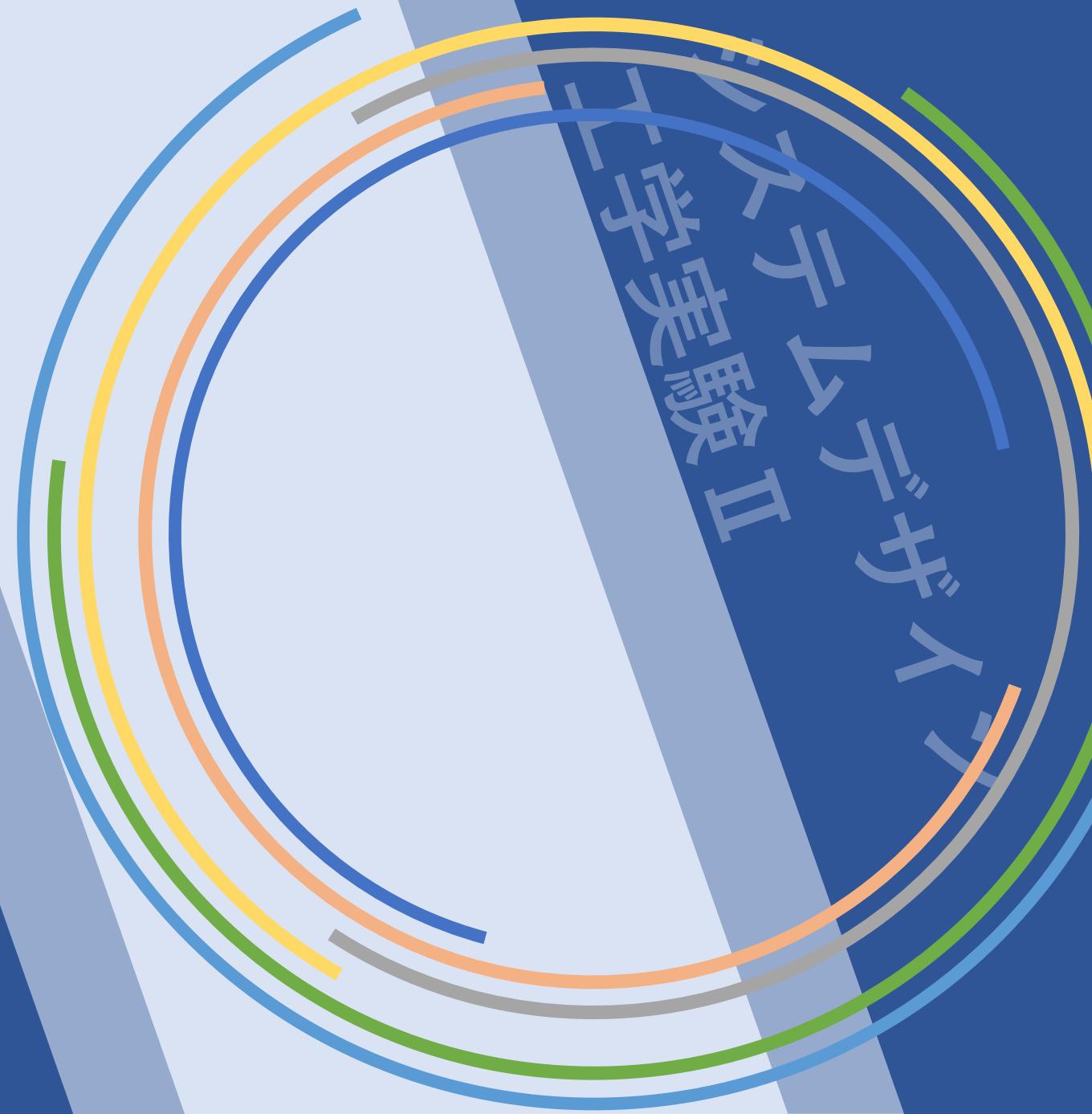




Internet of Things

担当： 西 宏章



この授業に関する情報はCanvasLMSで取り扱います
実験内容やレポートに関する質問などもLMSへ





実験内容を頭に入れておこう
実験内容と手順

IoTを学び作る — 実験の流れ

4

- Arduinoハンズオン
 - 多くの学生はすでに高校で扱っていることが多い
 - Raspberry PIを使って実習するのが筋ですが（検討中）
- 実際にみなさんがやること
 - 講義を聞く→実際に回路を作る→プログラムを書く→動作させる！
- IoTとは何か
 - IoTは技術もサービスも含む
 - Hiroaki Nishi, 2016
 - IoTとはシステムを考えることである
 - Masanori Takeshita, Fumi Koda, First IoT project textbook, 2016
 - “IT does not matter”
 - Nicholas G. Carr, Harvard Business Review, 2003
 - IoT技術には気づかなくなるぐらい身近になる



