4GIM（4G IoT Module）V1.0 利用マニュアル

本マニュアルは、4G IoT Module　V1.0 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 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
８）サンプルコード（第３章・第４章）zipファイル

世界最小クラス・省エネタイプ
４G通信モジュール（ボード）

株式会社 タブレイン

4GIM-V1.0R01

4 GIM V1.0の通信モジュールHL7539とは

- 世界最小サイズのLTE通信モジュール・ブレイクアウトボード
  - シエラワイヤレス社の「HL7539」（JATE/TELEC 取得済）・NTTドコモ（IOT取得済）を利用
  - サイズは 35mm × 25mm × 7mm，重量は7.5ｇ と非常に小型なボード
  - さまざまなIoTデバイスやゲートウェイとして利用できる携帯向けで消費電力が低い

HL7539（NTTドコモ用）の主な仕様	
RF	LTE B1:2100/B21:1500/B19:850 MHz
Speed	384Kbps(Download)/384Kbps(Upload)
その他	JATE　取得済み
サイズ・重さ	23㎜×22㎜×2.5㎜・2.5ｇ
電源電圧	3.2～4.5V　（推奨3.7V）
動作温度	-30℃ ～ 70℃

４GIM V1.0 の特長

４GIM V1.0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）４GIM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）エアプレーンモードを利用することで省電力化が可能
5）デバイスに組み込むためのカスタマイズが容易
6）利用料金の安いSIMカード(docomo系MVNOが提供するSIMカード)利用可能★
7）豊富な開発事例（多くが３GIMの開発事例）
8）試作から実運用まで幅広く利用可能
9）Arduinoの開発環境や豊富な資産が利用可能
10）IoT試作環境として技術情報が豊富
11）難しいATコマンドではなく、平易な$コマンドにより簡単にインターネット通信が利用可能
12）Arduinoおよび互換機向けのライブラリを用意したことで中学生などの初心者でも通信技術が利用可能

★ 一部特定サーバを利用するSIMカードでは利用できないものもあります。

この他、４GIM V1.0 の保守サポートは、以下のWikiページにて対応しています。

https://3gim.wiki/doku.php?id=4gim_v1

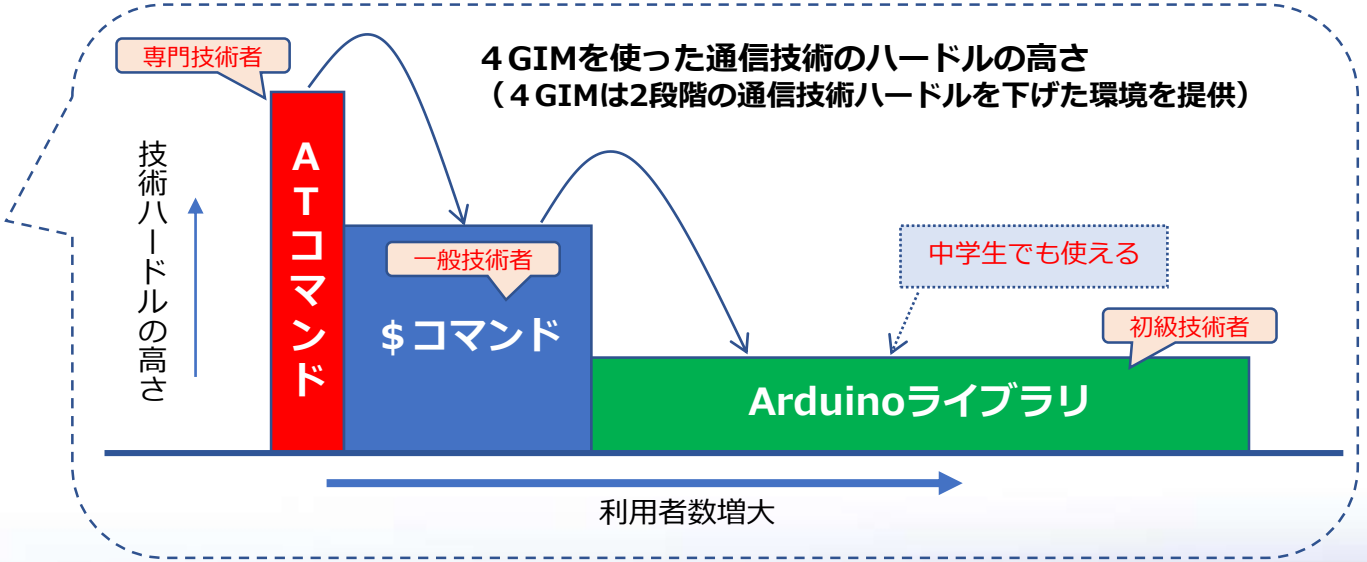
またFacebookにて、最新情報を掲載しています。

https://www.facebook.com/tabrain

Rev. 180129

４GIMはイノベーションを起こす製品

1. ４GIMは、創造を豊かにする製品づくりが可能
2. 最先端で高度な技術を簡単にした製品づくりが可能
3. ４GIM関連情報として、高度なプロトタイピング開発環境を提供
4. 分かり易いマニュアルと豊富なサンプルを提供
   →「真似て技術力を磨く」ことをコンセプト
5. 充実した保守サポート対応



目次

第1章
4GIM V1.0 の概要

目次

1．はじめに

- ４GIMは、M2MやIoTシステム開発での試作やから量産化に適したLTE通信モジュールです。

- 今後の新しいモノづくりにおいて、広域ワイヤレス通信（LTE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４GIMです。

- ４GIM（４G 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LTE通信モジュールです。

- ４GIMは、マイコンボード（Arudino関連ボード、mbedボード、GR-SAKURA、PIC、Intel Edisoni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「**a4gimライブラリ**」または「**a4gim2ライブラリ**」関連の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。また**3GIMシールド**や**IoTABシールド**などを活用することで簡単にArduino関連互換機上で４GIMが利用できるようになります。

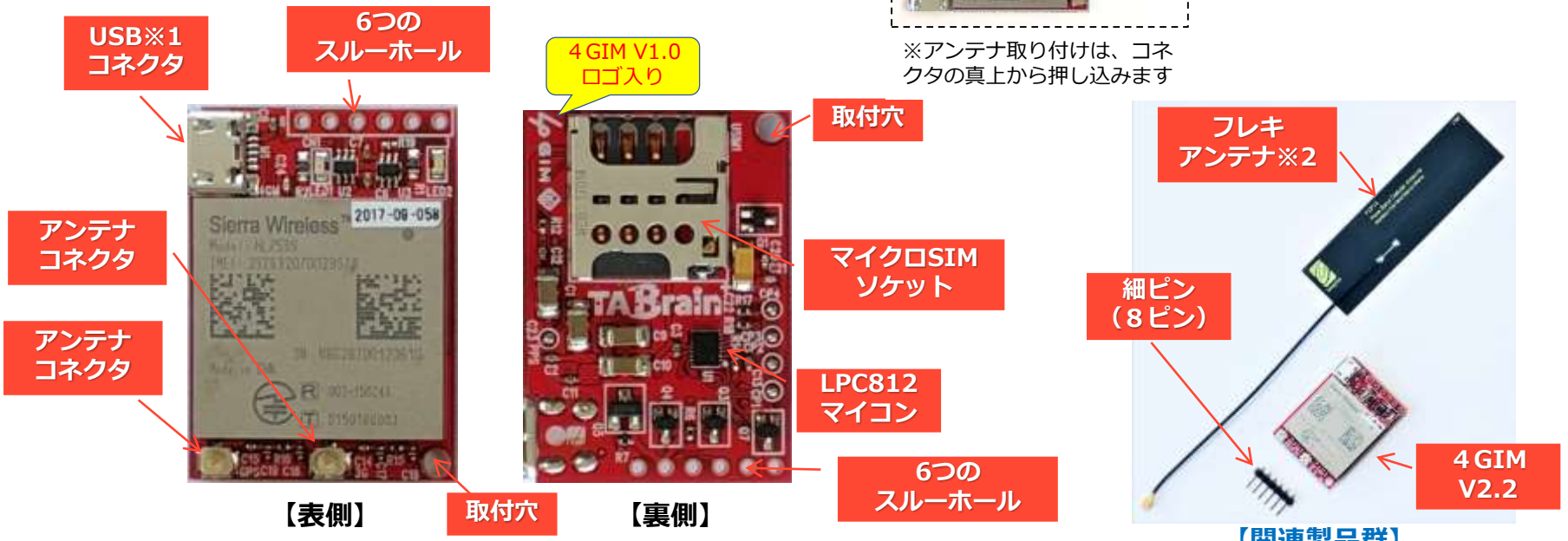
- 本製品の技術サポートおよび今後のマニュアル更新につきましては、以下のWikiページをご覧ください。

  https://a3gim.wiki/

- ４GIM V1.0のファームウェアgw4ｇのバージョンはV1.0です。これまでの3GIM V2.2に搭載されているgw3ｇ V3.3などとほとんど互換性があります（一部の機能が削除されたり追加されたりしています）。そのため特に３GIMで開発したソフトウェアの変更は少なくて済みます。詳しくは、補足資料7を参照ください。

2．４GIM V1.0の外観

■ ４GIMの外観写真およびその説明



- 表側には、**通信モジュール（HL7539）**、**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）
- ４GIM(LTE）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基板から剥がれることがありますので、ご注意ください。万が一コネクタが剥がれた場合には、有償で交換いたします。
※２：４Gフレキアンテナは、４GIMと組み合わせて日本で技適を取得したものです。
※３：アンテナは、２つ取付できます。２つのアンテナで感度が良くなります。

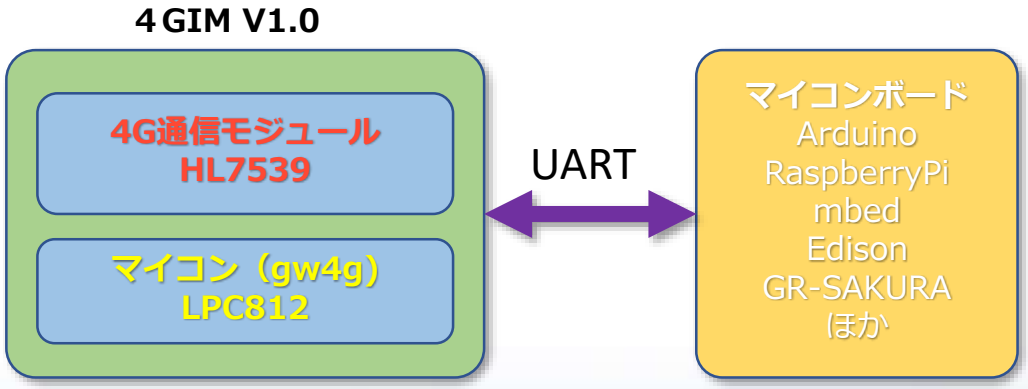
3．4GIM V1.0 の機能概要

- **4GIM V1.0は、下記の機能が提供されます。**
  1. 4G回線を介したインターネット接続（TCP/IPおよびHTTP、HTTPS※1）
  2. 4Gネットワークからの時刻取得
  3. プレーン（機内）モード機能による省電力化
  4. 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※2、APN※2の切り替え等）
  5. HL7539のATコマンドパススルー機能

  （SMS機能は利用できません。ご注意ください）

※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ものではありません。

※2：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。



4．4GIM V1.0 の仕様概要

項　目`	仕　様	補　足
外形寸法	幅25mm × 奥行35mm × 高さ**7**mm	**取付穴**はø2.6(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の電源は使用不可**）	安定したDC電源または3.7Vリチウムポリマ電池を推奨
消費電流	10～900mA（最大）	利用状況や電波状態に依存 （エアプレーンモード時は10mA程度）
通信規格・対応周波数	HL7539（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 **※1**)
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(４GIM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 **※2**	**初期設定**　**115200 bps**

※1：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。HL7539のATコマンド仕様書等を参照されてのご利用をお願いいたします。
※2：ソフトシリアル通信では、9600、19200、38400、57600bpsまで、ハードシリアル通信では、さらに115200bpsまで利用可能です。

5．4GIM V1.0 のピンコネクタ配置

6つのスルーホール

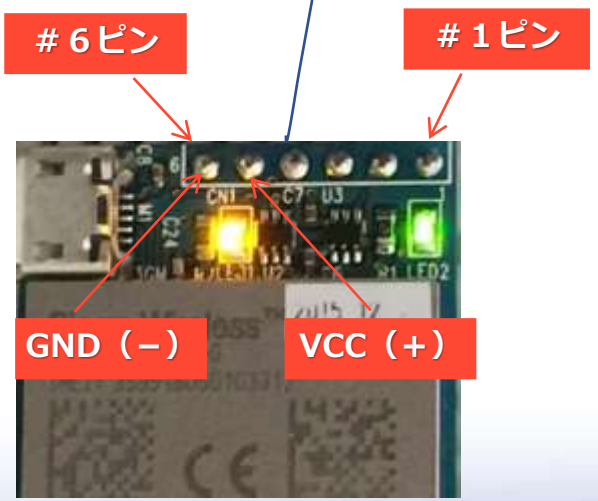
ピン番号	名称	機能など
#1	PWR_ON	電源のON/OFF制御(開放または0:LOWでON、1:HIGHでOFF)
#2	RX	UARTインタフェース(RX)：相手方のTxに接続
#3	TX	UARTインタフェース(TX)：相手方のRxに接続
#4	IOREF	ロジック電圧(任意、通常は**1.8V～5V**で利用)
#5	VCC（+）	**電源電圧(3.3～4.2V)　※瞬間的でも3.2V以下にならないこと**
#6	GND（−）	グラウンド

【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。
供給電流能力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）

6．４GIMを利用するソフトウェアについて

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

4 GIM V1.0

HL7539
（通信モジュール）

4 GIM V1.0
対応

アプリケーション
<gw4g (V1.0)>

Arduino以外
＄コマンド利用

第２章・第3章・第4章 参照

IoTAB シールド V４.0など利用

UART

Power

Arduino(UNO, 101, Zero, Pro, Mega, DUE..)

Arduino アプリケーション

Arduino ライブラリ <a4gim(R4.３)>

SoftwareSerial ライブラリ

Arduino ブートローダ

USB or
外部電源

Arduinoの場合
ライブラリ関数利用

第５章参照

Rev. 180129

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。）

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。詳細はタブレイン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。

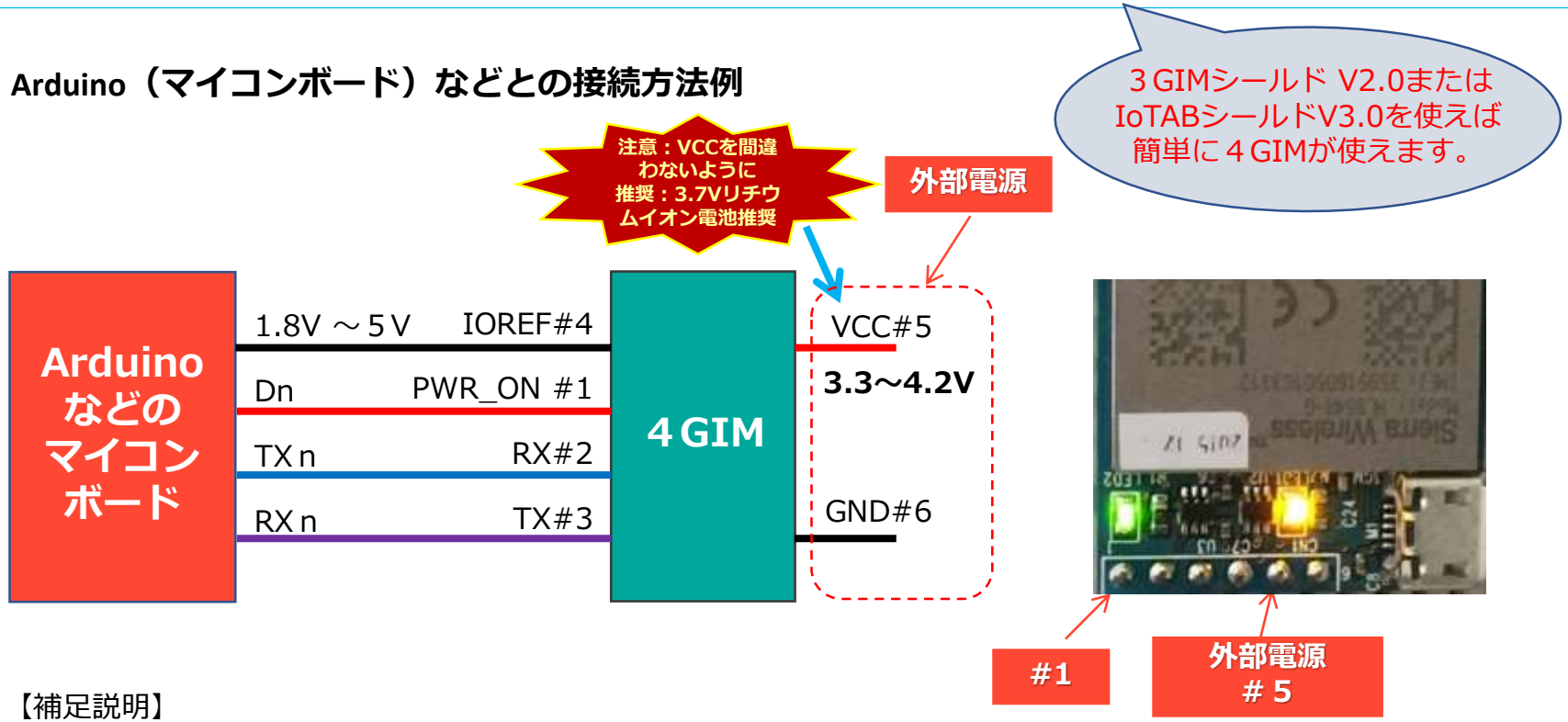
4. 回路図は、オープンソースとして公開します。将来的には、プリインストールしているファームウェアも公開する予定です。

5. GPS取得は、電波状態の良い屋外などで行ってください。また初回のGPS取得時では、特にPCなどの電波障害源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が有効）

   （初回のGPS取得には、数分間の時間が掛かります。ただし、Assisted GPS機能を使うと屋外では30秒前後で取得できます）

【参考】Arduinoで利用する際の接続

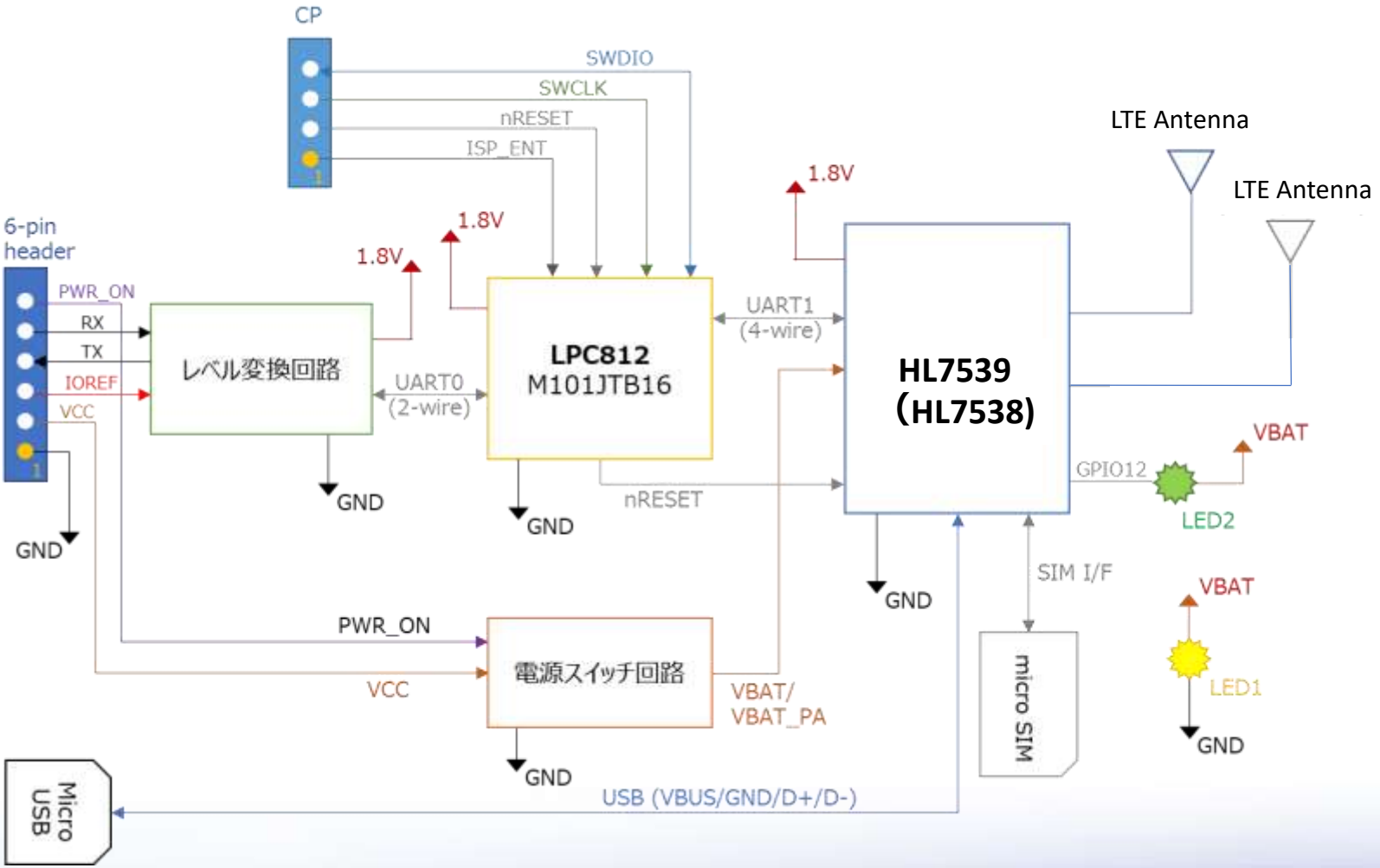
- **Arduino（マイコンボード）などとの接続方法例**



【補足説明】

- デジタル出力Dn（#1接続）をLOWにすることで、４GIMの電源をオンにします。なお、４GIM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に取る事で稼動できます**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>

