥");
```

※usernameとpasswordは、SIMカードのAPN情報のユーザ名とパスワード
 <本マニュアルP.50参照>

10. AT COMMAND ②

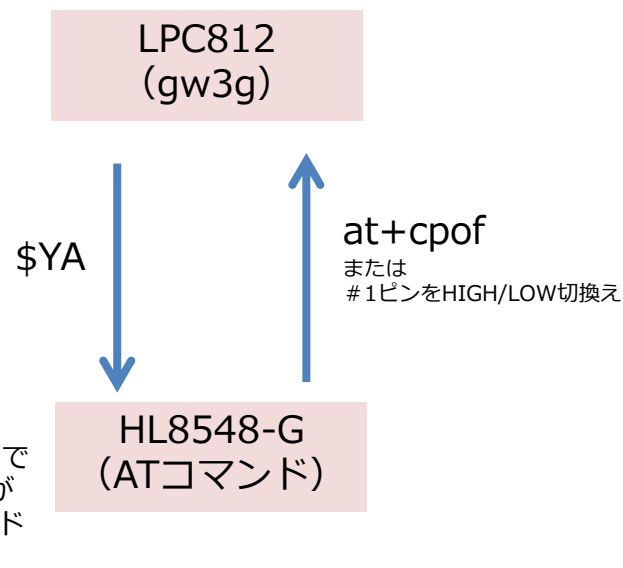
HL8548-GのATコマンドは、シエラワイヤレスサイトから以下の資料をダウンロードできます。
ドキュメント名 (AirPrime_HL6_and_HL8_Series_AT_Commands_Interface_Guide_Rev12_0.pdf)



【注意点】

ATコマンドでHL8548Gの特殊な設定を行った場合には、LPC812から制御不能となることがあり、3 GIMが利用できなくなる場合がございます。

ATコマンドで操作する場合には、十分な知識を得た上で、ご利用頂く事お勧めいたします。





2. SMS（ショートメッセージ）関連

〈注意〉 ショートメッセージは、SIMカードがショートメッセージ対応でないご利用いただけません。



1. SMS SEND ①

■ SMS(ショートメッセージ) の送信

| 項目 | 値など | 説明 | 補足 |
|--------|---------|--|-------------------------------------|
| 機能分類 | SMS | | |
| 機能名 | SEND | SMSを送信する | |
| コマンド形式 | | \$SS msn "message" [encode]¥n | encode=ASCII と同じ |
| 引数 | msn | 送信先の電話番号（ハイフオン無しの数字のみで指定） | |
| | message | 送信するメッセージ（制御文字は使用不可、UNICODEはサポートしない） | 「"」は「¥"」として記述、最大120バイトまで |
| | encode | "ASCII" または "UNICODE"のいずれか | |
| 応答値 | 【正常時】 | \$SS=OK¥n | |
| | 【エラー時】 | \$SS=NG errno ..¥n または \$SS=NG errcode¥n | |
| | errno | 401：引数指定エラー 402：BUSYエラー（既にSMS送信中、送信エラー等） 409：encodeでUNICODEは指定できない | 3GIM(V2)ではASCIIのみ利用可能 |
| 前提条件 | ① | 音声サービス(SMS含む)が利用できる状態であること。 | |
| | ② | 通常・SMSは従量課金制であるので、送信回数には留意が必要である。 | データ通信専用SIMでは利用できないケースが多い点も注意が必要である。 |
| 補足事項 | ① | 文字コードがASCIIの場合でも、SMSとして利用できない文字が存在する。 | |
| | ② | 送信できるメッセージの最大長は120バイト以内であるが、送信先の携帯事業者によっては文字数の制限がより厳しい場合がある。その場合は、メッセージの一部しか届かない場合もある。 | |

1. SMS SEND ②

■ 事例 : SMS(ショートメッセージ) 送信のプログラム

■ Arduinoでのサンプルスケッチ

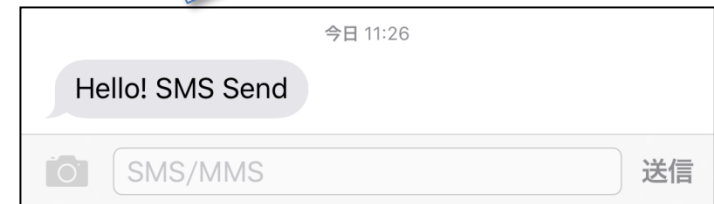
```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup(){
  Serial.begin(38400);
  Serial3g.begin(38400);
  delay(23000);
  Serial.println("begin SMS Send ");
  Serial3g.print("$SS 08012345678 ¥"Hello! SMS Send¥" ASCII¥n");
  String rt="";
  do{
    while (!Serial3g.available());
    rt = Serial3g.readStringUntil('\n');
    Serial.print(rt);
  } while (! rt.startsWith("$SS="));
  Serial.print("¥r¥nend");
}
void loop() {}
```

\$ SS=OKまたは \$ SS=NGでない場合繰り返し

■ 実行モニタ画面 (正常時)

```
begin SMS Send
$SS=OK
end
```

スマホに送られてきたSMS



※ SMS (ショートメッセージ) の送信の応答性は、必ずしもリアルタイム性・即時性があるとは限りません。(タブレットの調査による)
 ※ SMS (ショートメッセージ) は、NTTドコモでは1件あたり3円(税別)となっていますので、大量送信では気を付けるようにしてください。

2. SMS RECEIVE ①

■SMS(ショートメッセージ)の受信

| 項目e | 値など | 説明 | 補足 | |
|--------|-----------------|---|---|--|
| 機能分類 | SMS | | | |
| 機能名 | RECEIVE | 受信したSMSを読み出す | | |
| コマンド形式 | | \$SR¥n \$SR index [option] ¥n | | |
| 引数 | index option | SMSの格納スロット番号 (1~20) 日時出力の有無：1で出力あり、指定なし時は出力なし | 省略時は1 | |
| 応答値 | 【正常時】 msn | \$SR=OK msn "message"¥n 受信したSMSの送信元の電話番号 (ハイフオン無し) | 「"」は「¥」として記述 最大11バイト | |
| | message | 受信したSMSのメッセージ | ASCIIまたはUNICODE、最大100バイト | |
| | 【正常時2】 date | \$SR=OK msn date "message"¥n SMSの受信日時 | option=1指定時 日時の形式は\$YTと同一 | |
| | 【エラー時】 errno | \$SR=NG errno¥n 412：SMSを受信していない 413：内部エラー 401：コマンド形式のまちがい | | |
| | 前提条件 | ① | SIMカードがSMSオプション利用できること | |
| | 補足事項 | ① | 通常、複数のSMSを同時に受信しない場合は、特にindexを指定しない「\$SR」でSMSを読み出すことができる。 | |
| ② | | 受信したSMSは、20個ある格納スロットのいずれかに格納される。「\$SC」コマンドで受信したSMSがある場合は、通常はindex=1のスロットに格納されるが、そこにすでにSMSが格納されている場合は別の空いているスロットに格納される。そのため、index=1~20を順に指定して、各スロットにSMSが格納されているかどうかを調べる処理が必要となる。 | | |

2. SMS RECEIVE ②

■ 事例 : SMS (ショートメッセージ) の受信サンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup(){
  Serial.begin(38400);
  Serial3g.begin(38400);
  delay(23000);
  Serial.println("begin SMS Receive ");
  Serial3g.println("$SR");
  String rt="";
  while (!Serial3g.available() );
  rt = Serial3g.readStringUntil('\n');
  Serial.print(rt);
  Serial.print("\r\n");
}
void loop() {}
```

■ 実行モニタ画面 (正常時)

```
begin SMS Receive
$SR=OK 08012345678 " Hello! SMS Send "
end
```

※ エアプレーンモード時でも、SMS (ショートメッセージ) は受信できます。

3. SMS CHECK ①

■ SMS (ショートメッセージ) の受信確認

| 項目 | 値など | 説明 | 補足 |
|--------|-------|--|---|
| 機能分類 | SMS | | |
| 機能名 | CHECK | SMSを受信しているかどうかをチェックする | |
| コマンド形式 | チェック | \$SC¥n | |
| | SMS削除 | \$SC 1¥n | すでに受信しているSMSすべてを削除する |
| 引数 | - | | |
| 応答値 | 【正常時】 | \$SC=OK exist¥n | |
| | exist | 0 : SMSをまったく受信していない時 1 : SMSを1つ以上受信している時 | |
| 前提条件 | | | |
| 補足事項 | ① | SMS配送遅延や電波状態等によりSMSの再送が発生した場合、\$SCコマンドで受信したSMSがあるにも関わらず、\$SRコマンドで受信したSMSをうまく読み出せないケースが発生する。このような場合は、「\$SC 1」を実行することでおかしくなったSMSの受信状態をクリアすることができる。 | 左記のようなSMSの受信状態を避けるには、SMSは一度に一つずつ送信・受信を行い、3GIM側に複数のSMSを溜めないような利用方法を推奨する。 |

【SMS全般の留意事項】

- 1) 受信したSMSは、\$SRで読み出さない限り、HL8548-G内の不揮発性メモリ（電源を切っても内容が保持されるメモリ）に記録される。そのため、3GIM(V2)を利用するアプリケーションにて、適当なタイミング（例えば初期化時）で溜まっているSMSを適宜読み捨てる処理が必要となる。溜まっているSMSをすべて読み捨てるには、「\$SC 1」を実行する。
- 2) SMSの受信では、3GIMの電波状態や電源状態等により、あるいは通信事業者側の配信遅延等により、受信できなかったSMSを後で受け取る場合があるため即時性が要求されるような利用方法では十分に留意する必要がある。

3. SMS CHECK ②

■ 事例 : SMS (ショートメッセージ) 受信サンプルプログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup() {
  Serial.begin(38400);
  Serial3g.begin(38400);
  delay(23000);
  Serial.println("begin SMS Check ");
  Serial3g.println("$SC");
  while (! Serial3g.available() );
  String rt = Serial3g.readStringUntil('\n');
  Serial.print(rt);
  Serial.print("\r\nend");
}
void loop() {}
```

■ 実行モニタ画面 (正常時)

```
begin SMS Check
$SC=OK 0
end
```

未受信

```
begin SMS Check
$SC=OK 1
end
```

受信

※ SMS受信は、即時性がないため、しばらく何回か受信確認が必要となる場合があります。



3. GPS関連

GPS利用時は、GPS専用アンテナが必要です。
(別売のGPSフレキアンテナをご利用ください)



1. LOCATION GPS ①

■ GPS(全地球測位システム) の取得コマンド (次頁につづく)

| 項目 | 値など | 説明 | 補足 |
|--------|-------------|---|--------------------------------------|
| 機能分類 | GPS | | |
| 機能名 | LOCATE | 測位を行う。 | |
| コマンド形式 | | \$LG method [option [repeatCount]]¥n | |
| 引数 | method | 測位の方法 (下記のいずれかを指定: 現時点 任意文字受付可) MSBASED : GPSで測位、GPSが利用できない時は3Gネットワークを利用 MSASSISTED : 3Gネットワークを利用して測位 (A-GPS) STANDALONE : GPS単体で測位 | 任意文字入力時はMSBASEDとなる (本文列は3 GIM V1.0用) |
| | option | 測位のオプション 0 : 緯度・経度を表示 (V1.0と同じ) 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) | 省略時は 0 |
| | repeatCount | 測位・出力の回数 (option=0,1,2の場合) または行数 (option=3の場合) 1~100までの回数を指定 (出力間隔は測位状況によるが1秒間隔以上) | 省略値は 1 |

次頁につづく

※**NMEAフォーマット**は、GPSの出力フォーマットで、主に以下の様なセンテンスとなっています。(以下参考)

| センテンス | NMEAセンテンス内容(概要) | 備考 |
|----------|---|-----------------|
| \$ GPGGA | UTC,緯度,N,経度,E,品質,使用衛星数,水平精度低下率,海拔高さ,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 |

1. LOCATION GPS ②

■ GPS(全地球測位システム) の取得コマンド (つづき)

前頁よりつづく

| 項目 | 値など | 説明 | 補足 |
|----------|--|--|--------------------|
| 応答値 | 【正常時】 | \$LG=OK latitude longitude¥n | option=0の時 |
| | | \$LG=OK latitude longitude utc quality number height¥n | option=1の時 |
| | | \$LG=OK "gpgga"¥n | option=2の時 |
| | | \$LG=OK ¥nmea_sen¥n... | option=3の時 |
| | latitude | 緯度 (北緯、9.99999形式、ただし桁数は場合により可変) | option=0の時 |
| | 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 : differential GPS (干渉測位方式) モードの何れか | option=1の時 上記含む |
| | number | 捕捉衛星数 | |
| | height | GPSアンテナの海拔高さ (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) | option=2の時 | |
| nmea_sen | 各NMEAフォーマット出力 | option=3の時 | |
| 【エラー時】 | errno | \$LG=NG errno¥n | |
| | | 501 : 引数指定エラー | |
| | | 508 : GPS測位エラー (タイムアウトエラー : 3分間) 509 : BUSYエラー (すでに測位中) | |
| 前提条件 | ① | GPSアンテナが正しく装着されてること | |
| 補足事項 | ① | 測位には、初回には数分以上の時間がかかる場合がある。 | |
| | ② | AGPSサーバとして、Googleのロケーションサーバを利用する。 | |
| | ③ | repeatCountを2以上で指定した場合は、指定回数出力が終わるまでは制御は戻らない (次のコマンドは受け付けない) | |

※ 短時間でGPS取得するには、**Assisted GPS** をご利用ください。Assisted GPSについては、

1. LOCATION GPS ②

■ 事例 : GPS取得プログラム

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4, 5);
const unsigned long baudrate = 38400;

void setup() {
  Serial.begin(baudrate);
  Serial3g.begin(baudrate);
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH);
  Serial.println("begin Location GPS");
  delay(35000);
}

void loop() {
  static boolean sw=true;
  if(sw) {
    Serial3g.write("$LG MSBASED\r\n");
    Serial.println("$LG MSBASED");
  }
  sw=false;
  while(!Serial3g.available());
  String rt = Serial3g.readStringUntil('\r\n');
  Serial.println(rt);
  if(rt.startsWith("$LG=NG 508")){ sw=true;
  }
  else if ( rt.startsWith("$LG=OK")) {
    Serial.println("end "); while(1);
  }
  if(rt.startsWith("$LG=NG 509")) sw=false;
}
```

■ シリアルモニタ画面（正常時応答）

```
begin Location GPS
$LG MSBASED
$LG=OK 35.62212345 139.5912345
end
```

```
begin Location GPS
$LG x 9
$LG=OK
end
```

■ GPS取得関数

```
String GPSget() {
  Serial3g.println("$LG MSBASED");
  String rstr;
  unsigned long tim = millis(); // time set(ms)
  do{ while(!Serial3g.isListening());
    rstr=Serial3g.readStringUntil('\r\n');
    if(rstr.length()>0) Serial.println(rstr); //debug print...
    if(rstr.indexOf("$LG=NG 508")==0) Serial3g.println("$LG MSBASED");
  }while(!(rstr.indexOf("$LG=OK")==0));
  return(rstr.substring(7));
}
```

【補足説明：初回のGPS取得できない理由】

- ※ \$LG=NG 508 (T/O)が時間オーバーで返ってくる時
- 1) 屋内などのGPS衛星がとらえられない
→ なるべく屋外などで電波状態が良い環境で実行
- 2) 数分でGPS取得できない
→ 初回は3分以上ほどかかる場合もあるので何度か実行
- 3) アンテナや電源の不備
 - ・GPSアンテナが正しく接続されていない
 - ・GPSアンテナの向きが悪い
 - ・外部電源供給が不足している
- 4) パソコンなどの近くでは電波障害でGPS取得しにくい
→ パソコン本体などから3 GIMを離して実行する

2. LOCATION GPS (Assisted GPS)

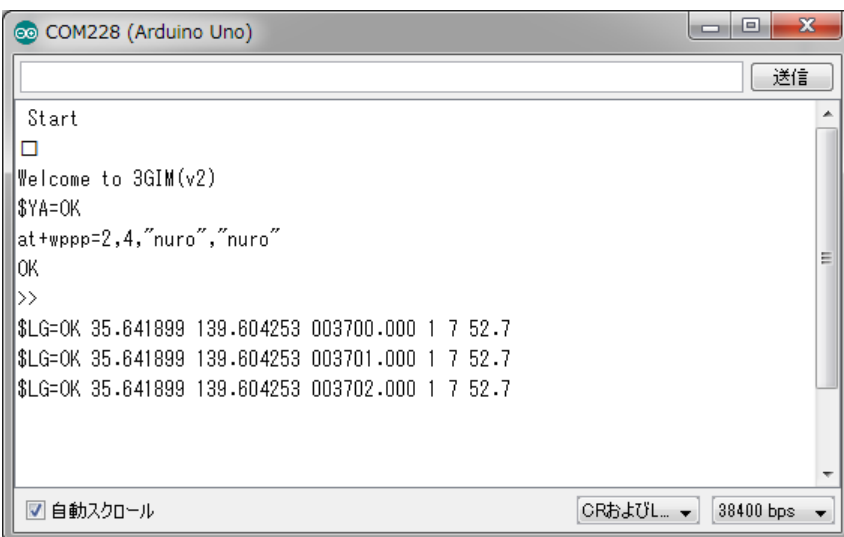
本**Assisted GPS**は、衛星の情報を3G通信基地局から受信し、いち早く位置情報を取得する技術です。

本3GIMでは、ATコマンドモードでのAssisted GSP機能を実行させるための以下の2行を実行しておくことで、切り替えることができます。

```
$YA 1
at+wppp=2,4,"ユーザ名","パスワード"
```

ここで「\$YA 1」は、ATコマンドモードに0.01秒間入ることになります。

次の「at+wppp=…」でのユーザ名とパスワードは、利用しているSIMカードのプロファイル情報となります。



```
COM228 (Arduino Uno)
Start
Welcome to 3GIM(v2)
$YA=OK
at+wppp=2,4,"nuro","nuro"
OK
>>
$LG=OK 35.641899 139.604253 003700.000 1 7 52.7
$LG=OK 35.641899 139.604253 003701.000 1 7 52.7
$LG=OK 35.641899 139.604253 003702.000 1 7 52.7
```

■ Arduinoでのサンプルスケッチ

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4, 5);
#define BAUDRATE 38400

void setup() {
  Serial.begin(BAUDRATE);
  Serial3g.begin(BAUDRATE);
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH); delay(100);
  digitalWrite(7,LOW);
  Serial.println(" Start");
  delay(14000);
  Serial3g.println("$YA 1");
  delay(5);
  Serial3g.println("at+wppp=2,4,¥\"ユーザ名¥\",¥\"パスワード¥\"");
  delay(100);
  Serial3g.println("$LG x 1 3");
}

void loop() {
  if (Serial3g.available() > 0)
    Serial.println(Serial3g.readStringUntil('\n'));
  if (Serial.available() > 0)
    Serial3g.println(Serial.readStringUntil('\n'));
}
```





4. Web関連

1. HTTP GET ①

■ HTTP GETによるネット接続

| 項目 | 値など | 説明 | 補足 |
|--------|----------|--|---|
| 機能分類 | Web | | |
| 機能名 | GET | HTTP/GETを指定されたURLへ送信して、レスポンスを取得する | ボディ部のみ取得できる |
| コマンド形式 | | \$WG url ["header"]¥n | カギ括弧 [] は、実際は不要 |
| 引数 | url | GETリクエストを送信するURL（例えば、"http://www.google.co.jp/"等） | URLエンコードされていること 先頭に"http://"または"https://"を含むこと |
| | header | ヘッダ情報（例えば、"Authorization: Basic QWxhZGRpbGl2ZmZlZQ=="等） | \$エンコードされていること。 Hostプロパティは省略可 |
| 応答値 | 【正常時】 | \$WG=OK nbytes¥nresponse¥n | |
| | nbytes | レスポンス文字列のバイト数（末尾の'¥n'は含まず） | 最大1023 |
| | response | レスポンスの文字列 | バイナリデータも取得可能 |
| | 【エラー時】 | \$WG=NG errno¥n | |
| 前提条件 | ① | 301：コマンド形式または引数指定エラー（urlの指定間違いも含む） | |
| | | 303～308：タイムアウトまたは内部エラー マイナス値：HTTPステータスコードの符号をマイナスにした値 | タイムアウトは30秒設定 値の範囲は-400～-599 |
| 補足事項 | ② | パケット通信サービスが利用できる状態であること。 | プロファイル設定が正しい事 |
| | ② | ヘッダ情報の指定は、gw3g R2.0以降のみで利用できる | |
| | ① | レスポンスにはヘッダ情報は含まれず、ボディ情報のみが含まれる。 | |
| | ② | レスポンスの文字コードは、urlで指定されたサーバに依存する。 | |
| | ③ | HTTPのバージョンは「1.1」としてGETまたはPOSTする。 | |
| | ④ | ヘッダにはUserAgentプロパティは含まれない。 | |
| | ⑤ | 本コマンドでは、レスポンスボディが大きい場合は、先頭の一部しか取得できない。ボディが大きい場合でもすべてを取得したい場合には、TCP機能を利用する。 | HTTP/GETのレスポンスボディが大きい場合、本コマンドの実行には最大35秒程度の時間がかかる |
| | ⑥ | urlの長さとはheaderの長さを合わせて、最大1024バイトまでとする。ただし、urlに含まれるホスト名は最大96バイトまでとする。 | |

1. HTTP GET ②

■事例：ネット接続サンプルプログラム

■Arduinoサンプルプログラム

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup() {
  Serial.begin(38400);
  Serial3g.begin(38400);
  delay(23000);
  Serial.println("begin HTTP GET");
  delay(1000);
  Serial3g.println(" ($WG http://tabrain.jp/demo/httpGET_test.txt");
  delay(1000);
  unsigned long tm = millis();
  while (millis() - tm < 35000) {
    while (Serial3g.available()) {
      char c = Serial3g.read();
      Serial.print(c);
    }
  }
  Serial.print("¥r¥nend");
}
void loop() {}
```

■シリアルモニタ画面（正常時応答）

```
begin HTTP GET
$WG=OK 44
Tabrain Web site
Complete access from 3GIM

end
```

■ www.tabrain.jp/demo/httpGET_test.txtのファイル内容

サンプルデータ

Tabrain Web site
Complete access from 3GIM

URLコードの変換が必要が文字（5文字）

| | | | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 文字 | | ! | " | # | \$ | % | & | ' | (|) | * | + | , | - | . | / | |
| コード | %20 | %21 | %22 | %23 | %24 | %25 | %26 | %27 | %28 | %29 | %2A | %2B | %2C | %2D | %2E | %2F | |
| 文字 | : | ; | < | = | > | ? | @ | [| ¥ |] | ^ | _ | ` | { | | } | ~ |
| コード | %3A | %3B | %3C | %3D | %3E | %3F | %40 | %5B | %5C | %5D | %5E | %5F | %60 | %7B | %7C | %7D | %7E |

2. HTTP POST ①

■ HTTP POSTによるネット接続

| 項目 | 値など | 説明 | 補足 | |
|--------|----------|-------------------------------------|--------------------------------------|----------------|
| 機能分類 | Web | | | |
| 機能名 | POST | HTTP/POSTを指定されたURLへ送信して、レスポンスを取得する | | |
| コマンド形式 | | \$WP url "body" ["header"]¥n | カギ括弧 [] は、実際は不要 | |
| 引数 | url | POSTリクエストを送信するURL | 最大256バイト（\$エンコードされていること） | |
| | body | POSTするボディ | 最大1024バイト（"） | |
| | header | ヘッダ情報 | 最大256バイト（"）、省略可 | |
| 応答値 | 【正常時】 | \$WP=OK nbytes¥nresponse¥n | | |
| | nbytes | レスポンス文字列のバイト数（デコード前のサイズ） | 最大1023バイト | |
| | response | レスポンスの文字列（エンコードされた文字列） | バイナリデータも取得可能 | |
| | 【エラー時】 | \$WP=NG errno ..¥n | | |
| 前提条件 | ① | 301：コマンド形式または引数指定エラー（urlの指定間違いも含む） | | |
| | | 303～308：タイムアウトまたは内部エラー | | |
| 補足事項 | ② | マイナス値：HTTPステータスコードの符号をマイナスにした値 | 値の範囲は-400～-599 | |
| | | ① | パケット通信サービスが利用できる状態であること。 | プロファイル設定が正しいこと |
| | | ② | 全ての引数の長さの合計は改行'¥n'を含み1088バイト以下であること。 | |
| 補足事項 | ③ | ① | レスポンスにはヘッダ部は含まれず、ボディ部のみが含まれる。 | |
| | | ② | レスポンスの文字コードは、urlで指定されたサーバの実装に依存する。 | |
| 補足事項 | ③～⑥ | HTTP GETの補足事項を参照のこと | | |

※ 「errno」は、Wikiページなどで補足説明していきます。

2. HTTP POST ②

■ 事例 : HTTP POSTによるツイート参照

本事例は、後述しています
「3GIMでのツイッター連携使用例」
を参考にしてください。

- 本関数は、\$WG または \$WPを含んだコマンド文字列を引数として送る関数

```
//===== $WG & $WP command =====  
boolean _3G_WGP(String command) {  
  Serial3g.println(command);  
  String rstr;  
  unsigned long tim = millis(); // time set(ms)  
  do{  
    while(!Serial3g.available ());  
    rstr=Serial3g.readStringUntil('\n');  
  } while (rstr.indexOf("$W") != 0 && (millis() - tim) < LIMITTIME);  
  return (rstr.indexOf("$W") == 0);  
}  
//=====
```

ここで Serial3g は、シリアル通信ポート
LIMITTIMEは、制限時間 (14000msec) を設定のこと



5. TCP/IP関連



1. TCP/IP READ

■ コネクションからのデータ読み込み

| 項目 | 値など | 説明 | 補足 | |
|----------------------------|-----------------|---|------------------------|--|
| 機能分類 | TCP/IP | | | |
| 機能名 | READ | 現在のコネクションからデータを読み出す | ノンブロッキングで動作する | |
| コマンド形式 | | \$TR maxbytes¥n | | |
| 引数 | maxbytes | 読み出すデータの最大長 (バイト) | 最大1024 | |
| 応答値 | 【正常時】 | \$TR=OK nbytes¥ndata¥n | | |
| | nbytes | 読み出したデータのバイト数 (≦maxbytes) 、このバイト数には末尾の¥nは含まない | 指定バイト数以下の場合もある | |
| | data | 読み出したデータ | gw3g R2.0からバイナリデータも取扱可 | |
| | 【エラー時】 | \$TR=NG errno ..¥n | | |
| | errno | 631 : コマンドの指定、または引数に誤りがある | | |
| | | 633 : READエラー | | |
| 635 : コネクションがない (未接続) | | | | |
| 636、637 : コネクションのステータスのエラー | | | | |
| | 639 : タイムアウトエラー | | タイムアウト時間は60秒 | |
| 前提条件 | ① | パケット通信サービスが利用できる状態であること。 | プロファイル設定が正しいこと | |
| | ② | TCP/IPコネクションが確立されていること | | |
| 補足事項 | ① | 相手から受信したデータをそのまま加工せず取得する | | |
| | ② | 呼び出された時に 3 GIMに届いているデータを、最大 msxbytes分まで読み出す。 常にブロッキングはせず、データがない時は nbytes=0 で直ちに戻る。 | R2.0から常にノンブロッキングで動作 | |
| | ③ | 読み出す前にコネクションの状態チェックを行うため、コネクションが切断されている場合には直ちに制御が戻る。 | | |

