

3 G通信技術を活用した

IoT教材キット V4.0

(オープンソースハードウェアArduinoによる試作開発期間短縮技術)

テキスト R4.1



Arduino UNO R3



Genuino101



Arduino M0 Pro



IoTABシールドV4.0

本ドキュメントについて

- ▶ 本ドキュメントは、株式会社タブレインが独自に作成してきたIoTプロトタイピング開発教育向け教材です。
- ▶ 本マニュアルの無断複写はお断り致します。
- ▶ 学校法人内でご利用の場合で、かつタブレイン直販による事前に購入者が分かっている場合には、問題ありませんが、代理店を通じてご購入され複写ご利用される時は、事前に info@tabrain.jp へお問い合わせお願いいたします。

【もくじ】

はじめに

第Ⅰ編 入門編

第Ⅱ編 Arduino編

第Ⅲ編 IoTABシールド編

第Ⅳ編 補足資料



Genuino101



IoTABシールドV4.0

IoT教材キット V4.0 マニュアル R4.0

作成・著作権 株式会社タブレイン

もくじ

1. 本資料の利用目的と進め方
2. 最新のモノづくり情報



はじめに

1. 本資料の利用目的と進め方

▶ 本資料のご利用目的

- ▶ **本資料は、オープンソースハードウェアArduino互換機**を簡単、かつ短時間に学ぶための情報を掲載しています。
- ▶ Arduino互換機を学ぶポイントは、以下のような項目となります。
 - ① システムとは、「入力」と「処理」と「出力」から成り立つ。
 - ② 電子部品には、「入力」用電子部品と「出力」用電子部品に分かれる。
 - ③ Arduinoには、「アナログ/デジタル」入出力ポートと、電源・GNDポートが用意されている。
 - ④ Arduinoの入出力は、「アナログ」と「デジタル」と「シリアル通信」が用意されている。
 - ⑤ Arduinoの処理（プログラミング）は、C（C++）言語に近いArduino言語によって、IDE（統合開発環境）上で開発し・Arduinoボードに書き込んで実行する。
 - ⑥ Arduinoの使い方をさらに加速化させるためにインターネットを活用し、自分なりに複雑なことへ挑戦してみる。

▶ 本資料を使った進め方

- ▶ まずは、① サンプルスケッチを動かし、② プログラムの内容を理解し、③ 変更・応用したプログラムを作成・実行してみる。
- ▶ 「真似て、技術を習得し、さらに変更・応用し、拡張すること」でレベルアップを図るようにしよう。

2. 最新のモノづくり情報

▶ オープンソースハードウェアの進化について

モノづくりにおいて、オープンソースハードウェアの波が押し寄せてきた。このことにより、誰もが、簡単に、類似製品を開発し、公開し、ネット上のSNS（ソーシャルネットワーク）を通じて、さらにモノづくりを安価で、簡単に、短時間で作れる環境が出来上がってきたことになる。

▶ モノづくりの統合的な環境（世界の動き）について

- ▶ 3Dプリンタが2012年に価格破壊が起きた。安価な3Dプリンタが出現した背景には、オープンソースハードウェアArduinoを使って、モータの制御やセンサー制御が簡単に行え、しかも短期間でモノづくりができたことによる。
- ▶ さらに2014年パソコンCPU最大手メーカのインテルが、オープンソースハードウェアの世界に乗り出してきたことは、広く知れ渡るようになった。
- ▶ インテルは、2014年1月にGalileoを販売開始し、10月にはEdisonを販売開始するようになった。ともにArduino互換機として動かすことができ、しかも多くのArduinoの資産が利用できる特長がある。さらに2016年にはArduino/Genuino101をArduinoUNOと同サイズで製品化し販売開始した。
- ▶ その他、FABLABと呼ばれるモノづくり工房が、世界中に起こり、レーザカッターや3Dプリンタなど、自分でも簡単に利用できる道具が用意された環境が増えてきた。
- ▶ また、これらのことで、個人によるモノづくりがどんどん増え続け、Maker Faireと呼ばれる「モノづくりイベント」が、これまた世界中に広がり始めていて、個人力が企業力を脅かすまでになってきている。
- ▶ さらに、個人力で開発した新しいモノづくりを資金面で支援するため、クラウドファンディングなるものもインターネット上で立ち上げた企業もあり、すでに企業設立やモノづくりの量産化支援で、期待以上の資金も集まるようになってきた。
- ▶ 一方、いろいろなところでのモノづくりを支援する企業や団体も増え、具体的なアイデアを出し合い、実際にモノづくりしてみるイベントのハッカソン、メイカソン、アイデアソンなども世界中に広がりはじめている。

もくじ

- 第1章 モノづくり革命
- 第2章 ArduinoとRaspberryPiの旋風
- 第3章 Arduinoの概要
- 第4章 モバイルM2M/IoTについて
- 第5章 センサネットワークによるM2M/IoT
- 第6章 期待されるM2M/IoTビジネス



Genuino101

第I編 入門編

日本のモノづくりから、今大きなビジネスチャンスのあるM2Mについて紹介、さらにオープンソースハードウェアArduino概要を説明

もくじ

1. 日本のモノづくりと国際競争
 2. 日本の最先端のモノづくりとは
 3. 日本のモノづくり現場の環境変化
 4. 世界の大きな波による技術改革
 5. メーカームーブメントの全貌
 6. メーカー・ブームのトリガー
 7. モノづくりを後押しする情報サイト
 8. モノづくりイベント
 9. オープンソースハードウェアの出現
 10. モノづくりを後押しする環境
 11. オープンイノベーションの波
- 【ブレイク】Arduino + 3 GIMの可能性



第1章 モノづくり革命

1. 日本のモノづくりと国際競争

- ・ 明治維新のモノづくり

- ▶ 鎖国時代から一挙に世界を知ること
- ▶ 士農工商の身分制度がなくなり自由な思想に

- ・ 戦後復興のモノづくり

- ▶ 復興に向けた日本の成長
- ▶ 「物まね」から「品質向上」、さらに「カイゼン」

- ・ 新たなモノづくりの革新（アナログからデジタルの世界へ）

- ▶ 日本の強みを活かしたモノづくりへの挑戦
- ▶ 最先端で高度・精密な技術を生かしたモノづくりへの挑戦

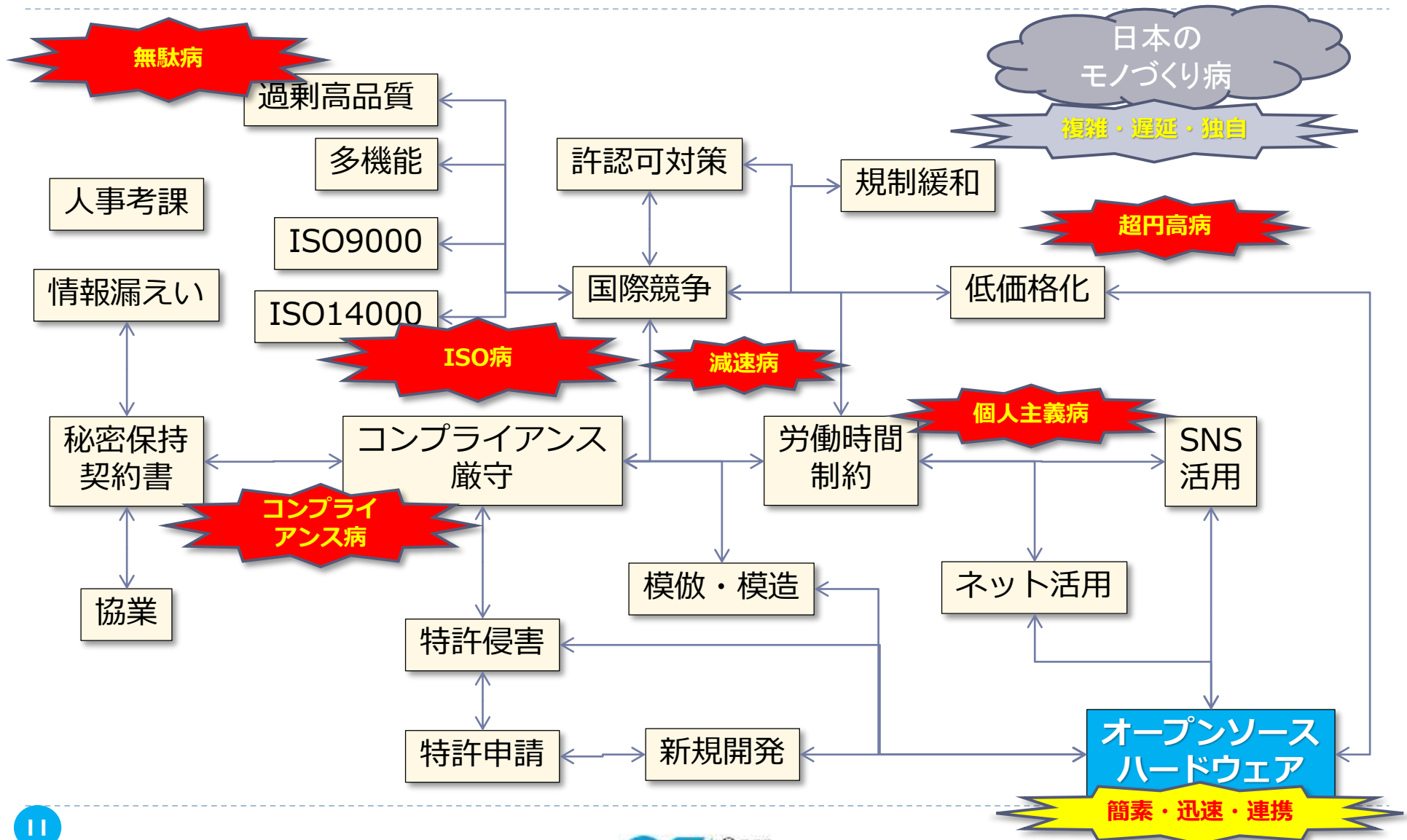
2. 日本の最先端のモノづくりとは

- ▶ 最先端の部品（モータや電子部品、MEMSなど）
 - ▶ 高性能で、高機能、高品質、しかも低価格なセンサ類・アクチュエータ類が日本では豊富に入手可能

- ▶ 最小化・高品質・高機能・多機能に向かう日本製品
 - ▶ モノづくりへのこだわりで生まれた技術
 - ▶ 国際標準化や業界標準化を忘れた日本技術

- ▶ ロボット産業とヒューマノイド・ロボット
 - ▶ 産業用ロボットは世界シェアNo.1だが、福島原発でのロボット活用では海外のものが最初に活躍
 - ▶ なぜ日本は、人間型（ヒューマノイド）ロボットを目指すのか？

3. 日本のモノづくり現場の環境変化



4. 世界の大きな波による技術改革

▶ インターネット

- ▶ あらゆるものがインターネットで接続しはじめた
- ▶ 互いに接続し、共有化し、協調しはじめる

▶ オープンソース

- ▶ プログラムのソースコードがオープン（無償、タダ、フリー）
- ▶ OCW（オープンコースウェア：MITによる無償による講演）
- ▶ TED（アイデア・プレゼンテーションを無償で公開）など

▶ ソーシャルネットワーク

- ▶ 時空間を超えた人間関係を再構築
- ▶ 知人の情報が信頼が高い（人に聞くことが重要）

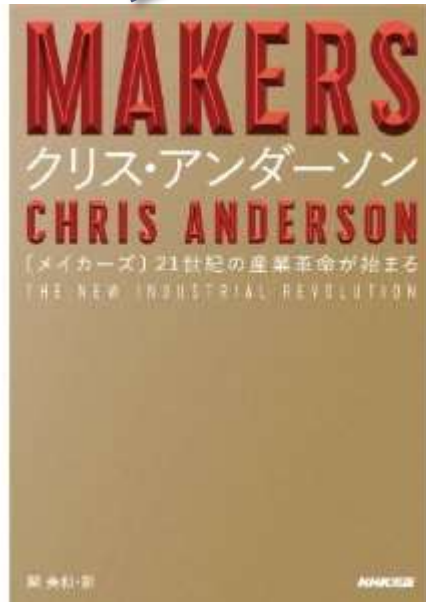
5. メーカームーブメントの全貌



6. メイカー・ブームのトリガー

もともと人間には、モノ作りする喜び（欲）がある

メイカームーブメントの
言葉の生みの親



2012年出版

ポイント

- ① 誰もが簡単にモノ作りできる環境ができたこと
- ② SNSで協業・共創しながらモノ作りできるようになったこと
- ③ 大量生産型から少量開発・一品モノ開発になってきたこと



オライリーの出版物「Make」

7. モノづくりを後押しする情報サイト

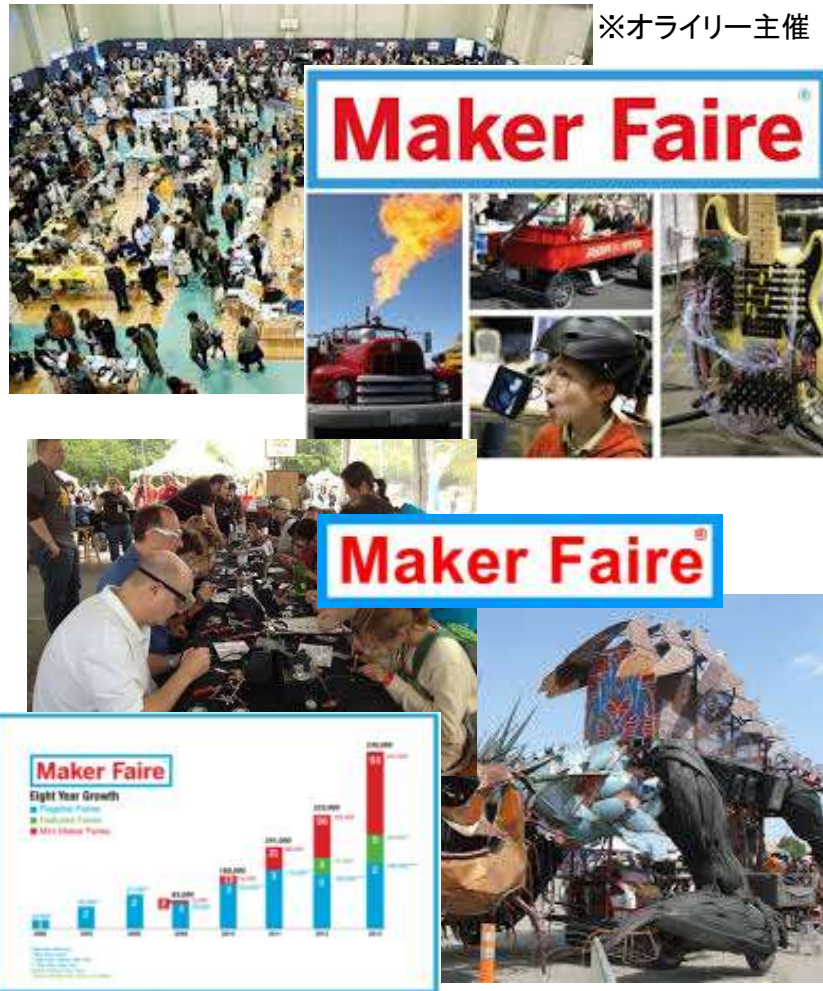
▶ ネット上のモノづくりの情報サイト(&展示会)



8. モノづくりイベント

■ Maker Faire（世界各国でブームに）

※オライリー主催



■ ハッカソン・メイカソン・アイデアソン マラソンとの造語で、みんなで集まってモノづくりをするイベント



人が出会ってモノづくりするイベント
共創・触発の環境・空間ができる

9. オープンソースハードウェアの出現

■ オープンソースハードウェア・マイコンボードの出現



Arduino Uno



Arduino Tre



GR-SAKURA
(ルネサス製CPU利用)



インテルGalileo
(現在Galileo 2 を販売中)



インテルEdison



Raspberry Pi



beagleBone Black



KURUMI
(ルネサス製CPU利用)



Geunino101

■ さまざまなワイヤレス関連拡張ボード・キットの出現



3 Gシールド



3 GIM
(世界最小クラス 3 G通信機器)



IoTABシールド



Wi-SUN



ZigBee



EnOcean



BLE

簡単、安価、短時間に
モノづくりできる環境 (ツール)

10. モノづくりを後押しする環境

■ FabLab（ファボラボ）

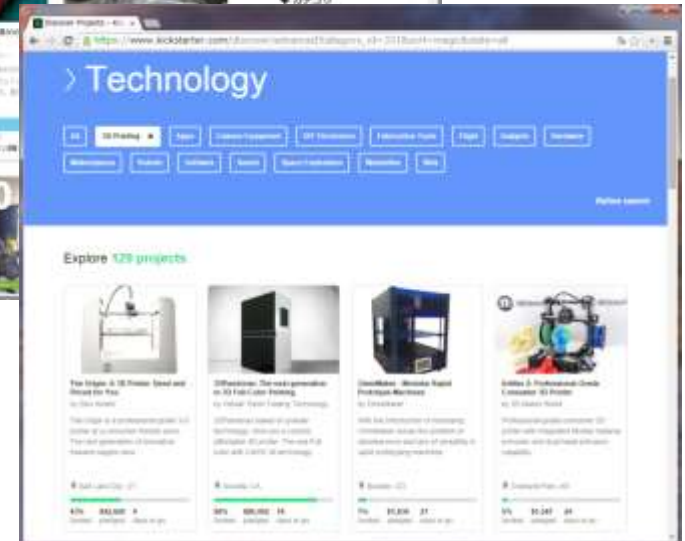
全国的に広がりを見せているモノづくり環境（施設）



直ぐにモノづくりする環境提供
安価・便利・豊富なツール・道具

■ クラウドファンディング

モノづくりを支援する資金調達のサイト



独自アイデアを紹介することで
資金調達をネットで行うサイト

11. オープンイノベーションの波

- ▶ 企業は、ISO取得やコンプライアンスなどに追われ、社外との協業体制も難しくなっている。
- ▶ さらに、円高などにより、開発費の削減も余儀なくされている。

いままでは

日本のモノづくりの成功の原点は「**組織**」にあったが

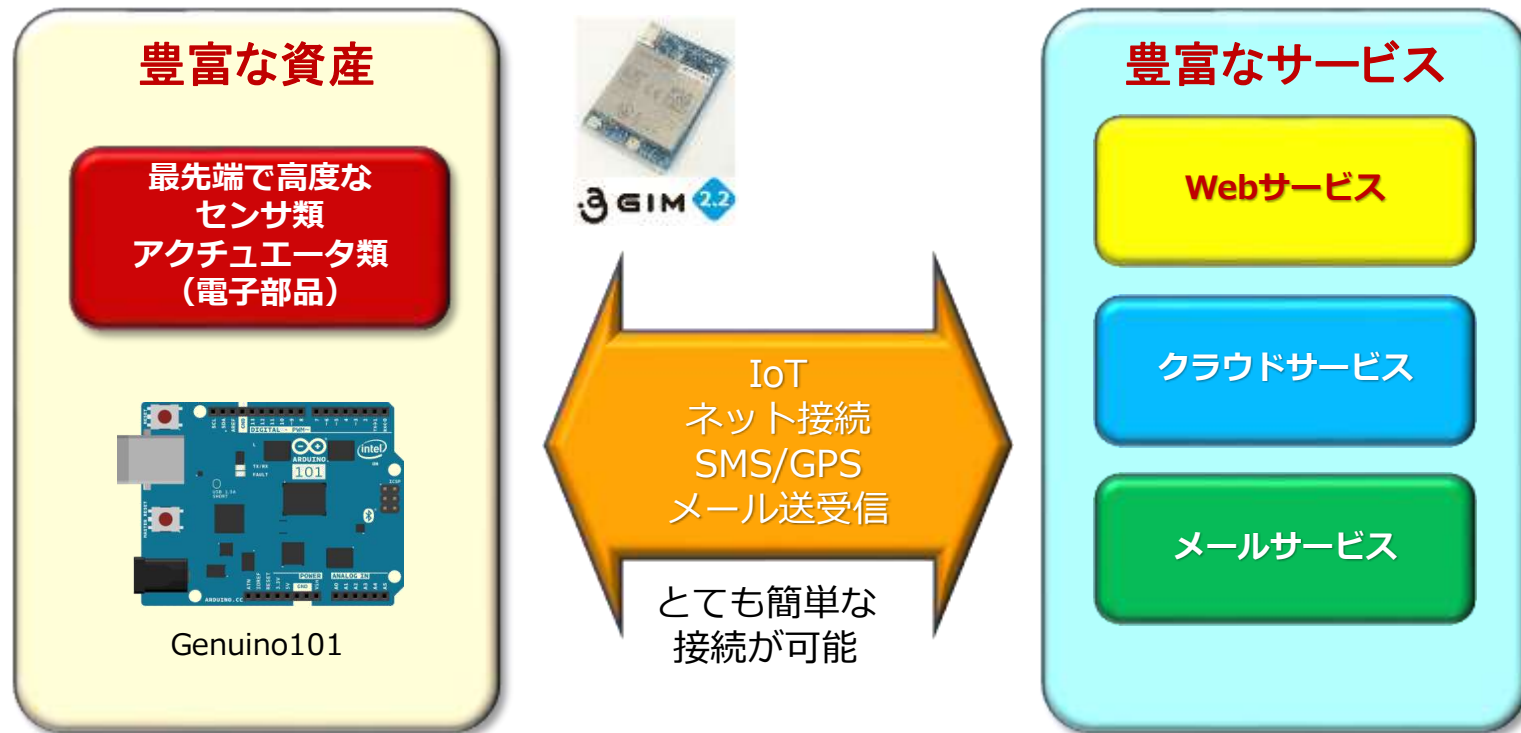
これからは

個々人が自由に自分のアイデアをさらけ出す時代に

そのためのオープンソースハードウェアは、強力な助っ人に

あらたな世界市場に「**メイカーズ市場**」が誕生しはじめている

【ブレイク】 Arduino+ 3 GIMの可能性



【Arduino + 3 GIMの可能性の方程式】

$$3 \text{ GIMの可能性} = \int_0^{\text{豊富なサービス}} \text{豊富な資産 (Arduino)}$$

アイデアによっては、無限大の可能性を紹介すること

**M2Mビジネス以外にも
教育・研究でも可能性は大きい**

もくじ

1. マイコンボードとボードコンピュータ
 2. Arduinoとは
 3. RaspberryPiとは
 4. ArduinoとRaspberryPiの魅力
 5. ArduinoとRaspberryPiのとの比較
 6. IoT向け利用のArduinoとRaspberryPi
- 【ブレイク】Arduino初心者向け参考書



オープンイノベーションに大きな影響を与えている2つのボード

第2章 ArduinoとRaspberryPiの旋風

1. マイコンボードとボードコンピュータ

共に**教育用ボード**として誕生。今、**IoT 向けの デバイス・ゲートウェイ・サーバ**で利用されはじめた

2005年イタリア生まれ



マイコンボード
(種類も豊富)

オープンソースハードウェア

多くの互換品や類似品・コピーが製造販売されている

2012年イギリス生まれ



インターネット付
ボードコンピュータ
(現在8種類)

2015年10月まで700万台販売

共に簡単に短時間で、しかも安価で、IoTデバイス・IoTノード・IoTゲートウェイ・IoTサーバなどが構築できる

Arduinoは簡単にマイコン技術を学ぶツールとして
2005年イタリアの大学の先生らが開発したボードです。

2. Arduinoとは

- ▶ Arduinoとは、マイコンボードと統合開発環境のこと



Atmel AVR (8ビットマイコン)

現在は、16ビット、
32ビットマイコンまで
シリーズで登場



統合開発環境 Arduino
(C言語風の開発環境)

- ▶ コンセプトは、**オープンソースハードウェア**
 - ▶ ハードウェアの回路図が公開され、誰でも同じもの（ボード）が開発・製造可能
 - ▶ 既に多くのクローン製品・類似製品が世の中に販売されている
 - ▶ さらに多くの拡張ボード（シールドと呼ばれる）も販売されている
 - ▶ 統合開発環境には、多くのサンプルプログラムが掲載され、誰でも利用可能
 - ▶ センサとアクチュエータが簡単に利用できる

3. RaspberryPiとは

RaspberryPiは2012年にイギリスのRaspberryPi財団が学校でのコンピュータ教育向けに開発したシングルボードコンピュータです。

- ▶ RaspberryPiは、ARMプロセッサ搭載し、Linuxで開発利用する。インターネット接続が容易。



- ▶ ボードコンピュータRaspberryPiは、OSを持つ超安価Linuxマシン
 - ▶ 開発環境はキーボード・ディスプレイを接続、インターネットに接続して利用
 - ▶ IoTゲートウェイやIoTサーバなどに利用
 - ▶ 高速な処理が可能だが、消費電力は大きい（電源ケーブルが必要）

4. ArduinoとRaspberryPiの魅力

- ① 価格が安い → Arduinoはシンプル・クローン品も多い
RaspberryPi Zeroは、僅か5ドル（510円程）
- ② 開発環境が充実 → 組み合わせが簡単
（多くのソフト・部品がある）
- ③ 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
- ④ デファクトなボード → 世界的に認められるデファクト（業界標準）
- ⑤ 魅力ある応用展開 → 3Dプリンタやドローン、ロボットのベースに

5. Arduino と RaspberryPi との比較

©tabrain	Arduino + 互換機	RaspberryPi
ボードの特徴	マイコンボード（OSなし） シングルタスク：リアルタイム性確保可能	ボードコンピュータ：Linux（OSあり） インターネット接続が必須、リアルタイム性確保なし
特徴	インタフェースが充実（各種センサ利用が容易） ネット上に豊富な事例が蓄積	GPIO・UART・I2C・SPI（アナログI/Oなし） サーバ構築・インターネット接続が容易
開発環境	専用IDE（JAVAベースの開発言語）、Scratch、Processing	Scratch, IDLE, Windows10IoTなど
IoTシステムでの利用	IoTデバイス・ノード向き（ 3Gシールド 付でIoTゲートウェイ構築可能）屋内・屋外利用可能	IoTゲートウェイ向き、サーバ構築向き ほぼ屋内向き（移動体には向かない）
利用者	初心者向き（小学生からお年寄りまで）、組み込み技術・制御技術が学べる、センサ技術+マイコン技術が学べる	中級・上級者向き（マニア、情報技術者）、インターネット接続関連が学べる、サーバ構築ができる
強みと弱み	強み：コンパクト・省エネ・IoTノード開発向き 弱み：動画などの高速処理が難しい	強み：高速インターネット接続可能（動画送信可能） 弱み：センサ接続などは大変・その他にも・・・

	Arduino Uno R3	Raspberry Pi 3 Model B+
価格（2019年2月時点）	3240円	5670円
サイズ	7.6×1.9×6.4(cm)	8.6×5.4×1.7(cm)【名刺サイズ】
メモリ	2 KB (0.002MB)	1 GB (1000MB)
CPU	Atmega328P 16MHz	ARM Cortex-A53 1.4GHz
オンボードネットワーク	無し	10/100/1000BASE-T (RJ45) 有線LAN 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN Bluetooth 4.2, BLE
マルチタスク	不可（リアルタイム性あり）	可（リアルタイム性なし）
入力電圧	USB 5V、外部7～12V	USB 5V
フラッシュメモリ	32KB	SDカード（2～16G）
USB	1口（入出力）	4口（一般的な周辺機器接続可能）
OS	無し	Linux、Raspbian、PIDORAほか
統合開発環境	Arduino IDE、Scratchほか	Scratch、IDLEなどのLinuxで動くもの

だからArduino+RaspberryPiの製品も出てきた



UDOO NEO (2015年5月-)

6. IoT向け利用のArduinoとRaspberryPi

利用項目	Arduinoおよび互換機	RaspberryPiシリーズ
センサ利用	○ アナログ・GPIO、UART、I2C、SPIのセンサ利用可能	△ アナログセンサの利用は不可（別途AD変換必要）
アクチュエータ利用	○ 制御がシングルタスクで簡単	△ マルチタスク処理によって制御が複雑
その他拡張	△ データ処理が遅いため動画処理は難しい	○ データ処理が早いため動画処理は得意
利用場所	○ 省エネ対応できることから、屋内から屋外、また小型化によって移動体に利用可能	△ 消費電力が大きいことから、電源確保できているところがほとんど
応用範囲	○ 省エネボード対応の小型から大型までの制御ボードまで開発可能で、応用分野は広い	△ 応用分野は制限され、屋内利用がほとんど。
試作段階 (プロトタイピング)	○ 既存Arduinoおよび互換ボードでの試作・プロトタイピング開発は簡単で短時間に開発可能	○ 高速な処理向きで、マルチタスク処理も利用可能
量産化段階	△ ボードのカスタマイズなどで期間や費用	○ RaspberryPiを組込んだシステム開発が安価に開発可能。
IoT向け利用	IoTノード・IoTデバイス・IoTゲートウェイと幅広く利用できる	IoTゲートウェイ・IoTサーバとして利用

©tabrain

【ブレイク】Arduino初心者向け参考書

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。

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-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



2014年2月17日発行
第4刷2016年8月発行
韓国翻訳版2016年6月出版
(アマゾンよりお買い求めできます)



もくじ

1. オープンソースハードウェアの紹介
 2. Arduinoを使ってできること
 3. Arduinoとは
 4. Arduino UNO と互換機について
 5. Arduinoの普及展開
 6. Arduinoの人気の秘密
 7. Arduinoのハードウェア
 8. 豊富なArduinoシールド
 9. Arduinoの構築環境
 10. Arduinoシリーズ
 11. Arduinoの魅力
- 【ブレイク】Arduino関連の参考本（邦書）



第3章 Arduinoの概要

1. オープンソースハードウェアの紹介

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



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 UNO



GR-SAKURA



What's Next Yellow

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)

2. Arduinoを使ってできること

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

3. Arduinoとは

- ▶ Arduinoは、オープンソースハードウェアにより開発されたマイコンボードで開発には統合開発環境 Arduino IDE を使用する。
- ▶ Arduino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>



ハード

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

+

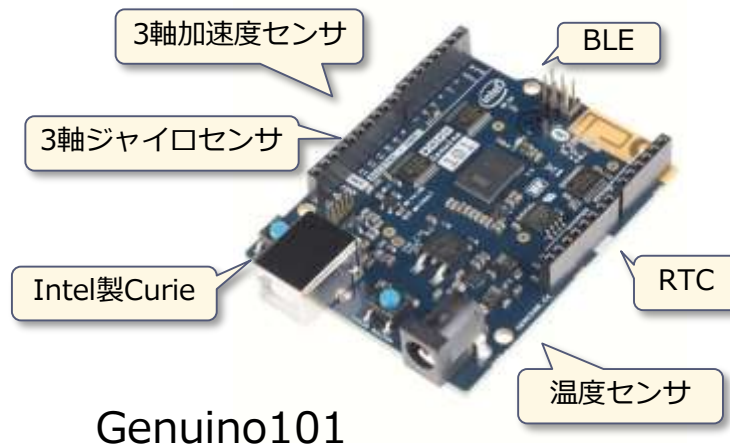


ソフト

統合開発環境 Arduino IDE
(C++言語風の開発環境)

4. Arduino UNO と互換機について

- ▶ Arduino UNOと互換機について
 - ▶ Arduino UNO は、Arduinoシリーズで最も利用されている8ビットマイコンボード
 - ▶ Arduino UNOとのピン互換のボードに、Arduino Leonardo、Arduino Zero、Arduino M0 Pro、Genuino 101など数多く出ています。
 - ▶ EEPROMやフラッシュメモリ、SRAMを備えています。
 - ▶ 開発環境はArduino IDE（統合開発環境）※ネットからダウンロード可

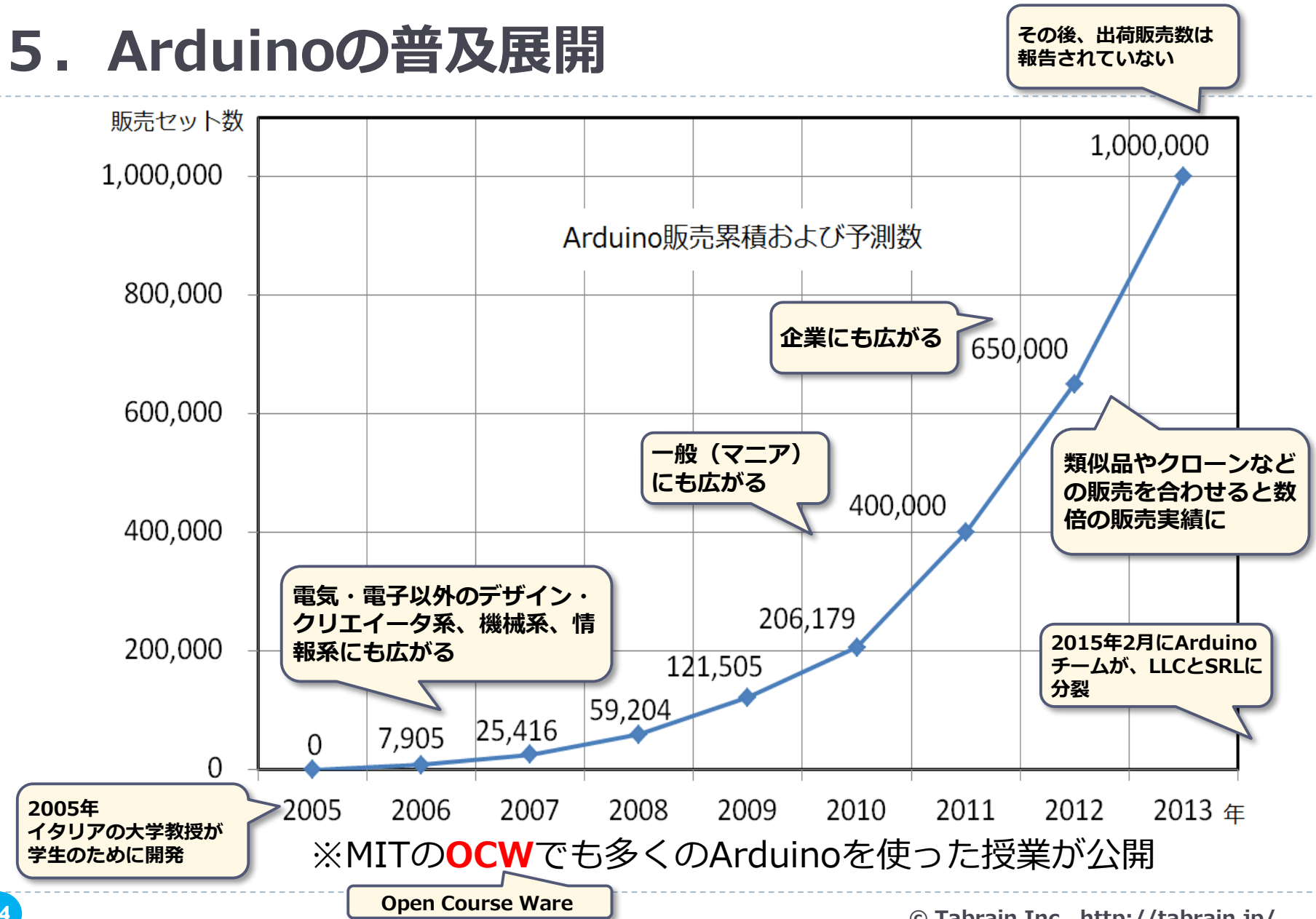


※Arduino UNOと同じコネクタ（ピン互換）を持つ
インテル製MPUのCurieを使った互換ボード



統合開発環境 Arduino IDE
が利用可能

5. Arduinoの普及展開



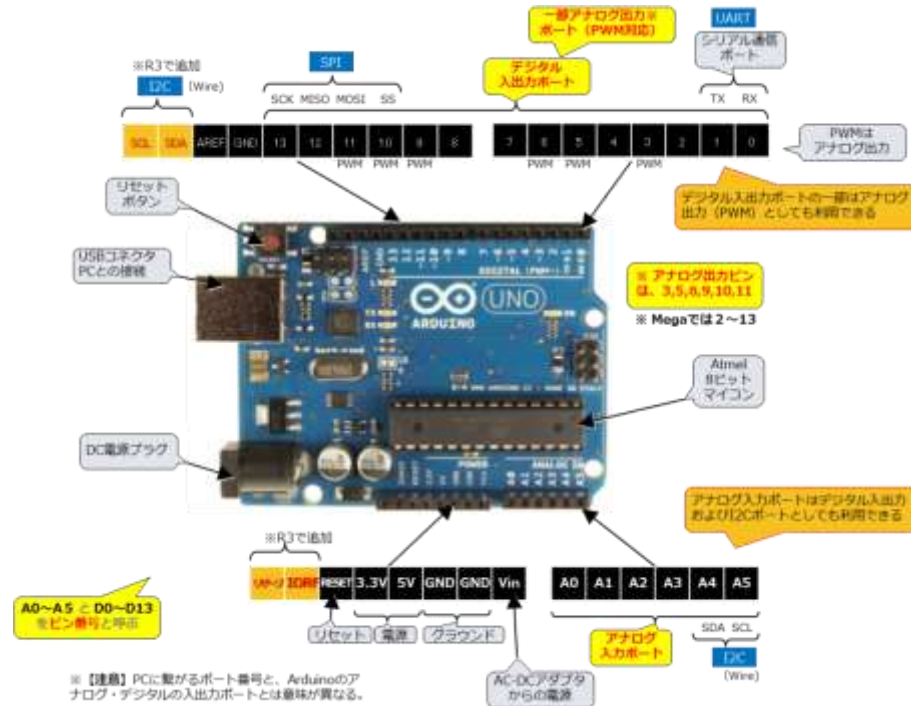
6. Arduinoの人気の秘密

- 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

7. Arduinoのハードウェア

基本は、Atmel AVRの8ビットマイコン。拡張製品の種類が豊富。

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。
機能は同等で小型化したものや、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）

Official Arduino boards

Official Arduino Shields



Arduino Uno



Arduino LilyPad



Arduino Ethernet Shield



Arduino Mega 2560



Arduino Fio



Arduino Wireless SD Shield



Arduino Mega ADK



Arduino Pro



Arduino Wireless Proto Shield



Arduino Ethernet



Arduino Nano



Arduino Motor Shield

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

8. 豊富なArduinoシールド

豊富なArduinoシールド（センサなどが搭載された拡張ボード）

3 GIM シールド

モータ駆動シールド



音声シールド



液晶表示シールド



Arduino/Genuino



カラーLCDシールド



温度センサシールド



心拍センサシールド



IoTABシールド

イーサネットシールド



9. Arduinoの構築環境

3Dプリンタ・ドローン・
ロボットなどの製作でも
多く利用されている。

豊富なスケッチが
Webサイトに

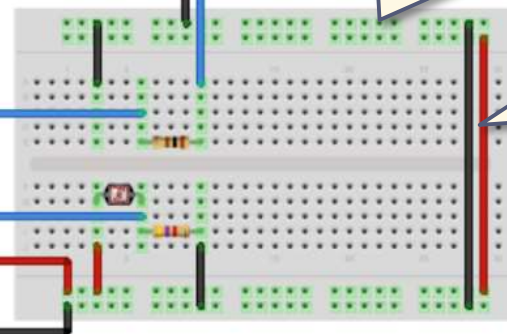
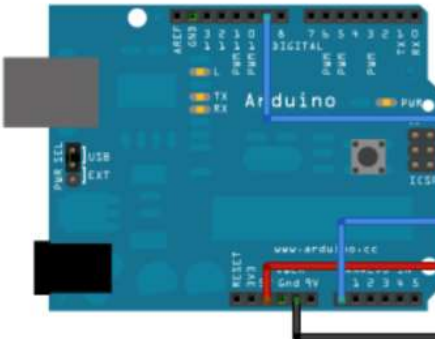
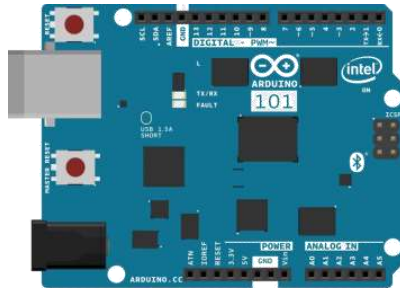
豊富なセンサ・
シールド類

ブレッドボード

ジャンパーコード

はんだ付け不要

試作が
安く・早く
簡単に
構築可能



10. Arduinoシリーズ

最新版の32ビットCPU使用（約6,000円弱）

最も多く販売（約3,000円以上）

Arduino 仕様	Genuino101	UNO R3	Leonardo	Mega ADK	Due	Pro
マイクロプロセッサ	Intel Curie	ATmega328P	ATmega32U4	ATmega2560	ATSAM3X8E	ATmega168/328P
動作電圧	3.3 V (5V tolerantI/O)	5 V			3.3V	3.3V/5V
推奨入力電圧	7－12V					3.35－12V(3.3V) 5-12V（5 V）
制限入力電圧	7－20V	6－20V				
GPIOピン数	14（うち4ピンPWM出力）	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）
PWMチャンネル	4	6	7	15	12	6
アナログI/Oピン	6	6	12	16	I：12/O:2(DAC)	6
I/Oピン電流	20mA	40mA	40mA	40mA	130mA	40mA
供給可能な最大電流	50mA	50mA	50mA	50mA	800mA	
フラッシュメモリ	196KB	32KB（うち0.5KBはブートローダ用）	32KB（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)
SRAM	24KB	2KB	2.5KB	8KB	96KB（2バンク：64KB・32KB）	1KB(168) 2KB(328)
EEPROM	-	1KB	1KB	4KB		512B(168) 1 KB(328)
クロック周波数	32MHz	16MHz	16MHz	16MHz	84MHz	8MHz(3.3V) 16MHz(5V)
備 考	ハードシリアル1 慣性センサ付 BLE対応			ハードシリアル3		

【arduino.ccサイト参照】

1 1. 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、デザイナー系・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」Arduinoの記事にて）
- ▶ 試作された作品の魅力
 - ▶ 限られ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・試作できることが魅力）
 - ▶ 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）

【ブレイク】 Arduino関連の参考本①（邦書）



日本のArduino先駆者
IAMAS 准教授小林茂先生

2014年2月に出版
2017年12月に第5刷
(韓国翻訳版も出版)

学研 金子茂氏と
スイッチサイエンス社
金本茂社長関与

もくじ

1. 3G通信網とセンサの利用技術
2. ワイヤレスによる通信網
3. 通信網の利用にむけて



第4章 モバイルM2M/IoTについて

1. 3G通信網とセンサの利用技術



1.1 モバイルの可能性

▶ モバイル【mobile】とは

- ▶ 携帯電話機、携帯できる、移動できる、動きやすい、といった意味の英単語。また、モバイルコンピューティングの略称で、もともとは携帯型コンピュータを持ち運んで利用すること。
- ▶ 現在では、携帯電話など持ち運びができる通信機器と、コンピュータ機能をもつ端末を組み合わせ、外出先や移動中にデータ通信を行ったり、インターネットを利用することをモバイルというのが一般的。

(NTT東日本Webサイトより)

▶ モバイルの利用価値

- ▶ 電子メール
- ▶ 画像・映像配信
- ▶ 情報収集
- ▶ ファクス送受信
- ▶ インターネット接続
- ▶ 各種予約
- ▶ モバイルバンキング

モバイルとセンサを繋ぎ合わせて新たな価値を生む可能性を探る

2. ワイヤレスによる通信網

	セルラー系			非セルラー系				
	LTE	3G	WiMAX	WiFi	Bluetooth	ZigBee	EnOcean	Wi-SUN
IEEE規格	Long Term Evolution	ITU-2000	802.16	802.11a/b/g/n	802.15.1	802.15.4		IEEE802.15.4g
周波数帯	1.25/5/10/20MHz	800MHz/2GHz	2.5GHz	2.4GHz/5GHz	2.4GHz	866MHz/915MHz/2.4GHz	315MHz (日本)	920MHz (日本)
通信距離	最大10km前後	最大10km前後	最大50km	最大100m程度	最大100m程度	100m以上 (2.4GHz)	ビル内30m 最大300m	約 1 kmほどまで
通信速度	最大85Mbps	最大 7 Mbps	最大70Mbps	最大300Mbps	最大3Mbps	最大250kbps	125kbps	~200 kbps
接続ノード数				32	7	65535		
電池寿命	機器依存	機器依存	機器依存	数時間	数日 (BLEは数年)	数年	長寿長 電池なし	特定小電力無線
ネットワーク構成	スター型	スター型	スター型	スター型	スター型	スター型・メッシュ型など	スター型	スター型・メッシュ型など
主な用途	広域エリア・スマートフォン対応	広域エリア・携帯電話	ほぼ広域エリア・モバイル通信	狭いエリア・無線LAN	携帯電話・コンピュータ付属・他	セキュリティ機器	バッテリーレス照明スイッチ他	ガス・水道・電気のメータ系端末
備考	今後IoT向けのCat1やCatMなどが登場	4Gでは、通信速度が下り100Mbps、上り50Mbps実現	速度と伝達距離が次世代技術と現行技術の中位にあり、今後のWiMAX 2に期待		機器やロボット制御などで利用 BLEは、商用・モニタリング・監視・見守りなど	ロボット・などで利用	物理的なスイッチング動作を電気エネルギーに変換 (建物内で利用)	東京電力でメータ監視に利用開始・マルチホップなど特徴



無線の魅力： ケーブル（電源・電波）なしでの通信 ⇒ 可搬性や移動が可能

3. 通信網の利用にむけて

▶ データ通信の価値

- ▶ 3Gでは音声通信とデータ通信の2種類利用
- ▶ 音声通信は有限個数での通信。
データ通信は待ち状況があるが、ほとんどのデータ送受信は可能。
- ▶ 端末側で取れたデータを3G通信でやり取り可能。
※ 地震などの災害時には、音声通話がフルとなり繋がり難くなる。

▶ データ通信での音声処理

- ▶ Skypeなど、データ通信を通じて電話を掛ける
- ▶ Wi-Fi通信可能な場所にあるスマートフォンは、無償Skypeでコンピュータや他のスマートフォンに繋ぐことが可能。

▶ データ通信の利用価値

- ▶ 動画や画像以外のテキストデータだと、瞬時に簡単にデータ転送可能
- ▶ 何ができるか ⇒ 異なる場所（エリア）での継続的・断片的なデータ転送が可能

もくじ


1. センサによる収集データの活用
2. 通信網とセンサの結合
3. ワイヤレスセンサネットワークについて
4. センサネットワークの構築ポイント
5. Arduinoを使ったセンサネットワーク
6. 3 GIMによるセンサネットワーク
7. センサ能力の分類
8. センサ応用事例
9. データの可視化・認識化
10. センサの採取データについて



第5章 センサネットワークによるM2M/IoT

1. センサによる収集データの活用

- ▶ **人間界のデータ収集（人・自動車・建物などの感知）**
 - ▶ 防犯（人感知など）
 - ▶ 介護・医療（健康データの収集）
 - ▶ 安全（建物の振動・老朽化、温度＜火災＞、煙探知。自動車：運転履歴など）
- ▶ **動物界のデータ収集（動物の感知）**
 - ▶ 動物の病気（口蹄疫・狂牛病・鳥インフルエンザなど）
 - ▶ 動物の育成・保護（生態系調査など）
 - ▶ 動物の動き・集団検知（魚群探知など）
- ▶ **植物界のデータ収集（植物の感知）**
 - ▶ 植物の病気（乾燥・病原菌など）
 - ▶ 植物の育成・保護（ビニールハウス環境管理、樹木の成長など）
- ▶ **自然界・環境・防災・保全のデータ収集（無人での感知）**
 - ▶ 防災（地震動、津波検知、風力・風速計測、雨量計測、気圧など）
 - ▶ エネルギー（太陽光、太陽熱、風力・風向きなど）
 - ▶ 環境（農作物まわりの温度・湿度・二酸化炭素、Ph値、汚染度など）
 - ▶ 保全（建設物の維持管理データなど）



センサ値が
価値を生む
時代に

スマホ・家電・自動車にセンサが取りつくようになり、
高機能で最先端のMEMSセンサも急激に安価に

2. 通信網とセンサの結合

- ▶ **センサ利用によって検知・測定・観察データを取得**
- ▶ **即時に利用するものと、集積して後で利用するもの**
 - ▶ 何かに反応してデータを取得送信
 - ▶ 多くのデータを集積し、タイミングを図って送信
- ▶ **単体データの活用と収集データの活用**
 - ▶ 単体データとは、オン・オフなどのスイッチ（判断）などのデータや、ある時点でのデータ値
 - ▶ 収集データとは、さまざまなデータを集積・蓄積して活用するデータ値。収集データは、それらの値の解析・分析などから重要な判断材料を抽出する必要がある。
- ▶ **センサネットワークの活用**
 - ▶ 無線によるセンサの値を集積し、互いに協調しあって連携しあうネットワークシステムのこと。
⇒ アイデアひとつで、世の中に役立つ価値あるものが出来上がる。
 - ▶ ローカルなセンサーネットワークと広域通信網を使ったグローバルなネットワークの実現（大規模センサネットワーク）
⇒ 小さいセンサネットワークの積み重ねで、大規模なセンサネットワークの実現

3. ワイヤレスセンサネットワークについて

▶ センサネットワークでの課題

- ▶ 大容量のデータ処理（ビックデータ） ⇒ サーバ管理の技術利用
- ▶ クラウド処理 ⇒ 遠隔地での監視・管理技術の利用
- ▶ センサ技術 ⇒ 部品メーカーの技術利用（安価・高機能・高品質・耐久性など）
- ▶ 通信技術 ⇒ 無線による広域およびローカルな通信技術と、その組み合わせ
- ▶ その他技術 ⇒ 規格・標準化、省電力化、低価格化、

▶ ワイヤレスセンサネットワークの活用

- ▶ センサネットワークは、ローカルと広域での無線によるセンサ活用が重要に
- ▶ プロトコルやトポロジの解決（利用者にはシンプルが重要）

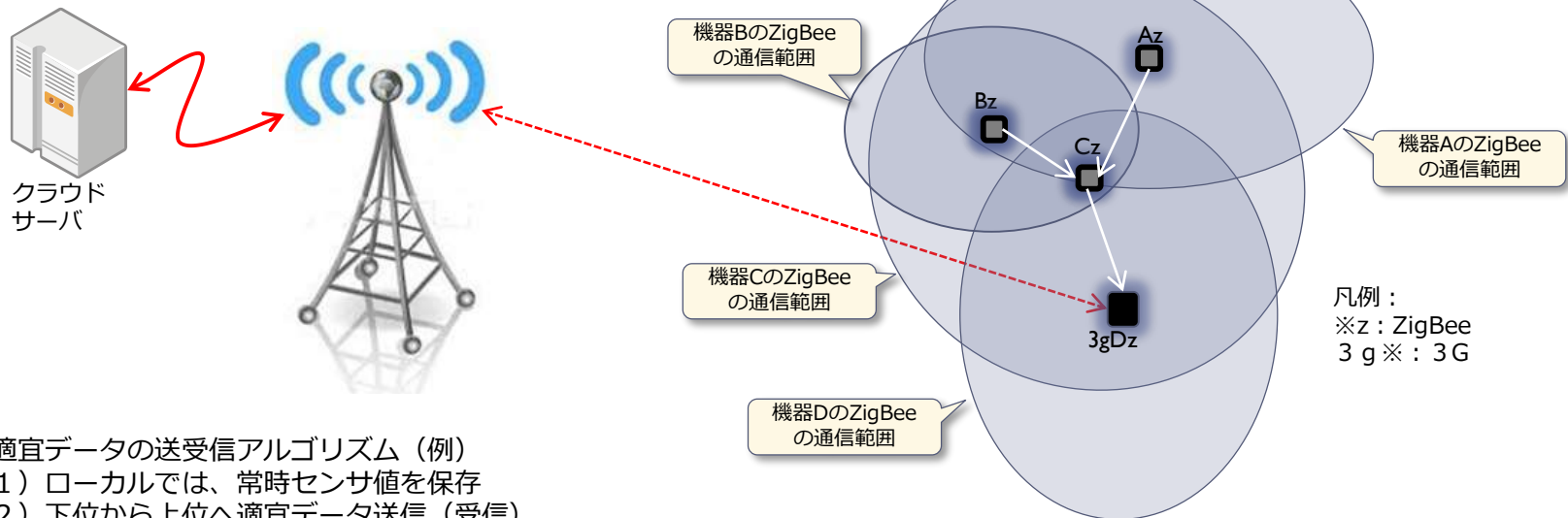
4. センサネットワークの構築ポイント

課題

- 1) 湿度によって通信状態が異なる
- 2) 季節によって通信状態が異なる
- 3) 通信状態に障害があったときに早期に発見
- 4) 適宜必要なときにデータ収集を実施
- 5) 障害がある時点ではローカルにてデータ蓄積保留
- 6) 必要な場合のみ集中してデータを受信

解決

- 1) 欠落の無いデータ収集が必要
- 2) 効率的なデータ収集が必要 → 通信効率（トポロジ最適化）
- 3) 小域と広域との通信網を使い分け
- 4) エラー（障害）発生を早期に発見
- 5) 低消費電力化対策、自然環境下対策、防虫・防水対策など



適宜データの送受信アルゴリズム（例）

- 1) ローカルでは、常時センサ値を保存
- 2) 下位から上位へ適宜データ送信（受信）
- 3) データ送信に不具合の場合：上位へアラーム発信
- 4) 親機と子機との通信設定状態
- 5) 加速度センサを使ったタグの動的状態把握

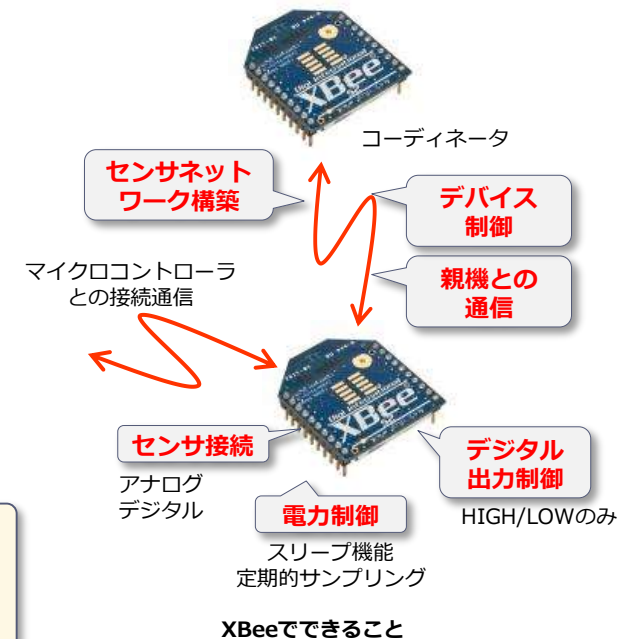
5. Arduinoを使ったセンサネットワーク

Arduinoで利用できるセンサ類 ⇒ 無限大に存在

- ▶ デジタル・アナログ、シリアル通信のセンサ類が簡単に利用可能
しかも、中高生でも簡単に構築できる（技術ハードルが低い）
- ▶ 豊富なセンサ部品が安価で入手可能（ネットでも簡単に入手可能）
温度、光、湿度、Ph値、二酸化炭素、人感、距離、傾斜、ジャイロ、加速度センサなどが、数円～数千円で入手可能
- ▶ ローカルワイヤレスセンサ技術で安価なZigBee技術が利用可能
ZigBee製品のひとつに安価で普及しているDigi International社の XBeeが存在
（価格は、1700円程度から）
- ▶ 試作が安価で、短期間で、容易に構築可能
既にネット上に多くのセンサ利用の事例（接続方法/配線方法とプログラム/スケッチが公開されている）が活用可能

【参考】

XBeeとは ZigBee規格の無線通信機能とマイコンを搭載した小型モジュールです。価格は約2000円前後、アンテナを含め、日本での技適取得済PAN製品のひとつです。



6. 3 GIMによるセンサネットワーク

- ▶ **Arduinoのセンサ技術・ローカル通信技術を活用し、3 GIM機能と連携することで広域によるセンサネットワークシステムが構築可能に**
- ▶ **農業・漁業での活用期待**
 - ・農業・漁業にITを活用する時代は、まだこれから科学的な生産性向上に向けた研究が盛んな時代に
 - ⇒ 田畑の土壌のPh値から、まわりの環境（温度、湿度、風速など）を継続的に観測していくことでより生産性の高い作物の作り方が見えてくる。
（多くの作物が対象に、その他樹木の育成でも調査の必要性が叫ばれている）
 - ⇒ 日本の広域で利用が広がる可能性大（背景に、農業人口の現象など）
- ▶ **建設・保全での活用期待**
 - ・保守・メンテナンスすべき建造物が増大し、安全・安心できる建設物の維持管理が必要な時代
 - ・橋梁・トンネルなどをはじめとする維持管理（検査含む）は、人力で行う時代からセンサ利用のIT活用時代に
 - ⇒ 科学的に腐食・変位・歪み・振動・音質を測定することによって保全観測を継続的に行い、人件費を削減した維持管理が必要な時代

