a3gim R4.3
（３GIM V2.2用）

本資料は、３GIM V2.2 ＋３GIMシールド Ｖ２.0 を簡単に利用するためのArduino用ライブラリ**a3gim R4.３** および**a3gim2 R4.3**の解説書です。

【注意】本３GIM V2.2モジュールの通信速度は、9600bpsに設定しています。(a3gim.h内)
本ライブラリa3gimR4.3は、３GIM V2.1以下のバージョンでは使えませんのでご注意ください。

Open Wireless Alliance

３GIM V2.2+3GIMシールド a3gim R4.3 ライブラリ仕様書

2012/10/01～2017/08/18

著作：NPO法人オープンワイヤレスアライアンス

TABrain

もくじ

第Ⅰ編 概要・導入

１．３GIM V2.2+３GIMシールドの概要

1．3GIM V2.2+３GIMシールドとは

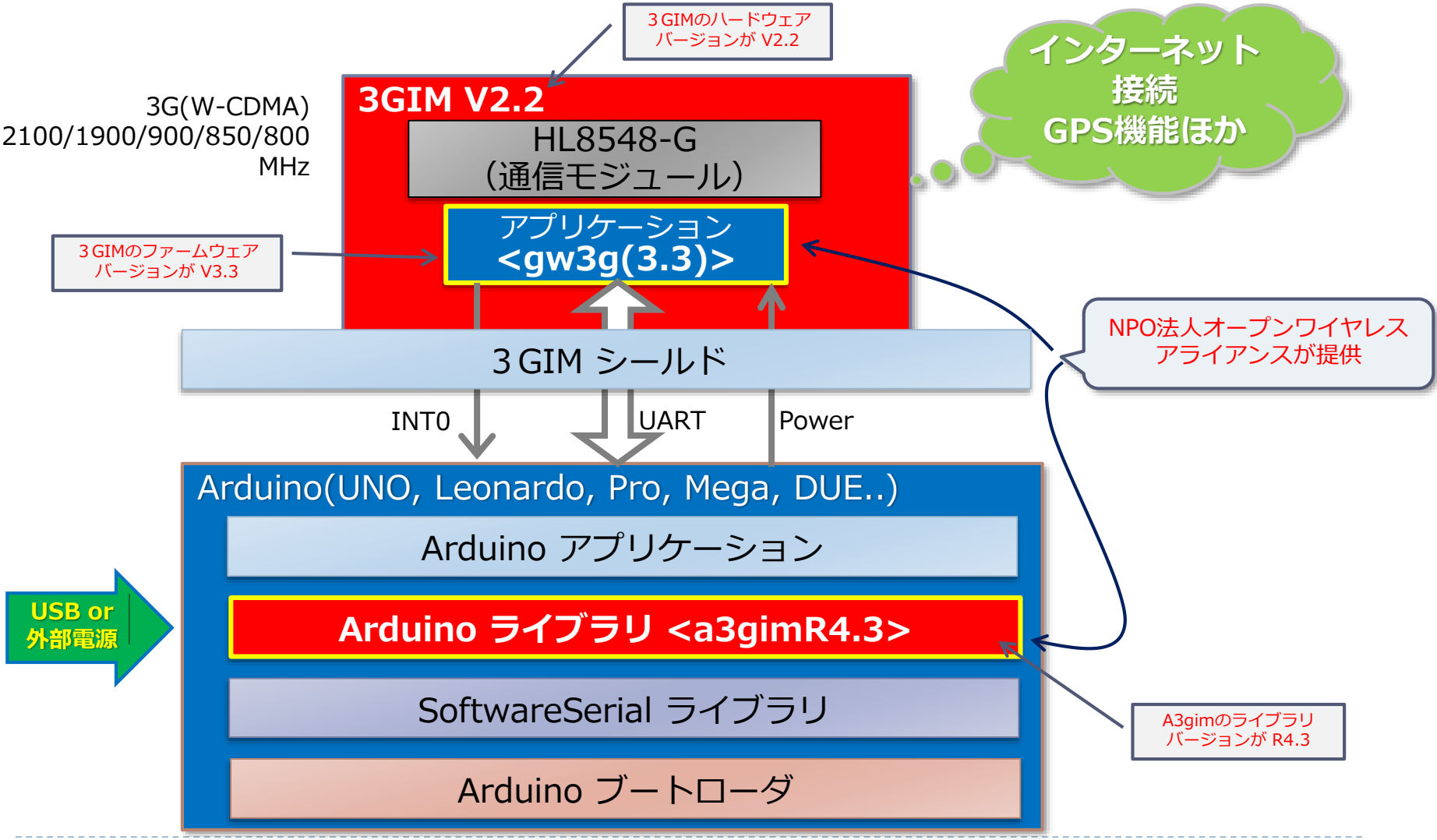
- Arduino上で簡単に3GIM V2.2を利用できるようにしたボードが３GIMシールドです。
  - ３GIM V2.2は、３G通信機能を持つFOMA通信をサポート
  - NTTdocomo、MVNOのIIJmio、DTI、SORACOM、ほかSO-NET（０SIM）等のSIMカードで利用可能

- メモリサイズの小さい8ビットマイコン「Arduino UNO R3」の上でも利用できる高機能なライブラリ“**a3gim**”を提供（Arduino Mega、Leonardo、Arduino Zero、Arduino M0、Arduino M0 Pro、Genuino101では、**a3gim2**をご利用ください）

- ライブラリが提供する機能は、下記の通り：
  - **Web通信**(HTTP/HTTPS GET/POST)
  - **TCP/IP通信**(connect, disconnect, read, write)
  - **現在位置情報（GPS）取得**(Assisted GPS)
  - **プロファイル確認・設定**(APN情報による設定・確認)
  - **その他**（初期化、電源制御、時刻取得、電波強度取得、IMEI取得、LED制御等）

- 3GIMシールドを利用するにあたっての留意点は下記の通り：
  - a3gim ライブラリでは、Arduino標準ライブラリのソフトウェアシリアル（SoftwareSerial.h）を利用する。
  - a3gim2ライブラリでは、ハードウェアシリアル（Serial1）を利用する。
  - ３GIMシールドでは、シリアル通信UARTは、標準的には４（Rx)と５（Ｔｘ）ピンを利用（変更可能）
  - 7番ピンで、３GIMの電源On/Offを切替えに利用する。

2. 3GIM V2.2+3GIMシールドの構成



2．a3gim R4.3ライブラリ概説

1．ライブラリを利用するハードウェア

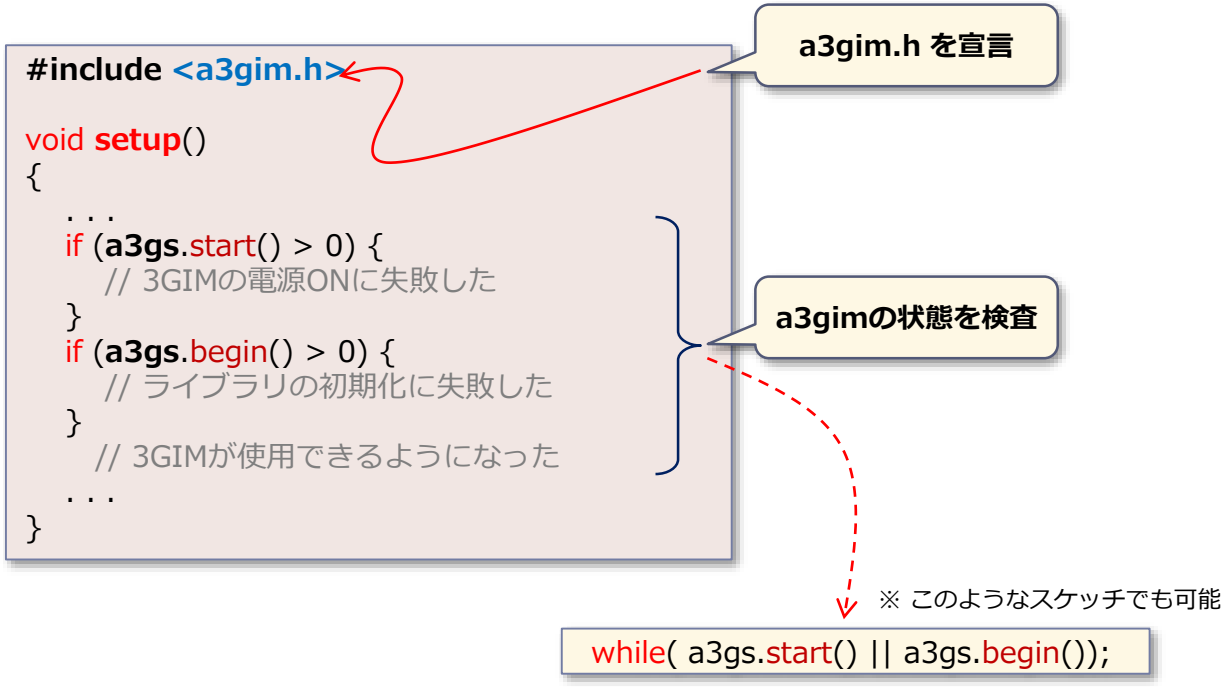
- Arduinoから3GIMを簡単に利用できるようにするために、ライブラリa3gimが提供されています。
  - a3gimライブラリは、Arduino UNO/Pro等でソフトウェアシリアルを使って3GIMを利用する際に使用します。
  - a3gim2ライブラリは、Arduino Mega/Due/M0 Pro/Leonardo/101/Zero等でハードウェアシリアル3GIMを利用する際に使用します。提供される機能は、a3gimと同等です。
  - 以下では、a3gim/a3gim2を総称してa3gimと呼びます。
  - ３GIM V2.2では、a3gim および a3gim2ともに 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3GIMの各機能を利用することができます。
- 本ライブラリ群は、タブレイン製の「IoTABシールド」または「3GIMシールド 」を利用し、Arduino UNO、Zero(M0) Pro、Genuino101、ArduinoMEGAなどの上で稼働できます。
- このa3gim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

- a3gimライブラリを使用する方法
  - ライブラリは、スケッチの中で以下の順序でコントロール用のメソッド（関数）を呼び出すことにより利用できます。




```
#include <a3gim.h>

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