【参考】 4GIM(V1.0)の内部ブロック図



第２章
$コマンドインタフェース

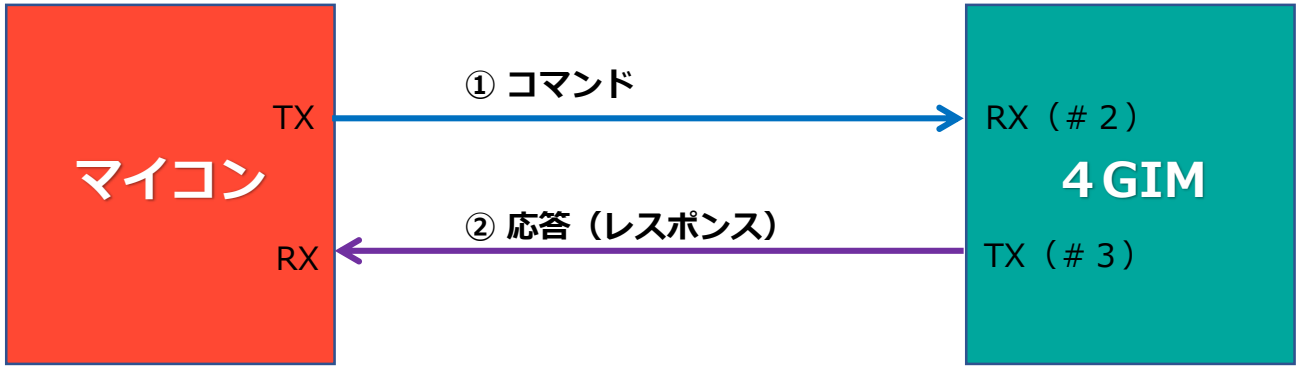
目次

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４GIM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、４GIM側で受信します。
- つぎに４GIM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４GIM側へのコマンド送信と、４GIM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、115200bps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。

2．4GIM V1.0 コマンド一覧

4 GIM V1.0 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		gw4 g アプリのバージョン情報の取得	
2		RSSI	$YR		電波受信強度(RSSI)の取得	
3		Service	$YS		利用可能サービスの取得	
4		IMEI	$YI		IMEIの取得	
5		LED	$YL		LED（RUN）の状態の取得、設定	
6		Baudrate	$YB		UARTの通信速度の変更	【拡張】リセット後に有効
7		Reset	$YE		リセット（初期化）	
8		Time	$YT		日時の取得	
9		Airplane mode	$YP		エアプレーン（機内）モードの切り替え	
10		ATcommand	$YA		ATコマンドパススルーモード切換え	【新規】
11	Web	Get	$WG		GETリクエストの送出、レスポンスの取得	ヘッダ指定可(R2.0から)
12		Post	$WP		POSTリクエストの送出、レスポンスの取得	
13	TCP/IP	Read	$TR		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
14		Write	$TW		TCP/IPコネクションへのデータの書き込み	同上
15		Connect	$TC		TCP/IPコネクションの接続	
16		Disconnect	$TD		TCP/IPコネクションの切断	
17		Status	$TS		TCP/IPコネクションの状態の取得、設定	【変更】
18		Get sockname	$TN		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
19		Tunnel Write	$TT		現在のコネクションへダイレクトにデータ書出し	
20		Set/Get Param	$TX		TCP/IPコネクションで使用するタイムアウト時間の取得・設定	【V2.1以降利用可】
21	UDP	Start	$UB		UDP機能の開始	
22		End	$UE		UDP機能の終了	
23		Send	$US		UDP/IPを使って指定した宛先へデータ送信	
24		Get Sockname	$UN		自ソケットのIPアドレスを取得	
25	Profile	SIM Profile	$ PS		使用するプロファイルのAPN/ユーザ名/PWの取得・設定	

4 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。

3．4GIM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ります。引数には制御コード（TABコード等）は含まないでください。また、$文字は先頭にスペースを入れずに指定してください。

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ことで対応できます。

LF(¥n) が改行コードとなります。CR(¥r) ではありませんのでご注意ください。

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。

【補足1】4GIM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 4GIM(v1)

この応答が無い場合は、電源電圧が不足しているか、＃1ピンのOn/Offが行われていないことなります。はんだ付け不良も原因しています。

【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

■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**rn**"

４．４GIMのインタフェース形式（補足事項①）

■初期起動時について

４GIM 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**14**秒後に４GIM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４GIMと接続できたことになります。

```
Welcome to ４GIM(v1)
```

※下線部は、４GIMのハードウェアのバージョンを示す。
一方、ソフトウェアのバージョン情報は $YVコマンド で取得できる。

■コマンドの応答（レスポン）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、ヘッダ部とボディ部が返ってきますので、複数行のレスポンスが返ってくる場合がありますので、ご注意ください。

```
$WG http://tabrain.jp/
$WG=OK 66
```

４．４GIMのインタフェース形式（補足事項②）

■４GIMを使ったArduinoのサンプル・スケッチ（起動プログラム）について

４GIM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４GIMから返ってくることを前頁で紹介しました。

ここでは、Arduinoでの４GIMのスケッチ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になります。falseの場合には、通信に異常があったことで、通信速度の設定が間違ったこと等に起因します。

```
boolean _4G_SETUP() {
  String str;
  uint32_t tim = millis();
  do {
    while (!Serial4G.available()) {
      if (millis() - tim > 15000) return false;
    }
    str = Serial4G.readStringUntil('¥n');
  } while (str.indexOf("４GIM")<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115200bpsとなっています。
もし、変更された場合には、メモをしておく必要があります。

上記は、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。　通常は起こりえませんが、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、HL7539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HL7539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。

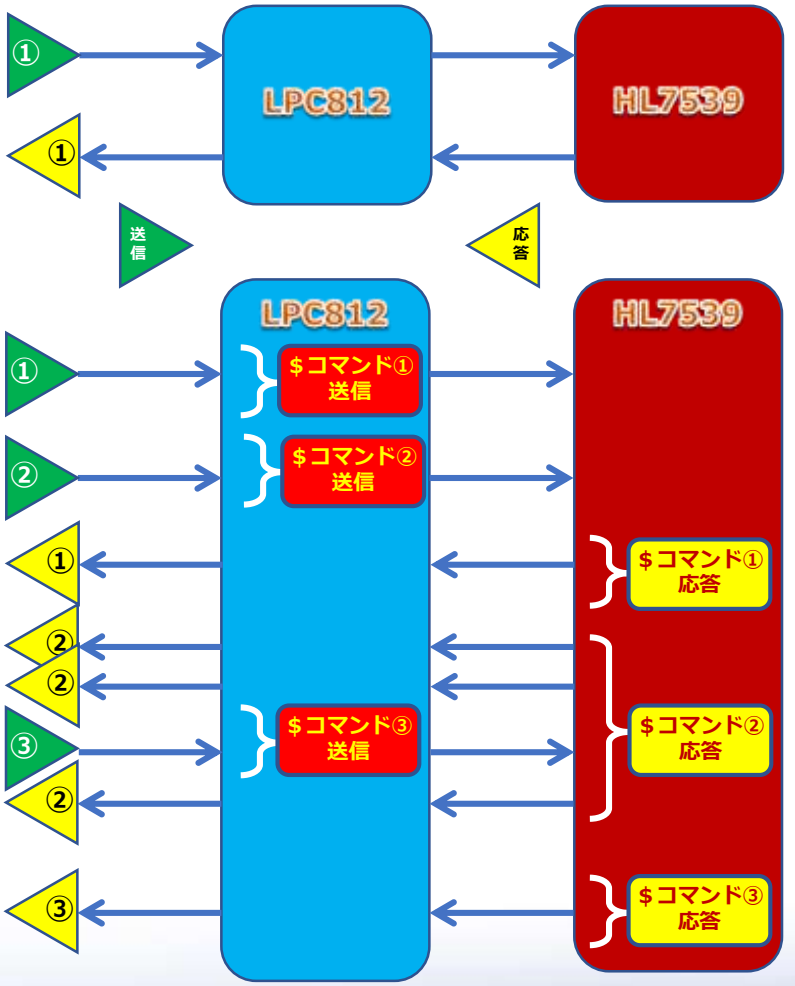
以下のサンプルスケッチは、送信後、応答を待つ２つのケースです。

■応答値を使う場合

```
Serial4G.println( "$WG " + str);
do { str = Serial4G.readStringUntil('¥n'); }
while (str.indexOf("$WG") < 0);
```

■応答値を使わない場合

```
Serial4G.println("$YL " + String((int)sw));
while (Serial4G.readStringUntil('¥n').indexOf("$YL") < 0);
```



＄コマンド送信と応答の関係（事例）

第３章
４GIMコマンド・応答（レスポン）

目次

１．システム関連

1．SYSTEM VERSION ①

■ 4GIM V1.0（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（gw4g）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（gw4g）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	①	本コマンドで取得できるバージョン情報は、ファームウェアgw4 g のバージョン情報である。４GIMのハードウェアのバージョン情報とは異なる。	

※ 2018年2月の出荷時点ファームウェアバージョンは「4.0」である。

1．SYSTEM VERSION ②

■事例：バージョン取得サンプルプログラム

スケッチ名：system_version.ino

■Arduinoでのサンプルスケッチ

```
#define  Serial4G Serial1
void setup() {
  Serial.begin(115200);
  delay(100);
  Serial4G.begin(115200);
  Serial.println("begin SYSTEM version");
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH);
  delay(100);
  digitalWrite(7,LOW);
  while(Serial4G.readStringUntil('¥n').indexOf("４GIM")<0);
  Serial4G.println("$YV");
  while(!Serial4G.available());
  Serial.println(Serial4G.readStringUntil('¥n'));
  Serial.println("end");
  pinMode(7,OUTPUT);digitalWrite(7,HIGH);
}

void loop() {}
```

通信速度設定

■実行モニタ画面（サンプル）

```
begin SYSTEM Version
$YV=OK 4.0
end
```

応答受信

【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
・Serial4G.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
・Serial4G.listen()は、受信状態占有関数
・Serial4G.available()は、受信バイト数を返す
・Serial4G.readStringUntil('¥n')は、
　　リターン値までの文字列読込み関数

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		**$YR¥n**	
引数	-		
応答値	【正常時】	**$YR=OK rssi¥n**	
	rssi	電波強度（-51～-113）[dBm]	rssiは常にマイナス値
	【エラー時】	**$YR=NG errno¥n**	
	errno	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	①	４GIMをONにした直後ではエラーとなる場合がある。そのため、確実にRSSIを取得するには、1秒程度の間隔を空けて３回程度のリトライを行うことを推奨する。	

RSSIとは、無線通信機器が受信する信号の強度を表す。（Received Signal Strength Indication, Received Signal Strength Indicator 別名：受信信号強度）

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

スケッチ名：system_rssi.ino

■Arduinoでのサンプルスケッチ

```
#define Serial4 Serial1
void setup() {
  Serial.begin(115200);
  Serial4G.begin(115200);
  Serial.println(“begin SYSTEM RSSI");
  pinMode(7,OUTPUT);
  digitalWrite(7, HIGH);
  delay(100);
  digitalWrite(7, LOW);
  while(Serial4G.readStringUntil('¥n').indexOf("４GIM")<0);
  Serial4G.println("$YR");
  while (!Serial4G.available()) ;
  Serial.println(Serial4G.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（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	４GIM(Ver2.x)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。	
	②	４GIMをONにした直後ではエラーとなる場合、あるいはパケット通信が利用できるにも関わらず0を返す場合がある。	

４GIM V1.0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

スケッチ名：system_service.ino

■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

```
#define Serial4 Serial1
void setup() {
  Serial.begin(115200);
  Serial4G.begin(115200);
  Serial.println("begin SYSTEM IMEI");
  pinMode(7,OUTPUT);
  digitalWrite(7, HIGH);
  delay(100);
  digitalWrite(7, LOW);
  while(Serial4G.readStringUntil('¥n').indexOf("４GIM")<0);
  Serial4G.println("$YI");
  Serial.println(Serial4G.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		**$YI¥n**	
引数	-		
応答値	【正常時】	**$YI=OK imei¥n**	
	imei	15桁の数字（４GIMを一意に識別できる数字列）	
	【エラー時】	**$YI=NG errno¥n**	
	errno	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

スケッチ名：system_imei.ino

■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

```
#define Serial4 Serial1
void setup() {
  Serial.begin(115200);
  Serial4G.begin(115200);
  Serial.println("begin SYSTEM IMEI");
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH);
  delay(100);
  digitalWrite(7,LOW);
  while(Serial4G.readStringUntil('¥n').indexOf("４GIM")<0);
  Serial4G.println("$YI");
  while (!Serial4G.available()) ;
  Serial.println(Serial4G.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

■ 4GIM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	状態取得	**$YL¥n**	
	設定	**$YL status¥n**	
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）	

5．SYSTEM LED ②

スケッチ名：system_led.ino

■事例：LED状態取得とLEDの点滅を10回繰り返すサンプルプログラム

■Arduinoでのサンプルスケッチ

```
#define Serial4 Serial1
void setup(){
  Serial.begin(115200);
  Serial4G.begin(115200);
  Serial.println("begin SYSTEM LED");
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH);
  delay(100);
  digitalWrite(7,LOW);
  while(Serial4G.readStringUntil('¥n').indexOf("４GIM")<0);
  Serial4G.println("$YL");
  String rt = Serial4G.readStringUntil('¥n');
  Serial.print(rt);
  for (int i=0; i<10; i++) {
    Serial4G.println("$YL 1");
    delay(500);
    Serial4G.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

■ 4GIM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	**$YB¥n**	
	設定	**$YB baudrate¥n**	
引数	baudrate	設定するボーレート(9600/19200/38400/57600/115200)	出荷時は115200bps
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

スケッチ名：system_baudrate.ino

■事例：４GIM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

```
#define Serial4 Serial1
#define NOWBAUDRATE  115200
#define NEWBAUDRATE  38400
void setup() {
  Serial.begin(NOWBAUDRATE );
  Serial.println("begin SYSTEM Baudrate");
  Serial4G.begin(NOWBAUDRATE);
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH);
  delay(100);
  digitalWrite(7,LOW);
  while(Serial4G.readStringUntil('¥n').indexOf("４GIM")<0);
  Serial4G.println("$YB " + String(NEWBAUDRATE));
  delay(20);
  Serial4G.begin(NEWBAUDRATE);
  Serial.println(Serial4G.readStringUntil('¥n'));
  Serial.println("end");
}

void loop() {}
```

この通信速度にする

この待機時間重要

ボーレートを115200bps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、その情報はメモ等で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

■ ４GIM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	４GIM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４GIM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にして電源をOFFすることでも実行できます。
電源がOFFとなった時は、４GIM上の橙LED(LED1)が消灯します。

7．SYSTEM RESET ②

スケッチ名：system_reset.ino

■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

```
#define Serial4 Serial1
void setup(){
  Serial.begin(115200);
  Serial4G.begin(115200);
  Serial.println(“begin SYSTEM Reset");
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH);
  delay(100);
  digitalWrite(7,LOW);
  while(Serial4G.readStringUntil('¥n').indexOf(" 4 GIM")<0);
  Serial4G.println("$YE");
  Serial.println(Serial4G.readStringUntil('¥n'));
  while(Serial4G.readStringUntil('¥n').indexOf(" 4 GIM")<0);
  Serial.println("restart:");
  Serial.println("end");
}
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 2018/05/14 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、HL7539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。	初回の実行時はインターネットに接続するため、正しいプロファイル情報の設定、SIMカードおよび3Gアンテナの装着が必要となる。

ファームウェア起動後に「＄YT」を起動すると初回のみ**5～10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

スケッチ名：system_time.ino

スケッチ名：system_time2.ino

■Arduinoでのサンプルスケッチ

```
#define Serial4 Serial1
void setup() {
  Serial.begin(115200);
  Serial4G.begin(115200);
  Serial.println("begin System Time");
  Serial4G.println("$YT");
  Serial.println(Serial4G.readStringUntil('¥n'));
  Serial.println(“end");
}
void loop() {}
```

■実行モニタ画面（正常時）

```
begin System Time
$YT=OK 2018/08/13 10:58:11
end
```

■時間取得関数（例）

```
String datetime() {
  String dtime;
  uint32_t tim = millis();
  do {
    Serial4G.println("$YT");
    dtime = Serial4G.readStringUntil('¥n');
    if (millis() - tim > 60000) return "error";
  } while (dtime.indexOf("20") < 0);
  return dtime.substring(7);
}
```

【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5～10 秒ほど掛る
※2回目以降は、瞬時に取得できる

※注意：正しいSIMカードとアンテナ接続がされてないと
時間取得ができません。

```
begin System Time
$YT=OK 2018/08/13 11:01:10
continue
2018/08/13 11:01:10
2018/08/13 11:01:15
2018/08/13 11:01:20
2018/08/13 11:01:25
2018/08/13 11:01:30
2018/08/13 11:01:35
2018/08/13 11:01:40
2018/08/13 11:01:45
```

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	４GIM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	**$YP¥n**	
	モード設定	**$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			
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。	

補足：本エアプレーンモード時は、**４GIMの消費電流を数ミリA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、４GIM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

スケッチ名：airplane_mode.ino

■Arduinoでのサンプルスケッチ

```
#define Serial4 Serial1
void setup() {
  Serial.begin(38400);
  Serial4G.begin(38400);
 Serial.println("begin AirPlane Mode");
  pinMode(7, OUTPUT);
  digitalWrite(7, HIGH);
  delay(100);
  digitalWrite(7, LOW);
  while (Serial4G.readStringUntil('¥n').indexOf("４GIM") < 0);
  Serial4G.println("$YP");
  Serial.println(Serial4G.readStringUntil('¥n'));
  Serial.println(“end");
}
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	**$YA【time】¥n**	
引数	time	待機時間：単位100ミリ秒；（例： time=300 は 30秒間を意味する）	
応答値	【正常時】	**$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、HL7539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gw4ｇファームウェアの動作に支障を来たす場合があるので、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、LTE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、４GIMの電源を入れなおす。 ②「**AT+CPOF**」コマンドを実行して４GIMをリセットする。	①・②共に初期起動時に戻る。「Welcome to ４GIM(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)

```
Serial4G.println("$YA 1");
delay(10);
Serial4G.println("at+wppp=2,4,¥"username¥",¥"password¥"");
```

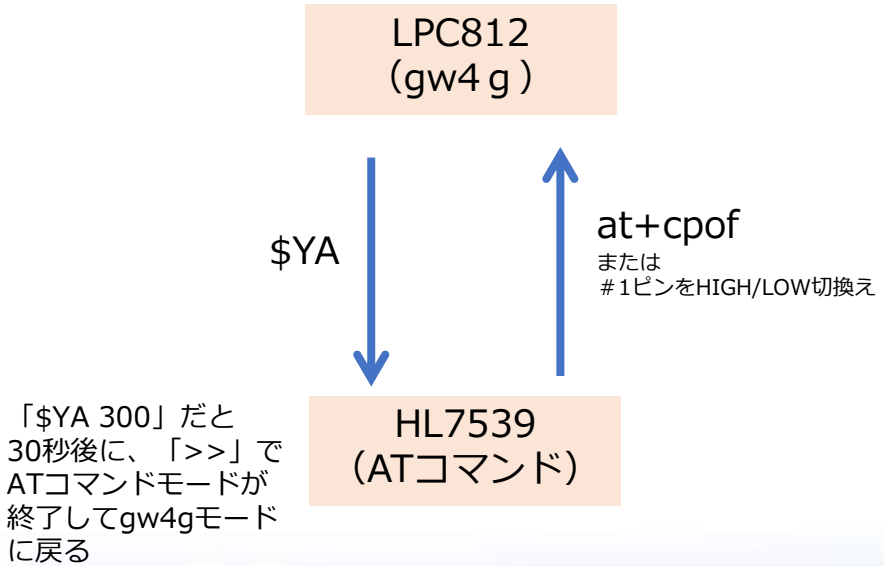
※usernameとpasswordは、SIMカードのAPN情報のユーザ名とパスワード
<本マニュアルP.50参照>

Rev. 180129

１０．AT COMMAND ②

HL7539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。

【注意点】
ATコマンドでHL7539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、４GIMが利用できなくなる場合があります。
ATコマンドで操作する場合には、充分な知識を得た上で、ご利用いただく事お薦めいたします。



2．Web関連

1．HTTP GET ①

■HTTP GETの実行

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、"https://www.arduino.cc/"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 QWxhZGRpblc2FtZQ=="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**	
	errno	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	タイムアウトは30秒設定
		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
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事
補足事項	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

スケッチ名：http_get.ino

■Arduinoサンプルプログラム