7. センサ能力の分類

センサ能力を使って何ができるか？

3種類のセンサ能力

▶ 感知能力

- ▶ 人間の代わりに感知
 - ▶ 監視、防犯、照度などの感知
- ▶ 人間にできない感知
 - ▶ 五感以外の感知（二酸化炭素濃度とか）、
 - ▶ 五感でも高精度な感知（超音波や紫外線・赤外線など）
 - ▶ 見えないものの感知（電波・ガスなど）
 - ▶ 危険な環境での感知など

▶ 計測能力

- ▶ データ計測し、集積・分析・解析から推測へ

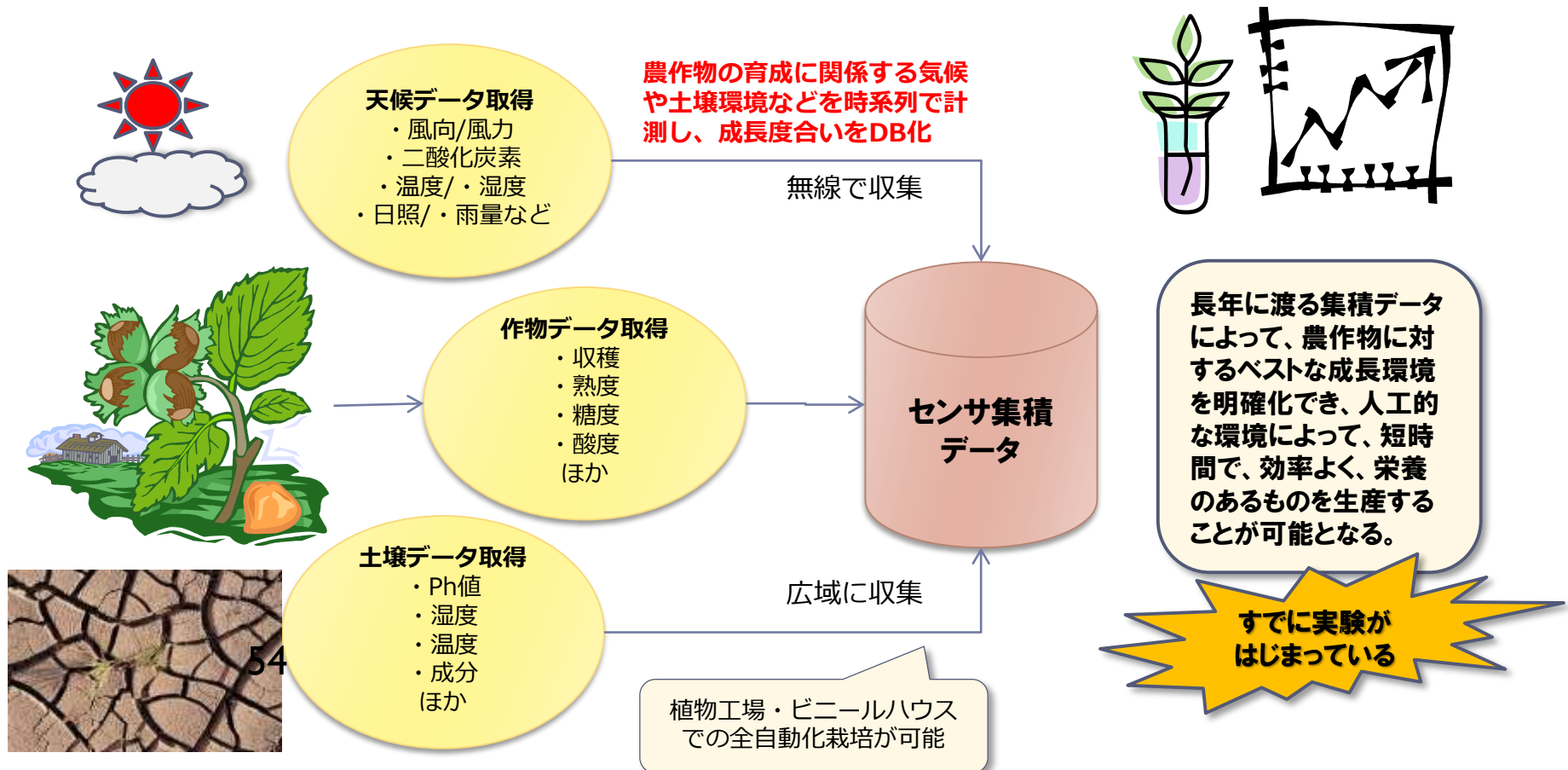
▶ 活用能力

- ▶ 即活用するものからデータ蓄積から活用するものまで

8. センサ応用事例（1）

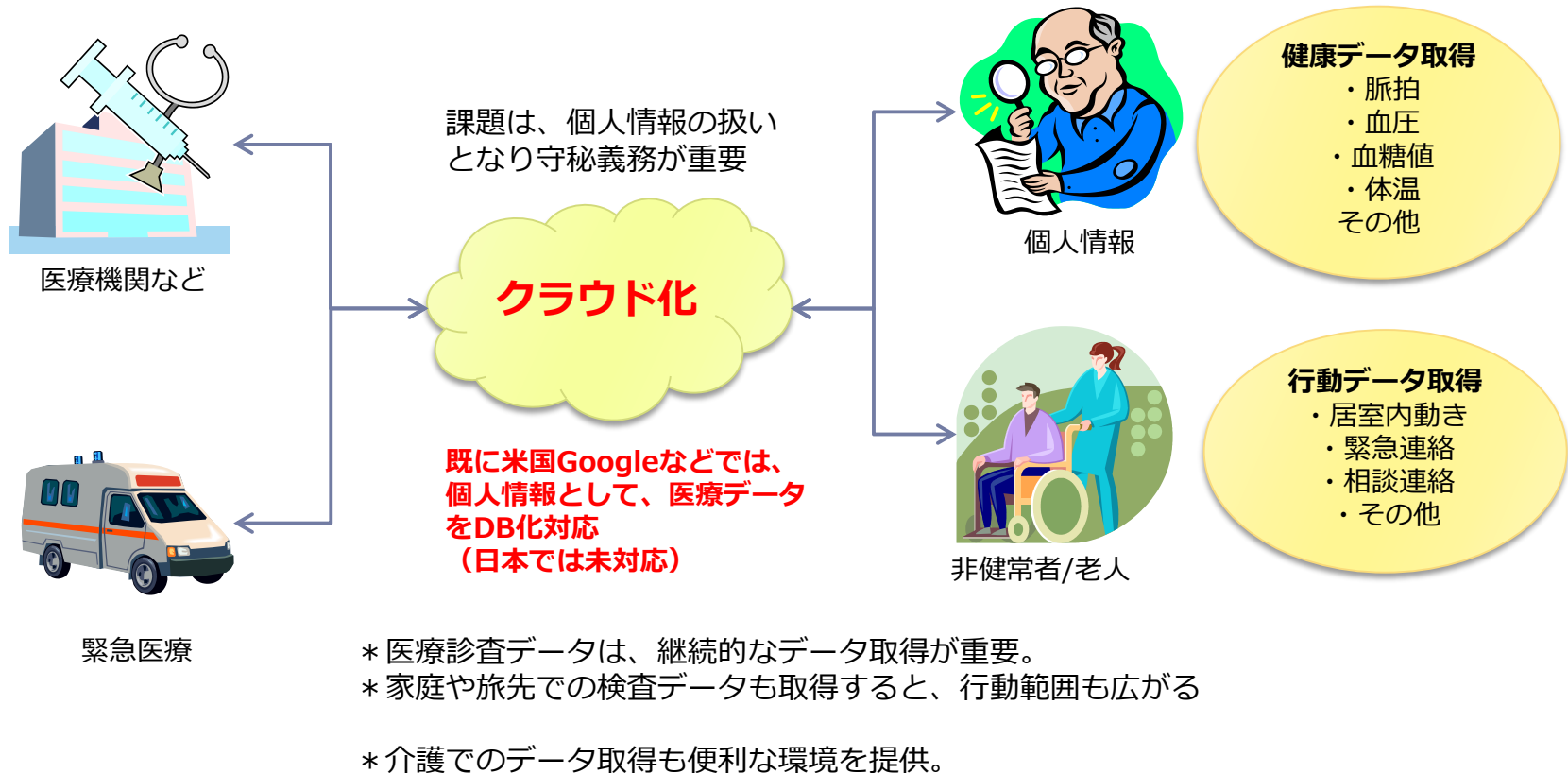
▶ 農業への応用

▶ 農業分野でのIT活用（スマートアグリ）は無限大へ



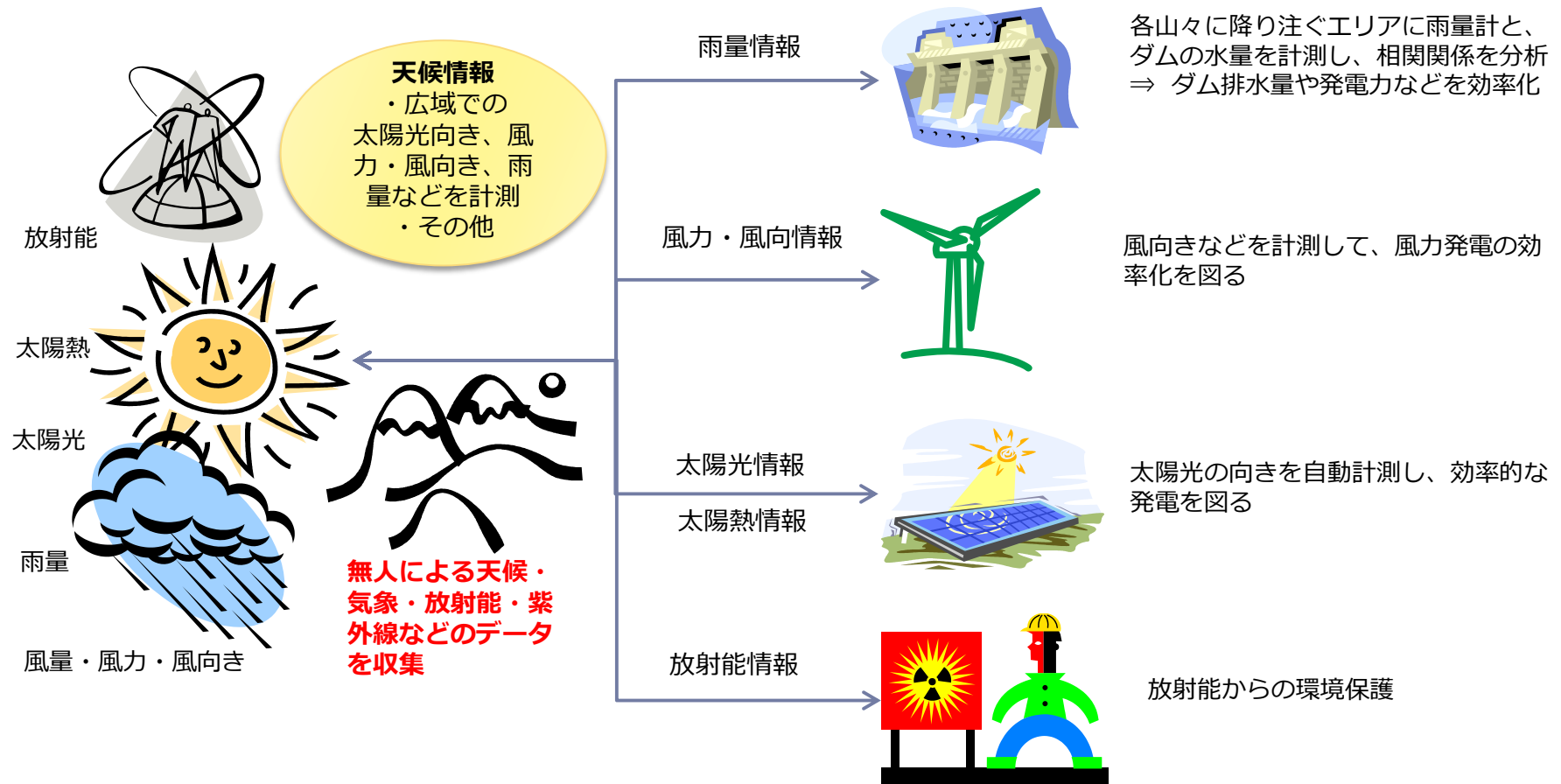
8. センサ応用事例（2）

▶ 医療・介護分野への応用



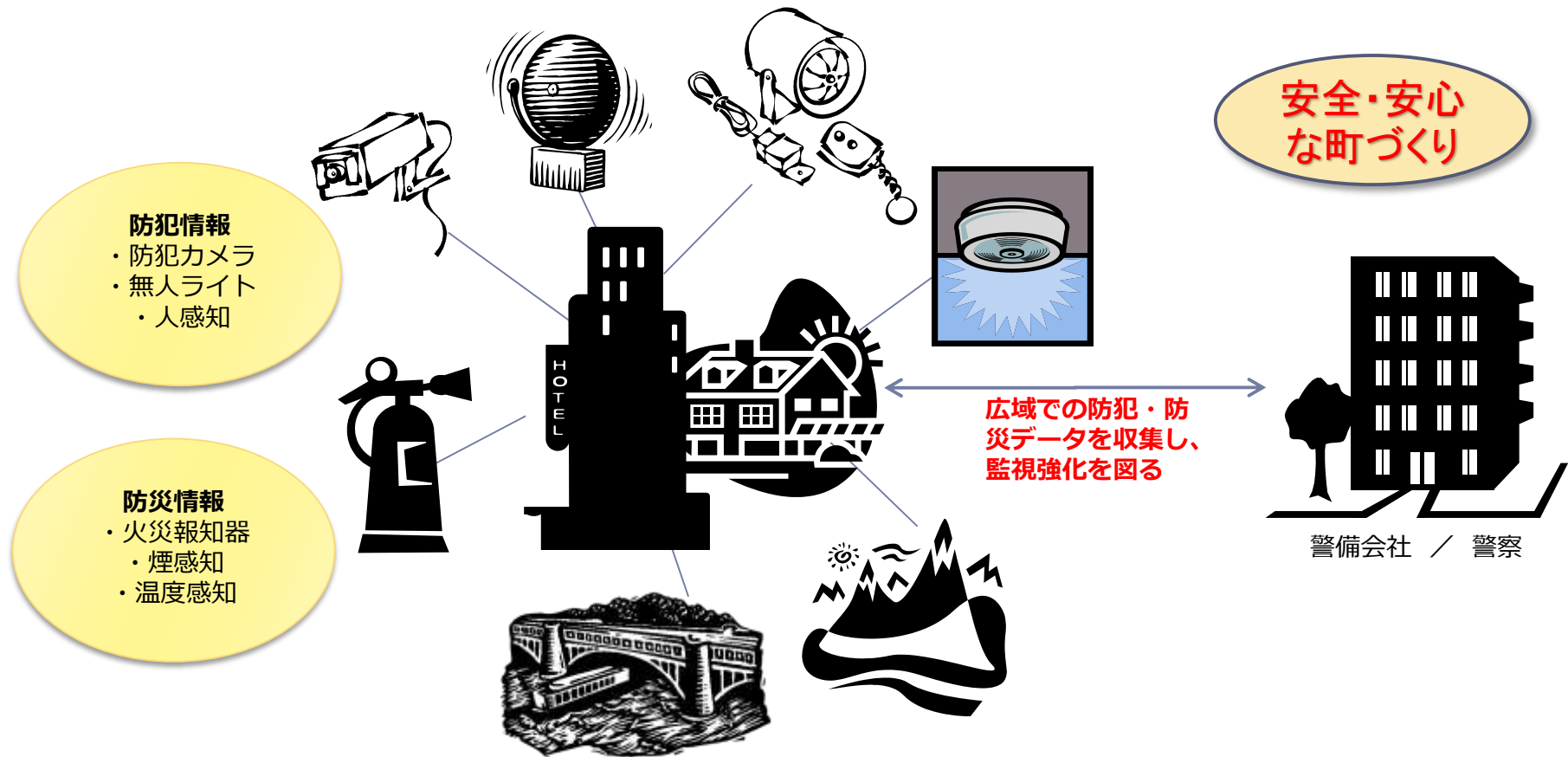
8. センサ応用事例（3）

▶ 環境・エネルギーへの応用



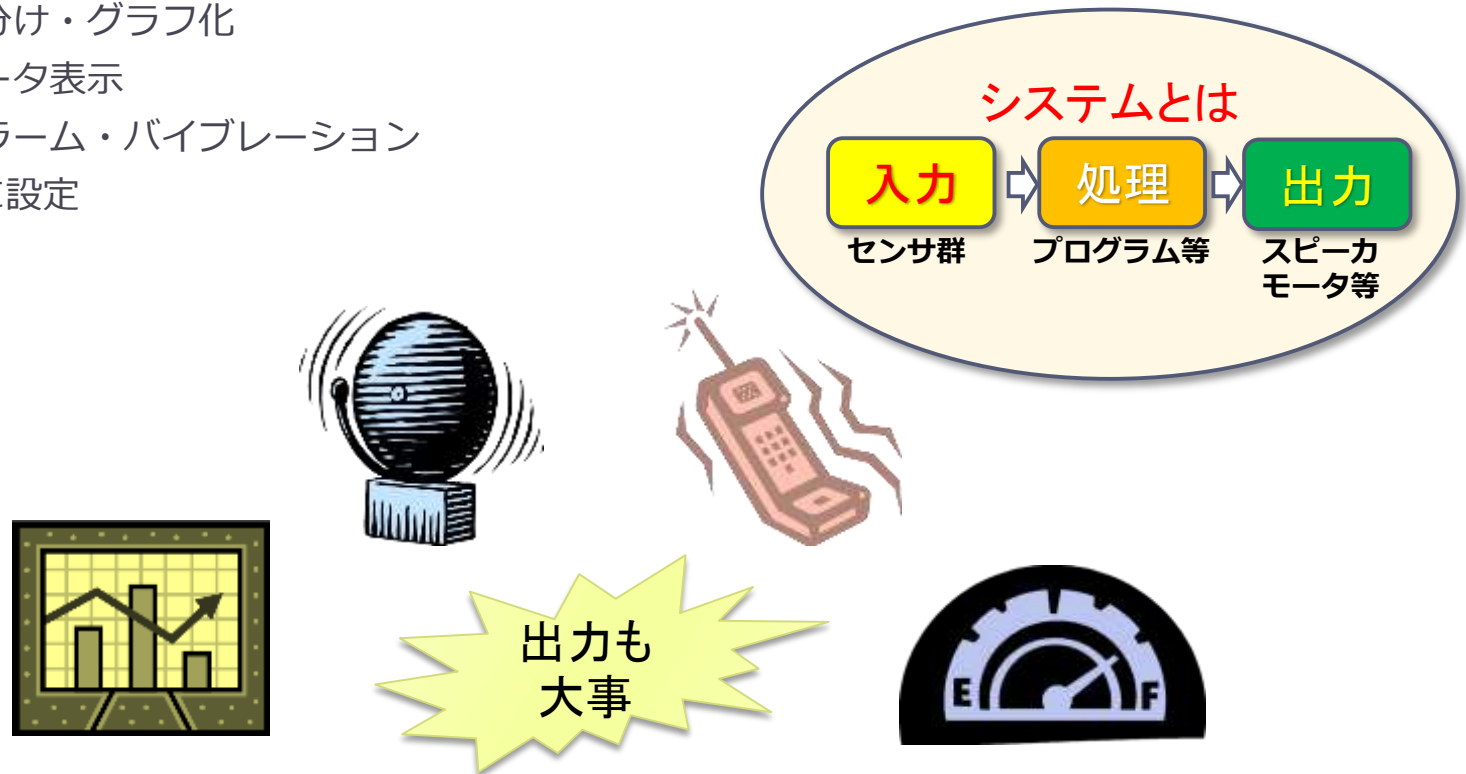
8. センサ応用事例（4）

▶ 防犯・防災への応用



9. データの可視化・認識化

- ▶ デジタルデータを分かりやすい・判断しやすい・処理しやすいように可視化・知覚化・認識化を行う
- ▶ システムでの出力を明確化
人の五感へ伝えるもの・訴えるもの
 - ▶ 可視化：色分け・グラフ化
 - ▶ 知覚化：メータ表示
 - ▶ 認識化：アラーム・バイブレーションなどを出力に設定



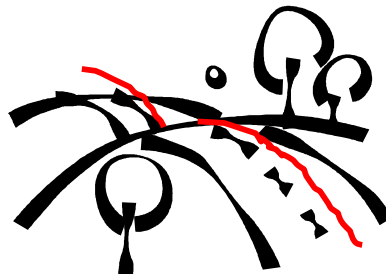
10. センサの採取データについて

データ採取の地点

- ▶ どこでデータを取得するか？
 - ▶ 単点測定・計測 <単なる地点観測など>
 - ▶ 多点計測・計測 <多量なデータ観測>
 - ▶ ライン計測 : 道路上、川沿い、ルート上など
 - ▶ エリア計測 : 多地点での対応<畑や田んぼ、室内など>
 - ▶ 3D空間計測 : ホール空間、地上から屋上まで



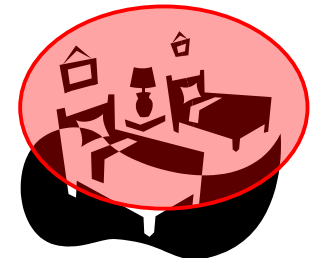
点



線分



面



空間

もくじ

1. M2M/IoTを実現にする技術と応用分野
 2. 医療・介護分野での3 GIM応用
 3. エネルギー分野での3 GIM応用
 4. 環境・エコ分野での3 GIM応用
 5. 防犯・防災分野での3 GIM応用
 6. 観光・娯楽分野での3 GIM応用
 7. 農業・漁業分野での3 GIM応用
 8. 建設・保全分野での3 GIM応用
 9. 監視・見守り分野での3 GIM応用
 10. 地方支援での3 GIM応用
- 【ブレイク】 Arduino.cc VS Arduno.org



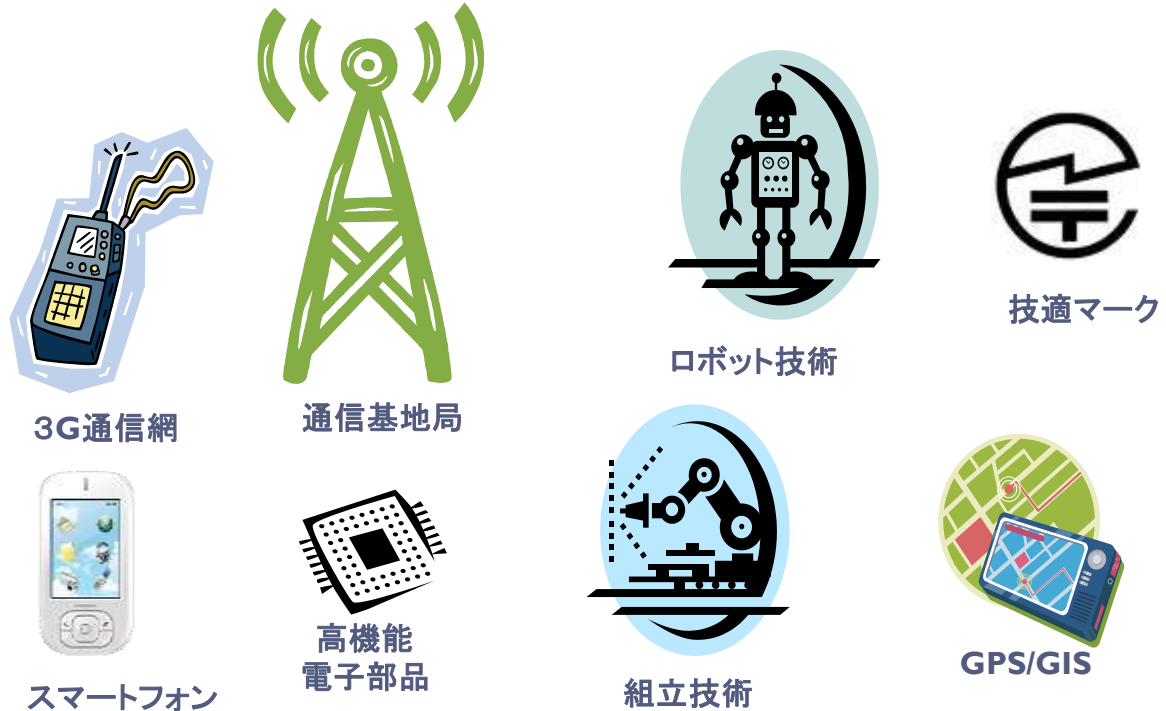
第6章 期待されるM2M/IoTビジネス

1. M2M/IoTを実現する技術と応用分野

携帯電話は、すでにひとり一台の時代となりましたが、自分の携帯電話の機能をフル活用している人たちは少ないでしょう。ある人たちにとっては、余分な機能や分かりづらいデザインと感じたり、さらにはキー操作や画面操作などで煩わしさを感じたりし、必ずしも満足いかない状況も出てきています。

3 GIMの概念では、地域や特定団体・特定企業、さらに特定コミュニティに特化した携帯電話が簡単に作れることを目指しています。その活用範囲も広く捉えて対応していくことを考えています。以下に示す応用範囲や事例は、ほんの一例でしかありません。現場で必要とされる携帯電話が誰でも簡単に作れること、これが3 GIMの考えです。

3 GIMは、1対1の双方向だけでなく、1対多での関係を持つことができ、情報を取りにいくだけでなく、収集したり分析したり、制御に使ったり、観測・観察に使ったり、ひとを誘導したり・危険を知らせたり、それはさまざまに携帯電話を特化したものに変身させることができます。特に日本の高度なセンサ技術やGPS/GIS技術と組み合わせることで可能性は無限に広がります。



センサ群(一例)



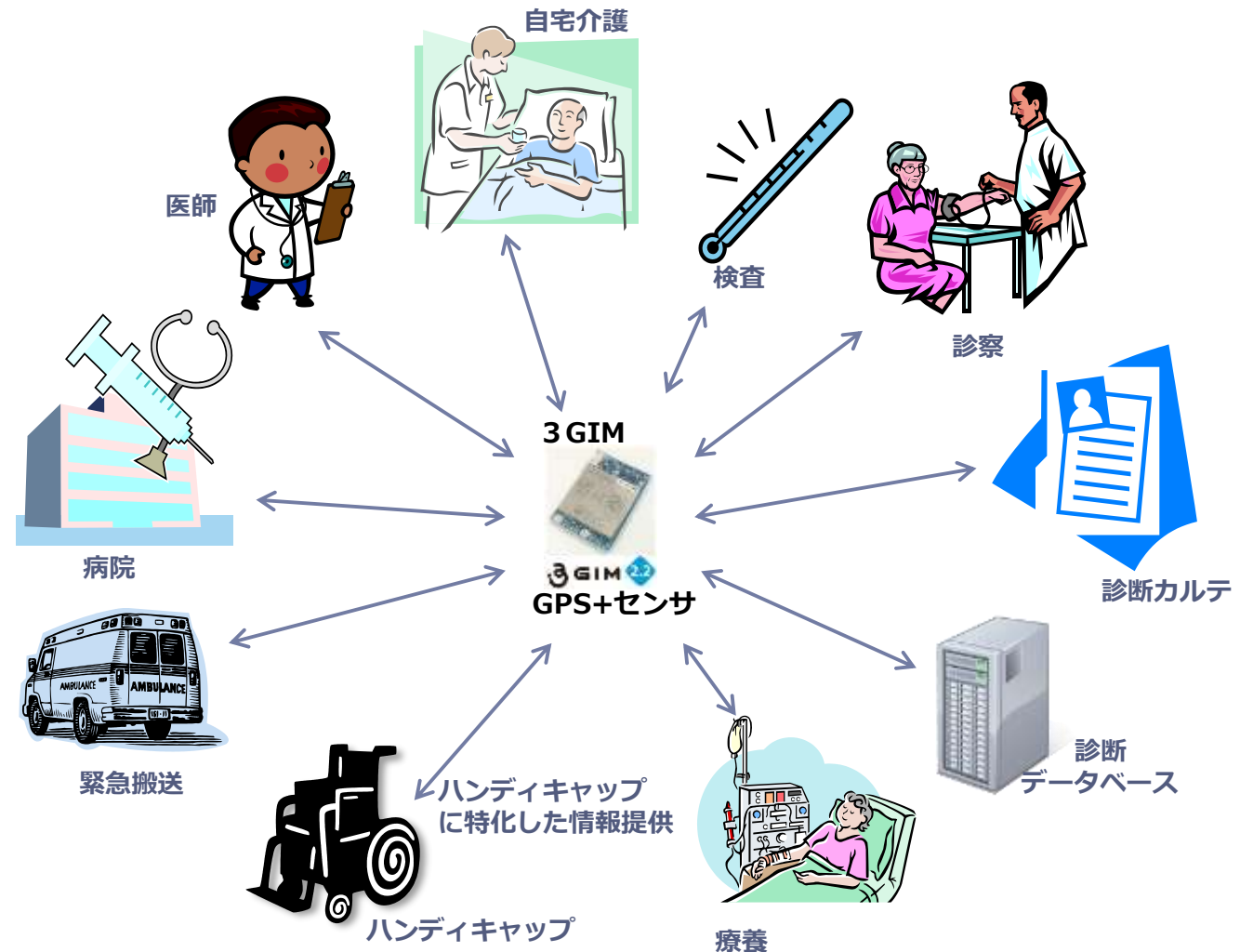
2. 医療・介護分野での3 GIM応用

一人暮らしのお年寄りが増え、医療や介護の効率化と有機的な対応が必要となります。応急連絡や定期的な検査などにモバイルを活用することで、より便利な社会にしていけることが期待されています。

自宅に居ながら検査した検診データをモバイルを使ってデータベースに送ることで、医師が病院内でそのデータをみて判断することが可能となります。

また軽病な患者では往診することなく、遠隔での対応に切り替え、緊急時に専用モバイルの簡単な操作で緊急連絡を可能とします。

遠隔操作での連絡網において、安心モバイルとして、医師と直結することも可能となります。



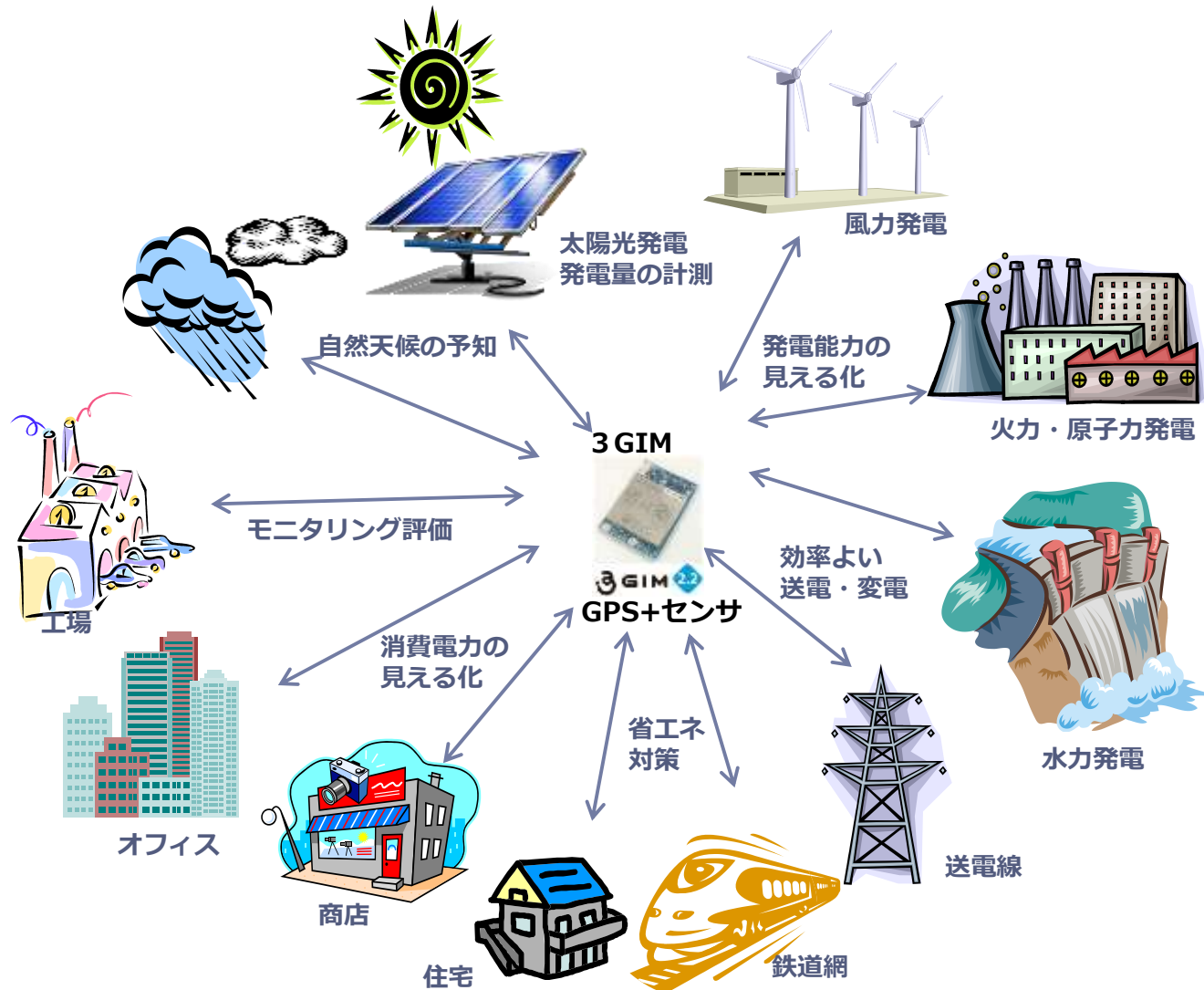
3. エネルギー分野での3 GIM応用

資源の乏しい日本でのエネルギー問題はさまざまなところで話題となってきました。可動式や指向性による自然エネルギーの効率よい吸収・収集において、広範囲で緻密な気象データをモバイルで収集し分析し、応用することが可能となります。

3.11の東日本大震災後の計画的な消費電力需要は重要となり、さまざまなところでのモニタリングが必要となります。

また太陽光発電を中心とした地域発電や企業内発電も盛んとなり、今後の自然天候の予知やモニタリング評価を、長期に渡って実施することで、より効率の良いエネルギー循環を構築できていくものと考えます。

もちろん発電量や消費電力のモニタリングも重要で、3 GIMによるワイヤレスでの計測装置機器が敷設でき、LANの敷設の問題や社内セキュリティの問題が無関係となります。



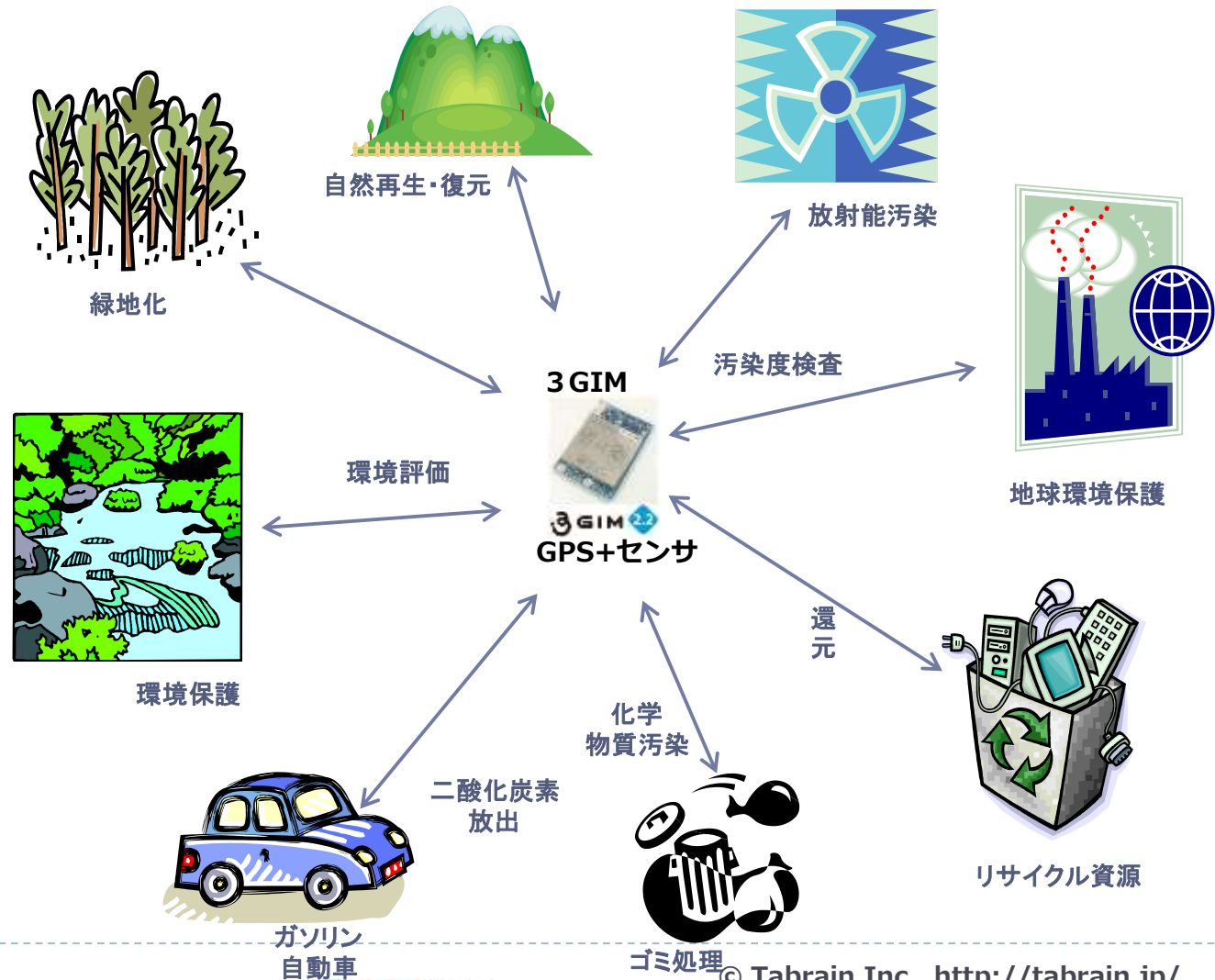
4. 環境・エコ分野での3 GIM応用

地球温暖化や生態系の急激な変化に対し、地球をやさしく守る活動が必要になってきました。モバイル技術を使うことで、幅広いエリアにおいて、無人での観測や保護などを行うことが可能となります。

環境評価・汚染度評価などのモニタリングは、さまざまなところで必要となってきます。

二酸化炭素だけにとどまらず、化学物質や放射能など目に見えないもので常時変化する地球上の土壌や河川や地下水・海水、さらに空気中のモニタリングは、幅広いエリアで必要となります。

場合によっては定点であったり、場合によっては移動点でのモニタリングが長年に渡って必要になってきました。このビッグデータによるセンサ値を有効収集するにも3 GIMは有効に利用することができます。



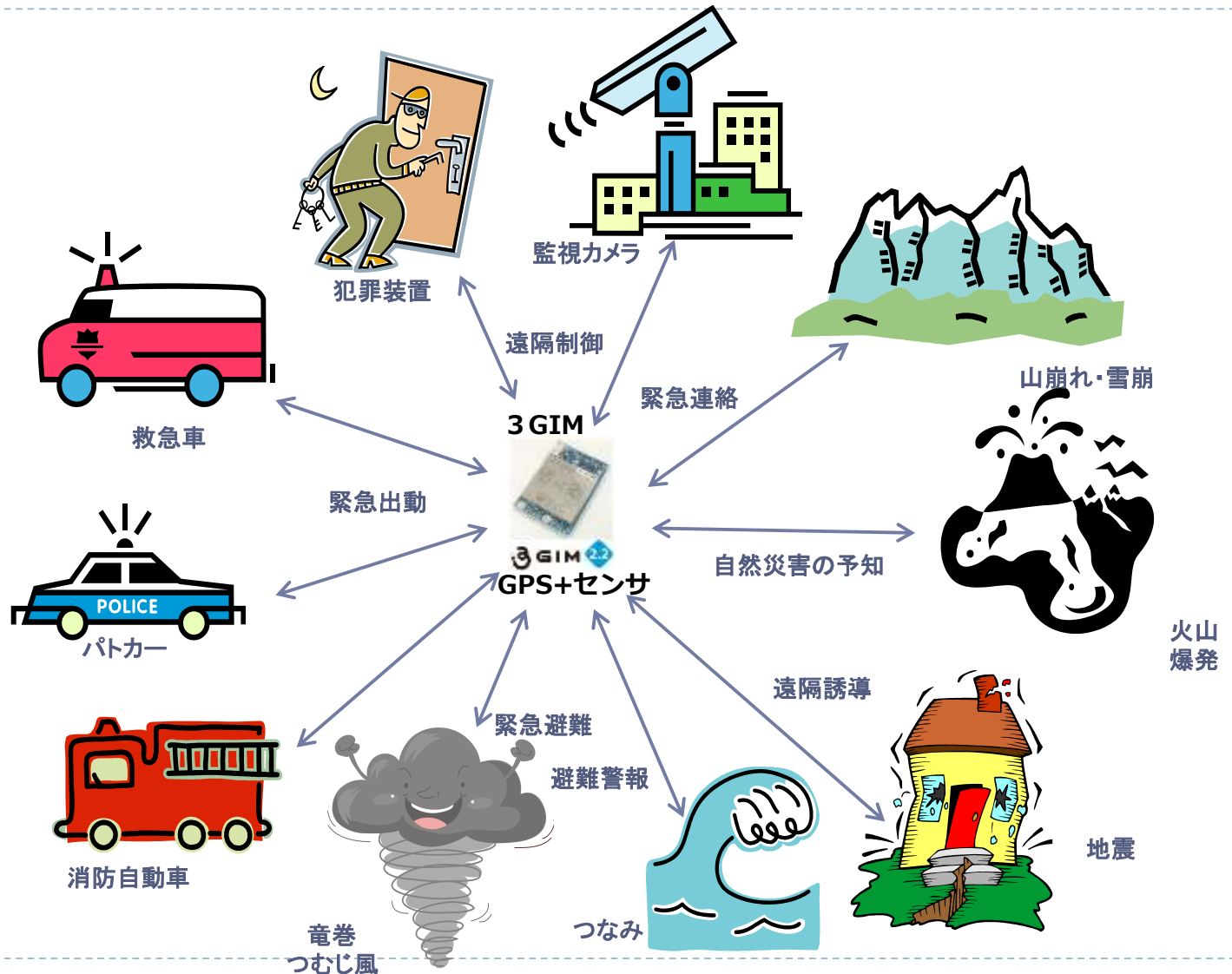
5. 防犯・防災分野での3GIM応用

古き良き日本人のコミュニティが崩れつつあり、隣人との付き合いも希薄化し、防犯の意識が高まりました。その対策として、防犯装置や監視カメラの設置が増え、その緊急連絡や遠隔制御がモバイルによって可能となりました。

また、東日本大震災によって新たな自然災害での防災の意識も高まってきました。より緻密な避難通報や避難経路の誘導など、地域に特化したモバイル活用は必須となってくるでしょう。

これら防犯・防災に特化したモバイル機器は、いろいろと期待されていて、GPS/GISの機能を活用することで、誘導避難だけでなく、安否確認などにも役立てることが可能となります。

この防犯・防災分野での3GIMの持つ双方向通信やGPS機能などを利用し、活用する可能性も無限に広がります。



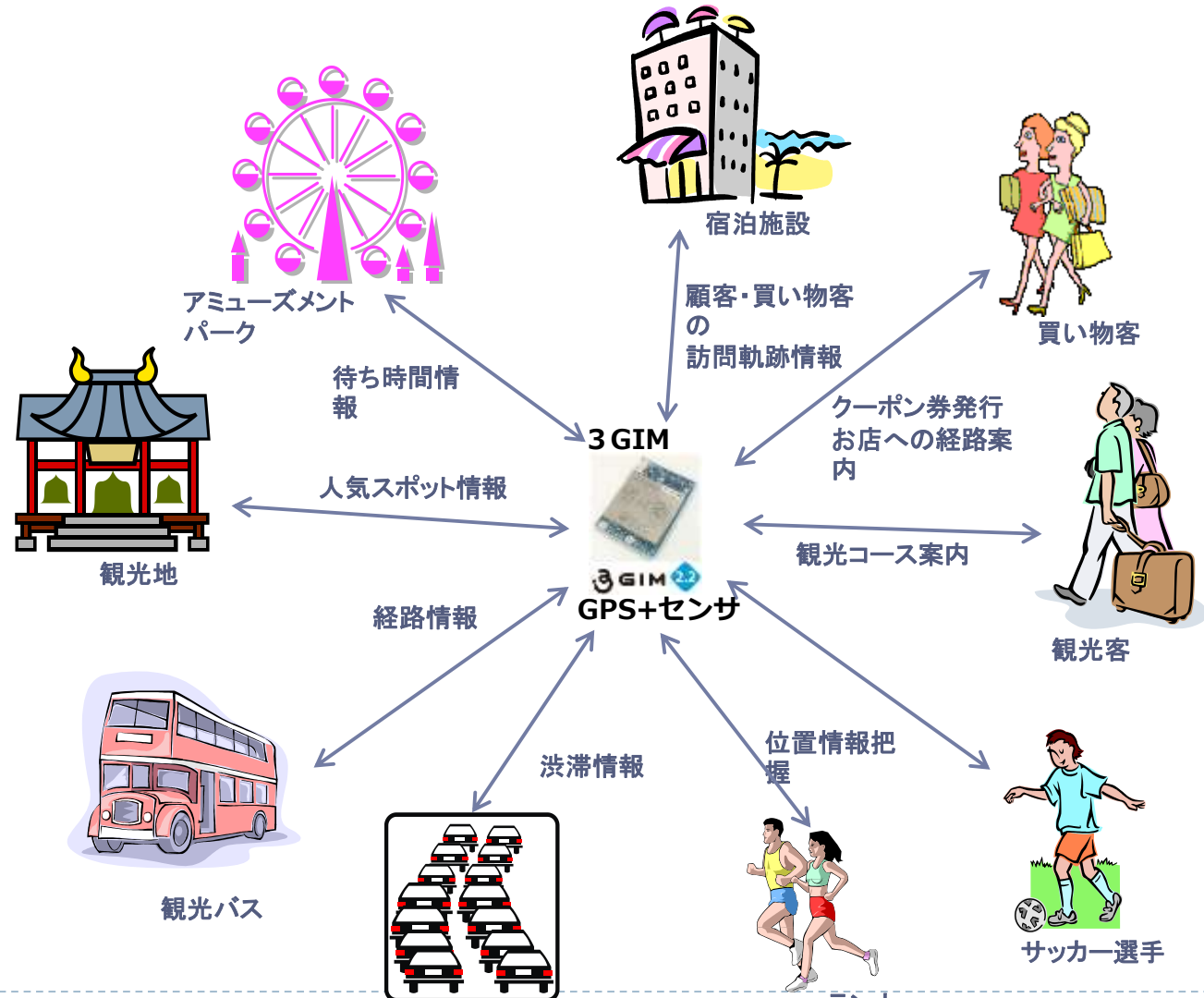
6. 観光・娯楽分野での3 GIM応用

観光地やアミューズメントパークでの付加価値のあるスポット情報や混雑情報などを独自モバイルで提供し、GPS機能を使って、顧客の誘導や軌跡分析などを行う機能などを提供することができます。さらに外国人へのきめ細かなサービス向上も可能となります。

観光バスやレンタカーでの地域に根ざしたサービス向上や、外国人向けの言語に対応したサービス向上において、デポジット式モバイルの貸し出しなどによる付加価値を持たせることができます。

また、観光客や買い物客が訪ねたお店や観光地の軌跡を分析するなど、新たな観光地発掘へと繋げることもできるようになります。

その他、スポーツ選手に付帯させたモバイルで、正確で瞬時の位置情報を取ることで、新たな楽しみ方も生まれます。



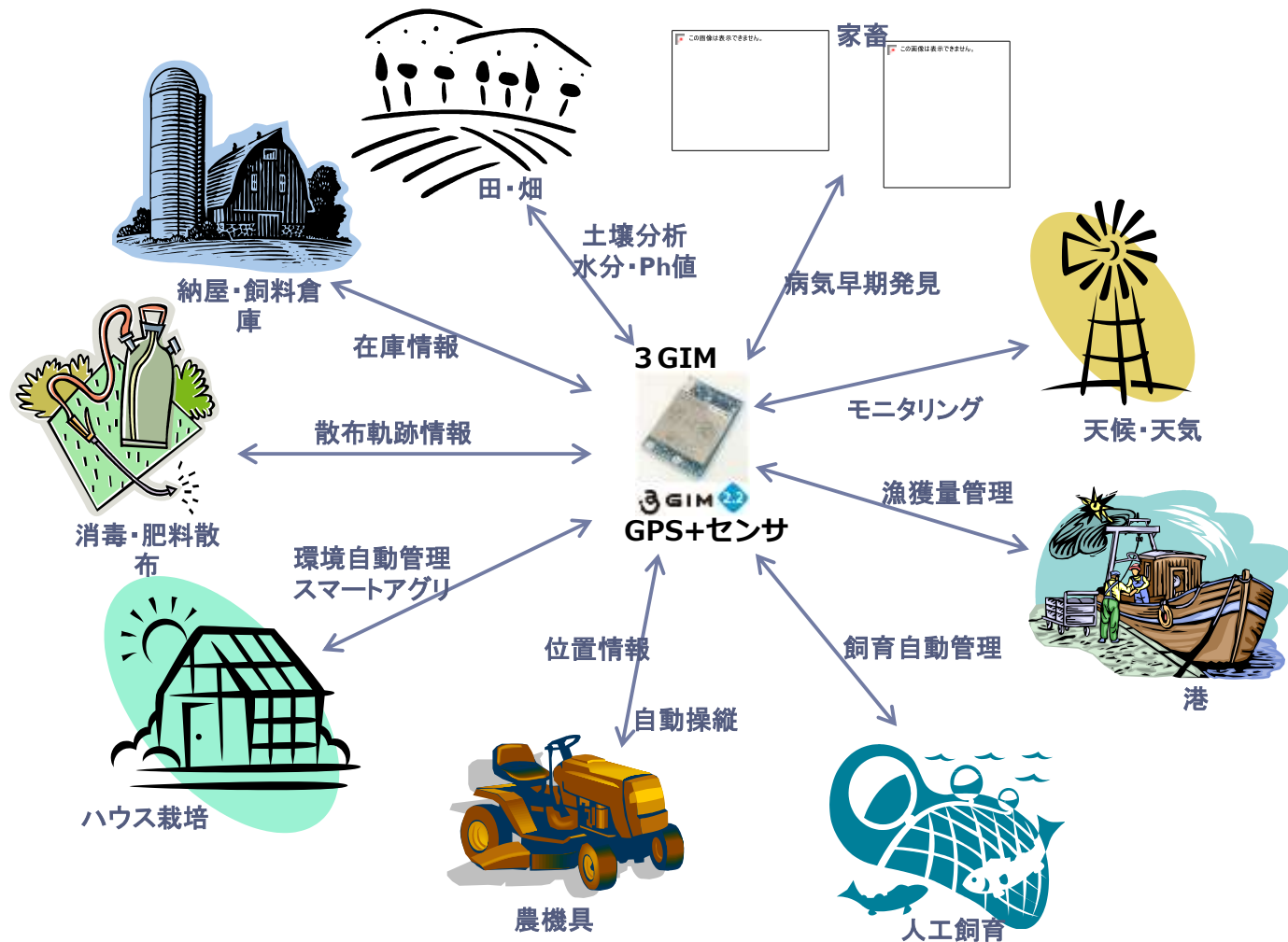
7. 農業・漁業分野での3 GIM応用

情報通信技術（ICT）の活用が遅れている農業や漁業分野においては、広いエリアでの監視や観測に伴い、モバイルの活用がいろいろと出てくるといわれています。

多くのセンサ技術との連携で、長期にわたるデータ収集・モニタリングも可能となり、新技術化を図った農業・漁業が開拓できるのではないのでしょうか。

最近の牛や豚などの口蹄疫や鶏や鳥類の鳥インフルエンザは、早期発見が重要となっています。すでに牛や鶏に付けた加速度センサや熱センサでの研究も進んでいて、多くの農家でのモニタリングも必要となってきました。

今後は、特に農業のさらなるICT活用にモバイルは欠かせないものとなるでしょう。



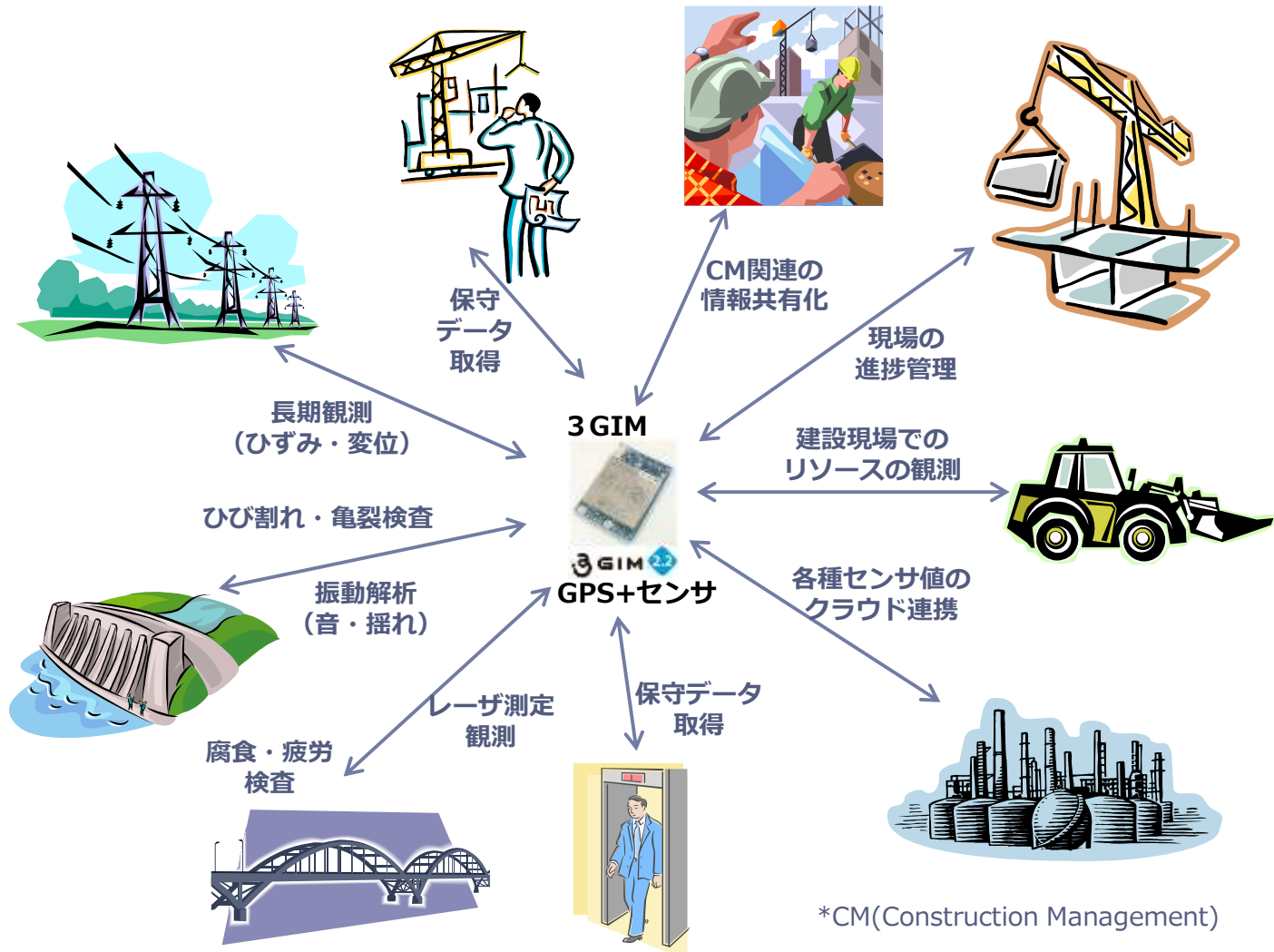
8. 建設・保全分野での3GIM応用

新築・改築・増築などでの建設現場は、一時的なものです。が、新たなLAN敷設が難しいところなどでは、3GIMを使ってのサーバ連携が強力な武器となります。

現場でのさまざまな稼動する機器や人的リソース、さらに資材などを有効活用するための現場でのワイヤレスを活用して収集するデータと、本社などがかかえるサーバとの連携も簡単に対応することができます。