2. TCP/IP WRITE

■ コネクションへのデータ書き出し

| 項目 | 値など | 説明 | 補足 |
|--------|--------|---|---|
| 機能分類 | TCP/IP | | |
| 機能名 | WRITE | 現在のコネクションへデータを書き出す | |
| コマンド形式 | | \$TW "data"¥n | ダブルクォートを省略することもできるが、推奨しない。 |
| 引数 | data | 書き出すデータ | 最大1080バイトまで。ダブルクォートで囲む場合は\$エンコードされていること。 |
| 応答値 | 【正常時】 | \$TW=OK nbytes¥n | |
| | nbytes | 書き出したデータのバイト数（実際に相手に書き出したサイズ） | 最大1024バイト |
| | 【エラー時】 | \$TW=NG errno ..¥n | |
| | errno | 621 : コマンドの指定、または引数に誤りがある 623 : WRITEエラー 625 : コネクションがない（未接続） 629 : タイムアウトエラー | タイムアウト時間は60 |
| 前提条件 | ① | パケット通信サービスが利用できる状態であること。 | プロファイル設定が正しいこと |
| | ② | TCP/IPコネクションが確立されていること | |
| 補足事項 | ① | dataとして指定できるデータは\$エスケープシーケンスにてエンコードされている必要がある。 | バイナリデータを送りたい場合は、最低限、0x00(ヌル)、0x0a(LF)、0x22(")、0x24(\$)の4つのバイト値を\$でエスケープすればよい。 |
| | ② | dataとして指定できるデータは、エンコード前の生データのサイズが1024バイト以下であること。ただし、指定できるデータdataの長さは、コマンド文字列やダブルクォート等を含みコマンド行の最大長(改行を含み1088バイト)以下であること。 | |
| | ③ | 書き出す前にコネクションの状態チェックは行わない。そのため、相手方がコネクションを切断している場合は、タイムアウト時間が経過した後にエラーとなって制御が戻る。 | 性能を優先するために、コネクションの存在チェックを毎行行わない仕様としている。 |

3. TCP/IP CONNECT

■ コネクション接続

| 項目 | 値など | 説明 | 補足 |
|--------|------------|---|--|
| 機能分類 | TCP/IP | | |
| 機能名 | CONNECT | TCP/IPコネクションを接続する | |
| コマンド形式 | | \$TC host_or_ip port¥n | |
| 引数 | host_or_ip | 接続するホスト名またはIPアドレス | |
| | port | 接続するポート番号 | 1~65535の範囲 |
| 応答値 | 【正常時】 | \$TC=OK¥n | |
| | 【エラー時】 | \$TC=NG errno ..¥n | |
| | errno | 601 : 引数がおかしい | |
| | | 603 : すでに接続済み 604 : コネクションエラー(ホストやポートが間違っている場合を含む) | タイムアウト時間は60秒 |
| 前提条件 | ① | パケット通信サービスが利用できる状態であること。 | プロファイル設定が正しいこと |
| | ② | TCP/IPコネクションが確立されていること | |
| 補足事項 | ① | TCP/IPコネクションは一度に一つだけ使用できる。コネクションはWeb機能とは独立しているため、Web機能と同時に利用することができる。 | 例えば、TCP機能であるサーバとコネクションをつないだままで、Web機能を利用できる |

GET / HTTP/1.1

Accept: image/gif, image/jpeg, */*

Accept-Language: ja Accept-Encoding: gzip, deflate User-Agent: Mozilla/4.0 (Compatible; MSIE 6.0; Windows NT 5.1;) Host: www.xxx.zzz

Connection: Keep-Alive

4. TCP/IP DISCONNECT

■ コネクション切断

| 項目 | 値など | 説明 | 補足 |
|----------------|------------|---------------------------|----------------|
| 機能分類 | TCP/IP | | |
| 機能名 | DISCONNECT | 現在のTCP/IPコネクションを切断する | |
| コマンド形式 | | \$TD¥n | |
| 引数 | | | |
| 応答値 | 【正常時】 | \$TD=OK¥n | |
| | 【エラー時】 | \$TD=NG errno ..¥n | |
| | errno | 611 : コマンドの形式に誤りがある | |
| | | 614 : 内部エラー(Close) | |
| 615 : 接続されていない | | | |
| 前提条件 | ① | パケット通信サービスが利用できる状態であること。 | プロファイル設定が正しいこと |
| | ② | TCP/IPコネクションが確立されていること | |
| | ③ | read中あるいはwrite中ではないこと | |
| 補足事項 | | | |

5. TCP/IP STATUS

■ コネクション状態の取得および状態設定

| 項目 | 値など | 説明 | 補足 | |
|--------|---------------|--|---|--|
| 機能分類 | TCP/IP | | | |
| 機能名 | STATUS | 現在のTCPコネクションの状態を取得する | | |
| コマンド形式 | 取得 | \$TS [option]¥n | | |
| 引数 | option | 1 : 付加情報を出力する | 省略時は詳細情報は出力しない | |
| 応答値 | 【正常時】 | \$TS=OK status¥n \$TS=OK status tcpnotif remainedBytes receivedBytes¥n | option省略時 option=1の時 | |
| | status | 0 : CLOSED (接続なし) 1 : DISCONNECTING 2 : DISCONNECTED (接続待ち) 3 : CONNECTING 4 : CONNECTED (送受信待ち) | | |
| | tcpnotif | 下記の【TCP機能全般の留意点】を参照 | オプション(1)が指定された時のみ出力される | |
| | remainedBytes | 未送信状態のデータのバイト数 | 同上 | |
| | receivedBytes | 受信状態のデータのバイト数 (\$TRコマンドでの読み出し可能なバイト数) | 同上 | |
| | 【エラー時】 | \$TS=NG errno ..¥n | | |
| | errno | 641 : 引数がおかしい 642 : ステータス取得エラー | | |
| | 前提条件 | | | |
| | 補足事項 | ① | 本コマンド実行時にTCPコネクションの状態を取得して結果を返却するため、正確な情報が得られる。 | |

6. TCP/IP GETSOCKNAME

■ コネクション状態のIPアドレスとポート番号取得

| 項目 | 値など | 説明 | 補足 |
|--------|--------------|---|----------------|
| 機能分類 | TCP/IP | | |
| 機能名 | Get Sockname | 自分のIPアドレスを取得する | ポート番号は取得できない |
| コマンド形式 | 取得 | \$TN¥n | |
| 引数 | | | |
| 応答値 | 【正常時】 | \$TN=OK ipAddr ¥n | |
| | ipAddr | 自分のIPアドレス(IP v4のみサポート) | |
| | 【エラー時】 | \$TN=NG errno ..¥n | |
| | errno | 663 : コマンドの形式に誤りがある 662 : 接続していない 661 : IPアドレス取得エラー | |
| 前提条件 | ① | パケット通信サービスが利用できる状態であること。 | プロファイル設定が正しいこと |
| | ② | TCP/IPコネクションが確立されていること | |
| 補足事項 | | | |

7. TCP/IP TUNNEL WRITE

■現在のコネクションにダイレクトにデータ書き出し

| 項目 | 値など | 説明 | 補足 |
|--------|--------------|--|---|
| 機能分類 | TCP/IP | | |
| 機能名 | TUNNEL WRITE | 現在のコネクションへダイレクトにデータを書き出す | サイズの大きいバイナリデータをサーバへ送信する手段として最適である。 |
| コマンド形式 | | \$TT nbytes¥ndata | dataの後に改行は不要である |
| 引数 | nbytes | 書き出すデータのバイト数 | 最大32000バイト |
| | data | 書き出すデータ | \$エンコードは不要、バイナリデータもそのままOK |
| 返却値 | 【正常時】 | \$TT=OK nbytes¥n | |
| | nbytes | 書き出したデータのバイト数 | |
| | 【エラー時】 | \$TT=NG errno ..¥n | |
| | errno | 621：コマンドの指定、または引数に誤りがある 623：WRITEエラー 625：コネクションがない（未接続） | |
| 前提条件 | ① | パケット通信サービスが利用できる状態であること。 | プロファイル設定が正しいこと |
| | ② | TCP/IPコネクションが確立されていること | |
| 補足事項 | ① | dataとして指定できるデータの内容はバイナリデータでもよい。\$エンコードの必要はなく、ヌルデータ(0x00)を含めてそのまま書き出すことができる。 | |
| | ② | 最初に指定したデータサイズ(バイト数)分を必ず書き出す必要がある。書き出すデータのサイズが最初に指定したサイズに満たない場合は、タイムアウト後に半角スペースが自動で補填される。 | |
| | ③ | 書き出す前にコネクションの状態チェックは行わない。そのため、相手方がコネクションを切断している場合は、タイムアウト時間が経過した後にエラーとなって制御に戻る。 | 性能を優先するために、コネクションの存在チェックを毎回行わない仕様としている。 |
| | ④ | ボーレートを115200bpsに設定している場合に限り、概ね1024バイトを送信するたびに10ミリ秒程度のディレイ時間を設けることを推奨する。 | |

8. TCP/IP 利用サンプルプログラム ①

■ 事例 : TCP/IP関連一覧のコマンドを使ったサンプルプログラム

■ Arduinoサンプルプログラム

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup() {
  pinMode(7,OUTPUT);
  digitalWrite(7,LOW); delay(100); digitalWrite(7,HIGH);
  Serial.begin(38400); Serial3g.begin(38400);
  Serial.println("Ready...");
  while (true) {
    unsigned long tim=millis();
    while (!Serial3g.available() && (millis() - tim) < 15000) ;
    String st=Serial3g.readStringUntil('\n');
    Serial.println(st);
    if (st.indexOf("3GIM")>0)
      break;
    else if ((millis()-tim) > 15000) {
      Serial.println("connect error");
      while(1) ;
    }
  }
  Serial.println("Start..");
  Serial.println("begin TCP/IP sample"); delay(100);
  while (!tcpprintln("$TC red www.tabrain.jp 80"));
  tcpprintln("$TW ¥"GET / HTTP/1.0$r$n¥");
  tcpprintln("$TW ¥"HOST: www.tabrain.jp$r$n$r$n¥");
  tcpprintln("$TR 200");
  tcpprintln("$TD");
  Serial.print("end");
}
void loop() {}
```

```
boolean tcpprintln(String ttc) {
  String rts="";
  uint32_t tm=millis();
  Serial3g.println(ttc);
  do {
    while (!Serial3g.available() && (millis()-tm<30000)) ;
    rts=Serial3g.readStringUntil('\n');
    Serial.println( rts );
  } while (rts.indexOf("$T") < 0);
  char ch;
  do {
    ch=Serial3g.read();
    if (0x20<ch && ch<0x80 ) Serial.print(ch);
  } while (Serial3g.available());
  return (rts.indexOf("=OK") > 0);
}
```

9. TCP/IP 利用サンプルプログラム ②

■ 事例 : TCP/IP関連一覧のコマンドを使ったサンプルプログラムの出力結果

■ シリアルモニタ画面 (正常時応答)

```
Ready...

Welcome to 3GIM(v2)
Start...
begin TCP/IP sample
$TC=OK

$TW=OK 16

$TW=OK 24

$TR=OK 200
HTTP/1.1 200 OK
Date: Su
,14Feb201622:07:39GMTServer:Apache/2.2.23(Unix)mod_ssl/2.2.3
OpnSSL/10.1mLas-Modified:ri,1Fe2010551:22MTTag:"384595-117-
2b8c1cad3"Acce$TD=NG 614

end
```

読み込みバッファの出力結果
(ここでは200バイト出力)

10. TCP/IP 機能の留意点

【TCP機能全般の留意点】

1) \$TR/\$TW/\$TT/\$TN/\$TSでコマンド形式エラー以外のエラーが発生した場合は、一旦、\$TDでコネクションを切断して、\$TCからやり直すこと。

2) \$TSコマンドで取得できるtcpnotifの値の意味は下記の通り：

- 0 Network error
- 1 No more sockets available; max. number already reached
- 2 Memory problem
- 3 DNS error
- 4 TCP disconnection by the server or remote client
- 5 TCP connection error
- 6 Generic error
- 7 Fail to accept client request's
- 8 Data sending is OK but KTCPSND was waiting more or less characters
- 9 Bad session ID
- 10 Session is already running
- 11 All sessions are used



6. Profile関連

1. PROFILE SET/GET①

■SIMカードのプロファイル情報の設定・取得

| 項目 | 値など | 説明 | 補足 |
|--------|-----------------|--|------------|
| 機能分類 | PROFILE | | |
| 機能名 | SET | デフォルトのプロファイル情報を設定・取得する | |
| コマンド形式 | 設定 | \$PS apn user password¥n | ※ |
| | 取得 | \$PS¥n | |
| 引数 | apn | APN情報 (例えば: iijmio.jp) | |
| | user | ユーザ名 (例えば: mio@iij) | |
| | password | 認証用パスワード (例えば: iij) | |
| 応答値 | 【正常時】 | \$PS=OK¥n \$PS=OK "apn","user","password"¥n | 設定時 取得時 |
| | 【エラー時】 errno | \$PS=NG errno¥n 211: コマンドの形式に誤りがある 212: 内部エラーまたはSIMカード・アンテナなし | |
| 前提条件 | ① | 有効なSIMカード (利用可能なSIMカード) を装着しておく必要がある。 | |
| 補足事項 | ① | apnの長さ、userの長さ、passwordの長さの合計は、最大51文字まで | |
| | ② | デフォルトプロファイルは、一つだけ保持することができる。本機能で設定したプロファイルは、電源を切っても3GIM内に保持される。 | |
| | ③ | 出荷時のデフォルト状態では、iijmioのプロファイルが設定されている。 | |

※: 例えば「**\$PS iijmio.jp mio@iij iij¥n**」と設定する (特にダブルクォートで囲む必要はないが、囲んでもよい)
一部、ユーザ名やパスワードが無いSIMカードもあり、その場合には設定する必要はありません。

補足: SIMカードのプロファイル情報は、内部メモリに保存されますので、電源を切っても保存されたままとなります。

1. PROFILE SET/GET②

■ SIMカードのプロファイル情報の設定

■ Arduinoサンプルプログラム

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
void setup(){
  Serial.begin(38400);
  Serial3g.begin(38400);
  delay(35000);
  Serial.println("begin Profaile Set");
  Serial3g.println("$PS iijmio.jp mio@iij iij");
  while(!Serial3g.available());
  String rt = Serial3g.readStringUntil('\n');
  Serial.print(rt);
  Serial.print("\r\nend");
}
void loop() {}
```

■ シリアルモニタ画面（正常終了の場合）

```
begin Profaile Set
$PS=OK
end
```

■ プロファイル設定サンプル（他MVNO製品も同様に設定）

| SIMメーカー製品名 | 設定方法 |
|---------------|---|
| DOCOMO mopera | \$PS mopera.net |
| iijmio | \$PS iijmio.jp mio@iij iij |
| iijmobile | \$PS iijmobile.jp mobile@iij iij |
| SONET | \$PS so-net.jp nuro nuro |
| SORACOM | \$PS soracom.io sora sora |
| EXCITE | \$PS vmobile.jp bb@excite.co.jp excite |
| HI-HO | \$PS vmobile.jp lte@hi-ho hi-ho |
| BMOBILE | \$PS bmobile.ne.jp bmobile@u300 bmobile |
| DTI | \$PS dream.jp user@dream.jp dti |
| MMT | \$PS mmtcom.jp mmt@mmt mmt |

もくじ

1. Arduinoでのシリアルモニタ操作
2. 3GIMでのツイッター連携使用例
3. 3GIMでのクラウド連携使用例（1）
4. 3GIMでのクラウド連携使用例（2）
5. 3GIMでのクラウド連携使用例（3）
6. GPS機能を使った応用例
7. Arduino関連ライブラリ（a3gim2）
8. サンプルツール群



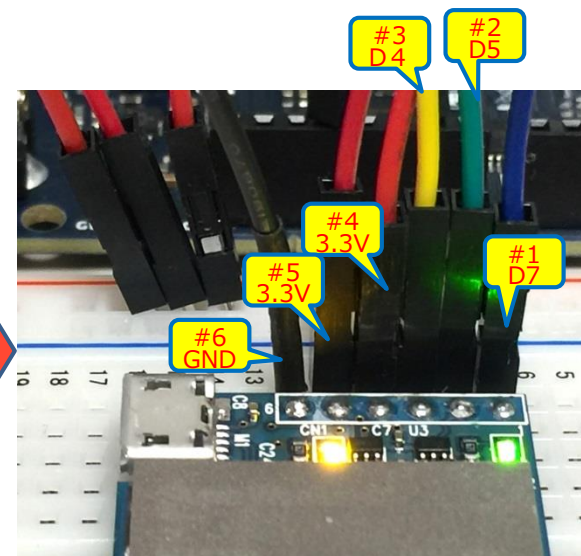
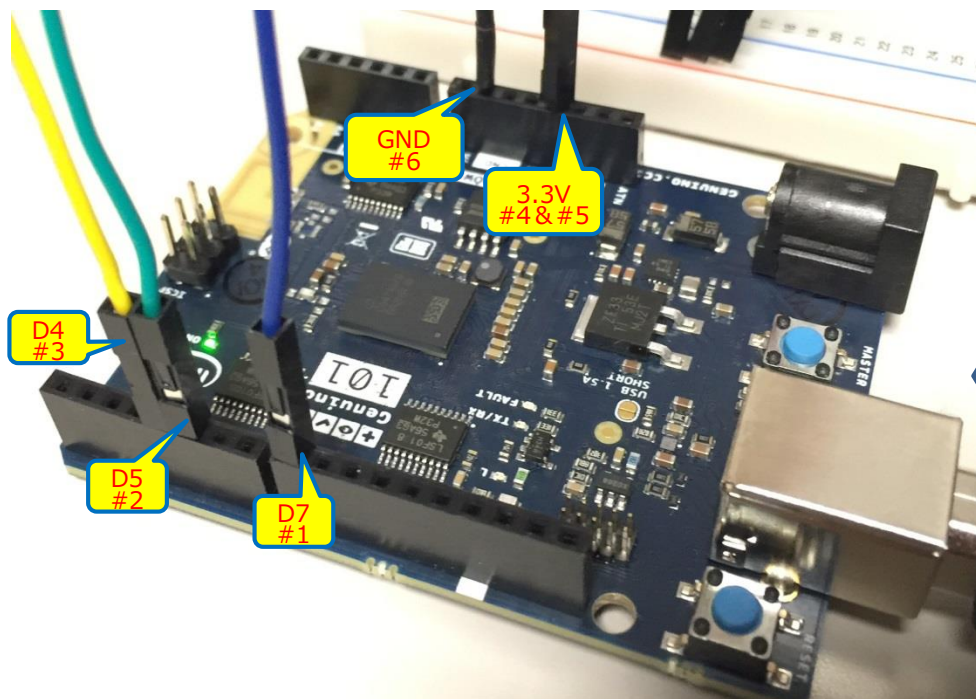
第4章 応用事例プログラミング



1. Arduinoでのシリアルモニタ操作



1. Genuino 101 との接続例



P.74 のサンプルスケッチを動かして
みてください。簡単に動くはずですよ。

※Genuino 101の3.3V出力が1 Aあるとのことで
3 GIMと接続してみました。

【注意】 場合によっては、外部電源を取る必要があるかもしれません。

2. Arduino上での簡単な利用例

■ソフトウェア（次頁の[monitor3gim.ino](#)）をコンパイルし、Genuino101に書き込んで動かしてみてください。

■必要な部品：

- ① Genuino101（IDEは、「[arduino.cc](#)」からダウンロードしてください）
- ② ジャンパワイヤおよびブレッドボード
- ③ マイクロSIMカード

■簡単な稼動テスト状況

- ① Arduino IDEのモニタ画面上に立上げメッセージ「Welcome to 3GIM(v2)」が表示された時点で立ち上がった段階です。
- ② 3GIMの各\$コマンドを入力して応答値を確認してみてください。

3. Arduinoシリアルモニタ画面操作スケッチ

- ▶ ArduinoUNO+3GIMシールド+ 3 GIM V2.1を接続し、シリアルモニタ画面上でコマンド入力して、その結果を見てみることにしてみましょう。
- ▶ Arduinoのスケッチは以下のとおりです。

■ シリアルモニタ画面での3 GIM入出力プログラム ([monitor3gim.ino](#) : *3Gシールド+Arduinoで利用可能)

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4, 5);
const unsigned long baudrate = 38400;

void setup() {
  Serial.begin(baudrate);
  Serial3g.begin(baudrate);
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH); delay(5);
  digitalWrite(7,LOW); //3GIMシールド電源ON
  Serial.println("Ready.");
}

void loop() {
  if (Serial3g.available() > 0) {
    char c = Serial3g.read();
    Serial.print(c);
  }

  if (Serial.available() > 0) {
    char c = Serial.read();
    Serial.print(c); // Echo back
    Serial3g.print(c);
  }
}
```

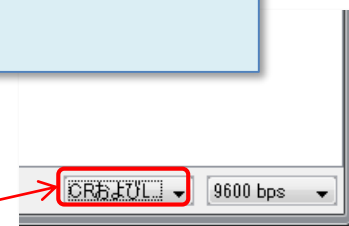
本サンプルスケッチは、シリアルモニタ画面で、コマンドをキー入力することで、応答（レスポンス）を表示確認できるもので、マニュアル操作でのコマンド/レスポンスが即座に見ることができます。

■ シリアルモニタ画面での操作例

```
Ready.
$YV
$YV=OK 2.0
$YI
$YI=OK 359516050164625
$YR
$YR=OK -68
$YT
$YT=OK 2016/02/06 17:55:11
$LG MSBASED
$LG=OK 35.64189613 139.6041995
$WG http://tabrain.jp/demo/httpGET_test.txt
$WG=OK 44
Tabrain Web site
Complete access from 3GIM
```

赤字：入力
青字：出力

補足：Arduino IDEのシリアルモニタ画面の改行モードは「CRおよびLF」に設定のこと





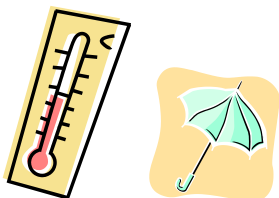
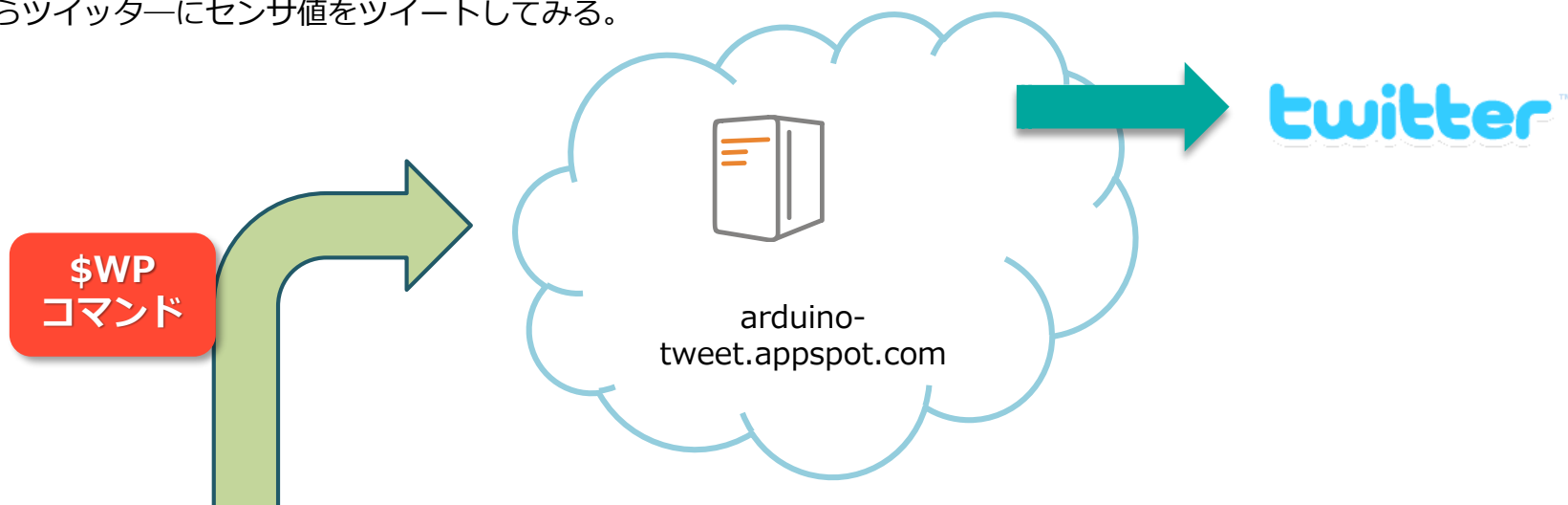
2. 3GIMでのツイッター連携使用例 (Arduinoの事例)



1. 3 GIMを使ったツイッター連携の利用イメージ

■ TLAの利用

Tweet Library for Arduino【TLA】を利用し、3 GIMからツイッターにセンサ値をツイートしてみる。



各種 センサ値
無償でアップ

【TLAの機能】

- twitterの認証を、トークンで代行してくれる
- http/POSTにより、twitterへ簡単に投稿できる

2. TLA利用手順①

- ①ブラウザで、下記のサイトにアクセスする
<http://arduino-tweet.appspot.com/>

Tweet Library for Arduino

Post messages to Twitter (tweet) from **Arduino** with **Ethernet Shield!**

How to begin:

Step 1: **Get a token to post a message using OAuth.**

Step 2: [Add some Libraries to your Arduino IDE.](#)

Step 3: [Run a sample sketch to tweet!](#)

Notice

- The library uses this site as a proxy server for OAuth stuff. Your tweet may not be applied during maintenance of this site.
- **Please avoid sending more than 1 request per minute** not to overload the server.
- Twitter seems to reject repeated tweets with the same content (returns error 403).

Reference

See [Arduino: Playground](#)

License

2. TLA利用手順②

- ② ツイッターのアカウント情報を入力する
(既にツイッターID登録済の場合には次に進んでください)

Authorize Arduino to use your account?

この連携アプリを認証すると、次の動作が許可されます。

- タイムラインのツイートを見る。
- フォローしている人を見る、新しくフォローする
- プロフィールを更新する。
- ツイートする

ユーザー名、またはメールアドレス
パスワード

保存する・パスワードを忘れた場合はこちら

連携アプリを認証 キャンセル

この連携アプリを認証しても、次の動作は許可されません。

- ダイレクトメッセージを見る。
- Twitterのパスワードを見る。

設定のアプリ連携からいつでも連携アプリの許可を取り消すことができます。
連携アプリを認証することでTwitterのサービス利用規約に同意したことになります。また、いくつかの連携アプリの利用情報はTwitterにも共有されます。詳細についてはプライバシーポリシーをご覧ください。

Arduino
開発者: NeoCat
arduino-tweet.appspot.com/
Twitter Library for Arduino: post a tweet easily using Arduino

新規登録

ユーザー名かメールアドレス と
パスワードを入力
(こちらが今後のID/PW)

次に選択

3. TLA利用手順③

③ トークンを記録しておく（コピー＆ペースト）



The screenshot shows a web browser window with the URL `arduino-tweet.appspot.com/oauth/twitter/callback?oauth_token=LThIi6Gck2dKKhL70Uud9dPJXEyov1nsjUsnLZKVDJs&oauth_verifier=wDEy0TMTHC9hPRm4XDL5t`. The page displays "Your token is:" followed by a text input field containing the token `134474081-wlUsVKUosDmeC6oVADV99...`. A red box highlights the token, and a red callout bubble points to it with the text "カット&ペーストで 選択・複写". A purple callout bubble points to the token with the text "こちらが必要なトークン".

4. 3 GIMでのツイッター連携使用例

■ 事例：ツイッターにメッセージアップのプログラム

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
const char *server = "http://arduino-tweet.appspot.com:80/update";
const char *token = "YOUR_TOKEN";

void setup() {
  Serial.begin(38400);
  Serial3g.begin(38400);
  Serial.print(" Plsese waite");
  delay(35000);
  Serial.println("Ready");
}

void loop() {
  char msgBuff[300];
  char msg[]="Twitter test";
  static int no = 0;// 番号のカウントアップ
  sprintf(msgBuff,"$WP %s ¥"token=%s&status=%s %d¥"¥n",server,token,msg,no++);
  Serial.println(msgBuff);
  Serial3g.println(msgBuff);
  String rt;
  do{
    while(!Serial3g.available());
    rt = Serial3g.readStringUntil('¥n');
    Serial.println(rt);
  } while(!rt.startsWith("$WP"));
  Serial.println("wait 1min");
  delay(60000);// ツイートは1分毎とする (制限事項)
}
```

