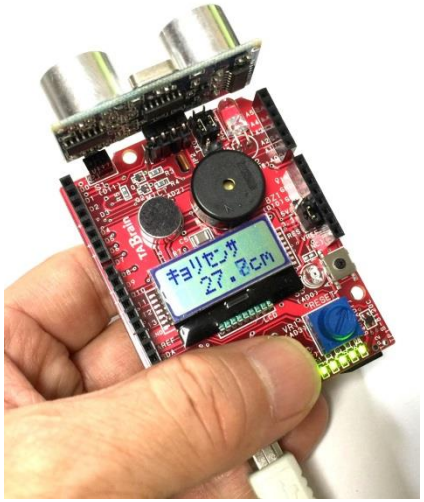
IoTAB SHIELD™ 4.0

UNO版



IoTABシールド V4.0は、全12個の電子部品を搭載し、オープンソースハードウェアの業界標準（デファクトスタンダード）であるArduino UNO R3やGenuino101上で、誰もが簡単に短時間にセンサ類やLED・LCDなどが活用できるようにした教材キットです。
さらにオプションの３GIMを利用すればIoT教材キットとして、メール送信やツイッタ連携、クラウド連携なども簡単に実現できる製品となります。

<参考資料>
・ ３GIM V2.２利用マニュアル

MCPC 公認版

IoTABシールド活用テキスト

サンプルスケッチ等は http://tabrain.jp/IoTABS/IoTABSV4.0sampleV.zipからダウンロードできます

製造・販売 ：株式会社タブレイン

IoTABシールドは「頭脳を活用支援するIoT技術」を意味する拡張ボードです。
この拡張ボードによって、自らの頭脳を鍛えてみては如何でしょうか？
これを利用することで、IoTデバイス関連のモノづくりの楽しさが分かるはずです。

Ver4.１ 2019/0１/12改訂

TABrain

※ 高度ポリテクセンター・ポリテクセンター中部などでの教材として利用中

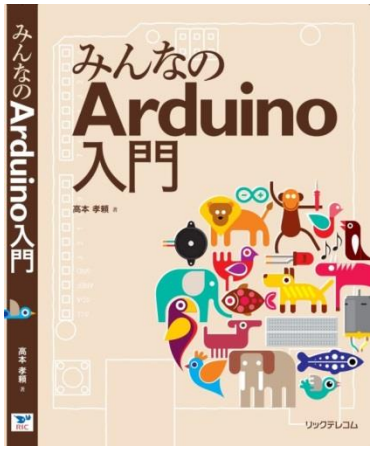
もくじ

第Ⅰ編　準備編
- 第１章　IoTABシールドの概要
- 第２章　Arduinoとは
- 第３章　Arduinoの基本

第Ⅱ編　技術編
- 第４章　Arduino UNO の初級利用
- 第５章　Arduino UNO の応用利用
- 第６章　IoTABシールド入力部品
- 第７章　IoTABシールド出力部品
- 第８章　赤外線リモコン
- 第９章　IoTABシールド応用事例

第Ⅲ編　展開編
- 第１０章　３GIMとは
- 第１１章　３GIM+IoTABシールド
- 第１２章　IoTシステム開発事例

第Ⅳ編　補足編
- 第１３章　Arduinoプログラミング文法
- 第１４章　補足資料
- 第１５章　ちょっとしたチップス
- 添付　IoTABシールド検査テスト



Arduino初心者の方は、
こちらの本を推奨します。
Arduinoの基本が学べます。
2018年5月時点 第１版5刷

（注意）
本マニュアルは、Arduino UNO をベースに記述しています。
よって、電圧 5 V系が基本となっています。
（一部3.3V系のArduino互換機では使えない部品もあります）
他のArduinoでお使いの場合には、注意が必要となります。

第Ⅰ編　準備編

第1章
IoTABシールドの概要

1．はじめに

オープンソースハードウェアArduinoによって、誰もが簡単にモノづくりの世界へ入っていくことができるようになりました。しかし、Arduino単体だけでは、たいしたことはできず、電子部品を買い揃えて、スケッチと呼ばれるプログラミングの作成が必要となります。初心者にとって、この電子部品を揃え、動くスケッチまでたどり着くには、ハードルがいくつもあり、多くの無駄な時間を過ごすことになります。

例えば

1）必要な電子部品をどこで揃えるの？

2）はんだ付けやブレッドボードとのケーブル接続はどうするの？

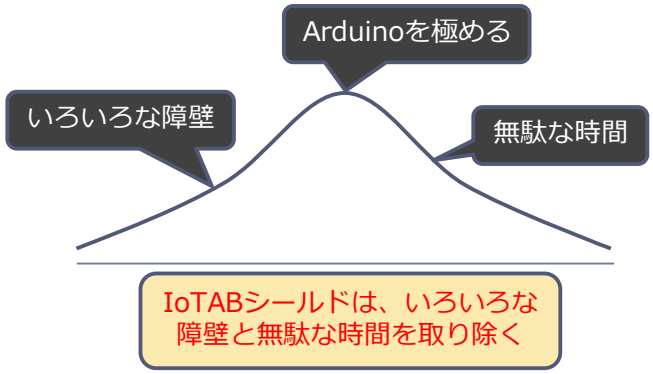
3）抵抗やコンデンサなどはどう使うの？

4）アナログ・デジタル・シリアル通信はどのように使い分けるの？

5）使う電子部品のサンプルスケッチはネット上のどこにあるの？

6）複数の電子部品の組み合わせだと複雑な配線とスケッチをどうしたらいいの？

などのようなことを多く経験します。



ここで紹介する**IoTABシールド**は、このようなことを一挙に解決する環境を提供致します。電子・電気の知識がほとんどない初心者でも、モノづくりを楽しむには、買ってきたキットが、なるべく興味のある間に、思い通りに動くことが重要です。しかも自分の思い通りに組合せ、変更し、あらたに成長させていく楽しみが必要となります。

この**IoTABシールド**には、多くのセンサ類をはじめとする入力電子部品やスピーカ、LED、それに液晶ディスプレイ（LCD）まで持ち合わせていて、豊富なサンプルスケッチで、すぐに思い通りのモノづくりに到達することが可能となっています。

IoTABシールドは、電子・電気が専門外である機械系、情報系、建築系などの工学系、理工系、さらには文系までの方々にも利用できるものとして開発しました。

是非とも、身の回りで役立つモノづくりに使ってみて、新たな発見をする喜びを感じ取ってみては如何でしょうか。

2．IoTABシールドとは

IoTABシールドは、**オープンソースハードウェア**Arduino上で電子部品を接続配線することなく、誰もが簡単に使えるようにした拡張ボードと本マニュアル（スケッチ込み）を含むキットです。

入力部品となる多くのセンサやスイッチ、可変抵抗などから、外部出力となるLCD、LED、スピーカなどを自由に組み合わせ、複雑かつ高度なシステムを、いちはやく構築することが可能です。システムの極意は「**システム＝入力＋処理＋出力**」で、

入力＝センサ類・可変抵抗器・スイッチなど

処理＝プログラミング（スケッチ）

出力＝LCD（液晶ディスプレィ）・LED・スピーカ

となります。

これらの組合せ数は、実に**8,000通り以上**。さらに外部の電子部品を使えば、無限大に広がります。

特に3GIM/4GIMなどを利用すれば、簡単にIoTデバイスの試作・開発、利用が可能となります。

また、３GIMと組み合わせれば、IoTシステム開発の教材としても利用できます。具体的には、センサネットワークによるクラウドにセンサ値を集めることや、遠隔での制御（操作）、それに遠隔での監視などが可能となります。

IoTABシールドのメリット

1）電子・電気の専門知識はまったく不要
2）プログラムだけで電子部品が利用可能
3）わずか数時間で使えるサンプルスケッチ付
4）さらに自由に電子部品を組み合わせ可能
5）３GIMとの連携による遠隔監視・制御
6）創造豊かなモノづくり環境提供

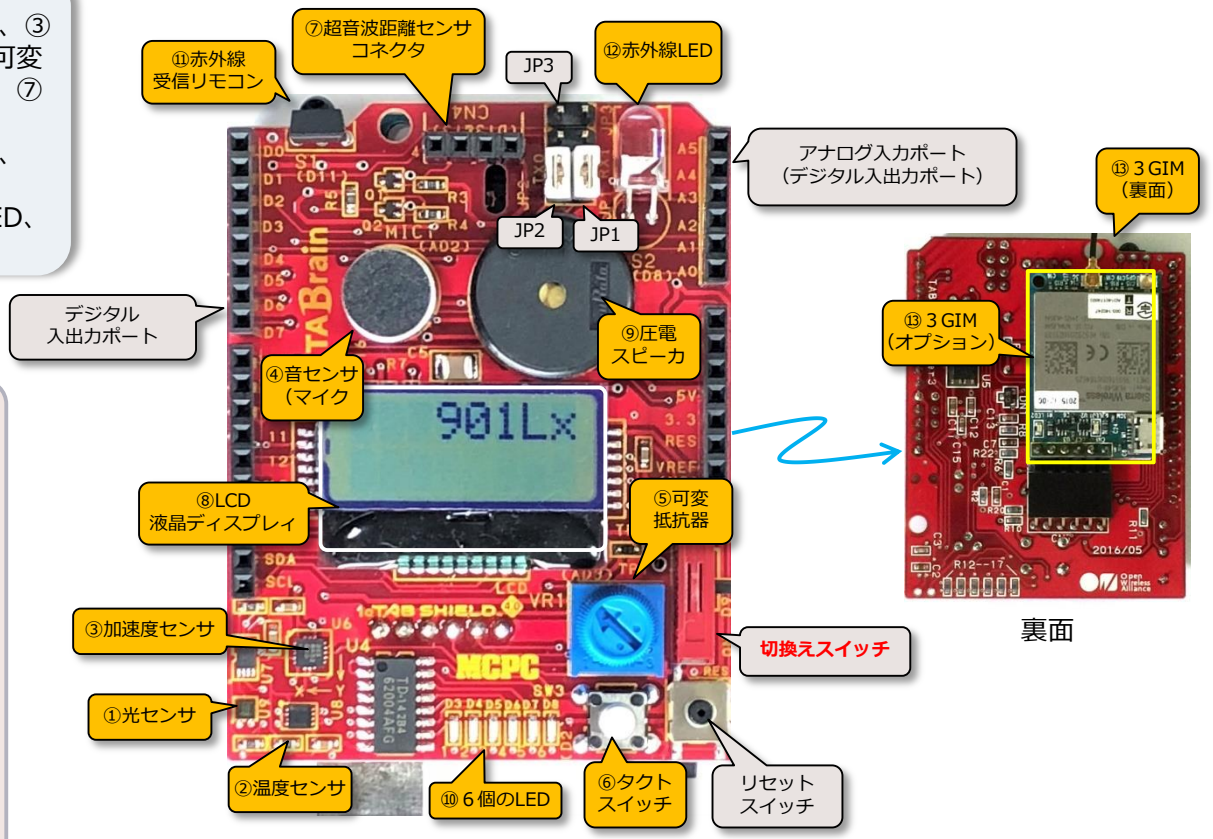
3．IoTABシールド概要機能 ①

IoTABシールド上装備の電子部品

入力電子部品：①照度センサ、②温度センサ、③加速度センサ、④音（マイク）センサ、⑤可変抵抗器（ボリューム）、⑥タクトスイッチ、⑦超音波距離センサ（外付け）

出力電子部品：⑧LCD（液晶ディスプレイ）、⑨圧電スピーカ、⑩６個のLED

その他：⑪赤外線受信リモコン、⑫赤外線LED、⑬３GIM用インタフェース



※基板上に、電子部品の接続ポートがシルクで記載されています。

Arduino上で何ができるか？

1．Arduino＋IoTABシールドを使うことで、さまざまな学習が可能となります。
　入力センサと出力電子部品との組合せによるシステムの構築が簡単にできます。

2．試作品開発でのツールとして利用できます。
IoTABシールドが持つ多くのセンサは、簡単に利用できる環境にあり、スケッチも揃っていることから、すぐに試作品に組み込んで利用できます。

3．DIYとしてオリジナルのモノづくりに利用できます。防犯システムや自動カウンタ、超音波距離メジャー、オリジナルテレビリモコンなど、発想によっては、多岐にわたるユニークな製品づくりができます。

その他（ジャンパピン）

JP1: UART Rx切換え（D0／D4）
JP2: UART Tx切換え（D1／D5）
JP3: Rx/Tx

3． IoTABシールド概要機能②

この一覧表では、各電子部品の製品・入出力ポートなどを掲載しています。

No.	電子部品	製品/入出力	概要（部品型名等）
①	光センサ	シリアルI2C 0x29	BH1780
②	温度センサ	シリアルI2C 0x4A	STS30
③	加速度センサ	シリアルI2C 0x1C	MMA8452Q
④	音センサ（マイク）	アナログ IN:A2	C9767BB422LFP
⑤	可変抵抗器	アナログ IN:A3	3386K-EY5-103TR
⑥	タクトスイッチ	デジタル IN:D2	akizuki P-03648
⑦	超音波距離センサ（外付け）	デジタル IN/OUT: D12/D13	HC－SR04

No.	電子部品	製品/入出力	概要（部品型名等）
⑧	LCD（液晶ディスプレィ）	シリアルI2C 0x7C	AQM0802A-RN-GBW
⑨	圧電スピーカ	デジタル OUT:D10	PKM17EPP-4001-B0
⑩	6個のLED	デジタル OUT:D3 ～ D8	OSYG1608C1A
⑪	赤外線リモコン受信モジュール	デジタル IN:D11	PL-IRM2161-XD1
⑫	赤外線LED	デジタル OUT:D8	OSIR5113A
⑬	3GIM	シリアル通信 UART	オプション製品

※④～⑫ までの電子部品の利用の**切換えスイッチ（D9）**があります。

【注意事項】
※ここで表記の A2とA3は、**アナログ入力ポート番号**で、D0～D13は、**デジタル入出力ポート番号**です。
スケッチ内では、アナログ関連の「A0」から「A5」は直接記述できますが、デジタル関連の「D0」から「D13」までは記述できません。デジタル関連では整数の「０」から「13」を使ってください。

参考：IoTABsシールドV4.0で利用している電子部品情報は、こちらからダウンロードください。
http://tabrain.jp/tabs/IoTABS4_parts_pdf.zip

4．３GIM/４GIM 利用時について

【注意】**３GIM/４GIM 利用時**では、Arduino+IoTABシールドは、利用するArduinoの種類によって以下の**ジャンパピンの切換え**が必要です。

（１）UART（シリアル通信）のジャンパピン

・Arduino UNO ：ソフトウェアシリアル通信（D4、D5）利用（写真１）

・Genuino101 ：ハードウェアシリアル通信（D0、D1）利用（写真２）

・Arduino MEGA：ハードウェアシリアル通信（写真３）

・Arduino M0 Pro：ハードウェアシリアル通信（D0、D1）利用（写真２）

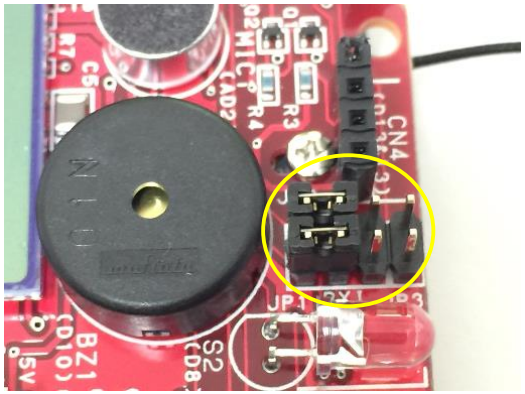


写真１：Arduino UNOの場合
<左端>

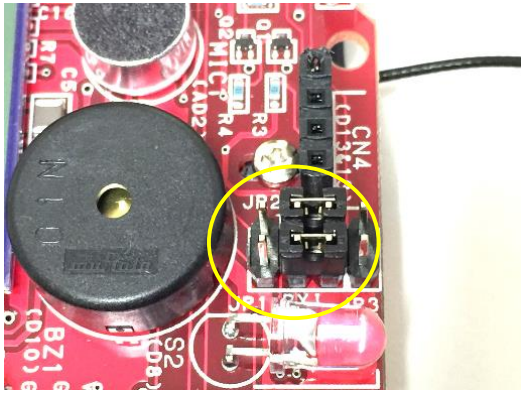


写真２：Genuino101の場合
Arduino M0 Proの場合
<中央>

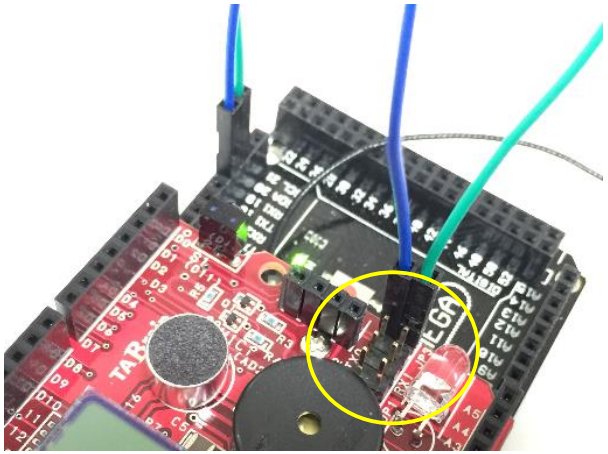
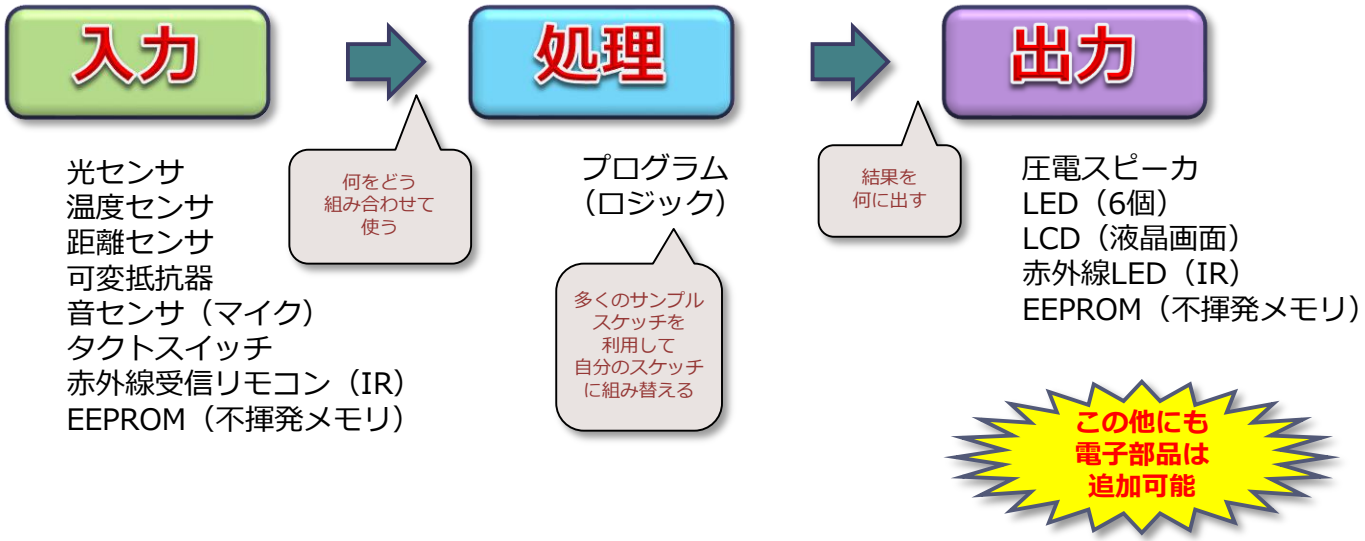


写真３：Arduino MEGAの場合
Rx/Dxをシリアル１に接続

5．IoTABシールドを使うに当たって

- **システム**とは、一般に、組織・組立て・体系・系統を意味します。
- コンピュータのシステムは、以下の３つによって構成されます。

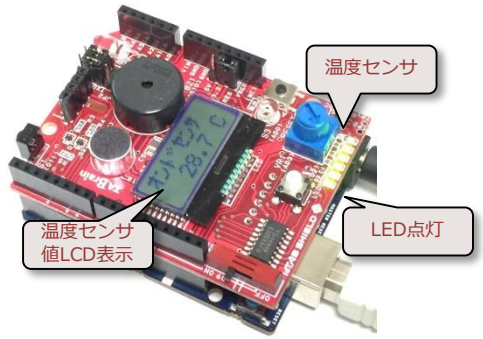


- システムは簡素化することが重要です。
  - シンプリシティ（Simplicity：簡単）にまとめることが重要
  - 分かり易くしたモジュール化が重要
  - 処理の流れを複雑にしないことが重要
    （複雑になるところだけはブラックボックス化）

6．用意されたサンプルスケッチ

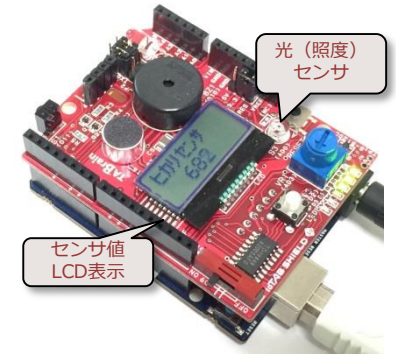
プログラムダウンロード
Program Download

■温度センサ値表示



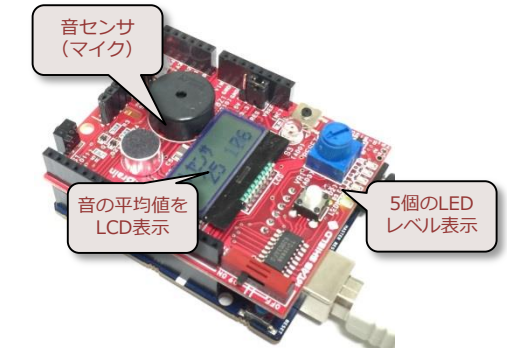
温度センサの値をLCDに表示。（アラーム出力も可）

■照度センサの値表示



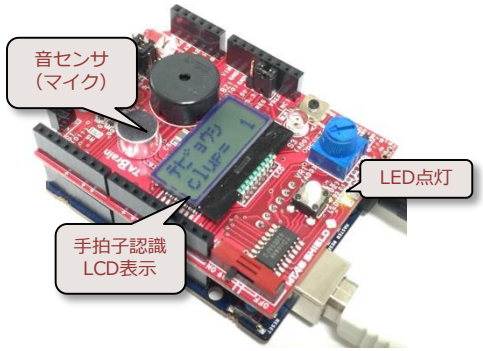
照度センサの値をLCDに表示。5個のLED点灯とも連動。（アラーム出力も可）

■音センサの値表示



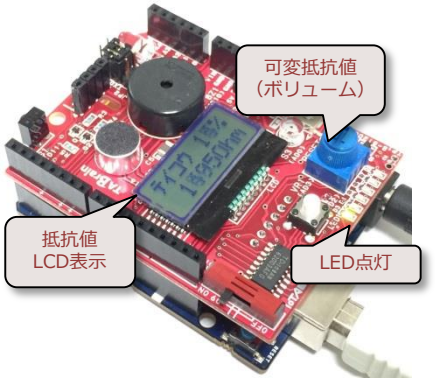
音センサの値をLCDに表示。同時に5個のLEDを使ってレベル表示。ある間隔の平均音量も並列して表示。

■手拍子の認識



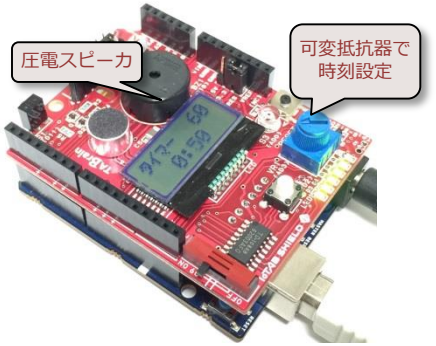
手拍子の数を認識して、その数をLCD・LEDに表示。赤外線リモコンと組み合わせ家電の制御などに利用可能。

■可変抵抗値の値表示



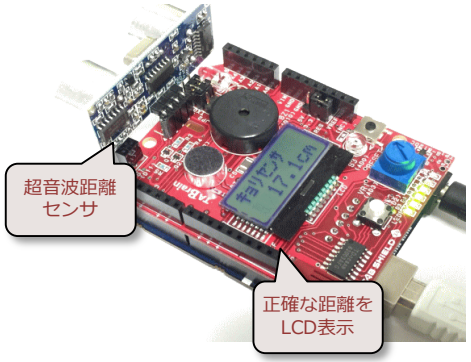
ボリューム値をLCDに表示。同時に5個のLEDも点灯。
このボリュームもいろいろな切換スイッチに利用可能。

■タイマー



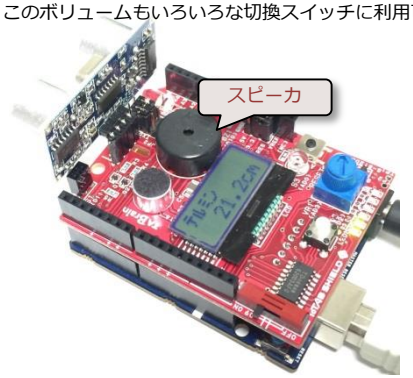
可変抵抗器、タクトスイッチ、LCD、LED、スピーカ、
それに時間関数を使って実現

■距離センサの値表示



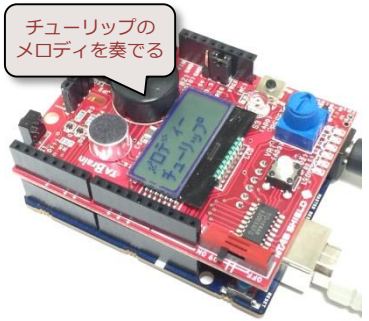
天井までの距離などを正確に表示。近距離の場合
アラームを出すことも。LEDで大まかな距離も表示。
家の見張り役にも使えるかもしれません。

■テルミン（距離センサ）



距離に応じた音階をスピーカから出力

■メロディを奏でる



スピーカを使えば、メロディも奏でることが可能。
その他、テレビ・照明のリモコンを読み取り、赤外線
LEDから送信可能。家庭リモコン替わりにも変身可能。

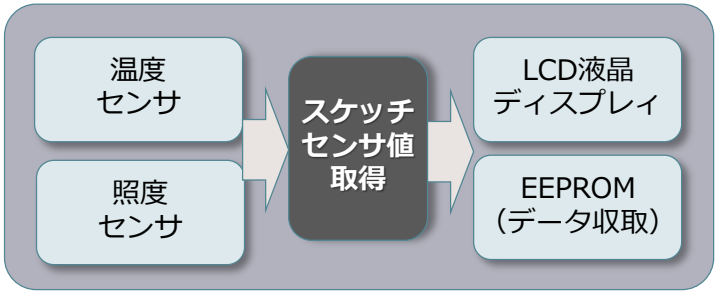
7．応用展開を考える

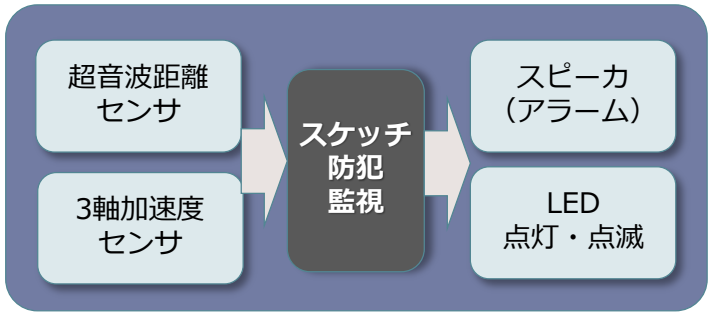
あなたもモノづくりの大発明者に

電子部品をいくつか組合せると、さまざまなアイデアが湧いてきます。

　四六時中、考え、思考し、熟慮することで、新しいアイデアが湧いてきます。

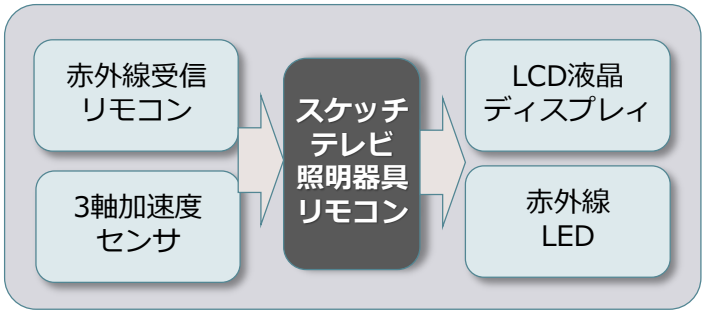
　常に考えていることで、アイデアはふとしたところから出てきます。その発見を楽しんでみてください。





モノづくりを考える
- 普段から困っていることを解決することを考えてみる
- もっと便利なものを考えてみる
- シーズからニーズを考える
- ニーズからシーズを考える
- もっとシンプルにしてみる
- 違った考えをしてみる

工夫する
- いろいろと組み合わせてみる
- 入力と出力を使い分ける
- 操作タイミング（時間差）を考える
- スイッチを簡単に組み入れる（ここがポイント）
- 五感を通じるような入出力を使う（見る、聞く、感じる）
- 音・光・熱 を捉える
- 記憶し、それを使う（学習する）
- ワイヤレス・無線（赤外線、３Gなど）を活用する
- 操作性を増やす（加速度センサやスイッチを活用）
- 自動化を考える



第２章
Arduinoとは

1．Arduinoの紹介 ①

オープンソースハードウェアとは

・ 設計図（回路図）が無償で公開
→ 類似の製品が作れる
→ クローンが出回る

・ ソフト開発する統合開発環境（IDE）は無償で提供される

・先駆者が既に実施してきたことに対して犯してはならない暗黙のルール


Arduino UNO


GR-SAKURA


What’s Next Yellow

Make: Japan

Make:Japan 2012/03/02 記事
オープンソースハードウェアの｛暗黙の｝ルール

Arduinoは、オープンソースハードウェアのデファクトスタンダード（業界標準）

1．Arduinoの紹介 ②

Arduinoで何ができる

- Arduinoで、電子工作ができる
- Arduinoで、簡単にLED・センサなどが操作できる
- Arduinoで、電子工作の制御（コントロール）できる
- Arduinoで、ロボット・ドローンが開発できる
- Arduinoで、３Dプリンタまで作れる
- Arduinoで、モノ作りが簡単になる
- Arduinoで、IoTデバイスが開発できる
- Arduino互換機が開発できる

- 安心・安全な世の中にするためのモノ作り革命が起きる
  誰もが、安価で、短期間で、簡単に モノ作りできる

1．Arduinoの紹介 ③

Arduino（ハード）　と　IDE（ソフト）の準備

- 予め必要な環境は、Arduinoの購入と、Arduinoの統合開発環境（IDE）インストールが必要。
  - Arduinoの購入は、ネット上から簡単に購入可能。
    - アマゾンからの購入:http://amazon.co.jp/
    - スイッチサイエンス社のサイトからの購入： http://www.switch-science.com/
  - ArduinoのIDEのインストールは、以下のサイトから（無償。寄付あり）
    - http://arduino.cc/en/Main/Software

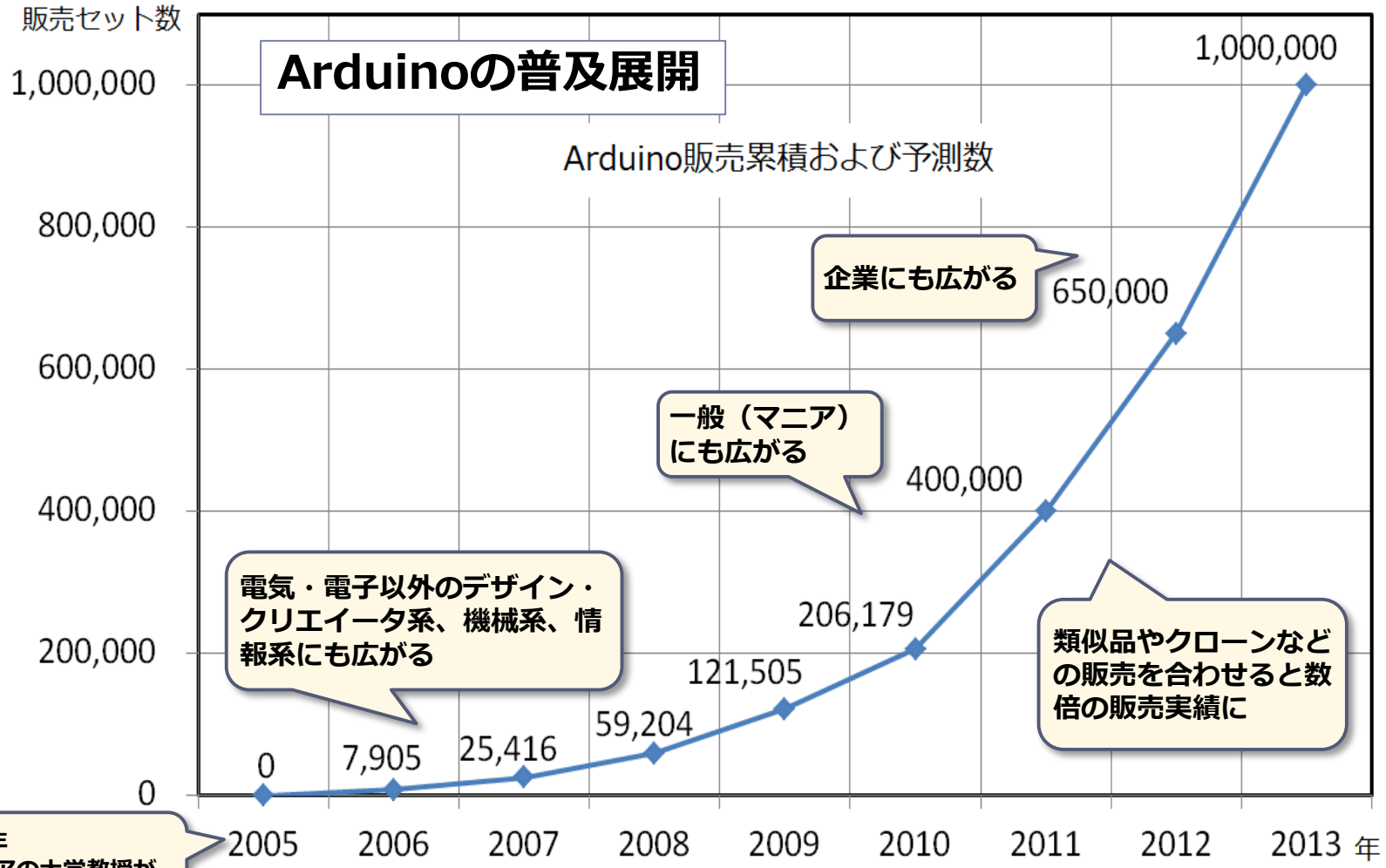
Atmel AVR（8ビットマイコン）など

現在は、32ビットマイコンも利用可能

＋

統合開発環境　Arduino
（C＋＋言語風の開発環境）

1．Arduinoの紹介 ④



※MITのOCWでも多くのArduinoを使った授業が公開
OCWとは、Open Couse Ware のこと

1．Arduinoの紹介 ⑤

Arduinoの人気の秘密

① 価格が安い　→　シンプル・クローン品も多い

② 短期間でできる　→　組み合わせが簡単

（多くのソフト・部品がある）

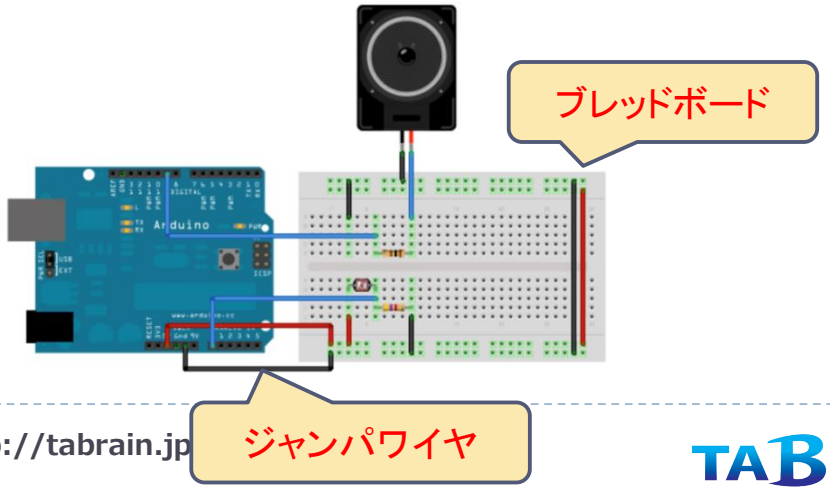
③ 技術ハードルが低い　→　専門的な知識はそれほど必要ない

（マイコン独自の知識もほとんど不要）

④ 利用できる財産が豊富→　多くのWebサイトの知的財産が利用可能

⑤ 製造リスクの軽減　→　特許侵害などの心配が少ない

⑥ 試作や量産の容易化　→　経費削減でき、迅速に対応可能

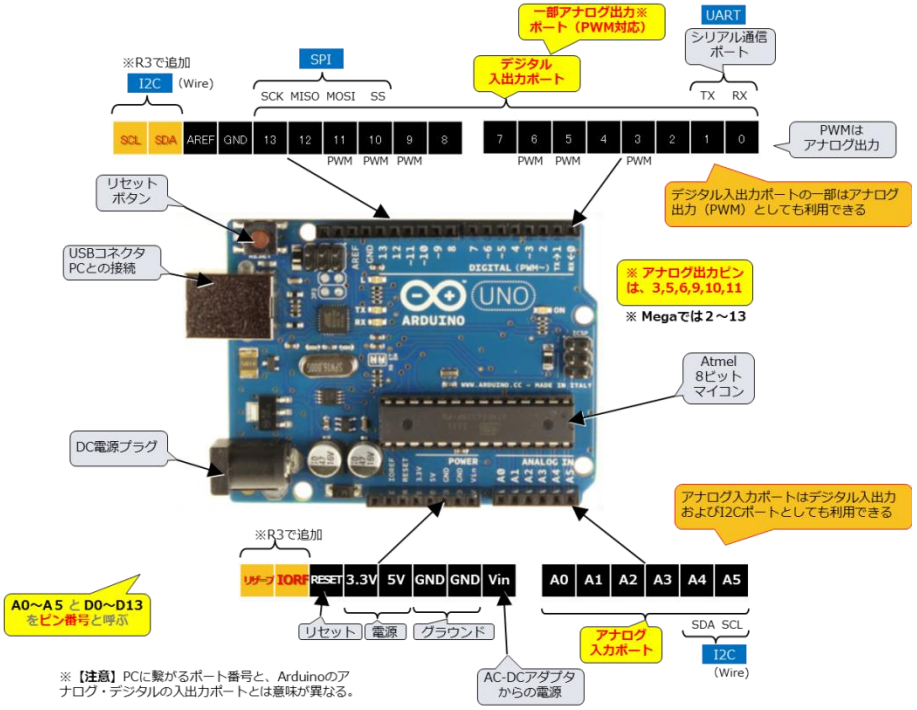


1．Arduinoの紹介 ⑥

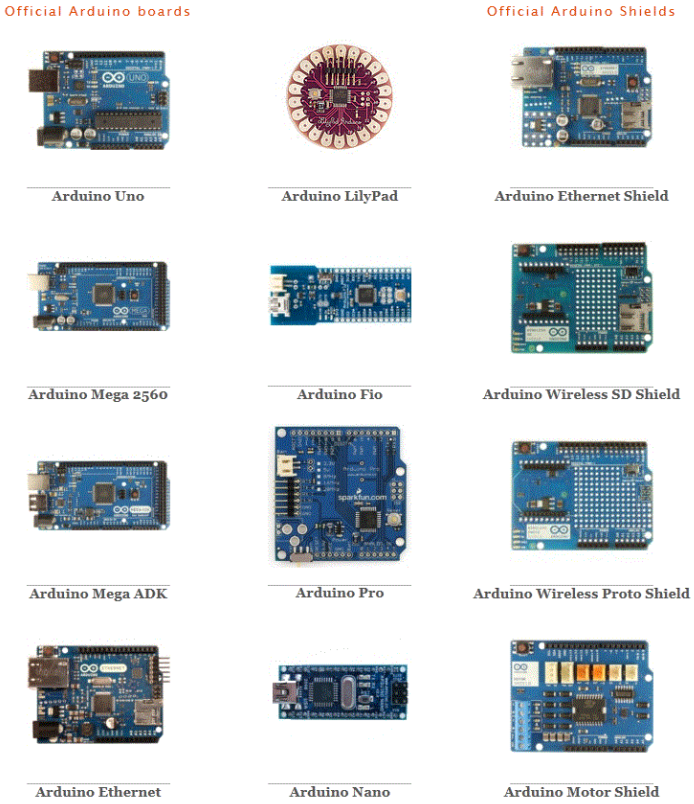
Arduinoのハードウェア

基本は、Atmel AVRの8ビットマイコン。拡張製品も品揃え

- これまで多くのバージョン・製品・拡張品が出てきているが、現時点での標準のArduinoは、UNO（Ver.3）となっている。この他にも機能的に同じ小型化したものや、拡張できるMEGAなども存在。今後は、ARMマイコンを使った製品も出てくる予定。



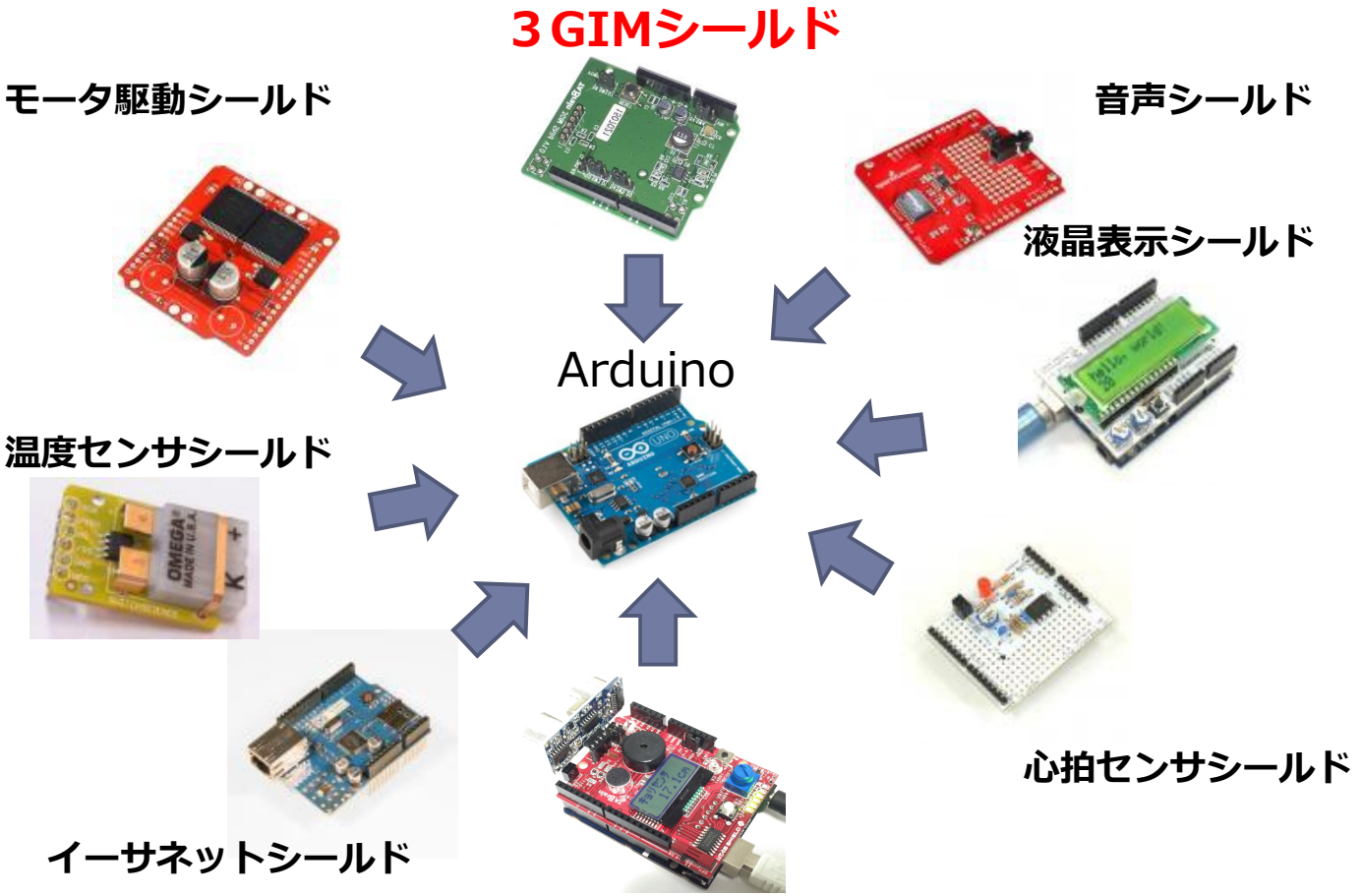
Arduinoファミリー（一部）



UART（シリアルデータ変換）		1K EEPROM
2K RAM（ワークメモリ）	CPU	32Kb フラッシュメモリ
入力/出力ポート		

1．Arduinoの紹介 ⑦

豊富なArduinoシールド（センサなどが搭載された拡張ボード）



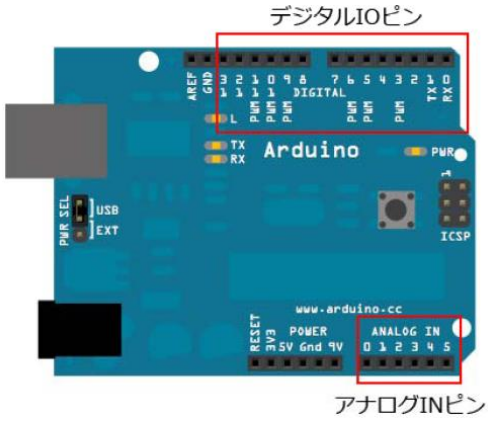
1．Arduinoの紹介 ⑧

Arduinoの構築環境

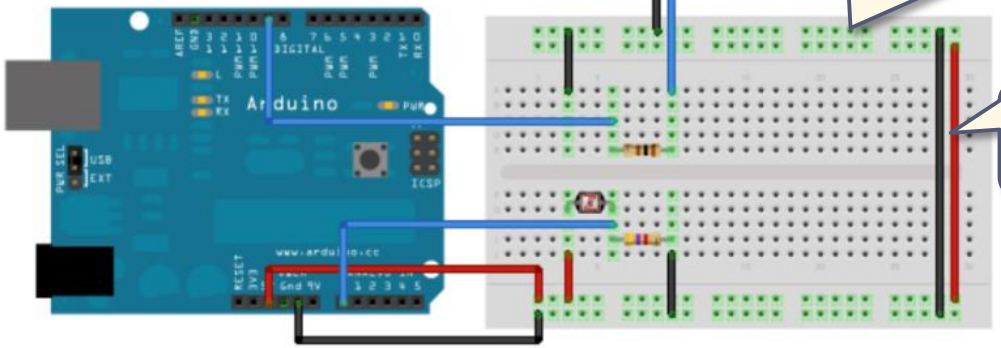
ロボットの製作でも
多く利用されている。

豊富なスケッチが
Webサイトに

豊富なセンサ・
シールド類



ブレッドボード



ジャンパワイヤ

試作が
安く・早く
簡単に
構築可能

はんだ付け不要

1．Arduinoの紹介 ⑨

Arduinoシリーズ

主力製品

最新版の32ビットCPU使用

Arduino　仕様	UNO R3	Leonardo	Mega 2560	Due	Pro	Genuino101
マイクロプロセッサ	ATmega328	ATmega32u4	ATmega2560	AT91SAM3X8E	ATmega168/328	Intel Curie
動作電圧	5V			3.3V	3.3V/5V	3.3V（5V耐性）
推奨入力電圧	7－12V				3.35－12V(3.3V) 5-12V（5V）	7－12V
制限入力電圧	6－20V					7-20V
デジタルI/Oピン数	14（うち6ピンPWM出力）	20	54（うち15ピンPWM出力）	54（うち12ピンPWM出力）	14（うち6ピンPWM出力）	14（うち4ピンPWM出力）
PWMチャネル	6	7	15	12	6	4
アナログI/Oピン	6	12	16	I：12/O:2(DAC)	6	6
I/Oピン電流	40mA	40mA	40mA	130mA	40mA	20mA
3.3V供給可能電流	50mA	50mA	50mA	800mA		
フラッシュメモリ	32K（うち0.5KBはブートローダ用）	32K（うち4KBはブートローダ用	256KB（うち8KBはブートローダ用）	512KB（ユーザアプリケーション用）	16KB(168) 32KB(328)	196kB
SRAM	2KB	2.5KB	8KB	96KB（2バンク：64KB・32KB）	1KB(168) 2KB(328)	24KB
EEPROM	1KB	1KB	4KB		512B(168) 1KB(328)	付属：6軸加速度・ジャイロセンサ
クロック周波数	16MHz	16MHz	16MHz	84MHz	8MHz(3.3V) 16MHz(5V)	付属：BLE、RTC 32MHz

【arduino.ccサイト参照】

1．Arduinoの紹介 ⑩

Arduinoの魅力

- 技術的なハードルが低く、簡単に利用可能
  - センサやアクチュエータなどを簡単に繋いでみることができる。
  - マイコン独自の知識がほとんど不要で、すぐに利用できる。
  - 短時間で、試作・プロトタイプ版開発ができる。
  - 多くのサンプル・プログラムやシールド（拡張キット）・部品群が繋がる
  - Arduino関連の価格が安い（リーズナブル）
- 普及の勢いがある理由・背景
  - 当初は電気・電子の学生のためのマイコンボードだったが、デザイナ系・機械系・情報系などへの**学生**も扱うようになった広がりがある。（回路図・基板図などの理解なしで利用可）
  - オープンソースハードウェアであることから個人（**マニア**）達のファンが急激に増えてきた
  - 難しいハードとプログラムの試作・プロトタイプが、簡単に、しかも低コストで、短期間にできるようになったメリットを感じるようになって、**企業**での利用も**試作・量産に**使い始めている。

    （2012年11月26日号「日経エレクトロニクス」にてArduinoの記事にて）
- 試作された作品の魅力
  - 限られえた数のロット開発などは、Arduinoを使ってそのまま実用にも使える。
  - 試作したものでは、ガイガカウンタ（放射線測定器）や、３Dプリンタなど高機能なものまで出てきている。

（考えたものが、直ぐに簡単に低コストで開発・試作できことが魅力）
  - 特許侵害などの心配が少ない（図面のオープン化でクローンなどの開発が容易）

2．IoTABシールド活用準備手順 ①

ここでは、IoTABシールドを使う上での準備手順となります。

1）Arduino および USBケーブルの準備（**購入**）

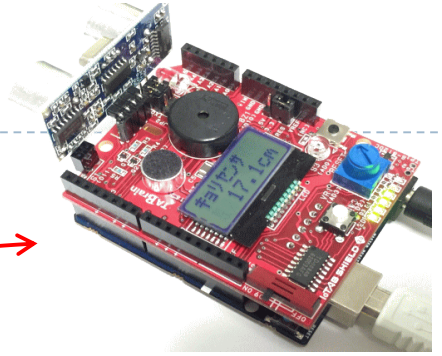
2）PC上に統合開発環境（IDE）の準備（**無償ダウンロード**）
① PC （Windows版・Mac版・Linux版など）の準備
② WebサイトよりIDEダウンロード・インストール
③ 開発環境の設定
④ ドライバー設定

初回のみ実施

3）**IoTABシールドを動かしてみる**
① PCとArduino接続
② 接続ポートの確認
③ サンプル（スケッチ）の起動・確認

※Arduino IDEは、arduino.cc または arduino.org からダウンロードできます。
この説明書では、arduino.cc サイトで紹介しています。

2．IoTABシールド活用準備手順 ②



1）Arduino 関連ハードの準備 （購入）

Arduino ボード + USBケーブル

+

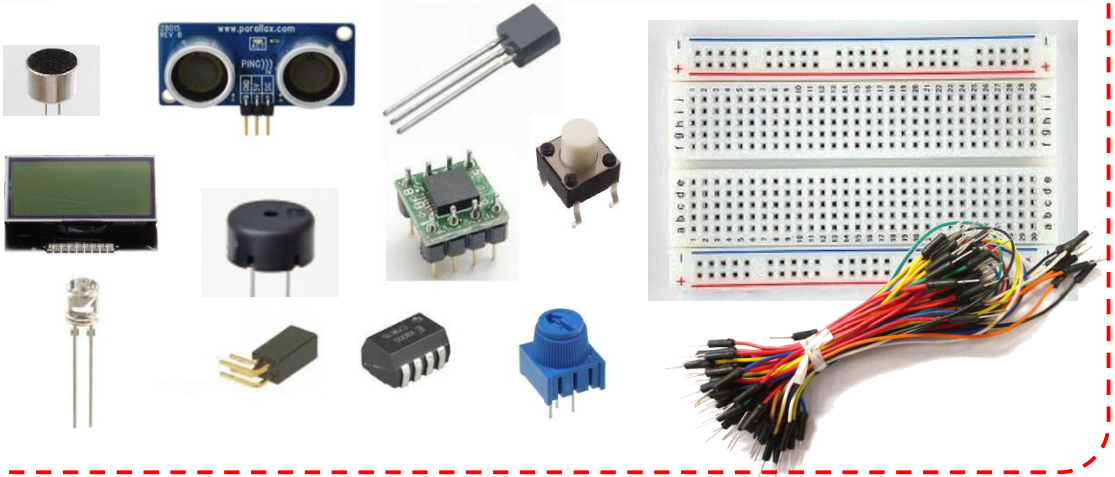
電子部品※
センサ類
モータ類
LCD・LED
ほか

+

ブレッド ボード + ジャンパ ワイヤ

Aタイプ

Bタイプ



IoTABシールドは、この部分が不要に

2．IoTABシールド活用準備手順 ③

2）PC上に統合開発環境（IDE）の準備

統合開発環境（IDE）は無償ダウンロード



Arduino.cc から 統合開発環境（IDE）をPC上にダウンロード

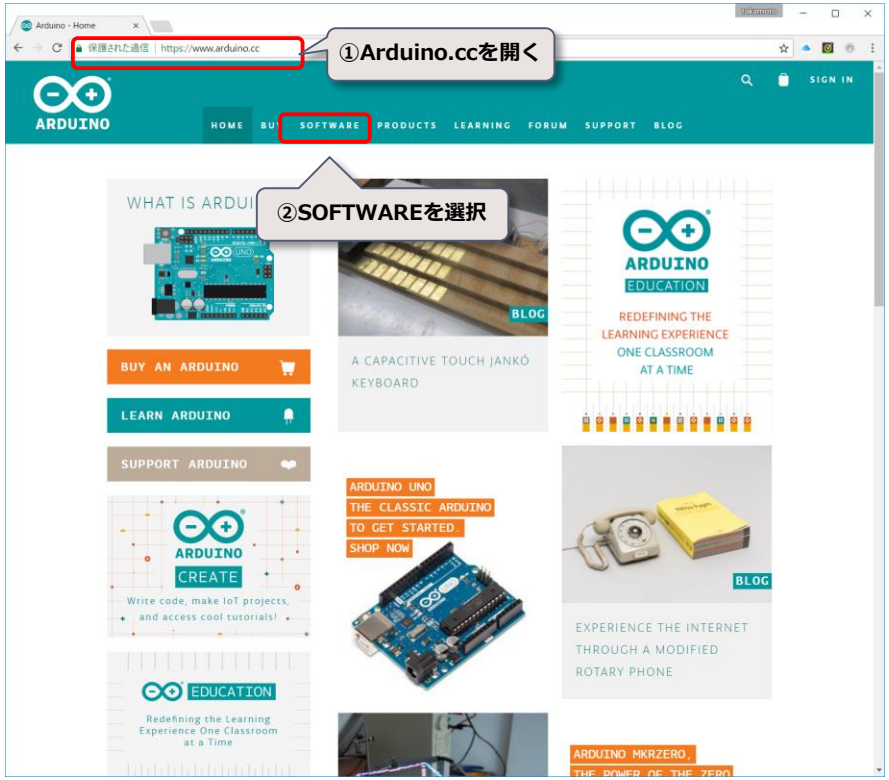
統合開発環境（IDE: Integrated Development Environment）とは
プログラム開発する上で必要な、エディタ、コンパイラ、ビルダ（リンカー）、デバッガなどが統合されて提供されている環境のこと

※ ArduinoのIDEは、C++言語をベースとする開発環境となる
　その他にも Java系のProcessing もある。

【インストールは次頁以降】

3．Arduino IDE インストール手順①

- Arduino ホームページにアクセス：　www.arduino.cc



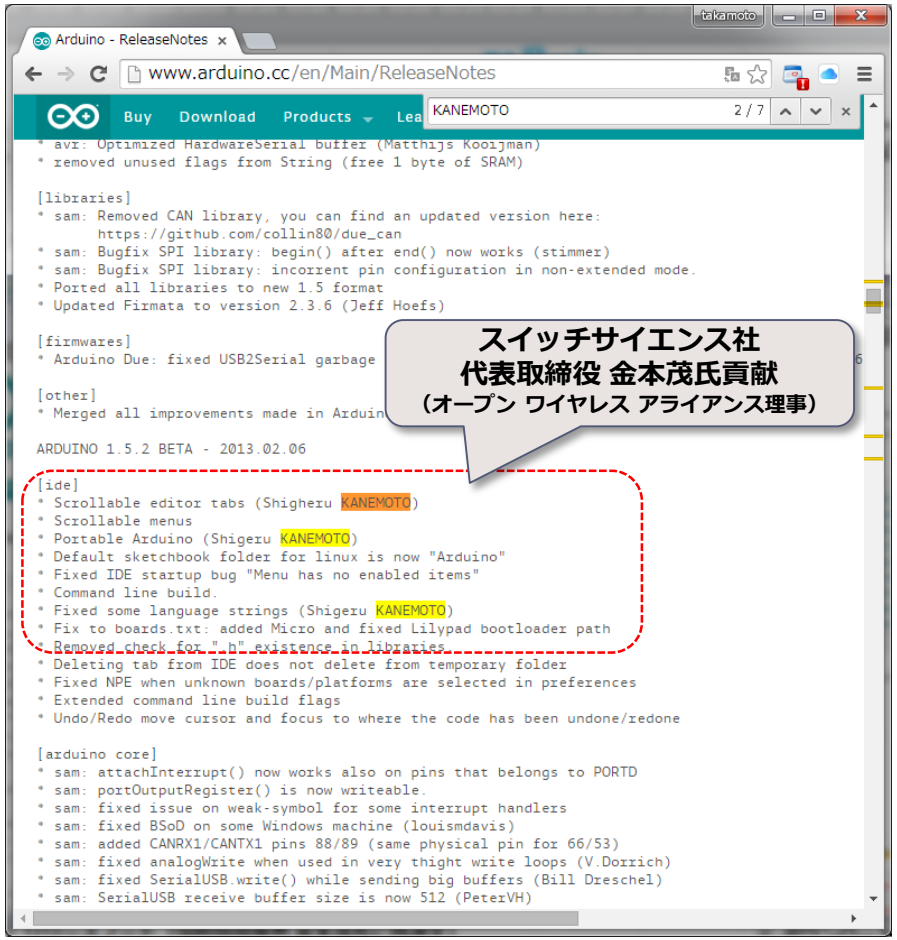
・ **Arduino.cc （Arduinoホームページ）**

・ **Arduino/Download**

Windowsの場合（以下から選択可能）
1)　Windows Installer （PC権限者として c:¥Program Files(x86)¥Arduinoにインストール）
2)　Windows ZIP file for non admin install (PC権限者でなくても解凍インストール可）
3)　Windows app （Windowsアプリとしてインストール）

3．Arduino IDE インストール手順②

- Arduino Ver1.8.7（日本語化対応版）のダウンロード（2018年９月時点最新版）

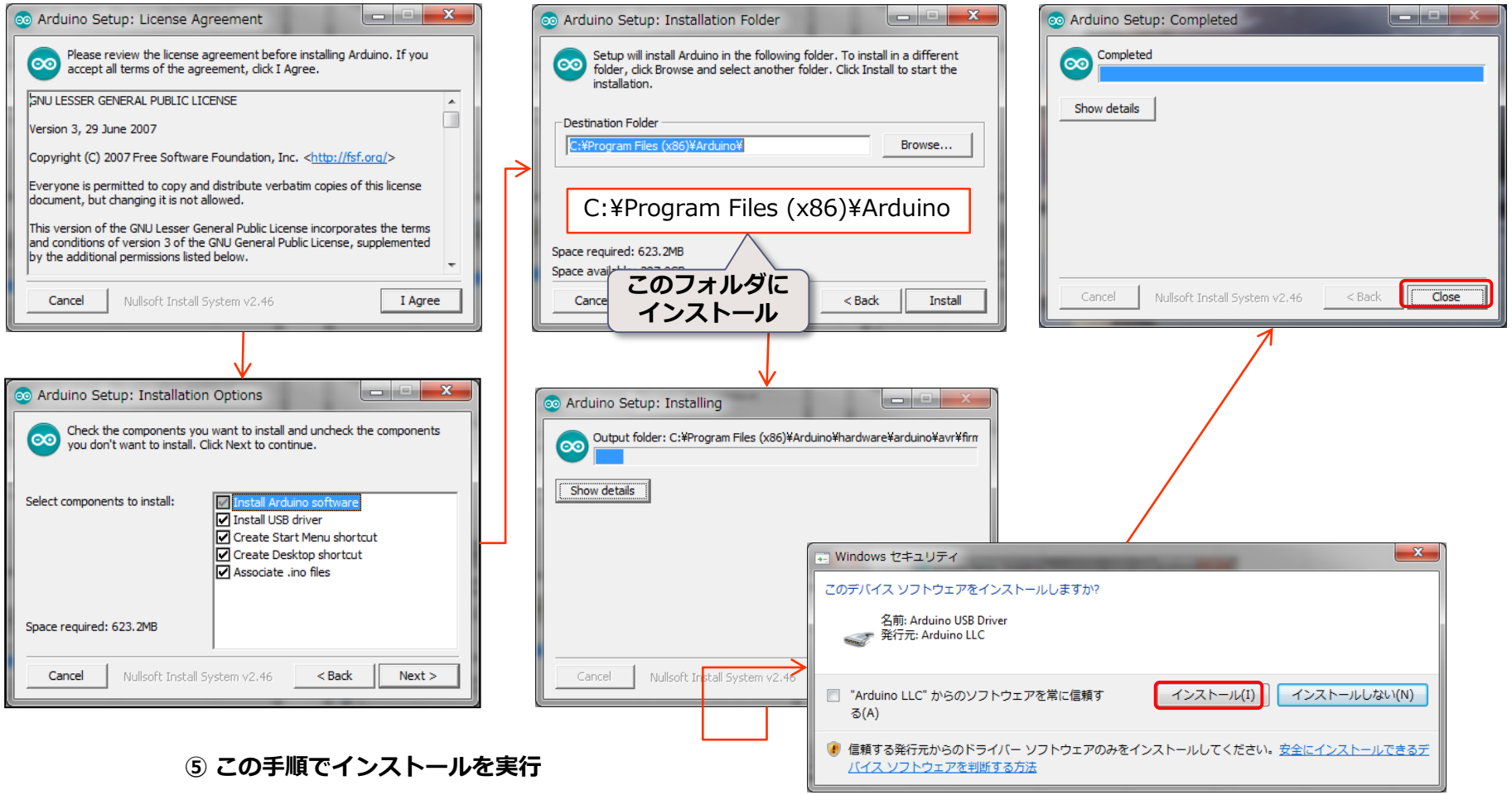


リリースノート（国際版開発者 金本茂氏に感謝文）

④ ダウンロード画面

3．Arduino IDE インストール手順③

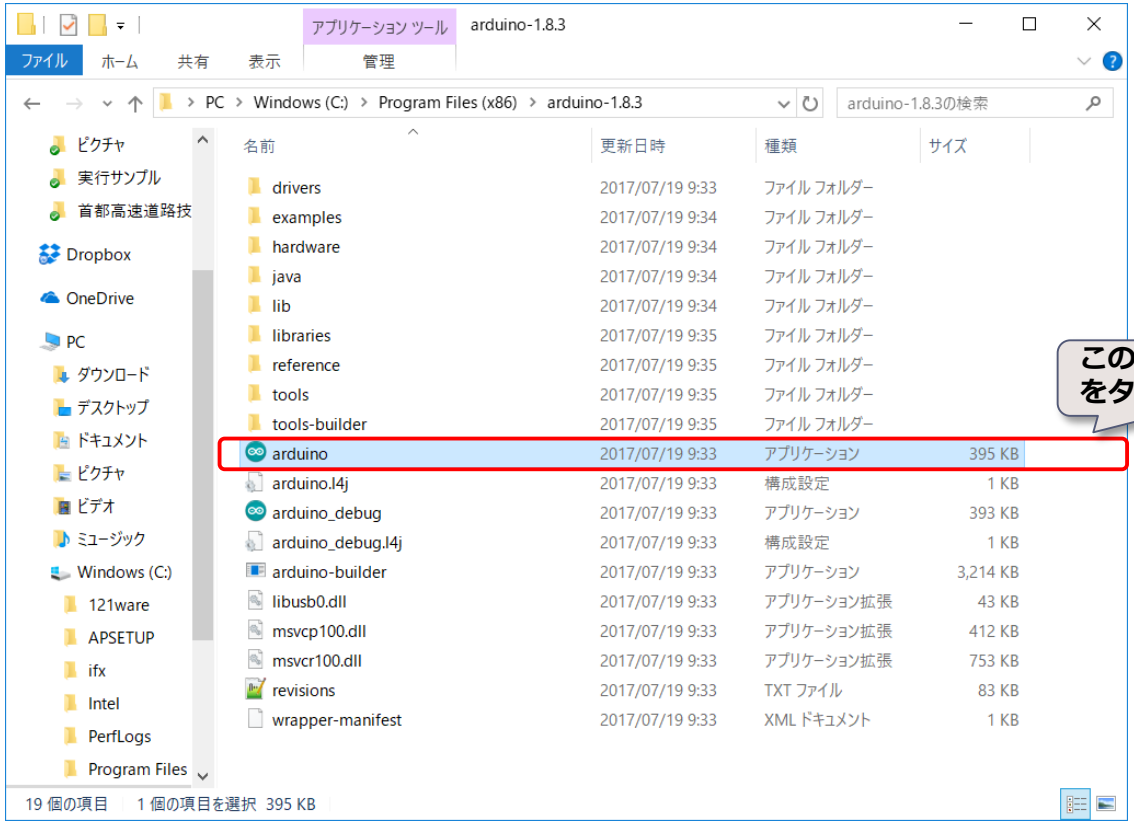
C:¥Program Files (x86)¥Arduinoにダウンロード （Windows installerの場合）



⑤ この手順でインストールを実行

3．Arduino IDE インストール手順④

C:¥Program Files (x86)にダウンロード （Windows zip ファイルの場合）

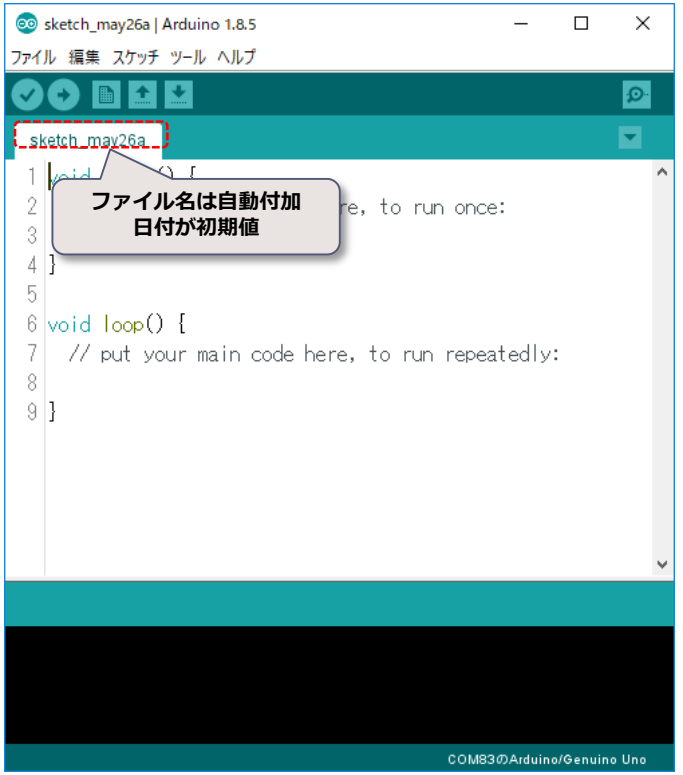


⑥ ダウンロードされたフォルダ
→ タスクバーに表示

4．Arduino IDE 画面説明①

① Arduino アイコンなどを選択

② Arduino 立ち上げ表示画面



② Arduino 初期画面

4．Arduino IDE 画面説明②

ツールバー

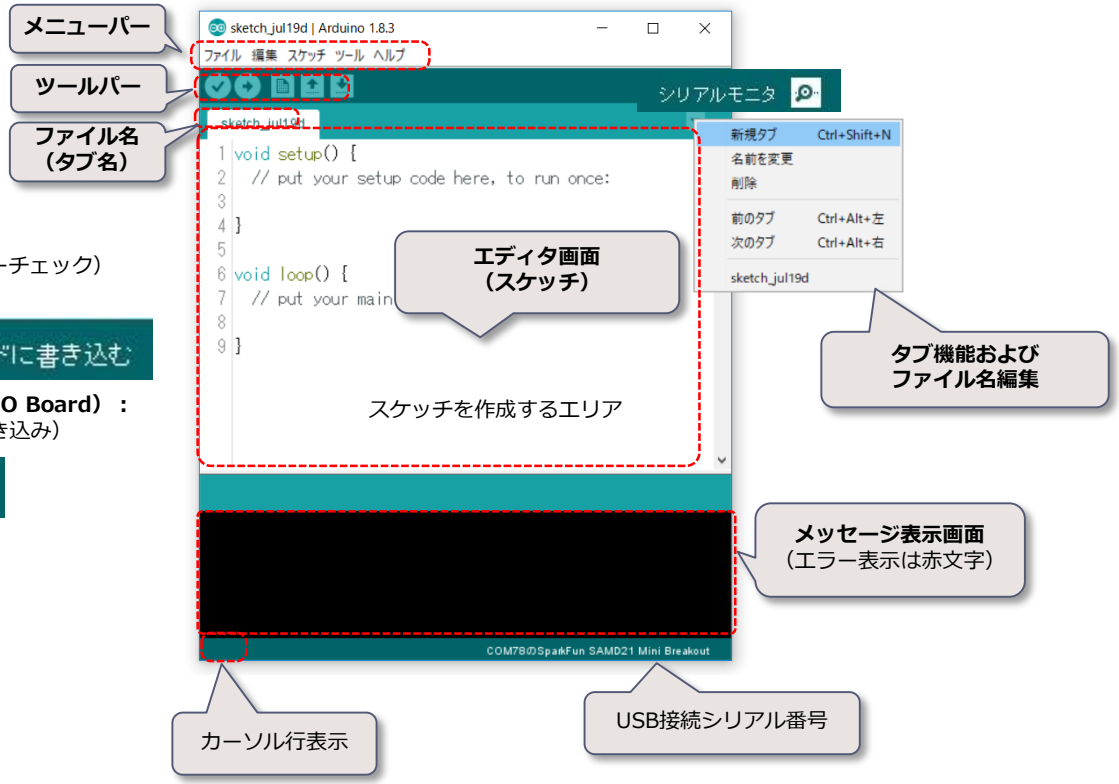
検証（Verify）： コンパイルの検証（エラーチェック）

マイコンボードに書き込む（Upload to I/O Board）：
① コンパイル検証 ＋ アップロード（書き込み）

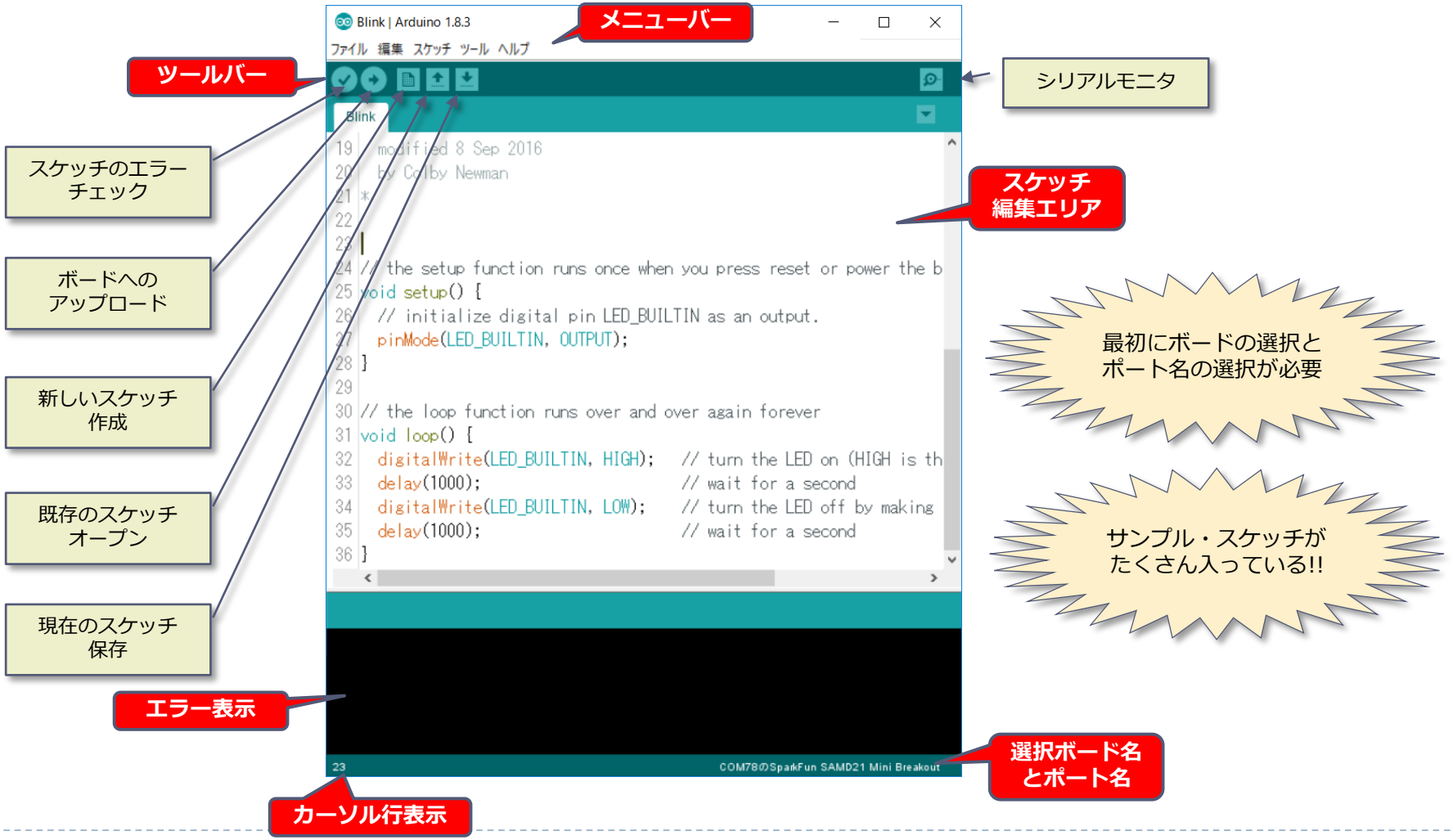
新規ファイル（New）：コンパイルの検証

開く（Open）：既存スケッチを開く

保存（Save）：スケッチを保存



4．Arduino IDE 画面説明③

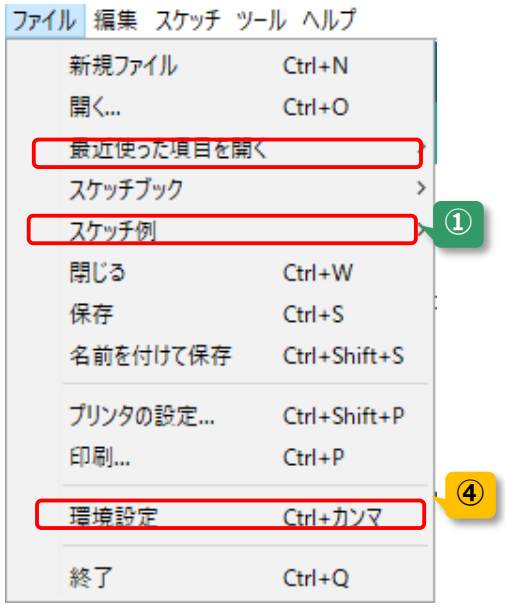


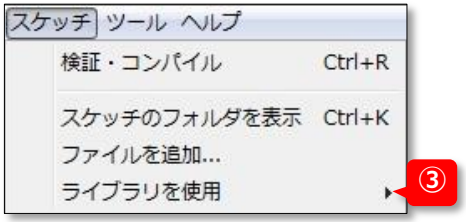
4．Arduino IDE 画面説明④

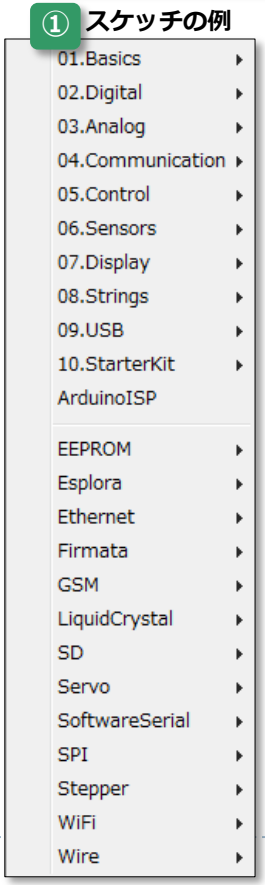
■IDEで使う主な機能だけを紹介しておきます。

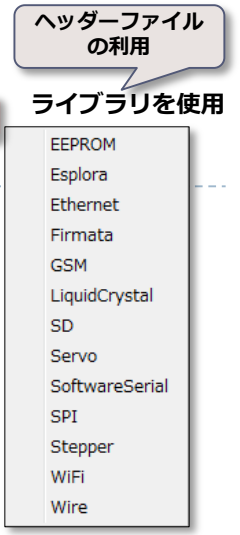
タブ機能は、プログラムのモジュール化で有効に働きます。よく使うスケッチ群をこのタブ画面で分けて利用します。

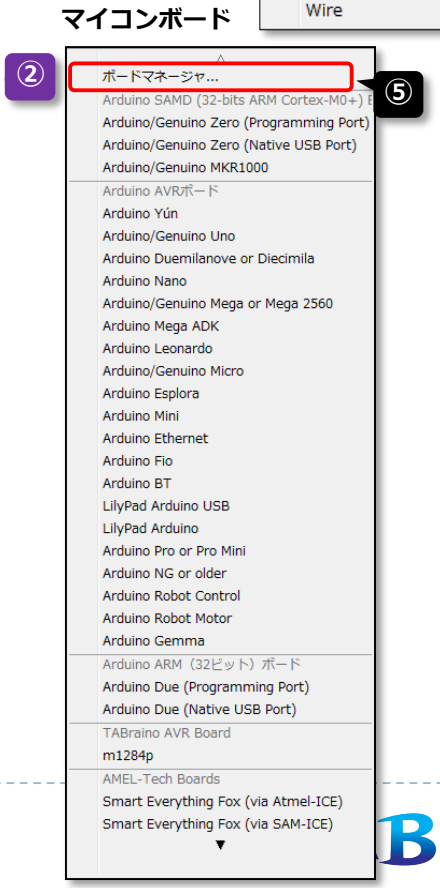
4．Arduino IDE 画面説明⑤











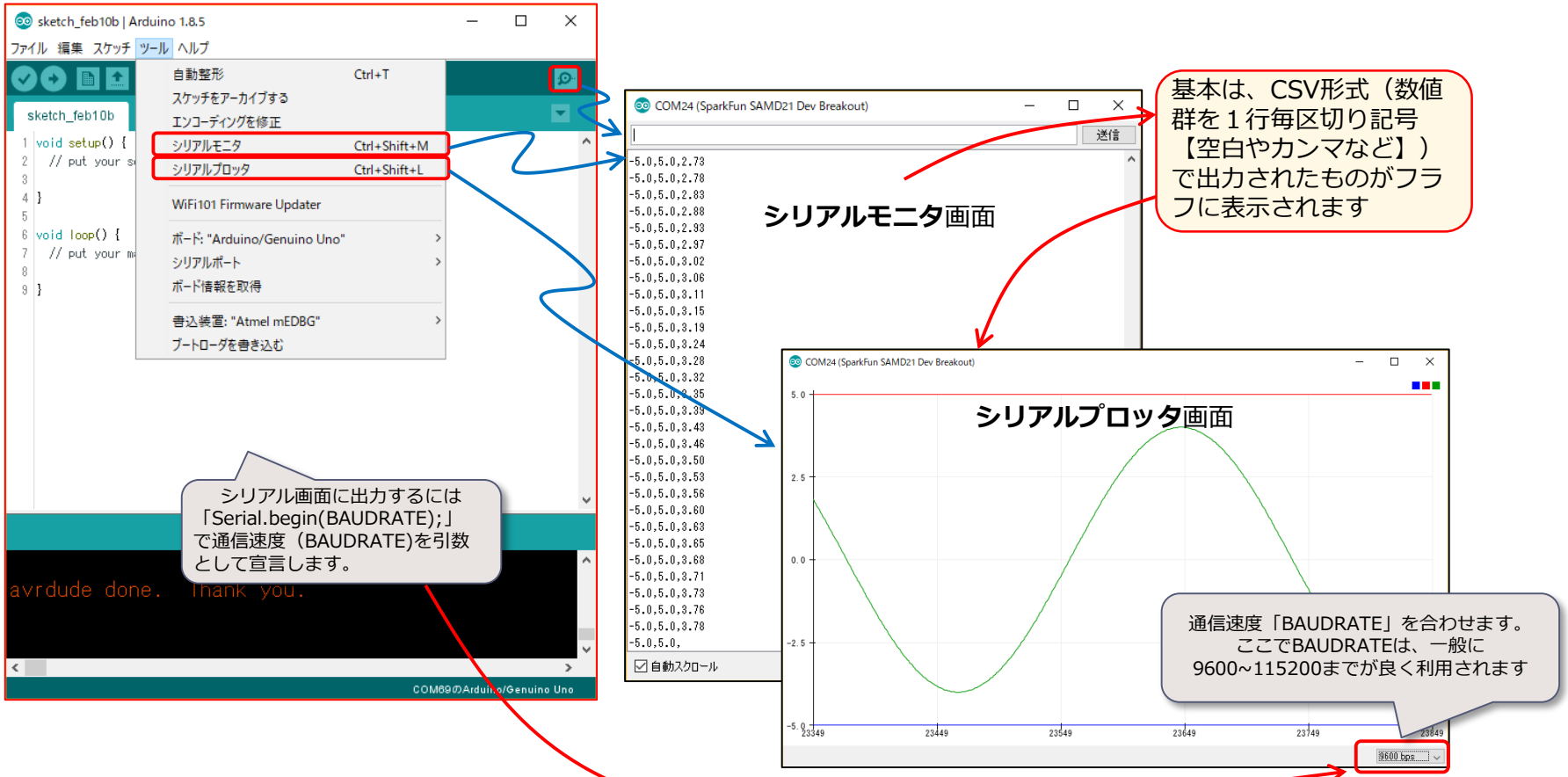
4．Arduino IDE 画面説明⑥

②

結果表示で利用するシリアルモニタ画面とシリアルプロッタ画面：

スケッチ（プログラム）をArduinoに書き込んで実行した場合、USBケーブルを通じてPC側に結果を出力表示させることができます。この場合、Arduino側のプログラムでは、シリアル通信（UART）を使って出力を行います。