またこれからの建設業界での保全（保守・点検）は重要な課題となってきました。いちいち人によるものだけでなく、一部センサと無線（ワイヤレス）を使った監視システムなどは、これから大きなニーズがあるものと考えます。

特に、橋梁やエレベータなどの保全は、遠隔での対応が3GIMによって可能となってきます。



*CM(Construction Management)

9. 監視・見守りでの3GIM応用

人の動きや動物の動きを監視したり、さらには植物などの成長を見守りする場面が多く出てきました。

子供らの安全な通学などを見守り父兄への連絡を行うシステムは、徐々にニーズが高まってきています。同様に社会問題となっている認知症の方々の動きを見守るものもあります。

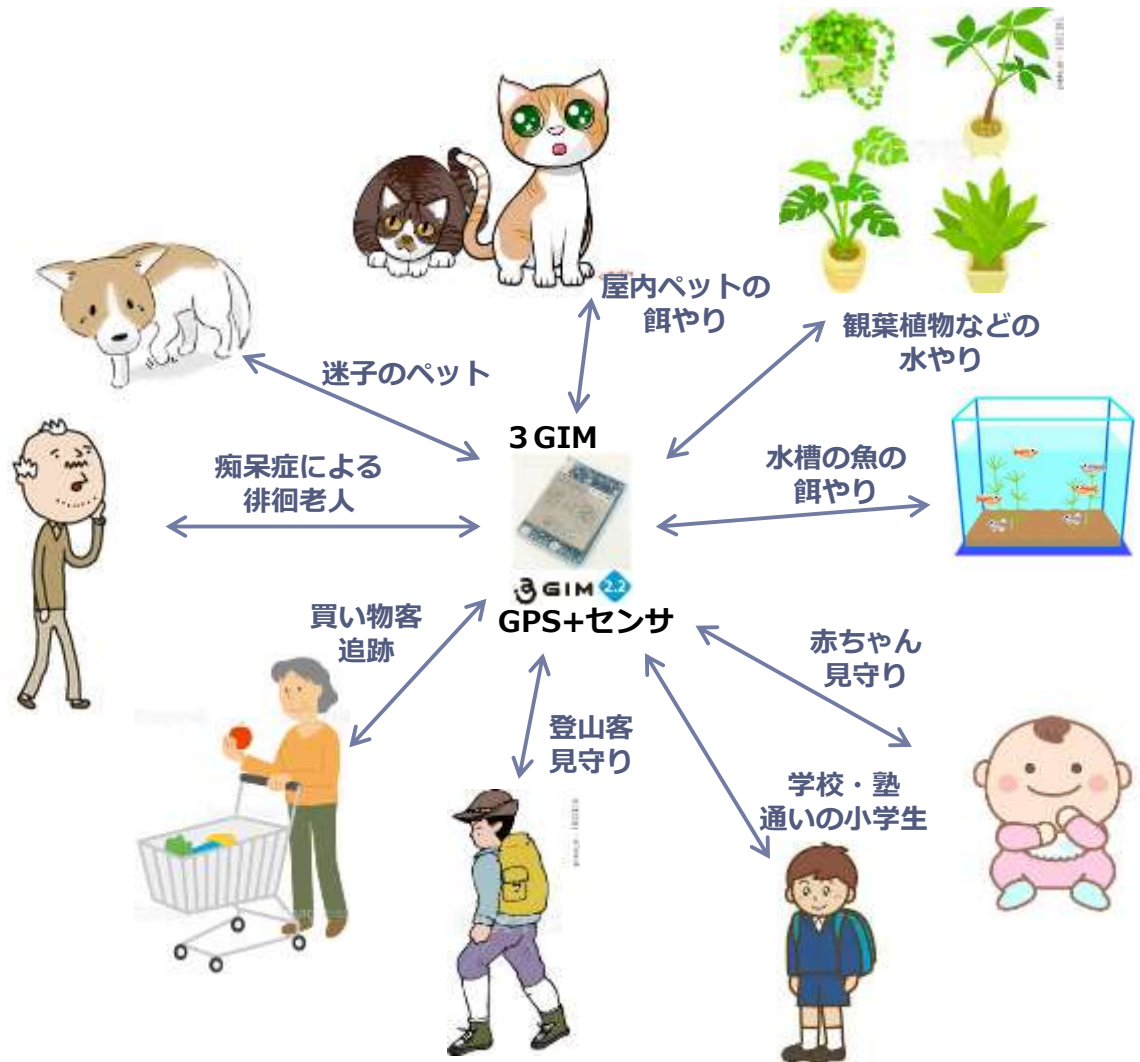
スーパーなどでの買い物客の動きによって、動線を感じたり、買い物パターンを調査したりするものも出てきています。

その他、長時間での外出によるペットの監視や観葉植物の監視、または赤ちゃんなどの監視は、ニーズが高まってきています。

動きや成長を把握することで、安心・安全な状況を把握でき、次の行動にも行かせるシステム構築ができるようになります。

現在、これらのシステムは、導入段階にあり、今後ますます期待が高まるものとなっています。

ビックデータへと結びつくことで、多くの価値あるデータが生まれ、社会インフラとしての安全・安心できる街づくりにも繋がるものとして期待されています。

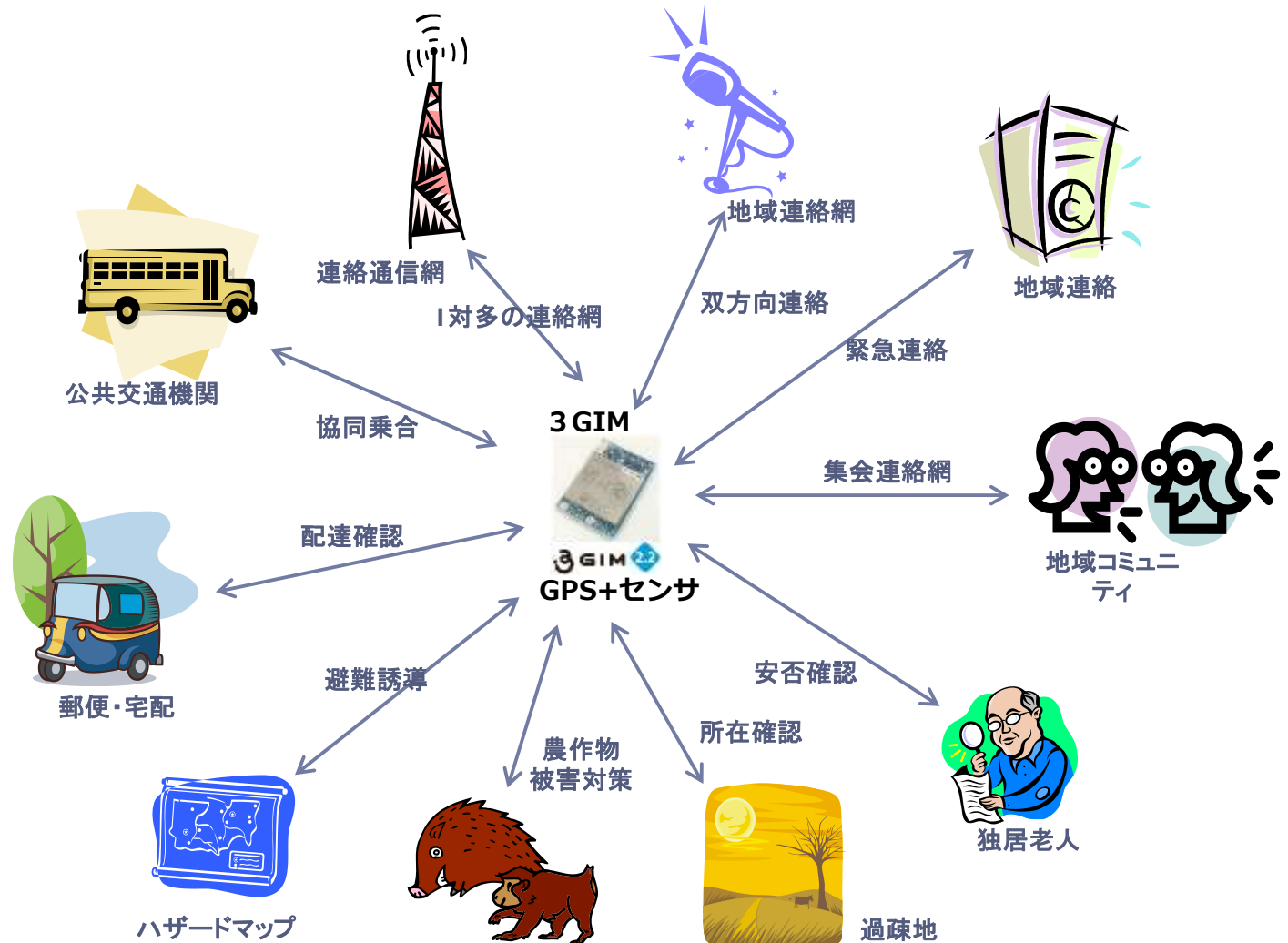


10. 地方支援での3 GIM応用

一人暮らしでお年寄りが多い過疎化が進む地方では、常時のコミュニケーションが重要となってきています。ひとり一台のモバイルにより、双方向や一対多などの連絡網や位置確認、さらに住民サービス向上での多くの価値・効果が期待されています。

地域生活で必要となる移動手段については、共通の連絡網によって、乗り合いバスを合理的に走らせたり、郵便・宅配も効率よく巡回させたりすることが可能となります。

過疎化での独居老人の安否や所在確認も、GPS機能や双方向の連絡によって常時可能となり、安心した生活を送ることが可能となります。



【ブレイク】 Arduino.cc VS Arduino.org

- ▶ 2015年はじめ、マネジメントチーム（Arduino LCC : arduino.cc）から製造チーム（Arduino Srl : arduino.org）が分裂、その後マネジメントチームが製造チームを訴訟。
- ▶ Arduino LCCは、USAのみで商標「Arduino」が利用可能、USA以外はArduino Srlが商標「Arduinio」を独占
- ▶ 互いに作る製品とIDEが分裂：
 - ▶ Arduion LCCは、Genuino101（インテル製）などを製品化し、IDEは 1.6.xxで展開。
 - ▶ Arduino Srlは、Arduino Zeroなどを製品化し、IDEは、1.7.xxで展開。

（IDEは、Arduino LCC側が高機能で進化、日本ではSrlの製品がArduino として販売）

- ▶ 2016年秋、互いが歩み寄り、再度連携ビジネスに合意し、再スタート
- ▶ 2016年12月20日 IDEが統一（共有化）され、バージョンが 1.8.0で再スタート
- ▶ 2017年11月時点の最新バージョンは、1.8.5 が Arduino.ccから提供
- ▶ 2018年1月時点 製造づくりに遅れ発生（製造チームの弱体化で製品の入手が難しくなる）

もくじ

- 第1章 Arduinoの基礎
- 第2章 Arduino IDEの準備
- 第3章 ソフトウェア編（文法）
- 第4章 ソフトウェア編（基礎）
- 第5章 Arduinoの基礎演習
- 第6章 Arduinoの拡張演習
- 第7章 Genuino101専用機能



Arduino互換機（Genuino101）

第Ⅱ編 Arduino編

ArduinoおよびIDEについて基本的な使い方を学ぶ。基本的なC言語から、ハードウェアの扱いまで、短時間で使い方を覚える。

もくじ

1. Arduinoによる電子工作とは
 2. Arduinoを効率よく学ぶ
 3. Arduinoによるシステムと制御
 4. インターフェ이스の基礎
 5. Arduino UNO インターフェイス
 6. Genuino101 インターフェイス
 7. Arduino M0 Pro インターフェイス
 8. 電子部品の取扱いについて
- 【ブレイク】Arduino関連の参考本 ②（洋書）



第1章 Arduinoの基礎

1. Arduinoによる電子工作とは

- ▶ 電子部品を使った組み合わせ・制御が学べる
 - ▶ 電子部品の組み合わせによって、抵抗・コンデンサ・センサなどを機能できるように配線（ハードウェアを学ぶ）
 - ▶ 電子部品の制御をプログラムによって、思った機能で働くようにする（ソフトウェアを学ぶ）
 - ▶ Arduinoでは、センサーやLEDなどの電子部品を簡単に、短時間で利用できるので、電子工作を楽しく学ぶことができる

Arduino上の電子工作で、ハードウェアとソフトウェアを学ぶことができる。

※但し、ロボットや3Dプリンタなどを作る場合には、機械を組み合わせた機構の知識が必要。

2. Arduinoを効率良く学ぶ

○ 初回のみ

● 段階的に拡張

①

Arduinoの
入出力ポートを知る ○

②

アナログ/デジタル
入出力を知る ○

③

IDEの
使い方を知る ○

④

電子部品の
特性を知る ●

⑤

Arduino言語を知る ●
関数・変数・式など

⑥

サンプル・スケッチ ●
を動かしてみる

⑦

スケッチ ●
を拡張してみる

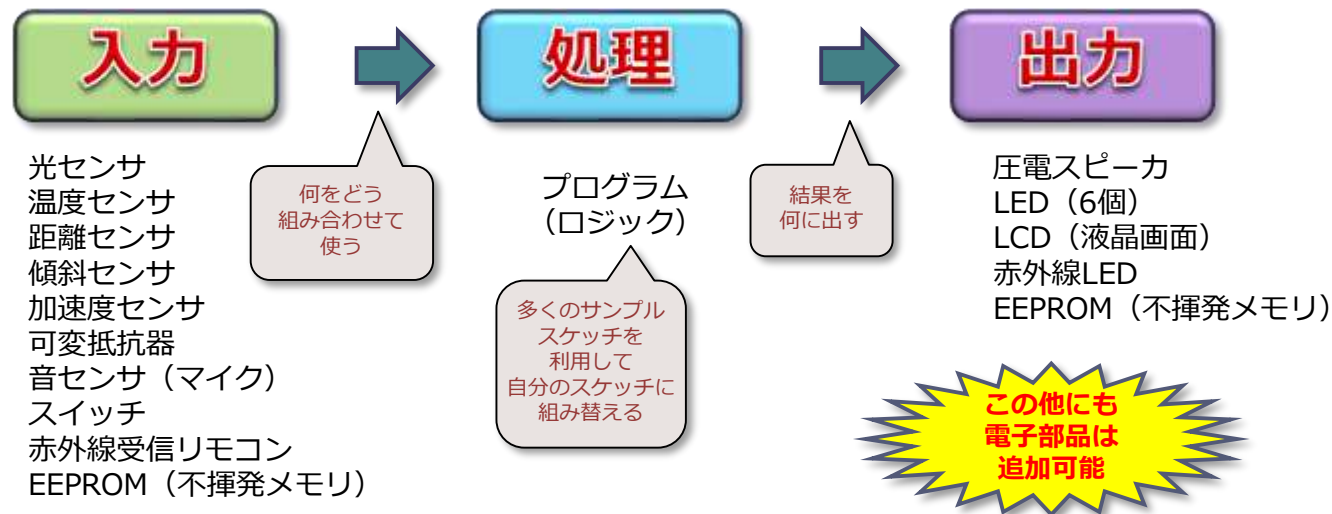
⑧

テクニックを
理解する

3. Arduinoによるシステムと制御（1）

- ▶ システムとは、一般に、組織・組立て・体系・系統を意味する
- ▶ コンピュータのシステムは3つによって構成

「システム = 入力 + 処理 + 出力」

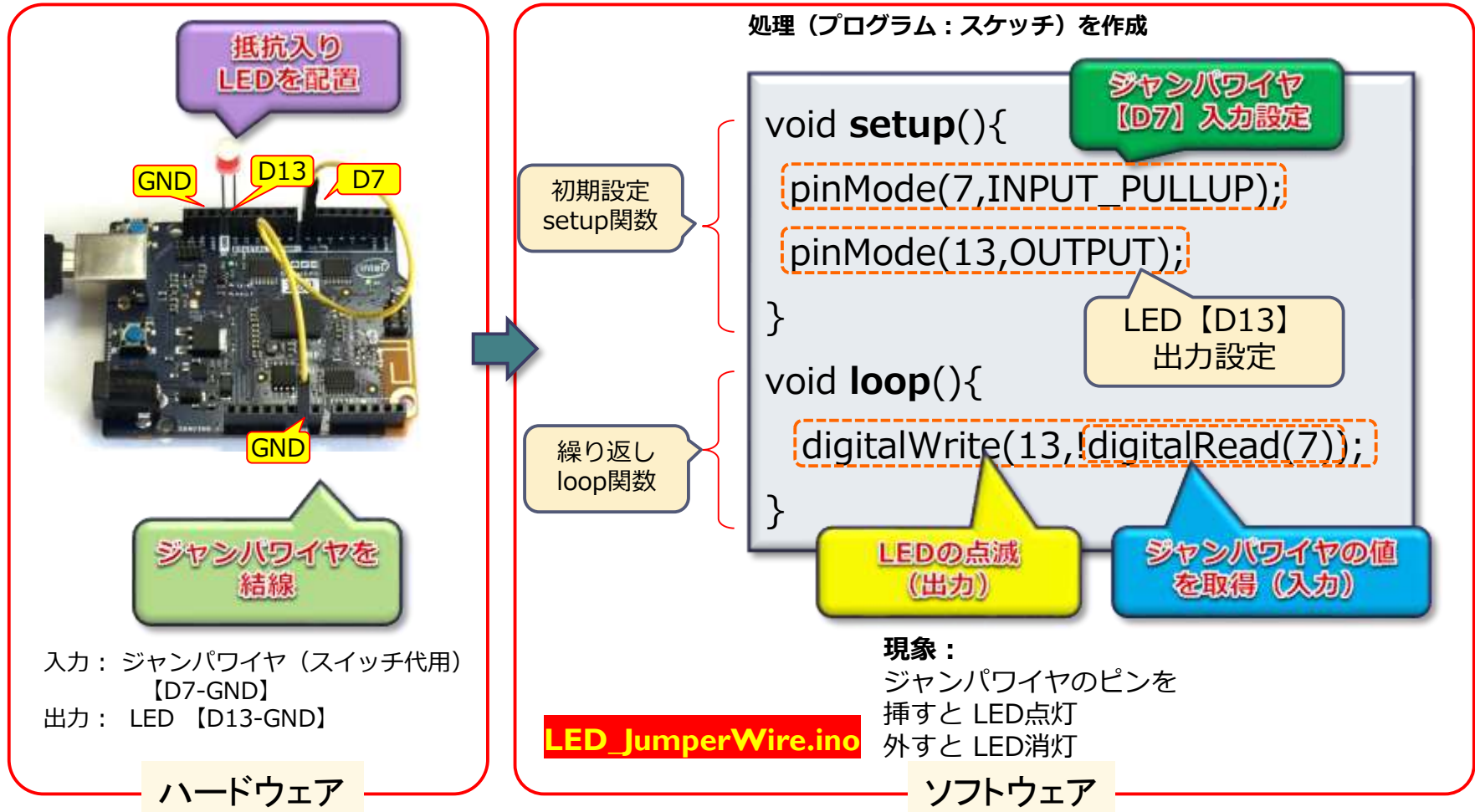


- ▶ 手順は、先に「入力」と「出力」を決め、「処理」は後

3. Arduinoによるシステムと制御（2）

※ジャンパワイヤを D7とGNDに接続
LEDをD13とGNDに接続（Arduino上LED代用可）

【補足注意】ここで使っているLEDは、抵抗入りのLEDを使っています。一般に市販されている抵抗なしのLEDを直接ポートに差し込むことは、LEDが壊れる原因となります。



3. Arduinoによるシステムと制御（3）

- ▶ システム（ソフトウェア）は簡素化（シンプル）すること
 - ▶ 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)

某Arduino関連本のサンプルスケッチ

```
unsigned long t1;
unsigned long t2;

void setup(){
  Serial.begin(9600);
  Serial.println("start");
  t1=0;
  t2=0;
}

void loop(){
  while( t2==t1 ) {
    t2 = millis()/1000;
  }
  t1 = t2;
  Serial.println(t1);
}
```

グローバル変数が
2つも存在

無駄に初期化

処理の理解が
しんどい

sample_complex.ino

見やすくしたスケッチ

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

void loop(){
  static unsigned long ts=0;
  Serial.println(ts);
  ts++;
  while(millis()/1000<ts);
}
```

2行でまとめる

ローカル変数採用

変数は1個・初期化

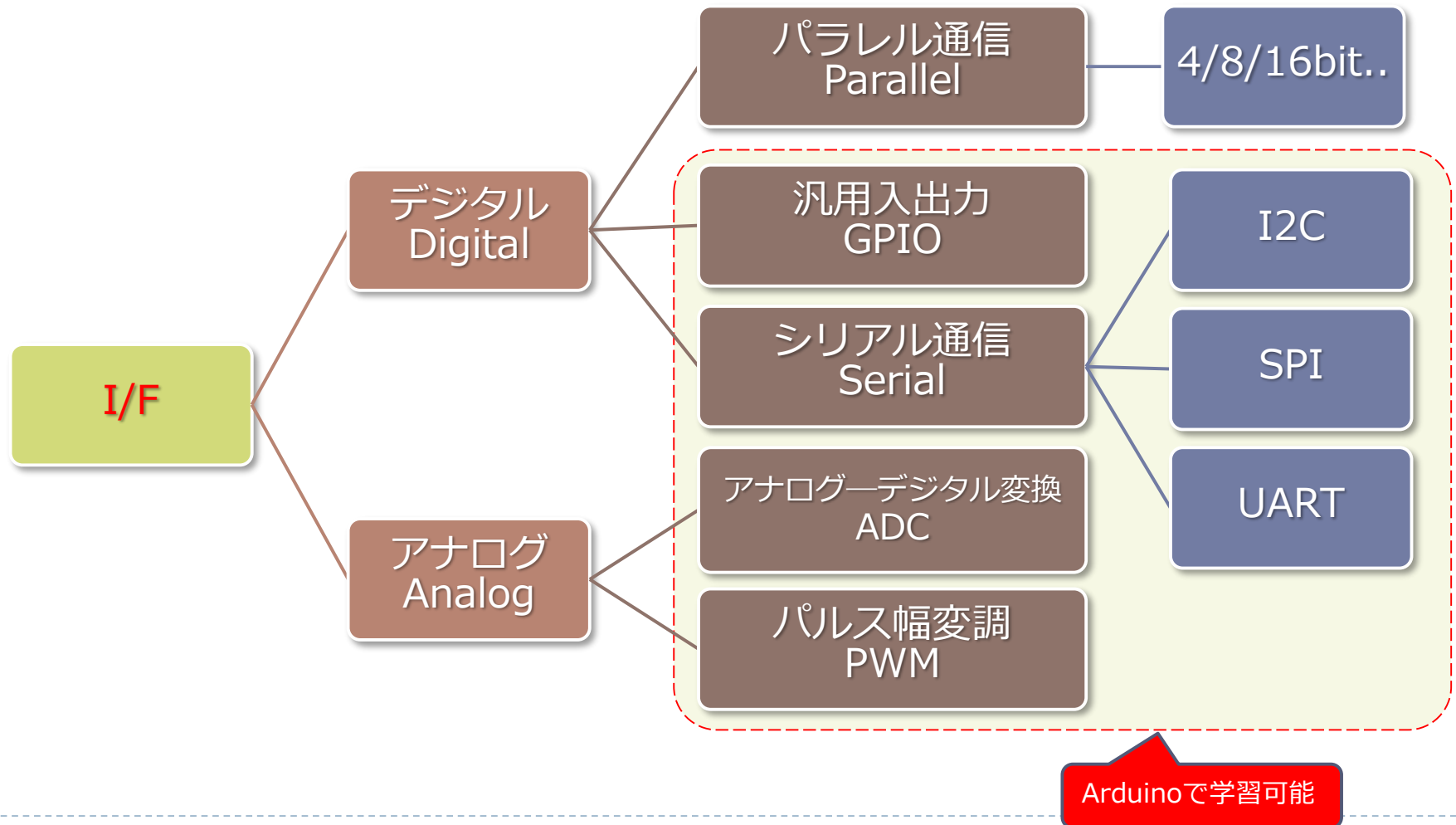
処理の理解が
分かりやすい

sample_simple.ino

- 1) 変数定義と初期化 (1回)
- 2) 処理は3行でまとめた

4. インターフェ이스の基礎（1）

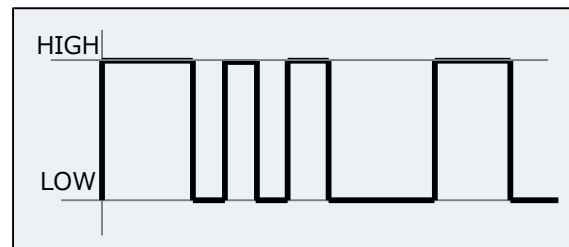
Arduinoでは下記のマイコンのインタフェースのうち、パラレル通信以外を学習することができます。



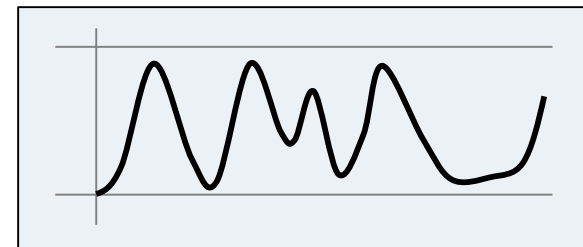
4. インターフェ이스の基礎（2）

- ▶ 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（analogReadResolution関数利用による）



デジタル



アナログ

4. インターフェ이스の基礎（3）

▶ シリアル通信／パラレル通信

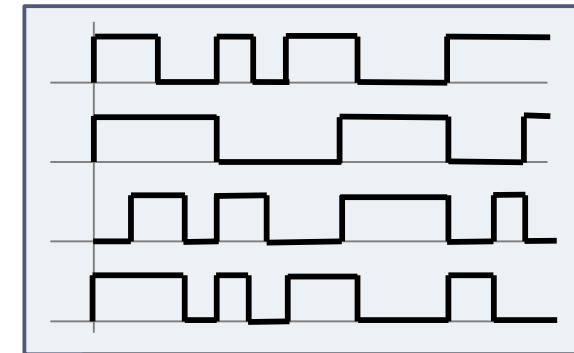
▶ シリアル通信（Serial）

- ▶ 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



デジタル

アナログ

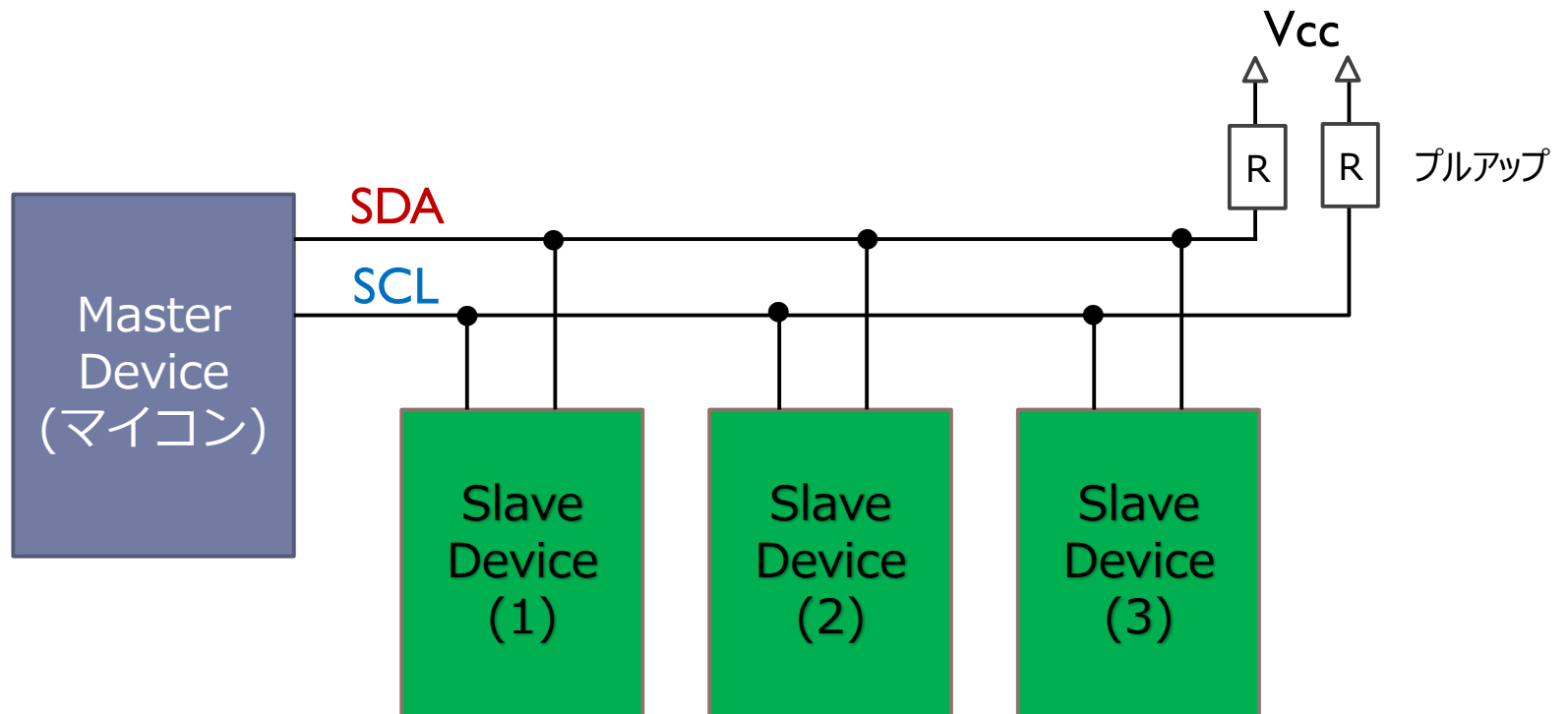
4. インターフェ이스の基礎（4）

▶ **I2C** (Inter-Integrated Circuit) とは

「アイ・スクエア・シー」とか
「アイ・ツー・シー」と呼ぶ

- ▶ 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

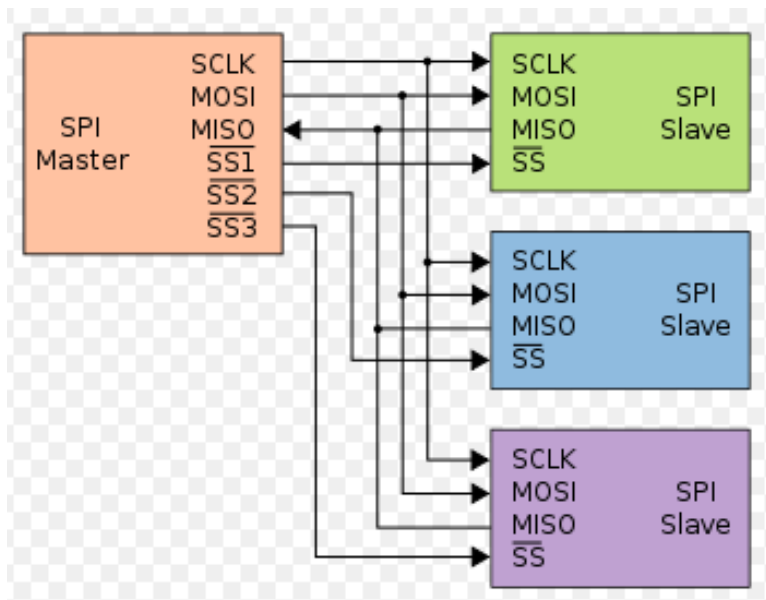
Arduino UNOの場合
A4-A5利用
(他専用ピン利用可)
IoTABシールドでは
A4-A5は未使用



4. インターフェ이스の基礎（5）

- ▶ **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)



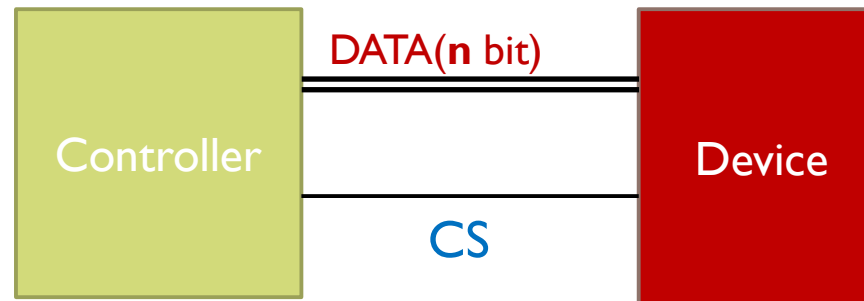
SCLK: Serial Clock
MOSI: Master-Out/Slave-In
MISO: Master-In/Slave-Out
SS: Slave Select

4. インターフェ이스の基礎（6）

▶ 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



4. インターフェ이스の基礎（7）

▶ **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

4. インターフェ이스の基礎（8）

▶ 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

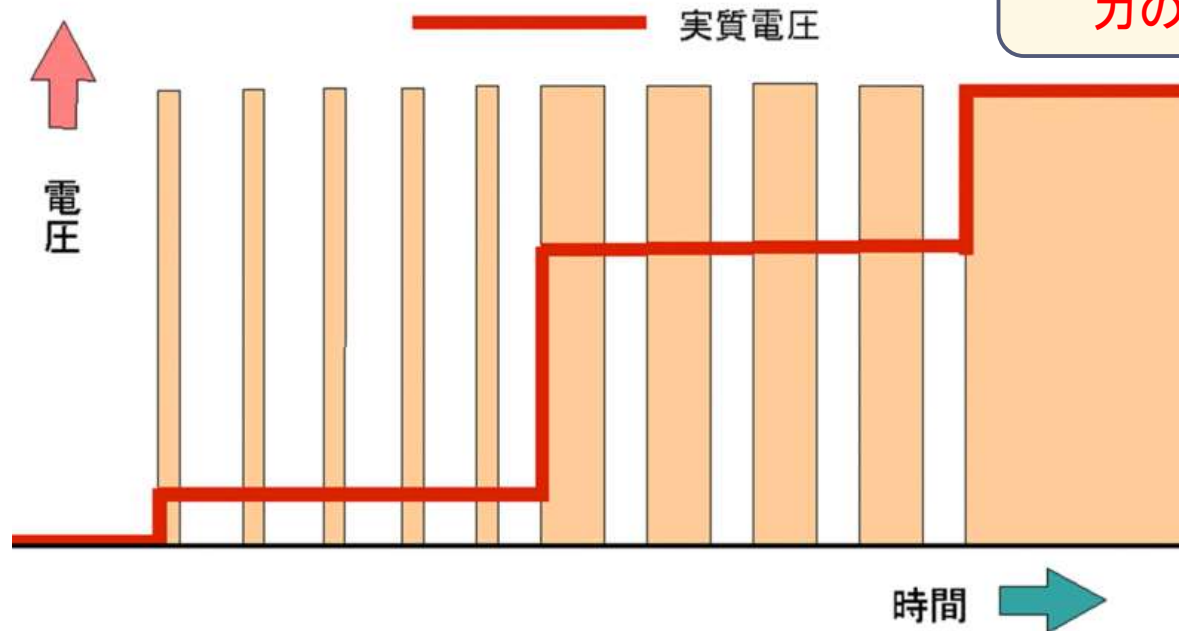
4. インターフェイスの基礎（9）

- ▶ **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）

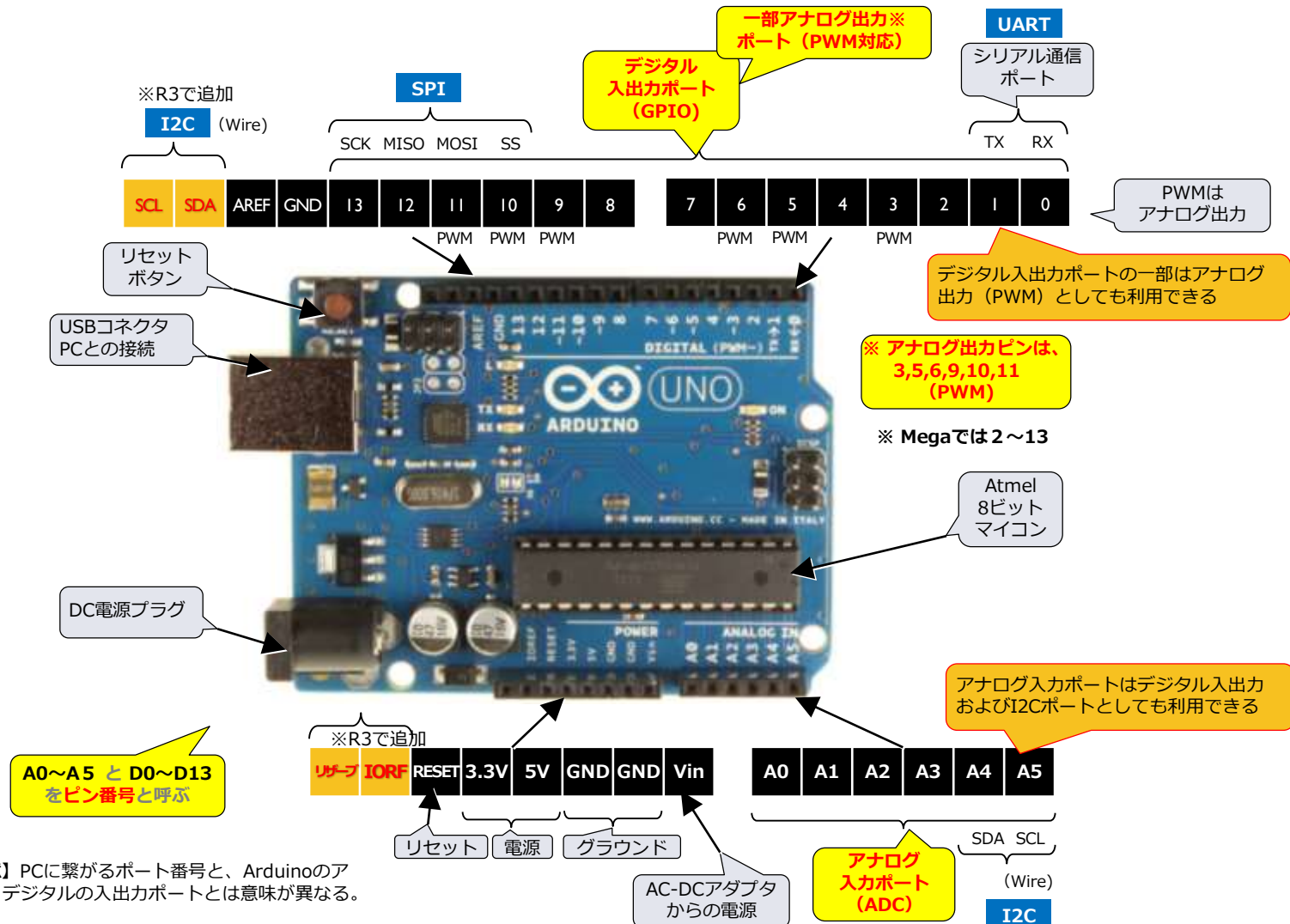
ArduinoUNOの
※ アナログ出力ピンは
D3,5,6,9,10,11

【注意】
Geuino101のPWMは、
D3,5,6,9

PWMは、アナログ出力のことを指す



5. Arduino UNO インタフェース (1)



5. Arduino UNO インタフェース（2）

▶ 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

▶ 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

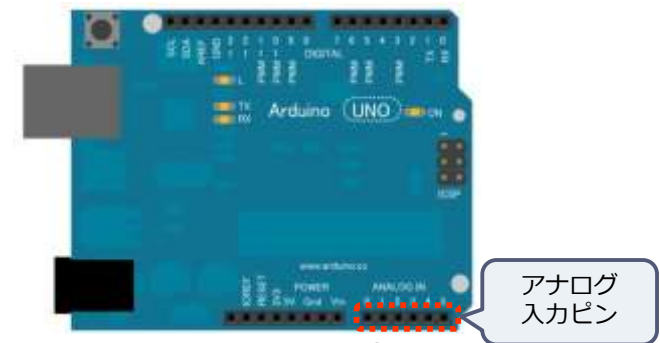
- ▶ `void analogWrite(int pin, int value)`

- pin 3,5,6,9,10,11
 - value 0~255 (0は0V、255は 5V)

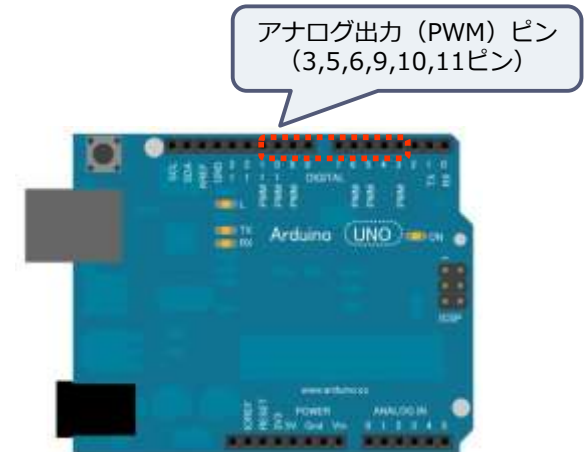
- ▶ アナログ値（PWM）を出力

- 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



Arduino UNOの場合

5. Arduino UNO インタフェース (3)

▶ 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)を読み込む

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)

「pinMode(pin,INPUT);」省略可

int **digitalRead**(pin)

- pin 0~13 (およびA0~A5)
- 戻り値 **LOW**(0) or **HIGH**(1)

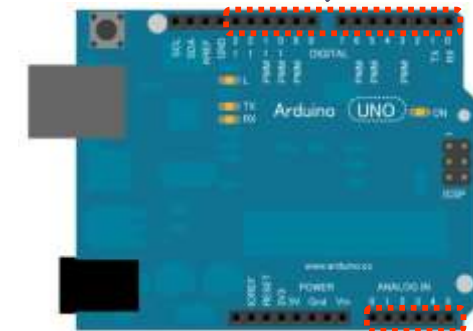
プルアップ抵抗を考慮 (モード)

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



デジタル
入力ピン
0~13

Arduino UNOの場合

デジタル入力ピン
14~19 (A0~A5)

5. Arduino UNO インタフェース (4)

▶ デジタル出力 (GPIO)

- ▶ 指定したピンへ0(LOW)または1(HIGH) を出力

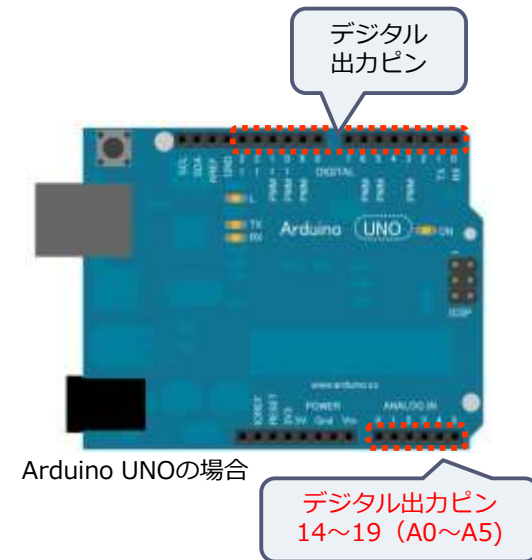
```
void pinMode(int pin, int mode)  
  □ pin      0~13およびA0~A5  
  □ mode     OUTPUT  
void digitalWrite(int pin, int value)  
  □ pin      0~13およびA0~A5  
  □ value     LOW(0) or HIGH(1)
```

宣言 : 3番ポートを利用

```
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

電源とGND
で利用可能



5. Arduino UNO インタフェース (5)

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>
```

5. Arduino UNO インタフェース（6）

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

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

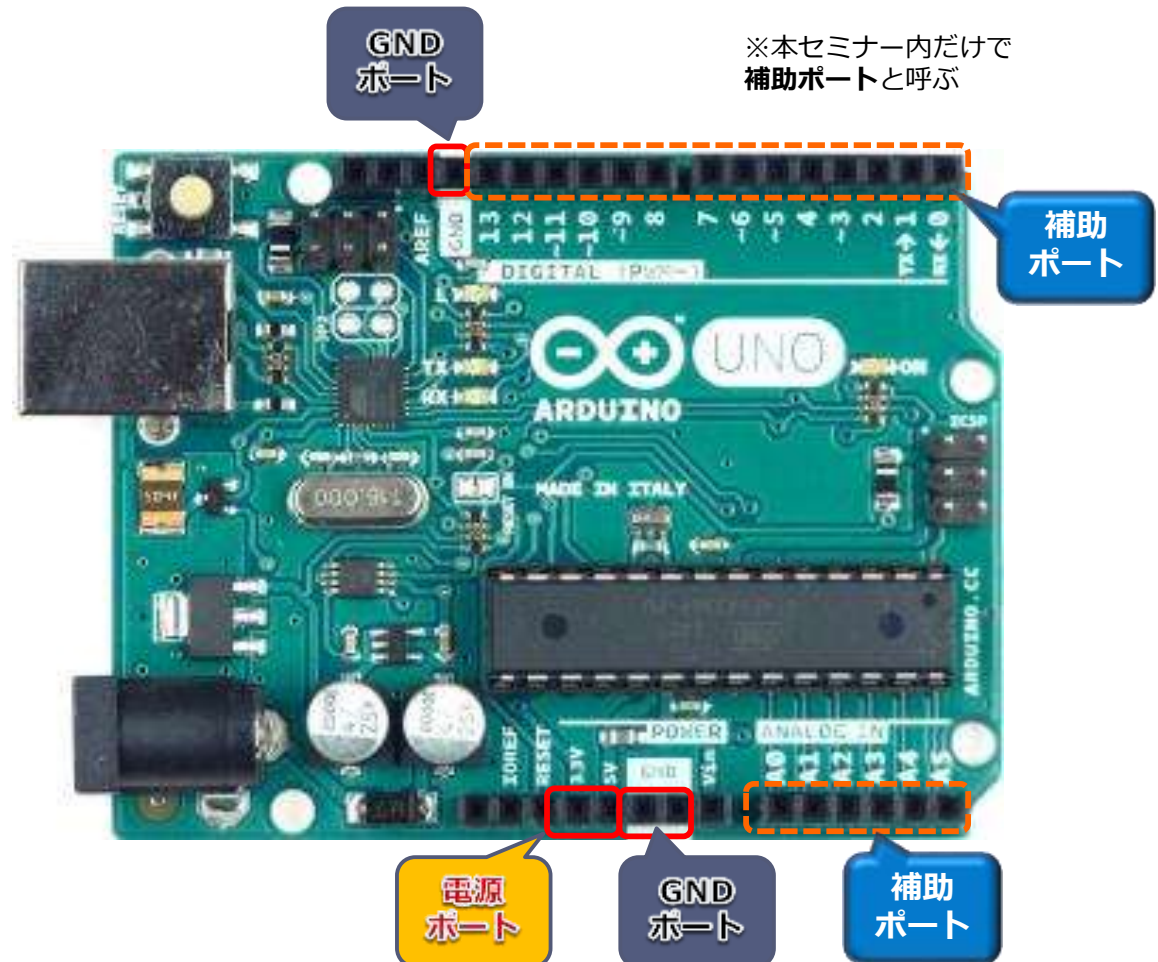
```
pinMode(Dx, OUTPUT);
```

// 電源ポートの場合

```
digitalWrite(Dx, HIGH);
```

// GNDポートの場合

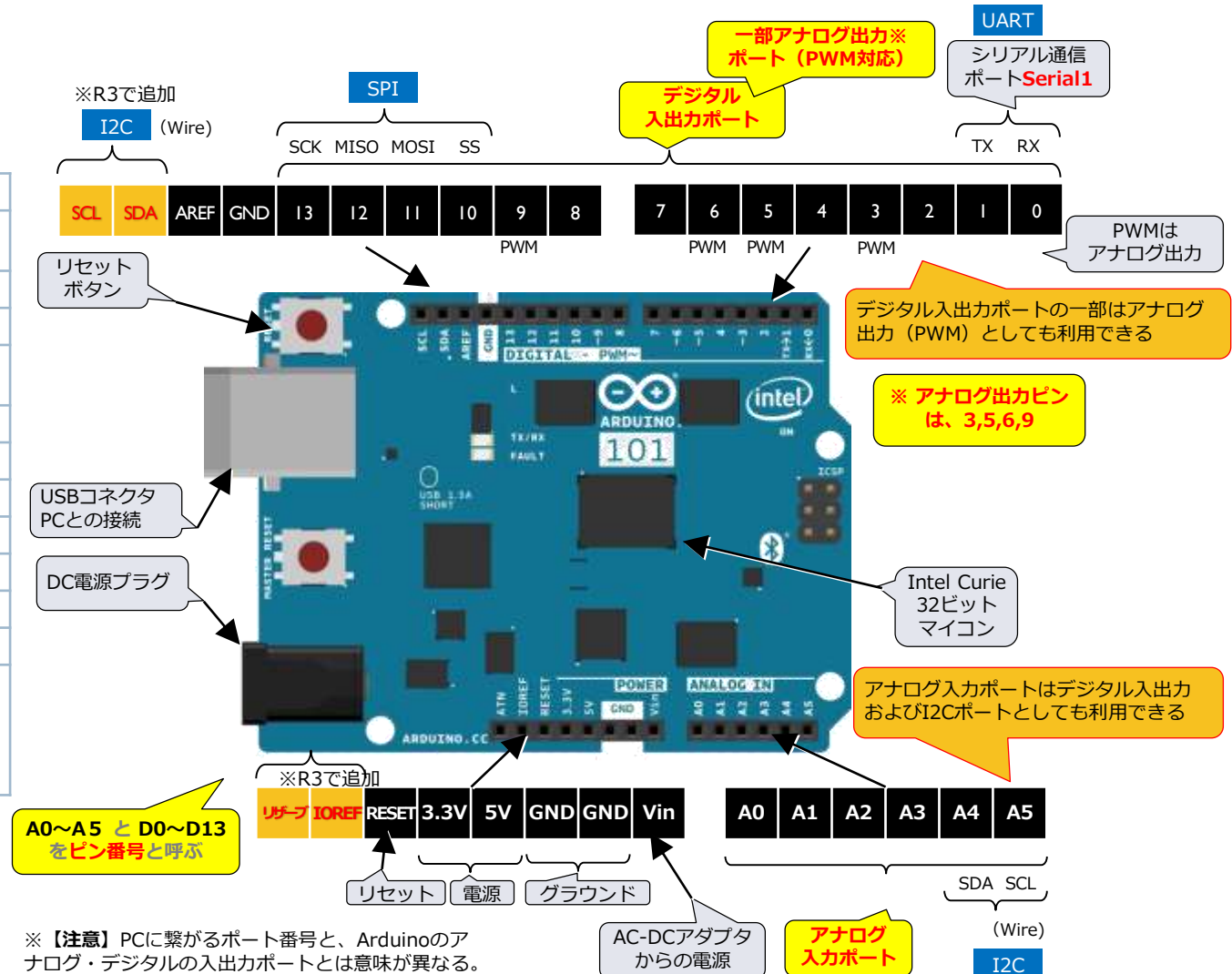
```
digitalWrite(Dx, LOW);
```



6. Genuino101 インタフェース (1)

■ Genuino101技術仕様

マイクロコントローラ	Intel Curie
制御電圧	3.3V (5V tolerant I/O)
入力電圧 (推奨値)	7-12V
入力電圧 (制限値)	7-17V
GPIO ピン数	14 (うち4ピンPWM)
PWMピン数	4
アナログ入力ピン	6
I/Oピンの電流	20 mA
フラッシュメモリ	196 kB
SRAM	24 kB
クロック速度	32MHz
LEDピン	D13
機能	BLE、EEPROM (2 K バイト)、リアルタイム クロック、 6軸加速度・ジャイロ センサ



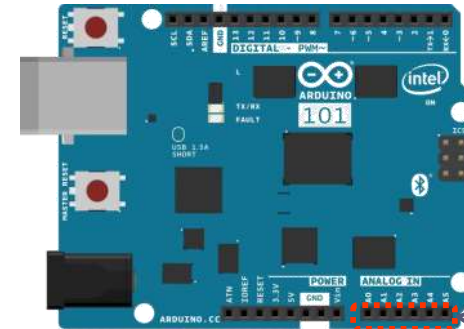
6. Genuino101 インタフェース (2)

▶ アナログ入力 (ADI: 6 ピン)

- ▶ Uno/Fioでは、6ピンまで利用可能(A0-A5)

- ▶ int **analogRead**(int pin)

- pin A0~A5
 - 戻り値 0~1023 (0は0V、1023は**3.3V**)



アナログ
入力ピン

▶ アナログ出力 (PWM : 4 ピン)

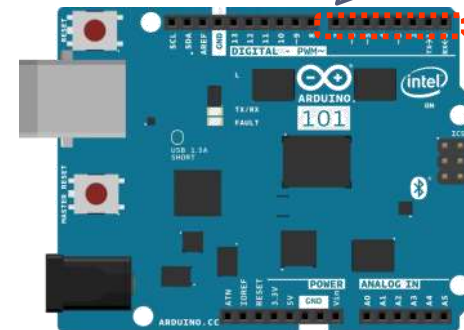
- ▶ Uno/Fioでは、デジタルピンと共用で、6ピンまで利用可能

- ▶ void **analogWrite**(int pin, int value)

- pin **3,5,6,9** ピン (注 : UNOより少ない)
 - value 0~255 (0は0V、255は**3.3V**)

- ▶ 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490Mhz



アナログ出力 (PWM) ピン
(3,5,6,9 ピン)

6. Genuino101 インタフェース (3)

▶ デジタル入力 (GPIO : 20ピン)

▶ Uno/Fioでは、13ピンまで利用可能(D0-D13) & A0-A5

▶ 指定したピンから、0 (LOW) または1 (HIGH) を読み込む

▶ void **pinMode**(int pin, int mode)

- pin 0~13 & A0~A5
- mode INPUT (デフォルトではINPUT)

「pinMode(pin,INPUT);」省略可

▶ int **digitalRead**(pin)

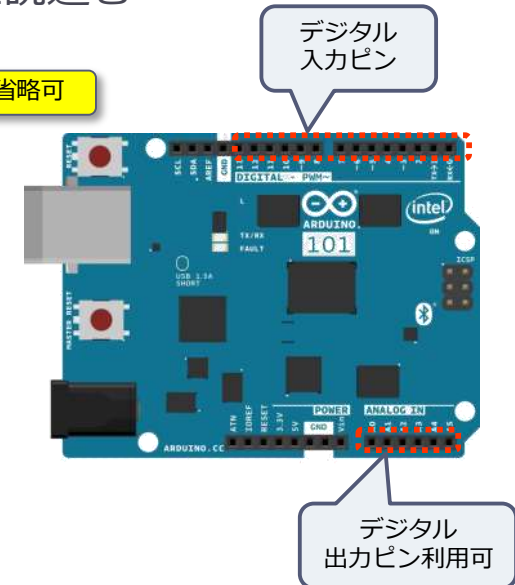
- pin 0~13 & A0~A5
- 戻り値 0 or 1

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



6. Genuino101 インタフェース（４）

▶ デジタル出力（GPIO：20ピン）

▶ 指定したピンへ0または1を出力

▶ void **pinMode**(int pin, int mode)

- pin 0~13 & A0~A5
- mode OUTPUT

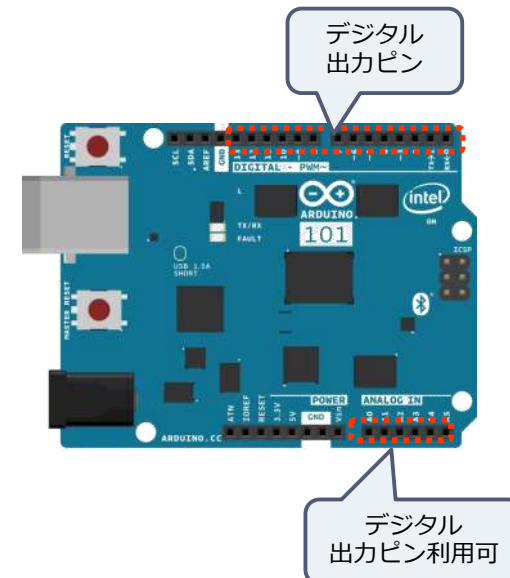
▶ void **digitalWrite**(int pin, int value)

- pin 0~13 & A0~A5
- value 0 or 1

宣言：3番ポートを利用

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

3番ポートを
出力として設定



6. Genuino101 インタフェース (5)

▶ **UART (Serial・Serial1) : ハードウェアシリアル通信**

▶ D0 (Rx) / D1 (Tx) ピン利用

- ❑ `Serial1.begin (baudrate);`
- ❑ `Serial1.readStringUntil('¥n');`
- ❑ `Serial1.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**

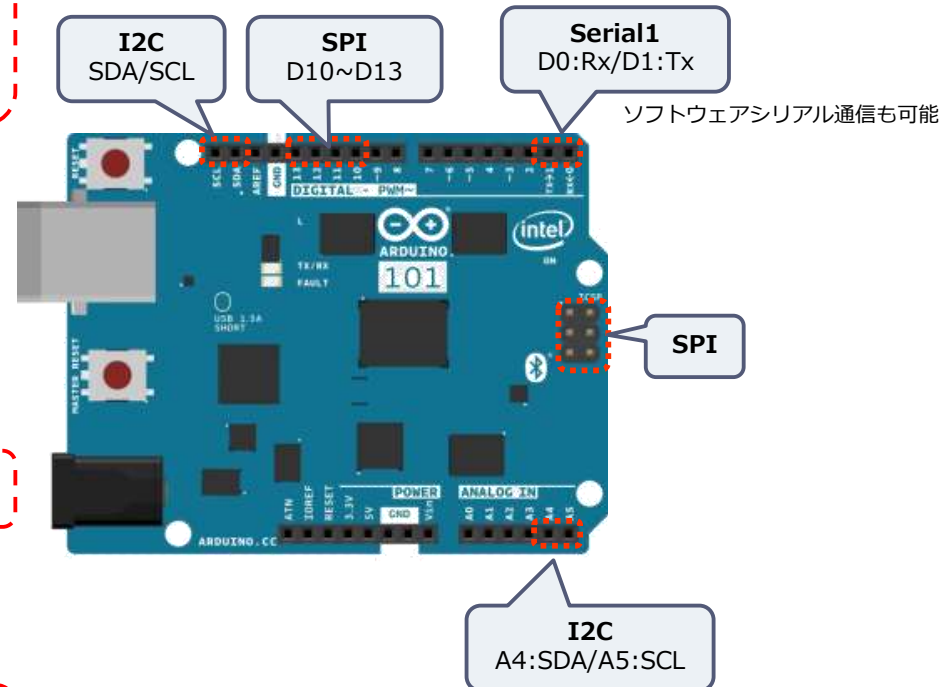
I2Cコネクタ2箇所 (A4/A5も利用可)