The Internet of Everything (IoE) – Aamer Azeemi, Cisco, 2013



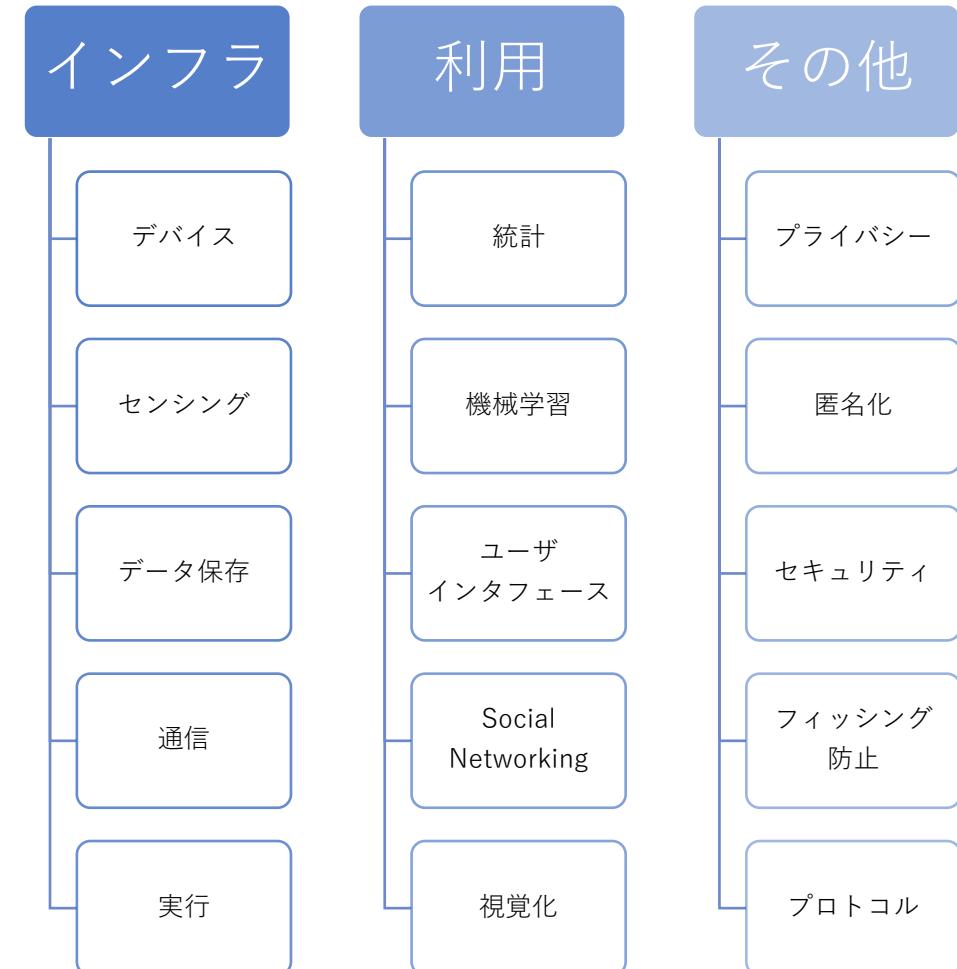
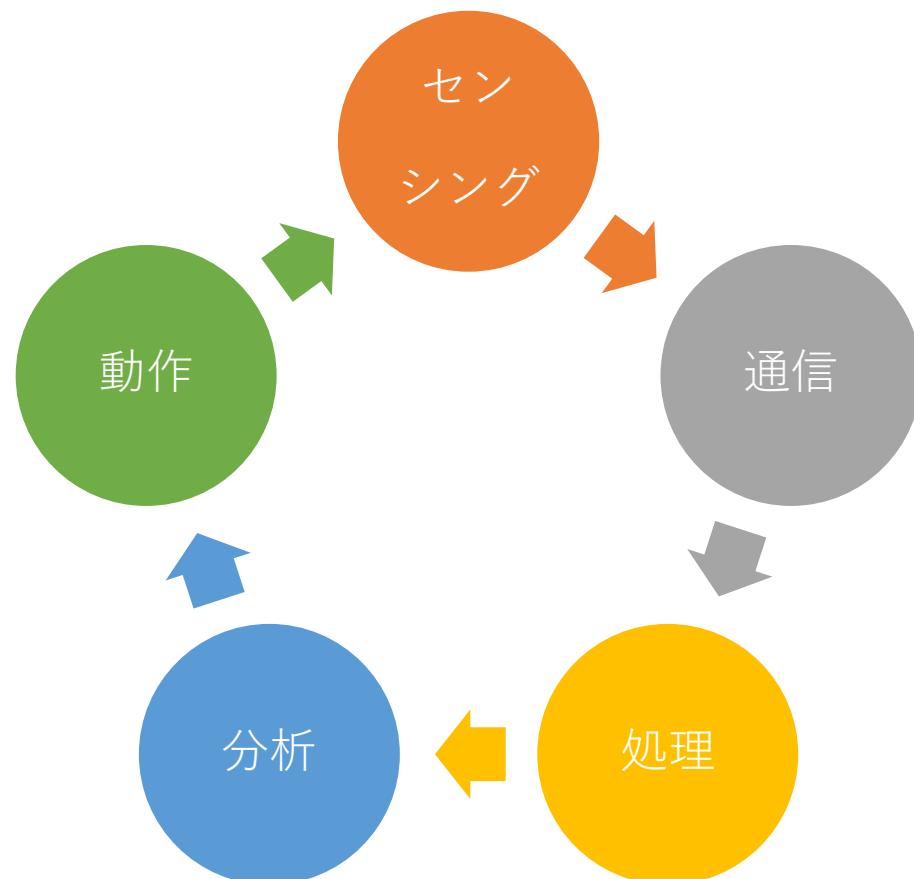
古くからの理想、技術進歩で実現へ