表示先は、**シリアルモニタ画面**または数値のみのグラフを表示させる場合には**シリアルプロッタ画面**を選択します。



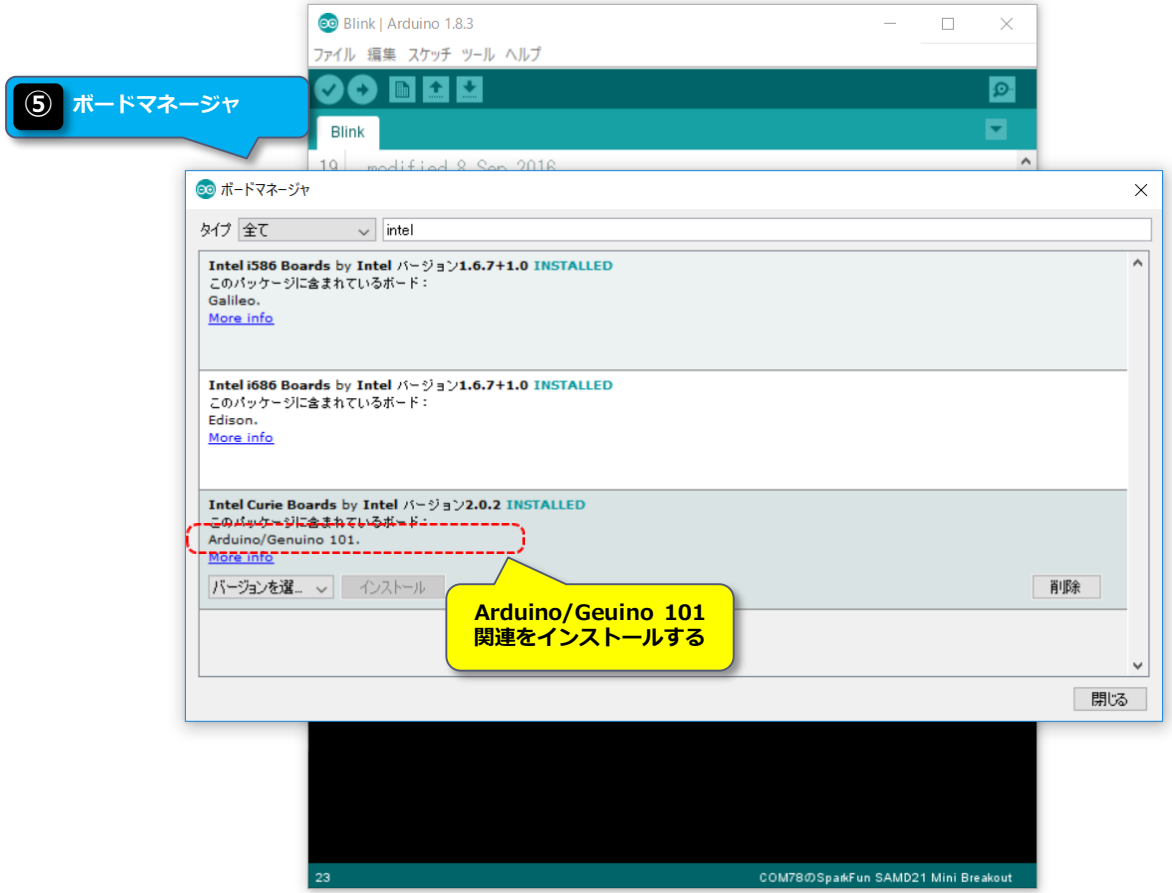
5．Arduino接続①

⑤

Genuino101をご利用の場合には、以下のボードマネージャより関連ファイルを読み込む必要があります。

① Arduino（Genuino101）とPCとをUSBケーブルで接続

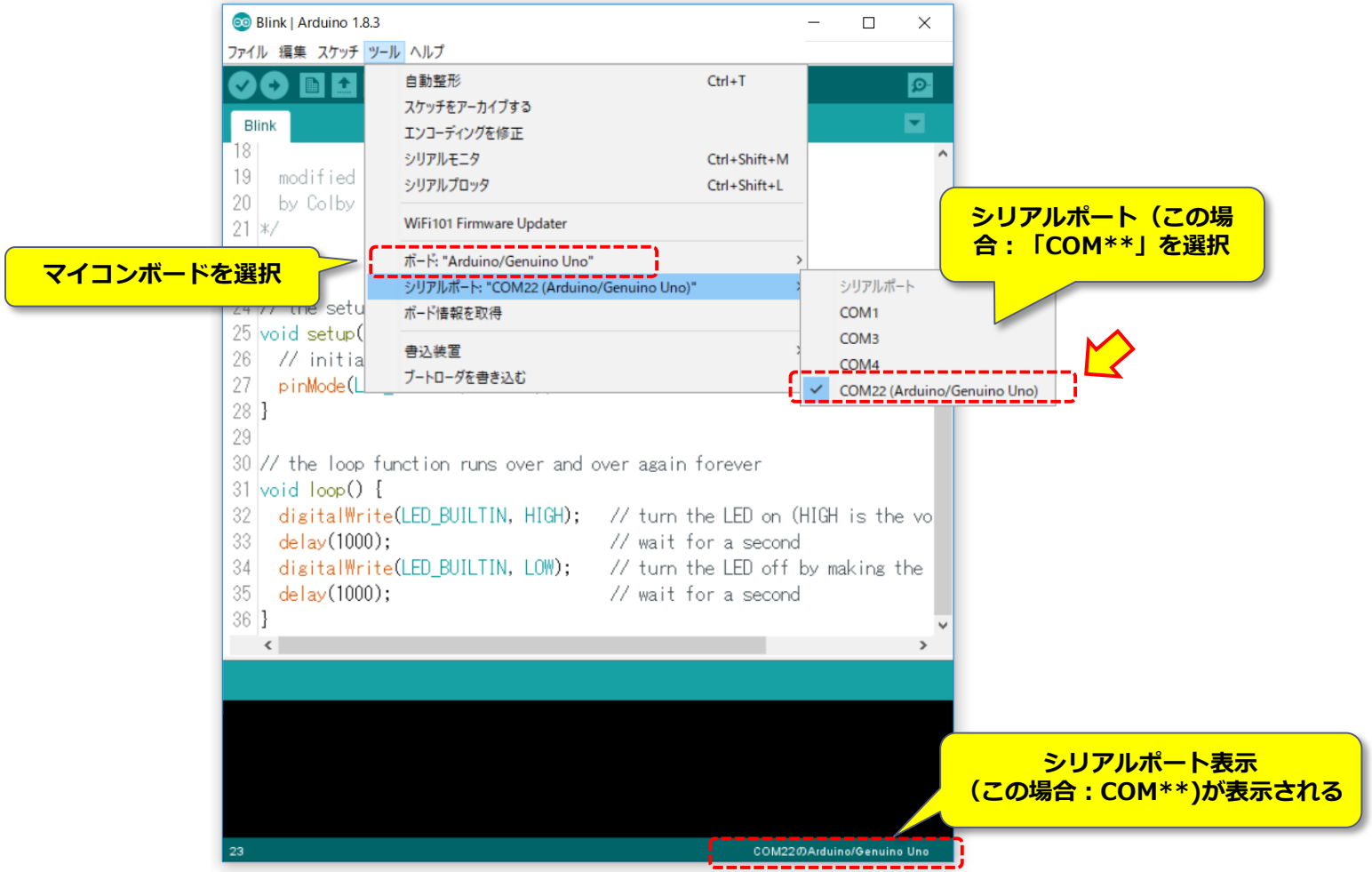
② 「ボードマネージャ」選択



5．Arduino接続②

⑤

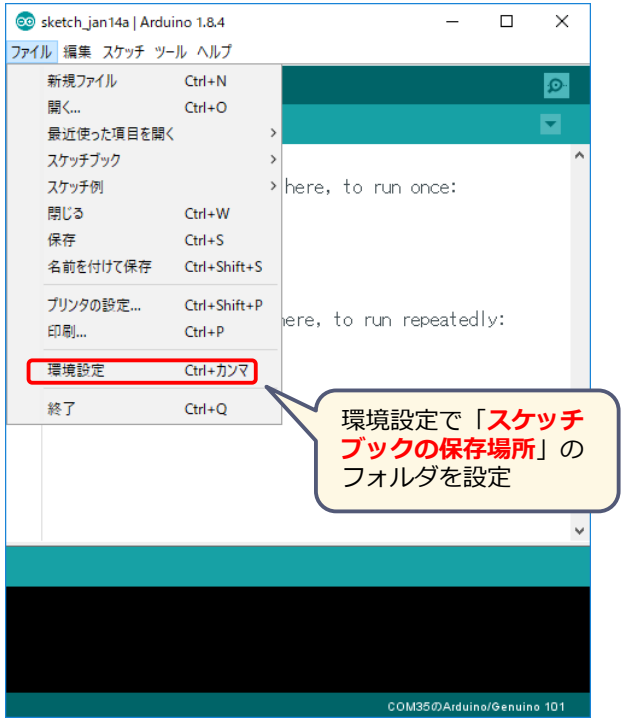
⑤　Arduino IDE側でのボードとシリアルポートの設定

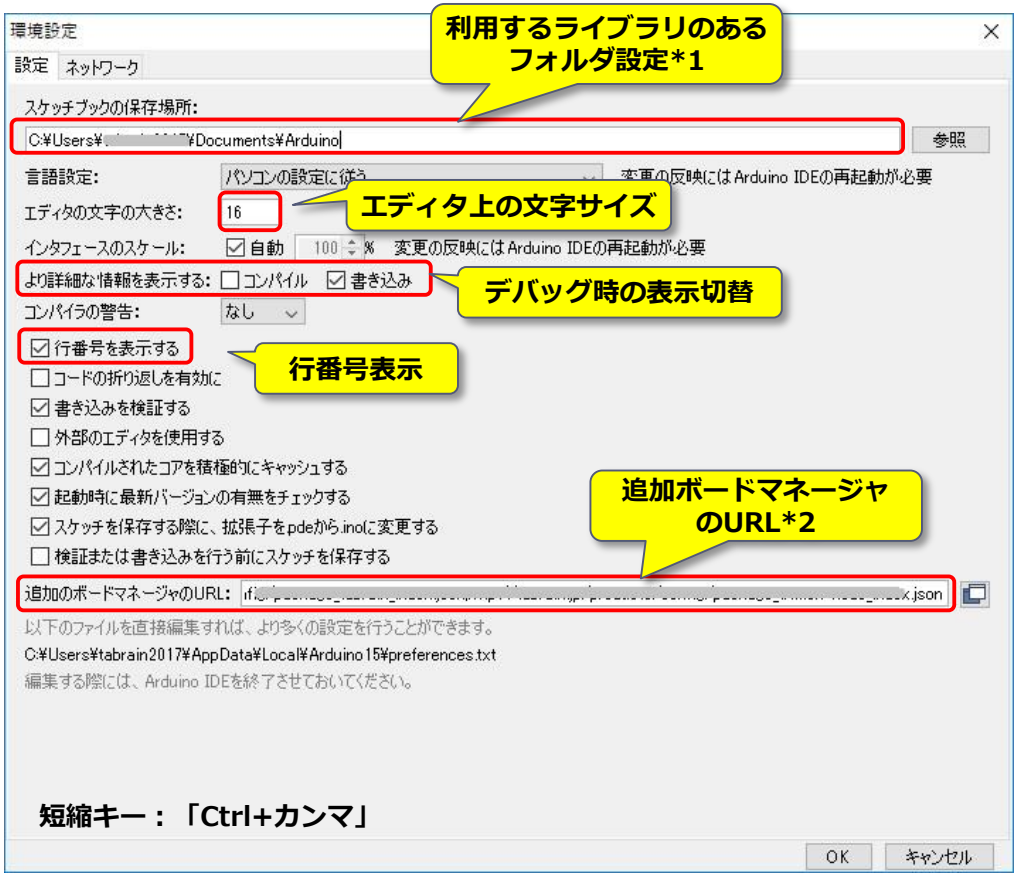


6．環境設定について

④

ここでは、メニューバー「ファイル」にある「環境設定」について、よく利用する項目について紹介しておきます。





【補足1】*1： フォルダの初期設定は、「＊＊￥ユーザ￥￥<アカウント名>￥ドキュメント￥Arduino」で、このフォルダにある「libraries」（ライブラリ群）が利用できるようになります。また独自のプロジェクト（仕事）で、環境を区別利用することができます。

【補足2】*2：追加ボードをJSON書式で定義し、ここで設定することで、オリジナルのボードの設定や利用環境が追加います。（本ボードでもこの機能を使って、前述のボード追加を行っています）

7．ライブラリのインストール

Arduinoのライブラリ群は、以下のところにインストールして利用することができます。

① Arduino IDE上のライブラリ群利用：Arduinoのライブラリマネージャから読み込み ③
② GitHub上のライブラリ群利用（zipファイル）：ZIPファイルでダウンロードしたものをインストールし利用
③ 独自ライブラリ利用：手動でインストールし利用

①の場合には、Arduino IDEのメニューバー「スケッチ」→「ライブラリをインクルード」→「ライブラリを管理...」で呼び出しが可能です。このときIDEを再起動します。

②の場合には、「ZIPファイル」で読み込んだライブラリ群を「スケッチ」→「ライブラリをインクルード」→「.ZIP形式のライブラリをインストール...」でインストールしたのち、IDEを再起動して利用できるようになります。
この場合、「環境設定」の「スケッチブックの保存場所」の「libraries」のフォルダ下に保存されます。

③の場合には、任意のフォルダ先にインストールして利用することができます。

操作	フォルダ	参　考
Arduino IDE インストール先 ライブラリ	C:¥Program Files (x86)¥Arduino¥libraries	自動で行う場合（手動だと「Arduino1.8.5」バージョン名が付く配下にインストールする） 「スケッチ例」での「あらゆるボード用のスケッチ例」で利用可能
Arduino IDE　「環境設定」先ライブラリ	C:¥Users¥<アカウント名>¥Documents¥Arduino¥libraries	「スケッチ例」での「カスタムライブラリのスケッチ例」で利用可能
手動で任意のフォルダ先にインストール	任意フォルダ	「#include “任意のフォルダ¥ヘッダファイル“」で呼び出しで利用可能

【補足１】GitHubは、ソフトウェア開発プロジェクトのため、フリーのライブラリなどをオープンソースとして公開しているサイトです。
【補足２】 ライブラリは、なるべく「環境設定」でのアカウント名以下のドキュメント以下に展開するのを推奨します。

8．呼び出しライブラリについて

ライブラリ群を呼び出す場合、つまり「#include」で呼ばれるライブラリ群は、
以下のフォルダ内の手順（①→②→③→④）で探しにいきます。

① 現在（カレント）のスケッチ保存フォルダ内（タブで宣言）
↓
② 環境設定による「スケッチブックの保存場所」フォルダ内
↓
③ ボード関連の「libraries」フォルダ内（特定カスタムボード限定）
↓
④ ArduinoIDEの「libraries」フォルダ内（あらゆるボード）

#include <***.h>　は、②「libraries」のフォルダから探しだします。
#include “***.h” は、①のカレントフォルダから先に探しに行きます。

①のカレントフォルダを設定しない場合には、②の設定フォルダがカレントとなります。

②の「スケッチブックの保存場所」を設定すると、プロジェクト毎などの整理が分かり易くなります。
　→できるだけ「環境設定」における「**スケッチブックの保存場所**」を利用して、開発を進めていくのをお薦めいたします。

③ 利用するボード毎で用意される「libraries」に専用のライブラリ群が保管されています

④ の中にあるライブラリおよびサンプルスケッチは、あらゆるボードで利用できます。ただし、書き込みはできませんので注意してください。

9．ライブラリ利用方法

ArduinoにI2C関連センサーなどの部品を追加して利用する場合や、スリープ機能などを追加したりする場合、既存のライブラリを利用することでプログラミングがとても簡単で短時間に構築できるようになります。

これらのライブラリは、製品毎や機能毎にさまざまなものが用意されていて、主にインターネット上から検索して利用するようになっています。

後述するように、加速度センサ、温湿度センサ、それに照度センサをはじめ、スリープ・割込み・ウェイアップ機能を使う場合、プログラム（スケッチ）上に、「#include」文で、ライブラリを読み込んでいます。

例えば、温湿度センサを使う場合

```
#include <Wire.h>
#include <HTS221.h>
```

上記の２つのライブラリ「Wire.h」と「HTS221.h」を、Arduinoのライブラリ群ファルダから探してきて、コンパイル・リンクします。

（このArduinoのライブラリ群フォルダは、２つの場所【IDEが置かれた「¥libraries」】【環境設定で設定されたスケッチブックが保存された「¥libraries」】から）

また<>でなく、以下の“”で囲んで使う場合は、

```
#include ”Wire.h“
#include ”HTS221.h”
```

この場合には、上記の２つの場所のほか、先にカレントのスケッチがある場所からも探し出して、読み込み、コンパイル・リンクします。

ライブラリ群の中身は、高度な技術者でないと紐解けないところも多く、詳細はネットで検索ください。

特に使う側としては、ライブラリ群を利用する上での関数群がどんなもので、それをどう利用するかが重要となります。

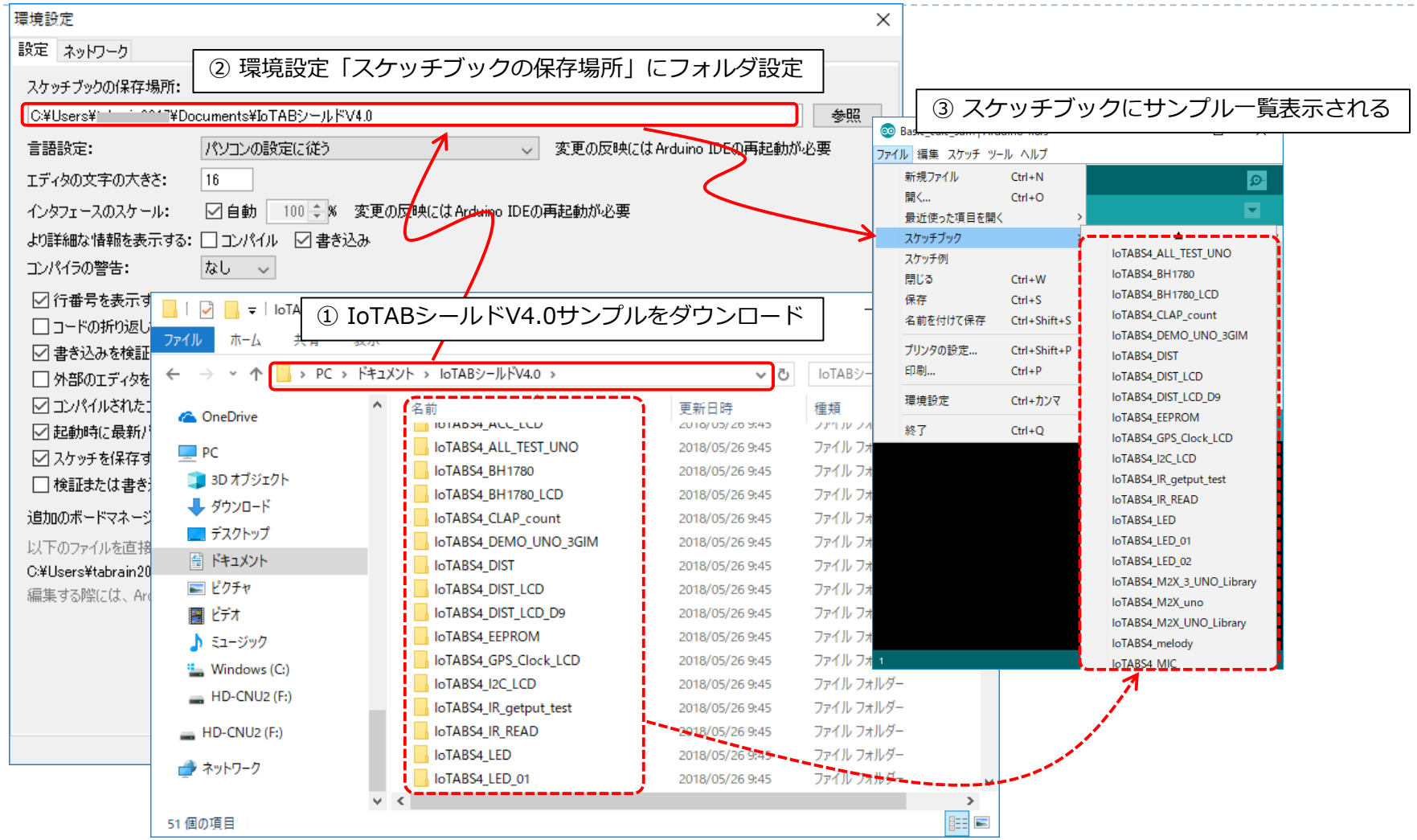
（つまり、ライブラリ群の中身は、ブラックボックスとして利用することで構わないといえます）

使う上でのヒントとしては、ライブラリ群はサンプルスケッチが用意されているものが多いので、その中身を参考にしながら、プログラミング（スケッチ）していく方法をおすすめします。

その他、インターネット上にライブラリ群を使った事例も豊富に検索できるものもありますので、参考になるところです。

ここでのボードを利用する場合のライブラリ群も後の章にて、その呼び出し方や使い方を紹介していますので、詳細についてはそちらの章をご参照ください。

１０．環境設定によるサンプルの呼び出し方法



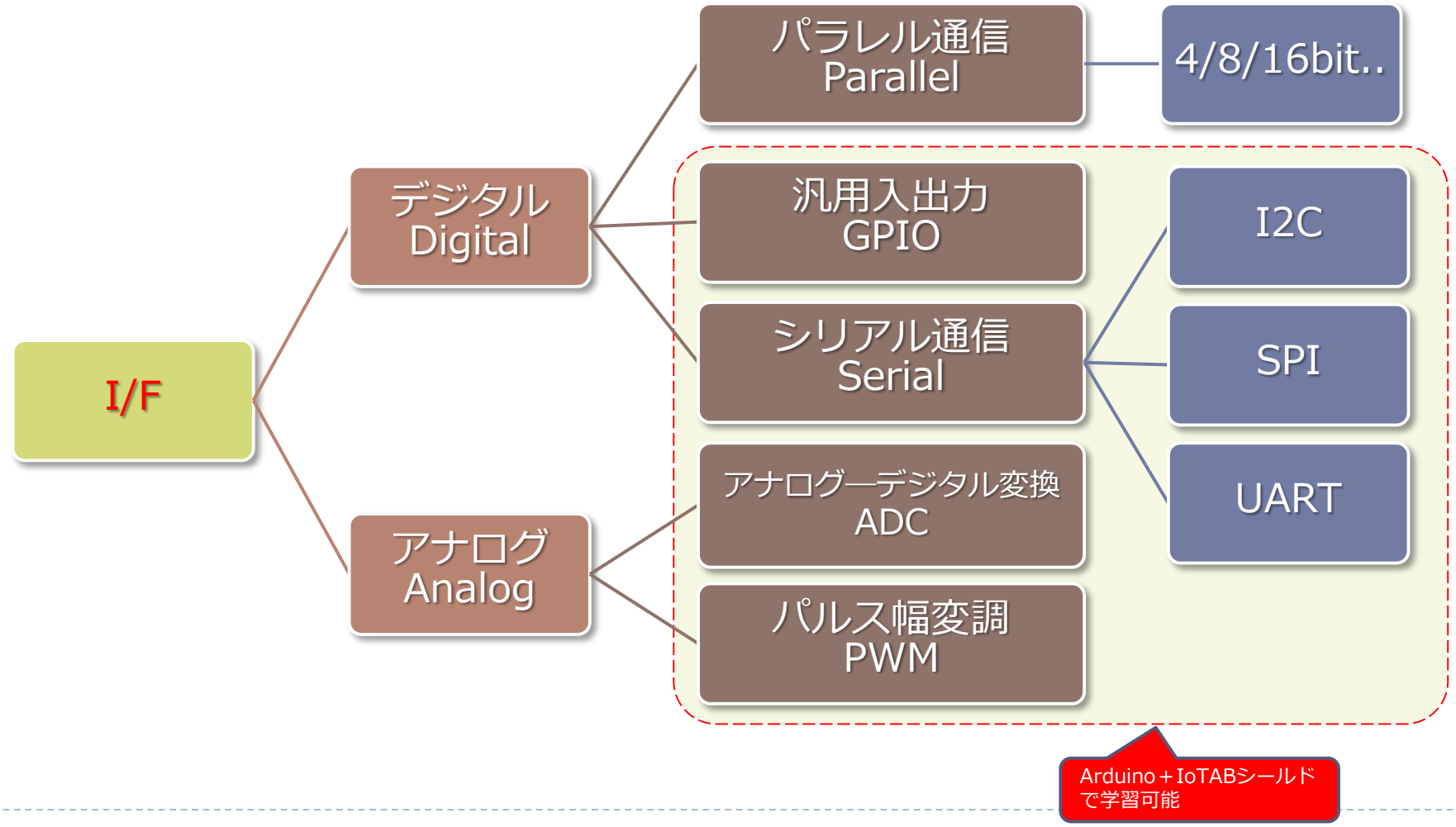
注意：ここでの「IoTABシールドV4.0」フォルダは独自に作成ください。
サンプルスケッチ群は、本ドキュメントの表紙にあるサイトからダウンロードしてください。

第３章
Arduinoの基本

マイコンのインタフェース
Arduinoのインタフェース
サンプルスケッチの実行

1．マイコンのインタフェース

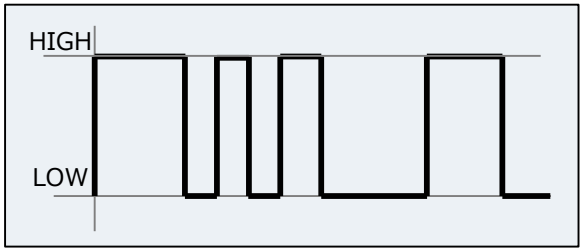
Arduinoの世界では、以下のマイコンのインタフェースのうち、パラレル通信以外で学習することができます。



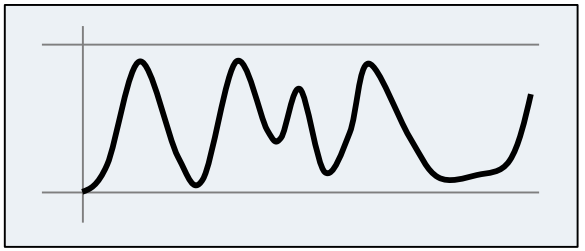
2．インタフェースの基礎 ①

- アナログ通信とデジタル通信
  - **デジタル通信（GPIO：General Purpose Input/Output）**
    - **“0”【LOW/False】** または **”1”【HIGH/True】** の データ表現
    - シリアルであれば、アナログと同様に1本のラインで伝送可
    - 雑音（ノイズ）に強い
  - **アナログ通信（ADC ・ PWM ）**
    - 電圧で、データを段階的な値で表現
    - 例えばArduinoでは、0V～5V(または3.3V)の範囲で、**入力（ADC）**は：2の10乗(1024段階)*1で、出力（**PWM**）は：２の８乗（256段階）でデータを表現
    - 雑音（ノイズ）に弱い
    - 直観的で分かり易い

＊１：32ビットマイコン（Due、Zero、MKRファミリー）では、階調が０～4095まで利用可能
（analogReqadResolution関数利用による）



デジタル


アナログ

2．インタフェースの基礎 ②

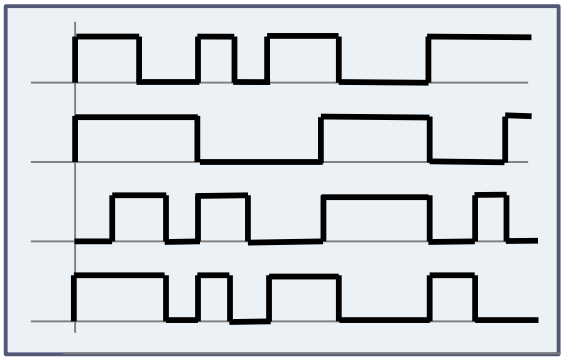
シリアル通信／パラレル通信

シリアル通信（Serial）

- 基本は、1本のラインでデータを伝送
- 送受信を同時に行う場合（全二重通信）は、送信用と受信用の2本が必要
- PCの例では、USB、SATA/SAS、Thunderbolt 等
- 伝送周波数を高くできる

パラレル通信（Parallel） ← Arduinoでは未対応

- 複数のラインで複数のデータを平行して伝送
- 通常は、4/8/16/32bitのいずれか
- シリアルに比べて1度に送れるデータ量は多いが、ビット数が多くなると配線が難しく、また速度を上げにくい
- PCの例では、RAM、PCI/PCI-Exp 等

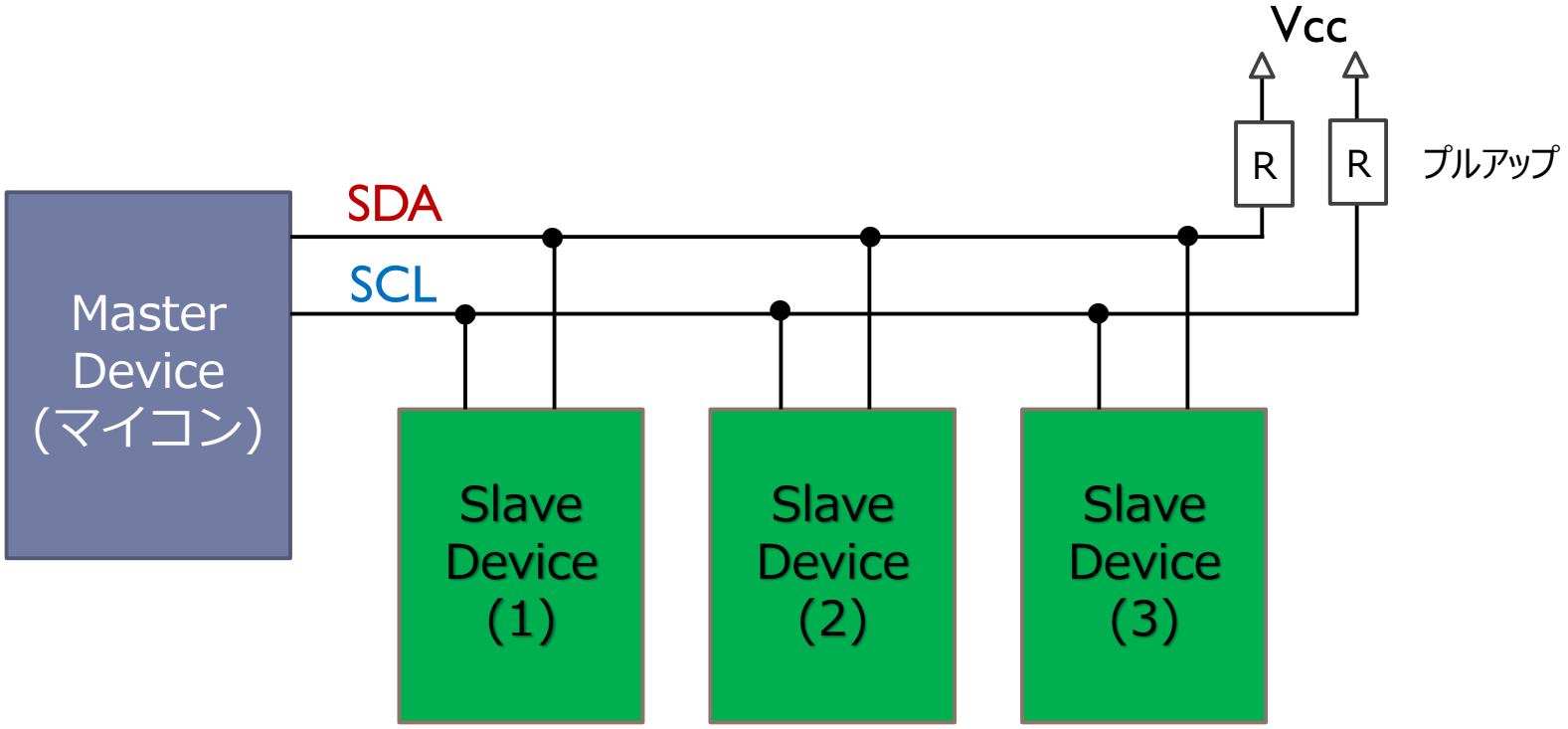


2．インタフェースの基礎 ③

「アイ・スクエア・シー」とか
「アイ・ツー・シー」と呼ぶ

- **I2C**（Inter-Integrated Circuit）とは
  - シリアルインタフェース（Arduinoで使える通信速度は最大400kbps）
  - マスタとスレーブで通信を行うプロトコル
  - 複数のデバイスをラインにぶら下げることができる（最大112個）
  - マスタ主導でコマンドをデバイスに送り、スレーブからの応答を受け取る

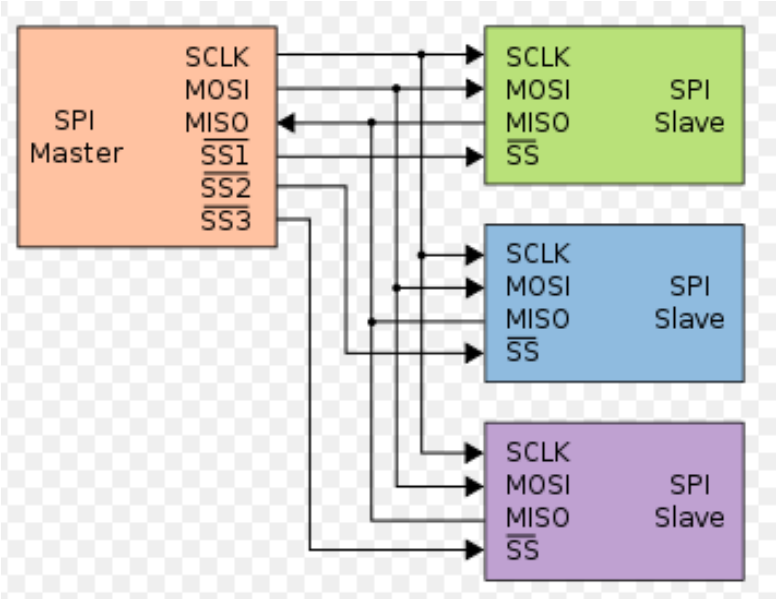
Arduino UNOの場合
A4-A5利用
（他専用ピン利用可）
IoTABシールドでは
A4-A5は未使用



2．インタフェースの基礎 ④

- **SPI**（Serial Peripheral Interface）とは
  - シリアルインタフェース
  - マスタとスレーブで通信を行うプロトコル、仕組みはI2Cとほぼ同じ
    （ただし、送受信は別ライン）
  - SDカードの読み書き等で利用（I2Cより高速）

Arduino UNOの場合
D10-D12利用
（他専用ピン利用可）

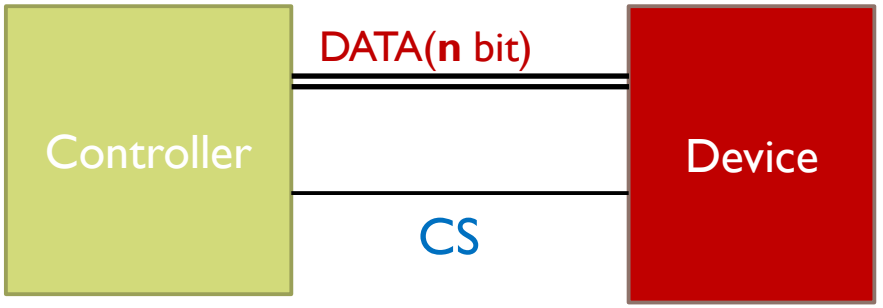


SCLK：　Serial Clock
MOSI：　Master-Out/Slave-In
MISO:　Master-In/Slave-Out
SS：　Slave Select

2．インタフェースの基礎 ⑤

- **Parallel（パラレル）通信**とは
  - いくつかの規格が存在
    - SCSI、PCI/PCI-Express
  - マイコンで利用することは少ない
    - 液晶ディスプレイ(8/16 bit)
    - カメラモジュール(8 bit)
  - 制御の方法
    - CS信号をLOWにしてからDATAを送り、CS信号をHIGH(確定)にする

Arduinoの場合
未対応



2．インタフェースの基礎 ⑥

- **GPIO**（General Purpose Input/Output：汎用入出力）とは
  - 汎用のデジタル入出力
  - 入力は、HIGH（ロジック電圧）とLOW（０V）
  - 出力も、HIGH（ロジック電圧）とLOW（０V）
  - Arduinoの場合、GPIOは D0－D13の他に、A0-A5などのアナログ入力ピンも利用可能
  - ピンの設定宣言は、pinMode関数にて行う

ArduinoUNOの場合
※ GPIOは、
D0-D13とA0-A5

2．インタフェースの基礎 ⑦

ADC（Analog Digital Converter）

- アナログ入力値をデジタルに変換する機能
- アナログデバイスをマイコンで扱うための手段
- 例えば、Arduino UNOでは、入力値（0～5V）を10ビットのデジタル値（0～1023）に変換する
- 入力電圧と出力値の関係は下表の通り：

Arduino UNOの場合
A0-A5利用

入力(mV)	出力(10進数)
0	0
4.89	1
9.78	2
…	…
5000.0	1023

1 段階（LSB）＝
4.89 mV

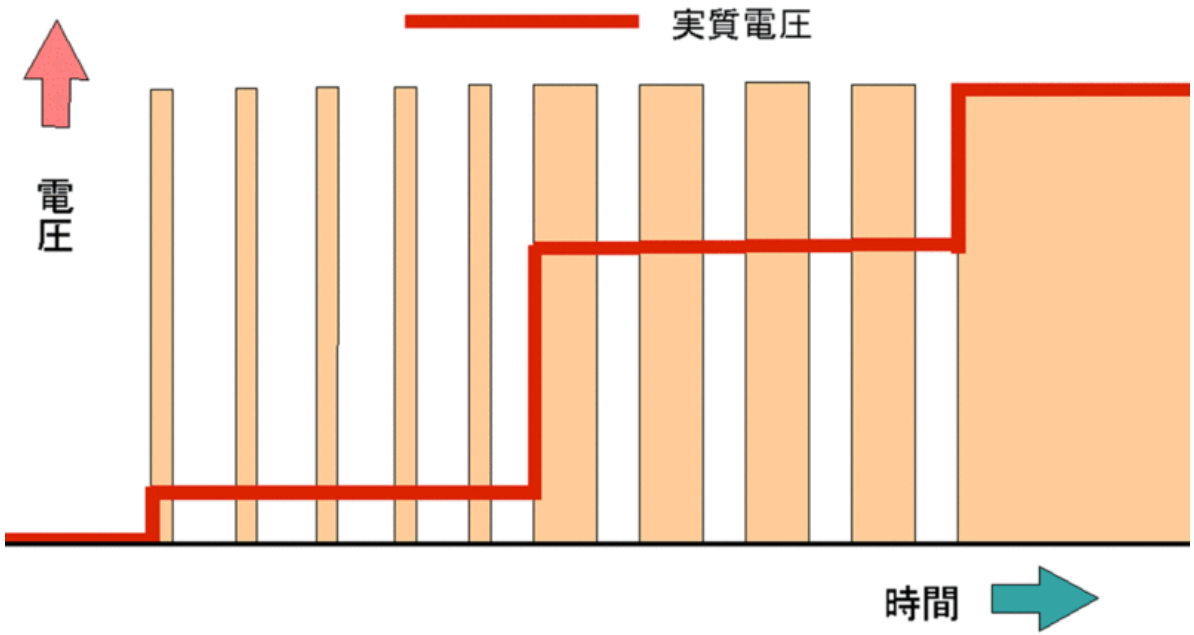
2．インタフェースの基礎 ⑧

- **PWM（Pulse Width Modulation）**とは
  - 電圧の高低を制御する方式
  - パルス波のデューティー比（オン／オフ時間の比率）を変化させて変調する
  - Arduinoでは、**256段階**で制御が可能（D3*など）

ArduinoUNOの
※ アナログ出力ピンは
D3,5,6,9,10,11

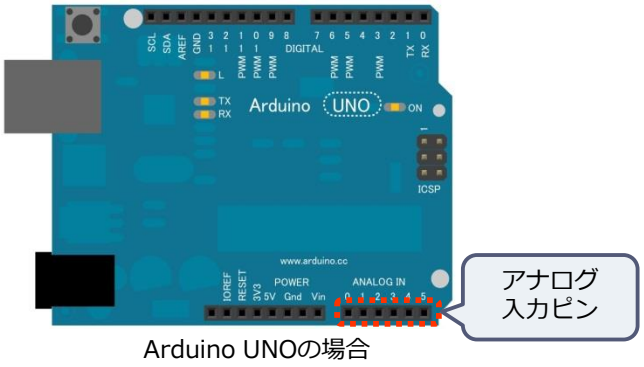
【注意】
Geuino101のPWMは、
D3,5,6,9

PWMは、アナログ出力のことを指す



3．Arduino のインタフェース ①

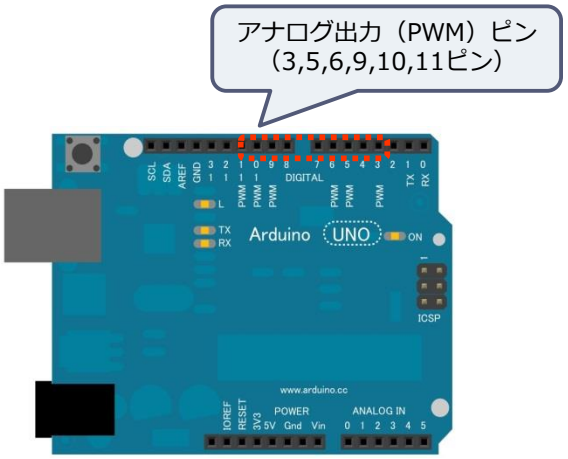
- **アナログ入力（ADC）**
  - Uno では、6ピンまで利用可能(A0-A5)
    - int **analogRead**(int pin)
      - pin　　**0～5**
      - 戻り値　　**0～1023**



Arduino UNOの場合

- **アナログ出力（PWM）**
  - Uno では、デジタルピンと共用で、6ピンまで利用可能
    - void **analogWrite**(int pin, int value)
      - pin　　**3,5,6,9,10,11**
      - value　　**0～255**（0は0V、255は 5V）
    - アナログ値（PWM）を出力
      - 利用は、LEDの明るさ制御、モータ回転スピード制御など

      ※ PWM信号の周波数は、ピン5と6は980Hz、それ以外は490Hz



Arduino UNOの場合

3．Arduino のインタフェース ②

- **デジタル入力（GPIO）**
  - Uno では、13ピンまで利用可能(D0-D13)、さらにA0-A5の20ピン利用可能
  - 指定したピンから、0(LOW)または1(HIGH)を読込む
    - void **pinMode**(int pin, int mode)
      - pin　0～13（およびA0～A5）
      - mode　**INPUT** または**INPUT_PULLUP**　（デフォルトではINPUT）
    - int **digitalRead**(pin)
      - pin　0～13（およびA0～A5）
      - 戻り値　**LOW**(0) or **HIGH**(1)

「pinMode(pin,INPUT);」省略可

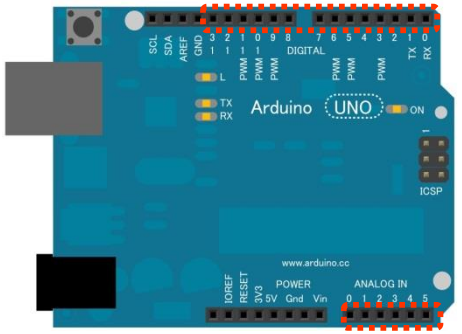
プルアップ抵抗を考慮（モード）

宣言：3番ポートを利用

```
const int sensorPin = 3;
// set up
pinMode(sensorPin, INPUT);
// Read digital value
int sw = digitalRead(sensorPin);
```

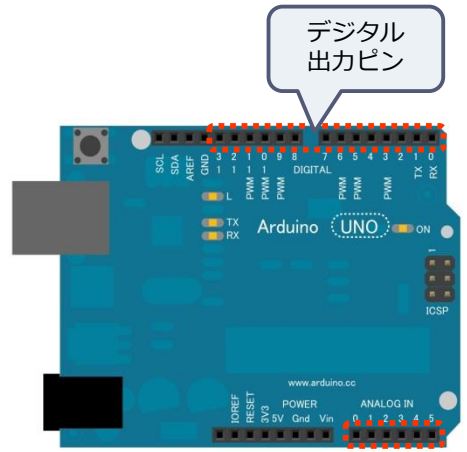
省略可

デジタル入力として設定



3．Arduino のインタフェース ③

- **デジタル出力（GPIO）**
  - 指定したピンへ0(LOW)または1(HIGH) を出力
    - void **pinMode**(int pin, int mode)
      - pin　　0～13およびA0～A5
      - mode　　**OUTPUT**
    - void **digitalWrite**(int pin, int value)
      - pin　　0～13およびA0～A5
      - value　　**LOW**(0) or **HIGH**(1)



Arduino UNOの場合

宣言：3番ポートを利用

**電源とGND
で利用可能**

```
const int sensorPin = 3;
// set up
pinMode(sensorPin, OUTPUT);
// Write digital value
digitalWrite(sensorPin, HIGHT);
```

3番ポートを
出力として設定

3．Arduino のインタフェース ④

UART（Serial）：ハードウェアシリアル通信

- D0（Rx）/ D1（Tx）ピン利用
  - **Serial.begin**（baudrate);
  - **Serial.readStringUntil**(‘¥n’);
  - **Serial.println**(string);

    など

他にソフトウェアシリアル通信も可能

```
#include <SoftwareSerial.h>
SoftwareSerial  SerialS(3,4); //例
```

I2C（Wire.h)

I2Cコネクタ2箇所（A4/A5も利用可）　＜その他：ソフトウェアI2C＞

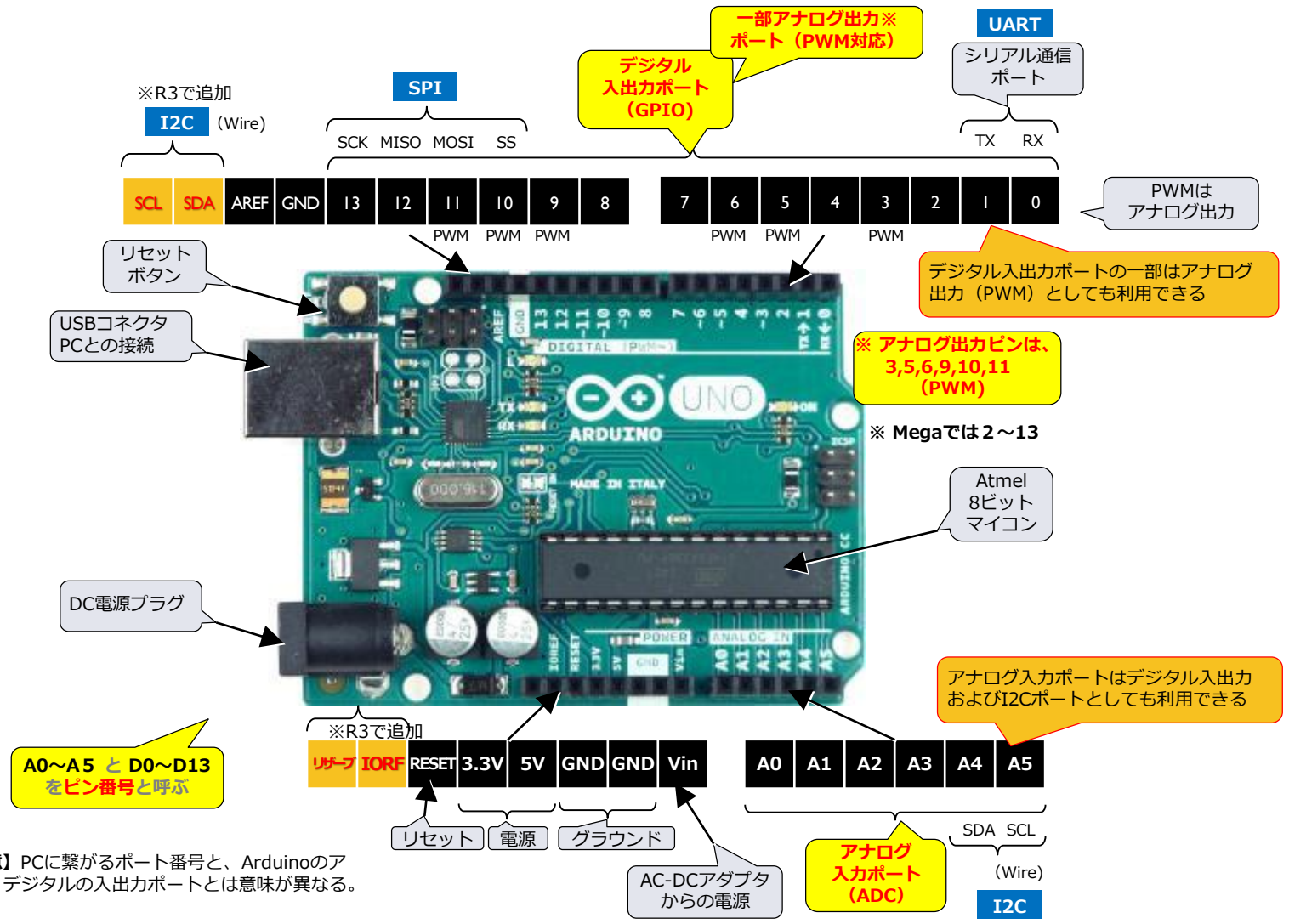
＃include <Wire.h>　を利用

SPI（SPI.h）

- SPIコネクタ利用（D10～D13）

```
#include <SPI.h>
#include <SPI_registers.h>
```

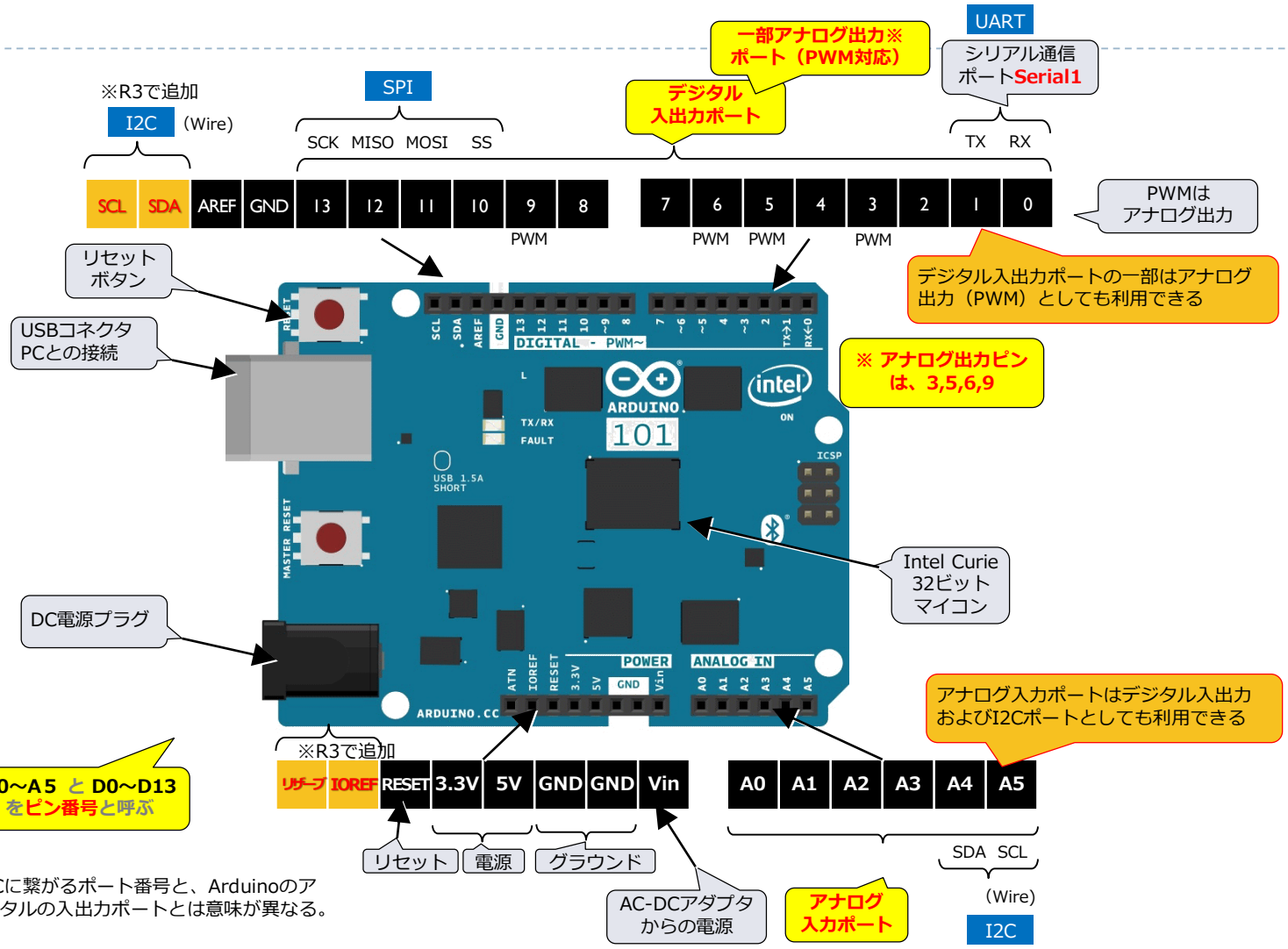
3．Arduinoのインタフェース ⑤ Arduino UNO R3



※【注意】PCに繋がるポート番号と、Arduinoのアナログ・デジタルの入出力ポートとは意味が異なる。

重要

3. Genuino101 インタフェース⑥ Genuino101



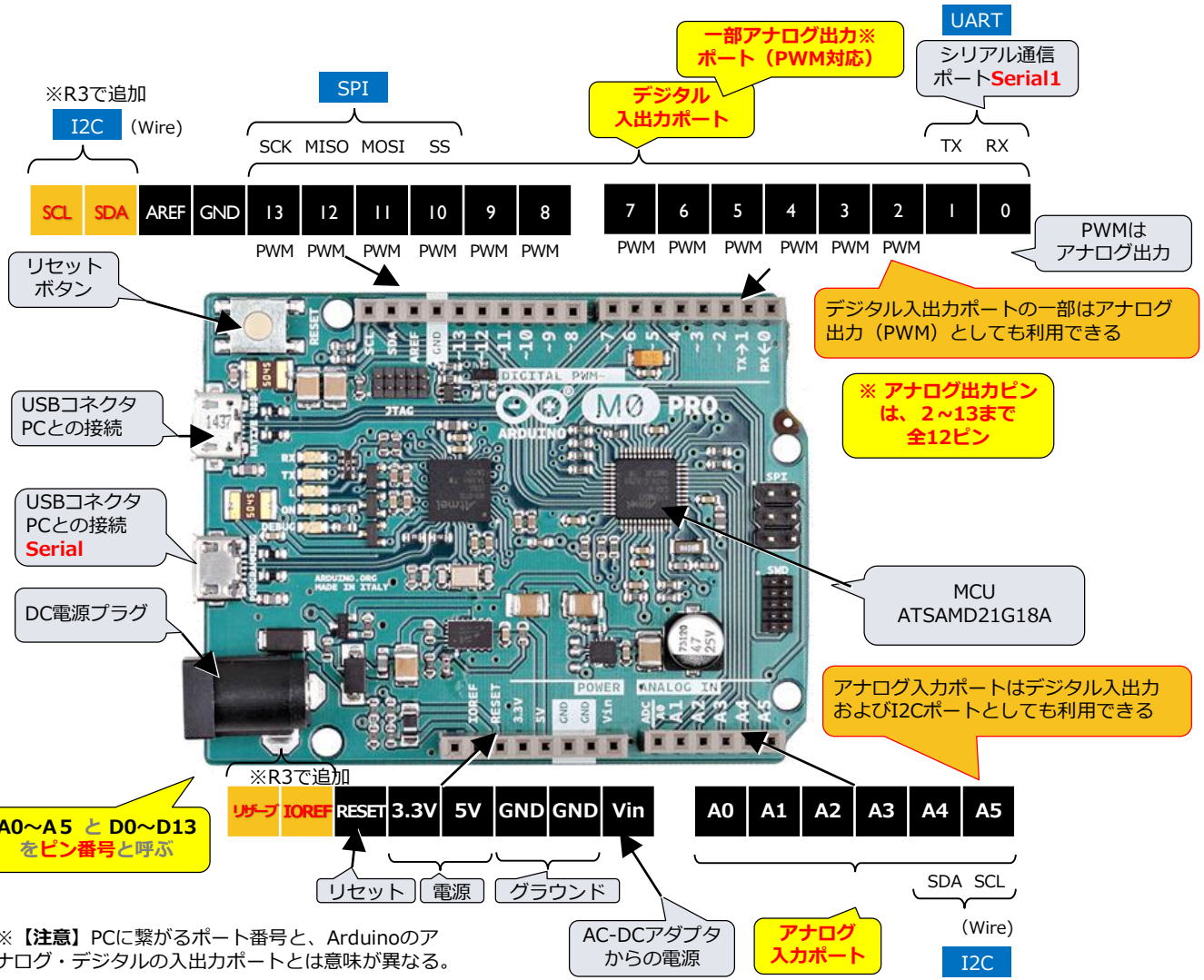
※【注意】PCに繋がるポート番号と、Arduinoのアナログ・デジタルの入出力ポートとは意味が異なる。

重要

3．Arduino M0 Pro インタフェース ⑦

■Arduino M0 Pro技術仕様

マイクロコントローラ	ATSAMD21G8A
制御電圧	3.3V
入力電圧（推奨値）	7-12V
入力電圧（制限値）	5-15V
GDIO ピン数	14（D）＋6（A） （うち12ピンPWM）
PWMピン数	12
アナログ入力ピン	6
I/Oピンの電流	7 mA
フラッシュメモリ	256 kB
SRAM	32 kB
クロック速度	48MHz
LEDピン	D13
機能	EEPROM（無し、エミュレーション16Kb）



※【注意】PCに繋がるポート番号と、Arduinoのアナログ・デジタルの入出力ポートとは意味が異なる。

3．Arduinoのインタフェース ⑧

電源とグラウンド（GND）のテクニック

GPIOを使って、電源（ロジック電圧）とGND（０V）に設定可能

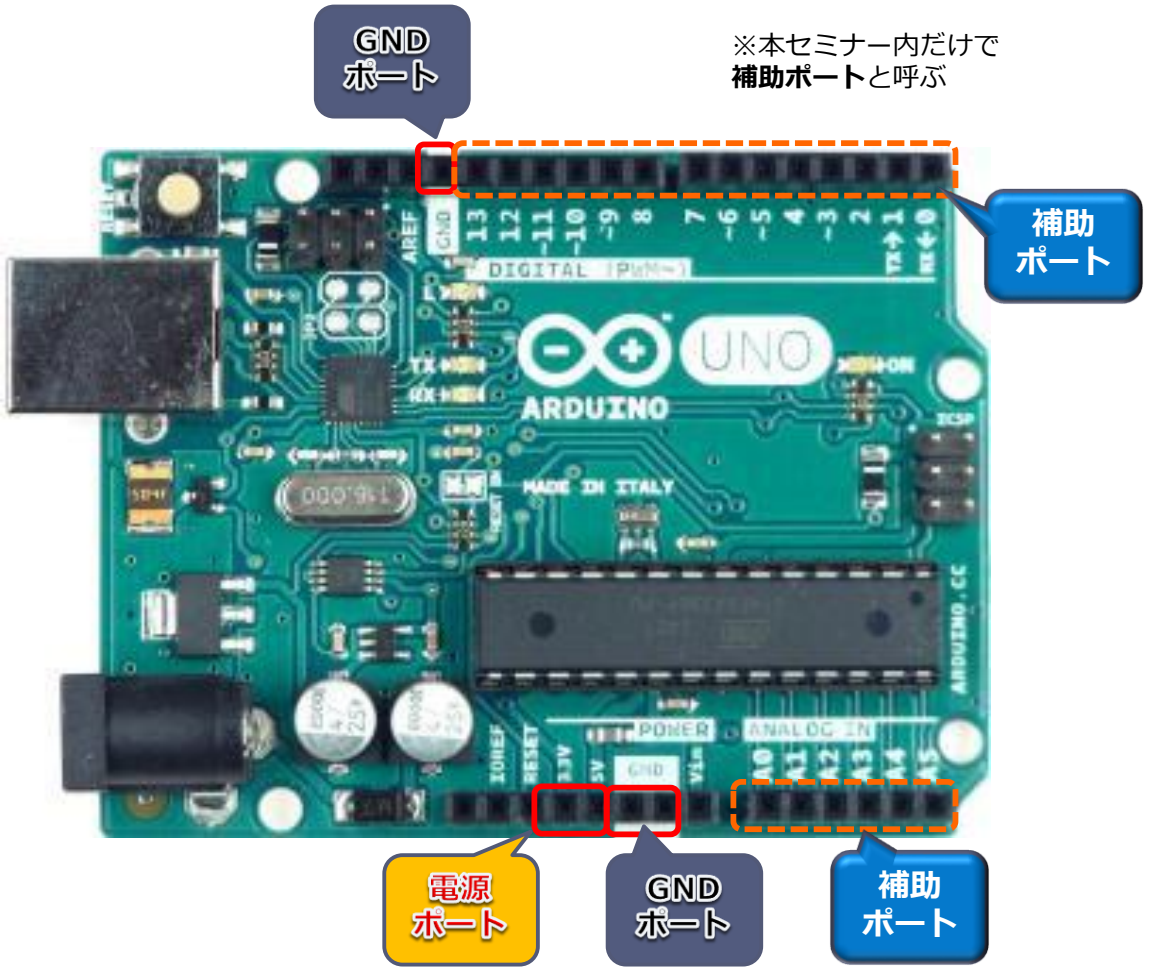
※本セミナー内だけで
補助ポートと呼ぶ

重要なテクニック

補助ポートを、
電源・GNDに設定可能

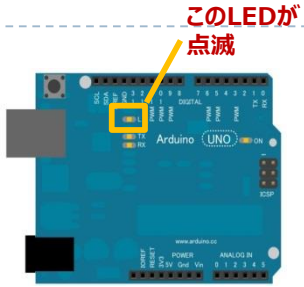
補助ポートの使い方

```
// 初期設定
pinMode(Dx,OUTPUT);
// 電源ポートの場合
digitalWrite(Dx,HIGH);
// GNDポートの場合
digitalWrite(Dx,LOW);
```



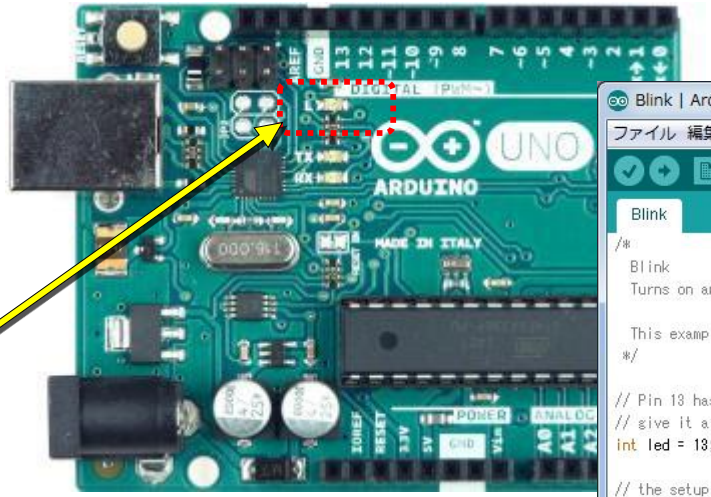
4．サンプル・スケッチを実行 ①

最も簡単な事例は、「Blink」を使った事例
・Arduino本体のみで接続・確認ができる（電子部品やブレッドボードは不要）

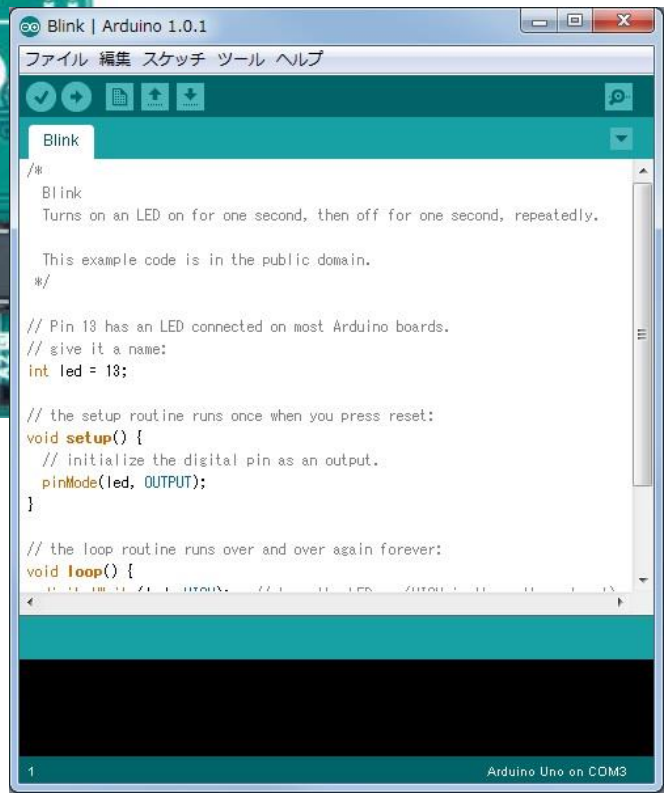


PCとの接続

デジタル ピン13に接続された
LEDと同じ扱いとなる
「Blink」スケッチを起動すると
点滅しはじめる。



写真は Arduino Uno R3



■サンプルスケッチ「Blink」を読み込み実行してみる。
メニューバーの「**ファイル**」→「**スケッチの例**」→
「**01.Basics**」 → 「**Blink**」を選択

その後、「書き込み」→「実行」を行う

4．サンプル・スケッチを実行 ②

※サンプルスケッチ「Blink.ino」を読み込んで動かしてみよう。メニューバー「ファイル」→スケッチの例」→「01．Basics」→「Blink」を選択

「//」より後の行も**コメント**

「/*」と「*/」で挟まれた間が**コメント**

```
24 // the setup function runs once when you press reset or power the board
25 void setup() {
26   // initialize digital pin LED_BUILTIN as an output.
27   pinMode(LED_BUILTIN, OUTPUT);
28 }
29
30 // the loop function runs over and over again forever
31 void loop() {
32   digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the volt
33   delay(1000);                       // wait for a second
34   digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the vo
35   delay(1000);                       // wait for a second
36 }
```

pinModeは**内部関数**

LED_BUILTINは、システム変数として、デジタル13番ポートが設定されている

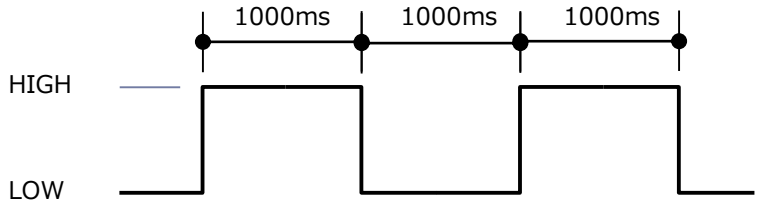
setup()は、最初の初期設定関数

loop（）は、無限ループを持った関数

この2つは必要な関数

digitalWriteとdelayは**内部関数**

「//」より後の行も**コメント**



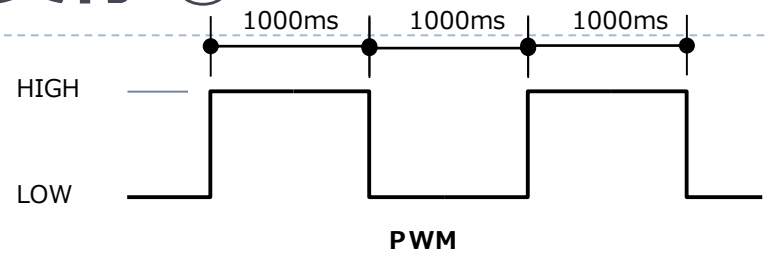
PWM（パルス幅変調：Pulse Width Modulation)
パルス波のディーティ比を変化させて変調すること

プログラムの動き
13ポートに接続されたところに、
1秒間（1000ms)ごとHIGHとLOWを繰り返す
このことで、13ポートとグラウンドに接続されたLEDは、点灯と消灯の繰り替えしを1秒間ごとに行う

【参考】**HIGH**と**LOW**の意味
HIGH（INPUTのとき）：ピンの入力電圧が3V以上のときHIGHになる
HIGH（OUTPUTのとき）：HIGHのときピンの出力電圧は5Vになる
LOW（INPUTのとき）：ピンの入力電圧が2V以下のときLOWになる
LOW（OUTPUTのとき）：HIGHのときピンの出力電圧は0Vになる

4．サンプル・スケッチを実行 ③

delayの中の数値を
変化させてみよう



PWM

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
delay(5);
digitalWrite(LED_BUILTIN, LOW);
  delay(5);
}
```
Blink01.ino

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
delay(1);
digitalWrite(LED_BUILTIN, LOW);
  delay(5);
}
```
Blink03.ino

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
delay(1);
digitalWrite(LED_BUILTIN, LOW);
  delay(10);
}
```
Blink05.ino

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
delay(5);
digitalWrite(LED_BUILTIN, LOW);
  delay(1);
}
```
Blink02.ino

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
delay(10);
digitalWrite(LED_BUILTIN, LOW);
  delay(10);
}
```
Blink04.ino

```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
delay(2);
digitalWrite(LED_BUILTIN, LOW);
  delay(2);
}
```
Blink06.ino

結果がどうなるか想像できるようになれば、プログラミングが理解できたことになります。

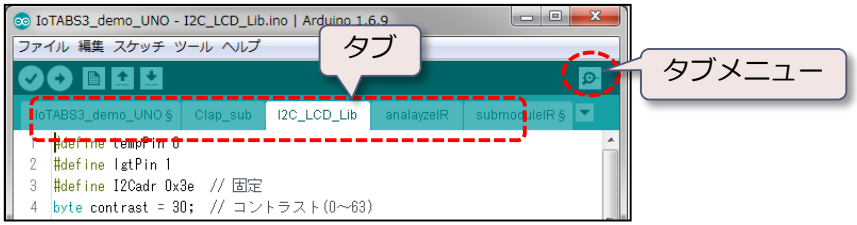
【補足】 delay は、ミリ秒単位です。
delayMicroseconds は、マイクロ秒単位です。

5．スケッチの作成方法

■IDEを使って、スケッチ作成では、初歩的なルール（文法）を知った上で、プログラミング化していきます。

（プログラミング文法参照）

■スケッチの行数が多くなっていくと、モジュール毎に整理し、**タブ画面**を利用していきます。



先頭には
1）ヘッダーファイルの読み込み
2）プリプロセッサの宣言
3）グローバル変数の宣言
などを行う

「setup」関数には
1）シリアルモニタの宣言
Serial.begin関数
2）デジタル入出力の宣言
3）初期化などなど
を行う

「loop」関数には
1）ループする制御アルゴリズムを記載する
2）同じコーディングは避け
3）短いコーディング、分かり易いコーディングを心がける
4）モジュール化を行う
など

```
/*  よく利用する関数群の定義    */
int val ( int x) {

/*  グローバル変数や
    プリプロセッサの設定   */
#define NoPin A2

/* setup関数の定義：
     初期設定を行う  */
void setup(){
  Serial.begin(9600);
}

/* loop関数の定義：
   繰り返しする実行文 */
void loop() {
  int v = val(analogRead(NoPin));
  delay(100);
  Serial.println(v);
}
```

function_tab_ex.ino

モジュール化は
1）**タブ画面**を使って整理する
2）モジュール化したりするものをタブ化を実施
（場合によっては、ヘッダーファイル化も同様）

その他
1）変数名は、分かり易い名前を採用のこと
2）グローバル変数とローカル変数を明確化すること
3）日本語（UTF-8コード）は、コメント以外で使わないこと
4）全角文字（特に空白文字）はエラーを起こしやすいので、注意が必要

6．プログラミングについて（変数を使ってみる）

方法１： 一般的な変数利用

変数timeを宣言

variable_sample01.ino

```
int tim = 100;   // 変数を利用
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(tim);
    digitalWrite(LED_BUILTIN, LOW);
    delay(tim);
}
```

変数timeを利用

変数timeを利用

変数を使うことで
変更が簡単に

【注意】一般的な変数宣言は、スコープ（利用できる範囲）内で、値を変えることができます。

方法２： プリプロセッサの＃defineを利用

プリプロセッサ
#defineでTIME宣言、PIN宣言

variable_sample02.ino

```
#define TIME 100  // プリプロセッサ
#define LED_BUILTIN  3   // ピン番号
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(TIME);
    digitalWrite(LED_BUILTIN, LOW);
    delay(TIME);
}
```

プリプロセッサ
#defineを利用
することで実行プ
ログラムは縮小化

【注意】プリプロセッサの#defineで宣言された値は、コンパイル時に置き換わります。変数扱いはできません。

6．プログラミングについて（ステップアップしてみる）

課題１：1秒間暗くし、1秒間明るくすることを繰り返す。

```
unsigned long time;
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  time = millis();
  do{
    digitalWrite(LED_BUILTIN, HIGH);   // set the LED on
    delay(5);              // wait for a second
    digitalWrite(LED_BUILTIN, LOW);    // set the LED off
    delay(15);    }         // wait for a second
  while(time+1000> millis());
  time=millis();
  do{
    digitalWrite(LED_BUILTIN, HIGH);   // set the LED on
    delay(15);             // wait for a second
    digitalWrite(LED_BUILTIN, LOW);    // set the LED off
    delay(1);    }         // wait for a second
  while(time+1000> millis());
}
```

LED_sample_01.ino

どこをどう変更するか
注意深く見てください

ここでのプログラムをじっくり観察し、
どう稼働するかを理解してください。

【参考】millsf関数は、
プログラムを起動してから経過した時間をミリ秒単位で返す。

6．プログラミングについて（関数を作ってみる）

関数を使うことで、プログラムがモジュール化でき、まとまった処理として、同じ処理の繰り返しや長くなる処理をまとめたりすることでプログラミングが短くなり見やすくなります。

関数を使って、LEDの明かりを暗くしたり、明るくしたりを繰り返す。

LED_sample_02.ino

```
#define LED_BUILTIN 13
#define TIME 1000
void BlinkFnc( boolean sw)
{ digitalWrite(LED_BUILTIN, sw); delay(1);
  digitalWrite(LED_BUILTIN, !sw ); delay(10);}

unsigned long time;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  time = millis();
  do{ BlinkFnc(true); }
  while(time+TIME> millis());
  time=millis();
  do{ BlinkFnc(false); }
  while(time+TIME> millis());
}
```

プリプロセッサ
#defineでTIME宣言

LED_sample_03.ino

```
#define LED_BUILTIN 13
#define TIME 1000
unsigned long time;

void BlinkFnc( boolean sw)
{  time = millis();
   do{ digitalWrite(LED_BUILTIN,  sw);  delay(1);
       digitalWrite(LED_BUILTIN, !sw ); delay(10);}
  while(time+TIME> millis());
}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  BlinkFnc(true);
  BlinkFnc(false);
}
```

関数（プロシジャ）宣言

関数宣言文の理解
型 関数名（引数）
｛ 処理群 ｝

関数も「システム」の考え方として、**引数**と**戻り値**を持ち、**引数**が「入力」で、**戻り値**が「出力」、そして内部のプログラミングが「処理」となります。

6．プログラミングについて（引数を変更してみる）

LEDの点滅を例に、明るさを変更する事例を紹介します。
ここでは、どんな処理を行っているか理解してみましょう。

```
#define LED_BUILTIN 13
#define TIME 500
unsigned long time;

void BlinkFnc( boolean sw)
{  time = millis();
   do{ digitalWrite(LED_BUILTIN,  sw);  delay(1);
       digitalWrite(LED_BUILTIN, !sw ); delay(10);}
  while(time+TIME> millis());
}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  BlinkFnc(true);
  BlinkFnc(false);
}
```

時間を変えてみる

LED_sample_04.ino

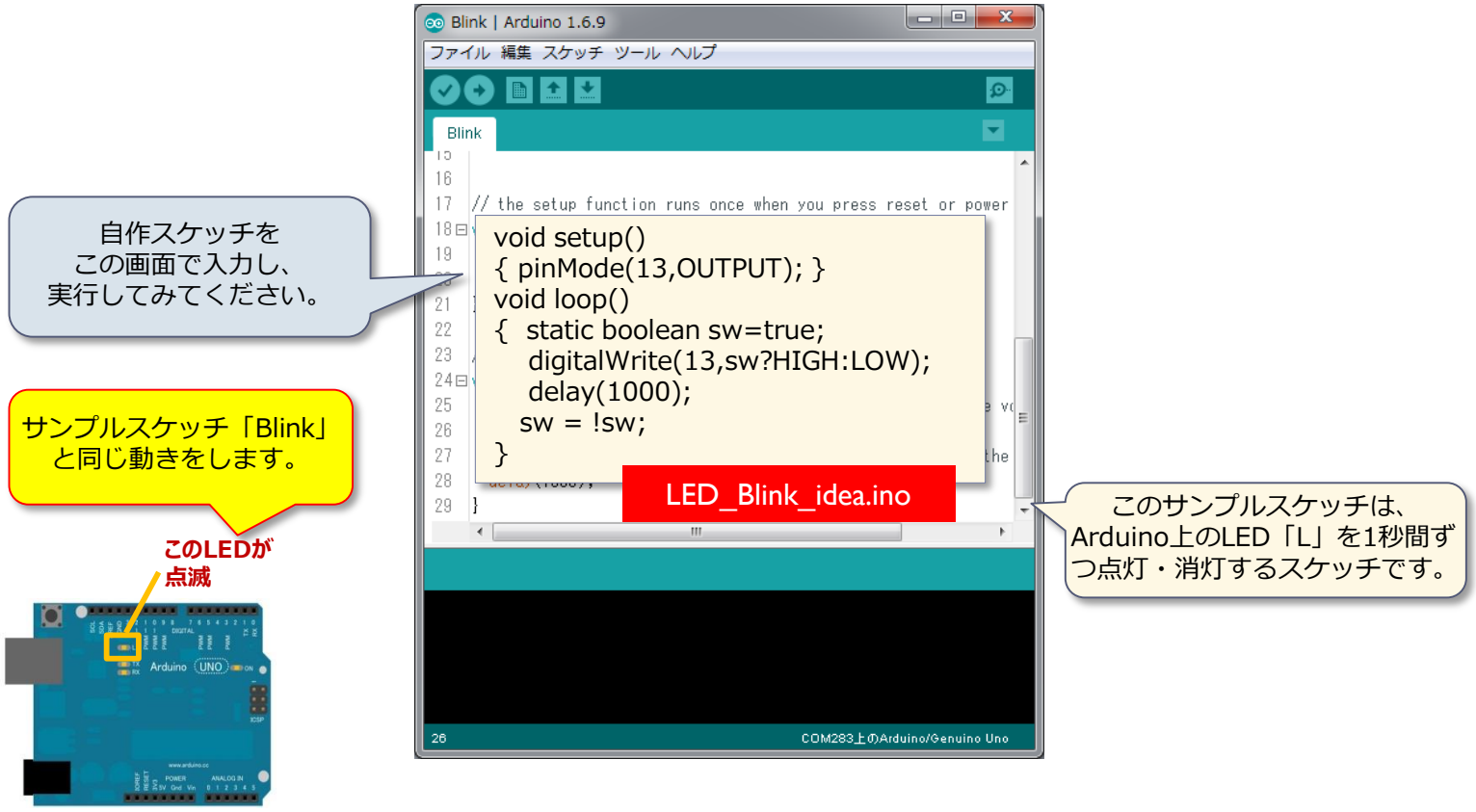
ここで使っている引数 swがどんな役割をしているか？
SW＝true　：HIGH
SW=false　：LOW

do{
　・・・・
}while（time + TIME>millis());

このことで、TIME （ミリ秒）間繰り返していることをご理解ください。

7．自作スケッチを実行

■さきにArduinoとUSBケーブルを繋いでおきます。
※IDEの右下に接続されたシリアルポートが表示されています。もし表示無い場合には、メニュー「ツール」の「シリアルポート」から、ArduinoのUSBケーブルを選択します。
■Arduinoプログラム（スケッチと呼ぶ）は、IDEを開いて、エディタ編集して作成していく。また、コンパイルし、Arduinoに書き込んで実行させます。



8．Arduinoプログラミングの可能性について

マイコンボード の Arduino は、ここで紹介する以下の簡単なプログラミングの勉強もできるようになっています。

１）四則演算 （整数・実数）

２）文字列処理

３）分岐（判断）・反復（繰り返し）の制御処理

４）さまざまなアルゴリズム検証処理

５）統計処理（平均・標準偏差など）

その他、多次元配列処理や、構造データ処理などの複雑な処理も可能となっています。

ボードを使った処理では、センサの値をどう処理するかといった、数値処理・統計処理・多次元処理など、数学的な処理が多く出てくるものと考えます。

これらのことも考え、ここでは　初心者向けに Arduinoの使い方を交えてサンプルスケッチを紹介していますので、ひとつひとつどのように動いているかを確認しながら進んでいってください。

（１）サンプルスケッチ①（整数の四則演算）

整数の変数を使った四則演算（和・差・積・商、加減乗除）を行ってみましょう。
その他剰余（%）や１つずつの加算・減算（++,--）についても紹介しましょう。

int 変数（ここでは l ,m,n）に、初期値として整数値を入力することが可能。また「,」で区切り複数入れる事が可能

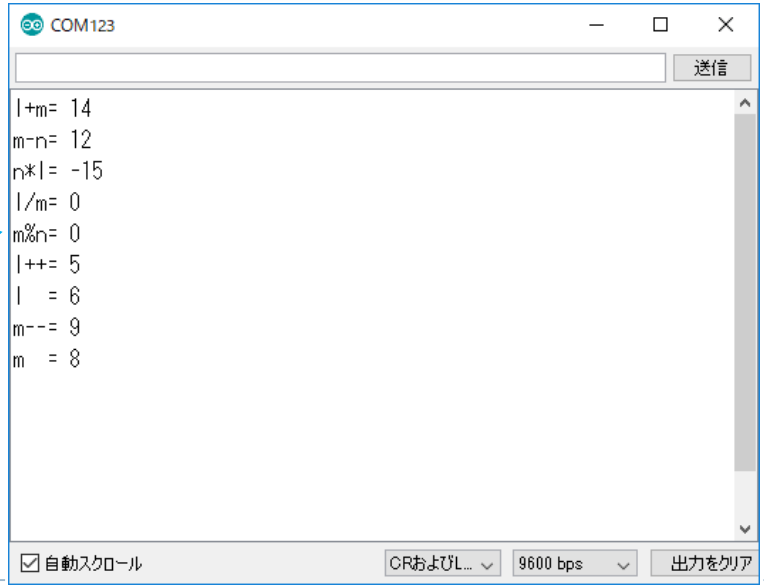
```
int l=5, m=9,  n= -3;
void setup() {
 Serial.begin(9600);
 Serial.print("l+m= "); Serial.println(l+m); // 5+9=14
 Serial.print("m-n= "); Serial.println(m-n); // 9-(-3)=12
 Serial.print("n*l= "); Serial.println(n*l); // -3*5=-15
 Serial.print("l/m= "); Serial.println(l/m); // 5/9 = 0
 Serial.print("m%n= "); Serial.println(m%n); // 9%-3 = 0
 Serial.print("l++= "); Serial.println(l++); // 5 but l=6;
 Serial.print("l  = "); Serial.println( l ); // l=6;
 Serial.print("m--= "); Serial.println(m--); // 9 but m=8
 Serial.print("m  = "); Serial.println( m ); // m=8
 }
void loop() {}
```

Basic_calc_sample1.ino

ここで、以下はシリアル出力機能で
Serial.print()：改行なし
Serial.println()：改行あり
このカッコ内で演算処理も可能

「m%n」はmをｎで割った時の余りを算出する　９は-3で割り切れるので０　mを10に変えると１と計算値が算出される

「＋＋」 変数にプラス１する ここではｌに１を足してｌの値を６に変更している



（２）サンプルスケッチ②（実数の四則演算）

実数の四則演算関係のサンプルプログラムを稼働してみましょう。
一部、型変換（実数から整数）や型変換関数を使っています。ともに切り捨て処理になっている点に気を付けましょう。

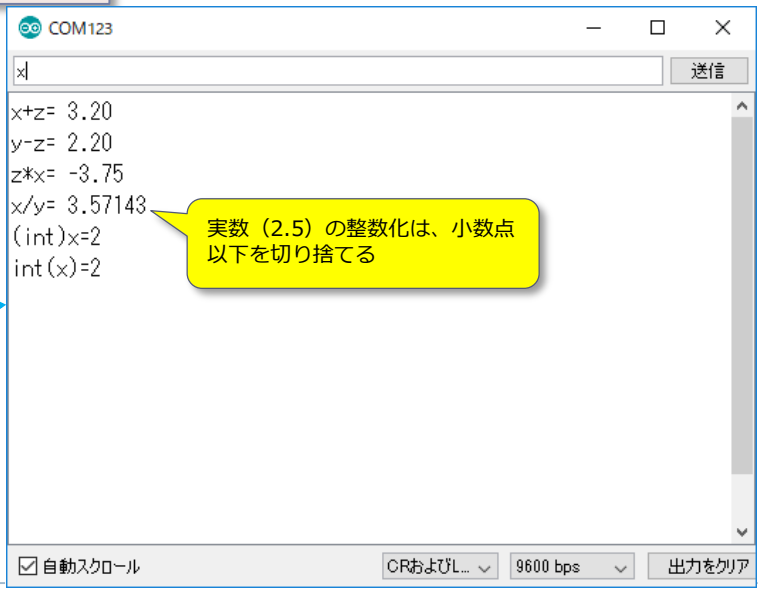
float 変数（ここではx,y,z）に４バイト実数値を入力　「,」で区切り複数入れる事ができる