```
#include <Wire.h> を利用
```

▶ **SPI**

▶ SPIコネクタ利用 (D10~D13)

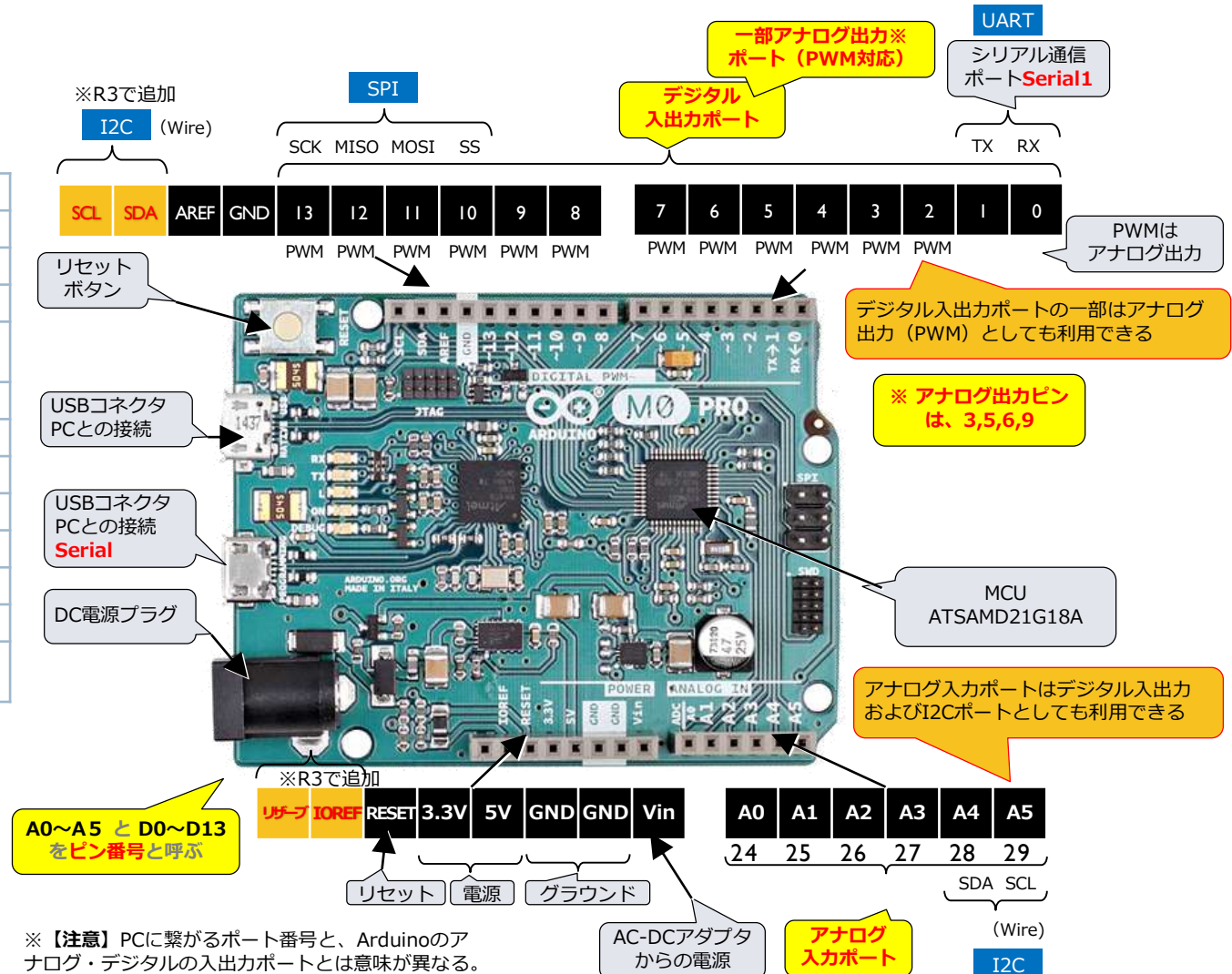
```
#include <SPI.h>
#include <SPI_registers.h>
```



7. Arduino M0 Pro インタフェース

■ 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 (うち4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)



8. 電子部品の取扱いについて

Arduino上で使う電子部品

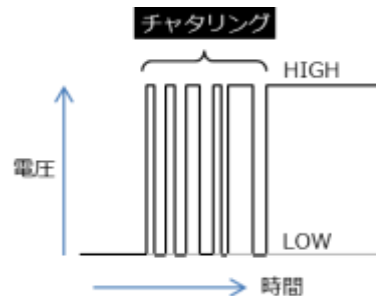
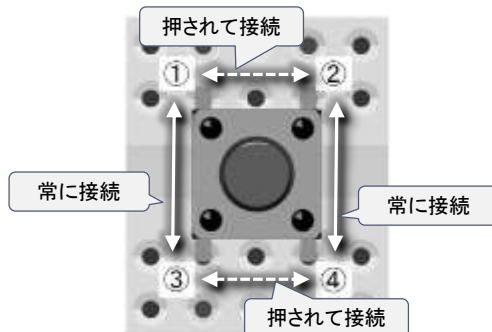
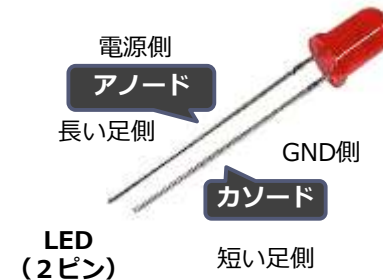
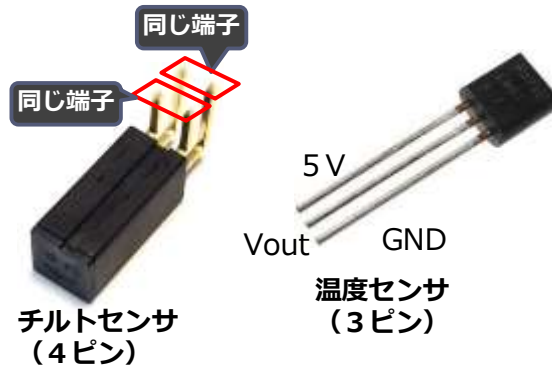
電子部品の利用上の注意点

- 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

入力

処理

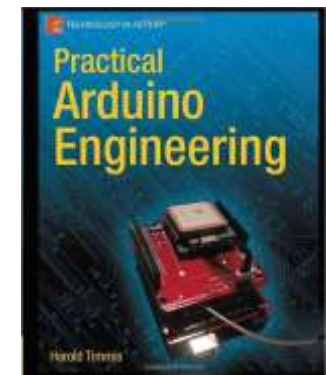
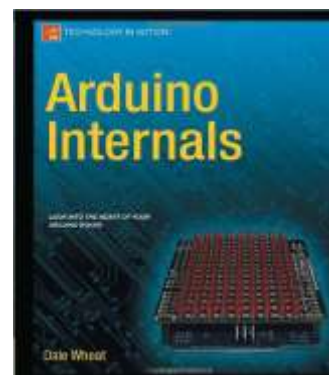
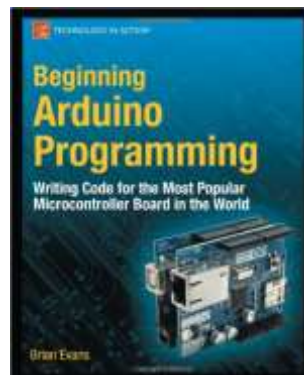
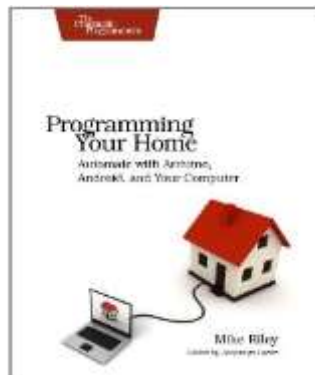
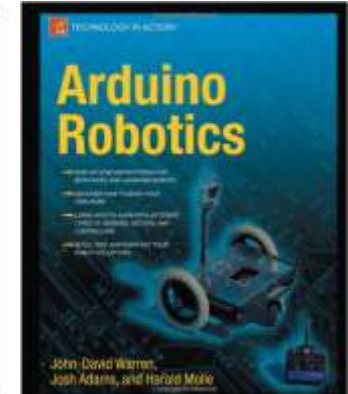
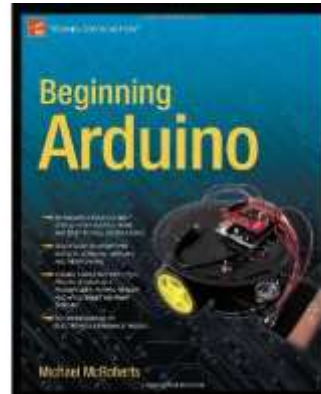
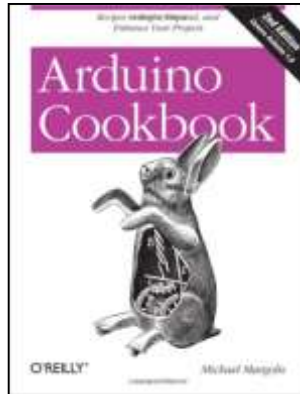
出力



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関連の参考本 ②（洋書）



Arduino関係の洋書
今後も多く販売予定

もくじ

1. 準備段階
2. Arduino IDE インストール手順
3. Arduino IDE 画面説明
4. Arduino接続
5. Arduino IDE 環境設定
6. ライブラリについて



第2章 Arduino IDEの準備

1. 準備段階

1) Arduino UNO・Genuino101・Arduino M0 Pro 関連ハードの準備（購入）

- ① Arduino + USBケーブル
- ② 電子部品類
- ③ ブレッドボード+ジャンパーコードなど

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
- ③ 開発環境の設定

3) 接続・確認

- 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

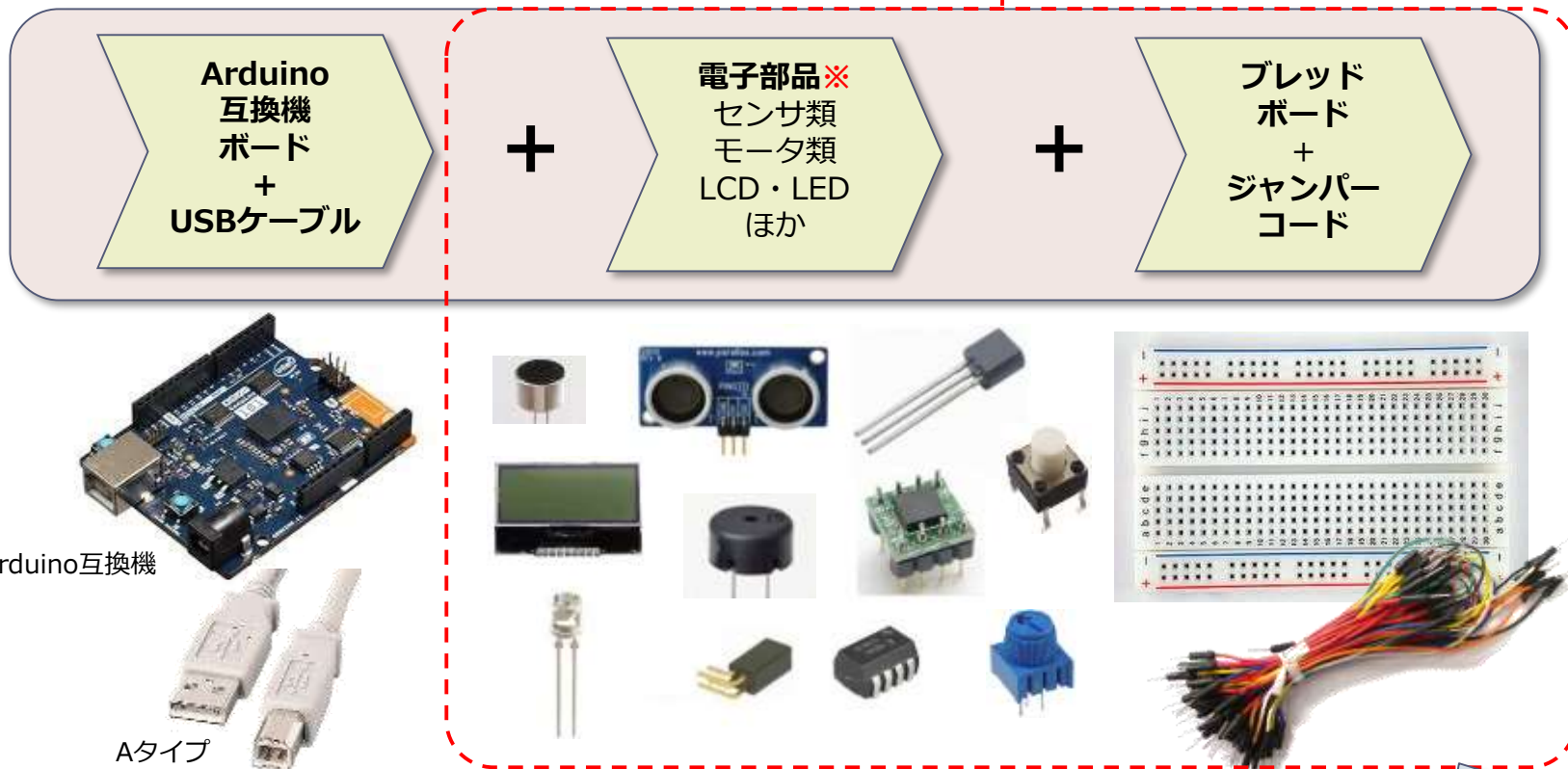
(1) Arduino 関連ハードの準備

▶ 準備段階（ハードウェアの準備）

1) Arduino 関連ハードの準備



IoTABシールド



購入先

秋葉原：秋月電商、千石電商、若松通L商など
ネット販売：スイッチサイエンスほか

(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



OSは、Mac OS X・Windows・Linuxに対応

arduino.cc から 統合開発環境 (IDE) の最新版を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。

2. 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/Downloads

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）

2. Arduino IDE インストール手順（2）

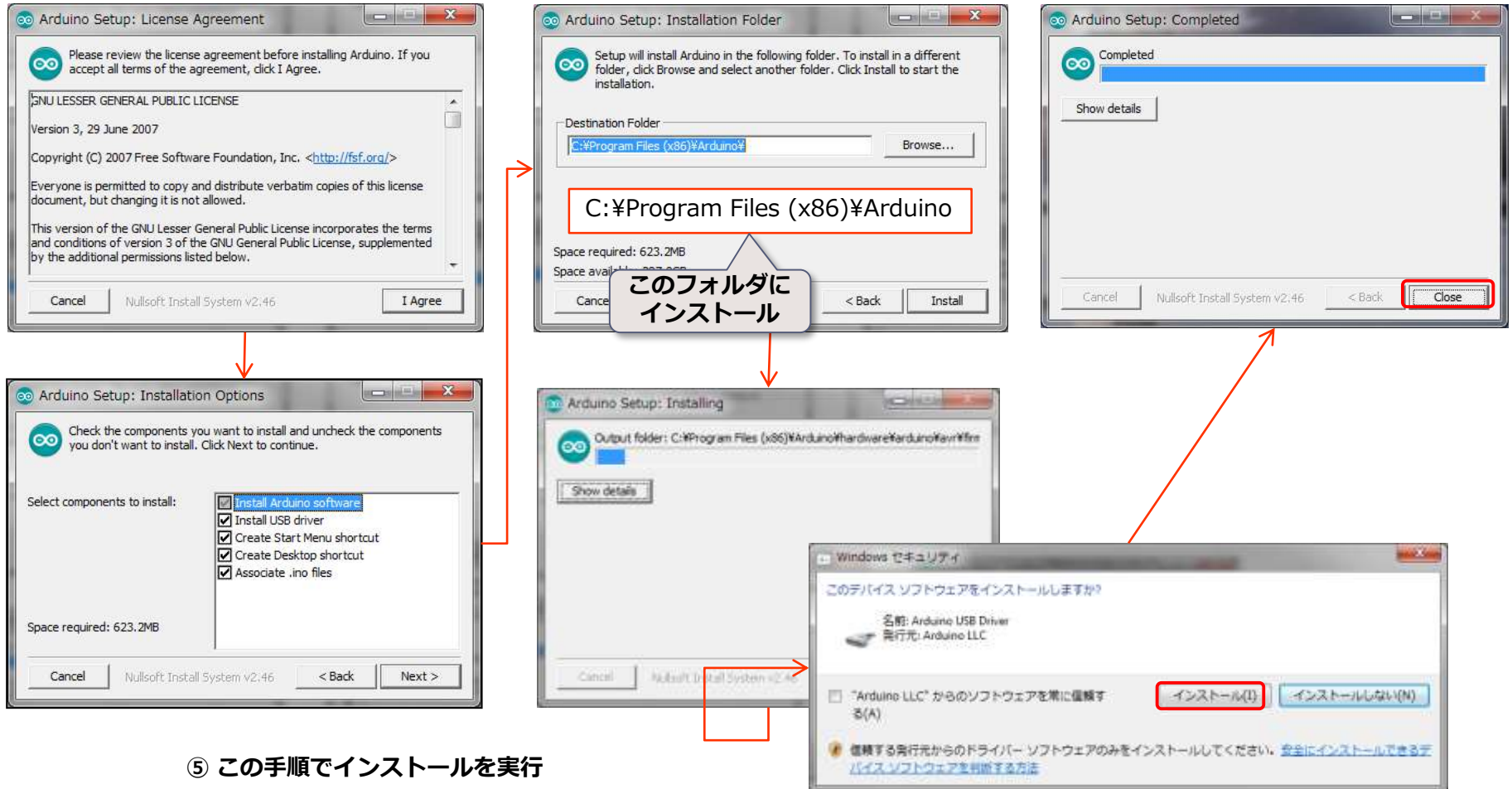
- ▶ Arduino IDE Ver1.8.8（日本語化対応版）のダウンロード（2019年2月時点最新版）

The screenshot shows the Arduino website's download page. On the left, a teal sidebar lists download options: 'Windows Installer, for Windows XP and up' (highlighted with a red box and a callout 'ここから自動インストール版をダウンロードする'), 'Windows ZIP file for non admin install' (highlighted with a red box and a callout 'ZIPファイル版 複数バージョンをダウンロードするとき便利'), 'Windows app Requires Win 8.1 or 10' (with a callout 'Windows アプリ版'), 'Mac OS X 10.8 Mountain Lion or newer' (with a callout 'Mac版'), and 'Linux 32 bits', 'Linux 64 bits', and 'Linux ARM' (grouped with a red box and a callout 'Linux版'). Below these are links for 'Release Notes', 'Source Code', and 'Checksums (sha512)'. On the right, the main content area has a 'Support the Arduino Software' section with a donation form. The form shows a progress bar for donations (\$3, \$5, \$10, \$25) and a 'JUST DOWNLOAD' button (highlighted with a red box and a callout '無償ダウンロード'). A red arrow points to the 'JUST DOWNLOAD' button. A callout '寄付をする場合' points to the donation options. The URL 'https://www.arduino.cc/en/Main/Donate' is visible in the browser address bar.

ダウンロード画面

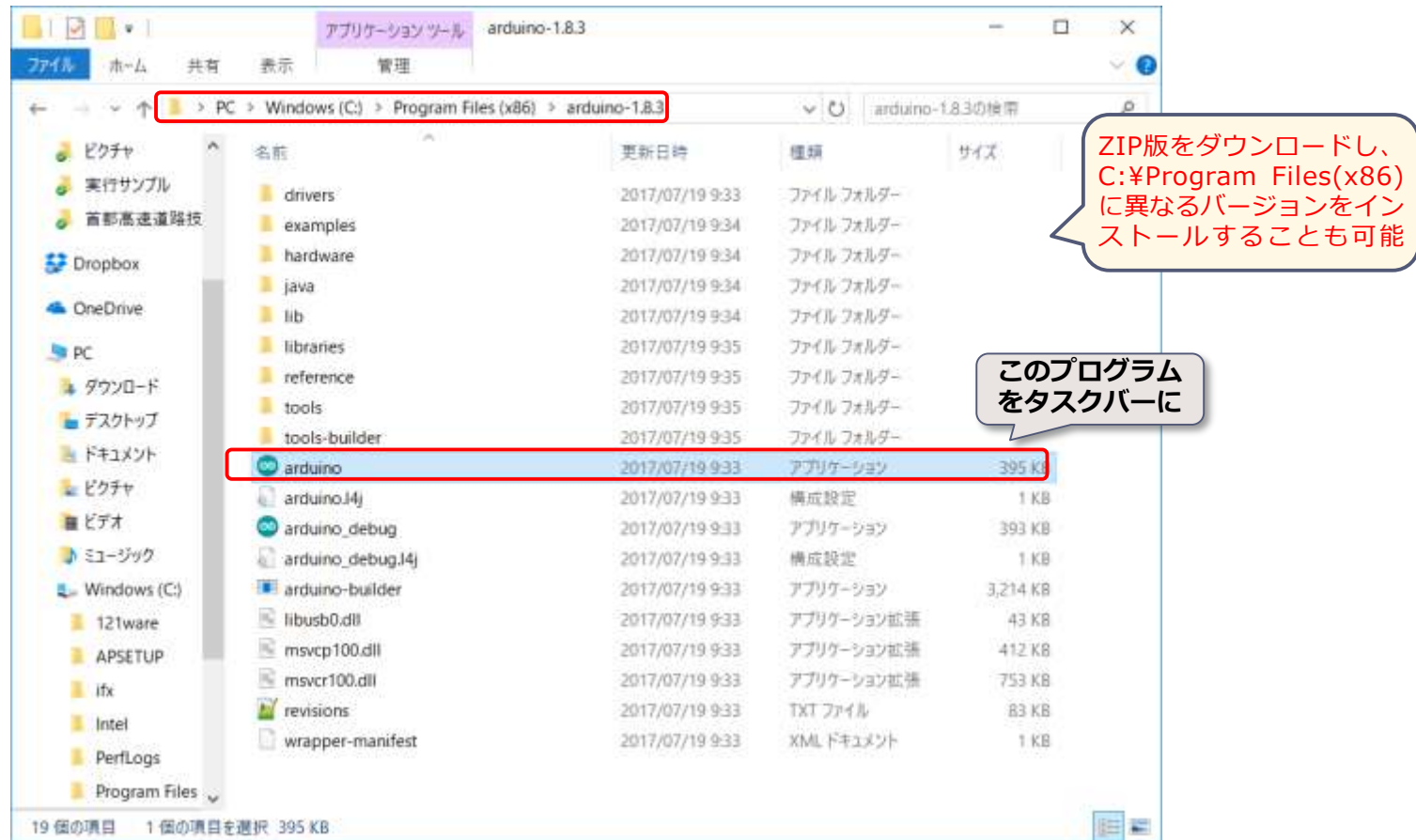
2. Arduino IDE インストール手順（3）

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



2. Arduino IDE インストール手順（４）

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



⑥ ダウンロードされたフォルダ
→ タスクバーに表示

3. Arduino IDE 画面説明（1）

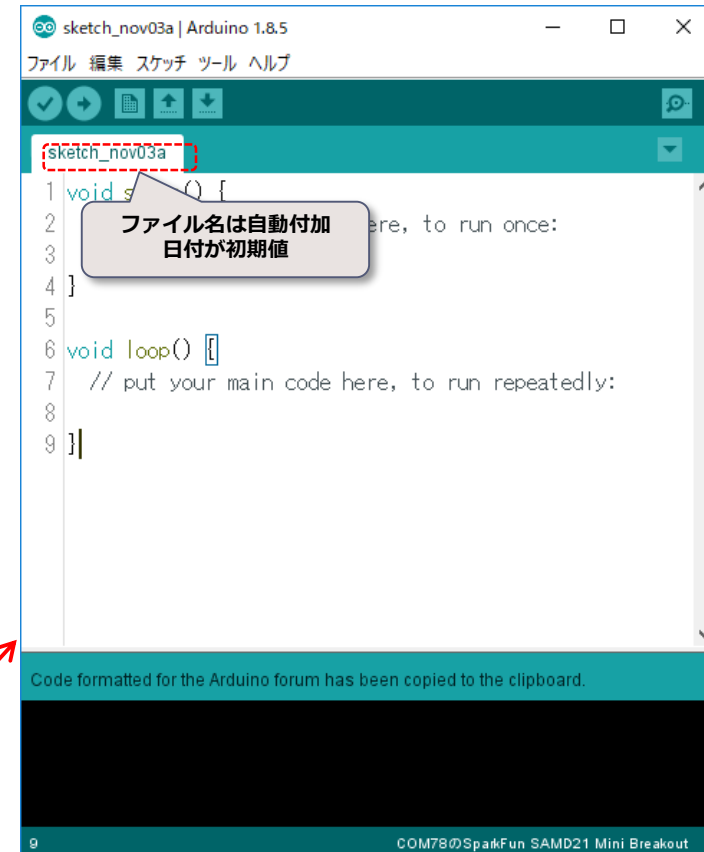
Arduino IDEのアイコンを選択（クリック等）することで起動



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

3. 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

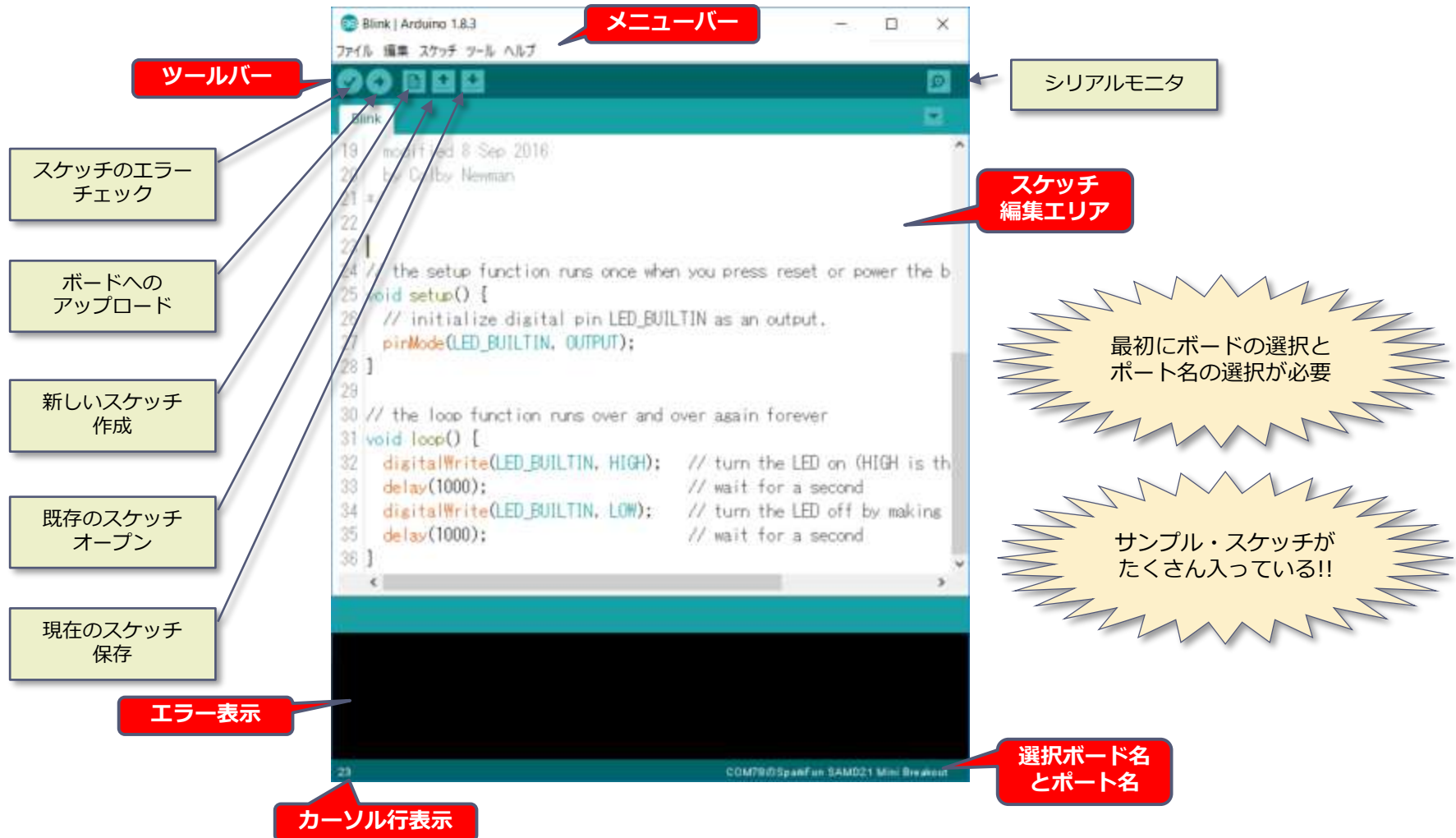
The screenshot shows the Arduino IDE window titled "sketch_nov03a | Arduino 1.8.5". The menu bar includes "ファイル", "編集", "スケッチ", "ツール", and "ヘルプ". The toolbar contains icons for Verify, Upload, New File, Open, and Save. The file name "sketch_nov03a" is shown in the title bar. The main editor area contains the following code:

```
1 void setup() {
2   // put your setup code here, to run once:
3 }
4
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8 }
9 }
```

Annotations include:

- メニューバー**: Points to the menu bar.
- ツールバー**: Points to the toolbar.
- ファイル名 (タブ名)**: Points to the file name in the title bar.
- エディタ画面 (スケッチ)**: Points to the main code editor area.
- スケッチを作成するエリア**: Points to the main code editor area.
- タブ機能およびファイル名編集**: Points to the tab list on the right, showing "sketch_nov03a" and a context menu with options like "新規タブ", "名前を変更", "削除", "前のタブ", and "次のタブ".
- メッセージ表示画面 (エラー表示は赤字)**: Points to the status bar area showing a message: "シリアルポートのパラメータを設定中にエラーが発生しました: 11".
- USB接続シリアル番号**: Points to the status bar showing "COM78のSparkFun SAMD21 Mini Breakout".

3. Arduino IDE 画面説明（3）



- スケッチを呼出
- スケッチの事例を呼び出す
- スケッチを保存

■スケッチのコンパイル

■スケッチのアップロード

ツールバー

メンバー

シリアルモニタ

■シリアルモニタ画面

タブ関連メニュー

■タブ（モジュール） タブ画面対応

自動整形

Ctrl+■ スケッチを自動整形

スケッチをアーカイブする

エンコーディングを修正

シリアルキータ

シリアルプロッタ

Ctrl+Shift+M

Ctrl+Shift+L

WiFi101 Firmware Updater

ボード: "Arduino/Genuino Uno"

■ Arduinoの種類選択

シリアルポート: "COM83 (Arduino/Genuino Uno)"

■シリアルポートの選択

ボード情報を取得

直达装运

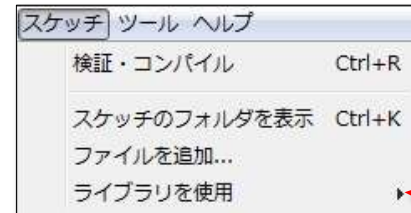
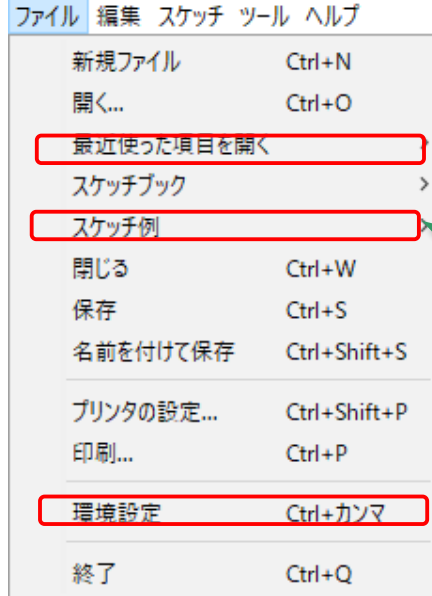
ブートローダを書き込む

3. Arduino IDE 画面説明 (5)

ヘッダーファイル
の利用

③

ライブラリを使用



① スケッチの例

- 01. Basics
- 02. Digital
- 03. Analog
- 04. Communication
- 05. Control
- 06. Sensors
- 07. Display
- 08. Strings
- 09. USB
- 10. StarterKit
- ArduinoISP

マイコンボード

②

ボードマネージャ...

⑤

- Arduino SAMD (32-bits ARM Cortex-M0+)
- Arduino/Genuino Zero (Programming Port)
- Arduino/Genuino Zero (Native USB Port)
- Arduino/Genuino MKR1000
- Arduino AVR512-D
- Arduino Yun
- Arduino/Genuino Uno
- Arduino Duemilanove or Diecimila
- Arduino Nano
- Arduino/Genuino Mega or Mega 2560
- Arduino Mega ADK
- Arduino Leonardo
- Arduino/Genuino Micro
- Arduino Esplora
- Arduino Mini
- Arduino Ethernet
- Arduino Flo
- Arduino BT
- LilyPad Arduino USB
- LilyPad Arduino
- Arduino Pro or Pro Mini
- Arduino NG or older
- Arduino Robot Control
- Arduino Robot Motor
- Arduino Gemma
- Arduino ARM (32ビット) ボード
- Arduino Due (Programming Port)
- Arduino Due (Native USB Port)
- Tabbrain AVR Board
- m1284p
- AMEL-Tech Boards
- Smart Everything Fox (via Atmel-ICE)
- Smart Everything Fox (via SAM-ICE)



Arduinoボード
USB接続ポート

3. Arduino IDE 画面説明（6）

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。

シリアルモニタ画面

シリアルプロッタ画面

基本は、CSV形式（数値群を1行毎区切り記号【空白やカンマなど】）で出力されたものがグラフに表示されます

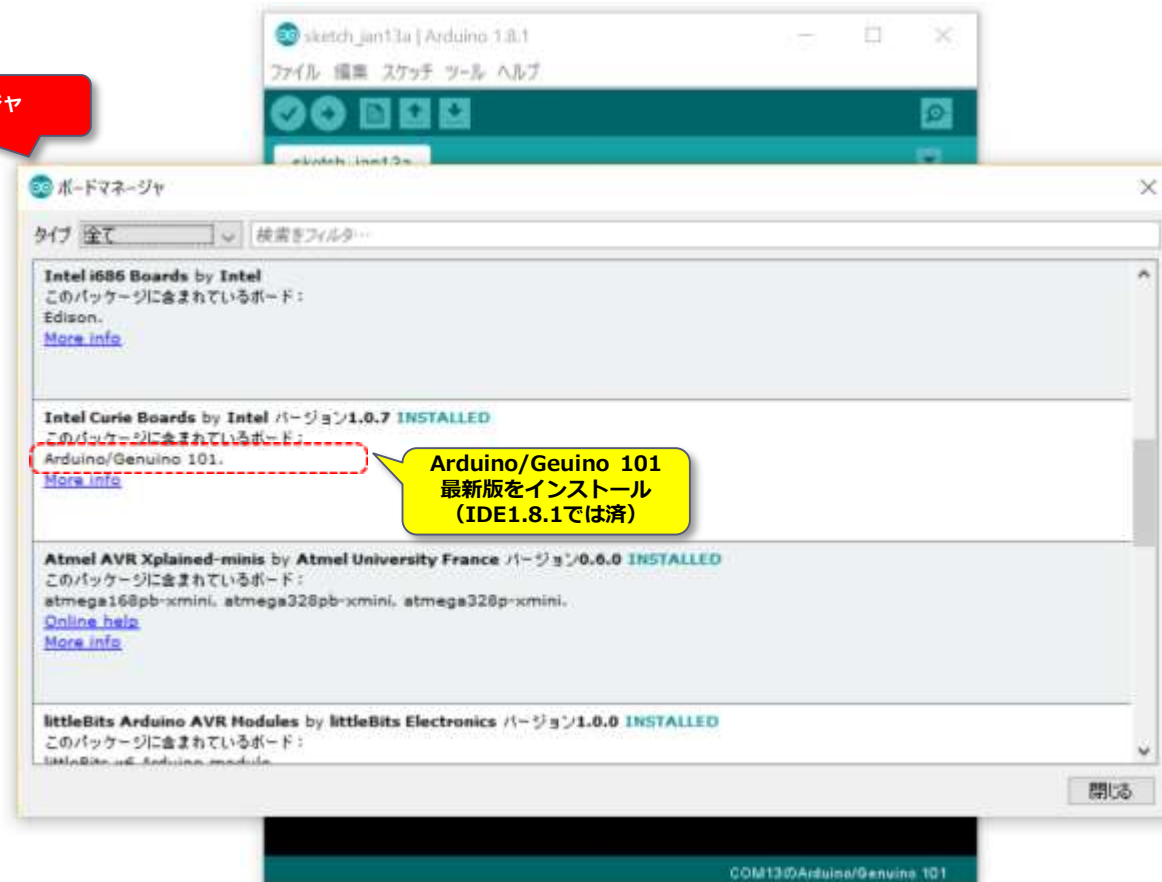
シリアル画面に出力するには「Serial.begin(BAUDRATE);」で通信速度（BAUDRATE）を引数として宣言します。

通信速度「BAUDRATE」を合わせます。
ここでBAUDRATEは、一般に9600~115200までが良く利用されます

4. Arduino接続（Genuino101の場合①）

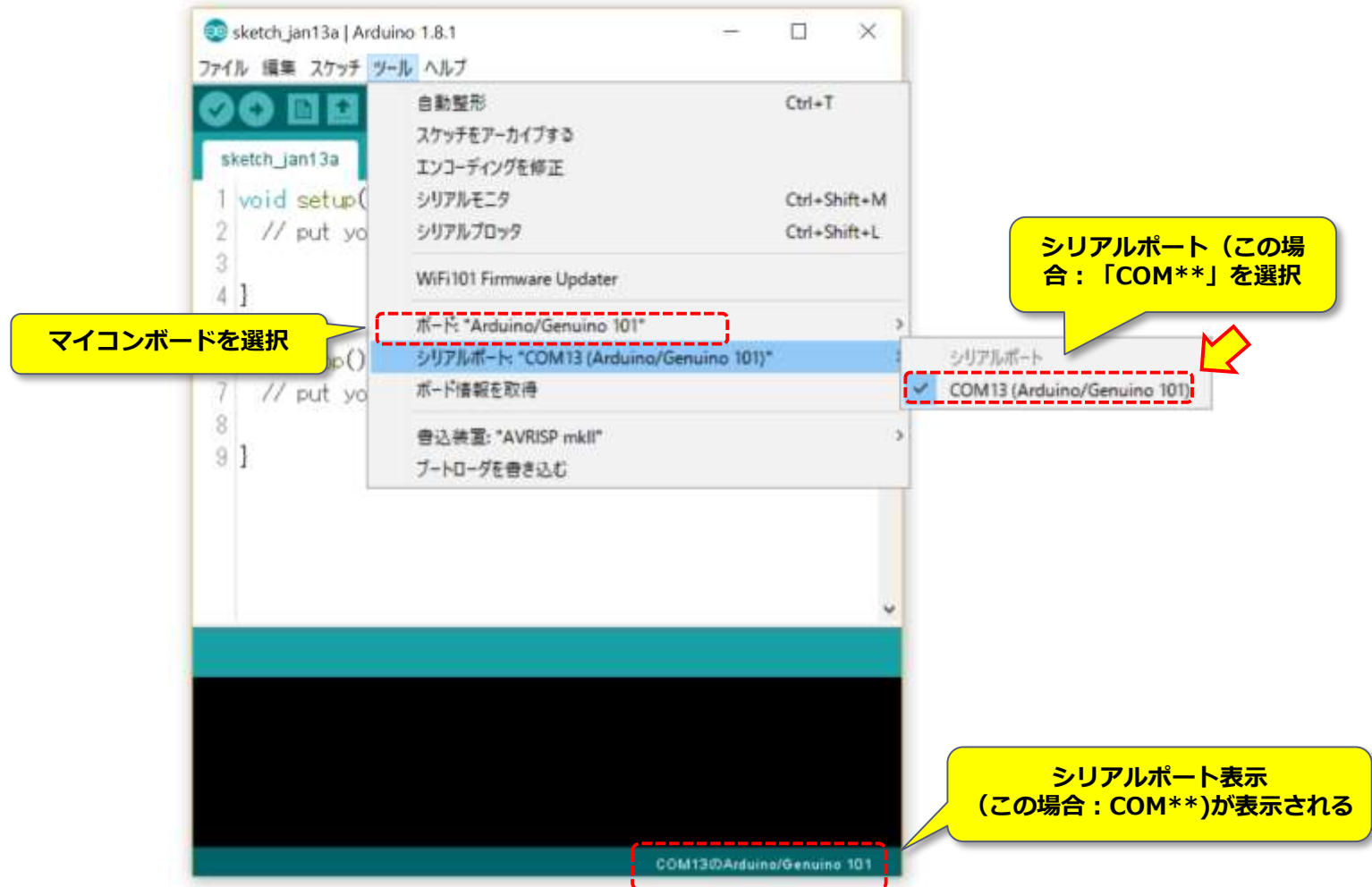
- ① Arduino（Genuino101）とPCとをUSBケーブルで接続
- ② 「ツール」→「ボード」→「ボードマネージャ」選択

② ボードマネージャ



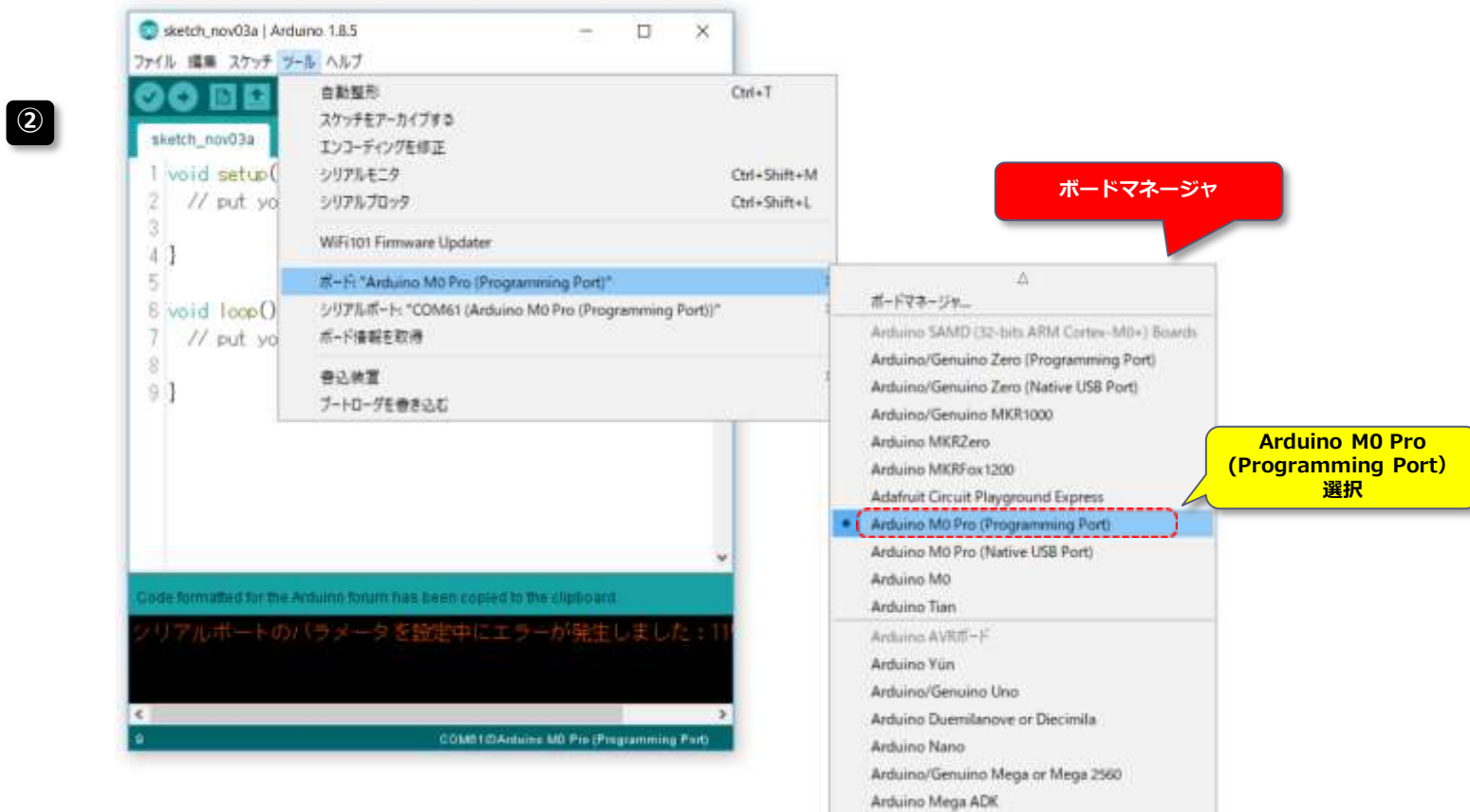
4. Arduino接続（Genuino101の場合②）

Arduino IDE側でのシリアルポートの設定



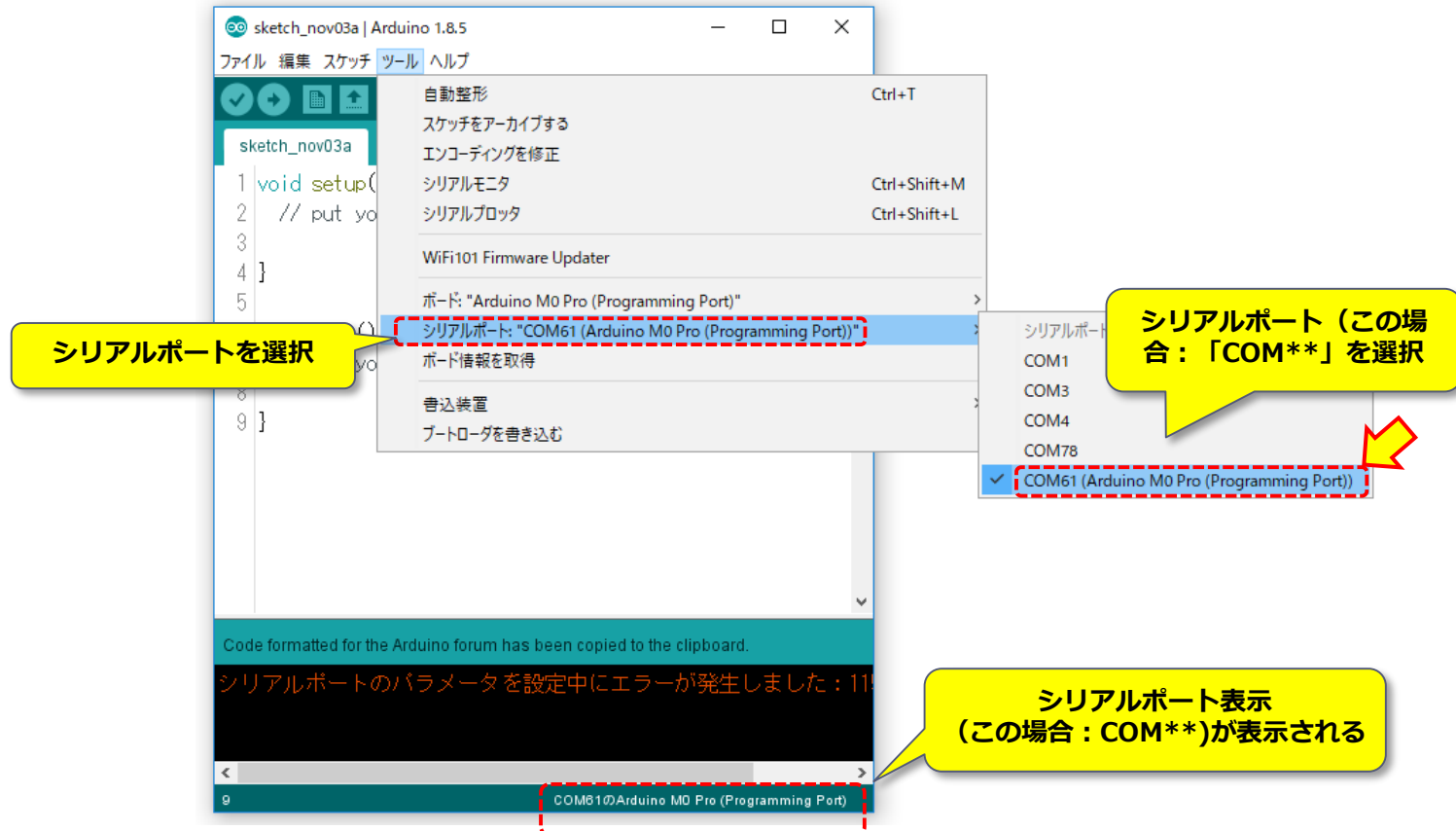
4. Arduino接続（Arduino M0 Proの場合①）

- ① Arduino（Arduino M0 Pro）とPCとをUSBケーブルで接続
- ② 「ツール」→「ボード」→「ボードマネージャ」選択



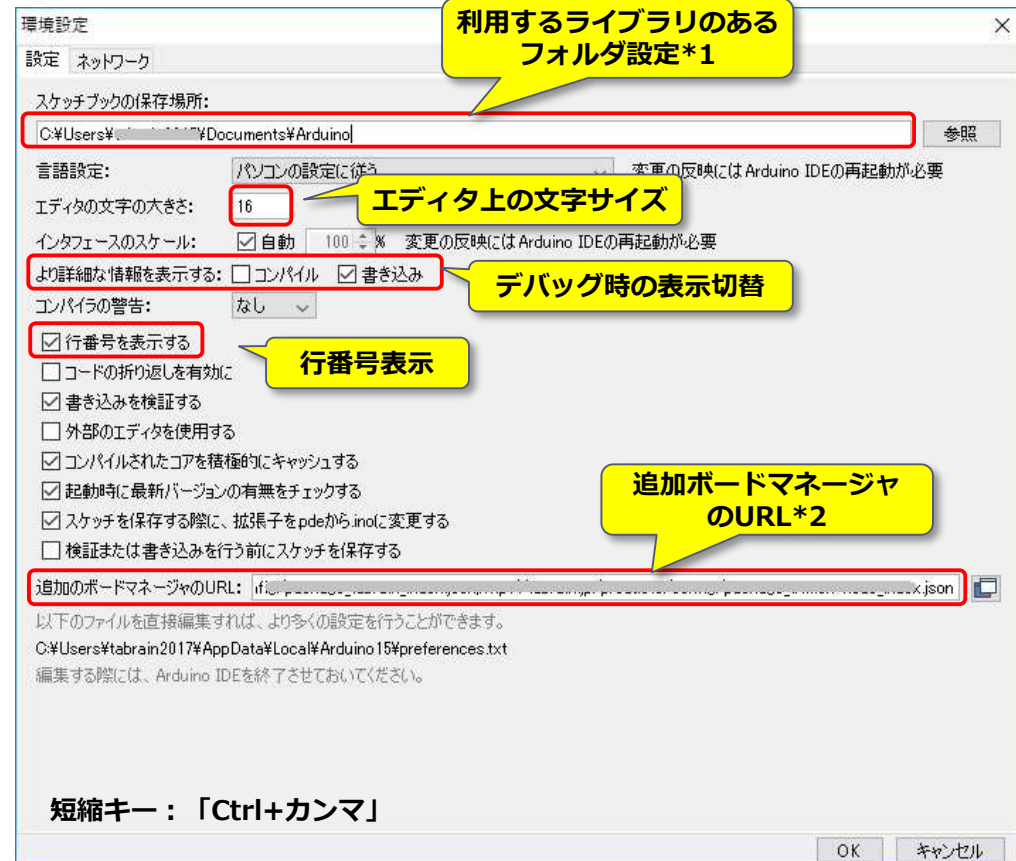
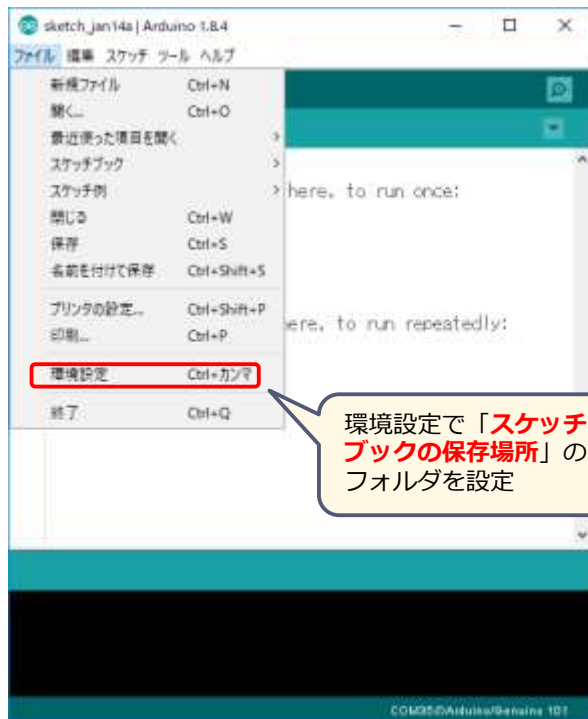
4. Arduino接続 (Arduino M0 Proの場合②)

Arduino IDE側でのシリアルポートの設定



5. Arduino IDE 環境設定

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できます。

6. ライブラリについて（1）

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
- 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」>「スケッチブックの保存場所」>「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"」で呼び出して利用可能

【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」で設定されている<アカウント名> ¥ ドキュメント以下に展開することを推奨します。

6. ライブラリについて（2）

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勧めします。

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。

6. ライブラリについて（3）

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に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でき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・ウェイク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のライブラリ群フォ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ありますので用途に合わせ検索して参考にしてください。

6. ライブラリについて（４）

【IoTABシールドV4.0サンプルをライブラリとして使用する例】

The image shows the Arduino IDE environment settings and a file explorer window. The environment settings window is titled "環境設定" (Environment Settings) and has the "設定" (Settings) tab selected. The "スケッチブックの保存場所:" (Sketchbook storage location) is set to "C:\Users\%user%\Documents\IoTABシールドV4.0". The "言語設定:" (Language settings) are set to "パソコンの設定に従う" (Follow computer settings). The "エディタの文字の大きさ:" (Editor font size) is set to 16. The "インタフェースのスケール:" (Interface scale) is set to 100%. The "より詳細な情報を表示する:" (Show more information) checkbox is checked. The "コンパイラの警告:" (Compiler warnings) are set to "なし" (None). The "行番号を表示する:" (Show line numbers) checkbox is checked. The "コードの折り返し:" (Code folding) checkbox is checked. The "書き込みを検証:" (Verify upload) checkbox is checked. The "外部のエディタを:" (External editor) checkbox is checked. The "コンパイルされた:" (Compiled) checkbox is checked. The "起動時に最新:" (Update on startup) checkbox is checked. The "スケッチを保存する:" (Save sketch) checkbox is checked. The "検証または書き込み:" (Verify or upload) checkbox is checked. The "追加のボードマネージャ:" (Additional board managers) checkbox is checked. The "以下のファイルを直接:" (Directly edit the following files) checkbox is checked. The "C:\Users\%user%\Documents\IoTABシールドV4.0" folder is selected in the file explorer. The file explorer shows the contents of the "IoTABシールドV4.0" folder, which includes a list of files and folders. The files and folders are listed in a table with columns for "名前" (Name), "更新日時" (Last modified), and "種類" (Type). The files and folders are listed as follows:

名前	更新日時	種類
IoTAB4_ALL_TEST_UNO	2018/05/26 9:45	ファイル フォルダ
IoTAB4_BH1780	2018/05/26 9:45	ファイル フォルダ
IoTAB4_BH1780_LCD	2018/05/26 9:45	ファイル フォルダ
IoTAB4_CLAP_count	2018/05/26 9:45	ファイル フォルダ
IoTAB4_DEMO_UNO_3GIM	2018/05/26 9:45	ファイル フォルダ
IoTAB4_DIST	2018/05/26 9:45	ファイル フォルダ
IoTAB4_DIST_LCD	2018/05/26 9:45	ファイル フォルダ
IoTAB4_DIST_LCD_D9	2018/05/26 9:45	ファイル フォルダ
IoTAB4_EEPROM	2018/05/26 9:45	ファイル フォルダ
IoTAB4_GPS_Clock_LCD	2018/05/26 9:45	ファイル フォルダ
IoTAB4_I2C_LCD	2018/05/26 9:45	ファイル フォルダ
IoTAB4_IR_getput_test	2018/05/26 9:45	ファイル フォルダ
IoTAB4_IR_READ	2018/05/26 9:45	ファイル フォルダ
IoTAB4_LED	2018/05/26 9:45	ファイル フォルダ
IoTAB4_LED_01	2018/05/26 9:45	ファイル フォルダ

The "IoTAB4_LED" folder is selected, and its contents are displayed in a list on the right side of the file explorer. The contents of the "IoTAB4_LED" folder are listed as follows:

名前	更新日時	種類
IoTAB4_LED	2018/05/26 9:45	ファイル フォルダ
IoTAB4_LED_01	2018/05/26 9:45	ファイル フォルダ
IoTAB4_LED_02	2018/05/26 9:45	ファイル フォルダ
IoTAB4_M2X_3_UNO_Library	2018/05/26 9:45	ファイル フォルダ
IoTAB4_M2X_uno	2018/05/26 9:45	ファイル フォルダ
IoTAB4_M2X_UNO_Library	2018/05/26 9:45	ファイル フォルダ
IoTAB4_melody	2018/05/26 9:45	ファイル フォルダ
IoTAB4_MIC	2018/05/26 9:45	ファイル フォルダ

The "IoTAB4_LED" folder is selected, and its contents are displayed in a list on the right side of the file explorer. The contents of the "IoTAB4_LED" folder are listed as follows:

名前	更新日時	種類
IoTAB4_LED	2018/05/26 9:45	ファイル フォルダ
IoTAB4_LED_01	2018/05/26 9:45	ファイル フォルダ
IoTAB4_LED_02	2018/05/26 9:45	ファイル フォルダ
IoTAB4_M2X_3_UNO_Library	2018/05/26 9:45	ファイル フォルダ
IoTAB4_M2X_uno	2018/05/26 9:45	ファイル フォルダ
IoTAB4_M2X_UNO_Library	2018/05/26 9:45	ファイル フォルダ
IoTAB4_melody	2018/05/26 9:45	ファイル フォルダ
IoTAB4_MIC	2018/05/26 9:45	ファイル フォルダ

もくじ

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）



第3章 ソフトウェア編（文法）

※Arduinoの開発言語は、C++言語のサブセットです。
ここでは、一般的なC言語に基づいて紹介しています。

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。（Arduino IDE上の開発言語を「Arduino言語」とも呼びます）

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）をご説明します。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) 組み込み関数群
- 5) キーワード等（一部）

1. 関数と引数

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

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 (組み込み関数)
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

【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
組み込み定数 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)

【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

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)

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> (= double)
 - 6) float (32ビット実数) :
 - 7) string (文字列)
 - 8) void (型なし)
- その他、配列や構造体などあり

型はエディタ上で
色が付く

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

Arduinoにない制御文

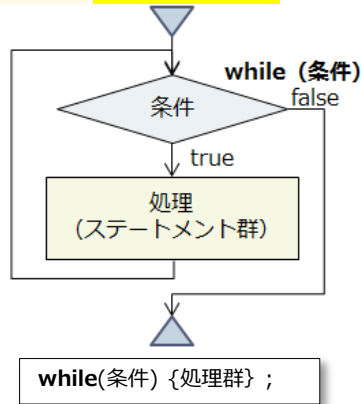
- repeat文 なし
- until文 なし

制御文（一部）

制御文は「分岐」と「反復」の2つ

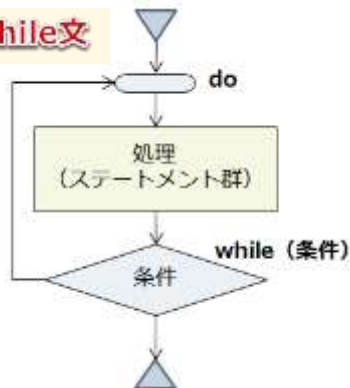
■while文

分岐&反復



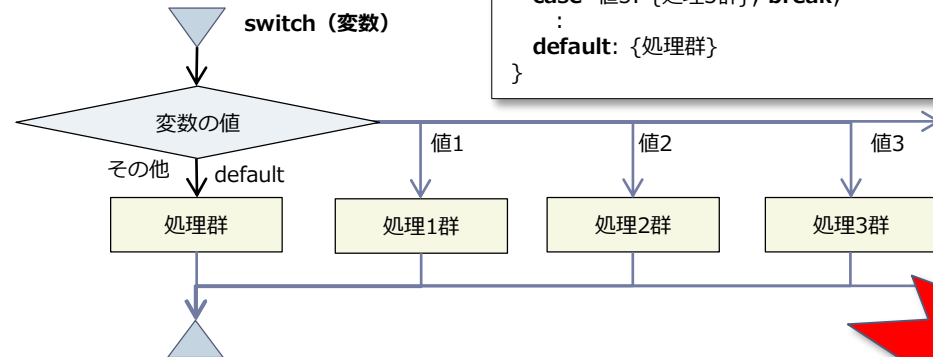
■do-while文

分岐&反復



■switch文

分岐



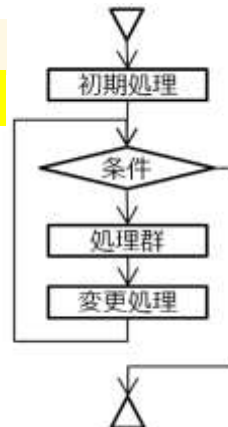
```