ツイートのトークン

■ 実行モニタ画面（正常時）

```
Plsese waite
Ready
$WP *****
$WP=OK 2
wait 1min
```

株式会社タブレイン @tabrain · 21分
Twitter test 4

株式会社タブレイン @tabrain · 23分
Twitter test 3

株式会社タブレイン @tabrain · 25分
Twitter test 2

株式会社タブレイン @tabrain · 26分
Twitter test 1

株式会社タブレイン @tabrain · 28分
Twitter test 0

ツイートされた画面

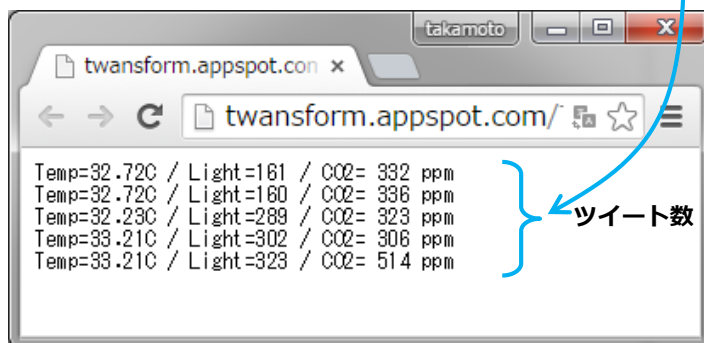
5. ツイートの読み込み①

- ツイートされた内容（値や文章）をクライアント側（3G端末側）に読み込むサンプル

ツイートされた内容を読むには、以下のアクセスで可能

<http://twansform.appspot.com/ツイッター名/text/5>

〈ブラウザでキー入力すると以下のような表示が返る〉



- 今度は、3G通信を使ってツイートされた内容（値や文章）を読み込むには、以下のスケッチなどで行う。

```
$WG http://twansform.appspot.com/ツイッター名/text/行数
```

※ここで ツイッター名は、@で始まるツイッター名で、@を取り除いた後ろの名前。行数は、最新版から取得するツイッター数

- 次頁のサンプルスケッチで実行した結果

```
COM9 (Arduino Uno)
>Ready.
  Initilaizing...
start
$WG http://twansform.appspot.com/tabrain/text/5
>W*=STARTING
>W*=GETHOSTBYNAME

>W*=CONNECT
>W*=SENDREQUEST
>W*=READRESPONSE
>WG=CONTENT_TYPE text/plain
>WG=READ(200Byte)
$WG=OK 200
  Twitter text :200
Temp=35.16C / Light=162 / CO2= 806 ppm
Temp=35.16C / Light=161 / CO2= 813 ppm
Temp=35.16C / Light=162 / CO2= 570 ppm
Temp=35.16C / Light=164 / CO2= 567 ppm
Temp=35.16C / Light=161 / CO2= 567 ppm
end ----

自動スクロール CRおよびL... 9600 bps
```

こちらがツイッター結果

5. ツイートの読み込み②

■ ツイートされたデータの読み込み

(3GIMの電源信号(ピン番号#1)は、Arduino側のD7ピンに接続)

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
SoftwareSerial D300(8,9);
const unsigned long baudrate = 38400;

#define LIMITTIME 15000 // ms (3G module start time)

String command = "$WG http://twansform.appspot.com/ツイッター名
/text/5";

//=====================================================
void setup() {
  Serial.begin(baudrate);
  Serial.println(">Ready. ¥r¥n Initilaizing...");
  if( _3Gsetup() ) {
    Serial.println("start");
  } else {
    Serial.println(" Connect Error ... Stop");
    while(1);
  }
  while(! _3G_WG(command));
  Serial.println("end ----");
}
//=====================================================
void loop () {}
```

成功するまで実行

■ 応用例

ツイートしたことで遠隔制御も可能

```
//===== 3G setup =====
boolean _3Gsetup() {
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH); delay(1000);// 3Gshield --> digitalWrite(7,LOW);
  digitalWrite(7,LOW); delay(100); // 3G shield --> digitalWrite(7,HIGH);
  Serial3g.begin(baudrate);
  //----- 3G module begin & connect -----
  String str;
  unsigned long tim = millis();
  do{ while(!Serial3g.isListening());
    str=Serial3g.readStringUntil('¥n');
  }while(!(str.indexOf("3GIM")>0) && (millis() - tim) < LIMITTIME);
  if( millis() -tim >= LIMITTIME) {
    return false;
  } else return true;
}
//===================================================== $WG command =====
boolean _3G_WG(String command) {
  Serial3g.println(command); Serial.println(command); // debug
  String rstr;
  unsigned long tim = millis(); // time set(ms)
  do{ while(!Serial3g.isListening());
    rstr=Serial3g.readStringUntil('¥n');
    Serial.println(rstr); //debug print...
  }while(!(rstr.indexOf("$WG=")==0));// $WP return check
  if(rstr.indexOf("$WG=OK")==0) {
    rstr=rstr.substring(7); int N=rstr.toInt();
    Serial.println(" Twitter text :" + rstr);
    for(int i=0; i<200; i++) {
      while(!Serial3g.available());
      char c=Serial3g.read();
      Serial.print(c);
    }
    return (true);
  }
  return(false);
}
```

3Gシールドの場合

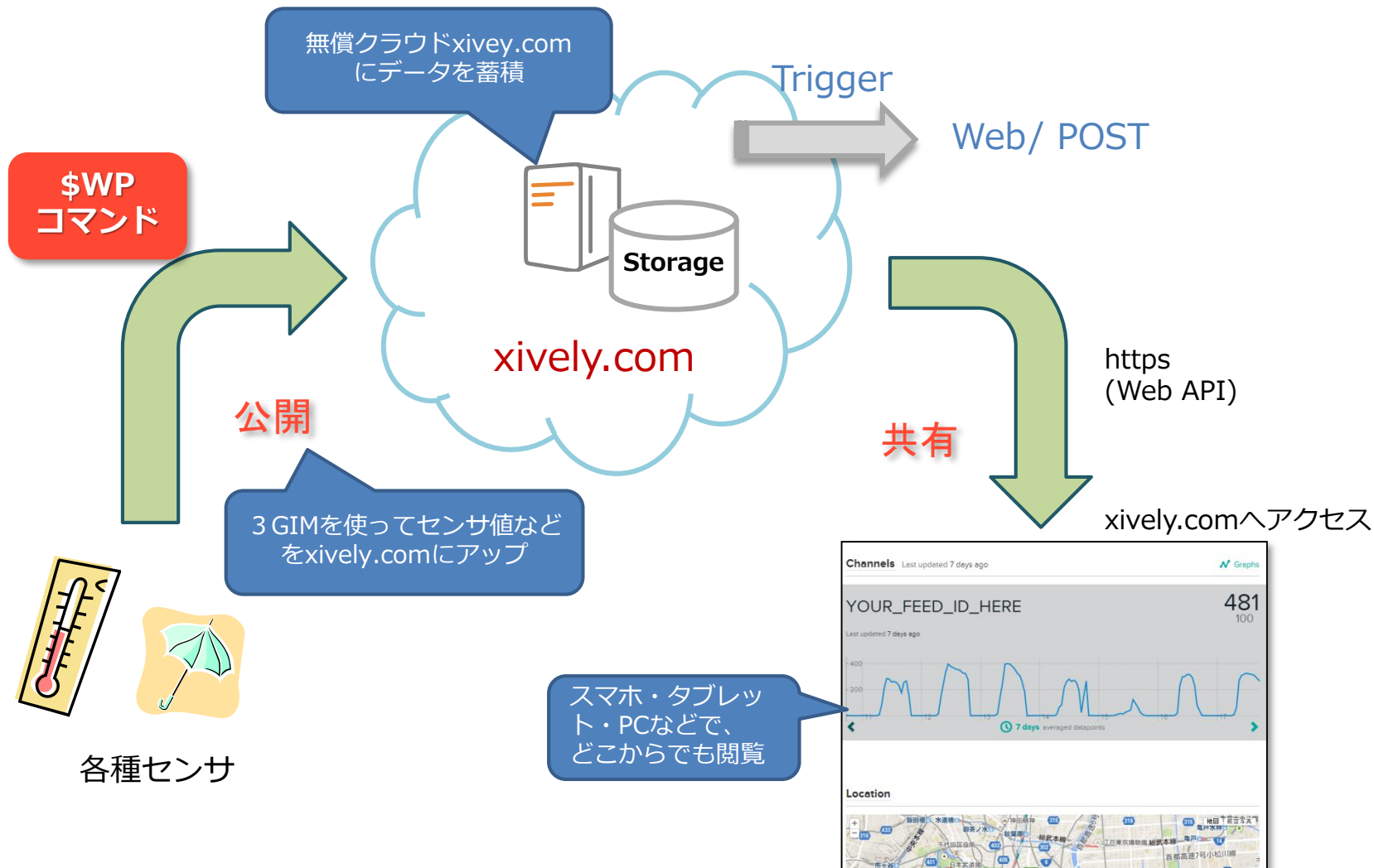
こちらを一部追加



3. 3GIMでのクラウド連携使用例① (xibely.com & Arduinoの事例)



1. xively.comの利用イメージ



2. xively.comの利用手順

注意：最近、新規登録に
長期間掛る状況にある

xively.comは、実績が豊富な無償のクラウドで、日本でも広く利用されています。まだ、英語版しかありませんが、グラフ表示やデータのアップやダウンロードだけの利用と考えると、問題なく簡単に使うことができます。

ここでは、本3GIMとセンサなどを使い、このxively.comにデータをアップしていくサンプルをご紹介します。

まずは、xively.comでの①ユーザ登録が必要で、その後各設定（②deviceの追加と③channelの追加）を行い、それら設定された値（④Feed IDとAPI Keyの確認）を使うことで、プログラミングしていきます。

【利用に当たっての注意点】

xively.comでは、無償の範囲での利用は、制限があります。特にデータのアップは、1分間に数回程度でしかできません。1秒毎とか頻繁にデータをアップしたりすると、利用できなくなることがあります。十分に気を付けてプログラミングしてください。

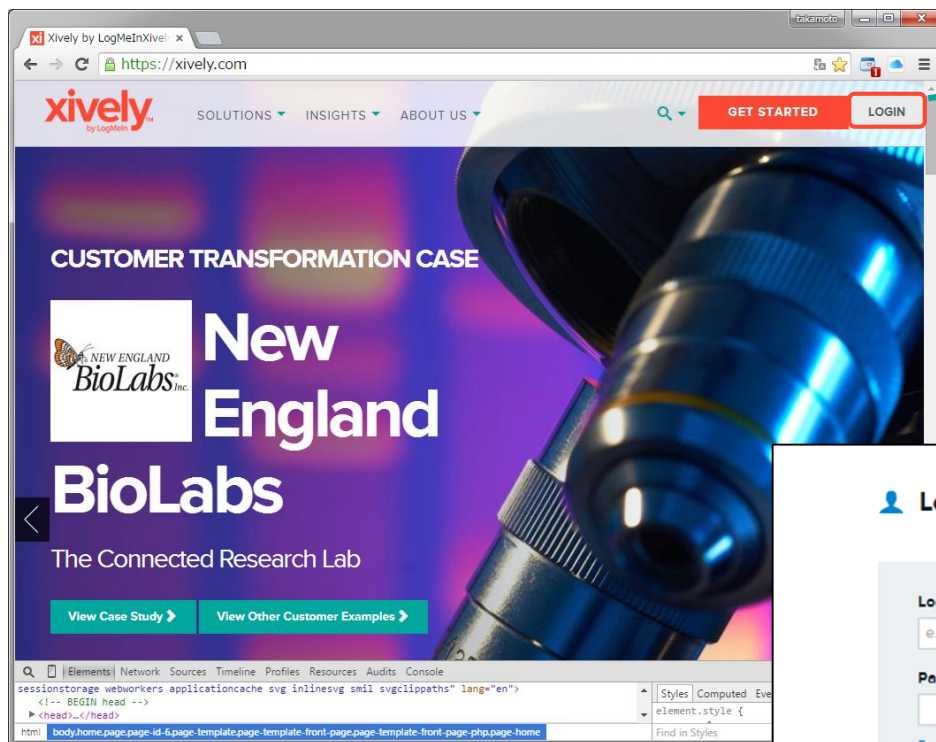
① ユーザの登録

② deviceの追加

③ channelの追加

④ Feed IDとAPI Keyの確認

3. xively.com ① ユーザ登録



http://xively.com/ にアクセス

Login

Login

e.g. johndoe

Login名

Password

Password

[Forgot Password](#)

Don't have an account? [Sign up here](#)

✓ Login

Remember me

予めユーザ登録しておいてください。

4. xively.com ②deviceの追加

The image shows two screenshots of the xively.com website. The left screenshot shows the 'Development Devices' page with a '+ Add Device' button. A red callout bubble points to the button with the text 'ここをクリックして次へ' (Click here to go next). A white box at the bottom of the left screenshot contains the text 'デバイスを追加してください' (Please add the device). A green arrow points from the bottom of the left screenshot to the top of the right screenshot.

The right screenshot shows the 'Add Device' form. It includes the following fields and options:

- Device Name:** A text input field with the placeholder 'e.g My Device'. A red callout bubble points to it with the text 'デバイス名(英数字記号)を入力' (Enter device name in alphanumeric characters).
- Device Description:** A text area with the placeholder 'Tell us more about this device'. A red callout bubble points to it with the text '説明文を入力' (Enter description).
- Privacy:** Two radio button options: 'Private Device' and 'Public Device'. A red callout bubble points to the 'Public Device' option with the text '機密性は無いのでPublicを選択' (Select Public because there is no confidentiality).

At the bottom of the form, there are two buttons: 'Add Device' (with a checkmark icon) and 'Cancel'.

5. xively.com ③chenellの追加

The image displays two screenshots of the xively.com web interface, illustrating the process of adding a channel to a device.

Left Screenshot: Shows the "Add Channels to your Device!" section. A red callout points to the "+ Add Channel" button with the text "ここをクリックして次へ". Below this, a blue callout says "つぎにチャンネルを追加".

Right Screenshot: Shows the "Add Channel" form. Three red callouts provide instructions:

- "ID(英数字・記号)を入力" points to the ID input field.
- "タグ、単位、記号を入力" points to the Tags, Units, and Symbol input fields.
- "初期値を入力" points to the Current Value input field.

6. xively.com ④ Feed IDとAPI Keyの確認

The screenshot displays the xively.com developer interface for a specific device. The browser address bar shows the URL: `https://personal.xively.com/develop/adHGhTHQKaJHg9lbMQLQ`. The page is divided into several sections:

- Public Device Information:** Lists various identifiers such as Product ID, Product Secret, Serial Number, and Activation Code.
- Feed Information:** Shows the Feed ID (highlighted with a red box and labeled "FEED_ID"), Feed URL, and API Endpoint.
- Channels:** A list of channels is shown, with "test01" (highlighted with a red box and labeled "CHANNEL_ID") selected. It displays a graph area with the text "No datapoints found for this channel".
- Request Log:** A table of recent requests, including a GET request for "channel test01" and a POST request for "feed".
- API Keys:** A section for managing API keys, showing an "Auto-generated 3GIM sample device key for feed" (highlighted with a red box and labeled "API_KEY") and its permissions.

7. 温度を測って定期的にxively.comへアップ

▶ 準備するもの

- ▶ Arduino UNO R3 など
- ▶ 温度センサ (LM61BIZ) など
- ▶ ブレッドボード
- ▶ ジャンパ線 (やわらかい線)
- ▶ 3GIM (あらかじめピンヘッダを半田付けしておく)
- ▶ マイクロSIMカード (3GIMで使えるもの)
- ▶ 3.7Vリチウムポリマ電池 (充電してあるもの) 、または3.7V出力可能なDC電源

▶ 接続方法

- ▶ 3GIMにマイクロSIMを挿入して、ブレッドボードにピンヘッダを刺す。
- ▶ #6(GND)を電源 (リチウムポリマ電池) のGNDとArduinoのGNDに接続、
- ▶ #5(VCC)を電源 (リチウムポリマ電池) の「+」に接続
- ▶ #4(IOREF)をArduinoの5V、#3(TX)をArduinoの**D4**、
#2(RX)をArduinoの**D5**、#1(PWR_ON)を**D7**に、それぞれジャンパ線で接続する。
- ▶ 温度センサをブレッドボードに刺して、センサのGNDをArduinoのGND、VddをArduinoの5V、VoutをArduinoの**A0**に、それぞれジャンパ線で接続する。

(※ここで、**D4**、**D5**、**D7**および**A0**は、Arduino I/Oポート入出力番号)

8. 温度を測って定期的にxively.comへアップ

▶ サンプルスケッチ

```
// Sample sketch for 3GIM
```

```
#include <SoftwareSerial.h>
```

```
const int PowerPin = 7; // D7
```

```
const int tmpPin = 0; // A0
```

```
const char *PostCmd = "$WP https://api.xively.com/v2/feeds/FEED_ID/datastreams/CHANNEL_ID?_method=put ";
```

```
const char *Header = "$X-ApiKey: API-KEY$r$nContent-Type: text/csv$r$n$";
```

```
// Global variables
```

```
uint32_t interval = 60000; // Interval time [mS] 1min
```

```
SoftwareSerial Serial3g(4, 5);
```

```
char body[20];
```

```
// setup() -- set up device
```

```
void setup() {
```

```
  pinMode(PowerPin, OUTPUT);
```

```
  digitalWrite(PowerPin, LOW); // 3GIM on
```

```
  Serial3g.begin(38400);
```

```
  delay(35000); // wait for start up 3gim
```

```
}
```

```
void loop() {
```

```
  // Sense temperature
```

```
  int tX10 = getTemperature() * 10;
```

```
  // upload sensing data to the xively.com
```

```
  uploadToCloud(tX10);
```

```
  // sleep a while
```

```
  delay(interval);
```

```
}
```

赤文字の箇所は、実際のxively.comの登録内容に沿って修正すること！

```
float getTemperature() {
```

```
  int mV = analogRead(tmpPin) * 4.88;
```

```
  return ((float)(mV - 600) / 10.0);
```

```
}
```

```
void uploadToCloud(int temp) {
```

```
  // upload temperature
```

```
  Serial3g.print(PostCmd);
```

```
  sprintf(body, "$%d.%d$", (temp / 10), abs(temp % 10));
```

```
  Serial3g.print(body);
```

```
  Serial3g.println(Header);
```

```
  Serial3g.flush();
```

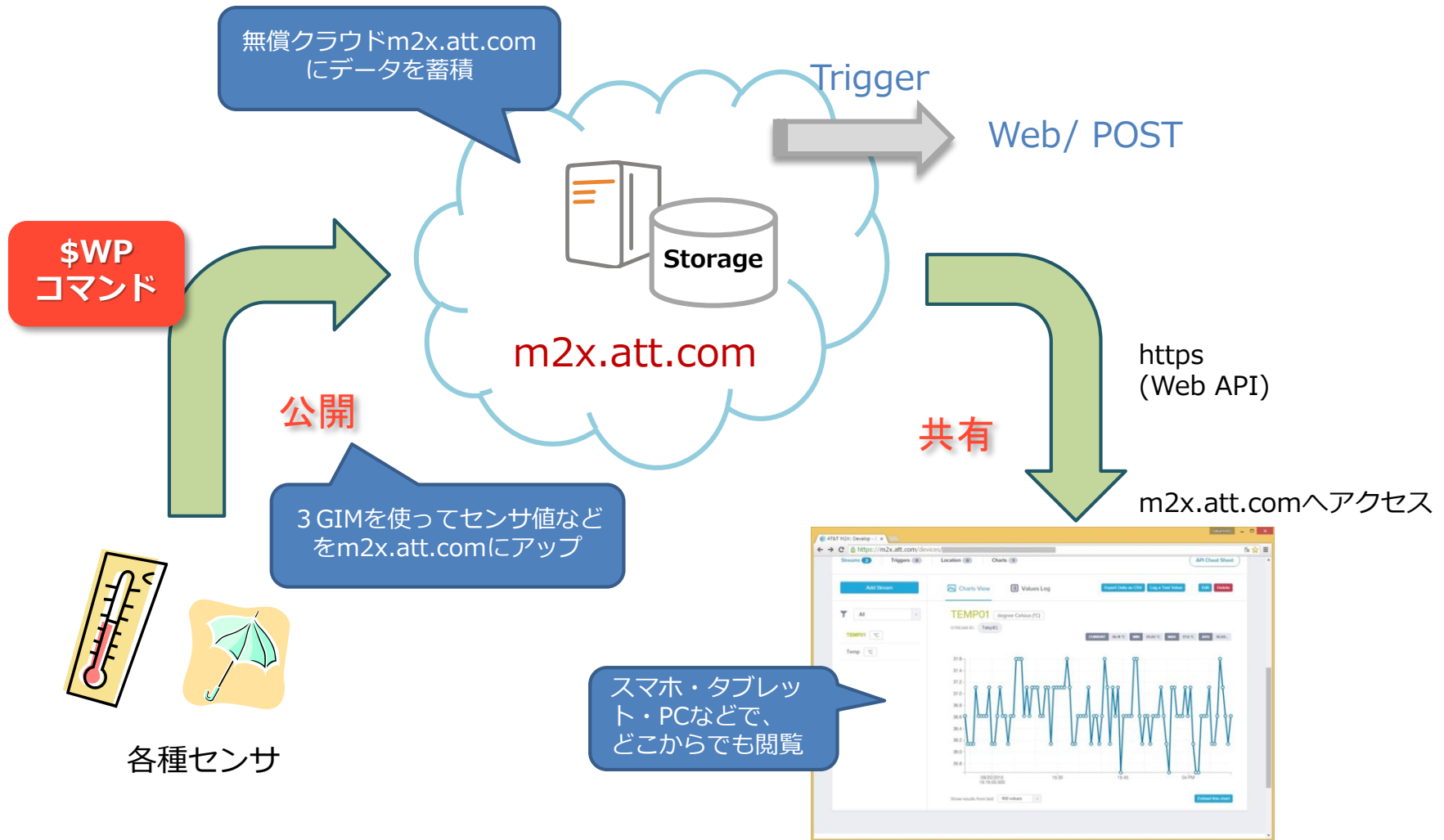
```
}
```



4. 3GIMでのクラウド連携使用例② (M2X & Arduinoの事例)



1. M2X (AT&T IoTサービス) の利用イメージ



2. M2X (AT&T IoTサービス) の利用手順

M2X (AT&T IoTサービス) は、2013年から米国通信事業最大手のAT&Tが、M2MおよびIoTビジネスに向けたサービスを開始したものです。

ArduinoやRaspberryPi、Mbedなど、多くのオープンソースハードウェアで利用できる環境を提供したものとなっています。

フリーで使え、センサデータの蓄積・グラフ表示、データのダウンロードなどができるようになっています。

ただ、時間設定が、世界標準のみで行なっていて、日本時間での表示や日本語表記が可能となりました。

グラフ表示やデータのアップやダウンロードだけの利用と考えると、問題なく簡単に使うことができます。

その他、トリガー機能が使え、センサ値が閾値を超えたときなどメールやツイッターを飛ばすことができます。

ここでは、本3Gシールドとセンサを使い、このm2x.att.comにデータをアップしていくサンプルをご紹介します。

詳細な規約等は、m2x関連の公開情報等をご参照ください。

① ユーザの登録

② device の追加

③ stream の追加

④ x-m2x-keyの確認

3. M2X (AT&T IoTサービス) のID登録

The image illustrates the steps to register for an M2X developer account. It consists of three overlapping screenshots of a web browser.

- Top Screenshot:** The M2X homepage at <https://m2x.att.com>. A red box highlights the "SIGN UP NOW" button, with a callout pointing to it that says "ID登録へ" (Go to ID registration).
- Middle Screenshot:** The "Create your Free Developer Account" page. A red box highlights the "Sign up with AT&T Developer Account" button, with a callout pointing to it that says "ID登録" (ID registration).
- Bottom Screenshot:** The "AT&T Developer Login" form. A red box highlights the "Email Address or Username" and "Password" input fields, with a callout pointing to them that says "メールアドレス (ユーザID) およびパスワード" (Email address (user ID) and password).

Arrows indicate the flow from the homepage to the sign-up page, and then to the login form.

http://m2x.att.com/ にアクセス

メールアドレス (ユーザID) およびパスワード

4. デバイス (Device) の作成登録

デバイス画面へ

新しいデバイス登録画面へ

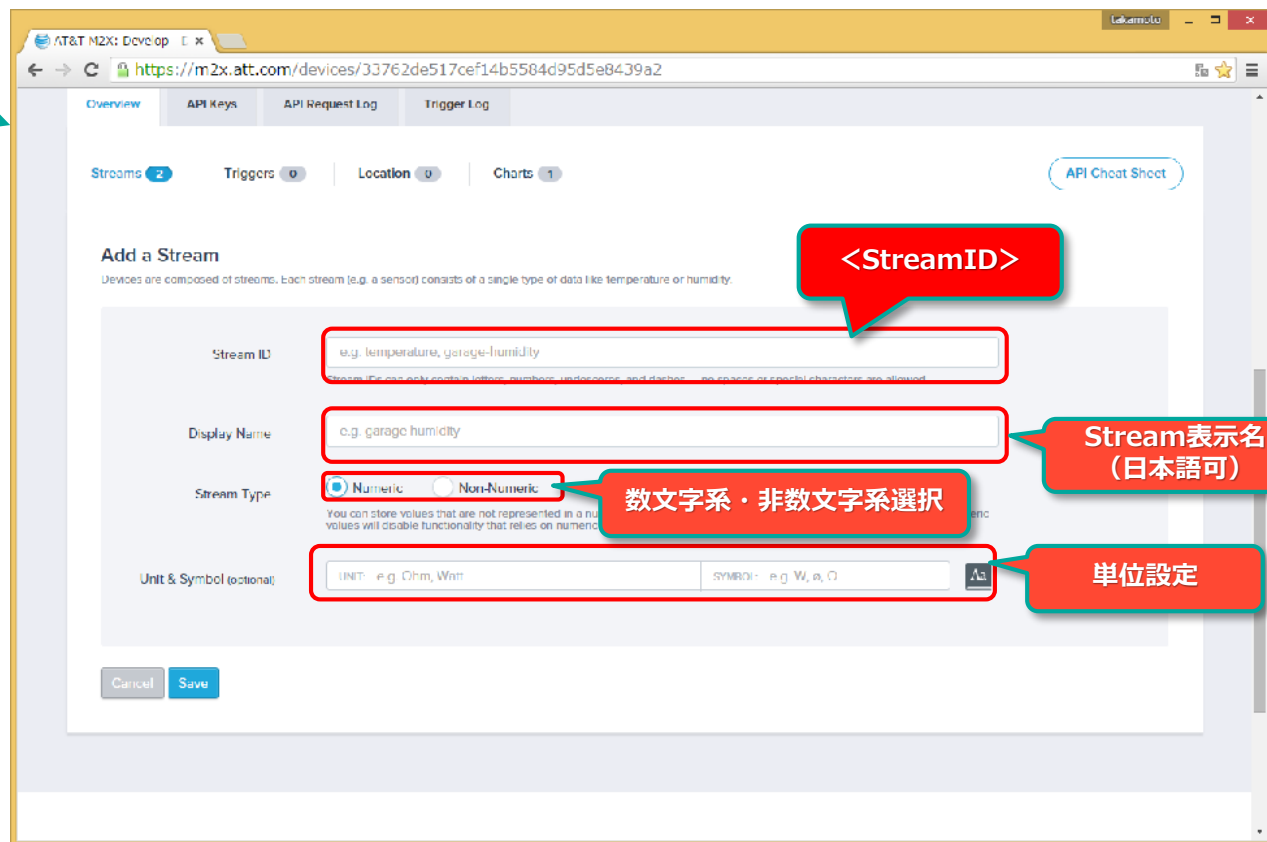
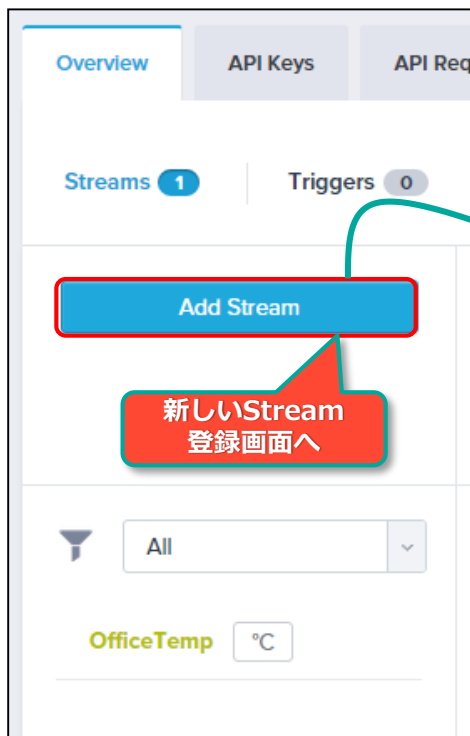
デバイス名登録 (日本語可)

非公開：個人利用

公開：共有利用

※ deviceIDは、英数文字のキーワードとして自動的に設定されます。

5. ストリーム (StreamID) の作成登録



※ StreamIDは、入力した名前が設定されます

6. デバイスIDとスキームIDの登録

The screenshot displays the AT&T M2X developer console interface. The main content area shows the details for a device named "OfficeTemp&Light".

- Device Name:** OfficeTemp&Light (indicated by a red box and a callout bubble labeled "デバイス名").
- Device ID:** A long alphanumeric string (indicated by a red box and a callout bubble labeled "<deviceID>").
- Primary API Key:** A long alphanumeric string (indicated by a red box and a callout bubble labeled "<x-m2x-key>").
- Stream ID:** Temp (indicated by a red box and a callout bubble labeled "<streamID>").

Additional callout bubbles provide context:

- A blue bubble on the left says "デバイス作成画面でデバイス作成" (Device creation on the device creation screen).
- A blue bubble on the right says "デバイス作成登録名称" (Device creation registration name).
- An orange bubble below the stream ID says "Stream表示名 (日本語可)" (Stream display name (Japanese supported)).

The interface also shows tabs for Overview, API Keys, API Request Log, and Trigger Log. The Overview tab is active, showing a summary of Streams (1), Triggers (0), Location (0), and Charts (1). There are buttons for "Add Stream", "Charts View", "Values Log", "Export Data as CSV", "Log a Test Value", "Edit", and "Delete".

7. M2Xへのデータアップの書式要件

M2Xへのセンサ値アップは、\$WPコマンドを使って行います。

\$WPコマンドを使って、以下の書式の例のような URL と body 、それに header を使って、M2Xクラウドにアップする。

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

※ 以下変数の説明

- <deviceID> : デバイスID
- <streamID> : ストリームID
- <x-m2x-key> : M2Xキー
- <val> : データアップするセンサ値
- <date-time> : 日時（文字列） 例 “2015-09-20T23:55:36\$+09:00”（\$+は特殊文字）

※ 日時は、日本時間を登録（ただしM2Xでの表示は、グリニッジ標準時となる）

8. サンプルプログラム①

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
const unsigned long baudrate = 38400;
```