```
float x=2.5, y=0.7,  z= -1.5;
void setup() {
 Serial.begin(9600);
 Serial.print("x+z= "); Serial.println(x+y);       // 2.5+0.7=3.2
 Serial.print("y-z= "); Serial.println(y-z);       // 0.7-(1.5)=2.2
 Serial.print("z*x= "); Serial.println(z*x);       // -1.5*2.5=-3.75
 Serial.print("x/y= "); Serial.println(x/y,5);     // 2.5/0.7=3.57143..
 Serial.print("(int)x="); Serial.println((int)x); // 型変換
 Serial.print("int(x)="); Serial.println(int(x)); // int関数
 }
void loop() {}
```

Basic_calc_sample2.ino

「(int)変数」にfloat型の変数を整数に変更している

int関数に引数xが引き渡されている。



（3）サンプルスケッチ③（文字列処理）

プログラミングの処理では、文字列処理が多く出てくる場面があります。文字列処理では、クラスライブラリとしてString関数を使うことで便利に処理できることもありますが、一部これまでの文字配列（型がchar）を使うことでも簡単に処理することができます。

```
String s = "Trillion-Node";
char c[15] ;

void setup(){
  while(!Serial);
  Serial.begin(9600);
  String sb = s.substring(9);
  Serial.println("substring = " + sb);
  int ln = s.length();
  Serial.println("length = " + String(ln));
  sb=s;   sb.replace("-"," ");
  Serial.println("replace = " + sb);
  s.toCharArray(c,ln+1);
  Serial.println(c);
  sb = String(c).substring(0,8);
  Serial.println(sb);
  float f = 23.024;
  sprintf(c,"x=%2d.%03d",(int)f,(int)(f*1000)%1000);
  Serial.println(c);
}
void loop() {}
```

文字列変数「S」宣言し、文字列入力

文字配列（15文字）変数「c」宣言

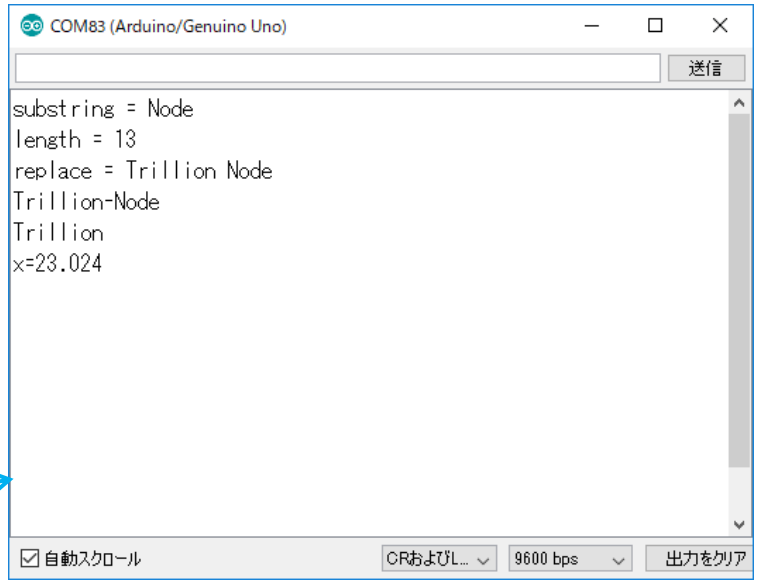
「while(!Serial); 」はシリアル画面が有効になるまで待機

シリアル通信の速度（9600bpsのこと）

実数表示で、整数部と小数点以下の数値を分けて処理している

ここでのString 関連の関数群紹介
s.substring(i)　：文字列の分割１
s.substring(i,j)：文字列の分割２
s.length()：文字列長さ
s.replace（s0,s1)：文字列の変更
s.toCharArray(c,l):文字列から文字配列変換
sprintf(c,s,v0..Vn)：文字配列への変換
詳細は、別途ネット参照

Basic_String_char_sample.ino



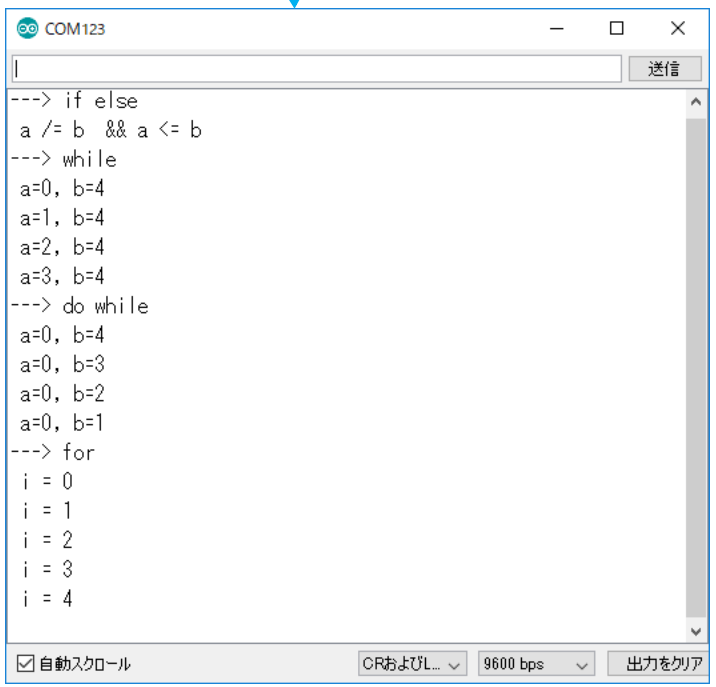
（４）サンプルスケッチ④（制御文）

Arduinoでは、制御文として「if-else」、「while」、「do-while」、「for」、「switch」文などがあります。
これらを使いこなすことで、複雑な処理を簡素なプログラミングにまとめることができます。特にこの制御文は効率良い使い方が重要となりますので、良質な例文を数多く参考にすることをおすすめします。

```
int a = 0, b = 4, c = -2;
char ch[20];

void setup() {
 Serial.begin(9600);
 Serial.println("---> if else");
 if ( a==b )  Serial.println ( " a== b ");
 else if( a>b )  Serial.println( " a>b ");
 else Serial.println( " a /= b  && a <= b");
 Serial.println("---> while");
 while( a != b ) {
   sprintf(ch," a=%d, b=%d",a,b);
   Serial.println(ch);
   a++;
 }
 Serial.println("---> do while ");
 a=0;
 do {
     sprintf(ch," a=%d, b=%d",a,b);
     Serial.println(ch);
     b--;
 } while(a!=b);
 Serial.println("---> for");
 for ( int i=0; i< 5; i++ ) {
   Serial.print(" i = ");
   Serial.println(i);
 }
}

void loop() { }
```

Basic_control_sample1.ino

別途　文法の「制御文」参照
if..else if..else.. : 条件
while.. : 条件＋繰り返し
do..while : 繰り返し＋条件
for .. : 繰り返し



（５）サンプルスケッチ⑤（アルゴリズム検証）

Arduinoのマイコンボードでも簡単に**並び替え**（**ソート**）などのアルゴリズム検証にも使えます。
ここでは、**クイックソート**のアルゴリズムを使ってプログラミング化し、8個のデータを並び替えてみます。
この程度だと、瞬時に結果を出すことができます。

```
//  クイックソート
// 参考「C言語によるアルゴリズムとデータ構造」柴田望洋＋辻亮介著
#define swap(type,x,y) do { type t=x; x=y; y=t;} while(0)

void setup() {
  int i;
  int x[] = { 175, 170, 160, 168, 165, 173, 155, 150 };
  int nx = sizeof(x) / sizeof(x[0]);
  while(!Serial);
  Serial.begin(9600);
  Serial.println(String(nx) + " data  sort");
  Serial.println("before -> ");
  for(i=0; i<nx; i++) Serial.print(String(x[i]) + " ");
  Serial.println();
  quick(x, 0, nx-1);
  Serial.println("after  -> ");
  for(i=0; i<nx; i++) Serial.print(String(x[i]) + " ");
}

void loop(){}
```

初期設定

xのサイズ算出

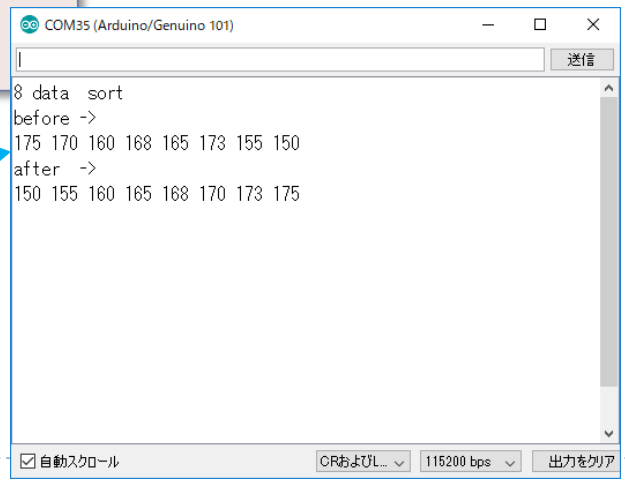
シリアル画面が起動するまで待機

クイックソート関数呼び出し

Basic_quick_sort.ino

```
void quick(int a[], int left, int right) {
   int pl = left;
   int pr = right;
   int x = a[(pl+pr)/2];
   do{
     while(a[pl]<x) pl++;
     while(a[pr]>x) pr--;
     if (pl <=pr) {
        swap(int , a[pl], a[pr]);
        pl++;
        pr--;
     }
   } while(pl<=pr);
   if (left<pr) quick(a, left,pr);
   if(pl<right) quick(a, pl,right);
}
```

ソートの昇順や降順の並び替えのアルゴリズムはいろいろ存在します。ここでは一般に高速な処理として利用されている「クイックソート」を紹介しています。



「プログラミング」は、「**データ構造**」と「**アルゴリズム**」の２つから成り立ちます。ここではアルゴリズムの勉強として広く紹介されているソートの中のひとつの「クイックソート」を紹介しています。詳細は、別途ネット上から検索してご理解ください。

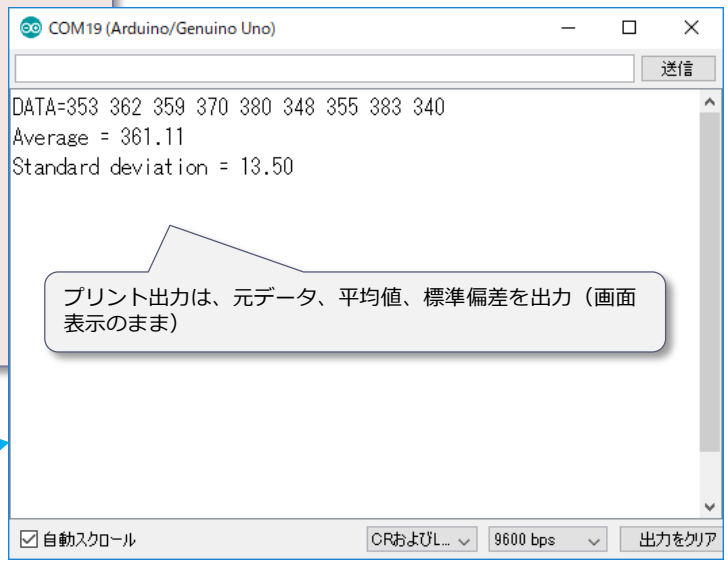
（6）サンプルスケッチ⑥（平均と標準偏差）

センサ値を複数回取得するとき、必ずしも一つの値で正確とならず、平均を取ることや、そのばらつき（標準偏差）の算出が重要になることがあります。ここでは、平均と標準偏差の処理を記載しておきます。
＜例えば、河川の水位を距離センサで計測した場合、常に水面が揺れているのを計測する場合、複数回計測し、その平均値や、水面の揺れの大きさを標準偏差で認識することができます＞

```
void setup()
{
  Serial.begin(9600);
  int dat[] = {353, 362, 359, 370, 380, 348, 355, 383, 340};
  int NN = sizeof(dat)/sizeof(dat[0]);
  Serial.print ("DATA=");
  for (int i = 0; i < NN; i++) {
    Serial.print(String(dat[i]) + " ");
  }
  Serial.println();
// 平均値
  float sum=0;
  for (int i = 0; i < NN; i++)  sum += dat[i];
  float mean = sum / (float)NN;

// 標準偏差
  float dev  = 0;
  for (int i = 0; i <NN; i++)  dev += sq((float)dat[i] - mean);
  float std = sqrt(dev / NN );

  Serial.print("Average = "); Serial.println(mean);
  Serial.print("Standard deviation = "); Serial.println(std);
}
void loop(){}
```

Basic_Ave_StDev.ino

【補足】
「a = a+1」 は「a++」・「++a」と同じ
だだし、応答値が異なる、前者は「a」後者は「a+1」
「a = a+b」 は 「a += b 」と同じ
「a = a-b」は「a -= b」と同じ



【補足】「sq(x)」関数は、平方(x＊x)を算出する関数です。
「sqrt(x)」関数は、平方根(√x)を算出する関数です。
その他「pow(x,y)」関数は、xのべき乗(y)を算出する関数です。

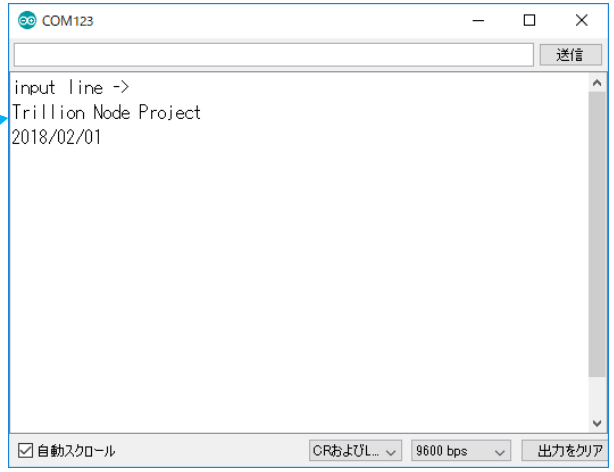
（７）簡単なサンプルプログラム集①

シリアルモニタ画面でのキーボード入力をそのまま画面に表示するスケッチです。
改行コードがあれば、それまでの一行を読み込んで、シリアル画面に表示します。

```
void setup() {
  Serial.begin(9600);
  Serial.println("input line ->");
}

void loop() {
  if (Serial.available())
    Serial.println(Serial.readStringUntil('¥n'));
}
```

sample_1.1.ino

【補足】
'¥n' は、改行文字コード
'¥r'は、ラインフィード文字コード
'¥t'は、タブ文字コード



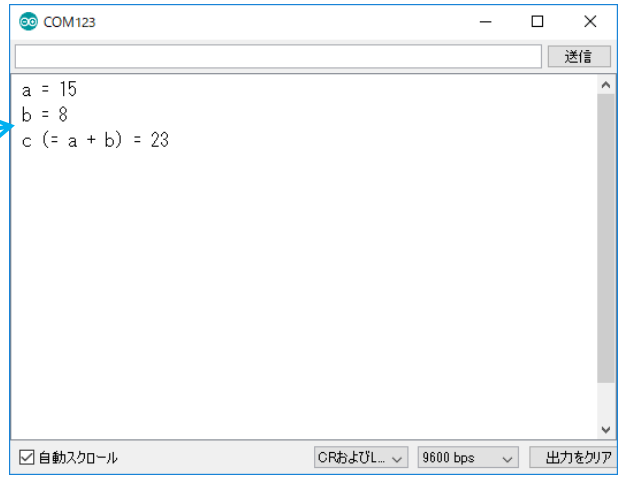
（８）簡単なサンプルプログラム集②

変数を使って、加算を行うもので、一連の流れをシリアルモニタ画面への表示を行います。

```
void setup() {
  Serial.begin(9600);
  int a,b,c;
  a=15;
  b=8;
  c=a+b;
  Serial.println(" a = " + String(a));
  Serial.println(" b = " + String(b));
  Serial.println(" c (= a + b) = " + String(c));
}

void loop() { }
```

sample_1.2.ino

【補足】
Serial.println(s0+… + sn);
ここでs0~snは、文字列を連結して表示
数値や文字などは、一旦文字列関数で変換必要
そのためString(v)　を利用
ここで v は、整数・実数などの数値や文字配列など



（９）簡単なサンプルプログラム集③

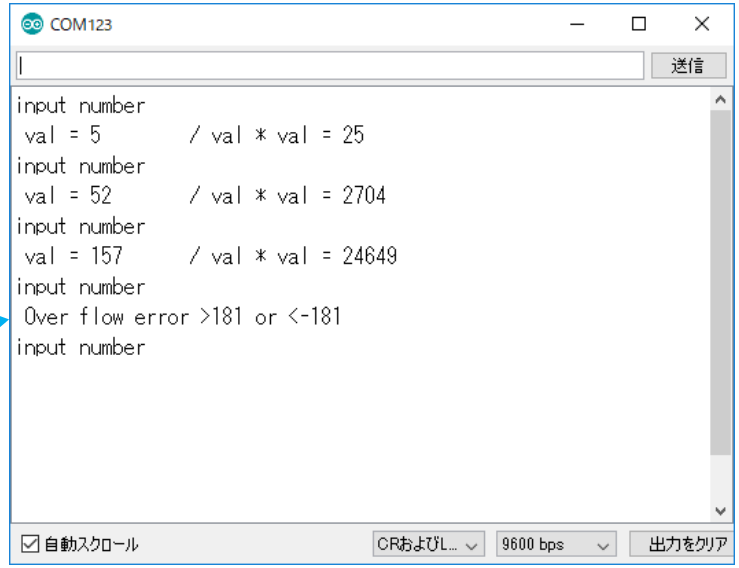
入力された整数の二乗（ n*n ）を行うプログラミングです。単精度整数のため入力値が-181～181の制限があります。
これを超えるとエラー処理を行います。
ここでは、タブ（0x09）コードを使ったりした処理も行っています。

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println("input number");
  while (!Serial.available());
  int val = Serial.readStringUntil('¥n').toInt();
  if( val > 181 || val < -181) {
      Serial.println(" Over flow error >181 or <-181");
      return ;
  }
  Serial.print( " val = " + String(val) );
  Serial.write(0x09);
  Serial.println("/ val * val = " + String(val * val));
}
```

sample_1.3.ino

【補足】
数値の文字列を数値に変換するとき
s.toInt() で 整数文字列sを整数に変換
String.write(x)は、文字コードｘで出力

「while(!Serial.available()); 」は、シリアル通信接続可能になるまで待機の意味



（１０）簡単なサンプルプログラム集④

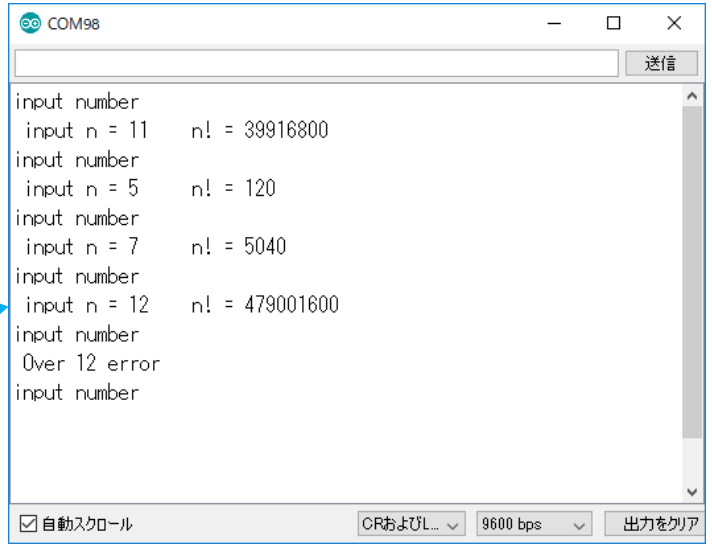
入力された整数の階乗（n！ ＝ １＊２＊３.. ＊ n ）の計算を 行うプログラミングです。単精度整数のため入力値が１～12の制限があります。
これを超えるとエラー処理を行います。

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println("input number");
  while (!Serial.available());
  int n = Serial.readStringUntil('¥n').toInt();
  if (n > 12) {
    Serial.println(" Over 12 error");
    return;
  }
  long val = 1;
  for ( int i = 1; i <= n; i++)  val = val * i;
  Serial.print( " input n = " + String(n) + char(0x09));
  Serial.println(" n! = " + String(val));
}
```

sample_1.4.ino

【補足】
「val = val * i」　は、　「val *= i 」に変更可能

【補足】
loop関数内の「return」は、関数終了の意味で、先頭に戻る



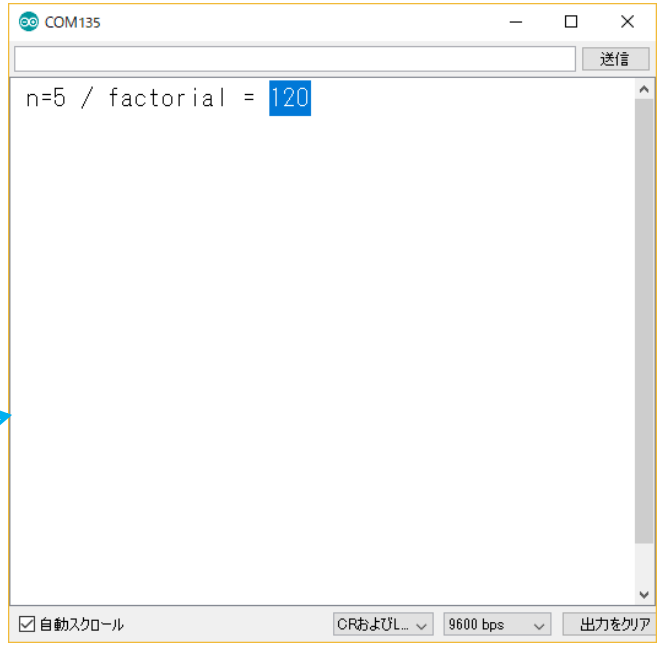
（１１）簡単なサンプルプログラム集⑤

ここでのプログラムは、階乗（n!＝1*2*・・・*n）計算を **再起呼び出し**（リカーシブ）で処理するものです。再起呼び出しは、関数が自分の関数を呼び出すことを意味します。この再起呼び出しは、プログラムが簡単になり、理解しやすくなる半面、呼び出しの深さが深くなるとメモリーを消費しいくことから、スピード面で注意が必要となります。

```
void setup() {
  Serial.begin(9600);
  int n= 5;
  Serial.print(" n=" + String(n) + " / factorial = ");
  Serial.print(factorial(n));
}

void loop() {
}

long factorial( long n ) {
  if ( n==0 ) return 1;
  else return n*factorial(n-1);
}
```

sample_1.6.ino

【補足】
factorial　関数の中で、自分を呼び出しています



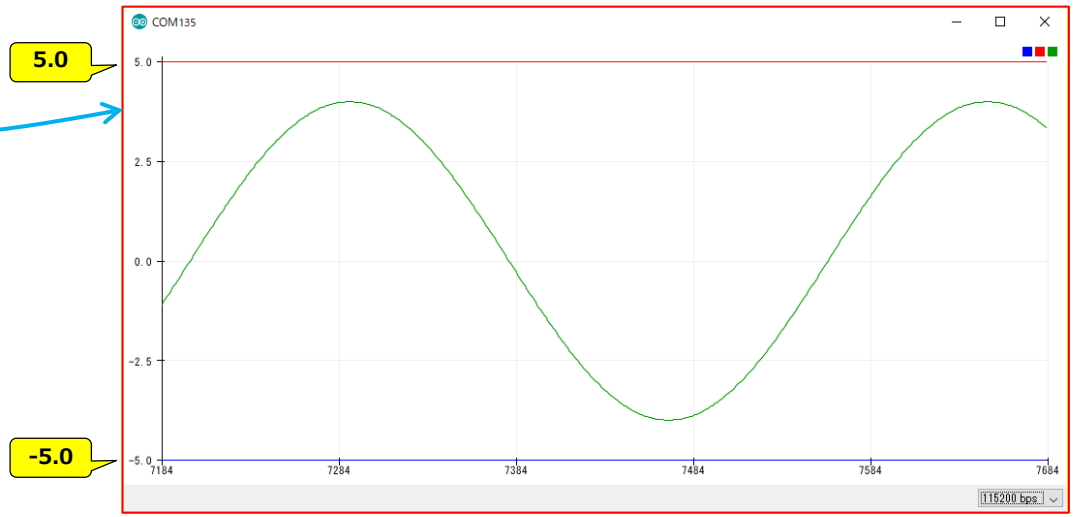
（１２）簡単なサンプルプログラム集⑥

ここでは、プログラミングによって正弦波であるSINカーブを描いてみましょう。ダイナミック（動的）に動き続けるサインカーブを描きます。ここでは、最大最小を５と-５とし、振幅を８に設定して描いています。

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  static float x = 0.0;
  Serial.print("-5.0,5.0,");
  Serial.println(4.0*sin(x/180.0*PI));
  x+= 1;
}
```

Serial.printl(“-5.0,5.0”);は、グラフの最小値（-5.0）と最大値(5.0)を表示させるためのものです

sample_2.1.ino



【補足説明】右の図のように最大値を「5.0」と最小値を「-5.0」に設定することで、ダイナミックなグラフを表示させるときには見やすくなります。

【演習】振幅100の余弦波（COS波）を描いてみましょう。

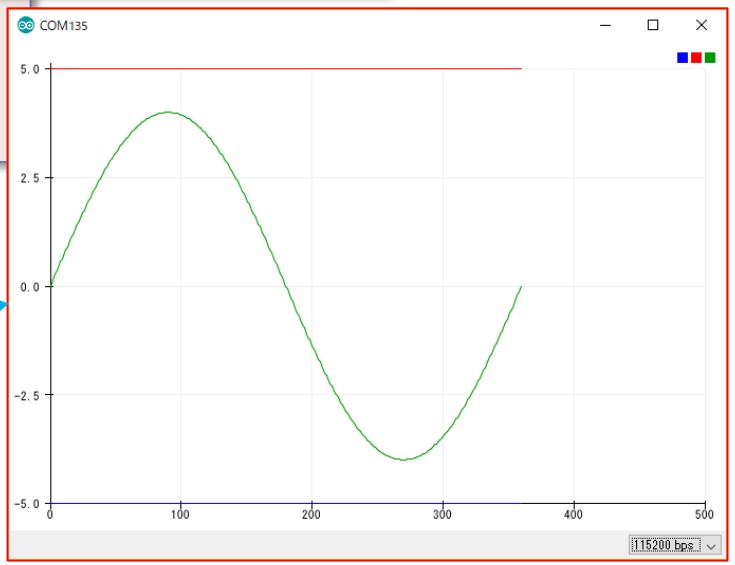
（１３）簡単なサンプルプログラム集⑦

SINカーブを１周期である０度から360度で描いてみましょう。

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  static float x = 0.0;
  Serial.print("-5.0,5.0,");
  Serial.println(4.0*sin(x/180.0*PI));
  x+= 1;
  if(x>360) while(1);
}
```

sample_2.2.ino

【補足】
角度x（度）が360を超えたとき
while(1)の処理となるが、
これ「while(1)」は終了し、無限に留まることで、その後の処理は何もないこと



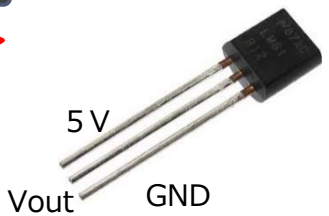
9．電子部品の取扱いについて

電子部品の利用上の注意点
1） 部品の種類について
・類似品があるが、性能・規格を確認のこと
2） 部品の注意点について
・最大値（最大定格）を守って利用のこと
・電源とグラウンドの極性を守ること
・足ピンの接続においては注意を払うこと
3）熱の考慮について
・発熱する電子部品（抵抗など）は要注意

Arduino上で使う電子部品

入力 → 処理 → 出力

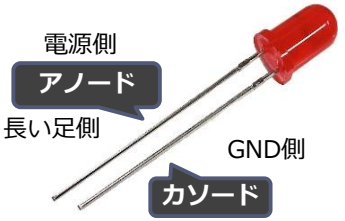
チルトセンサ（４ピン）



温度センサ（３ピン）

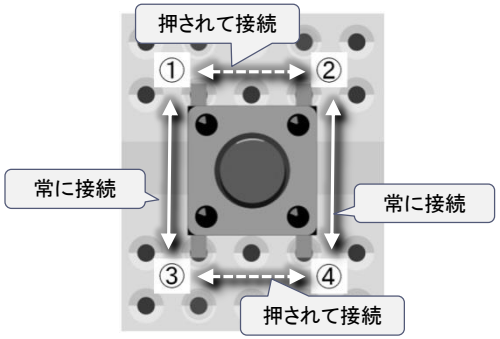

可変抵抗器（３ピン）


超音波距離センサ（４ピン）

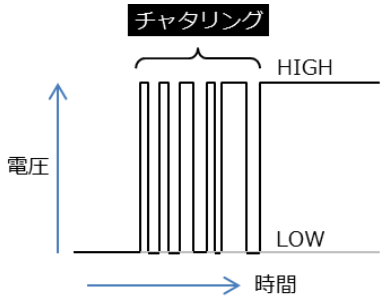


LED（２ピン）


スピーカ（２ピン）



タクトスイッチ（４ピン）



チャタリング


光センサ（２ピン）
<＋10KΩ抵抗>

Arduino上での利用の注意点
1） 極性のある電子部品の取扱い注意
2）抵抗などが必要な電子部品に注意
3）アナログ・デジタル・シリアル通信に配慮
4）足ピンの意味を理解して利用

[ブレイク] Arduino方言ほか

- Arduinoでは、一部の特殊な語彙を用いている
  - プログラムのことを　→　スケッチ
  - 拡張ボードのことを　→　シールド
  - 秘法や秘訣のことを　→　レシピ

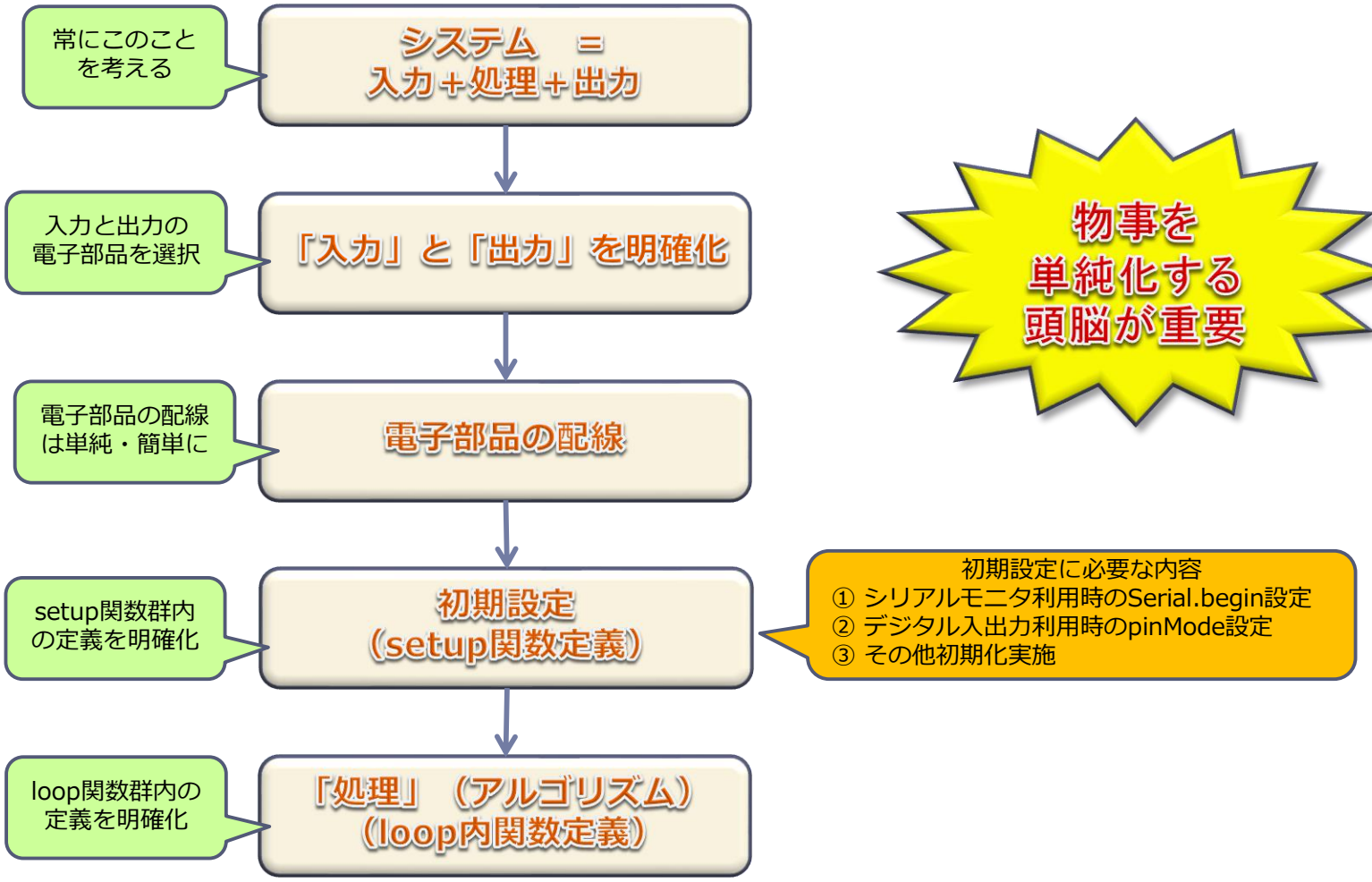
Arduinoが芸術系（アート、クリエイターによく利用されるため）

- フィジカルコンピューティングとは
  - ニューヨーク大学から始まった教育プログラム、研究指針で、既存のPCのグラフィカル・ユーザー・インタフェース（ウィンドウ、マウス、アイコンなど）を超えて、身の回りの生活環境によりそった身体的なコンピュータのあり方を模索する研究の動向を言い表す。（ネット上から）
  - エレクトロニクスを使ってデザイナーやアーティストのために新しい素材を生み出す。

第Ⅱ編　技術編

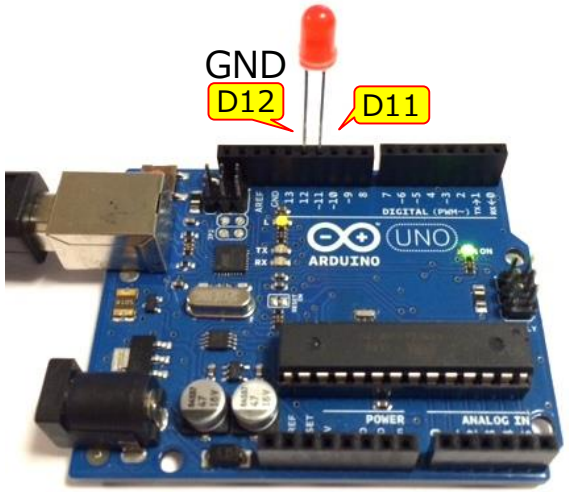
第４章
Arduino UNO の初級利用

1．簡単にArduinoを学ぶフロー



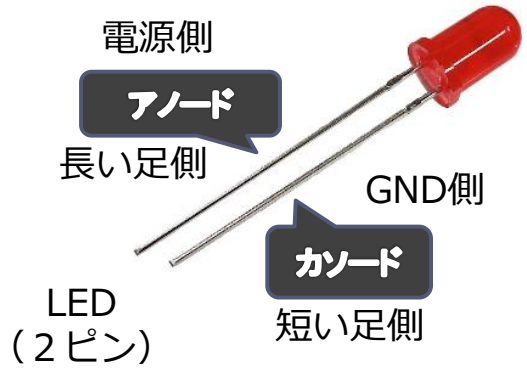
デジタル出力
アナログ出力

2．Arduinoの基本（単体LED ①）



LED　D11 ＆ D12（GND）

部品接続



LED部品説明図

製品番号：OSR6LU5VB64A-5V

課題１： デジタル出力での段階的に明るくLED点灯
課題２：アナログ出力での段階的に明るくLED点灯
課題３： 特殊なLED点滅 （tone関数を使ってみよう）

IoTABシールド
LED：D3-D8

デジタル出力
アナログ出力

2． Arduinoの基本（単体LED ②）

【課題１】段々と明るくデジタル出力

```
void setup(){
  pinMode(12,OUTPUT);
  digitalWrite(12,LOW);
  pinMode(11,OUTPUT);
}
void loop(){
  for(int i=0; i<255; i++ ) {
    long tm = millis();
    do{
     digitalWrite(11,HIGH);
     delayMicroseconds(i);
     digitalWrite(11,LOW);
     delayMicroseconds(255-i);
    }while(millis()-tm<10);
  }
}
```

D11 にアノード（電源）
D12 にカソード（GND）

繰り返し（0～255）
・段々と明るく

IV-02LED-01.ino

【課題２】段々と明るくアナログ出力

```
void setup(){
  pinMode(12,OUTPUT);
  digitalWrite(12,LOW);
}
void loop(){
  for(int i=0; i<256; i++){
    analogWrite(11,i);
    delay(10);
  }
}
```

D12 に
カソード（GND）

繰り返し（0～255）
・段々と明るく

IV-02LED-02.ino

【課題３】tone関数で点滅（特殊）

```
void setup(){
  pinMode(12,OUTPUT);
  digitalWrite(12,LOW);
  pinMode(11,OUTPUT);
}
void loop(){
  tone(11,250,1000);
  delay(2000);
}
```

IV-02LED-03.ino

tone関数を使って
1秒間隔でLED点滅

LED取扱いの注意点：
1）一般にLEDには抵抗が必要
（抵抗付きLEDも存在）
2）アナログ出力とデジタル出力
の両方で制御可能
3）D13にArduino上のLEDと連動

アナログ出力

3．Arduinoの基本（圧電スピーカ）

■ ポイント

- **アナログ出力電子部品**
  - ① LED
  - ② スピーカ（特殊ケース）
- **アナログ出力の接続ポート**
  - ain: D3,D5,D6,D9,D10,D11
- **アナログ出力関数（PWM）**
  - analogWrite(ain,val);
  - ここで　書き込み値 val = 0～255

■ 事例

128=256/2

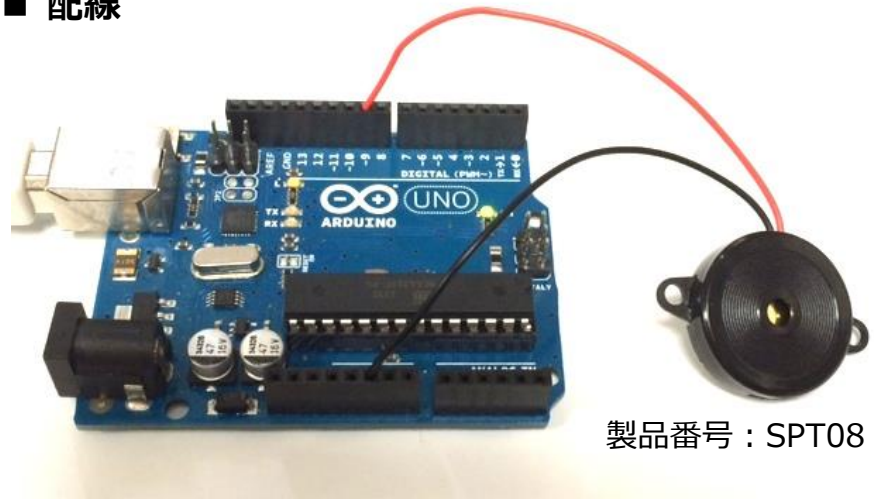
IV-03SPK.ino

```
void setup() { }
void loop() {
  analogWrite(9,128);
  delay(500);
  analogWrite(9,0);
  delay(500);
}
```

0.5秒間ごとに音を出したり消したりする

IoTABシールド
スピーカ：D9

■ 配線



製品番号：SPT08

スピーカ
GND & D9

■ 注意点

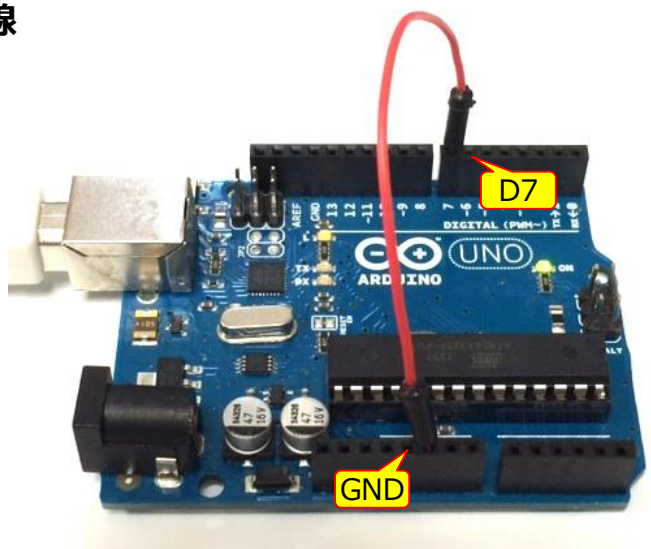
① スピーカは、本来デジタル出力
② LEDもアナログ出力できるが、
　デジタル出力がより安心

4．Arduinoの基本（ジャンパワイヤ）

■ ポイント

- **デジタル入力電子部品**
  - ① タクトスイッチ
  - ② スライドスイッチなど
  - ③ 超音波距離センサなど
- **デジタル入力の接続ポート**
  - din: 0～13 または A0~A5（14~19）
- **デジタル入力設定・読込関数**
  - pinMode(din,INPUT/INPUT_PULLUP);
  - digitalWrite(ain,HIGH/LOW);

■ 配線



ケーブル（スイッチ）
GND-D7

■ 事例

```
void setup(){
  pinMode(7,INPUT_PULLUP);
  Serial. begin(9600);
}
void loop(){
  Serial.println(digitalRead(7));
  delay(100);
}
```

pinMode設定と
シリアルモニタ設定

0.1秒毎シリアルモニタに
センサ値表示

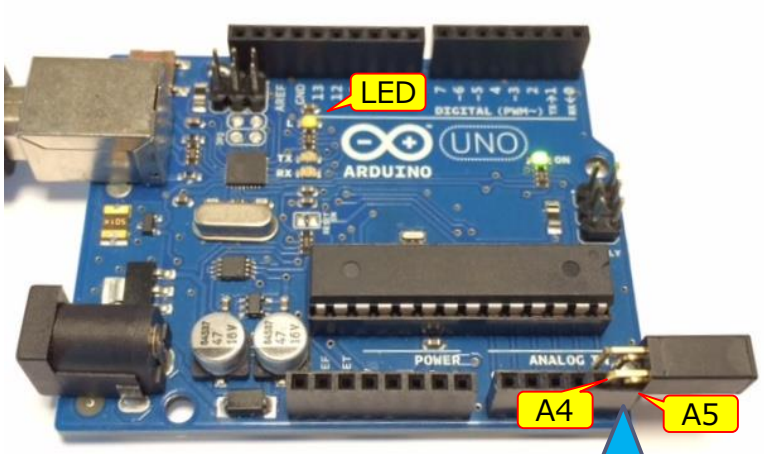
IV-04SWT.ino

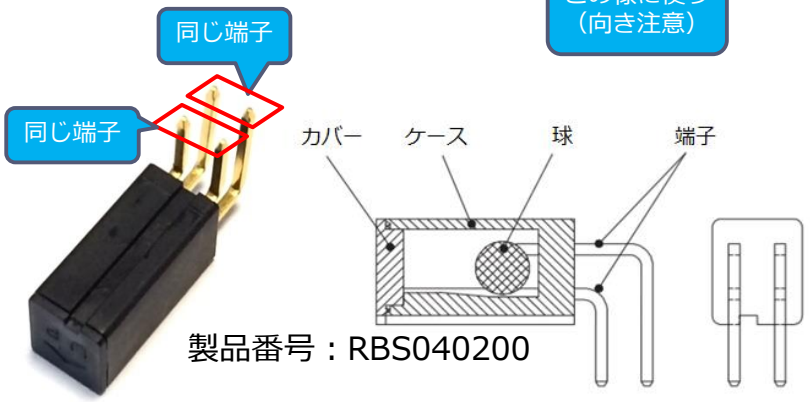
IoTABシールド
タクトスイッチ：D2

■ 注意点

① スイッチなどはプルアップ抵抗を考慮
② Offの状態は「HIGH」で、
Onの状態は「LOW」となる。

5．Arduinoの基本（チルトセンサ）





製品番号：RBS040200

チルトセンサは、傾きによって、スイッチが入る

注意点：

① スイッチが入ったとき　LOW （＝0）となり
　スイッチが切れたとき　HIGH（＝1）となる

② チルトセンサの片方をGNDにする。

③ pinMode関数で、第２引数に「INPUT_PULLUP」
　（プルアップ抵抗）を利用すること

課題：
　チルトスイッチによってスイッチがOn/Offの状態で
LED（D13）の 点灯・消灯を行う

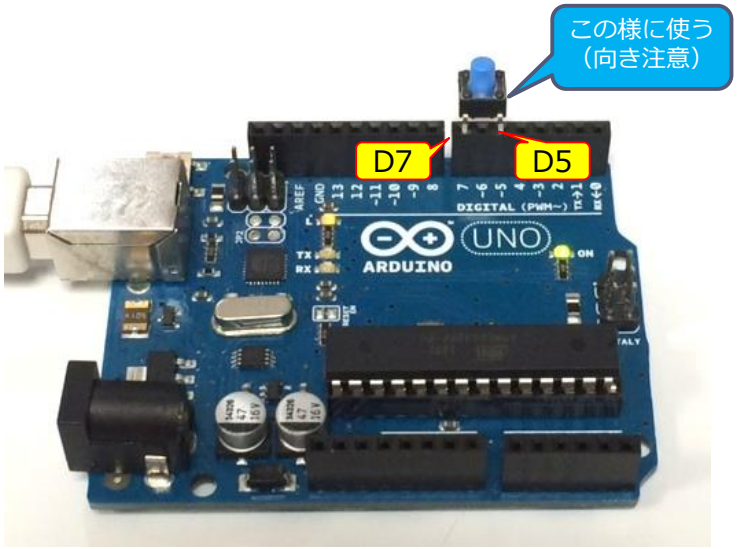
```
 // チルトセンサ
void setup(){
  pinMode(18,OUTPUT);  // A4
  digitalWrite(18,LOW);
  pinMode(19,INPUT_PULLUP); // A5
  pinMode(13,OUTPUT);
}
void loop(){
  digitalWrite(13,!digitalRead(19));
}
```

チルトセンサとLEDのpinMode設定

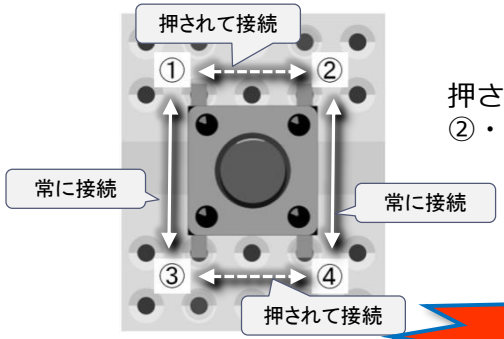
センサ値でLEDを点滅させる

IV-05TLT.ino

6．Arduinoの基本（タクトスイッチ①）



タクトスイッチ
D5,D7



注意
押された時、①・③側と
②・④側に電流が流れる

IoTABシールド
タクトスイッチ：D2

タクトスイッチも「ジャンパワイヤ」と同じ働き

注意点：
① スイッチが入ったとき（押した状態）　LOW （＝0）となり
　スイッチが切れたとき（離れた状態）　HIGH（＝1）となる

② タクトスイッチの片方をGNDにする。

③ pinMode関数で、第２引数に「INPUT_PULLUP」
　（プルアップ抵抗）を利用すること

課題：
　タクトスイッチによってスイッチがOn/Offの状態で
LED（D13）の 点灯・消灯を行う

```
// スイッチD5･D7
// LED D13
void setup(){
  pinMode(5,OUTPUT);
  digitalWrite(5,LOW);
  pinMode(7,INPUT_PULLUP);
  pinMode(13,OUTPUT);
}
void loop(){
  digitalWrite(13,digitalRead(7)?LOW:HIGH);
}
```

タクトスイッチとLEDの
pinMode設定

タクトスイッチが
押された時、LED点灯

IV-06SWT.ino

演習課題：
　タクトスイッチの一方をGND、もう一方をD12に挿し込んだ場合のスケッチはどうなる？

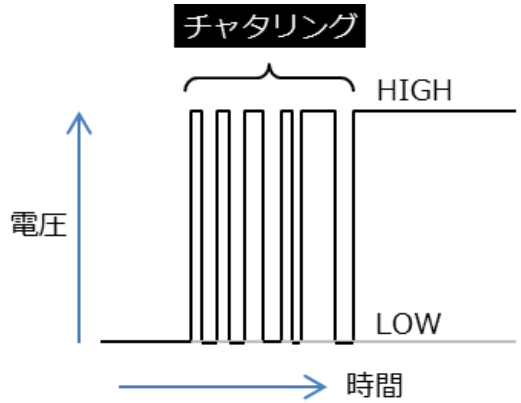
6．Arduinoの基本（タクトスイッチ②）

チャタリング(chattering)について

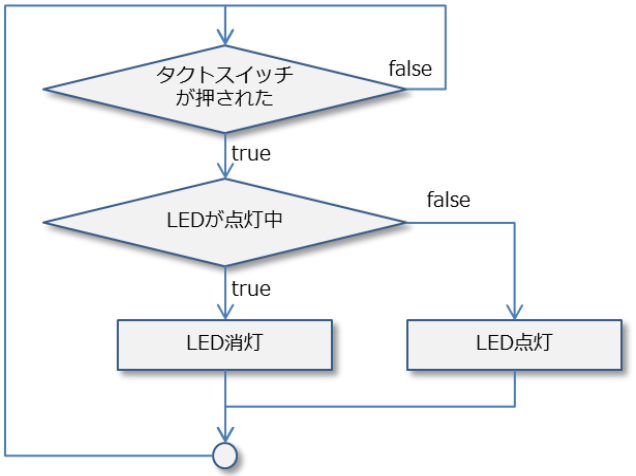
スイッチが押されたときに、　短い時間では、接触不安定な状態となる
これをチャタリングと呼び、その考慮が必要な場合がある。

課題：
　タクトスイッチが押される度に、LEDを点滅を繰り返す

注意：チャタリングを考慮してスケッチを作成する必要がある
　特に、押したときの時間を考慮



タクトスイッチのボタンを離す瞬間



IV-06CTR.ino

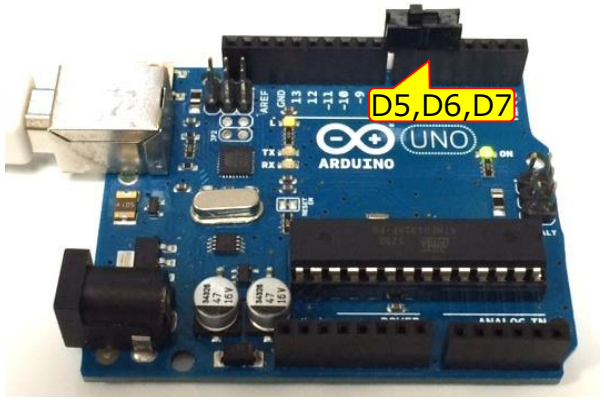
タクトスイッチとLEDのpinMode設定

```
// チャタリング処理(LED点灯・点滅)
void setup(){
  pinMode(5,INPUT_PULLUP);
  pinMode(7,OUTPUT);
  //digitalWrite(7,LOW);
  pinMode(13,OUTPUT);
}
void loop(){
  static boolean sw=HIGH;
  digitalWrite(13,sw);
  while(digitalRead(5));
  delay(****);
  sw=!sw;
}
```

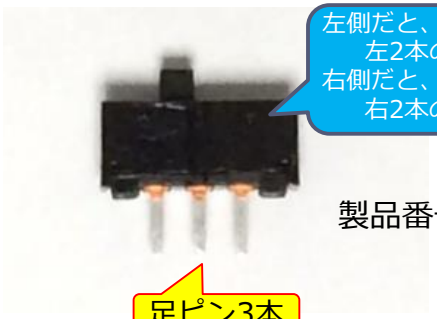
チャタリングを考慮したLED点灯

チャタリング考慮
（時間を変更してみよう）

7．Arduinoの基本（スライドスイッチ）



スライドスイッチ
D5,D6,D7



製品番号：SLB1208

スライドスイッチは、２つの「ジャンパワイヤ」があるようなもの

注意点：
① 両側ピンをGNDにして、中央ピンを「INPUT__PULLUP」にするか

② 中央ピンをGNDにして、両側ピンを「INPUT_PULLUP」にする

<ただ、片方だけの設定でも問題なし>

課題：
スライドスイッチによって
LED（D13）の 点灯・消灯を行う

IV-07SLS.ino

```
// スライドスイッチD5-D7
// LED D13
void setup(){
  pinMode(5,OUTPUT);
  digitalWrite(5,LOW);
  pinMode(6,INPUT_PULLUP);
  pinMode(13,OUTPUT);
}
void loop(){
  digitalWrite(13,digitalRead(6));
}
```

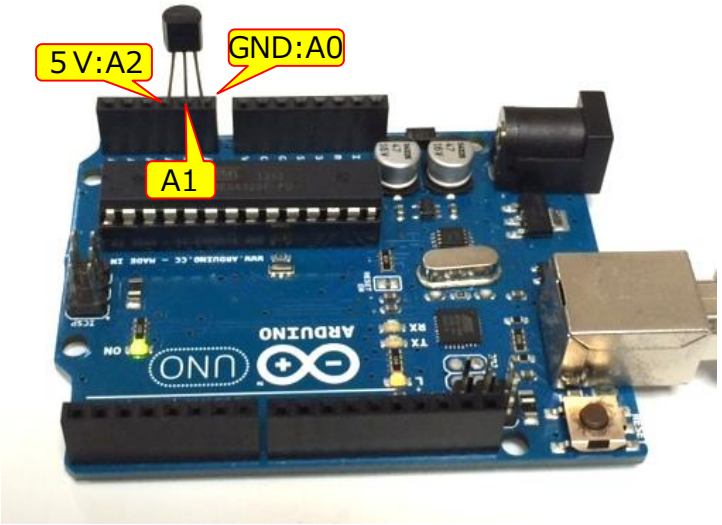
スライドスイッチとLEDのpinMode設定

スライドスイッチとLEDで点滅

演習課題：
右と左のスライドによって、LEDの点滅を変えてみよう

アナログ入力

8． Arduinoの基本（温度センサ）



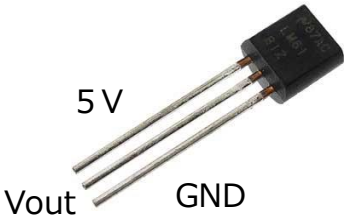
温度センサ（LM61BIZ)
A0(GND),A1(Vout),A2(5V)

温度センサには、アナログ・デジタルと様々存在。
ここでは、安価なアナログ温度センサ（LM61BIZ)を利用

注意点：
① 電源とGNDを間違えないように

② 平らな面を見て、左側ピンをA2（電源）に、右側ピンをA0（GND）に接続

③ 中央ピン（A1接続）から温度（電圧）値を出力⇒ 摂氏温度に変換する式が必要



製品番号：LM61BIZ

課題：
温度センサによる値を、シリアルモニタ画面に表示

演習課題：
温度センサのある閾値をもって、LEDを点灯、消灯してみよう

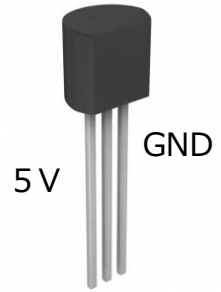
IoTABシールド
温度センサ：A1
（異なる製品）

```
float tmp () {    // 温度センサ値出力関数
  return(analogRead(A1)/1023.0*500 - 60);
}
void setup(){
  pinMode(A2,OUTPUT);  // 温度センサ電源
  digitalWrite(A2,HIGH);
  pinMode(A0,OUTPUT);  // 温度センサGND
  digitalWrite(A0,LOW);
  Serial.begin(9600);
}
void loop() {
  Serial.println(tmp());
  delay(300);
}
```

温度センサの両端ピンをGNDと電源設定
シリアルモニタの初期設定

温度値を0.3秒毎にシリアルモニタ表示

IV-08TMP.ino



アナログ入力

9．Arduinoの基本（光センサ）

■ ポイント

- **アナログ入力電子部品**
  - ① 多くのセンサ類
  - ② 可変抵抗器
- **アナログ入力の接続ポート**
  - ain: A0～A5
- **アナログ入力関数**
  - int val = analogRead(ain);
    ここで 読み込み値 val = 0～1023

■ 事例

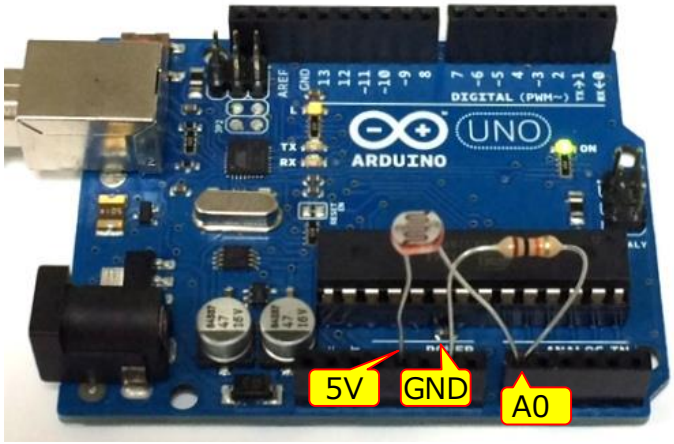
シリアルモニタ画面の初期化（ボーレート設定）

光センサ値をシリアルモニタ画面に表示

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(analogRead(A0));
  delay(100);
}
```

IV-09LGT.ino

■ 配線



1）光センサ（CdS） 5V & A0
2）抵抗（10KΩ）GND & A0

■ 注意点

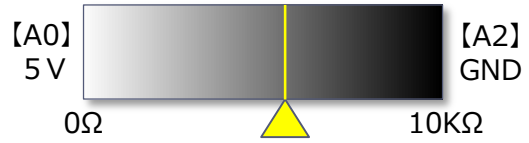
① 抵抗と光センサのピンを同じA0に挿し込む
② シリアルモニタ画面に値を表示出力

IoTABシールド
光センサ：A0

アナログ入力

10. Arduinoの基本（可変抵抗器）

可変抵抗器（10KΩ）
A0,A1,A2
製品番号：TSR-3386U



【A1】の値(a)
0～1023
変換式 a*10000/1023
=a*10/1.023

何故
a*10000/1023
ではだめか

IoTABシールド
可変抵抗：A3

可変抵抗器は、アナログ入力として利用する。

注意点：
① 両端を電源【A0】とGND【A2】に設定

② 中央ピンの値をアナログ入力として
値を取得し、変換

課題：
出力値をシリアルモニタ画面に表示

可変抵抗器のA0,A1,A2のピン設定
シリアルモニタの初期設定

```
void setup(){
  pinMode(A0,OUTPUT);
  pinMode(A2,OUTPUT);
  digitalWrite(A0,HIGH);
  digitalWrite(A2,LOW);
  Serial.begin(9600);
}
void loop(){
  Serial.println((float)analogRead(A1)*10/1.023);
  delay(300);
}
```

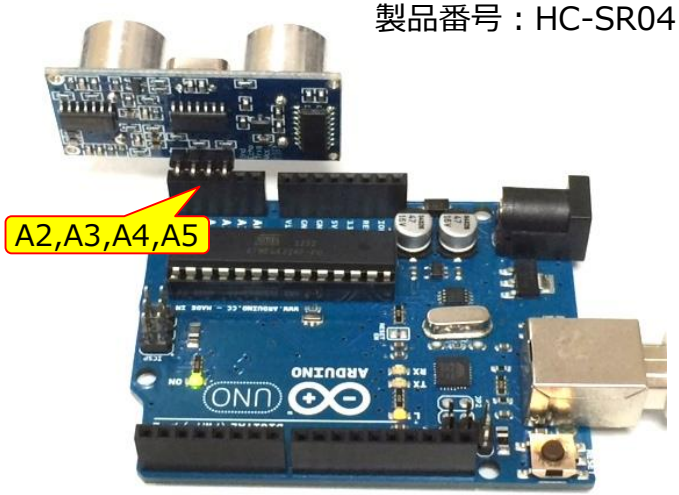
IV-10VRS.ino

0.3秒ごとに可変抵抗器の値を
シリアルモニタに表示

演習課題：
可変抵抗器の値を見て、LEDの明るさを変えてみよう

11. Arduinoの基本（超音波距離センサ①）

製品番号：HC-SR04

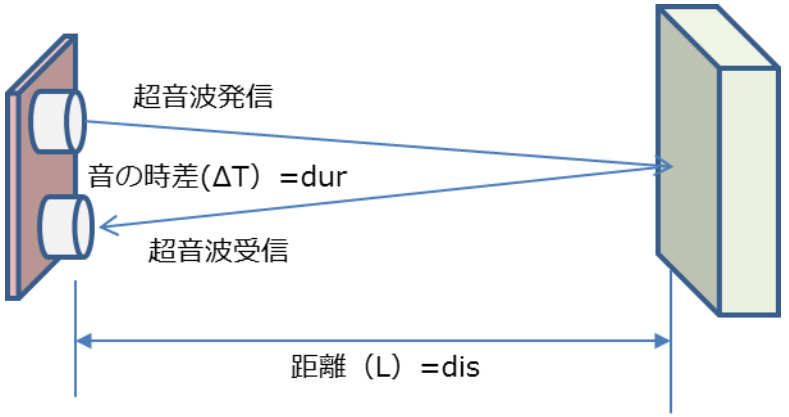


赤外線距離センサ
Vcc(A2) ： 5 V
Trig(A3)：送信トリガー
Echo(A4)：受信エコー
GND(A5)： 0 V

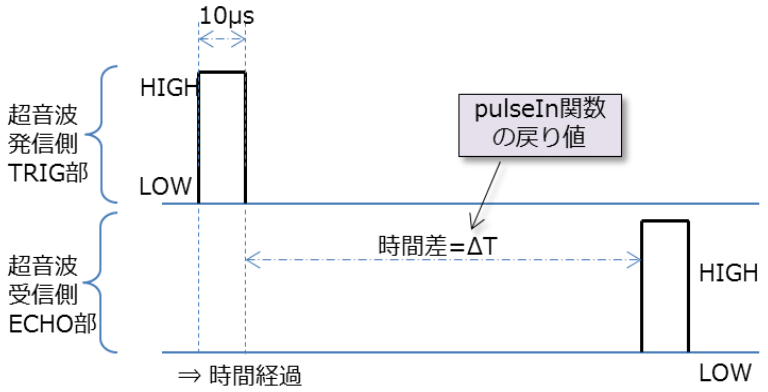
プログラミング上は、超音波のHIGH/LOWの時差を計測して、距離を算出。

この場合、「pulseIn関数」を利用する。

IoTABシールド
超音波センサ：D12-D13

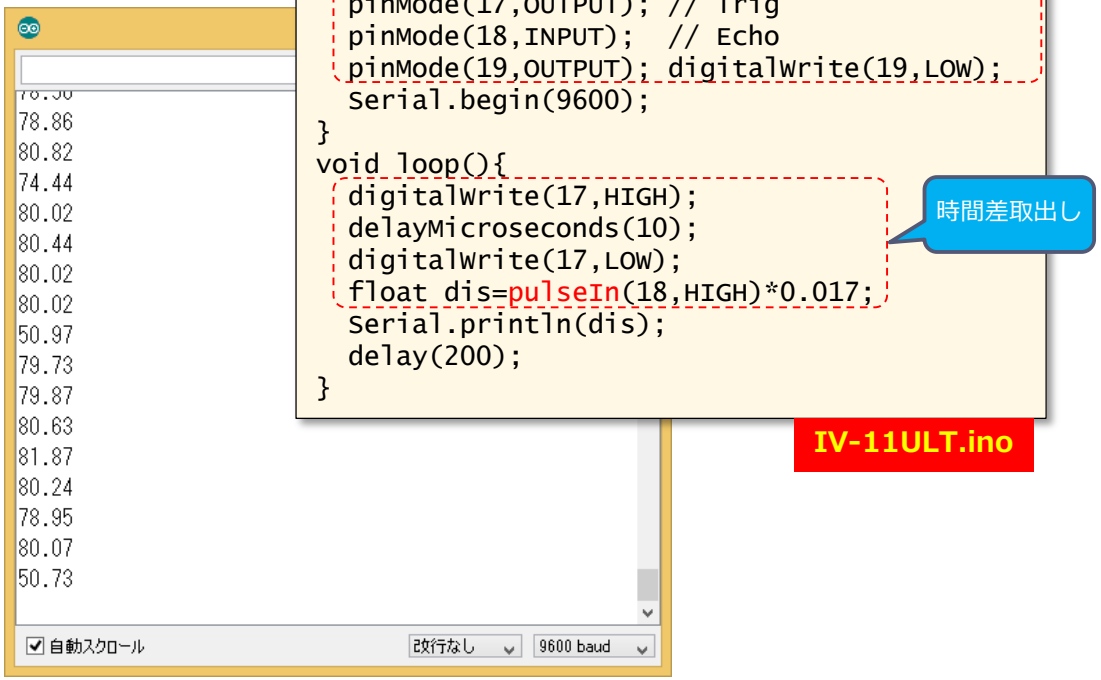


L = C × ΔT／2
ここで、Lは、障害物までの距離
Cは、音速（概算簡易式は 331＋ 0.6× t ：単位m/s）
ΔTは、超音波の発信から受信までの時間差



11. Arduinoの基本（超音波距離センサ②）

pulseIn 関数
unsigned long pulseIn(pin,val,tout);
ここで、 pin: パルスを入力するピン番号
val: 測定するパルスの種類（HIGHまたはLOW）
tout: タイムアウト時間（省略可）
戻り値： パルスの長さ（マイクロ秒）




```
void setup() {
  pinMode(16,OUTPUT); digitalWrite(16,HIGH);
  pinMode(17,OUTPUT); // Trig
  pinMode(18,INPUT);  // Echo
  pinMode(19,OUTPUT); digitalWrite(19,LOW);
  Serial.begin(9600);
}
void loop(){
  digitalWrite(17,HIGH);
  delayMicroseconds(10);
  digitalWrite(17,LOW);
  float dis=pulseIn(18,HIGH)*0.017;
  Serial.println(dis);
  delay(200);
}
```

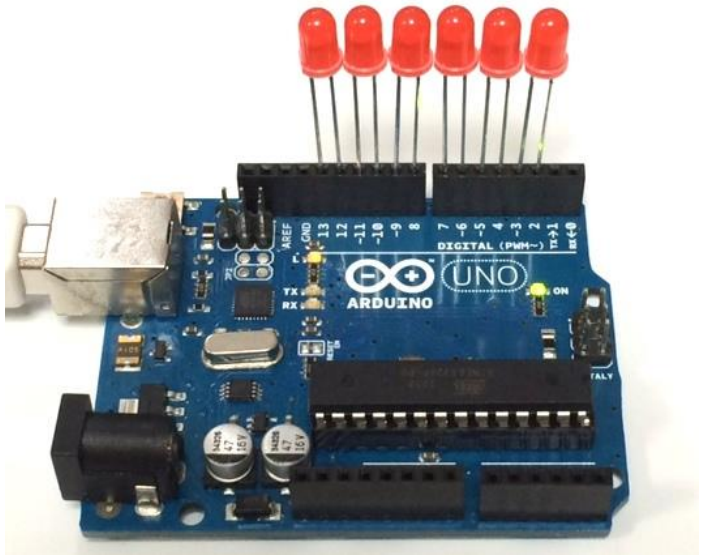
ポート設定

時間差取出し

IV-11ULT.ino

デジタル出力
アナログ出力

12. Arduinoの拡張（複数LED ①）



6個のLED
アノード側
D2,D4,D6,D8,D10,D12
カソード側
D3,D5,D7,D9,D11,D13

※デジタル出力の場合

接続LED	アノード	カソード
LED 1	D2	D3
LED 2	D4	D5
LED3	D6	D7
LED 4	D8	D9
LED 5	D10	D11
LED 6	D12	D13

課題：
1．6個のLEDを順次点滅
2．6個のLEDを右左に点滅

※アナログ出力の場合

※注意：ここでは抵抗付LEDを利用しているため直接挿して利用可能

IoTABシールド
LED：D3-D8

デジタル出力
アナログ出力

１２．Arduinoの拡張：複数LED ②

課題1： 6個のLEDを順次右から左へ、また逆に左から右に点滅させる

回答スケッチ例：

6個のLEDのアノードをデジタル出力（GND）宣言

```
void setup(){
   for(int i=2; i<14; i++) {
     pinMode(i,OUTPUT);
     digitalWrite(i,LOW);
   }
}
void loop() {
   static int i=2, j=2;
   digitalWrite(i,HIGH);
   delay(100);
   digitalWrite(i,LOW);
   if(i==12) {j=-2;}
   else if (i==2) {j=2;};
   i=i+j;
}
```

IV-12LED-01.ino

D02～D12 まで6個のLEDを順次０.1秒ごとに点灯（D12まで行ったらD02に戻る）

課題２： 下のスケッチはどんな動きをするでしょう？

```
void setup(){
   for(int i=2; i<14; i++) {
     pinMode(i,OUTPUT);
     digitalWrite(i,LOW);
   }
}
void loop() {
  for( int i=2; i<14; i=i+2) {
    digitalWrite(i,HIGH);
    delay(50);
   }
   for( int i=12; i>1; i=i-2){
     digitalWrite(i,LOW);
     delay(50);
   }
}
```

IV-12LED-02.ino

課題３： 下のスケッチはどんな動きをするでしょう？

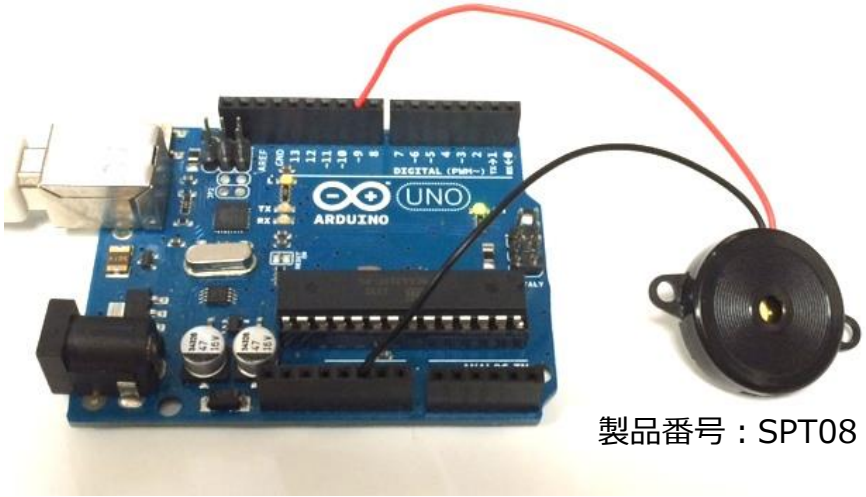
```
void setup(){
   for(int i=2; i<14; i++) {
     pinMode(i,OUTPUT);
     digitalWrite(i,LOW);
   }
}
void loop() {
  for( int i=2; i<14; i=i+2) {
    digitalWrite(i,HIGH);
    delay(50);
   }
   for( int i=12; i>1; i=i-2){
     digitalWrite(i,LOW);
     delay(50);
   }
  for( int i=12; i>1; i=i-2) {
     digitalWrite(i,HIGH);
     delay(50);
  }
  for( int i=2; i<14; i=i+2) {
    digitalWrite(i,LOW);
    delay(50);
  }
}
```

IV-12LED-03.ino

※注意：ここでは抵抗付LEDを利用しているため直接挿して利用可能

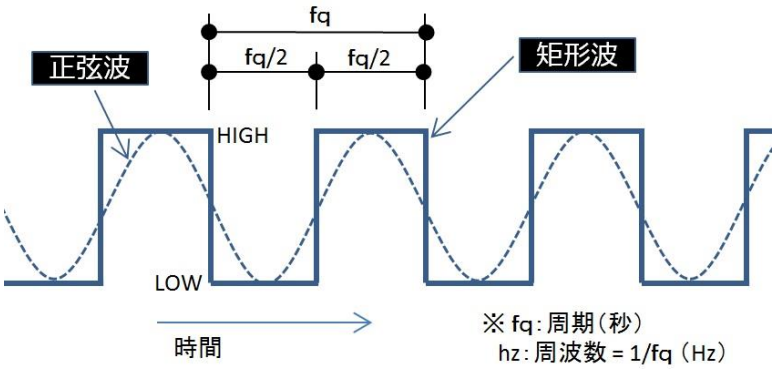
13. Arduinoの拡張（圧電スピーカ ①）

デジタル出力
アナログ出力



製品番号：SPT08

スピーカ
GND ＆ D9



重要： スピーカは、膜の鼓動（振動）で音を発生
スピーカに電源のOn/Offすることで振動を起こす。

Arduinoでは、 ３つの方法で音を出すことが可能

1）digitalWrite関数を使った場合
2）analogWrite関数を使った場合
3）tone関数を使った場合

課題：
1．デジタル出力で音を鳴らす
2．アナログ出力で音を鳴らす
3．tone関数を使って音を鳴らす
4．自らtone関数をつくってみよう

IoTABシールド
スピーカ：D9

13. Arduinoの拡張（圧電スピーカ ②）

デジタル出力
アナログ出力

■課題１：デジタル出力による音発生

```
void setup(){
  pinMode(9,OUTPUT);
}
void loop(){
  digitalWrite(9,HIGH);
  delay(2);
  digitalWrite(9,LOW);
  delay(10);
}
```

pinModeによる初期設定

0.01秒ごとにHIGH/LOWの振動で音を発生

IV-13SPK-01.ino

■課題２：アナログ出力による音発生

```
// アナログ出力によるスピーカ
void setup(){
}
void loop(){
 analogWrite(9,255/2);
  delay(100);
  analogWrite(9,0);
  delay(100);
}
```

0.2秒ごとに0.1秒の音発生

IV-13SPK-02.ino

■課題３：デジタル出力（tone関数利用）

```
void setup(){
  pinMode(9,OUTPUT);
}
void loop()
{
  tone(9,250,500);
  delay(700);
}
```

pinModeによる初期設定

IV-13SPK-03.ino

tone関数を使って
1秒ごとに0.5秒の音発生

音階の周波数

■課題４： tone関数を作ってみよう

音階	3	4	5	6	7	8	9
B	247	494	988	1976	3951	7902	
A#	233	466	932	1865	3729	7459	
A	220	440	880	1760	3520	7040	14080
G#		415	831	1661	3322	6645	13290
G		392	784	1568	3136	6272	12544
F#		370	740	1480	2960	5920	11840
F		349	698	1397	2794	5588	11176
E		330	659	1319	2637	5274	10548
D#		311	622	1245	2489	4978	9956
D		294	587	1175	2349	4699	9397
C#		277	554	1109	2217	4435	8870
C		262	523	1047	2093	4186	8372

※ここで、ド：C、レ：D、ミ：E、ファ：F、ソ：G、ラ：A、シ：Bとなります。

デジタル出力
アナログ出力

13. Arduinoの拡張（圧電スピーカ ③）

課題５：チューリップ（メロディ）を作ってみよう

IoTABS4_melody.ino

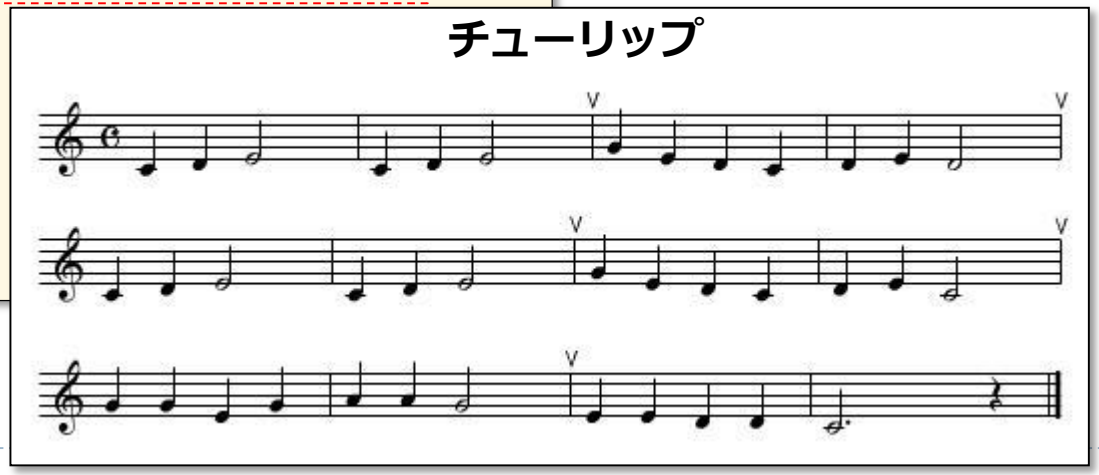
```
#define SPKPIN 10
#define TC 0 // ド
#define TD 1 // レ
#define TE 2 // ミ
#define TF 3 // ファ
#define TG 4
#define TA 5
#define TB 6 // シ
#define TX 7
int fq[] = {262, 294, 330, 349, 392, 440, 494, 523}; // 音階の周波数(Hz)ドレミ･･･
int mo[38][2] = {
  {TC,500},{TD,500},{TE,1000},{TC,500},{TD,500},{TE,1000},
  {TG,500},{TE,500},{TD,500},{TC,500},{TD,500},{TE,500},{TD,1000},
  {TC,500},{TD,500},{TE,1000},{TC,500},{TD,500},{TE,1000},
  {TG,500},{TE,500},{TD,500},{TC,500},{ TD,500},{TE,500},{TC,1000},
  {TG,500},{TG,500},{TE,500},{TG,500},{TA,500},{TA,500},
  {TG,1000},{TE,500},{TE,500},{TD,500},{TD,500},{TC,1000}
};
void setup() {
  for (int i = 0; i < 38; i++) {
    if(mo[i][0] != TX) {
      tone(SPKPIN, fq[mo[i][0]], mo[i][1]);
    }
    delay(mo[i][1]+20);
  }
}
```

ドレミの番号設定

ドレミの音階の定義

チューリップのメロディ

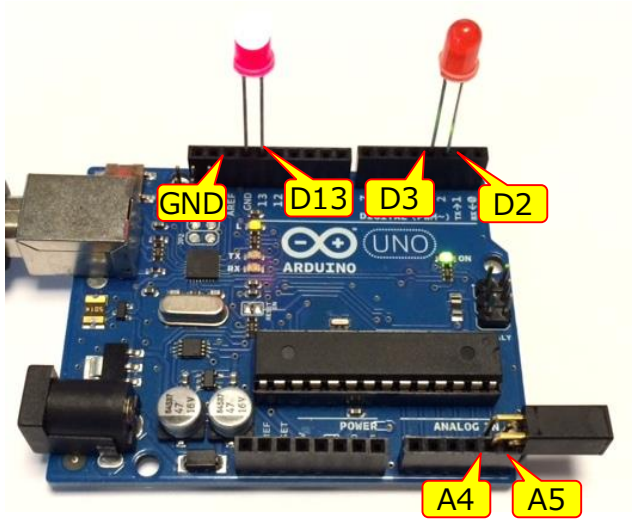
上記の配列moをtone関数を使って実行



TABrain

第5章
Arduino UNO の応用利用

課題１：チルト（傾斜）センサとLED



課題：
チルトセンサの傾きによって、２つのLEDのいずれかを点灯、もう片方を消灯させるスケッチを作成

ヒント：
初期設定は、D2,D3,D13,D4,D5
電源の初期設定：
GNDの初期設定：

IoTABシールド
LED：D3-D8

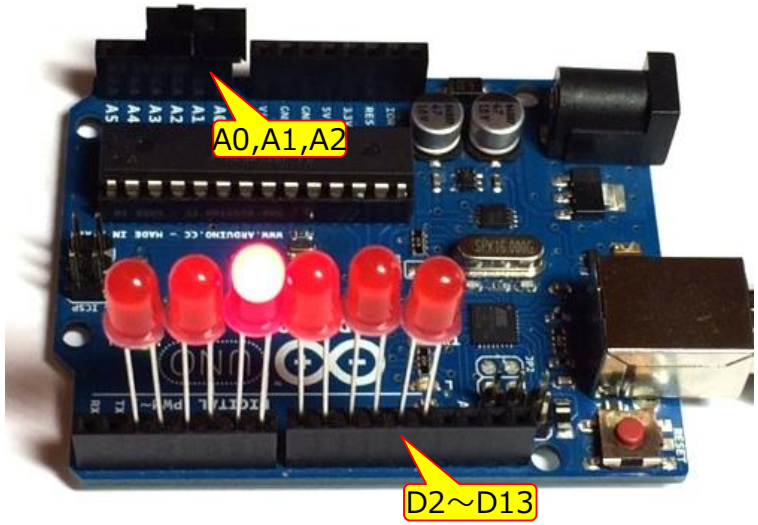
V-01SMP.ino

```
// チルトセンサの傾斜方向でLEDを点滅
void setup(){
 pinMode(18,OUTPUT);
 digitalWrite(18,LOW);
 pinMode(19,INPUT_PULLUP);
 pinMode(13,OUTPUT);
 pinMode(2,OUTPUT);
 pinMode(3,OUTPUT);
 digitalWrite(3,LOW);
}
void loop(){
 digitalWrite(13,digitalRead(19));
 digitalWrite(2,!digitalRead(19));
}
```

1）A4,A5のチルトセンサ初期設定
2）D2－D3のLED初期設定
3）D13のLED初期設定

チルトセンサで、一方のLED点灯、もう一方を消灯

課題２：LED6個とスライドスイッチ



1）6個のLED（D2-D13）
2）スライドスイッチ
（A0,A1,A2）

IoTABシールド
LED：D3-D8

課題：スライドスイッチの方向（右・左）に応じて、
6個のLEDを順次流れるように点滅させる

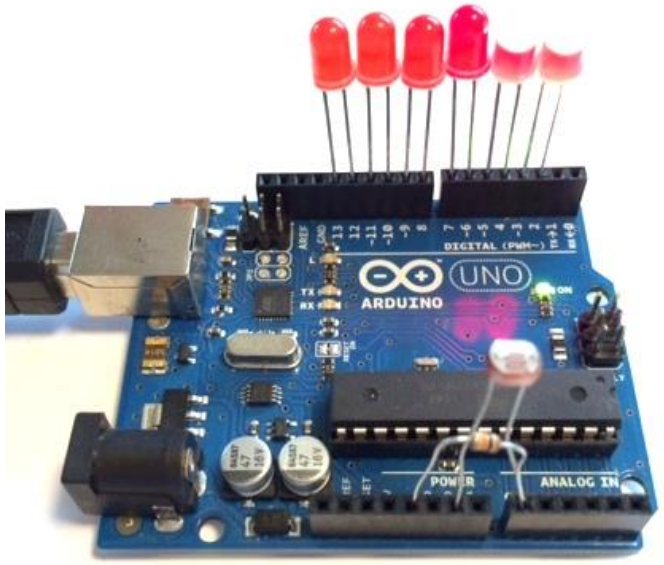
V-02SMP.ino

```
// A0-A2 SW-set
// D2-D13 LED-set
// A4-A5 Speaker
void setup(){
  for(int i=2; i<14; i++) {
    pinMode(i,OUTPUT);
    digitalWrite(i,LOW);
  }
  pinMode(14,OUTPUT);
  pinMode(15,INPUT_PULLUP);
  pinMode(14,OUTPUT); digitalWrite(A4,LOW);
  pinMode(15,OUTPUT);
}
void loop() {
  static int i=2,j;
  j=(digitalRead(15)?-2:2);
  digitalWrite(i,HIGH);
  tone(A5,131*i/2,50);
  delay(100);
  digitalWrite(i,LOW);
  i=i+j;
  if(i<2) i=12;
  else if(i>12) i=2;
}
```