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

■for文

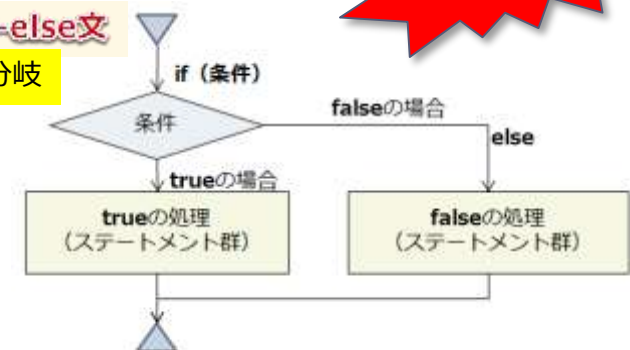
分岐&反復



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

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】

```
until(SENSOR_1==1); //エラーなし
until(SENSOR_1 == 1); //エラーなし
until(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
'c' //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. 組み込み関数群

【参考】Web上で
Arduino リファレンス
参照

【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: 読み込み (INPUT_PULLUP, INPUT)
        書き込み (OUTPUT)
    ※ modeが INPUTの場合は省略可
```

【デジタル読み込み (入力) 関数】

```
digitalRead(pin);
    pin: ピン番号
    戻り値: HIGH / LOW
```

【デジタル書き込み (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

【アナログ読み込み (入力) 関数】

```
analogRead(pin);
    pin: ピン番号
    戻り値: 0 ~ 1023
```

【アナログ書き込み (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、
デジタル入出力の場合: 0 ~ 19
アナログ入力の場合: A0 ~ A5
アナログ出力の場合: 3, 5, 6, 9, 10, 11
(Arduino UNOの場合)

5. キーワード等 (一部)

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
EEPROM.read	EEPROM読み込み	引数: 番地
EEPROM.write	EEPROM書き込み	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)	
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	

もくじ

1. サンプル・スケッチを実行
 2. スケッチの作成方法
 3. プログラミングについて
 4. 自作スケッチの実行
 5. Arduinoプログラミングの可能性について
 6. サンプルスケッチ
- 【ブレイク】プログラミングで設計書が必要か？
【ブレイク】Arduino方言ほか

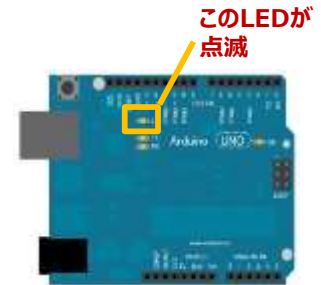


第4章 ソフトウェア編 (基礎)

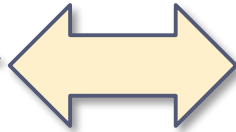
1. サンプル・スケッチを実行（1）

最も簡単で分かりやすいのは「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）



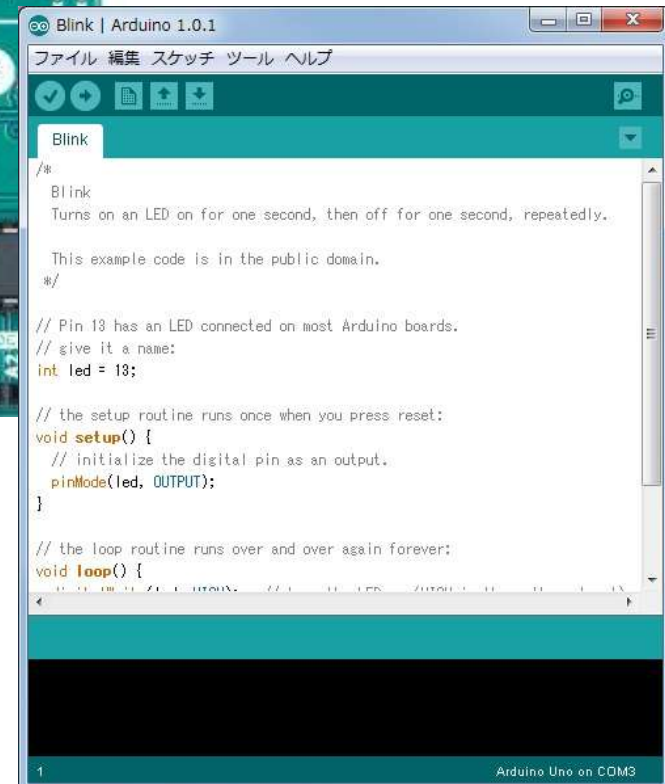
PCとの接続



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

1. 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

pinModeは
内部関数

「//」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 }

```

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

1000ms 1000ms 1000ms

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

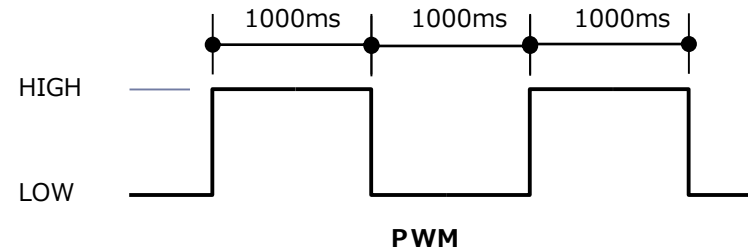
プログラムの動き

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
の繰り返しを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

2. スケッチの作成方法



```
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);
}
```

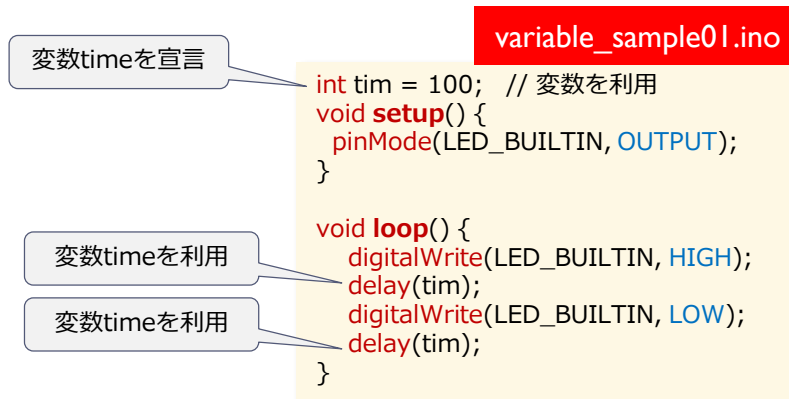
Blink06.ino

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。

3. プログラミングについて（変数を使ってみる）

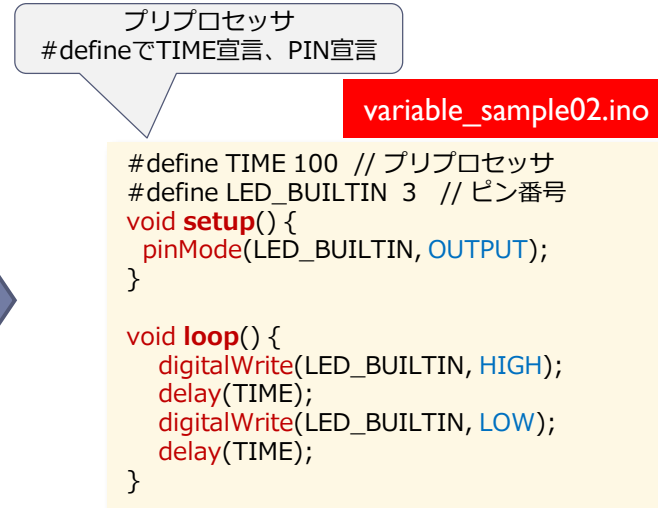
方法 1：一般的な変数利用



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 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。

3. 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

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。

どこをどう変更するか
注意深く見てください

【参考】 millis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。

3. 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。

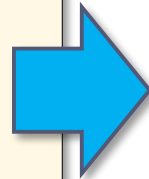
LED_sample_02.ino

```
#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。

3. 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。

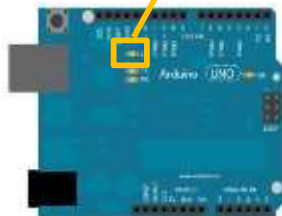
4. 自作スケッチの実行

- 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。
- コンパイルし、Arduinoに書き込んで実行させます。

自作スケッチを
この画面で入力し、
実行してみてください。

サンプルスケッチ「Blink」
と同じ動きをします。

このLEDが
点滅



```
Blink | Arduino 1.6.9
ファイル 編集 スケッチ ツール ヘルプ

Blink
10
16
17 // the setup function runs once when you press reset or power
18 void setup()
19 { pinMode(13,OUTPUT); }
20
21 void loop()
22 { static boolean sw=true;
23   digitalWrite(13,sw?HIGH:LOW);
24   delay(1000);
25   sw = !sw;
26 }
27
28
29 }
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2
```

5. 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、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。

今後はセンサの値をどう処理していくか、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きます。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。

6. 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す。

```
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

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

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に変わると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

```
COM123
送信
l+m= 14
m-n= 12
n*l= -15
l/m= 0
m%n= 0
l++= 5
l = 6
m--= 9
m = 8
☒ 自動スクロール CRおよびLF... 9600 bps 出力をクリア
```

6. 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。

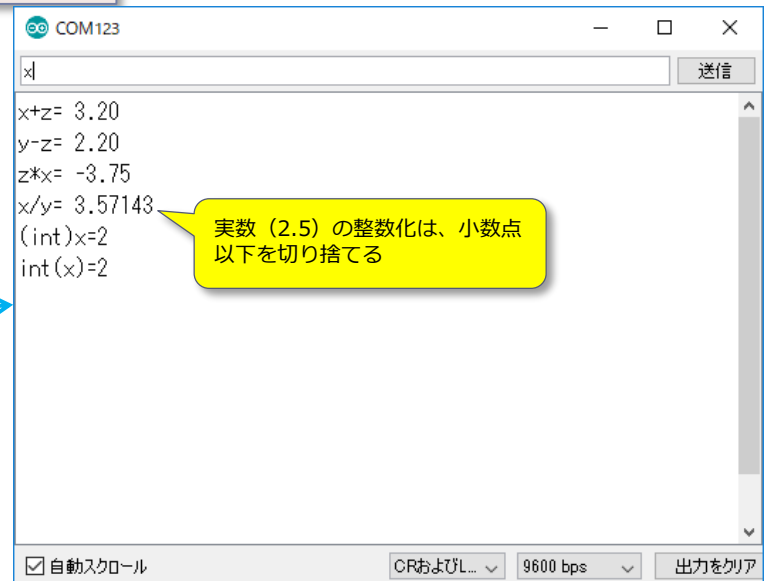
```
float x=2.5, y=0.7, z= -1.5;
void setup() {
  Serial.begin(9600);
  Serial.print("x+z= "); Serial.println(x+y); // 2.5+0.7=3.2
  Serial.print("y-z= "); Serial.println(y-z); // 0.7-(1.5)=2.2
  Serial.print("z*x= "); Serial.println(z*x); // -1.5*2.5=-3.75
  Serial.print("x/y= "); Serial.println(x/y,5);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

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

「(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。

Basic_calc_sample2.ino



6. 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合があります。文字処理では、クラスライブラリとしてString関数を使う処理が可能ですが、文字配列（型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

Basic_String_char_sample.ino

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

```
COM83 (Arduino/Genuino Uno)
substring = Node
length = 13
replace = Trillion Node
Trillion-Node
Trillion
x=23.024
```

送信

自動スクロール CRおよびLF 9600 bps 出力をクリア

6. 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() { }
```

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

```
---> if else
a /= b  && a <= b
---> while
a=0, b=4
a=1, b=4
a=2, b=4
a=3, b=4
---> do while
a=0, b=4
a=0, b=3
a=0, b=2
a=0, b=1
---> for
i = 0
i = 1
i = 2
i = 3
i = 4
```

Basic_control_sample1.ino

6. 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を行うことができます。
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。

```
COM35 (Arduino/Genuino 101)
8 data sort
before ->
175 170 160 168 165 173 155 150
after ->
150 155 160 165 168 170 173 175
```

「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、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(){}

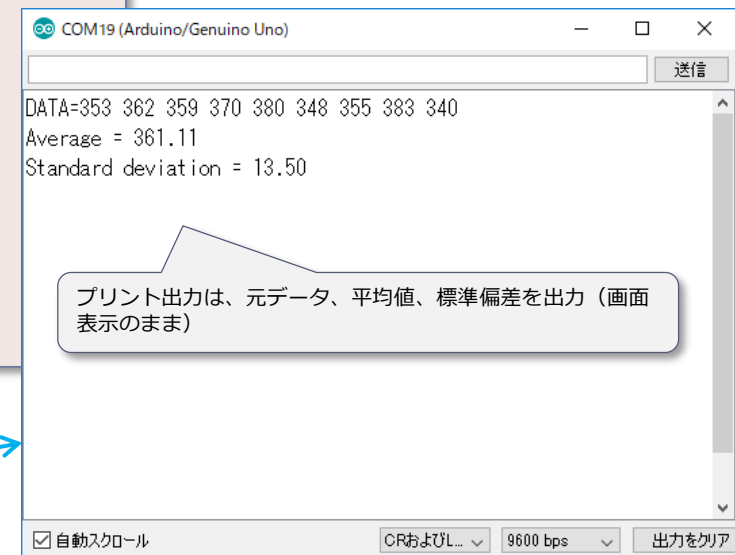
```

Basic_Ave_StDev.ino

【補足】「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(\sqrt{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。

【補足】

「a = a+1」は「a++」・「++a」と同じ
 ただし、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



6. サンプルスケッチ（7）

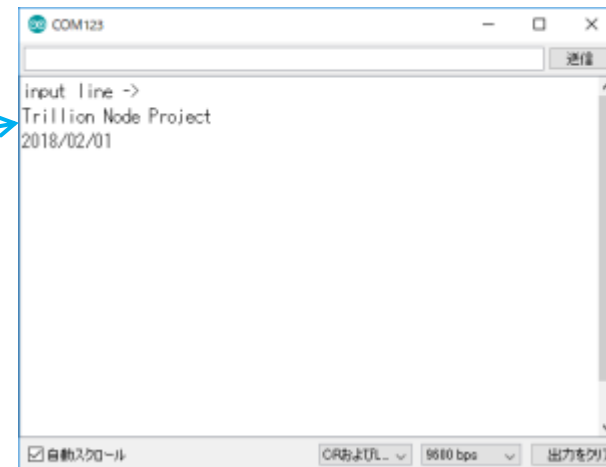
シリアルモニタ画面でのキーボード入力をそのまま画面に表示するスケッチです。
改行コードがあれば、それまでの一行を読み込んで、シリアル画面に表示します。

```
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



6. サンプルスケッチ（8）

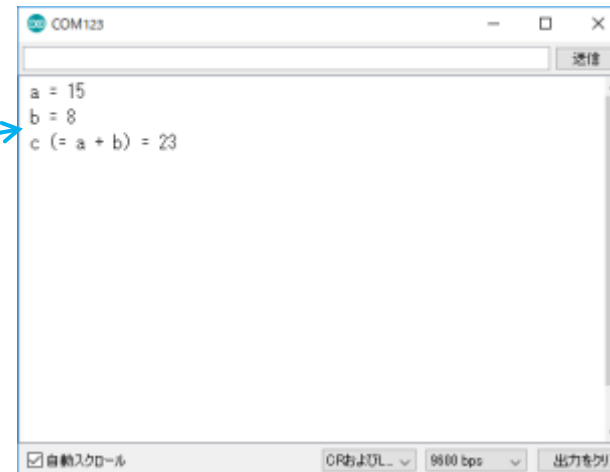
変数を使って、加算を行うもので、一連の流れをシリアルモニタ画面への表示を行います。

```
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



6. サンプルスケッチ（9）

入力された整数の二乗（ $n \times n$ ）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));  
}
```

【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

sample_1.3.ino

6. サンプルスケッチ（10）

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));  
}
```

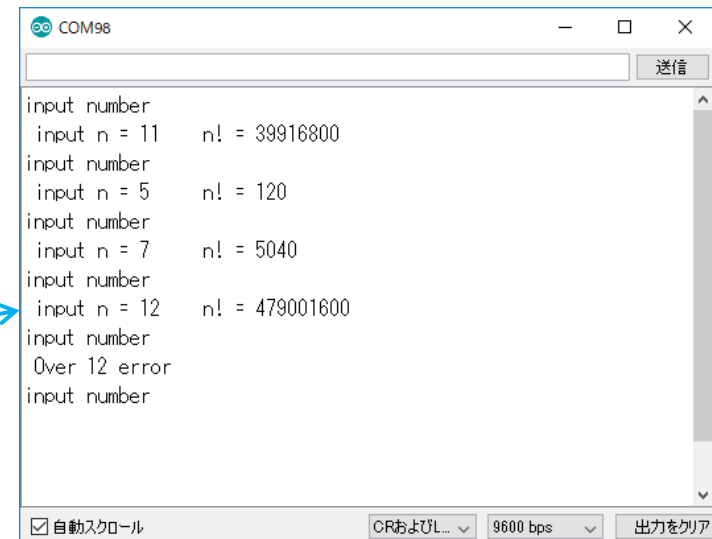
sample_1.4.ino

【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



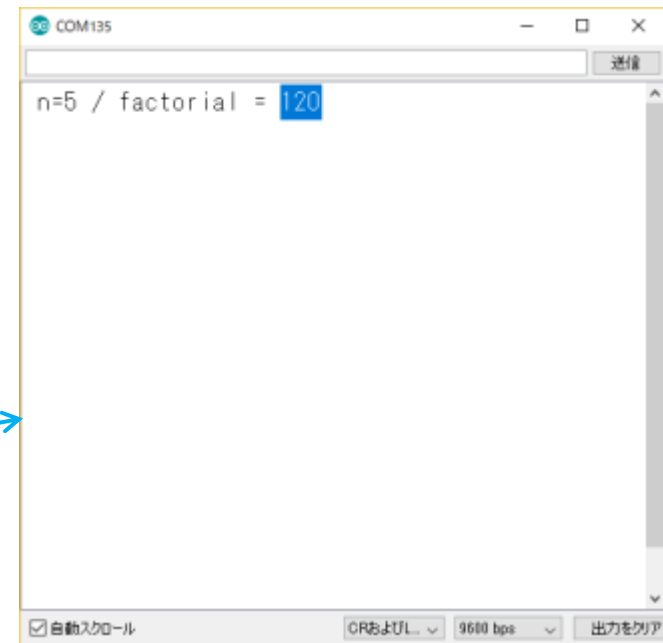
6. サンプルスケッチ（1 1）

ここでのプログラムは、階乗（ $n! = 1 \times 2 \times \dots \times n$ ）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



6. サンプルスケッチ（12）

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;  
}
```

sample_2.1.ino

Serial.print("-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

【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するときには見やすくなります。

5.0

-5.0



【演習】振幅100の余弦波（COS波）を描いてみましょう。

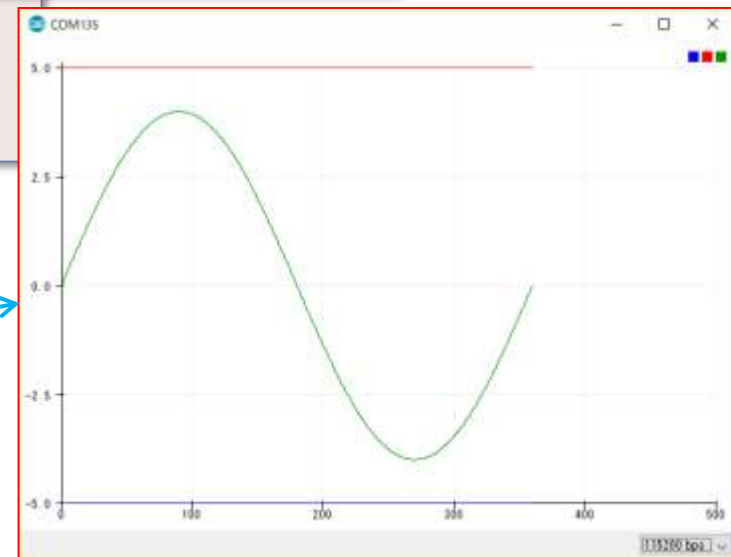
6. サンプルスケッチ（13）

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



[ブレイク] プログラミングで設計書が必要か？

- ▶ プログラミングに設計図は必要か？
 - ▶ 大きなプログラミング開発では必須。設計段階は、トップダウンで考えて行き、実際にプログラム（コーディング）では、ボトムアップで仕上げていく。
- ▶ 一般の設計では、どんな図面があるか？
 - ▶ What to make? と How to make? の両方の図面が必要となってくる。完成図を表した図面と、作る手順などを表す図面がある。建築では、基本設計図と施工図と呼んだりする。
- ▶ プログラミングに必要な仕様書とは？
 - ▶ 以前は、外部設計書（入出力部分を明確にしたもの）と内部設計書（処理部分を明確にしたもの）を作成していたが、最近では複雑なものとなりUMLなどを使って設計書をまとめている。

[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。

もくじ

1. 簡単にArduinoを学ぶフロー
 2. Arduinoの基本 (単体LED)
 3. Arduinoの基本 (圧電スピーカ)
 4. Arduinoの基本 (ジャンパワイヤ)
 5. Arduinoの基本 (チルトセンサ)
 6. Arduinoの基本 (タクトスイッチ)
 7. Arduinoの基本 (スライドスイッチ)
 8. Arduinoの基本 (温度センサ)
 9. Arduinoの基本 (光センサ)
 10. Arduinoの基本 (可変抵抗器)
 11. Arduinoの基本 (超音波距離センサ)
 12. Arduinoの拡張 (複数LED)
 13. Arduinoの拡張 (圧電スピーカ)
- 【ブレイク】抵抗値のカラー識別



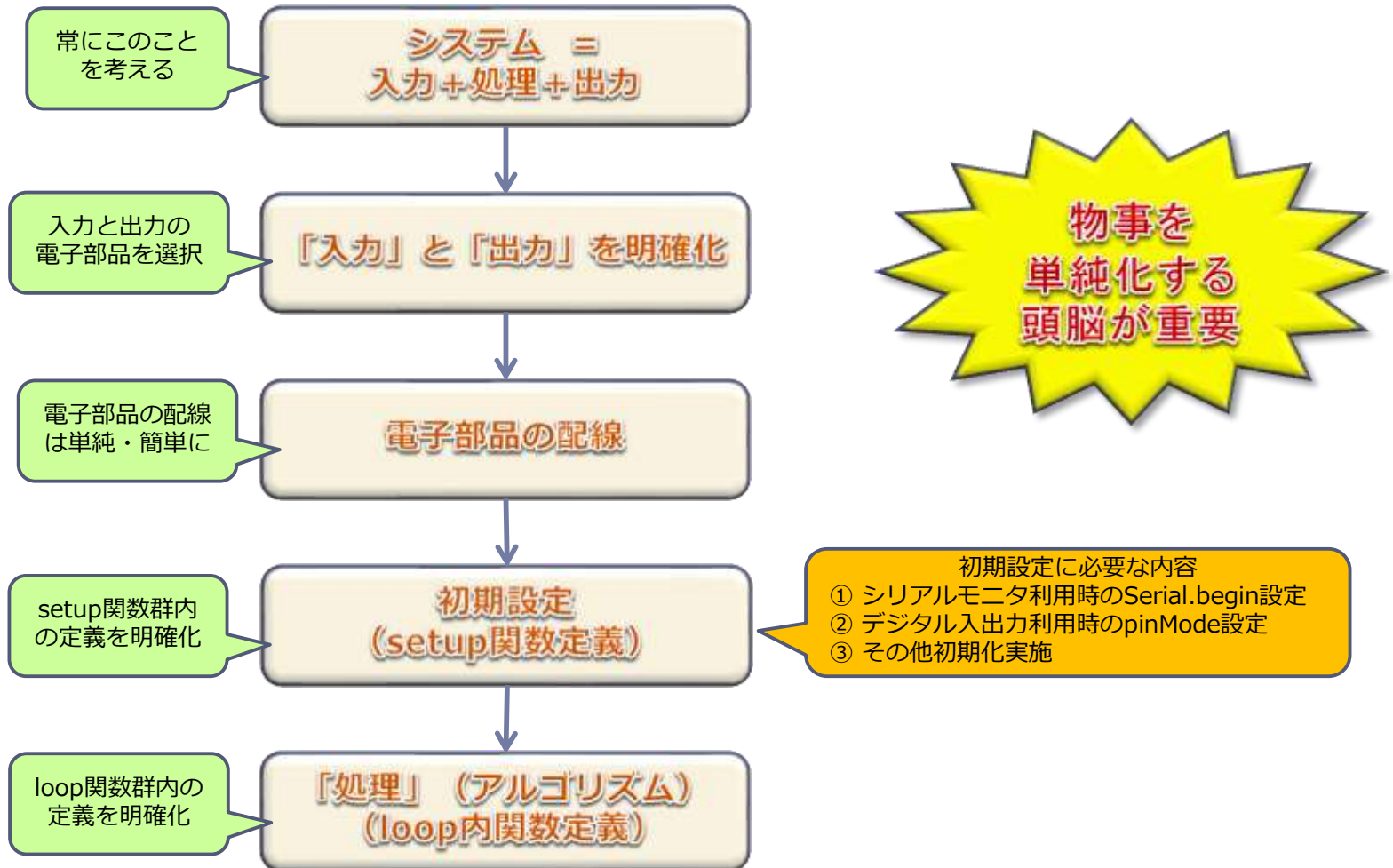
第5章 Arduinoの基礎演習

システムは、
「入力」+「処理」+「出力」
で構成される。

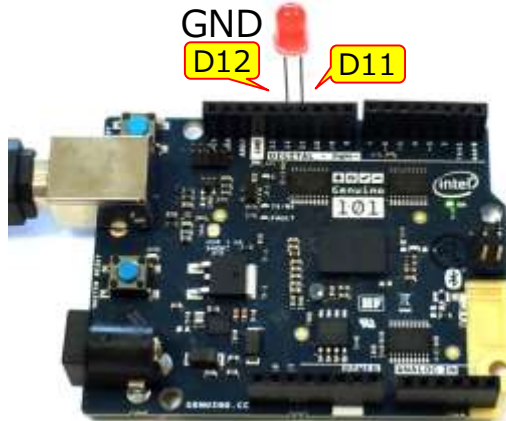
センサは「入力」部分となる



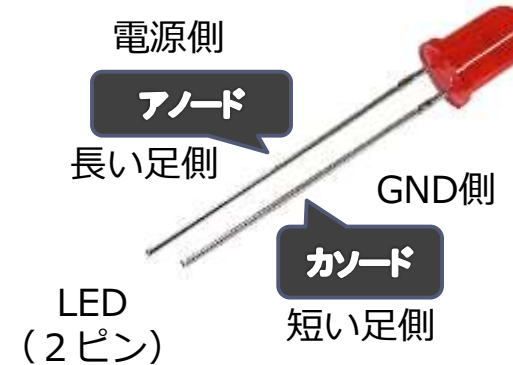
1. 簡単にArduinoを学ぶフロー



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

【補足注意】 ここで使っているLEDは、抵抗入りのLEDを使っています。一般に市販されている抵抗なしのLEDを直接ポートに差し込むことは、LEDが壊れる原因となります。

2. Arduinoの基本（単体LED ②）

Basic_LED_01.ino

【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);
  }
}
```

繰り返し（0~255）
・段々と明るく

Basic_LED_02.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)

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

Basic_LED_03.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);
}
```

D11 にアノード (電源)
D12 にカソード (GND)

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

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

※Geunino101では、D10/D11は未対応

※Arduino M0 Proでは、D2～D13が対応

▶ アナログ出力関数

`analogWrite(ain,val);`

ここで 書き込み値 val = 0～255

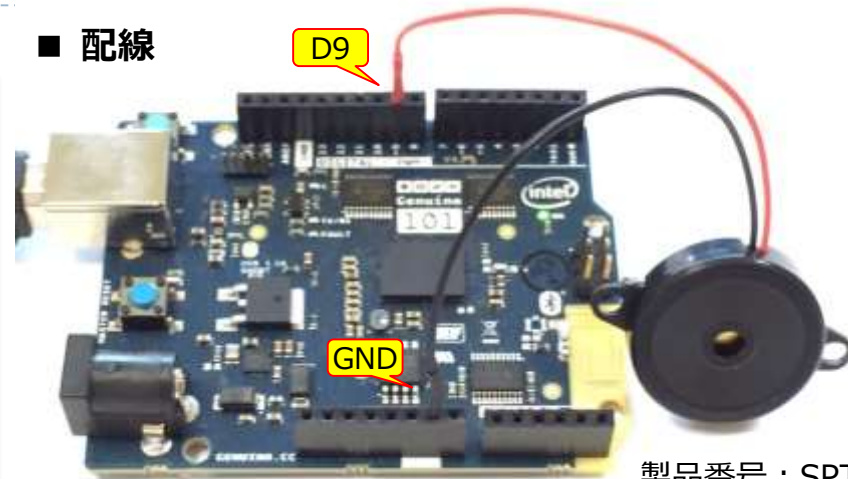
■ 事例

Basic_Speaker.ino

```
void setup() { }
void loop() {
  analogWrite(9,157);
  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
(ヒント: 同じ周期で音を発生させる)

■ 配線



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

**IoTABシールド
圧電スピーカ : D10**

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

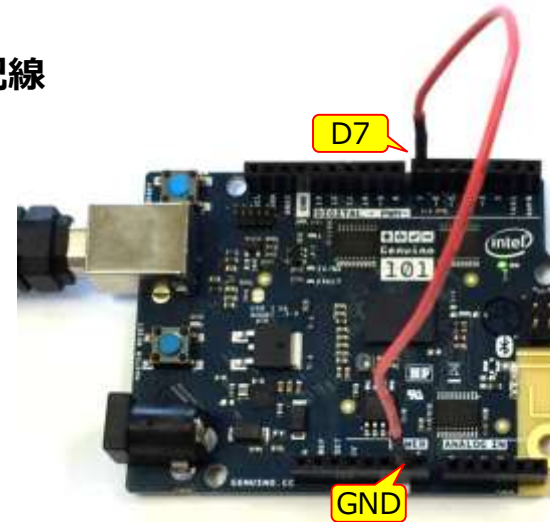
```
pinMode(din, INPUT/INPUT_PULLUP);  
digitalWrite(ain, HIGH/LOW);
```

■ 事例

Basic_JumperWire.ino

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

■ 配線



ケーブル（スイッチ）
GND-D7

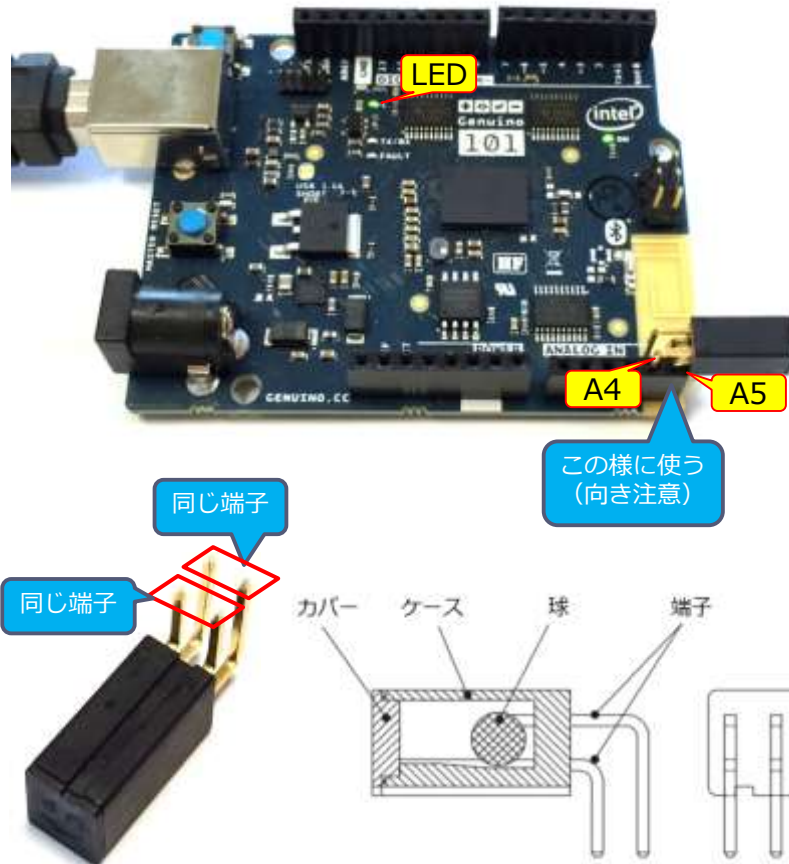
■ 注意点

- ① スwitchなどは**プルアップ抵抗（INPUT_PULLUP）**を考慮
- ② Offの状態は「HIGH」で、
Onの状態は「LOW」となる。

IoTABシールド
タクトスイッチ：D2

5. Arduinoの基本（チルトセンサ）

■ 配線



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
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

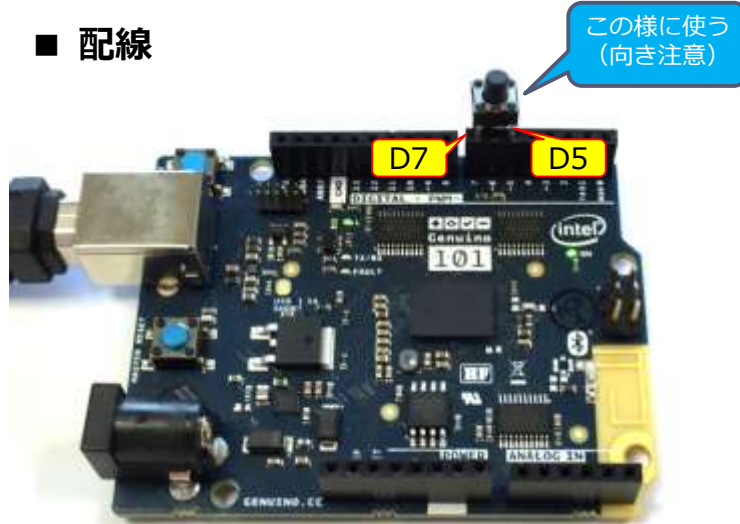
ヒント
「!」を利用

Basic_Tilt.ino

センサ値でLEDを点滅させる

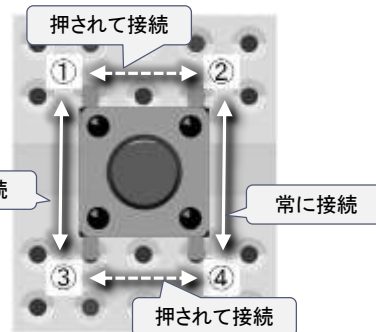
6. Arduinoの基本（タクトスイッチ①）

■ 配線



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
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

Basic_TactSwitch_01.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）

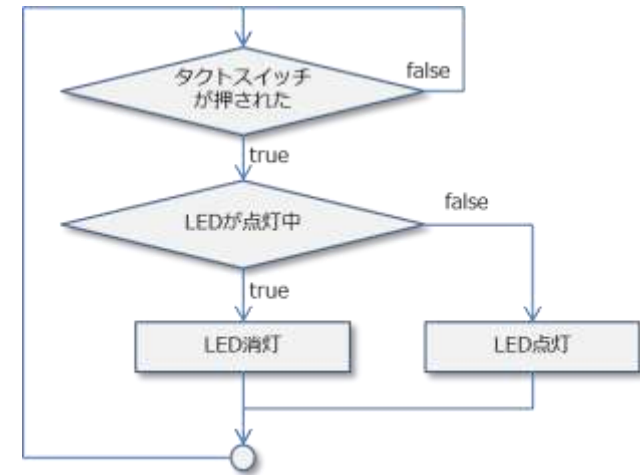
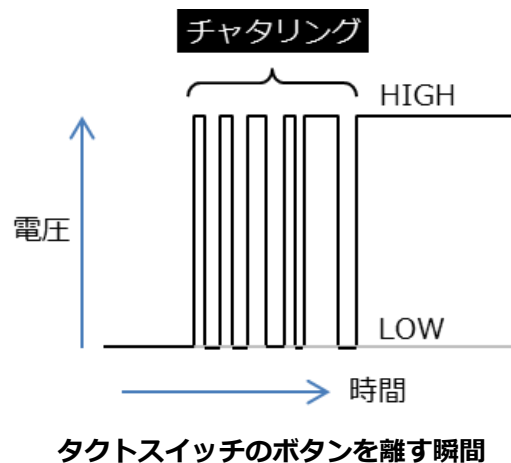
チャタリング(chattering)について

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



Basic_TactSwitch_02.ino

```
// 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;
}
```

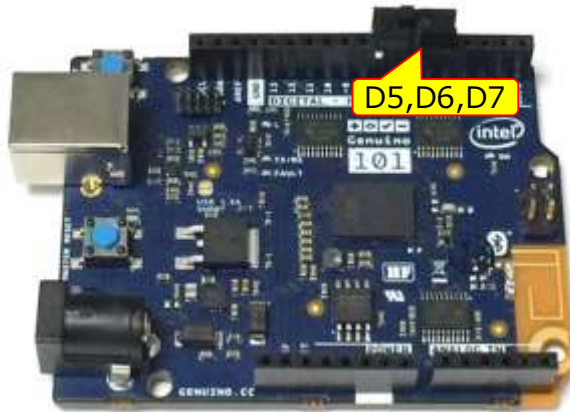
タクトスイッチとLEDのpinMode設定

チャタリングを考慮したLED点灯

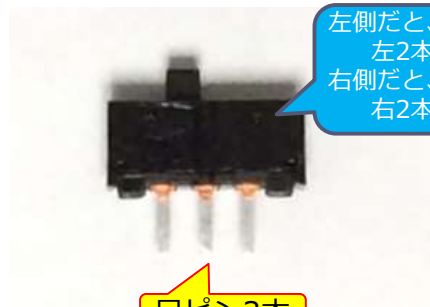
チャタリング考慮
(時間を変更してみよう)

7. Arduinoの基本（スライドスイッチ）

■ 配線



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

```
// 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

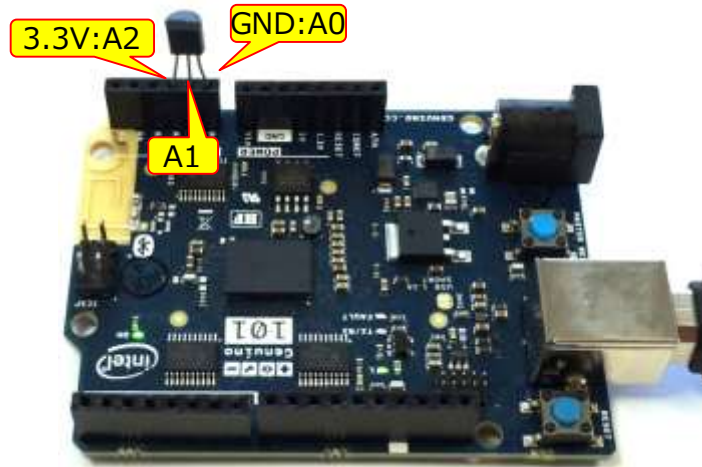
Basic_SlideSwitch.ino

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

8. Arduinoの基本（温度センサ）

■ 配線

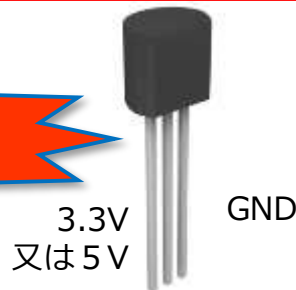


温度センサ（LM61BIZ）
A0(GND), A1(Vout), A2(5V or 3.3V)

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
＜変換式が異なる＞



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

```
float tmp () { // 温度センサ値出力関数
  return(analogRead(A1)/1023.0*500 - 60);
  // return (analogRead(A1)/1023.0*330 - 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

Basic_Temperature.ino

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

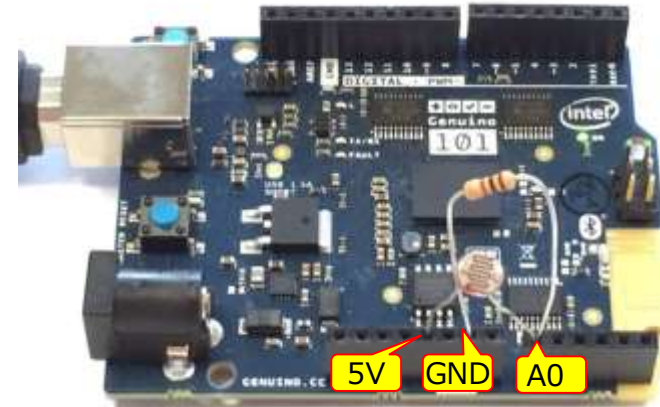
▶ アナログ入力の接続ポート

- ▶ 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`

■ 配線



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

■ 事例

```
void setup() {  
  Serial.begin(9600);  
}  
void loop() {  
  Serial.println(analogRead(A0));  
  delay(10);  
}
```

シリアル通信出力の初期化（ボーレート設定）

光センサ値をシリアルモニタ画面または
シリアルプロッタ画面に表示

Basic_Light.ino

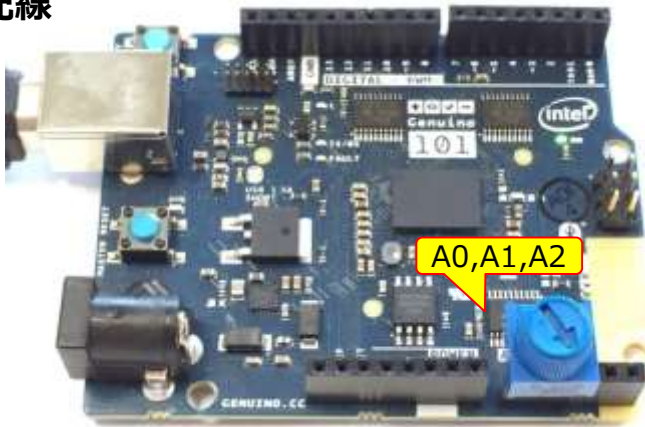
■ 注意点

- ① 抵抗と光センサのピンを同じA0に挿し込む
- ② シリアルモニタまたは**シリアルプロッタ**画面に出力

IoTABシールド
光センサ : A0

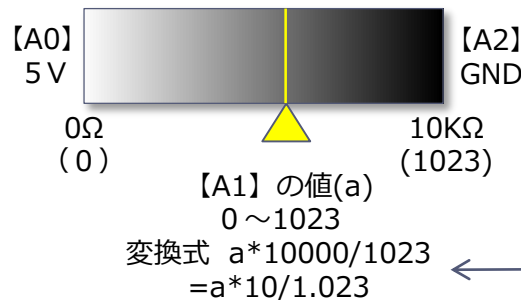
10. Arduinoの基本（可変抵抗器）

■ 配線



可変抵抗器（10KΩ）
A0,A1,A2

製品番号：TSR-3386U



何故
 $a \cdot 10000 / 1023$
ではだめか

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

【注意】
ピンの接触が
悪いですよ

IoTABシールド
可変抵抗器：A3

```
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);
}
```

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

Basic_Resistance.ino

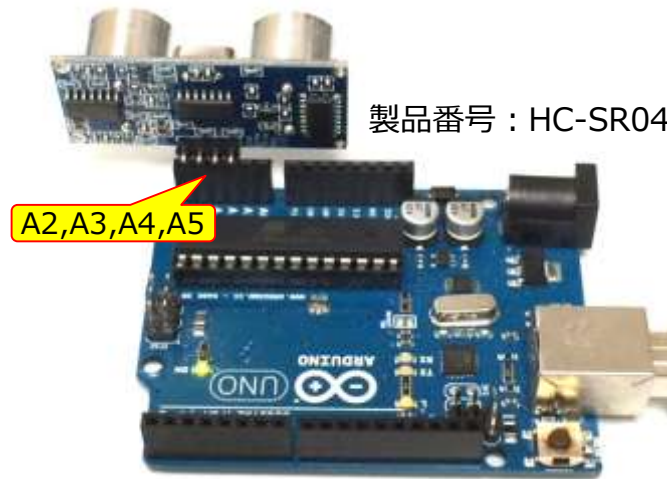
0.3秒ごとに可変抵抗器の値を
シリアルモニタに表示

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）

ArduinoUNO（5V系）では、超音波距離センサは、ポートに直接ピンを差し込む「直挿し」ができる。



製品番号：HC-SR04

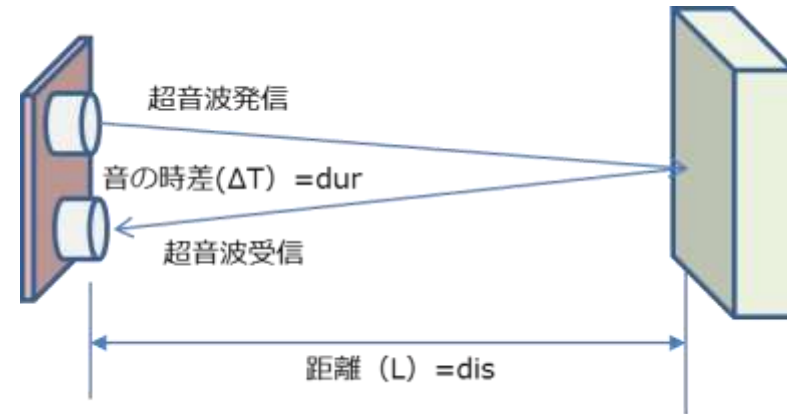
A2,A3,A4,A5

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

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**

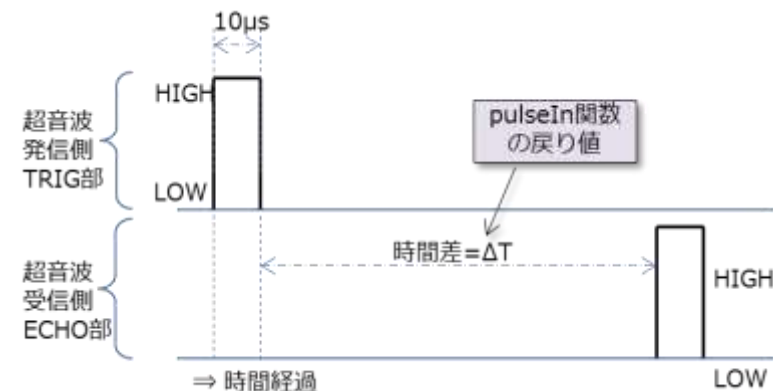


$$L = C \times \Delta T / 2$$

ここで、Lは、障害物までの距離

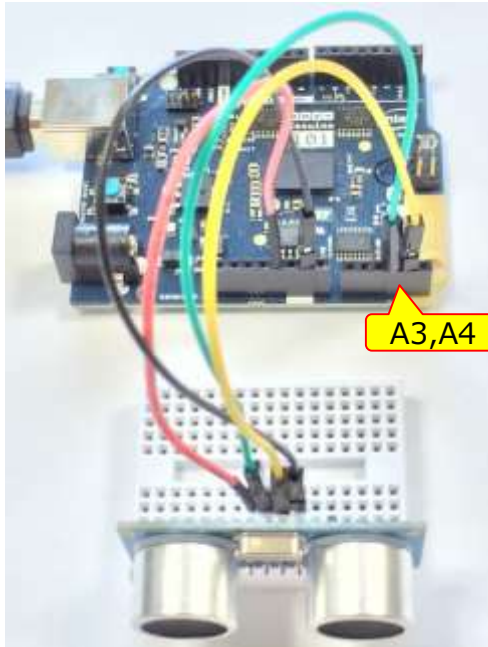
Cは、音速（概算簡易式は $331 + 0.6 \times t$: 単位m/s）

ΔTは、超音波の発信から受信までの時間差



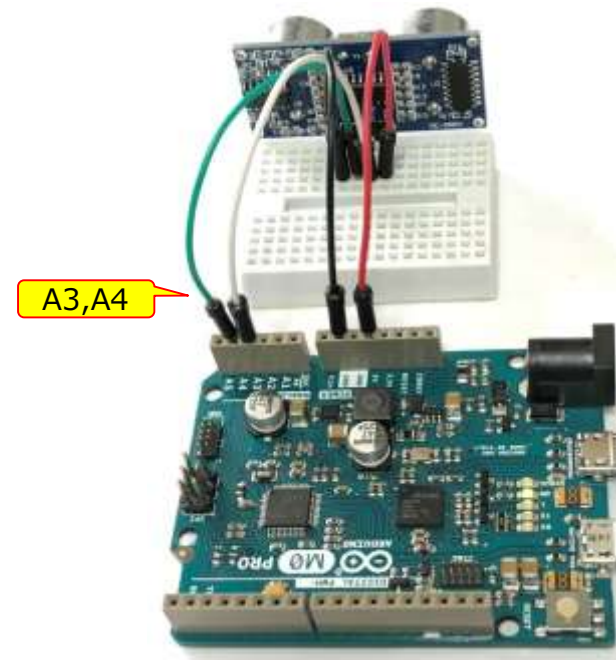
11. Arduinoの基本（超音波距離センサ②）

■ 配線（Geunino101の場合）



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

■ 配線（Arduino M0 Proの場合）



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

11. Arduinoの基本（超音波距離センサ③）

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）

The screenshot shows the Arduino IDE interface. On the left, the serial monitor displays a list of distance measurements in centimeters. On the right, the sketch editor shows the code for Basic_Distance.ino. Two callout boxes highlight specific parts of the code: 'ポート設定' (Port Setting) points to the setup() function, and '時間差取出し' (Time Difference Extraction) points to the pulseIn function call in the loop().

```
void setup() {  
  pinMode(17,OUTPUT); // Trig  
  pinMode(18,INPUT);  // Echo  
  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);  
}
```

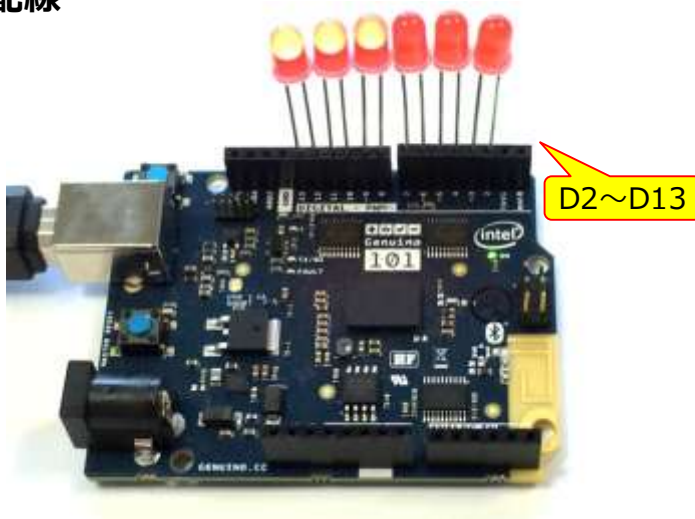
70.50
78.86
80.82
74.44
80.02
80.44
80.02
80.02
50.97
79.73
79.87
80.63
81.87
80.24
78.95
80.07
50.73

自動スクロール 改行なし 9600 baud

Basic_Distance.ino

12. Arduinoの拡張（複数LED ①）

■ 配線



6 個のLED
 アノード側（長い足）
 D2,D4,D6,D8,D10,D12
 カソード側（短い足）
 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

IoTABシールド
LED : D3 - D8

※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

1 2. 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：

```
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;
}
```

Combined_LEDs_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);
  }
}
```

Combined_LEDs_02.ino

6個のLEDのアノードを
デジタル出力（GND）宣言

課題3： 下のスケッチはどんな動き
をするでしょう？

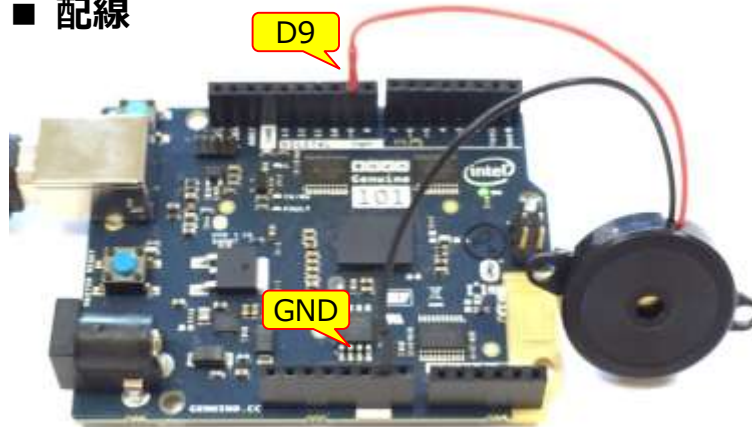
```
void setup(){
  for(int i=2; i<14; i++) {
    pinMode(i,OUTPUT);
    digitalWrite(i,LOW);
  }
}
void loop() {
  for( int i=2; i<14; i=i+2) {
    digitalWrite(i,HIGH);
    delay(50);
  }
  for( int i=12; i>1; i=i-2){
    digitalWrite(i,LOW);
    delay(50);
  }
  for( int i=12; i>1; i=i-2) {
    digitalWrite(i,HIGH);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

Combined_LEDs_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）

■ 配線



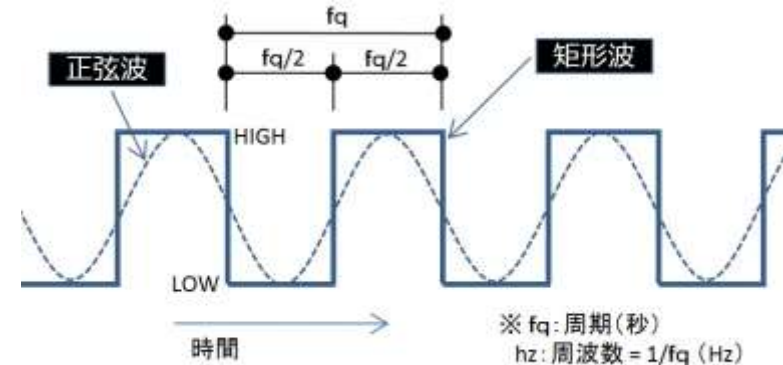
製品番号：SPT08

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
圧電スピーカ：D9

13. Arduinoの拡張（圧電スピーカ ②）

■課題1：デジタル出力による音発生

pinModeによる初期設定

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

0.01秒ごとにHIGH/LOWの振動で音を発生

■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

■課題3：デジタル出力（tone関数利用）

pinModeによる初期設定

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

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

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。

■課題4：tone関数を作ってみよう

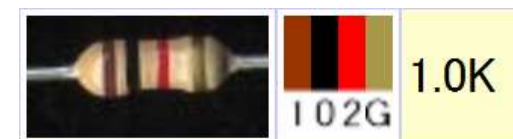
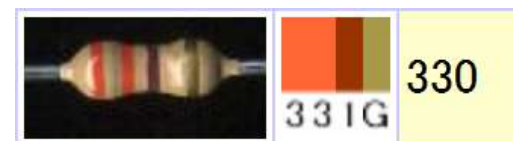
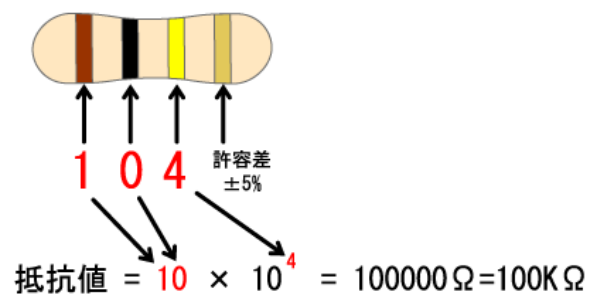
課題5：チューリップ^o（メロディ）を作ってみよう

The image displays three staves of musical notation for the song "The Rose Tree" in G major, 3/4 time. The first staff contains the melody, the second the alto part, and the third the bass part. Each staff has a 'V' marking above the final measure, indicating a vocal entry point.