```
while( a3gs.start() || a3gs.begin());
```

3．ライブラリ機能一覧 ライブラリが提供する関数(1/3)

分類	メソッド名※1	機能概要	補足
コントロール（Control）	**begin**※	ライブラリの初期化	
	end※	ライブラリの終了	
	restart※	3GIMのリセット	
	start※	3GIMの電源ON	
	shutdown※	3GIM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	**出荷時は9600bps**
	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．ライブラリ機能一覧 ライブラリが提供する関数(2/3)

分類	メソッド名	機能概要	補足
Web機能	**httpGET**※	GETメソッドの要求	http/httpsを利用可
	httpPOST	POSTメソッドの要求	同上
	tweet※	Twitterへの投稿	＊
現在位置取得（GSP)	**getLocation**	現在位置の取得	緯度経度情報
	getLocation2	現在位置の取得２	緯度経度他情報
	setLocationParams	GPS機能関連設定	**※追加**
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	3GIMのバージョンの取得	
	getResult	3GIMからの応答値受信	
	sendCommand	3GIMへのコマンド送信	
	sendData	3GIM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	３GIM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）
※ **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

3．ライブラリ機能一覧 ライブラリが提供する関数(3/3)

分類	メソッド名	機能概要	補足
TCP/IP機能	**connectTCP**※	TCPコネクションを接続	
	disconnectTCP※	TCPコネクションを切断	
	getStatusTCP	TCPコネクション最新状況取得	
	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

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“a3gim.h”で定義されています。

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

※1 これらの定数は、ATmega328*/32U*（Unoなど）を利用したArduinoではSRAMのサイズが小さいことからかなり制約が厳しい。ATmega2560/1280（Megaなど）またはADKを利用することで、これらの最大値を大きくすることができる。
※2 大きなデータを読み書きする場合は、複数回に分けてread/writeを実行する。

3．a3gimインストール方法

1．a3gim.zipのインストール方法

GitHubからa3gim.zipをダウンロードして、以下のフォルダに解凍してください。
Windowsの場合：　ドキュメントの中にある￥Arduinoフォルダの下にある¥libralies

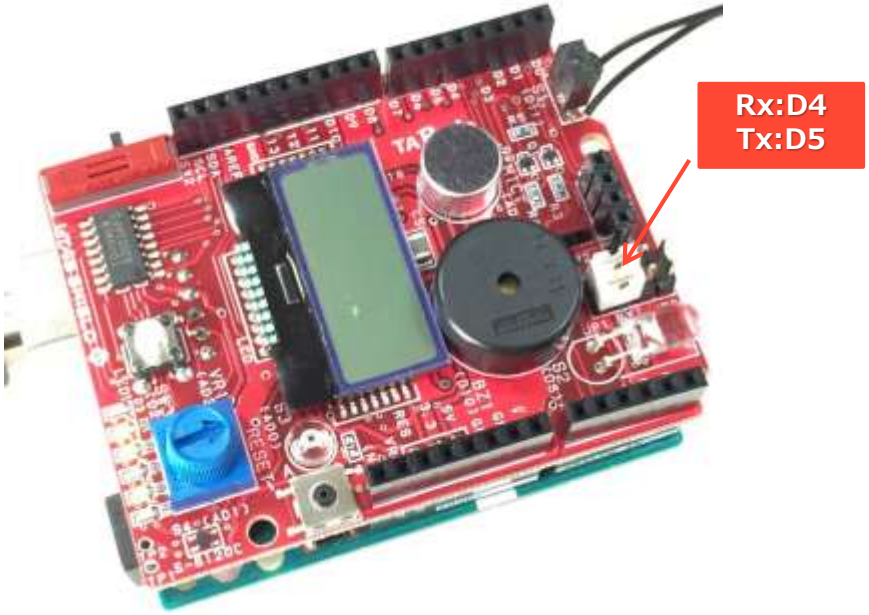
2．a3gimサンプルスケッチ一覧

a3gimのサンプルスケッチを動かしてみましょう。
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。

- blink_led
- check_baudrate
- check_rssi
- check_service
- get_imei
- get_location
- get_location_agps
- get_status
- get_time
- get_time2
- http_get
- m2x_sample
- sample_TCPIP
- set_baudrate
- set_defaultprofile
- tweet_sample

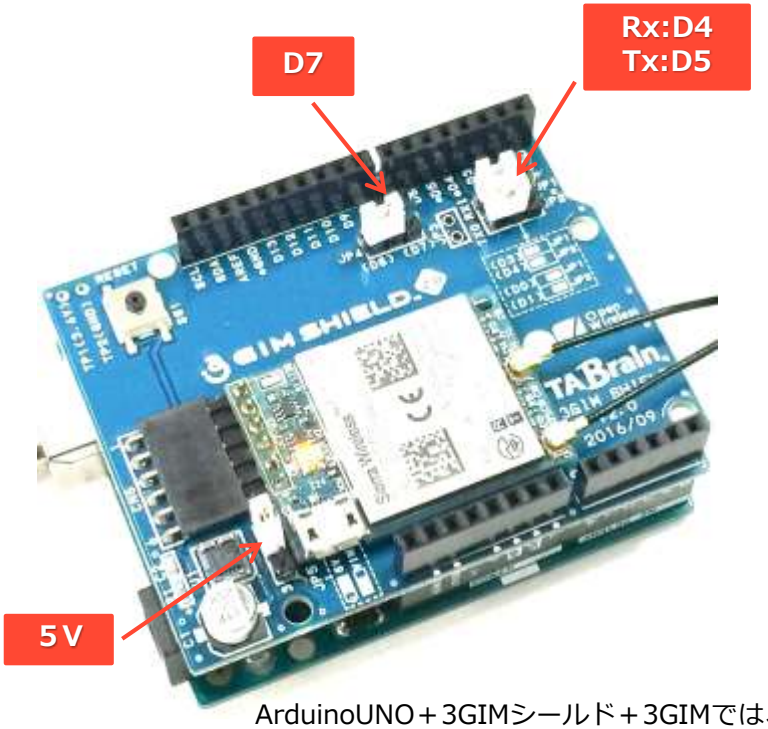
3．3GIMシールド・IoTABシールド設定方法

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



ArduinoUNO＋IoTABシールド＋3GIMでは、

① JP1とJP2は、RxとTxのジャンパピンをD4とD5に接続



ArduinoUNO＋3GIMシールド＋3GIMでは、

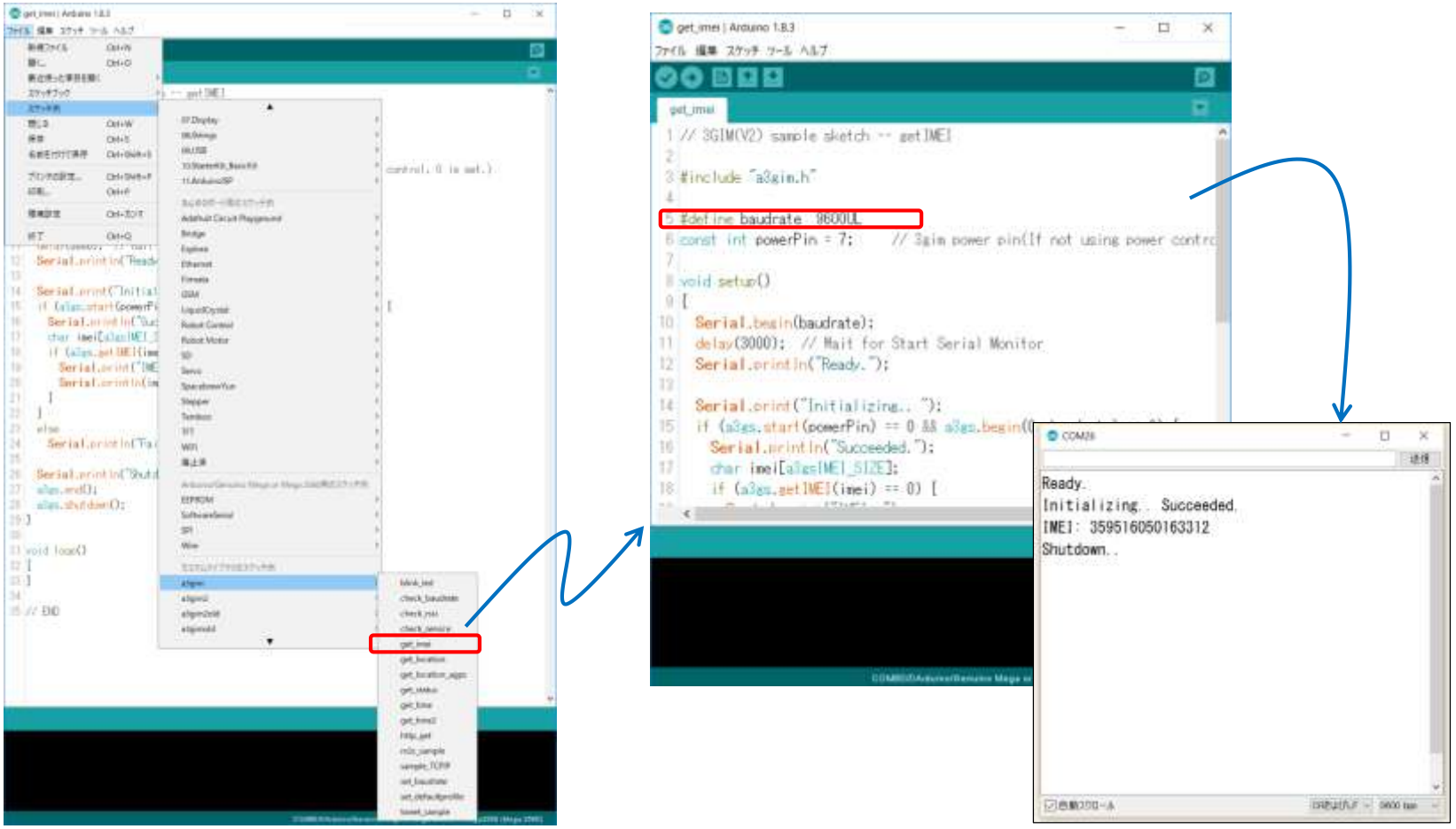
① JP１とJP2のRxとTxのジャンパピンをD4とD5に接続

② JP４はD7に接続

③ JP5は　５Vに接続

4．サンプルスケッチの読込・書込・実行

- 前述の環境か、ほぼ同等の状態で、a3gimのサンプルスケッチをArduinoIDEに読み込み、コンパイルし、Arduino＋（IoTABまたは3GIM）シールド＋3GIM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。



第Ⅱ編 機能説明

４．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9600bps**
	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**
  - a3gimライブラリの初期化・終了、ライブラリの状態の取得、3GIMシールドのリセット、電源ON/OFF、IMEIの取得、LED1のON/OFF、UARTの通信速度の設定を行う
- **留意点**
  - 電源ONには、15秒程度の時間が掛かる。リセットには、5秒程の時間が掛かる。
  - 電源OFFには、1秒ほどの時間が掛かる
  - **setBaudrate**による通信速度の変更には、十分留意すること

1．コントロール関連の関数　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：HL8548-G上のgw3gアプリのバージョンが古い（ライブラリは使用不可）
補足	3Gシールドの電源がONの状態で、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：HL8548-G上のgw3gアプリのバージョンが古い（ライブラリは使用不可）
補足	3Gシールドの電源がONの状態で、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, 9600) == 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	HL8548-G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HL8548-G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HL8548-GのIMEIを取得する
引数	imei　:取得したIMEI（サイズは a3gimIMEI_SIZE バイト）
戻り値	0：正常に取得できた時
	0以外：エラーが発生した時（取得できない時）
補足	IMEIとは3G通信モジュール(HL8548-G)の識別IDである（電話番号とは無関係） 引数imeiが指す結果格納場所のスペース（a3gimIMEI_SIZEバイト＝16桁）は、あらかじめ呼び出し側で確保しておくこと。

【使い方の例】

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

【出力結果例】

354563020267950

IMEI番号

このIMEIの中に、HL8548-G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3GIM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、3GIM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 9600(bps) となっている。 通信速度をデフォルトの設定値よりも高くするには、ハードウェアシリアルの利用を推奨する。 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(9600) == 0)  {
  Serial.println("Baudrate was changed.");
  Serial.println("Please reset me now.");
}
```