```
#define Serial4 Serial1
void setup() {
  Serial.begin(38400);
  Serial4G.begin(38400);
  pinMode(7, OUTPUT);
  digitalWrite(7, HIGH);
  delay(100);
  digitalWrite(7, LOW);
  Serial.println("Start");
  while (Serial4G.readStringUntil('¥n').indexOf(" 4 GIM") < 0);
  Serial.println("begin HTTP GET");
  Serial4G.println("$WG http://tabrain.jp/demo/httpGET_test.txt");
  delay(1000);
  unsigned long tm = millis();
  while (millis() - tm < 35000) {
    while (Serial4G.available()) {
      char c = Serial4G.read();
      Serial.print(c);
    }
  }
  Serial.print("¥r¥nend");
}
```

■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  4 GIM



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  4 GIM
```

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の実行

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**	
	errno※	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	値の範囲は-400～-599
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	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	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。	
	③～⑥	HTTP GETの補足事項を参照のこと	

※「errono」は、WiKiページなどで補足説明しています。

2．HTTP POST ②

■事例：HTTP POSTによるツイート参照

スケッチ名：twitter_sample.ino

本事例は、後述しています
「4 GIM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 & $WP command ===============
boolean _4G_WGP(String command) {
  Serial4G.println(command);
  String rstr;
  unsigned long tim = millis();  // time set(ms)
  do{
       while(!Serial4G.available ());
       rstr=Serial4G.readStringUntil('¥n');
  } while (rstr.indexOf("$W") != 0 && (millis() - tim) < LIMITTIME);
  return (rstr.indexOf("$W") == 0);
}
//=============================================
```

ここで Serial4G は、シリアル通信ポート
LIMITTIMEは、制限時間（14000msec）を設定のこと

４．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	gw4ｇ R2.0からバイナリデータも取扱可
	【エラー時】	**$TR=NG errno ..¥n**	
	errno	631：コマンドの指定、または引数に誤りがある	
		633：READエラー	
		635：コネクションがない（未接続）	
		636、637：コネクションのステータスのエラー	
		639：タイムアウトエラー	タイムアウト時間は60秒（$TXコマンドで変更可能）
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に取得する	
	②	呼び出された時に４GIM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。	常にノンブロッキングで動作する
	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回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】	**$TC=NG errno ..¥n**	
	errno	601：引数がおかしい	
		603：すでに接続済み	
		604：コネクションエラー(ホストやポートが間違っている場合を含む)	タイムアウト時間は60秒（$TXコマンドで変更可能）
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

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**	option省略時
		$TS=OK status tcpnotif remainedBytes receivedBytes¥n	option=1の時
	status	0：CLOSED（接続なし）	
		1：DISCONNECTING	
		2：DISCONNECTED（接続待ち）	
		3：CONNECTING	
		4：CONNECTED（送受信待ち）	
	tcpnotif	下記の【TCP機能全般の留意点】を参照	オプション(1)が指定された時のみ出力される
	remainedBytes	未送信状態のデータのバイト数	同上
	receivedBytes	受信状態のデータのバイト数（$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、正確な最新情報が得られる。	

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へのデータの直接書き出し

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	$エンコードは不要、バイナリデータもそのままでOK
返却値	【正常時】	**$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．SET/GET PARAMETERS

■TCP/IP機能のパラメータの設定・取得

項目	値など	説明	補足
機能分類	TCP/IP		
機能名	Control	TCP/IP関連コマンドでのタイムアウト時間を取得・設定する。	
コマンド形式	パラメータ取得	**$TX¥n**	
	パラメータ設定	**$TX timeout¥n**	
引数	timeout	設定するタイムアウト時間（10ミリ秒単位、0～6000の範囲）	デフォルトは6000
応答値	【正常時】	**$TX=OK¥n** **$TX=OK timeout¥n**	
	timeout	取得したタイムアウト時間（10ミリ秒単位）	
	【エラー時】	**$TX=NG errno¥n**	
	errno	671：コマンド形式がおかしい 672：指定されたタイムアウト時間がおかしい	
前提条件			
補足事項	①	本コマンドで設定したタイムアウト時間は、４GIMの電源をOFFにした時やリセットした時に、デフォルトの時間（6000=60秒間）に戻る。	
	②	極端に短いタイマアウト時間に設定することは推奨しない。TCP/IPネットワークの世界では、ネットワーク上でデータをやり取りする関係で、本質的にエラーを検出するためには一定の時間（通常は数秒から30秒程度で、ネットワークや3Gの電波状態に大きく依存する）が掛かる。この時間よりも短いタイムアウト時間を設定した場合、例えば$TCコマンドからエラーが返ってきた後に、正常に接続できた状態となったりする等の挙動があり得る。従って、設定するタイムアウト時間は、いろいろと試してみて具体的な値を決めるといった手順を踏むことを推奨する。	

9．TCP/IP 利用サンプルプログラム ①

スケッチ名：tcp_ip.ino

■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

```
#define Serial4G Serial1
void setup() {
  Serial.begin(115200);  Serial4G.begin(115200);
  pinMode(7, OUTPUT);
  digitalWrite(7, HIGH); delay(100); digitalWrite(7, LOW);
  Serial.println("Ready...");
  while (Serial4G.readStringUntil('¥n').indexOf("4GIM") < 0);
  Serial.println("begin TCP/IP sample"); delay(100);
  Serial4G.println("$TC www.tabrain.jp 80");
  Serial4G.println("$TW ¥"GET / HTTP/1.1$r$n¥"");
  Serial4G.println("$TW ¥"HOST: www.tabrain.jp$r$n$r$n¥"");
  tcpip("$TR 500");
  Serial4G.println("$TD");
  Serial.print("¥r¥nend");
}

void loop() {}
```

```
String tcpip(String cmd ) {
  Serial.print(cmd + " -> ");
  Serial4G.println(cmd);
  String str;
  do{
    str = Serial4G.readStringUntil('¥n');
    if(str.length()>0 )Serial.println(str);
  } while( str.indexOf(cmd.substring(0,3))<0);
  int n = str.substring(6).toInt();
  if(cmd.indexOf("TR")>0 && n>0) {
    for(int i=0; i<n ; i++) {
      while(!Serial4G.available());
      Serial.write(Serial4G.read());
    }
  }
}
```

10. 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...
begin TCP/IP sample
$TR 500 -> $TC=OK
$TW=OK 16
$TW=OK 24
$TR=OK 251
HTTP/1.1 200 OK
Date: Sun, 13 Aug 2017 03:37:08 GMT
Server: Apache
Last-Modified: Sat, 20 May 2017 00:44:24 GMT
Accept-Ranges: bytes
Content-Length: 66
Content-Type: text/html

<meta http-equiv="refresh"
content="1;URL=http://tabrain.jp/new/">
end
```

読込みバッファの出力結果
（ここでは251バイト出力）

11. 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
3)	$TR/$TCコマンドのタイムアウト時間は、デフォルトで60秒である。この時間は、$TXコマンドによって変更することができる。 また同様に、$TW/$TTコマンドのエラー発生時のタイムアウト時間は、デフォルトで60秒である。この時間も、$TXコマンドで変更することができる。（このタイムアウト時間は$TR/$TCのタイムアウト時間と共通であるため、片方だけを変更することはできない）	

5．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**	設定時
		$PS=OK ”apn”,“user”,”password”¥n	取得時
	【エラー時】 **errno**	**$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	本コマンドの実行にあたっては、有効なSIMカード（利用可能なSIMカード）と3Gアンテナ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４GIM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（引数をダブルクォートで囲む必要はないが、囲んでもよい）
ユーザ名やパスワードが無いSIMカードもあるが、その場合には適当な文字列を指定すること（引数を省略することはできない）。

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

スケッチ名：profile_set.ino

■Arduinoサンプルプログラム

```
#define Serial4G Serial1
void setup() {
  Serial.begin(115200);
  Serial4G.begin(115200);
  pinMode(7, OUTPUT);
  digitalWrite(7, HIGH);
  delay(100);
  digitalWrite(7, LOW);
  Serial.println("begin Profile Set");
  while (Serial4G.readStringUntil('¥n').indexOf("４GIM") < 0);
  String str;
  Serial4G.println("$PS iijmio.jp mio@iij iij");
  do { str = Serial4G.readStringUntil('¥n');
  } while (str.indexOf("$PS")<0);
  Serial.println(str);
  Serial4G.println("$PS");
  do { str = Serial4G.readStringUntil('¥n');
  } while (str.indexOf("$PS")<0);
  Serial.println(str);
  Serial.println("end");
}
void loop() {}
```

■シリアルモニタ画面（正常終了の場合）

```
begin Profaile Set
$PS=OK
$PS=OK "iijmio.jp","mio@iij","iij"
end
```

■プロファイル設定サンプル（他MVNO製品も同様に設定）

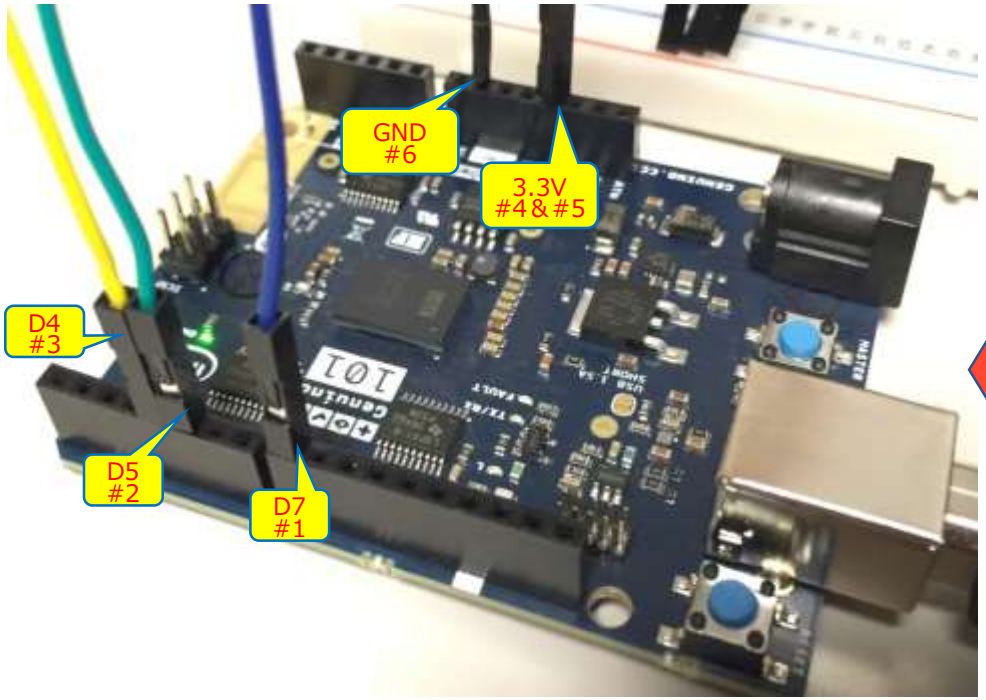
SIMメーカ製品名	設定方法
DOCOMO mopera	$PS mopera.net
iijmio	$PS iijmio.jp mio@iij iij
iijmobile	$PS iijmobile.jp mobile@iij iij
SONET	$PS so-net.jp nuro nuro
SORACOM	$PS soracom.io “” “”
EXCITE	$PS vmobile.jp bb@excite.co.jp excite
HI-HO	$PS vmobile.jp lte@hi-ho hi-ho
BMOBILE	$PS bmobile.ne.jp bmobile@u300 bmobile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

第４章
応用事例プログラミング

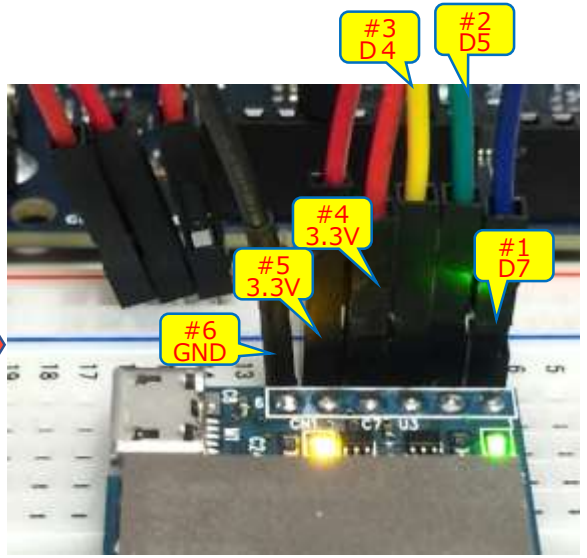
目次

１．Arduinoのシリアルモニタによる操作

1．Genuino/Arduino 101 との接続例



接続



※Genuino 101の3.3V出力が１Aあるとのことで
４GIMと接続してみました。

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。

【注意】電源（電流）不足の場合は、外部電源を取る必要があるかもしれません。
（電源不足の場合、４GIMが立ち上がらなかったり、インターネット接続ができないことがおきます）

電源不足の場合には、３GIMシールドやIoTABシールドなどをご利用ください。

2．Arduino上での簡単な利用例

■ソフトウェア（次頁の monitor４GIM.ino ）をビルドし、Arduino UNO　や Genuino/Arduino 101に書き込んで動かしてみてください。

■必要な部品：

① Genuino/Arduino 101　（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 ４GIM(v2.2)」が表示された時点で立ち上がった段階です。

② ４GIMの各＄コマンドを入力して、応答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

- Arduino UNO+３GIMシールド＋４GIM V1.0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。

スケッチ名：monitor４GIM.ino

■ シリアルモニタ画面での４GIM入出力プログラム
（monitor４GIM.ino：＊３GIMシールド＋Arduinoで利用可能）

```
#define Serial4G Serial1
const unsigned long baudrate = 115200;

void setup() {
  Serial.begin(baudrate);
  Serial4G.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 (Serial4G.available() > 0) {
    char c = Serial4G.read();
    Serial.print(c);
  }

  if (Serial.available() > 0) {
    char c = Serial.read();
    Serial.print(c);    // Echo back
    Serial4G.print(c);
  }
}
```

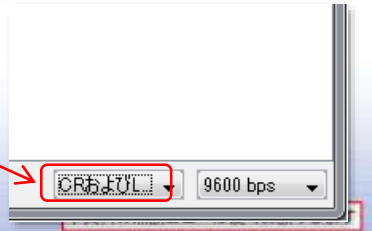
本サンプルスケッチは、
シリアルモニタ画面で、コマンドをキー入力することで、応答（レスポンス）を表示確認できるもので、マニュアル操作でのコマンド/レスポンスが即座に見ることができます。

■ シリアルモニタ画面での操作例

```
Ready.

Welcome to  4 GIM(v1)
$YV
$YV=OK 3.3
$YI
$YI=OK 359516050532664
$YR
$YR=OK -77
$YB
$YB=OK 38400
$LG x 1
$LG=OK 35.641889 139.604123 040056.756 1 7 83.4
$YT
$YT=OK 2017/08/13 13:01:11
$WG http://tabrain.jp
$WG=OK 66
<meta http-equiv="refresh"
content="1;URL=http://tabrain.jp/new/">
```

赤字：入力
黒字：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

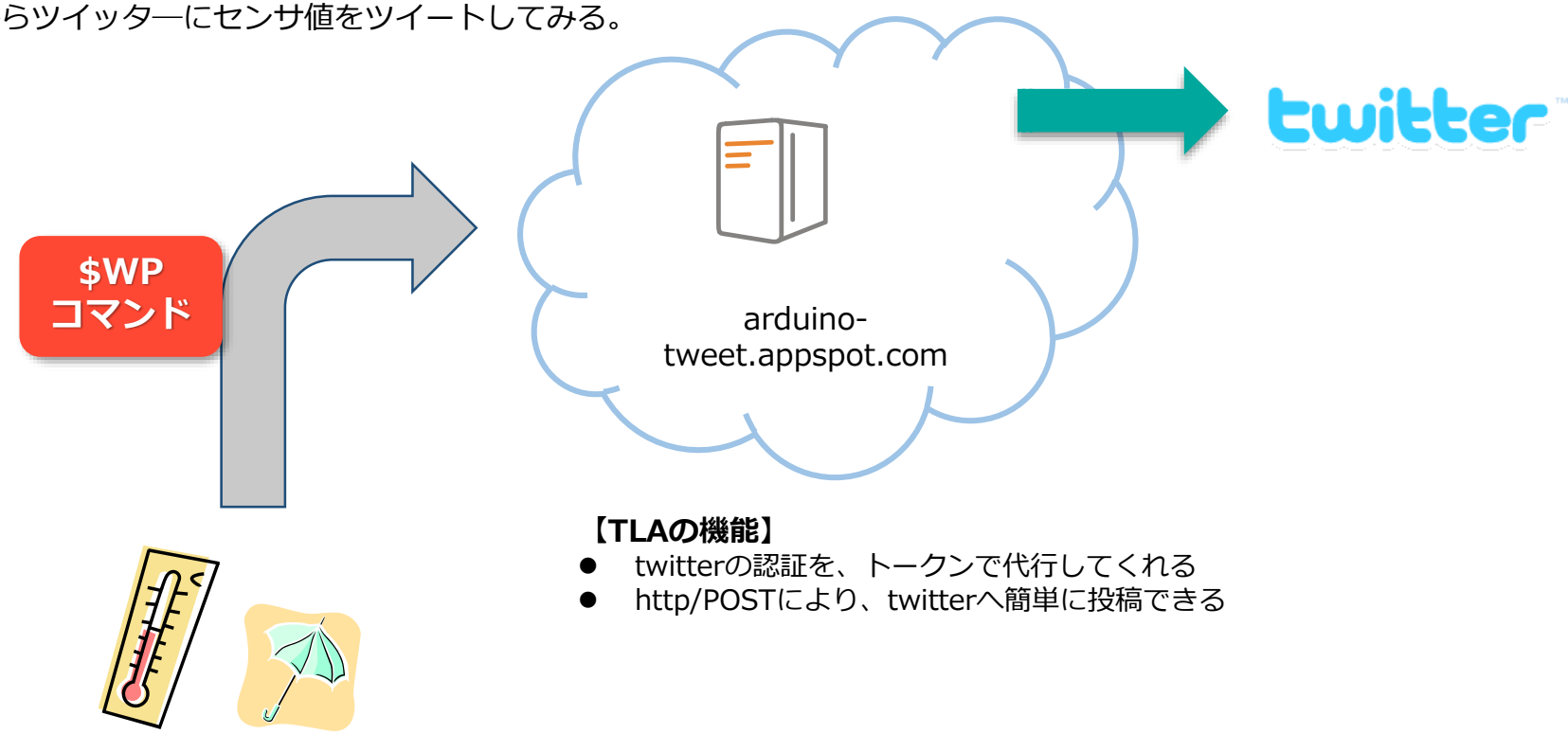


２．４GIMでのツイッター連携使用例（Arduinoの事例）

1．４GIM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、４GIMからツイッターにセンサ値をツイートしてみる。



【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/

画面のハードコピーイメージや登録内容等は、2016年4月時点のものです。その後変更されている可能性がありますが、ご了承ください。

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）

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）

4．4GIM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

スケッチ名：twitter_sample.ino

```
#include <SoftwareSerial.h>
SoftwareSerial Serial4G(4,5);
#define INTERVAL 60000UL   //ツイートは1分毎とする（制限事項）
String URL   = "http://arduino-tweet.appspot.com:80/update";
String TOKEN = "token=<YOUR-TOKEN>&status=";

void setup() {
  Serial.begin(38400);
  Serial4G.begin(38400);
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH);
  delay(100);
  digitalWrite(7,LOW);
  Serial.println("Ready...");
  while (Serial4G.readStringUntil('¥n').indexOf("４GIM") < 0);
 Serial.println("Start");
}
```

ツイートのトークン

```
void loop() {
  static uint32_t tm = millis();
  String dtm;
  do{  Serial4G.println("$YT");
    while(!Serial4G.available());
       dtm = Serial4G.readStringUntil('¥n');
  } while(dtm.indexOf("$YT=OK")<0);
  dtm= dtm.substring(7);
  dtm.replace(" ","%20");
  String body = TOKEN  +  dtm + "%20light=" + String(analogRead(A0));
  String message = "$WP " + URL + " ¥"" + body + "¥"";
  Serial.println(message);
  Serial4G.println(message);
  String str;
  do{
     while(!Serial4G.available());
     str = Serial4G.readStringUntil('¥n');
  } while(str.indexOf("$WP")<0);
  Serial.println(str + "¥r¥n wait " + String(INTERVAL/1000) + "sec");
  while(millis() - tm <INTERVAL);
  tm = millis();
}
```