1）6個のLED初期設定
2）スライドスイッチ初期設定

LED点滅が流れるようにするには？

課題３：光センサとLED６個



1）LED D2-D13まで 6個
2）光センサ GND ＆ A0
3）抵抗 5V ＆ A0

IoTABシールド
LED：D3-D8
光センサ：D10

課題：　光センサの値に応じて、6個のLEDを点滅させる
（6段階で、LEDを点灯）

V-03SMP.ino

```
void setup(){
  for(int i=2; i<14; i++) {
    pinMode(i,OUTPUT);
    digitalWrite(i,LOW);
  }
}
void loop(){
  int n = map(analogRead(A0),0,1023,1,6);
  for(int i=0; i<6; i++) {
    digitalWrite(i*2+2,(i<n?HIGH:LOW));
  }
}
```

6個のLED初期設定

光センサの値に
応じてLEDを点灯

課題４：可変抵抗器とスピーカ

1）スピーカ D7　GND
2）可変抵抗器 A0,A1,A2

TABシールド
LED：D3-D8
スピーカ：D9
可変抵抗器：A3

課題：　可変抵抗器のつまみを回すことで、スピーカの音を高音にしたり、低音にしたりする。

V-04SMP.ino

1）スピーカの初期設定
2）可変抵抗器の初期設定

```
void setup(){
 pinMode(7,OUTPUT); // ブザーD7
 pinMode(A0,OUTPUT);  //可変抵抗器
 pinMode(A2,OUTPUT);  //可変抵抗器
 digitalWrite(A0,HIGH); //電源
 digitalWrite(A2,LOW);  // GND
}
void loop(){
   tone(7,analogRead(A1),100);
}
```

抵抗に応じて
高低差を付けた音発生

課題５：LED6個と超音波距離センサ

1）LED D2-D13まで 6個
2）赤外線距離センサ A2,A3,A4,A5

IoTABシールド
LED：D3-D8
距離センサ：D12,D13

課題： 距離センサの値に応じて、6個のLEDを点滅させる

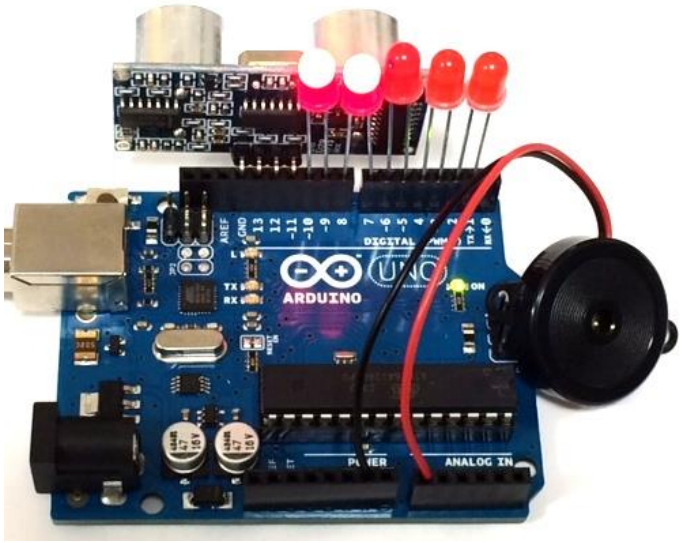
V-05SMP.ino

1）6個のLEDの初期設定
2）超音波距離センサの初期設定

```
void setup() {
  for( int i=2; i<14; i++)
    pinMode(i,OUTPUT);
  pinMode(16,OUTPUT); digitalWrite(16,HIGH);
  pinMode(17,OUTPUT); // Trig
  pinMode(18,INPUT); // Echo
  pinMode(19,OUTPUT); digitalWrite(19,LOW);
  Serial.begin(9600);
}
void loop(){
  digitalWrite(17,HIGH);
  delayMicroseconds(10);
  digitalWrite(17,LOW);
  float dis=pulseIn(18,HIGH)*0.017;
  for(int i=1; i<7; i++) {
    digitalWrite(i*2,dis>i*10?HIGH:LOW);}
}
```

超音波距離センサにより距離を取出し
LEDの点滅を制御する

課題６：LED5個・スピーカ・距離センサ



1）LED D1-D10まで 5個
2）赤外線距離センサ D11,D12,D13,GND
3）スピーカ A0、GND

IoTABシールド
LED：D3-D8
スピーカ：D9
距離センサ：D12,D13

課題：超音波距離センサの値によって、スピーカから音の高低を出してみよう。それと同時に、LEDの点灯個数を変えてみよう。
テルミンのような仕組みを考えてみよう

V-06SMP.ino

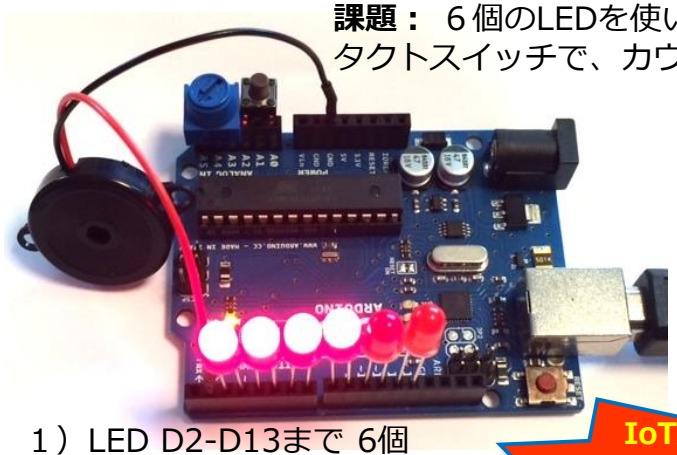
1）D1～D10 までLED初期設定
2）D11－D13＆GNDに
超音波距離センサ初期設定
3）A0にスピーカ設定

```
void setup(){
  for(int i=1; i<11; i++) {
    pinMode(i,OUTPUT); digitalWrite(i,LOW);
  };
  pinMode(11,OUTPUT); digitalWrite(11,HIGH);
  pinMode(12,OUTPUT);
  pinMode(13,INPUT);
  pinMode(A0,OUTPUT);
}
void loop(){
  digitalWrite(12,HIGH);
  delayMicroseconds(10);
  digitalWrite(12,LOW);
  int dis = pulseIn(13,HIGH)*0.017;
  digitalWrite(1,dis<10);
  digitalWrite(3,dis<20);
  digitalWrite(5,dis<30);
  digitalWrite(7,dis<40);
  digitalWrite(9,dis<50);
  if(dis<50) tone(A0,1500 - dis*20,100);
  delay(200);
}
```

1）超音波距離センサの値を取出し
2）10cmから10cm刻みで、音程を変え
3）tone関数で音を出してみよう

課題７：タイマーを作る

課題： ６個のLEDを使い、可変抵抗器によるタイマー時間を設定し、タクトスイッチで、カウントダウンし、時間が来たらブザーを鳴らす。



1）LED D2-D13まで 6個
2）スピーカ D1 GND
3）タクトスイッチ A0、A2
4）可変抵抗器 A3,A4,A5

IoTABシールド
LED：D3-D8
スイッチ：D2
スピーカ：D9
可変抵抗器：A3

1）可変抵抗器の値を読み取りLEDを点灯させる時間は、１分から6分 （変数 n）まで
2）タクトスイッチが押されるタイミング
3）時間の初期設定
4）一旦スタートのブザーを鳴らす
※時間は、unsigned long tm = millis();で開始

```
void loop(){
  byte n;  // タイマー時間(分)
  do{   //可変抵抗器による時間設定(分)
    n = map(analogRead(A4),0,1023,1,6);
    for(int i=1; i<7; i++) {// 設定時間:1～6分
      digitalWrite(i*2,i<n+1?HIGH:LOW);
    }
  } while(digitalRead(16));//設定終了ボタン
  unsigned long tm = millis();
  tone(1,250,500);
  do {  // カウントダウン(LED点滅)
    long ts = millis()-tm;
    digitalWrite(n*2,HIGH);
    delay(1000-ts/60);
    digitalWrite(n*2,LOW);
    delay(ts/60);
    if(millis()-tm>60000){
      tm=millis();  n--;
    }
  }while(n>0);
  digitalWrite(12,HIGH);
  pinMode(1,OUTPUT);
  do {
    tone(1,250,500);
    delay(1000);
  }while(digitalRead(16));
  digitalWrite(12,LOW);
  delay(1000);
}
```

1）時間の経過とともにLEDを1分ごとに消灯
2）ただし、点滅の繰り返しを行うが、1分間のカウントダウンでも点灯の時間と消灯の時間を変化させていく

1）時間が来たことで、LED（D12－D13）を点灯
2）ブザーを鳴らす
3）タクトスイッチが押されるまで１）に戻る
4）LED（D12-D13)消灯

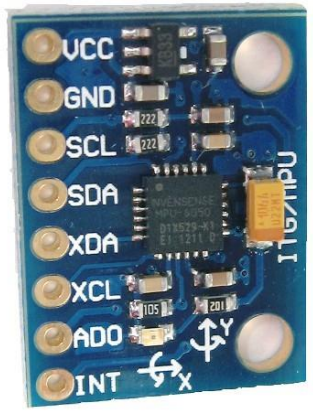
V-07SMP.ino

```
void setup() {
  for(int i=2;i<14; i++){  // LED6個の初期設定
    pinMode(i,OUTPUT);  digitalWrite(i,LOW);
  }
  pinMode(1,OUTPUT);          //スピーカ初期設定
  pinMode(14,OUTPUT);         //タクトスイッチ
  digitalWrite(14,LOW);       //タクトスイッチ(GND)
  pinMode(16,INPUT_PULLUP);   //タクトスイッチ(入力)
  pinMode(17,OUTPUT);         //可変抵抗器
  digitalWrite(17,LOW);       //可変抵抗器(GND)
  pinMode(19,OUTPUT);         //可変抵抗器
  digitalWrite(19,HIGH);      //可変抵抗器(電源)
}
```

1）6個のLEDの初期設定
2）スピーカの初期設定
3）タクトスイッチの初期設定
4）可変抵抗器の初期設定

課題８：慣性センサを使う

GY-521 は、6軸加速度センサと呼ばれ、3軸加速度・3軸ジャイロ・温度センサを持つブレイクアウトボード となる。

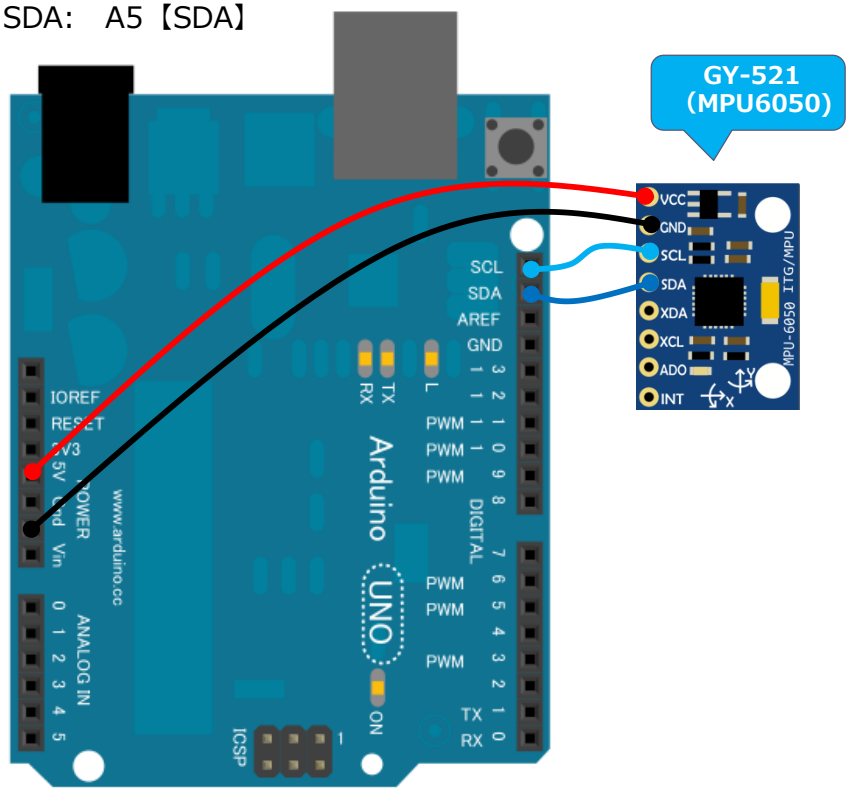


【参考サイト】
http://playground.arduino.cc/Main/MPU-6050
https://www.switch-science.com/catalog/1208/

TABシールドには
付属していません

GY-521 は、シリアル通信I2Cによって、3軸加速度・3軸ジャイロ・温度の値を取り出すことができる。

【配線】Arduinoと接続するピンは、以下の４つ
VCC： 5V
GND: GND
SCL： A4【SCL】
SDA: A5【SDA】



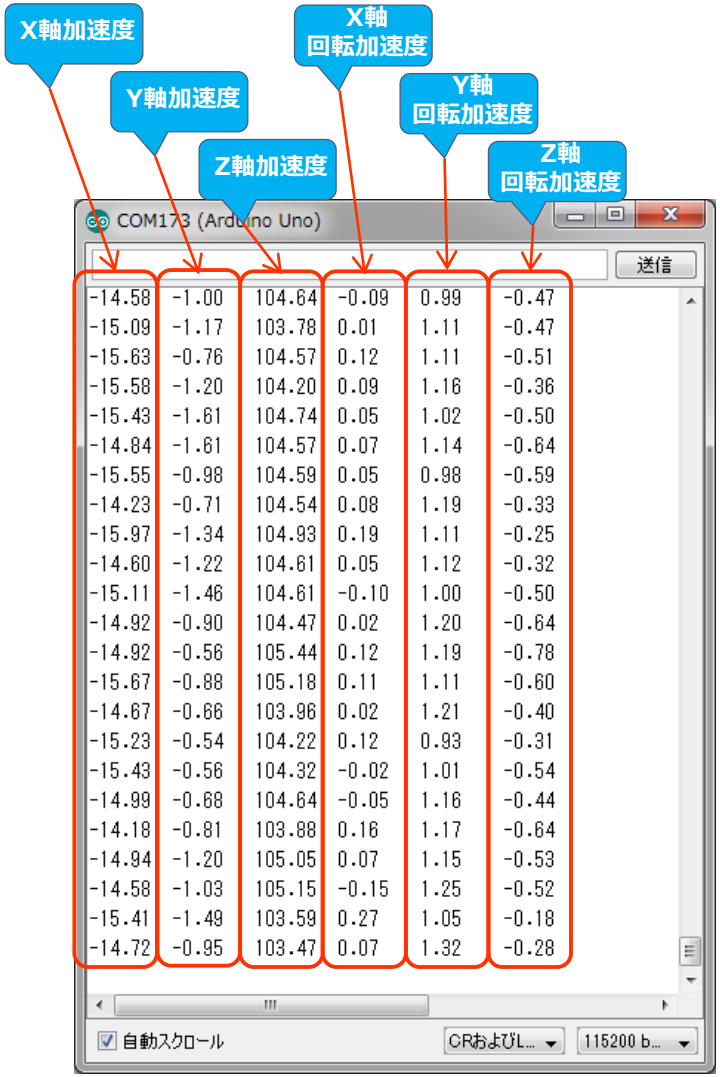
課題８：慣性センサを使う

【課題】ここでは、GY-521を繋いでシリアルモニタ画面で
センサ値を表示させてみるスケッチを作成してみよう

以下のセンサ値は、間にタブ「¥t」付でシリアルモニタ表示

加速度			ジャイロ		
x	y	z	x	y	z

のように表示させてみる。（右図参照）



課題８：慣性センサを使う

※ヘッダー部分

```
// GY-521(MPU-6050) & LCD for IoTABShield V4.0
#include <Wire.h>
#define DTM 10  // Freeqence time (msec)

#define MPU6050_ACCEL_XOUT_H       0x3B   // R
#define MPU6050_WHO_AM_I           0x75   // R
#define MPU6050_PWR_MGMT_1         0x6B   // R/W
#define MPU6050_I2C_ADDRESS        0x68

typedef union accel_t_gyro_union{
  struct{
    uint8_t x_accel_h;
    uint8_t x_accel_l;
    uint8_t y_accel_h;
    uint8_t y_accel_l;
    uint8_t z_accel_h;
    uint8_t z_accel_l;
    uint8_t t_h;
    uint8_t t_l;
    uint8_t x_gyro_h;
    uint8_t x_gyro_l;
    uint8_t y_gyro_h;
    uint8_t y_gyro_l;
    uint8_t z_gyro_h;
    uint8_t z_gyro_l;
  }
  reg;
  struct{
    int16_t x_accel;
    int16_t y_accel;
    int16_t z_accel;
    int16_t temperature;
    int16_t x_gyro;
    int16_t y_gyro;
    int16_t z_gyro;
  }
  value;
};
```

setup関数

```
void setup(){
  Wire.begin();
  Serial.begin(115200);
  int error;
  uint8_t c;
  error = MPU6050_read (MPU6050_WHO_AM_I, &c, 1);
  error = MPU6050_read (MPU6050_PWR_MGMT_1, &c, 1);
  MPU6050_write_reg (MPU6050_PWR_MGMT_1, 0);
}
```

V-08SMP.ino

MPU6050_read関数

```
// MPU6050_read
int MPU6050_read(int start, uint8_t *buffer, int size){
  int i, n, error;
  Wire.beginTransmission(MPU6050_I2C_ADDRESS);
  n = Wire.write(start);
  if (n != 1)
    return (-10);
  n = Wire.endTransmission(false);    // hold the I2C-bus
  if (n != 0)
    return (n);
  // Third parameter is true: relase I2C-bus after data is read.
  Wire.requestFrom(MPU6050_I2C_ADDRESS, size, true);
  i = 0;
  while(Wire.available() && i<size){
    buffer[i++]=Wire.read();
  }
  if ( i != size)
    return (-11);
  return (0);  // return : no error
}
```

MPU6050_wrie関数

```
// MPU6050_write
int MPU6050_write(int start, const uint8_t *pData, int size){
  int n, error;
  Wire.beginTransmission(MPU6050_I2C_ADDRESS);
  n = Wire.write(start);        // write the start address
  if (n != 1)
    return (-20);
  n = Wire.write(pData, size);  // write data bytes
  if (n != size)
    return (-21);
  error = Wire.endTransmission(true); // release the I2C-bus
  if (error != 0)
    return (error);
  return (0);         // return : no error
}
```

MPU6050_wrie_reg関数

```
// MPU6050_write_reg
int MPU6050_write_reg(int reg, uint8_t data){
  int error;
  error = MPU6050_write(reg, &data, 1);
  return (error);
}
```

課題８：慣性センサを使う

loop関数

```
void loop(){
  int error;
  float dT;
  accel_t_gyro_union accel_t_gyro;
  error = MPU6050_read (MPU6050_ACCEL_XOUT_H, (uint8_t *) &accel_t_gyro,
sizeof(accel_t_gyro));

  uint8_t swap;
#define SWAP(x,y) swap = x; x = y; y = swap
  SWAP (accel_t_gyro.reg.x_accel_h, accel_t_gyro.reg.x_accel_l);
  SWAP (accel_t_gyro.reg.y_accel_h, accel_t_gyro.reg.y_accel_l);
  SWAP (accel_t_gyro.reg.z_accel_h, accel_t_gyro.reg.z_accel_l);
  SWAP (accel_t_gyro.reg.t_h, accel_t_gyro.reg.t_l);
  SWAP (accel_t_gyro.reg.x_gyro_h, accel_t_gyro.reg.x_gyro_l);
  SWAP (accel_t_gyro.reg.y_gyro_h, accel_t_gyro.reg.y_gyro_l);
  SWAP (accel_t_gyro.reg.z_gyro_h, accel_t_gyro.reg.z_gyro_l);

  dT = ( (float) accel_t_gyro.value.temperature + 12412.0) / 340.0;
 float acc_x = accel_t_gyro.value.x_accel / 16384.0; //FS_SEL_0 16,384 LSB / g
  float acc_y = accel_t_gyro.value.y_accel / 16384.0;
  float acc_z = accel_t_gyro.value.z_accel / 16384.0;

  char pr[8];

  float acc_angle_x = atan2(acc_x, acc_z) * 360 / 2.0 / PI;
  float acc_angle_y = atan2(acc_y, acc_z) * 360 / 2.0 / PI;
  float acc_angle_z = atan2(acc_x, acc_y) * 360 / 2.0 / PI;

  float gyro_x = accel_t_gyro.value.x_gyro / 131.0;  //FS_SEL_0 131 LSB / (°/s)
  float gyro_y = accel_t_gyro.value.y_gyro / 131.0;
  float gyro_z = accel_t_gyro.value.z_gyro / 131.0;
```

```
  Serial.print(acc_x*100, 2);
  Serial.print("¥t");
  Serial.print(acc_y*100, 2);
  Serial.print("¥t");
  Serial.print(acc_z*100, 2);
  Serial.print("¥t");
  Serial.print(gyro_x, 2);
  Serial.print("¥t");
  Serial.print(gyro_y, 2);
  Serial.print("¥t");
  Serial.print(gyro_z, 2);
  Serial.print("¥t¥r¥n");

 while(millis()-tm>DTM);
  tm=millis();
}
```

この部分がシリアルモニタ画面表示

課題９．I2C_LCDを使う

秋月電商で販売しているI2C_LCD（AQM0802A-RN-GBWを使って光センサ値を表示させてみよう。

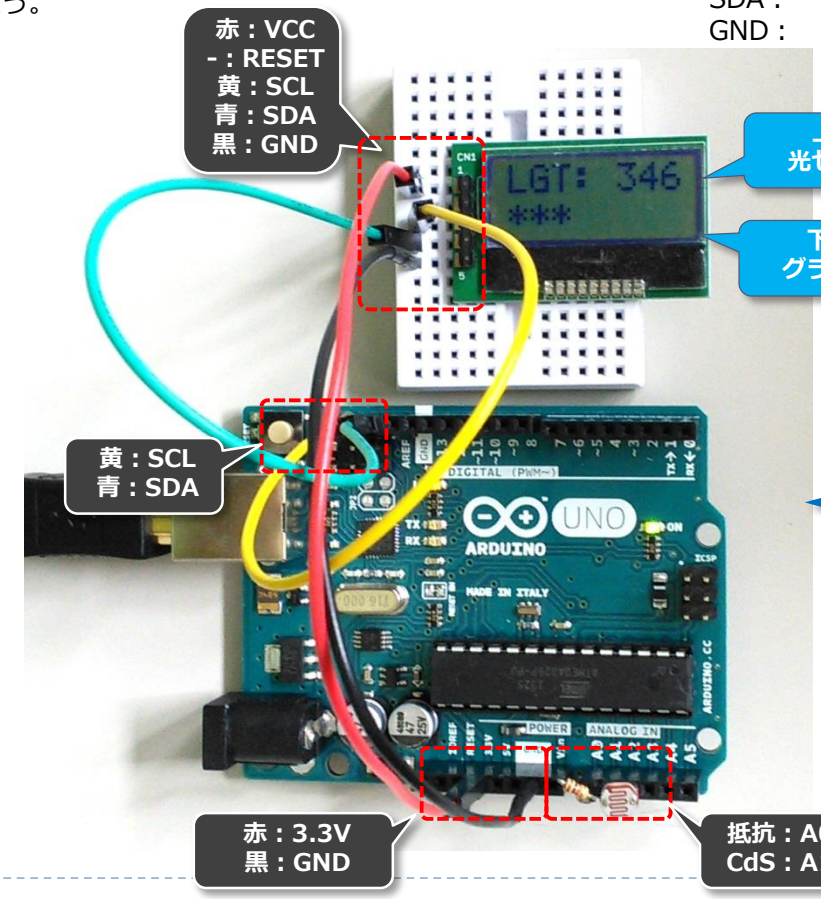
【課題】 光センサ（CdS）をArduinoのA0、A1、A2に直接挿しこんで、その値を表示させると同時に、文字表示で明るさの変化を8文字グラフで表示してみよう。

明るい
＊＊＊＊＊＊＊＊
＊＊＊＊＊＊＊
＊＊＊＊＊＊
＊＊＊＊＊
＊＊＊＊
＊＊＊
＊＊
＊

暗い

【配線】

LCD配線		**CdS（光センサ）**
VCC：	3.3V	A1
RESET:	-	A2
SCL:	SCL（A5）	**１KΩ抵抗**
SDA：	SDA（A4）	A0
GND：	GND	A1



IoTABシールド
光センサ：A0
LCD：I2C

課題９．I2C_LCDを使う

サンプルスケッチ

```
#include <Wire.h>

#define tempPin 0
#define lgtPin 0


void setup() {
  pinMode(A0,OUTPUT);digitalWrite(A0,LOW);
  pinMode(A2,OUTPUT);digitalWrite(A2,HIGH);
  Wire.begin();
  i2c_powerup();
  delay(100);
  lcd_init();
}
void loop(){
 char pr[9];
 uint8_t val = i2c_readlux();
 sprintf(pr,"LGT:%4d",val);
 lcd_setCursor(0,0);
 lcd_printStr(pr);
 String st;
 if(val>800) st ="********";
 else if(val>700) st = "******* ";
 else if(val>600) st = "******  ";
 else if(val>500) st = "*****   ";
 else if(val>400) st = "****    ";
 else if(val>300) st = "***     ";
 else if(val>200) st = "**      ";
 else if(val>100) st = "*       ";
 else st = "        ";
 delay(100);
 char pr0[9];
 st.toCharArray(pr0,8);
 lcd_setCursor(0,1);
 lcd_printStr(pr0);
 delay(100);
}
```

A0：GND
A2: 5 V電圧

上段：１行目
センサ値表示

下段：2行目
グラフ表示

IoTABS4_BH1780_LCD.ino

ここでは、BH1780.inoと
I2C_LCD.inoを利用

第６章
IoTABシールド入力部品

1．入力電子部品について

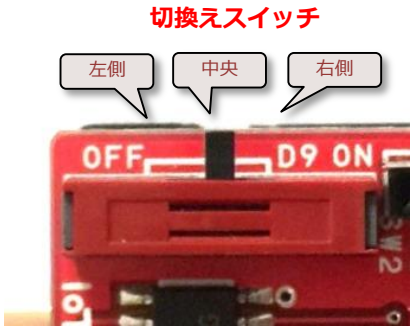
ここでは、アナログ切換えスイッチとデジタル切換えスイッチ、デジタル拡張専用入力ポートの説明

IoTABシールドに装備された入力電子部品は以下のとおりです。

■アナログ電子部品

※ NC：接続なし

アナログ入力ポート	切換えスイッチ	
	左側	右側
A2	NC	音センサ（マイク）
A3	NC	可変抵抗器



切換えスイッチ（重要）

右側：IoTABシールドの電子部品が全て利用可

中央：D9（HIGH）で利用可、D9（LOW）でD10-D13,A0-A3が利用不可

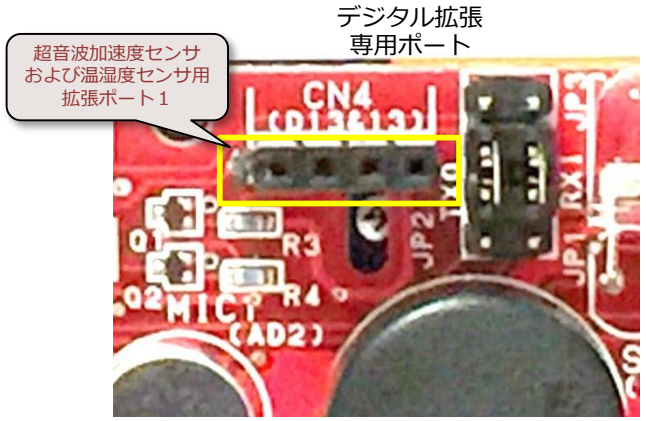
左側：IoTABシールドのD10-D13、A0-A3電子部品が利用不可

※D9 の切り替えは**P＊＊参照**

■デジタル電子部品

デジタル入力ポート	接続電子部品
D2	タクトスイッチ
D11	赤外線受信リモコン
D12	超音波距離センサ(Echo)
D13	超音波距離センサ(Trig)
I2C（0x1C)	加速度センサ
I2C（0x29）	照度センサ
I2C（0x4A)	温度センサ

※ I2Cのカッコ内数字は、アドレス



このデジタル拡張専用入力ポートには、直接、超音波距離センサや湿度センサなどを指すことができるようにしました。

（注意）
デジタル切換えスイッチ
D8はONで利用します

2．タクトスイッチを使う

デジタル

- 注意：一般にタクトスイッチを利用する場合には、プルアップ抵抗（もしくはプルダウン抵抗）を考慮する必要がある。
  ⇒　考慮しないと、デジタルセンサ値が不安定になる。
  IoTABシールドのタクトスイッチは、プルアップ抵抗が取り付けています。

```
pinMode(2,INPUT_PULLUP);
boolean sw = digitalRead(2);
```

■ **スイッチ**

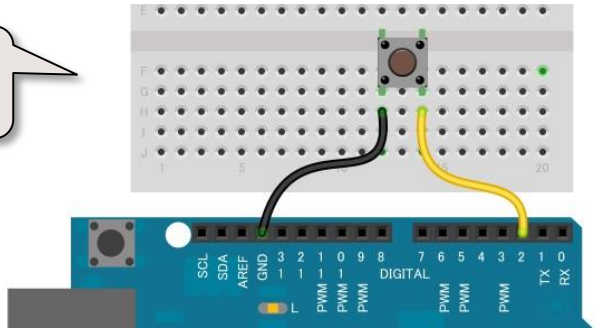
項　目	説　明
製品名	タクトスイッチ

I/O	**デジタル出力（D2）**
	pinMode(2,INPUT_PULLUP);
	sw = digitalRead(2);

備　考	swの値（注意） HIGH: スイッチOFF LOW: スイッチON

ポイントは、pinModeの2番目の引数が、「INPUT_PULLUP」になること

右の事例は、単独でタクトスイッチを付けた事例



スイッチが押されたとき、sw=LOWに
それ以外は、sw=HIGHに

```
void setup(){
  Serial.begin(9600);
  pinMode(2,INPUT_PULLUP);
}
void loop() {
  boolean sw = digitalRead(2);
  delay(100);
  Serial.println(sw?"Off":"On");
}
```

IoTABS4_SWITCH.ino

タクトスイッチの位置

```
pinMode(2,INPUT_PULLUP);
boolean sw = digitalRead(2);
```

ケーブルが繋がれたとき、sw＝LOWに
離れたときは、sw=HIGHになる

TABrain

（注意）
アナログ切換えスイッチ
は**左側**で利用します

アナログ

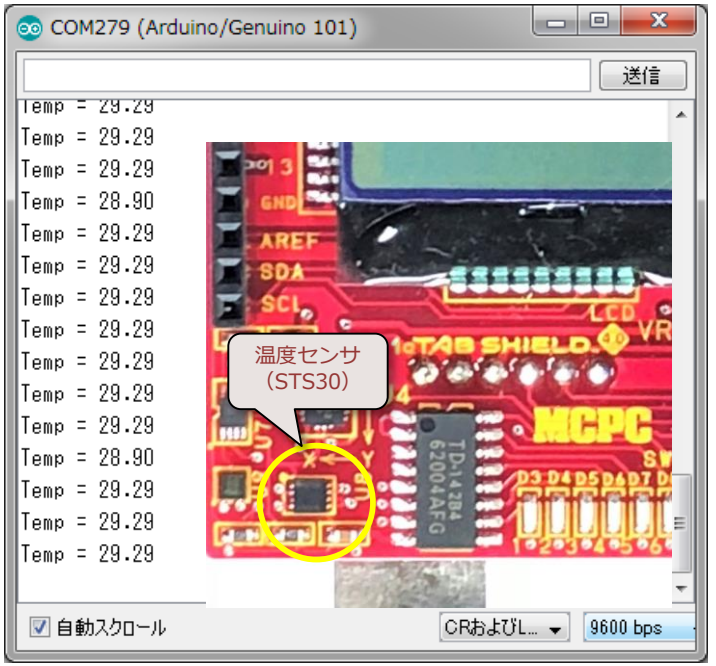
3．温度センサを使う

■ 温度センサ

項　目	説　明
製品名	STS30 メーカ（）
I/O	**シリアル通信（I2C）** （Address＝0x4A
備　考	（-30℃ ～ +100℃）

アナログ値から温度にする変換式がポイント

■ シリアルモニタ画面



■ サンプルスケッチ

```
//サンプルスケッチ
#include <Wire.h>
#include <STSSensor.h>
#define STS30addr 0x4A
STSSensor sts30( Wire );
void setup()
{
  Wire.begin();
  Serial.begin(9600);
  sts30.init(Wire) ;
}
void loop()
{
   Serial.print("Temp = ");
   Serial.println(sts30.getTemperature());
   delay(1000);
}
```

■初期宣言：
#include <STSSensor.h>
#define STS30addr 0x4A
STSSensor sts30(Wire);
■設定
■ループ

IoTABS4_TEMP.ino

■ 補足

1）ここで紹介する温度は、空気中などの温度を測定するもので液体や固体などを直接測定することはできません。
2）待機時間（delay関数）を入れて使うようにしてください。

■ 注意点

Arduino(AVR)のA/D変換の特性により、電源供給と温度センサの計測で、トラブルが発生する場合がある。対策として、一度「空読み」をすると正しくセンサー値を取り出すことができる。
参照：http://homepage3.nifty.com/sudamiyako/zk/AVR_ADC/AVR_ADC.html

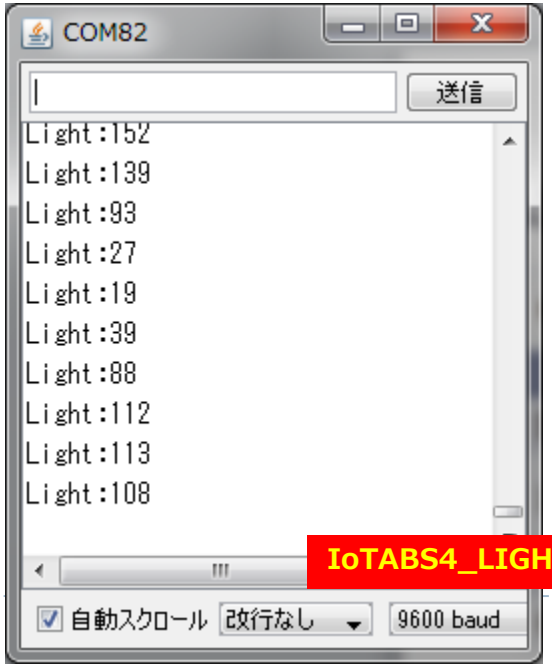
4．照度（光）センサを使う

シリアル（I2C）

■ 照度センサ

項　目	説　明
製品名	BH1780
I/O	**シリアル通信** (0x29)
変換式	特に利用しない
備　考	明るい・暗いの目安値

■ シリアルモニタ画面

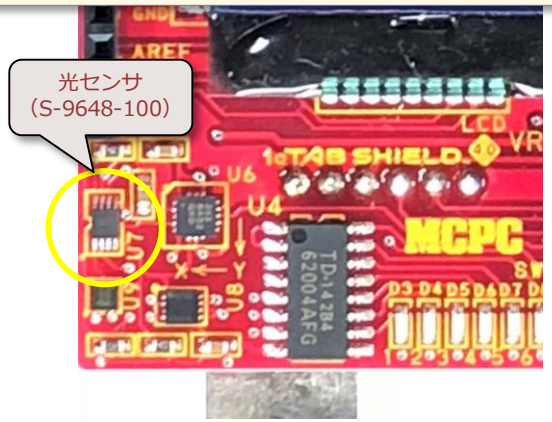


IoTABS4_LIGHT.ino

■ サンプルスケッチ

```
#include <Wire.h>
void setup() {
  Serial.begin(9600);
  Wire.begin();                // join i2c bus with address #8
  delay(10);
  Serial.println("start");
  i2c_powerup();
  delay(100);
  lcd_init();
}

void loop() {
  lcd_setCursor(0,0);
  char pr[9];
  sprintf(pr,"%6dLx",i2c_readlux());
  lcd_printStr(pr);
  Serial.println("Light: " + String(i2c_readlux()) + " lux");
}
```

明るくなると 値が大きくなり、
暗くなると 値が小さいくなる



TABrain

（注意）
アナログ切換えスイッチ
は**左側**で利用します

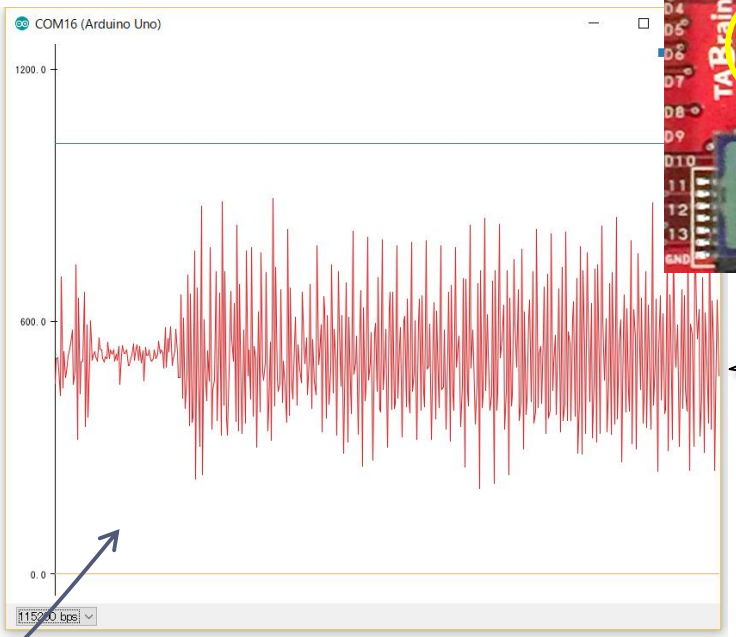
5．音センサ（マイク）を使う

アナログ

■ 音センサ（マイク）

項　目	説　明
製品名	C9767BB422LFP メーカ（DB）
I/O	**アナログ入力（A2）** mic = analogRead(A2);
変換式	特に利用しない
備　考	音の大小の目安値

■ シリアルプロッタ画面



512の値を中心
した波形となる

■ サンプルスケッチ

```
//サンプルスケッチ
#define MicPin A2

void setup () {
 Serial.begin(115200);
}

void loop() {
  Serial.println("1023¥t0¥t" + String(analogRead(MicPin)));
}
```

1023と0に線を入れる

IoTABS4_MIC.ino

■ 補足

1）音は、波形として出力されるので、平均化する必要がある。
　　場合によっては、最初の数秒間などで平均値を調べ、その平均値との差を出して波形を捉えることもできる。
2）雑音のあるところでは、注意が必要となる。

TABrain

（注意）
アナログ切換えスイッチ
は**左側**で利用します

アナログ

6．可変抵抗器を使う

■ 可変抵抗器

項　目	説　明
製品名	3386K-EY5-103TR (メーカ：SUNTAN)
I/O	**アナログ入力（A3）** reg= analogRead(A3); 0 ≦ reg ≦ 1023
変換式	特に利用しない
備　考	抵抗値は、０～10KΩ

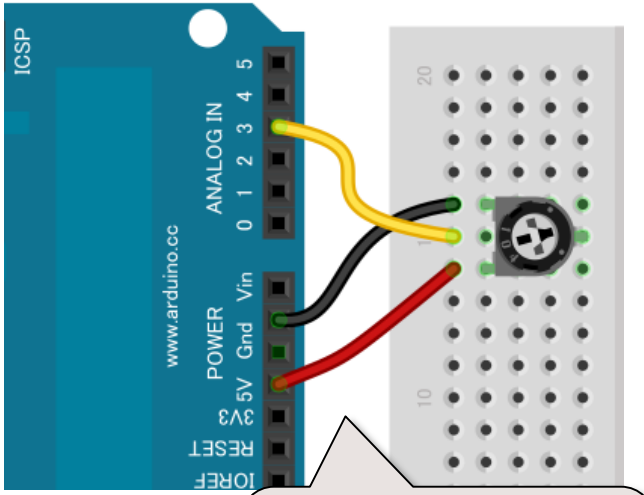
■ サンプルスケッチ

IoTABS4_REG.ino

```
//サンプルスケッチ
#define RegPin A3

void setup () {
 Serial.begin(9600);
}

void loop() {
  int reg= analogRead(MicPin);
  char pr[10];
  sprintf(pr,“Reg:%d",map(reg,0,1023,0,10000));
  Serial.println(pr);
  delay(100);
}
```

抵抗値の出し方は、map関数を使って、出力値の０～1023を、０～10KΩに変換している

■ 単独での事例



この事例は、単独に可変抵抗器を使った場合の接続例。注意すべきことは、中央のピンがアナログ入力ポートに接続すること。両端が、GNDと電源となる。

可変抵抗器は様々な切換で活用可能

7．超音波距離センサを使う

デジタル拡張
専用入力ポートを使います

デジタル

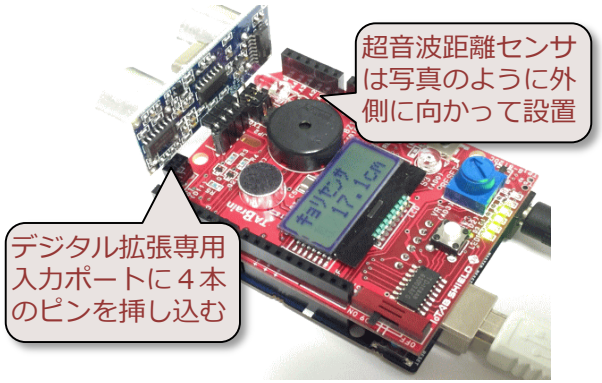
■ 超音波距離センサ

項　目	説　明
製品名	HC-SR04 (メーカ：多数)

I/O	**デジタル入出力（D12,D13）**
	pinMode(13,OUTPUT);
	Trig= digitalWrite(13,HL);
	pinMode(12,INPUT);
	Echo=pulseIn(12,HL);

変換式	dis=(float)Echo*0.017;
備　考	

■ 接続方法

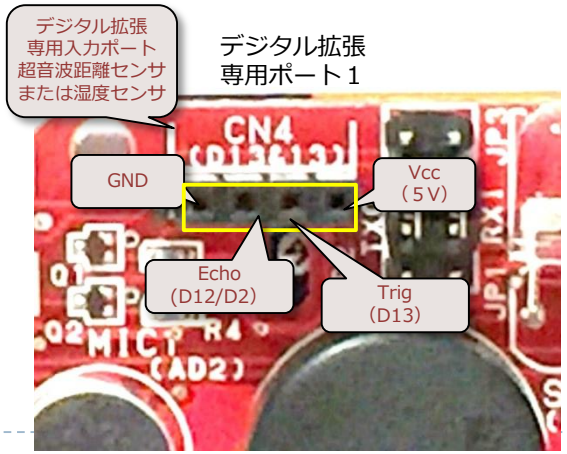


■ サンプルスケッチ

IoTABS4_DIST.ino

```
#define TrigPin 13
#define EchoPin 12
void setup(){
   pinMode(TrigPin,OUTPUT);
   pinMode(EchoPin,INPUT);
   Serial.begin(9600);
}
void loop() {
   digitalWrite(TrigPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(TrigPin, LOW);
   float dist =(float)pulseIn(EchoPin,HIGH)*0.017;
   Serial.print(" DIst = ");
   Serial.println(dist);
   delay(100);
}
```

超音波の速度を340m/secで計算した換算式



TABrain

8．加速度センサを使う

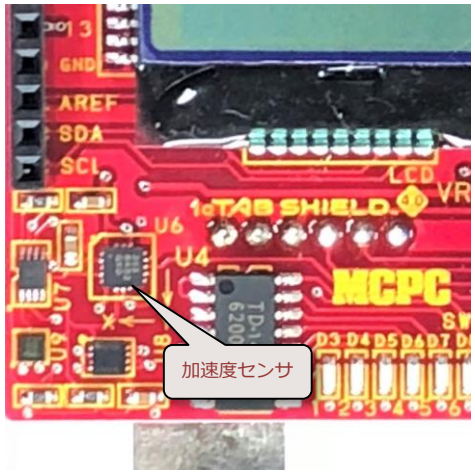
デジタル拡張
専用入力ポートを使います

デジタル

■ 加速度センサ

項　目	説　明
製品名	MMA8452Q (メーカ：)
I/O	**I2C （0x1C）**
利用方法	ライブラリ利用
単位	G
備　考	２G、４G、８G 利用可

■ 加速度センサの位置



■ サンプルスケッチ

IoTABS4_ACC.ino

```
#define TrigPin 13
#define EchoPin 12
void setup(){
   pinMode(TrigPin,OUTPUT);
   pinMode(EchoPin,INPUT);
   Serial.begin(9600);
}
void loop() {
   digitalWrite(TrigPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(TrigPin, LOW);
   float dist =(float)pulseIn(EchoPin,HIGH)*0.017;
   Serial.print(" DIst = ");
   Serial.println(dist);
   delay(100);
}
```

超音波の速度を340m/secで計算した換算式

ここでの加速度センサ部品 MMA8452Qは、12ビット対応で、０～4095までの精度となっています。

9．温湿度センサを使う（オプション）

デジタル

■ **温湿度センサ（DTH11又はDTH22）**

項　目	説　明
製品名	DTH11またはDTH22（AM2302） (メーカ：

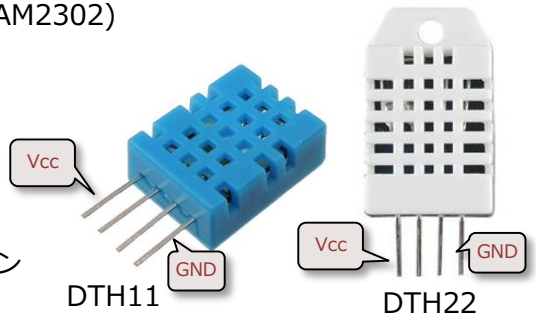
I/O	**デジタル拡張入出力（D12,D13）**
	pinMode(10,INPUT_PULLUP);
	sw=digitalRead(10);

変換式	なし
備　考	

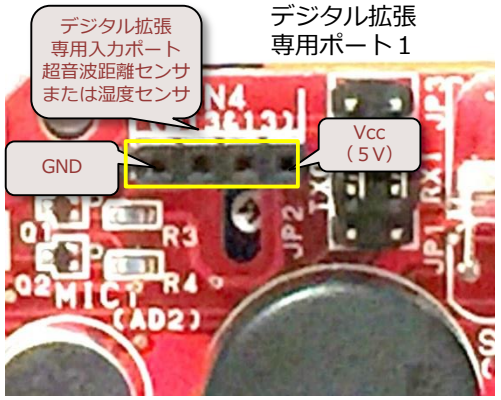
挑戦してみよう。
本スケッチは、ネット上で探し出してみてください。

参照：サンプルスケッチ
IoTABS4_DTH11.ino を添付

温湿度センサDTH11 / DTH22(AM2302)
1）Vcc（3.5V～5.5V）
2）SDA
3）NC（接続無し）
4）GND
※ 正面から見て、左側が1番ピン



超音波距離センサと同じポートを使い、USB側ケーブル側に表面（写真面）を向けて差し込みます。



１０．切換えスイッチを使う

切換えスイッチを利用することで、デジタルピンのD10-D13およびアナログピンのA0-A3に取りつく電子部品を利用可能としたり不可能としたりができます。

ここで紹介するサンプルスケッチを使って、切換えスイッチを切り替えて様子をみてみましょう。

【ON】側：常に距離センサが稼働

【D9】側： 5秒間隔で距離センサが稼働切換え

【OFF】側：距離センサが稼働不可

この場合、LCD上には、以下のような表示が出てきます。

（超音波距離センサは、取り付けておいてください）

超音波距離センサ稼働時

超音波距離センサ不稼働時

IoTABS4_DIST_LCD_D9.ino

```
// 切換えスイッチを中央にして利用
// ５秒間隔で距離センサを測定
#include <Wire.h>

#define TrigPin 13
#define EchoPin 12
#define D9PIN    9

void setup(){
    pinMode(TrigPin,OUTPUT);
    pinMode(EchoPin,INPUT);
    pinMode(D9PIN,OUTPUT);
    Wire.begin();
    lcd_init();
    lcd_clear();
    lcd_setCursor(0,0);
    lcd_printStr("Distance");
}
void loop() {
  static boolean sw;
  static uint32_t tm = millis()/1000;
    static char pr[9];
    digitalWrite(TrigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(TrigPin, LOW);
    sprintf(pr,"%4d cm ",int(pulseIn(EchoPin,HIGH)*0.017));
    lcd_setCursor(0,1);
    lcd_printStr(pr);
    delay(100);
    if( millis()/1000 -tm > 5 ) {
      digitalWrite(D9PIN,sw); sw = !sw;
      tm = millis()/1000;
    }
}
```

第７章
IoTABシールド出力部品

1．出力電子部品の概要

ここでは、デジタル切換えスイッチも含めて説明

IoTABシールドに装備された出力電子部品は以下のとおりです。

■アナログ電子部品 （圧電スピーカ：D9とLED２，４，５）

■デジタル電子部品（D２～D７）

デジタル出力ポート	接続電子部品
D3	LED１（PWM）
D4	LED２
D5	LED３（PWM）
D6	LED４（PWM）
D7	LED５
D8	LED６ ＆ 赤外線LED

LED１，３，４は
PWM制御可能
（PWM機能を使って明るさを変更可能）

■デジタル電子部品（D10）

デジタル出力ポート	接続電子部品
D9	切換えスイッチ
D10	圧電スピーカ（PWM）

切換スイッチを中央位置にした場合にのみ、D9の切り替えでIoTABシールドの電子部品のOn/Offが切替できます

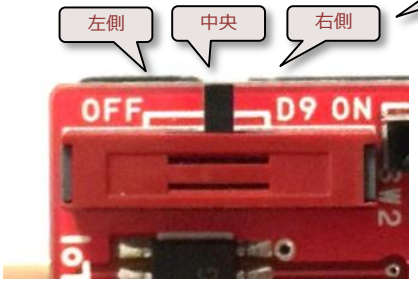
■シリアル通信電子部品（I2C：A4,A5）

シリアル通信	接続部品
A4(I2C:SDA)	I2C-LCD
A5(I2C:SCL)	I2C-LCD

※I2Cは、ArduinoUNOの場合A4,A5でも利用可

（注意）LED６と赤外線LEDは、同じD7を利用

左側からLED１（D3)から
LED6（D8)と配置



中央位置
HIGH： 電子部品On
LOW： 電子部品Off

切換えスイッチ

右側： 電子部品On
左側： 電子部品Off

アナログ
デジタル

2．圧電スピーカを使う

■ 圧電スピーカ

項　目	説　明
製品名	C9767BB422LFP メーカ（DB）

I/O	**デジタル出力（D10）**
	pinMode(10,OUTPUT);　必須
	tone(10,hz,dly);
	noTone(10);

備　考	hz: 音の高さ（右欄参考） dly: 時間(ms)

アナログ出力でも
音が出せる

■ 音の高さ（単位：Hz）

音階	3	4	5	6	7	8	9
B	247	494	988	1976	3951	7902	
A#	233	466	932	1865	3729	7459	
A	220	440	880	1760	3520	7040	14080
G#		415	831	1661	3322	6645	13290
G		392	784	1568	3136	6272	12544
F#		370	740	1480	2960	5920	11840
F		349	698	1397	2794	5588	11176
E		330	659	1319	2637	5274	10548
D#		311	622	1245	2489	4978	9956
D		294	587	1175	2349	4699	9397
C#		277	554	1109	2217	4435	8870
C		262	523	1047	2093	4186	8372

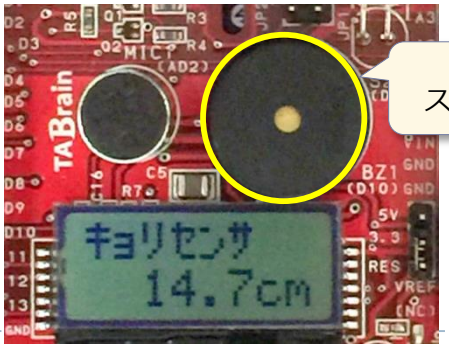
※ここで、ド：C、レ：D、ミ：E、ファ：F、ソ：G、ラ：A、シ：Bとなります。

■サンプルスケッチ

```
void setup(){}

void loop() {
  tone(10,262,800);
  delay(1000);
  tone(10,294,800);
 delay(1000);
  tone(10,330,800);
  delay(2000);
}
```

IoTABS4_SPK.ino

ドレミを連続して繰り
返すスケッチ



圧電
スピーカ

TABrain

3．６個のLEDを使う

デジタル

■ ６個のLED

項　目	説　明
製品名	ＯＳＹＧ１６０８

I/O	**デジタル出力（D3~D８）**
	pinMode(Dx,OUTPUT);
	digitalWrite(Dx,HL);

備　考	Dx: 3～８（つまりD3～D８） HL: HIGHまたはLOW

必須

左側からLED１からLED６と配置

■ サンプルスケッチ１

```
void setup(){
  for(int i=3; i<9; i++)
    pinMode(i,OUTPUT);
}

void loop(){
  for(int i=3; i<9; i++){
    digitalWrite(i,HIGH);
    delay(50);
  }
  for(int i=3; i<9; i++){
    digitalWrite(i,LOW);
    delay(50);
  }
}
```

６個のLEDを順に点灯していき、そのあと消灯していくスケッチ

IoTABS4_LED__01.ino

■ サンプルスケッチ２

流れ星が左から右に移動していく状態のスケッチ

IoTABS4_LED_02.ino

```
void setup() {
  for(int i=3; i<9; i++) {
    pinMode(i,OUTPUT);
  }
}
byte id[6]={0,1,2,3,4,5};
void loop() {
  static unsigned long tm=millis();
  for(int i=3; i<8; i++) digitalWrite(id[i-3]+3,HIGH);
  for(int i=3; i<8; i++){
    delayMicroseconds(pow(2,i-3)*10);
    digitalWrite(id[i-3]+3,LOW);
  };
  if(millis() - tm >80) {
    for(int i=0; i<6; i++) {id[i]= id[i]+1; if(id[i]>5) id[i]=0; };
    tm=millis();
  }
 }
```

【注意】LCD上に透明なフィルムが貼り付けてあり、傷がついている場合があります。不良品ではありません。剥がして使えます。

4．I2C－LCD（液晶ディスプレイ）を使う

I2C

■ **I2C-LCD（8文字×2行）**
キャラクタ液晶ディスプレイ

項　目	説　明
製品名	AQM0802A-RN-GBW メーカ(Xiamen Zettler Electronics)
I/O	**I2C（アナログ出力ピンのA4、A5利用）** #include <Wire.h> それにI2C_LCD専用モジュールの読込みが必要
備　考	専用モジュール「I2C_LCD.ino」添付資料

■ **I2C_LCD.ino 関数群**

関数群	概要説明
lcd_init()	I2C_LCDの初期化
lcd_clear()	画面消去
lcd_DisplayOff()	画面の非表示
lcd_DisplayOn()	画面の表示
lcd_printStr(str)	文字列表示 str：表示する文字列
lcd_setCursor(x,y)	カーソル位置設定 x: 文字カラム(0～7) y: 行数（0～1）

I2C－LCD

スイッチサイエンス社
I2C－LCD

秋月電商
I2C－LCD

※ 「I2C_LCD.ino」をタブ画面に配置のこと
（添付資料参照）

■ **サンプルスケッチ１**

```
#include <Wire.h>
void setup() {
  lcd_init();
}
void loop() {
  lcd_clear();
  delay(500);
  lcd_setCursor(0, 0);
  lcd_printStr("IoTAB V4");
  delay(1000);
  lcd_setCursor(0, 1);
  lcd_printStr("test SCR");
  delay(1000);
}
```

IoTABS4_I2C_LCD.ino

5．EEPROMを使う

■ **EEPROM**
不揮発性メモリー

項　目	説　明
製品名	Arduino/Geuino独自
I/O	**I2C（アナログ出力ピンのA4、A5利用）** #include <EEPROM.h>
備　考	専用ライブラリ「EEPROM. h 」参照

- Arduino UNO R3には、不揮発性のメモリーEEPROMが１Kバイト内蔵されています。
- Genuino 101にも、不揮発性のメモリーEEPROMが２Kバイト内蔵されています。

■ **EEPROM.h 関数群**

関数群	概要説明
EEPROM.write(ad,dat)	データ書き込み ad: アドレス（0～1023・2047） dat: データ（バイト）
EEPROM.read(ad)	データ読み込み ad: アドレス（0～1023・2027）

■ **サンプルスケッチ１**

```
#include <EEPROM.h>
void setup() {
  Serial.begin(9600);
  for (int i = 0; i < 10; i++)
    EEPROM.write(i, 10 - i);
  for (int i = 0; i < 10; i++)
    Serial.println(EEPROM.read(i));
}

void loop() { }
```

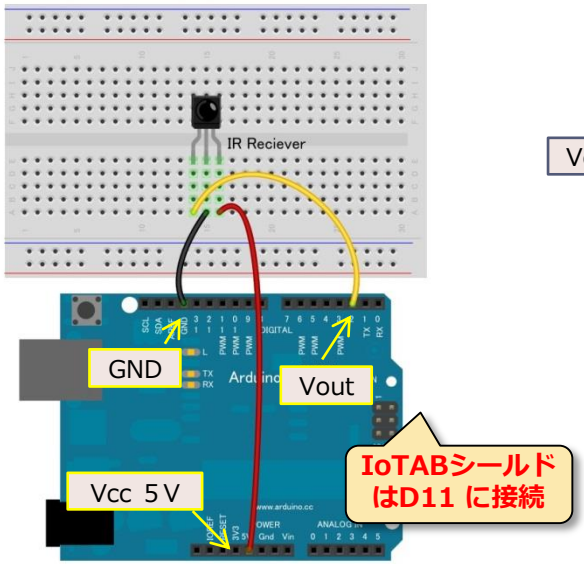
IoTABS4_EEPROM.ino

第8章
赤外線リモコン

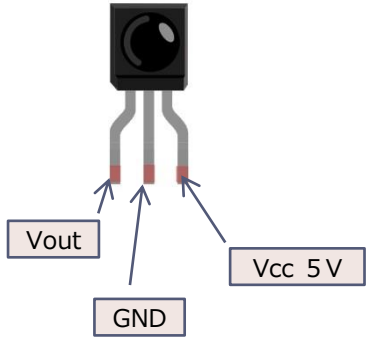
1．赤外線リモコン受信モジュールを使う

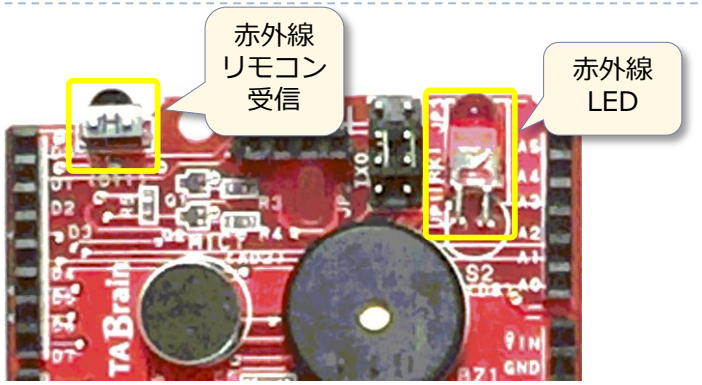
■赤外線リモコンの学習

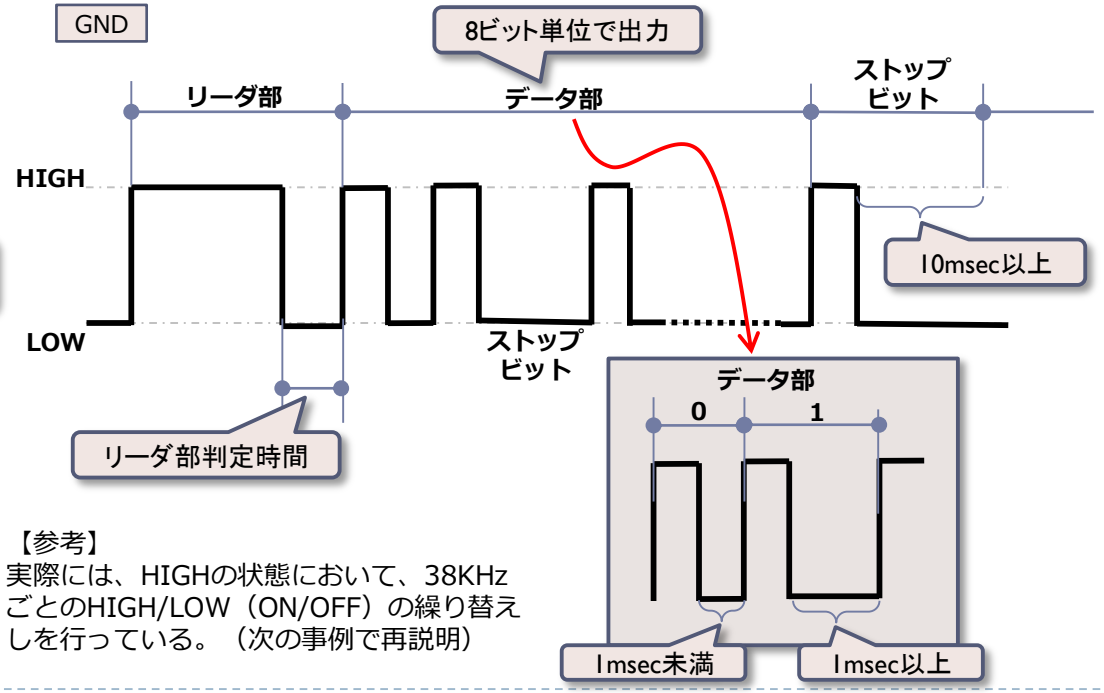
■赤外線リモコン受信モジュール を使って、テレビリモコンなどの信号を読み取ってみましょう。



【注意】テレビリモコンは、メーカによって送信データフォーマットが異なります。よって、ここで紹介するプログラムが稼働しない場合もあります。異なるデータとしては、
① リード部が無い場合
② 押しっ放し操作でデータ部が無い場合
③ リーダ部LOWが4000より小さい場合（プログラム変更にて利用可能）







【参考】
実際には、HIGHの状態において、38KHzごとのHIGH/LOW（ON/OFF）の繰り替えしを行っている。（次の事例で再説明）

2．赤外線リモコン受信モジュールのスケッチ

```
#define IR_Pin      11                  // 赤外線受信モジュールピン番号
void setup()
{
  Serial.begin(9600) ;                  //シリアルモニタ画面表示速度
  pinMode(IR_Pin,INPUT) ;// 赤外線受信モジュールVout表示
}
void loop()
{
  unsigned long t ;
  int i , cnt ;
  boolean  IRbit[64] ;// 読込バッファ
  t = 0 ;
  if (digitalRead(IR_Pin) == LOW) {   // リーダ部チェック
    t = micros() ;                    // 現在の時刻(us)を得る
    while (digitalRead(IR_Pin) == LOW) ;// HIGH(ON)になるまで待つ
    t = micros() - t ;// LOW(OFF)の部分をはかる
  }
  // リーダ部有りなら処理する(3.4ms以上のLOWにて判断する)
  if (t >= 3400) {
    i = 0 ;
    while(digitalRead(IR_Pin) == HIGH) ;// ここまでがリーダ部(ON部分)読み飛ばす
    // データ部の読み込み
    while (1) {
      while(digitalRead(IR_Pin) == LOW) ;// OFF部分は読み飛ばす
      t = micros() ;
      cnt = 0 ;
      while(digitalRead(IR_Pin) == HIGH) {// LOW(OFF)になるまで待つ
        delayMicroseconds(10) ;
        cnt++ ;
        if (cnt >= 1200) break ;        // 12ms以上HIGHのままなら中断
      }
      t = micros() - t ;
      if (t >= 10000) break ;           // ストップデータ
      if (t >= 1000)  IRbit[i] = HIGH ;// ON部分が長い
      else            IRbit[i] = LOW ;    // ON部分が短い
      i++ ;
    }
```

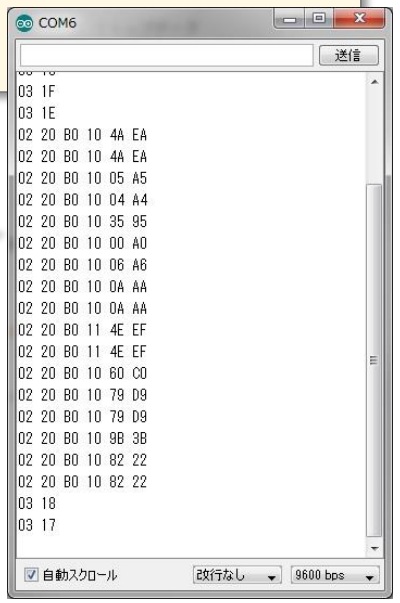
IoTABシールドはD11 に接続

IoTABS4_IR_READ.ino

```
// データ有りなら表示を行う
if (i != 0) {
  IRbit[i] = 0 ;
  int l=0, x;
  x = i / 8 ;
  // ビット文字列データから数値に変換する
  for (int j=0 ; j < x ; j++) {
    int dt = 0 ;
    for (int k=0 ; k < 8 ; k++) {
      if (IRbit[l++] == HIGH)  bitSet(dt,k) ;
    }
    if( dt<16 ) Serial.print('0');
    Serial.print(dt,HEX) ;              // ＨＥＸ(16進数)で表示
    Serial.write(' ') ;
  }
  Serial.println() ;
 }
}
```

ビット設定関数

テレビリモコンから読み取ったデータ事例

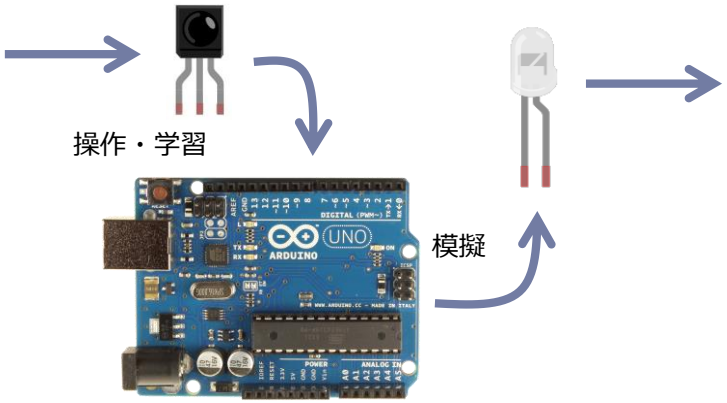


3．照明器具やテレビのリモコンについて

■赤外線リモコンの学習と送信

■つぎは、家庭内のテレビや照明器具のリモコンを使って、その信号を読み取ったものをArduinoに一旦保存し、
赤外線LEDで出力する方法を行ってみましょう。
■また、今回は、読込んだ信号データを永続的に記憶させる場所として、Arduinoに標準で搭載されているEEPROMを使います。

照明器具用
赤外線リモコン



照明器具

■家電製品では、いくつかの赤外線形式が仕様されています。
その中で、最も普及している家電製品協会フォーマットおよび
NECフォーマットを今回学習します。

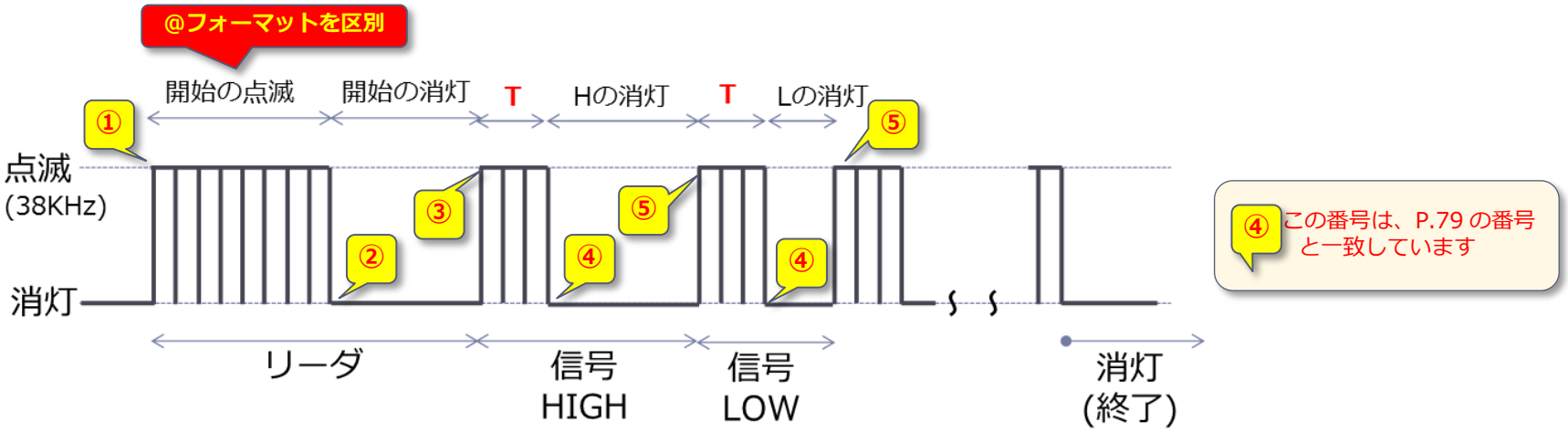
■多くの赤外線リモコンは、波長920~950nmの赤外線を使い、
また自然界の赤外線と区別できるように38KHzの周波数（搬送
波といいます）で点滅させて使います。

■各方式で点灯と消灯の時間が定められていており、その関係
は、①信号の開始、②信号のHIGH/LOWの送出（１または０）、
となっています。

■家電製品により、制御するときに送出するビット列（文字
コードの列）が定められています。その定義は多岐にわたり、
膨大なパターン数となります。そのため、本教材では、その内
容には触れず、赤外線リモコンから読み取った値を、再現する
にとどめます。

【注意】
スマホなどによる外出先からの遠隔操作ができるようになりま
すが、利用にあたっては十分注意してください。（スケッチの
バグや赤外線LEDの設置方法等によっては、うまく電源の
ON/OFF等の制御ができなくなる可能性があります）

4．照明器具やテレビリモコンの赤外線信号



項　目	家電協会フォーマット	NECフォーマット
点滅信号の特性	38KHz、デューティ比50% (HIGH:13ns, LOW:13ns)	38KHz、デューティ比33% (HIGH:9nm, LOW:17nm)
基準となる時間T	0.35~0.5mS	0.56mS
@ 開始の点滅時間※	T×８（=2.8~4mS）	T×16（=9mS）
開始の消灯時間	T×４（=1.4~2mS）	T×8（=4.5mS）
Hの消灯時間	T×３（=1.05~1.5mS）	T×３（=1.68mS）
Lの消灯時間	T×１（=0.35~0.5mS）	T×１（=0.56mS）

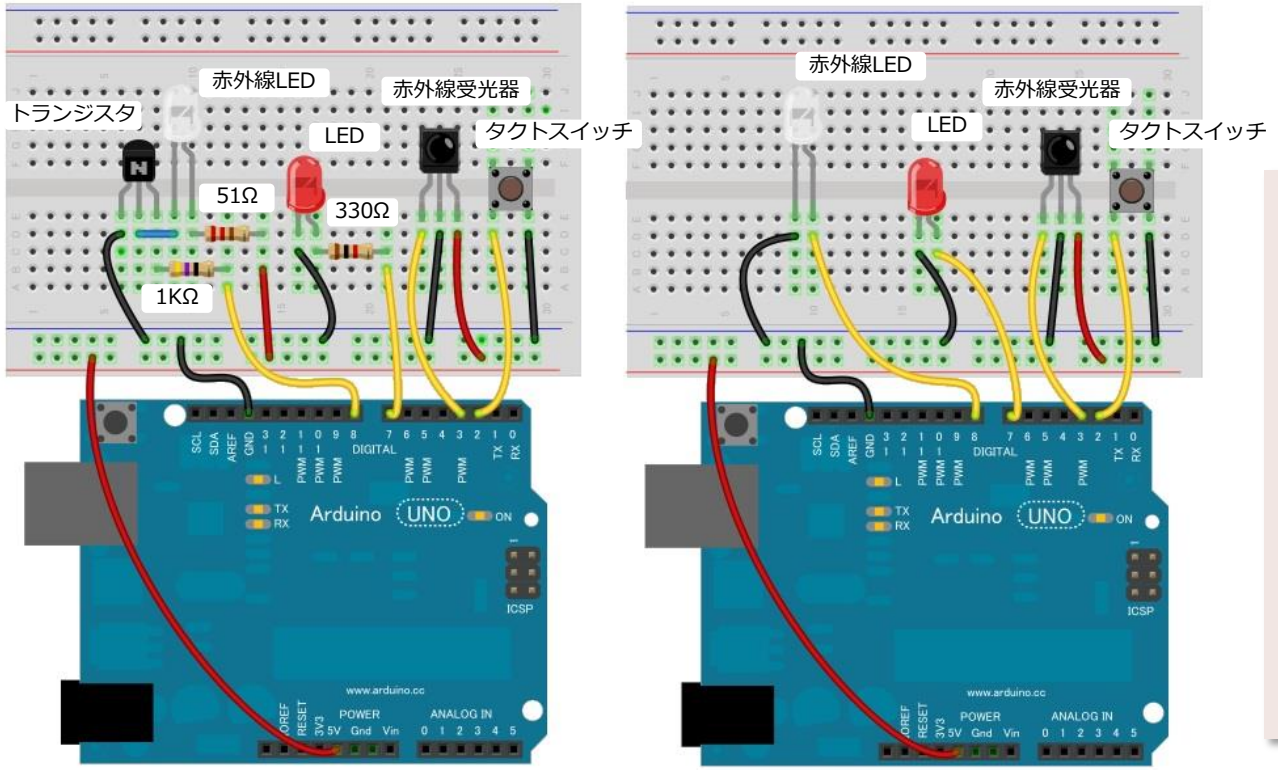
※「開始の点滅時間」の長さで、いずれかのフォーマットを判定

■作成するスケッチの仕様
・家電協会またはNECフォーアットの赤外線リモコンを扱う
・一つの操作だけを学習する
・利便性を考えて、一つのスケッチで、赤外線リモコンの学習と、その操作の両方を実行できるようにする
・タクトスイッチとLEDを使い、学習と操作のOn/Offを切り換える。
・タクトスイッチを押したままリセット（もしくはシリアルモニタ表示）すると、学習モードとなり、LEDが点灯する。
・初期起動時は、タクトスイッチを押した状態で、赤外線リモコンの学習モードとする。
・一度、学習した情報は、EEPROMに書き込まれ、次回以降の立ち上げで、値を読込んでくる。

5．赤外線リモコン受信・送信モジュール組み立て

■配線図
　ここでは、２つの案を提示します。
　周りの環境による雑音の問題や、電流制限などを配慮した場合が、左側となります。

右側の場合には、先ずは簡単な接続として提示しました。



【注意】
本キットで利用している赤外線LEDは、以下のものとなります。

http://akizukidenshi.com/catalog/g/gI-04311/

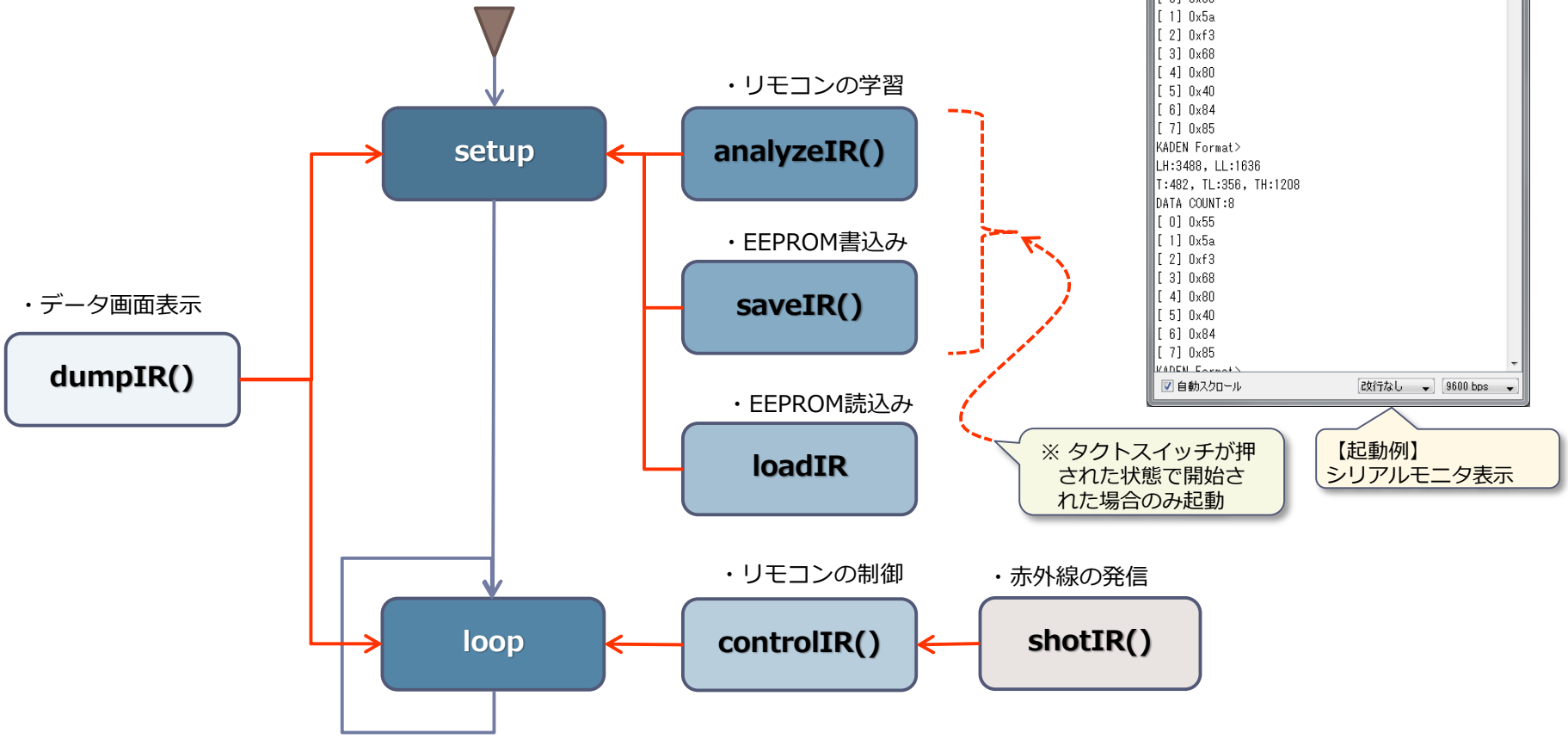
この製品の特性として、「半減角：15度（狭角）」、つまり、15度離れると赤外線のパワーは1/2となります。

よって、リモコンの制御は、赤外線LEDを家電製品の受光部に向けてうまく方向・角度を調整する必要があります。

6．リモコンのプログラミング構成

■スケッチの概要

ここでは、サブモジュール（関数）として、６個用意しました。どれもリモコン操作として重要な機能群となりますので、いろいろと活用してみてください。



7．赤外線リモコンのスケッチ（1）

初期設定（宣言）

```
// IR remote contoroller demo
//  (*) This sketch is 16MHz MCU Only
//#define GENUINO101 1
#ifdef GENUINO101
#include <CurieEEPROM.h>
#else
#include <EEPROM.h>
#endif

#define  MAX_SIGNALS  16

// 接続ピン設定番号（IoTABシールド仕様）
const int  LedPin = 7;
const int  ButtonPin = 2;
const int  IRInputPin = 3;
const int  IROutputPin = 8;

// グローバル変数設定
volatile uint8_t *ir_out, *ir_in;
uint16_t t_ldr_high, t_ldr_low;  // Leader time [100uS]
uint16_t t, t_min, t_max;        // Basic time [10uS]
uint16_t t_high, t_high_min, t_high_max; // High time [10uS]
uint16_t t_low, t_low_min, t_low_max;    // Low time [10uS]
uint8_t  count_signals;
uint8_t  signals[MAX_SIGNALS];
```

EEPROMヘッダー

setup()

```
void setup()
{
  pinMode(LedPin, OUTPUT);
  digitalWrite(LedPin, LOW);       // Led Off
  pinMode(IROutputPin, OUTPUT);
  digitalWrite(IROutputPin, LOW);  // IR Off
  pinMode(ButtonPin, INPUT_PULLUP);
```

IoTAB4_IR_getput_test.ino

```
 while(!Serial);
  Serial.begin(9600);

  if (digitalRead(ButtonPin) == LOW) {
    // ボタンスイッチが押された状態→セットアップ
#ifdef GENUINO101
    EEPROM.clear();
#endif    delay(2000);
    digitalWrite(LedPin, HIGH);  // Led On
    Serial.println("Ready to catch.");

    // Analyze IR signals
    while (! analyzeIR()) Serial.println("Fail to catch, retry.");

    digitalWrite(LedPin, LOW);  // Led Off

   saveIR();    // Save result to eeprom

   // Output analayed results
   dumpIR();
  }

  // Normal mode
  uint8_t nCounts = EEPROM.read(0);
  if (nCounts == 0 || nCounts > MAX_SIGNALS) {
    Serial.println("No IR signals in eeprom.");
    while (1) ;  // Halt here
  }

  // Load signal data
  loadIR();
}
```

タクトスイッチが押された状態で起動

リモコンの学習

EEPROM書込み

データ画面表示

EEPROMデータ無しエラー処理

EEPROM読込み

7．赤外線リモコンのスケッチ（2）

loop()

```
void loop()
{
  if (digitalRead(ButtonPin) == LOW) {
    dumpIR();
    digitalWrite(LedPin, HIGH);  // Led On
    controlIR();    // Send IR signal
    delay(100);
    digitalWrite(LedPin, LOW);   // Led Off
  }
  delay(30);
}
```

データ画面表示

リモコンの制御

analayzeIR()：リモコンの学習

```
// analyzeIR() --
boolean  analyzeIR()
{
  // Initilize variables
  t_min = t_low_min = t_high_min = 9999;
  t_max = t_low_max = t_high_max = 0;
  // Analyze leader
  uint32_t ts = micros();
  while (micros() - ts <= 2500) {
    if (digitalRead(IRInputPin))
      ts = micros();
  }
  while (! digitalRead(IRInputPin))  ;
  t_ldr_high = micros() - ts;
  ts = micros();
  while (digitalRead(IRInputPin)) ;
  t_ldr_low = micros() - ts;
 // Analyze data
  int i;
  uint16_t tt;
```

マイクロ秒単位で波長を読み取り

初期設定

① 赤外線LED ON

② 赤外線LED OFF

開始の点滅時間

③ 赤外線LED ON

開始の消灯時間

```
for (i = 0; i < MAX_SIGNALS; i++) {
    signals[i] = 0;  // Clear
    for (int b = 0; b < 8; b++) {
      ts = micros();
      while (!digitalRead(IRInputPin))   ;
      tt = micros() - ts;
      if (tt > t_max) t_max = tt;
      if (tt > 0 && tt < t_min) t_min = tt;
      ts = micros();
      while (digitalRead(IRInputPin)) {
        if (micros() - ts >= 2000) {
          tt = 0;
          goto _stop;
        }
      }
      tt = micros() - ts;
      signals[i] <<= 1;
      if (tt > 800) {
        signals[i] |= 1;
        if (tt > t_high_max) t_high_max = tt;
        if (tt > 0 && tt < t_high_min) t_high_min = tt;
      }
      else {
        if (tt > t_low_max) t_low_max = tt;
        if (tt > 0 && tt < t_low_min) t_low_min = tt;
      }
    }
_stop:
    if (tt == 0)
      break;  // stop
  }
  count_signals = i; // data signals count
  t = (t_max + t_min) / 2;
  t_high = (t_high_max + t_high_min) / 2;
  t_low = (t_low_max + t_low_min) / 2;
  return true;  // OK!
}
```