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

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、3G通信は実行できないが、SMSの受信のみ可能である（ただし、受信したSMSの読み出しやSMSの送信はできない） エアプレーンモードをONにすることで、消費電力を大幅に節約することができる。 リセットまたは電源のOFF/ONで、デフォルトの設定(OFF)に戻る。

【使い方の例】

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

5．Web関連関数

5．Web関連の関数

提供関数ライブラリ

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

※　Arduino GSM/GPRSシールド用ライブラリと互換性がある関数
＊　無償サービス「http://arduino-tweet.appspot.com/」を利用（要Twitterの登録）

概要

- http/httpsを簡単に利用できる。
- GET/POSTメソッドを利用できる。
- Web機能の関数は、すべて同期処理である。そのため、レスポンスが取得できるまで、あるいは通信がタイムアウト(30秒程度)するまで呼び出し元には制御は戻らない。
- tweetは、サードパーティのフリーサービスを利用することで使用できる（ユーザ登録が必要、利用条件はそのサービスに従う）。
  - 詳細は http://arduino-tweet.appspot.com/　（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a3gimMAX_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a3gimMAX_HEADER_LENGTHバイト)	
	body : HTTPのボディ文字列(最大a3gimMAX_BODY_LENGTHバイト)	
	result : [OUT] レスポンスの格納先（スペースは呼び出し側で確保）	
	resultlength : [IN/OUT] resultのサイズを指定、呼び出し結果のサイズが返却	
	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a3gimMAX_TWEET_LENGTHバイト）	
戻り値	0 : 正常に投稿できた時	
	0以外 : 投稿できなかった時	
補足	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。	

6．位置情報取得（GPS）関連関数

現在位置取得（GPS）関連の関数

- **提供関数ライブラリ**

分類	メソッド名	機能概要	補足
現在位置取得（GPS)機能	getLocation	現在位置の取得	内蔵GPSを使用
	getLocation2	現在位置の取得２	
	setLocationParams	GPS関連機能設定	アシストGPSおよび アクティブGPSアンテナ

- **概要**
  - HL8548-G内蔵GPSや3Gネットワークを利用して位置を測位
  - 引数の指定により、下記のいずれかの測位方法を選択できる（３GIM V2.0以降は、区別なしで、任意文字対応）
    - **MPBASED**
      - GPSを利用して現在位置を測位する。GPSが利用できない場合は、3Gネットワークを利用する。
    - **MPASSISTED**
      - 3Gネットワーク上のロケーションサーバを利用して現在位置を測位する。
    - **MPSTANDALONE**
      - GPSのみを利用して現在位置を測位する。
- **留意点**
  - 通信サービス（例えば、IIJmio等）によっては、3Gネットワーク上のロケーションサーバ（Googleサーバ利用）を利用することができない。その場合は、GPS単独の測位のみが利用できる。
  - 測位方法としてアシストGPSを利用する場合は、通信料金（通常、定額プランの範囲内だが、SIMカードの通信サービスによる）が発生する
  - 屋内や都心等のように、上空にある衛星の電波がGPSアンテナで補足できない場所では、正しく測位できない場合がある。

1．現在位置取得（GPS）関連の関数　getLocation

int getLocation(int method, char* latitude, char* longitude)	
機能概要	現在位置を取得する
引数	method : 測位方法（a3gimMPBASED / a3gimMPASSISTED / a3gimMPSTANDALONE のいずれか）を指定（現在任意文字対応）
	latitude : [OUT] 緯度(北緯) dd.dddddd形式、ただし桁数は場合により可変
	longitude : [OUT] 経度(東経) ddd.dddddd形式、同上
戻り値	0 : 正常に取得できた時
	0以外 : 取得できなかった時（latitude, longitudeの値は不定）
補足	本関数の実行には、数十秒～3分程度の時間が掛かる。 AGPSサーバは、3GIM(V2.2)ではGoogleのサーバを利用する（今後変更となる可能性がある） 本関数は、同期処理である。そのため、測位処理が完了するまで、あるいは測位が失敗するまで呼び出し元には制御は戻らない。

2．現在位置取得（GPS）関連の関数　getLocation2

int getLocation2(char* latitude, char* longitude, char *height, char *utc, int *quality, int *number)	
機能概要	現在位置を取得する
引数	latitude :[OUT] 緯度(北緯：度)　dd.dddddd形式、ただし桁数は場合により可変
	longitude : [OUT] 経度(東経：度) ddd.dddddd　同上
	height : [OUT]　アンテナの海抜高さ（単位：m）
	utc : [OUT] 協定世界時での時刻（hhmmss）
	quality : [OUT] 位置特定品質。0 = 位置特定できない、1 = SPS（標準測位サービス）モード、2 = differenctial GPS（干渉測位方式）モード
	number : [OUT] 使用GPS衛星数
戻り値	0：正常に取得できた時
	0以外：取得できなかった時（latitude, longitudeの値は不定）
補足	本関数の実行には、数十秒～3分程度の時間が掛かる。 AGPSサーバは、3GIMV2.2ではGoogleのサーバを利用する（今後変更となる可能性がある） 本関数は、同期処理である。そのため、測位処理が完了するまで、あるいは測位が失敗するまで呼び出し元には制御は戻らない。

3．GPSの初期設定関数　setLocationParams

int setLocationParams(int timeout, boolean useAGPS, boolean useActiveAntenna)	
機能概要	AGPS（アシストGPS）やアクティブGPSアンテナの設定など関数
引数	timeout:GPS取得の制限時間（10ミリ秒）【省略値：180000（3分間）】
	useAGPS:AGPS(アシストGPS)の設定、ONにする時はTRUE（=1）、OFFにする時はFALSE（=0）を指定する
	useActiveAntenna：アクティブGPSアンテナの設定、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：設定できなかった時
補足	本関数の実行では、SIMカードの設定やアクティブGPSアンテナの接続が必要。

7．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	3GIMからの応答値受信	
	sendCommand	3GIMへのコマンド送信	
	sendData	3GIMへのデータ送信	
	enterAT	ATコマンドパススルーモード	
	discardUntil	３GIM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） a3gimSRV_NO（＝0）: サービス利用不可 a3gimSRV_PS（＝1）: パケット通信サービスのみ a3gimSRV_CS（＝2）: 音声通信(+SMS)サービスのみ a3gimSRV_BOTH（＝3）: パケット通信、音声通信(SMS)いずれも可
戻り値	0: 正常に取得できた時
	0以外: 取得できなかった時（statusの値は不定）
補足	3GIM(V2.2)では、a3gimSRV_CSとa3gimSRV_BOTHは返却しない。つまり、SMSが利用できるかどうかは判断できない。

【使い方事例】

```
int status;
if (a3gs.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 (a3gs.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 (a3gs.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	3GIM ファームウェアgw3g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． 3GIMからの応答値受信関数　getResult

int getResult(char *buf, int *len, uint32_t timeout)	
機能概要	3GIM 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	

7．３GIMへのコマンド送信　sendCommand

void sendCommand(const char* cmd)	
機能概要	3GIM 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	

8．３GIMへのデータ送信関数　sendData

void sendData(const char* data)	
機能概要	3GIM 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、HL8548-Gに対して直接ATコマンドを送信して、そのままの結果を取得することができる。 ATコマンドの使用に制限はないため、HL8548-Gの設定を任意に変更することができる。しかし、変更した設定等によっては、gw3gファームウェアの動作に支障をきたす場合があるので、十分留意すること。 利用できるATコマンドの詳細は、HL8548-Gの開発元であるSierra Wireless社のサイトで公開されている「AT Commands Interface Guide - AirPrime HL6 and HL8 Series」を参照のこと。 ※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。

１０．３GIMからの文字の読み捨て関数　discardUntil

void discardUntil(const char match)	
機能概要	3GIM 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	

8． TCP/IP関連関数

TCP/IP関連の関数

提供関数ライブラリ

分類	メソッド名	機能概要	補足
TCP/IP	connectTCP※	TCPコネクションを接続	
	disconnectTCP※	TCPコネクションを切断	
	getStatusTCP	TCPコネクション最新状況取得	
	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 (a3gs.connectTCP(svr, 80) == 0) {
   // GET Request for google site
   a3gs.write("GET / HTTP/1.0$n");
   a3gs.write("HOST:");
   a3gs.write(svr);
   a3gs.write("$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、3GIMが自動的に0x00バイトを充当する。 通信速度を115,200bpsに設定している場合は、フロー制御を行っていないため1024バイトのデータ送るたびに5ミリ秒以上のディレイを挿入する必要がある。

【使い方の例】

```
char *svr = “someserver.aaa";
uint8_t data[256];
if (a3gs.connectTCP(svr, 80) == 0) {
  a3gs.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Iバイトのデータ
	-I：　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 (a3gs.connectTCP(svr, 80) == 0) {
  // GET Request for google site
  a3gs.write("GET / HTTP/1.0$n");
  a3gs.write("HOST:");
  a3gs.write(svr);
  a3gs.write("$n$n");
  handleResponse();
}
else
  Serial.println("Error: can't connect");
```

```
void handleResponse(void) {
  int c;
  while ((c = a3gs.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[a3gimMAX_RESULT_LENGTH+1];
  int nbytes;
  while(a3gs.read(res,a3gimMAX_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 (a3gs.connectTCP(svr, 80) == 0) {
  // GET Request for google site
  a3gs.write("GET / HTTP/1.0$n");
  a3gs.write("HOST:");
  a3gs.write(svr);
  a3gs.write("$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 (a3gs.connectTCP(svr, 8080) == 0) {
  a3gs.write(binaryData, sizeof(binaryData));
}
```

９．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） 3GIM V2.2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 (a3gs.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3G通信モジュール（HL8548-G、以下3Gモジュールと呼ぶ）は，付属している3Gアンテナとの組合せで，日本の技適（技術基準適合証明※1）を取得をしています。よって，日本以外の海外での利用や，アンテナの取り換えやケーブルの取り外し等を行った使い方は，電波法違法利用となりますので，絶対行わないでください。

- 3GアンテナおよびGPSアンテナ，それにそれぞれのケーブルとコネクタは小さく，壊れやすいため，取扱いには，十分注意してください。特に，頻繁な取り外し・取り付けは行わないようにお願い致します。（GPSアンテナ関係は別売オプションとなります）

- Arduinoと3GIMを接続して、電源をONあるいはリセットを掛けた場合，利用できるようになるまで15秒程度の時間が掛かります。

- 3Gモジュールは瞬間的に消費電力が高くなる場合があるため、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）

第Ⅲ編 サンプル・スケッチ

１０．サンプルスケッチ説明

- 【注意事項】本稿で扱うスケッチ（プログラム）は、IDEバージョン Arduino 1.8.3 上で作成・確認したものです。その他のバージョンでは稼働しない場合があります。

1．blink_led.ino 【3GIM上のLED点滅】

- 3GIM上の緑色LEDが1秒間隔で点滅します。

【サンプルスケッチ（blink_led.ino）】