光センサ（アナログA0)をアップ

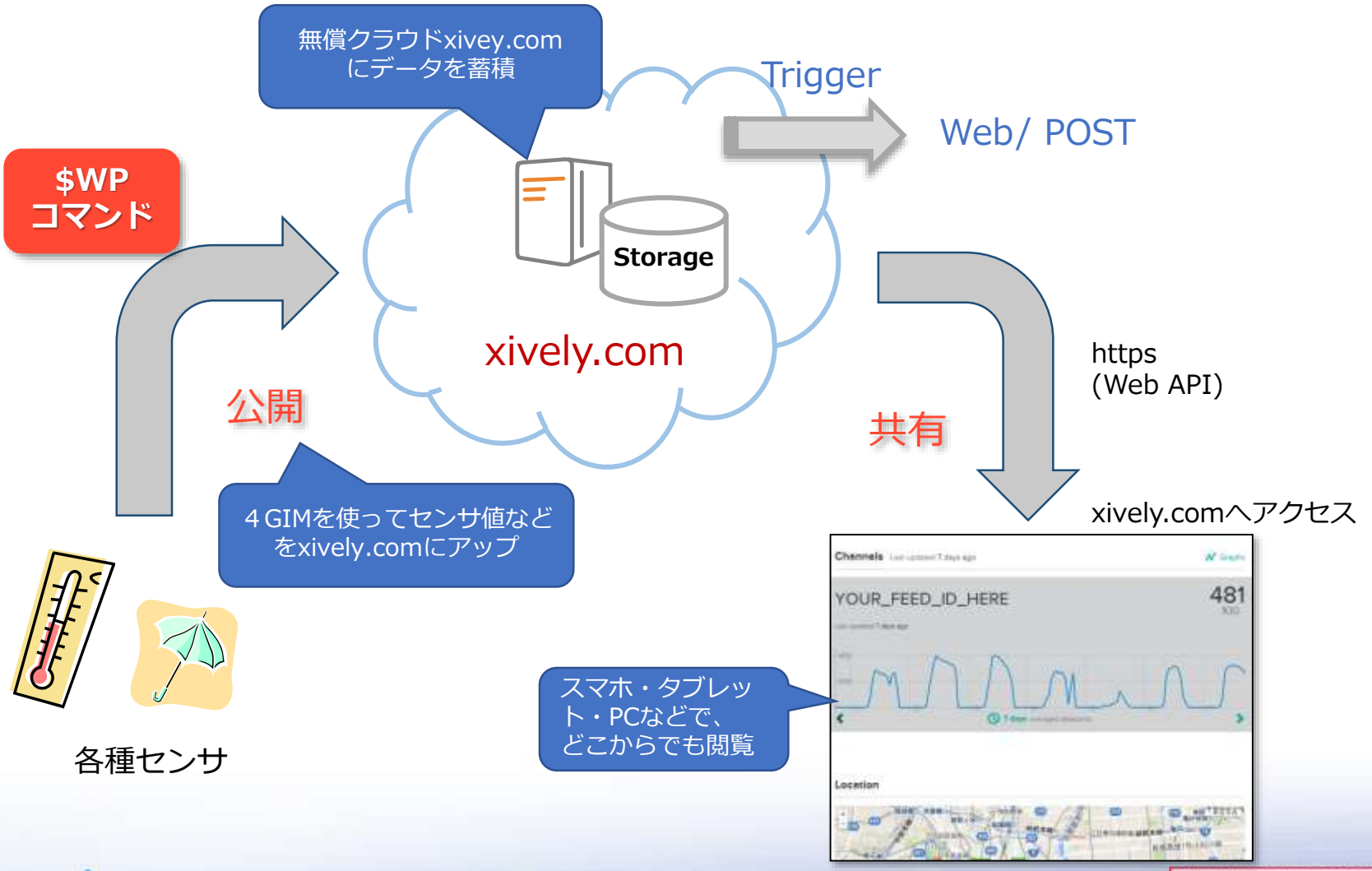
ツイートされた画面

■実行モニタ画面（正常時）

```
Ready…
Start
$WP ************
$WP=OK 2
 wait 60sec
```

３．４GIMでのクラウド連携使用例①
（xively.com ＆ Arduinoの事例）

1．xively.comの利用イメージ



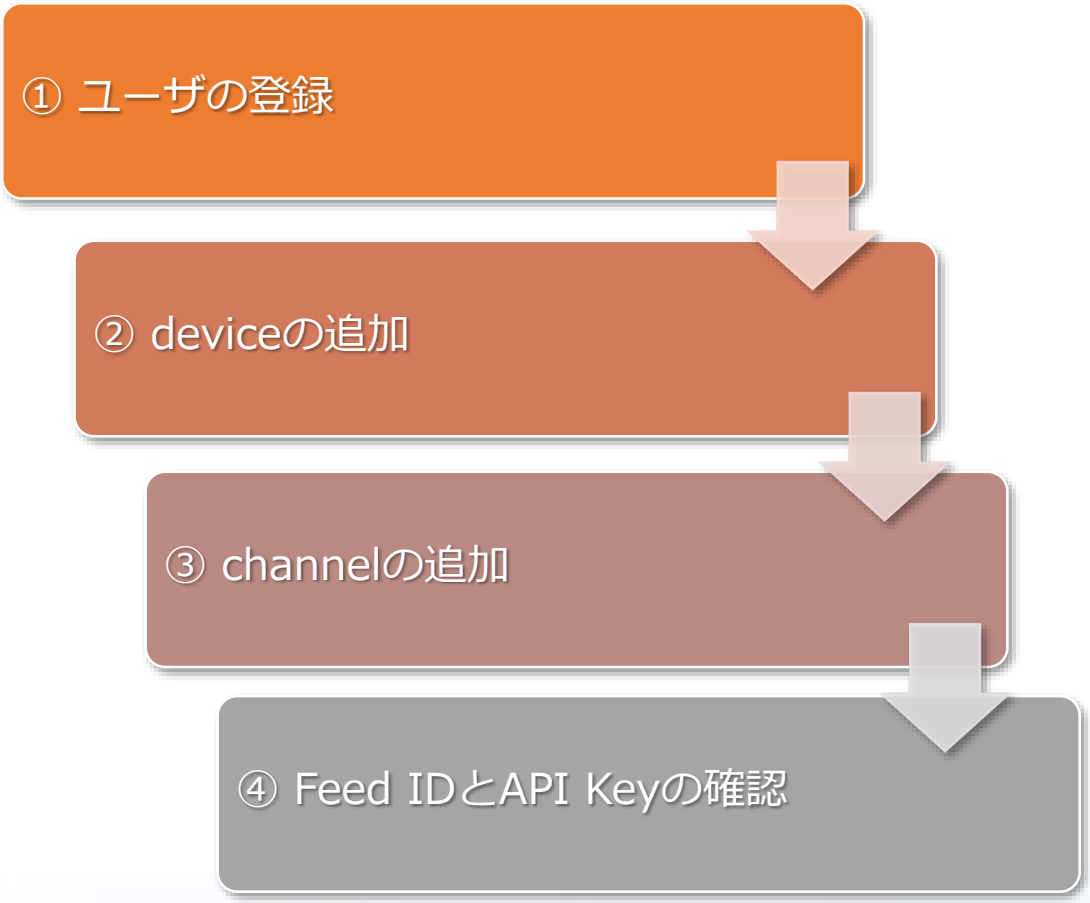
2．xivey.comの利用手順

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４GIM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。

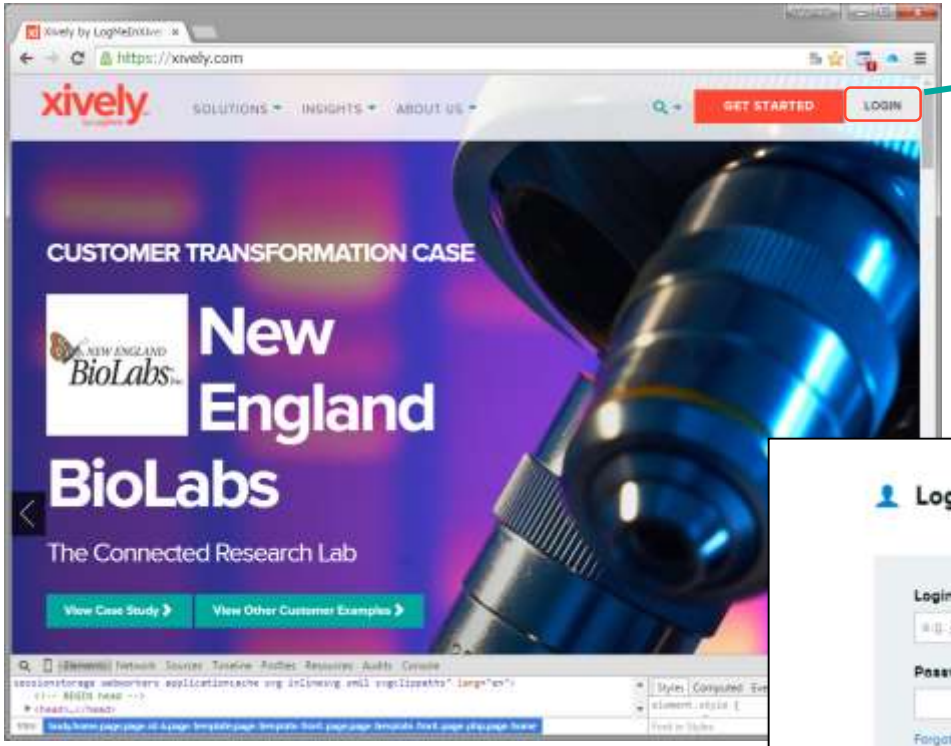
先ずは、xively.comでの①ユーザ登録が必要で、その後各設定（②deviceの追加と③channelの追加）を行い、それら設定された値（④Feed IDとAPI Keyの確認）を使うことで、プログラミングしていきます。



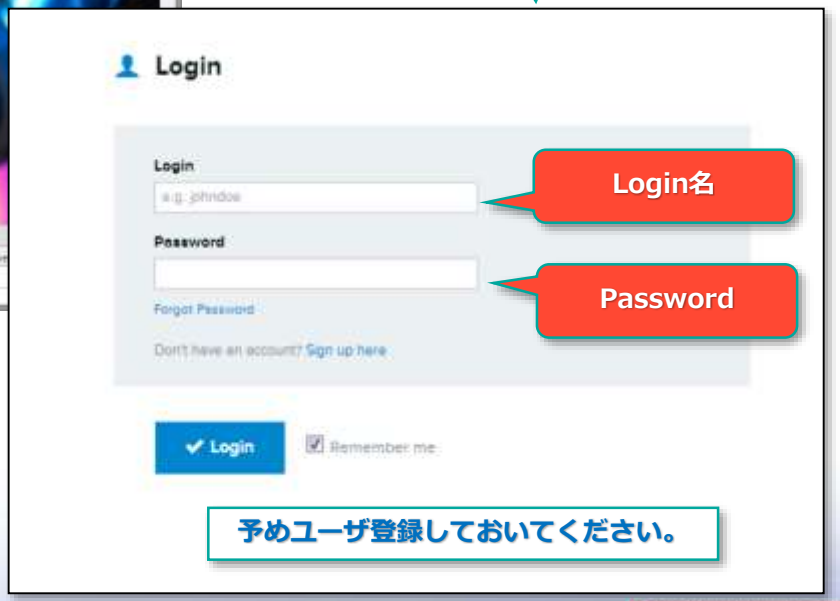
【利用に当たっての注意点】
xively.comでは、無償の範囲での利用は、制限があります。特にデータのアップは、1分間に数回程度でしかできません。
1秒毎とか頻繁にデータをアップしたりすると、利用できなくなることがあります。
充分に気を付けてプログラミングしてください。

3．xively.com ①ユーザ登録

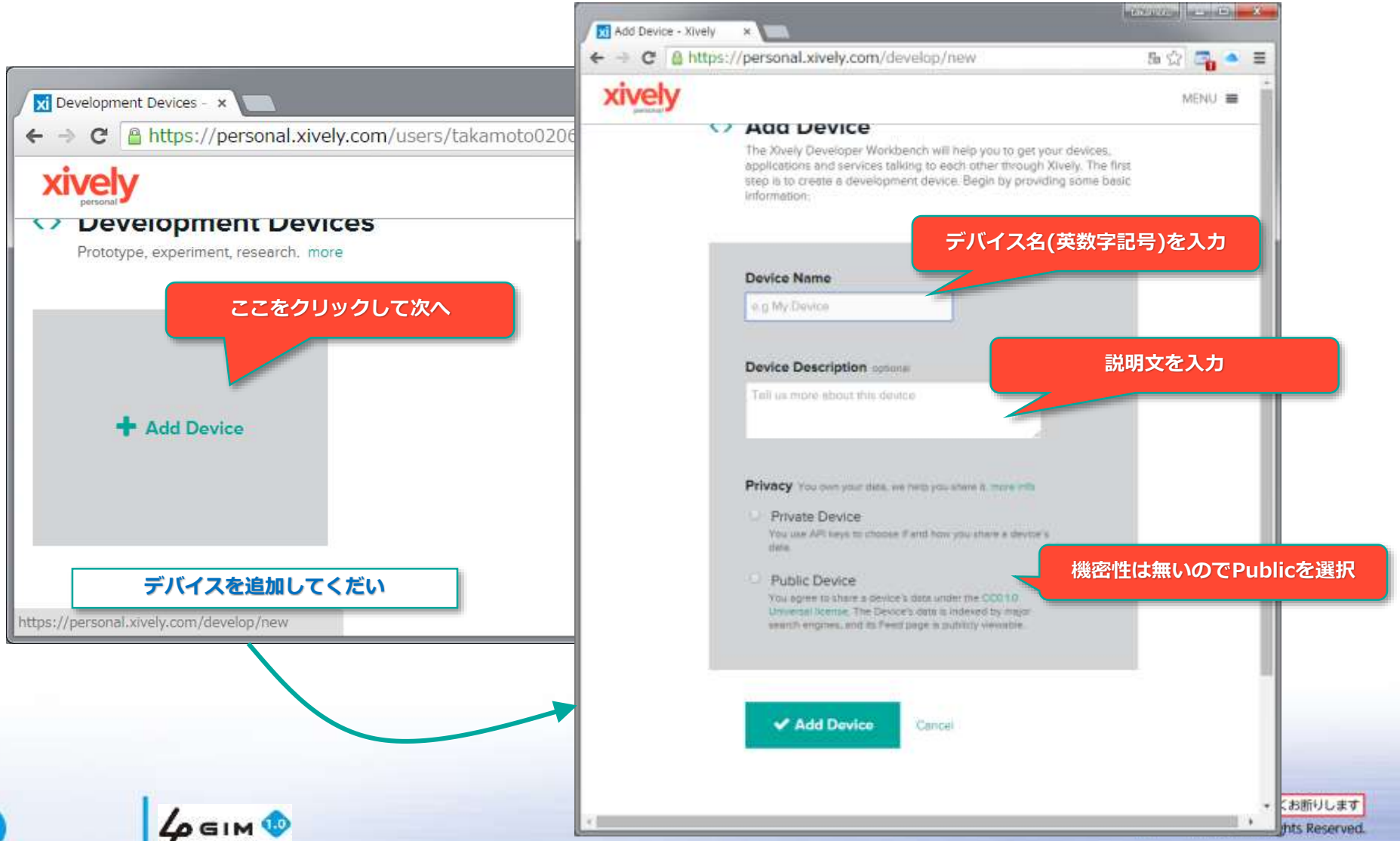
画面のハードコピーイメージや登録内容等は、2016年4月時点のものです。その後変更されている可能性がありますが、ご了承ください。



http://xively.com/ にアクセス

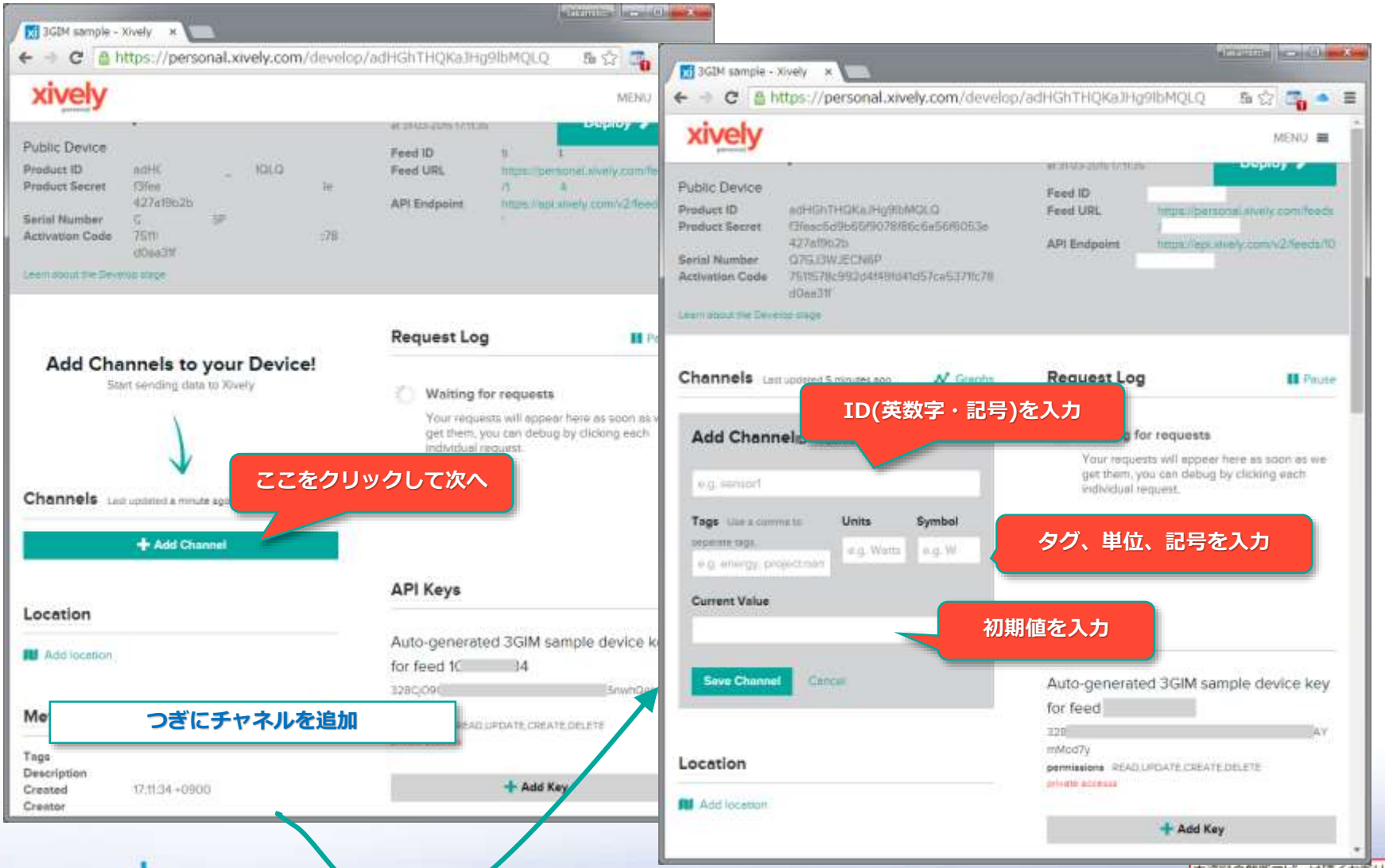


4．xively.com ②deviceの追加

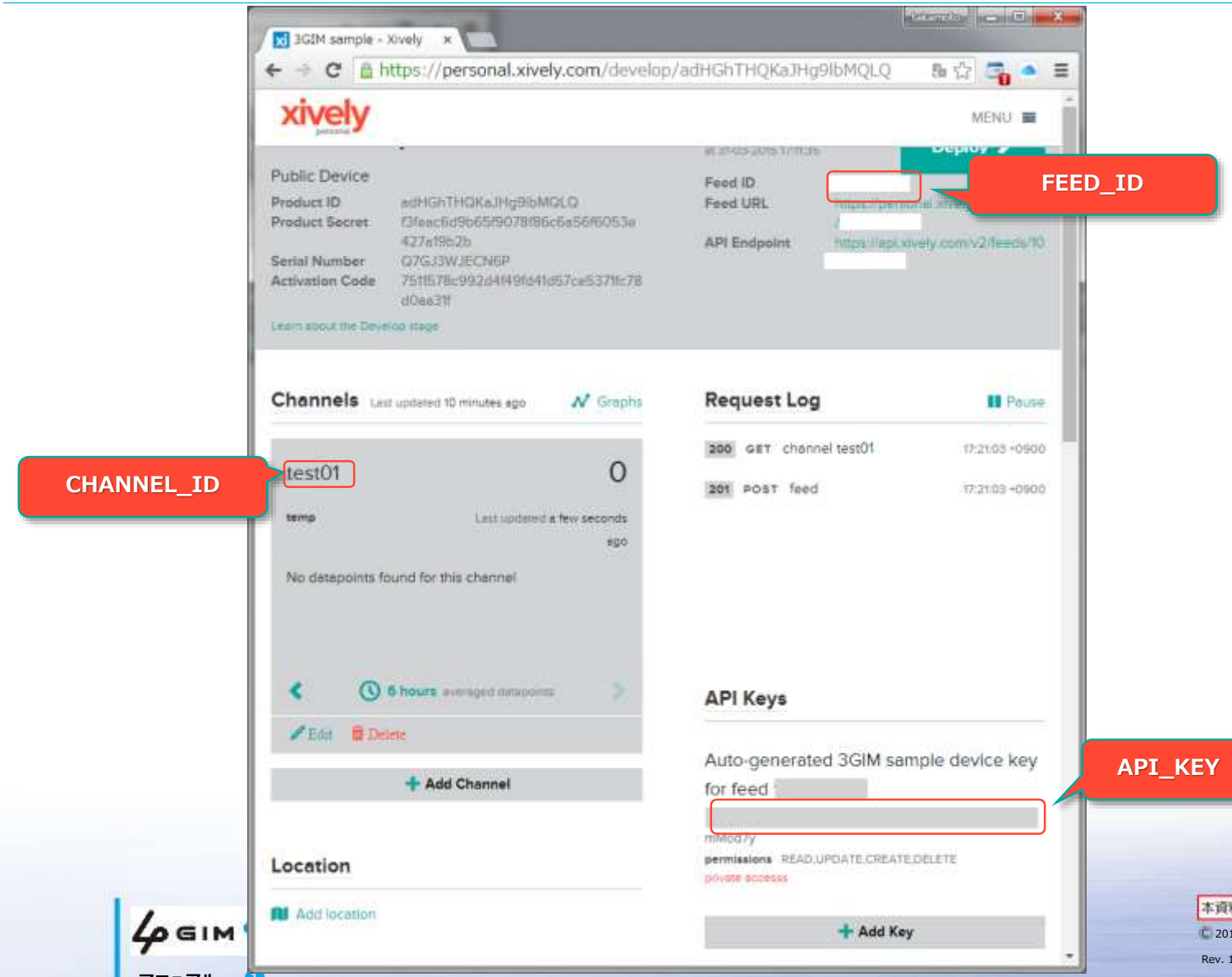


Rev. 180129

5．xively.com ③chenellの追加



6．xively.com ④Feed IDとAPI Keyの確認



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）
  - ４GIM（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（４GIM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**
  - ４GIM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