【ブレイク】 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



もくじ

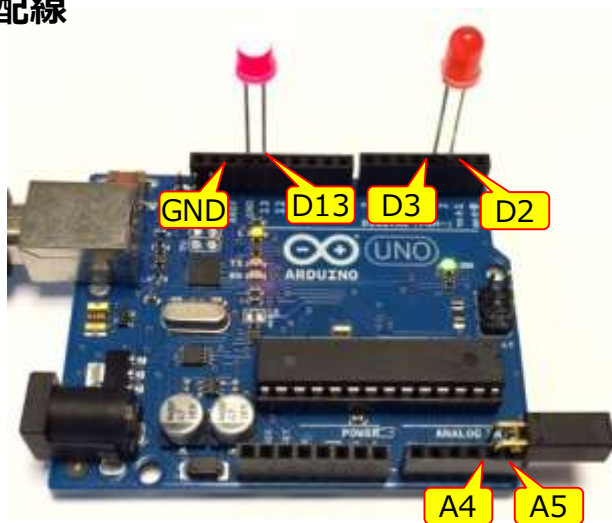
- 課題 1. チルトセンサとLED
- 課題 2. LED6個とスライドスイッチ
- 課題 3. LED6個とリミットスイッチ
- 課題 4. 光センサとLED6個
- 課題 5. 可変抵抗器とスピーカ
- 課題 6. LED6個と超音波距離センサ
- 課題 7. LED5個・スピーカ・距離センサ
- 課題 8. タイマーを作る



第6章 Arduinoの拡張演習

課題 1 : チルトセンサとLED

■ 配線



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：

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

```
// 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);
```

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

```
}
void loop(){
  digitalWrite(13,digitalRead(19));
  digitalWrite(2,!digitalRead(19));
}
```

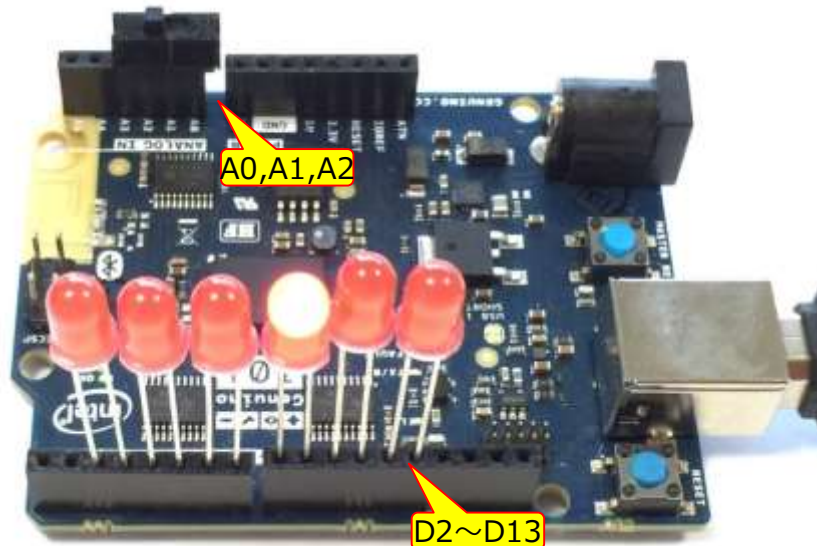
チルトセンサで、
一方のLED点灯、もう一方を消灯

Combined_LEDs_Tilt.ino

**IoTABシールド
LED : D3 - D8**

課題 2 : LED6個とスライドスイッチ

■ 配線



- 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

```
// 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;
}
```

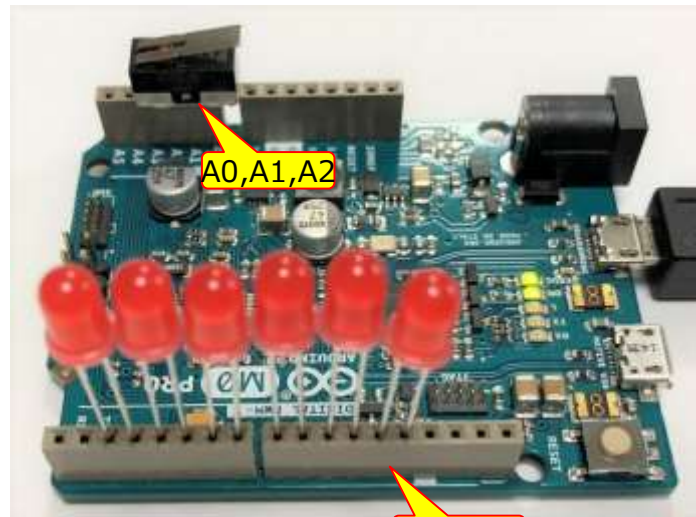
- 1) 6個のLED初期設定
- 2) スライドスイッチ初期設定

LED点滅が流れるようにするには？

Combined_LEDs_SlideSwitch.ino

課題3 : LED6個とリミットスイッチ

■ 配線



- 1) 6個のLED (D2-D13)
- 2) リミットスイッチ (A0,A2,A4)

**IoTABシールド
LED : D3 - D8**

課題 : リミットスイッチの設置に応じて、
6個のLEDを順次流れるように点滅させる

```
// A2-A4 SW-set
// D2-D13 LED-set
void setup() {
  pinMode(A2, OUTPUT);
  pinMode(A4, INPUT_PULLUP);
  for (int i = 2; i < 14; i++) {
    pinMode(i, OUTPUT);digitalWrite(I,LOW);
  }
}

void loop() {
  if (digitalRead(A4)) {
    for (i = 2; i < 14; i += 2) {
      digitalWrite(i, HIGH); delay(100);
      digitalWrite(i, LOW);
    }
  } else {
    for (i = 12; i > 1; i -= 2) {
      digitalWrite(i, HIGH); delay(100);
      digitalWrite(i, LOW);
    }
  }
}
```

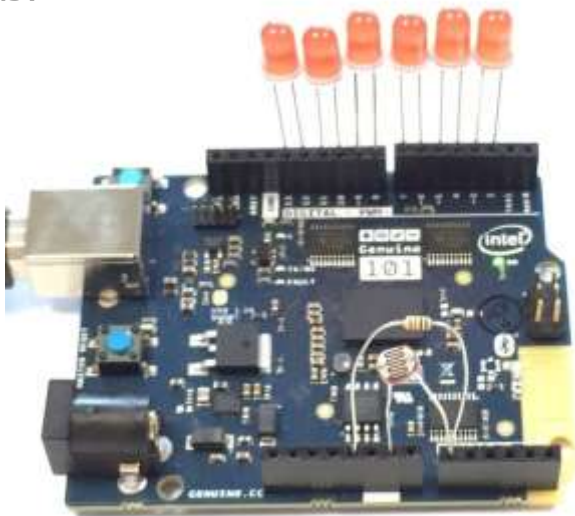
- 1) リミットスイッチ初期設定
- 2) 6個のLED初期設定

LED点滅が流れるようにするには？

Combined_LEDs_LimitSwitch.ino

課題4：光センサとLED 6個

■ 配線



- 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 : A0

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)

```
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));
  }
}
```

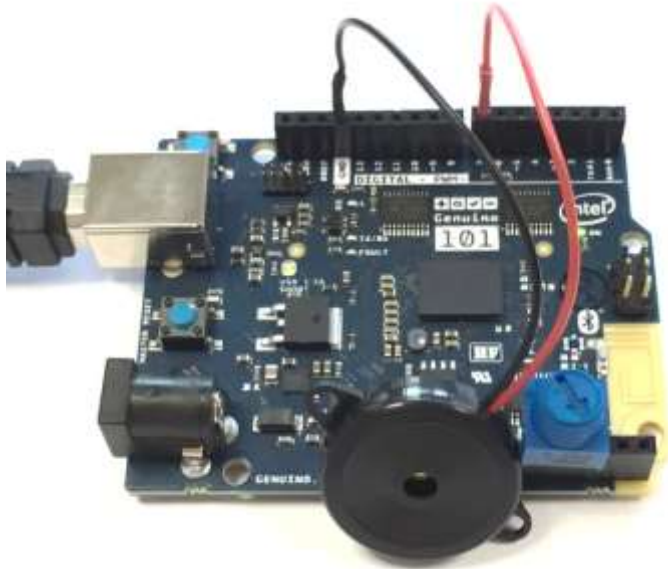
6個のLED初期設定

Combined_LEDs_Light.ino

光センサの値に応じてLEDを点灯

課題 5 : 可変抵抗器とスピーカ

■ 配線



- 1) スピーカ D7 GND
- 2) 可変抵抗器 A0,A1,A2

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。

- 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);  
}
```

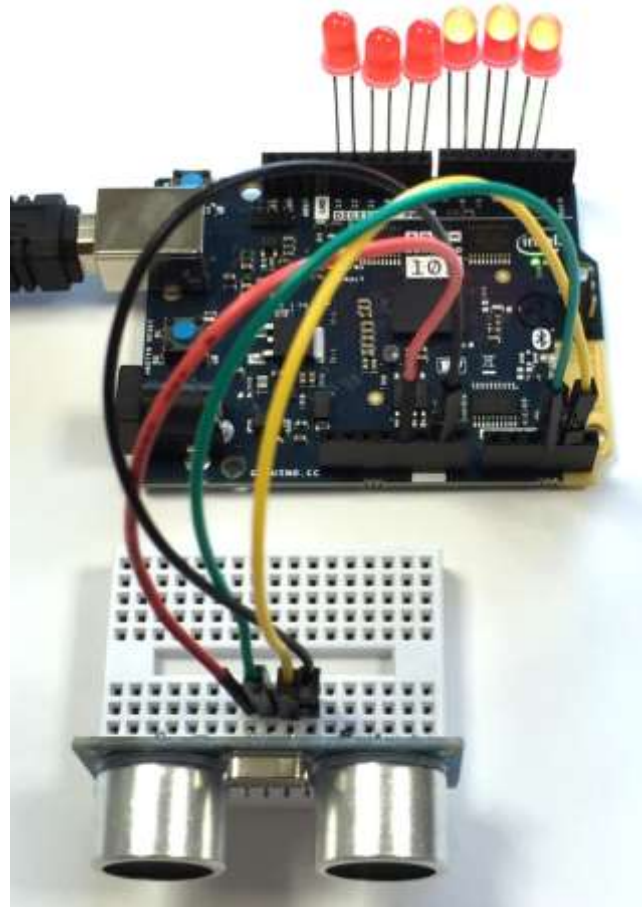
Combined_Speaker_Resistance.ino

抵抗に応じて高低差を付けた音発生

IoTABシールド
圧電スピーカ : D10
可変抵抗器 : A3

課題 6 : LED6個と超音波距離センサ

■ 配線



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

課題： 距離センサの値に応じて、6個のLEDを点滅させる

- 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); digitalWrite(i,LOW);}
  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);}
}
```

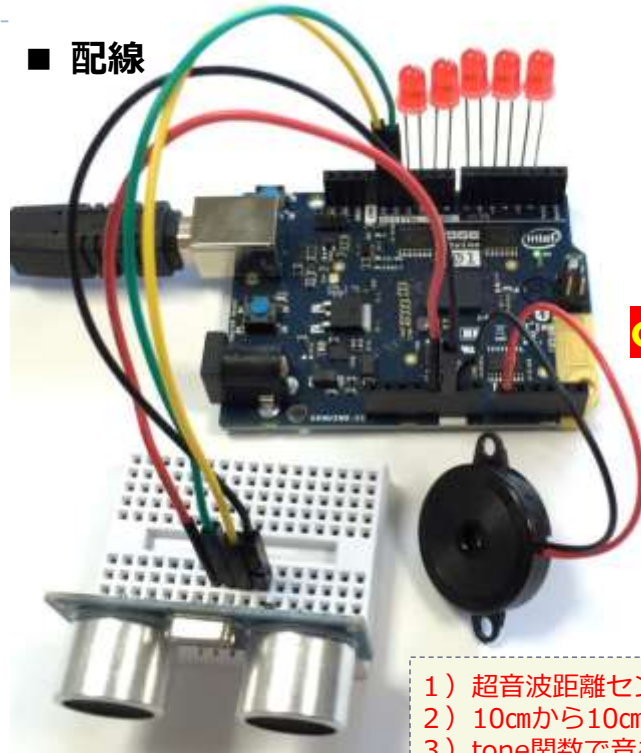
超音波距離センサにより距離を取出し
LEDの点滅を制御する

Combined_LEDs_Distance.ino

IoTABシールド
 超音波距離センサ : D12,D13
 LED : D3 - D8

課題7：LED5個・スピーカ・距離センサ

■ 配線



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

IoTABシールド
 超音波距離センサ : D12,D13
 圧電スピーカ : D10
 LED : D3 - D8

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

Combined_LEDs_Distance_Speaker.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

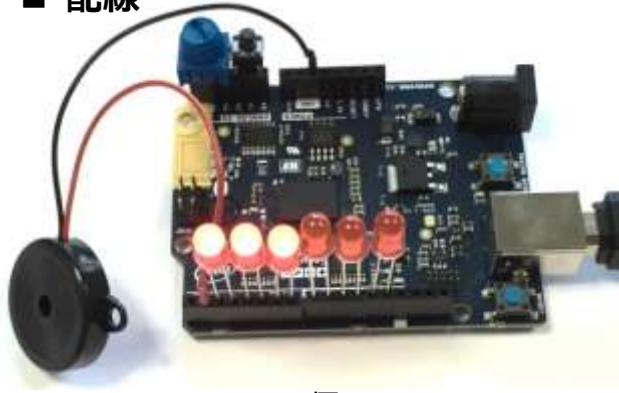
```
void setup(){
  for(int i=1; i<11; i++) {
    pinMode(i,OUTPUT); digitalWrite(i,LOW);
  };
  pinMode(11,OUTPUT); digitalWrite(11,HIGH);
  pinMode(12,OUTPUT);
  pinMode(13,INPUT);
  pinMode(A0,OUTPUT);
}

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抽出し
- 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 8 : タイマーを作る

■ 配線



- 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

- 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

```
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)
}
```

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。

Combined_Timer.ino

```
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) 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

- 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

もくじ

1. Genuino101 専用機能とは
2. 慣性センサ利用
3. 温度センサ利用
4. BLE機能
5. EEPROM機能
6. RTC（リアルタイムクロック）機能

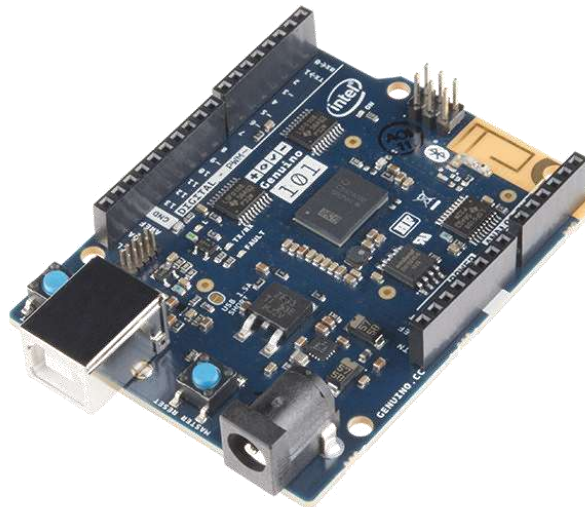


第7章 Genuino101専用機能

1. Genuino101 専用機能とは

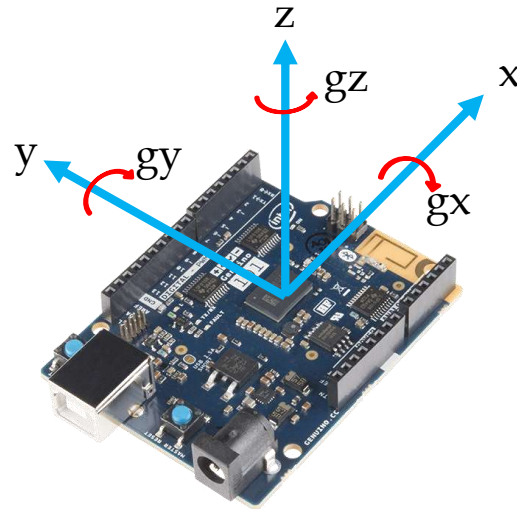
Genuino101は、以下の機能を持ち合わせています。これらの機能は、参照ライブラリを使って利用するようになっています。

	Genuino101機能	内 容	参照ライブラリ
1	慣性センサ	3軸加速度 + 3軸ジャイロセンサ	CurieIMU.h
2	温度センサ	float : CurieIMU.readTemperature()/32767.0 + 23.0;	CurieIMU.h
3	BLE	BlueTooth Low Enagy機能	CurieBLE.h
4	EEPROM	2048バイト（2 Kバイト）メモリ	EEPROM.h
5	リアルタイムクロック	日時の設定・読込	CurieTime.h



2. 慣性センサ利用①

Genuino101は、以下の図の方向が正方向となる3軸加速度センサと3軸ジャイロセンサ、計6軸の慣性センサを持ち合わせています。



加速度センサは、最大値を2G、4G、8G、16Gの重力加速度の倍数で値を読み取ることができます。これらは、設定によって切り替えができます。

ジャイロセンサは、角速度として1秒間の角度測定の設定を、25、50、100、200、400、800、1600、3200（度/秒）まで設定によって切り替えができます。

Genuino101では、加速度センサによる特性ごとによるサンプルをいろいろと用意しています。

クラスライブラリとしては、約40近く用意されています。

加速度センサやジャイロセンサは、技術的に高度なレベルでの利用ができます。

加速度センサは、衝撃や地震動、傾きなどに利用できます。このことで、人の動き、地盤の動き、斜面の動き、などなど多くの場面での利用ができます。

ジャイロセンサは、ロボットの動きなどでの角速度が取得でき、さらに角速度を角度に変換することで、傾きや回転角度を求めることができます。ただし、角速度による角度の積分の計算では、誤差が生じることから、誤差（ドリフト）が出ない計算方法が必要となります。

2. 慣性センサ利用②

加速度センサの取得について

```
#include "CurieIMU.h"

void setup() {
  Serial.begin(9600); // initialize Serial communication
  while (!Serial); // wait for the serial port to open

  // initialize device
  Serial.println("Initializing IMU device...");
  CurieIMU.begin();

  // Set the accelerometer range to 2G
  CurieIMU.setAccelerometerRange(2);
}

void loop() {
  float ax, ay, az; //scaled accelerometer values

  // read accelerometer measurements from device, scaled to the configured range
  CurieIMU.readAccelerometerScaled(ax, ay, az);

  // display tab-separated accelerometer x/y/z values
  Serial.print("a:");
  Serial.print(ax);
  Serial.print("\t");
  Serial.print(ay);
  Serial.print("\t");
  Serial.print(az);
  Serial.println();
}
```

CurieIMU_Acc_sample.ino

6軸加速度・ジャイロセンサの値の取得について

■ 宣言

```
#include "CurieIMU.h"
```

■ 初期設定

- `CurieIMU.begin();`
- `CurieIMU.setAccelerometerRange(2);`

※引数は、最大振動値で2 G、4 G、8 G、16 Gのいずれか

■ センサ値取得関数

- `CurieIMU.readAccelerometerScaled(ax, ay, az);`

2. 慣性センサ利用③

ジャイロセンサの取得について

```
#include "CurieIMU.h"

void setup() {
  Serial.begin(9600); // initialize Serial communication
  while (!Serial); // wait for the serial port to open

  // initialize device
  Serial.println("Initializing IMU device...");
  CurieIMU.begin();

  // Set the accelerometer range to 250 degrees/second
  CurieIMU.setGyroRange(250);
}

void loop() {
  float gx, gy, gz; //scaled Gyro values

  // read gyro measurements from device, scaled to the configured range
  CurieIMU.readGyroScaled(gx, gy, gz);

  // display tab-separated gyro x/y/z values
  Serial.print("g:");
  Serial.print(gx);
  Serial.print("\t");
  Serial.print(gy);
  Serial.print("\t");
  Serial.print(gz);
  Serial.println();
}
```

CurieIMU_Gyro_sample.ino

■宣言

#include "CurieIMU.h"

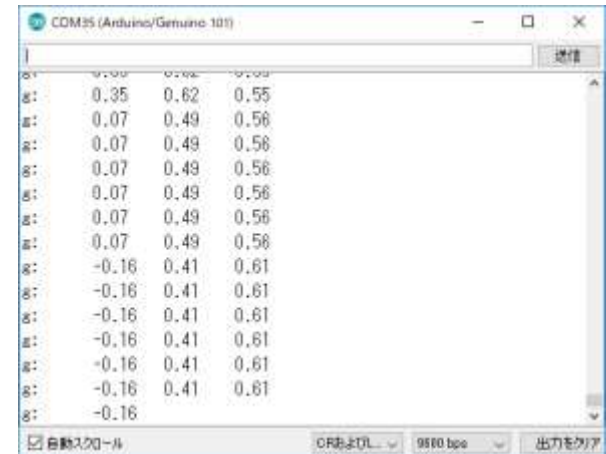
■初期設定

- CurieIMU.begin();
- CurieIMU.setGyroRange(250);

※引数は、毎秒事の最大角度（最大角速度）は、25、50、100、200、400、800、1600、3200度/秒

■センサ値取得関数

- CurieIMU.readGyroScaled(gx, gy, gz);



3. 温度センサ利用

Genuino101は、温度センサを持つ、その取得について

```
#include "CurieIMU.h"

void setup() {
  Serial.begin(9600); // initialize Serial communication
  while (!Serial);   // wait for the serial port to open

  // initialize device
  Serial.println("Initializing IMU device...");
  CurieIMU.begin();

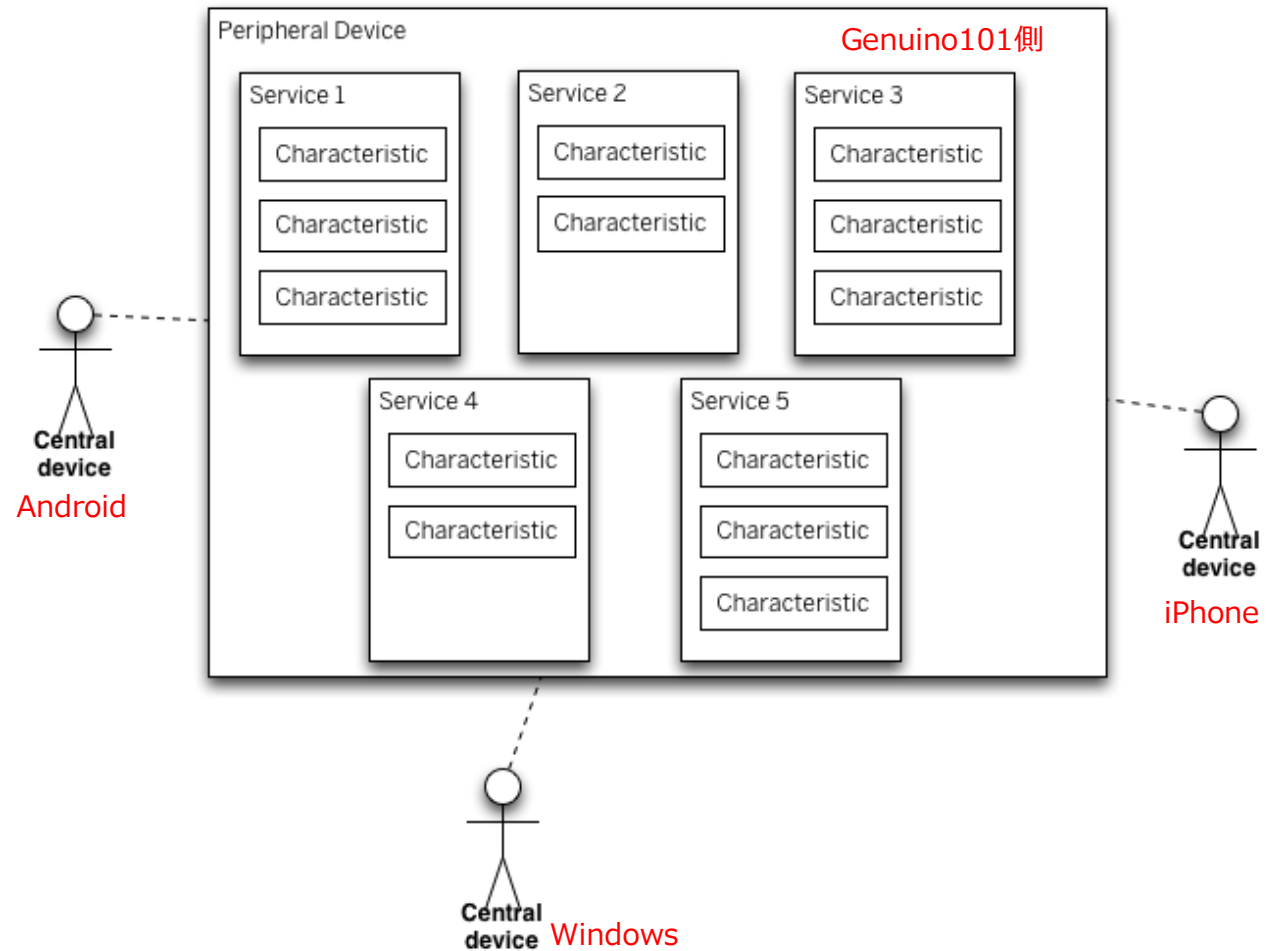
  void loop() {
    float val = CurieIMU.readTemperature()/32767.0 + 23.0;
    Serial.println(" Temp = " + String(val) + " C");
    delay(1000);
  }
```

CurieIMU_Tempertue.ino

- 宣言
#include "CurieIMU.h"
- 初期設定
 - CurieIMU.begin();
- センサ値取得関数
 - CurieIMU.readTemperature()

4. BLE機能①

BLEの設定について



4. BLE機能② BLEクラス（一部参考）

参考 : <https://www.arduino.cc/en/Reference/CurieBLE>

BLEPeripheral Class	処理内容	戻り値
<i>yourBlePeripheralName</i> .begin()	BLE初期化	true: 初期化完了
<i>yourBlePeripheralName</i> .poll()		
<i>yourBlePeripheralName</i> .end()	セントラルとの接続終了	
<i>yourBlePeripheralName</i> .setAdvertisedServiceUuid(advertisedServiceUuid)	: TBD	
<i>yourBlePeripheralName</i> .setLocalName(localName)	ローカル名設定	
<i>yourBlePeripheralName</i> .setDeviceName(deviceName)	デバイス名設定	
<i>yourBlePeripheralName</i> .setAppearance(appearance)	: TBD	
<i>yourBlePeripheralName</i> .setEventHandler(attributeName)		
<i>yourBlePeripheralName</i> .setAttribute(attributeName)	ペリフェラルに属性追加	
<i>yourBlePeripheralName</i> .disconnect()	セントラルとの通信終了	
<i>yourBlePeripheralName</i> .central()	セントラルとの通信チェック	
<i>yourBlePeripheralName</i> .connected()	セントラルとの通信状態確認	true : 通信状態、false:遮断状態

4. BLE機能③

BLEサービス

SpecificationName	SpecificationType	AssignedNumber	内 容
Alert Notification Service	org.bluetooth.service.alert_notification	0x1811	
Automation IO	org.bluetooth.service.automation_io	0x1815	
Battery Service	org.bluetooth.service.battery_service	0x180F	バッテリーサービス
Blood Pressure	org.bluetooth.service.blood_pressure	0x1810	血圧
Body Composition	org.bluetooth.service.body_composition	0x181B	
Bond Management	org.bluetooth.service.bond_management	0x181E	
Continuous Glucose Monitoring	org.bluetooth.service.continuous_glucose_monitoring	0x181F	
Current Time Service	org.bluetooth.service.current_time	0x1805	現時刻
Cycling Power	org.bluetooth.service.cycling_power	0x1818	
Cycling Speed and Cadence	org.bluetooth.service.cycling_speed_and_cadence	0x1816	
Device Information	org.bluetooth.service.device_information	0x180A	
Environmental Sensing	org.bluetooth.service.environmental_sensing	0x181A	環境センサ
Generic Access	org.bluetooth.service.generic_access	0x1800	
Generic Attribute	org.bluetooth.service.generic_attribute	0x1801	
Glucose	org.bluetooth.service.glucose	0x1808	
Health Thermometer	org.bluetooth.service.health_thermometer	0x1809	体温
Heart Rate	org.bluetooth.service.heart_rate	0x180D	心拍数（脈拍）
HTTP Proxy	org.bluetooth.service.http_proxy	0x1823	
Human Interface Device	org.bluetooth.service.human_interface_device	0x1812	
Immediate Alert	org.bluetooth.service.immediate_alert	0x1802	
Indoor Positioning	org.bluetooth.service.indoor_positioning	0x1821	
Internet Protocol Support	org.bluetooth.service.internet_protocol_support	0x1820	
Link Loss	org.bluetooth.service.link_loss	0x1803	
Location and Navigation	org.bluetooth.service.location_and_navigation	0x1819	
Next DST Change Service	org.bluetooth.service.next_dst_change	0x1807	
Object Transfer	org.bluetooth.service.object_transfer	0x1825	
Phone Alert Status Service	org.bluetooth.service.phone_alert_status	0x180E	
Pulse Oximeter	org.bluetooth.service.pulse_oximeter	0x1822	
Reference Time Update Service	org.bluetooth.service.reference_time_update	0x1806	
Running Speed and Cadence	org.bluetooth.service.running_speed_and_cadence	0x1814	
Scan Parameters	org.bluetooth.service.scan_parameters	0x1813	
Transport Discovery	org.bluetooth.service.transport_discovery	0x1824	
Tx Power	org.bluetooth.service.tx_power	0x1804	
User Data	org.bluetooth.service.user_data	0x181C	
Weight Scale	org.bluetooth.service.weight_scale	0x181D	

4. BLE機能④ 基本サンプル

Genuino101によるBLEの基本サンプル事例

```
#include "CurieBLE.h"
BlePeripheral BlePeripheral;
BleService BLEService("19B10010-E8F2-537E-4F6C-D104768A1214");
BleCharacteristic ReadWriteData("19B10011-E8F2-537E-4F6C-D104768A1214", BLERead | BLEWrite, 20);
```

BLEPeripheral インスタンス設定
BLEService UUID設定
BLECharacteristic 特性設定

読込 (READ) と書込
(WRITE)、バイト長

```
void setup(){
  BlePeripheral.setLocalName("BLE_NAME");
  BlePeripheral.setAdvertisedServiceUuid(BLEService.uuid());
  BlePeripheral.addAttribute(BLEService);
  BlePeripheral.addAttribute(ReadWriteData);
  BlePeripheral.begin();
}
```

BLE名 (LocalName)の設定
サービスUUID設定
属性設定 : BLEサービス、特性
BLE開始

```
void loop() {
  if (BlePeripheral.central()) {
    while (BlePeripheral.central().connected()) {
      // ここに処理スケッチを記入
    }
  }
}
```

セントラル側からの
・セントラル検索
・セントラルとの接続

GI01_BLE_sample.ino

5. EEPROM機能① EEPROMクラス関数

EEPROMは、Arduino上にある不揮発性メモリで、電源を切ってもメモリ保存された状態となります。

センサ値などのコンフィギュレーションなどの値や、赤外線リモコン（学習リモコン）などで読み込んだIR値などを保管しておくことで、初期設定を電源入れるたびに行う必要がなくなります。

Genuino101のEEPROMは2 K（2048）バイト、利用方法はArduino UNO（1 Kバイト）などと同じ利用で読み書き可能

EEPROMは、ヘッダーファイル「EEPROM.h」を宣言する必要があります。

ただし、このEEPROMオブジェクト関数群は、処理スピードが遅く、制限として1,000,000回程度となっています。（使いすぎに注意してください）

EEPROMオブジェクト関数	引数説明	戻り値
EEPROM.read(address)	アドレス値（address:int）のデータ読込	バイト:0~255byte
EEPROM.write(address,value)	アドレス値のデータ書込、value:0~255byte	
EEPROM.update（address,value）	アドレス値のデータ変更、value:0~255byte。write関数の延命	
EEPROM.get(address,data)	アドレス値のデータ読込、data:int型・float型・構造体など	データ値
EEPROM.put(address,data)	アドレス値にデータ書込、data:int型・float型・構造体など	
EEPROM.length()	EEPROMの容量（バイト）	EEPROMサイズ(int)
EEPROM[address]	高速なEEPROMアドレスでの読込・書込	バイト:0~255byte

5. EEPROM機能② サンプルスケッチ

本サンプルスケッチは、ライブラリとして用意されている

EEPROM_iteration.ino

```
#include <EEPROM.h>

void setup() {
  /** Iterate the EEPROM using a for loop.  */
  for (int index = 0 ; index < EEPROM.length() ; index++) {
    //Add one to each cell in the EEPROM
    EEPROM[ index ] += 1;
  }
  /** Iterate the EEPROM using a while loop.  */
  int index = 0;
  while (index < EEPROM.length()) {
    //Add one to each cell in the EEPROM
    EEPROM[ index ] += 1;
    index++;
  }
  /** Iterate the EEPROM using a do-while loop.  */
  int idx = 0; //Used 'idx' to avoid name conflict with 'index' above.

  do {
    //Add one to each cell in the EEPROM
    EEPROM[ idx ] += 1;
    idx++;
  } while (idx < EEPROM.length());
} //End of setup function.

void loop() {}
```

■ 宣言

#include "EEPROM.h"

■ 初期設定

■ センサ値取得関数

• CurieIMU.readGyroScaled(gx, gy, gz);

EEPROMを使うことでread/writeより高速で、寿命も長くなる

6. RTC（リアルタイムクロック）機能

Genuino101は、リアルタイムクロック機能を持ち、内部時間を取得することができる。

3 GIMで取得した時刻を、Genuino101のRTCに設定するサンプルスケッチを紹介しましょう。

■ 宣言

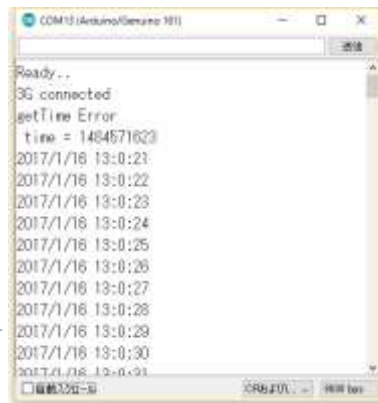
```
#include "CuireTime.h"
```

■ 日時設定関数

```
setTime(H,M,S, D, M, Y);  
setTime(t); // 累積秒
```

■ 日時取得関数

```
year();  
month();  
day();  
hour();  
minute();  
second();  
now(); // 1970年1月からの累積秒
```



setTime_3GIM_G101

```
#include <CurieTime.h>  
#include <a3gim2.h>  
#define baudrate 9600UL  
  
void setup() {  
  while(!Serial);  
  Serial.begin(9600);  
  Serial.println("Ready..");  
  unsigned long seconds;  
  if(a3gs.start(7)==0 && a3gs.begin()==0) {  
    Serial.println("3G connected ");  
    while(a3gs.getTime2(seconds) !=0)  
      Serial.println("getTime Error");  
    Serial.println(" time = " + String(seconds));  
    a3gs.getTime2(seconds);  
    setTime(seconds);  
  } else Serial.println("3G not connected");  
  if(seconds-10.0<0.0) {  
    Serial.println(" error ");  
  }  
}  
  
void loop() {  
  static int s;  
  if(second()!=s) {  
    s=second();  
    Serial.print(String(year()) + "/");  
    Serial.print(String(month()) + "/");  
    Serial.print(String(day()) + " ");  
    Serial.print(String(hour()) + ":");  
    Serial.print(String(minute()) + ":");  
    Serial.print(String(second()));  
    Serial.println();  
  }  
}
```

もくじ

- 第1章 IoTABシールドの基本利用
- 第2章 IoTABシールドの応用利用
- 第3章 IoTABシールド・デモ・スケッチ
- 第4章 応用デモ・スケッチ
- 第5章 温度ステーション



IoTAB SHIELD 3.0

第Ⅲ編 IoTABシールド編

もくじ

- (1) IoTABシールド概説
- (2) 入力部品
- (3) 出力部品
- (4) 赤外線 (IR) リモコン



第1章 IoTABシールドの基本利用



(1) IoTABシールド概説

1. IoTABシールドV4.0とは

IoTABシールド・キットは、**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。これまでのバージョンは、V1.0、V1.1、V2.0、V3.0があり、広く利用されてきています。

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、いち早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（液晶ディスプレイ）・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。

（Genuino101の独自機能と組み合わせれば、さらに広がります）

また、3 GIMと組み合わせれば、M2Mビジネスの材料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) 3 GIMとの連携によるIoT遠隔監視・制御

2. IoTABシールド仕様

IoTABシールド上装備の電子部品

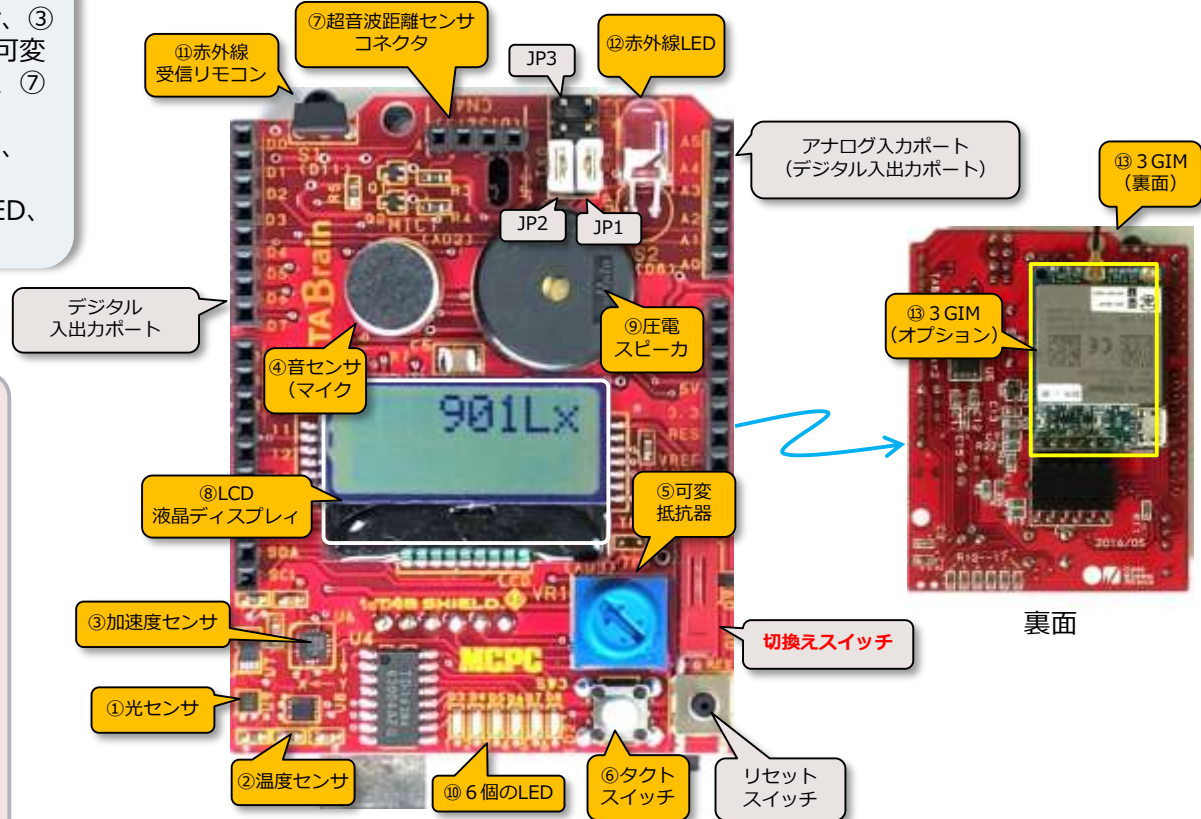
入力電子部品：①照度センサ、②温度センサ、③加速度センサ、④音（マイク）センサ、⑤可変抵抗器（ボリューム）、⑥タクトスイッチ、⑦超音波距離センサ（外付け）

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、⑬3 GIM用インタフェース

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、さまざまなユニークな製品づくりができます。



※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。

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シールド上の電子部品

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	照度センサ	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（液晶ディスプレイ）	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. Arduino+IoTABシールドの接続について

【注意】 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。（**3 GIM 利用時**）

（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 M0 Pro** : 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）



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＞

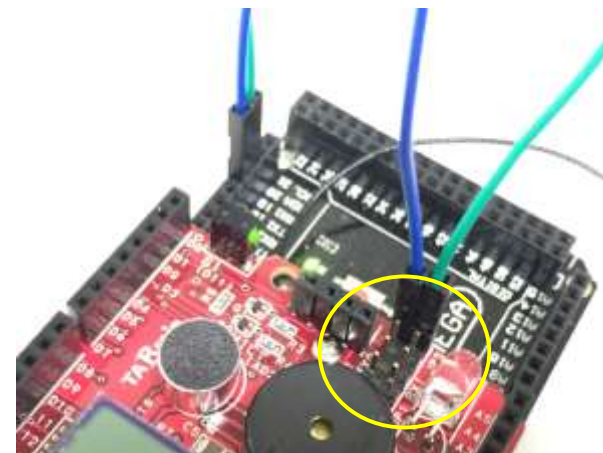


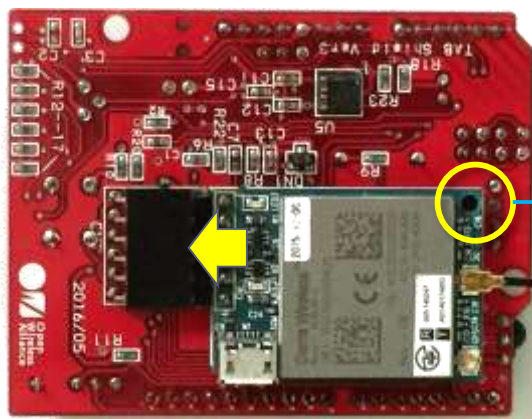
写真3 : Arduino MEGAの場合
Rx/Dxをシリアル1に接続

5. 3 GIMをIoTABシールドに設置

3 GIM は、IoTABシールドの裏面に接続配置します



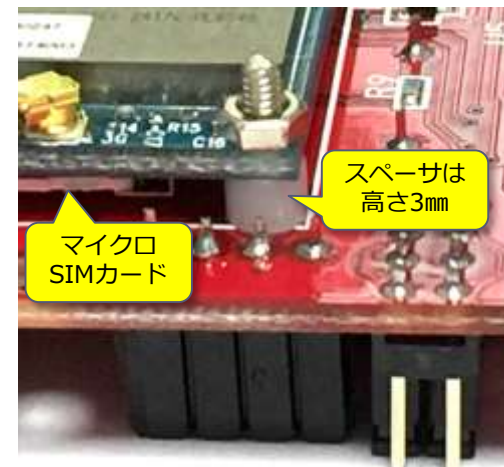
L型ピンを半田付け



2mm×10mmネジ利用
スペーサは3mm利用



スペーサは
高さ3mm



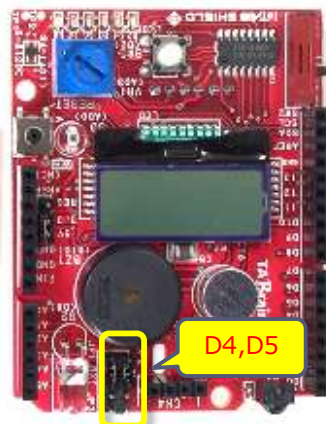
マイクロ
SIMカード

スペーサは
高さ3mm

マイクロSIMカード挿入後、ネジ止め



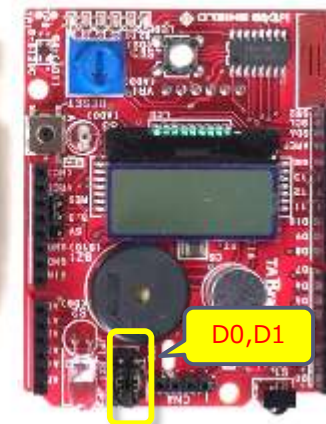
Arduino UNOに搭載する場合



D4,D5



Genuino101/Arduino M0 Proに搭載する場合



D0,D1



(2) 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をします。

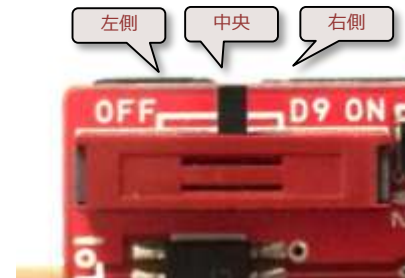
IoTABシールドに装備された入力電子部品は以下のとおりです。

■ 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



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

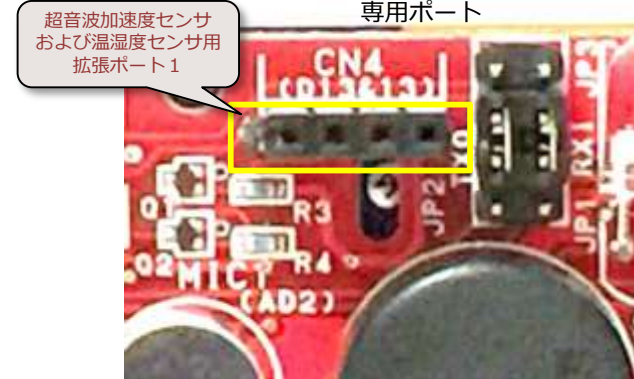
左側 : IoTABシールドのD10-D13、A0-A3電子部品が利用不可

■ 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

デジタル拡張 専用ポート



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温度センサなどを指すことができるようにしました。

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、デジタルセンサ値の値が不安定になる。

IoTABSシールドのタクトスイッチにはプルアップ抵抗が取り付けられています。

■ 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

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

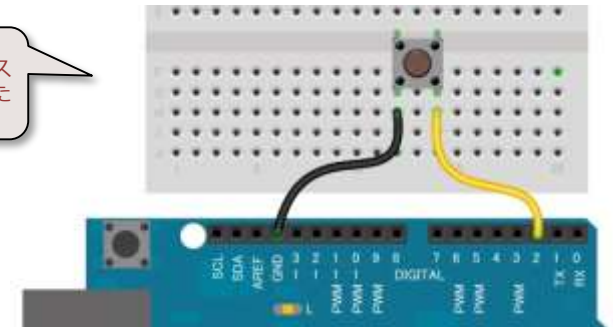
ケーブルが繋がれたとき、sw = LOWに
離れたときは、sw = HIGHになる

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



タクトスイッチ
の位置



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

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

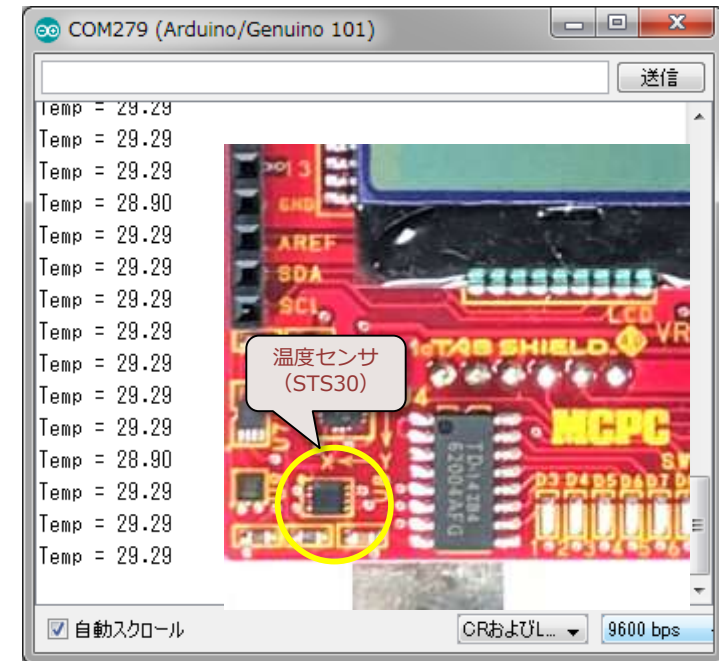
■ サンプルスケッチ

```
//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

■ シリアルモニタ画面



■ 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ります。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ます。

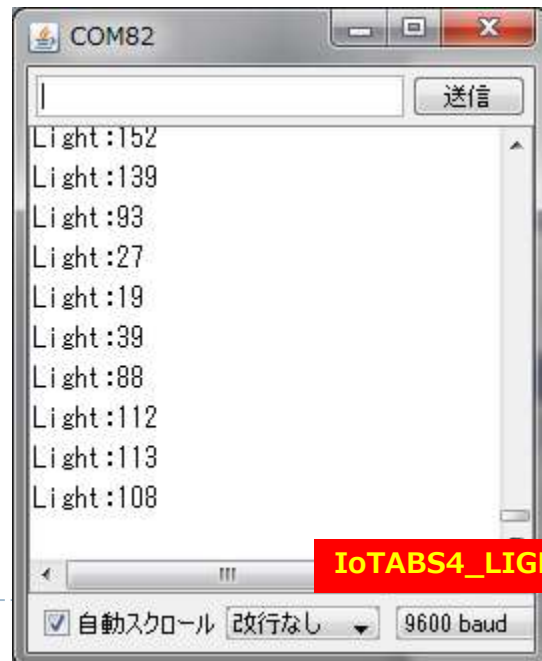
4. 照度（光）センサを使う

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



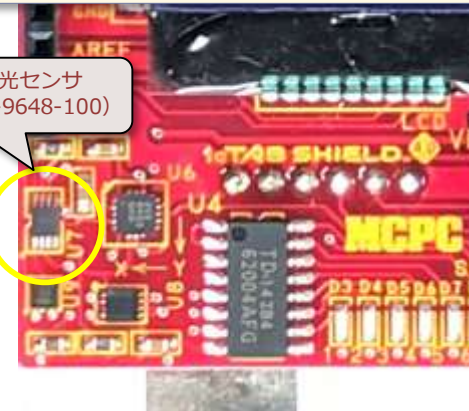
■ サンプルスケッチ

```
#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 値が小さくなる

光センサ
(S-9648-100)



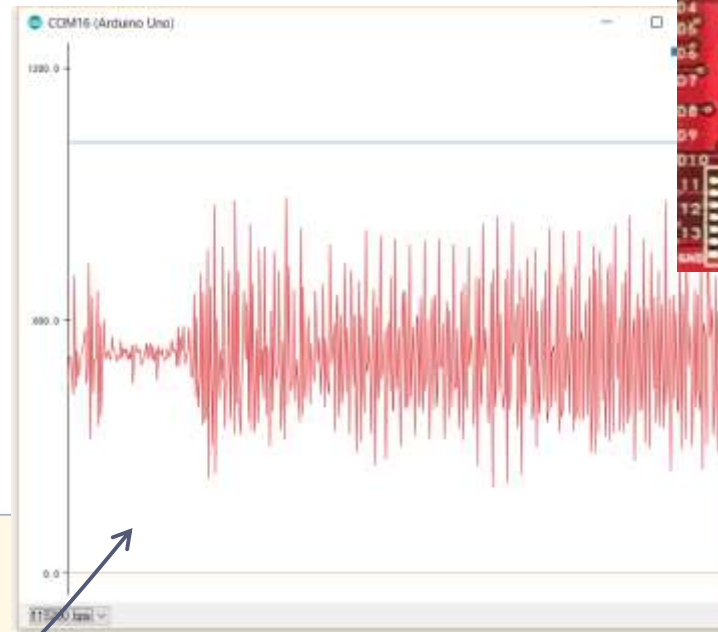
5. 音センサ（マイク）を使う

アナログ

■ 音センサ（マイク）

項 目	説 明
製品名	C9767BB422LFP メーカー（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