```
// 3GIM(V2) sample sketch -- setLED

#include <a3gim.h>

#define  INTERVAL  1000  // Blink interval
#define baudrate                  9600UL
const int powerPin = 7;     // 3gim power pinIf not using power control, 0 is set.

void setup()
{
  Serial.begin(baudrate);
  delay(100);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

Serial.print("Initializing.. ");
  if (a3gs.start(powerPin) == 0 && a3gs.begin(0, baudrate) == 0)
    Serial.println("Succeeded.");
  else {
    Serial.println("Failed.");
    Serial.println("Shutdown..");
    a3gs.end();
    a3gs.shutdown();
    return;
  }
  Serial.println("Blinking..");
}
```


```
void loop()
{
  a3gs.setLED(true);
  delay(INTERVAL);
  a3gs.setLED(false);
  delay(INTERVAL);
}

// END
```

2．check_baudreate.ino 【3GIM上のLED点滅】

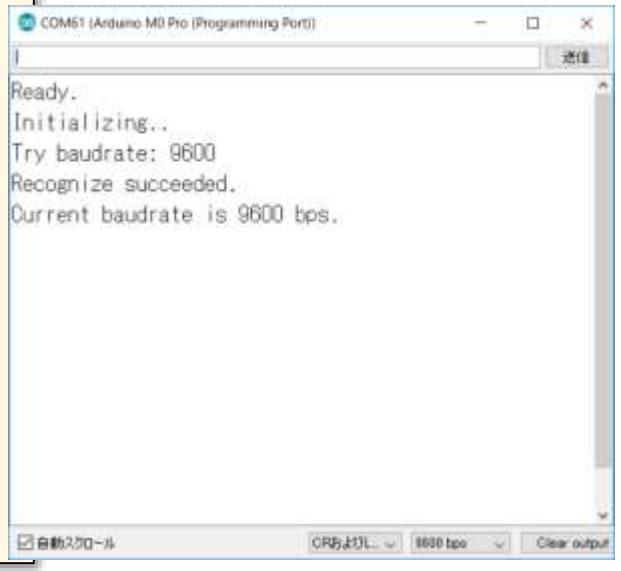
- 3GIMの設定された通信速度（ボーレート）を探しだします。

【サンプルスケッチ（check_rssi.ino）】

```
// 3GIM(V2) sample sketch -- check current baudrate

#include <a3gim.h>

const int powerPin = 7;     // 3gim power pin(If not using power control, 0 is set.)
long baudrates[5] = { 9600, 19200, 38400, 57600, 115200 };
                    //-- 2400bps and 4800bps are not supported in 3GIM(V2)

void setup()
{
  Serial.begin(9600);
  Serial.println("Ready.");

  Serial.println("Initializing.. ");
  for (int i = 0; i < 5; i++) {
    if (a3gs.start(powerPin) == 0) {
      Serial.print("Try baudrate: ");
      Serial.println(baudrates[i]);
      if (a3gs.begin(0, baudrates[i]) == 0) {
        Serial.println("Recognize succeeded.");
        Serial.print("Current baudrate is ");
        Serial.print(baudrates[i]);
        Serial.println(" bps.");
        return;
      }
      Serial.println("Failed.");
      a3gs.end();
      a3gs.shutdown();
    }
  }
  Serial.println("Can't recognize baudrate.");
}

void loop(){ }
```

３GIMに設定された通信速度が分からなくなったときに、探し当てます。

ただし、シリアル通信（UART）の接続が正しく設定されていることが前提となります。



3．check_rssi.ino 【受信信号（電波）強度取得】

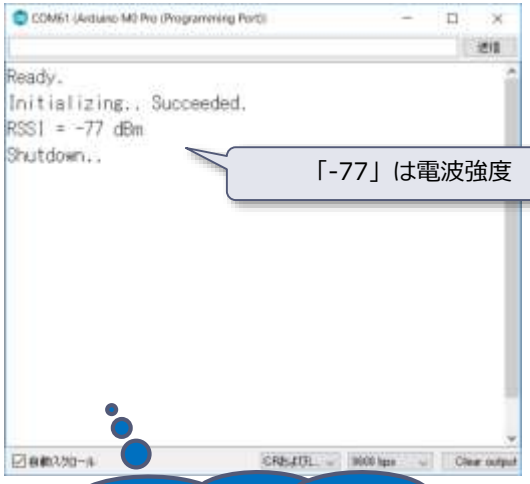
- Arduino上の３Gシールドが、電波強度「rssi」について、３G電波を捉えているかを、数値によって返します。

【サンプルスケッチ（check_rssi.ino）】

```
// 3GIM(V2) sample skech -- getRSSI

#include <SoftwareSerial.h>
#include <a3gim.h>

void setup()
{
  Serial.begin(9600);
  delay(100);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    int rssi;
    if (a3gs.getRSSI(rssi) == 0)
      Serial.print("RSSI = ");
      Serial.print(rssi);
      Serial.println (" dBm");
  }
  else
    Serial.println("Failed.");
  Serial.println("Shutdown.. ");
  a3gs.end();
  a3gs.shutdown();
}

void loop()
{
}
```

電波強度を引数rssiに返す

シリアルモニタに
表示される内容（例）



電波強度は必ずマイナス値で返却
「0」に近いほど電波強度は強い

※値が「-70」以下になると電波強度は強い

「-110」に近い場合は、アンテナが外れているかもしれません。

4．check_service.ino 【SIMカードサービス形態取得】

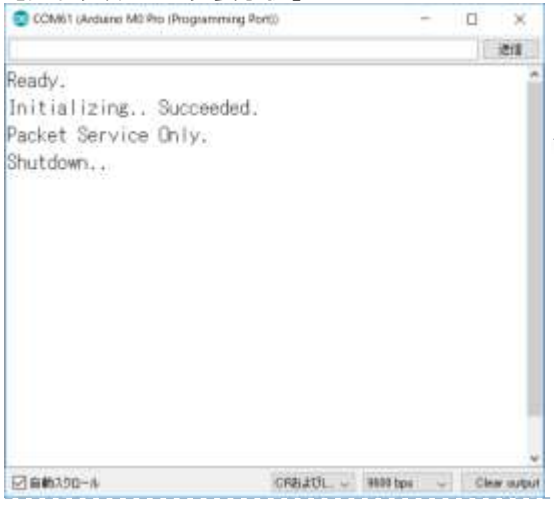
- SIMカードのサービス形態を取得し表示するスケッチです。

【【サンプルスケッチ（check_sercvce.ino）】

```
// 3GIM(V2) sample sketch  -- getServices
#include <a3gim.h>

#define baudrate 9600UL
const int powerPin = 7;

void setup()
{
  Serial.begin(baudrate);
  delay(100);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
```

```
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    int rstatus;
    if (a3gs.getServices(status) == 0) {
      switch (status) {
        case a3gsSRV_NO :
          Serial.println("No Service.");     break;
        case a3gimSRV_PS :
          Serial.println("Packet Service Only.");       break;
        case a3gimSRV_CS :
          Serial.println("Voice Service Only."); break; // also SMS
        case a3gimSRV_BOTH :
          Serial.println("Packet And Voice Services."); break;
        default :      break;
      }
    }
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop() {}
```

【シリアルモニタ表示例】



a3gimSRV_NO（＝0）：サービス利用不可
a3gimSRV_PS（＝1）：パケット通信サービスのみ
a3gimSRV_CS（＝2）：音声通信(+SMS)サービスのみ
a3gimSRV_BOTH（＝3）：パケット通信、音声通信(SMS)いずれも

5．get_imei.ino【HL8548-Gモジュール携帯端末識別番号取得】

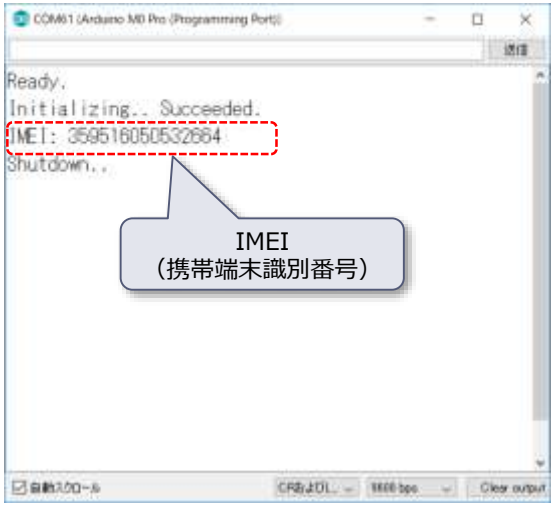
- HL8548-Gモジュールの固有ID（IMEI：携帯端末識別番号）を取得し表示するスケッチです。

【サンプルスケッチ（get_time.ino）】

```
// 3GIM(V2) sample sketch -- getIMEI

#include <a3gim.h>

#define baudrate  9600UL
const int powerPin = 7;

void setup()
{
  Serial.begin(9600);
  delay(100);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    char imei[a3gsIMEI_SIZE];
    if (a3gs.getIMEI(imei) == 0)
      Serial.print("IMEI: ") ;
      Serial.println(imei) ;
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop()  { }
```

IMEI取得関数

【シリアルモニタ表示例】



6．get_location.ino 【現在位置取得】

- 現在位置情報をGPSより取得表示するサンプルです。

```
// 3GIM(V2) sample sketch -- getLocation

#include <a3gim.h>

#define baudrate  9600UL
#define TIMEOUT 9000
const int powerPin = 7;

void setup()
{
  Serial.begin(baudrate );
  delay(3000);    // Wait for start serial monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start(powerPin) == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded. It maybe takes several minutes.");
    char lat[15], lng[15];
    if (a3gs.getLocation(a3gimMPBASED, lat, lng) == 0) {
      Serial.print("OK: ");
      Serial.print(lat);
      Serial.print(", ");
      Serial.println(lng);
    }
    else
      Serial.println("I don't know this location.");
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}
void loop()  {}
```

【注意事項】 GPS機能は、アンテナ特性が影響しますので、アンテナの感度の良いもので、なるべく周りの障害物を避け、さらに周辺にノイズ発生源が無いなどの条件で、位置情報が取得できます。

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded. It maybe takes several minutes.
OK: ****** ******
Shutdown..
```