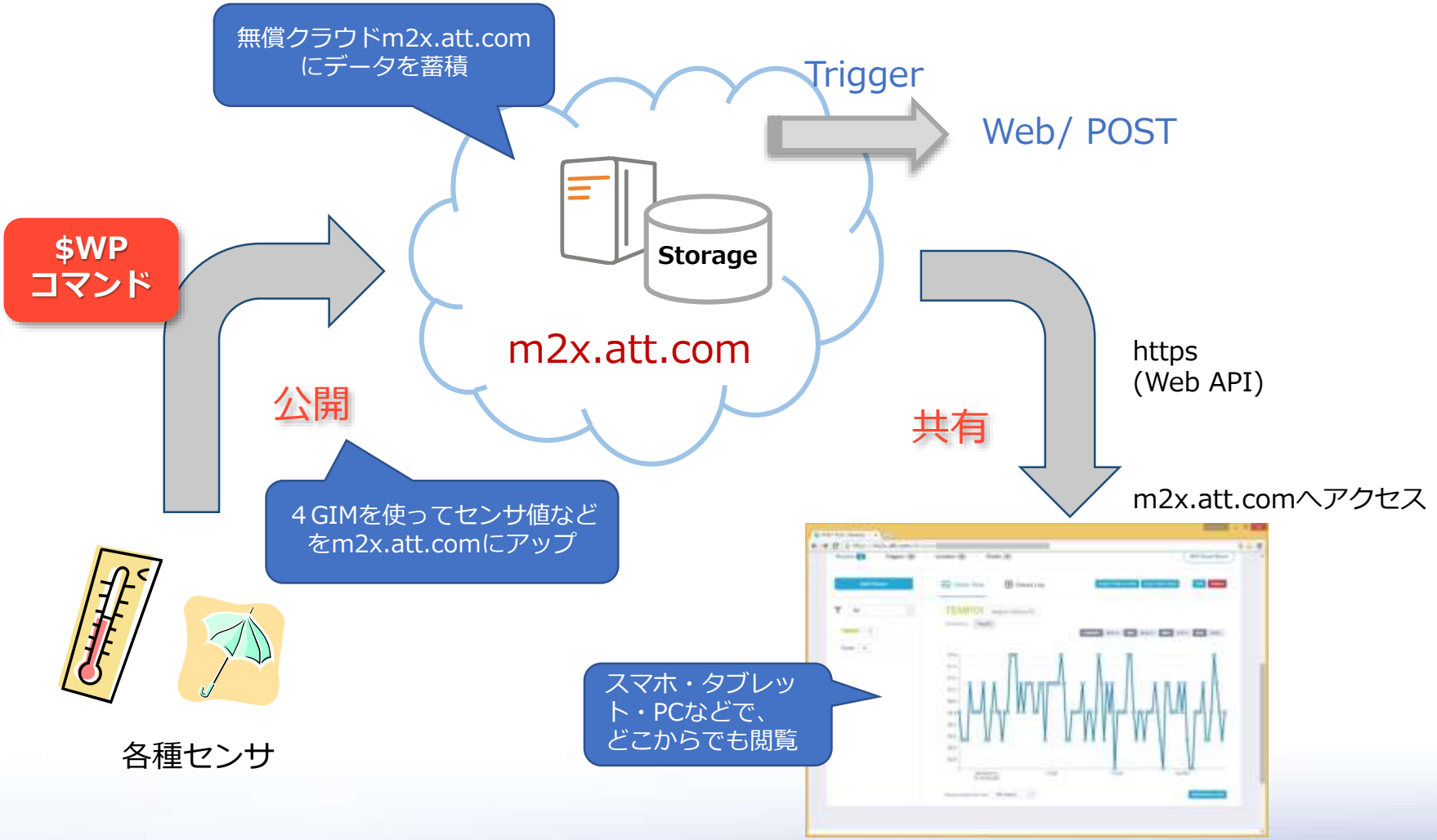
赤文字の箇所は、実際のxively.comの登録内容に沿って修正すること！

```
//  Sample sketch for  4 GIM

#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  Serial4G(4, 5);
char body[20];

// setup() -- set up device
void setup() {
   pinMode(PowerPin, OUTPUT);
   digitalWrite(PowerPin, LOW);    //  4 GIM on
   Serial4G.begin(38400);
   delay(35000);    // wait for start up  4 GIM
}
void loop() {
   // Sense temperature
   int tX10 = getTemperature() * 10;
   // upload sensing data to the xively.com
   uploadToCloud(tX10);
   // sleep a while
   delay(interval);
}
```

```
float getTemperature() {
   int mV = analogRead(tmpPin) * 4.88;
   return ((float)(mV - 600) / 10.0);
}

void uploadToCloud(int temp) {
   // upload temperature
   Serial4G.print(PostCmd);
   sprintf(body, "¥"%d.%d¥" ", (temp / 10), abs(temp % 10));
   Serial4G.print(body);
   Serial4G.println(Header);
   Serial4G.flush();
}
```

４．４GIM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

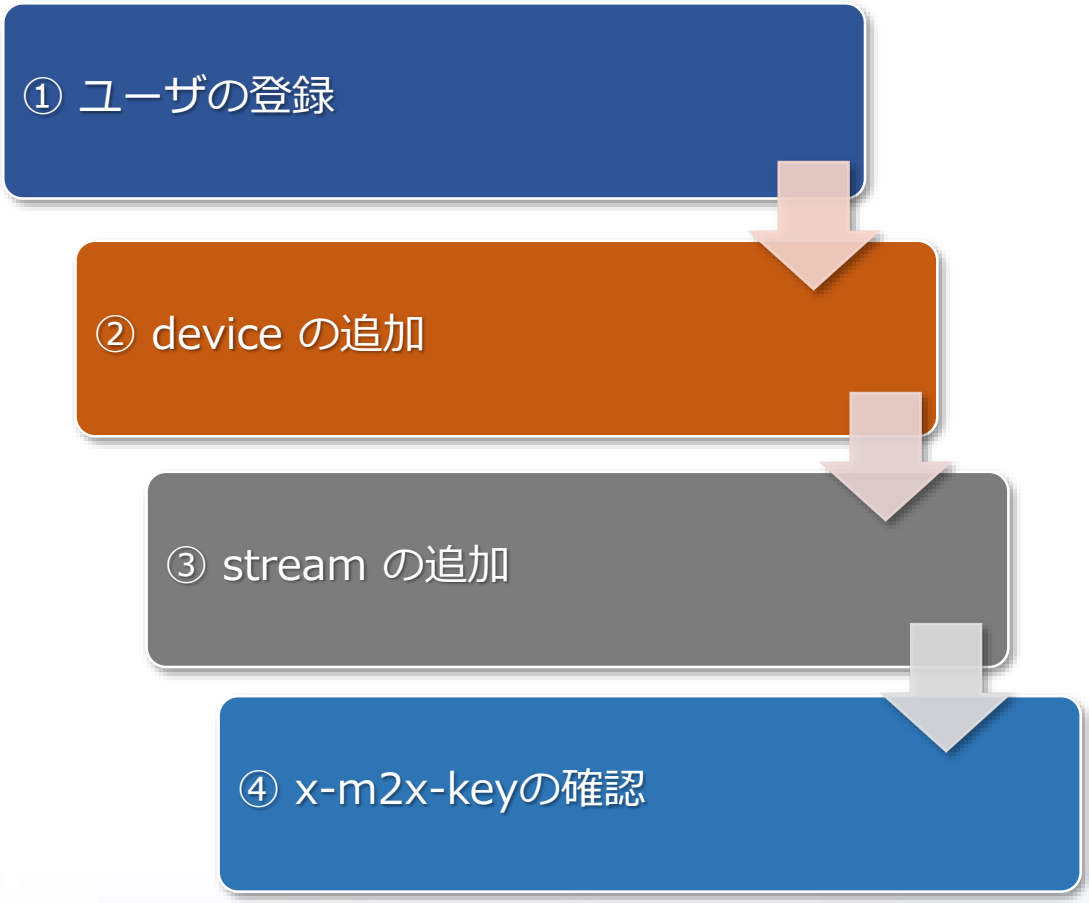


2． M2X（AT&T IoTサービス）の利用手順

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RasberryPi、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。

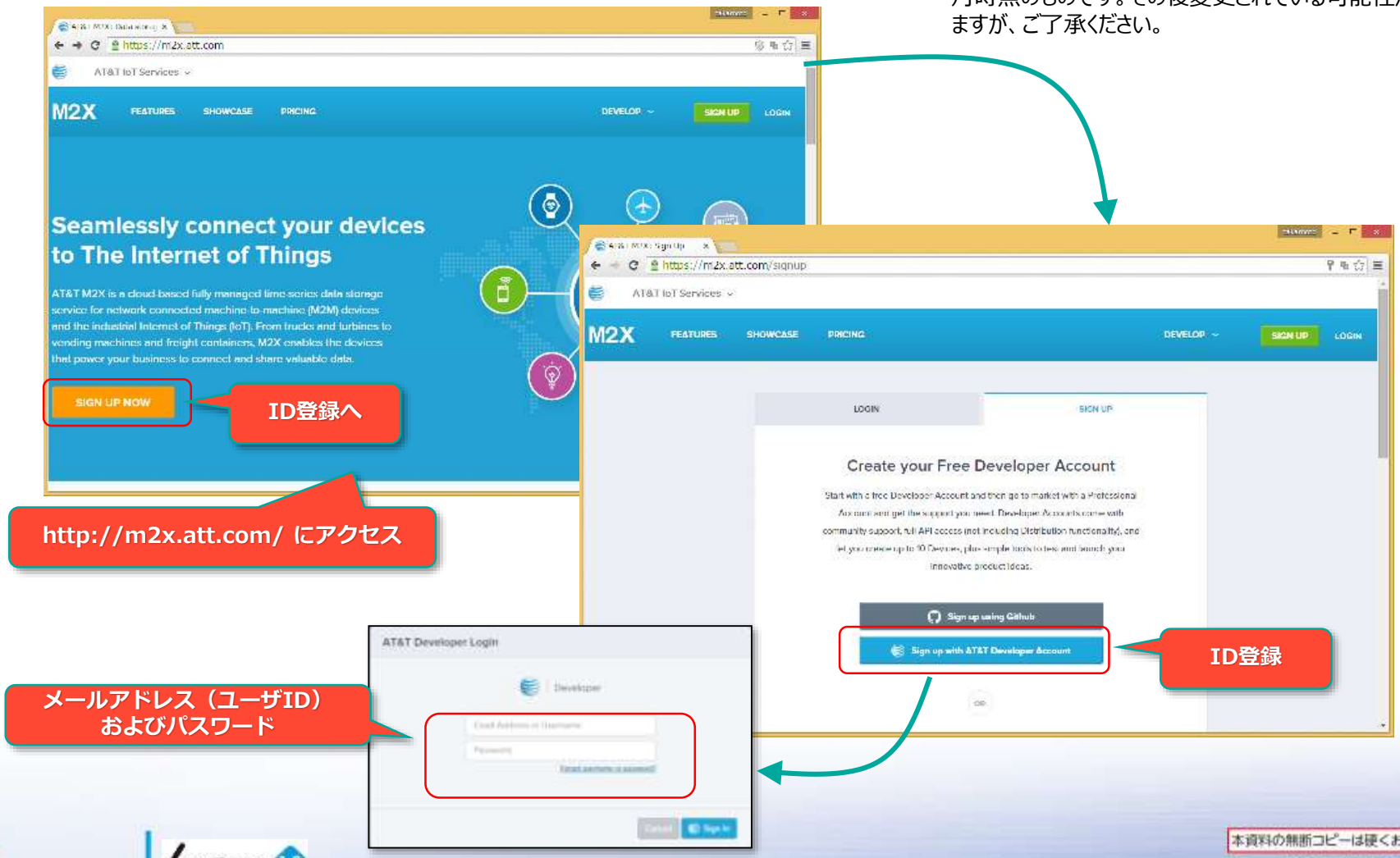
ここでは、本3Gシールドとセンサを使い、このm2x.att.comにデータをアップしていくサンプルをご紹介します。

詳細な規約等は、m2x関連の公開情報等をご参照ください。

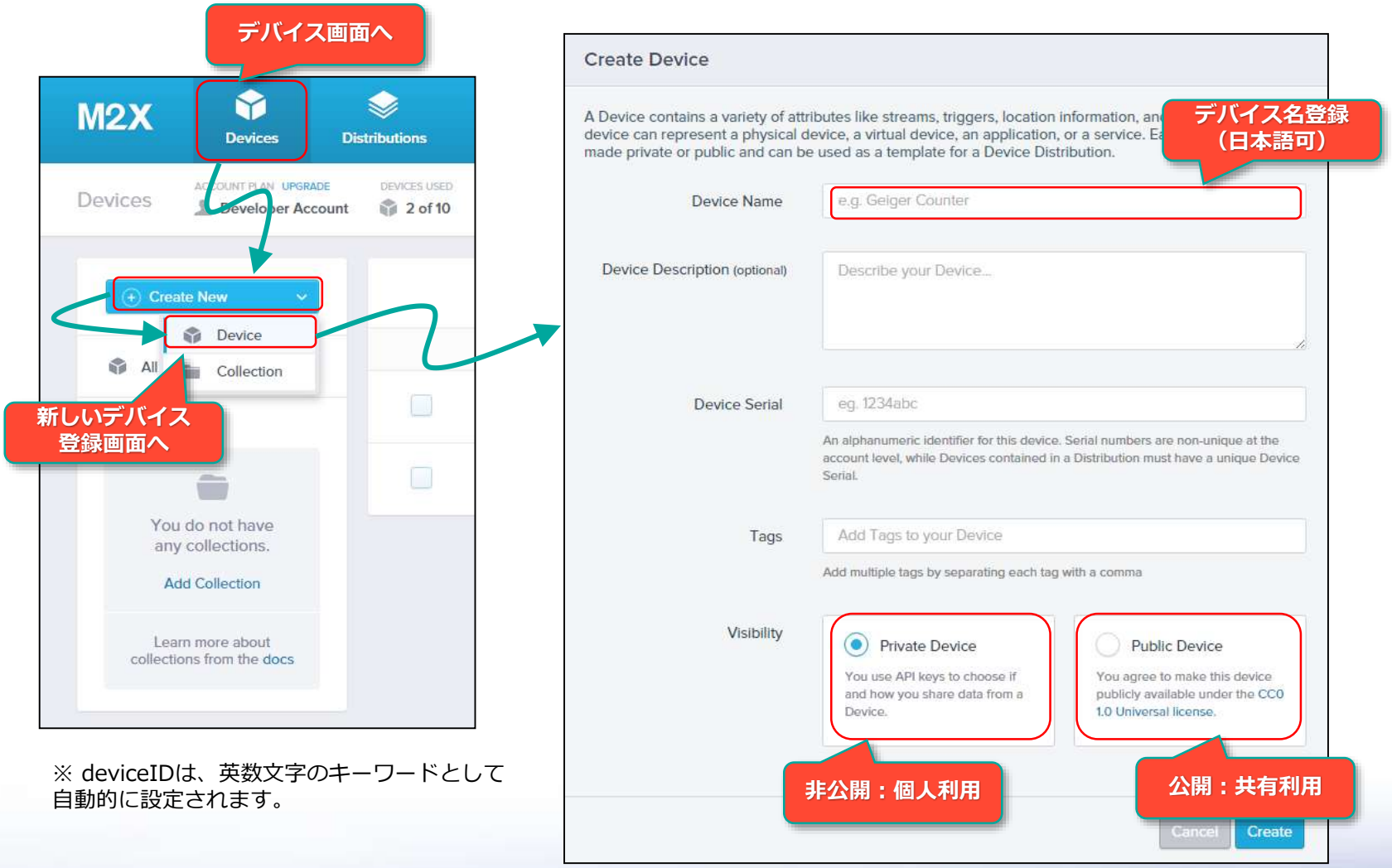


3． M2X（AT&T IoTサービス）のID登録

画面のハードコピーイメージや登録内容等は、2016年4月時点のものです。その後変更されている可能性がありますが、ご了承ください。



4．デバイス（Device）の作成登録

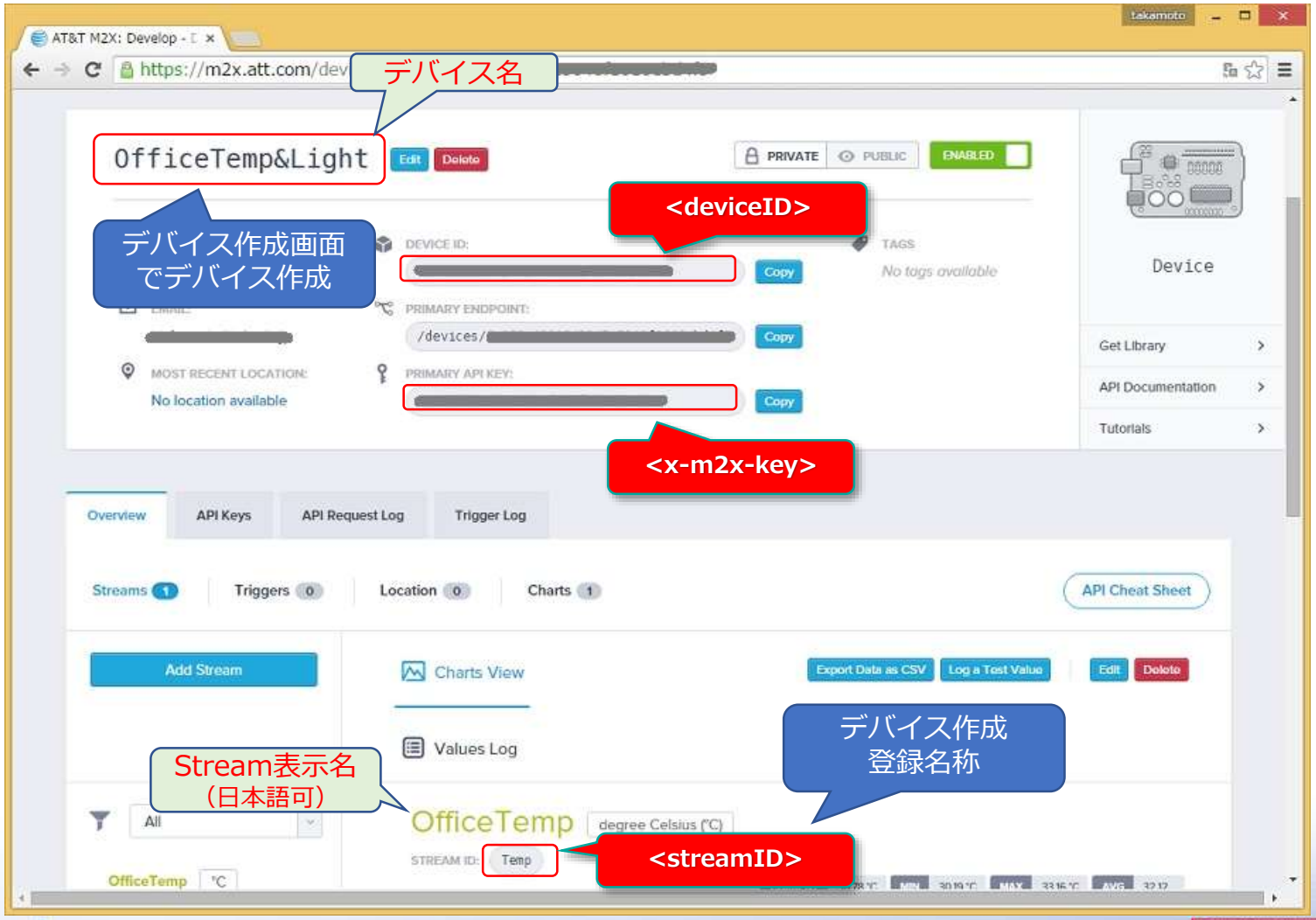


※ deviceIDは、英数文字のキーワードとして自動的に設定されます。

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



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>**rnContent-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 “2015-09-20**T**23:55:36**$+**09:00”（$+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