- 技術的進歩 (“IoT” by Kevin Ashton in 1999)
 - 電子回路、電気回路の製造技術進歩
 - 無線通信の進歩 4G/LTE/5G, Wi-Fi, BLE
 - バッテリ性能の向上 LIBなどの普及・LIBの管理もIoT
 - 取り扱いが簡単に Linux搭載・IoTプラットフォーム
 - 小型化 回路を直接触って作ることが難しくなっている
 - 低価格化・高性能化・信頼性向上
 - 様々な利用例・応用例
 - ケーブルレス（無線も電源も）
- 様々な用途
 - 趣味、研究、サービス
 - 実世界に近い場所で情報を扱う = 新鮮





IoTのエコシステム





センシング

- 2022年にセンサ1兆個
 - - Trillion sensor summit 2013
- 温度
- 湿度
- 照度
- エネルギ消費
- ガス(CO₂, SO_x, NO_x)
- 匂い
- 圧力
- 赤外線(行動検出など)
- 風向、風速など



IoTとは?(英訳)
– MONO WIRELESS, http://mono-wireless.com/jp/tech/Internet_of_Things.html

物理量
現象
行動

アナログ

デジタル

電圧
データ





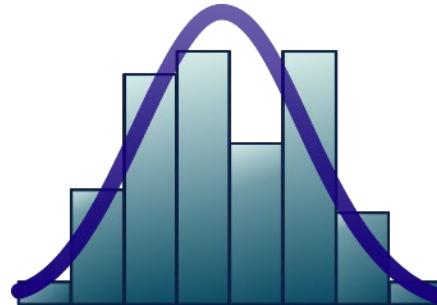
分析（データシステムの知能化とデザイン）

- データの処理には3段階ある - Hideyuki Tokuda, 2016

- 1 : 視覚化
- 2 : 最適化
- 3 : 予測

- 統計

- 機械学習



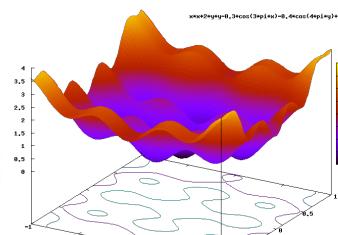
https://en.wikipedia.org/wiki/Wikipedia:Meetup/DC/Statistics_Edit-a-thon#/media/File:Fisher_iris_versicolor_sepalwidth.svg

Visualization



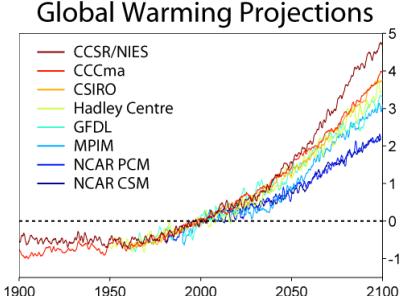
https://upload.wikimedia.org/wikipedia/commons/9/9b/Social_Network_Analysis_Visualization.png