④ 赤外線LED OFF

⑤ 赤外線LED ON

データ終了

HIGHの信号処理

LOWの信号処理

7．赤外線リモコンのスケッチ（3）

saveIR()：EEPROM書込み

```
// saveIR() --
void  saveIR()
{
  EEPROM.write(0, count_signals);
  EEPROM.write(1, highByte(t_ldr_high));
  EEPROM.write(2, lowByte(t_ldr_high));
  EEPROM.write(3, highByte(t_ldr_low));
  EEPROM.write(4, lowByte(t_ldr_low));
  EEPROM.write(5, highByte(t));
  EEPROM.write(6, lowByte(t));
  EEPROM.write(7, highByte(t_high));
  EEPROM.write(8, lowByte(t_high));
  EEPROM.write(9, highByte(t_low));
  EEPROM.write(10, lowByte(t_low));
  for (int i = 0; i < count_signals; i++)
    EEPROM.write(i + 11, signals[i]);
}
```

loadIR()：EEPROM読込み

```
// loadIR() --
void  loadIR()
{
  count_signals = EEPROM.read(0);
  t_ldr_high = (EEPROM.read(1) << 8) | EEPROM.read(2);
  t_ldr_low = (EEPROM.read(3) << 8) | EEPROM.read(4);
  t = (EEPROM.read(5) << 8) | EEPROM.read(6);
  t_high = (EEPROM.read(7) << 8) | EEPROM.read(8);
  t_low = (EEPROM.read(9) << 8) | EEPROM.read(10);
  for (int i = 0; i < count_signals; i++)
    signals[i] = EEPROM.read(i + 11);
}
```

dumpIR()：データ画面表示

```
void dumpIR()  // read IR data dump
{
  char buf[100];
  if (t_ldr_high < 5000)
    Serial.println("KADEN Format>");
  else
    Serial.println("NEC Format>");
  sprintf(buf, "LH:%d, LL:%d", t_ldr_high, t_ldr_low);
  Serial.println(buf);
  sprintf(buf, "T:%d, TL:%d, TH:%d", t, t_low, t_high);
  Serial.println(buf);
  sprintf(buf, "DATA COUNT:%d", count_signals);
  Serial.println(buf);
  for (int i = 0; i < count_signals; i++) {
    sprintf(buf, "[%2d] 0x%02x", i, signals[i]);
    Serial.println(buf);
  }
}
```

7．赤外線リモコンのスケッチ（4）

congrolIR()：リモコン制御

```
// controlIR() --
void  controlIR()
{
  // Shot leader sinal
  shotIR(t_ldr_high);
  delayMicroseconds(t_ldr_low);

  // Shot data signals
  for (int i = 0; i < count_signals; i++) {
    for (int b = 0; b < 8; b++) {
      shotIR(t);
      if (signals[i] & (0x80 >> b))
        delayMicroseconds(t_high - 10);
      else
        delayMicroseconds(t_low - 10);
    }
  }
  shotIR(t * 10);  // epilogue
}
```

開始の点滅

開始の消灯

HIGHの消灯

LOWの消灯

shotIR()：リモコン送信

```
// shotIR() -- Shot IR signal
void  shotIR(uint16_t width)
{
  uint16_t us;
  uint32_t ts = micros();

  while (micros() - ts <= (uint32_t)width) {
    ir_out = portOutputRegister(irout_port);
    if (t_ldr_high > 8000) {
      // NEC format(ON 9uS/OFF 17uS)
      digitalWrite(IROutputPin,HIGH);
      delayMicroseconds(9);
      digitalWrite(IROutputPin,LOW);
      delayMicroseconds(17);
    }
    else {
      // KADEN format(ON 13uS/OFF 13uS)
      digitalWrite(IROutputPin,HIGH);
      delayMicroseconds(13);
      digitalWrite(IROutputPin,LOW);
      delayMicroseconds(13);
    }
  }
}
```

NECフォーマット

家電協会フォーマット

**リモコン制御の場合
赤外線LEDと器具との距離が離れ過ぎていると
制御できない場合があります。
また一部指向性があります。
器具の受光器に近いところで
赤外線LEDを向けて操作してください。**

**IoTABシールドのサンプルスケッチには
IoTABシールドの内臓するEEPROMに
複数のリモコンを保存し、また活用する
ものとなっています。
参考にしてください。**

第９章
IoTABシールド応用事例

IoTABシールド組み合わせ学習

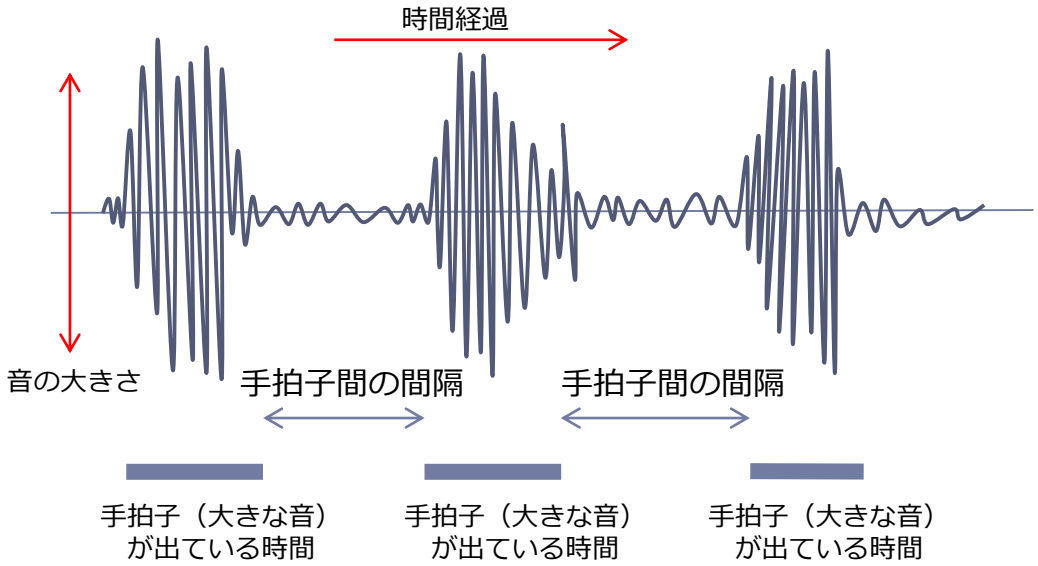
IoTABシールドのもつ多くの電子部品を組み合わせ、入力と出力とで結びつけることで、さまざまなアイデア製品の開発が学べます

ここでは、参考として、以下のソフトウェアによる開発事例を紹介いたします。

	システム	利用する電子部品	組合せて試作する製品（案）
1	手拍子カウント	音センサ、LCD、 ６個のLED	手拍子のような連続した大きな音を感知し、一定時間、音がしなくなると、それまでのカウント数を出す。再度、カウントする状態になり、手拍子が鳴るのを待つ。
2	音速測定	２つのIoTABシールド上の音センサ 6個のLCD	２つのIoTABシールドを１本のケーブルで接続し、互いの音センサの距離を正確に１m離し、その間での大きい音の時間差をもって音速を測定する。
3	学習リモコン	赤外線リモコン 赤外線LED LCD、可変抵抗器 タクトスイッチ EEPROM	テレビリモコンや照明LEDリモコンなどの赤外線信号を「読取モード」状態でEEPROMに書き込み。「呼出モード」状態でEEPROMから読み込んで、スイッチで赤外線LEDで送信。この場合、リセットとスイッチを同時に押した状態で、「読取モード」とし、リセットのみの場合には、「呼出モード」とする。
4	タイマー	LCD タクトスイッチ 可変抵抗器 圧電スピーカ	**タイマー**：可変抵抗器を使って、計測する時間を設定し、タクトスイッチで、開始し、時間をカウントダウンさせる。時間がゼロになったときに、アラームをブザーから出す。 **ストップウオッチ**：タクトスイッチを押した段階から、時刻を刻み、次にスイッチを押したら、押された時刻を表示。その間もラップを刻み続け、再度スイッチを押すと、つぎのラップを刻むといった表示を継続させる。
5	万歩計	Genuino101 LCD、LED	Genuino101に搭載された慣性センサ（加速度センサ）による振動を捉え、万歩計サンプルプログラムをベースに、IoTABシールド上のLCDに万歩計の数値を表示する。

1．手拍子カウント ①

ここでは、手拍子をカウントするプログラムを紹介します。
まずは音センサで取れる手拍子の波形を想定してみましょう。この大きな波形の数をカウントします。



課題は、波形の大きい音と小さい音を区別することです。また音センサでは、アナログ値として、０～１０２３の値で、Arduino UNOでは、値が512を中心として上下した値が波形で収集できます。

サンプルスケッチは、以下のところからダウンロードできます。

http://tabrain.jp/tabs/TABshieldAllTest.zip

手拍子の音が出ている間は、大きな振動が起きています。その間は、僅か1秒もない間で、継続した音の大きさを捉えることで、手拍子として認識させます。

また、手拍子と手拍子の間隔が、ある一定以上になると、それまでの手拍子のカウントをします。つまり、短い間だと、つぎの手拍子をカウントすることにします。

挑戦してみよう
サンプルスケッチに手拍子をカウントするプログラムが入っています。自分なりにアレンジして、手拍子のカウント数で何かを制御してみてください。

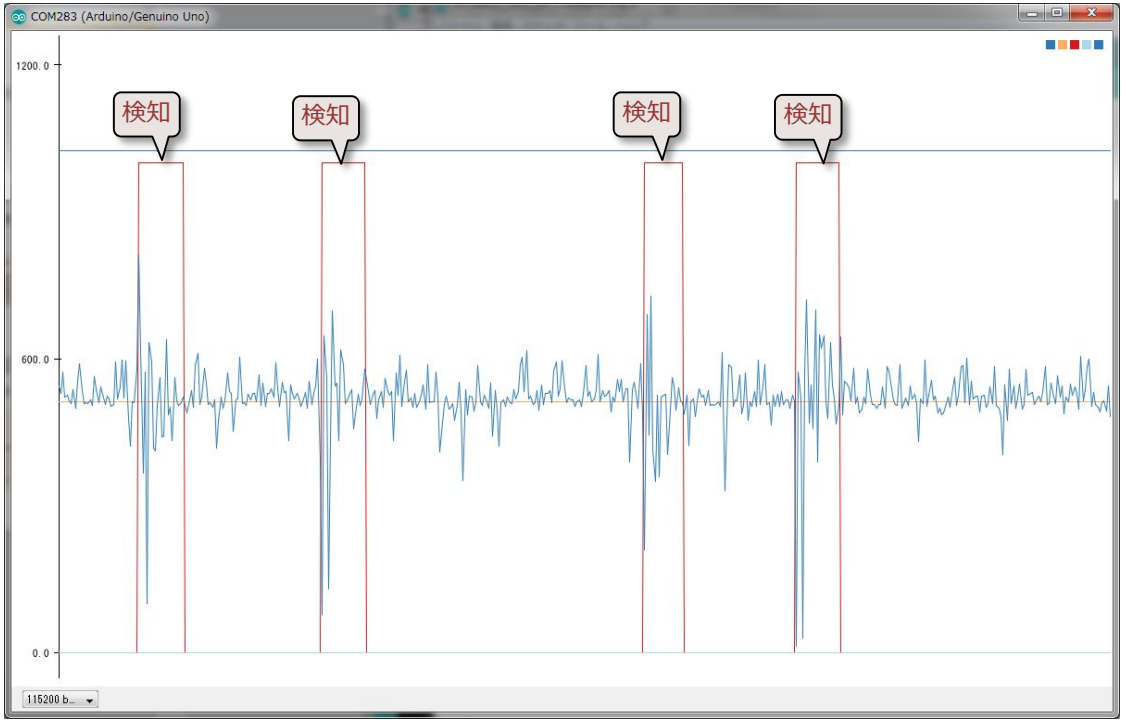
1．手拍子カウント ②

IoTABS4_CLAP_count.ino

```
void setup() {
 Serial.begin(115200);
}

void loop() {
 int val = analogRead(A2);    // IoTABシールドの音センサI/O
 static unsigned long tm = millis();
 static int sw;
 if( abs(val-512) > 250){ sw=1;tm=millis(); }
 else if ( millis()-tm>100 ) sw=0;
 Serial.println("1024¥t512¥t"+ String(sw*1000) + "¥t0¥t" + String(val));
 delay(5);
}
```

arduino.cc の IDEでは、グラフモニタ画面があり、音センサの値を時刻歴で表示させることが可能となりました。

このことを利用して、手拍子のタイミングをこのグラフ表示させてみましょう。



2．タクトスイッチとLED

タクトスイッチを押す度に、LEDを点灯したり、
消灯したりします。

IoTABS4_SW_LED01.ino

```
// チャタリング処理（LED点灯・点滅）
#define SWPin 2

void setup(){
  pinMode(SWPin,INPUT_PULLUP);
  for(int i=3;i<9; i++) pinMode(i,OUTPUT);
}
void loop(){
  static boolean sw=HIGH;
  while(digitalRead(SWPin));
  for(int i=3; i<9; i++) digitalWrite(i,sw);
  sw=!sw;
  delay(300);
}
```

チャタリング
を考慮した
待機時間

タクトスイッチを押すと、LEDを点灯し、
離すと、LEDが消灯します。

IoTABS4_SW_LED02.ino

```
#define SWPin 2

void setup(){
  pinMode(SWPin,INPUT_PULLUP);
  for(int i=3;i<9; i++) pinMode(i,OUTPUT);
}
void loop(){
  for(int i=3; i<9; i++)
    digitalWrite(i,!digitalRead(SWPin));
}
```

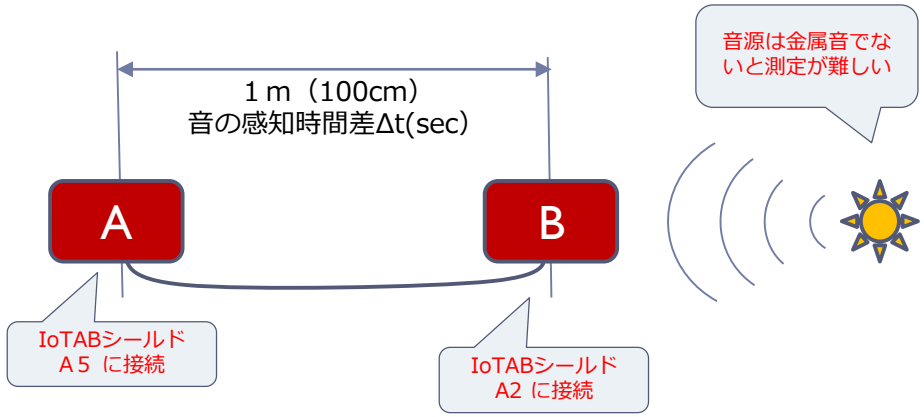
3．音速測定

２つのIoTABシールド「A」と「B」を使い、間隔１mで離し、音の感知の時間差（Δｔ：単位秒）を測定し、以下の計算式で音速を求めます。
　音速（m）＝ 1/Δt
２つのIoTABシールドは「A」のA5ポートと「B」のA5ポートを接続します。

実験結果は、ほぼ 340m/秒 近くで、PC上に表示されれば問題ありません。ただ正しく音が取得しやすい状況を作ることがポイントとなります。

IoTABS4_SOUND_SPEED_A.ino

```
//本スケッチは、２台のArduino+IoTABシールドで
//音速を測定するもの
//奥のArduinoに本プログラムを書き込み
//奥のArduinoからシリアル画面に値を表示させる
void setup() {
  for (int i = 3; i < 9; i++) pinMode(i, OUTPUT);
  Serial.begin(115200);//音速をシリアルモニタ画面で表示
}
void loop() {
  static int val;
  uint32_t tim;
  val = analogRead(A5) - 512;//手前のTABシールドの音センサ値
  val = val < 0 ? -val : val;
  if (val > 300) {
    tim = micros();
    do {
      val = analogRead(A2) - 512; //奥のTABシールドの音センサ値
      val = val < 0 ? -val : val;
    } while ((val < 300)&& (micros() - tim < 3200));
    tim = micros() - tim;
    if (tim < 3200) {
      Serial.print(1000000.0 / float(tim));
      Serial.println(" m ");
      for (int i = 3; i < 9; i++) digitalWrite(i, HIGH);
      delay(3000);
      for (int i = 3; i < 9; i++) digitalWrite(i, LOW);
    }
  }
}
```



IoTABS4_SOUND_SPEED_B.ino

```
// 音源に近い手前のArduinoに本プログラムを書き込む
void setup() {
   for(int i=3; i<9; i++) pinMode(i,OUTPUT);
   Serial.begin(115200);
}

void loop() {
 int val = analogRead(A2)-512;
 val = val<0?-val:val;
// Serial.println(val);
 for(int i=3; i<9; i++) digitalWrite(i,val>300);
}
```

4．テレビ学習リモコン ①

テレビの学習リモコンとして、タクトスイッチと3軸加速度センサを使い、傾きによってテレビの操作ができるものを作成してみましょう。

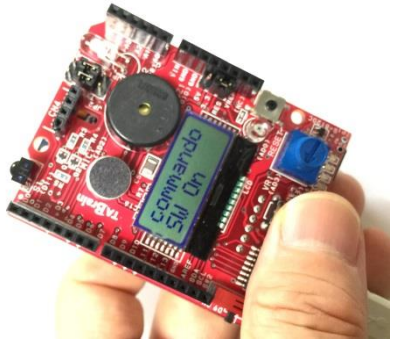
最初にテレビのリモコンを読み込み、電源スイッチ、ボリュームのUP／DOWN，チャンネルのUP／DOWNの５個をLCDに表示した内容で操作するようにします。

【リモコン学習モード時】
このリモコンを読み込む操作は、タクトスイッチ（ボタン）を押した状態で、一緒にリセットボタンを押した時に...実行されます。

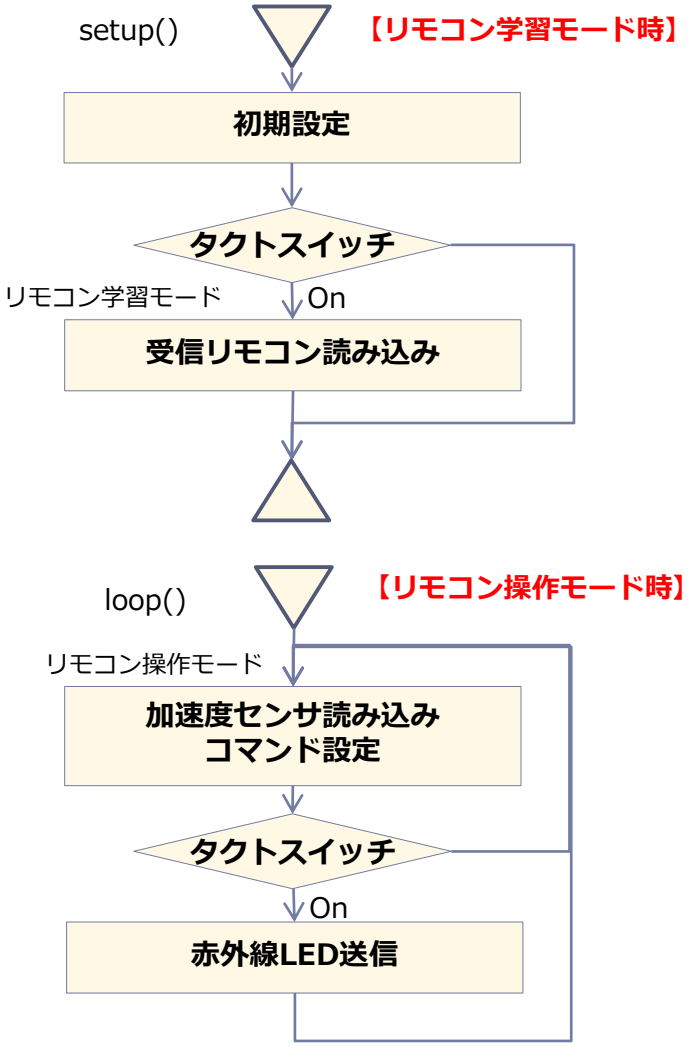
【リモコン操作モード時】

後は、写真のように、傾きによって、これらのコマンドを選択できるようにし、スイッチを押すことで実行できるようにしました。おまけとして、コマンドをLED点灯でも分かるようにします。

注意点として、IoTABシールドの赤外線LEDは指向性が強いために、テレビのリモコン受光位置に向けて操作する必要があります。



【リモコン操作】　以下の状態でタクトスイッチを押す
水平状態　電源ON/OFF
右に少し傾ける　ボリュームUP
右に大きく傾ける　ボリュームDOWN
左に少し傾ける　チャンネルUP
左に大きく傾ける チャンネルDOWN

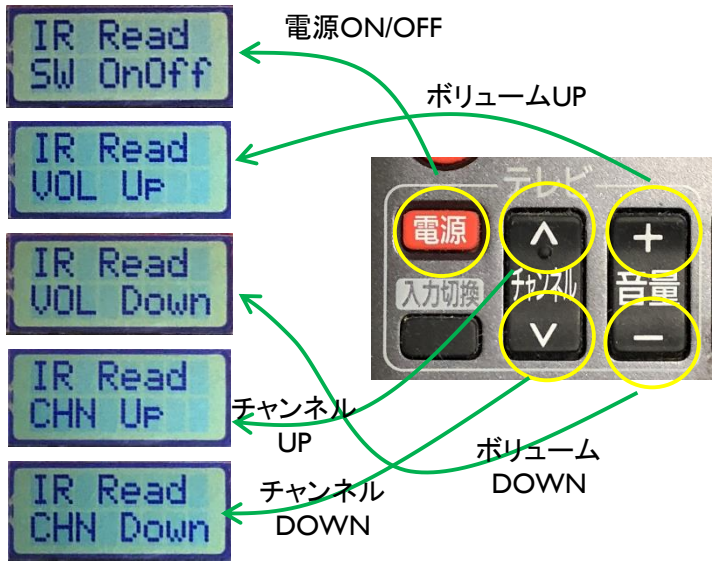


4．テレビ学習リモコン ②

本プログラムでは、テレビリモコンの５つのボタン「**電源ON/OFF**」「**チャンネルUP**」「**チャンネルDOWN**」「**ボリュームUP**」「**ボリュームDOWN**」を読込み、IoTABシールド上の**加速度センサによる傾き**によって、操作コマンドを切り替えるスケッチを考えてみました。

【リモコン学習モード時】

IoTABシールドの**タクトスイッチを押した状態**で、リセット（または電源ON）し、その後タクトスイッチを離します。その後５つのボタンをLCD表示に従ってリモコンのボタン押します。



【リモコン操作モード時】

IoTABシールドを傾け（傾斜させ）ると、５つのコマンドが表示されます。またLEDも表示数が１から５個までが表示されます。タクトスイッチをLCDに表示されたコマンド状態で押すと、赤外線LEDから、リモコン操作が照射されます。以下５つの傾斜状態で切換えられます。

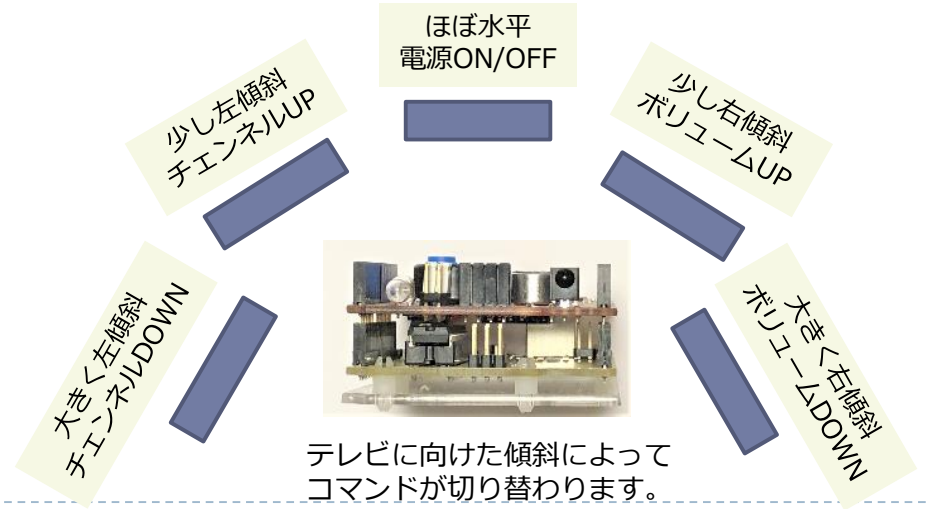
１）ほぼ水平の場合：電源ON/OFFモード

２）少し右に傾けた場合：ボリュームUP

３）大きく右傾斜した場合：ボリュームDOWN

４）少し左に傾けた場合：チャンネルUP

５）大きく左に傾けた場合：チャンネルDOWN



テレビに向けた傾斜によって
コマンドが切り替わります。

4．テレビ学習リモコン ③

本テレビ学習リモコンでは、EEPROMを使って、読み込んだ赤外線IRを記憶します。

一旦読み込んだ操作ボタンは、Arduinoの電源を切っても保存されたままで、つぎに電源を入れた状態では、前頁の「学習リモコン操作」となります。

ここでは、ライブラリとして、以下の外部ライブラリ

1）Wire.h

2）EEPROM. h

それに内部ライブラリ（タブ画面）として

3）I2C_LCD（LCD画面表示）

4）MMA8452Q（加速度センサ）

5）analayzeIR（学習リモコン分析）

6）submoduleIR（IR関連モジュール）

を使っています。

IoTABS4_TV_IR_STUDY.ino

```
//   テレビ学習リモコン
//  タクトスイッチを押した状態で、リセットを押すと リモコン学習モード
//   何もせずにリセットすると、リモコン操作モード

#include <Wire.h>
#include <EEPROM.h>
#define Spk_Pin 10
#define But_Pin  2
#define Led1Pin 3
#define Led2Pin 4
#define Led3Pin 5
#define Led4Pin 6
#define Led5Pin 7
#define Led6Pin 8
#define  MAX_SIGNALS  16
#define MMA8452Qaddr 0x1C
struct MMA8452Q { int x,y,z;} acc;

const int  IRInputPin = 11;
const int  IROutputPin = 8;

// グローバル変数設定
volatile uint8_t *ir_out, *ir_in;
uint8_t  irout_bit, irout_port;  // IROutputPin's PORT and Bit
uint8_t  irin_bit, irin_port;    // IRInputPin's PORT and Bit
uint16_t t_ldr_high, t_ldr_low;  // Leader time [100uS]
uint16_t t, t_min, t_max;        // Basic time [10uS]
uint16_t t_high, t_high_min, t_high_max; // High time [10uS]
uint16_t t_low, t_low_min, t_low_max;    // Low time [10uS]
uint8_t  count_signals;
uint8_t  signals[MAX_SIGNALS];

char pr[9];
char cmd[5][9] = { "SW OnOff","VOL Up  ", "VOL Down","CHN Up  ","CHN Down"};
```

4．テレビ学習リモコン ④

```
void setup() {
  Wire.begin();
  Serial.begin(9600);
  pinMode(Spk_Pin,OUTPUT);
  pinMode(Led1Pin,OUTPUT);
  pinMode(Led2Pin,OUTPUT);
  pinMode(Led3Pin,OUTPUT);
  pinMode(Led4Pin,OUTPUT);
  pinMode(Led5Pin,OUTPUT);
  pinMode(Led6Pin,OUTPUT);
  pinMode(But_Pin,INPUT_PULLUP);
  MMA8452Q_init();
  irout_bit = digitalPinToBitMask(IROutputPin);
  irout_port = digitalPinToPort(IROutputPin);
  irin_bit = digitalPinToBitMask(IRInputPin);
  irin_port = digitalPinToPort(IRInputPin);

  lcd_init();// I2C LCD 初期化
  //リモコン学習モード
  while(digitalRead(But_Pin)==LOW) {
    for(int i=0; i<5; i++) {
      lcd_setCursor(0,0);
      lcd_printStr("IR Read ");
      while(digitalRead(But_Pin)==LOW); // ボタンOFFまで待機
      delay(300);
      lcd_setCursor(0,1);
      lcd_printStr(cmd[0,i]);
      while (! analyzeIR())  {
        lcd_setCursor(0,1);
        lcd_printStr("Read Err");
      };
      lcd_setCursor(0,0);
      lcd_printStr("IR GET  ");
      delay(1000);
      saveIR(i);
    }
  }
}
```

```
//リモコン操作モード
void loop() {
  lcd_setCursor(0,0);
  lcd_printStr("commando");
  MMA8452Q_readAcc();
  int x=acc.x;//x方向しか利用しない
  int y=acc.y;
  int z=acc.z;
  Serial.print(x); Serial.print("¥t");
  Serial.print(y); Serial.print("¥t");
  Serial.print(z); Serial.print("¥n");

  byte com;
  lcd_setCursor(0,1);
  if(abs(x)<201 ) {
    com = 0;
  } else if( x<-650 ) {
      com = 4;
  } else if( x<-200 ) {
      com = 3;
  } else if( x>650 ) {
     com = 2;
  } else if( x>200 ) {
      com= 1;
  } else com=5;
   ;
  digitalWrite(com+2,HIGH);
  if( com<5 ) {
  loadIR(com);
  lcd_printStr(cmd[0,com]);

  while(digitalRead( But_Pin)==LOW ) {
     tone(Spk_Pin,200+com*100,100);
     loadIR(com);
     dumpIR();controlIR();
  }
  delay(300);}
  digitalWrite(com+2,LOW);
}
```

5．タイマーを作る ①

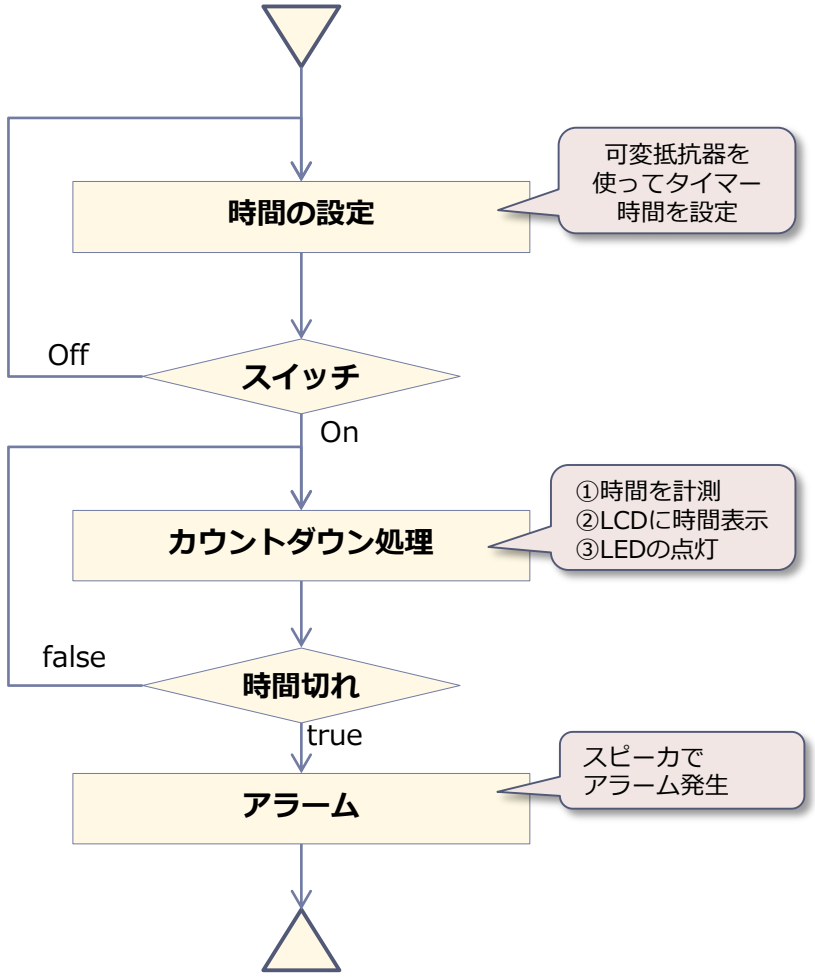
時間（1秒～60分）設定後、アラームを鳴らすタイマーを作ります。
タイマーのプログラムは、以下の順序で行います。

1） 時間の設定（可変抵抗器を使って、LCDでタイマー時間表示）
2）スタート（タクトスイッチを使って）
3）カウントダウン（５個のLED、LCDを使って）
4）時間になって（スピーカでアラーム）

ここで時間設定での設定時間
以下の間隔で設定を行う

1）1秒～60秒までは 1秒間ごと
2）1分～3分までは5秒間ごと
3）3分～5分までは15秒間ごと
4）5分～10分までは30秒間ごと
5）10分から60分までは1分間ごと

モジュール化(関数)
long set_timer()



5．タイマーを作る ②

```
#include <Wire.h>
#include <EEPROM.h>

#define SpkPin 10
#define ButPin 2
#define RegPin A3
```

```
void setup(){
  lcd_init();// I2C LCD 初期化
  pinMode(SpkPin,OUTPUT);
  pinMode(ButPin,INPUT_PULLUP);
  for(int i=3;i<9;i++)
    pinMode(i,OUTPUT);
  lcd_setCursor(0,0);
  lcd_printStr("TIME");
  int limtime;
  char pr[8];
  do {// set timer
    limtime=set_timer();
    sprintf(pr," %2d:%2d",limtime/60, limtime%60);
    lcd_setCursor(0,1);
    lcd_printStr(pr);
    sprintf(pr,"%4d",limtime);
    lcd_setCursor(4,0);
    lcd_printStr(pr);
  }while(digitalRead(ButPin)==HIGH);
  long time=millis();
  int stime,ostime=0;
  do{ // count down
     stime=(millis()-time)/1000;
     if(ostime!=stime) {
       sprintf(pr," %2d:%2d",(limtime-stime)/60,(limtime-stime)%60);
       lcd_setCursor(0,1);
       lcd_printStr(pr);
       ostime=stime;
       int val=map(stime,0,limtime,6,0);
       for(int i=3; i<9; i++)
          digitalWrite(i,(val>i-2)?HIGH:LOW);
     }
  }while(time+(long)limtime*1000>millis());
    digitalWrite(2,LOW);
    lcd_setCursor(0,1);
    lcd_printStr("TimeOver");
  do{ // speaker
    tone(SpkPin,400,500);
    delay(500);
    noTone(SpkPin);
    delay(500);
  }while(digitalRead(ButPin)==HIGH);
}
```

時間設定

カウントダウン

時間切れアラーム

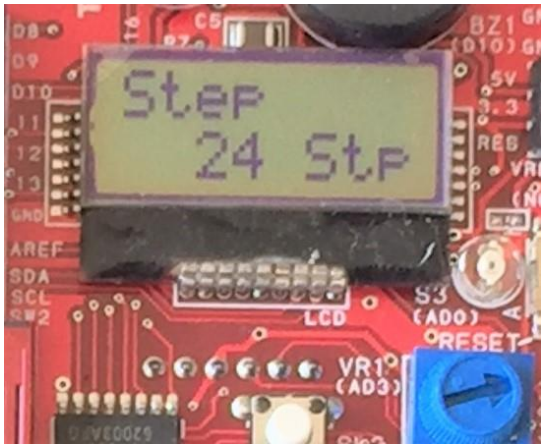
IoTABS4_TIMER.ino

時間設定のモジュール

```
void loop(){}

long set_timer(){
  int reg = analogRead(RegPin);
  if(reg<375) return(map(reg,0,374,1,60));//1-60s@1s
  else if( reg<600 )
    return(60+5*map(reg,375,599,0,36));//1-3m@5s
  else if( reg<650 )
    return(180+15*map(reg,600,649,0,8));//3-5m@15s
  else if( reg<712 )
    return(300+30*map(reg,650,711,0,10));//5-10m@30s
  else return( 600+60*map(reg,712,1023,0,50));//10-60m@1m
}
```

6．加速度センサを使った万歩計 ①

加速度センサを使った万歩計のサンプルスケッチを紹介します。これを使って、万歩計をLCDに表示するスケッチに変更してみましょう。



IoTABシールドの加速度センサ（MMA8452Q)を使って、1.25G以上（20Hz

6．IoTABシールドV4.0による万歩計 ②

別途「I2C_LCD.ino」「MMA8452Q]ライブラリをタブに追加が必要です。

ライブラリ追加

LEDのピン番号変更

```
#include <Wire.h>
#define  MAX_SIGNALS  16
#define MMA8452Qaddr 0x1C
struct MMA8452Q {
  int x, y, z;
} acc;

const int ledPin = 3;
boolean stepEventsEnabeled = true;   // whether you're polling or using events
long lastStepCount = 0;              // step count on previous polling check
boolean blinkState = false;          // state of the LED

void setup() {
  Serial.begin(9600);
  lcd_init();
  lcd_setCursor(0, 0);
  lcd_printStr("Step    ");
  lcd_setCursor(0, 1);
  lcd_printStr("Counter ");
  delay(1000);
  pinMode(ledPin , OUTPUT);
  // intialize the sensor:
  MMA8452Q_init();

  lcd_setCursor(0, 1);
  lcd_printStr("Start  ");
}
```

LCD初期化・表示追加

LCD表示追加

IoTABS4_STEP_COUNT.ino

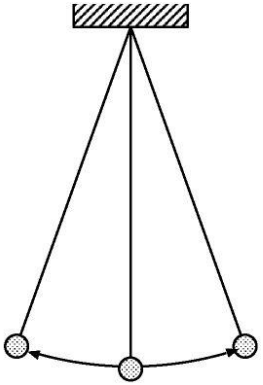
```
void loop() {
 uint32_t tim = millis();
  uint32_t val, x, y, z;
  do {
    MMA8452Q_readAcc();
    x = (uint32_t)acc.x;
    y = (uint32_t)acc.y;
    z = (uint32_t)acc.z;
    val = sqrt(x * x + y * y + z * z);
    if (val > 1250) break;
  } while (millis() - tim < 300);
  if (val > 1250 ) {
    updateStepCount();
    digitalWrite(ledPin , blinkState);
    blinkState = !blinkState;
  }
  delay(400);
}

static void updateStepCount() {
  static  int stepCount = 0;
  stepCount++;
  lcd_setCursor(0, 1);
  char pr[9];
  sprintf(pr, "%4d Stp", stepCount);
  lcd_printStr(pr);
}
```

LCD表示追加

7．IoTABシールドを使った応用事例

タイマー

メロディ



振り子周期計測

音叉振動数計測

光（照度）計測

万歩計

音（ノイズ）計測

超音波距離計測

熱中症観測

モールス信号

学習リモコン

第Ⅲ編 展開編

※３GIMは別売品となっています。

第１０章
３GIM とは

３GIMに関するマニュアルは別途以下からダウンロードしてください。

http://tabrain.jp/3GIM_V2.0/3GIM%20V2.0R01manual.pdf

1．３GIMのコンセプト

- 誰もが簡単に利用できること
  - 電気・電子の知識が無くても３G通信技術とセンサ技術が学べること
  - 小中学生からでもサンプルスケッチを真似て応用展開できること
  - DIYで簡単にシステムが構築できること

今回のコンテストで中学生が優秀賞を受賞

- 最先端で高度な技術を理屈なしに学べること
  - 理論や理屈なしで、最先端の高度な技術を、利用・活用できること
  - 無駄な学習時間なしに、実技を身に着けられること

東大教授森川先生から「M2Mはニッチなビジネス、簡単に試作できる環境が必要」

- センサ技術＋ワイヤレス技術＋ネット技術の応用展開へ繋げること
  - オープンソースハードウェアを使ったセンサ技術との連携を容易にすること
  - Webサービスやクラウドサービスなどとの連携を容易にすること

M2M研究会会長挨拶から「技術的な課題、運用面の課題、ビジネス上の課題が多くある」

- 人材育成と雇用創出へつなげること
  - 多くの雇用創出につなげた人材育成での教育につながること
  - 新たなビジネスチャンスを生む雇用創出につながること

MCPC モバイルM２M委員会の方々から「３GIMは、M2Mの裾野を広げてくれる可能性が大きい」⇒３GIMを使って簡単にM２Mが行えるツールが欲しい

3GIMは、技術的な課題を大幅に縮小

2．３GIM V2.2とは ①

- Arduino上で簡単に3G通信を行うことができる３G通信モジュール
  - FOMA系の通信をサポート（docomo、MVNOのIIJmio、IIJmobile、sonet nuro LTE、 b mobile、DTI、Soracom等）
  - クアルコム設計・AnyData製造の**IEMモジュール版**（既販売）と、**USBドングル版**(予定）を使った２つの３GIMを用意

- 低いCPU性能で少ないメモリを持つArduino上で利用できるよう、高機能なAPIをライブラリ”a3gim”として提供

  （※海外でも３GIMが販売されているが、ATコマンドで利用：技術ハードルは高い）
  - http://tabrain.jp/3GIM_V2.2/a3gimR4.0manual.pdf

- ライブラリが提供する予定の機能は、下記の通り：
  - SMS送受信(send, receive, check, onReceived)
  - Web通信(HTTP GET/POST)
  - GPS位置取得(GPS Standalone, AGPS)
  - TCP/IP通信(connect, disconnect, read, write)
  - その他（時刻やIEM取得等）

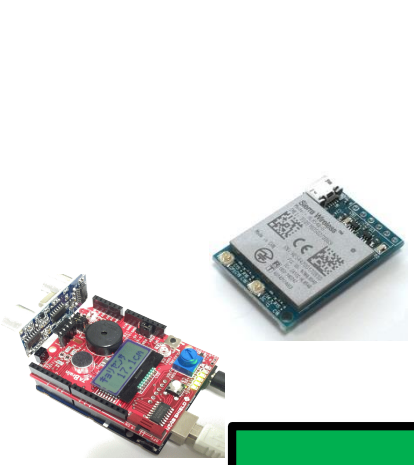
    （※TCP/IPなどを利用してメール送受信なども可能）

3．3GIM V2.2とは②

- 世界最小サイズの3G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「HL8548-G」（JATE/TELEC 取得済）・NTTドコモ（IOT取得済）を採用
  - サイズは 35mm × 25mm × 7mm , 重量は7.5ｇ と超小型な３G通信モジュール
  - 32ビットARMマイコン（LPC812M101JTB16）を搭載、独自のファームウェアを開発できる
  - GPS/AGPSが利用可能
  - 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

HL8548-Gの主な仕様	
UMTS	Band 1/6/19
EDGE/GPRS/GSM	850/900/1800/1900 MHz
GPS	GPS（1575.42MHz） GLONASS（1602MHz）、 Assisted GPS
Speed	7.2Mbps(Download)/5.76Mbps(Upload)
その他	JATE　取得済み(docomo IOT取得済み)
サイズ	23㎜×22㎜×2.5㎜
動作温度	-30℃ ～ 70℃

2．3GIMとは ③



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

3GIM V2.1

HL8548-G
(通信モジュール)

アプリケーション
<gw3g(V3.1)>

インターネット
接続
GPS機能ほか

NPO法人オープンワイヤレス
アライアンスが提供

IoTAB シールド V3.0

UART

Power

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

Arduino アプリケーション

Arduino ライブラリ <a3gim(R4.0)>

SoftwareSerial ライブラリ

Arduino ブートローダ

**USB or
外部電源**

2．３GIMとは④

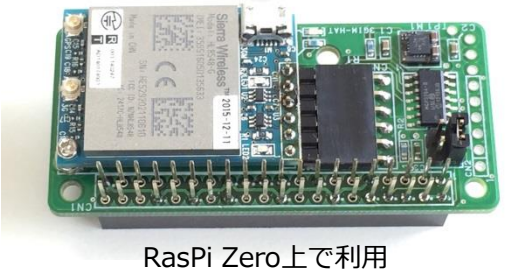
- Arduinoで簡単に3G通信を行うことができる３G通信モジュール
  - FOMA系の通信をサポート（docomo、MVNOのIIJmio、IIJmobile、ｂmobile、DTI、ソネット、ソラコムなど多くの種類のSIMカードが利用可）
  - シエラワイヤレス製の３G通信モジュール（HL8548-G）を搭載し、Assisted GPS機能も利用可能

- 低いCPU性能で少ないメモリを持つArduino上で利用できるよう、高機能なAPIをライブラリ”a3gim”として提供

  ※海外でも３GIMが販売されているが、ATコマンドで利用：技術ハードルは高い（MCPCのモバイルM2Mワーキンググループの方によると「多くの技術者がATコマンド習得で挫折している」とのこと）

- ライブラリが提供する予定の機能は、下記の通り：
  - SMS送受信(send, receive, check, onReceived)
  - Web通信(HTTP GET/POST)
  - GPS位置取得(GPS Standalone, AGPS)
  - TCP/IP通信(connect, disconnect, read, write)
  - その他（時刻やIEM取得等）

３GIMはRaspberryPiでも利用可能
（３GIM HAT＋３GIM）



RasPi Zero上で利用

RasPi ３B 上で利用

3．３GIM用 a3gimライブラリの概要

HL8548-Gとは

HL8548-Gは、小型の３G通信モジュールで、その特徴は
▼ シエラワイヤレス社の３G通信モジュール（JATE/TELEC取得済）
サイズは 23mm × 22mm × 2.5mm、重量は4.5ｇ（超小型）
携帯向けに設計されたモジュールであり，消費電力が低い
シエラ ワイヤレス モジュール製品 - 株式会社アルティマ

HL8548-Gの主な仕様	
無線周波数	800/850/900/1900/2100 MHz
GPS	Standalone GPS, AGPS
Speed	(UL)5.76Mbps/(DL)7.2Mbps
その他	JATE/TELEC　取得済み
動作温度	-45℃ ～ 85℃

Arduino用ピン接続

ピン	用途	補足
Vin	電源供給	電源切替SWにより切り替え可能
Vcc	参照電圧	
GND	グラウンド	
Tx	UARTのTxD	ソフトウェアシリアルとして使用
Rx	UARTのRxD	同上

動作環境

項目	動作環境	補足
Arduino	UNO	
	Leonard	特殊設定
	Pro(5V)	
	Pro(3.3V)	
	Mega 2560/ADK	特殊設定
IDE	バージョン 1.6 以降	1.6.8以上を推奨
電源	USB	800mA以上の供給能力が必要
	ACアダプタ(5V用)	7～9Vで1A以上のものを推奨
	ACアダプタ(3.3V用)	5～6Vで1A以上のものを推奨

※Arduino 互換機の GR-SAKURA（ルネサス製）でも稼働

ライブラリ概要

３GIMでは、Arduino側から利用する主なライブラリ群
を以下に分類分けして紹介します。
（基本は、Arduinoライブラリ標準のものをベースとしています）

機能分類	機能概要	補足
コントロール関連	３GIMの電源制御，初期化等	
ショートメッセージ関連	SMS（ショートメッセージ）の送受信	SIMカード限定による利用
Web関連	GET/POSTのメソッド発行，Tweet	HTTP GET/POST
現在位置取得（GPS）関連	GPS位置情報取得	GPS，AGPS
プロファイル関連	プロファイルの読み書き	
通信その他機能	電波強度，時計，サービス関連	

【注意】a3gimライブラリは継続的にバージョンアップを重ねていく予定でいますので、
インターネットなどによって最新の情報を取得するようにしてください。

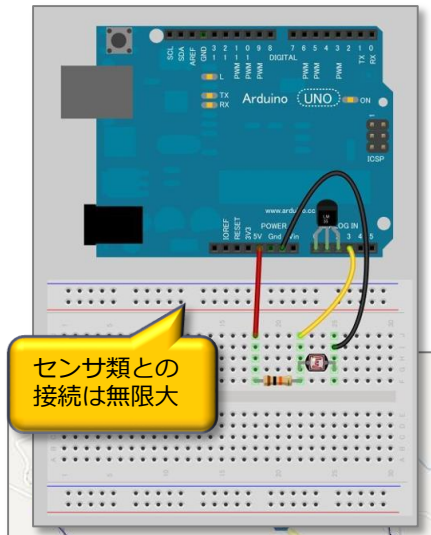
関数例（tweet）

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
	msg：投稿するメッセージ
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

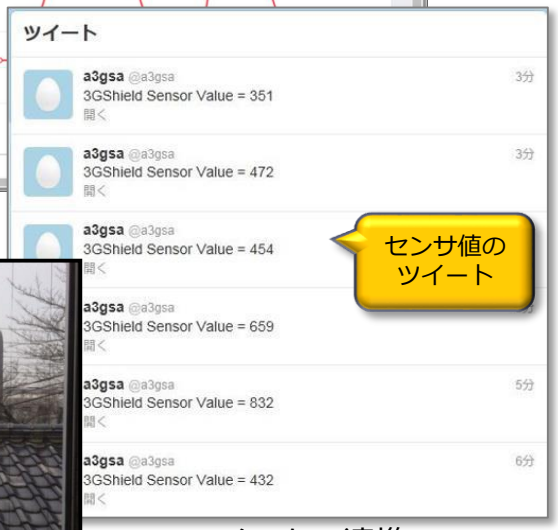
※tweet関数は、Web関連の関数群の中のひとつの機能となります。

4．3GIM V2.2の主な機能

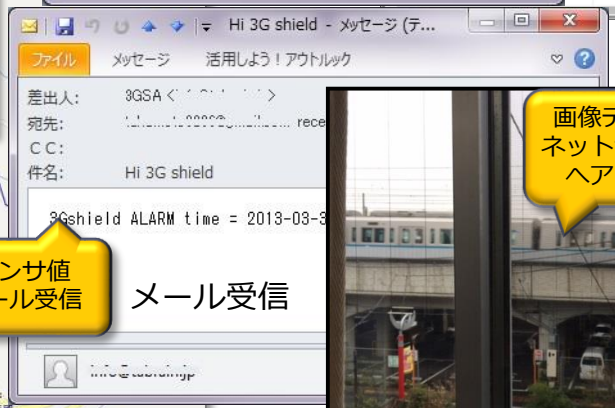
Arduino上のセンサや
アクチュエータなどとの連携



クラウド連携（M2Xは無料で利用可能）



ツイッター連携



画像データアップ

GPS機能

その他：SMS、メール送受信も可能
HTTP技術やTCP/IP技術を応用することで、可能性は無限大

5．3GIM a3gim.h ライブラリー覧表

※ 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）

分類	メソッド名	機能概要	補足
コントロール関係	getStatus※	3Gシールドの状態取得	
	begin※	ライブラリの初期化	
	end※	ライブラリの終了	
	restart※	3Gシールドのリセット	
	start※	3Gシールドの電源ON	
	shutdown※	3Gシールドの電源OFF	
	getIMEI	IMEIの取得	携帯端末固有番号
	setLED	緑色LEDのON/OFF	
	setBaudrate	通信速度の設定	
	setAirplaneMode	エアプレーン（機内）モードのON/OFF	
	setResult	通信結果を取得	
インターネット関係（Web）	httpGET※	GETメソッドの要求	https取得も可能
	httpPOST	POSTメソッドの要求	
	tweet※	Twitterへの投稿	＊
インターネット関係（TCP）	connectTCP※	TCPコネクションを接続する	
	disconnectTCP※	TCPコネクションを切断する	
	writeBegin	シリアル通信で直接書込み	
	read※	データを読み込む	
	write※	データを書き出す	
位置情報取得（GPS）関係	getLocation	現在位置の取得	内蔵GPSを使用
	getLocation2	Assisted GPSを使った現在位置の取得	
その他ライブラリ	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(gw3gアプリ)のバージョンの取得	
プロファイル関係	setDefaultProfile	デフォルトプロファイル設定	
	getDefaultProfile	デフォルトプロファイル取得	
ATコマンドパススルー	enterAT	ATコマンドパススルーモード	

6．ワイヤレス通信技術の普及に向けて

NPO法人オープンワイヤレスアライアンス設立<2013年3月設立>

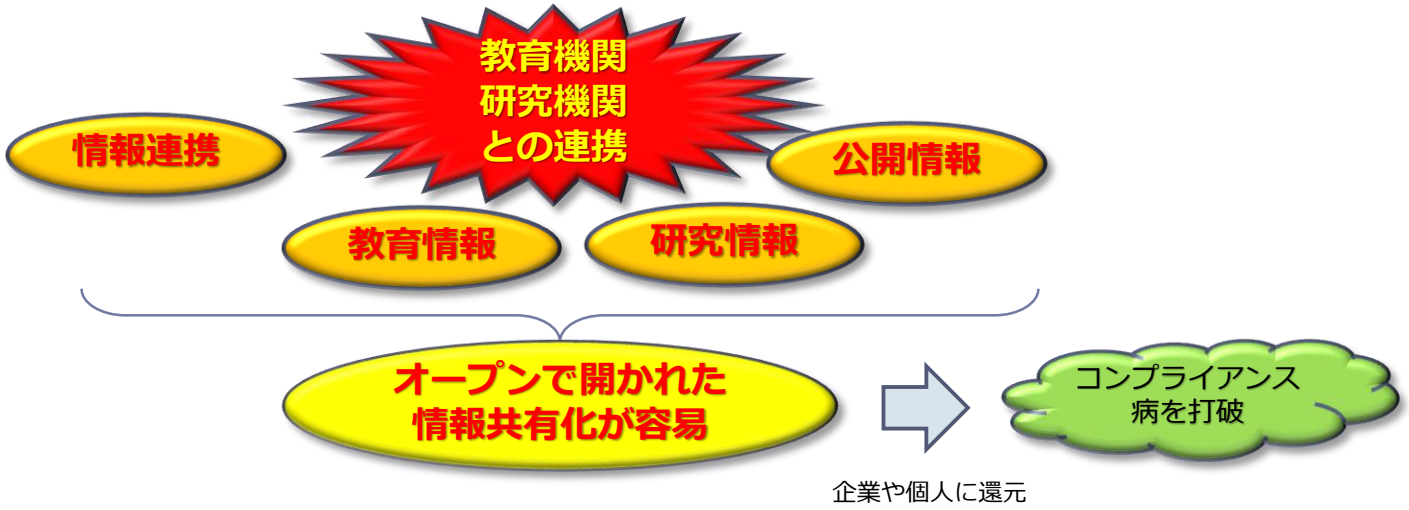
ワイヤレス通信技術の普及を目的

① 最先端で高度な技術をArduino/RaspberryPi上で簡単に学べる

② あらゆるものをネットでつなぐアイデアの場を提供

③ センサ技術との連携で将来に役立つデータを収集・分析

Open Wireless Alliance



7．他の無線機器との比較

多量生産でないと
融通が利かない
メーカへ技術流出

M2Mでの展開でのメリット

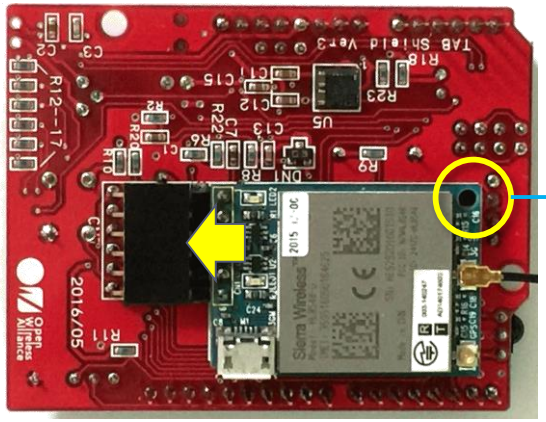
比較項目	3GIM	モバイル・ルータ機器	スマホ
通信エリア	○ 広域な3G通信網のみ （将来はLTEも視野に）	○ 3G+LTE / WiMAX	○ 3G+LTE / WiMAX
通信速度	△ （3G利用SIMカードに依存）	○ （高速だと通信費大）	○ （高速だと費用大）
通信費用	○ 安価なMVNOのSIMが利用可能	× 通信費はキャリア依存	△ スマホのキャリア依存（SIMフリーであれば安価なSIM利用可能）
消費電力	◎ 省電力化の技術対策が容易	△ 機器依存、省電力化が難しい	× 省電力化は難しい
開発環境（試作）	○ 収得しやすい開発環境	× 他に機器利用が必要	△ Androidなどの環境利用できるが技術的レベルは高い （M2Mではスマホの画面は不要）
機器量産	◎ 安価に抑えることが可能 （アライアンス対応）	△ メーカ依存 （多量生産でないと融通不可）	△ メーカ依存
M2Mでの利用評価	◎ 短時間に高機能の試作品実現可能 （量産も容易） 中小企業でも導入開発が可能	△ 他の機器との連携で価値が出るもの （機器開発・ソフト開発などはメーカに依存）	× M2Mでのスマホ利用は不向き

第11章
3GIM + IoTABシールド

1．３GIMをIoTABシールドに設置

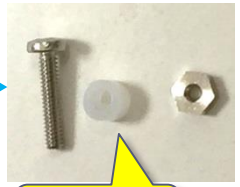
３GIM は、 IoTABシールドの裏面に接続配置します

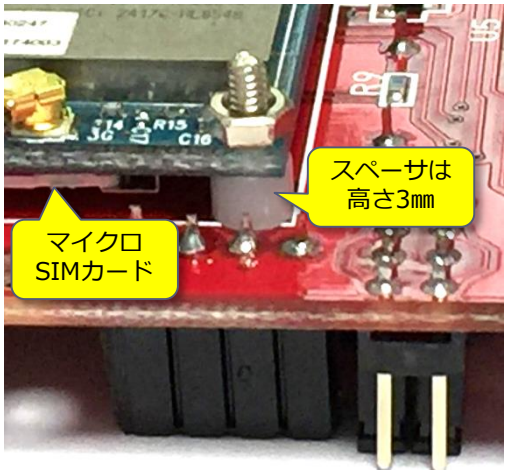
L型ピンを半田付け

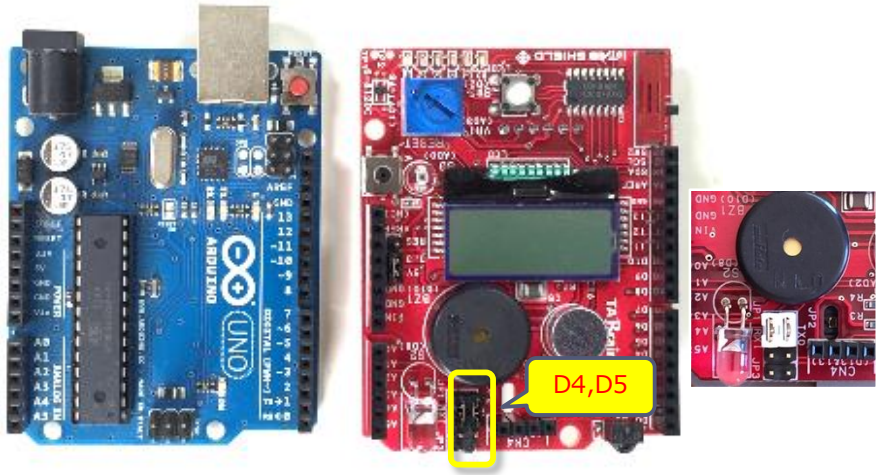


マイクロSIMカード挿入後、ネジ止め

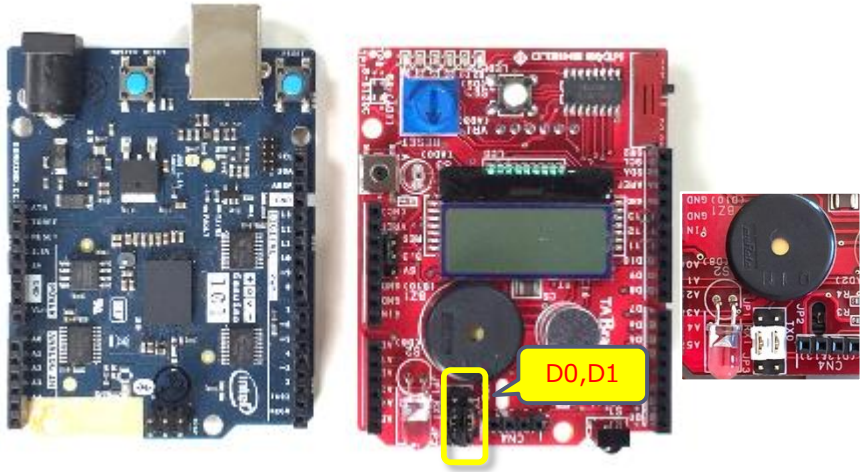
2㎜×10㎜ネジ利用
スペーサは3㎜利用







Arduino UNOに搭載する場合



Genuino101に搭載する場合

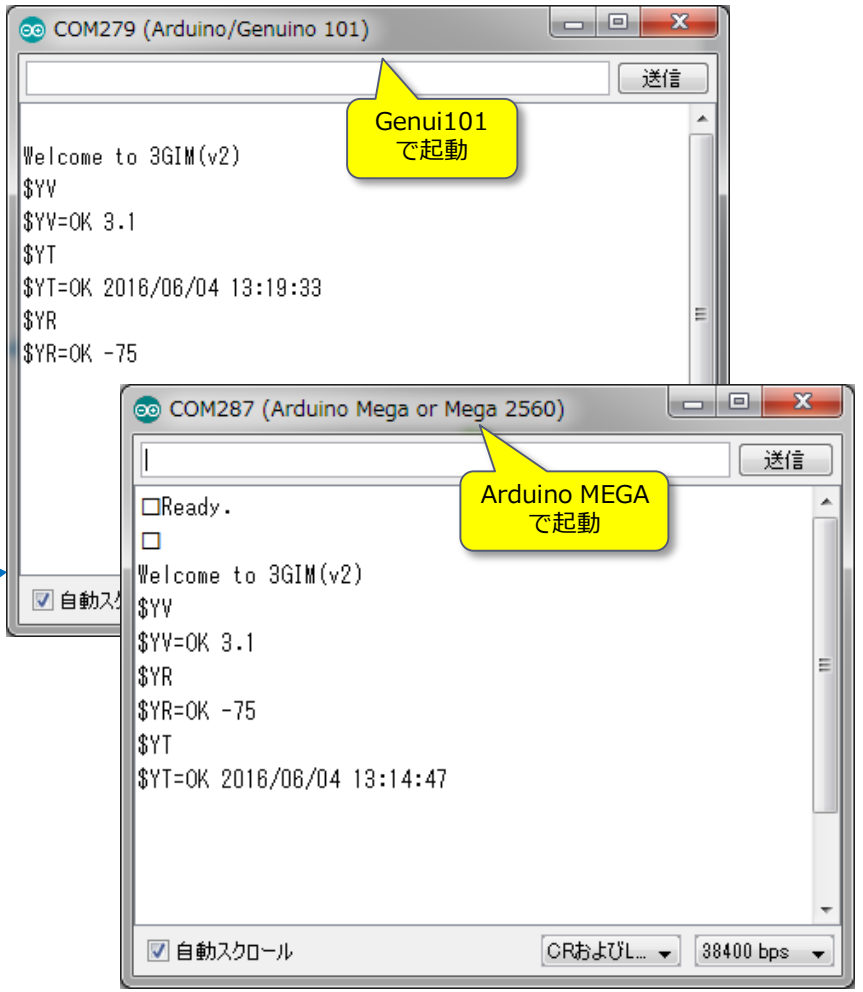
2．３GIM+IoTABシールド接続テスト

モニター出力画面を使った簡単なスケッチを動かしてみましょう。

以下のスケッチは、Arduino UNOで動きます。Genuino101およびArduinoMEGAで動かす場合は、先頭2行をコメントしてください。

```
#include <SoftwareSerial.h>
SoftwareSerial Serial1(4,5);
const unsigned long baudrate = 9600;

void setup() {
  Serial.begin(baudrate);
  Serial1.begin(baudrate);
  pinMode(7,OUTPUT);
  digitalWrite(7,HIGH); delay(100);
  digitalWrite(7,LOW);
  Serial.println("Ready.");
}

void loop() {
  if (Serial1.available() > 0) {
    char c = Serial1.read();
    Serial.print(c);
  }

  if (Serial.available() > 0) {
    char c = Serial.read();
    Serial.print(c);    // Echo back
    Serial1.print(c);
  }
}
```

IoTABS4_monitor_3GIM.ino



3．メール送信テスト

IoTABシールド上の温度センサ値をメール送信してみましょう。

温度センサ値を送るメール送信スケッチ（Genuino101の場合）

```
//#include <SoftwareSerial.h>
//SoftwareSerial Serial1(4,5);
String server= “http://*******/sendmail.php?email=" ;
String email=“*******@****.jp";

void setup() {
 Serial.begin(BAUDRATE);
  Serial1.begin(BAUDRATE);
  pinMode(7,OUTPUT); digitalWrite(7,HIGH); delay(100);
  digitalWrite(7,LOW);
  Serial.println("Ready");
  String str="";
  do{ str=Serial1.readStringUntil('¥n');
  } while(str.indexOf("3GIM")<0);
  Serial.println(str);
  delay(2000);// 待機（重要）

  float temp= 207.26 - 0.3923 * analogRead(A1);// IoTABシールド 温度センサ値
  digitalWrite(3,HIGH);
  String sr = "$WG "+ server+email +"&cont=temp=%20" + String(temp) + "%20C";
  Serial.println(sr);
  Serial1.println(sr);

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

void loop() {}
```

IoTABS4_sendmail_temp_3GIM.ino

サーバ側 （メール送信PHP：sendmail.php）

```
<HTML>
<HEAD><TITLE> 3G send Light sensor e-mail </TITLE><HEAD>
<BODY>

<H3>  3G light sensor get </H3>
<?php
 if(mail($_GET["email"], // メール送信先アドレス(to)
        '3GIM sensor E-mail' ,// タイトル
        ' 3GIM send ALARM ' . "¥r¥n" . 'DATE = ' . date('Y-m-d') . '
        TIME = ' . date('H:i:s') . "¥r¥n" . 'cont : ' . $_GET["cont"] ,//本文
        'From: 3GSA<temp@tabrain.jp>' . "¥n" .//送信元アドレス
        'X-Mailer: PHP/' . phpVersion()))
{     echo ‘<B>SUCCESS TO SEND</B><BR>’; }
else
      echo '<B>faile to mb_send_mail</B><BR>';
 ?>

</BODY>
```

4．ツイート・テスト

IoTABシールド上の温度センサ値をツイートしてみましょう。

温度センサ値をツイートするスケッチ（Genuino101の場合）

```
String server = "arduino-tweet.appspot.com:80/update ";
String token = "<トークン>"; // トークン取得　http://arduino-tweet.appspot.com/

void setup() {
  Serial.begin(38400);
  Serial1.begin(38400);
  pinMode(7,OUTPUT); digitalWrite(7,HIGH); delay(100);
  digitalWrite(7,LOW);
  String str="";
  do{ str=Serial1.readStringUntil('¥n');
  }while ( str.indexOf("3GIM")<0 );
  Serial.println(str);
  delay(2000);
  float temp= 207.26 - 0.3923 * analogRead(A1);
  str="$WP http://" + server + "¥"token=" + token +"&status=temp=" + String(temp) + " C ¥"";
  Serial.println(str);
  Serial1.println(str);
  do{ str = Serial1.readStringUntil('¥n');
  } while(!str.startsWith("$WP"));
  Serial.println(str);
  Serial1.println("$YE");
  Serial.println("END");
}
void loop() {}
```

※トークンは、あらかじめツイッター登録したIDで取得します。Tweet Library for Arduino【TLA】を利用します。http://arduino-tweet.appspot.com/

シリアルモニタ画面

ツイート内容

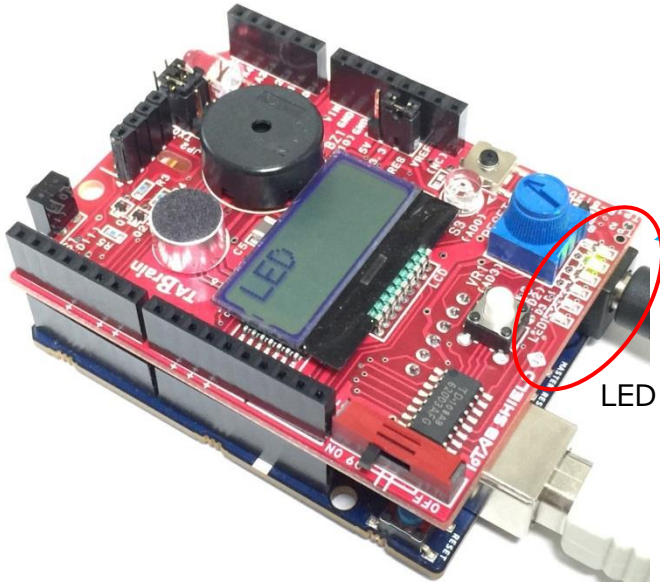
演習課題
HTML＋PHPによる３GIM操作

遠隔操作で３GIM上の入出操作＋メール送信

1．LEDを点滅させる

＊ARDUINO+３GIM上にあるLEDを遠隔操作で
On/Off させる

課題：LED点滅を遠隔からOn/Offする

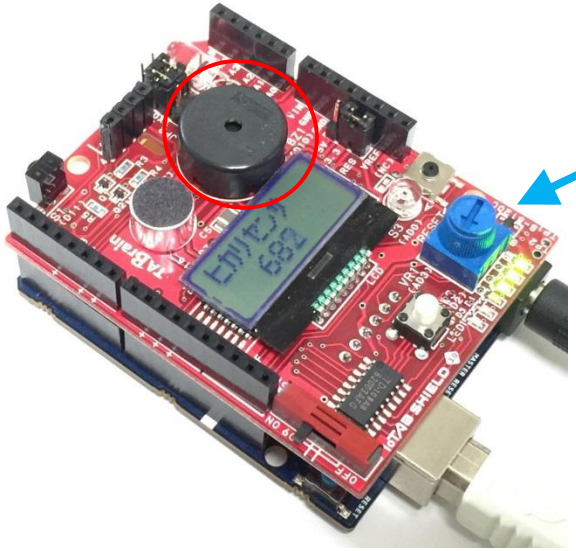


LED１～６はD3～D8対応

2．ブザー（スピーカ）を鳴らす

＊ARDUINO+３GIM上にあるスピーカを遠隔操作で５秒ほど鳴らす

課題：スピーカを5秒ほど鳴らす

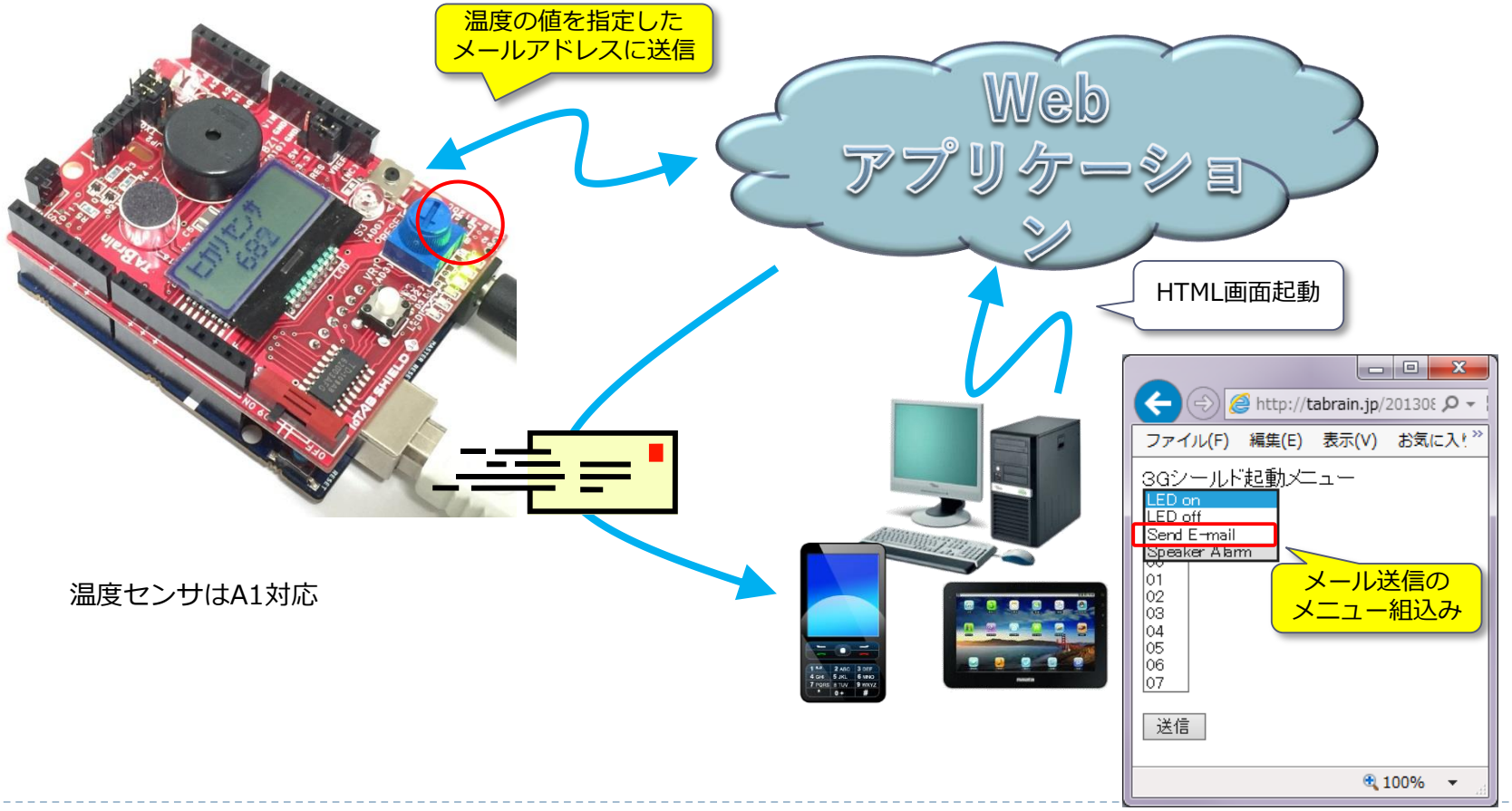


スピーカはD10対応

3．温度センサの値をメールで受信する

＊ARDUINO+３GIM上にある温度センサ値を自分の
メールアドレスに送る

課題：指定したメールアドレスに「温度センサーの値」返す



温度センサはA1対応

4．総合メニューによる遠隔操作

＊IoTABシールド上にある温度センサ値を自分のメール
アドレスに送る

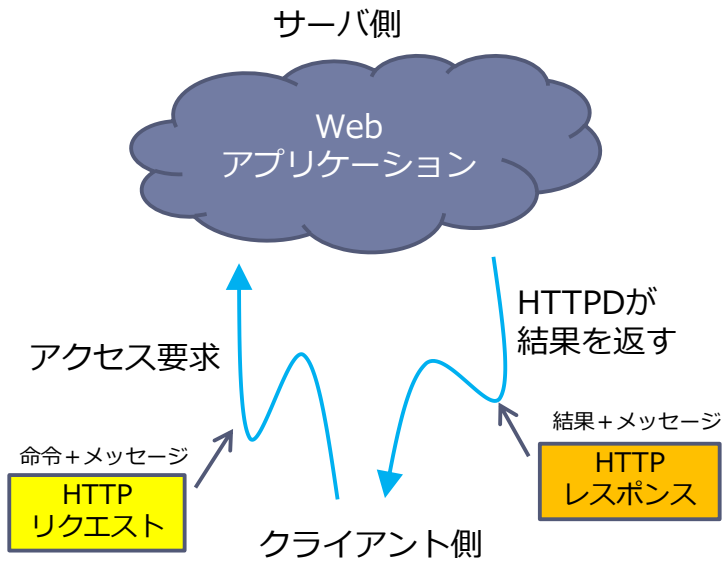
5．プログラミング

演習課題の回答例

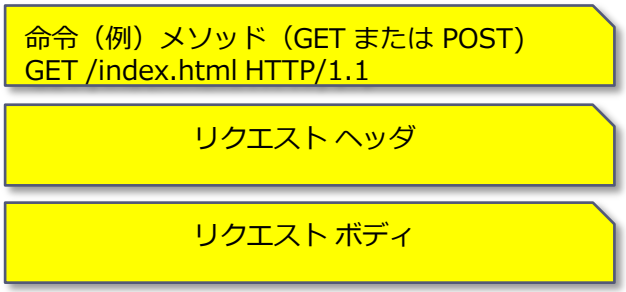
遠隔操作で３GIM上の入出操作＋メール送信

1．HTTPの基礎（1）

リクエストとレスポンス



HTTPリクエスト

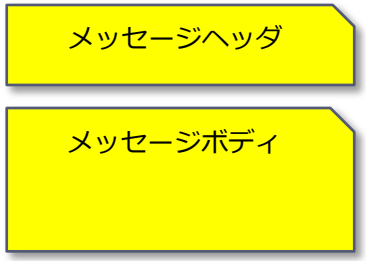


GETメソッドによるデータ送信（パラメータを渡す方法）
http://www.***.co.jp/index.html?para1=123&para2=345
※ ここでは、変数「para1」に123を、「para2」に345を値として渡す

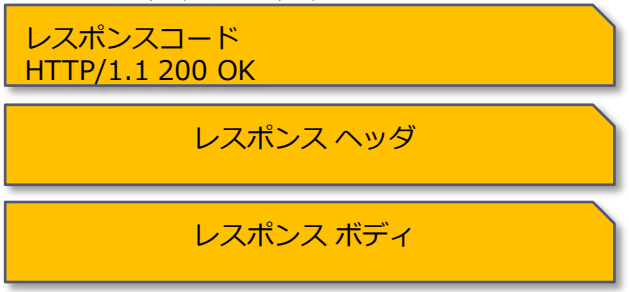
POSTメソッドによるデータ送信（パラメータを渡す方法）
HTMLフォームで、「method」属性にPOSTを与えて実現

```
<FORM action="http://www.tabrain.jp/form.php"  method="POST">
＊＊＊＊
＊＊＊＊
</FORM>
```

メッセージの構造



HTTPレスポンス



1．HTTPの基礎（2）

POSTによるデータ送信とレスポンス

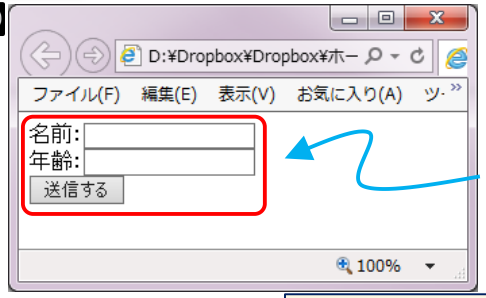
クライアント側

①
```
<FORM action="http://www.tabrain.jp/form.php" method="POST">
 名前：<INPUT type="text" name="name" /><BR>
 年齢：<INPUT type="text" name="age" /><BR>
<INPUT type="submit" name="buttonName" value="送信する" />
<FORM>
```

サーバ側に
あっても同じ

このファイル名を test.html として
IEなどのブラウザで実行すると

②



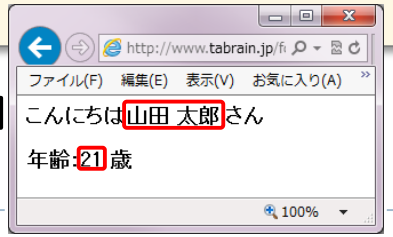
名前と年齢を入力し
「送信する」ボタン
を押す

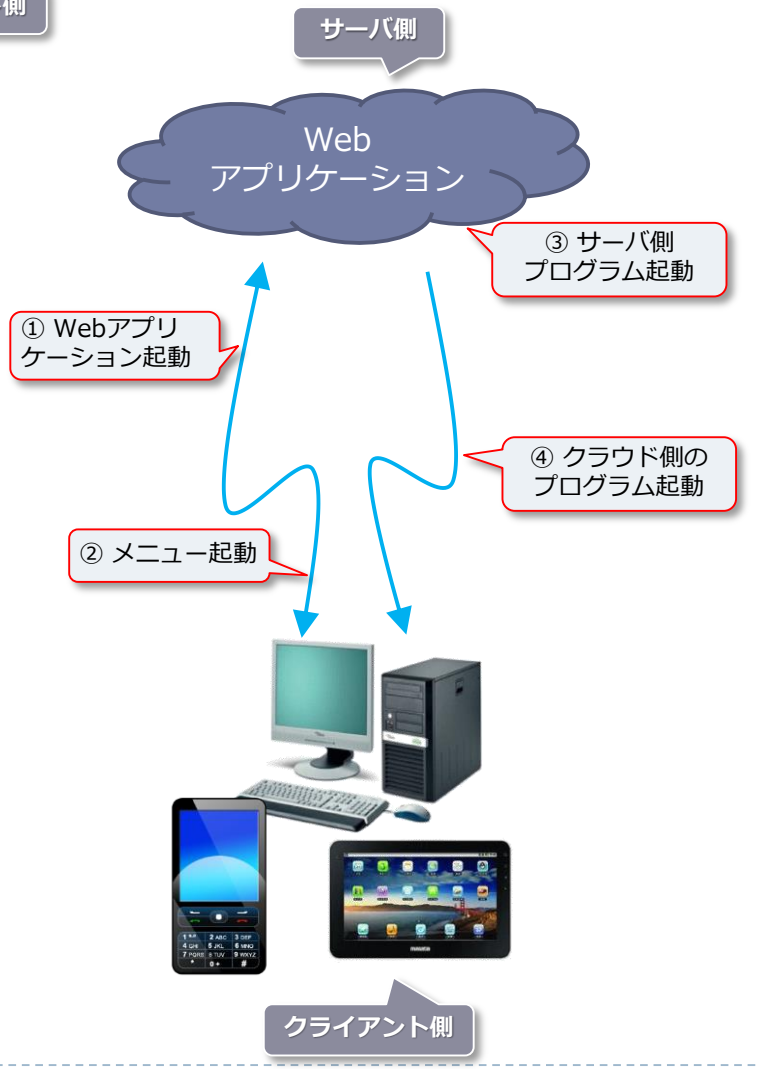
サーバ側

サーバ側のプログラムが起動（この場合、
http://www.tabrain.jp/form.php）

③
```
<HTML>
<BODY>
<H3>  こんにちは<?= $_POST["name"]  ?> さん </H3>
<H3>  年齢：<?= $_POST["age"] ?> 歳 </H3>
</BODY>
</HTML>
```

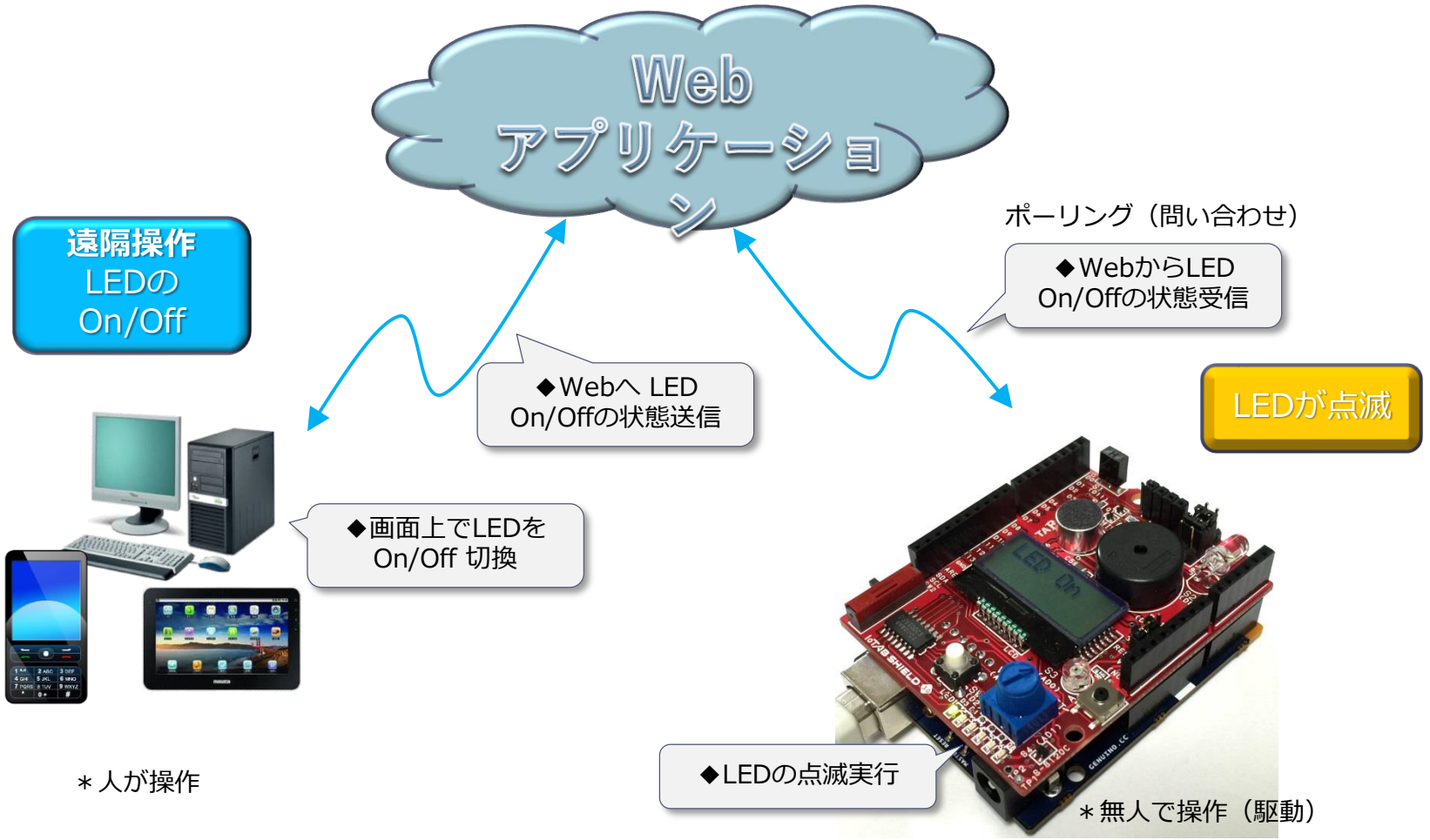
④





2．３GIM上のLED配線と課題解決

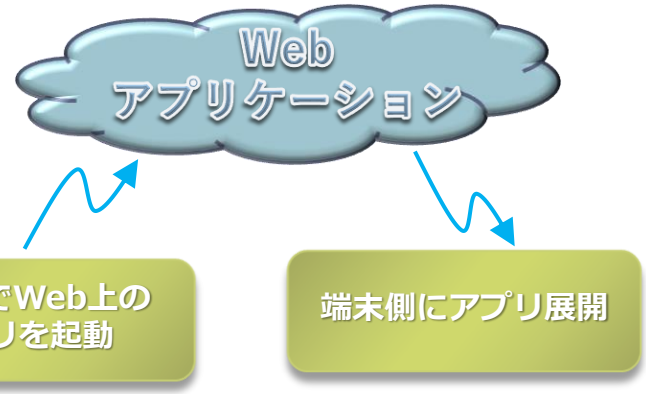
＊端末（PC、タブレット、スマホなど）から、遠隔地にある３GIM上のLEDを点滅させる