スケッチ名：m2x_sample.ino

```
#include <SoftwareSerial.h>
SoftwareSerial Serial4G(4,5);
const unsigned long baudrate = 38400;
#define INTERVAL  180000UL // 3分間隔でデータアップ
#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);
  Serial4G.begin(baudrate);
  pinMode(7, OUTPUT);
  digitalWrite(7, HIGH);
  delay(100);
  digitalWrite(7, LOW);
  Serial.println("Ready");
  while (Serial4G.readStringUntil('¥n').indexOf("4 GIM") < 0);
  Serial.println("Start");
}

void loop () {
  static uint32_t tim = millis();
  String dtime;
  while ( (dtime = datetime())== "") ;
  Serial.println(dtime);
  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:" + Serial4G.readStringUntil('¥n'));
  }
  else Serial.println("Data Update false...");
  while(millis() - tim < LIMITTIME);
  tim = millis();
}
```

url,header,body の設定

setup 3G初期化

温度センサ値を3分間隔空けてM2Xにアップ

8．サンプルプログラム②

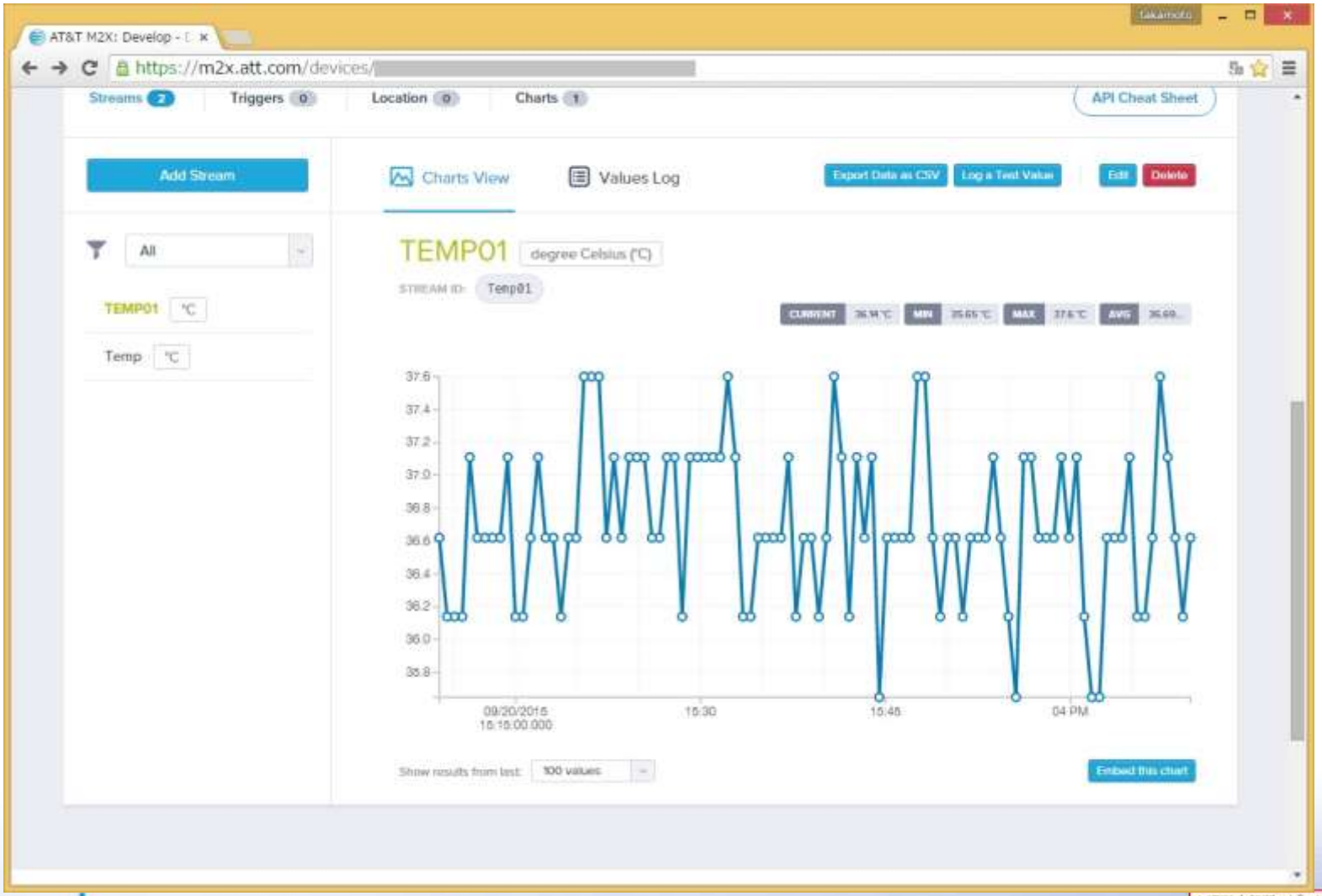
```
//=============== $WP command =================
boolean _3G_WP(String command) {
  Serial.println(command); // debug
  Serial4G.println(command);
  String rstr;
  unsigned long tim = millis();  // time set(ms)
  do {
    while (!Serial4G.isListening());
    rstr = Serial4G.readStringUntil('¥n');
    Serial.println(rstr);  //debug print....
  } while (!(rstr.indexOf("$WP=") == 0) && (millis() - tim) < LIMITTIME); // $WP return check
  return (rstr.indexOf("$WP=OK") == 0);
}
// Get Date & Time ( 4 GIM command)
//  return --> string "2015-12-23T01:23:45%2B09:00"
String datetime() {
  Serial4G.println("$YT");
  while (!Serial4G.available());
  String dtime = Serial4G.readStringUntil('¥n');
  Serial.println(dtime);
  if (dtime.indexOf("NG") > 0) return "";
  dtime.replace(" ", "T"); dtime.replace("/", "-");
  return (dtime.substring(7) + "$+09:00");
}
```

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



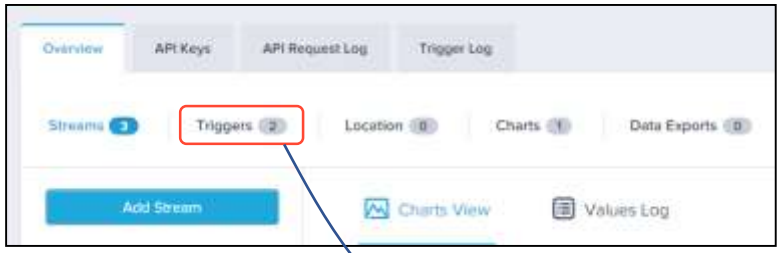
１０．　M2Xからトリガーでツイートする方法

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

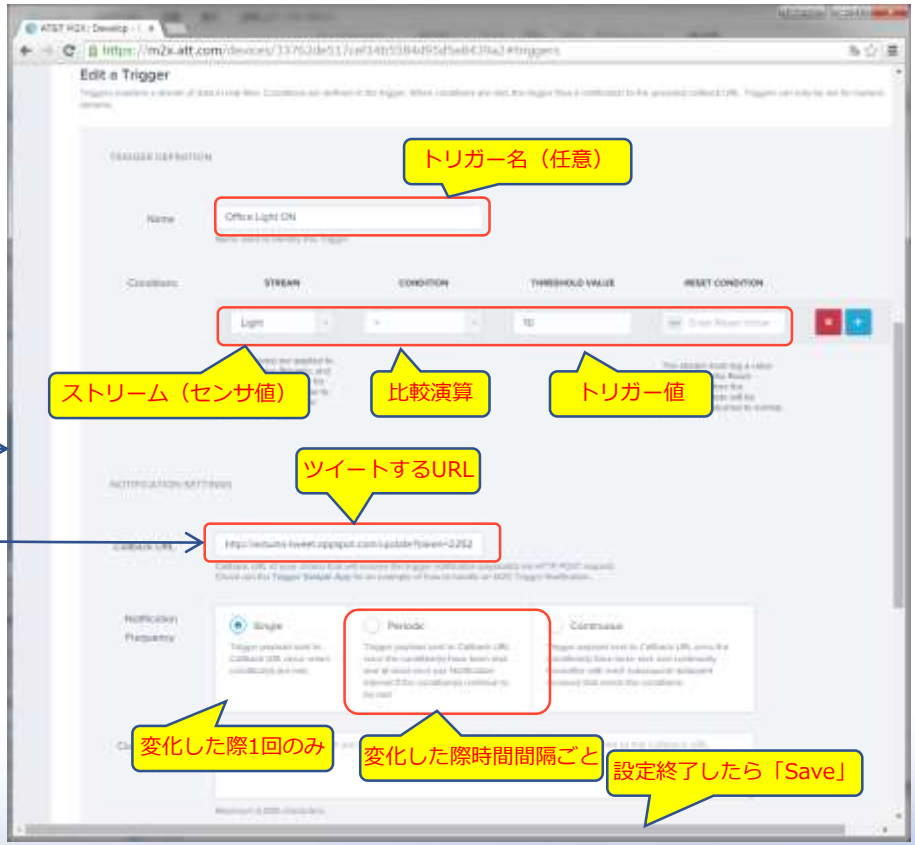
ツイートするURLは、以下の通り

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

トリガー設定の選択



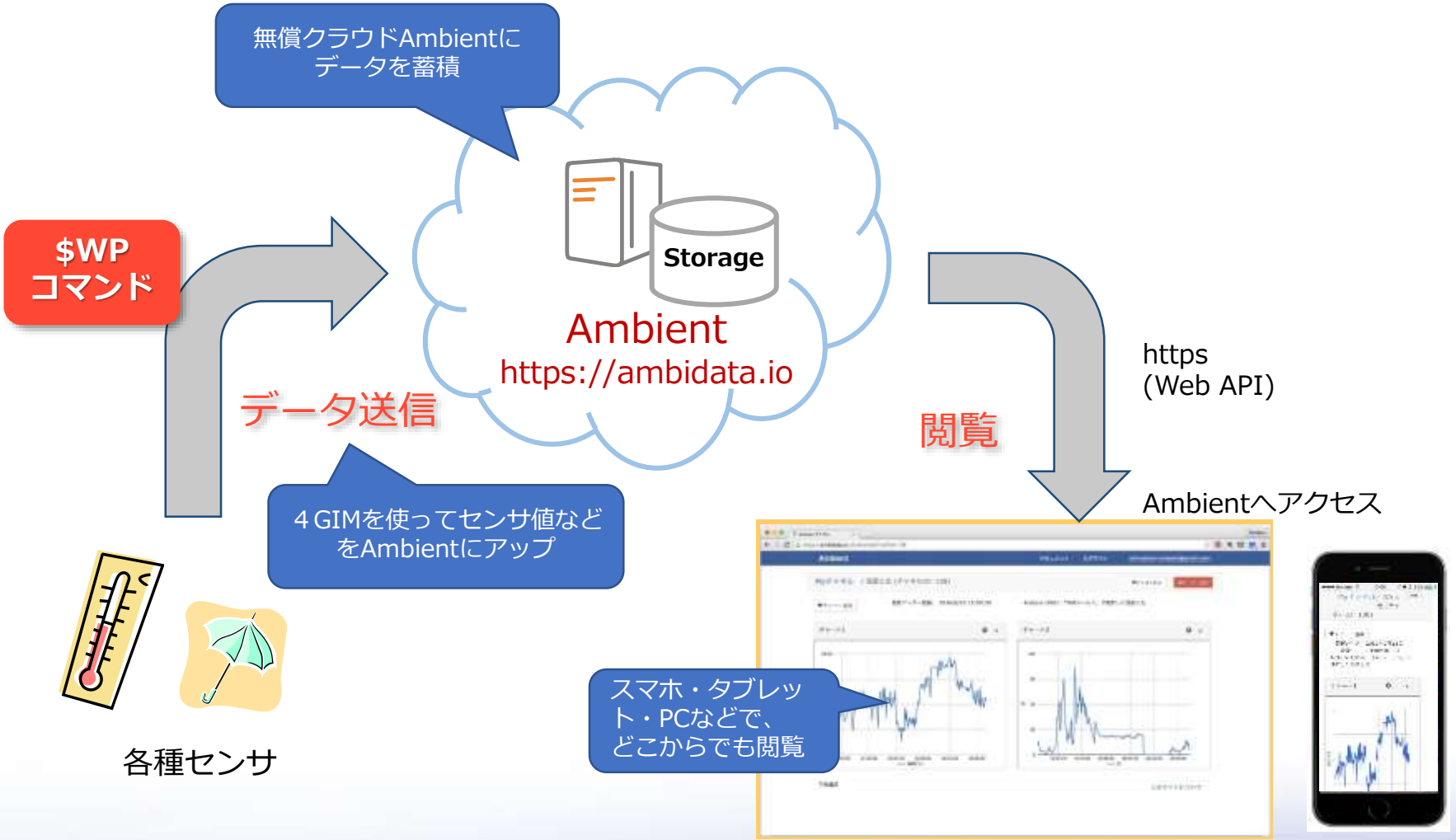
トリガーの設定



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。

５．４GIMでのクラウド連携使用例③
（Ambient ＆ Arduinoの事例）
＜現在未対応：接続予定＞

1．Ambientの利用イメージ

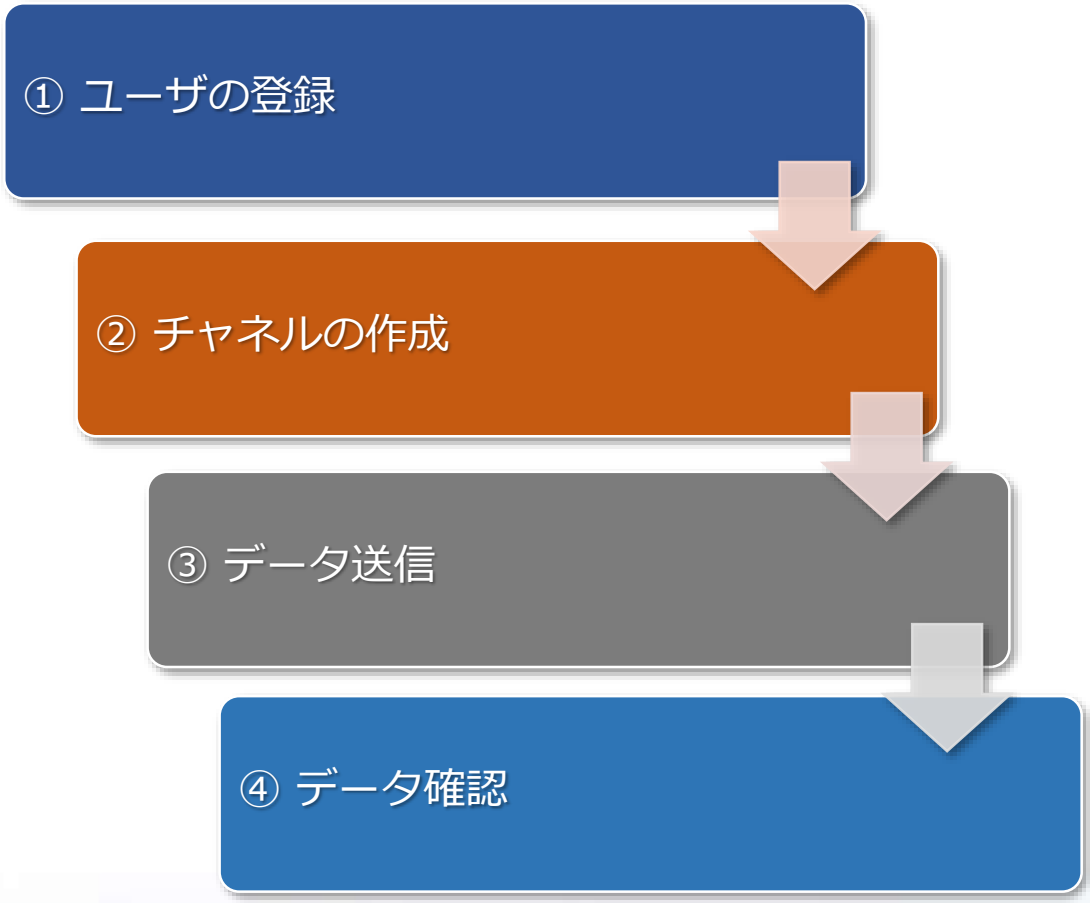


2． Ambientの利用手順

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。

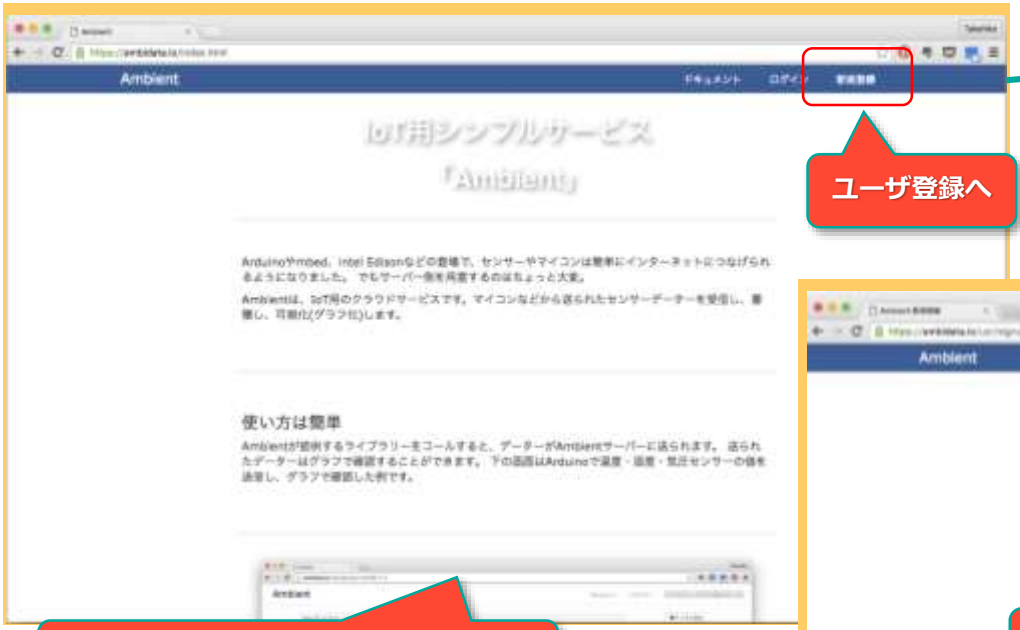
ここでは、本４GIMとセンサなどを使い、このAmbientにデータをアップしていくサンプルをご紹介します。