コンデンサ
マイク

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、平均化が必要がある。
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。

6. 可変抵抗器を使う

■ 可変抵抗器

項目	説明
製品名	3386K-EY5-103TR (メーカー: 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

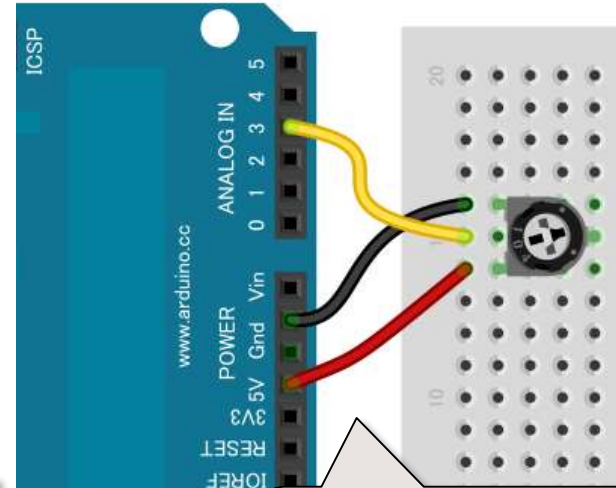
```
//サンプルスケッチ
#define RegPin A3
```

IoTABS4_REG.ino

```
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);
}
```

■ 単独での事例



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、中央のピンを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

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

7. 超音波距離センサを使う

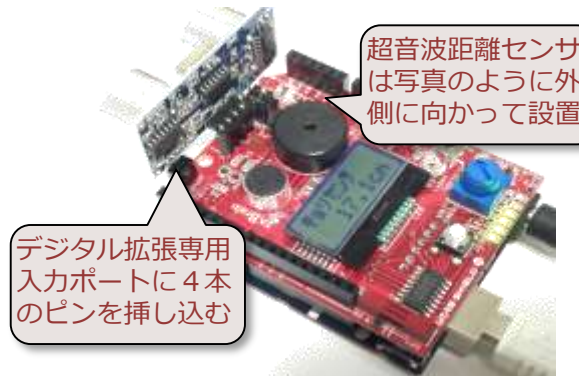
デジタル拡張
専用入力ポートを使います

デジタル

■ 超音波距離センサ

項 目	説 明
製品名	HC-SR04 (メーカー：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

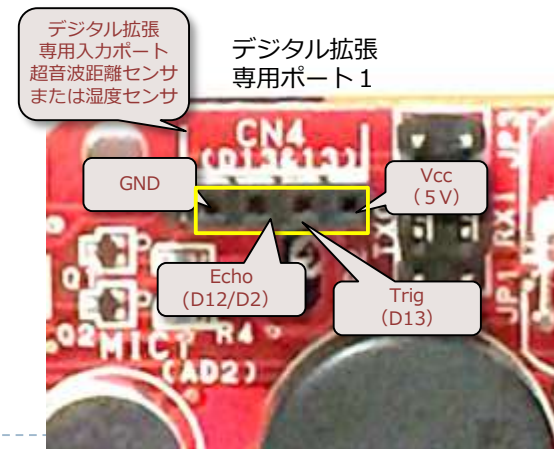


■ サンプルスケッチ

```
#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);
}
```

IoTABS4_DIST.ino

超音波の速度を340m/secで計算した換算式



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 (メーカー:)
I/O	I2C (0x1C)
利用方法	ライブラリ利用
単位	G
備 考	2 G、4 G、8 G 利用可

■ 加速度センサの位置



■ サンプルスケッチ

```
#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);
}
```

IoTABS4_ACC.ino

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. 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） （メーカー：
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てください。

参照：サンプルスケッチ

IoTABS4_DTH11.ino を添付

湿度センサDTH11 / DTH22(AM2302)

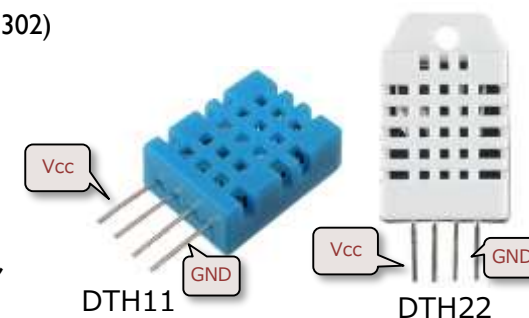
1) Vcc(3.5V～5.5V)

2) SDA

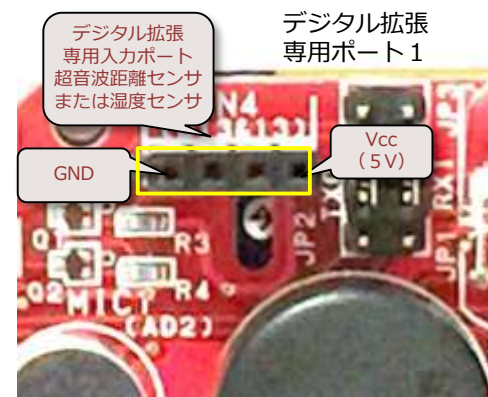
3) NC(接続無し)

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

IoTABS4_DIST_LCD_D9.ino

切換えスイッチを利用することで、デジタルピンのDI0-DI3およびアナログピンのA0-A3に接続する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

```
// 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;
  }
}
```

【ブレイク】 Genuino101 の再起動

- ▶ 今までのArduinoUNOなどでのプログラムが起動するタイミングは、以下のとおりでした。
 - 1) 電源を入れたとき
 - 2) プログラムを書き込んだとき
 - 3) IDEの中のシリアルモニタ画面を開いたとき
 - 4) リセットボタンを押したとき
- ▶ しかし、Genuino101では、諸条件によって違う動きをします。
 - ① **シリアルモニタ画面を使う場合**
 - ・プログラムを書込み後、`setup`関数の先頭に、「`while(!Serial);`」を入れておくことで、シリアルモニタ画面を開いたときにプログラムが起動します。
 - ・ただし、COMの認識する前にシリアルモニタ画面を開いてもプログラムは起動しません。この場合、『シリアルポート「COM13」が開けません。(Port not found)』が表示され、シリアルモニタ画面が開いても、プログラムは起動しません。
 - ・もしこの `while`文を入れていない場合には、シリアルモニタ画面を開くことなくプログラムは起動しはじめます。
 - ・また、一度シリアルモニタ画面を閉じ、再度画面を開いたとした場合には、プログラムは稼働したままの状態となっています。(再起動は置きません)
 - ・シリアル画面を開いた状態で、まだ表示されない場合には、再度シリアルモニタ画面のボタンを選択してみてください。
 - ② **再起動をしたい場合**
 - ・プログラムの再書き込みをすると再起動します。
 - ・Master Resetボタンを再度押すと、約5秒後に再起動できます。
 - ③ **シリアルモニタ画面で再起動したい場合**
 - この場合には、上記した「`while(!Serial);`」を挿入した上で、
 - ・プログラムを書き込んだ後に、シリアルモニタ画面を開くか
 - ・Master Resetボタンを押した後に、シリアルモニタ画面を開くときに、再起動がかかります。

(3) 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

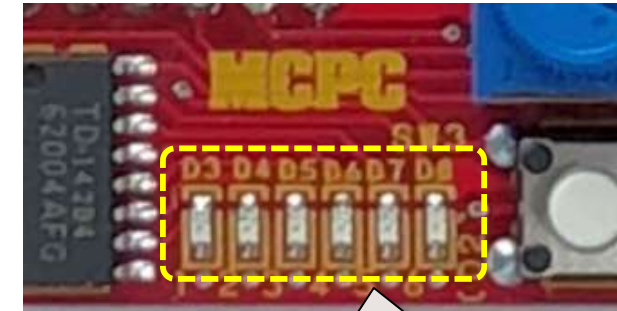
IoTABシールドに装備された出力電子部品は以下のとおりです。

(注意) LED 6 と赤外線LEDは、同じD7を利用

- アナログ電子部品 (圧電スピーカ : D9とLED 2 , 4 , 5)
- デジタル電子部品 (D 2 ~D 7)

デジタル出力ポート	接続電子部品
D3	LED 1 (PWM)
D4	LED 2
D5	LED 3 (PWM)
D6	LED 4 (PWM)
D7	LED 5
D8	LED 6 & 赤外線LED

LED 1 , 3 , 4 は
PWM制御可能
(PWM機能を使って明
るさを変更可能)



左側からLED 1 (D3)から
LED6 (D8)と配置

- 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)

中央位置
HIGH : 電子部品On
LOW : 電子部品Off

右側 : 電子部品On
左側 : 電子部品Off

切換えスイッチ

左側 中央 右側

切換えスイッチを中央位置に
した場合にのみ、D9の切
り替えてIoTABシールドの
電子部品のOn/Offが切替
えます



- 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

2. 圧電スピーカを使う

■ 圧電スピーカ

項目	説明
製品名	C9767BB422LFP メーカー (DB)
I/O	デジタル出力 (D10) pinMode(10,OUTPUT); tone(10,hz,dly); noTone(10);
備考	hz: 音の高さ (右欄参考) dly: 時間(ms)

必須

■ 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

アナログ出力でも
音が出せる

ドレミを連続して繰り返すスケッチ

■ 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。

圧電
スピーカ

3. 6個のLEDを使う

デジタル

■ 5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,HIGH);
備考	Dx: 3~8 (つまりD3~D8) HL: HIGHまたはLOW

必須



左側からLED 1 から
LED 6 と配置

流れ星が左から右に
移動していく状態の
スケッチ

■ 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

```
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();
  }
}
```

IoTABS4_LED_02.ino

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 メーカー(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み込みが必要
備 考	専用モジュール「I2C_LCD.ino」添付資料

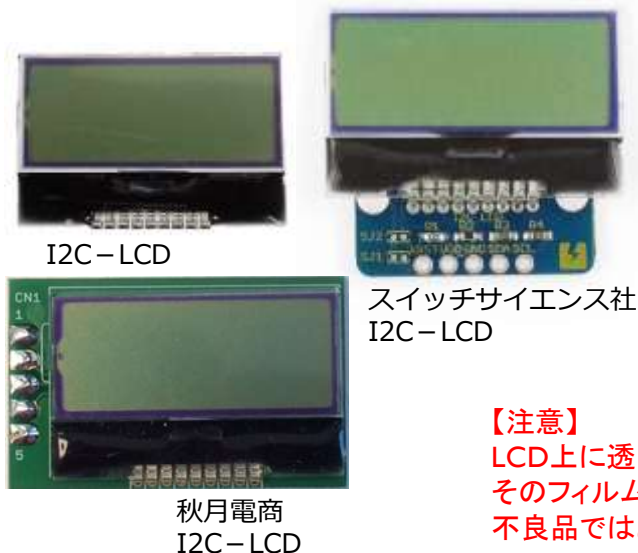
■ I2C_LCD.ino 関数群

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)

■ 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



【注意】

LCD上に透明なフィルムが貼り付けてあり、そのフィルムに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。

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() { }
```

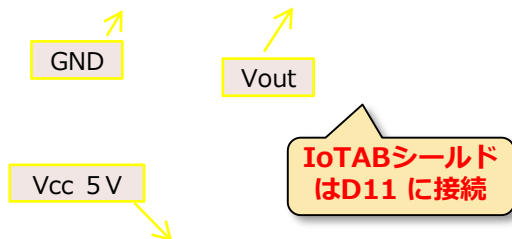
IoTABS4_EEPROM.ino

(4) 赤外線 (IR) リモコン

1. 赤外線リモコン受信モジュールを使う

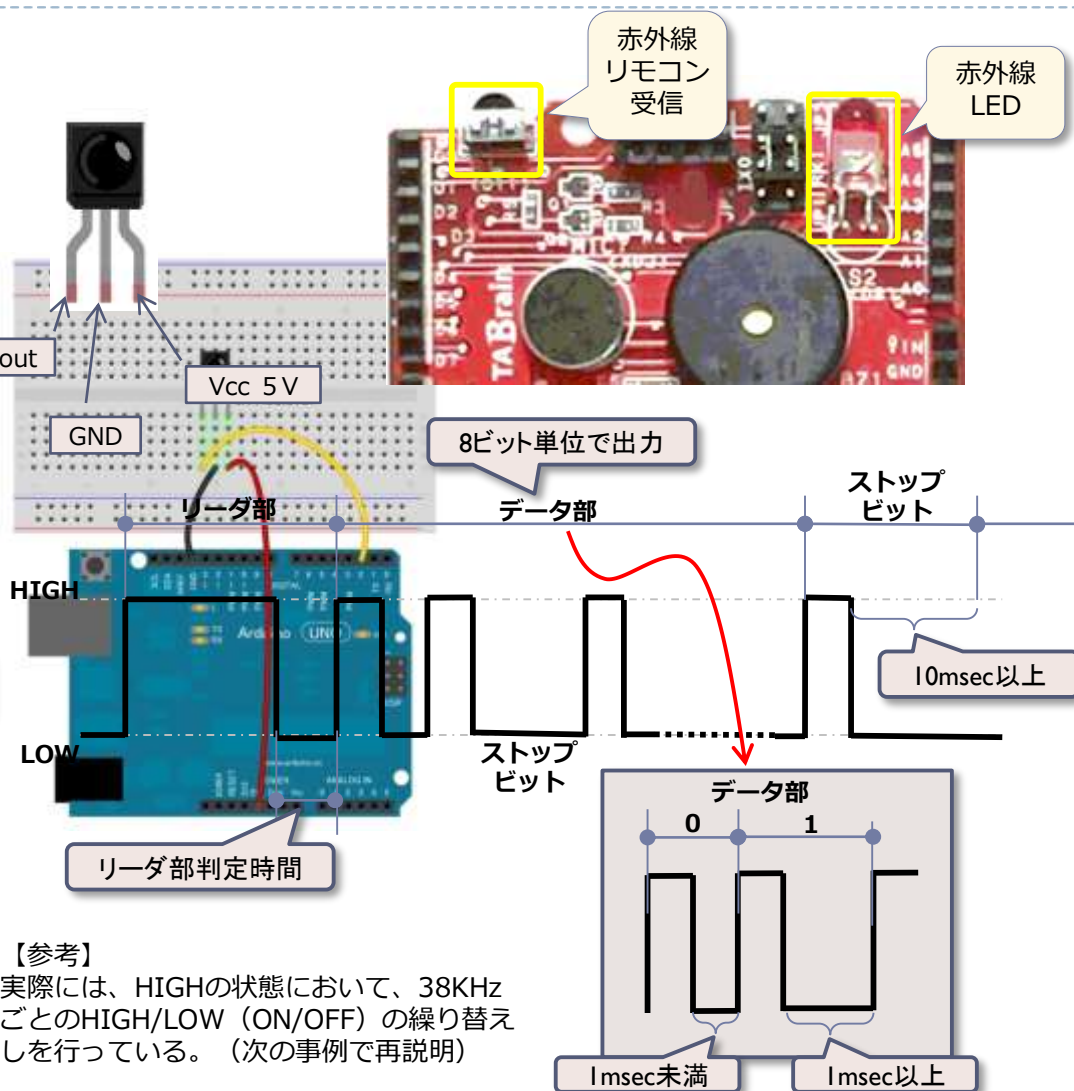
■ 赤外線リモコンの学習

■ 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、メーカー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 リード部LOWが4000より小さい場合（プログラム変更にて利用可能）



2. 赤外線リモコン受信モジュールのスケッチ

```
#define IR_Pin    2           // 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++;
    }
  }
}
```

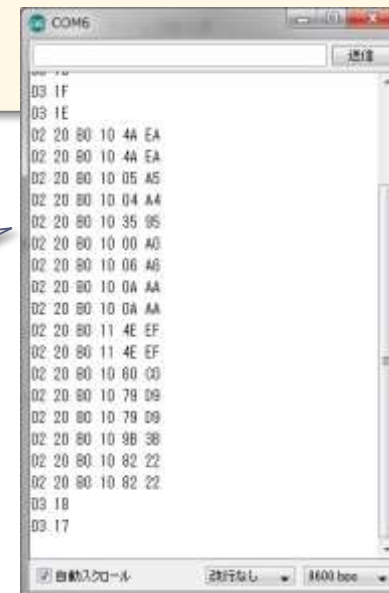
IoTABシールド
はD11 に接続

```
// 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); // H E X(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

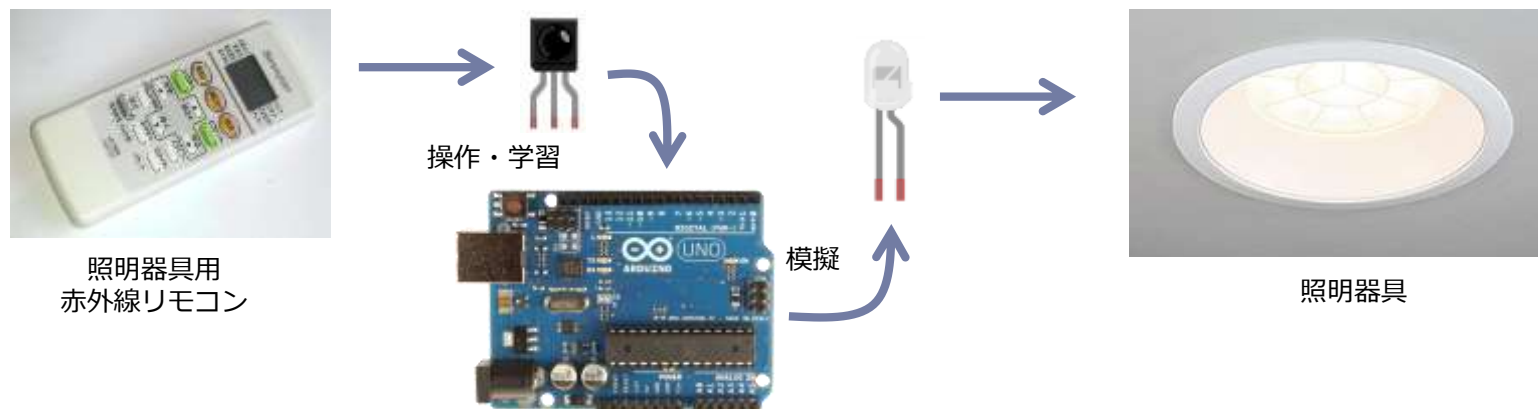
Sample_IR.ino



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。



■ 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。

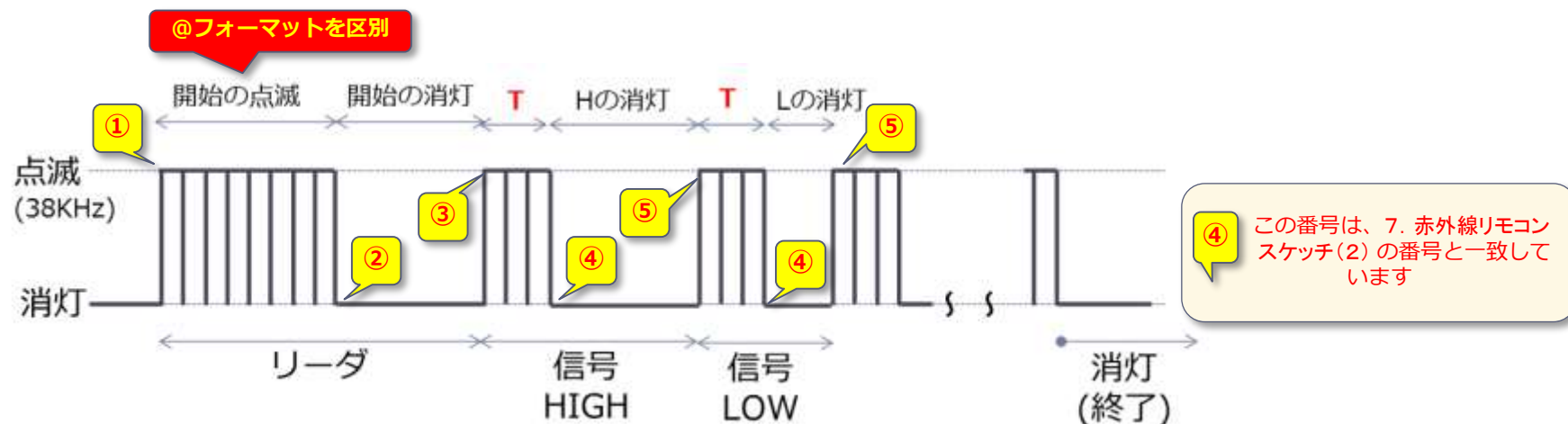
■ 各方式で点灯と消灯の時間が定められており、その関係は、①信号の開始、②信号のHIGH/LOWの送出（1または0）、となっています。

■ 家電製品により、制御するときに出すビット列（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。（スキ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

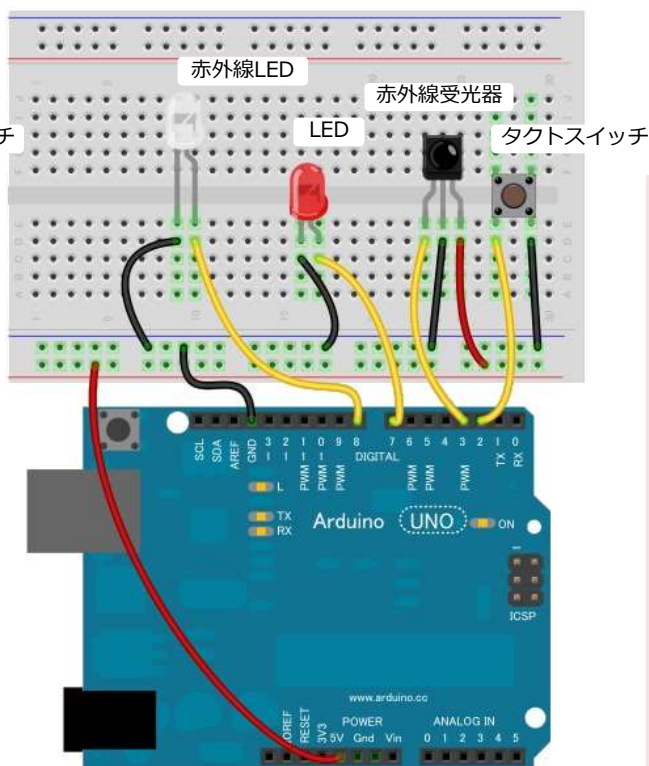
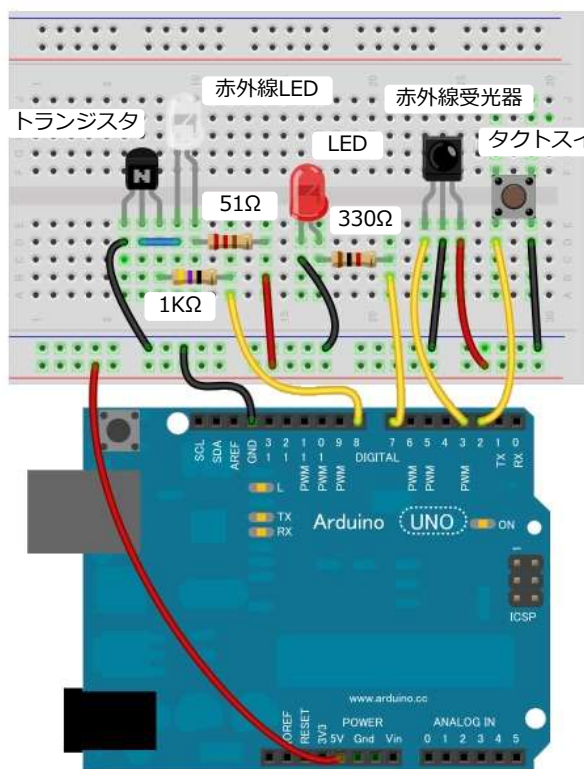
- ・家電協会またはNECフォーマットの赤外線リモコンを扱う
- ・一つの操作だけを学習する
- ・利便性を考えて、一つのスケッチで、赤外線リモコンの学習と、その操作の両方を実行できるようにする
- ・タクトスイッチとLEDを使い、学習と操作のOn/Offを切り換える。
- ・タクトスイッチを押したままリセット（もしくはシリアルモニタ表示）すると、学習モードとなり、LEDが点灯する。
- ・初期起動時は、タクトスイッチを押した状態で、赤外線リモコンの学習モードとする。
- ・一度、学習した情報は、EEPROMに書き込まれ、次回以降の立ち上げで、値を読み込んでくる。

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。

右側は簡単でシンプルな接続です。



【注意】

本キットで利用している赤外線LEDは型番OSI5LA113A、下記サイトで購入可能です。

<http://akizukidenshi.com/catalog/g/gI-12612/>

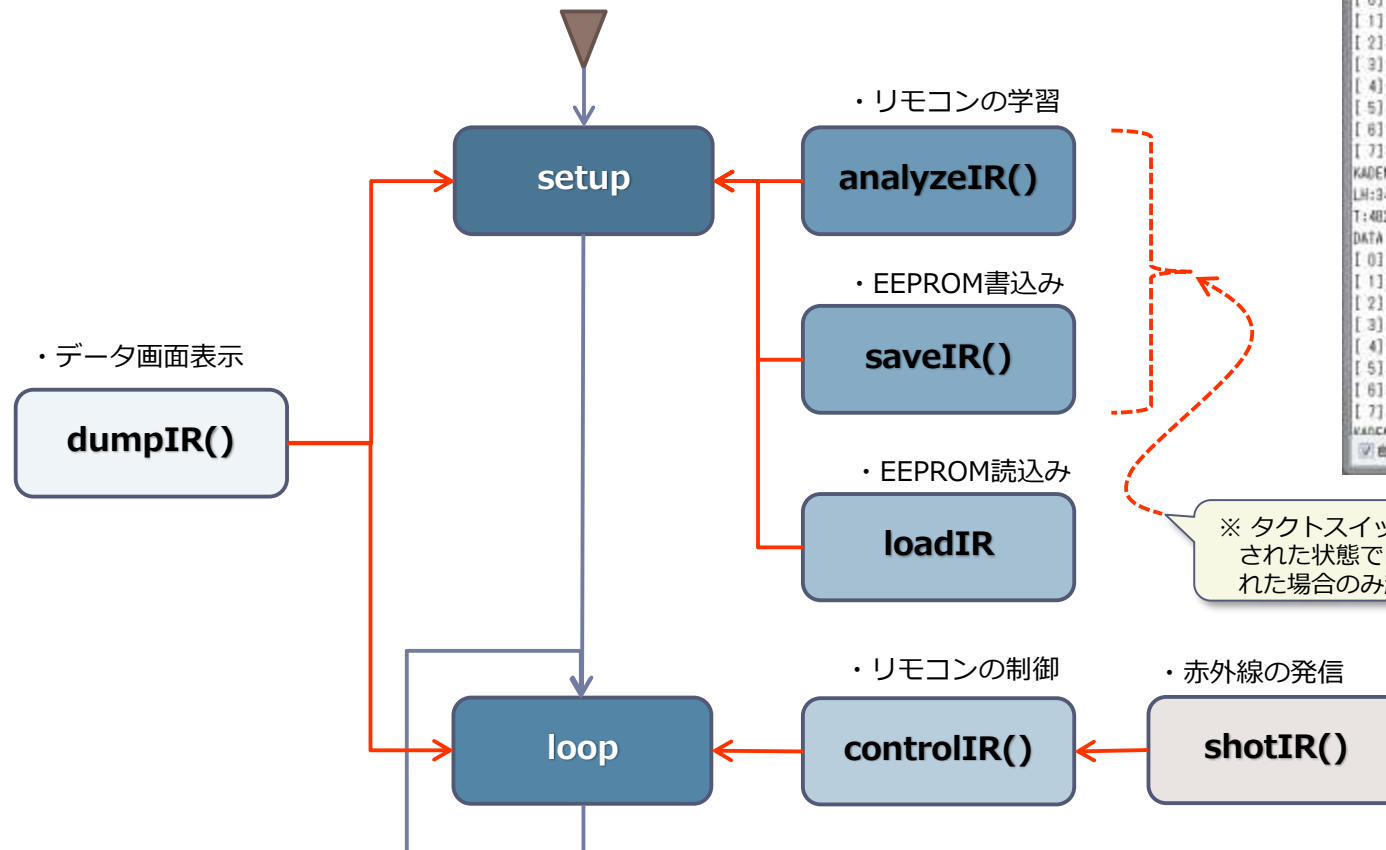
この製品の特性として、「半減角：15度（狭角）」、つまり、15度離れると赤外線のパワーは1/2となります。

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。



```
COM44
Ready to catch.
KADEN Format>
LH:3488, LL:1836
T:402, TL:356, TH:1208
DATA COUNT:8
[ 0] 0x55
[ 1] 0x5a
[ 2] 0xf3
[ 3] 0x88
[ 4] 0x80
[ 5] 0x40
[ 6] 0x84
[ 7] 0x85
KADEN Format>
LH:3488, LL:1836
T:402, TL:356, TH:1208
DATA COUNT:8
[ 0] 0x55
[ 1] 0x5a
[ 2] 0xf3
[ 3] 0x88
[ 4] 0x80
[ 5] 0x40
[ 6] 0x84
[ 7] 0x85
```

※ タクトスイッチが押された状態で開始された場合のみ起動

【起動例】
シリアルモニタ表示

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
#include <EEPROM.h>

#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_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];
```

setup()

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

IoTAB3_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
  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 analayzed 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();
```

EEPROMヘッダー

リモコンの学習

タクトスイッチが
押された状態で起動

データ画面表示

EEPROM書込み

EEPROM読み込み

EEPROMデータ無しエラー処理

7. 赤外線リモコンのスケッチ (2)

loop()

```
void loop()
{
  if (digitalRead(ButtonPin) == LOW) {
    dumpIR();           データ画面表示
    digitalWrite(LedPin, HIGH); // Led On
    controlIR();        リモコンの制御
    delay(100);
    digitalWrite(LedPin, LOW); // Led Off
  }
  delay(30);
}
```

analyzeIR() : リモコンの学習

```
// analyzeIR() -- マイクロ秒単位で波長を読み取り
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;
```

```
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!
```


7. 赤外線リモコンのスケッチ（3）

saveIR() : EEPROM書き込み

```
// 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読み込み

```
// 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)

controlIR() : リモコン制御

```
// controlIR() --
void controlIR()
{
    // Shot leader signal
    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は
IoTABシールドの内蔵する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2章 IoTABシールドの応用利用

IoTABシールド組み合わせ学習

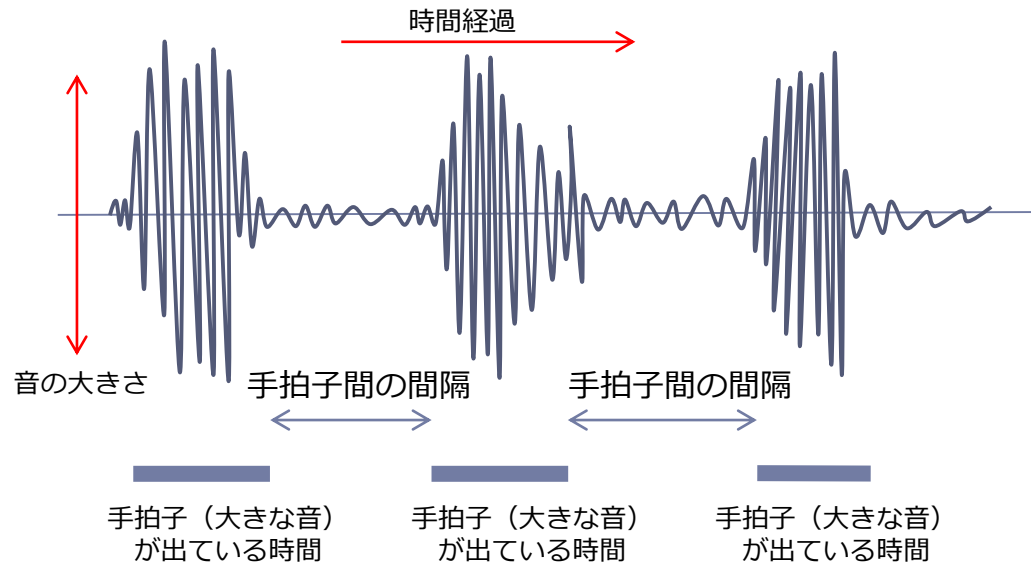
IoTABシールドのもつ多くの電子部品を組み合わせ、入力と出力とで結びつけることで、さまざまなアイデア製品の開発が学べます。

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	タクトスイッチ	タクトスイッチ LED	タクトスイッチを押す度に、LEDを点灯・消灯させる。
3	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。
4	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。
5	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。 ストップウォッチ ：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。
6	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。まずは手拍子のが音センサで取れる波形を想定してみましょう。以下のようなになることが想像つきます。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

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。

```
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);  
}
```



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。

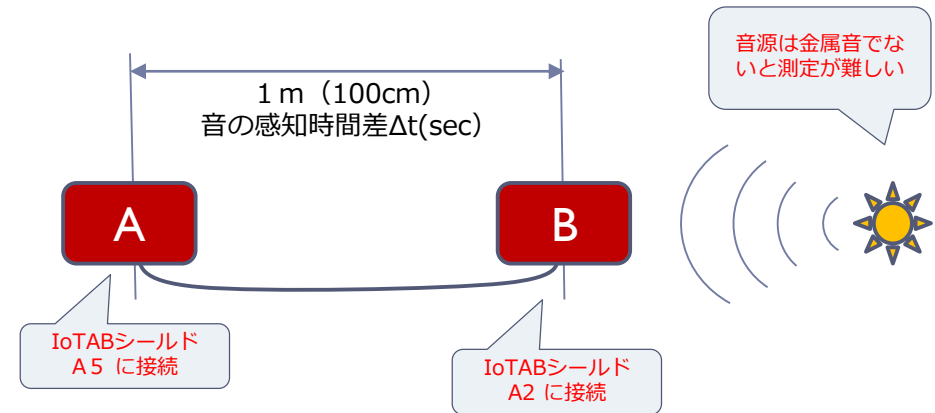
$$\text{音速 (m)} = 1/\Delta t$$

2つのIoTABシールドは、「A」のA5ポートと「B」のA5ポートを接続する

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って、簡単に、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むのを簡単にして、連続して、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。

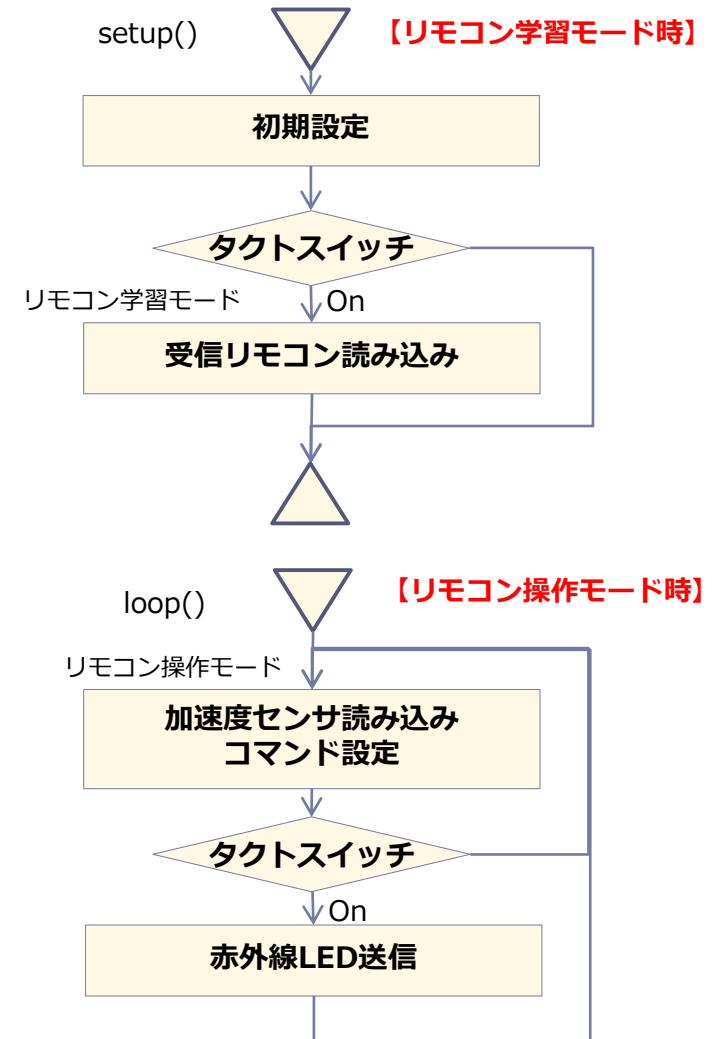
【リモコン操作モード時】

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

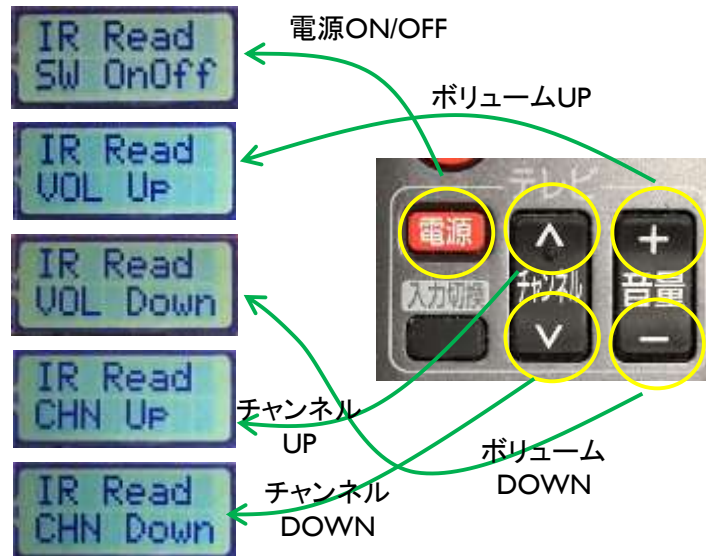


4. テレビ学習リモコン ②

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」を読み込み、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】

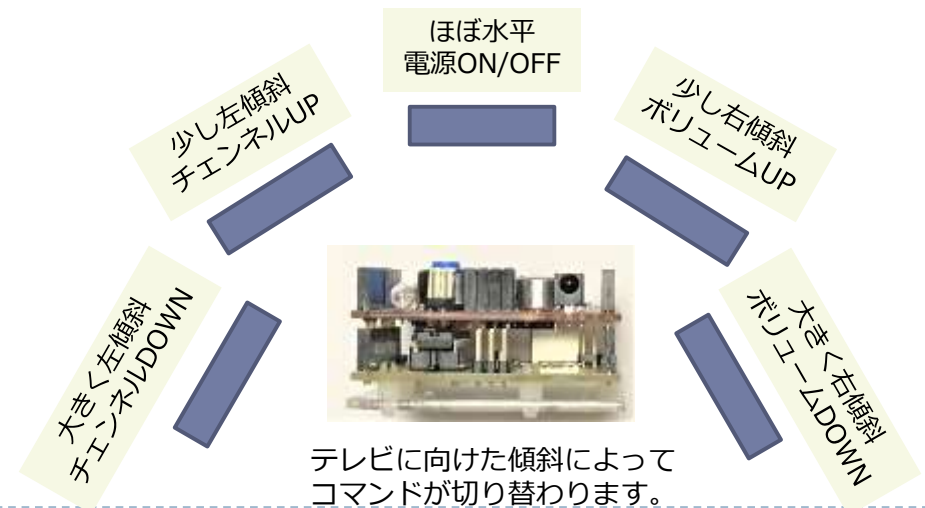
IoTABシールドのタクトスイッチを押した状態で、リセット（または電源ON）し、その後タクトスイッチを離します。その後LCD表示に従って5種類のリモコンのボタンを押します。



【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



4. テレビ学習リモコン ③

IoTABS4_TV_IR_STUDY.ino

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) analyzeIR (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。

```
// 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(); controllIR();
    }
    delay(300);
    digitalWrite(com+2,LOW);
  }
}

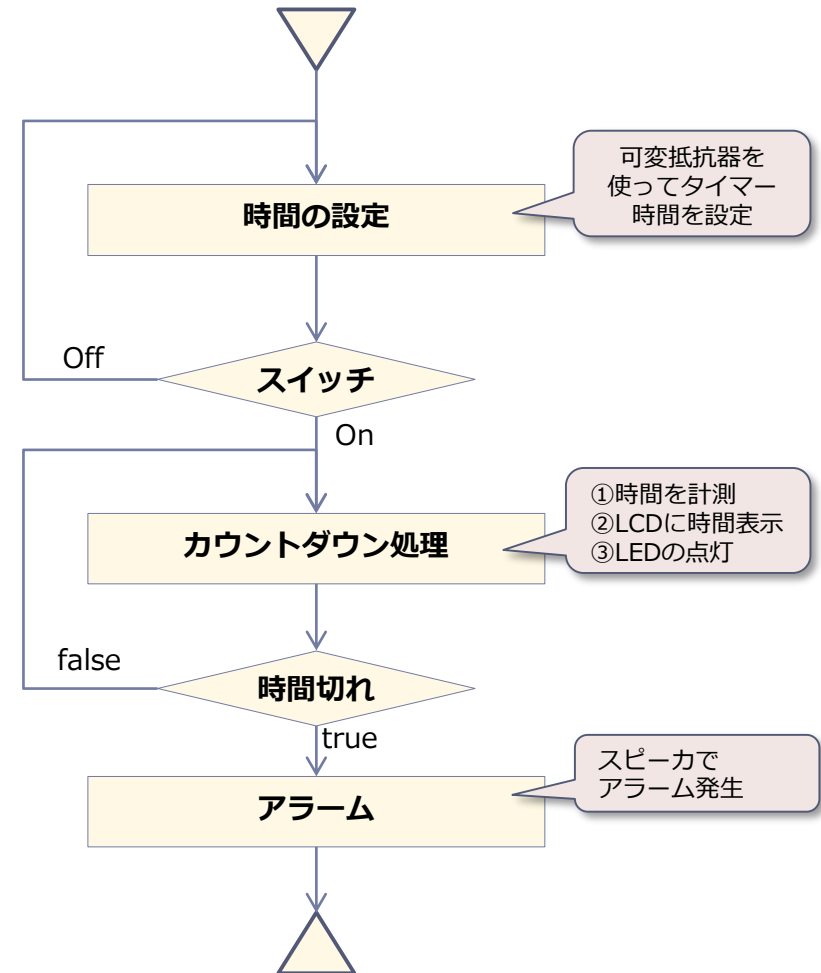
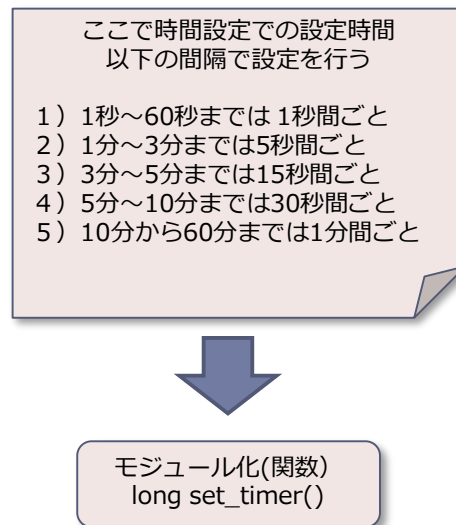
```

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）



5. タイマーを作る ②

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

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

時間設定

カウントダウン

時間切れアラーム

```
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);
}
```

IoTABS4_TIMER.ino

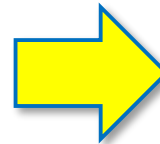
時間設定のモジュール

```
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。

IoTABS4_STEP_COUNT.ino

```
#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);
  // initialize the sensor:
  MMA8452Q_init();

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

ライブラリ追加

LEDのピン番号変更

LCD初期化・表示追加

LCD表示追加

```
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

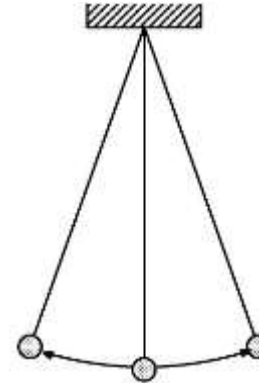
7. IoTABシールドを使った応用事例



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



熱中症観測



音（ノイズ）計測



超音波距離計測



モールス信号



学習リモコン

第3章 IoTABシールド・デモ・スケッチ

1. デモ・スキッチの紹介①

ここで紹介するサンプルスキッチは、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。

- 1) 「wat-sen」 温度センサのLCD表示とLED点滅
- 2) 「hikari-sen」 光センサのLCD表示とLED点滅
- 3) 「aki-sen」 音センサによるLCD表示（平均値表示付）とLED点滅
- 4) 「teyou-shi」 音センサを使った手拍子認識によるLCD表示とLED点滅
- 5) 「teiwari」 可変抵抗器を使ったLCD表示とLED点滅
- 6) 「tama-」 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) 「wakari-sen」 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) 「fumi-shi」 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) 「kodye-i」 スピーカによるメロディ「チューリップ」
- 11) 「sekiga-ten」 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、設定は上記 **【IR設定】** 参照）
- 12) 「3G Tweet」 温度値をツイート【ツイートするトークンに変更】※
- 13) 「3 G Mail」 温度値を 3 GIMを使って指定したメールアドレスに送信【自分のメールアドレスに変更】※
- 14) 「3G A-GPS」 GPS（位置情報値）をメール送信【自分のメールアドレスに変更】※

※印は、Genuino101上でIoTABシールド + 3 GIMが取り付けられた場合に有効なものです。

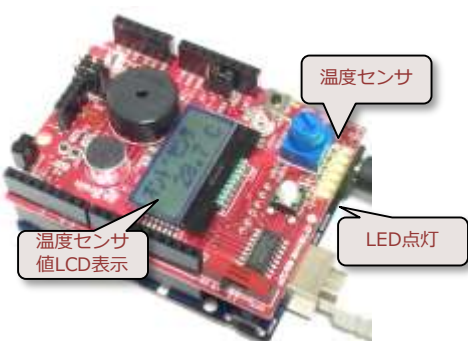
以上のように、温度センサ、湿度センサ、超音波距離センサ、可変抵抗器、スイッチ、音センサ、スピーカ、LED、LCDを使ったデモとなります。

【IR設定】赤外線学習リモコン読み込み操作

- 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の設定を行います。（コマンド学習）
学習させるリモコンをIoTABシールドのIR受信を向けて発信します。
- 2) テストプログラムの最後に、「sekiga-ten」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。

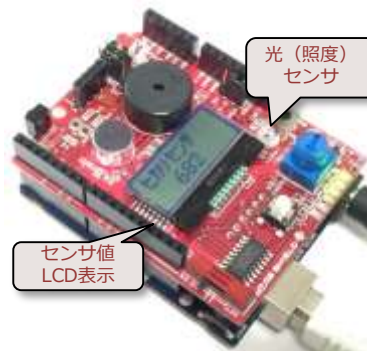
1. デモ・スケッチの紹介②

■温度センサ値表示



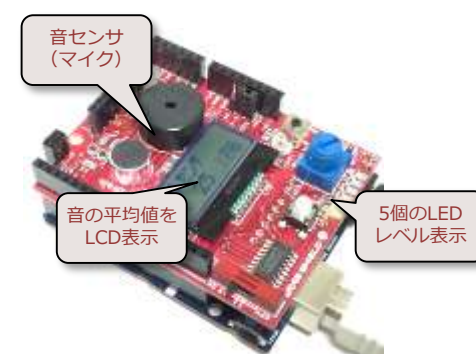
温度センサの値をLCDに表示。(アラーム出力も可)

■照度センサの値表示



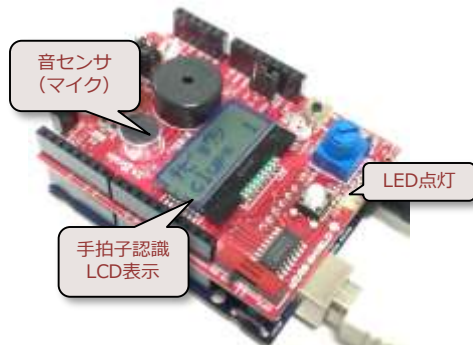
照度センサの値をLCDに表示。5個のLED点灯とも連動。(アラーム出力も可)

■音センサの値表示



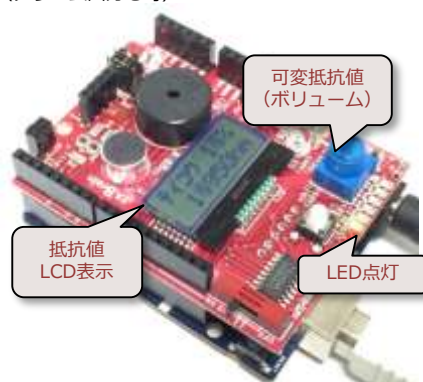
音センサの値をLCDに表示。同時に5個のLEDを使ってレベル表示。ある間隔の平均音量も並列して表示。

■手拍子の認識



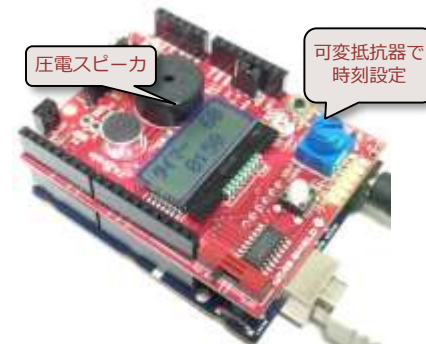
手拍子の数を認識して、その数をLCD・LEDに表示。赤外線リモコンと組み合わせ家電の制御などに利用可能。

■可変抵抗値の値表示



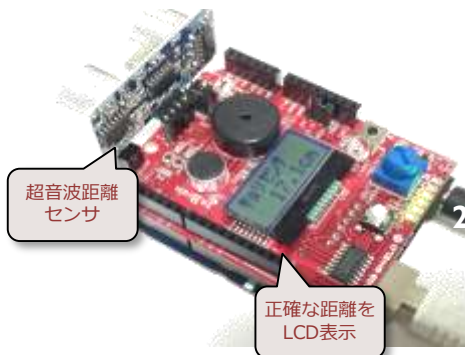
ボリューム値をLCDに表示。同時に5個のLEDも点灯。このボリュームもいろいろな切換スイッチに利用可能。

■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



■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。

2. デモ・スケッチ宣言部①

```
#include <Wire.h>
#include <EEPROM.h>
#include <arduino.h>
#define BAUDRATE 9600
```

初期設定 (1)

```
//===== 初期設定
=====
```

```
#define iemSerial Serial1
String server = "arduino-tweet.appspot.com:80/update ";
String mail_server = "sendmailServer";
String token = "YOUR-TOKEN"; // @owalliance_org
String e_mail = "your_mail_address"; // mail address
String SIM_APN = "¥"YOUUSER¥",¥"PW¥";
```

初期設定 (2)

```
// mopera.net,,
// iijmio.jp,mio@iij,iij
// vmobile.jp,bb@excite.co.jp,excite
// so-net.jp,nuro,nuro
// soracom.io,sora,sora
```

デジタルセンサ設定

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

超音波距離センサ設定

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

アナログセンサ設定

#define But_Pin 2

I2C_LCD初期設定

```
#define Led1Pin 3
#define Led2Pin 4
#define Led3Pin 5
#define Led4Pin 6
#define Led5Pin 7
#define Led6Pin 8
#define TABSPin 9
#define Spk_Pin 10
```

LED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
```

ドレミの設定

3 GIMを利用する際に必要となる設定

sendmailServer : sendmail.php を置くサーバ
YOUR-TOKEN : Twtterのトークン (下巻参考)
Your_mail_address : あなたのメールアドレス
YOUUSER : 利用するSIMカードのユーザ名
PW : 利用するSIMカードのパスワード

IoTABS3_demo.ino

2. デモ・スケッチ宣言部②

// 赤外線リモコン

```
#define MAX_SIGNALS 16
const int IRInputPin = 11;
const int IROutputPin = 8;
```

初期設定 (1)

// 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];
```

超音波距離センサ設定

初期設定 (2)

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;
```

```
char cmd[5][9] = { "SWon/off", "Comd 00 ", "Comd 01 ", "Comd 02 ", "Comd 03 "};
```

IRコマンド名5個

3. デモ・スケッチ (setup関数)

setup関数

```
void setup() {
  delay(1000);
  while (!Serial && digitalRead(2));
  Serial.begin(BAUDRATE);
  lcd_init();// I2C LCD 初期化
  lcd_clear();
  lcd_setCursor(0, 0); lcd_printStr("IoTAB ");
  lcd_setCursor(0, 1); lcd_printStr("SW on ");
  pinMode(2, INPUT_PULLUP);
  while (!Serial && digitalRead(2));
```

デジタルピン関連
初期設定

```
pinMode(TrigPin, OUTPUT);
pinMode(EchoPin, INPUT);
pinMode(Spk_Pin, OUTPUT);
pinMode(Led1Pin, OUTPUT);
pinMode(Led2Pin, OUTPUT);
pinMode(Led3Pin, OUTPUT);
pinMode(Led4Pin, OUTPUT);
pinMode(Led5Pin, OUTPUT);
pinMode(Led6Pin, OUTPUT);
pinMode(TABSPin, OUTPUT); digitalWrite(TABSPin, HIGH);
digitalWrite(Led5Pin, HIGH); // D07 pin 3GIM SW off
char pr[8]="";
```

LEDの初期設定

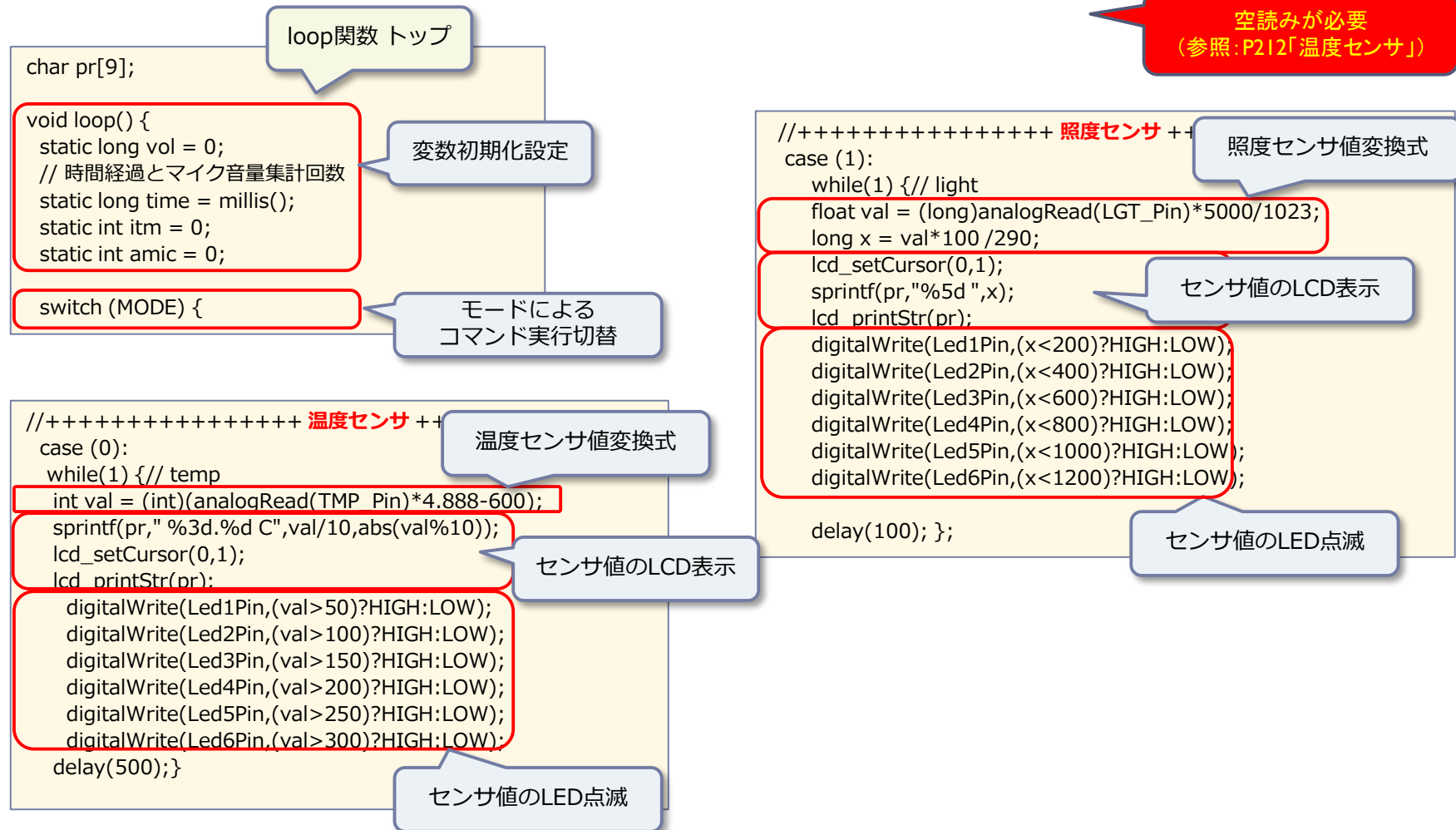
※ 可変抵抗器を使いLCDに表示されたコマンドを選択し、タクトスイッチを押す