緯度、経度を表示

ここで、表示された緯度・経度を、Googleの地図上で、直接カット&ペーストで、座標値を表示させることができます。

7．get_location_agps.ino 【A-GPSによる現在位置取得】

- 現在位置情報をA-GPSおよびアクティブGPSアンテナより取得表示するサンプルです。

```
// 3GIM(V2) sample sketch -- getLocation

#include <a3gim.h>

#define baudrate  9600UL
const int powerPin = 7;
const boolean useAGPS = true;          // Use AGPS function
const boolean useActiveAntenna = true;  // Not use active GPS antenna
void setup()
{
  Serial.begin(baudrate );
  delay(100);    // Wait for start serial monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start(powerPin) == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
```

A-GPS利用

アクティブGPSアンテナ利用

```
  // Set timeout and make AGPS function available
    a3gs.setLocationParams(TIMEOUT, useAGPS, useActiveAntenna);

    // Get location with specified timeout
    char lat[15], lng[15], utc[7], height[8];
    int quality, number;
    if (a3gs.getLocation2(lat, lng, height, utc, &quality, &number) == 0) {
      Serial.print("OK: ");
      Serial.print(lat);
      Serial.print(",");
      Serial.print(lng);
      Serial.print(",");
      Serial.print(height);
      Serial.print(",");
      Serial.print(utc);
      Serial.print(",");
      Serial.print(quality);
      Serial.print(",");
      Serial.println(number);
    }
    else
      Serial.println("Sorry, I don't know this location.");
  }else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}
void loop()  {}
```

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded. It maybe takes several minutes.
OK:35.641936,139.603996,52.5,032043,1,6
Shutdown..
```

緯度、経度を表示

8．get_status.ino【時刻取得1】

- 機能状態を取得するスケッチです。

【サンプルスケッチ（get_time.ino）】

```
// 3GIM(V2) sample sketch for Mega/Leonardo.. -- getStatus

#include <a3gim2.h>

#define baudrate  9600UL
const int powerPin = 7;     // 3gim power pin(If not using power control, 0 is set.)
```

【シリアルモニタ表示例】

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

```
void setup()
{
  Serial.begin(baudrate);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");

  if (a3gs.start(powerPin) == 0 && a3gs.begin(0, baudrate) == 0) {
    Serial.println("Succeeded.");
    int  status;
    status = a3gs.getStatus();
    Serial.print("Status is ");
    switch (status) {
      case A3GS::ERROR :
        Serial.println("ERROR");
        break;
      case A3GS::IDLE :
        Serial.println("IDLE");
        break;
      case A3GS::READY :
        Serial.println("READY");
        break;
      case A3GS::TCPCONNECTEDCLIENT :
        Serial.println("TCPCONNECTEDCLIENT");
        break;
      default :
        Serial.println("Unknown");
        break;
    }
  }

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop() {  }
```

9．get_time.ino【時刻取得1】

- 3GIMのタイマー機能を使って、年月日および時刻を取得するスケッチです。

【サンプルスケッチ（get_time.ino）】

```
// 3GIM(V2) sample sketch      getTime

#include <a3gim.h>
#define baudrate 9600UL
const int powerPin = 7;

void setup()
{
  Serial.begin(baudrate);
  delay(100);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start(powerPin) == 0 && a3gs.begin(0,baudrate) == 0) {
    Serial.println("Succeeded.");
    char date[a3gimDATE_SIZE], time[a3gimTIME_SIZE];
    if (a3gs.getTime(date, time) == 0) {
      Serial.print(date);
      Serial.print(" ");
      Serial.println(time);
    }
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop()  { }
```

時刻取得

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
2017/08/19 13:11:24
Shutdown..
```

【注意事項】 現在時刻の取得は、3G通信回線よりNICTから取得設定されますのです。

10. get_time2.ino【時刻取得2】

- 3GIMのタイマー機能を使って、1970年1月1日からの総数秒を取得するスケッチです。

【サンプルスケッチ（get_time2.ino）】

```
// 3GIM(V2) sample sketch – getTime2

#include <a3gim.h>

#define baudrate 9600UL
const int powerPin = 7;

void setup()
{
  Serial.begin(9600);
  delay(100);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start  (powerPin) == 0 && a3gs.begin(0,baudrate) == 0) {
    Serial.println("Succeeded.");
    unit32_t  seconds;
    if (a3gs.getTime2(seconds) == 0) {
      Serial.print(seconds);
      Serial.println("  Sec.");
    }
   else
      Serial.print ("Can't get seconds.");
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop()  { }
```

時刻取得

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
1503148616 Sec.
Shutdown..
```

【注意事項】 現在時刻の取得は、3G通信回線よりNICTから取得設定されますのです。

11. http_get.ino 【サーバのレスポンス返却】

本スケッチでは、arduino.ccにアクセスし、レスポンスを表示させています。

【サンプルスケッチ（http_get.ino）】

```
// 3GIM(V2) sample sketch  -- httpGET

#include <a3gim.h>
#define baudrate 9600UL
const int powerPin = 7;
const char *server = "www.arduino.cc";
const char *path = "";
int port = 80;

char res[a3gimMAX_RESULT_LENGTH+1];
int len;
```

Arduino サーバ

```
void setup()
{
  Serial.begin(baudrate);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print ("Initializing.. ");
  if (a3gs.start(powerPin) == 0 && a3gs.begin(0, baudrate) == 0) {
    Serial.println("Succeeded.");
    Serial.print ("httpGET() requesting.. ");
    len = sizeof(res);
    if (a3gs.httpGET(server, port, path, res, &len) == 0) {
      Serial.println("OK!");
      Serial.print ("[");
      Serial.print (res);
      Serial.println("]");
    }
    else
      Serial.println("NG!");
  }
  else
      Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();

}

void loop() { }
```

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
httpGET() requesting.. OK!
[<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
  <title>Arduino - HomePage </title>
  <link rel="shortcut icon" type="image/x-icon"
href="http://arduino.cc/en/favicon]
Shutdown..
```

1 2．m2x_sample.ino ① m2xクラウドアップ

本サンプルスケッチは、クラウドにセンサ値をアップするものです。

ここで使うm2xクラウドは、AT&T社がサービスするIoTクラウドで、フリー（無償）で利用できるサービスが用意されています。本サンプルでは、無償範囲での利用で、温度＋光センサの２つの値をクラウドにアップすることを行っています。

あらかじめ、m2xへの登録は、添付資料を参考にしてください。

```
//   AT&T m2x cloud server upload example using a3gim.h
//   upload data are temperature data and illumination data on the IoTABshield V3.0 (produced by Tabrain Inc.)
//   設定  <DEVICE-ID> および <API-KEY>   m2xで設定のこと
//   <STREAM ID> は、 温度センサ値「TEMP」および光センサ値「LIGHT」の２つ m2xで設定のこと

#include “a3gim.h”
#define BAUDRATE 9600UL
#define TMPPRA 0.594        // IoTABシールド ５V系マイコンボードの場合
//#define TMPPRA 0.392      //          ３．３V系マイコンボードの場合

const char *server = “api-m2x.att.com”;
const char *path = “v2/devices/<DEVICE-ID>/updates/”;
const char *header=“X-M2X-KEY:<API-KEY>$r$nContent-Type:application/json$r$n”;
const int port = a3gsDEFAULT_PORT;
const char *bodyTemplate = “{$¥”values$¥” : {$¥”TEMP$¥" : [{$¥"timestamp$¥" : $¥"%s$¥", $¥"value$¥" : $¥"%d.%d$¥"}], $¥"LIGHT$¥" :
[{$¥"timestamp$¥" : $¥"%s$¥", $¥"value$¥" : $¥"%d$¥"}] }}";
const int powerPin = 7;     // 3gim power pin
const int tempPin = 1, illPin = 0;  // Sensors pin number on IoTABshield V3.0
char res[100], body[256];

int len;
```

```
void setup() {
 delay(100);  // Wait for Start Serial Monitor
 Serial.begin(9600);
 Serial.println(">Ready.");

  Serial.print("Initializing.. ");
 if (a3gs.start(powerPin) == 0 && a3gs.begin(0,BAUDRATE) == 0)
   Serial.println("Succeeded.");
 else {
   Serial.println("Failed.");
   while (1) ;  // STOP
 }
}
```

m2xの登録は、ユーザ登録（ユーザID、パスワード）、

デバイスID、ストリームIDを登録します。

デバイスID（DEVICE-ID)を登録すると自動的にAPI－KEYが割り当てられます。ストリームIDは、スケッチの中の「TEMP」と「LIGHT」になります。このまま登録してください。

12．m2x _sample.ino ② m2xクラウドアップ

```
void loop() {
  char datetime[30];
  Serial.print(">httpPOST requesting: ");
  if (getDateAndTime(datetime) != 0) {
    Serial.println("getDateAndTime() Failed.");
    return;
  }
  int temp = getTemperature();
  int ill = getIlluminance();

  sprintf(body, bodyTemplate, datetime, (temp/10), (temp%10), datetime, ill);
  Serial.println(body);
  len = sizeof(res);

  int stat;
  if ((stat = a3gs.httpPOST(server, port, path, header, body, res, &len)) == 0) {
    Serial.println("Succeeded.");
    Serial.print(">Response=[");
    Serial.print(res);
    Serial.println("]");
  }
  else {
    Serial.print("Failed: ");
    Serial.println(stat);
  }

  delay(60000);  // take an interval
}
```

looop関数では、単純にデータをアップし、待機時間を1分間隔（60000ミリ秒）としています。

その他の関数としては、

1）時間を設定する関数getDateAndTime関数、

2）温度センサ値を取得する関数getTermperature関数

3）光センサ値を取得する関数getIlluminance関数

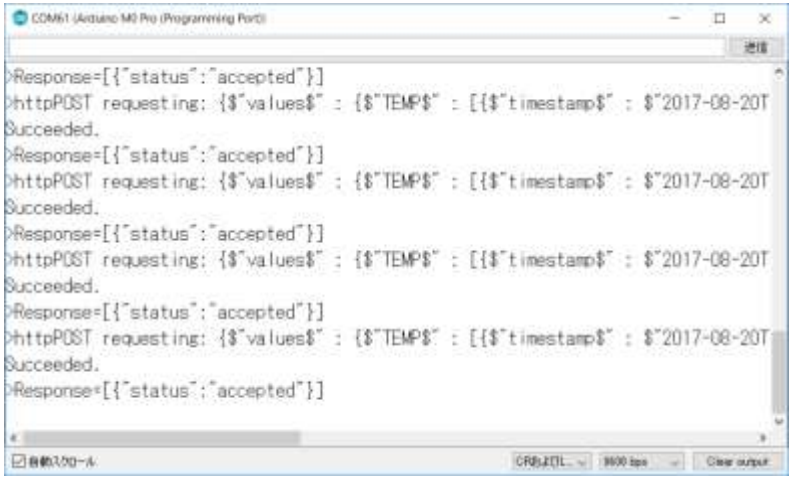
を用意しています。

```
int getDateAndTime(char *datetime) {
  char date[a3gsDATE_SIZE], time[a3gsTIME_SIZE];
  if (a3gs.getTime(date, time) != 0) {
    Serial.println("getTime() Failed.");
    return 1;    // error
  }
  date[4] = '-';
  date[7] = '-';
  sprintf(datetime, "%sT%s$+09:00", date, time);
  return 0;    // ok
}

int getTemperature(void) {   // get temperature(x10 C)
  for(int i=0; i<10; i++) analogRead(A1); // 空読み
  return (207.26 - TMPPRA*analogRead(A1))*10.0; // IoTABシールド V3.0利用の場合
}

int getIlluminance(void) {  // get illuminance(x1 lux)
  float mV = analogRead(illPin) * 3.23;        // IoTABシールド V3.0利用の場合
  return (mV / 2.9);
}
```