詳細な規約等は、Ambient関連の公開情報等をご参照ください。



3． Ambientのユーザ登録

画面のハードコピーイメージや登録内容等は、2016年4月時点のものです。その後変更されている可能性がありますが、ご了承ください。

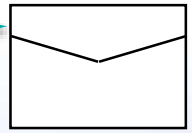


ユーザ登録へ

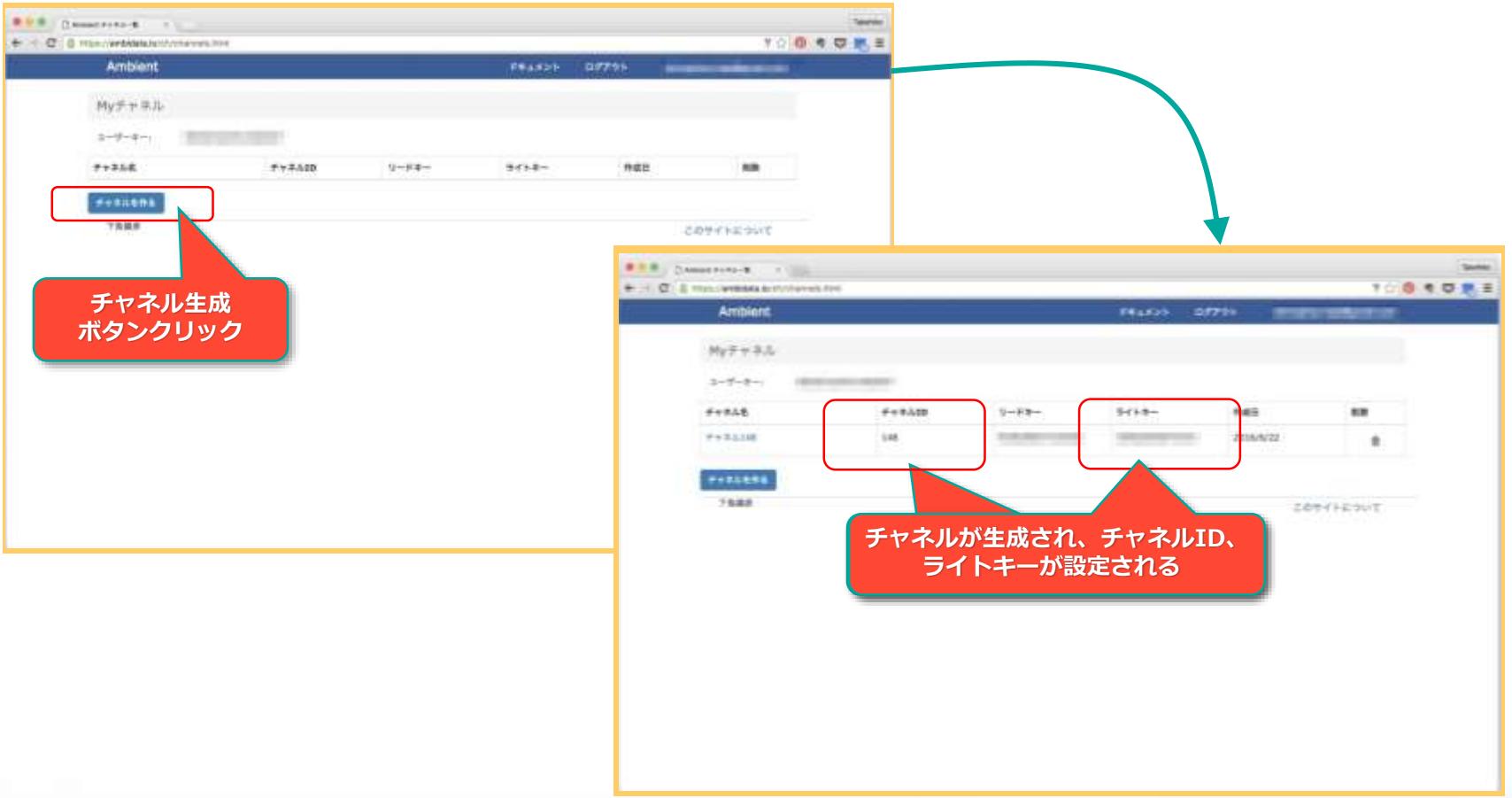
https://ambidata.io/ にアクセス

メールアドレス（ユーザID）
およびパスワード

確認メール



4．チャネル作成



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(チャネルID, ライトキー, SoftwareSerial *s);
初期化関数。チャネル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。
データー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 "Ambient4GIM.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("4GIM")>0) && (millis() - tim) <LIMITTIME);
  delay(1000);
  Serial.println(str);
  if( millis() -tim >= LIMITTIME) {
     return false;
  } else return true;
}
```

4GIM用Ambientライブラリ
ヘッダーファイル

チャネル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4GIM文字
が返却されるまで待機

3G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

データをパケットにセットし、送信

7．Ambientにデータアップした事例

8．チャネルとグラフをカスタマイズした事例

第５章
Arduino用a4gimライブラリ群

目次

１．a4gimライブラリとは

1．a4gimとは

a4gimとは、Arduinoやその互換ボード上で、４GIM V1.0を簡単に利用できるようにしたライブラリです。

これらのライブラリ群は、前述したさまざまな「＄コマンド」の利用・設定を簡単な関数呼び出しで利用できるようにしたライブラリで、４GIMの詳細を知ることなく簡単に利用することができます。

このライブラリは、次ページ以降に記載する通り、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ように環境を設定します。

※ a4gim2ライブラリも併せて提供しています。a4gimライブラリは主にUNO用となっており、４GIMとの通信にはSoftwareSerialライブラリを使用します。一方、a4gim2ライブラリは、ハードウェアシリアル(Serial/Serial1/Serial2..)を使用する仕様となっており、MegaやDue、ZeroやM0、101等のハードウェアシリアルを複数持つArduinoで使用することを想定しています。両方のライブラリは、常に同時に同じバージョンで公開していく予定です。

注意：４GIM V1.0では、 a4gim のバージョンは、R4.3 となります。

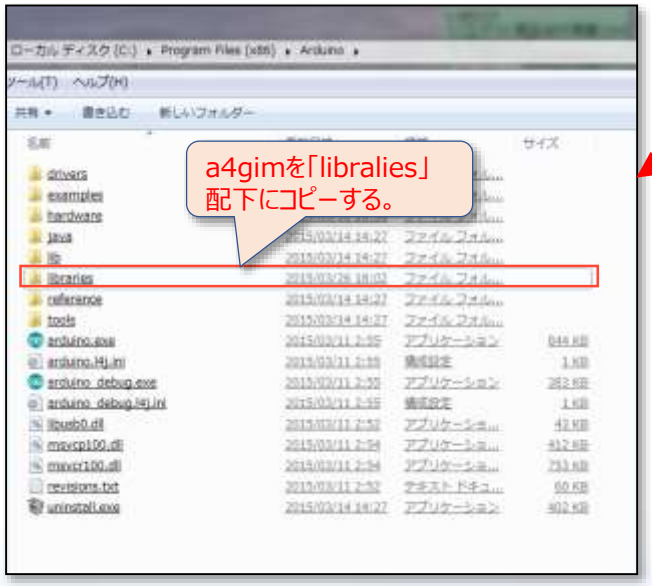
a4gim Version	４GIM V1.0のライブラリが利用できるボード	内部電圧
a4gim　R1.0	Arduino UNO R3	5V系
a4gim2　R1.0	Arduino Mega,Leonardo,Zero Pro, M0 Pro、Genuino101ほか	3.3V系又は５V系

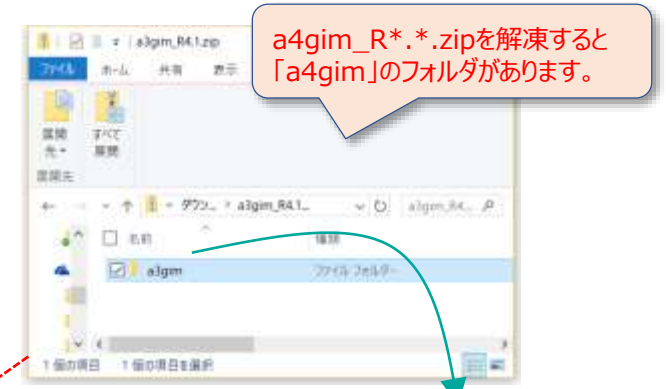
2．ライブラリa4gim.zipのダウンロード

1）このZIPファイルを、以下のサイトから**ダウンロード**してください
https://3GIM.wiki/doku.php?id=downloads

2）このzipファイルを、以下のいずれかのフォルダーに解凍してください。
① Arduino IDE環境下の「‥¥libraries」配下
② スケッチブックの保存場所（環境設定）の「..¥libraries」配下

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、「a4gim」が表示されます。
「a4gim2」ライブラリのインストールも、必要に応じて同様に行ってください。





ダウンロード先：
下記のページから最新版をダウンロードしてください。
https://3GIM.wiki/doku.php?id=downloads

3．a4gimライブラリを利用する方法

- **概要**
  - 提供するa4gimライブラリ機能は、以前の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４GIMとの接続を工夫することで、下記のライブラリを使用することができます。
    ＜本バージョンR1.0から「４GIM用ライブラリ」と呼びます＞
    - a4gim　UNO/Pro用（SoftwreSerialを使用）：４GIM専用に改訂
    - a4gim2 Mega/M0/Zero/101用（Serial1を使用）：４GIM専用に改訂

- **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)
    - a4gim.hのシンボル a4gsBAUDRATE の定義は、「115200」となっています。
    - a4gim2.hのシンボル a43gsBAUDRATEの定義は、「115200」となっています。

- **４GIMとArduinoとの接続方法（例）**
  - **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
  - **Zero（M0) Pro/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。

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

4．a4gim R1.0ライブラリー覧表

分類	メソッド名	機能概要	補足
コントロール（Control）	begin※	ライブラリの初期化	
	end※	ライブラリの終了	
	restart※	４GIMのリセット	
	start※	４GIMの電源ON	
	shutdown※	４GIMの電源OFF	
	getServices	現在使用できる通信サービス	
	getIMEI	IMEI-IDの取得	
	setBaudrate	UARTの通信速度の設定	初期は115200bps
	setLED	LED1のON/OFF	
	setAirplaneMode	エアプレーンモードのON/OFF	
インターネット関係（Web）	httpGET※	GETメソッドの要求	http/https利用可能
	httpPOST	POSTメソッドの要求	同上
	tweet※	Twitterへの投稿	
通信機能その他	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	４GIMのバージョンの取得	
	getResult	４GIMからの応答値受信	
	sendCommand	４GIMへのコマンド送信	
	sendData	４GIMへのデータ送信	
	enterAT	ATコマンドパススルーモード	**＊仕様変更**
	discardUntil	４GIMからの文字の受信	
その他ライブラリ	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	
	setTCPParams	TCPパラメータ設定	**＊追加**
	writeBegin	シリアル通信で直接書込み	
	read※	データの読込み	2つのﾊﾞﾘｴｰｼｮﾝ有
	write※	データの書出し	3つのﾊﾞﾘｴｰｼｮﾝ有
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	

※ Arduino GSM/GPRSシールド用ライブラリと互換性がある関数
＊ **追加** ： V2.2で追加となったもの
＊ **仕様変更** ： V2.2で以前のものから仕様変更されたもの

２．a4gimライブラリ概説

1．ライブラリを利用するハードウェア

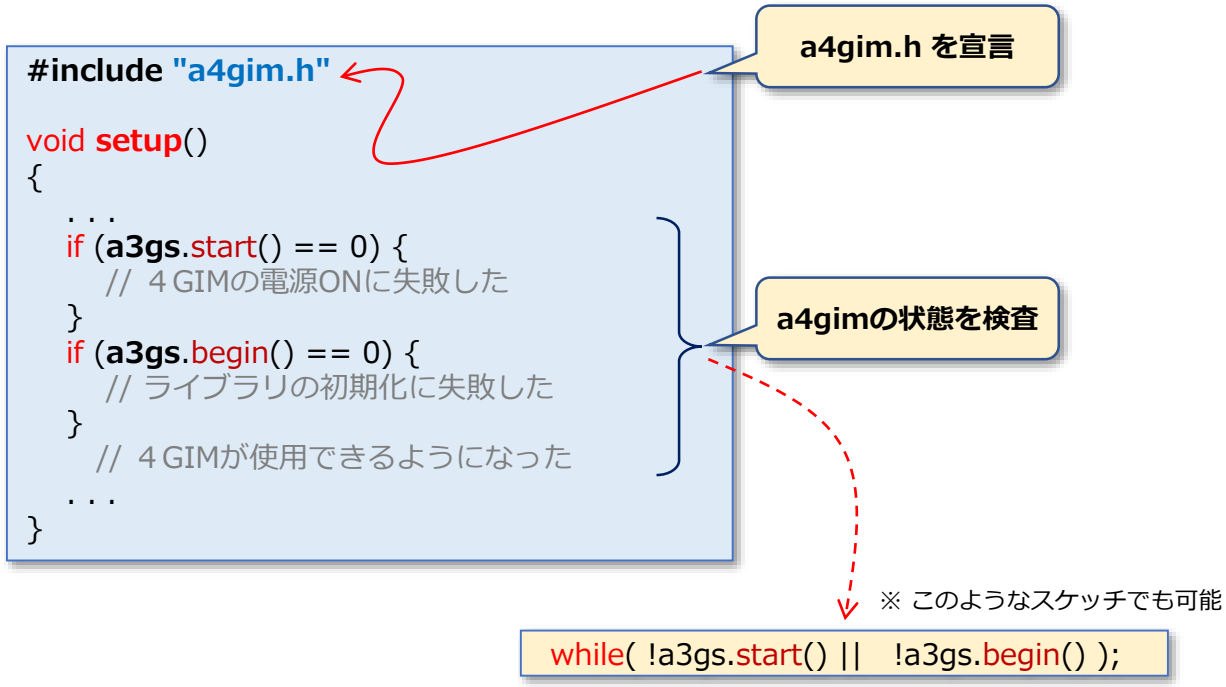
- Arduinoから４GIMを簡単に利用できるようにするために、ライブラリa4gimが提供されています。
  - a4gimライブラリは、Arduino UNO/Pro等でソフトウェアシリアルを使って４GIMを利用する際に使用します。
  - a4gim2ライブラリは、Arduino Mega/Due/M0 Pro/Leonardo/101/Zero等でハードウェアシリアル４GIMを利用する際に使用します。提供される機能は、a4gimと同等です。
  - 以下では、a4gim/a4gim2を総称してa4gimと呼びます。
  - ４GIM V1.0では、a4gim および a4gim2ともに バージョン R4.3を使用してください。
- 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４GIM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、Zero(M0) Pro、Genuino101、ArduinoMEGAなどの上で稼働できます。
- このa4gimはCPUアーキテクチャに依存しないライブラリであるため、Arduino互換機のMCUがAVRでもARMでも利用可能となっています。

IoTABシールドV3.0

３GIMシールドV2.0

2．ライブラリ概要

- a4gim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。



3．ライブラリ提供関数(1/3)

分類	メソッド名※1	機能概要	補足
コントロール（Control）	**begin**※	ライブラリの初期化	
	end※	ライブラリの終了	
	restart※	４GIMのリセット	
	start※	４GIMの電源ON	
	shutdown※	４GIMの電源OFF	
	getServices	現在使用できる通信サービス	
	getIMEI	IMEI-IDの取得	
	setBaudrate	UARTの通信速度の設定	**初期は115200bps**
	setLED	LED1のON/OFF	
	setAirplaneMode	エアプレーンモードのON/OFF	

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

3．ライブラリ提供関数(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	＊
通信機能その他	**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	４GIMのバージョンの取得	
	getResult	４GIMからの応答値受信	
	sendCommand	４GIMへのコマンド送信	
	sendData	４GIMへのデータ送信	
	enterAT	ATコマンドパススルーモード	
	discardUntil	４GIMからの文字の受信	

※　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．ライブラリ提供関数(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	
	setTCPParams	TCPパラメータ設定	
	writeBegin	シリアル通信で直接書込み	
	read※	データの読込み	2つのﾊﾞﾘｴｰｼｮﾝ有
	write※	データの書出し	3つのﾊﾞﾘｴｰｼｮﾝ有
プロファイル	**setDefaultProfile**	デフォルトプロファイルを設定	
	getDefaultProfile	デフォルトプロファイルを取得	

※　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“a4gim.h”で定義されています。

分類	定数名	意味	設定	補足
Web	a4gimMAX_URL_LENGTH	URLの最大バイト数	256	※1
	a4gimMAX_HEADER_LENGTH	POSTのヘッダの最大バイト数	512	※1
	a4gimMAX_BODY_LENGTH	POSTのボディの最大バイト数	1024	※1
	a4gimMAX_RESULT_LENGTH	GET/POSTのレスポンスの取得可能な最大バイト数	192	※1
	a4gimMAX_TWEET_LENGTH	ツイートメッセージの最大バイト数	60	※1
TCP/IP	a4gimMAX_HOST_LENGTH	ホスト名の最大バイト数	96	※1
	a4gimMAX_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）	**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	
	getServices	現在使用できる通信サービス	
	getIMEI	IMEI-IDの取得	
	setBaudrate	UARTの通信速度の設定	**初期は115200bps**
	setLED	LED1のON/OFF	
	setAirplaneMode	エアプレーンモードのON/OFF	

※　Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

- **概要**
  - a4gim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　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gw4 g 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の状態で、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.");
```

1．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gw4ｇ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の状態で、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、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。 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。

【使い方の例】

```
if (a3gs.begin(0, 115200) == 0)
  Serial.println("Succeeded.");
```

2．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

3．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.");
```

4．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.");
```

5．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();
```

6．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IEMのIMEIを取得する
引数	imei ：取得したIMEI（サイズは a4gimIMEI_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LTE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（a4gimIMEI_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[a4gimIMEI_SIZE];
if (a3gs.begin() == 0) {
   a3gs.getIEI(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。

7．コントロール関連の関数　setBaudrate

int setBaudrate(int baudrate)	
機能概要	Arduinoと４GIM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	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、４GIMと通信できなくなる。 工場出荷時の通信速度は、安定動作が可能な 115200(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。 本関数による通信速度の変更は、直ちに有効となる。 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）

【使い方の例】

```
if (a3gs.setBaudrate(115200) == 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。
a4gim.h のスケッチ内の「a4gimBAUDRATE」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。

8. 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.");
}
```

9．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、LTE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． Web関連関数

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/　（タブレインとは直接関係のないサービスです）
- **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）	
	header : 特殊なヘッダの指定(最大a4gimMAX_HEADER_LENGTHバイト)	
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	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）	

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 QWxhZGRpbjpvcGVuIHNlc2FtZQ==**rn**X-Myheder: XYZ**rn**"

  - 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a4gimMAX_HEADER_LENGTHバイト)	
	body : HTTPのボディ文字列(最大a4gimMAX_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	
	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a4gimMAX_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	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。 【**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。	

5．サービス他関連関数

サービス他関連関数

- **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	
	getResult	４GIMからの応答値受信	
	sendCommand	４GIMへのコマンド送信	
	sendData	４GIMへのデータ送信	
	enterAT	ATコマンドパススルーモード	
	discardUntil	４GIMからの文字の受信	

- **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．サービス他関連関数　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） a4gimSRV_NO（＝０）：サービス利用不可 a4gimSRV_PS（＝１）：パケット通信サービスのみ a4gimSRV_CS（＝２）：音声通信(+SMS)サービスのみ a4gimSRV_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	４GIM(V1.0)では、a4gimSRV_CSとa4gimSRV_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 (a4gim.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．サービス他関連関数　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アンテナを付けないで測定した場合では「-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 (a4gim.getRSSI(rssi) == 0)
    Serial.println(rssi);
```

3．サービス他関連関数　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 (a4gim.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．サービス他関連関数　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される問題。
【詳細はウィキペディア参照】

5．サービス他関連関数　getVersion

int getVersion(char* version)	
機能概要	４GIM ファームウェアgw4 g アプリのバージョンを取得する（最新版V3.3）
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。

6．４GIMからの応答値受信関数　getResult

int getResult(char *buf, int *len, uint32_t timeout)	
機能概要	４GIM からの応答値受信
引数	buf : 結果を返すバッファ
	len：バッファサイズ
	timeout：タイムアウト時間
戻り値	0：正常に取得できた時
	１：取得できなかった時（タイムアウト）
補足	

7．４GIMへのコマンド送信　sendCommand

void sendCommand(const char* cmd)	
機能概要	４GIM へのコマンド送信
引数	cmd: コマンド送信（「＄コマンド」などを送信）
戻り値	戻り値なし
補足	

8．４GIMへのデータ送信関数　sendData

void sendData(const char* data)	
機能概要	４GIM へのデータ送信
引数	data: 配列文字
戻り値	戻り値なし
補足	

9．ATコマンド実行　enterAT

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	ATコマンドパススルーモードでは、HL7539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。 ATコマンドの使用に制限はないため、HL7539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、gw4ｇ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。 利用できるATコマンドの詳細は、HL7539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。 ※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。

１０．４GIMからの文字の読み捨て関数　discardUntil

void discardUntil(const char match)	
機能概要	４GIM からの文字の読み捨て関数
引数	match: 読み捨てするまでの文字（この文字と一致したら処理終了）
戻り値	戻り値なし
補足	

6． 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	
	setTCPParams	TCPパラメータ設定	
	writeBegin	シリアル通信で直接書込み	
	read※	データの読込み	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

※ Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

【注意事項】

- 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デフォルトで60秒が設定されている
- readやwriteでエラーが発生した時は、一旦disconnectTCPを呼び出した後、connectTCPによる接続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(80/443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関数はノンブロッキング（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。

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 (a4gim.connectTCP(svr, 80) == 0) {
  // GET Request for google site
  a4gim.write("GET / HTTP/1.0$n");
  a4gim.write("HOST:");
  a4gim.write(svr);
  a4gim.write("$n$n");
  //  Get resopnse..
}
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付けエラー*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　setTCPParams

int setTCPParams(int timeout1, int timeout2)	
機能概要	TCPコネクションのパラメータ設定
引数	timeout1：connectTCP関数のタイムアウト時間（設定：10m秒）
	timeout2 : write（）/read（）関数のタイムアウト時間（設定：10ミリ秒）
戻り値	0 : 正常処理
	1 : パラメータの間違い
	2 : 内部処理エラー発生
補足	引数は、ともに0より大きく6000（1分）より短いこと

5．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	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で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、４GIM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。

【使い方の例】

```
char *svr = “someserver.aaa";
uint8_t data[256];
if (a4gim.connectTCP(svr, 80) == 0) {
  a4gim.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。

6．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 (a4gim.connectTCP(svr, 80) == 0) {
  // GET Request for google site
  a4gim.write("GET / HTTP/1.0$n");
  a4gim.write("HOST:");
  a4gim.write(svr);
  a4gim.write("$n$n");
  handleResponse();
}
else
  Serial.println("Error: can't connect");
```

```
void handleResponse(void) {
  int c;
  while ((c = a4gim.read()) > 0) {
    // 読み出した文字cを処理する
  }
}
```