3．PC側のWeb上のLED On/Off選択

http://tabrain.jp/20130801/3gform**.html

フリーエディタ：TeraPad
http://www.forest.impress.co.jp/library/software/terapad/



3gfrom.php

起動プログラム

```
<form action = "3gmakefile.php" method = "post">
<p>  3 GIM起動メニュー <br>
<select name="cmd" id="cmd">
<option value="LED on">LED on</option>
<option value="LED off">LED off</option>
</select></p>
<p>番号：<br>
<select name="no" size ="8">
<option value="00">00</option>
<option value="01">01</option>
<option value="02">02</option>
<option value="03">03</option>
<option value="04">04</option>
<option value="05">05</option>
<option value="06">06</option>
<option value="07">07</option>
</select></p>
<input type="submit" value="送信">
</form>
```

メニュー画面

個人ID

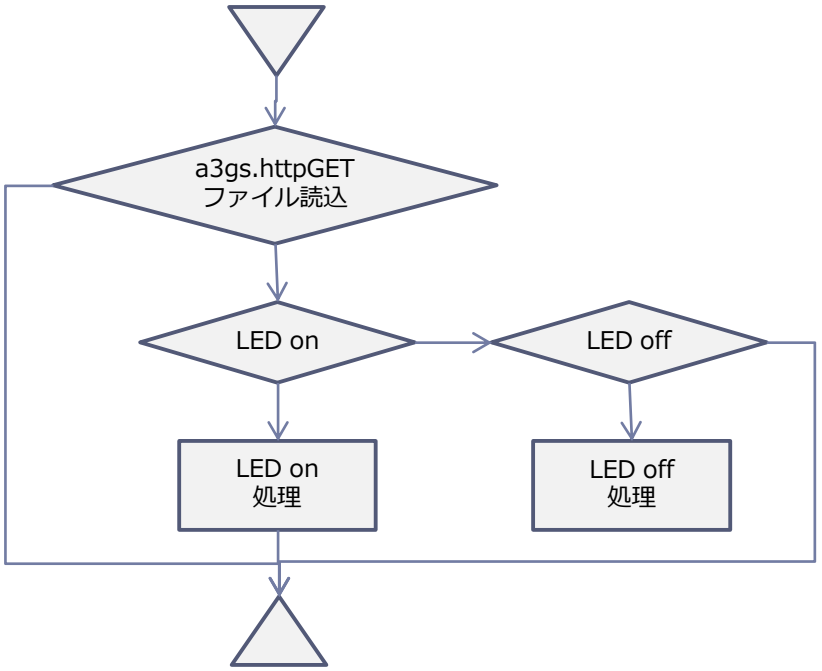
3gmakefile.php

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<?php
// echo  $_POST["cmd"];
$x=$_POST["cmd"] . "";
$fn1="temp" . $_POST["no"] . ".xxx";
$fp=fopen($fn1,'w');
fputs($fp, $x);
fclose($fp);
$fn2 = '3gsa' . $_POST["no"] . '.xxx';

copy($fn1 , $fn2);
echo '実行コマンド = [' . $x . ']';

?>
<br>
<input type="submit" name="buttonName" value="戻る" onClick="history.back()" />
```

temp**.xxx作成

3gsa**.xxx作成

4．３GIM上のLED配線と課題解決



3gsaLED.ino

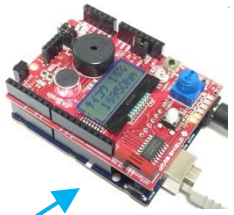
```
#include <SoftwareSerial.h>
#include "a3gs.h"

const char *server = "www.tabrain.jp";
const char *path = "/20130801/3gsa01.xxx";
int port = 80;

void setup()
{
  Serial.begin(9600);
  pinMode(13,OUTPUT);
  Serial.println("Ready..");
  while(!(a3gs.start() == 0 && a3gs.begin()==0)) ;
  Serial.println("Start..");
}
void loop()
{
  int len = 15;
  char res[15];
  if (a3gs.httpGET(server, port, path, res, len) == 0)
{
    Serial.println(res);
    if(strncmp(res,"LED on",6)==0 )  {
      digitalWrite(13,HIGH);
    }
    else if(strncmp(res,"LED off",7)==0)  {
      digitalWrite(13,LOW);
    }

  } else { Serial.println(" httpGET error"); }
  delay(3000);
}
```

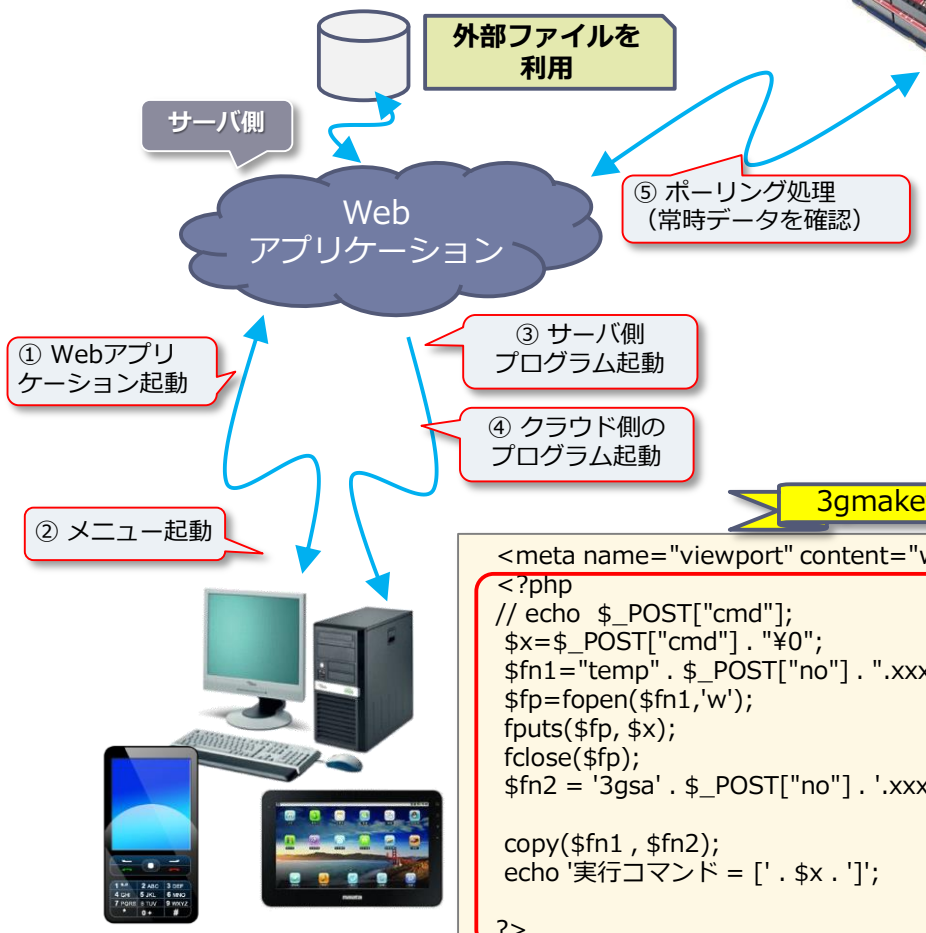
各自IDファイル
に書き替え

各自IDファイル
から読み込み

5．起動画面とファイル出力

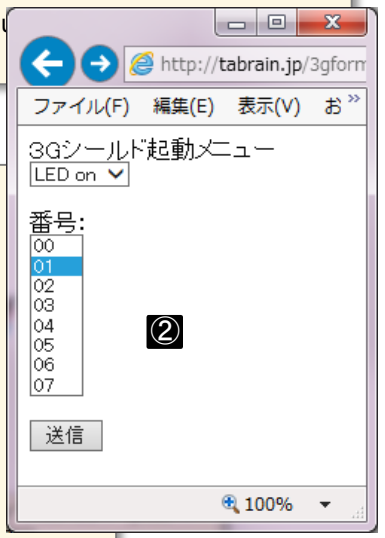
- クライアント＆サーバ操作





3gform*.html

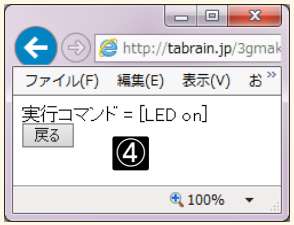
```
<form action = "3gmakefile.php" method = "post">
<p>  3 GIM起動メニュー <br>
<select name="cmd" id="cmd">
<option value="LED on">LED on</option>
<option value="LED off">LED off</option>
</select></p>
<p>番号：<br>
<select name="no" size ="8">                    ①
<option value="00">00</option>
<option value="01">01</option>
<option value="02">02</option>
<option value="03">03</option>
<option value="04">04</option>
<option value="05">05</option>
<option value="06">06</option>
<option value="07">07</option>
</select></p>
<input type="submit" valu
</form>
```



3gmakefile.php

```
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <?php
 // echo  $_POST["cmd"];
  $x=$_POST["cmd"] . "¥0";
  $fn1="temp" . $_POST["no"] . ".xxx";
  $fp=fopen($fn1,'w');
  fputs($fp, $x);
  fclose($fp);
  $fn2 = '3gsa' . $_POST["no"] . '.xxx';

  copy($fn1 , $fn2);
  echo '実行コマンド = [' . $x . ']';

 ?>
 <br>
 <input type="submit" name="buttonName" value="戻る" onClick="history.back()" />
```

③



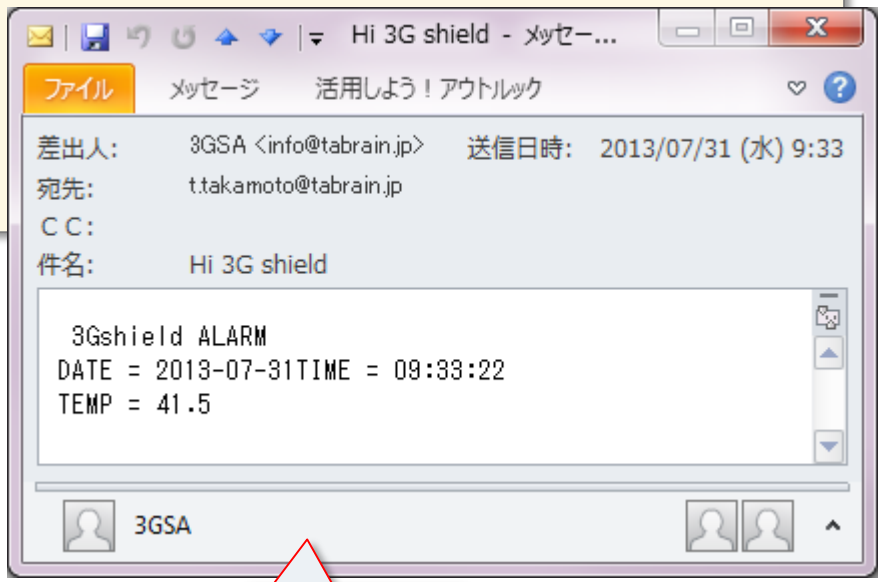
6．PHPによるメール送信

3gsendmail.php

```
<HTML>
<HEAD><TITLE> 3G send e-mail </TITLE><HEAD>
<BODY>

<H3>  3G shield temp get </H3>
<?php
 if(mail($_GET["email"], // to
        'Hi 3G shield' ,// タイトル
        ' 3Gshield ALARM ' . "¥r¥n" . 'DATE = ' . date('Y-m-d') . 'TIME = ' . date('H:i:s') . "¥r¥n" . 'TEMP = ' . $_GET["temp"] ,//本文
        'From: 3GSA<info@tabrain.jp>' . "¥n" .
        'X-Mailer: PHP/' . phpVersion()))

    echo '<B>SUCCESS TO SEND</B><BR>';

else
    echo '<B>faile to mb_send_mail</B><BR>';
 ?>

</BODY>
```

メールアドレスの引数設定

メール本文



画面表示

7．Arduino＋３GIM側のスケッチの解説

```
#include <SoftwareSerial.h>
#include "a3gs.h“

// LM61BIZ output pin: A1
#define LM61BIZ_Pin 1

int port = 80;
int spk=8;


//char res[15];
char path2[100]; //メールアドレス＋温度センサ値バッファ

int getTemp(void)
{
  int mV = analogRead(LM61BIZ_Pin) * 4.88;
  return (mV - 600);
}

void spkAlarm() {
  for(int i=0; i<500; i++) {
    digitalWrite(spk,HIGH);
    delay(3);
    digitalWrite(spk,LOW);
    delay(2);
  }
}
```

温度センサー A0

温度計算式

スピーカ
アラーム

```
void setup()
{
  pinMode(A0, OUTPUT);  // A0(LM61BIZ - GND)
  digitalWrite(A0, LOW);
  pinMode(A2, OUTPUT);  // A2(LM61BIZ - VSS+)
  digitalWrite(A2, HIGH);

  Serial.begin(9600);
  pinMode(8,OUTPUT);
  pinMode(13,OUTPUT);
  while(!(a3gs.start() == 0 && a3gs.begin()==0)) ;
  Serial.println("Start..");
  a3gs.setLED(true);
}
```

A0/A2 GND、Vss+

```
void loop()
{
  static boolean swa=true, swm=true;
  if(swa==false || swm==false) a3gs.setLED(false);
  const char *server = "www.tabrain.jp";
  const char *path = "/20130801/3gsa**.xxx";
  char res[15];
  const int len = 15;
    Serial.print("httpGET:");
if (a3gs.httpGET(server, port, path, res, len) == 0) {
    Serial.println(res);
   if(strncmp(res,"LED on",6)==0 )       {
     swa=true; swm= true;
     digitalWrite(13,HIGH);
   }
   else if(strncmp(res,"LED off",7)==0)  {
     swa=true; swm= true;
     digitalWrite(13,LOW);
   }
   else if(strncmp(res,"Send E-mail",11)==0) {
     if ( swm ) {
       int temp = getTemp();
       sprintf(path2, "/20130801/3gsendemail.php?email=*****&temp=%d.%d", temp/10, temp%10);
       Serial.print("temp httpGET:");
      if (a3gs.httpGET(server, port, path2, res, len) == 0)
        Serial.println( res );
       delay(5000);
     }
     swm = false;
   }
   else if(strncmp(res,"Speaker Alarm",13)==0) {
     if (swa) {   spkAlarm(); } ;
     swa = false;
   }
   delay(100);
  }
  Serial.print(".");
  delay(3000);
}
```

各自IDファイル

LED on処理

LED off処理

温度センサ値
メール送信

メールアドレス

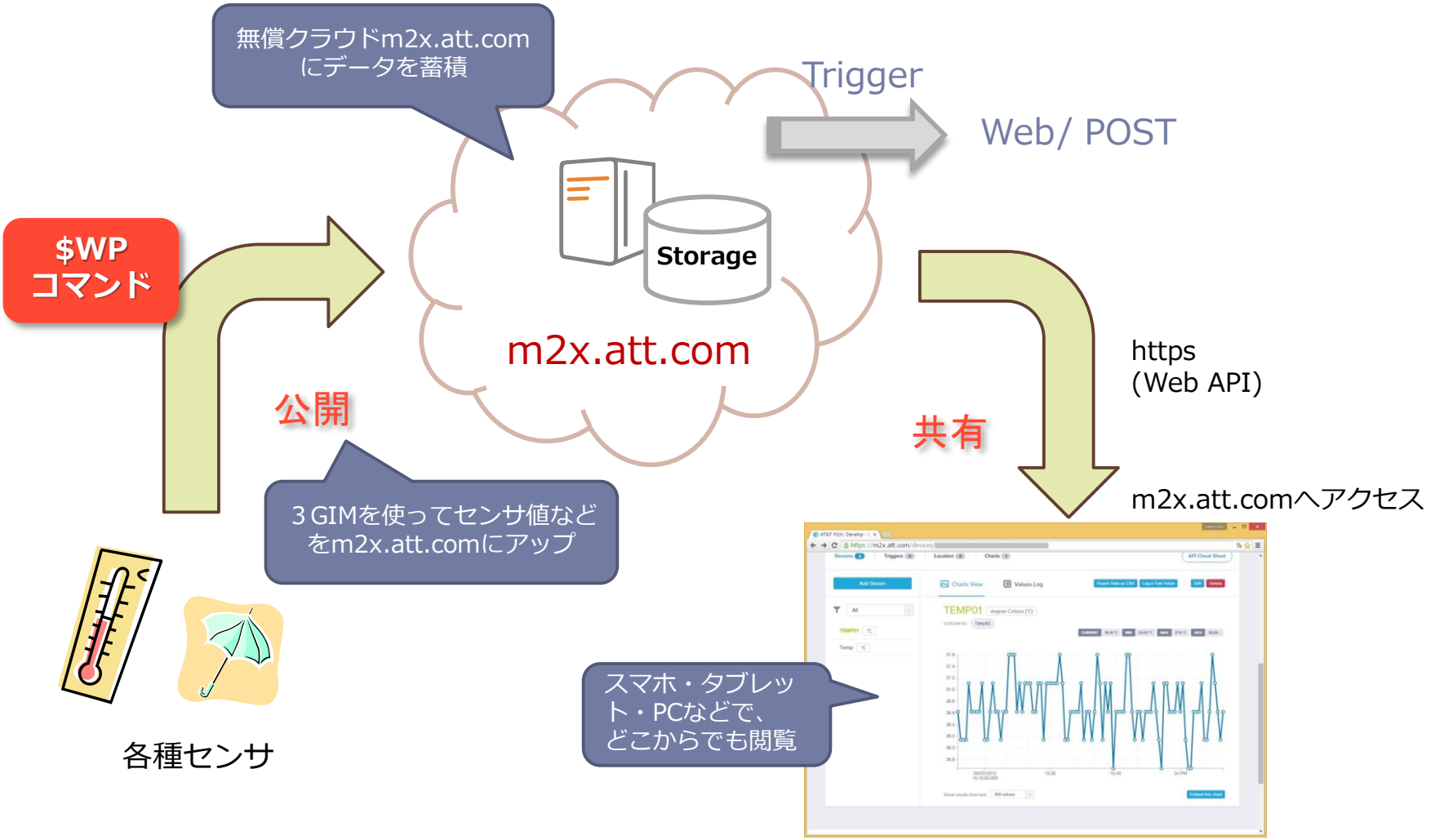
温度センサ値

スピーカ発音

３GIM+クラウド

※３GIMは別売品となっています。

1．M2X（AT&T IoTサービス）の利用イメージ

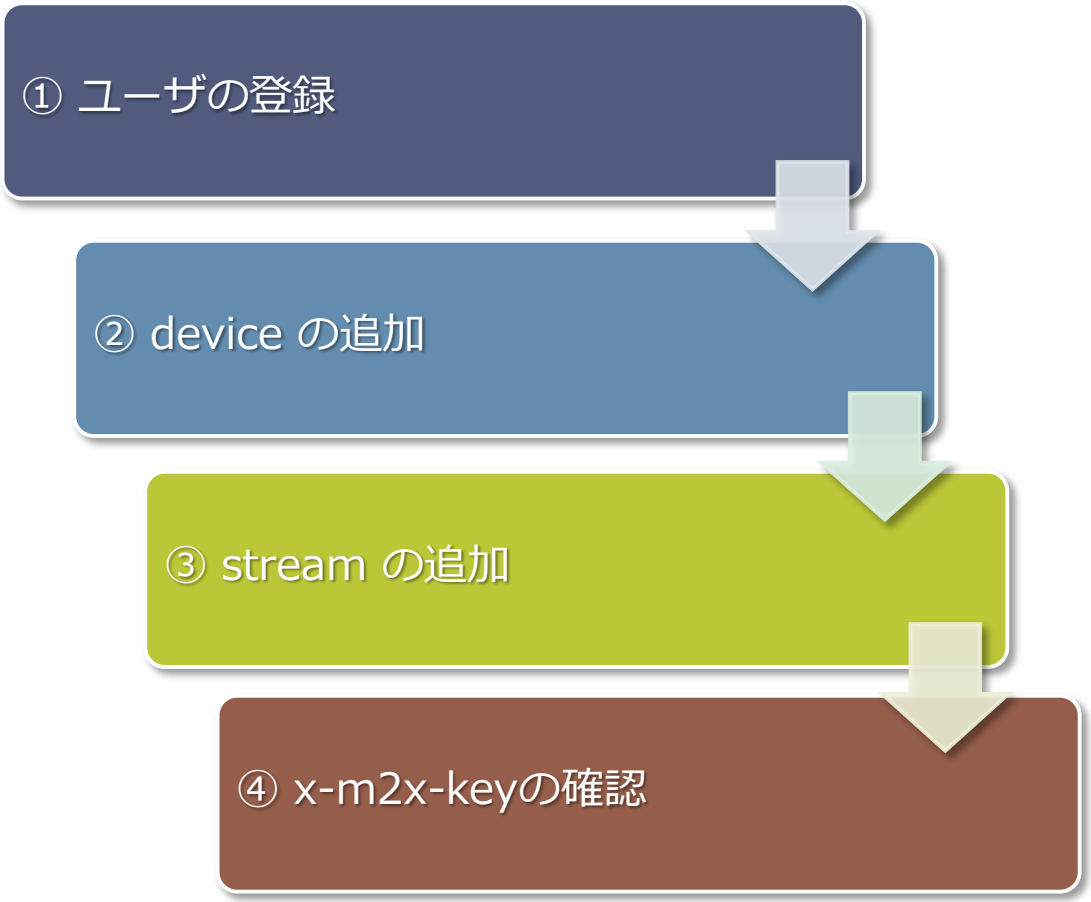


2． M2X（AT&T IoTサービス）の利用手順

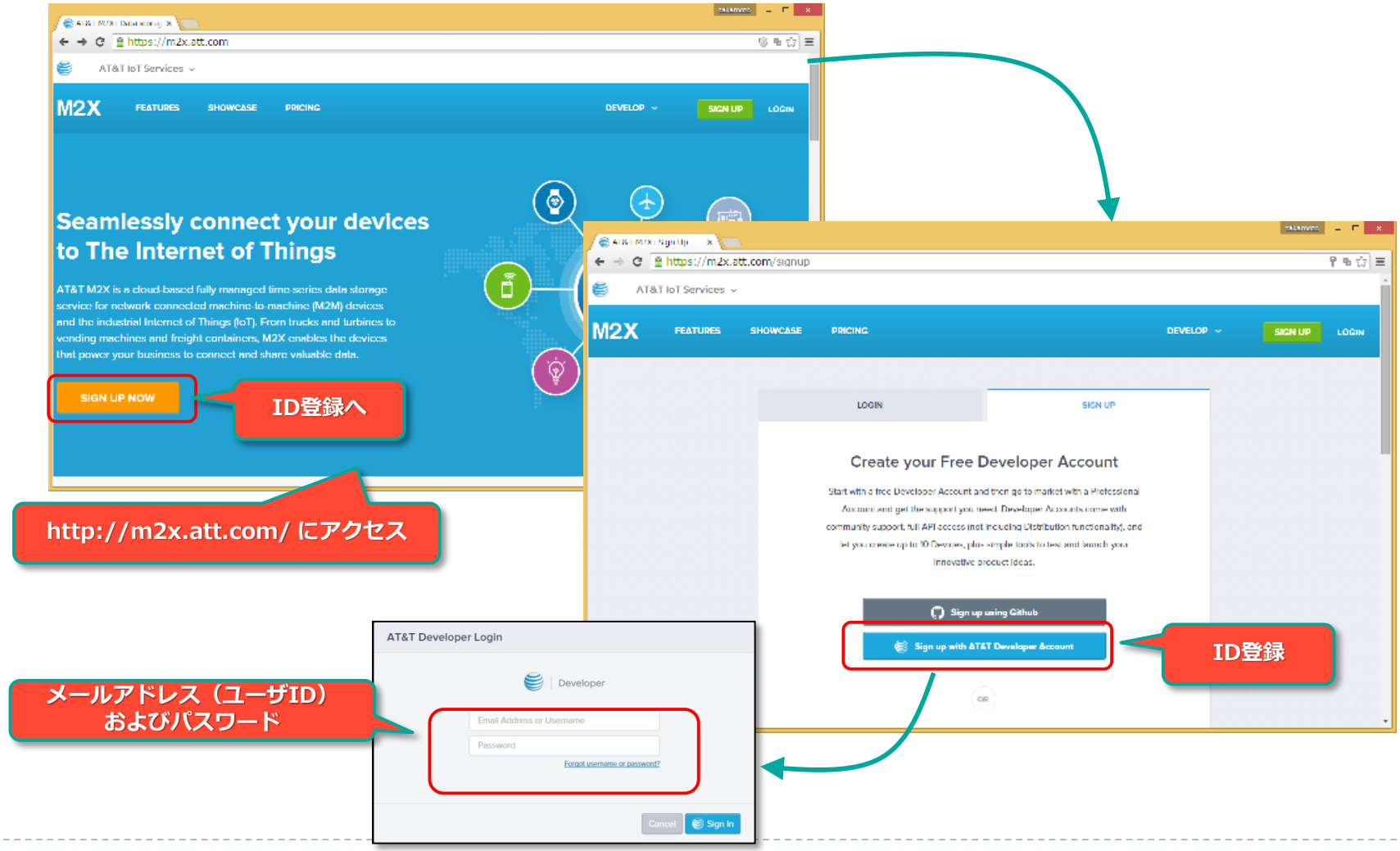
M2X（AT&T IoTサービス）は、2013年から米国通信事業最大手のAT&Tが、M2MおよびIoTビジネスに向けたサービスを開始したものです。
ArduinoやRasberryPi、Mbedなど、多くのオープンソースハードウェアで利用できる環境を提供したものとなっています。
フリーで使え、センサデータの蓄積・グラフ表示、データのダウンロードなどができるようになっています。
ただ、時間設定が、世界標準のみで行なっていて、日本時間での表示が現時点できないのが難点となっています。
ただ、登録の簡単さは
まだ、英語版しかありませんが、グラフ表示やデータのアップやダウンロードだけの利用と考えると、問題なく簡単に使うことができます。

ここでは、本３GIMとセンサなどを使い、この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。

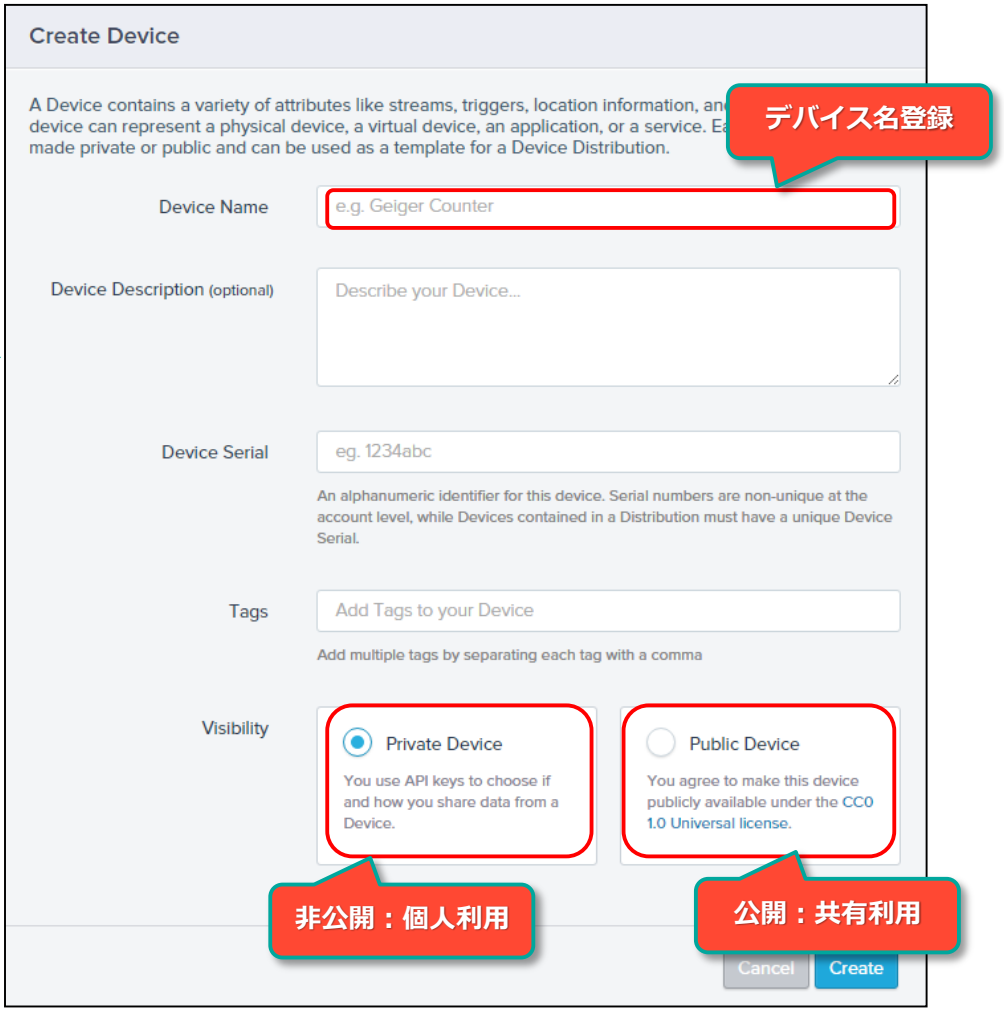


3． M2X（AT&T IoTサービス）のID登録

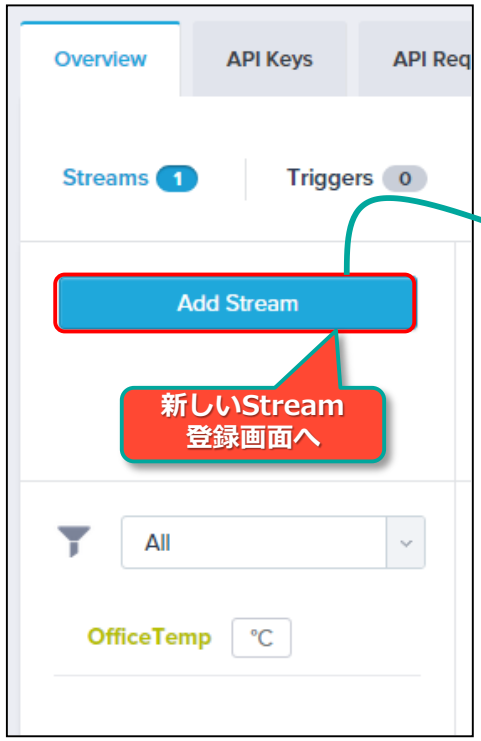


4．デバイス（Device）の作成登録

※ 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。

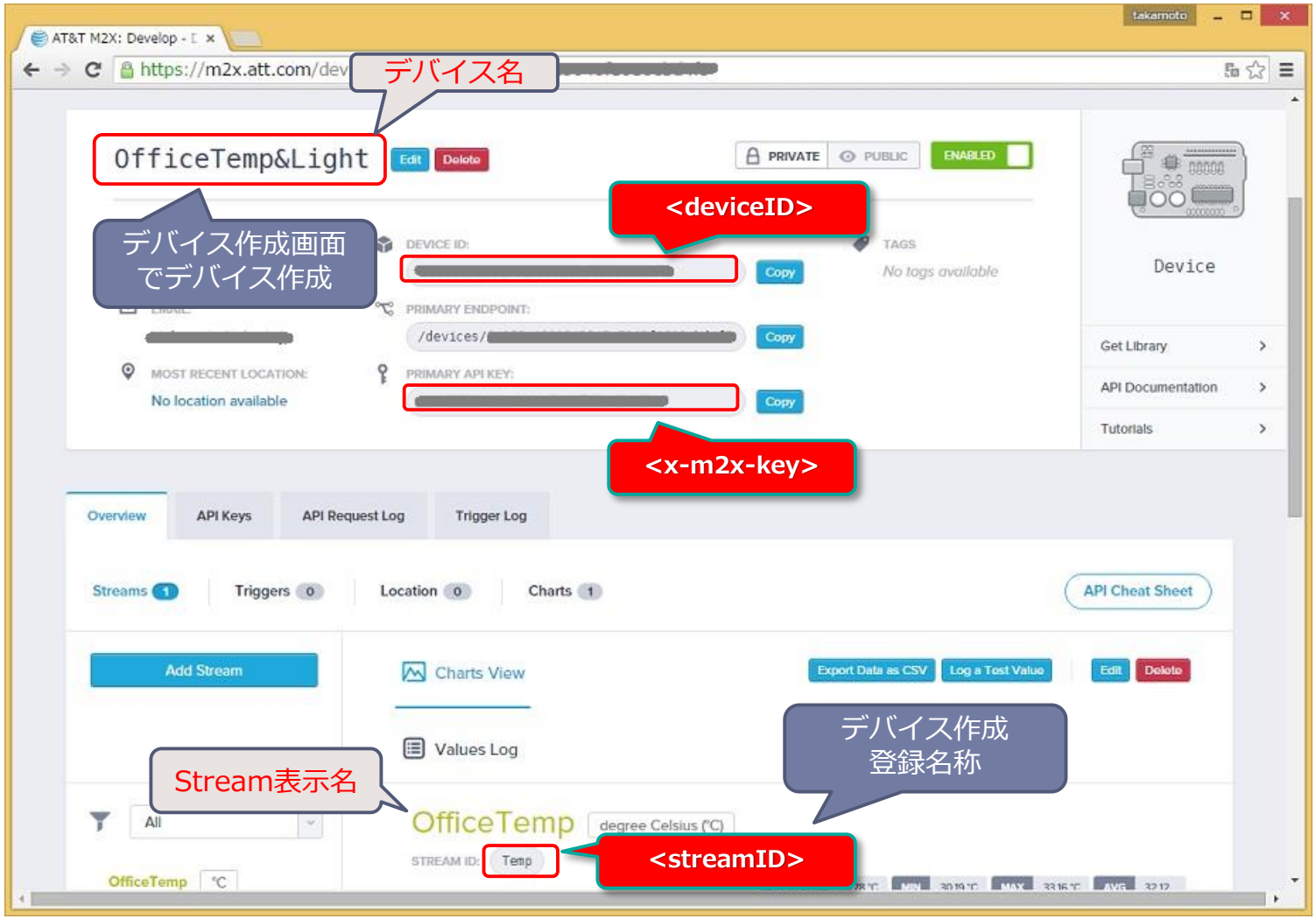


5．ストリーム（StreamID）の作成登録



※ 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

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

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

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

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

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

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

```
//============== 3G setup =================
boolean _3Gsetup() {
  pinMode(7,OUTPUT);
  digitalWrite(7,LOW); delay(1000);
  digitalWrite(7,HIGH); delay(100);
  Serial3g.begin(baudrate);
 //------- 3G module begin & connect ---------
  String str;
  unsigned long tim = millis();
  do{  while(!Serial3g.isListening());
       str=Serial3g.readStringUntil('¥n');
  }while(!(str.indexOf("3GIM")>0) && (millis() - tim) <LIMITTIME);
  if( millis() -tim >= LIMITTIME) {
     return false;
  } else return true;
}
//============== $WP command =================
boolean _3G_WP(String command) {
   Serial.println(command); // debu
  Serial3g.println(command);
  String rstr;
  unsigned long tim = millis();  // time set(ms)
  do{ while(!Serial3g.isListening());
       rstr=Serial3g.readStringUntil('¥n');
       Serial.println(rstr);  //debug print....
  }while(!(rstr.indexOf("$WP=")==0) && (millis() - tim) <LIMITTIME);// $WP return check
  return (rstr.indexOf("$WP=")==0);
}
// Get Date & Time (3GIM command)
//  return --> string "2015-12-23T01:23:45%2B09:00"
String datetime() {
  Serial3g.println("$YT");
  while(!Serial3g.available());
  String dtime = Serial3g.readStringUntil('¥n');
  dtime.replace(" ","T"); dtime.replace("/","-");
  return(dtime.substring(7) + "$+09:00");
}
```

3Gシールド用（電源ON）

電源On状態から3GIM文字が返却されるまで待機

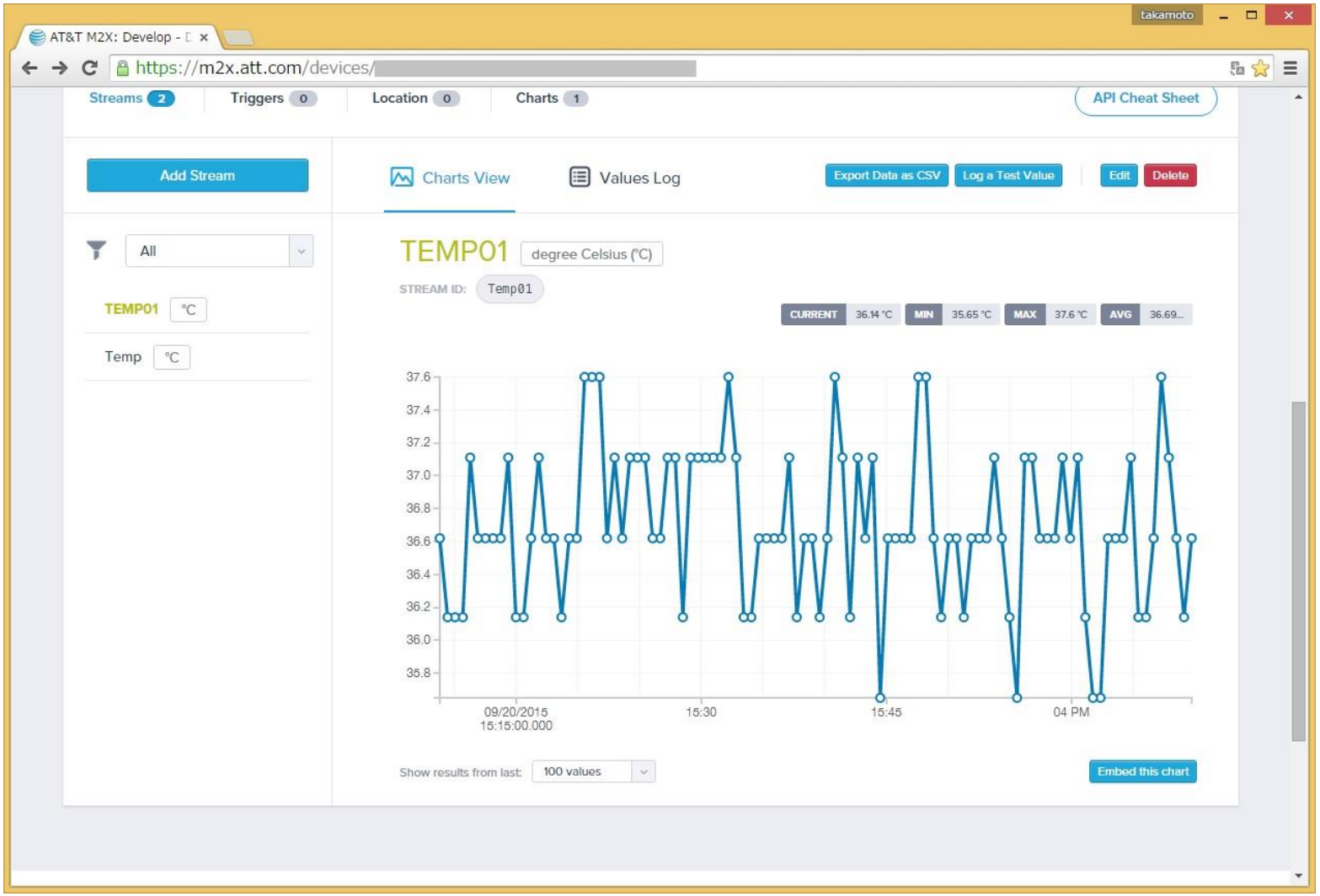
３G接続状態返却

３G POST処理

$WPコマンド返却処理

$YTによる時間取得設定機能

9．M2Xにデータアップした事例



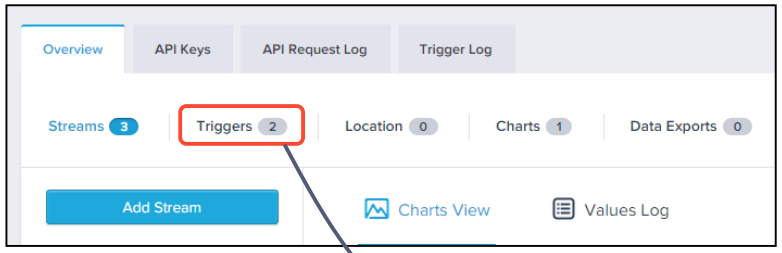
１０．M2Xからトリガーでツイートする方法

M2Xにアップしているセンサの値をトリガーにして、ツイートする方法を紹介

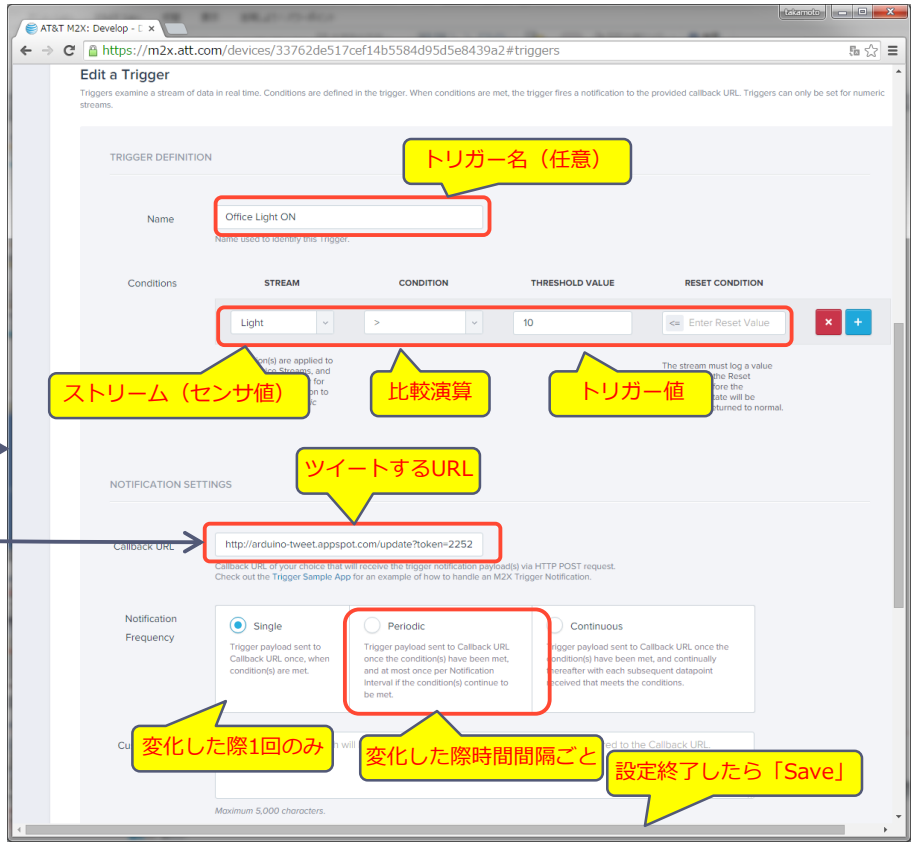
ツイートするURLは、以下の通り

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

トリガー設定の選択



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。

第１２章
IoTシステム開発事例

1．監視・見守り系システム

□今後、独居高齢者の増加にともなう見守りシステムのニーズ増大

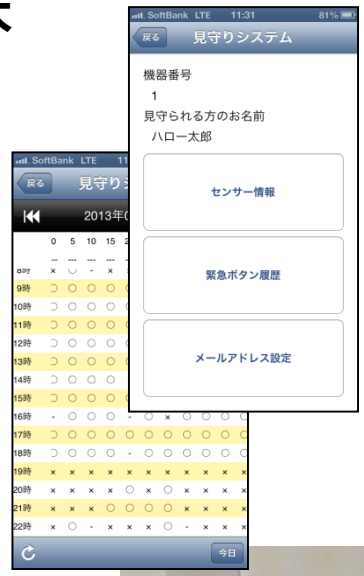
① 遠方にいる親族などにも状況を逐次知らせるシステムが必要
② クラウドサービスによる「いつでも・どこでも」情報把握が可能
③ 設置および運用が簡単であること
④ 運用コスト（SIMカードや機器レンタルなど）が安いこと

□アライアンス企業の開発事例紹介

① アライアンスメンバー（株式会社ハローシステム様）による開発事例
② 親機（３GIM利用）と複数の子機を使ったシステム
　・各部屋などでの動き・温度・明るさなどが分かる
③ 独居高齢者の状況（動きや明るさなど）をスマホで確認可能
④ 独居高齢者からのアラーム発信（メール送信）も可能
⑤ 設置が簡単（電源入れるのみ）→ あとはクラウド確認のみ

□発展・展開について

① 温度センサと湿度センサによる「熱中症」などのアラームを発信（警告）
② 見守り側との双方向での連絡も充実化（音と文字情報なども追加可能）
③ ペット（猫や犬など）の見守りシステムにも可能に



＊ 株式会社ハローシステム様の開発事例

2．スマートグリッド関連(IEEE1888)

□消費電力の見える化およびエネルギー削減へのニーズ増大

① スマートグリッドによる消費電力の意識が高まる

・具体的な見える化によって、対応策や節電を取ることが可能に

② 既存のメーカシステムとの差別化必要

・データの蓄積だけでなく、データ分析・解析が自由に行えること

・必要に応じて、ユーザが加工できること（現状、販売システムは難しい）

・安価なSIMカードおよびクラウドシステムで利用できること

③ ３GIM活用のメリット

・LANだとセキュリティ問題と敷設・維持に課題が多い。３GIM版は、課題解決し、簡単・迅速に設置可能。

□東京大学とIIJ、およびオープン ワイヤレス アライアンスで昨年10月にプレスリリース

① 「 世界初：IEEE1888対応の組込み３G通信モジュールを開発/M2Mクラウドサービスとの接続に成功」プレスリリースに発表

② 今後スマートグリッド（ＢＥＭＳやＨＥＭＳ）で標準化として利用

③ IEEE1888採用で標準的なクラウドのデータ相互運用が可能に

電力見える化システム（東大・フタバ企画）

□今後の発展系

① データ標準化により、相互運用が可能となり、応用展開が容易に

② M2Mの課題解決のひとつに

・クラウド関連での多くの分析・解析ツールとの連携が容易に

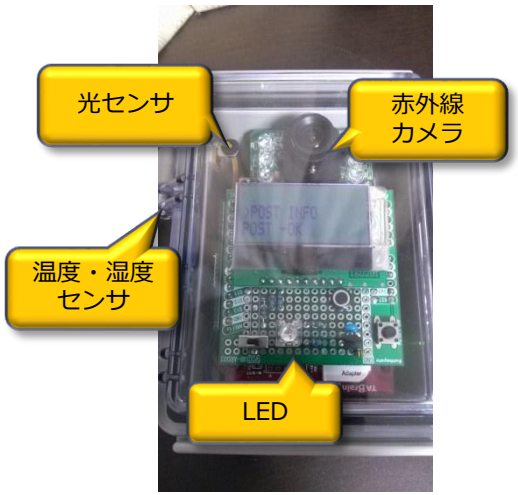
3．スマートアグリ関連の開発（オーダ品）

□農業分野でのIT活用が活発化

① スマートアグリにより、自動制御による食糧生産が現実に
- ・全自動による管理下での農業が実現　⇒ しかしとても高価で、一般農家では利用できない
- ・はたして、日本の農業に全自動化が必要か？（現状は、採算が合わない）

② 日本の農家に高価なスマートアグリの製品（IT化）は無駄
- ・すでにスマートアグリに挑戦した企業が倒産する事態も
- ・ビジネスモデルがまだ、農業IT化では、苦戦中

③ 現実にやれるIT化により農家が助かることとは何か
- ・離れた農地やビニールハウスなどで起きている現象を知りたい
- ・温度・湿度・土壌状態などから、農作物を荒らす泥棒や野生動物など



プラント見守りシステム
（拓殖大・構造計画）

□農家として本当に必要なシステムとは何か？

① 農家の老齢化により人手不足などが深刻化

② 農産物の被害も深刻化（泥棒や野生動物の被害）

③ 人手を補うもので、安価で手軽で、簡単に使えるシステムがまずはIT化

□遠隔による見守り・監視システムがまずは必要か？

① 温度・湿度・照度・二酸化炭素・土壌などの状況を遠隔地に知らせる（常時観測）
- ・場合によっては、メールにてアラーム送信

② 異常事態でのカメラ撮影や環境変化を遠隔地に知らせる（臨時観測・監視）
- ・盗難・野生動物被害などの防止、画像伝送による物象の転送

4．ICT百葉箱（IEEE1888利用）の開発

□インドにおけるDISANETプロジェクト (JICA/JST)

・南インドの都市ハイデラバードに気象センサ20台を高密度に設置し、データ収集を行い、これからの気象防災に役立てる

・M2Mゲートウェイによるデータ収集

・設置された気象センサからの情報はM2Mゲートウェイを通してIEEE1888形式に変換され、IEEE1888通信プロトコルにより、インド気象局内のサーバに転送。

□気象観測項目

Temperature：気温/Humidity：湿度/Pressure：気圧/RainFall：雨量/DayRainFall: 積算雨量/WindDir：風向/WindSpeed：風速

□デジタル百葉箱の特徴

・太陽光発電と蓄電による自律システム

・IEEE1888対応のM2Mゲートウェイの使用

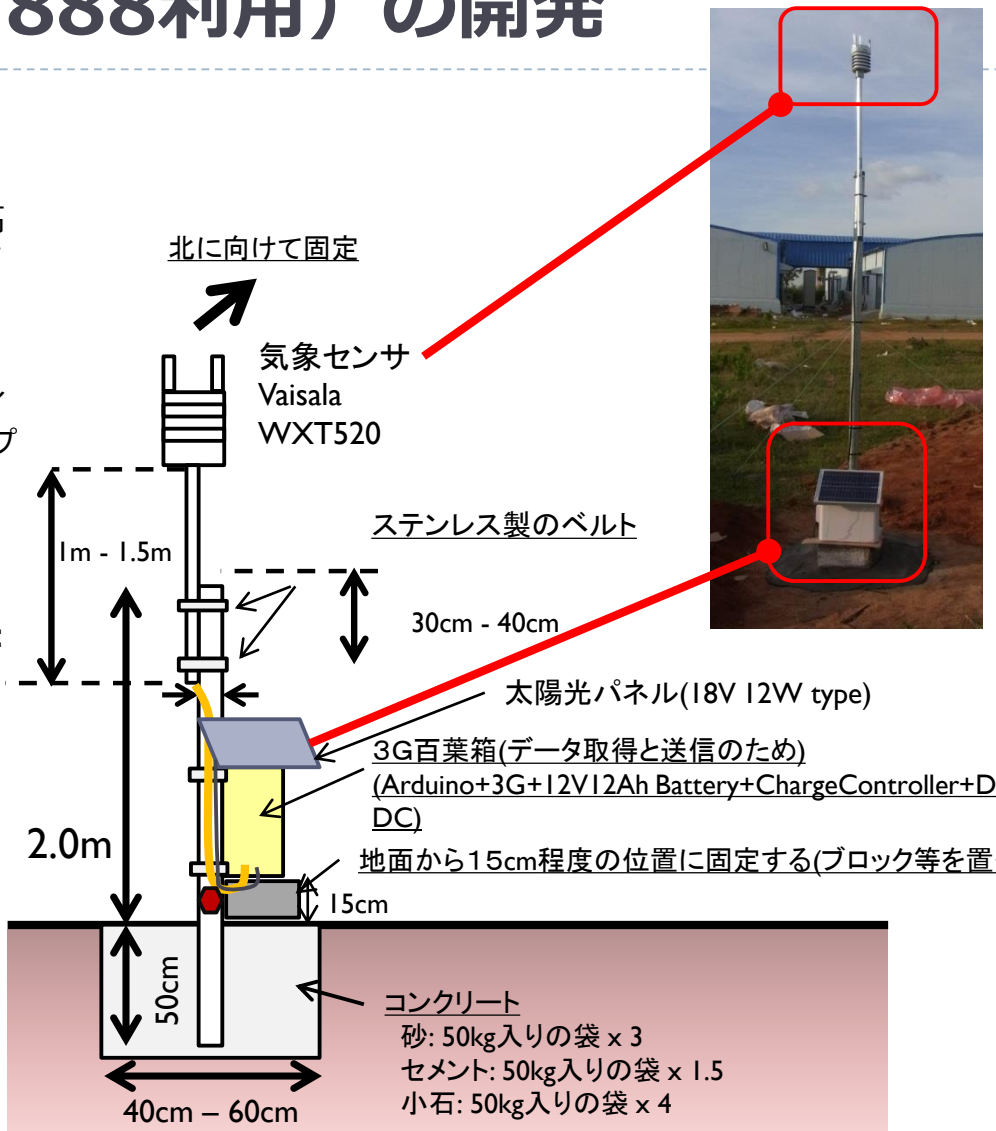
・3G通信によるデータ収集

□ M2Mゲートウェイの役割

・RS232Cシリアル通信によるセンサからのデータ収集

・収集されたデータのIEEE1888フォーマットへの変換

・3Gを使ったIEEE1888通信によるデータ転送



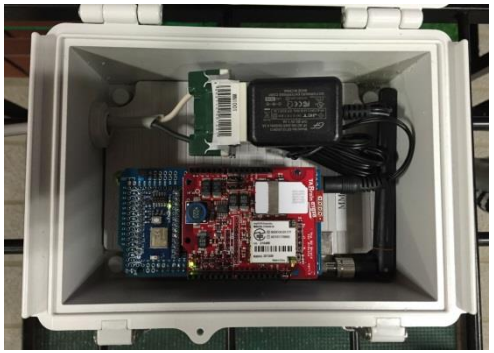
（東大・フタバ企画）

5．子ども見守りシステム（オーダ品）

小学生の登下校の見守りシステム

学校の校門に親機を設定

子どもがタグ（BLE）を付けて校門前を通過すると
親機が感知し、３Gでサーバに送信。
サーバ側では、保護者へ通過のメールを送信。



■開発プロセス
試作１→ 試作２→ 試作３ で開発進める

■課題
・登下校時の子供集団のトラフィック
・堅牢なシステム構築

試作を2回繰り返し、3回目のボードで
実運用に入る。ただ、試作１、試作２で開
発したボードも実運用で利湯尾中

ナビゲーションズ株式会社様提供（大阪）

6．農業用モニタリング（オーダ品）

遠隔での農業用ビニールハウス向け環境モニタリング開発

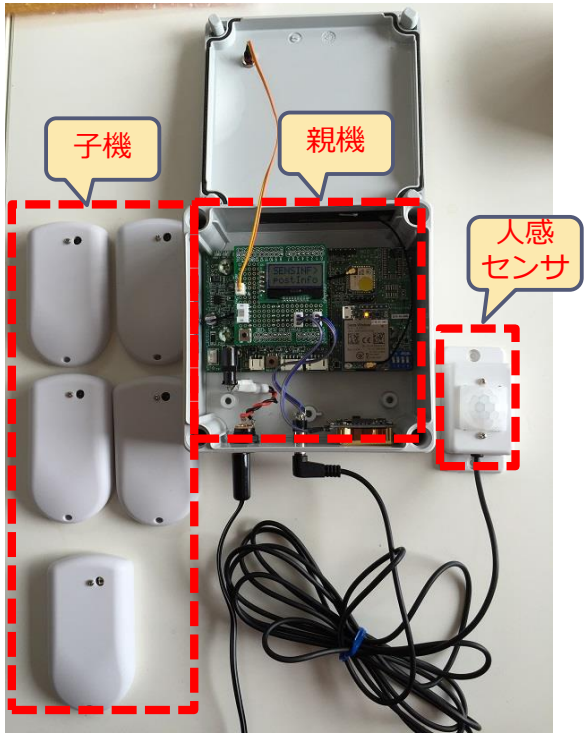
課題は、センサの設置場所、電源の確保、計測間隔など

有限会社歩産業様提供（熊本）

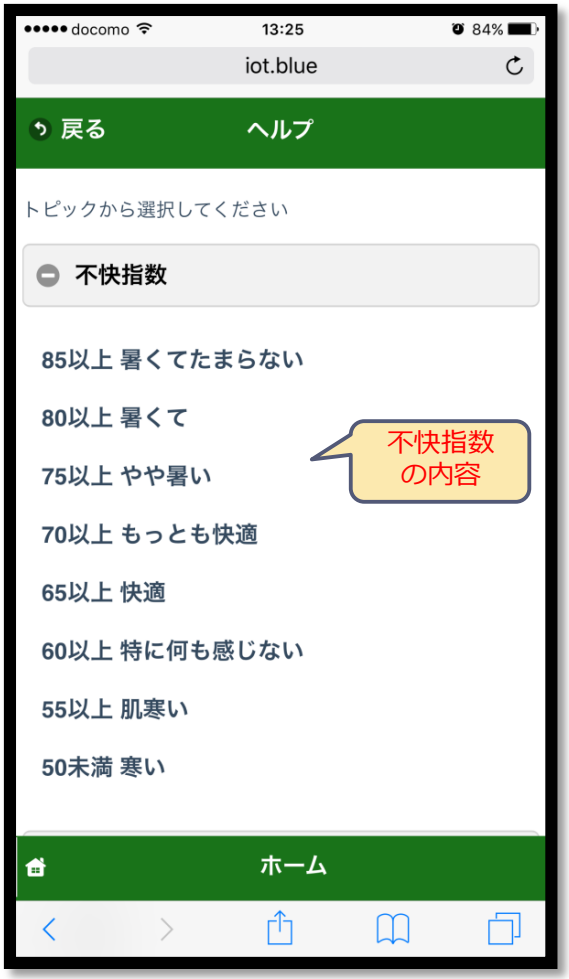
7．会議室環境モニタリング（オーダ品）

某メガバンクの会議室の利用状況把握
及び環境モニタリング

僅か1週間で、親機・子機5セット開発
クラウドも同じ１週間後サービス開始



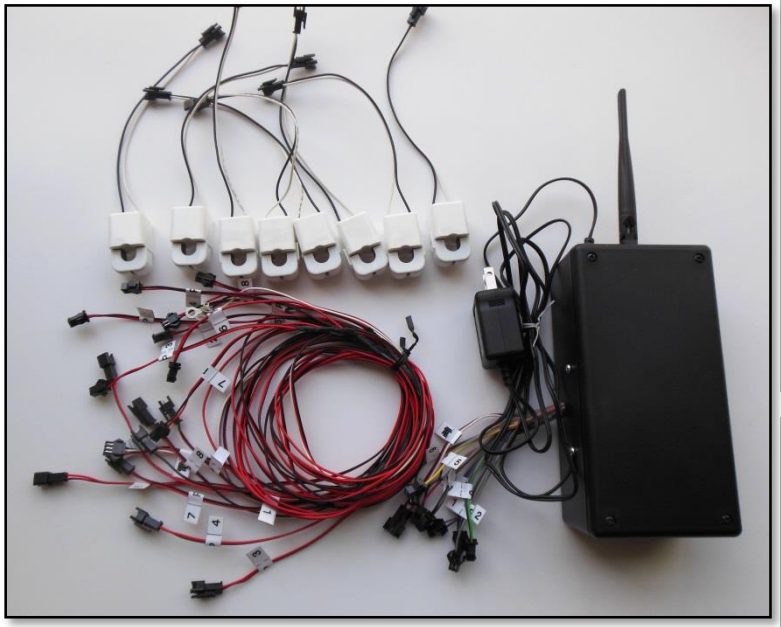
株式会社大和ビジネスサポート様提供（東京）

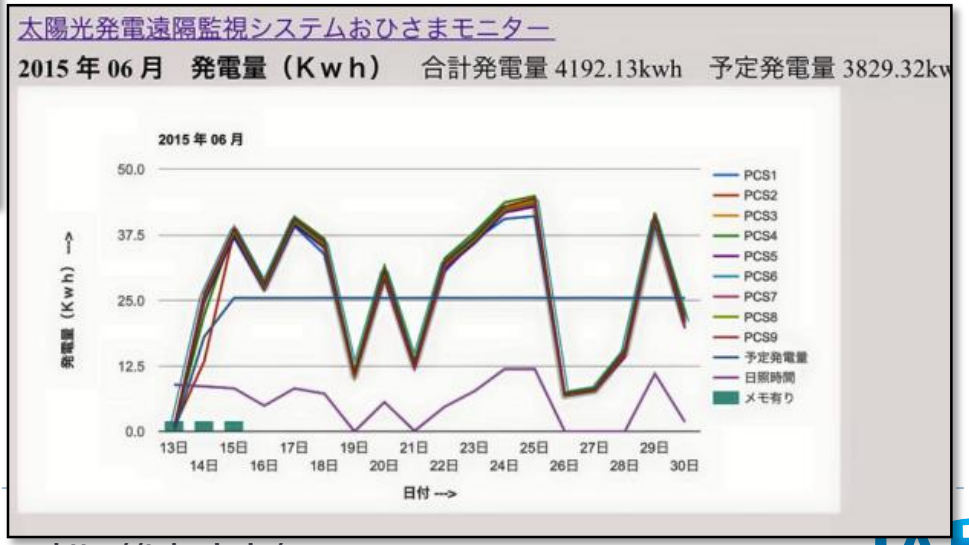


不快指数値の内容

8．太陽光発電量モニタリング

太陽光発電量モニタリング
2015年6月5日３Gシールド購入後、僅か1週間ほどで
クラウドにデータがアップされインターネット上で配信

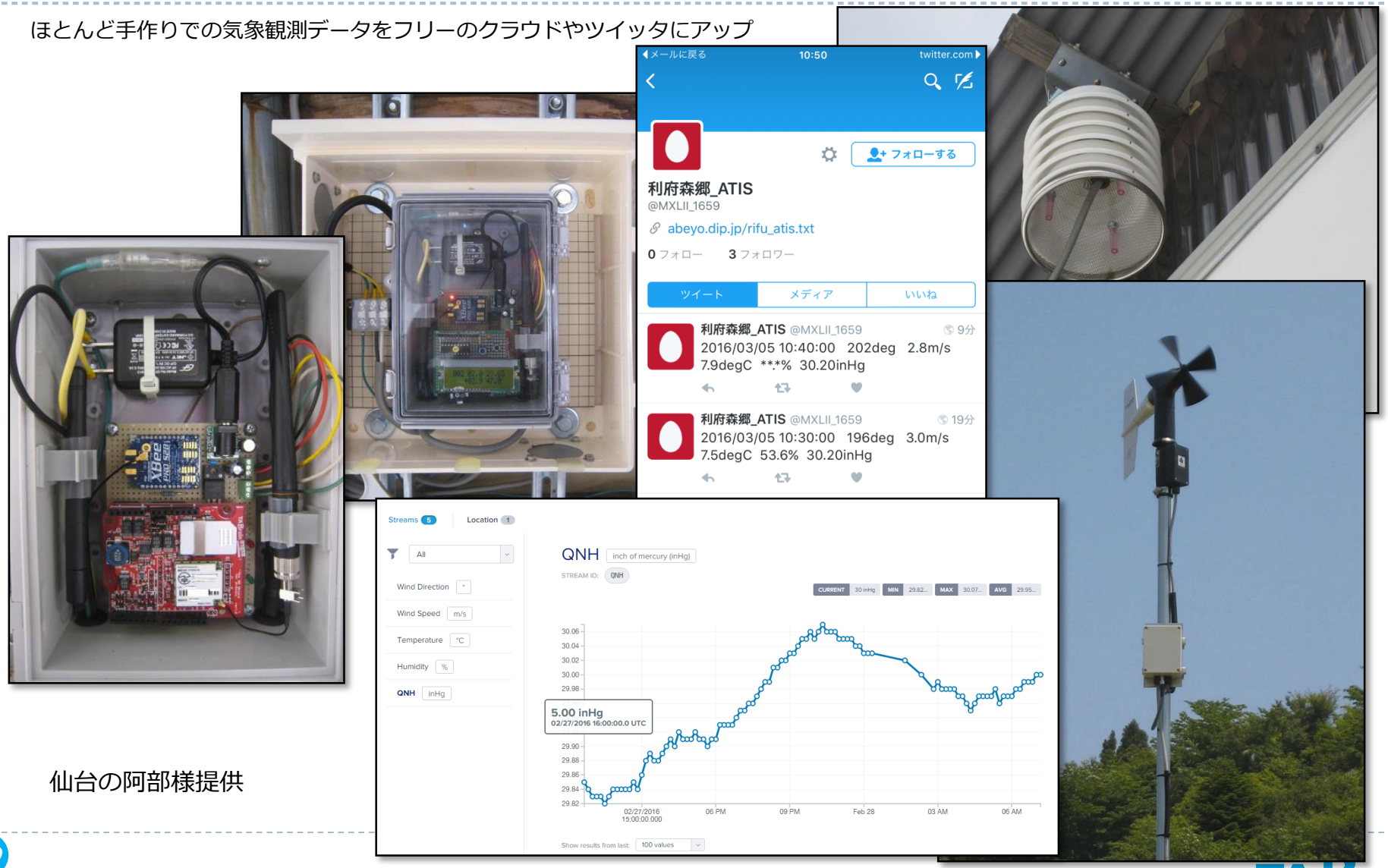




国井システム開発社様ご提供資料（石川）

9．気象観測モニタリング１

ほとんど手作りでの気象観測データをフリーのクラウドやツイッタにアップ

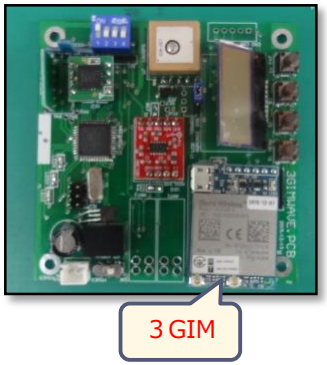


仙台の阿部様提供

１０．気象観測モニタリング２

某気象関連企業からの依頼での開発製品
今後、設置場所を増やすとのこと

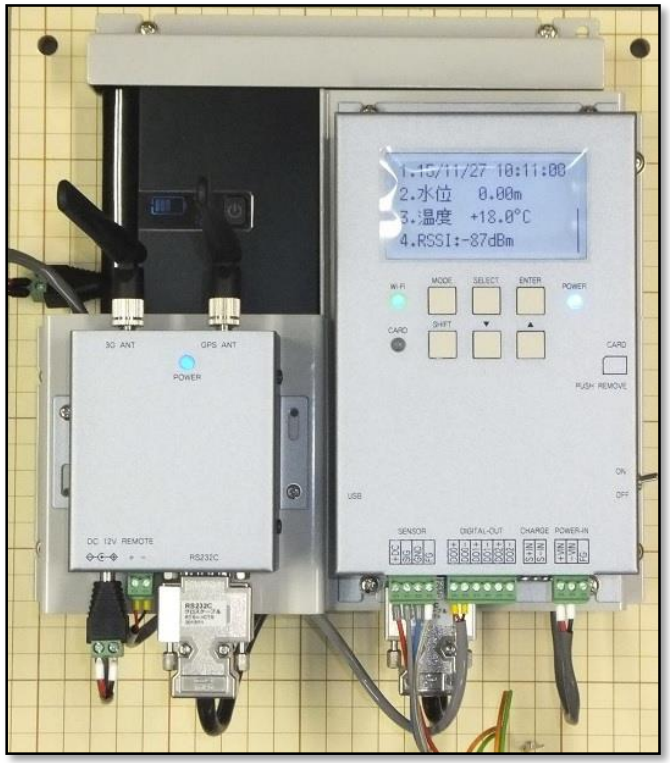
アマチュア無線でテレメータ開発してきたが、
３GIMを使いはじめたら簡単に開発できること
を確認し、すべて３GIM開発に切換えた

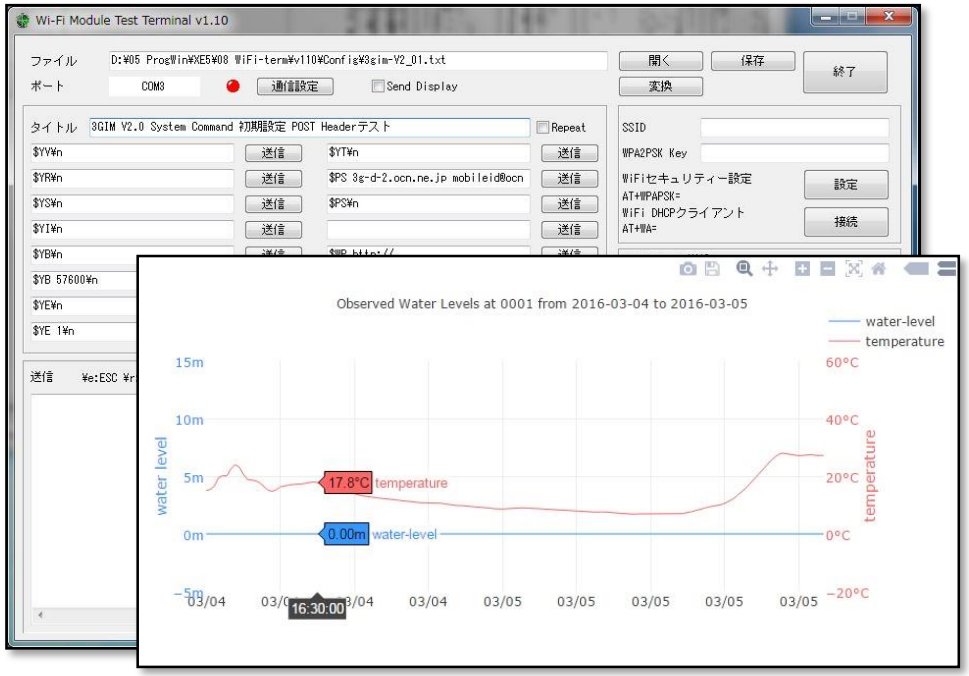


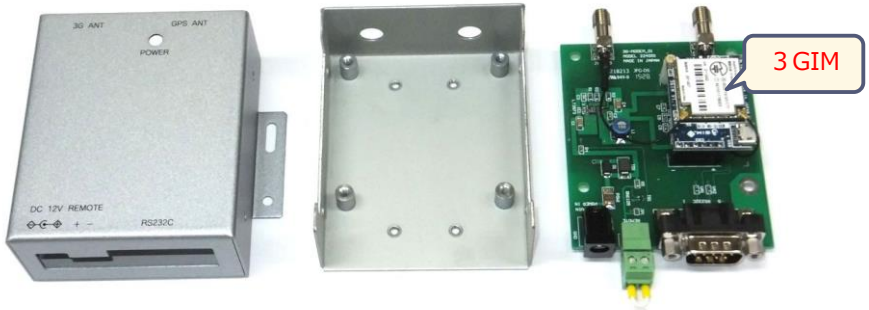
株式会社与論電子様提供

１１．水位観測モニタリング

河川水位計モニタリング機器







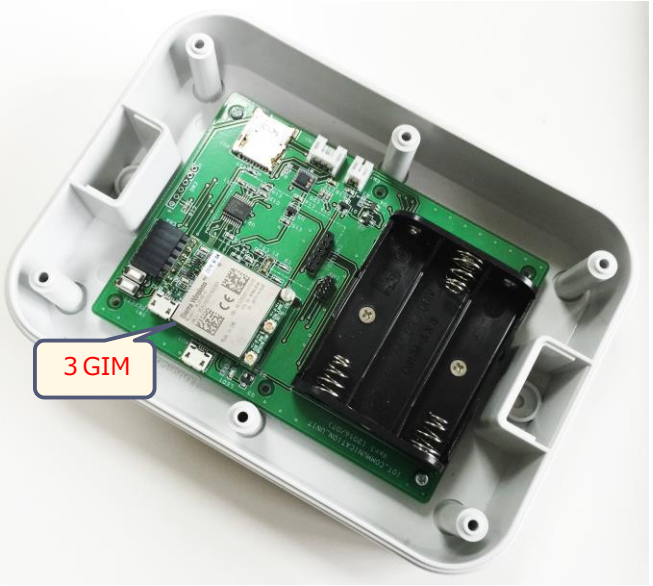
2015年夏WiFiで試作していたが、プログラム量の問題、
ルータ設置の問題から、即時に３GIMに切り替え

大谷計器株式会社様提供

１２．遠隔保守監視システム（オーダ品）

汎用的な遠隔保守監視システム

１） 屋外機器（太陽光発電駆動）の監視
・機器の信号を取得
・バッテリの容量監視
・低バッテリ時のセンサ値保存（SDメモリ）
２）低バッテリ時の非常時対応
・非常時バッテリ時の駆動
３）信号線と電源接続だけで即起動
・簡単な現場設置対応

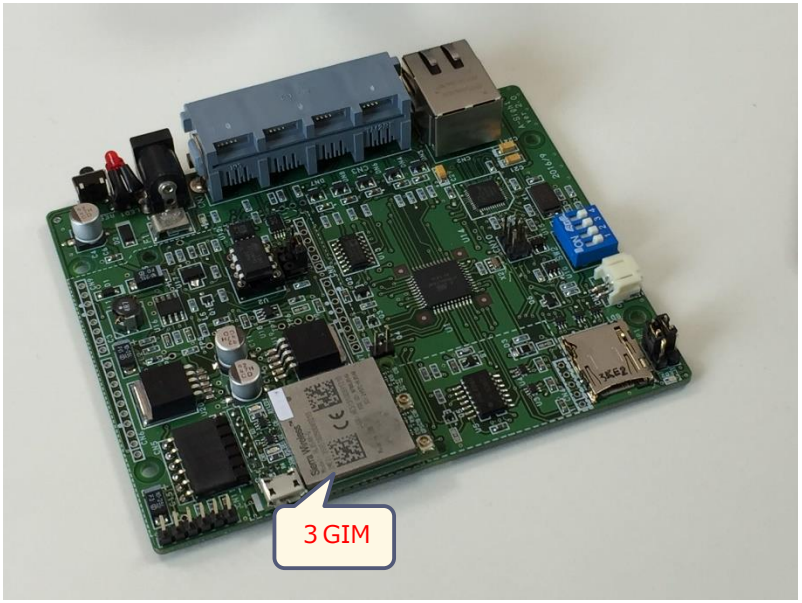


利用目的
・遠隔地の屋外にある設備機器の監視モニタリング
・これまで定期的な人的保守サポートが不要に
（大幅な人件費のコストダウン化）

主な顧客：
・某大手でのNTT関連野外機器の保守メンテ用として
・JR関連の線路切り替え信号機の保守メンテ用として
・電力会社所有の風力発電機振動機器の無線化として
・防塵建屋にとりつく機器の保守メンテ用として

１３．工場用IoT汎用ボード（オーダ品）

汎用的なIoTゲートウェイボード



１） 屋内および屋外機器の各種監視
- ・機器の信号を取得
- ・LAN・３G対応
- ・RS485、I2C、SPI、UART対応
- ・SDメモリ
- ・WDT機能（ハードウェアリセット）
- ・Arduino互換機
- ・３GIMライブラリ利用

２）豊富なI/F機能
- ・各種機器対応可能

３）堅牢なシステム
- ・システムの安定的な動きを展開
- ・ハードウェアリセット対応
- ・非通信時のSDメモリー保管

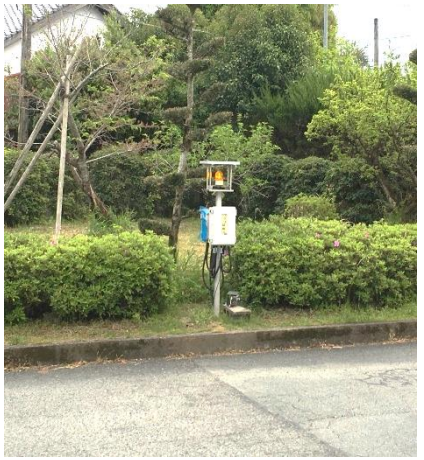
利用目的
- ・屋内・屋外での各種機器の保守サポート対応向け機器
- ・安定的で堅牢なシステム利用現場向け機器

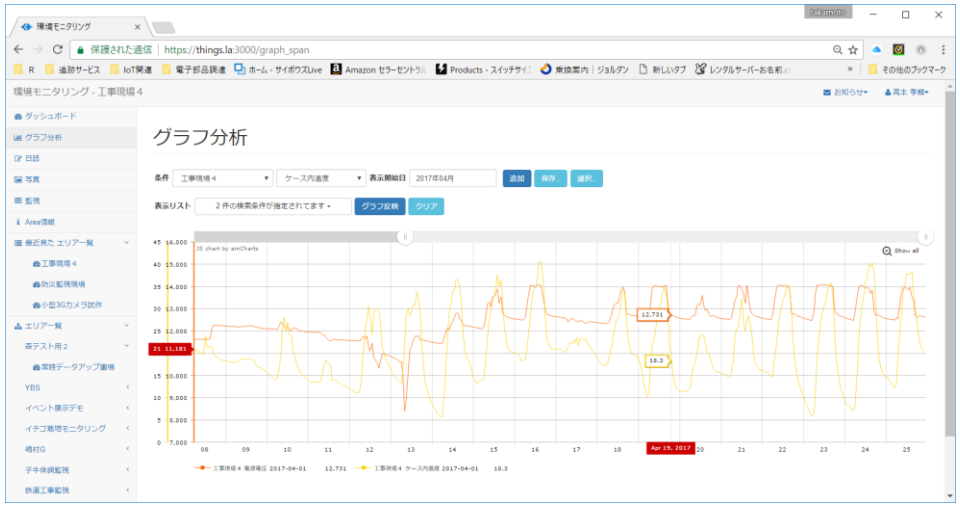
主な顧客：
- ・工場（製造現場）向け対応
- ・移動体（車載器）向け対応
- ・遠隔監視・モニタリング向け対応

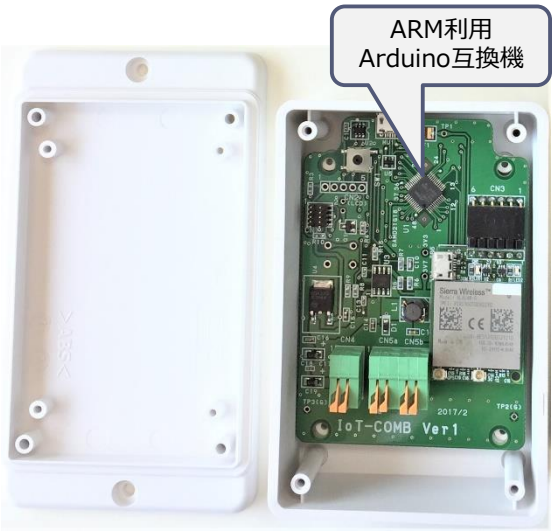
※専用ケースにて対応

１４．横取り失念装置遠隔監視システム（オーダ品）

超省エネタイプの屋外バッテリの遠隔監視システム
１） 鉄道線路切り替え工事中のリミッタスイッチ監視
２）装置
・３GIM＋IoT-COMB(Arduino互換機）対応
・IoT-COMB（ARM利用）
・温度センサ
・電源電圧測定
・リミッタスイッチ情報取得
３）既存装置に敷設
・ケーブル接続のみで電源ONの状態に
４）検査モードと運用モード
・現場設置前の検査モード（1分間隔）と
・現場設置での運用モード（1時間間隔を区別

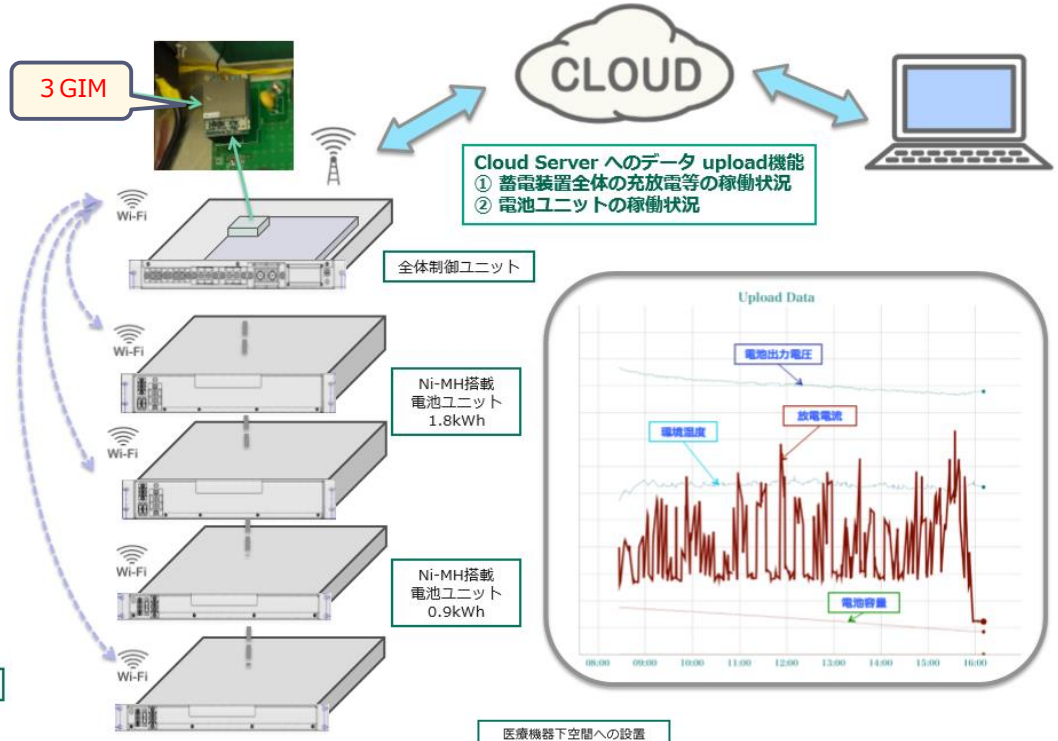






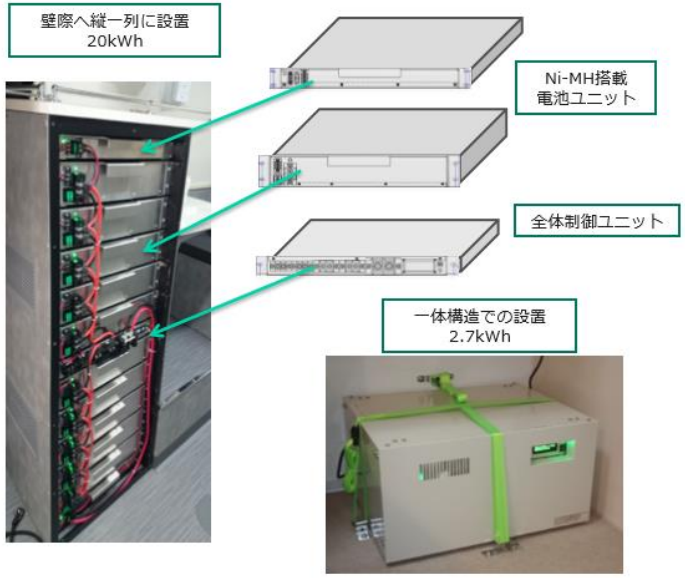
１５．Ni-MH蓄電装置の監視モニタリング装置

検診車のバッテリ遠隔モニタリング装置
・遠隔でのモニタリングをすることで
バッテリ追尾車を
・機器の信号を取得
・LAN・３G対応



特徴

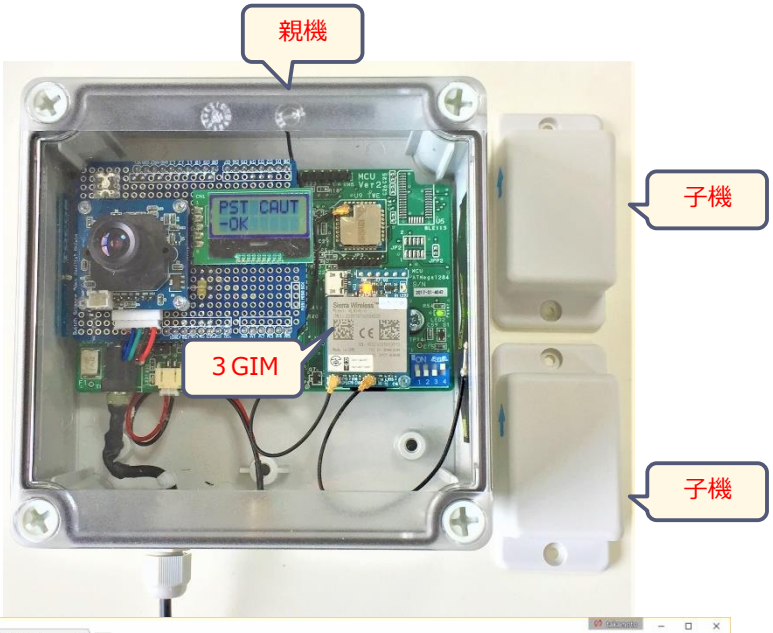
- **電池容量及び設置場所に柔軟な対応が可能**
- **蓄電装置稼働状況をリアルタイムで管理可能**

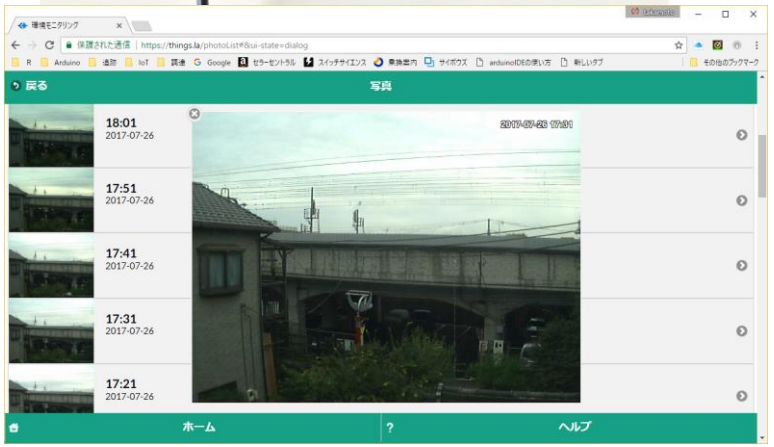


mhBSLabo
Confidential

１６．防災カメラ監視システム試作（オーダ品）

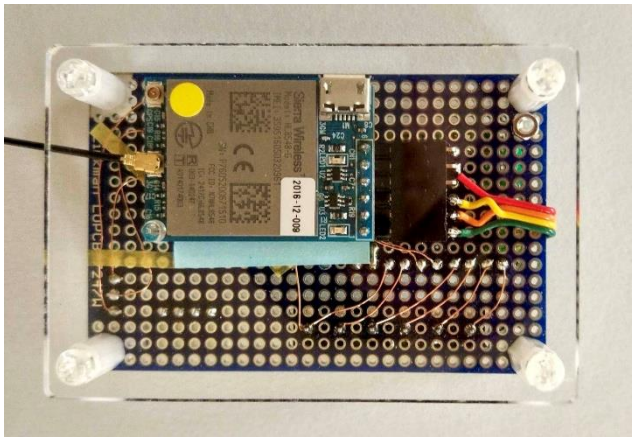
防災用監視カメラ試作
１）バッテリのみで稼働する可搬性のあるシステム
２）仕様
・３GIM＋TabrainoV1.1対応
・TWE-Lite（親機と子機との関係）
・温度センサ・外部電源電圧測定
・子機：加速度センサ・電源電圧
３）子機の加速度センサ利用
・微振動の抽出
・傾斜角度を抽出
４）重要仕様：長期安定での防災モニタリング
５）課題は、消費電力とバッテリとの関係

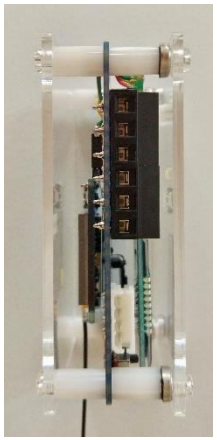




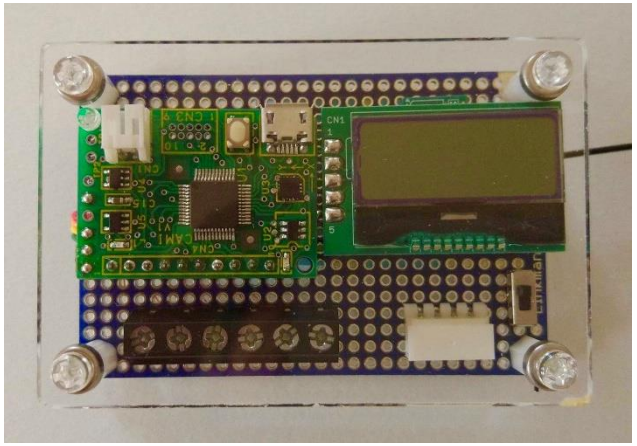
１７．ひび割れ監視システム試作（オーダ品）

高速道路柱脚・橋梁コンクリート部のひび割れ監視
１）バッテリのみで稼働する可搬性のあるシステム
２）仕様
・３GIM＋Tabraino２対応
・歪ゲージ＋距離センサ＋加速度センサ
・温度センサ・外部電源電圧測定
３）加速度センサ
・大きな振動時を捉えてメール送信
・常時観測が重要（ひび割れ進行を観測）
４）重要仕様：バッテリのみで長期間での稼働





※長期間での省エネ対応は
１）常時消費電力を抑える（0.5mA程度）
２）ソフト制御
① スリープ処理
② ウェイクアップ処理
③ 割込み処理
④ 消費電力を食わないセンサ取得術
⑤ 効率的な３Gでのデータアップ処理



１８．車両専用移動体IoTボード（オーダ品）

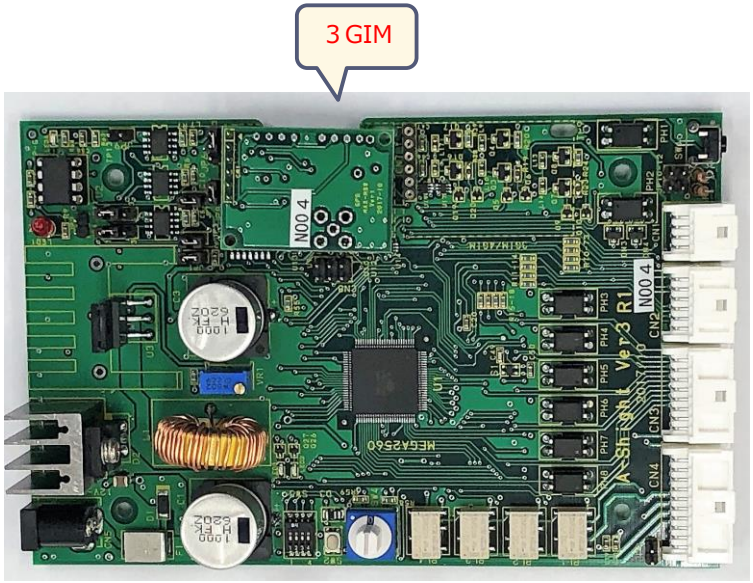
移動体用IoTボード

１） バス・トラック等のGPS監視モニタリング
・自動車内さまざまな装置の信号を取得
・３G/LTE（４GIM）対応
・I2C、SPI、UART対応
・SDメモリ
・WDT機能（ハードウェアリセット）
・Arduino互換機
・GPSモジュール搭載
２）豊富なI/F機能
・各種機器対応可能
３）堅牢なシステム
・システムの安定的な動きを展開
・ハードウェアリセット対応
・非通信時のSDメモリー保管

利用目的
・バス・車両等の各種機器の対応向け機器
・安定的で堅牢なシステム利用現場向け機器

主な顧客：
・バス・車両等に搭載対応
・その他移動体向け対応
・遠隔監視・モニタリング向け対応

※専用ケースにて対応



１９．ひび割れ振動距離監視モニタリング（オーダ品）

首都高速道路柱脚ひび割れ監視モニタリング装置

１）IoTボード

・３GIM/４GIM敷設コネクタ付き
・MCU： SAMD21G18A
・加速度センサ/温度センサ/距離センサ付き
・ひび割れセンサ付き
・３GIMライブラリ利用可能
・開発期間（ハードウェア1.5ヶ月間）
・ソフトウェアは事前の試作品のものを活用
・長期間稼働するための超省エネタイプ

２）利用目的

利用目的：バッテリで数年間稼働のこと

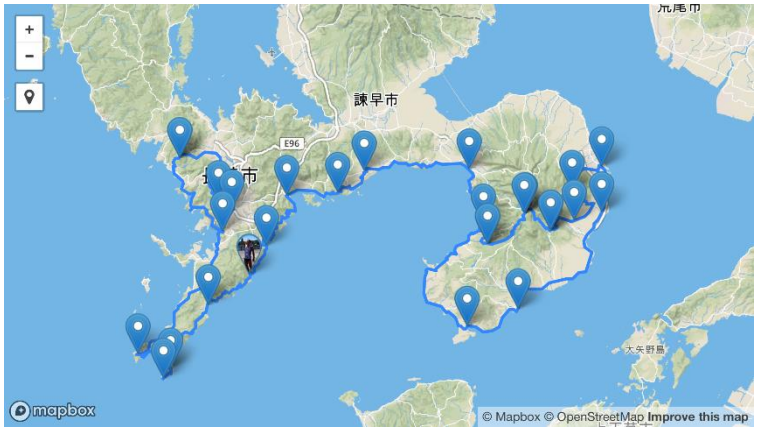
2017年11月21-22日 テクノハイウェイ展で展示

２０．ランナー追跡モニタリング

長崎での２７３キロのマラソンランナーに着けて
追跡モニタリングに利用

・デバイス重さは、わずか50ｇ前後から120ｇ程度
・バッテリは３日間継続して動くことが求められた
利用デバイスは
　３GIM＋IoTABボード＋リチウムイオン電池

（作成者・ランナー：長崎市の菅崎氏）



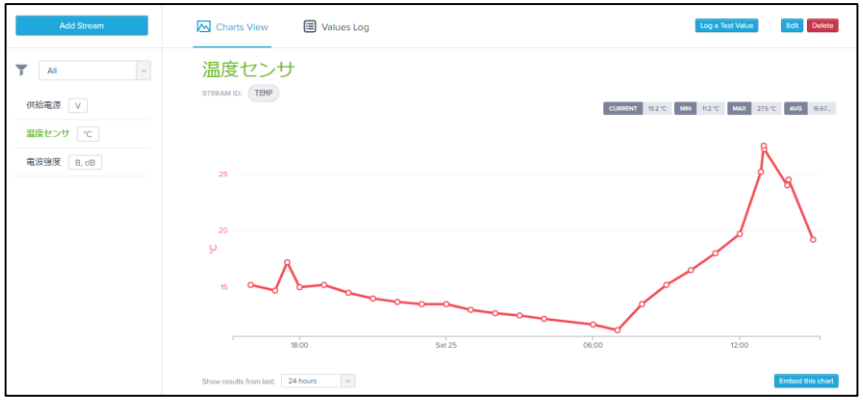
２１．イノシシ罠の感知システム（自社品）

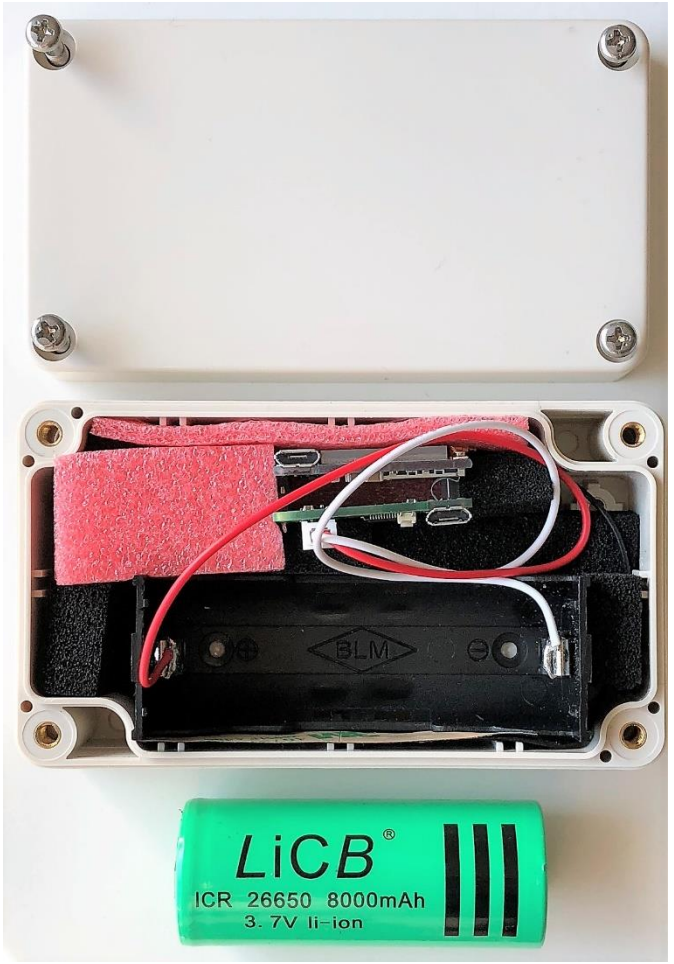
最近は、害獣駆除のためにさまざまな対応策がとられています。イノシシも農作物を荒らし、被害も甚大なものとなってきていて、その駆除・対策も多く行われています。

本件では、イノシシの罠を畑や山林に設置し、罠にかかるとそれをメールで知らせるIoTデバイスを作成しました。

バッテリのみで長期間稼働し続けるもので、そのバッテリ電圧も常時監視できるものとしています。

１時間に１回程度、バッテリ電圧および温度をクラウドにアップし、罠にイノシシが掛かるとメールが関係者に飛んでくるものとしています。ここでは、加速度センサによる振動で、メールを送信します。





上記写真の８０００mAhのリチウムイオン電池で約４ヶ月ほど稼働し続けます

22．IoTAB & IoTAB Proボード（自社品）

3 GIM/ 4 GIM対応用IoT利用ボード

1）IoTABボード

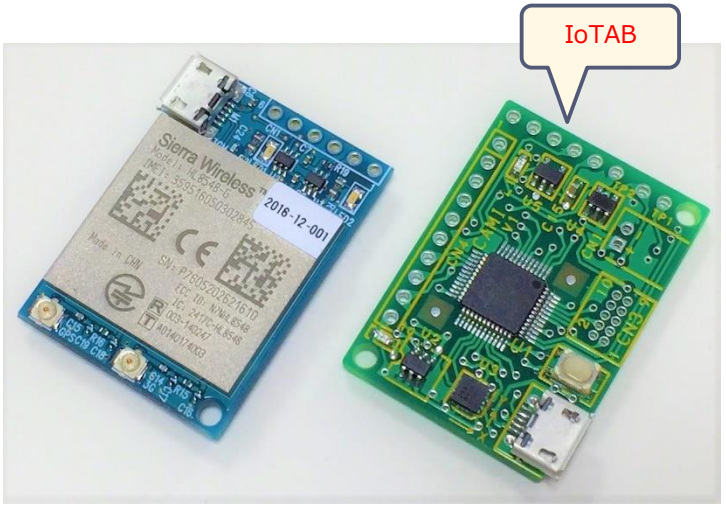
- ・3 GIM/ 4 GIM敷設コネクタ付き
- ・MCU： SAMD21G18A
- ・加速度センサ/温度センサ
- ・Arduino互換機
- ・3 GIMライブラリ利用可能
- ・アナログIN 4 （+ 2）ピン
- ・GPIO 2 （+ 6）ピン
- ・I2C/UART*3

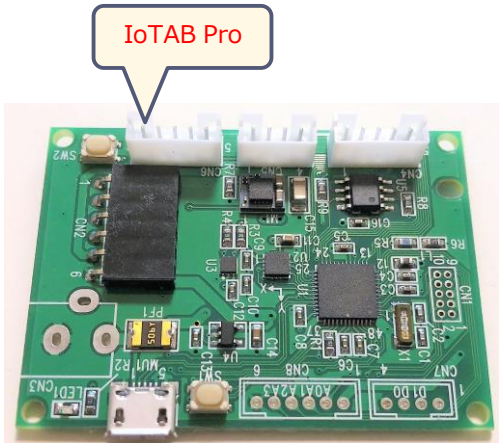
利用目的：超省エネ・超小型開発向け
利用事例：マラソン走者追跡・動物（サル）追跡

2）IoTAB Proボード

- ・3 GIM/ 4 GIM敷設コネクタ付き
- ・MCU： SAMD21G18A
- ・加速度センサ/温度センサ
- ・Arduino互換機
- ・3 GIMライブラリ利用可能
- ・アナログIN 4 ピン
- ・GPIO 10ピン
- ・I2C/UART*3/SPI
- ・システムの安定的な動きを展開
- ・ハードウェアリセット対応
- ・非通信時のSDメモリー保管

利用目的：汎用IoT省エネ開発ボード
利用事例：高速道路柱脚ひび割れ監視、





２３．IoTABを用いた小型IoTカメラ試作（自社品）

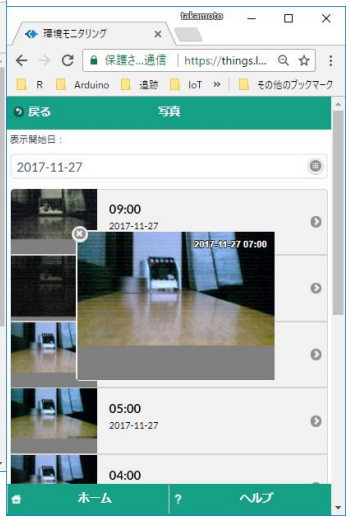
IoTABボード＋３GIMを使ったIoT小型カメラ試作

筐体（ケース）サイズは、わずか5㎝×5㎝×2.5㎝

利用が特に簡単にし、ただバッテリ充電して設置するのみ。
300mAh程度のバッテリで、30分のセンサ値取得、1時間ごとのカメラ画像アップで、2日間ほど連続使用可能。
また加速度センサによる割込みで起動する。

今後の改良点
　外部電源利用対応（オプション）
　遠隔制御可能

利用目的：一時的な防犯・防災・見守りなど



２４．河川水位監視デバイス開発（オーダ品）

国土交通省では、最近の多くの水害に対し、上流河川でも水位監視のニーズが高まり、そのデバイス設置が急がれています。
今回開発した水位監視デバイスは、すでに新潟県、神奈川県、高知県、熊本県などでの試験運用として、Arduino+３GIM/４GIM＋IoTABボードをベースに開発したものです。
ここでは、安価で、耐久性があり、長期（5年）に渡って稼働することが要求されています。ここでは、太陽光発電パネルとバッテリによって長期安定稼働する仕組みをとっています。

現在（2018年6月）では、カメラ付きのデバイスも開発し、超音波距離センサ（５～１０m）を使って水面までの距離を計測し、クラウドにデータをアップし続けています。
本システムでの水位監視間隔は、水位が低い場合には１時間ごとに、危険水位に達すると５分ごとにクラウドにデータをアップします。
（危険水位距離や計測時間間隔は設置後クラウドで調整可能）

【機能】
クラウドにアップするデータは、水位（水面までの距離）、電源電圧（バッテリ容量監視）、温度、電波強度などで、雨天が１ヶ月継続しても継続して稼働する仕組みとなっています。

【特長】
・現場危険水位（河川水面までの高さ調整）がクラウド調整可能
・監視調整（危険水位監視間隔調整）がクラウド対応可能
　<デバイスのID管理によって設置場所と紐づけ>
・Arudino互換機＋３GIM/４GIMで製造コストは安価
・超省エネのArudino互換機IoTABボードを搭載
・雨天時の長期対応可能（１ヶ月ほど稼働）※

※電波強度なども影響し、１ヶ月以下になることもありえる。

太陽光発電パネル
（５W）

水位センサは、MaxBotix社の
MB7093 や MB7386利用
（５m～１０mまで監視可能）

河川水位監視デバイス
防水対応 IP65

カメラ対応版

現場設置事例

２５．量産化農業用IoT監視デバイス開発（オーダ品）

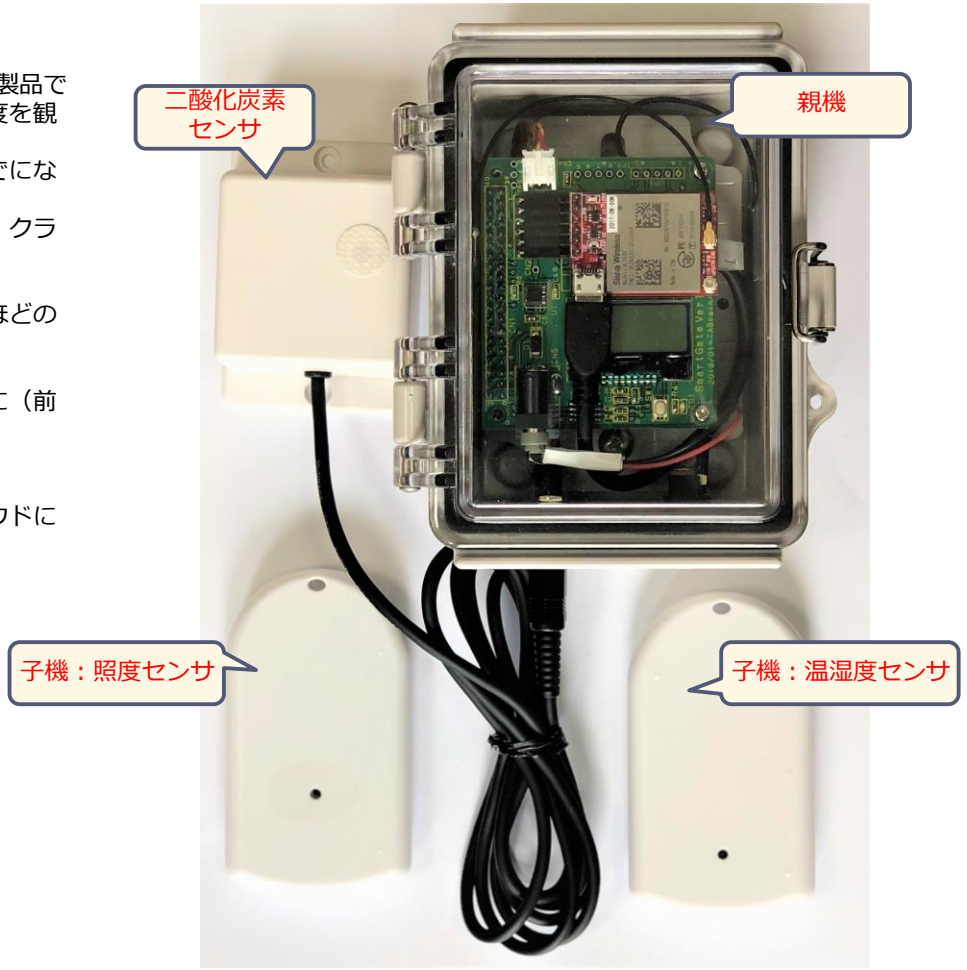
本デバイスは、RaspberryPi Zeroをベースに開発した農業用監視デバイス製品です。量産化対応として、ビニールハウス内の温度、湿度、照度、二酸化濃度を観測し、そのほか子機のバッテリや電波強度をモニタリングします。
これによって安価で利用できるデバイスとなり、広くご利用いただけるまでになりました。
ここでも安定して稼働することが重要ですが、親機の電源を入れるだけで、クラウドにデータがアップされることから、使い勝手も超簡単となっています。

温湿度や照度については、子機として複数台増やすことができ、約５０mほどの距離でも親子間での無線が利用できるようになっています。

こちらは某国立大学の農学部による研究開発で開発したものですが、すでに（前頁での）試作も多いことから、量産用として開発したものとなっています。

【機能】
・農業用専用として、二酸化炭素濃度および温湿度・照度のデータをクラウドにアップして監視可能となる
・監視は、5分から１０分間隔で設定（調整）が可能
・閾値をクラウドに設定し、メール送信が可能
・子機側は、数10台まで増やすことが可能
・親機と子機の間の通信距離は５０mほどまで可能
・電源ON/OFFだけで利用可能

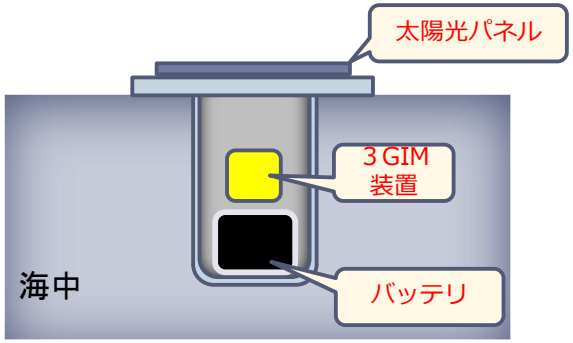
【特長】
・安価なデバイス提供
・超簡単な利用（電源On/Offで利用開始・終了が切り替え）
・子機側のバッテリ切れの監視が可能



農業用IoT監視デバイス

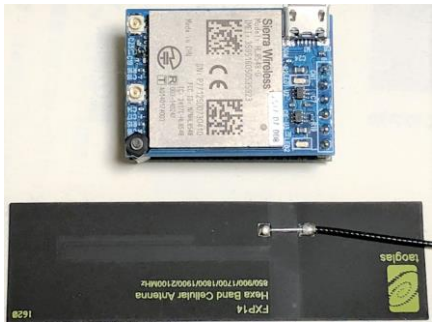
２６．その他 特殊IoTデバイスの試作

① 海洋調査用IoTデバイス



海洋調査のひとつで、海上に浮かせて海中の
温度や海水分析を行うデバイス開発
（詳細は秘密）

② 野鳥に取り付けた気象観測デバイス



大型野鳥に取り付けた観測装置
超小型化（軽量化）を目指し、バッテリのみ
で長期間稼働し続けること
特にGPS機能を使ったデバイスが特徴

③ GPS機能を使ったデバイス

移動体監視・動態監視のデバイス開発が多発
・徘徊老人向けデバイス開発
・害獣調査デバイス開発
・観光者調査向けデバイス開発
などなど

→ 来年から「みちびき」対応可能に
誤差が６cm程度に

④ 機器保全対応デバイス

工学な機器製造メーカが顧客向け保守対応と
して、機器にIoTデバイスを組込み
保全対応を遠隔で行うケース増大
→ 消耗品の計画的な対応
→ 故障発生などの早期対応
→ 総合的なメンテナンス対応
など多くのメリットが存在

２７．その他試作・プロトタイプ開発ほか

□人や動物の動きをキャッチしデータとして収集・分析

① 親機と子機との関係でシステム開発（課題は子機の長寿命化）
　→ 親機にゲートウェイ機能と子機受信器、子機は発信機能のみ
② クラウド連携において動きをモニタリング分析
　→ 子機固有IDの動きをモニタリング、
③ 関係者にはメールによって動きを知らせる

□特殊環境下の維持装置モニタリングシステム試作

① 助成金によって特殊環境下の維持管理をモニタリングするシステムの試作

② 複数のセンサ値をクラウドにアップし、その状況を把握できる環境下に

③ 異常な状態をいち早く知らせるシステムとして開発中（端末系はスマホや携帯のメール）

□オープンソースハードウェア関連での試作・プロトタイプ開発支援

① 既存システムに組み込む高価な機材を安価なシステムに切り替えるために試作・プロトタイプ開発
　・通信モジュールを用いてモニタリング機能を対応
② 工場内機器の保守モニタリングや設定制御などを遠隔操作によって自動化
　・自社内LANとは別系統でセキュリティ対応に無関係

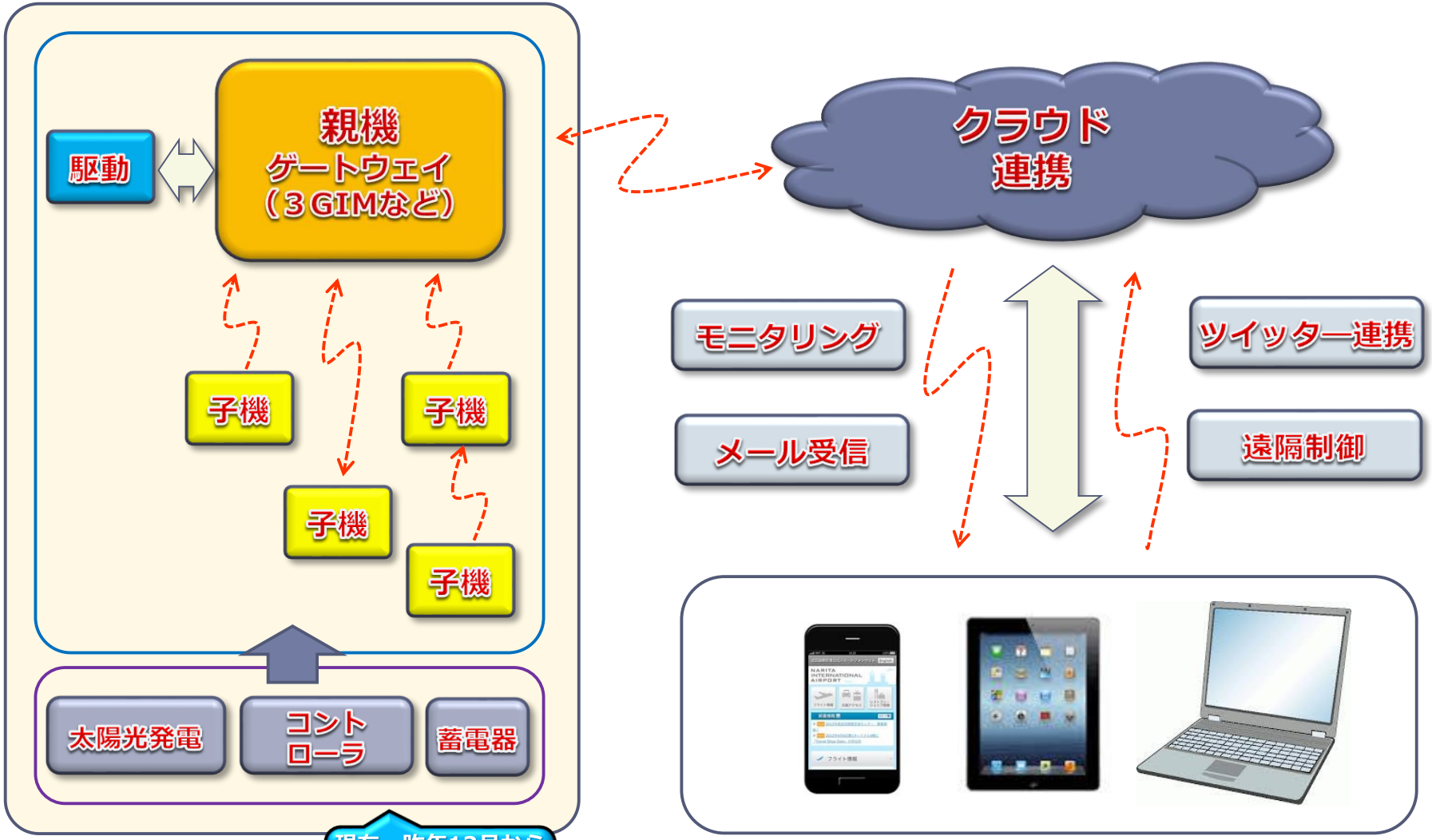
２８．大学・高専での開発事例

- 信州大学（電磁波解析と磁界発電の研究）
- 東京大学（腐食センサーによる橋梁保全研究）
- 徳島大学（橋梁の保全研究）
- 東京海洋大学（近海小型船避難緊急発信装置）
- 沼津工業高専（農業用エリアモニタリング研究）
- 千葉大学（農業用ビニールハウスモニタリング）
- 東海大学（農業用モニタリング研究）
- 東海大学（EVカー蓄電モニタリング）
- 拓殖大学（EnOceanと３GIMの連携）
- 和歌山県立海南高校（缶サットに搭載）

その他多くの大学・高専で、３GIMを使った研究活動実施

電磁解析
（信州大学・田代研究室）

２９．３GIM関連の共通技術提供



３０．第１回アイデアコンテスト(2013年度)

http://3gsa.org/information.html にて公開

・最優秀賞　３GSコミュニティバスお知らせシステム　（資料）　飯島幸太氏
・準優秀賞　クラウドコレクタ　（資料）　山本三七男氏
・努力賞　愛車の記録　（資料）　小林康晃氏
・その他応募作　相互見守り（コミュニケーション）システム

・最優秀賞　Himawari3搭載 ３G火山ガスモニタリングシステム　拓殖大学 瀬谷鮎太氏
・優秀賞　ペット管理システム　（資料）　東京都立小石川中等教育学校　小島和将君
・準優秀賞　ArduinoをつかったFOXテーリング　（資料）　東京都立総合工科高校　高橋君・土屋君・平林氏
・努力賞　山の幸、獲ったどー　（資料）　拓殖大学　南川俊氏ほか
・特別賞　わいっち目　（資料）　拓殖大学　井上龍氏ほか
・その他応募作
　自動車防犯装置　（資料）　東京都立総合工科高校　土屋君・高橋君・平林氏
　水田あんばい　（資料）　拓殖大学　金山祐大氏ほか
　アマモ場の生育環境観測システム　（資料）　広島商船高専　芝田研究室

３１．第２回アイデアコンテスト(2014年度)

http://3gsa.org/information.html にて公開

平成26年11月16日開催されました第2回３GIM・アイデア・コンテストの結果報告となります。以下のFacebookなどで掲載しています。

・最優秀賞　「快適マネージャー」
　東京都立小石川中等教育学校 小川広水君（中学一年生）　資料/ビデオ
・優秀賞　「ポチっとじょうろ」
　東京都立小石川中等教育学校 中野龍太君・中本一輝君・小林俊介君（中学二年生）資料/ビデオ
・特別賞「Rubyを用いたマイコンプログラムの遠隔書き換えシステム」チーム海南 山本三七男氏
　他和歌山県立海南高校（瀧本君、若勇君、和田君、筈谷君、岸田先生）/ルアリダワークス 資料/ビデオ
・特別賞　「Dustino(ダスティーノ）～ゴミ箱管理システム」
　九州工業大学 備後博生君、城戸翔兵君、張思嘉君　資料/ビデオ
・アイデア賞「冷蔵庫を使ったお年寄り見守りシステム」
　東京都立小石川中等教育学校 金子知洋君（高校二年生）資料/ビデオ
・アイデア賞「熱中症予防散システム」
　九州工業大学 待野翔太君・友永健太君 資料/ビデオ
・技術賞「３GIMを使用したGPS+GLONASS vs GPSの位置精度比較」
　チームmochi 望月康平氏 資料/ビデオ
・技術賞「自動散水システム」
　九州工業大学 中山一平君、永山雄一君、 Avinash Dev Nagumanthri君 資料/ビデオ
・技術賞「心拍数測定システム」
　東京都立小石川中等教育学校 佐藤和哉君（高校二年生）　資料/ビデオ

このほかにも入選からもれたのもありましたが、どれも素晴らしい作品でした。

第Ⅳ編　補足編

第１３章
Arduinoプログラミング文法

Arduino IDEを使う上での基本文法

プログラミング開発は、コンピュータ言語を使って行います。Arduino IDEでは、ほぼC言語（C++)に近い言語で開発を行っていきます。（ここではArduino IDE上の開発言語を 「Arduino言語」とも呼んでいます）

ここではArduino言語を使う上での、基本的な文法（記述する上での決まり事）を覚えていく必要があります。必ずしも文法を覚えてからでないと開発ができないことはありませんが、文法を覚えることで、効率良く、短時間で開発ができるようになっていきます。

ここでは、基本的な文法として、以下の５つについてご紹介していきます。

１）関数と引数

２）制御文

３）表記法とデータ・演算子

４）組込み関数群

５）キーワード等（一部）

1．関数と引数

重要

Arduino言語は、Javaで作られた言語で、C言語系・C++言語系である。

Arduino言語
（手続き型言語処理）
1）順次処理
2）条件処理（制御文）
3）反復処理（制御文）

関数と引数

カッコ内は引数