url,header,body
の設定

```
#define LIMITTIME 35000 // ms (3G module start time)
```

```
String url = "http://api-m2x.att.com/v2/devices/<deviceID>/updates/ ";
String header = "¥"X-M2X-KEY:<x-m2x-key> $r$nContent-Type:application/json$r$n¥"";
String body = "¥"{"¥$¥"values¥$¥" : {"¥$¥"<streamID>¥$¥" : [{" ¥$¥"timestamp¥$¥" : ¥$¥"}] }¥"";
```

```
//=====
```

```
void setup() {
  Serial.begin(baudrate);
  Serial.println(">Ready. ¥r¥n Initilaizing...");
  if( _3Gsetup() ) {
    Serial.println("start");
  } else {
    Serial.println(" Connect Error ... Stop");
    while(1);
  }
}
```

setup
3G初期化

温度センサ値を3分間隔
空けてM2Xにアップ

```
void loop () {
  String dtime = datetime(); // Serial.println(dtime); //debug
  float temp = analogRead(A1)*0.488 - 60.0; // TABshield temp sensor
  if( _3G_WP("$WP " + url + body + dtime + "¥$¥", ¥$¥"value¥$¥" : ¥$¥"" + String(temp) + "¥$¥"}] }¥" " + header)){
    Serial.println("Data Update complete:" + Serial3g.readStringUntil('\n')); }
  else Serial.println("Data Update false...");
  delay(180000); //waiting 3min
}
```

8. サンプルプログラム②

```
//===== 3G setup =====
boolean _3Gsetup() {
  pinMode(7,OUTPUT);
  digitalWrite(7,LOW); delay(1000);
  digitalWrite(7,HIGH); delay(100);
  Serial3g.begin(baudrate);
  //----- 3G module begin & connect -----
  String str;
  unsigned long tim = millis();
  do{ while(!Serial3g.isListening());
    str=Serial3g.readStringUntil('\n');
  }while(!(str.indexOf("3GIM")>0) && (millis() - tim) <LIMITTIME);
  if( millis() -tim >= LIMITTIME) {
    return false;
  } else return true;
}

//===== $WP command =====
boolean _3G_WP(String command) {
  Serial.println(command); // debug
  Serial3g.println(command);
  String rstr;
  unsigned long tim = millis(); // time set(ms)
  do{ while(!Serial3g.isListening());
    rstr=Serial3g.readStringUntil('\n');
    Serial.println(rstr); //debug print....
  }while(!(rstr.indexOf("$WP=")==0) && (millis() - tim) <LIMITTIME);// $WP return check
  return (rstr.indexOf("$WP=")==0);
}

// Get Date & Time (3GIM command)
// return --> string "2015-12-23T01:23:45%2B09:00"
String datetime() {
  Serial3g.println("$YT");
  while(!Serial3g.available());
  String dtime = Serial3g.readStringUntil('\n');
  dtime.replace(" ", "T"); dtime.replace("/", "-");
  return(dtime.substring(7) + "+09:00");
}
}
```

3Gシールド用 (電源ON)

電源On状態から3GIM文字が返却されるまで待機

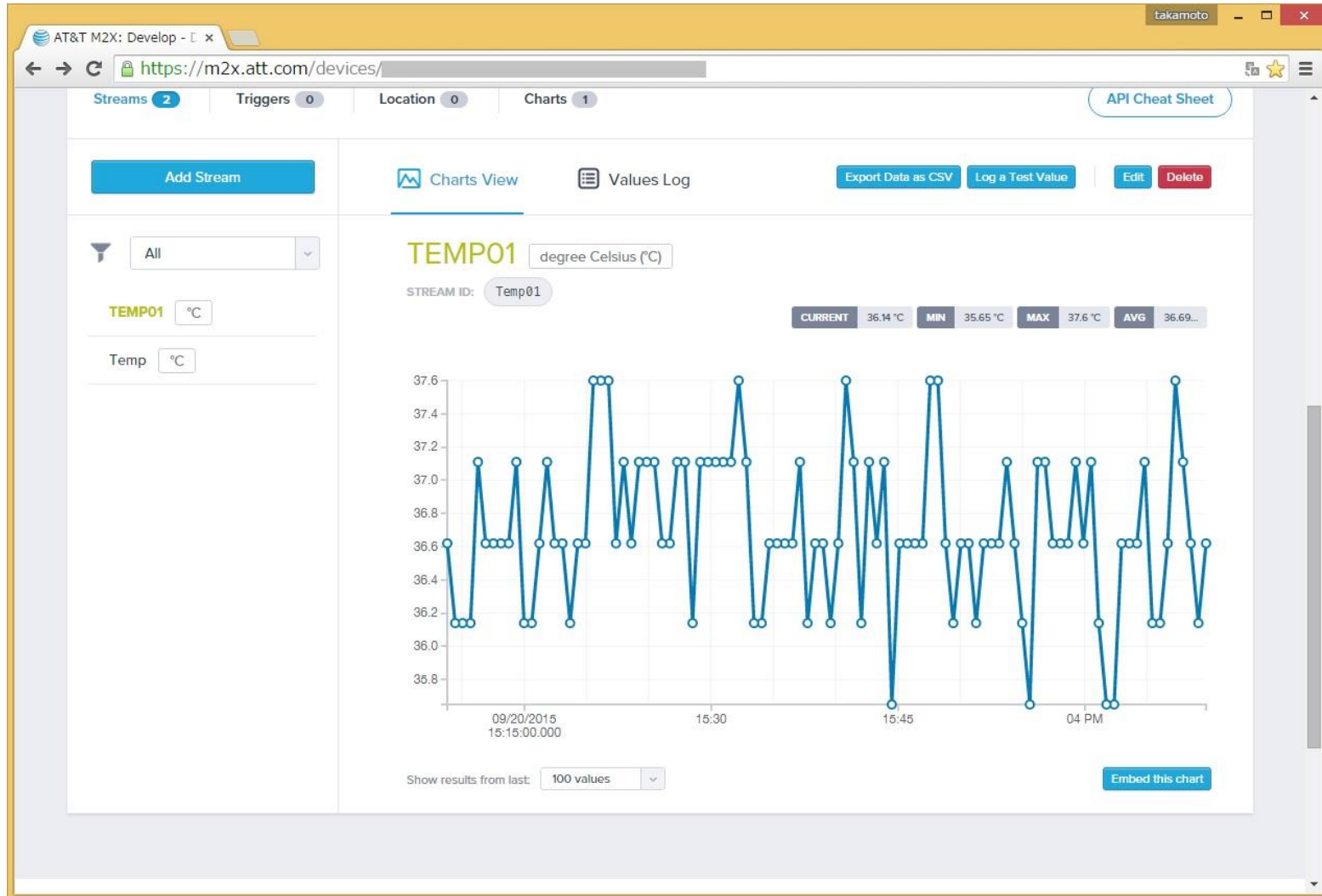
3G接続状態返却

3G POST処理

\$WPコマンド返却処理

\$YTによる時間取得設定機能

9. M2Xにデータアップした事例



10. M2Xからトリガーでツイートする方法

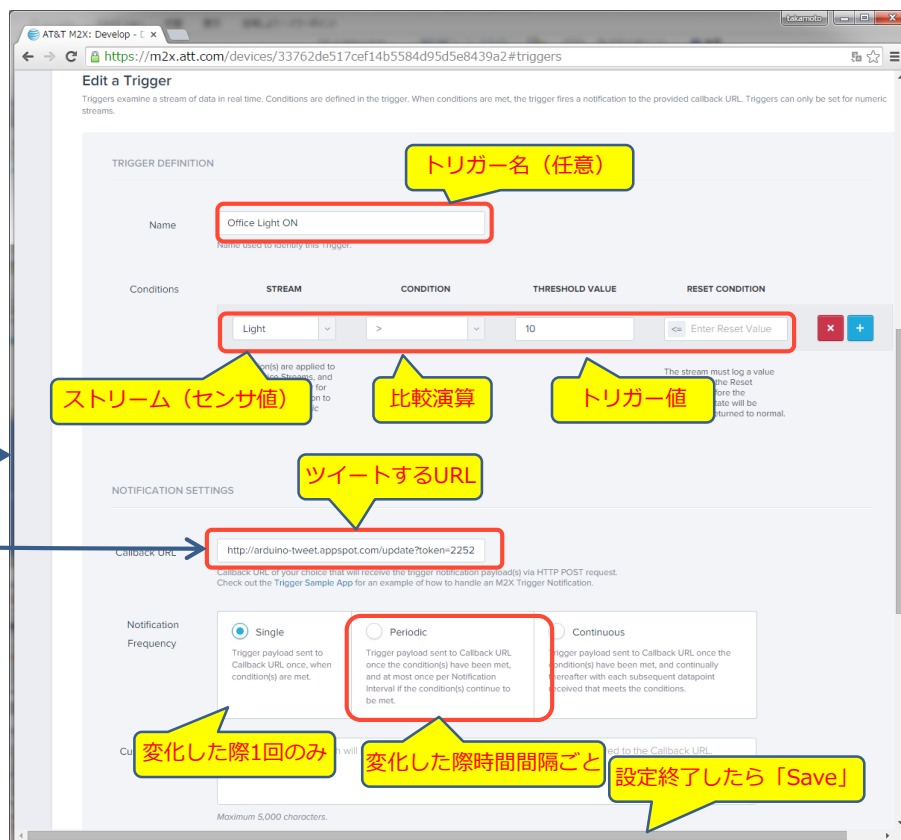
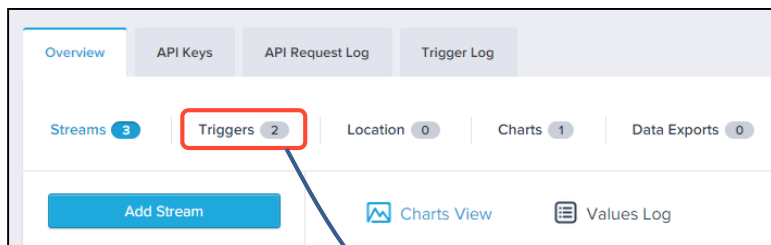
M2Xにアップしているセンサの値をトリガーにして、ツイートする方法を紹介

ツイートするURLは、以下の通り

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

トリガーの設定

トリガー設定の選択

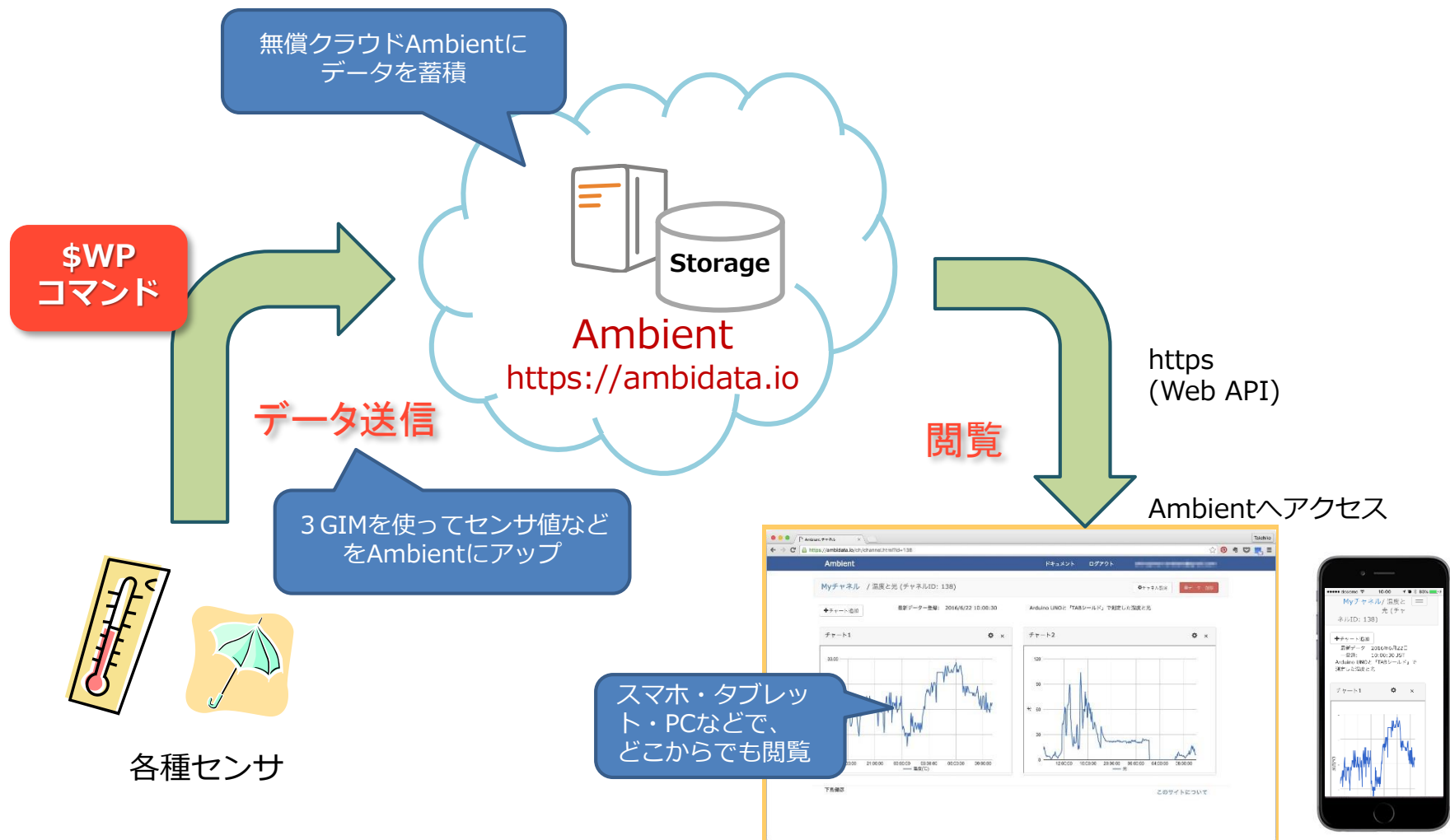


トリガーは、M2Xに送られてきたセンサ値の変化を捉え、アクションを起こすものです。
 この場合、センサ値がある値より大きいか、小さいかで、URLを起動します。
 ここでは、センサ値を見て、ツイッターにツイートするものです。



5. 3GIMでのクラウド連携使用例③ (Ambient & Arduinoの事例)

1. Ambientの利用イメージ



2. Ambientの利用手順

AmbientはArduino、mbedなどのオープンソースハードウェアから利用できるIoT用のクラウドサービスです。無料で使え、センサデータの蓄積・グラフ表示などができるようになっています。

チャンネル生成後、データを送信すると、煩雑な設定をしなくても自動的にデータがグラフ表示されます。ユーザ登録からグラフ表示までが非常に簡単に行えることが特長の一つです。一方でグラフ種類、2軸表示、表示件数設定などグラフのカスタマイズも強力に行えます。

日本で開発されたサービスで、ユーザインタフェースだけでなく、チュートリアルや事例などもすべて日本語で用意されているのも安心です。

ここでは、本3GIMとセンサなどを使い、このAmbientにデータをアップしていくサンプルをご紹介します。

詳細な規約等は、Ambient関連の公開情報等をご参照ください。

① ユーザの登録

② チャンネルの作成

③ データ送信

④ データ確認

3. Ambientのユーザ登録

ユーザ登録へ

メールアドレス (ユーザID) およびパスワード

確認メール

https://ambidata.io/ にアクセス

4. チャネル作成

The first screenshot shows the 'Myチャンネル' page with a 'チャンネルを作る' button highlighted. A red callout box points to this button with the text: **チャンネル生成 ボタンクリック**.

The second screenshot shows the same page after a channel has been created. The 'チャンネルID' and 'ライトキー' fields in the table are highlighted. A red callout box points to these fields with the text: **チャンネルが生成され、チャンネルID、ライトキーが設定される**.

| チャンネル名 | チャンネルID | リードキー | ライトキー | 作成日 | 削除 |
|----------|---------|-------|-------|-----------|----|
| チャンネル148 | 148 | | | 2016/6/22 | |

5. Ambient へのデータ送信ライブラリ

Ambientへのセンサ値アップには、ライブラリが用意されています。

```
Ambient::begin(チャンネルID, ライトキー, SoftwareSerial *s);
```

初期化関数。チャンネルID, ライトキー, ソフトウェアシリアルへのポインターを指定します。

```
Ambient::set(フィールド, データ);
```

データをパケットにセットする関数。1~8のフィールド番号とデータを指定します。
データは予め文字列に変換しておきます。

```
Ambient::send();
```

データをAmbientに送信する関数。

6. サンプルプログラム①

```

#include <SoftwareSerial.h>
#include "Ambient3GIM.h"

SoftwareSerial iemSerial(4,5);
const unsigned long baudrate = 38400;

#define LIMITTIME 35000 // ms (3G module start time)

unsigned int channelId = 100;
const char* writeKey = "...writeKey...";
Ambient ambient;

//===== 3G setup =====
boolean _3Gsetup() {
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH); delay(100);
  digitalWrite(7,LOW); delay(100); // 3G shield --> digitalWrite(7,HIGH);
  //----- 3G module begin & connect -----
  String str="";
  unsigned long tim = millis();
  do{ str=iemSerial.readStringUntil('\n');
  }while(!(str.indexOf("3GIM")>0) && (millis() - tim) <LIMITTIME);
  delay(1000);
  Serial.println(str);
  if( millis() -tim >= LIMITTIME) {
    return false;
  } else return true;
}

```

3 GIM用Ambientライブラリ
ヘッダーファイル

チャンネルID、ライトキー定義
Ambientオブジェクト

3Gシールド用
(電源ON)

電源On状態から3GIM文字
が返却されるまで待機

3 G接続状態返却

6. サンプルプログラム②

```
void setup() {  
  Serial.begin(baudrate);  
  iemSerial.begin(baudrate);  
  Serial.println(">Ready. ¥r¥n Initilaizing...");  
  if( _3Gsetup() ) {  
    Serial.println("start");  
  } else {  
    Serial.println(" Connect Error ... Stop");  
    while(1) { digitalWrite(7,LOW); delay(500); digitalWrite(7,HIGH); delay(500); }; // エラーアラーム  
  }  
  ambient.begin(channelId, writeKey, &iemSerial);  
  delay(1000);  
}  
  
void loop () {  
  char tempbuf[8], lightbuf[8];  
  float temp = 207.26 - 0.594 * analogRead(A1);  
  int light = analogRead(A0);  
  
  sprintf(tempbuf, "%2d.%1d", (int)temp, (int)(temp*10)%10);  
  sprintf(lightbuf,"%3d", light);  
  Serial.print("temp: "); Serial.print(tempbuf);  
  Serial.print(", light: "); Serial.println(lightbuf);  
  
  ambient.set(1, tempbuf);  
  ambient.set(2, lightbuf);  
  ambient.send();  
  
  delay(300000UL);  
}
```

setup
3G初期化

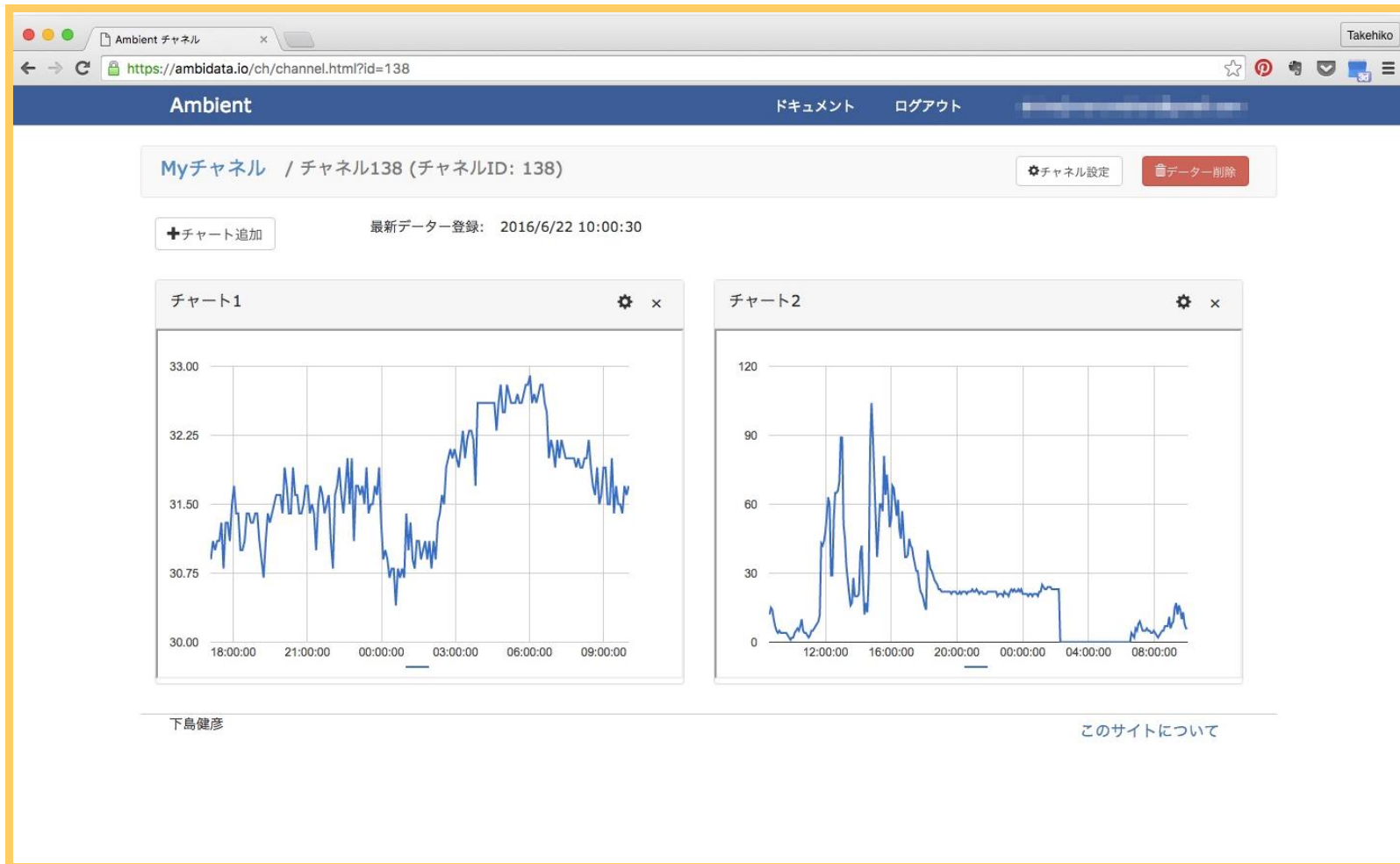
Ambient初期化

温度、明るさセンサー値を読み込み

データを文字列に変換

データをバケットにセットし、送信

7. Ambientにデータアップした事例



8. チャンネルとグラフをカスタマイズした事例





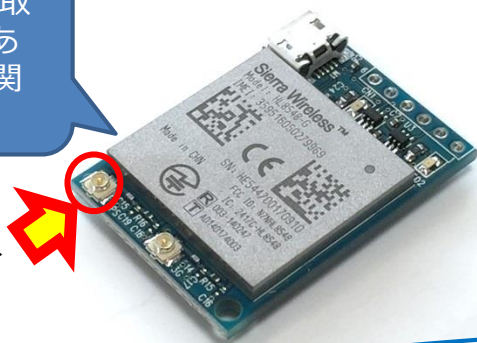
6. GPS機能を使ってGoogleマップに表示

1. GPSによる位置情報取得

GPSを使って位置情報の取得を行い、その情報をインターネット上のサーバにアップし、さらにGoogleマップ上にプロットすることを行ってみましょう。

GPSによる位置情報の初回取得には3分以上掛る場合があります。（3G通信とは無関係に位置情報取得可能※）

GPSアンテナコネクタ部分
(GPSアンテナは別売です)