1 2．m2x _sample.ino ③ m2xクラウドアップ

実行結果は、右図が、シリアルモニタ画面。

下２つの画面コピーが、光センサ（ストリームID: LIGHT）と温度センサ（ストリームID: TEMP）の値がアップされた時刻歴のグラフ表示となります。



13. sample_TCPIP.ino 【TCP/IP機能のサンプル】

本スケッチは、TCP/IP機能を使ってhttp通信を行う事例です。
この例では、Arduinoの公式ページから、titleタグの中身を抜き出して、シリアルへ出力します。
改行文字等は、$文字を使ったエスケープシーケンスを使って記述します。

```
// a3gim sample skech.15-- connectTCP/disconnectTCP/read
//   A title is extracted from a homepage.

#include <a3gim.h>
#define baudrate        9600UL
#define MAX_RETRY  3

const in powerPin = 7;

const char *server = “tabrain.jp";  // URL to extract a title
const char *path = “/new/”;
const int port = 80;
char res[200];

boolean getTitle(char *res);

void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

 _redo:
  Serial.print("Initializing.. ");
  if (a3gs.start(powerPin) == 0 && a3gs.begin(0, baudrate) == 0) {
    Serial.println("Succeeded.");
 // Connect to server
    int nCount = 0;
    while (nCount++ < MAX_RETRY && a3gs.connectTCP(server, port) != 0) {
      Serial.println("connectTCP() can't connect, retry..");
      delay(100);
    }
    if (nCount > MAX_RETRY) {
      Serial.println("connectTCP() abort");
      goto _end;
    }
 // Send GET request
    a3gs.write("GET ");
    a3gs.write(path);
    a3gs.write(" HTTP/1.0$r$n");
    a3gs.write("HOST:");
    a3gs.write(server);
    a3gs.write("$r$n$r$n");
// Receive response
    do {
      int nbytes;
      if ((nbytes = a3gs.read(res, sizeof(res)-1)) < 0) {
        Serial.println("read() failed");
        break;
      }
    } while (! getTitle(res));
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();

  delay(15000);
  goto _redo;  // Repeat

 _end:
  while (1) ;
}

void loop() { }
```

TCP/IPの接続処理

Titleタグが取得できるまで、レスポンスを読み続ける

出力結果

```
Ready.
Initializing.. Succeeded.
tabrain.jp : Page title is "株式会社タブレイン"
Shutdown..
```

14．set_baudrate.ino 【通信速度の変更】

本スケッチは、通信速度を変更するスケッチです。

出荷時の9600pbsを19200、38400，57600、115200に変更することができます。

ただし、ArduinoUNOなどのソフトウェアシリアルで使う場合には、115200bpsは設定しないようにしてください。

```
// a3gim sample skech.12 -- setDefaultProfile and getDefaultProfile

#include <a3gim.h>

#define NEW_BAUDRATE              38400
#define CURRENT_BAUDRATE     9600
const int powerPin = 7;
void setup()
{
  Serial.begin(CURRENT_BAUDRATE );
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
   if (a3gs.start(powerPin) == 0 && a3gs.begin(0, baudrate) == 0){
    Serial.println("Succeeded.");
 if (a3gs.setBaudrate(NEW_BAUDRATE) == 0) {
      Serial.print("Baudrate was changed as ");
      Serial.print(NEW_BAUDRATE);
      Serial.println(" bps.");
    }
  }
  else
    Serial.println("Failed, baudrate was not changed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop()  {  }
```

注意事項

シリアルモニタ画面の通信速度と、３GIMの通信速度が大幅に違うと、文字化けや通信エラーが生じます。

できるだけ、互いの通信速度は同じか近い通信速度で処理するようにしてください。

誤って、設定した通信速度が分からなくなった場合には、

別なスケッチ「check_baudrate.ino」を実行してみてください。

15．set_defaultprofile.ino 【プロファイル変更】

- 本スケッチは、SIMカードのプロファイルを書き替えるもので、NTTドコモおよびそのMVNOのSIMカードのプロファイル情報（APN情報、ユーザ名、パスワード）を設定することで、通信ができるようになります。
- この場合、アンテナと正しいSIMカードの設定が必要です。

```
// 3GIM(V2) sample sketch -- setDefaultProfile and getDefaultProfile

#include <a3gim.h>

#define baudrate    9600

const char *newApn = "x.y";      //apn情報
const char *newUser = "u";       // ユーザ名
const char *newPassword = "p";  // パスワード
const int powerPin = 7;
```

プロファイル設定

【シリアルモニタ表示例】

```
Succeeded.
Default Profile Number is old_x.y,old_u,old_p
Succeed.
Shutdown..
```

古いプロファイル出力

```
void setup()
{
  Serial.begin(baudrate);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start(powerPin) == 0 && a3gs.begin(0, baudrate) == 0) {
    Serial.println("Succeeded.");
                        // Get current default profile
    char apn[20], user[20], password[20];
    if (a3gs.getDefaultProfile(apn, user, password) == 0) {
      Serial.print("Default Profile Number is ");
      Serial.print(apn);
      Serial.print(",");
      Serial.print(user);
      Serial.print(",");
      Serial.println(password);
    }

                        // Set new default profile
    if (a3gs.setDefaultProfile(newApn, newUser, newPassword) == 0) {
      Serial.println("Succeed.");
    }
    else
      Serial.println("Failed.");

  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}
void loop(){}
```

16．tweet_sample.ino①【ツィートデータ送信】

- ツイートは、センサー値をアップすることで、便利な利用方法ができるようになります。
- ツイートにメールアドレスで登録しておくと、ツイートされれば、自動的にメールが送られてきます。
- また複数の関係者もこのツイートによって、情報が拡散させることもできます。

- ただし、注意事項として以下の制限がありますので気を付けてご利用ください。

1）1分間隔以内でツイートしない

2）同じ内容のツイートをしない

```
// 3GIM(V2) sample sketch for Mega/Leonardo.. -- tweet

#include <a3gim.h>

#define baudrate 9600UL
const int powerPin = 7;     // 3gim power pinIf not using power control, 0 is set.
const char *token = "YOUR_TOKEN_HERE";
const char *message = "TWEET_MESSAGE_HERE";
  //-- Note: can't tweet same message continuously.

void setup()
{
  Serial.begin(baudrate);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start(powerPin) == 0 && a3gs.begin(0, baudrate) == 0) {
    Serial.println("Succeeded.");
    Serial.print("tweet() requesting.. ");
    if (a3gs.tweet(token, message) == 0)
      Serial.println("OK!");
    else
      Serial.println("Can't tweet.");
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop(){}
```