```
// 関数呼び出し例
digitalWrite(13, HIGH);
delay(500);
digitalWrite(13, LOW);
```

関数と引数を覚える

ここで
digitalWrite ： 内部関数 （組込み関数）
delay ： 内部関数

内部関数はあらかじめ登録されている関数群
「Arduinoをはじめよう」の付録を参照

内部関数は、IDE上では色が付く

外部関数は、ユーザが独自に開発した関数群

外部関数の定義

関数には、引数があり、**戻り（リターン）値**を設定
※一般的な関数の定義文

```
型 関数名（引数、・・・・）
 {
    処理群
 }
```

関数も、システムと同じで
入力： 引数
処理：処理群
出力： 関数の型

型： 型がvoidの場合は戻り値なし（**プロシジャ**:手続き）
それ以外では、return値で値を返す必要がある
関数名： システム定義以外の名前を表記（英数字文字を使う）
（大文字と小文字を区別するので注意）
引数： 型と変数名のペアで定義
たとえば、**【インチ換算関数】**の例

```
float inch (int cm) { return ( cm/2.54 ); }
```

float は型で実数のこと、
inch は関数名
int は引数の型
cm は引数の名前
（関数の中だけで利用可能）

【定数と変数】
定数：定義した値の内容が変更できるもの
＃define、 constant などで設定
組込み定数 HIGH、LOW、INPUTなど
変数：定義した値の内容が変更できるもの
型 変数名=初期値; で設定（一般）

型（タイプ）には、
1) boolean（ブーリアン値） ： True(=1), False(=0)
2) byte（バイト） ： 0..256
3) char（文字） ：
4) int（16ビット整数）<unsigned int> :
5) long（32ビット整数）<unsigned long> (= float)
6) float（32ビット実数）または 「double」 でも可
7) string （文字列）
8) void (型なし)
その他、配列や構造体、クラスなどあり

型はエディタ上では色が付く

【Arduinoの基本となる２つの必須関数】

```
void setup()
{  処理群    } // 初期設定群
void loop()
{   処理群      } // ループする本体処理
```

setup関数は、初期設定で一度だけ処理される

loop関数は、setup関数処理後、繰り返し処理される関数

コメント
1）一行コメント： // コメントになる
2）コメント間 ： /* ここから、ここまで */

コメントはプログラムを見易くする方法のひとつ

2．制御文

重要

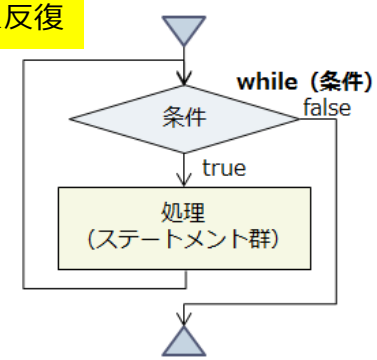
制御文は「分岐」と「反復」の２つのみとなります。
また、Arduinoでは、以下の５つの制御文が使えます。

Arduinoにない制御文
■repeat文 なし
■until文 なし

制御文（一部）

■while文

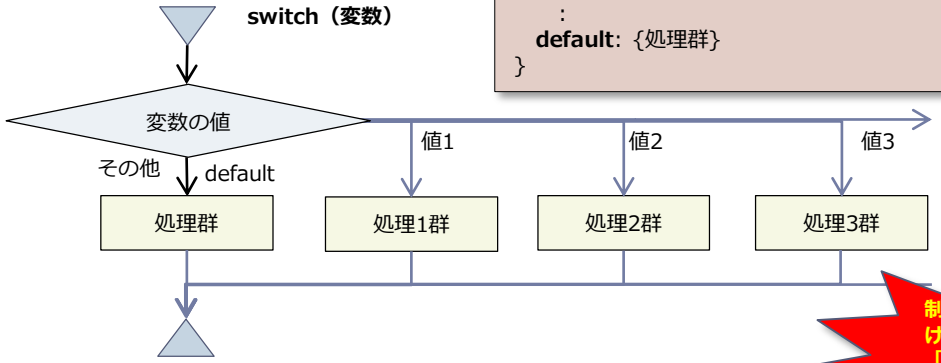
分岐＆反復




```
while(条件) {処理群} ;
```

■switch文

分岐

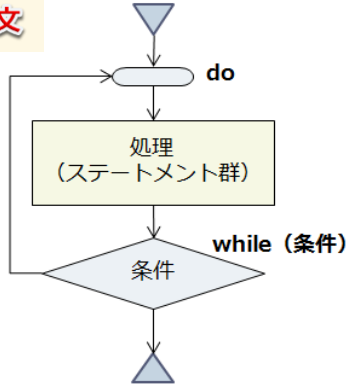



```
switch（変数）
{ case 値1: {処理1群}; break;
  case 値2: {処理2群}; break;
  case 値3: {処理3群}; break;
    :
  default: {処理群}
}
```

制御文から抜け出るには、「break」文を利用する

■do-while文

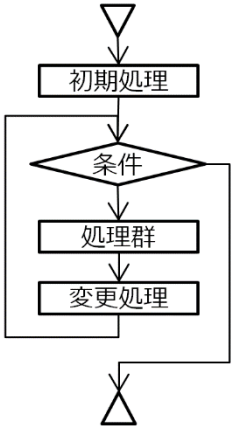
分岐＆反復




```
do {処理群}
while（条件） ;  //条件が真の場合処理群を実施
```

■for文

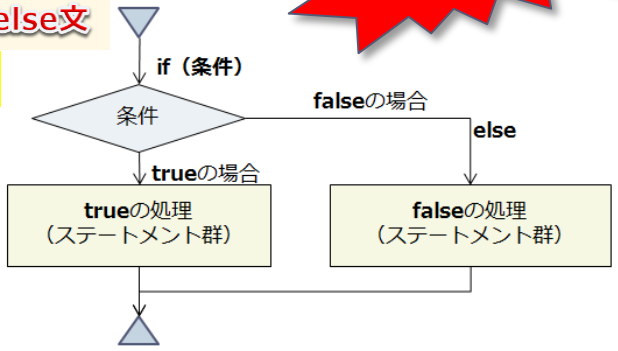
分岐＆反復




```
for(初期処理；条件；変更処理) { 処理群 }
```

■if-else文

分岐




```
if（条件） {処理群a} // 条件が真のとき処理群aを実行
else {処理群b} // 条件が偽のとき処理群ｂを実行
```

3．表記法とデータ・演算子

重要

表記法とデータ、演算子

【空白文字例】
```
x=123;    // 空白文字なし
x = 123;  // 空白文字あり
```
この場合、空白は自由に入れられる

【注意すべき空白文字例】
```
if(SENSOR_1==1 );  //エラーなし
if(SENSOR_1 == 1); //エラーなし
if(SENSOR_1 = = 1); //エラー発生
```
この場合、比較演算子「==」に、空白は入れられない

【数値表現例】
```
124     // 10進数整数
0xA0    // 16進数整数（=160）
1.234   // 実数
```
基本は、10進数で表記
16進数を使うのは少ない

【文字列と文字の例】
```
"Arduino  &  Tabrain" //文字列定数
't'                   //文字定数
```
文字列はダブルクオート
文字はシングルクオートで囲む

【文字列と文字の宣言】
文字列は「String」
文字は「Char」

【変数名と初期値の設定例】
```
int x=0, y=2, z =3;   // 複数変数設定
string s = "BricxCC & NXC" ;
float d=19.5602;
```
【基本の表記法】
型 変数名=初期値；
型には、**int**, **string**, **float** など
「=初期値」は省略可能（初期値なし）

配列は、データ表現のひとつ
使う場面が出てきた場合には強力

【配列の利用例】
```
int marray [];  // １次元整数配列（データサイズは不定）
bool ts[][];  // ２次元ブーリアン配列（データサイズは不定）
int x[] = { 1, 2, 3, 5, 7, 9 }; // 初期設定（サイズ６）
int y[][]={{2,3,1,}{2,3,4}};//2次元配列初期設定サイズ（3×2）
string s[5]; // １次元配列、サイズ５、初期設定なし
```

プリプロセッサ(コンパイル前に置き換え)
```
#include    // ヘッダーファイル呼び出し
#define     // 定数などの置き換え
```
よく使うこの２つは覚えること
特に「#define」は重要

キーワードを覚える
・bool、 break、byte、case、char、const, do、if-else、false、for、intなど

キーワードは、エディタの中で、文字列の色が変わるので、一目で分かる。
（注意：変数には利用できない）

【演算子】を覚える（特に◎は重要）
- **算術演算子**◎
  +、－、＊、/、%など
- **関係演算子**◎
  ==、!=、＞、＜、＞＝、＜＝
- **論理演算子**◎
  &、｜、^、||、&&、
- **代入演算子**◎
  +=、-=、*=、/=、%=
- **ビット**
  >>、<<
- **条件**
  ?:

【演算子の使い方例】
```
x = 5/y ＋（z＊2－6）；
if (x != y ) {z=5};
x = 5^ y ;
x += 3 ;  // x=x+３のこと
```

4．組込み関数群

重要

【シリアルモニタ表示関数】
```
Serial.begin(baud);  //初期設定
   baud: 通信速度（ボーレート）
Serial.println(value);          //改行有
Serial.println(value, base); //改行有
Serial.print(value);            // 改行無
Serial.print(value,base);    // 改行無
  value： 出力する値
   base：DEC  10進数
         HEX   16進数
         BIN   2進数
```

【デジタルI/O設定関数】
```
pinMode(pin, mode);
 pin： ピン番号
 mode： 読込み（INPUT_PULLUP,INPUT）
        書込み （OUTPUT）
    ※ modeが INPUTの場合は省略可
```
【デジタル読込み（入力）関数】
```
digitalRead(pin);
pin: ピン番号
 戻り値： HIGH / LOW
```
【デジタル書込み（出力）関数】
```
digitalWrite(pin, value);
  pin：ピン番号
  value：出力する値（HIGH / LOW）
```
電源とGNDに活用可能

【アナログ読込み（入力）関数】
```
analogRead(pin);
  pin：ピン番号
  戻り値： 0～1023
```
【アナログ書込み（出力）関数】PWM
```
analogWrite(pin, value);
  pin：ピン番号
  value：出力する値（0～255）
```

【時間関数】
```
millis();  // 実行時からのミリ秒数
micros();  // 実行時からのマイクロ秒数
delay(tm); //  待機時間（tm: ミリ秒）
delayMicroseconds(tm);  同上（tm:マイクロ秒）
```

【音関連関数】
```
tone(pin,hz,tm);  // トーン（音）発生
   pin: ピン番号
   hz:  周波数(Hz)
   tm:  発生長さ（ミリ秒：省略可）
noTone();  // トーン（音）の中止
```

【値変換関数】
```
map(val, smin, smax, tmin, tmax);
  val: 変換する元の値（整数）
 smin: 元の値の最小値（整数）
 smax: 元の値の最大値（整数）
 tmin: 返還後の最小値（整数）
 tmax: 変換後の最大値（整数）
 ex: val の値が、0～1023の値で出てきたものを、
0 V～5 Vに変換する場合
 map( val, 0, 1023, 0, 5 );
 ただし、すべて整数扱いとなる。
```

```
// 余談
void setup()
{
  Serial.begin(9600);
  byte i=-1;
  char j=-1;
  Serial.println((int)i); // 255
  Serial.println((int)j); // -1
}
void loop(){}
```

【文字列関数群の使い方１】
```
int len = mystring.length();
char x = mystring.charAt(2);
```

【文字列関数群の使い方２】
```
String s = “abcdefgh”;
Serial.println(s.substring(3,6)) ; // def と表示
```

【文字列結合処理】
```
String a=“456”,
       b=String( “123”+ a + “789”);
```

【文字列の処理事例】
```
 String a= "abcdefg";
  int x = 1234;
  Serial.println(String(" a = " +a));
  a = String(x,DEC);
  Serial.println(" x = "+ a);
```

【char型→String型変換】
```
char  oldstring[] = “string array”;
String newstring = String(oldstring);
```

【String型→char型変換】
```
 String str = "hello world";
 char  chr[str.length()];
 str.toCharArray(chr,str.length()+1);
```

ピン（ポート）番号は、
デジタル入出力（GPIO）の場合：　0～19
アナログ入力（AD)の場合　：A0 ～ A5
アナログ出力（PWM)の場合　：3,5,6,9,10,11
（Arduino UNO・ボードの場合）

【参考】Web上で
Arduino リファレンス
参照

5．キーワード等（一部）

重要

Arduinoで使う主なキーワード一覧

利用関数	内容	備考
setup	初期設定関数	必須関数
loop	繰り返し関数	必須関数
pinMode	デジタル入出力ポート設定	デジタル入出力必須
digitalWrite	デジタル出力（HIGH/LOW）	引数：ピン番号、値
digitalRead	デジタル入力（HIGH/LOW）	引数：ピン番号
analogWrite	アナログ出力(0~255)	引数：ピン番号：値
analogRead	アナログ入力(0~1023)	引数：ピン番号
pulseIn	パルス検知（HIGH/LOW）	引数：ピン番号、値
tone	トーン関数（周波数）	引数：ピン番号、値
millis	時間関数（ミリ秒）	引数なし
micros	時間関数（マイクロ秒）	引数なし
delay	待機時間（ミリ秒）	引数：時間
delayMicroseconds	待機時間（マイクロ秒）	引数：時間
map	範囲置換	<引数別途参照>
Serial.begin	シリアル通信速度設定	初期設定
Serial.print	シリアル出力	引数：値
Serial.println	シリアル出力（改行付）	引数：値
EEPROMread	EEPROM読込み	引数：番地
EEPROMwrite	EEPROM書込み	引数：番地、値

キーワード	内容	備考
HIGH/LOW	5V（=1）/ 0V（=0）	
true/false	真（=1）/ 偽（=0）	
INPUT	pinMode引数（入力）	pinMode省略可
INPUT_PULLUP	pinMode引数（入力）	プルアップ抵抗付
OUTPUT	pinMode引数（出力）	

文 法	内容	備考
文	{} と；で区切る	
関数	関数/手続き	
変数	グローバル/ローカル	
コメント・空白	//、/*　*/、半角空白	

型（タイプ）	内容（容量）	備考
boolean	ブーリアン（1バイト）	true/false
char	文字（1バイト）	‘a’･･･’z’など
byte	バイト(1バイト)	
int	整数（2バイト）	
long	整数（4バイト）	
float	実数（4バイト）	double でも可
void	型なし	

制御文	内容	備考
for文	反復	
while文;	分岐＋反復	
if-else文	分岐	
do-while文	反復＋分岐	
switch	分岐	

演算子	内容	備考
+,-,*,/,%,++,--	算術	
==,!=,>,<,>=,<=	関係	
&,|,!,||,&&	論理	

その他	内容	備考
数字（実数、整数）	5,1.25,1.2E2など	
文字・文字列	char ・ char[]	文字列は配列利用
配列	一次から多次配列	

第１４章
補足資料

【補足1】I2C_LCD.inoライブラリについて

```
#define I2Cadr 0x3e  // 固定
byte contrast = 30;  // コントラスト(0～63)

void lcd_init(void) {// I2C_LCDの初期化
   Wire.begin();
  lcd_cmd(0x38); lcd_cmd(0x39); lcd_cmd(0x4); lcd_cmd(0x14);
  lcd_cmd(0x70 | (contrast & 0xF)); lcd_cmd(0x5C | ((contrast>>4) &0x3));
  lcd_cmd(0x6C); delay(200); lcd_cmd(0x38); lcd_cmd(0x0C); lcd_cmd(0x01);
  delay(2);
}

void lcd_cmd(byte x) {// I2C_LCDへの書き込み
  Wire.beginTransmission(I2Cadr);
  Wire.write(0x00); // CO = 0,RS = 0
  Wire.write(x);
  Wire.endTransmission();
}
void lcd_clear(void) {
  lcd_cmd(0x01);
}
void lcd_DisplayOff() {
  lcd_cmd(0x08);
}
void lcd_DisplayOn() {
  lcd_cmd(0x0C);
}

void lcd_contdata(byte x) {
  Wire.write(0xC0); // CO = 1, RS = 1
  Wire.write(x);
}

void lcd_lastdata(byte x) {
  Wire.write(0x40); // CO = 0, RS = 1
  Wire.write(x);
}
```

タブ画面に、このモジュールを読み込んでおいて利用

```
// 文字の表示
void lcd_printStr(const char *s) {
  Wire.beginTransmission(I2Cadr);
  while (*s) {
    if (*(s + 1)) {
      lcd_contdata(*s);
    } else {
      lcd_lastdata(*s);
    }
    s++;
  }
  Wire.endTransmission();
}

// 表示位置の指定
void lcd_setCursor(byte x, byte y) {
  lcd_cmd(0x80 | (y * 0x40 + x));
}
```

■ 関数の説明

関数群	概要説明
lcd_init()	I2C_LCDの初期化
lcd_clear()	画面消去
lcd_DisplayOff()	画面の非表示
lcd_DisplayOn()	画面の表示
lcd_printStr(str)	文字列表示 str：表示する文字列
lcd_setCursor(x,y)	カーソル位置設定 x: 文字カラム(0～7) y: 行数（0～1）

【補足２】Arduinoトラブルシューティング

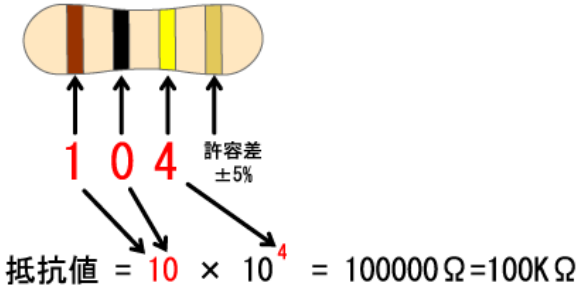
Arduinoトラブルシューティングの参考サイト
原本：http://arduino.cc/en/Guide/Troubleshooting
翻訳：http://garretlab.web.fc2.com/arduino_guide/trouble_shooting.html#toc1

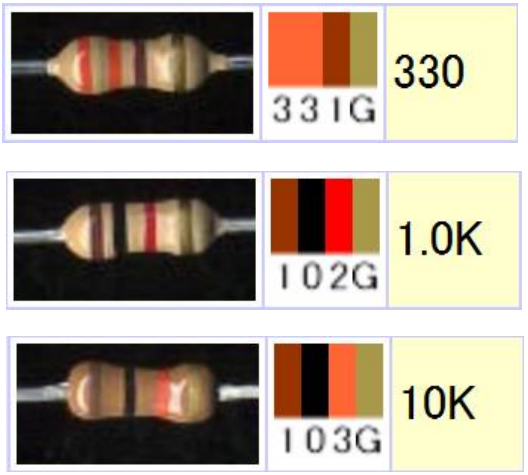
1 Arduinoボードにプログラムをアップロードできない
2 (Mac OS Xで)"Build folder disappeared or could not be written"が出る
3 MacでJavaを更新した後，Arduinoソフトウェアを起動できない
4 プログラムをコンパイルするときに，java.lang.StackOverflowErrorが出る
5 (Arduino Diecimilaより古い場合)Arduinoボードに外部電源から電力を供給したときにスケッチが開始しない
6 (Windowsで)プログラムをアップロードするときにArduinoソフトウェアが固まる
7 Arduinoボードが起動しない(緑の電源LEDが点灯しない)
8 Diecimilaがスケッチを開始するのに長時間(6から8秒)かかる
9 WindowsでArduino.exeを実行したときにエラー発生
10 古いMac OS XでArduinoソフトウェアが動かない
11 Arduinoソフトウェアを起動するとき，ネイティブライブラリであるlibrxtxDerial.jnilib 関連して，UnsatisfiedLinkError が出る
12 “Could not find the main class.”というエラー発生
13 Windowsでcygwinと競合する
14 (Windowsで)Arduinoソフトウェアの起動や Tools メニューを開くのに時間がかかる
15 ArduinoボードがTools > Serial Portメニューに現れない
16 (Macで)コードをアップロードしたりシリアルモニタを使うときに，gnu.io.PortInUseExceptionが出る
17 FTDI USBドライバの問題
18 Arduinoボードの電源を入れたときやリセットしたときにスケッチが開始しない
19 スケッチのアップロードは成功したように見えるのに，何も起こらない
20 スケッチの大きさを小さくするには
21 analogWrite()を3，5，6，9，10，11番以外のピンに対して実行したときに，PWM(アナログ出力)が得られない
22 関数や変数が未定義というエラー発生
23 スケッチをアップロードする際に，invalid device signatureというエラー発生

【補足３】抵抗値のカラー識別

数値	色	覚え方
0	黒	黒い礼（0）服
1	茶	茶を１杯
2	赤	赤いに（2）んじん
3	橙	ダイダイみ（3）かん
4	黄	四季（黄）の色

数値	色	覚え方
5	緑	ミドリゴ
6	青	あおむし
7	紫	紫式部
8	灰	ハイヤー（8）
9	白	ホワイトク（9）リスマス





【補足４】知っていると便利なこと①

- RAMサイズを知る方法、節約する方法
  - 下記のサイトにTipsがある：
    http://arduino.sugakoubou.com/tips
    - メモリを制限すれすれに使用している場合、上記のサイトにあるTipsで具体的なサイズが分かる
  - Arduino UNOは、ATmega328P を搭載
    - Flash ROM　　32KB(うち4KBはブートローダが使用、実質28KB)
    - SRAM　　2KB　←　あっという間に使い切り、意味不明なバグに！
    - EPROM　　1KB
    - 動作周波数　　16MHz(約16MIPS)

【メモリサイズ節約の方法】
- 大きなサイズの変数は、ローカルではなく、グローバルとして定義する
- 必要最低限だけのサイズを確保する
- 実数(float)はできるだけ使用せず、整数(int or int32_t)で代用する

【補足５】知っていると便利なこと②

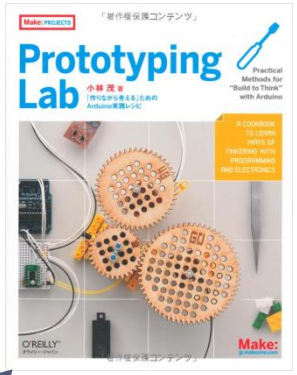
- **情報源のサイト**

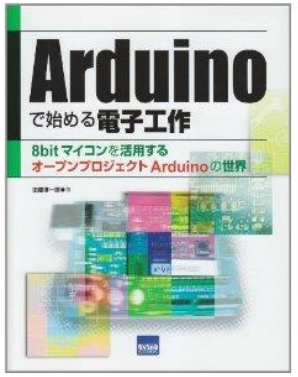
  - **Arduino日本語リファレンス**
    http://www.musashinodenpa.com/arduino/ref/

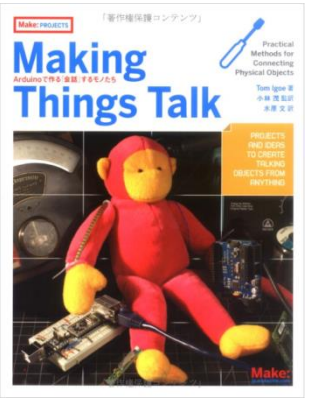
  - **Arduino Wiki**
    http://www.musashinodenpa.com/wiki/

  - **建築発明工作ゼミ2008**
    Arduino　http://kousaku-kousaku.blogspot.jp/2008/07/arduino.html
    Processing　http://kousaku-kousaku.blogspot.jp/2008/07/processing_10.html

【補足６】Arduino関連の参考本 ①（和書）



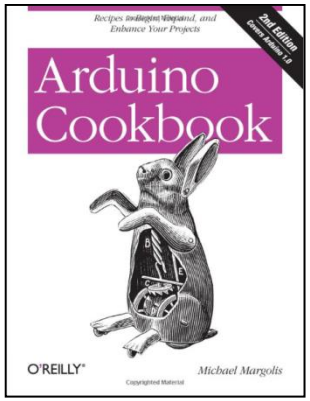


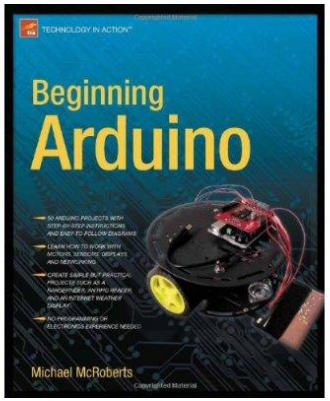


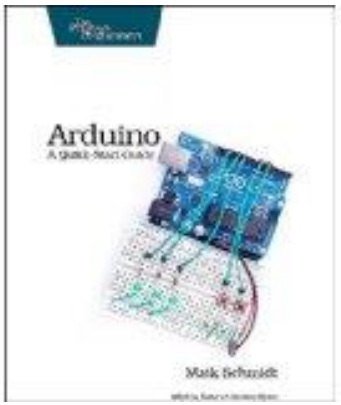
日本のArduino先駆者
IAMAS 准教授小林茂先生

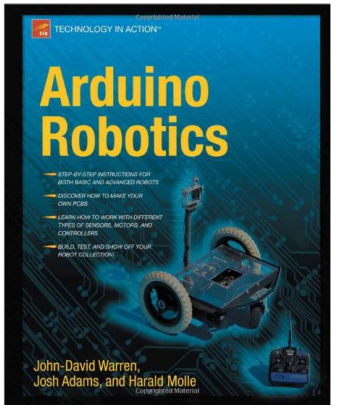
学研 金子茂氏と
スイッチサイエンス社
金本茂社長関与

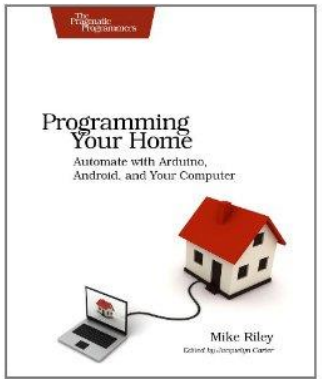
【補足６ 】Arduino関連の参考本 ②（洋書）

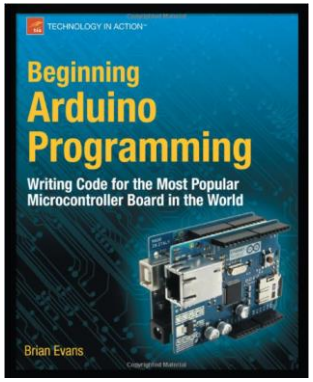


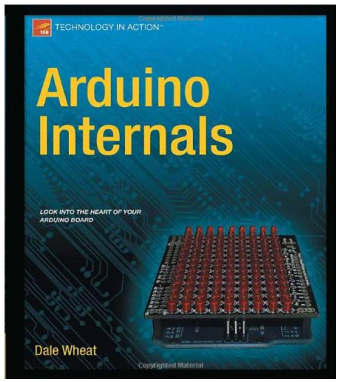


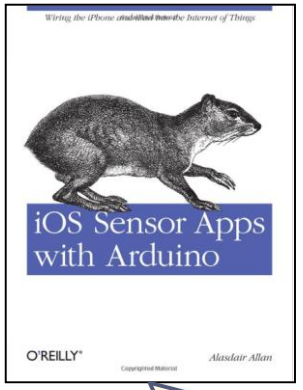


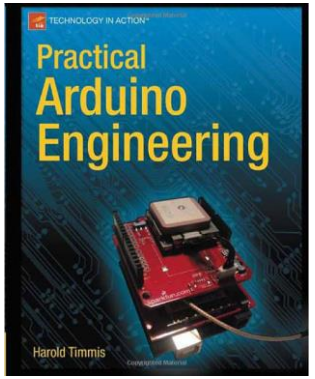












Arduino関係の洋書
今後も多く販売予定

【補足7 】総合サンプルスケッチの紹介①

本サンプルスケッチ「IoTABS3demo」は、以下の機能をリセットスイッチで切り替えて利用する総合的なテストプログラムとなります。

「IoTABS3_demo_UNO」（赤外線学習リモコン読込み操作）
1）リセットボタンと一緒にタクトスイッチを押した状態で、リセットボタンを先に離し、次にタクトスイッチを押すと、５つのコマンドの学習リモコンを行います。（コマンド学習）
2）テストプログラムの最後に、「ｾｷｶﾞｲｾﾝ」のコマンドがあります。こちらで、学習した５つのリモコン操作を操作できます。

1）「ｵﾝﾄﾞｾﾝｻ」温度センサのLCD表示とLED点滅
2）「ﾋｶﾘｾﾝｻ」光センサのLCD表示とLED点滅
3）「ｵﾄｾﾝｻ」音センサによるLCD表示（平均値表示付）とLED点滅
4）「ﾃﾋﾞｮｳｼ」音センサを使った手拍子認識によるLCD表示とLED点滅
5）「ﾃｲｺｳ」可変抵抗器を使ったLCD表示とLED点滅
6）「ﾀｲﾏｰ」可変抵抗器とスイッチを使ってタイマーによるLCD表示とLED点滅、それにスピーカによるアラーム発生
7）「LED ON」 LEDの点滅をデモ
8）「ｷｮﾘｾﾝｻ」超音波距離センサを使ったLCD表示とLED点滅、それにスピーカによる近接アラーム
9） 「ﾃﾙﾐﾝ」 超音波距離センサを使って音の諧調とLED点灯を変える
10）「ﾒﾛﾃﾞｨｰ」スピーカによるメロディ「チューリップ」
11）「ｾｷｶﾞｲｾﾝ」学習リモコンによる赤外線LED操作（可変抵抗器を使ってコマンド選択）

以上のように、温度センサ、湿度センサ、超音波距離センサ、可変抵抗器、スイッチ、音センサ、スピーカ、LED、LCDを使ったデモとなります。

サンプルスケッチは、以下のところからダウンロード（Arduino UNO R3用）できます。
http://tabrain.jp/tabs/IoTABS3demo.zip

【補足7 】総合サンプルスケッチの紹介②

IoTABS4_DEMO_UNO_3GIM.ino

初期設定（１）

```
#include <Wire.h>
```

I2C_LCD初期設定

```
// #no.5 I2C_LCD set
#define sdaPin A4    // ArduinoA4
#define sclPin A5    // ArduinoA5
```

超音波距離センサ設定

```
#define TrigPin 13    // Sonar Trig
#define EchoPin 12    // Sonar Echo
#define CTM 10
```

加速度センサ設定

アナログセンサ設定

```
#define TMP_Pin A1
#define LGT_Pin A0
#define Mic_Pin A2
#define Reg_Pin A3
```

```
// SW 2
#define AccX_Pin A1
#define AccY_Pin A2
#define AccZ_Pin A3
```

デジタルセンサ設定

```
#define InfRim_Pin 11
#define Tlt_Pin 10
#define Spk_Pin 9
#define But_Pin 8
//#define Vib_Pin 12
//#define Spk_Pin 11
```

LED設定

```
#define Led1Pin 2
#define Led2Pin 3
#define Led3Pin 4
#define Led4Pin 5
#define Led5Pin 6
#define Led6Pin 7
```

初期設定（２）

ドレミの設定

```
#define TC 0 //  ド
#define TD 1 //  レ
#define TE 2 //  ミ
#define TF 3 //  ファ
#define TG 4 //  ソ
#define TA 5 //  ラ
#define TB 6 //  シ
#define TX 7
```

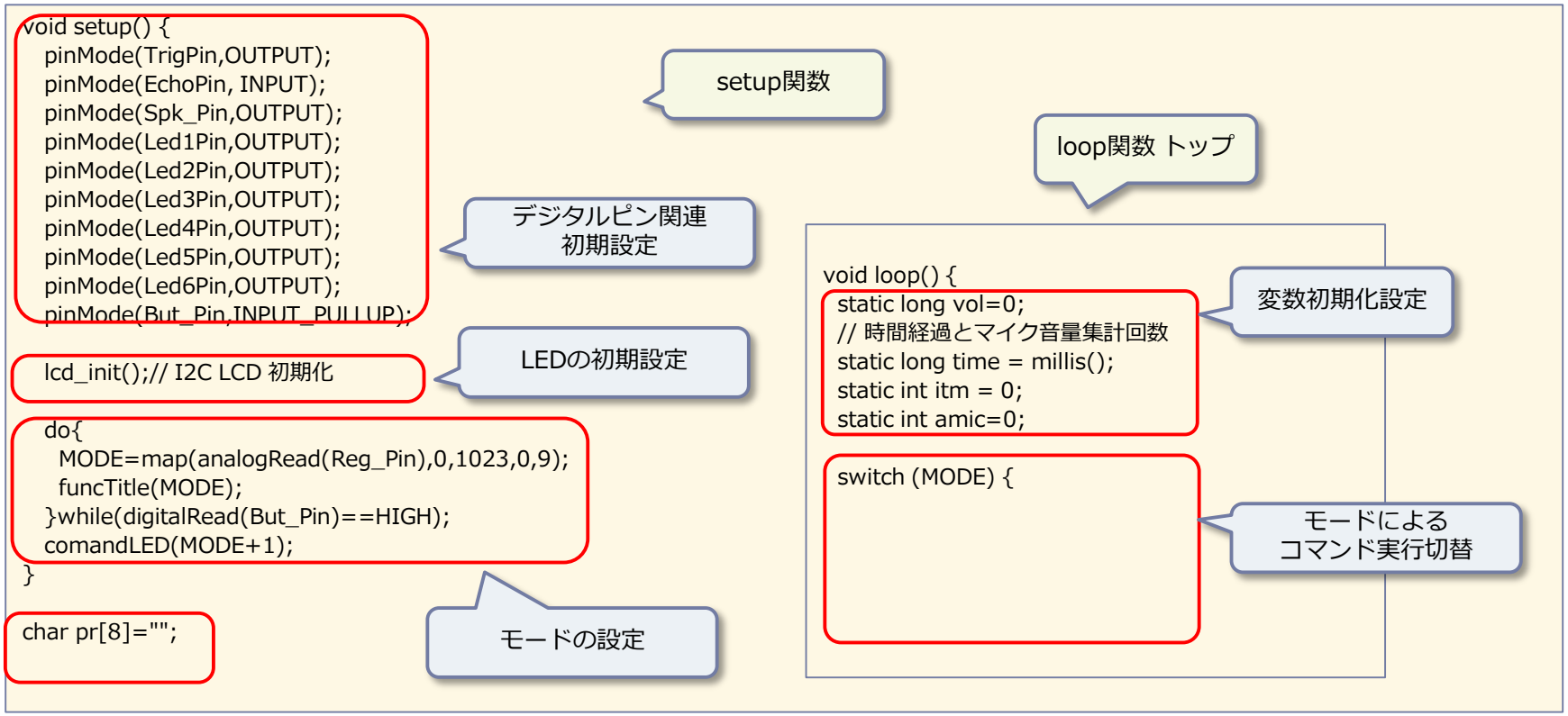
チューリップのメロディ設定

```
int fq[]={262,294,330,349,392,440,494,0}; // 音階の周波数（Hz)ドレミ・・・
int mo[45][2]={{TC,500},{TD,500},{TE,1000},{TX,1000}, {TC,500},{ TD,500}, {TE,1000},{TX,1000},
        {TG,500},{TE,500},{TD,500}, {TC,500},{ TD,500}, {TE,500},{TD,1000},{TX,1000},
        {TC,500},{TD,500},{TE,1000}, {TX,1000},{TC,500},{ TD,500}, {TE,1000},{TX,1000},
        {TG,500},{TE,500},{TD,500}, {TC,500},{ TD,500}, {TE,500},{TC,1000},{TX,1000},
        {TG,500},{TG,500},{TE,500},{TG,500},{TA,500},{TA,500},{TG,1000},{TX,1000},
        {TE,500},{TE,500},{TD,500},{TD,500},{TC,1000}};
```

モード変数

```
byte MODE;
```

【補足7 】総合サンプルスケッチの紹介③



※ 可変抵抗器を使いLCDに表示されたコマンドを
選択し、タクトスイッチを押す

【補足7 】総合サンプルスケッチの紹介④

```
//+++++++++++++++++ 温度センサ ++++++++++++
 case (0):
  while(1) {// temp
  int val = (int)(analogRead(TMP_Pin)*4.888-600);
  sprintf(pr," %3d.%d C",val/10,abs(val%10));
  lcd_setCursor(0,1);
  lcd_printStr(pr);
    digitalWrite(Led1Pin,(val>50)?HIGH:LOW);
    digitalWrite(Led2Pin,(val>100)?HIGH:LOW);
    digitalWrite(Led3Pin,(val>150)?HIGH:LOW);
    digitalWrite(Led4Pin,(val>200)?HIGH:LOW);
    digitalWrite(Led5Pin,(val>250)?HIGH:LOW);
    digitalWrite(Led6Pin,(val>300)?HIGH:LOW);
   delay(500);}
```

温度センサ値変換式

空読みが必要
（参照：P59「温度センサ」）

センサ値のLCD表示

センサ値のLED点滅

```
//+++++++++++++++++ 照度センサ ++++++++++++
 case (1):
    while(1) {// light
    float val = (long)analogRead(LGT_Pin)*5000/1023;
    long x = val*100 /290;
    lcd_setCursor(0,1);
    sprintf(pr,"%5d ",x);
    lcd_printStr(pr);
    digitalWrite(Led1Pin,(x<200)?HIGH:LOW);
    digitalWrite(Led2Pin,(x<400)?HIGH:LOW);
    digitalWrite(Led3Pin,(x<600)?HIGH:LOW);
    digitalWrite(Led4Pin,(x<800)?HIGH:LOW);
    digitalWrite(Led5Pin,(x<1000)?HIGH:LOW);
    digitalWrite(Led6Pin,(x<1200)?HIGH:LOW);

    delay(100); };
```

照度センサ値変換式

センサ値のLCD表示

センサ値のLED点滅

【補足7 】総合サンプルスケッチの紹介⑤

```
//+++++++++++++++++ 音センサ（マイク） ++++++++++++
  case (2): // MIC
   for(int i=0; i<100; i++) {
      int v = analogRead(Mic_Pin);
      vol += (v>0?v:-v); delay(10);}
   vol /= 100;
   while(1){
    // Mic
   int mic = 0;
   for(int i=0; i<10; i++) {
     int v = analogRead(Mic_Pin)-vol;
     mic += (v>0)?v:-v;}
   mic /= 10;
   amic += mic; itm++;
   if(time+3000<millis()) {
      amic /= itm;
      lcd_setCursor(0,1);
      sprintf(pr,"%4d",amic);
      lcd_printStr(pr);
      amic = 0; itm=0;
      time=millis();
   }
   lcd_setCursor(4,1);
   sprintf(pr,"%4d ",mic);
   lcd_printStr(pr);
   digitalWrite(Led1Pin,(mic>50)?HIGH:LOW);
   digitalWrite(Led2Pin,(mic>100)?HIGH:LOW);
   digitalWrite(Led3Pin,(mic>150)?HIGH:LOW);
   digitalWrite(Led4Pin,(mic>200)?HIGH:LOW);
   digitalWrite(Led5Pin,(mic>250)?HIGH:LOW);
   digitalWrite(Led6Pin,(mic>300)?HIGH:LOW);

    delay(100);};
    break;
```

音の平均値取得

音センサ値の取得

3秒間の音センサの平均値取得　LCD表示

音センサ値取得 LCD表示

音センサ値による LED点滅

```
//+++++++++++++++++ 手拍子カウント ++++++++++++
  case (3): // MIC
   while(1){
     lcd_setCursor(0,1);
     lcd_printStr("Ready...");
     int nclup = checkSound();
     lcd_setCursor(0,1);
     sprintf(pr,"clup= %2d",nclup);
     lcd_printStr(pr);
   digitalWrite(Led1Pin,(nclup>0)?HIGH:LOW);
   digitalWrite(Led2Pin,(nclup>1)?HIGH:LOW);
   digitalWrite(Led3Pin,(nclup>2)?HIGH:LOW);
   digitalWrite(Led4Pin,(nclup>3)?HIGH:LOW);
   digitalWrite(Led5Pin,(nclup>4)?HIGH:LOW);
   digitalWrite(Led6Pin,(nclup>5)?HIGH:LOW);
     delay(2000);
   };
```

手拍子カウント モジュール

手拍子カウント数 LED表示

手拍子のカウント数 によるLED点滅

【補足7 】総合サンプルスケッチの紹介⑥

```
//+++++++++++++++++ 可変抵抗器 ++++++++++++
   case(4): //Volume
//    Serial.println("Volulme");
    while(1) {
      vol=analogRead(Reg_Pin);
//      Serial.println(vol);
    digitalWrite(Led1Pin,(vol>146)?HIGH:LOW);
    digitalWrite(Led2Pin,(vol>292)?HIGH:LOW);
    digitalWrite(Led3Pin,(vol>438)?HIGH:LOW);
    digitalWrite(Led4Pin,(vol>584)?HIGH:LOW);
    digitalWrite(Led5Pin,(vol>730)?HIGH:LOW);
    digitalWrite(Led6Pin,(vol>876)?HIGH:LOW);
    lcd_setCursor(4,0);
    sprintf(pr,"%3d",map(vol,0,1023,0,100));
    pr[3]='%';
    lcd_printStr(pr);
    lcd_setCursor(0, 1);
    sprintf(pr,"%5dOhm", map(vol,0,1023,0,10000));
    lcd_printStr(pr);
    delay(100);  }
```

可変抵抗値読み込み

可変抵抗器の
値によるLED点滅

可変抵抗器の値
LCD表示

【補足7 】総合サンプルスケッチの紹介⑦

```
//+++++++++++++++++ タイマー ++++++++++++
//  可変抵抗器を使ったタイマー
  case(5):
  while(1){
    int limtime;
    do {// set timer
      limtime=set_timer();
      sprintf(pr," %2d:%2d",limtime/60, limtime%60);
      lcd_setCursor(0,1);
      lcd_printStr(pr);
      sprintf(pr,"%4d",limtime);
      lcd_setCursor(4,0);
      lcd_printStr(pr);
    }while(digitalRead(But_Pin)==HIGH);
    long time=millis();
    int stime,ostime=0;
```

時間設定

時間設定値LCD表示

ボタン押し

```
do{ // count down
      stime=(millis()-time)/1000;
      if(ostime!=stime) {
        sprintf(pr," %2d:%2d",(limtime-stime)/60,(limtime-stime)%60);
        lcd_setCursor(0,1);
        lcd_printStr(pr);
        ostime=stime;
        int val=map(stime,0,limtime,6,0);
        digitalWrite(Led1Pin,(val>0)?HIGH:LOW);
        digitalWrite(Led2Pin,(val>1)?HIGH:LOW);
        digitalWrite(Led3Pin,(val>2)?HIGH:LOW);
        digitalWrite(Led4Pin,(val>3)?HIGH:LOW);
        digitalWrite(Led5Pin,(val>4)?HIGH:LOW);
        digitalWrite(Led6Pin,(val>5)?HIGH:LOW);
      }
    }while(time+(long)limtime*1000>millis());
    digitalWrite(Led1Pin,LOW);
    lcd_setCursor(0,1);
    lcd_printStr("TimeOver");
    do{ // speaker
      tone(Spk_Pin,400,500);
      delay(500);
      noTone(Spk_Pin);
      delay(200);
    }while(digitalRead(But_Pin)==HIGH);
  }
```

カウントダウン

時間のLCD表示

時間のLED表示

タイムオーバー処理

```
long set_timer(){
  int reg = analogRead(Reg_Pin);
  if(reg<375) return(map(reg,0,374,1,60));//1-60s@1s
  else if( reg<600 )
    return(60+5*map(reg,375,599,0,36));//1-3m@5s
  else if( reg<650 )
    return(180+15*map(reg,600,649,0,8));//3-5m@15s
  else if( reg<712 )
    return(300+30*map(reg,650,711,0,10));//5-10m@30s
  else return( 600+60*map(reg,712,1023,0,50));//10-60m@1m
}
```

時間設定関数
可変抵抗器を利用
0 －60秒：1秒間隔
1ー３分：5秒間隔
3 － 5分：15秒間隔
5 － 10分：30秒間隔
10－60分：1分間隔

【補足7 】総合サンプルスケッチの紹介⑧

```
//+++++++++++++++++ 3軸加速度センサ ++++++++++++
//  スイッチを中央に切り替え、終わったら左に切り替え
  case (6):
   while(1){// Accオフセット調整（４号機）
    int x = (long)analogRead(AccX_Pin)*2500/1023-855;
    int y = (long)analogRead(AccY_Pin)*2500/1023-875;
    int z = (long)analogRead(AccZ_Pin)*2500/1023-855;
    lcd_setCursor(4,0);
    sprintf(pr,"%4d",x);
    lcd_printStr(pr);
    lcd_setCursor(0,1);
    sprintf(pr,"%4d%4d",y,z);
    lcd_printStr(pr);
    if (x>250 || y>250 || z >250) tone(Spk_Pin,1000,100);
    else if (x>200 || y>200 || z >200) tone(Spk_Pin,500,100);
    else if (x>150 || y>150 || z >150) tone(Spk_Pin,200,100);
//else noTone(Spk_Pin);
    delay(50);}
```

補正値設定

LCD表示

異常加速度でのアラーム

※3軸加速度センサの補正値処理は、各製品ごとに異なることから別途調整必要。

```
//+++++++++++++++++ 超音波距離センサ ++++++++++++
  case (7):
//     Serial.println("Distance");
   while(1){// Distance
    digitalWrite(TrigPin, HIGH);
    delayMicroseconds(CTM);
    digitalWrite(TrigPin, LOW);
    float dis =(float)pulseIn(EchoPin,HIGH)*0.017;
    if(dis<10) tone(Spk_Pin,400,60);
    digitalWrite(Led1Pin,(dis>10)?HIGH:LOW);
    digitalWrite(Led2Pin,(dis>12)?HIGH:LOW);
    digitalWrite(Led3Pin,(dis>14)?HIGH:LOW);
    digitalWrite(Led4Pin,(dis>16)?HIGH:LOW);
    digitalWrite(Led5Pin,(dis>18)?HIGH:LOW);
    digitalWrite(Led6Pin,(dis>20)?HIGH:LOW);
    sprintf(pr,"%4d.%1dcm",(int)dis,(int)(dis*10.0)%10);
//    Serial.println( pr );
    lcd_setCursor(0, 1);
    lcd_printStr(pr);
    delay(100); };
```

HC-SR04センサの距離算出処理

LED点灯

LCD表示

【補足7 】総合サンプルスケッチの紹介⑨

```
//+++++++++++++++++ 傾斜センサ ++++++++++++
  case(8): //傾斜（TILT)センサ
 //   Serial.println("Tilt");
    while(1) {
      vol=digitalRead(Tlt_Pin);
//      Serial.println(vol);
   digitalWrite(Led1Pin,vol?HIGH:LOW);
   vol?tone(Spk_Pin,800,10):noTone(Spk_Pin);
   lcd_setCursor(0,1);
   lcd_printStr(vol?"  Off":"  On ");
   delay(100);
   }
```

チルトセンサ値

LED点滅

LCD表示

```
//+++++++++++++++++ メロディ ++++++++++++
   case(9):
   while(1) { //Melody チューリップ
     char melody[]={0xC1,0xAD,'-',0xD8,0xAF,0xCC,0xDF,' '};
     lcd_setCursor(0,1);
     lcd_printStr(melody);
     for(int i=0; i<45; i++){
      tone(Spk_Pin,fq[mo[i][0]],mo[i][1]);
      delay(500);}
     while(digitalRead(But_Pin)==HIGH);
    }
   }
}
```

メロディ発生

※メロディデータは、グローバルデータとして定義済（前ページ）

【補足7 】総合サンプルスケッチの紹介⑩

■ タイトルLCD表示（カタカナ表示）