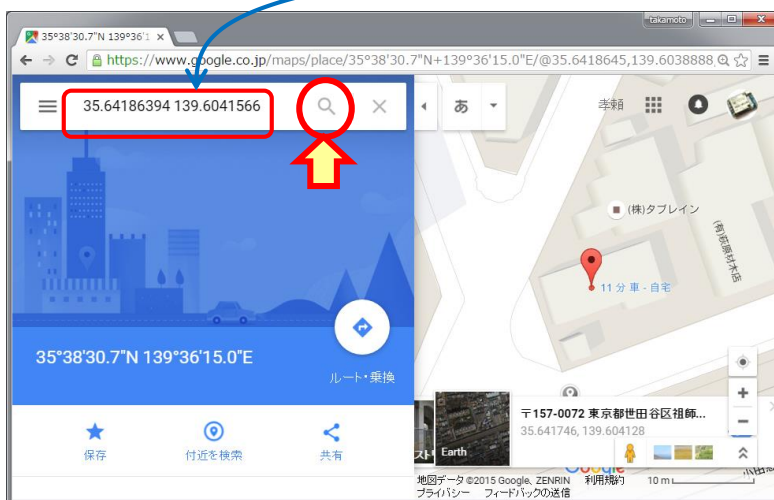


参考：P.72のmonitor3gim.ino を使って、Arduino+3 GIMでのマニュアル操作で、GPSによる位置情報を取得してきましょう。

■ GPS機能による位置情報取得

P.72参照
monitor 3 gim.ino
で稼動した場合

```
Ready.
Welcome to 3GIM(v2)
$LG MSBASED
$LG=OK 35.64186394 139.6041566
```

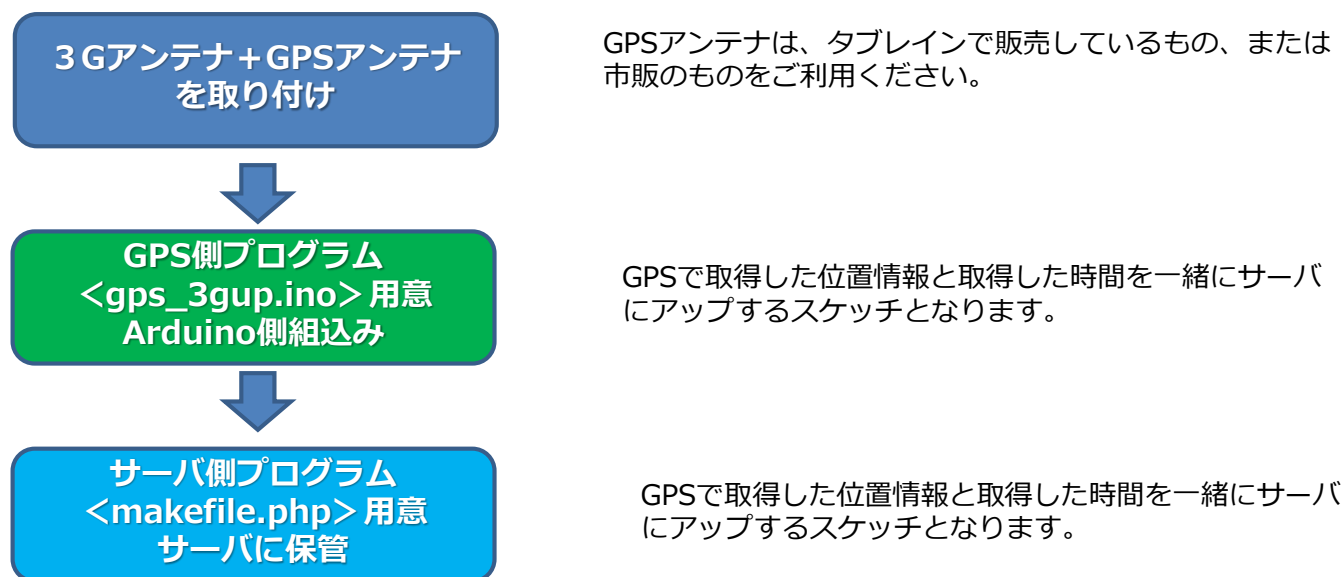


取得した緯度・経度をGoogleマップにプロットしてきましょう。

注意：初回のGPS取得では、数分から3分以上時間が掛りますが、2回目以降でのGPS取得は短時間になります。

2. GPS機能による位置情報群をサーバにアップ

GPS機能を使って、位置情報を定期的にアップするプログラムを紹介しましょう。



3. GPS機能による位置情報アッププログラム

クライアント側プログラム
<gps_3gup.ino①>

```
#include <SoftwareSerial.h>
SoftwareSerial Serial3g(4,5);
const unsigned long baudrate = 38400;

#define LIMITTIME 35000 // ms (3G module start time)

String URL = " http://<サーバ・アドレス>/makefile.php?file=";
String fname;
String dtime;
String gps;

// ----- setup -----
void setup() {
  Serial.begin(baudrate);
  Serial.println(">Ready. ¥r¥n Initilaizing...");
  delay(10);
  while( !_3G_setup() ) {
    Serial.println(" Connect Error ... Stop");
  }
  Serial.println("Connected");
  Serial3g.println("$YL 1"); delay(100);
  dtime = datetimes(); // Serial.println(dtime); //debug
  fname=dtime.substring(2,10) + ".dat";
  Serial.println("Fname = " + fname);
  delay(1000);
}
// ----- loop -----
void loop () {
  unsigned long tim = millis();
  gps = GPSget();gps.replace(" ","");
  Serial.println("gps= " + gps);
  dtime = datetimes(); // Serial.println(dtime); //debug
  Serial.println("dtime= " + dtime); delay(1);
  //----- make file -----
  if(gps!="") {
    Serial3g.println("$YL 1");
    _3G_WGP("$WG " + URL + fname + "&data=" + dtime + "G" + gps + "¥r¥n");
    while(millis()-tim<60000);
  } else if(Serial3g.println("$YL 0")); delay(100);
}
```

1

クライアント側プログラム
<gps_3gup.ino②>

```
// =====
// Get Date & Time (3GIM command)
// return --> string "2015-12-23T01:23:45"
String datetimes() {
  String dtim;
  do {
    Serial3g.println("$YT");
    while(!Serial3g.available());
    dtim = Serial3g.readStringUntil('¥n');
    delay(10);
  } while ( !(dtim.indexOf("201") ==7) );
  dtim.replace(" ", "T"); dtim.replace("/", "-");
  return(dtim.substring(7));
}
//===== 3G setup =====
boolean _3G_setup() {
  String str;
  uint32_t tim = millis();
  do {
    while (!Serial3g.available()) {
      if (millis() - tim > 40000) return false;
    }
    str = Serial3g.readStringUntil('¥n');
  } while (str.indexOf("Hello") != 0);
  return true;
}
//===== $WG command =====
boolean _3G_WGP(String command) {
  delay(10);
  Serial.println(command); // debug
  delay(10);
  Serial3g.println(command);
  String rstr;
  unsigned long tim = millis(); // time set(ms)
  do{ while(!Serial3g.isListening());
    rstr=Serial3g.readStringUntil('¥n');
    Serial.println(rstr); //debug print...
  }while(!(rstr.indexOf("$W")==0) && (millis() - tim) <LIMITTIME);
  return (rstr.indexOf("$W")==0);
}
```

2

ここではAssisted GPSは利用していません。
利用する場合には、P.50を参照ください

4. プログラム続き & 実行サンプル

```
//=====
String GPSget() {
  delay(10);
  Serial.println("GPS getting...");
  Serial3g.println("$LG MSBASED");
  String rstr;
  unsigned long tim = millis(); // time set(ms)
  do{ while(!Serial3g.isListening());
    rstr=Serial3g.readStringUntil('\n');
    if(rstr.length()>0) Serial.println(rstr); //debug print....
    if(rstr.indexOf("$LG=NG 508")==0) Serial3g.println("$LG MSBASED");
  }while(!(rstr.indexOf("$LG=OK")==0) && (millis() - tim) < 180000); // waite 3min check
  if(rstr.indexOf("$LG=OK")==0) return(rstr.substring(7));
  else return ""; // time over.
}
```

クライアント側プログラム
<gps_3gup.ino③>

3

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

サーバ側プログラム
<makefile.php>

```
15-12-11.dat - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
2015-12-11T10:57:54G35.40226758,136.8549192
2015-12-11T10:58:54G35.40297568,136.8683088
2015-12-11T10:59:54G35.40328145,136.8709803
2015-12-11T11:00:54G35.40277719,136.8699342
2015-12-11T11:01:51G35.40390909,136.8704277
2015-12-11T11:02:53G35.40413439,136.8703204
2015-12-11T11:03:51G35.40412366,136.8704921
2015-12-11T11:04:54G35.4041183,136.8705136
2015-12-11T11:05:52G35.4049176,136.8705082
2015-12-11T11:06:54G35.40461183,136.8691885
2015-12-11T11:07:55G35.40567935,136.8706745
2015-12-11T11:08:55G35.40658057,136.8705511
2015-12-11T11:10:21G35.40785193,136.8696606
2015-12-11T11:11:16G35.40821135,136.8699503
2015-12-11T11:11:58G35.40704191,136.8703312
2015-12-11T11:12:53G35.40768027,136.8706155
2015-12-11T11:13:56G35.40754616,136.8723482
2015-12-11T11:14:54G35.40780902,136.872654
2015-12-11T11:15:56G35.40932178,136.8715972
2015-12-11T11:16:58G35.40894628,136.8712807
2015-12-11T11:17:56G35.40853858,136.8748534
2015-12-11T11:18:54G35.40846884,136.8752772
2015-12-11T11:19:58G35.40835619,136.8753737
2015-12-11T11:20:55G35.40914476,136.8758029
2015-12-11T11:21:58G35.40941834,136.8756795
2015-12-11T11:22:55G35.4092896,136.8769455
2015-12-11T11:23:59G35.40927351,136.8770099
2015-12-11T11:24:56G35.4093647,136.8779486
2015-12-11T11:26:00G35.4091984,136.8771976
2015-12-11T11:27:04G35.40960073,136.8781042
2015-12-11T11:28:01G35.4098475,136.8780506
```

Google MAPS API
を使って出図



もくじ

1. a3gimライブラリとは
2. a3gimライブラリ関数群
3. コントロール関連関数
4. ショートメッセージ関連関数
5. Web関連関数
6. 現在位置取得（GPS）関連関数
7. 通信その他機能関数
8. TCP/IP関連関数
9. プロファイル関連ほか関数



第5章 Arduino用a3gimライブラリ群



1. a3gimライブラリとは

1. a3gimとは

a3gimとは、Arduinoやその互換上の拡張ボード上で、3GIM V2.1を簡単に利用できるようにした関数群となります。(参照図)

これらのライブラリ群は、上述したさまざまな「\$コマンド」の利用・設定を一括した関数群で、これらが中学生でも通信技術が利用できるものとしています。

具体的には、Arduino IDE上で利用できる3GIM関連のライブラリ関数群となります。

このライブラリ群は、次のページに記載しているように、ネット上からダウンロードし、Arduino IDE上にコピーして利用できるように環境設定します。

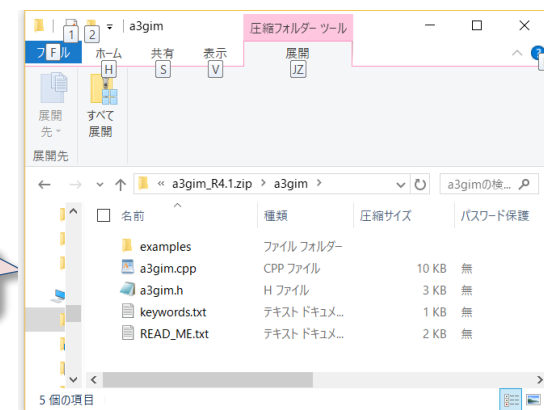
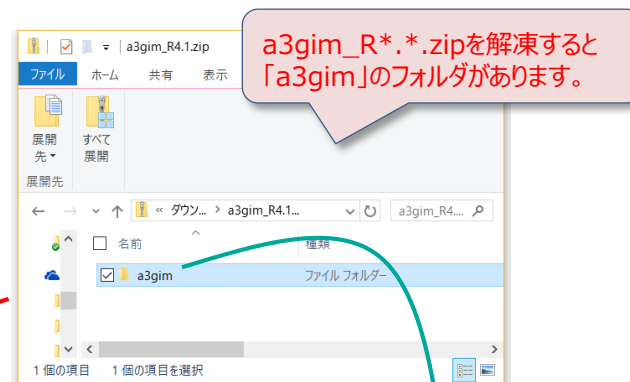
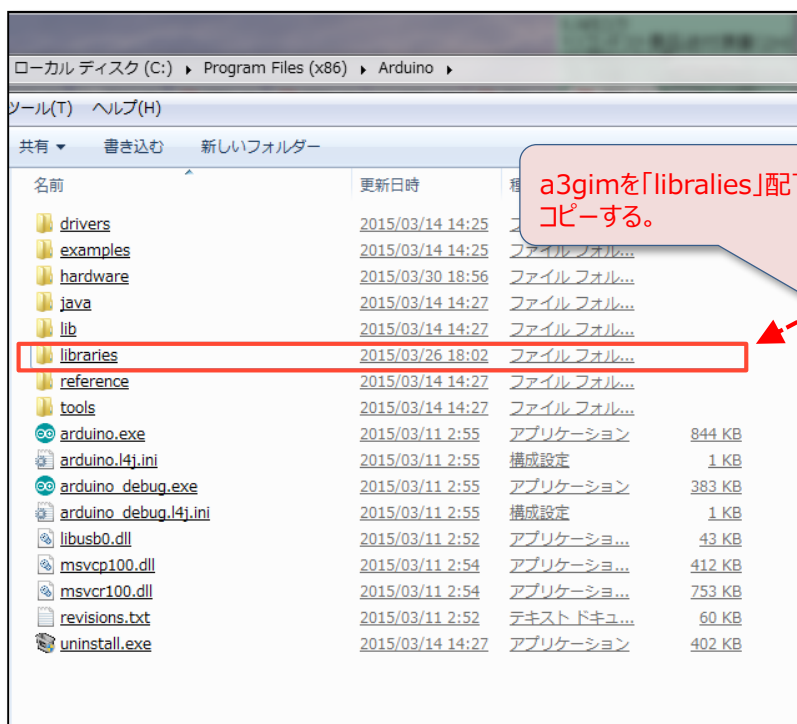
2. ライブラリ群a3gim.zipのダウンロード

このZIPファイルを、Arduino IDE環境下の「`..\libraries`」配下にコピーしてください。コピーした後、Arduino IDEを起動すると、メニュー「ファイル」⇒「スケッチの例」に、「a3gim」が表示されます。

ダウンロード先：

Arduino UNO/Pro用：http://a3gs.wiki.fc2.com/ref/a3gim_R4.1.zip

Arduino Mega/Due/101用：http://a3gs.wiki.fc2.com/ref/a3gim2_R4.2.zip



3. a3gimライブラリを利用する方法

概要

- ▶ 提供するa3gimライブラリ機能は、3Gシールドの提供ライブラリとほぼ同等です。
- ▶ そのため、Arduinoと3GIMとの接続を工夫することで、3Gシールド用の下記のライブラリを使用することができます：
 - ▶ a3gim UNO/Pro用（SoftwreSerialを使用）：3 GIM専用に改訂
 - ▶ a3gim2 Mega/Due/101用（Serial1を使用）：3 GIM専用に改訂
 - ▶ a3gs UNO/Pro用（SoftwreSerialを使用）：3 Gシールド専用
 - ▶ a3gs2 Mega/Due/Leonardo用（Serial3gまたは3を使用）

互いのライブラリの違い

- ▶ ヘッダファイル(デフォルトのボーレートの違い)
 - ▶ a3gim.hのシンボル a3gsBAUDRATE の定義は、「9600」となっています。
 - ▶ a3gim2.hのシンボル a3gsBAUDRATEの定義は、「9600」となっています。
 - ▶ a3gs.hのシンボル a3gsBAUDRATE の定義は、以前「4800」で、最新では「9600」としています。
 - ▶ a3gs2.hのシンボル a3gsBAUDRATE の定義は、「57600」となっています。

【注意】3 GIMの出荷時は、9600bpsとしています。

3GIMとArduinoとの接続方法

- ▶ UNOの場合
 - #6をGND、#4を5V、#3をD4、#2をD5、#1を開放(何も接続しない：常時電源ON)、に接続する
- ▶ **Mega/Dueの場合（ハードウェアシリアル通信利用）**
 - #6をGND、#4を5V、#3をRX3、#2をTX3、#1を開放(何も接続しない：常時電源ON)、に接続する
- ▶ **Leonardoの場合（ハードウェアシリアル通信利用）**
 - #6をGND、#4を5V、#3をRX1、#2をTX1、#1を開放(何も接続しない：常時電源ON)、に接続する
- ▶ **Genuino101の場合（ソフトウェアシリアル通信利用）**
 - #6をGND、#4、#5をV3.3に、#2をD0、#3をD1、#1を開発（何も接続しない：常時電源ON）、に接続する。

【補足】ハードウェアシリアル利用のため高速設定可能

【補足】3 GIMのボーレートを57600bpsまで高速設定可能
※環境が良ければ115200bpsも可能



2. a3gimライブラリ関数群

1. ライブラリを利用するハードウェア

- ▶ Arduinoから3GIMを簡単に利用できるようにするために、ライブラリa3gimが提供されています。
 - ▶ a3gimライブラリは、Arduino UNO/Pro等でソフトウェアシリアルで3GIMを利用する際に使用します。
 - ▶ a3gim2ライブラリは、Arduino Mega/Due/Leonardo/101/Zero等でハードウェアシリアル3GIMを利用する際に使用します。提供される機能は、a3gimと同等です。
 - ▶ Geunino101 は、一部
 - ▶ 以下では、a3gim/a3gim2を総称してa3gimと呼びます。
- ▶ このライブラリを利用することで、関数の呼び出しという形で3GIMの各機能を利用することができます。
- ▶ 本ライブラリ群は、タブレイン製の「IoTABシールド」または「3GIMシールド」を利用し、Arduino UNO、Genuino101、ArduinoMEGAなどの上で稼働できます。



IoTABシールドV3.0



IoTABシールドV2.1

2. ライブラリ概要

▶ a3gimライブラリを使用する方法

- ▶ ライブラリは、スケッチの中で以下の順序でコントロール用のメソッド（関数）を呼び出すことにより利用できます。

```
#include "a3gim.h"

void setup()
{
  ...
  if (a3gs.start() == 0) {
    // 3Gシールドの電源ONに失敗した
  }
  if (a3gs.begin() == 0) {
    // ライブラリの初期化に失敗した
  }
  // 3Gシールドが使用できるようになった
  ...
}
```

a3gim.h を宣言

a3gimの状態を検査

※ このようなスケッチでも可能

```
while( a3gs.start() == 0 && a3gs.begin() ==0 );
```

3. ライブラリ機能一覧 ライブラリが提供する関数(1/3)

| 分類 | メソッド名※ ¹ | 機能概要 | 補足 |
|--------------------------|---------------------|------------------|-------------|
| コントロール (Control) | getStatus※ | 3Gシールドの状態取得 | |
| | begin※ | ライブラリの初期化 | |
| | end※ | ライブラリの終了 | |
| | restart※ | 3Gシールドのリセット | |
| | start※ | 3Gシールドの電源ON | |
| | shutdown※ | 3Gシールドの電源OFF | |
| | getIMEI | IMEI-IDの取得 | |
| | setLED | LED1のON/OFF | |
| | setBaudrate | UARTの通信速度の設定 | 初期は9600bps |
| | setAirplaneMode | エアプレーンモードのON/OFF | |
| ショート メッセージ (SMS) * | getResult | 通信結果を取得 | ※追加 |
| | sendSMS※ | SMSの送信 | ※仕様変更 |
| | availableSMS※ | SMSの受信状態チェック | |
| | readSMS※ | SMSの読出し | |
| | onSMSReceived | SMS着信時のコールバック設定 | V4.0では何もしない |

※ Arduino GSM/GPRSシールド用ライブラリと互換性がある関数です。

* 利用するSIMカードによって使えない場合があります。

3. ライブラリ機能一覧 ライブラリが提供する関数(2/3)

| 分類 | メソッド名 | 機能概要 | 補足 |
|-----------------|---------------------|-----------------|----------------|
| Web機能 | httpGET * | GETメソッドの要求 | http/httpsを利用可 |
| | httpPOST | POSTメソッドの要求 | 同上 |
| | tweet * | Twitterへの投稿 | * |
| 現在位置取得 (GSP) | getLocation | 現在位置の取得 | 緯度経度情報 |
| | getLocation2 | 現在位置の取得 2 | 緯度経度他情報 ※追加 |
| 通信機能その他 | getServices | 利用可能サービスの取得 | |
| | getRSSI | 電波強度の取得 | |
| | getTime | 現在時刻の取得 | 日付・時刻形式 |
| | getTime2 | 現在時刻の取得 | 通算秒形式 |
| | getVersion | 3Gシールドのバージョンの取得 | |

※ Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

* 無償サービス「<http://arduino-tweet.appspot.com/>」を利用（要登録）

3. ライブラリ機能一覧 ライブラリが提供する関数(3/3)

| 分類 | メソッド名 | 機能概要 | 補足 |
|----------|--------------------------|-----------------|-------------|
| TCP/IP機能 | connectTCP ※ | TCPコネクションを接続 | |
| | disconnectTCP ※ | TCPコネクションを切断 | |
| | getStatusTCP | TCPコネクション最新状況取得 | ※追加 |
| | writeBegin | シリアル通信で直接書込み | ※追加 |
| | read ※ | データの読込み | 2つのバリエーション有 |
| | write ※ | データの書出し | 3つのバリエーション有 |
| プロファイル | setDefaultProfile | デフォルトプロファイルを設定 | ※仕様変更 |
| | getDefaultProfile | デフォルトプロファイルを取得 | ※仕様変更 |
| ATコマンド | enterAT | ATコマンドパススルーモード | ※追加 |

※ Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

4. ライブラリ定数一覧 ライブラリが定義している主な定数

- 各ライブラリを利用する上で、留意すべき定数（主に最大値の定義）を下表に示す。これらは、ヘッダファイル“a3gim.h”で定義されています。

| 分類 | 定数名 | 意味 | 設定 | 補足 |
|--------|------------------------|----------------------------|------|----|
| SMS | a3gimMAX_SMS_LENGTH | SMSメッセージの最大バイト数 | 120 | |
| | a3gimMAX_MSN_LENGTH | 電話番号の最大バイト数(最大桁数) | 11 | |
| Web | a3gimMAX_URL_LENGTH | URLの最大バイト数 | 256 | ※1 |
| | a3gimMAX_HEADER_LENGTH | POSTのヘッダの最大バイト数 | 512 | ※1 |
| | a3gimMAX_BODY_LENGTH | POSTのボディの最大バイト数 | 1024 | ※1 |
| | a3gimMAX_RESULT_LENGTH | GET/POSTのレスポンスの取得可能な最大バイト数 | 192 | ※1 |
| | a3gimMAX_TWEET_LENGTH | ツイートメッセージの最大バイト数 | 60 | ※1 |
| TCP/IP | a3gimMAX_HOST_LENGTH | ホスト名の最大バイト数 | 96 | ※1 |
| | a3gimMAX_DATA_LENGTH | 一度に読み書きできるデータの最大バイト数 | 1024 | ※2 |

※1 これらの定数は、ATmega328*/32U*（Unoなど）を利用したArduinoではSRAMのサイズが小さいことからかなり制約が厳しい。ATmega2560/1280（Megaなど）またはADKを利用することで、これらの最大値を大きくすることができる。