16．tweet_sample.ino②【ツィートデータ送信】

- 30秒ごとのセンサ値は、「ツイート」に送信され、またシリアル画面にも表示されます。

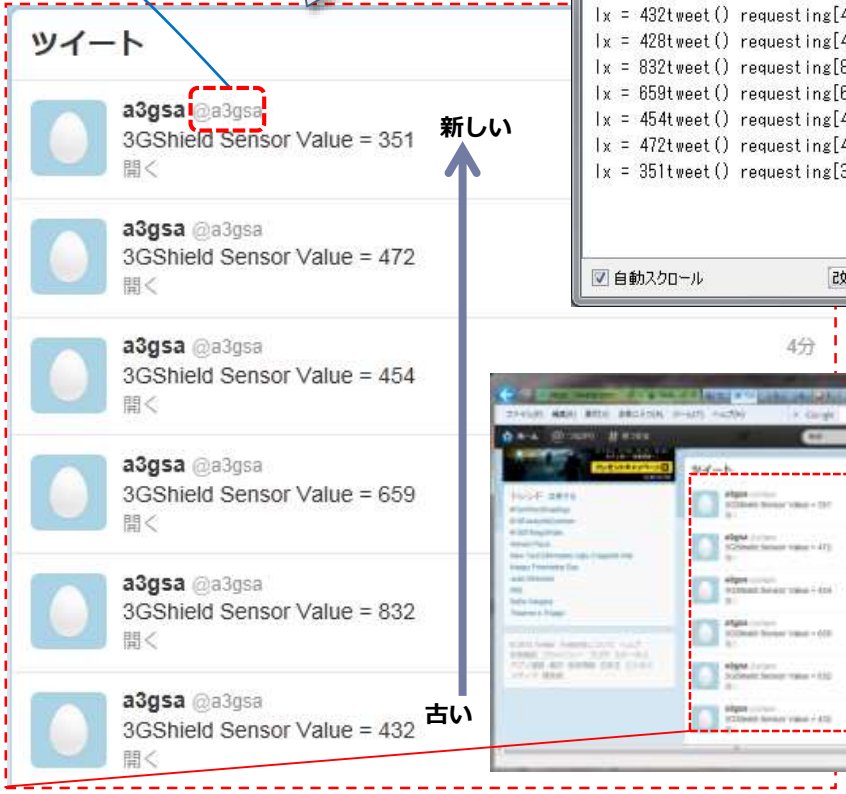
※ 以下の事例でのセンサ値は、光センサー（CdS）を利用

テキストで確認することもできる。
http://twansform.appspot.com/a3gima/text/1

Twitterにアップされた
ツィート内容（テキスト）

Twtter受信は次ページ参照

Tweet画面



シリアル
画面表示

添付資料

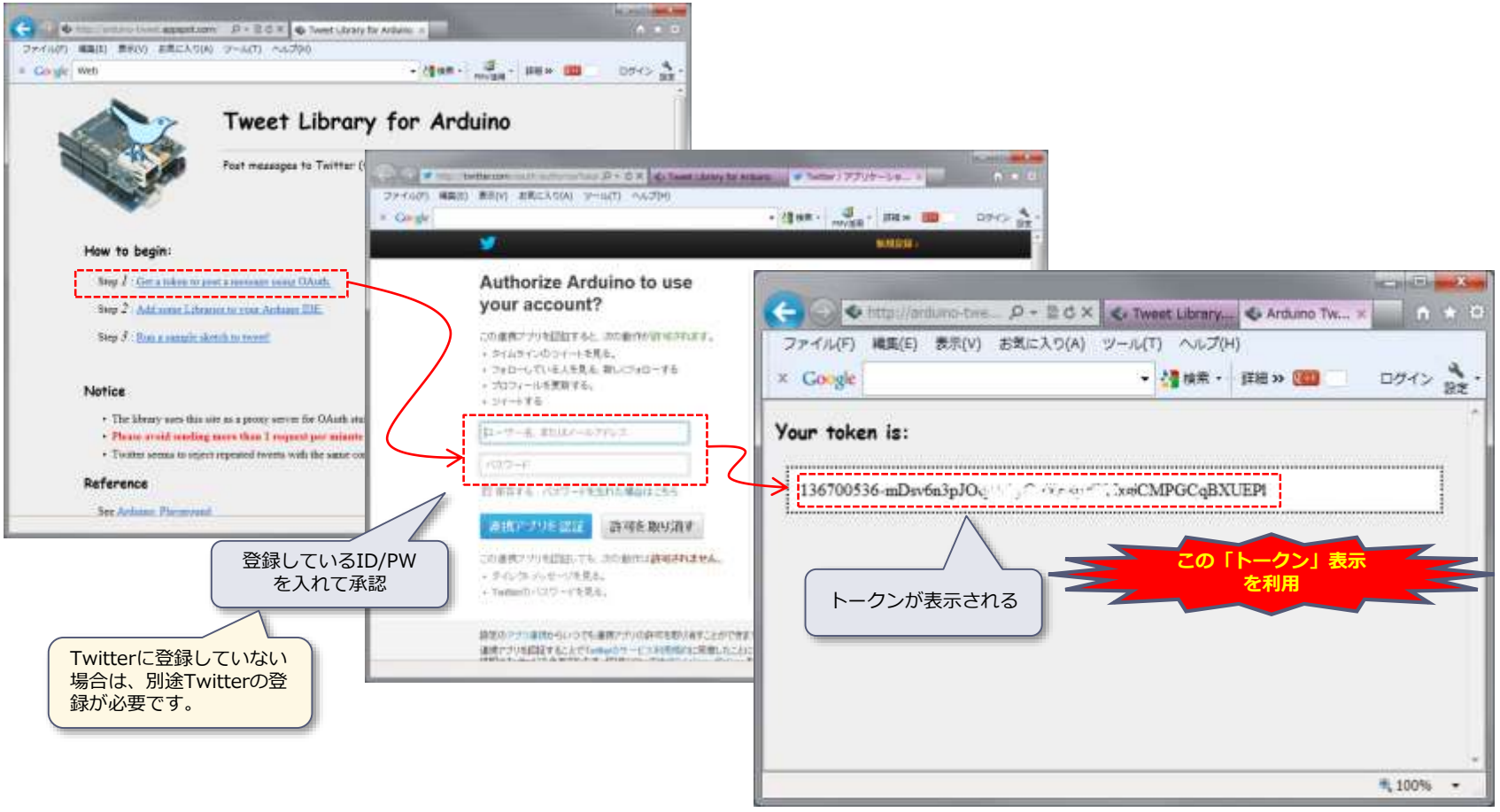
【添付資料.1】a3gimR4.3 ライブラリー覧表

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※	3GIMのリセット	
	start※	3GIMの電源ON	
	shutdown※	3GIM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9600bps**
	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	3GIMのバージョンの取得	
	getResult	3GIMからの応答値受信	
	sendCommand	3GIMへのコマンド送信	
	sendData	3GIM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	３GIMからの文字の受信	
現在位置取得（GPS）	**getLocation**	現在位置の取得	緯度経度情報
	getLocation2	現在位置の取得２	緯度経度他情報
	setLocationParams	GPS機能関連設定	**※追加**
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