6．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[a4gimMAX_RESULT_LENGTH+1];
  int nbytes;
  while(a4gim.read(res,a4gimMAX_RESULT_LENGTH+1) > 0) {
    Serial.print(res);
  }
}
```

6．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	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。 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。

7．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。

7．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	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。 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。

【使い方の例】

```
char *svr = “arduino.cc";
if (a4gim.connectTCP(svr, 80) == 0) {
  // GET Request for google site
  a4gim.write("GET / HTTP/1.0$n");
  a4gim.write("HOST:");
  a4gim.write(svr);
  a4gim.write("$n$n");
  //  Get resopnse..
}
```

7．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：型 size_t）
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 (a4gim.connectTCP(svr, 8080) == 0) {
  a4gim.write(binaryData, sizeof(binaryData));
}
```

７．プロファイル関連ほか関数

プロファイル関連の関数

提供関数ライブラリ

分類	メソッド名	機能概要	補足
プロファイル	setDefaultProfile	デフォルトプロファイルを設定	
	getDefaultProfile	デフォルトプロファイルを取得	

【注意事項】プロファイルは、通信サービス事業体が提供する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 car *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） ４GIM V1.0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： IIJ様の iijmio サービス(iijmio.jp/mio@iij/iij)

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 (a4gim.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");
}
```

【補足資料】注意点

- 本製品で利用しているLTE通信モジュール（HL7539、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にお願い致します。（GPSアンテナ関係は別売オプションとなります）

- Arduinoと４GIM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。

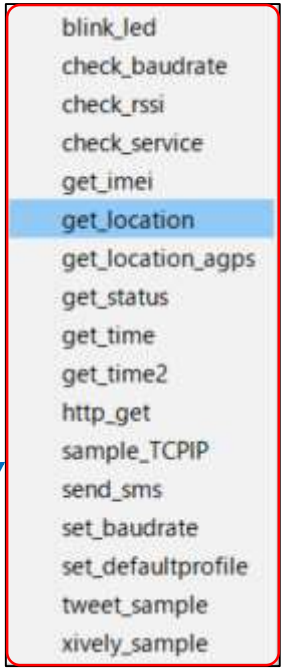
- 3Gモジュールは瞬間的に消費電力が高くなる場合があるため、なるべく外部電源(ACアダプタ)をご利用いただくことをお薦めいたします。詳細は2章を参照ください。
  - ご利用されるパソコンの特性により、Arduino側へのUSB接続からの電力供給だけでは、3Gシールドが利用できない場合がありますのでご注意ください。動作が不安定となる場合は、外部電源(ACアダプタ)の利用をお勧めします。

※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）

８．サンプルスケッチの実行例

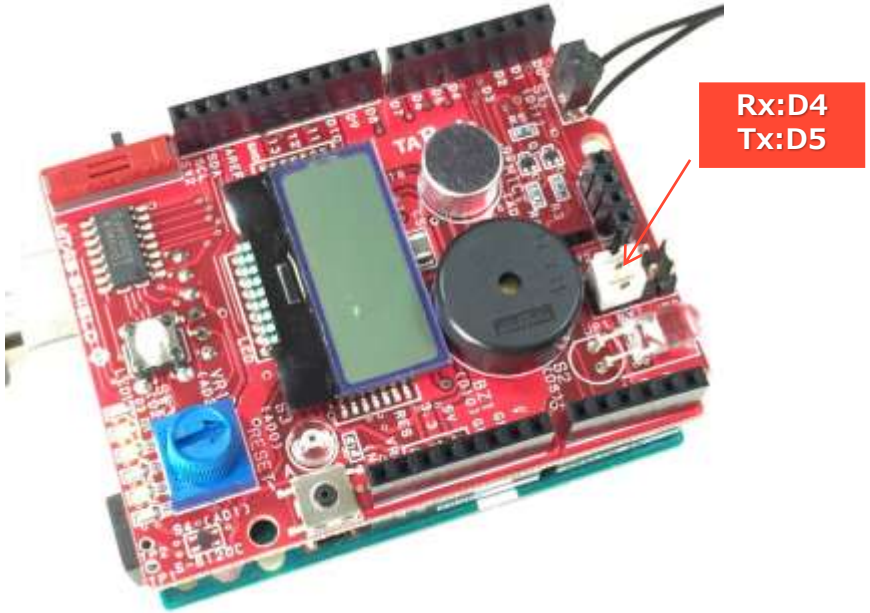
1．a4gimサンプルスケッチ一覧

a4gimのサンプルスケッチを動かしてみましょう。
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。



2．３GIMシールド・IoTABシールド設定方法

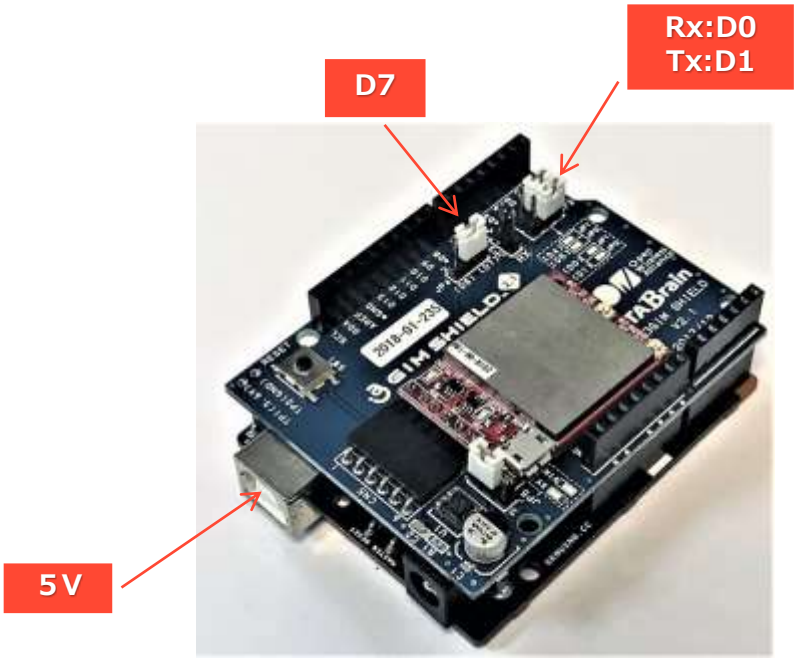
a4gimのサンプルスケッチをArduino IDEで読み込み、ArduinoUNOやArduino M0(Pro)、Genuino101ほか互換機にコンパイルし、書き込んで、実行してみてください。ここでは、ArduinoUNO＋IoTABシールド＋４GIM　または　ArduinoUNO＋３GIMシールド＋４GIM　を使った設定を紹介しておきます。



ArduinoUNO＋IoTABシールド＋４GIMでは、

① JP1とJP2は、RxとTxのジャンパピンをD4とD5に接続

（ただし通信速度は、57600pbsから9600bpsで利用）



Arduino M0(Pro)＋３GIMシールド＋４GIMでは、

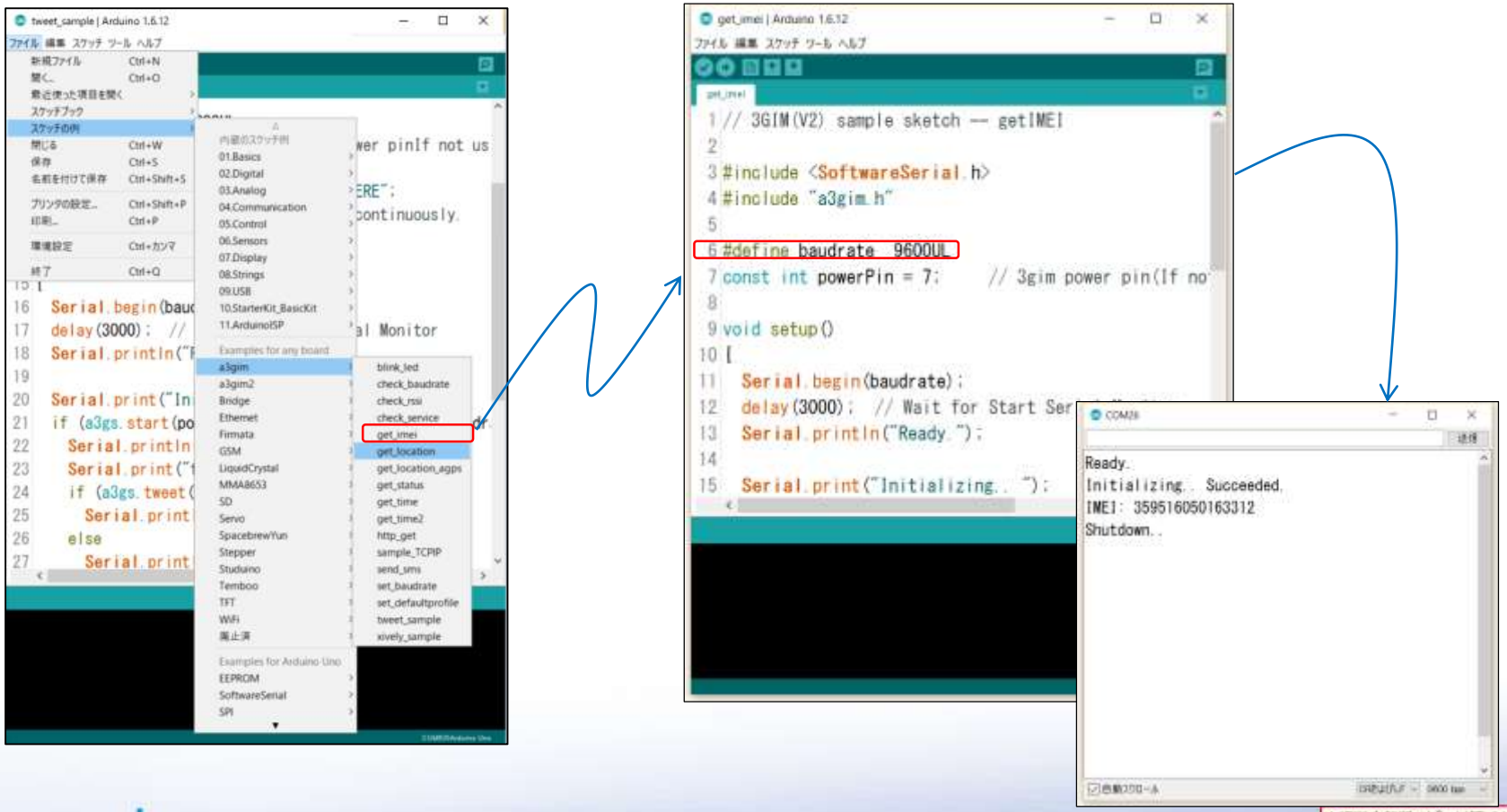
① JP１とJP2のRxとTxのジャンパピンをD0とD1に接続

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、a4gim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４GIM）シールド＋４GIM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・実行を行ってみています。



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: 115200
Recognize succeeded.
Current baudrate is 115200 bps.
```

■check_service.ino
（SIMカードのサービス情報）

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

■check_rssi.ino
（電波強度取得）

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

■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..
```

■get_status.ino
（機能状態取得）

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

■get_time.ino
（現日時取得）

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

■get_time2.ino
（現日時取得２）

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

■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..
```

■tweet_sample.ino
（ツイート送信確認）

※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）

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

Rev. 180129

補足資料

目次

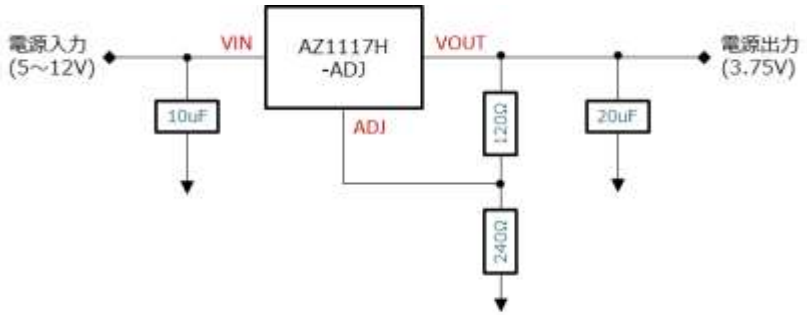
【補足資料１】４GIM V1.0 コマンド・応答一覧表

No	分類	機能	コマンドの形式	応答（レスポンス）：正常時	応答（レスポンス）：エラー時
1	System	Version	$YV¥n	$YV=OK version¥n	$YV=NG errno¥n
2		RSSI	$YR¥n	$YR=OK rssi¥n	$YR=NG errno¥n
3		Service	$YS¥n	$YS=OK service¥n	$YS=NG errno¥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	$YA=OK	$YA=NG errno¥n
11	Web	Get	$WG url ["header"]¥n	$WG=OK nbytes¥nresponse¥n	$WG=NG errno ..¥n
12		Post	$WP url "body" ["header"]¥n	$WP=OK nbytes¥nresponse¥n	$WP=NG errno ..¥n
13	TCP/IP	Read	$TR maxbytes¥n	$TR=OK nbytes¥ndata¥n	$TR=NG errno ..¥n
14		Write	$TW "data"¥n	$TW=OK nbytes¥n	$TW=NG errno ..¥n
15		Connect	$TC host_or_ip port¥n	$TC=OK¥n	$TC=NG errno ..¥n
16		Disconnect	$TD¥n	$TD=OK¥n	$TD=NG errno ..¥n
17		Status	$TS [status]¥n	$TS=OK status¥n	$TS=NG errno ..¥n
18		Get sockname	$TN¥n	$TN=OK ipAddr portNo¥n	$TN=NG errno ..¥n
19		Tunnel Write	$TT nbyte¥ndata	$TT=OK nbytes¥n	$TT=NG errno ..¥n
20		Set/Get Param	$TX [timeout]¥n	$TX=OK [timeout]¥n	$TX=NG errno¥n
21	UDP		$UB	$UB=OK	$UB=NG errno ..¥n
22			$UE	$UE=OK	$UE=NG errno ..¥n
23			$US	$US=OK	$US=NG errno ..¥n
24			$UN	$UN=OK	$IM=NG errno ..¥n
25	Profile	Set/Read	$PS APN user pw¥n	$PS=OK¥n	$PS=NG errno¥n

※V2.2では、SMS関連機能は削除されました。

【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、４GIM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

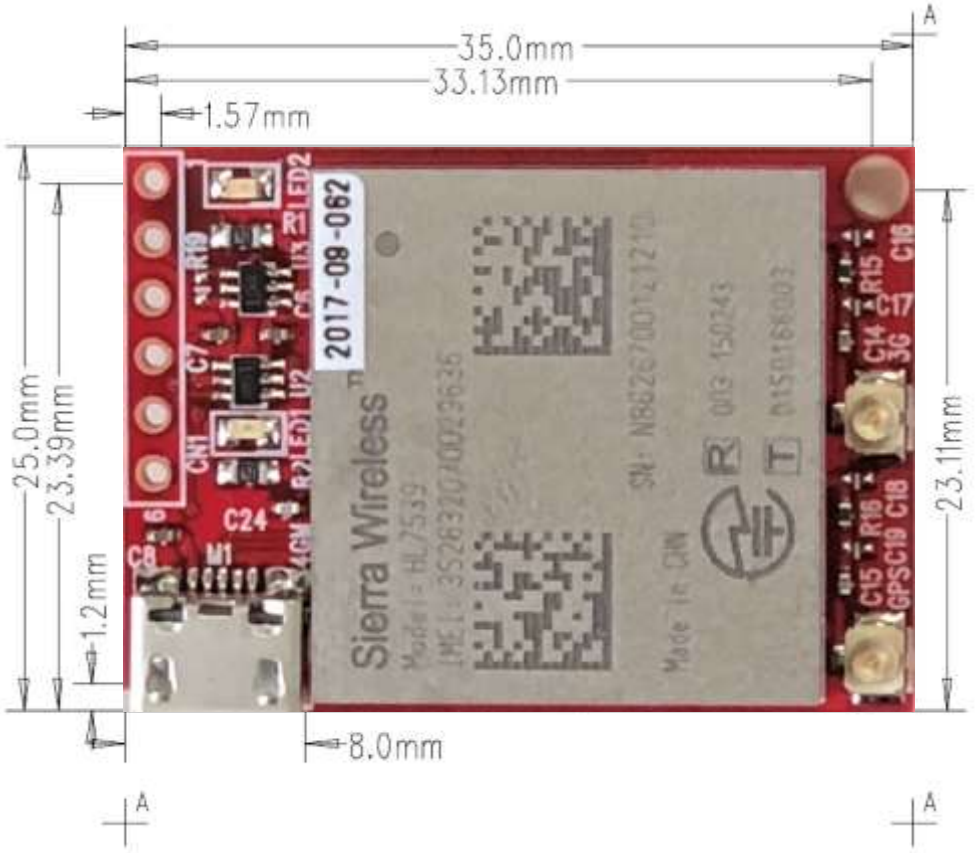
【補足資料３】トラブルシューティング

#	課題	現象	問題点・対応策	補足
1	配線・接続	・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	・４GIM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）	・#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）	・コマンドを送っても、返信がない ・正しい応答でない	・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 ４GIM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	・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 ※はんだ付け不良も多く発生
3	エラー頻発	・#=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	・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（電流不足が原因として多い）	・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	・電波強度が取得できない	・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	・同 上
5	GPS取得	・GPS取得ができない ・GPS取得に時間が掛る ・アクティブGPSアンテナで時間が掛る	・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 ・アクティブ用の「at+gpsconf=1,1」が設定されていない	・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
6	SIMプロファイル設定	・プロファイル設定でエラー発生	・正しいSIM（通信できる）やアンテナが接続されていない	・場合によっては何度も実行して正しく設定されることあり
7	ネット接続	・Webコマンド群やTCP/IPコマンド群が正しく応答しない	・3G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	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

※その他トラブルが有った場合には、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４】 4GIM V1.0 外形寸法

3GIM(V2.2)と寸法は変わっていません。



【補足資料５】 ４GIM サポートサイト

４GIMに関する技術情報が盛りだくさん掲載されています。
https://3gim.wiki/

【補足資料６】４GIM関連商品のご紹介

４GIMを**Arduino** UNO や Genuino101 上、それに**RaspberryPi**（B+/2B/3B/Zero)でも簡単に使用することができる３GIMシールドや３GIM 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

３GIMシールドは、Arduino UNOやGenuino101、さらにArduino MEGAなどの上で簡単に４GIMを利用することができます。

３GIMシールドは、電源電圧3.7Vを４GIMに安定供給し、安心して４GIMを稼動させることができます。

IoTAB SHIELD 3.0

IoTABシールドV3.0（最新版はV4.0）は、これまでのセンサ拡張ボードTABシールドに、４GIM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

３GIM HATは、Raspberry Pi の多くの種類で稼動する４GIM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との相違点

- ハードウェアの違い
  - 搭載されている通信モジュールが異なります
  - ３GIMではシエラワイヤレス社のHL8548、４GIMでは同社のHL7539が搭載されています。

- ファームウェアの違い
  - ファームウェアgw4ｇのバージョン番号が「1.0」となりました。
  - GPS機能がなくなりました。
  - UDP機能が追加となりました。

【補足資料８】 ラズベリーパイでの４GIMの利用

４GIM 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４GIMを3Gモデムとして利用するための手順をご紹介します。

４GIM を通信モデムとして利用することで、 ４GIM 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（https://3GIM.wiki/doku.php？id=3GIM_v2＃ラズベリーパイで使用する方法）

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】すぐに利用されたい方へ

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 ４GIM V1.0およびアンテナ（４GIM専用通信アンテナ）

② Arduino UNO または Arduino M0 Pro（Zero）、 Genuino101など

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タイプの標準USBケーブル、もしくはマイクロUSBケーブル

・SIMカードは、iijmioプリペイドまたはsoracomなどのSIMカード（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。（販売予定）

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）最新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 ４GIMにマイクロSIMカード挿入し、４GIM専用のアンテナを装着する

② IoTABシールドまたは３GIMシールド上に①の４GIMを取り付ける

③ 上記②のシールドを、Arduino UNO またはArduino M0 Pro（Zero）、 Genuino101などに取り付ける

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などに書き込み、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

【補足資料10】参考資料および参考サイト

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
8）サンプルコード（第３章・第４章）zipファイル

４GIM関連保守サービスほか

- ４GIMに関する技術的なお問い合わせ・ご相談は、以下のメールにて対応しています。

  info@tabrain.jp

- マニュアルやWikiページ等で分かりにくいところなどで、ご支援いたします。あらかじめマニュアルやWikiページによる熟読をお願いいたします。マニュアルに分かり難い点がございましたら、ご指摘頂けますようお願い申し上げます。

（注意）お客様で作成されたプログラムなどのご相談は、対象外とさせて頂きます。

また、独自サーバとの接続での問い合わせも対象外とさせて頂きます。

（サーバ側のプログラムとの関連もあることから）