※2 大きなデータを読み書きする場合は、複数回に分けてread/writeを実行する。



3. コントロール関連関数

コントロール関連の関数

提供関数ライブラリ

| | | | |
|---------------------|------------------------|------------------|------------|
| コントロール (Control) | getStatus ※ | 3Gシールドの状態取得 | |
| | begin ※ | ライブラリの初期化 | |
| | end ※ | ライブラリの終了 | |
| | restart ※ | 3Gシールドのリセット | |
| | start ※ | 3Gシールドの電源ON | |
| | shutdown ※ | 3Gシールドの電源OFF | |
| | getIMEI | IMEI-IDの取得 | |
| | setLED | LED1のON/OFF | |
| | setBaudrate | UARTの通信速度の設定 | 初期は9600bps |
| | setAirplaneMode | エアプレーンモードのON/OFF | |
| | getResult | 通信結果を取得 | ※追加 |

※ Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

概要

- ▶ a3gimライブラリの初期化・終了、ライブラリの状態の取得、3GIMシールドのリセット、電源ON/OFF、IMEIの取得、LED1のON/OFF、UARTの通信速度の設定を行う

留意点

- ▶ 電源ONには、15秒程度の時間が掛かる。リセットには、5秒程の時間が掛かる。
- ▶ 電源OFFには、1秒ほどの時間が掛かる
- ▶ **setBaudrate**による通信速度の変更には、十分留意すること

1. コントロール関連の関数 `getStatus`※

| int getStatus(void) | |
|---------------------|---|
| 機能概要 | 現在のライブラリの状態を取得 |
| 引数 | なし |
| 戻り値 | 現在の状態 (ERROR, IDLE, READY, TCPCONNECTEDCLIENT のいずれか) |
| 補足 | 各状態の意味は下記の通り： a3gim::ERROR : エラーが発生した a3gim::IDLE : 空いている (機能の実行が可能) a3gim::READY : 同上 a3gim::TCPCONNECTEDCLIENT : TCPコネクションが接続した |

【利用例】

```
int status;  
status = a3gs.getStatus();  
Serial.print("Status is ");  
switch (status) {  
  case a3gim::ERROR :    Serial.println("ERROR");    break;  
  case a3gim::IDLE :     Serial.println("IDLE");     break;  
  case a3gim::READY :    Serial.println("READY");    break;  
  case a3gim::TCPCONNECTEDCLIENT :  
    Serial.println("TCPCONNECTEDCLIENT");    break;  
  default : Serial.println("Unknown");    break;  
}
```

【出力結果例】

Status is IDLE

2. コントロール関連の関数 `begin`※

- バリエーション1

| int begin(char* pin) | |
|----------------------|--|
| 機能概要 | ライブラリを初期化する |
| 引数 | pin : 未使用(指定は不要) |
| 戻り値 | 0 : 正常に初期化を実行できた時 |
| | 1 : エラーが発生した時 (ライブラリは使用不可) |
| | 2 : IEM上のgw3gアプリのバージョンが古い (ライブラリは使用不可) |
| 補足 | 3Gシールドの電源がONの状態、本ライブラリの使用に先立って一度だけ本関数を呼び出す必要がある。 初期化に失敗した場合は、end()関数を呼び出して、時間をおいて何度かリトライすることで成功する場合がある。 終了関数end()を呼び出した後は、再度、本関数を呼び出すことができる。 本関数の中では、デフォルトの通信速度で標準ライブラリ「SoftwareSerial」を初期化(begin)する。 |

【使い方の例】

```
if (a3gs.begin() == 0)
  Serial.println("Succeeded.");
```

3. コントロール関連の関数 `begin`※

- バリエーション2

| int begin(char* pin, uint32_t baudrate) | |
|---|--|
| 機能概要 | ライブラリを初期化する |
| 引数 | pin : 未使用(指定は不要) |
| | baudrate : 設定する通信速度 (1200/2400/4800/9600/19200/38400/57600/115200) |
| 戻り値 | 0 : 正常に初期化を実行できた時 |
| | 1 : エラーが発生した時 (ライブラリは使用不可) |
| | 2 : IEM上のgw3gアプリのバージョンが古い (ライブラリは使用不可) |
| 補足 | <p>3Gシールドの電源がONの状態、本ライブラリの使用に先立って一度だけ本関数を呼び出す必要がある。</p> <p>初期化に失敗した場合は、end()関数を呼び出して、時間をおいて何度かリトライすることで成功する場合がある。</p> <p>終了関数end()を呼び出した後は、再度、本関数を呼び出すことができる。</p> <p>本関数の中では、指定された通信速度で標準ライブラリ「SoftwareSerial」を初期化(begin)する。</p> <p>通信速度の変更は、setBaudrate()関数を使って事前に行っておく必要がある。</p> |

【使い方の例】

```
if (a3gs.begin(0, 9600) == 0)
  Serial.println("Succeeded.");
```

4. コントロール関連の関数 `end`※

| int end(void) | |
|---------------|---|
| 機能概要 | ライブラリの使用を終了する |
| 引数 | なし |
| 戻り値 | 0 : 正常に終了処理を実行できた時 |
| | 0以外 : エラーが発生した時 |
| 補足 | 本関数の中では、標準ライブラリであるSoftwareSerialを終了(end)する。 |

【使い方の例】
・次頁参照

5. コントロール関連の関数 restart※

| int restart(char* pin) | |
|------------------------|--|
| 機能概要 | 3Gシールドを再起動（リセット）する |
| 引数 | pin : 未使用(指定は不要) |
| 戻り値 | 0 : 正常にリセットを実行できた時 |
| | 0以外 : エラーが発生した時（リセットできない時） |
| 補足 | IEM全体をリセットする。 通常、本関数を呼び出してから10秒程度でIEMはリセット処理を開始し、40秒程度で利用可能な状態となる。 本関数によるリセット後に再度ライブラリを利用する場合は、一旦、終了関数end()を呼び出した後に、初期化関数begin()を呼び出すこと。 |

【使い方の例】

```
if (a3gs.restart() == 0) {  
    Serial.println("Restarting..");  
    a3gs.end();  
    if (a3gs.begin() == 0)  
        Serial.println("I'm OK.");  
}  
else  
    Serial.println("Restart Failed.");
```

6. コントロール関連の関数 start※

| int start(char* pin) | |
|----------------------|--|
| 機能概要 | 3Gシールドの電源をONにする |
| 引数 | pin : 未使用(指定は不要) |
| 戻り値 | 0 : 正常に電源ONを実行できた時 0以外 : エラーが発生した時 (電源ONできない時) |
| 補足 | 本関数を呼び出しには、40秒程度掛かる (本関数の呼び出しが完了した時点で、3Gシールドが利用可能な状態となっている)。その後、初期化関数begin()を呼び出すことで本ライブラリを利用することができる。 |

【使い方の例】

```
if (a3gs.start() == 0 && a3gs.begin()) {  
    Serial.println("Succeeded.");  
    // 成功処理  
}  
else  
    Serial.println("Restart Failed.");
```

7. コントロール関連の関数 shutdown※

| int shutdown(void) | |
|--------------------|---|
| 機能概要 | 3Gシールドの電源をOFFにする |
| 引数 | なし |
| 戻り値 | 0 : 正常に電源OFFを実行できた時 |
| | 0以外 : エラーが発生した時（電源OFFできない時） |
| 補足 | 本関数を呼び出しには、15秒程度掛かる 本関数を呼び出した後は、再度、電源ON関数start()を呼び出すことで3Gシールドを利用することができる。 |

【使い方の例】

```
a3gs.end();  
a3gs.shutdown();
```

8. コントロール関連の関数 getIMEI

| int getIMEI(char* imei) | |
|-------------------------|---|
| 機能概要 | 3Gシールドに装着されているIEMのIMEIを取得する |
| 引数 | imei : 取得したIMEI (サイズは a3gimIMEI_SIZE バイト) |
| 戻り値 | 0 : 正常に取得できた時 |
| | 0以外 : エラーが発生した時 (取得できない時) |
| 補足 | IMEIとは3G通信モジュール(IEM)の識別IDである (電話番号とは無関係) 引数imeiが指す結果格納場所のスペース (a3gimIMEI_SIZEバイト = 16桁) は、あらかじめ呼び出し側で確保しておくこと。 |

【使い方の例】

```
char imei[a3gimIMEI_SIZE];
if (a3gs.begin() == 0) {
  a3gs.getIMEI(imei);
  Serial.println(imei);
}
```



【出力結果例】

354563020267950

IMEI番号

このIMEIの中に、IEMモジュールに記載されたIDが含まれています。



9. コントロール関連の関数 setLED

| int setLED(boolean sw) | |
|------------------------|---|
| 機能概要 | 3Gシールドに搭載されているLED1を制御する |
| 引数 | sw : ONにする時はTRUE、OFFにする時はFALSEを指定する |
| 戻り値 | 0 : 正常に設定できた時 |
| | 0以外 : エラーが発生した時 |
| 補足 | LED1(緑色のLED)が3Gシールドのどこの位置に配置されているか等は、「取扱説明書」を参照のこと。 |

【使い方の例】

```
if (aFlag) {
  a3gs.setLED(TRUE);
  led1_status = TRUE;
  Serial.println("LED1 is turned on.");
}
```



こちらのLEDが点灯

10. コントロール関連の関数 setBaudrate

| int setBaudrate(int baudrate) | |
|-------------------------------|---|
| 機能概要 | Arduinoと3Gシールドを仲介するUARTの通信速度を設定する |
| 引数 | baudrate : 設定する通信速度(9600/19200/38400/57600/115200のいずれか) |
| 戻り値 | 0 : 正常に変更できた時 |
| | 0以外 : エラーが発生した時 |
| 補足 | <p>本関数の利用には十分留意すること。不適切な値を設定した場合は、3Gシールドを利用することができなくなる可能性がある。</p> <p>工場出荷時の通信速度は、安定動作が可能な 9600(bps) となっている。通信速度をデフォルトの設定値よりも高くするには、ハードウェアシリアルの利用を推奨する。</p> <p>本関数による通信速度の変更は、直ちに有効となる。</p> <p>デフォルトの通信速度と異なる通信速度を設定した場合は、次回の初期化の際には通信速度を指定してbegin()を呼び出すこと（詳細はbegin()の項を参照）</p> |

【使い方の例】

```
if (a3gs.setBaudrate(9600) == 0) {
  Serial.println("Baudrate was changed.");
  Serial.println("Please reset me now.");
}
```

注意 : 設定した通信速度をスケッチ内で呼び出す必要があります。a3gim.h のスケッチ内の「a3gimBAUDRATE」の設定となります。間違った場合には、サンプルスケッチの「check_baudrate.ino」で確認してみてください。

1 1. コントロール関連の関数 setAirplaneMode

| int setAirplaneMode(boolean sw) | |
|---------------------------------|---|
| 機能概要 | 3Gシールドのエアプレーン（機内）モードを制御する |
| 引数 | sw : ONにする時はTRUE (=1)、OFFにする時はFALSE (=0) を指定する |
| 戻り値 | 0 : 正常に設定できた時 |
| | 0以外 : エラーが発生した時 |
| 補足 | エアプレーンモードがONの時は、3G通信は実行できないが、SMSの受信のみ可能である（ただし、受信したSMSの読み出しやSMSの送信はできない） エアプレーンモードをONにすることで、消費電力を大幅に節約することができる。 リセットまたは電源のOFF/ONで、デフォルトの設定(OFF)に戻る。 |

【使い方の例】

```
if (aFlag) {  
    a3gs.setAirplaneMode(true);  
    airplaneMode = true;  
    Serial.println("Airplane mode on");  
}
```



4. ショートメッセージ関連関数

〈注意〉 ショートメッセージは、SIMカードがショートメッセージ対応でないをご利用いただけません。

ショートメッセージ関連の関数

▶ 提供関数ライブラリ

| 分類 | メソッド名※1 | 機能概要 | 補足 |
|------------------------|---------------|------------------|-------------|
| ショート メッセージ (SMS) | sendSMS※ | SMSの送信 | |
| | availableSMS※ | SMSの受信状態チェック | |
| | readSMS※ | SMSの読出し | |
| | onSMSReceived | SMS着信時に呼び出す関数を設定 | V4.0では何もしない |

▶ 概要

- ▶ 3GのSMS（Short Message Service : 120文字程度までの簡易メッセージングサービス）を利用する
- ▶ SMSは、通信キャリアにまたがって送受信できる

▶ 留意すべき点

- ▶ 使用するSIMカード（通信サービス）により、SMSが利用できない場合がある
- ▶ 通信料金（送信側に課金、一般には定額プランの範囲外）に注意すること
- ▶ 通信回線の状態によっては、SMSの配送遅延が起こる場合がある（常に即時配送できるとは限らない）
- ▶ 3Gネットワークとの相性により、何度も同一のSMSを受信してしまう現象が発生する場合がある
- ▶ SMSが届かない場合は、受信側のSMS受信拒否設定にも注意すること
- ▶ SMSのメッセージの中に利用できない文字が存在することに注意すること（例えば、"@"文字等は使えない）

1. ショートメッセージ関連の関数 `sendSMS`※

| <code>int sendSMS (const char* to, const char* msg, int encode)</code> | |
|--|--|
| 機能概要 | SMS（ショートメッセージ）を指定した宛先へ送信する |
| 引数 | to : 送付先の電話番号（ハイフオンなしの10ケタの半角数字） |
| | msg : 送信するメッセージ（最大 <a3gimmax_sms_length文字）< td=""> </a3gimmax_sms_length文字）<> |
| | encode : メッセージに使用するエンコード方法（ <a3gimcs_*で指定）。省略可（省略時はasciiエンコードと解釈）< td=""> </a3gimcs_*で指定）。省略可（省略時はasciiエンコードと解釈）<> |
| 戻り値 | 0 : 正常に送信できた時 |
| | 0以外 : 送信できなかった時 |
| 補足 | 本機能を利用するためには、SMSが利用できる通信サービス(SIMカード)を利用する必要がある。 関数getServices()にて、 a3gimSRV_CS または a3gimSRV_BOTH のいずれかの返却値が取得できる場合にはSMSは利用できる。 3 GIM(V2.1)では、日本語のSMSは取り扱いできない。 |

2. ショートメッセージ関連の関数 availableSMS※

| boolean availableSMS (void) | |
|-----------------------------|----------------------------|
| 機能概要 | SMSが届いているかをチェックする |
| 引数 | なし |
| 戻り値 | true : SMSが届いている時 |
| | false : SMSが届いていない時 |
| 補足 | |

【使い方の例】

```
char msg[a3gimMAX_SMS_LENGTH+1], msn[a3gimMAX_MSN_SIZE+1];
void loop() {
  if (a3gs.availableSMS()) {
    int msgLen = sizeof(msg);
    int msnLen = sizeof(msn);
    a3gs.readSMS(msg, msgLen, msn, msnLen);
    // 受信したSMSの処理
  }
  delay(1000);
}
```

3. ショートメッセージ関連の関数 readSMS※

| boolean readSMS (char* msg, int msglength, char* number, int nlength) | |
|---|---|
| 機能概要 | 受信したSMSを読み出す |
| 引数 | msg : [OUT] 読み出したメッセージ (最大a3gimMAX_SMS_LENGTH文字) |
| | msglength : msgのサイズ (バイト数) |
| | number : [OUT] SMSの送信元の電話番号 |
| | nlength : numberのサイズ (バイト数) (通常、a3gimMAX_MSN_LENGTH文字を確保) |
| 戻り値 | true : 正常に読み出した時 |
| | false : 読み出し不可の時 (SMSを未受信、SMS使用不可 等) |
| 補足 | 正常に読み出せた場合は、msgおよびnumberは'¥0'文字で終端される。 メッセージはASCIIで読み出すことができる。 上記以外は、sendSMS()を参照 |

4. ショートメッセージ関連の関数 onSMSReceived

| int onSMSReceived (void (*handler)(void)) | |
|---|---|
| 機能概要 | SMSが着信した時に呼び出される関数を設定する (V4.0では何もしない) |
| 引数 | handler : 呼び出される関数 (戻り値・引数は無) へのポインタ |
| 戻り値 | 0 : 正常に設定できた時 |
| | 0以外 : 設定できなかった時 |
| 補足 | 本関数は互換性を維持するために残しているものである。実際には何もしない。 |



5. Web関連関数

5. Web関連の関数

▶ 提供関数ライブラリ

| 分類 | メソッド名 | 機能概要 | 補足 |
|-------|----------|-------------|----------------|
| Web機能 | httpGET* | GETメソッドの要求 | http/httpsを利用可 |
| | httpPOST | POSTメソッドの要求 | |
| | tweet* | Twitterへの投稿 | * |

※ Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

* 無償サービス「<http://arduino-tweet.appspot.com/>」を利用（要Twitterの登録）

▶ 概要

- ▶ http/httpsを簡単に利用できる。
- ▶ GET/POSTメソッドを利用できる。
- ▶ Web機能の関数は、すべて同期処理である。そのため、レスポンスが取得できるまで、あるいは通信がタイムアウト(30秒程度)するまで呼び出し元には制御は戻らない。
- ▶ tweetは、サードパーティのフリーサービスを利用することで使用できる（ユーザ登録が必要、利用条件はそのサービスに従う）。
 - ▶ 詳細は <http://arduino-tweet.appspot.com/> （3Gシールドアライアンスとは関係のないサービス）

▶ 留意点

- ▶ 使用するSIMカードで、3Gパケット通信が利用できること
- ▶ 通信料金（http/https通信の利用は、通常、定額プランの範囲内）に留意すること
- ▶ 日本語の取り扱いにはご注意ください。リクエストを送る相手サーバにより、日本語の文字コードが決まりますが、Arduinoでは日本語の処理を簡単に記述することができません。英語のみを取り扱うことを推奨します。

1. Web関連の関数 httpGET ※

```
int httpGET (const char* server, uint16_t port, const char* path, char* result, int resultlength, boolean ssled=false, const char* header=NULL)
```

| | |
|------|--|
| 機能概要 | 指定したサーバ、ポート、パスに対して、httpまたはhttps/GETリクエストを発行して、そのレスポンスを返却する |
| 引数 | server : サーバのドメイン名またはIPアドレス |
| | port : サーバのポート番号 (通常は80を指定) |
| | path : URLのパス |
| | result : [OUT] レスポンスの格納先 (スペースは呼び出し側で確保) |
| | resultlength : resultのサイズを指定 |
| | ssled : httpsを利用する場合はtrue、httpを利用する場合はfalseを指定 (省略可能で、省略時はfalseと同じ) |
| 戻り値 | 0 : 正常にGETできた時 |
| | 0以外 : GETできなかった時 |
| 補足 | <p>本関数の実行時間は、状況によって最大30秒程度掛かる。 サーバからのレスポンスのサイズが大きい場合は、resultlength以降のデータは破棄される。 resultは、'¥0'文字で終端される。文字コードは、接続先のサーバに依存する。 また、レスポンスにはヘッダは含まれない (ボディ部分のみ)</p> |

1. Web関連の関数 httpGET ※

▶ 補足

- ▶ 特にヘッダの指定がない場合は、リクエストのヘッダは下記の通りである：

Host: **server:port**

- ▶ ヘッダを指定する場合は、下記のように指定する。2つ以上のヘッダを指定する場合は、それらの間を「`rn`」で区切り、末尾も「`rn`」で終わる（終端の改行は省略可能）。

```
char *header = "Authorization: Basic QWxhZGRpbjpvYVUyIHNoeXQ= $r$nX-Myheder: XYZ$r$n"
```

- ▶ path引数にはクエリー文字列を含めることができるが、事前にURLエンコードを施しておく必要がある。また、httpGET()関数では'\$'文字を特殊扱いするため、'\$'文字を文字列に含める時は、httpPOST()で解説しているようなエスケープシーケンスに従うこと。

2. Web関連の関数 httpPOST

```
int httpPOST (const char* server, uint16_t port, const char* path, const char* header, const char* body, char* result, int* resultlength, boolean ssled=false)
```

| | |
|------|---|
| 機能概要 | 指定したサーバ、ポート、パスに対して、httpまたはhttps/POSTリクエストを発行して、そのレスポンスを返却する |
| 引数 | server : サーバのドメイン名またはIPアドレス |
| | port : サーバのポート番号 (通常は80を指定) |
| | path : URLのパス |
| | header : HTTPのヘッダ文字列(最大a3gimMAX_HEADER_LENGTHバイト) |
| | body : HTTPのボディ文字列(最大a3gimMAX_BODY_LENGTHバイト) |
| | result : [OUT] レスポンスの格納先 (スペースは呼び出し側で確保) |
| | resultlength : [IN/OUT] resultのサイズを指定、呼び出し結果のサイズが返却 |
| 戻り値 | 0 : 正常にPOSTできた時 |
| | 0以外 : POSTできなかった時 |

次ページ
へ続く

2. Web関連の関数 httpPOST

前ページ
から続く

```
int httpPOST (const char* server, uint16_t port, const char* path, const char* header, const char* body, char* result, int* resultlength)
```

補足

引数headerでは、HostとContent-Lengthの指定は不要である。
引数bodyでは、最後の空行は指定不要である。
本関数の実行時間は、状況によって最大30秒程度掛かる。
サーバからのレスポンスのサイズが大きい場合は、resultlength以降のデータは破棄される。
resultは'¥0'文字で終端される。文字コードは、接続先のサーバに依存する。
レスポンスには、ヘッダは含まれない（ボディ部分のみ）
引数headerおよびbodyでは、下記の'\$'文字を使ったエスケープシーケンスをサポートする。直接、制御文字を指定することはできないので注意すること：
\$t : TAB(0x09)
\$r : CR(0x0d)
\$n : NL(0x0a)
\$" : "そのもの
\$\$: \$そのもの
\$xhh または \$Xhh : 16進数hh(スケッチでの文字列における"0xhh"と同義)
引数headerやbodyでは、バイナリデータをそのまま取り扱うことはできない。また、レスポンスは文字列として返却するため、'¥0'文字を含むことはできない。バイナリデータを透過的に扱った通信を行いたい場合は、TCPIP関数を利用する。

3. Web関連の関数 tweet ※

| int tweet (const char* token, const char* msg) | |
|--|--|
| 機能概要 | Twitterへ投稿する |
| 引数 | token : アクセスに必要なトークン (認証情報) |
| | msg : 投稿するメッセージ (最大a3gimMAX_TWEET_LENGTHバイト) |
| 戻り値 | 0 : 正常に投稿できた時 |
| | 0以外 : 投稿できなかった時 |
| 補足 | <p>tweetは、下記のフリーサービスを利用することで使用できる。ユーザ登録が必要で、利用条件はこのサービスに従う。 詳細は http://arduino-tweet.appspot.com/ を参照のこと。</p> <p>【注意】 上記サービスの制限により、同一メッセージを連続して投稿することはできない。また、一定時間内に投稿できるメッセージ数に制限がある(2012/10時点の制限では、1分間に1回の投稿まで)。この制限を守らない場合は、正しく引数を指定している場合でも本関数はエラーを返却する。</p> |



6. 現在位置取得 (GPS) 関連関数

現在位置取得（GPS）関連の関数

提供関数ライブラリ

| 分類 | メソッド名 | 機能概要 | 補足 |
|---------------|--------------|-----------|----------|
| 現在位置取得（GPS）機能 | getLocation | 現在位置の取得 | 内蔵GPSを使用 |
| | getLocation2 | 現在位置の取得 2 | |

概要

- ▶ HL8548-G内蔵GPSや3Gネットワークを利用して位置を測位
- ▶ 引数の指定により、下記のいずれかの測位方法を選択できる：
 - ▶ **MPBASED**
 - GPSを利用して現在位置を測位する。GPSが利用できない場合は、3Gネットワークを利用する。
 - ▶ **MPASSISTED**
 - 3Gネットワーク上のロケーションサーバを利用して現在位置を測位する。
 - ▶ **MPSTANDALONE**
 - GPSのみを利用して現在位置を測位する。

留意点

- ▶ 通信サービス（例えば、IIJmio等）によっては、3Gネットワーク上のロケーションサーバを利用することができない。その場合は、GPS単独の測位のみが利用できる。
- ▶ 測位方法としてa3gimMPSTANDALONEを指定した場合は、通信料金（通常、定額プランの範囲内だが、SIMカードの通信サービスによる）が発生する
- ▶ 屋内や都心等のように、上空にある衛星の電波がGPSアンテナで補足できない場所では、正しく測位できない場合がある。

1. 現在位置取得 (GPS) 関連の関数 getLocation

| int getLocation(int method, char* latitude, char* longitude) | |
|--|--|
| 機能概要 | 現在位置を取得する |
| 引数 | method : 測位方法 (a3gimMPBASED / a3gimMPASSISTED / a3gimMPSTANDALONE のいずれか) を指定 |
| | latitude : [OUT] 緯度(北緯) dd.dddddd形式、ただし桁数は場合により可変 |
| | longitude : [OUT] 経度(東経) ddd.dddddd形式、同上 |
| 戻り値 | 0 : 正常に取得できた時 |
| | 0以外 : 取得できなかった時 (latitude, longitudeの値は不定) |
| 補足 | 本関数の実行には、数十秒～3分程度の時間が掛かる。 AGPSサーバは、3 GIM(V2.1)ではGoogleのサーバを利用する (今後変更となる可能性がある) 本関数は、同期処理である。そのため、測位処理が完了するまで、あるいは測位が失敗するまで呼び出し元には制御は戻らない。 |

2. 現在位置取得 (GPS) 関連の関数 getLocation2

```
int getLocation2(char* latitude, char* longitude, char *height, char *utc, int *quality, int *number)
```

| | |
|------|---|
| 機能概要 | 現在位置を取得する |
| 引数 | latitude : [OUT] 緯度(北緯 : 度) dd.dddddd形式、ただし桁数は場合により可変 |
| | longitude : [OUT] 経度(東経 : 度) ddd.dddddd 同上 |
| | height : [OUT] アンテナの海拔高さ (単位 : m) |
| | utc : [OUT] 協定世界時での時刻 (hhmmss) |
| | quality : [OUT] 位置特定品質。0 = 位置特定できない、1 = SPS (標準測位サービス) モード、2 = differential GPS (干渉測位方式) モード |
| | number : [OUT] 使用GPS衛星数 |
| 戻り値 | 0 : 正常に取得できた時 |
| | 0以外 : 取得できなかった時 (latitude, longitudeの値は不定) |
| 補足 | <p>本関数の実行には、数十秒～3分程度の時間が掛かる。</p> <p>AGPSサーバは、3 GIM(V2.1)ではGoogleのサーバを利用する (今後変更となる可能性がある)</p> <p>本関数は、同期処理である。そのため、測位処理が完了するまで、あるいは測位が失敗するまで呼び出し元には制御は戻らない。</p> |



7. 通信その他機能関数

通信その他機能の関数

▶ 提供関数ライブラリ

| 分類 | メソッド名 | 機能概要 | 補足 |
|---------|-------------|-----------------|---------|
| 通信その他機能 | getServices | 利用可能サービスの取得 | |
| | getRSSI | 電波強度の取得 | |
| | getTime | 現在時刻の取得 | 日付・時刻形式 |
| | getTime2 | 現在時刻の取得 | 通算秒形式 |
| | getVersion | 3Gシールドのバージョンの取得 | |