```
void funcTitle( byte mode ) {
  char title[8];
  switch (mode) {
   case 0:{ char ttl[]={0xB5,0xDD,0xC4,0xDE,0xBE,0xDD,0xBB,' '}; strcpy(title,ttl);};  break; // ｵﾝﾄﾞ
   case 1:{ char ttl[]={0xCB,0xB6,0xD8,0xBE,0xDD,0xBB,' ',' '}; strcpy(title,ttl) ;};  break;//ﾋｶﾘｾﾝｻ
   case 2:{ char ttl[]={0xB5,0xC4,0xBE,0xDD,0xBB,' ',' ',' '}; strcpy(title,ttl) ;};  break;//ｵﾄｾﾝｻ
   case 3:{ char ttl[]={0xC3,0xCB,0xDE,0xAE,0xB3,0xBC,' ',' '}; strcpy(title,ttl) ;};  break;//ﾃﾋﾞｮｳｼ
   case 4:{ char ttl[]={0xC3,0xB2,0xBA,0xB3,' ',' ',' ',' ',' '}; strcpy(title,ttl) ;};  break;//ﾃｲｺｳ
   case 5:{ char ttl[]={0xC0,0xB2,0xCF,'-',' ',' ',' ',' ',' '  }; strcpy(title,ttl) ;};  break;//ﾀｲﾏｰ
   case 6:{ char ttl[]={0xB6,0xBF,0xB8,0xC1,' ',' ',' ',' ',' '}; strcpy(title,ttl) ;};  break;//ｶｿｸﾁ
   case 7:{ char ttl[]={0xB7,0xAE,0xD8,0xBE,0xDD,0xBB,' ',' '}; strcpy(title,ttl) ;};  break;//ｷｮﾘｾﾝｻ
   case 8:{ char ttl[]={0xC1,0xD9,0xC4,0xBE,0xDD,0xBB,' ',' '}; strcpy(title,ttl) ;};  break;//ﾁﾙﾄｾﾝｻ
   case 9:{ char ttl[]={0xD2,0xDB,0xC3,0xDE,0xA8,'-',' ',' '}; strcpy(title,ttl) ;};//ﾒﾛﾃﾞｨ
  }
  lcd_setCursor(0,0);
  lcd_printStr(title);
}
```

タイトル
LCD表示

■ コマンド LED点灯表示

```
void comandLED(byte cmd){
 for(int i=0; i<6; i++) digitalWrite(i+2,LOW);
 for(int i=0; i<10; i++) {
    if(cmd&0x1) digitalWrite(2,HIGH);
    if(cmd&0x2) digitalWrite(3,HIGH);
    if(cmd&0x4) digitalWrite(4,HIGH);
    if(cmd&0x8) digitalWrite(5,HIGH);
    if(cmd&0x10) digitalWrite(6,HIGH);
    delay(200);
    for(int j=2;j<7;j++){
      digitalWrite(j,LOW);}
    delay(100);
 }
}
```

コマンド番号を
LEDでビット表示

【補足7 】総合サンプルスケッチの紹介⑪

■ 手拍子のスケッチ

```
#define SoundLEV 150
boolean SoundSW = false;

int checkSound(void)          // 音センサーの制御モジュール（スレッド）
{
  int sw = 0;                 //   手拍子数（初期値）=0
  long tms;                   //  手拍子の音が鳴り終わった（しきい値以下）ときの時間
  long  sds, sde;//  初めの音声値、後の音声値
  tms = millis();//  現時点の時間を設定
  while(true) {
    //  delay(10);
    long sds=0;
    long sde=0;
    int val;
    for(int i=0; i<10; i++){
      val= analogRead(Mic_Pin)-512;
      sds +=(val>0?val:-val);
    }
    sds/=10;      // 初めの音声値
    //      Serial.print("s=");Serial.print(sds);
    delay(20);
    for(int i=0; i<10; i++){
      val= analogRead(Mic_Pin)-512;
      sde +=(val>0?val:-val);
    }
    sde/=10;        // 後の音声値
      char pr[100];
      //      snprintf(pr,100,"%4d,%4d",sds,sde);
      //      Serial.println(pr);
      //       Serial.print(" e=");Serial.println(sde);
      if( sds < SoundLEV && sde < SoundLEV )// sds と sde がともに低い音の場合
      {
        if((millis()-tms> 1200) && sw>0){
          SoundSW = false;
          return sw;
        }
        else if ( sw==0 ) tms=millis();
      } // 低い音が0.5秒以上続いた場合
      else if ( sds < SoundLEV && sde >= SoundLEV) // sdsが低く、sdeが高くなった場合
      {
        if ( SoundSW ){
          sw++ ;
        }// 既に高い音が鳴っていた場合　手拍子数を追加
        else {
          SoundSW=true;
          sw=1;
        }// 前回は音がなかった場合で、最初の音として１を設定
      }
      else if ( sds >= SoundLEV && sde < SoundLEV) // 高い音から、音が低くなった場合
      {
        tms = millis();
      }//  現在時間を設定
    }
  }
```

最初の音センサ値
平均値

最初の音センサ値
平均値

【補足7 】総合サンプルスケッチの紹介⑫

■ I2C_LCD.ino

[注意] 本スケッチ利用の際は、「＃include <Wire.h>」の宣言必要

```
#define I2Cadr 0x3e  // 固定
byte contrast = 30;  // コントラスト(0～63)

void lcd_init(void) {// I2C_LCDの初期化
   Wire.begin();
  lcd_cmd(0x38); lcd_cmd(0x39); lcd_cmd(0x4); lcd_cmd(0x14);
  lcd_cmd(0x70 | (contrast & 0xF)); lcd_cmd(0x5C | ((contrast>>4) &0x3));
  lcd_cmd(0x6C); delay(200); lcd_cmd(0x38); lcd_cmd(0x0C); lcd_cmd(0x01);
  delay(2);
}

void lcd_cmd(byte x) {// I2C_LCDへの書き込み
  Wire.beginTransmission(I2Cadr);
  Wire.write(0x00); // CO = 0,RS = 0
  Wire.write(x);
  Wire.endTransmission();
}
void lcd_clear(void) {
  lcd_cmd(0x01);
}

void lcd_DisplayOff() {
  lcd_cmd(0x08);
}

void lcd_DisplayOn() {
  lcd_cmd(0x0C);
}
```

関数群	概要説明
lcd_init()	I2C_LCDの初期化
lcd_clear()	画面消去
lcd_DisplayOff()	画面の非表示
lcd_DisplayOn()	画面の表示
lcd_printStr(str)	文字列表示 str：表示する文字列
lcd_setCursor(x,y)	カーソル位置設定 x: 文字カラム(0～7) y: 行数（0～1）

```
void lcd_contdata(byte x) {
  Wire.write(0xC0); // CO = 1, RS = 1
  Wire.write(x);
}

void lcd_lastdata(byte x) {
  Wire.write(0x40); // CO = 0, RS = 1
  Wire.write(x);
}

// 文字の表示
void lcd_printStr(const char *s) {
  Wire.beginTransmission(I2Cadr);
  while (*s) {
    if (*(s + 1)) {
      lcd_contdata(*s);
    } else {
      lcd_lastdata(*s);
    }
    s++;
  }
  Wire.endTransmission();
}

// 表示位置の指定
void lcd_setCursor(byte x, byte y) {
  lcd_cmd(0x80 | (y * 0x40 + x));
}
```

【補足7 】総合サンプルスケッチの紹介⑬

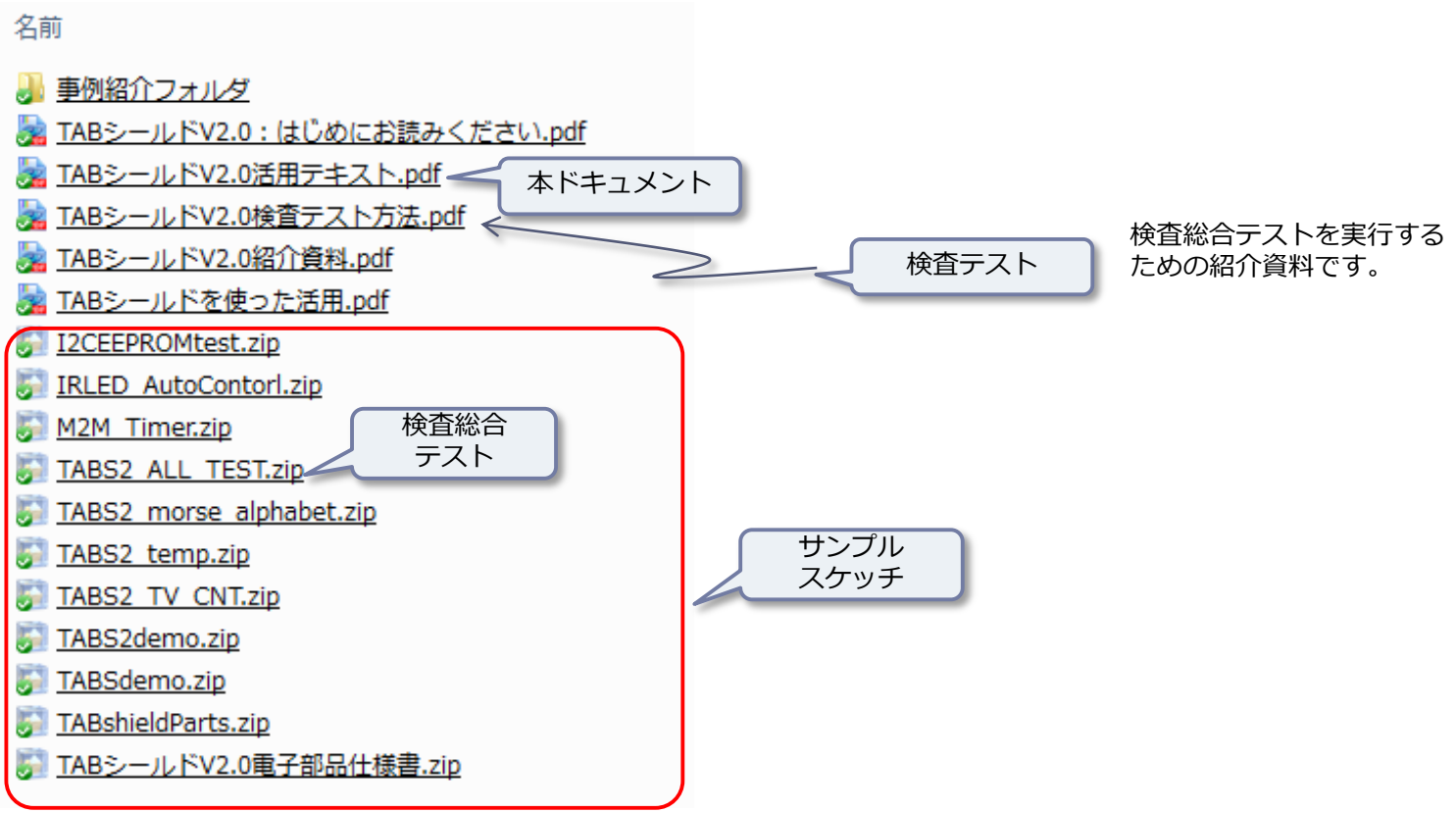
■ EEPROM.ino

```
#define I2C_EEPROM 0x57

void writeEEPROM(unsigned int eeaddress, byte data )
{

  Wire.beginTransmission(I2C_EEPROM);
  Wire.write((int)(eeaddress >> 8));   // MSB
  Wire.write((int)(eeaddress & 0xFF)); // LSB
  Wire.write(data);
  Wire.endTransmission();

  delay(5);
}

byte readEEPROM(unsigned int eeaddress )
{
  byte rdata = 0xFF;

  Wire.beginTransmission(I2C_EEPROM);
  Wire.write((int)(eeaddress >> 8));   // MSB
  Wire.write((int)(eeaddress & 0xFF)); // LSB
  Wire.endTransmission();

  Wire.requestFrom(I2C_EEPROM,1);

  if (Wire.available()) rdata = Wire.read();

  return rdata;
}
```

本スケッチ利用の際は、「＃include <Wire.h>」の宣言必要

関数群	概要説明
writeEEPROM(adr,data)	EEPROMへの書込み adr:アドレス（容量に応じたアドレス） data：データ（1バイト）
readEEPROM(adr)	EEPROMからの読み込み adr：アドレス（容量に応じたアドレス）

【補足８】 内容物まとめ



[コラム] 分かり易いプログラミング手法

1）コメントを利用
分かりにくい仕様などは、特にコメントをプログラムの中に記述する。関数などでは引数や戻り値も明確に記載する。

2）インデント（段下げ）を利用
インデントは、括弧などの位置づけを分かりやすくし、プログラムの実行の流れが見やすくなる。

3）分かり易い変数名・関数名を利用
変数名や関数名などは、分かり易い名前を付けることで、プログラムを理解しやすくできる。また大文字と小文字の使い分けも見やすくすることができる。

4）モジュール化対応
関数やサブルーチンによるモジュール化をすることで、プログラムを短くして、分かりやすくする。（**モジュール**とは、まとまりをもった機能部品のこと）

※CQ出版社「知的LEGO Mindstorms NXTプログラミング入門」から

もくじ

1．タイマー機能を使う
2．複数スケッチによるタブ画面
3．不揮発性メモリーEEPROMを使う
4．割込み機能を使う
5．シリアル通信機能を使う
6．ソフトウェアリセットの実現方法
7．ソフトウェアリセットと割込み処理の使い方
8．Arduino 電子部品利用早見表

第１５章
ちょっとしたチップス

「みんなのArduino入門」より

1．タイマー機能を使う

- Arduinoには、電源が入った時から、時間をカウントアップする機能がある。
- Arduino UNO では、0.004ミリ秒 （4マイクロ秒）単位で時刻を読み取る。

unsingned long millis() : Arduino上のプログラムが実行したときからの継続時間（ミリ秒）を返す
約50日間でオーバフローし、ゼロに戻る。　この時点で、戻り値：実行時からの時間（ミリ秒）を返す
unsingned long micros() : Arduino上のプログラムが実行したときからの継続時間（マイクロ秒）を返す
約70分間でオーバフローし、ゼロに戻る。ただし、Arduino UNO R3だと、4マイクロ秒間隔でカウントアップする。この時点で、戻り値：実行時からの時間（マイクロ秒）を返す

- また、プログラムをある間隔で中断（停止）する関数もある。

void delay(ms)：プログラムを指定した時間（m s ミリ秒）だけ中断
ms：待機時間（ミリ秒）、　戻り値：なし
void delayMicroseconds(us)：プログラムを指定した時間（usマイクロ秒）だけ中断
us：待機時間（マイクロ秒）、戻り値：なし

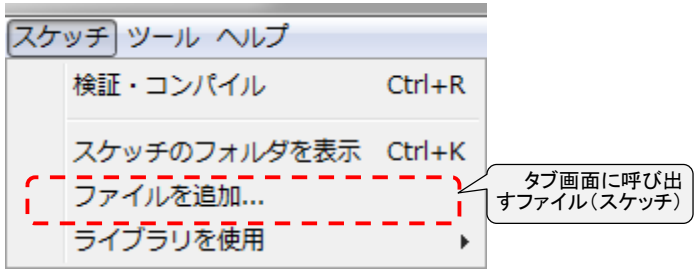
- 一定間隔のセンサ値の取得

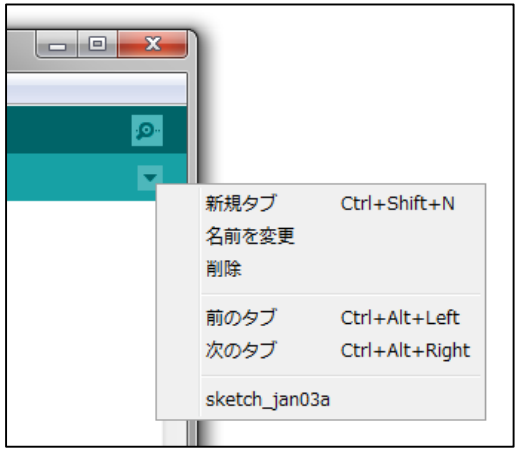
```
void setup()  {
    pinMode(13,OUTPUT);
}
void loop(){
  digitalWrite(13,HIGH);
  delay(1000);
  digitalWrite(13,LOW);
  delay(1000);
  }
```

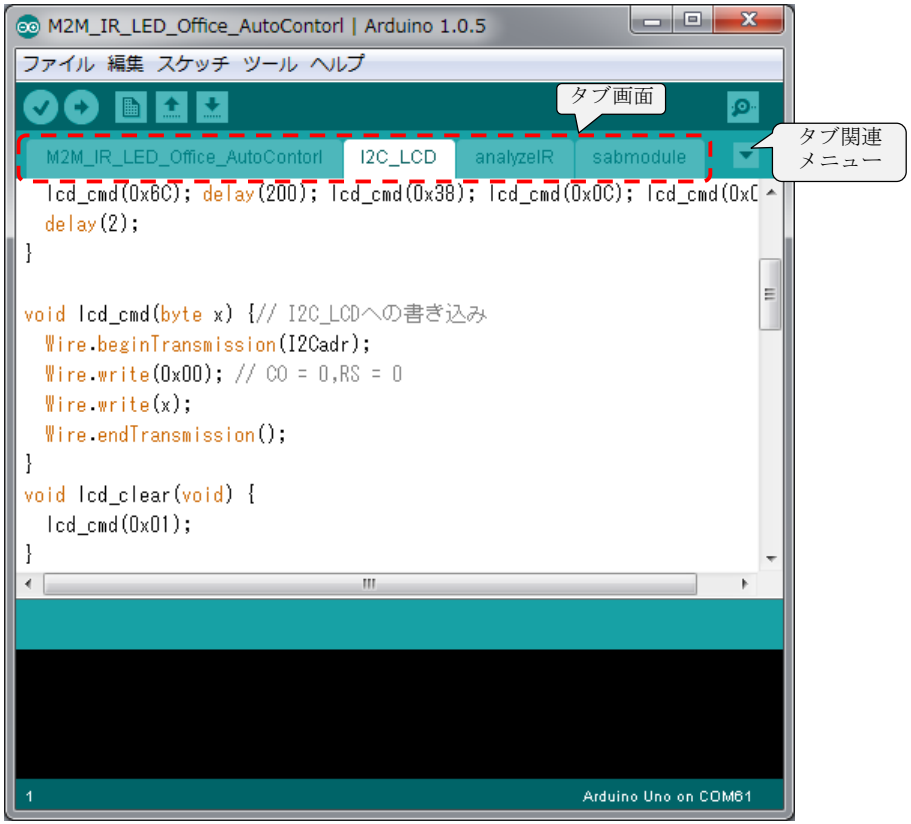
```
void setup() {
  pinMode(13,OUTPUT);
 }
 void loop() {
  static unsigned long tm=millis();     // 時刻初期化
  digitalWrite(13,HIGH);      // LED点灯
  while(tm+1000>millis());   //  1 秒以内
  digitalWrite(13,LOW);      //  LED消灯
  while(tm+2000>millis());  //  1秒以内
  tm=tm+2000;              // 時刻再設定
}
```

2．複数スケッチによるタブ画面

- タブ画面を使って、複数のスケッチを管理することができる。







3．不揮発メモリーEEPROMを使う①

- Arduino UNO R3には、１Kバイトの不揮発性メモリーEEPROMが備わっている。　電源を切っても、データをArduino上に保管し、つぎに電源を入れて再利用できる。
- 使い方として、呼出しヘッダーファイル<EEPROM.h>を読み込んでおく

```
#include <EEPROM.h>
```

```
void EEPROM.write (int adr, byte val)
   ここで、adr：アドレス（Arduino UNOの場合は、０から1023）
            val：書き込み値（バイト:0から255）
戻り値：なし
   byte EEPROM.read (int adr)
        ここで、adr：アドレス（Arduino UNOの場合は、０から1023）
          戻り値：指定したアドレスの読み込み値
```

EEPROMを使う時の注意点
１．100,000回まで
２．書き込み時間は、3.3ミリ秒掛る

- EEPROMを使った事例を紹介

リセットボタンを押したり、電源を切っても、メモリーの値がカウントアップされる

```
#include <EEPROM.h> // EEPROM.hの読み込み宣言
void setup(){
  Serial.begin(9600);
  byte val = EEPROM.read(0);// EEPROMからの読み込み
  Serial.print("Memory value: ");
  Serial.println(val);
  EEPROM.write(0,++val); // EEPROMへの書き込み
}
void loop(){}
```

Basic_EEPROM_sample00.ino

リセットボタンを、途中押して確認してみよう。

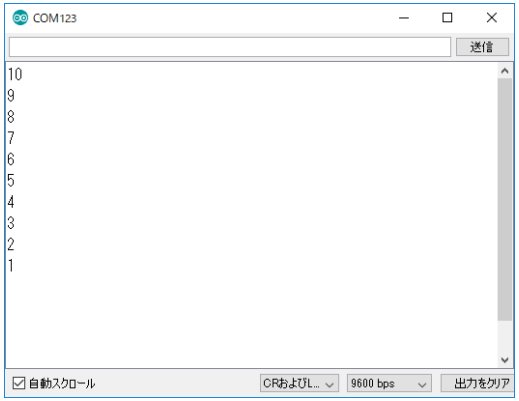
3．不揮発メモリーEEPROMを使う②

Arduino UNOでは、EEPROMとして１K(1024）バイトの不揮発メモリーが利用できます。
Arduino UNOの電源を切ったり、リセットしても消えることはありません。
大事なデータを保管・格納・利用するときに使うことができます。

■ **サンプルスケッチ**

```
#include <EEPROM.h>
void setup() {
  Serial.begin(9600);
  for (int i = 0; i < 10; i++)
    EEPROM.write(i, 10 - i);
  for (int i = 0; i < 10; i++)
    Serial.println(EEPROM.read(i));
}

void loop() { }
```

Basic_EEPROM_sample01.ino



① **必要ライブラリ**　EEPROM. h
② **初期化設定**　不要
③ **値の書き込み・読み込み**
　書き込み EEPROM.write
　読み込み EEPROM.read
などを利用

■ **EEPROM.h の基本関数群**

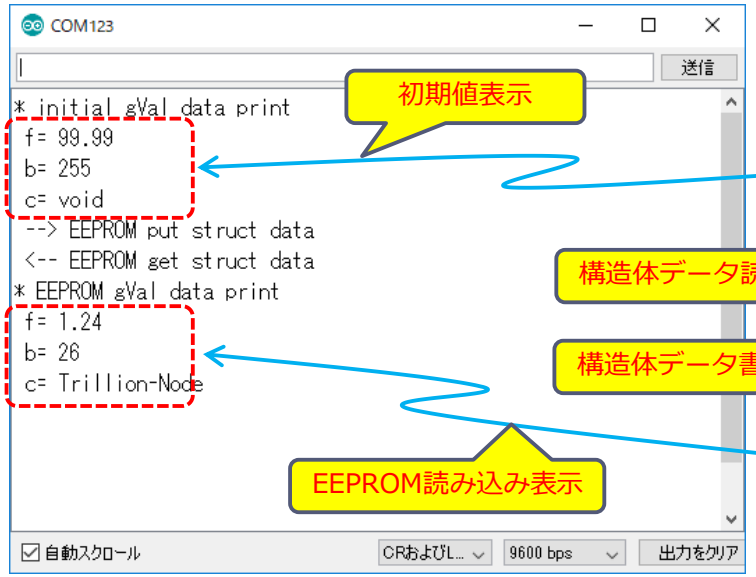
関数群（一部）	概要説明
Void EEPROM.write(ad,dat)	データ書き込み ad: アドレス（0～1023・2047） dat: データ（バイト）
byte EEPROM.read(ad)	データ読み込み ad: アドレス（0～1023・2027）
int EEPROM.length()	EEPROMの容量 （ボードの場合1024バイト）
EEPROM.put(ad,struct);	EEPROMに構造体で書き込み ad: アドレス struct:構造体データ
EEPROM.get(ad,struct);	EEPROMの保存された構造体を読み込み ad: アドレス struct:構造体データ

【補足】EEPROMの読み書きの処理速度は遅く（1バイト 3.3μ秒）、また読み書きの回数も10万回ほどと制限がありますので、ご注意ください。

3．不揮発メモリーEEPROMを使う③

構造体データを書き込み・読み込みするサンプルを紹介します。

① **必要ライブラリ**　EEPROM. h
② **初期化設定**　不要
③ **値書き込み・読み込み**
　書き込み EEPROM.put
　読み込み EEPROM.get



■ **サンプルスケッチ**

構造体定義

```
#include <EEPROM.h>
#include <EEPROM.h>
struct stdata {
  float f;
  byte  b;
  char  c[15];
};

void setup() {
  Serial.begin(9600);
  stdata pVal, gVal={99.99,255,"void"};
  pVal.f = 1.235;
  pVal.b = 26;
  sprintf(pVal.c,"%s","Trillion-Node");
  Serial.println("* initial gVal data print");
  Serial.println(" f= " + String(gVal.f));
  Serial.println(" b= " + String(gVal.b));
  Serial.println(" c= " + String(gVal.c));
  Serial.println(" --> EEPROM put struct data");
  EEPROM.put(100, pVal);

  Serial.println(" <-- EEPROM get struct data");
  EEPROM.get(100, gVal);
  Serial.println("* EEPROM gVal data print");
  Serial.println(" f= " + String(gVal.f));
  Serial.println(" b= " + String(gVal.b));
  Serial.println(" c= " + String(gVal.c));
}

void loop() { }
```

15文字の文字配列

変数設定
gValには初期値設定

Basic_EEPROM_sample02.ino

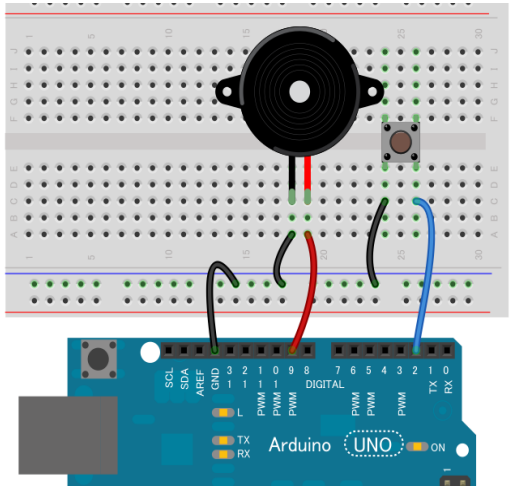
4．割込み機能を使う

- Arduino には、割込み機能があり、センサの値などによって、特別な処理を並行して行うことができる。
- Arduino UNOでは、デジタル入出力ピンの「D2」と「D3」の値の変化によって、指定した関数を呼び出す。

```
void attachInterrup(byte int, void (*fun)(void), int mode)
  ここで、 int : 割込み番号 （ 0 または 1）
              fun: 割込みする関数名（実際にはポインタ）
                   この関数には、引数も戻り値もなしとする。
             mode:割込み関数を実行する条件
「LOW」 ピンの値がLOWの場合
                「CHANGE」ピンの値が変わって場合
                「RISING」ピンの値がLOWからHIGHに変わった場合
                「FALLING」ピンの値がHIGHからLOWに変わった場合
                「HIGH」ピンの値がHIGHの場合
```

■ 事例：Arduino上のLEDを点滅させた状態で、割込み番号０の「D2」に取り付けたタクトスイッチの値が変化するたびに、ブザーを鳴らすサンプルスケッチ

・



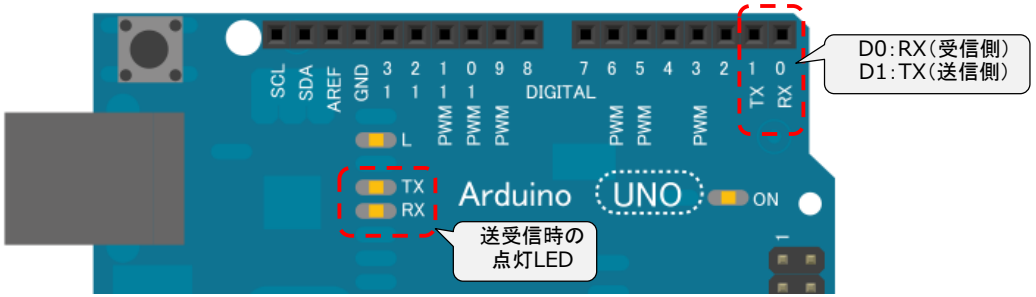
■ 割込みを使った事例紹介

```
boolean sw=false;
void setup(){
  pinMode(2,INPUT_PULLUP); // 割込みピン（タクトスイッチ）
  pinMode(9,OUTPUT);
  pinMode(13,OUTPUT); // Arduino 上のLED
  attachInterrupt(0,buzzer,CHANGE); //割込み処理関数
}
void loop() {
  digitalWrite(13,HIGH);
  delay(1000);
  digitalWrite(13,LOW);
  delay(1000);
  if( sw ) {
    sw=false;
  tone(9,255,1000);
  }
}
void buzzer(){
  sw=true;
}
```

Extend_Interrupt.ino

5．シリアル通信機能を使う①

- シリアル通信UARTを使って、２つのArduino間で、通信を行ってみる。
- もともとArduino UNOには、ハードウェアシリアルが、D0とD1に割り当てられている。
- 別途、ソフトウェアシリアル通信を使って、２つのArduino間での通信を行ってみる。



■ シリアル通信関係の関数（主なもの）

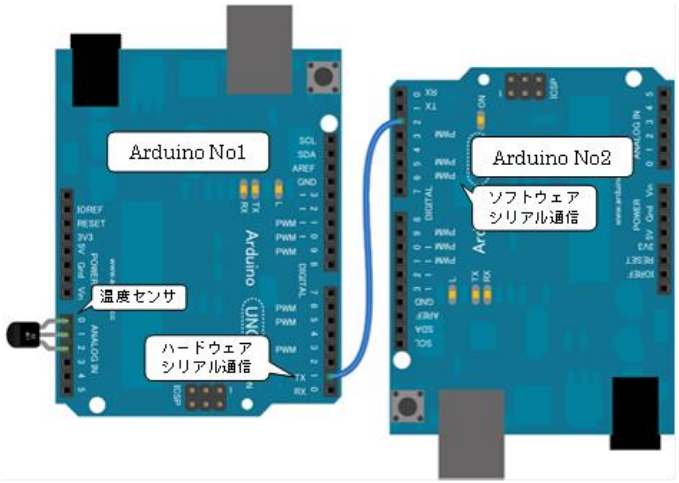
```
void Serial.begin(int speed) ： 通信速度を設定し、通信を有効にする
   ここで speed：通信速度（単位pbs：ビット/秒）で、300、1200～115200まで
void Serial.end()：通信を無効（中断）とし、「D0」「D1」がデジタル入出力ピンとして有効利用可能
int Serial.available() ：シリアルポートに到着しているバッファのバイト数を返す
   戻り値：シリアルバッファにあるデータのバイト数
int Serial.read()：受信データの読み込み （ポインタをずらす）
   戻り値：読み込み可能なデータの最初の1バイト。-1の場合はデータは存在しない。
```

■ ソフトウェアシリアル通信の割り当て

```
void SoftwareSerial（int rxPin, int txPin）：通信ポート（送信と受信）を設定
        ここで rxPin：データを受信するピン
               txPin：データを送信するピン
```

5．シリアル通信機能を使う②

- 事例：２つのArduinoでシリアル通信を実現（Arduino No1にある温度センサ値を、Arduino No2に送って、値をPC上で確認）



■ Arduino No1 のスケッチ

Extend_Serial_No1.ino

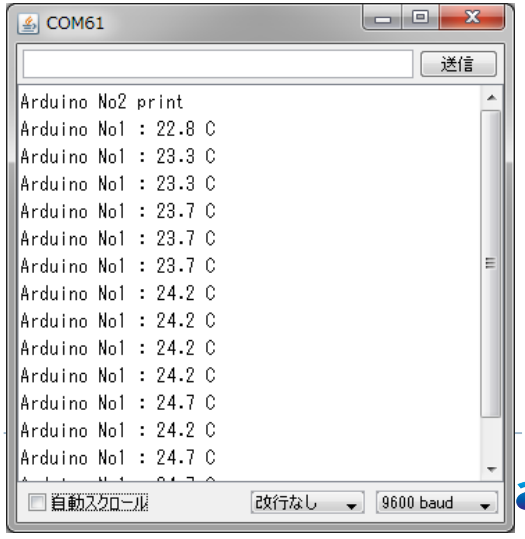
```
void setup(){
  Serial.begin(9600);    //Arduino No2との通信速度設定
  pinMode(A0, OUTPUT); //A0に温度センサ「GND」ピン設定
  digitalWrite(A0,LOW);
  pinMode(A2, OUTPUT); //A2に温度センサ「５V」ピン設定
  digitalWrite(A2,HIGH);
}
void loop() {
  float cel = ((float)analogRead(A1)/1023.0)*487.0-60.0;  //A1から温度センサ値取得
  char sc[25];
  sprintf(sc, "Arduino No1 : %d.%d C", (int)cel, (int)(cel*10)%10);
  Serial.println(sc);   // 温度センサ値を含む文字列をシリアル通信で送信
  delay(500);
}
```

■ Arduino No２ のスケッチ

Extend_Serial_No2.ino

```
#include <SoftwareSerial.h>       //ソフトウェアシリアル通信ライブラリの設定
SoftwareSerial No2Arduino (2, 3);  // 受信側RX：D2，送信側TX：D3に設定

void setup(){
  No2Arduino.begin(9600);        // ArduinoNo1との通信速度設定
Serial.begin(9600);              // シリアルモニタ画面への表示通信速度設定
Serial.println("Arduino No2 print"); // ArduinoNo2からの送信の表記文字
}
void loop(){
  if(No2Arduino.available())
    Serial.write(No2Arduino.read()); // ArduinoNo2で受信した文字をシリアルモニタ画面表示
}
```

■ 出力結果（例）

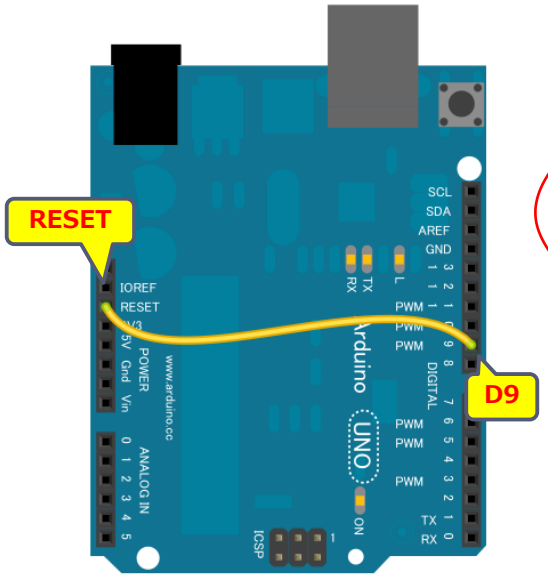


6．ソフトウェアリセットの実現方法

ソフトウェア・リセットは「Reset」I/OポートをLOWにするだけで実現可能
⇒ 実際には、以下のような配線とプログラムで実現可能

ソフトウェアリセットで再立上げ
（再度 setup関数とvoid loop関数を起動）

■Arduinoの配線



RESETピンとデジタルピン（D9)を接続

■サンプルスケッチ

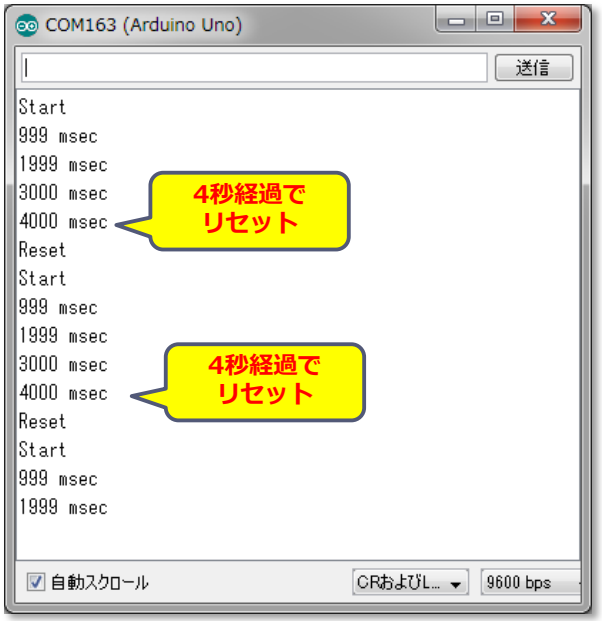
Extend_Softwere_Reset.ino

```
void setup() {
  Serial.begin(9600);
  Serial.println("Start");
}

void loop() {
  delay(1000);
  if( millis()>4000 ) {
     Serial.println("Reset");
     delay(100);
     pinMode(9,OUTPUT);
  }
  Serial.println(String(millis()) + " msec ");
}
```

pinMode設定でリセット

時間がある値（4秒）以上になると、
ソフトウェア・リセットする

■実行例（シリアルモニタ画面）



RESETピンと接続したピンのpinMode設定でリセット

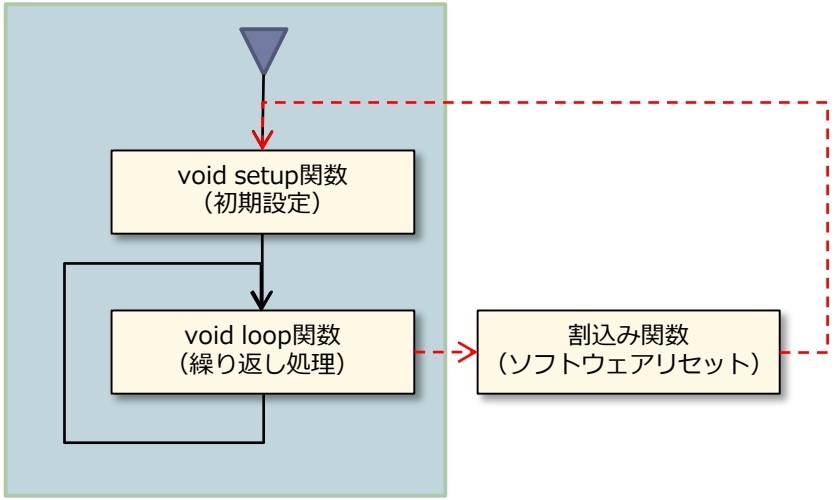
7．ソフトウェアリセットと割込み処理の使い方①

外部の変化に応じて、特殊処理をする場合には、**if**制御文などを使ったりしますが、もう一方では割込みを使う方法があります。

ある外部の変化が起きた時に、割込み処理を起動させ、そこでソフトウェアリセットを行いプログラムを再起動させてみるスケッチを実現してみましょう。

■課題
本サンプルスケッチでは、LED（D13）を1秒間隔で点滅させる一方で、10秒間隔でソフトウェアリセットを割込み処理によって行っています。

またこの10秒間の間に、Arduino上のLED（D13）を1秒間隔で点滅させています。

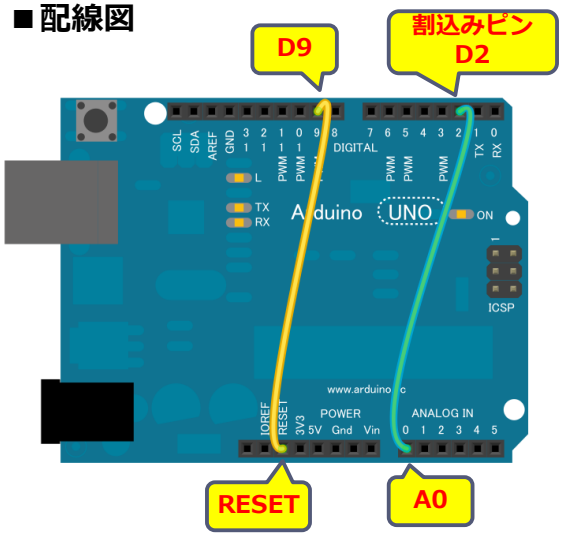


▼２つの技術ポイント
ここでは、２つの技術ポイントが必要となります。
１）割込み処理を起こすタイミング
２）割込み関数によるソフトウェアリセットの実現
これらを組み合わせて、何らかの外部の変化に応じて、ソフトウェアリセットを実現することが可能となります。

この場合、**Arduinoの割込みピン**を知っておく必要があります。Arduino UNO の場合は、D2(INT０）と、D3（INT1)となっています。 Arduino Megaの場合は、上記に加えD21（INT2)、D20（INT3)、D19（INT４）となっています。

7．ソフトウェアリセットと割込み処理の使い方②

■配線図



■スケッチ　Extend_SoftReset_Interrupt.ino

```
void setup() {
 Serial.begin(9600);
 Serial.println("start");
 delay(100);
 pinMode(A0,OUTPUT);
 delay(100);
 pinMode(2,INPUT_PULLUP);

 pinMode(13,OUTPUT);
 attachInterrupt(0,check,HIGH);
}

void loop() {
  static long tim = millis();
  digitalWrite(13,HIGH);
  delay(1000);
  digitalWrite(13,LOW);
  delay(1000);
  if(millis()-tim>10000) digitalWrite(A0,HIGH);
}

void check() {
  pinMode(9,OUTPUT);
}
```

（注意）pinModeは、A0を先にしてD2を後に設定

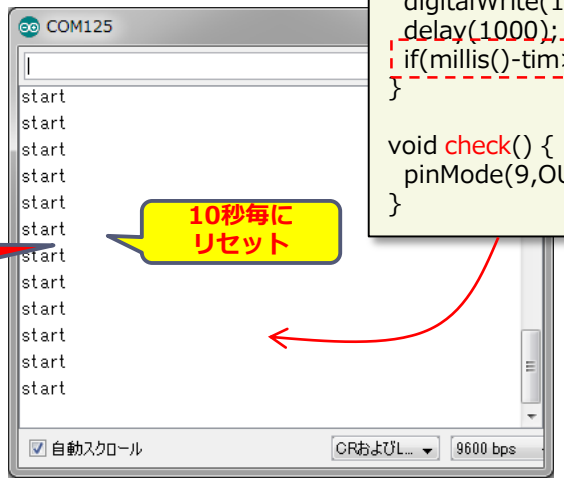
Tabraino LED 3

割込み関数定義

時間で割込み（10秒後）

pinMode設定でリセット

■シリアルモニタ画面



10秒毎にリセット

ある間隔ごとに確認して、リセットを掛けることが可能に

（応用１）ある時間以上経過しても通信が行われなかったときリセットする場合

（応用２）致命的なエラーになった時、再起動（リセット）させる場合

（応用３）一連の流れの処理を終え、再起動させたい場合

8．Arduino 電子部品利用早見表①

- さまざまな電子部品をArduino上で使う時の早見表となる。（「みんなのArduino入門」からの抜粋）

電子部品	利用I/O※３	スケッチ利用まとめ（変換式含む）	結果・変換式＆備考
可変抵抗器（4.2節）	入力A0~A5	float val=analogRead(Ax)*/1023.0 * R	R（抵抗値）
タクトスイッチ（4.3節）	入力D0~D19	pinMode(Dx,INPUT_PULLUP); boolean sw = digitalRead(Dx);	スイッチOn： LOW スイッチOff：HIGH
チルト（傾斜）センサー（4.3節）	入力D0~D19	同上	スイッチOn： LOW スイッチOff：HIGH

LED（5.2節、5.3節）	出力D0~D19	pinMode(Dx,OUTPUT);
		digitalWrite(Dx,hl);
		delay(sc);//必要な場合挿入
			hl:HIGH（＝５V）または
			LOW（=0V）
			sc:待機時間（ミリ秒）

	出力※１ PWM	analogWrite(Pwm,Px);	Px：0(0V)～255(5V)
圧電スピーカ（SPT08）（5.2節、5.4節）	出力D0~D19	pinMode(Dx,OUTPUT); tone(Dx,hz,sc);	hz:周波数（Hz） sc:時間（ミリ秒）
小型DCファン（モータ）（NidecD02X-05TS1）（5.5節）	出力※１ PWM	analogWrite(Pwm,Px);	Px：0(0V)～255(5V)
アナログ温度センサー(LM61BIZ)（6.1節）	入力A0~A5	int val=analogRead(Ax); float cel=(float)val*0.488-60.0	val：0(0V)～1023(５V)
アナログ光センサー(CdS)（6.2節）	入力A0~A5	int val=analogRead(Ax);	val：0(0V)～1023(５V)

※1．アナログ出力ポートのPWM（デジタル入出力ポート：Pwm）はD３、5、6、9、10、11の何れか。
※2．超音波距離センサーは、超音波の入出力によって、距離を算出。
※3．デジタル入出力ポートのD14からD19は、アナログ入力ポートのA0からA5と同じ。

8．Arduino 電子部品利用早見表②

電子部品	利用I/O※３	スケッチ利用まとめ（変換式含む）	結果・変換式＆備考

3軸加速度センサー
(KXR94-2050)（6.3節）
	入力
	A0~A5
		float Xa=digitalRead(Ax)*5.0/1023.0-2.5;
		float Ya=digitalRead(Ay)*5.0/1023.0-2.5;
		float Za=digitalRead(Az)*5.0/1023.0-2.5;
			Ax,Ay,Azは、A0～A5
			重力加速度も含まれる

超音波距離センサー
(HC-SR04/SEN136B5B)
（6.4節）
	入力※2
	D0~D19
		pinMode(TrigPin,OUTPUT);
		pinMode(EchoPin,INPUT);
		digitalWrite(TrigPin,HIGH);
		delayMicroseconds(CTM);
		digitalWrite(TrigPin,LOW);
		int dur = pulseIn(EchoPin,HIGH);
		float dis=(float)dur*0.017;
			TrigPin:トリガーピン
			EchoPin:エコーピン
			CTM:待機時間（マイクロ秒）
			測定距離は、数センチから４m程度

赤外線距離センサー
(GP2Y0A21YK)（6.5節）
	入力
	A0~A5
		float Vcc=5.0;
		float val=Vcc*analogRead(Ax)/1023;
		float dis= 26.549*pow(val,-1.2091)
			測定距離は、数センチ～80㎝程度

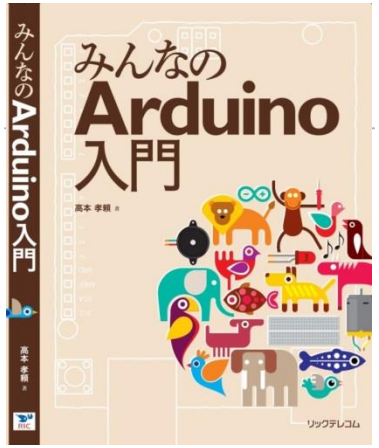
液晶ディスプレィ （SSCI-014076など）（6.6節）	A4/A5 (I2C)	#include<Wire.h> (I2C_LCD.inoのライブラリ群利用)	５V系と3.3V系に注意
EEPROM（7.3節）	-	#include <EEPROM.h> EEPROM.write(ad,val);　　//書き込み byte val=EEPROM.read(ad); //読み込み	ad: アドレス：0～1023 val：値（バイト）

シリアルモニタ画面
（7.5節）
	D0(RX)
	D1(TX)
		Serial.begin(spd); //通信速度設定
		Serial.print(str); // 改行なし
		Serial.println(str); // 改行あり
			spd: 通信速度9600等
			str：出力文字列

※1．アナログ出力ポートのPWM（デジタル入出力ポート：Pwm）はD３、5、6、9、10、11の何れか。
※2．超音波距離センサーは、超音波の入出力によって、距離を算出。
※3．デジタル入出力ポートのD14からD19は、アナログ入力ポートのA0からA5と同じ。

参考書

この「みんなのArduino入門」には、基本的な電子部品の使い方をまとめています。
ご参考にして頂けますと幸いです。



2014年2月17日発刊
（アマゾンよりお買い求めできます）

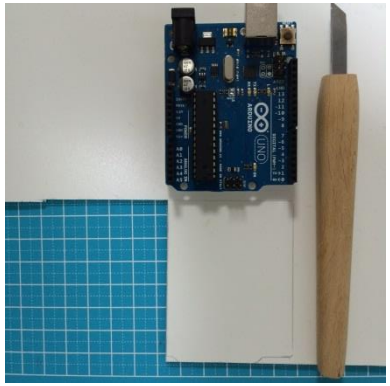
第 I 部　準備編
第1章　Arduinoってどんなもの？
1.1　Arduinoの誕生と背景
1.2　Arduinoとは
1.3　Arduinoの特長
1.4　Arduinoの機能
1.5　Arduinoの準備
1.6　統合開発環境（IDE）の準備
1.7　Arduinoを効率よく学ぶ
第2章　Arduinoを動かしてみよう
2.1　PCとArduinoとのUSBケーブル接続確認と注意事項
2.2　サンプル・スケッチを動かしてみよう
2.3　PCとArduino間のシリアル通信（シリアルモニタ表示）
2.4　ブレッドボードとジャンパワイヤを使ってみよう
2.5　アナログ・デジタル入出力とシリアル通信を知る
第3章　プログラミングの基本を知ろう
3.1　はじめに知っておくべきこと
3.2　C言語の基本的な決まりごとを知ろう
3.3　変数を使ってみよう
3.4　制御文を知ろう
3.5　関数を使ってみよう
3.6　よく使うものを知っておこう

第 II 部　基礎編
第4章　入力部品を使いこなそう
4.1　アナログとデジタルの入力系を知る
4.2　アナログ入力（可変抵抗器と電圧測定）を知る
4.3　デジタル入力（タクトスイッチとチルトセンサー）を知る
第5章　出力部品を使いこなそう
5.1　デジタルとアナログの出力系を知る
5.2　PWMによるアナログ出力（LEDと圧電スピーカの制御）を知る
5.3　デジタル出力によるLEDの制御
5.4　デジタル出力による圧電スピーカの制御
5.5　モータ（ファン）をアナログ出力で動かす

第 III 部　ステップアップ編
第6章　高度な入力出力部品を使ってみよう
6.1　温度センサー（アナログ）を使ってみよう
6.2　光センサー（アナログ）を使ってみよう
6.3　加速度センサー（アナログ）を使ってみよう
6.4　超音波距離センサー（デジタル）を使ってみよう
6.5　赤外線距離センサー（アナログ）を使ってみよう
6.6　液晶ディスプレイ（LCD）を使ってみよう
第7章　ちょっとしたティップス
7.1　タイマー機能を使う
7.2　複数スケッチによるタブ画面を使う
7.3　不揮発性メモリーEEPROMを使う
7.4　割込み機能を使う
7.5　シリアル通信機能を使う
7.6　知ってて得するArduino情報
付録
付録1　この本で扱った電子部品（教材キット）
付録2　この本で扱った電子部品のスケッチ利用まとめ（早見表）
付録3　IoTABシールドの紹介
付録4　Arduino関連情報サイト

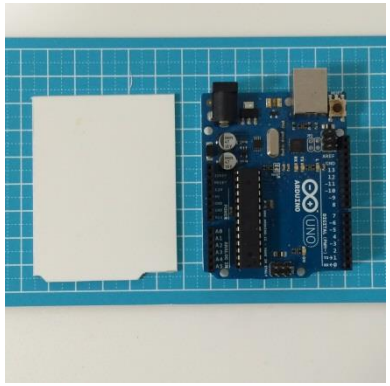
工作：静電気カバーシート（下駄）

- IoTABシールドを利用する場合、Arduinoごと手に取って操作する場合などがあります。その時静電気により誤動作する恐れもあり、その対策が必要となります。
- ここでは、その対策の一つとして、静電気カバーシート（下駄）の工作をご紹介します。
- 用意するもの：　材料：厚手のプラスチック板（２㎜ほど）、厚手の両面テープ
  工具：カッター、金定規など　　（材料費は、１枚当たり５０円以下に抑えられます）



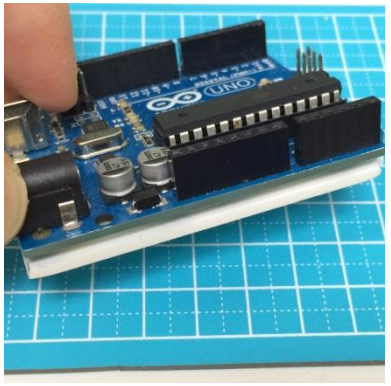
厚さ2㎜ほどのプラスチック製厚板とカッターを用意

厚板の上にArduinoを載せ、鉛筆で型を写し取ります。
その後、金定規とカッターを使って、型（下駄）を切り取ります。



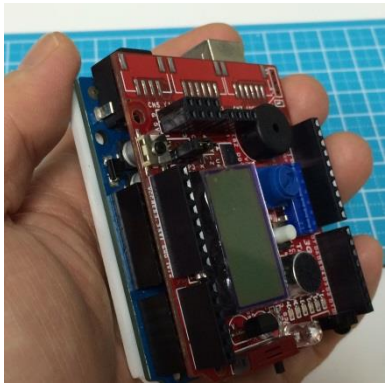
切り取ったら、さらに図のように凹凸のある部分も揃えて切り取ります。

その後、厚さのある両面テープを用意し、Arduinoの裏に貼っていきます。



両面テープを貼る場合、USBコネクタや電源コネクタに近い部分は、ピンが数ミリ裏に出ているので、１枚の両面テープを貼ります。

またArduino形状の凹凸のある反対側には、２枚ほど両面テープを重ねて貼ります。



このように、手で持って操作してもArduinoの裏面にあるピンに触れることなく、操作できるようになり、静電気も気にする必要はありません。

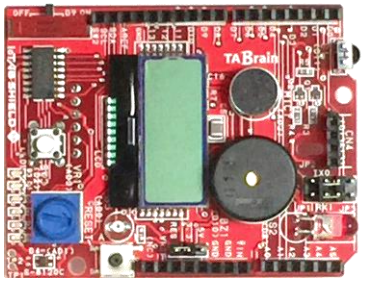
IoT教材キットについて

ワイヤレス（無線）を使った
センサネットワーク技術の応用

誰もが、短時間で、簡単に、IoTデバイス構築からインターネット接続が学べる教材キット
（既に2014年から多くの実績を持つM2M/IoT教材キットです）

IoT教育キット

センサ＋LCD/LED
IoTABシールドV3.0



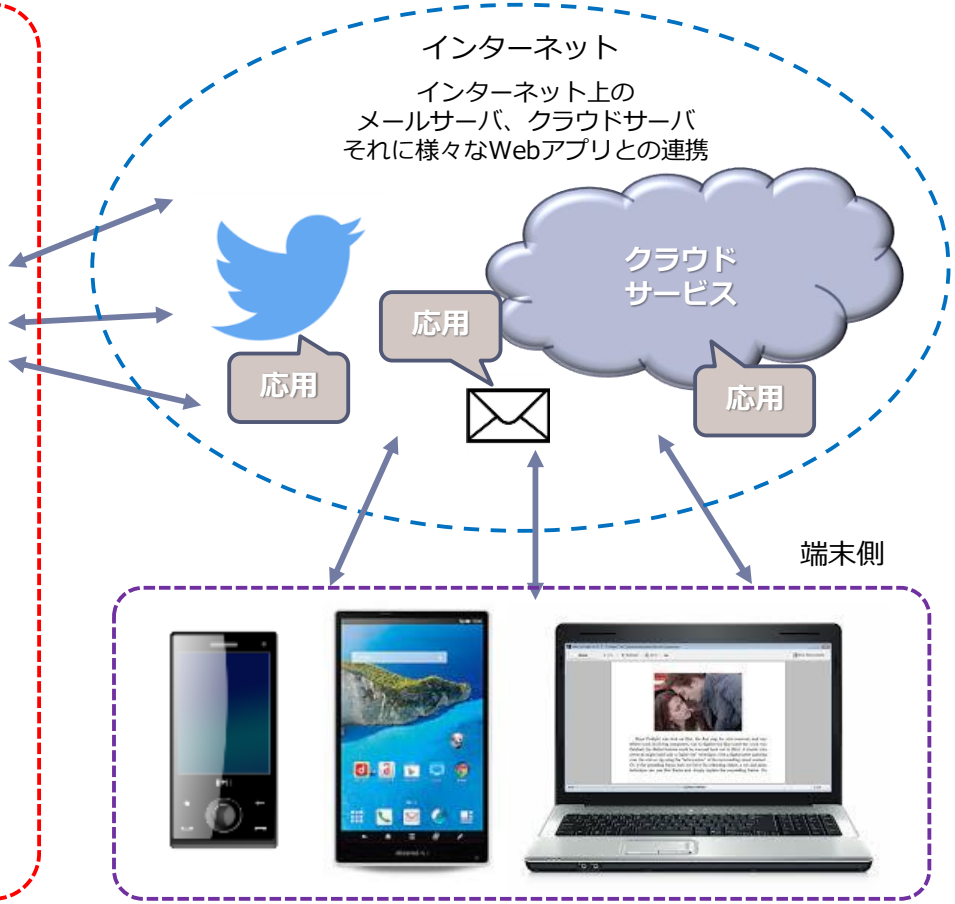
＋

3G通信機器
3GIM

＋

Genuino101
・インテルCurie
・慣性センサ搭載
・BLE搭載
・EEPROM

※その他、さまざまな
ボードに対応致します



IoT教材キット専用のマニュアル（豊富なサンプル付）も付属しています。

添付
IoTABシールド検査テスト

1．出荷検査テストについて

- 本IoTABシールドは、出荷時にすべての電子部品が正常に機能しているかを確認した上で出荷しています。
- 出荷検査テストは、IoTAB4_ALL_TEST.zipによるもので、ご購入様の方でも簡単にテストを行うことができます。

- テストスケッチは、http://tabrain.jp/tabs/IoTABS4_ALL_TEST.zipよりダウンロードしてご利用ください。
- このテストは、Arduino UNO上でIoTABシールドV４.0を搭載したうえで検査します。
- スケッチを書込み終了した時点で、プログラムは以下の動きをします。

１）圧電スピーカから音が1秒間鳴ります
２）TEST1: LCD画面上に３つの数字が表示されます
　　上段左は温度センサ値（C）、 上段右は照度センサ値
　　下段には音センサ値が表示（同時に音の大きさでLEDが点灯します）されます。
　　おのおのセンサ値の値が変わるように操作してみてください。
　　→温度センサに指を載せる。照度センサを手で覆って暗くしてみる。声を出してみる。
３）タクトスイッチを押す
４）TEST2: LCD画面上に以下の値が表示されます
　　上段左には、距離センサの値（CM）<超音波距離センサを挿入しておいてください>
　　上段右、下段左右には、加速度センサ値が、-100~100範囲で表示（１Gを100で換算）
５）タクトスイッチを押す
６）TEST３：赤外線リモコンと赤外線LEDのテスト
　　まず照明LEDやテレビなどのリモコンのボタンを2種類選択して送信してください。
　　あとは、自動的に入力した赤外線リモコンの値が赤外線LEDから1秒間隔で出てきます。
　　（赤外線LEDやテレビに向けて操作してください）
　　※ 本赤外線リモコンのスケッチは、エアコンなどの操作はできません。

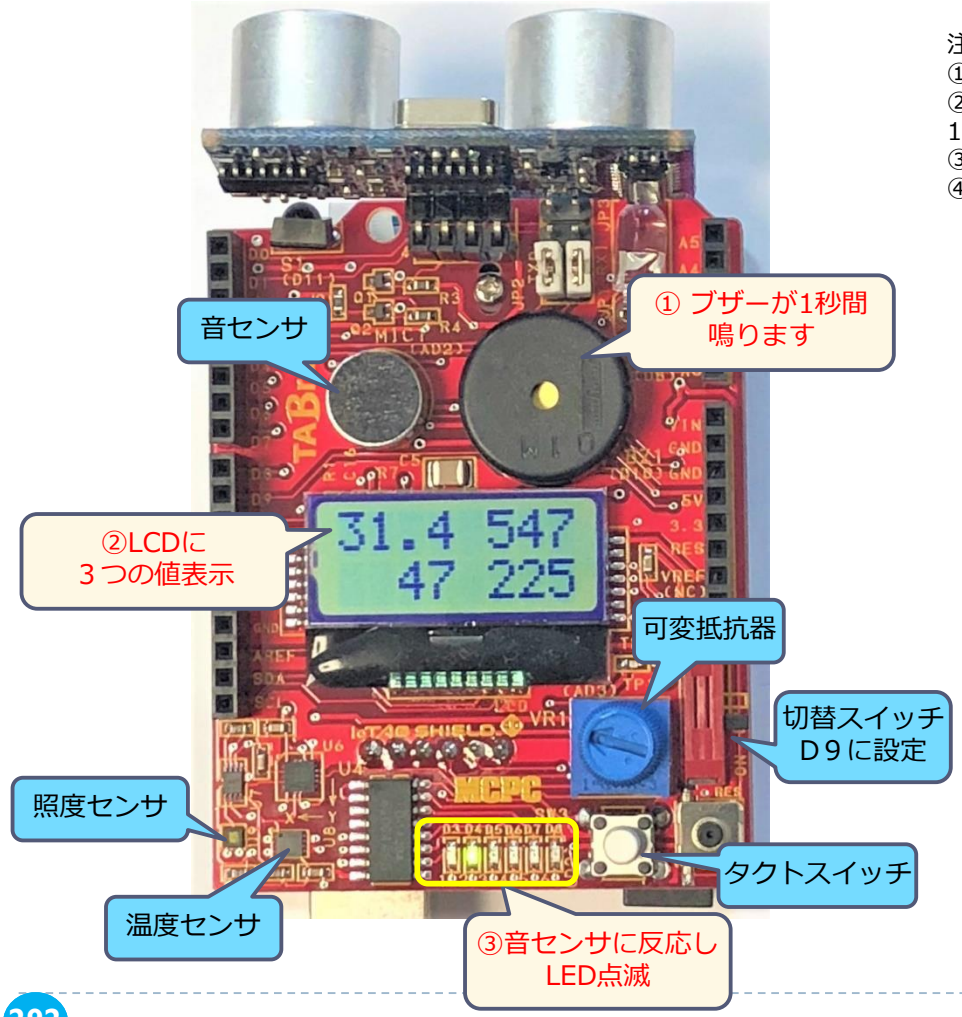
本出荷検査テストで以下の電子部品をテストしています。

１．照度センサ（I2C）
２．温度センサ（I2C）
３．音センサ
４．可変抵抗器
５．超音波距離センサ
６．加速度センサ（I2C）
７．タクトスイッチ
８．６個のLED
９．スピーカ
１０．LCD
１１．赤外線受信リモコン
１２．赤外線LED

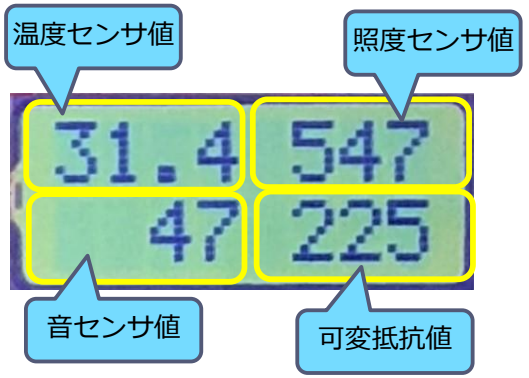
※出荷時すべて機能する製品を合格品としています。

2．出荷検査テストの操作画面①TEST1

TEST1：スピーカ、温度センサ、照度センサ、音センサ、LCD、LED、タクトスイッチ確認
※切替スイッチは、D9側に設定しておいてください。



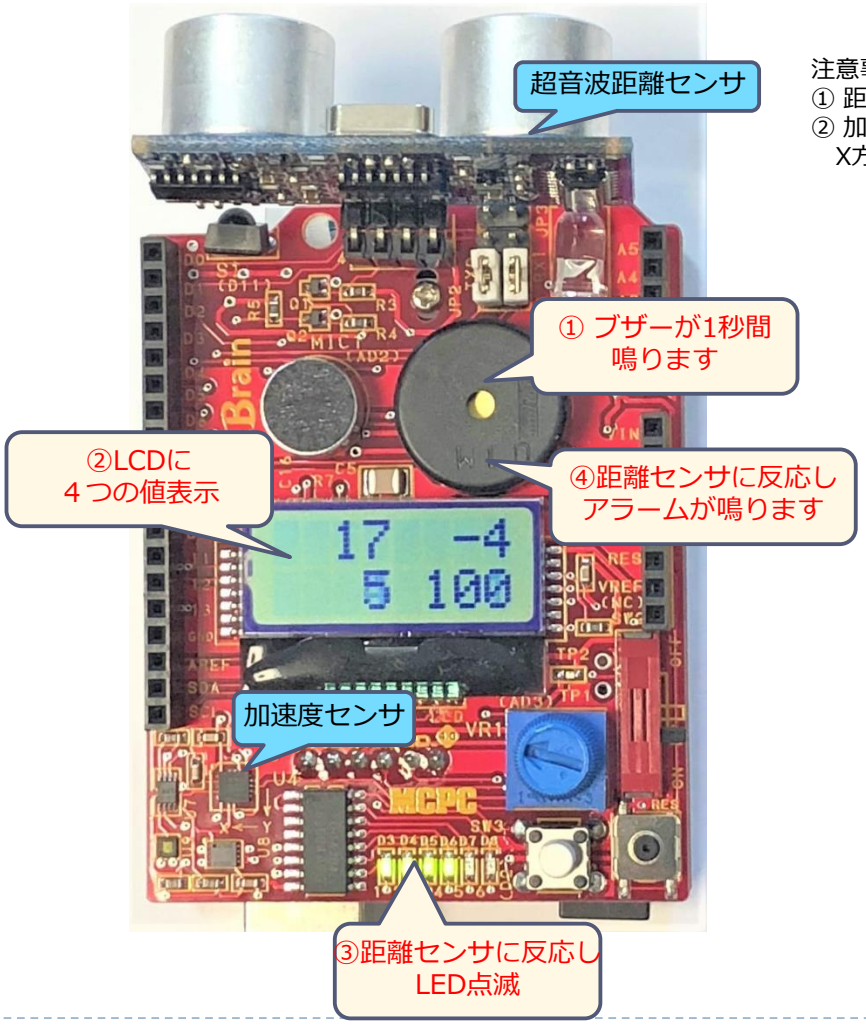
注意事項：
① 温度センサ値は、4桁で表示しています。
② 照度センサ値は、9999Lxまで表示しますが、太陽光などの直射日光は10000Lxを超えますので、正しく表示されない場合があります。
③ 音センサ値は、アナログ値の512を中心とした絶対値の平均を表示しています。
④ 可変抵抗値は、可変抵抗器の右周り端０から左周り端1023まで表示します。



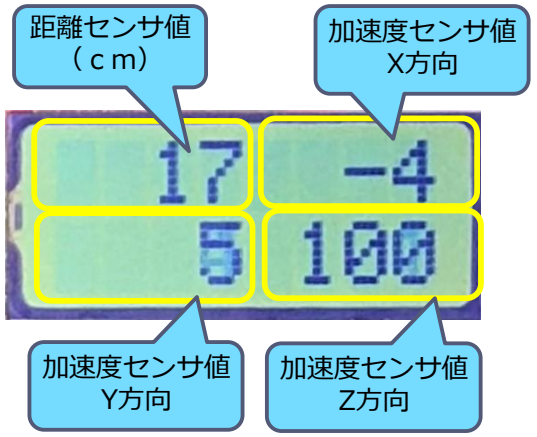
値が確認できたら、タクトスイッチを押して、次のTEST２に移ってください

2．出荷検査テストの操作画面②TEST２

TEST２：スピーカ、超音波距離センサ、加速度センサ、LCD、LED、タクトスイッチ確認



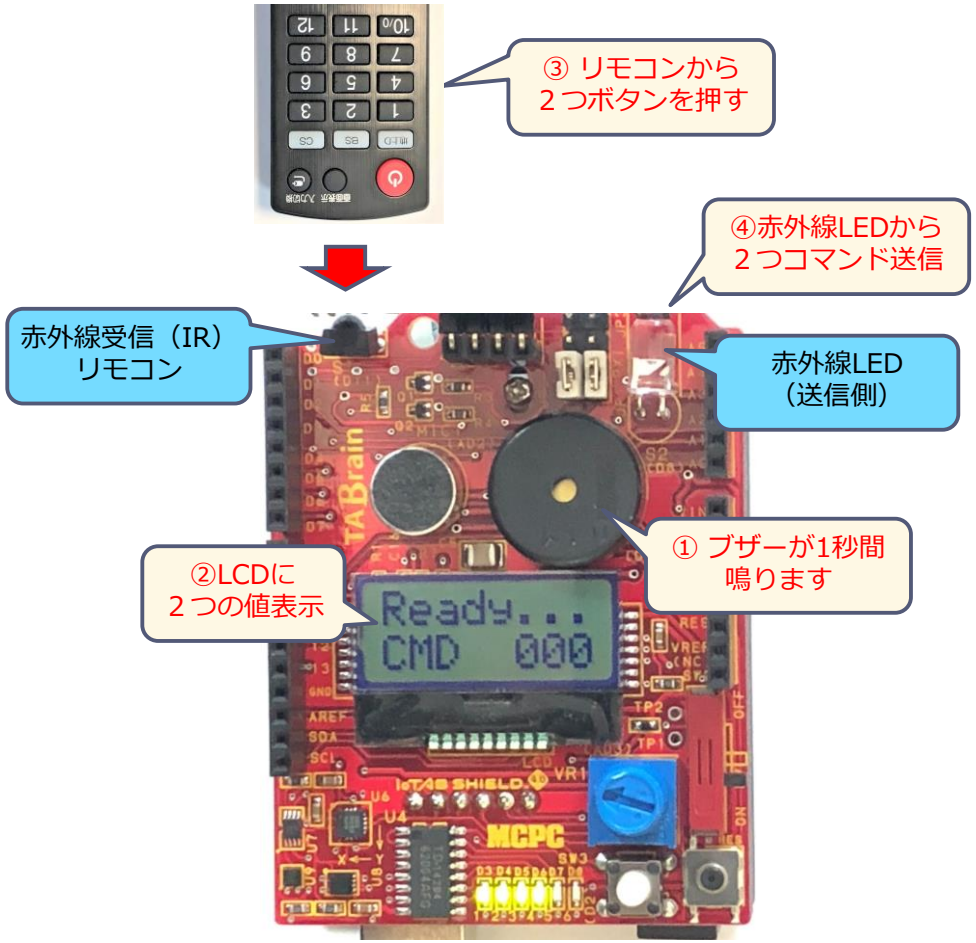
注意事項：
① 距離センサ値は、３桁で表示しています。単位はcmです。
② 加速度センサ値は、１Gを１００として表示しています。
　X方向（左上が正）、Y方向（下部が正）、Z方向（表上が正）となっています。



値が確認できたら、タクトスイッチを押して、次のTEST３に移ってください

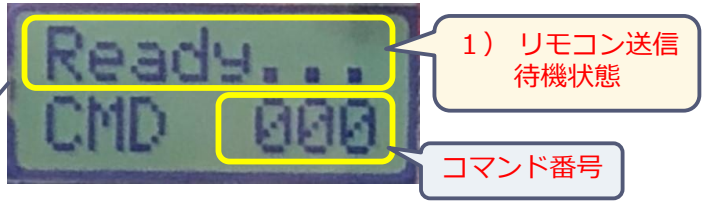
2．出荷検査テストの操作画面②TEST３

TEST３：赤外線受信（IR）リモコン、赤外線LED、LCDの確認テスト

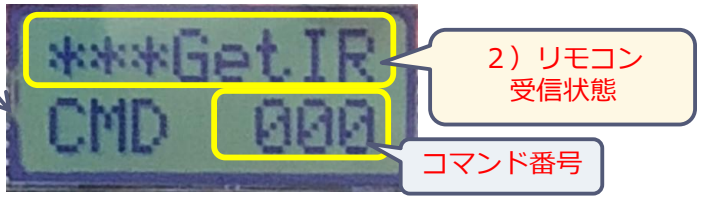


IR受信待機時

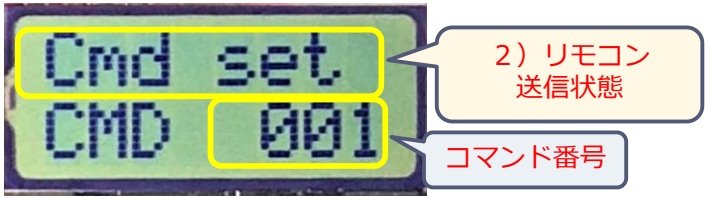
１）IR受信待機時（２つのコマンドを受信）



２）IR受信時

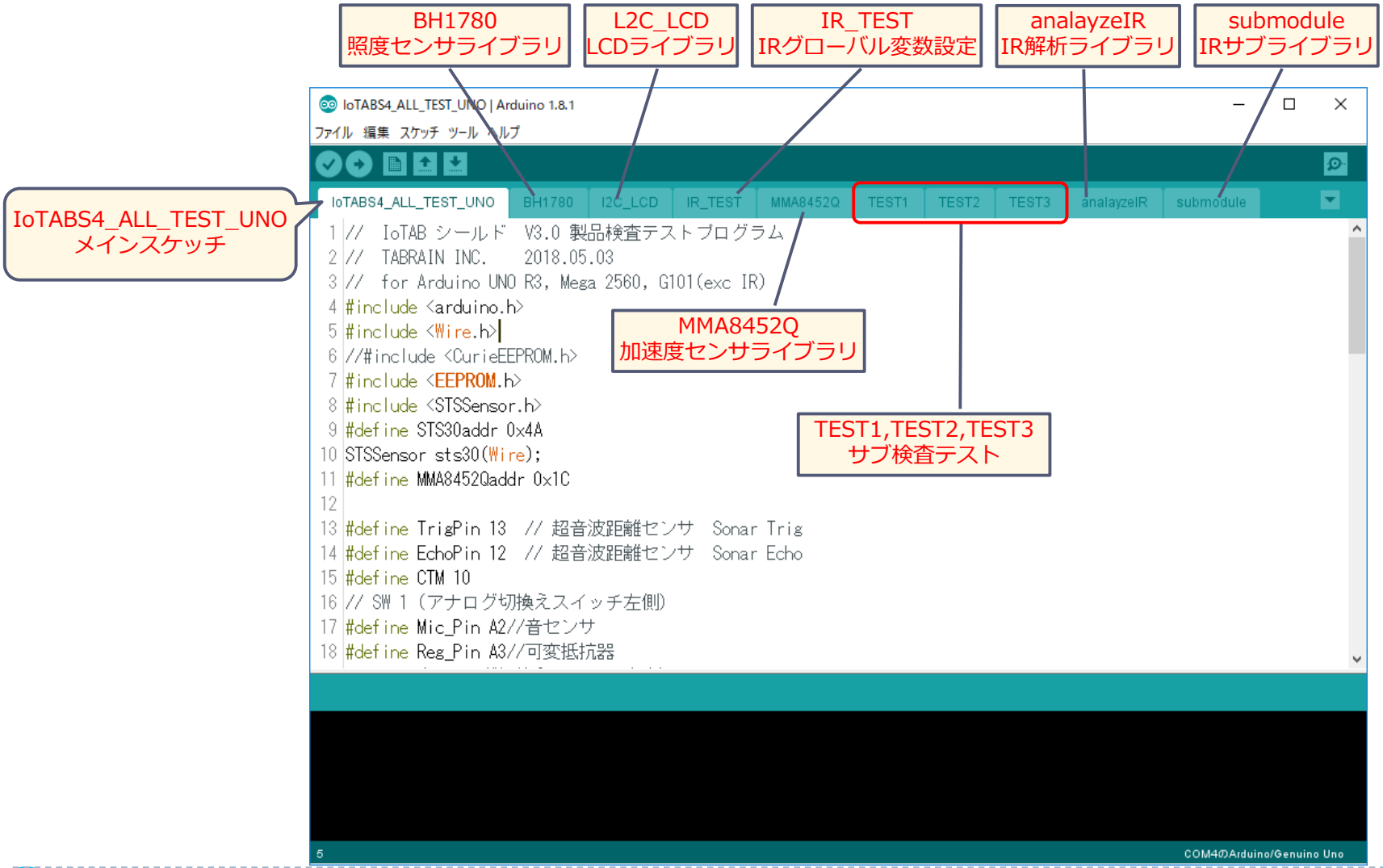


３）IR送信時（繰り替えし操作）



※最後に終了するには、タクトスイッチを押します。

3．検査テストスケッチ ①



3．検査テストスケッチ ②

メインスケッチ： 初期宣言、初期設定などを行い、TEST1、TEST2およびTEST3を実行します。

```
//   IoTAB シールド   V4.0 製品検査テストプログラム
//  TABRAIN INC.    2018.05.03
//  for Arduino UNO R3, Mega 2560, G101(exc IR)
#include <arduino.h>
#include <Wire.h>
//#include <CurieEEPROM.h>
#include <EEPROM.h>
#include <STSSensor.h>
#define STS30addr 0x4A
STSSensor sts30(Wire);
#define MMA8452Qaddr 0x1C

#define TrigPin 13  // 超音波距離センサ   Sonar Trig
#define EchoPin 12  // 超音波距離センサ   Sonar Echo
#define CTM 10
// SW 1 (アナログ切換えスイッチ左側)
#define Mic_Pin A2//音センサ
#define Reg_Pin A3//可変抵抗器
// SW 2（アナログ切換えスイッチ中央）

#define InfRim_Pin 11
#define Spk_Pin 10  //***** V4.0***
#define But_Pin 2     //***** V4.0***


#define Led2Pin 3
#define Led3Pin 4
#define Led4Pin 5
#define Led5Pin 6
#define Led6Pin 7

char pr[9];
byte MODE;
void lcd_init();
void test1();
void test2();
void test3();
```

初期宣言（任意）

```
void setup() {
  Serial.begin(9600);
   Wire.begin();
   pinMode(TrigPin,OUTPUT);
   pinMode(EchoPin, INPUT);
   pinMode(Spk_Pin,OUTPUT);
   pinMode(Led2Pin,OUTPUT);
   pinMode(Led3Pin,OUTPUT);
   pinMode(Led4Pin,OUTPUT);
   pinMode(Led5Pin,OUTPUT);
   pinMode(Led6Pin,OUTPUT);
   pinMode(But_Pin,INPUT_PULLUP);

   sts30.init(Wire);
   BH1780_powerup();
   MMA8452Q_init();
   lcd_init();
   test1();//   温度センサ、光センサ、音センサ、可変抵抗値表示
   test2();//   超音波距離センサ、３軸加速度センサ
   pinMode(But_Pin,INPUT_PULLUP);
   test3();//   赤外線受信リモコン、赤外線送信LED、EEPROM
}

void loop() {}
```

初期宣言

I2C部品関連初期設定

TEST1,TEST2,TEST3実行

3．検査テストスケッチ ②

TEST1のスケッチ
① ブザーを鳴らし、② 温度センサ値を算出表示、③ 照度センサ値を算出表示、④音センサ値の算出表示、⑤ 音センサ値によるLED点灯、⑤ 可変抵抗器の読み取り表示 を処理しています。

ブザーを
1秒間鳴らす

音の初期
平均音算出

温度センサ値
算出表示

照度センサ値
算出表示

音の平均音算出

```
void test1(){
 int vol=0;
 tone(Spk_Pin,255,1000);
 for(int i=0; i<30; i++) {
     int v = analogRead(Mic_Pin);
     vol += (v>0?v:-v); delay(10);}
 vol /= 30;
 do{
  // 温度センサ（LCD上段左側に表示）
   sts30.readSample();
   float temp = sts30.getTemperature();
   sprintf(pr,"%2d.%1d",int(temp),int(temp*10)%10);
   lcd_setCursor(0,0);
   lcd_printStr(pr);
   //照度センサ（LCD上段右側に表示）
   uint16_t x = BH1780_readlux();
   lcd_setCursor(4,0);
   sprintf(pr,"%4d ",x);
   lcd_printStr(pr);
   //音センサ（LCD下段左側に表示）
  int mic = 0;
  for(int i=0; i<10; i++) {
     int v = analogRead(Mic_Pin)-vol;
     mic += (v>0)?v:-v;}
   mic /= 10;
   lcd_setCursor(0,1);
   sprintf(pr,"%4d ",mic);
   lcd_printStr(pr);
```

音の大小で
LED点灯

可変抵抗器の
読み取り表示

```
  digitalWrite(Led2Pin,(mic>50)?HIGH:LOW);
     digitalWrite(Led3Pin,(mic>100)?HIGH:LOW);
     digitalWrite(Led4Pin,(mic>150)?HIGH:LOW);
     digitalWrite(Led5Pin,(mic>200)?HIGH:LOW);
    // チルト（傾斜）センサ（LEDの6番ピンで表示）
    // digitalWrite(Led6Pin,(digitalRead(Tlt_Pin)==LOW?HIGH:LOW));
     //可変抵抗器（LCD下段右側に表示）
     int vi2=analogRead(Reg_Pin);
     lcd_setCursor(4,1);
     sprintf(pr,"%4d ",vi2);
     lcd_printStr(pr);
     delay(100);
  } while(digitalRead(But_Pin));
 delay(500);
}
```

3．検査テストスケッチ ③

TEST 2 のスケッチ
① ブザーを鳴らし、② 超音波距離センサ値の取得表示、③ 加速度センサ値の算出表示を行います。

```
// 超音波距離センサ値、 3軸加速度センサ値の表示
// アナログ切換えスイッチを中央に、デジタル切換えスイッチ
void test2()
{
  tone(Spk_Pin,1000,1000);

   do{
    // 超音波距離センサ（LCD上段右側表示）
    digitalWrite(TrigPin, HIGH);
    delayMicroseconds(CTM);
    digitalWrite(TrigPin, LOW);
    float dis =(float)pulseIn(EchoPin,HIGH)*0.017;
    if(dis<10) tone(Spk_Pin,400,60);
//    digitalWrite(Led1Pin,(dis>10)?HIGH:LOW);
    digitalWrite(Led2Pin,(dis>10)?HIGH:LOW);
    digitalWrite(Led3Pin,(dis>12)?HIGH:LOW);
    digitalWrite(Led4Pin,(dis>14)?HIGH:LOW);
    digitalWrite(Led5Pin,(dis>16)?HIGH:LOW);
    digitalWrite(Led6Pin,(dis>18)?HIGH:LOW);
    MMA8452Q_readAcc();
    sprintf(pr,"%4d%4d",(int)dis,acc.x/10);
    lcd_setCursor(0, 0);
    lcd_printStr(pr);
    sprintf(pr,"%4d%4d",acc.y/10,acc.z/10);
    lcd_setCursor(0,1);
    lcd_printStr(pr);
     delay(100);
  }while(digitalRead(But_Pin));
  delay(500);
}
```

ブザーを
1 秒間鳴らす

超音波距離センサ
値の算出・表示

加速度センサ値
の算出・表示

3．検査テストスケッチ ④

TEST 3 のスケッチ
① ブザーを鳴らし、② 超音波距離センサ値の取得表示、③ 加速度センサ値の算出表示を行います。

```
#define RegPin A3
#define SpkPin 9
#define  MAX_SIGNALS  16
//const int  LedPin = 2;
const int  ButPin = 2;
boolean analyzeIR();
void  saveIR(byte cmd);
void  loadIR(byte cmd );
void dumpIR();
void  controlIR();
void  shotIR(uint16_t width);


char *control[2]={"CMD  000","CMD  001"};
// グローバル変数設定


void test3()
{
// 接続ピン設定番号
  tone(Spk_Pin,500,1000);
  irout_bit = digitalPinToBitMask(IROutputPin);
  irout_port = digitalPinToPort(IROutputPin);
  irin_bit = digitalPinToBitMask(IRInputPin);
  irin_port = digitalPinToPort(IRInputPin);
  //   設定
  pinMode(IROutputPin,OUTPUT);
  byte cmd;
```

LCD表示文字設定

ブザーを1秒間鳴らす

IR関連初期設定

```
char pr[8];
    for(int i=0; i<2; i++) {
        lcd_setCursor(0,0); lcd_printStr(">IR Read");
        cmd = i;
        lcd_setCursor(0,1); lcd_printStr(control[cmd]);
        // ボタンスイッチが押された状態→セットアップ
         delay(10);
      //  while(!digitalRead(ButPin)==LOW);
        delay(100);
        while(!digitalRead(ButPin)==HIGH);
        lcd_setCursor(0,0); lcd_printStr("Ready...");
        while (! analyzeIR())  {lcd_setCursor(0,1); lcd_printStr("Read Err");};
        lcd_setCursor(0,0); lcd_printStr("***GetIR");
        saveIR(cmd);    // Save result to eeprom
        lcd_setCursor(0,1); lcd_printStr(control[cmd]);
        delay(500);
     }
    lcd_setCursor(0,0); lcd_printStr("Cmd set ");
  boolean sw;
  do{
     lcd_setCursor(0,1); lcd_printStr(control[sw]);
     loadIR(sw); dumpIR();
     controlIR();
     delay(3000);
     sw = !sw;
   } while (digitalRead(ButPin));
    lcd_setCursor(0,1); lcd_printStr("TEST END") ;
    loadIR(1);
    dumpIR();controlIR();
    delay(100);
}
    //
```

おわりに

本マニュアルは、タブレインで培ってきたArduino関連技術の集大成となるものです。
皆様に自由にご利用いただくために無償で配布しています。

内容物について、ご相談については、受け付けていません。
ただ、IoTABシールド V4.0のご利用においてご質問等は、自由に受け付けていますので、気軽にご連絡いただけましたらと思います。

特に、アイデアによる新たな使い方があれば、是非ともご連絡いただけましたらと思います。お待ちしています。

タブレイン
into@tabrain.jp