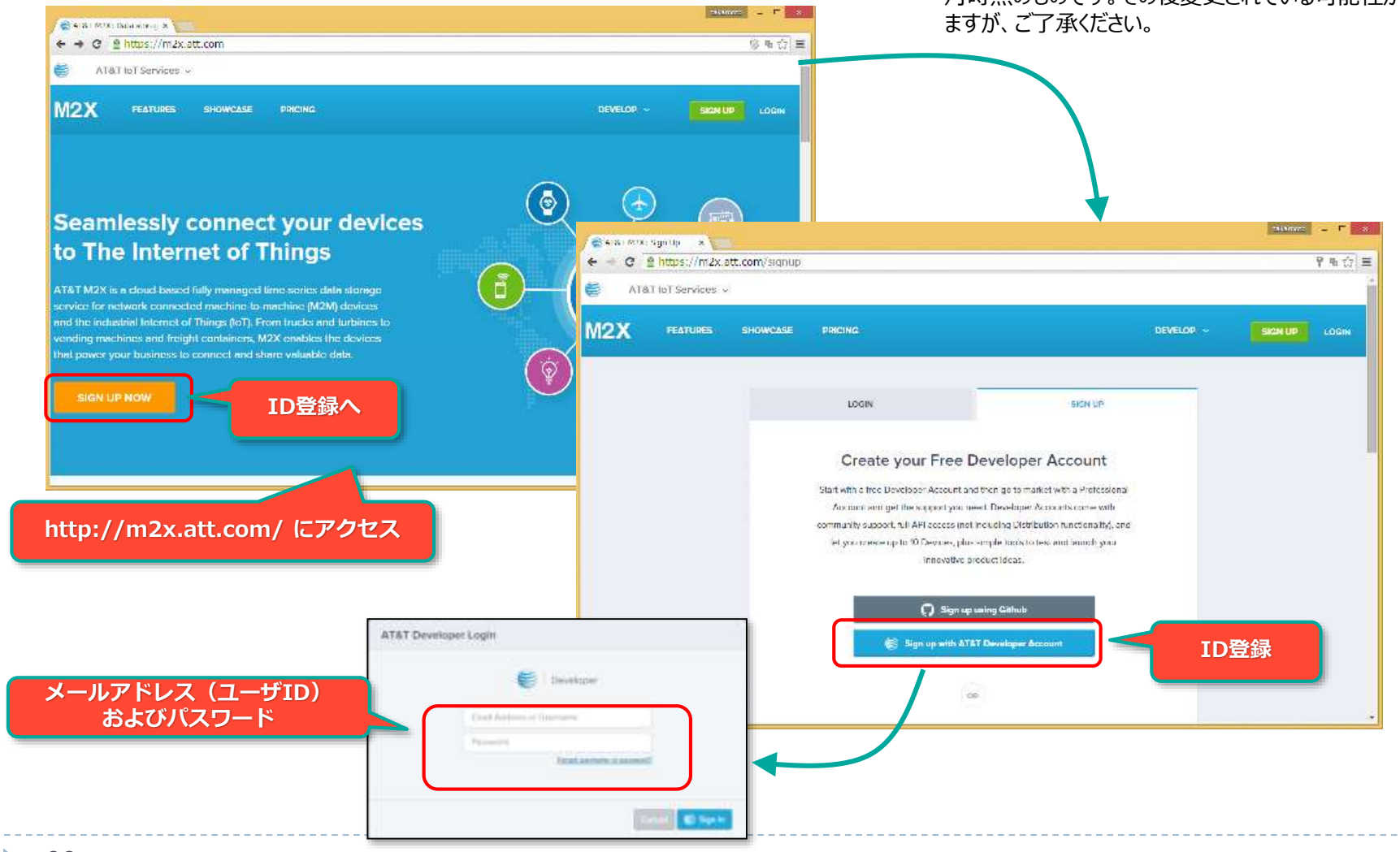
【添付資料.2】tweetトークン取得

- 以下のサイトからトークン取得を行う。
  http://arduino-tweet.appspot.com/

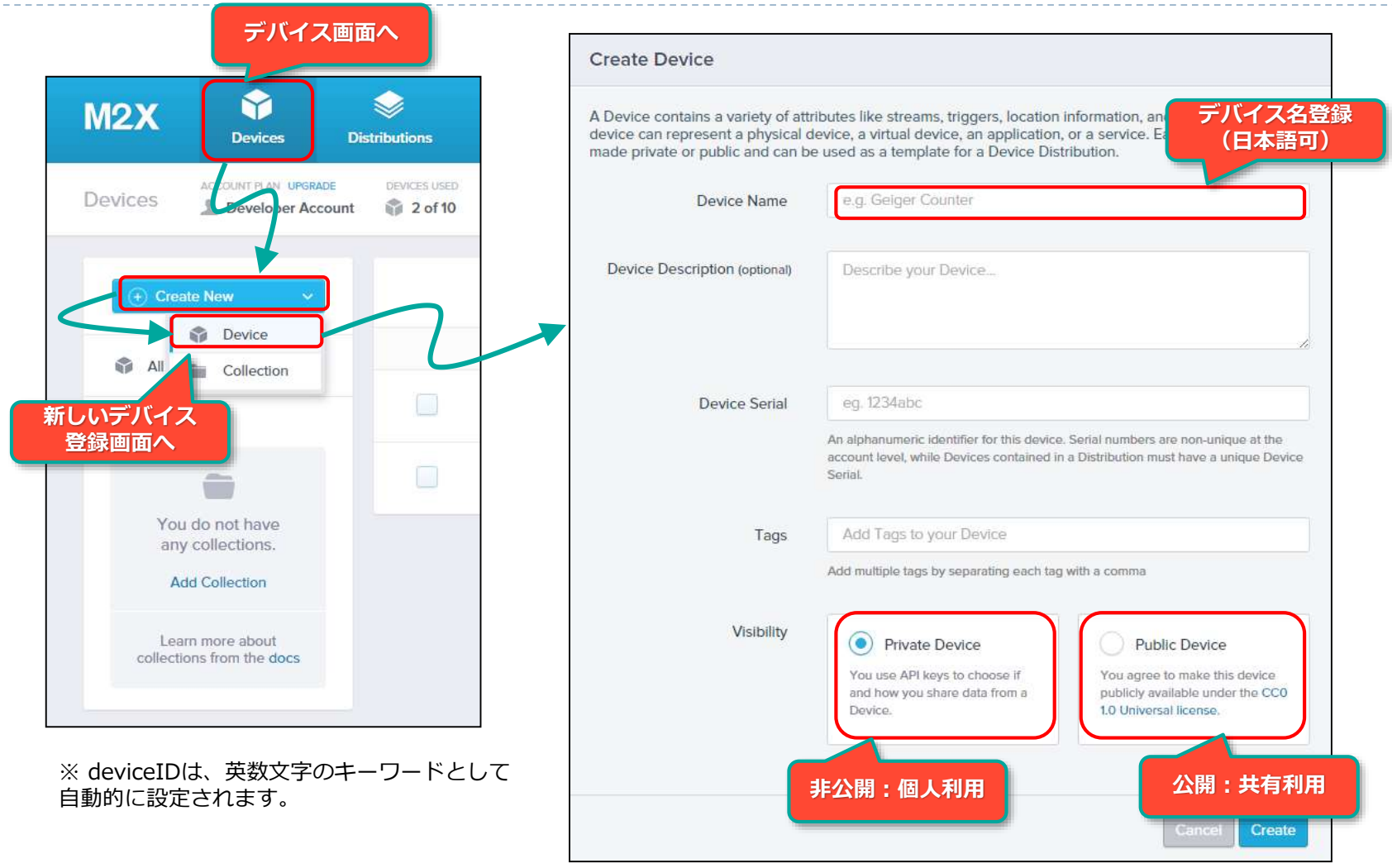


【添付資料.3】 m2xの利用登録 ①ユーザID登録

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。



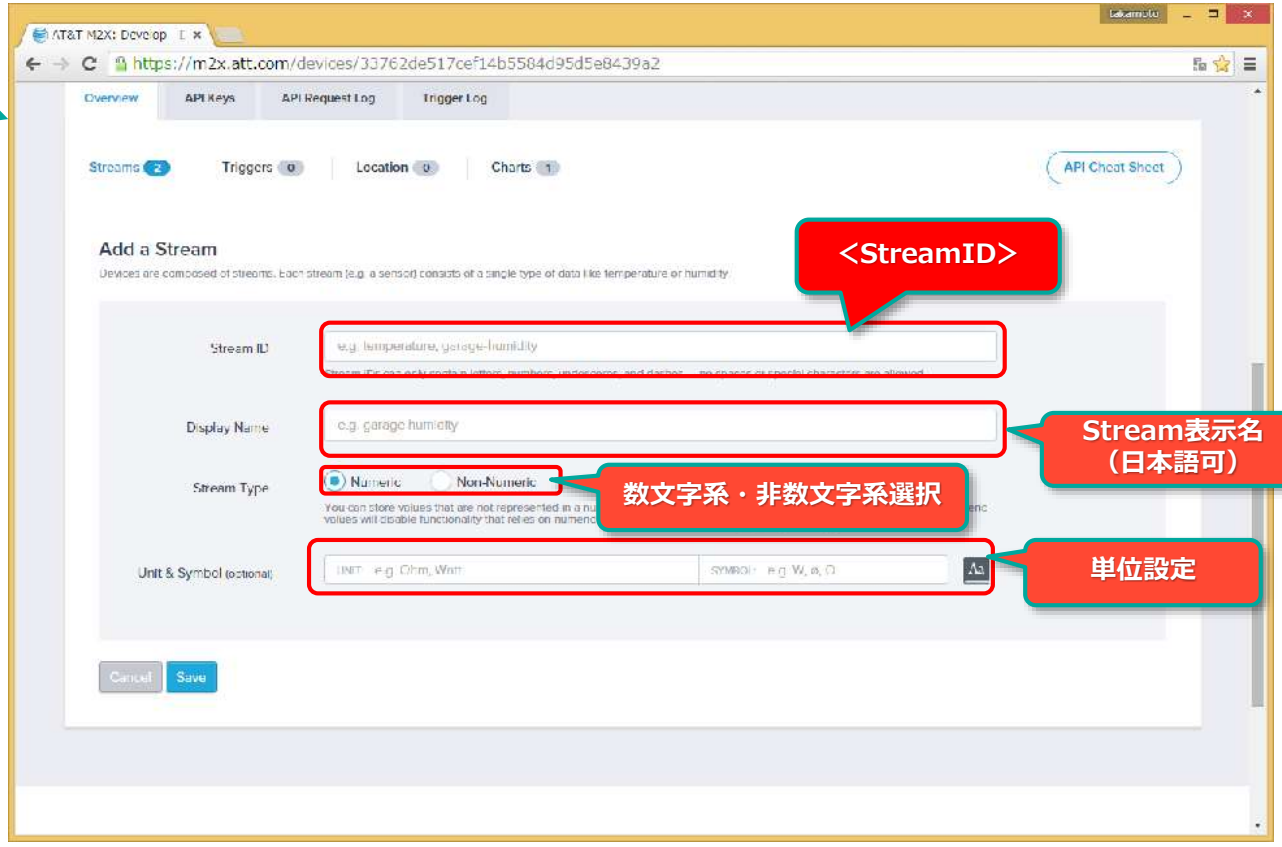
【添付資料.3】 m2xの利用登録　②デバイス登録



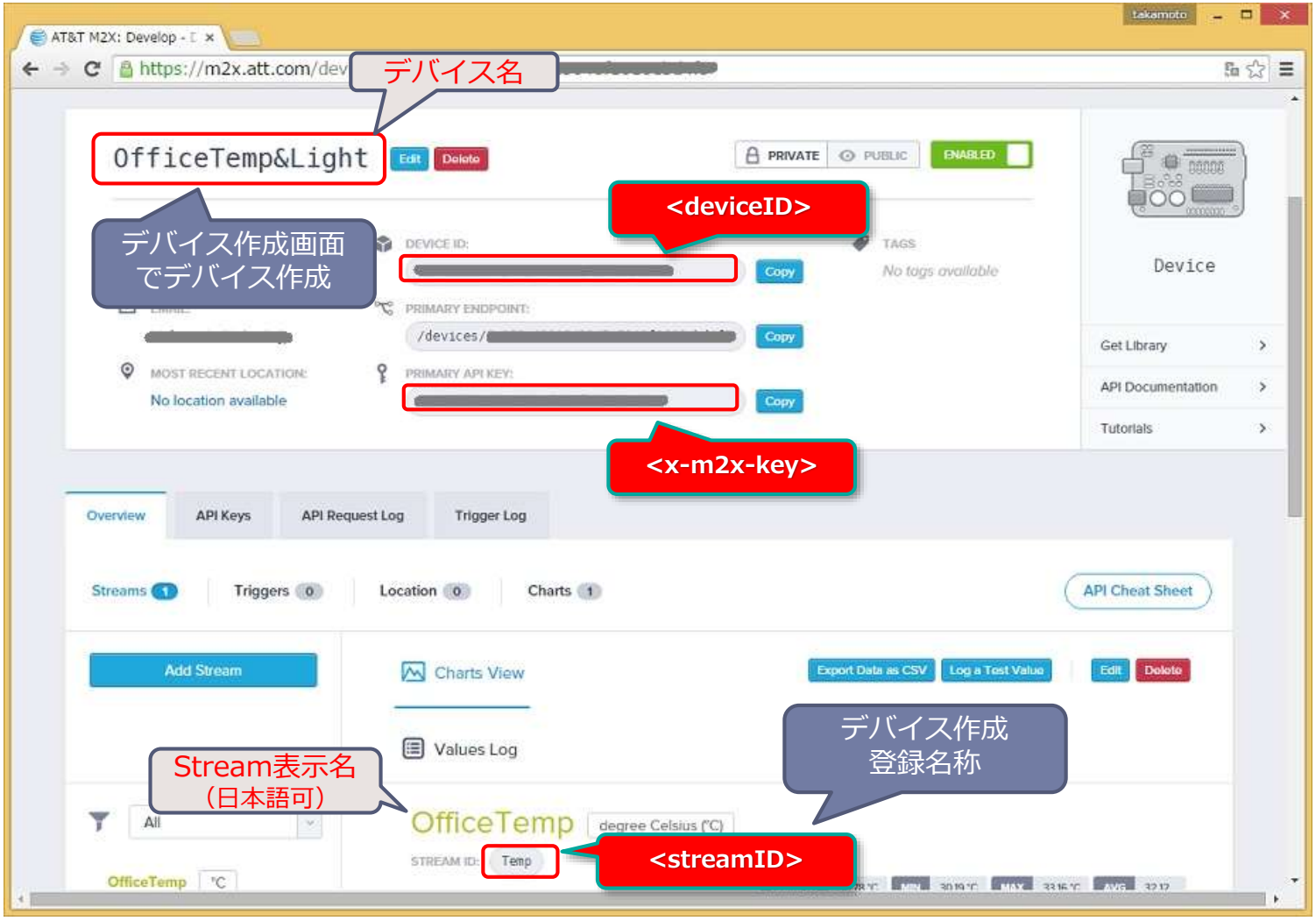
※ deviceIDは、英数文字のキーワードとして
自動的に設定されます。

【添付資料.3】 m2xの利用登録 ③ストリームID登録

※ StreamIDは、入力した名前が
設定されます



【添付資料.3】 m2xの利用登録 ④登録表示画面



【添付資料.3】 m2xの利用登録 ⑤postデータ書式

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）

【添付資料.4】語彙説明

- シールドとは、Arduino本体に装着できる拡張ボードのことを指す。
- スケッチとは、Arduino上で稼働するプログラムのこと。
- レシピとは、Arduino上での配線図やスケッチのこと。作りながら考える意味で、「秘訣」などの意味も含まれる。
- プロファイルとは、SIMカードを認識するためのもの（APN）で、予めHL8548-Gモジュールに、どのSIMカードで利用するかを設定する必要がある。（省略時設定は、別途記載している通り）
- a3gimとは、arduino 3G shield の略で、Arduino上で3gシールドを利用するためのライブラリ群となる。
- ３GIMとは、３G IoT Module の略で、世界最小の３G通信機器（タブレイン製）となる。
- APN（プロファイル）とは、Access Point Name（アクセス・ポイント・ネーム）の略で、携帯電話ネットワーク上のデータ通信で必要となる接続先を指定する文字列となる。通信キャリアごとやMNVO（仮想移動体通信事業者）ごとに固有の名前が付けられている。
- GPSとは、Global Positioning Systemの略で、高度約2万Kmの複数のGPS衛星から電波を受信し、緯度と経度を割り出すシステム。
- gw3gとは、GateWay to 3Gの略で、HL8548-Gモジュールに組み込んだBrewMPのアプリケーション群。
- IDEとは、統合的な開発環境のことを指す。
- HL8548-Gとは、Internet of Everything Moduleの略で、クアルコム設計の通信モジュールの概念となる。
- IMEIとは、International Mobile Subscriber Identityの略で、携帯端末の識別番号で、HL8548-G製品版のIMEIは３G通信モジュール（HL8548-G）の固有識別番号となる。
- MVNO（仮想移動通信事業者）とは、Mobile Virtual Network Operatorの略で、携帯電話やPHSなどの物理的な移動体回線網を自社で持つことなく、通信キャリアから借りて、自社ブランドで通信サービスを行う事業者のこと。
- RSSI値とは、Received Signal Strength Indicatorの略で、「受信信号強度」のことを意味し、受信機入力に入る受信信号の強度を示す数値となる。
- UNICODEは、世界的な標準の文字コードのひとつである。
- UTF-8は、UCSとUNICODEで使える8ビット符号単位の文字符号化形式及び文字符号化スキーム。

【添付資料.5】電源供給について

- ３Gシールドは、これまでの調査で、一時的に約200mA~600mAの電力を消費することが分かりました。
- このことで、電波状態やセンサ類の利用などの状況によっては、PCによるUSB電源（５V500mA）では、電源供給できない状態になり、通信できない場合が発生します。この障害の場合には、外部電源（推奨は９V1.3A）を別途、取って頂くことで、通信が可能となります。
- この他、USB接続ではなく、外部電源のみで利用される場合には、消費電力（特に電流）をチェック（測定）して、安定したバッテリー電力を使ってご使用ください。（購入時に提供される別資料「**３Gシールド＆３GIM利用のための電源供給について**」を参照ください）
- 二次電池（ニッケル水素やリチウムイオン）などのバッテリーをご利用される場合には、電池の電圧・電流（その積による電力）能力を十分満たすものを外部電源としてご利用ください。外部電源は、電源コネクタ部分、あるいはVinとGNDの２つのピンから供給することができます。