▶ 留意点

- ▶ 時刻は、使用している3Gネットワークを介してインターネット上のサーバから取得する（タイムゾーンや時刻精度は利用するネットワークに依存する）

1. 通信その他機能の関数 `getServices`

| int getServices(int& status) | |
|------------------------------|---|
| 機能概要 | 現在利用できるネットワークサービスを取得する |
| 引数 | status : [OUT] 利用できるネットワークサービス（下記のいずれか） a3gimSRV_NO (= 0) : サービス利用不可 a3gimSRV_PS (= 1) : パケット通信サービスのみ a3gimSRV_CS (= 2) : 音声通信(+SMS)サービスのみ a3gimSRV_BOTH (= 3) : パケット通信、音声通信(SMS)いずれも可 |
| 戻り値 | 0 : 正常に取得できた時 |
| | 0以外 : 取得できなかった時 (statusの値は不定) |
| 補足 | 3GIM(V2.1)では、 a3gimSRV_CS と a3gimSRV_BOTH は返却しない。つまり、SMSが利用できるかどうかは判断できない。 |

【使い方事例】

```
int status;
if (a3gim.getServices(status) == 0)
    Serial.println(status);
```

SIMカードによる提供サービスの違いを取得

2. 通信その他機能の関数 getRSSI

| int getRSSI(int& rssi) | |
|------------------------|---|
| 機能概要 | 3Gの電波強度 (RSSI) を取得する |
| 引数 | rssi : [OUT] 取得した電波強度 (単位はdBm) <範囲 : -1 ~ -113> |
| 戻り値 | 0 : 正常に取得できた時 |
| | 0以外 : 取得できなかった時 (rssiの値は不定) |
| 補足 | 電波強度は必ずマイナス値が返却される。0に近いほど電波強度は強い。 |

【電波受信レベルの測定サンプル】

実際に測定した実績では、3Gアンテナを付けずに測定した場合には「-113dBm」を計測、また電波状態の良いところでは「-68dBm」程度を計測できている。

【使い方事例】

```
int rssi;  
if (a3gim.getRSSI(rssi) == 0)  
    Serial.println(rssi);
```

3. 通信その他機能の関数 getTIME

| int getTime(char* date, char* time) | |
|-------------------------------------|--|
| 機能概要 | 現在の日付・時刻を取得する |
| 引数 | date : [OUT] 取得した日付 ("YYYY-MM-DD"形式) |
| | time : [OUT] 取得した時刻 ("HH:MM:SS"形式) |
| 戻り値 | 0 : 正常に取得できた時 |
| | 0以外 : 取得できなかった時 (date/timeの値は不定) |
| 補足 | 日付・時刻は使用している3Gネットワークを介して日本のサーバから取得するため、精度およびタイムゾーンはネットワークに依存する。(日本国内で利用する場合は、タイムゾーンは日本(JST)となる) 時刻は24h制(固定)である。ただし、自動調整出力となるため、変更は不可。 |

【使い方事例】

【利用例】

```
if (a3gim.getTime(date, time) == 0) {
    Serial.print(date);
    Serial.print(" ");
    Serial.println(time);
}
```



【出力結果例】

2012/12/01 12:10:43

【注意：時刻自動調整機能について】

長く電源を入れていないと、時刻が一旦1980年1月5日16:00:00に戻る。
電源を入れて動かしている間に、一時的に現在時刻より16時間遅れで表示され、さらに、日本時間で表示されるようになる。

4. 通信その他機能の関数 getTIME2

| int getTime2(uint32_t& seconds) | |
|---------------------------------|---|
| 機能概要 | 現在の時刻（1970/1/1からの通算秒：「UNIX時間」とも言う）を取得する |
| 引数 | seconds : [OUT] 取得した通算秒 |
| 戻り値 | 0 : 正常に取得できた時 |
| | 0以外 : 取得できなかった時（date/timeの値は不定） |
| 補足 | 時刻を秒単位で取得できるため、時刻の比較処理等が簡単となる。 日付・時刻の精度およびタイムゾーンについては、getTime()を参照。 秒への換算処理では、閏（うるう）年も考慮している。 2038年※問題が発生する可能性がある。 |

※ **2038年問題**は、ISOの通算秒の定義に1970年1月1日からとしていて、C言語の標準で32ビット符号付intを採用している場合、2038年1月19日3時14分7秒（UTC、以下同様）を過ぎると、この値がオーバーフローし、負と扱われるため、コンピュータが誤動作する可能性があるとする問題。

【詳細はウィキペディア参照】

5. 通信その他機能の関数 getVersion

| int getVersion(char* version) | |
|-------------------------------|---|
| 機能概要 | 3 GIM ファームウェアgw3gアプリのバージョンを取得する |
| 引数 | version : [OUT] 取得したバージョン("9.9"形式) |
| 戻り値 | 0 : 正常に取得できた時 |
| | 0以外 : 取得できなかった時 (versionの値は不定) |
| 補足 | begin()の処理の中で3Gシールドのバージョンと本ライブラリの整合性をチェックしているため、通常、本関数を利用する必要はない。 |



8. TCP/IP関連関数

TCP/IP関連の関数

提供関数ライブラリ

| 分類 | メソッド名 | 機能概要 | 補足 |
|--------|----------------|-----------------|-----------|
| TCP/IP | connectTCP* | TCPコネクションを接続 | |
| | disconnectTCP* | TCPコネクションを切断 | |
| | getStatusTCP | TCPコネクション最新状況取得 | |
| | writeBegin | シリアル通信で直接書込み | |
| | read* | データの読み込み | 3バリエーション有 |
| | write* | データの書出し | 3バリエーション有 |

【注意事項】

* Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

- TCP/IP v4のみサポートする。
- 一度に一つのコネクションだけを利用できる。（Web機能とは独立して使用できる）
- 本機能で提供する関数では、すべて同期的に処理する。そのため、サーバから結果が得られるまで、エラーが発生するまで、あるいは通信がタイムアウトするまで呼び出し元には制御が戻らない。
- 接続や通信では、タイムアウト時間として30秒が設定されている
- readやwriteでエラーが発生した時は、接続(connectTCP)からやり直す必要がある
- TCP/IP機能を利用するためには、利用するSIMカードでパケット通信が利用できる必要がある。また、契約プランによっては通信料金が高額になる場合があるため注意すること
- 利用するSIMカードによっては、使用できるポート番号に制限(80番のみ等)がある場合がある。
- 1バイト単位でのread/writeは、実行効率が悪く、かつ実行速度が遅い。そのため、できるだけ複数バイト単位でread/write関数を呼び出して処理することが望ましい。
- 一部のread/write関数で、バイナリデータを透過的に取り扱うことができる。（Ver2.0以降）
- read関数がノンブロッキング（読み出すべきデータがない場合は、待たずに直ちに呼び出し元へリターンする）
- で動作するように変更した。（Ver2.0以降）

1. TCP/IP機能の関数 connectTCP

| int connectTCP(const char* server, int port) | |
|--|---|
| 機能概要 | 指定したサーバ、ポート番号へ接続して、TCPコネクションを確立する |
| 引数 | server : 接続するサーバのホスト名またはIPアドレス port : 接続するポート番号 |
| 戻り値 | 0 : 正常に接続できた時 0以外 : エラーが発生した時 (戻り値はエラー番号を表す) |
| 補足 | 本機能の処理には、状況によって最大30秒程度掛かる。 serverには、IPv4アドレス("x.x.x.x"形式)またはホスト名を指定することができる。 |

【使い方の例】

```
char *svr = "arduino.cc";
if (a3gim.connectTCP(svr, 80) == 0) {
    // GET Request for google site
    a3gim.write("GET / HTTP/1.0\n");
    a3gim.write("HOST:");
    a3gim.write(svr);
    a3gim.write("\n\n");
    // Get response..
}
else {
    Serial.println("Error: can't connect");
}
```

2. TCP/IP機能の関数 disconnectTCP

| int disconnectTCP(void) | |
|-------------------------|---|
| 機能概要 | 接続しているTCPコネクションを切断する |
| 引数 | なし |
| 戻り値 | 0 : 正常に切断できた時 0以外 : エラーが発生した時 (戻り値はエラー番号を表す) |
| 補足 | 本機能の処理には、状況によって1～数秒程度掛かる。 本ライブラリの終了関数endでは、TCPコネクションは自動的に切断しない。そのため、必要に応じて必ず本関数を呼び出してTCPコネクションを明示的に切断すること。 |

3. TCP/IP機能の関数 getStatusTCP

```
int getStatusTCP(int *status,int *tcpnotif, logn *remainedBytes, long *receivedBytes)
```

| | |
|------|---|
| 機能概要 | 接続しているTCPコネクションを切断する |
| 引数 | status: TCPコネクションのステータス(別表①参照) tcpnotif: TCPコネクションで最後に発生したエラーの内容(別表②参照) remainedBytes: 相手に送信できていないデータのサイズ(バイト数) receivedBytes: 受信したがまだread()していないデータのサイズ(バイト数) |
| 戻り値 | 0 : 正常に切断できた時 0以外 : エラーが発生した時 (戻り値はエラー番号を表す) |
| 補足 | 本機能の処理には、状況によって1～数秒程度掛かる。 本ライブラリの終了関数endでは、TCPコネクションは自動的に切断しない。そのため、必要に応じて必ず本関数を呼び出してTCPコネクションを明示的に切断すること。 |

3. TCP/IP機能の関数 getStatusTCP

表① statusの意味

| status | 意味 |
|--------|--------------------|
| 0 | 未接続 |
| 1 | 接続済み |
| 2 | 接続失敗 |
| 3 | クローズした |
| 4 | 接続中 |
| 5 | アイドルタイム*2のカウント開始 |
| 6 | アイドルタイム*2のカウント取り消し |

*2 TCPコネクションが切断された時からアイドルタイムのカウントが開始される。アイドルタイムが30秒となった時に、3Gネットワークのセッションが解放される。アイドルタイム中にread()やwrite()を行うと、アイドルタイムはいったんリセットされる。

【エラー発生時の対処方法】

何らかのエラーが発生した場合は、通常、TCPコネクションを切断して、再度接続からやり直すことを推奨する。

表② tcpnotifの意味

| tcpnotif | 意味 |
|----------|-----------------------|
| 0 | ネットワークエラー |
| 1 | ソケットエラー |
| 2 | メモリ問題 |
| 3 | DNS問題 |
| 4 | TCPが相手から切断された |
| 5 | TCPコネクションエラー |
| 6 | 一般的なエラー |
| 7 | クライアントからのリクエスト受けエラー*1 |
| 8 | AT+KTCPSNDで文字待ちが発生*1 |
| 9 | セッションIDがおかしい*1 |
| 10 | セッションは使用中である |
| 11 | すべてのセッションは使用中である*1 |

*1 通常は発生しない内部エラー

4. TCP/IP機能の関数 writeBegin

| int writeBegin(size_t sz) | |
|---------------------------|--|
| 機能概要 | 指定したサイズszのデータをTCPコネクションに対して書き込む準備をする |
| 引数 | sz : データのサイズ(バイト数)、最大 a3gsMAX_TUNNEL_DATA_LENGTH |
| 戻り値 | 0 : 正常に準備ができた時 |
| | -1 : エラーが発生した時 (引数szがおかしい) |
| 補足 | <p>本機能の処理には、状況によって数秒程度掛かる。 バイナリデータを相手へ書き込むための最速の関数である。\$エスケープが不要であり、一度の書き込めるサイズも大きいいため、write()関数よりも高速である。 本関数を呼び出した後は、指定したサイズ分のデータ(バイナリデータもそのままOK)を、シリアルa3gsに対して直接書き込む。正確にszバイト分のデータを書き込む必要がある。szに満たない場合は、30秒のタイムアウト後に、szバイトに満たない不足分のデータとして、3GIMが自動的に0x00バイトを充当する。 通信速度を115,200bpsに設定している場合は、フロー制御を行っていないため1024バイトのデータ送るたびに5ミリ秒以上のディレイを挿入する必要がある。</p> |

【使い方の例】

```
char *svr = "someserver.aaa";
uint8_t data[256];
if (a3gim.connectTCP(svr, 80) == 0) {
    a3gim.writeBegin(sizeof(data));
    for (int i = 0; i < sizeof(data); i++)
        a3gs.write(data[i]);
    int len = sizeof(buf) - 1;
    char buf[20];
    if (a3gs.getResult(buf, &len, 60000) != 0)
        // Error handling ..
}
```

左記の使い方にあるように、writeBegin()を呼んだ後は、シリアルa3gsに対して直接write()やprint()を使って所定のサイズ分のデータを書き込む。
書き込んだ後は、getResult()関数を呼び出して、書き込みの成否をチェックする。

5. TCP/IP機能の関数 read

● バリエーション1 (バイナリデータの読み出し可)

| int read(void) | |
|----------------|---|
| 機能概要 | 現在のTCPコネクションから1バイトのデータを読み出す |
| 引数※ | なし |
| 戻り値 | 0~0xFF : 読み出した1バイトのデータ |
| | -1 : エラーが発生した時 (コネクションがcloseされた、エラーが発生した、タイムアウトした) |
| | -2 : データが読み出せなかった時 (データがない時) |
| 補足 | 本関数は常にノンブロッキングで動作する。そのため、読み出せるデータがない時は直ちに呼び出し元へリターンする。 本関数は、コネクションがcloseされた時やエラーが発生した時は、直ちに呼び出し元へリターンする。 |

【使い方の例】

```
char *svr = "arduino.cc";
if (a3gim.connectTCP(svr, 80) == 0) {
  // GET Request for google site
  a3gim.write("GET / HTTP/1.0$n");
  a3gim.write("HOST:");
  a3gim.write(svr);
  a3gim.write("$n$n");
  handleResponse();
}
else
  Serial.println("Error: can't connect");
```

```
void handleResponse(void) {
  int c;
  while ((c = a3gim.read()) > 0) {
    // 読み出した文字cを処理する
  }
}
```

5. TCP/IP機能の関数 read

- バリエーション2 (テキストデータの読み出しのみ)

| int read(char* result, int resultlength) | |
|--|--|
| 機能概要 | 現在のTCPコネクションから最大resultlengthバイトのデータを読み出す |
| 引数 | result : [OUT] 読み出したデータを格納するバッファアドレス resultlength : 呼び出し側で確保したバッファのサイズ (バイト数) |
| 戻り値 | 1~(resultlength-1) : 正常に読み出した時 (読み出したバイト数を返す) |
| | 0 : データが読み出せなかった時 |
| | 0未満 : エラーが発生した時 (コネクションがcloseされた、エラーが発生した) |
| 補足 | 本関数は常にノンブロッキングで動作する。そのため、読み出せるデータがない時は直ちに呼び出し元へリターンする。 本関数は、コネクションがcloseされた時やエラーが発生した時は、直ちに呼び出し元へリターンする。 resultで返却するデータは、ヌル文字('¥0')で終端させる。 |

【使い方の例】

```
void handleResponse(void) {
    char res[a3gimMAX_RESULT_LENGTH+1];
    int nbytes;
    while(a3gim.read(res,a3gimMAX_RESULT_LENGTH+1) > 0) {
        Serial.print(res);
    }
}
```

5. TCP/IP機能の関数 read

- バリエーション3 (バイナリデータの読み出し可)

| int read(uint8_t* buffer, size_t sz) | |
|--------------------------------------|---|
| 機能概要 | 現在のTCPコネクションから最大szバイトのデータを読み出す |
| 引数 | buffer : [OUT] 読み出したデータを格納するバッファアドレス sz : 呼び出し側で確保したバッファbufferのサイズ (バイト数) |
| 戻り値 | 1~sz : 正常に読み出した時 (読み出したバイト数を返す) |
| | 0 : データが読み出せなかった時 |
| | 0未満 : エラーが発生した時 (コネクションがcloseされた、エラーが発生した) |
| 補足 | <p>本関数は常にノンブロッキングで動作する。そのため、読み出せるデータがない時は直ちに呼び出し元へリターンする。</p> <p>本関数は、コネクションがcloseされた時やエラーが発生した時は、直ちに呼び出し元へリターンする。</p> <p>機能はバリエーション2と同じであるが、バイナリデータを扱う場合は本関数を使用すること。バリエーション2と異なり、読み出したデータをヌル文字('¥0')で終端することはしない。</p> |

6. TCP/IP機能の関数 write

- バリエーション1 (バイナリデータの書き出し可能)

| int write(uint8_t c) | |
|----------------------|---|
| 機能概要 | 現在のTCPコネクションへ1バイトのデータを書き出す |
| 引数 | c : 書き出すデータ |
| 戻り値 | 1 : 正常に書き出した時 |
| | 0未満 : エラーが発生した時 (コネクションがcloseされた、エラーが発生した、タイムアウトした) |
| 補足 | 本関数の処理には、状況によって最大30秒程度掛かる データcとして、制御文字、ヌル文字、あるいは特殊文字(ダブルクォート、\$)等も そのまま指定することができる。 本関数は同期処理であるため、コネクションヘデータを書き出すまで、コネクション がcloseされるまで、エラーが発生するまで、あるいはタイムアウトするまで呼び 出し元に制御は戻らない。 |

6. TCP/IP機能の関数 write

- バリエーション2 (テキストデータの書き出しのみ)

| int write(const char* str) | |
|----------------------------|--|
| 機能概要 | 現在のTCPコネクションへ文字列データを書き出す |
| 引数 | str : 書き出す文字列データ('%0'で終端する文字配列) |
| 戻り値 | 1以上 : 正常に書き出した時 (書き出したバイト数を返す) |
| | 0未満 : エラーが発生した時 (コネクションがcloseされた、エラーが発生した、タイムアウトした) |
| 補足 | <p>本関数の処理には、状況によって最大30秒程度掛かる。 本関数は同期処理であるため、コネクションへデータを書き出すまで、コネクションがcloseされるまで、エラーが発生するまで、あるいはタイムアウトするまで呼び出し元に制御は戻らない。</p> <p>バリエーション1・3と異なり、バイナリデータを取り扱うためのエスケープ処理を実行しないため、これらに比べて高速に実行できる。</p> |

【使い方の例】

```
char *svr = "arduino.cc";
if (a3gim.connectTCP(svr, 80) == 0) {
    // GET Request for google site
    a3gim.write("GET / HTTP/1.0\n");
    a3gim.write("HOST:");
    a3gim.write(svr);
    a3gim.write("\n\n");
    // Get response..
}
```

6. TCP/IP機能の関数 write

- バリエーション3 (バイナリデータの書き出し可能)

| int write(const uint8_t* buffer, size_t sz) | |
|---|--|
| 機能概要 | 現在のTCPコネクションへ指定したバイト数のデータを書き出す |
| 引数 | buffer : 書き出すデータ (バイト配列) sz : データのサイズ (バイト数) |
| 戻り値 | 1以上 : 正常に書き出した時 (書き出したバイト数を返す) 0未満 : エラーが発生した時 (コネクションがcloseされた、エラーが発生した、タイムアウトした) |
| 補足 | バリエーション2ではデータの中にヌル文字('¥0')を取り扱うことができないが、本バリエーションではデータをエスケープ処理するため、データの中に制御文字、ヌル文字、あるいは特殊文字(ダブルクォート、\$)をそのまま含めることができる。本関数の処理には、状況によって最大30秒程度掛かる。本関数は同期処理であるため、コネクションへデータを書き出すまで、コネクションがcloseされるまで、エラーが発生するまで、あるいはタイムアウトするまで呼び出し元に制御は戻らない。 |

【使い方の例】

```
char *svr = "192.168.1.1";
uint8_t binaryData[] = { 0x0, 0x1, 0x2, 0x3, ..};
if (a3gim.connectTCP(svr, 8080) == 0) {
    a3gim.write(binaryData, sizeof(binaryData));
}
```



9. プロファイル関連ほか関数

プロフィール関連の関数

▶ 提供関数ライブラリ

| 分類 | メソッド名 | 機能概要 | 補足 |
|--------|-------------------|----------------|----|
| プロフィール | setDefaultProfile | デフォルトプロフィールを設定 | |
| | getDefaultProfile | デフォルトプロフィールを取得 | |

【注意事項】 プロファイルは、通信サービス事業者が提供するSIMカードを利用するための設定情報である。
詳細は、setDefaultProfile 関数の説明を参照。

1. プロファイルの関数 setDefaultProfile

int setDefaultProfile(const char *apn, const char *user, const char *password)

| | |
|-------------|---|
| 機能概要 | デフォルトのプロファイルを設定する |
| 引数 | apn: SIMカードのAPN情報 user: SIMカードのユーザ名 password : SIMカードのパスワード |
| 戻り値 | 0 : 正常に設定できた時 0以外 : 設定できなかった時 (引数の指定が間違っている等) |
| 補足 | 設定したデフォルトのプロファイル番号は、内臓マイコンのフラッシュROMに記録されるため、電源をOFFにしても維持される (再度、本ライブラリにて設定するまで有効である) 3 GIM V2.1の出荷時の設定プロファイル情報は、下記の通りである : 出荷時デフォルト : IJ様の iijmio サービス(iijmio.jp/mio@ijj/ijj) |

2. プロファイルの関数 getDefaultProfile

int getDefaultProfile(char *apn, char *user, char *password)

機能概要 デフォルトのプロファイル番号を取得する

引数
apn: SIMカードのAPN情報
user: SIMカードのユーザ名
password: SIMカードのパスワード

戻り値
0: 正常に取得できた時
0以外: 取得できなかった時

補足

【使い方の例】

```
int pfNum;
if (a3gim.getDefaultProfile(&pfNum) == 0) {
    Serial.print("Default Profile No is ");
    switch (pfNum == 1)
    case 1 :
        Serial.println("docomo mopera");
    case 2 :
        Serial.println("IIJmio");
    case 3 :
        Serial.println("IIJmobile");
```

```
case 11 :
    Serial.println("bmobile");
case 15 :
    Serial.println("DTI ServerMan");
default :
    Serial.println("unknown profile");
}
```

ATコマンド実行

| int enterAT(unit32_t duration) | |
|--------------------------------|---|
| 機能概要 | ATコマンドパススルーモードに切り替え |
| 引数 | duration : ATコマンドパススルーモードから戻るまでの時間 (単位 : 0.1秒) 例 : 300=30秒 600=1分 = 0 の場合には、ATコマンドパススルーモードに切り替わったまま |
| 戻り値 | 0 : ATコマンドモードに切り替わった場合 |
| | 1 : 引数の指定がおかしい時など (モードは切り替わらない) |
| 補足 | <p>ATコマンドパススルーモードでは、HL8548-Gに対して直接ATコマンドを送信して、そのままの結果を取得することができる。</p> <p>ATコマンドの使用に制限はないため、HL8548-Gの設定を任意に変更することができる。しかし、変更した設定等によっては、gw3gファームウェアの動作に支障をきたす場合があるので、十分留意すること。</p> <p>利用できるATコマンドの詳細は、HL8548-Gの開発元であるSierra Wireless社のサイトで公開されている「AT Commands Interface Guide - AirPrime HL6 and HL8 Series」を参照のこと。</p> <p>※ATコマンドそのものに関しては、Tabrainでは技術的なサポートは致しかねますので、ご了承ください。</p> |

【補足資料】 注意点

- ▶ 本製品で利用している3G通信モジュール（HL8548-G、以下3Gモジュールと呼ぶ）は、付属している3Gアンテナとの組合せで、日本の技適（技術基準適合証明※1）を取得をしています。よって、日本以外の海外での利用や、アンテナの取り換えやケーブルの取り外し等を行った使い方は、電波法違法利用となりますので、絶対行わないでください。
- ▶ 3GアンテナおよびGPSアンテナ、それにそれぞれのケーブルとコネクタは小さく、壊れやすいため、取扱いには、十分注意してください。特に、頻繁な取り外し・取り付けは行わないようお願い致します。（GPSアンテナ関係は別売オプションとなります）
- ▶ Arduinoと3GIMを接続して、電源をONあるいはリセットを掛けた場合、利用できるようになるまで15秒程度の時間が掛かります。
- ▶ 3Gモジュールは瞬間的に消費電力が高くなる場合があります、なるべく外部電源(ACアダプタ)をご利用頂くことをお勧めいたします。詳細は2章を参照ください。
 - ▶ ご利用されるパソコンの特性により、Arduino側へのUSB接続からの電力供給だけでは、3Gシールドが利用できない場合がありますのでご注意ください。動作が不安定となる場合は、外部電源(ACアダプタ)の利用をお勧めします。

※1 技術基準適合証明とは、特定無線設備（総務省令「電波法施行規則」で定める小規模な無線局に使用するための無線設備）が電波法令の技術基準に適合していることを証明（電波法第38条の2）することである。（Wikipediaより）

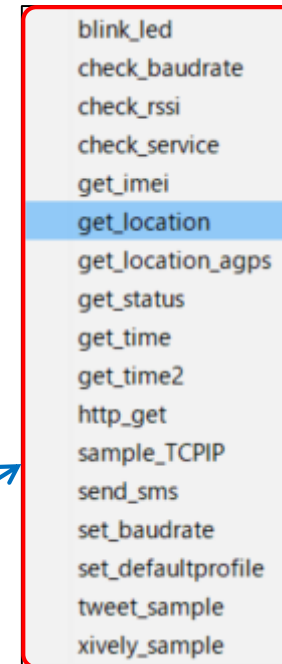


10. サンプルスケッチ群の実行例

1. a3gimサンプルスケッチ一覧

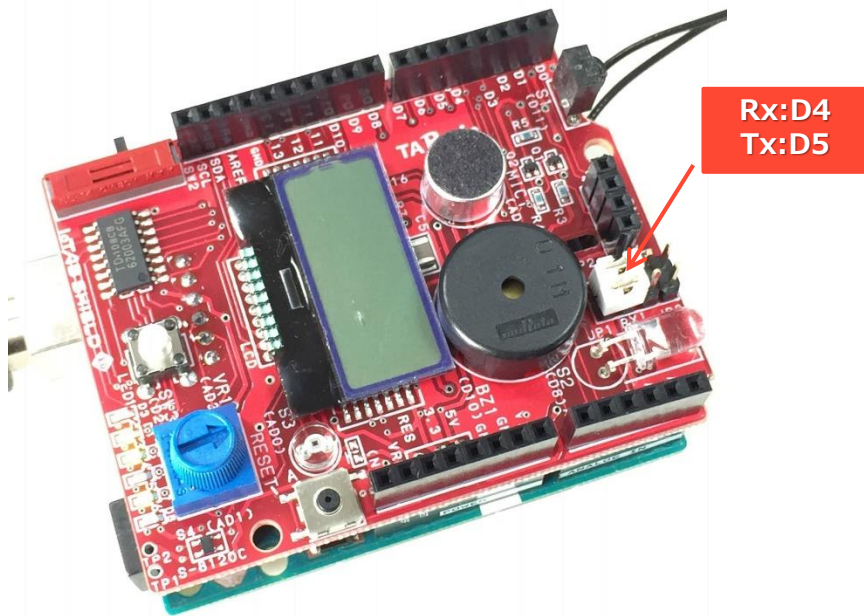


A3gimのサンプルスケッチを動かしてみましよう。
あらかじめ設定したサンプルスケッチは、以下の「スケッチの例」に表示されます。

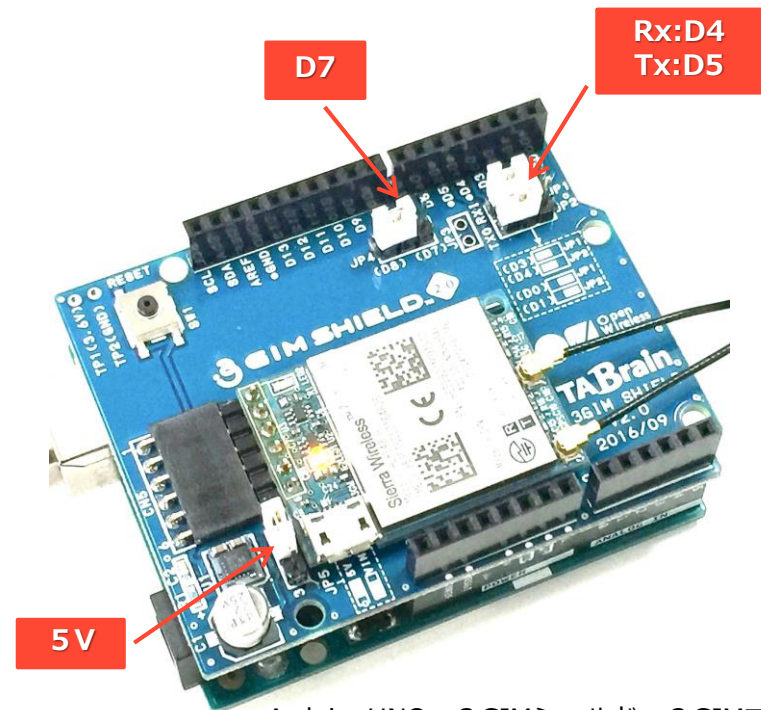


2. 3 GIMシールド・IoTABシールド設定方法

a3gimのサンプルスケッチをArduino IDEで読み込み、ArduinoUNOやGenuino101ほか互換機にコンパイルし、書き込んで、実行してみてください。ここでは、ArduinoUNO+IoTABシールド+ 3 GIM または ArduinoUNO+ 3 GIMシールド+ 3 GIM を使った設定を紹介しておきます。



ArduinoUNO+IoTABシールド+ 3 GIMでは、
① JP1とJP2は、RxとTxのジャンパピンをD4とD5に接続



ArduinoUNO+ 3 GIMシールド+ 3 GIMでは、
① JP1 とJP2のRxとTxのジャンパピンをD4とD5に接続
② JP4 はD7に接続
③ JP5は 5Vに接続

3. サンプルスケッチの読込・書込・実行

- ▶ 前述の環境か、ほぼ同等の状態で、a3gimのサンプルスケッチをArduinoIDEに読み込み、コンパイルし、Arduino+（IoTABまたは3 GIM）シールド+ 3 GIMに書込みます。そのあと実行してみてください。
- ▶ ここでは、「get_imei.ino」を読込・書込・実行を行って試しています。

The image illustrates the process of loading and running a sample sketch in the Arduino IDE. It consists of three main parts:

- File Selection:** The 'Sketch' menu is open, and 'get_imei.ino' is selected from the 'Examples for any board' list.
- Code Editor:** The sketch content is shown in the editor. The line `#define baudrate 9600UL` is highlighted with a red box. The sketch code is as follows:


```

1 // 3GIM(V2) sample sketch -- getIMEI
2
3 #include <SoftwareSerial.h>
4 #include "a3gim.h"
5
6 #define baudrate 9600UL
7 const int powerPin = 7; // 3gim power pin (If no
8
9 void setup()
10 {
11   Serial.begin(baudrate);
12   delay(3000); // Wait for Start Ser
13   Serial.println("Ready.");
14
15   Serial.print("Initializing.. ");

```
- Serial Monitor:** The serial monitor shows the output of the sketch:


```

Ready.
Initializing.. Succeeded.
IMEI: 359516050163312
Shutdown..