```
// IR setting
/* irout_bit = digitalPinToBitMask(IROutputPin);
   irout_port = digitalPinToPort(IROutputPin);
   irin_bit = digitalPinToBitMask(IRInputPin);
   irin_port = digitalPinToPort(IRInputPin);*/
pinMode(But_Pin, INPUT);
```

```
// IR set
while (digitalRead(But_Pin) == LOW) {
  // EEPROM.clear();
  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);
    dumpIR();
    saveIR(i);
  }
  lcd_clear();
}
```

モードの設定

4. デモ・スケッチ (loop関数①)



4. デモ・スケッチ (loop関数②)

//+++++++ 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;
```

音の平均値取得

```
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;
```

音センサ値の取得

```
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();
}
```

3秒間の音センサの平均値取得 LCD表示

```
lcd_setCursor(4,1);
sprintf(pr,"%4d ",mic);
lcd_printStr(pr);
```

音センサ値取得
LCD表示

```
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);
```

音センサ値による
LED点滅

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

//+++++++ 手拍子カウント +++++++

case (3): // MIC

```
while(1){
  lcd_setCursor(0,1);
  lcd_printStr("Ready...");
  int nclup = checkSound();
```

手拍子カウント
モジュール

```
lcd_setCursor(0,1);
sprintf(pr,"clup= %2d",nclup);
lcd_printStr(pr);
```

手拍子カウント数
LED表示

```
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);
};
```

手拍子のカウント数
によるLED点滅

4. デモ・スケッチ (loop関数③)

```
//+++++++ 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

4. デモ・スケッチ (loop関数④)

```
//+++++++ 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

```
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
}
```

```
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

時間設定関数
可変抵抗器を利用
0 - 60秒 : 1秒間隔
1 - 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

4. デモ・スケッチ (loop関数⑤)

```
//+++++++ LED ++++++
```

4種類のLED点滅

```
case (6):
  while (1) {
    for (int j = 0; j < 3; j++) {
      for (int i = 3; i < 9; i++) {
        digitalWrite(i, HIGH);
        delay(50);
      }
      for (int i = 3; i < 9; i++) {
        digitalWrite(i, LOW);
        delay(50);
      }
      for (int j = 0; j < 3; j++) {
        for (int i = 3; i < 9; i++)
        {
          digitalWrite(i, HIGH);
          delay(100);
          digitalWrite(i, LOW);
        }
      }
      for (int i = 3; i < 9; i++) {
        for (int j = 0; j < 3; j++) {
          digitalWrite(i, HIGH); delay(50);
          digitalWrite(i, LOW); delay(50);
        }
      }
    }
  }
}
```

```
//+++++++ 超音波距離センサ ++++++
```

HC-SR04センサの
距離算出処理

```
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); };
```

LED点灯

LCD表示

4. デモ・スケッチ (loop関数⑥)

```
//+++++++ デルミン ++++++
case (8): //
  while (1) {
    digitalWrite(TrigPin, HIGH);
    delayMicroseconds(CTM);
    digitalWrite(TrigPin, LOW);
    delayMicroseconds(CTM); digitalWrite(TrigPin, LOW);
    float dis = (float)pulseIn(EchoPin, HIGH) * 0.017;
    if (dis < 10) { tone(Spk_Pin, fq[0], 100); }
    else if (dis < 15) { tone(Spk_Pin, fq[1], 100); }
    else if (dis < 20) { tone(Spk_Pin, fq[2], 100); }
    else if (dis < 25) { tone(Spk_Pin, fq[3], 100); }
    else if (dis < 30) { tone(Spk_Pin, fq[4], 100); }
    else if (dis < 35) { tone(Spk_Pin, fq[5], 100); }
    else if (dis < 40) { tone(Spk_Pin, fq[6], 100); }
    else if (dis < 45) { tone(Spk_Pin, fq[7], 100); }
    digitalWrite(Led1Pin, dis > 15);
    digitalWrite(Led2Pin, dis > 20);
    digitalWrite(Led3Pin, dis > 25);
    digitalWrite(Led4Pin, dis > 30);
    digitalWrite(Led5Pin, dis > 35);
    digitalWrite(Led6Pin, dis > 40);
    sprintf(pr, "%4d.%1dcm", (int)dis, (int)(dis * 10.0) % 10);
    // Serial.println( pr );
    lcd.setCursor(0, 1);
    lcd_printStr(pr);
  }
}
```

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）

4. デモ・スケッチ (loop関数⑦)

```
//+++++++ 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)

4. デモ・スケッチ (loop関数⑧)

```
//+++++++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

```
//+++++++赤外線IR+++++++
case (10):
while (1) {
    lcd_setCursor(0, 0);
    lcd_printStr("commando");
    byte com;
    do {
        com = map(analogRead(Reg_Pin), 0, 1023, 0, 5);
        lcd_setCursor(0, 1);
        lcd_printStr(cmd[0, com]);
    } while (digitalRead(But_Pin) == HIGH);
    loadIR(com);
    dumpIR();
    controlIR();
}
```

可変抵抗による
コマンド選択

※IR関連処理は「赤外線リモコン」ページにて説明

※メロディデータは、グローバルデータとして定義済 (宣言ページ)

4. デモ・スケッチ (loop関数⑨)

```
//+++++++ 3G ツイッター ++++++
case (11):// 3G ツイッター
while (1) {
  lcd.setCursor(0, 0);
  lcd_printStr("3G Ready");
  Serial1.begin(BAUDRATE);
  pinMode(7, OUTPUT); digitalWrite(7, HIGH); delay(100);
  digitalWrite(7, LOW);
  unsigned long tim = millis();
  String str = "";
  do {
    str = Serial1.readStringUntil('\n');
  }
  while (str.indexOf("3GIM") < 0 && millis() - tim < 15000);

  if (millis() - tim >= 15000) {
    lcd.setCursor(0, 1);
    lcd_printStr("Error..."); delay(3000); break;
  } else {
    while (1) {
      lcd.setCursor(0, 1);
      lcd_printStr("Connect.");
      Serial1.println("$YT");
      while (!Serial1.available());
      str = Serial1.readStringUntil('\n').substring(7);
      analogRead(A1); // ダミー読み込み
      float temp = 207.26 - 0.3923 * analogRead(A1);
      str = "$WP http://" + server + "%token=" + token +
"&status=TIME=" + str + " / temp=" + String(temp) + " C ¥";
      delay(100);
    }
  }
}
```

3G立上げ

ツイートデータ

```
lcd.setCursor(0, 0);
lcd_printStr("$WPtweet");
iemSerial.println(str);
delay(100);
str = String(temp) + " C ";
str.toCharArray(pr, str.length() + 1);
lcd.setCursor(0, 1);
lcd_printStr(pr);
str = "";
String st = "";
tim = millis();
do {
  str = iemSerial.readStringUntil('\n');
  str.toCharArray(pr, str.length() + 1);
  lcd.setCursor(0, 1); lcd_printStr(pr);
  // Serial.println(str);
} while (str.indexOf("$WP") < 0 && millis() - tim < 40000);
delay(1000);
lcd.setCursor(0, 1);
if (str.indexOf("OK") > 0) {
  lcd_printStr("Tweet OK");
  while (1);
}
else {
  lcd_printStr("Tweet NG");
}
}
```

ツイート

ツイートの応答

4. デモ・スケッチ (loop関数⑩)

```
//+++++ 3G メール送信 ++++++
case (12): // 3G メール
  while (1) {
    lcd.setCursor(0, 0);
    lcd_printStr("3G Ready");
    Serial1.begin(BAUDRATE);
    pinMode(7, OUTPUT); digitalWrite(7, HIGH); delay(100);
    digitalWrite(7, LOW);
    unsigned long tim = millis();
    String str = "";
    do {
      str = Serial1.readStringUntil('\n');
    } while (str.indexOf("3GIM") < 0 && millis() - tim < 15000);

    if (millis() - tim >= 15000) {
      lcd_printStr("Error..."); delay(3000); break;
    } else {
      while (1) {
        lcd.setCursor(0, 1);
        lcd_printStr("Connect.");
        lcd.setCursor(0, 0);
        lcd_printStr("$WG mail");
        analogRead(A1); // ダミー読み込み
        float temp = 207.26 - 0.3923 * analogRead(A1);
        str = "$WG http://://" + mail_sever + "/?email=" +
          e_mail + "&cont=TEMP=%20" + String(tesendmail.phpmp) +
          "%20C";
        delay(200);
```

3G立上げ

メールデータ

```
Serial1.println(str);
str = String(temp) + " C ";
str.toCharArray(pr, str.length() + 1);
lcd.setCursor(0, 1);
lcd_printStr(pr);
tim = millis();
do {
  str = Serial1.readStringUntil('\n');
} while ((str.indexOf("$WG") < 0) && (millis() - tim < 40000));
lcd.setCursor(0, 1);
if (str.indexOf("OK") > 0) {
  lcd_printStr("Mail OK");
  while (1);
}
else {
  lcd_printStr("Mail NG");
}
delay(1000);
}
```

メール送信

メールの応答

4. デモ・スケッチ (loop関数⑪)

```
//+++++ Assisted GPS ++++++
case (13): // GPS
  while (1) {
    lcd_setCursor(0, 0);
    lcd_printStr("3G Ready");
    Serial1.begin(BAUDRATE);
    pinMode(7, OUTPUT); digitalWrite(7, HIGH); delay(100);
    digitalWrite(7, LOW);
    unsigned long tim = millis();
    String str = "";
    do {
      str = Serial1.readStringUntil('\n');
    } while (str.indexOf("3GIM") < 0 && millis() - tim < 15000);

    if (millis() - tim >= 15000) {
      lcd_printStr("Error..."); delay(3000); break;
    } else {
      while (1) {
        Serial1.println("$YA 1"); delay(20);
        Serial1.println("at+wppp=2,4," + SIM_APN); delay(10);
        while (Serial1.readStringUntil('\n').indexOf(">>") < 0);
        lcd_setCursor(0, 1);
        lcd_printStr("Connect.");
        lcd_setCursor(0, 0);
        lcd_printStr("GPS get ");
        String gps = GET_GPS();
        if (gps.length() > 0) {
          str="$WGhttp://" + mail_sever +
            "/sendmail.php?email="+e_mail+"&cont=https://www.google.co.jp/m
            aps/place/" + gps;
          //https://www.google.co.jp/maps/place/35°38'47.3"N+139°35'57.5"E
```

3G立上げ

A-GPS設定

httpGETデータ

```
delay(200);
Serial1.println(str);
tim = millis();
do {
  str = Serial1.readStringUntil('\n');
} while ((str.indexOf("$WG") < 0) && (millis() - tim < 40000));
lcd_setCursor(0, 1);
if (str.indexOf("OK") > 0) {
  lcd_printStr("GPS OK");
  while (1);
}
else {
  lcd_printStr("GPS NG");
}
delay(1000);
}
}

//===== GET GPS =====
String GET_GPS() {
  Serial1.println("$LG x 0");
  String gps;
  do {
    gps = Serial1.readStringUntil('\n');
    // Serial.println("GPS = " + gps);
  } while (gps.indexOf("$LG") < 0);
  if (gps.indexOf("OK") > 0) {
    gps = gps.substring(7);
    gps.replace(" ", ",");
  } else gps = "";
  // Serial.println("GPS = " + gps);
  return gps;
}
```

GPSのhttpGET

httpGET応答

5. デモ・スケッチ (funcTitle関数)

■ 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; // オト
    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; // ホセンサ
    case 3: { char ttl[]={0xC3,0xCB,0xDE,0xAE,0xB3,0xBC,' ',' '}; strcpy(title,ttl);}; break; // テレメータ
    case 4: { char ttl[]={0xC3,0xB2,0xBA,0xB3,' ',' ',' '}; strcpy(title,ttl);}; break; // タイム
    case 5: { char ttl[]={0xC0,0xB2,0xCF,' ',' ',' '}; strcpy(title,ttl);}; break; // タイマ
    case 6: { char ttl[]={0xB6,0xBF,0xB8,0xC1,' ',' ',' '}; strcpy(title,ttl);}; break; // カリタ
    case 7: { char ttl[]={0xB7,0xAE,0xD8,0xBE,0xDD,0xBB,' ',' '}; strcpy(title,ttl);}; break; // キリセンサ
    case 8: { char ttl[]={0xC1,0xD9,0xC4,0xBE,0xDD,0xBB,' ',' '}; strcpy(title,ttl);}; break; // フルトセンサ
    case 9: { char ttl[]={0xD2,0xDB,0xC3,0xDE,0xA8,' ',' '}; strcpy(title,ttl);}; break; // メディ
  }
  lcd_setCursor(0,0);
  lcd_printStr(title);
}
```

タイトル
LCD表示

■ コマンド LED点灯表示

コマンド番号を
LEDでビット表示

```
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);
  }
}
```

6. デモ・スケッチ (checkSound関数)

■ 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);
    }
  }
}
```

最初の音センサ値
平均値

最初の音センサ値
平均値

```
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
}
}
```


7. デモ・スケッチ (i2c_lcdライブラリ)

■ I2C_LCD_Lib.ino

【注意】本スケッチ利用の際は、「#include <Wire.h>」の宣言が必要

```
#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.beginTransaction(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.beginTransaction(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));
}
```

第4章 応用デモ・スケッチ

1. Genuino101 (BLE機能) + IoTABシールド

IoTABシールド上のLEDとスピーカをスマホから制御するスケッチ

```
#include <CurieBLE.h>

BLEPeripheral blePeripheral; // create peripheral instance

BLEService ledService("19B10000-E8F2-537E-4F6C-D104768A1214"); // create service

// create switch characteristic and allow remote device to read and write
BLECharCharacteristic switchChar("19B10001-E8F2-537E-4F6C-D104768A1214", BLERead | BLEWrite);

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

  // set the local name peripheral advertises
  blePeripheral.setLocalName("LEDCB");
  // set the UUID for the service this peripheral advertises
  blePeripheral.setAdvertisedServiceUuid(ledService.uuid());

  // add service and characteristic
  blePeripheral.addAttribute(ledService);
  blePeripheral.addAttribute(switchChar);

  // assign event handlers for connected, disconnected to peripheral
  blePeripheral.setEventHandler(BLEConnected, blePeripheralConnectHandler);
  blePeripheral.setEventHandler(BLEDisconnected, blePeripheralDisconnectHandler);

  // assign event handlers for characteristic
  switchChar.setEventHandler(BLEWritten, switchCharacteristicWritten);
  // set an initial value for the characteristic
  switchChar.setValue(0);

  // advertise the service
  blePeripheral.begin();
  Serial.println("Bluetooth device active, waiting for connections...");
}

void loop() {
  // poll peripheral
  blePeripheral.poll();
}

void blePeripheralConnectHandler(BLECentral& central) {
  // central connected event handler
  Serial.print("Connected event, central: ");
  Serial.println(central.address());
}

void blePeripheralDisconnectHandler(BLECentral& central) {
  // central disconnected event handler
  Serial.print("Disconnected event, central: ");
  Serial.println(central.address());
}

void switchCharacteristicWritten(BLECentral& central, BLECharacteristic& characteristic) {
  byte in = switchChar.value();
  Serial.println("Characteristic event, written: " + String(in));
  switch (in) {
    case 0:
      for(int i=3; i<9; i++) {pinMode(i,OUTPUT); digitalWrite(i,LOW); }
      break;
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
    case 6:
      pinMode(in+2,OUTPUT); digitalWrite(in+2,HIGH);
      break;
    case 7:
      tone(10,250,1000); delay(1000);
      break;
  }
}
```

G101_IoTAB_BLE_LED_BUZZER_LCD

第5章 温度ステーション

BLEを使った演習。
Arduino側で計測した温度をBLEを使って近くのスマホやタブレットへ送信して、表示する。

1. BLE温度ステーション（概要）

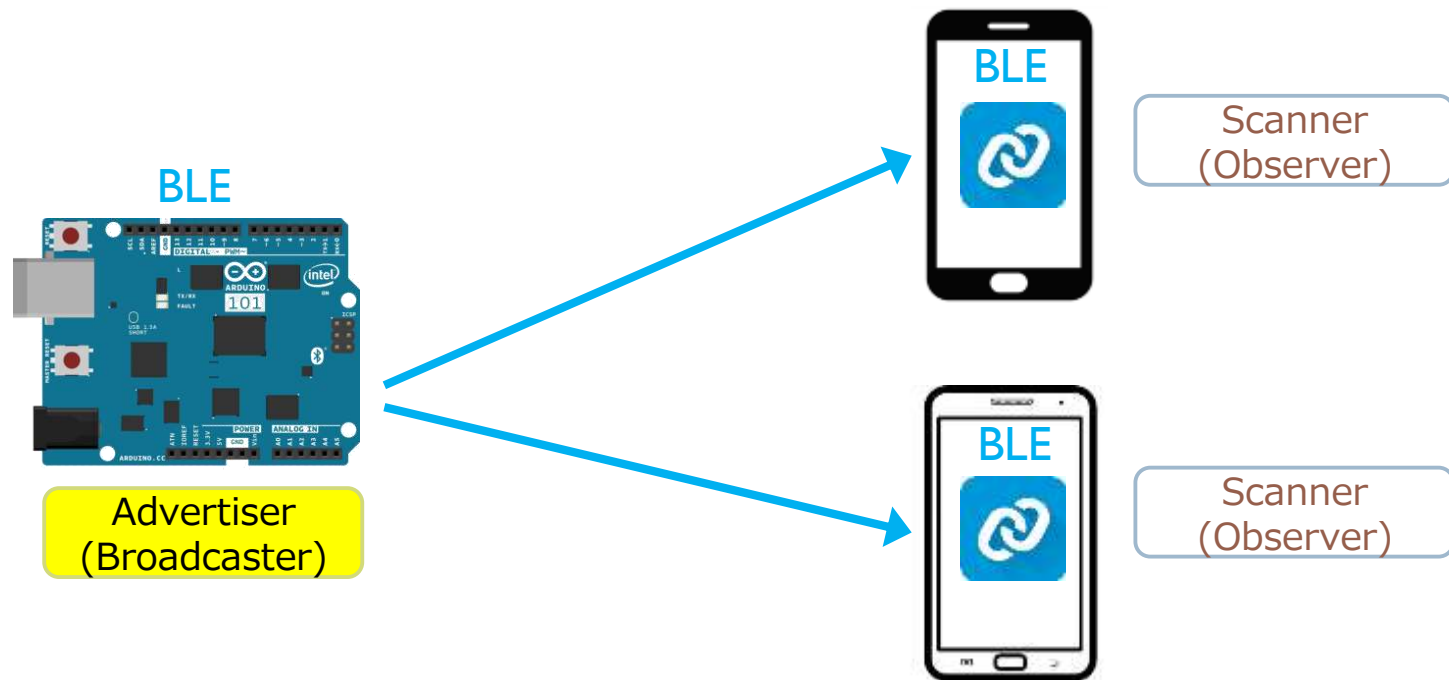
- ▶ スマホのアプリ「nRF Master Control」を使って動作を確認する
 - ▶ Google Playで入手&インストール（App Storeでは「nRF Control」でプログラムは少し異なる）



- ▶ IoTABシールド上の温度センサで温度を計測して、周囲へBLEを使って温度情報を発信する
 - ▶ BLEは、Gendduino I01に標準搭載されている
- ▶ BLEの規格はかなり複雑なので、サンプルスケッチCurieBLEをベースに改造するのが良い
 - ▶ 今回は、GATTプロファイルの一つである「環境センシング(UUID=0x181A)」を利用する

1. BLE温度ステーション（概要）

- ▶ 101から環境センシングサービスを提供していることをアドバタイズで周りへ発信
- ▶ スマホ側で「CONNECT」すると、101から温度を2秒間隔で計測・送信する
 - ▶ 同時に1対n



2. BLE温度ステーション（スケッチ）

▶ スケッチ (Temperature_Station_with_BLE.ino)

```

/*
 * Temperature Station with BLE
 */

#include <CurieBLE.h>

const int PIN_CONNECTED_LED = 13;
const char *BLE_LOCAL_NAME = "IoTShield <NO>";
const long UPDATE_INTERVAL_MS = 2000L;

BLEPeripheral myBLE;
BLEService bleEnvironmental("181A");
BLEUnsignedShortCharacteristic bleHumidityPC("2A6F", BLERead |
BLENotify);
BLEShortCharacteristic bleTemperatureC("2A6E", BLERead |
BLENotify);
long whenSensedMs = 0;    // time when the sensor was last read

void setup() {
  delay(3000);
  Serial.begin(9600);
  pinMode(PIN_CONNECTED_LED, OUTPUT);
  digitalWrite(PIN_CONNECTED_LED, LOW);
  // Configure the BLE stack
  myBLE.setLocalName(BLE_LOCAL_NAME);
  myBLE.setAdvertisedServiceUuid(bleEnvironmental.uuid());
  myBLE.addAttribute(bleEnvironmental);
  myBLE.addAttribute(bleHumidityPC);
  myBLE.addAttribute(bleTemperatureC);
  // Initialize our BLE Characteristics
  setTemperatureAndHumidity();
  myBLE.begin();
}

```

自分の番号に
置き換える

温度は2秒間隔
で更新

BLEを環境センシング
として利用するための
各種設定

スマホにつながると
1を返す

```

void loop() {
  BLECentral central = myBLE.central();
  if (central) {
    digitalWrite(PIN_CONNECTED_LED, HIGH);
    Serial.print("Connected to central: ");
    Serial.println(central.address());
    while (central.connected()) {
      // Connected !
      long nowMs = millis();
      if (nowMs - whenSensedMs >= UPDATE_INTERVAL_MS) {
        whenSensedMs = nowMs;
        setTemperatureAndHumidity();
      }
    }
    // Disconnected !
    digitalWrite(PIN_CONNECTED_LED, LOW);
    Serial.print("Disconnected from central: ");
    Serial.println(central.address());
  }
}

```

スマホにつながっている
間は、一定間隔で温
度を更新する

湿度は計測できない
ので、0を設定する

```

void setTemperatureAndHumidity() {
  bleHumidityPC.setValue(0);
  float mV = analogRead(1) * 3.23;
  float temperatureC = (mV - 1712.5) / (-8.20);
  short newTemperatureC = (short) (temperatureC * 100.0 + 0.5);
  bleTemperatureC.setValue(newTemperatureC);
  Serial.print("Temperature: ");
  Serial.println(temperatureC);
}

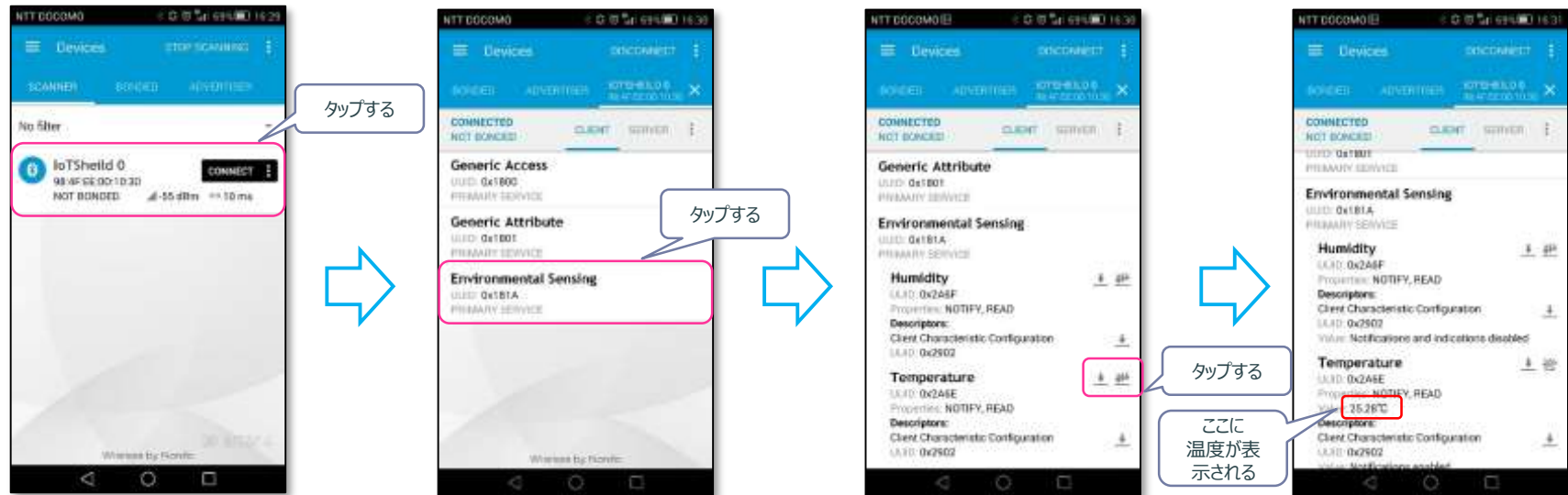
```

IoTABシールド上の
温度センサを読み取
り、BLEに設定する

Temperature_Station_with_BLE.ino

3. BLE温度ステーション（アプリの使い方）

- ▶ アプリ「nRF Master Control」(Androidのみ)を使った動作の確認方法は、下記の通り(Android用アプリの例)

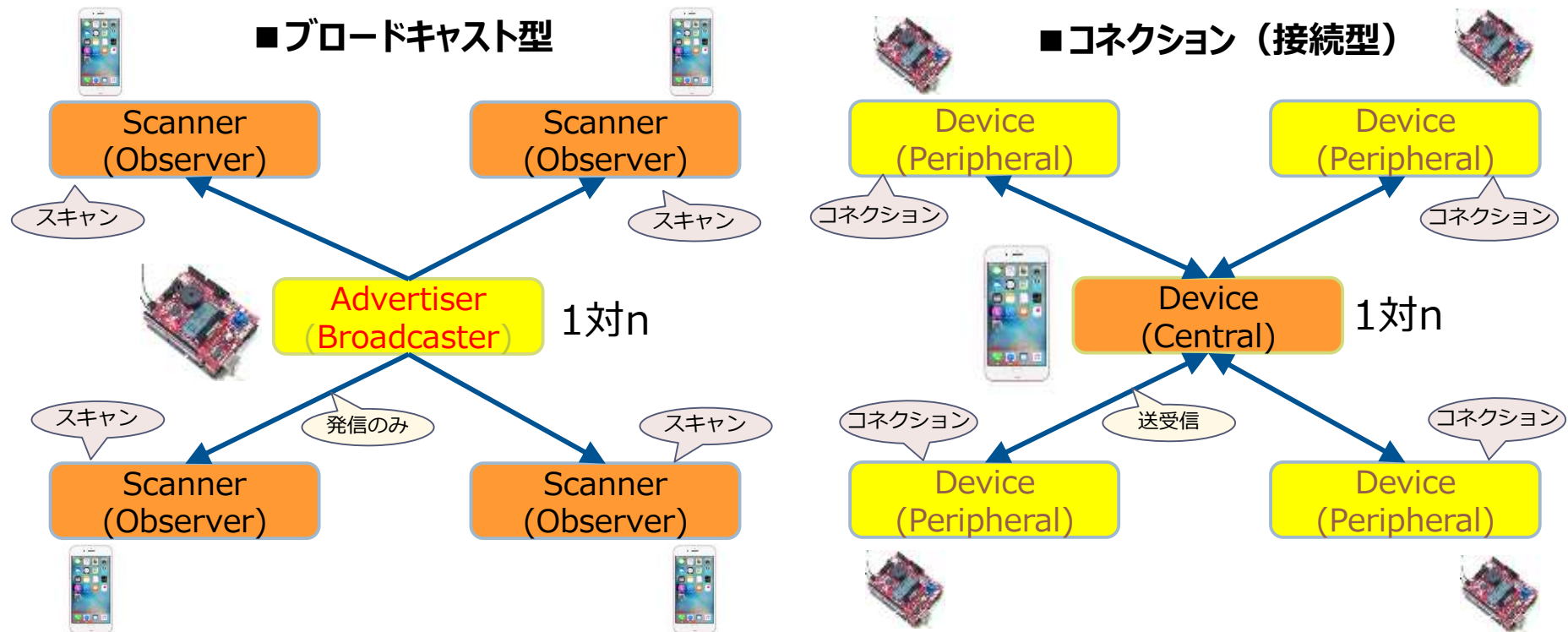


- ▶ スマホのBLEをONにしておく必要がある

4. BLE温度ステーション（BLEについて）

Genuino101 BLEのネットワークトポロジーについて（複合的に設定も可能）

この役割はGAP（汎用アクセスプロファイル）によって設定（GAPによって役割・モード・手順・セキュリティなどを設定）



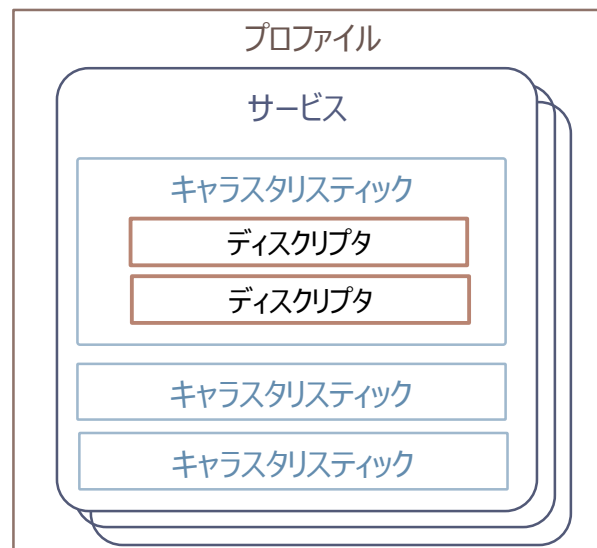
- ブロードキャストで情報を周りに発信する（受信なし）
- 通信を暗号化できない（iBeaconの考え）

- セントラル（スマホなど）と周りのペリフェラル（G101など）と送受信する（逆も可能）
- 暗号化も可能

4. BLE温度ステーション（BLEについて）

▶ GATT、プロファイル

- ▶ BLE端末同士が接続してデータの送受信を行う際は、GATTという仕様に基づいてやりとりします。
 - ▶ データを転送する際の最小単位を「キャラクタリスティクス」と呼ぶ
 - ▶ 「ディスクリプタ」はキャラクタリスティクスを構成する属性を定義する
 - ▶ キャラクタリスティクスが集まって、1つの機能を構成するものを「サービス」と呼ぶ
 - ▶ 複数のサービスが集まって構成される全体の機能を「プロファイル」と呼ぶ
 - ▶ プロファイルに基づいて、BLE機器同士が通信を行う
- ▶ プロファイルはBluetooth SIGという業界団体が標準を策定しているが、メーカーが独自のプロファイルを申請することもできる。



もくじ
Arduino裏ワザ技術

第Ⅳ編 補足技術編



Arduino裏ワザ技術

「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。

unsigned 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日間でオーバーフローし、ゼロに戻る。ここで、戻り値：実行時からの時間（ミリ秒）

unsigned 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分間でオーバーフローし、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。ここで、戻り値：実行時からの時間（マイクロ秒）

- ▶ 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（ms ミリ秒）だけ中断

ここで、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

ここで、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

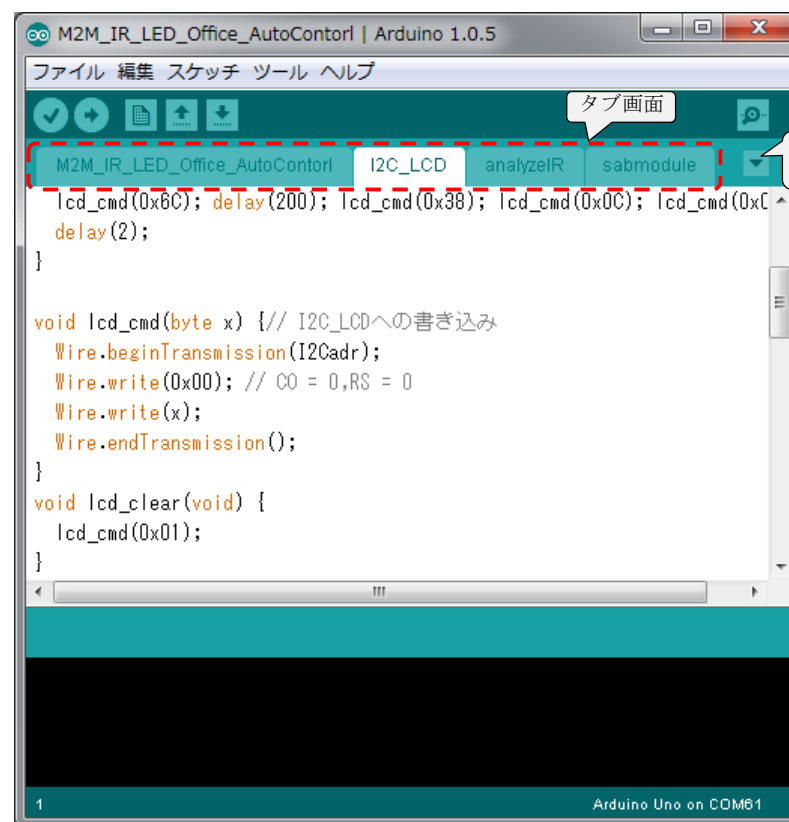
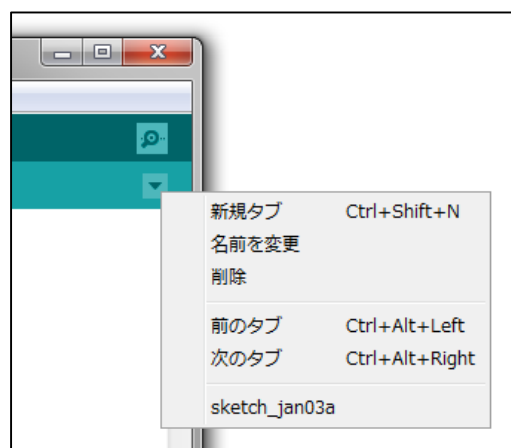
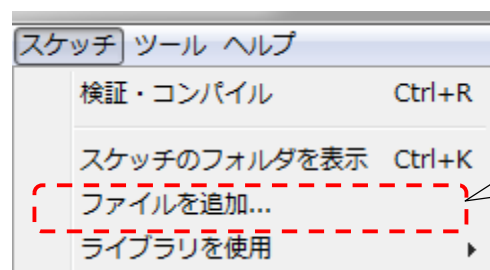
```
void setup() {
  pinMode(13,OUTPUT);
}
void loop(){
  digitalWrite(13,HIGH);
  delay(1000);
  digitalWrite(13,LOW);
  delay(1000);
}
```



```
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、1 K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。

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



3. 不揮発メモリーEEPROMを使う②

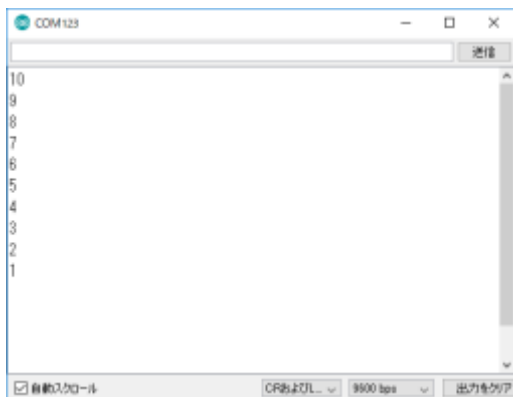
Arduino UNOでは、EEPROMとして1K(1024) バイトの不揮発メモリーが利用できます。
 Arduino UNOの電源を切ったり、リセットしても消えることはありません。
 大事なデータを保管・格納・利用するときに使うことができます。

■ 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 (リーフの場合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

```

COM123
* initial gVal data print
f= 99.99
b= 255
c= void
--> EEPROM put struct data
<-- EEPROM get struct data
* EEPROM gVal data print
f= 1.24
b= 26
c= Trillion-Node
  
```

■ サンプルスケッチ

```

#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() { }
  
```

構造体定義

15文字の文字配列

変数設定
gValには初期値設定

初期値表示

構造体データ読み込み

構造体データ書き込み

EEPROM読み込み表示

Basic_EEPROM_sample02.ino

4. 割り込み機能を使う

- ▶ Arduino には、割り込み機能があり、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 attachInterrupt(byte int, void (*fun)(void), int mode)
```

ここで、int : 割り込み番号 (0 または 1)

fun: 割り込みする関数名 (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: 割り込み関数を実行する条件

「LOW」 ピンの値がLOWの場合

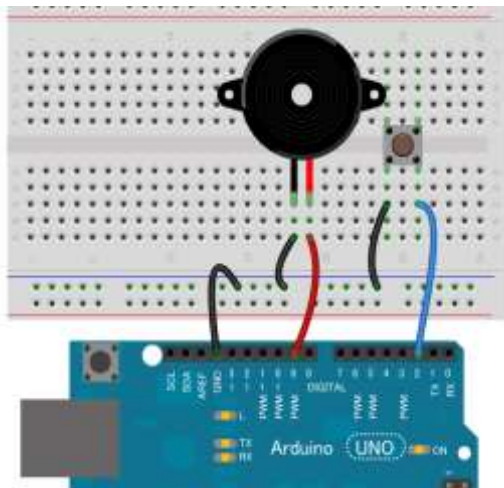
「CHANGE」 ピンの値が変わって場合

「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、割り込み番号 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



■ 割り込みを使った事例紹介

```
boolean sw=false;
void setup(){
  pinMode(2,INPUT_PULLUP); // 割り込みピン (タクトスイッチ)
  pinMode(9,OUTPUT);
  pinMode(13,OUTPUT); // Arduino 上のLED
  attachInterrupt(0,buzzer,CHANGE); // 割り込み処理関数
}
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）

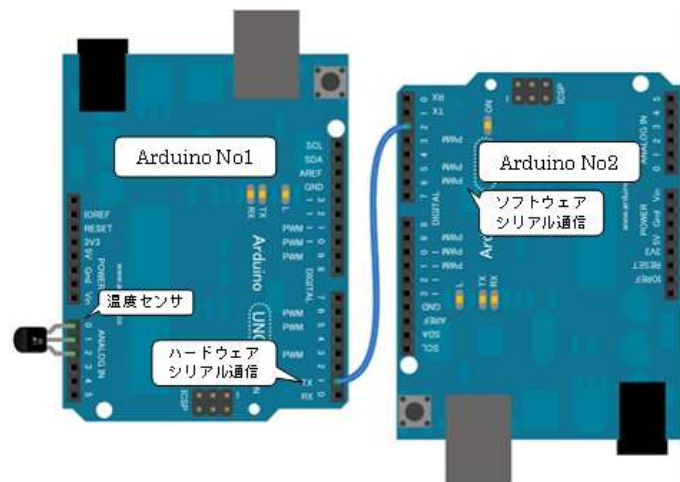
```
void Serial.begin(int speed) : 通信速度を設定し、通信を有効にする
    ここで speed : 通信速度（単位pbs : ビット/秒）で、300、1200～115200まで
void Serial.end() : 通信を無効（中断）とし、「D0」「D1」がデジタル入出力ピンとして有効利用可能
int Serial.available() : シリアルポートに到着しているバッファのバイト数を返す
    戻り値 : シリアルバッファにあるデータのバイト数
int Serial.read() : 受信データの読み込み（ポインタをずらす）
    戻り値 : 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

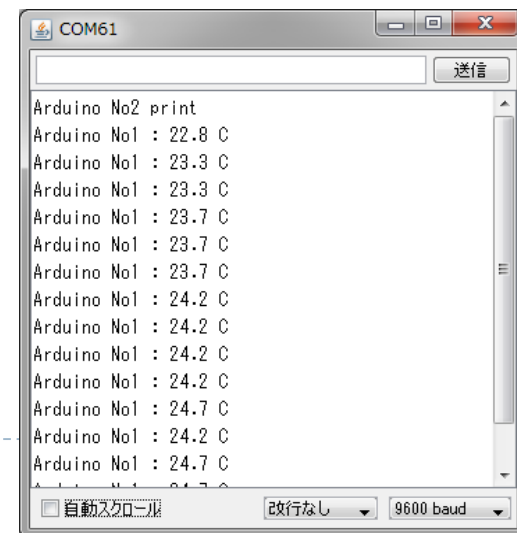
Extend_Serial_No2.ino

```
#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）

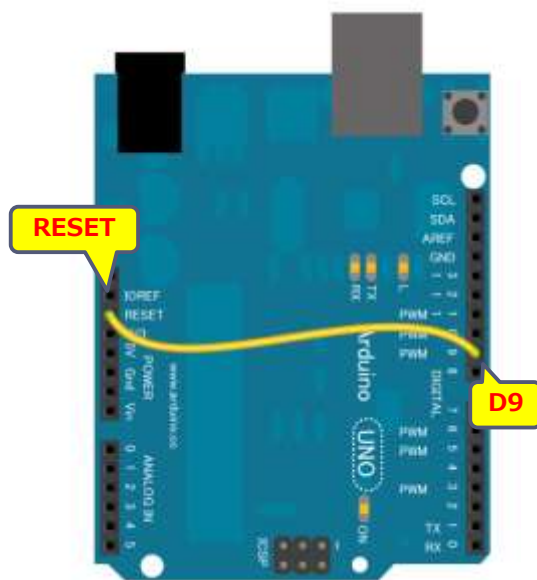


6. ソフトウェアリセットの実現方法

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_Software_Reset.ino

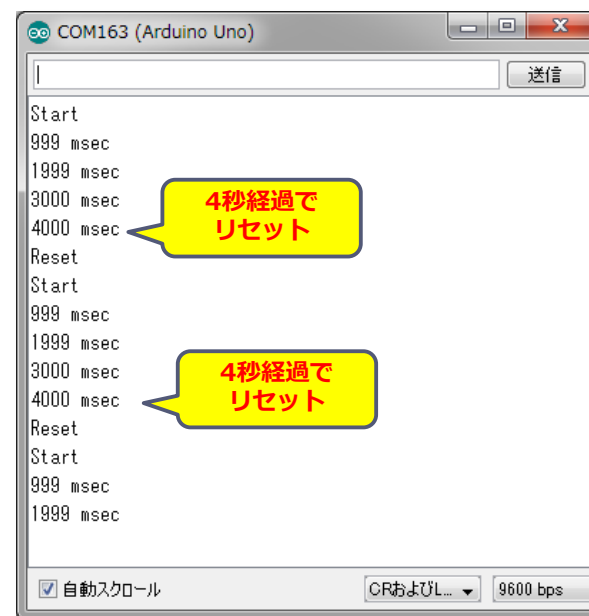
```
void setup() {
  Serial.begin(9600);
  Serial.println("Start");
}

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)

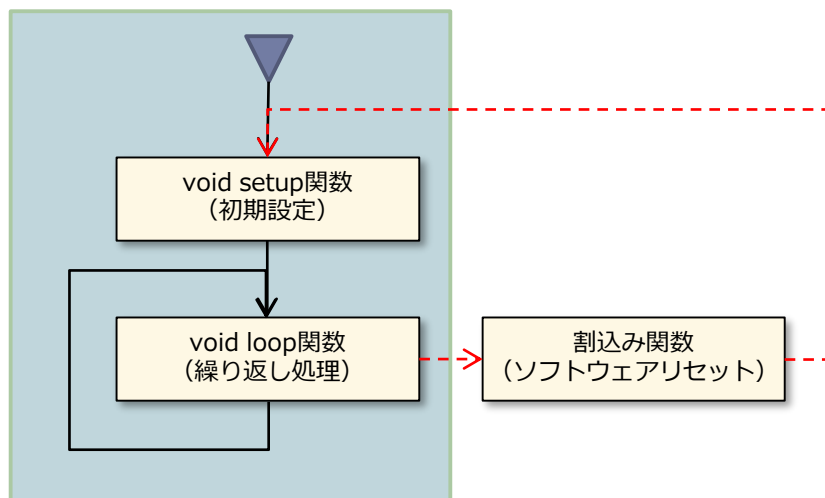


RESETピンと接続
したピンの
pinMode設定で
リセット

7. ソフトウェアリセットと割り込み処理の使い方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、もう一方では割り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、割り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り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

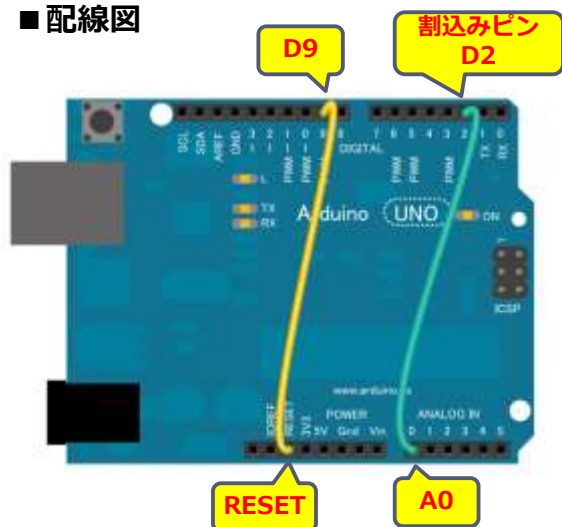
ここでは、2つの技術ポイントが必要となります。

- 1) 割り込み処理を起こすタイミング
 - 2) 割り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の割り込みピン**を知っておく必要があります。Arduino UNO の場合は、D2(INT 0) 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 (INT 4) となっています。

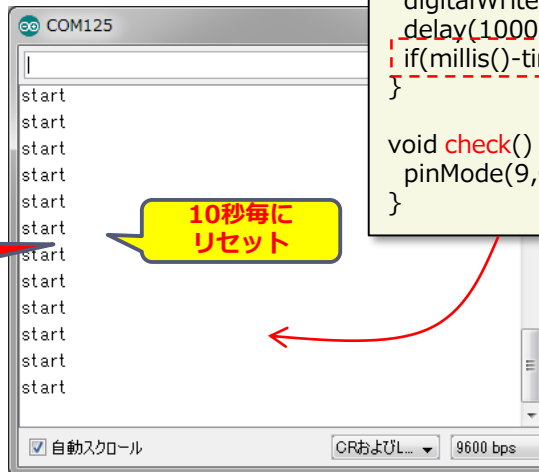
7. ソフトウェアリセットと割り込み処理の使い方②

■ 配線図



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

■ シリアルモニタ画面



■ 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D2を後に設定

Tabrain
LED 3

割り込み関数定義

時間で割り込み
(10秒後)

pinMode設定
でリセット

(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	<code>float val=analogRead(Ax)*1023.0 * R</code>	R（抵抗値）
タクトスイッチ（4.3節）	入力D0～D19	<code>pinMode(Dx,INPUT_PULLUP);</code> <code>boolean sw = digitalRead(Dx);</code>	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	<code>pinMode(Dx,OUTPUT);</code> <code>digitalWrite(Dx,hl);</code> <code>delay(sc);</code> //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	<code>analogWrite(Pwm,Px);</code>	Px：0(0V)～255(5V)
圧電スピーカ（SPT08）（5.2節、5.4節）	出力D0～D19	<code>pinMode(Dx,OUTPUT);</code> <code>tone(Dx,hz,sc);</code>	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	<code>analogWrite(Pwm,Px);</code>	Px：0(0V)～255(5V)
アナログ温度センサー（LM61BIZ）（6.1節）	入力A0～A5	<code>int val=analogRead(Ax);</code> <code>float cel=(float)val*0.488-60.0</code>	val：0(0V)～1023(5V)
アナログ光センサー（CdS）（6.2節）	入力A0～A5	<code>int val=analogRead(Ax);</code>	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4 m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～80cm程度
液晶ディスプレイ (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D 3、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。

9. Arduinoの環境設定について

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→⑤）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
↓
- 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 「**スケッチブックの保存場所**」にある「libraries」フォルダ内
↓
- ④ ボード関連の「Libraries」フォルダ内（特定ボード限定）
↓
- ⑤ 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。

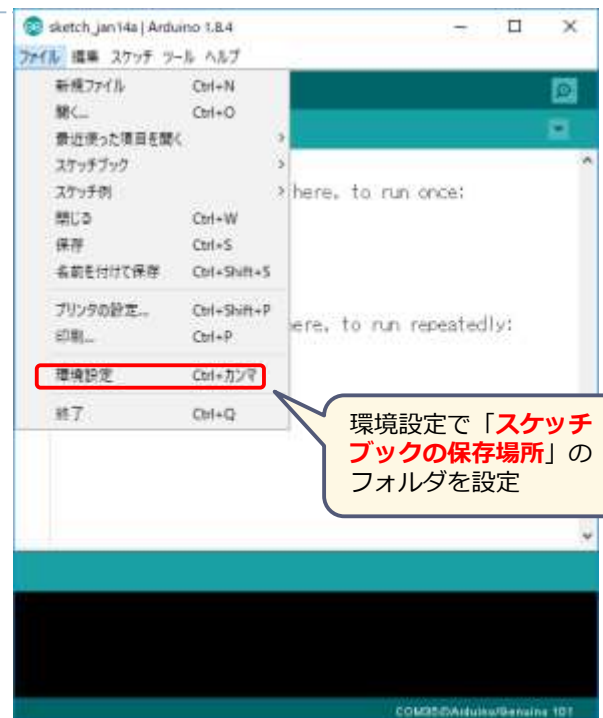
①のカレントフォルダを設定しない場合には、②の設定フォルダがカレントとなります。

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やすくなります。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勧めいたします。

③「スケッチブックの保存場所」に「libraries」フォルダを用意し、この中にライブラリ群を準備することで、管理しやすくなります。

④は、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

⑤の中にあるサンプルスケッチは、読み込みはできますが、書き込みはできませんので注意してください。



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。

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プ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

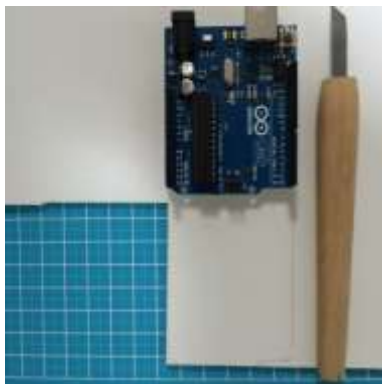
- 付録1 この本で扱った電子部品 (教材キット)
- 付録2 この本で扱った電子部品のスケッチ利用まとめ (早見表)
- 付録3 IoTABシールドの紹介
- 付録4 Arduino関連情報サイト



2014年2月17日発刊
(アマゾンよりお買い求めできます)

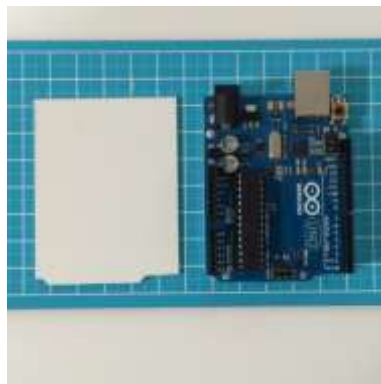
工作：静電気カバーシート（下駄）

- ▶ 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（2mm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）



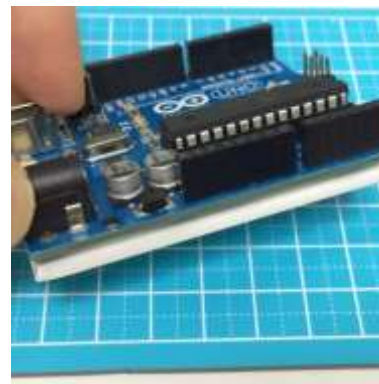
厚さ2mmほどのプラスチック製厚板とカッターを用意

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。



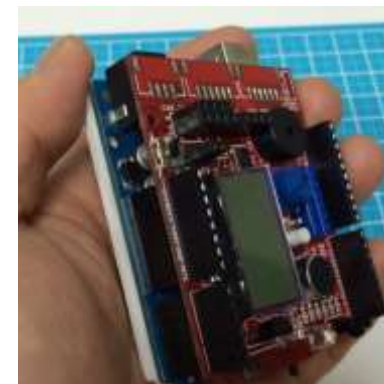
切り取ったら、さらに図のように凹凸のある部分も揃えて切り取ります。

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。

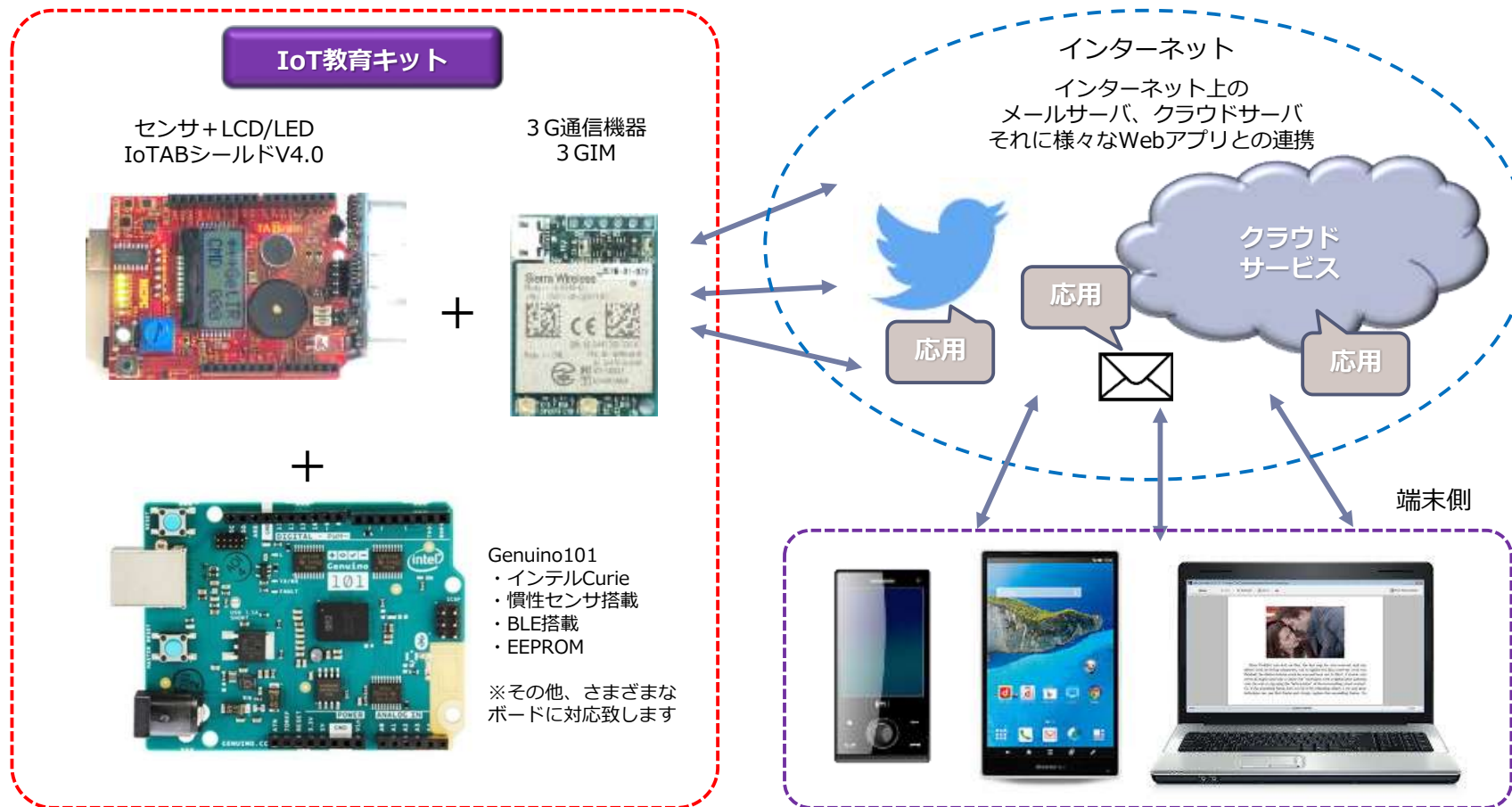


このように、手で持って操作してもArduinoの裏面にあるピンに触れることなく、操作できるようになり、静電気も気にする必要はありません。

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。

[ブレイク]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librxTxSerial.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

まとめ

- ▶ システムは、入力と処理と出力からなる
- ▶ デバイスは、センサとコンピュータとアクチュエータからなる
- ▶ モノ事は、シンプル（単純化）して見る
- ▶ モノ事は、シンプルに構築する（上から下に）
- ▶ 設計は トップダウンに、製造は ボトムアップに
- ▶ 理論（理屈・机上）と実践（実験・現場）との違いを知る（追及する）