```

4. サンプルスケッチの実行結果例

■ check_baudrate.ino (通信ボーレート確認)

```
Ready.
Initializing..
Try baudrate: 9600
Recognize succeeded.
Current baudrate is 9600 bps.
```

■ get_status.ino (機能状態取得)

```
Ready.
Initializing.. Succeeded.
Status is IDLE
Shutdown...
```

■ http_get.ino (httpgetによるサーバデータ取得確認) ※8行目のサーバを tabrain.jp に変更

```
Ready.
Initializing.. Succeeded.
httpGET() requesting.. OK!
[<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=Shift]
Shutdown..
```

■ check_service.ino (SIMカードのサービス情報)

```
Ready.
Initializing.. Succeeded.
Packet Service Only.
Shutdown..
```

■ get_time.ino (現日時取得)

```
Ready.
Initializing.. Succeeded.
2016/10/22 12:47:58
Shutdown..
```

■ sample_tcpip.ino (tcpipテスト確認)

```
Ready.
Initializing.. Succeeded.
www.arduino.org : Page title is "Arduino - Open Source Products for Electronic
Projects"
Shutdown..
Initializing..
```

■ check_rssi.ino (電波強度取得)

```
Ready.
Initializing.. Succeeded.
RSSI = -81 dBm
Shutdown..
```

■ get_time2.ino (現日時取得 2)

```
Ready.
Initializing.. Succeeded.
1477140604 Sec.
Shutdown..
```

■ tweet_sample.ino (ツイート送信確認)

```
Ready.
Initializing.. Succeeded.
tweet() requesting.. OK!
Shutdown..
```

※10行目のtokenを設定し、11行目に文章挿入
(11行目の文章には空白はエラーとなり、「%20」に変更する)

■ get_location.ino (GPS:位置情報・緯度・経度取得)

```
Ready.
Initializing.. Succeeded. It maybe takes several minutes.
OK: 35.641996, 139.604198
Shutdown..
```



もくじ

- 【補足資料1】 3 GIMコマンド・応答一覧表
- 【補足資料2】 5Vから3.3Vを作り出す回路例
- 【補足資料3】 トラブルシューティング
- 【補足資料4】 3 GIM V2.1 外形寸法
- 【補足資料5】 3 GIM サポートサイト
- 【補足資料6】 3 GIM関連商品のご紹介
- 【補足資料7】 3 GIM V1 との相違点
- 【補足資料8】 ラズベリーパイでの3 GIM利用



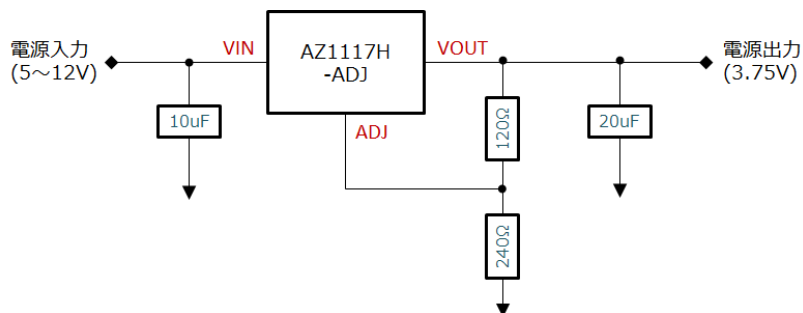
補足資料

【補足資料1】 3GIM V2.1コマンド・応答一覧表

| No | 分類 | 機能 | コマンド送信 | 応答 (レスポンス) 正常受信 | 応答 (レスポンス) エラー受信 |
|----|---------|---------------|-------------------------------|------------------------------|--|
| 1 | System | Version | \$YV¥n | \$YV=OK version¥n | |
| 2 | | RSSI | \$YR¥n | \$YR=OK rssi¥n | \$YR=NG errno¥n |
| 3 | | Service | \$YS¥n | \$YS=OK service¥n | |
| 4 | | IMEI | \$YI¥n | \$YI=OK imei¥n | \$YI=NG errno¥n |
| 5 | | LED | \$YL¥n | \$YL=OK status¥n | \$YL=NG errno¥n |
| 6 | | Baudrate | \$YB [baudrate]¥n | \$YB=OK baudrate¥n | \$YB=NG errno¥n |
| 7 | | Reset | \$YE [level]¥n | \$YE=OK level¥n | \$YE=NG errno¥n |
| 8 | | Time | \$YT¥n | \$YT=OK datetime¥n | \$YP=NG¥n |
| 9 | | Airplane mode | \$YP [mode]¥n | \$YP=OK mode¥n | \$YP=NG errno¥n |
| 10 | | ATcommand | \$YA¥n | OK | |
| 11 | SMS | Send | \$SS msn "message" [encode]¥n | \$SS=OK¥n | \$SS=NG errno ..¥n / \$SS=NG errtype errcode¥n |
| 12 | | Receive | \$SR [index[1]]¥n | \$SR=OK msn "message"¥n | \$SR=NG errno¥n |
| 13 | | Check | \$SC [1]¥n | \$SC=OK rtn¥n | |
| 14 | GPS | GPS | \$LG method [op [rp]]¥n | \$LG=OK latitude longitude¥n | \$LG=NG errno¥n |
| 15 | Web | Get | \$WG url ["header"]¥n | \$WG=OK nbytes¥nresponse¥n | \$WG=NG errno ..¥n |
| 16 | | Post | \$WP url "body" ["header"]¥n | \$WP=OK nbytes¥nresponse¥n | \$WP=NG errno ..¥n |
| 17 | TCP/IP | Read | \$TR maxbytes¥n | \$TR=OK nbytes¥ndata¥n | \$TR=NG errno ..¥n |
| 18 | | Write | \$TW "data"¥n | \$TW=OK nbytes¥n | \$TW=NG errno ..¥n |
| 19 | | Connect | \$TC host_or_ip port¥n | \$TC=OK¥n | \$TC=NG errno ..¥n |
| 20 | | Disconnect | \$TD¥n | \$TD=OK¥n | \$TD=NG errno ..¥n |
| 21 | | Status | \$TS [status]¥n | \$TS=OK status¥n | \$TS=NG errno ..¥n |
| 22 | | Get sockname | \$TN¥n | \$TN=OK ipAddr portNo¥n | \$TN=NG errno ..¥n |
| 23 | | Tunnel Write | \$TT nbyte¥ndata | \$TT=OK nbytes¥n | \$TT=NG errno ..¥n |
| 24 | Profile | Set/Read | \$PS APN user pw¥n | \$PS=OK¥n | \$PS=NG errno¥n |

【補足資料2】 5Vから3.7Vを作り出す回路例

- ◆ 5～12Vの電源(ACアダプタ等)から3GIMが必要とする3.7V電源を作り出す回路の例を以下に示す：



【図】 5Vから3.7Vを出力する電源回路例

- ◆ 必要な部品は下記の通り：

| No | 分類 | パーツ | 数量 | 実売価格(円) | 補足・販売店 |
|----|----------------------------|--------------------------|----|---------|----------------|
| 1 | 3端子レギュレータ | AZ1117H-ADJ | 1個 | 30 | 秋月電子にて10個単位で販売 |
| 2 | 抵抗 | 1/4W抵抗(240Ω) | 1個 | 10 | 秋月電子・千石電商等で販売 |
| 3 | 抵抗 | 1/4W抵抗(120Ω) | 1個 | 10 | 秋月電子・千石電商等で販売 |
| 4 | 積層セラミックコンデンサ | 25V 10μF | 1個 | 80 | 秋月電子・千石電商等で販売 |
| 5 | タンタルコンデンサ(または積層セラミックコンデンサ) | 10V 22μF | 1個 | 42 | 千石電商等で販売 |

【補足資料3】 トラブルシューティング

| # | 課題 | 現象 | 問題点・対応策 | 補足 |
|---|------------|---|--|--|
| 1 | 配線・接続 | <ul style="list-style-type: none"> UART (Tx:送信、Rx:受信)、電源およびGNDが正しく理解できていない | <ul style="list-style-type: none"> 3 GIMコネクタ部の#1~#6までを正しく理解して上で配線・接続のこと #1 (電源On/Off: 任意)、#2 (RX)、#3 (Tx)、#4 (1.8~5V 電源)、#5 (3.3~4.2V電源)、#6 (GND) | <ul style="list-style-type: none"> #5(VCC)で外部電源を利用する場合には、3.7Vリチウムイオン電池を推奨 |
| 2 | 応答 (レスポンス) | <ul style="list-style-type: none"> コマンドを送っても、返信がない 正しい応答でない | <ul style="list-style-type: none"> 通信モジュールとマイコンボードとの通信、またはマイコンボードとPCとの通信において以下の原因が考えられる ① 3GIMの配線が正しくできていない (配線・接続確認) ② 電源供給に問題がある (電源電圧の確認) ③ UART通信速度の設定が間違っている (確認設定) ④ 初期電源後の待ち時間を考慮不足 (15秒以上待機) ⑤ プログラムに間違いがあることで再確認 ⑥ Arduino IDE シリアルモニタ画面の改行コード変更 | <ul style="list-style-type: none"> 応答が正しく表示されない場合の原因は、配線ミスや配線での接触不良が考えられる ②の電源供給で、VCCの3.3~4.2Vを間違えるケースが多発 ⑥の場合、改行選択メニューで「CRおよびLF」を選択のこと |
| 3 | エラー頻発 | <ul style="list-style-type: none"> #=NGが多発 立ち上げタイミングの問題 電源供給 (電流が小さい) 問題 | <ul style="list-style-type: none"> 配線・接続が正しくできていること 適正なSIMカードの挿入されていること 正しく電源供給できていること | <ul style="list-style-type: none"> RSSI (電波強度測定) やSIMカードのサービス確認 \$YRや\$YSコマンドで確認 |
| 4 | 電波強度測定の取得 | <ul style="list-style-type: none"> 電波強度が取得できない | <ul style="list-style-type: none"> 正しいSIMカードとアンテナ接続によって正しく設定される 正しい電波強度を取得するにはしばらく時間が掛る | <ul style="list-style-type: none"> 同上 |
| 5 | SMS送受信 | <ul style="list-style-type: none"> SMSの送受信ができない SMSの応答が無い | <ul style="list-style-type: none"> SIMカードが、SMS対応になっていない (切替え必要) SMSサーバとのやり取りでの不備 (何度か読み込み必要) | <ul style="list-style-type: none"> 同上 |
| 6 | GPS取得 | <ul style="list-style-type: none"> GPS取得ができない GPS取得に時間が掛る | <ul style="list-style-type: none"> GPSアンテナが正しく接続されていること GPS電波状態が良い所 (屋外・PCから離す) で実施のこと 初期立上げでは数分から10分ほど掛る場合がある 電源供給が正しくできていること | <ul style="list-style-type: none"> 一度GPS取得でき、電源が入った状態だと、次からは即取得可能 |
| 7 | ネット接続 | <ul style="list-style-type: none"> Webコマンド群やTCP/IPコマンド群が正しく応答しない | <ul style="list-style-type: none"> 3 Gアンテナを正しく接続する 正しいSIMカードが挿入されていない SIMカードの接続不良 (再度再挿入などを実施) 電源供給が正しくできていること | <ul style="list-style-type: none"> 正しいSIMカードとは、\$ PSコマンドを使ってプロファイル設定されたSIMカードであること。Wikiページで情報公開 |

※その他トラブルがあった場合には、Wikiページにてお問い合わせください。

<http://form1.fc2.com/form/?id=816242>

基本的なことは、これまでWikiページサイトや、本資料等にて掲載していますので、そちらをご覧ください。基本的なことでのお問い合わせは、返答を控えさせていただくことがあります。**\$コマンドのエラーコード一覧表**は、[こちら](#)となります。

【補足資料5】 3 GIM サポートサイト

3 GIMに関する技術情報が盛りだくさん掲載されています。

<http://a3gs.wiki.fc2.com/wiki/3GIMの紹介>

The screenshot shows a web browser window displaying the 3GIM Wiki page. The page title is "3GIMの紹介 - 3Gシールド". The main heading is "3Gシールド/3GIM Wikiページ". The page is divided into sections: "3GIMの紹介" and "3GIM(3G IoT Module)について". The "3GIM(3G IoT Module)について" section has a sub-section "概要" (Overview) which states: "3GIM(スリージム)は、様々なマイコンを使って、簡単にインターネット接続することができるSDカードサイズの超小型3G通信モジュールです。" (3GIM (Sleejim) is a super-small 3G communication module that can be easily connected to the Internet using various microcontrollers on an SD card size.) It also mentions that it can be used with various microcontrollers like mbed, GR-Sakura, PIC, and Raspberry Pi. The "外観" (Appearance) section mentions that the image of the 3GIM (beta version) is uploaded and that the color, layout, and dimensions may change. A sidebar on the right contains a "トップメニュー" (Top Menu) with links to "Open Wireless Alliance", "メニュー" (Menu), "トップ" (Top), "News", "セミナー・イベント" (Seminar/Event), and "本サイト目的" (Purpose of this site).

内容 (もくじ)

■ 3 GIM (3G IoT Module)について

- ・ 概要
- ・ 外観
- ・ 提供する機能
- ・ 3 GIMスペック
- ・ ピン配置

■ 機能一覧(UART経由で利用する場合)

- ・ UARTコマンドインタフェースの概要
- ・ コマンド一覧

■ 5Vから3.7Vを作り出す回路例

■ 利用上の留意点

■ トラブルシューティング

■ ダウンロード

■ 事例

- ・ Intel Edisonで使ってみました
- ・ Arduinoを使ったモノ
- ・ mbedを使ったモノ

■ 3GShield & 3GIM Lab

- ・ Intel Edisonを使った事例
- ・ 3 GIMを使った環境モニタ
- ・ ラズベリーパイで3Gシールドを使ってみました
- ・ Intel Galileoで3Gシールドを使ってみる
- ・ ハウス向け監視モジュールの試作(その2)
- ・ ハウス向け監視モジュールの試作
- ・ GR SAKURAでの利用
- ・ 簡易監視装置の試作
- ・ メール読み上げ機の試作
- ・ 3Gシールドを使ったセンサネットワークの試作

【補足資料6】 3 GIM関連商品のご紹介

3 GIMを**Arduino** UNO や Genuino101 上、それに**RaspberryPi** (B+, 2 B、3 B、Zero)でも簡単に稼働させることができる3GIMシールドや3 GIM HAT、それに12種類センサ類やLED、スピーカなどを搭載したIoTABシールドV3.0を使うことで、誰もが、簡単に、短時間で「IoTデバイス」のモノづくりできるプロトタイピング開発環境が揃います。

(センサ値などをメールで送信し、ツイッター連携、クラウド連携が、容易に学べます)

3 GIM SHIELD™ 2.0

3 GIMシールドは、Arduino UNOや Genuino101、さらにArduino MEGA などの上で簡単に3 GIMを利用することができます。

3 GIMシールドは、電源電圧3.7Vを3 GIMに安定供給し、安心して3 GIMを稼働させることができます。



IoTAB SHIELD 3.0

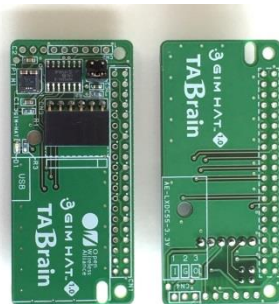
IoTABシールドV3.0は、これまでのセンサ拡張ボードTABシールドに、3 GIMが取り付けられる仕様としました。このことで、IoTデバイスの試作に取り掛かることができ、そのままセンサネットワークとしても利用できるようになります。



3 GIM HAT™ 1.0

3 GIM HATは、Raspberry Pi の多くの種類で稼働する3 GIM専用の拡張ボードで、USBケーブルによるインターネット接続と、UART接続による\$コマンド通信ができるものです。

Raspberry Piを野外のゲートウェイとして利用もでき、アナログセンサ対応ボードとしても利用できます。



【補足資料 7】 3GIM V1.1/V2.0 との相違点

- ▶ ハードウェアの違い (V2.0との違い)
 - ▶ LED点灯を少し暗くしました。
 - ▶ GPSのパルス信号を取り出す半田面を取り付けました。
- ▶ ハードウェアの違い (V1.1との違い)
 - ▶ 通信モジュール (HL8548-G) を搭載しました。(V1は、DTW400Wを搭載)
 - ▶ 基板サイズは、厚さが7mmと僅かに薄くなりました。(V1は、8mm)
 - ▶ ご利用いただける電源電圧を、**3.3~4.4V**としました。(V1は、**3.6~4.4V**)
※Arduino UNOの3.3Vを使った場合には、マイコンボードの固体差もあり、PC側のUSBによる電源電力の差もあり、通信できたり、できなかったりします。
- ▶ ファームウェアの違い (V1.1との違い。V2.0とは同じ)
 - ▶ 電源投入後 (#1をLOW)、緑LEDが点滅した後、初期立ち上げメッセージが表示されます。
 - ▶ 初期立ち上げメッセージは「Welcome to 3GIM(v2)」となりました。(V1は、「Hello, I'm gw3g(Ver 2.0)」)
 - ▶ 初期立ち上げ時間が、ほぼ15秒と短くなりました。(V1では、約25~35秒ほど)
 - ▶ 一部のコマンドでの応答が変わりました。各コマンドの応答値をご覧ください。
 - ▶ 通信ボーレートは、変更されると、リセットする必要なく、そのままご利用頂けます。(V1は、ハードウェアリセットが必要)
つまり¥YBを使ってボーレートを変更したのち、シリアル通信を同じく変更 (Arduinoではbegin関数など) して即利用できます。
 - ▶ 通信ボーレートは、9600・38400・57600・115200bpsとしました (推奨)。(V1は、4800pbs ~57600pbsまで設定可能)
 - ▶ ご利用頂けるSIMカードは、ユーザ設定できるようになりました。(V1は、限定された既存設定SIMカードから選択)
SIMカードのプロファイル (APN情報、ユーザ名、パスワード) 設定・参照は、\$PSコマンドのみでできるようになりました。
 - ▶ GPS機能 (精度も向上、コマンドオプション追加) を拡張しました。(V1は、V2に比べ精度が悪かった)
 - ▶ 一部機能を削減 (ストレージ機能) しました。(V1は、ストレージ機能保有)
 - ▶ 一部機能を追加 (TCP/IP機能の32Kバイト送信可能な\$TTコマンドなど) しました。(V1は、最大1024バイトで送信)
- ▶ V1で開発したソフトの変更点 (参考)
 - ▶ 初期立ち上げ時の待機時間やメッセージを受信した立ち上げでの配慮が必要となります。
 - ▶ コマンドの応答が一部異なることへの配慮が必要となります。
 - ▶ a3gim.zipのバージョンが変わり、一部機能変更がありますので、別途関連資料をご参考ください。

【補足資料 8】 ラズベリーパイでの3GIM利用

3GIM(V2)は、Arduinoだけではなく、ラズベリーパイ等のLinuxでも簡単に利用できます。ここでは、ラズベリーパイで3GIM(V2)を3Gモデムとして利用するための手順をご紹介します。

3GIM(V2)を3Gモデムとして利用することで、3GIM(V2)をUART経由で利用する方法に比べて、下記の利点が得られます：

1) 高速な通信が可能

デフォルトで460kbpsの通信速度（設定ファイルで変更可能）

2) Linux上の豊富なネットワーク系コマンドが利用が可能

ブラウザ等のインターネットアプリケーションや、いわゆるSocket APIが利用できるのでC/C++やPython/JavaScript等で簡単にインターネット通信が利用できる

●情報サイト (http://a3gs.wiki.fc2.com/wiki/3gimV2_sample2)

ここでは、ラズベリーパイ 2/B/B+ で使用する例をご説明します。A/A+、さらにはZeroでも同様に利用できます。ただし、A/A+ではUSBコネクタの口が一つしかないため、（少なくとも設定する時には）USBハブが別途必要になります。

お問い合わせは、info@tbrain.jp まで



【補足資料 9】 ファームウェアのバージョンアップ対応

3 GIM(V2およびV2.1) では、ファームウェアのバージョンアップを無償にて行っています。

- ・ GPSの改良およびHL8548-Gのファームウェア改良などを行っています。

ファームウェアのバージョンアップ対応は、以下の対応で無償で行います。（期限は、2017年12月末日までです）

- 1) ファームウェアが V2 の場合（\$ YV でのバージョンが「V3.0」または「V3.1」の場合）
 - ・ GPS \$ LG x 1 と \$ LG x 2 の改良となります。（\$ YVでの改訂バージョンが「V3.2」となります）
- 2) ファームウェアが V2.1の場合、HL8548-Gのファームウェアの改訂
 - ・ \$ WP (httpPOST) でのエラーコードの改良となります。

※ 2017年2月以降ご購入いただいたお客様は、対象外となります。

（スイッチサイエンスやアマゾンでの販売での確認対応では、上記のバージョンアップは不要となっています）

■ 対応方法

- 1) 先にメールにて「バージョンアップ依頼希望」連絡 (info@tabrain.jp) を入れてください。

・ 記載事項：購入時期、IMEI、モジュール上のシールNo、その他開示できるお客様情報

- 2) 送料は、お客様での対応をお願いします。

・ 3 GIM送付のとき、返却時の受け取り住所を記載した受取人支払いの宅配用紙を入れてください。

<住所間違いなどを避けるため、よろしくお願いいたします>

- 3) 対応期間は、弊社稼働日において、受領日から2日間から1週間で対応いたします。

【補足資料10】 すぐに実運用されたい方に

購入後、直ぐにご利用されたい方は、以下の手順対応をご推奨いたします。

■ 1) 購入品として、以下のものを揃える

- ① 3GIM V2.1およびアンテナ（3Gアンテナ+GPSアンテナ）
- ② Arduino UNO または Genuino101
- ③ IoTABシールドまたは3GIMシールド
- ④ その他製品（USBケーブル、SIMカード）
 - ・ A-BタイプのUSBケーブル
 - ・ SIMカードは、iijmioプリペイドまたはsoracomSIMカードなど（NTTドコモ対応マイクロSIMカードです）

※IoT教材キット（[アマゾン販売](#)）は、上記のすべて揃っています。

こちらの教材キットには、サンプルプログラム（Arduinoスケッチ）も豊富についています。

■ 2) m2x（フリークラウド）に温度・光センサ値をアップしてグラフ表示してみる。

参考資料は、[ここ](#)に記載してあります。

■ 3) Arduino IDEを、Arduino.ccからダウンロードして環境を構築（IoTABシールドなどのマニュアル参照）

■ 4) 手順対応

- ① 3GIMにマイクロSIMカード挿入し、アンテナ（3G+GPS）を装着する
- ② IoTABシールドまたは3GIMシールド上に①の3GIMを取り付ける
- ③ 上記②のシールドを、Arduino UNO または Genuino101に取り付ける
- ④ シールド上のジャンパピンの切り替えが正しく接続されているのを確認する
- ⑤ サンプルプログラムをダウンロードして、Arduino IDE上でコンパイルして、ArduinoUNOまたはGenuino101書き込み、実行する
- ⑥ 正しくプログラムが実行されていることをシリアルモニタで確認する