Optimization

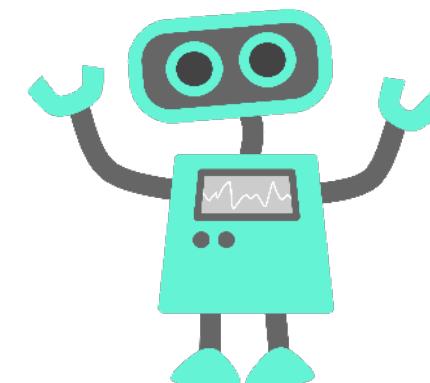


https://upload.wikimedia.org/wikipedia/commons/7/7a/B2_optimization_function.png

Prediction



https://upload.wikimedia.org/wikipedia/commons/a/aa/Global_Warming_Projections.png



<https://www.goodfreephotos.com/albums/vector-images/blue-robot-vector-art.png>



<https://pixabay.com/en/learn-note-sign-directory-897410/>





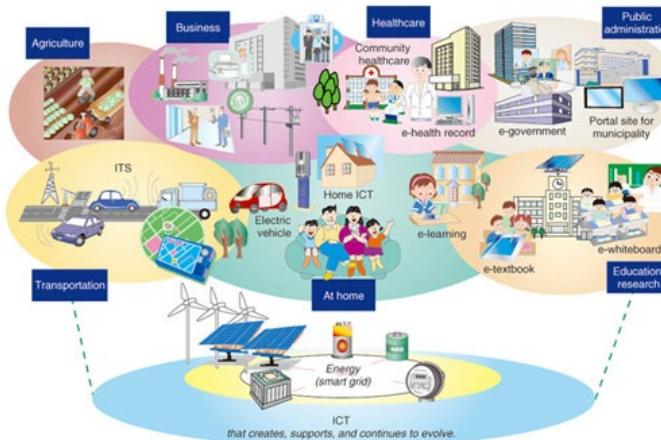
IoT的サービス例（マルチメディアデザイン）

家庭電力消費利用リコメンドサービス



Smart meter -
<http://www.tepco.co.jp/ep/private/smarterlife/smartmeter.html>

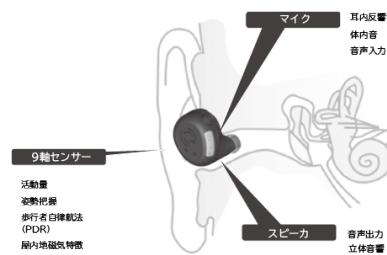
Smart community – NTT, https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201204fa1_s.html



ポケモンGO - <http://www.pokemongo.jp/>



ヒアラブルデバイス -
<https://jpn.nec.com/techrep/journal/g17/n01/170110.html>



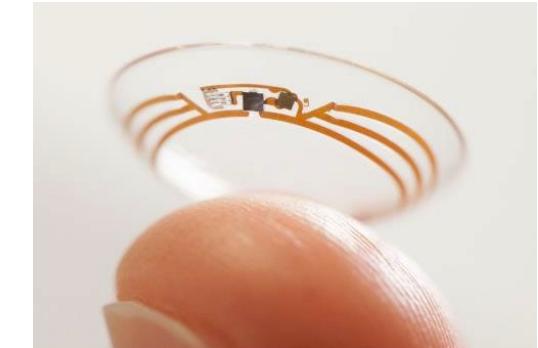
Food computer – OpenAg,
<http://openag.media.mit.edu/>



WestLab SD/Keio

スマートコンタクトレンズ

- <http://androidlover.net/androidnews/google-smart-contact-lens-monitoring-glucose-level.html>



Parrot Flower Power

- <http://www.parrot.com/jp/products/flower-power/>



- <https://tsst.scim-service.jp/>

<https://sii.or.jp/house2014/file/report02.pdf> 西研究室の取り組み



WestLab SD/Keio



スマート農業・スマート工場 (2020~)

The collage illustrates several projects from the Environmental Engineering Department at Keio University:

- 西研究室の取り組み**: 環境情報センシング (Environmental Sensing) - Includes a diagram of a building with sensor locations and two close-up photos of sensors.
- 西研究室の取り組み**: OMソーラー(太陽熱室内循環装置) (OM Solar (Solar Thermal Indoor Circulation Device)) - Shows a solar panel system connected to a control board.
- 西研究室の取り組み**: AiSEG - Shows a white control unit and a screenshot of a control interface featuring a penguin and a sun.
- 西研究室の取り組み**: 制御におけるプロジェクト成果 (Project Results in Control) - A list of bullet points detailing project outcomes.
- 主な取り組み**: 慶應共進化住宅スマートハウス (2014) (Keio Co-Development Residential Smart House (2014)) - Features a group photo of students, a modern house exterior, and a screenshot of a management application showing usage details and graphs.
- SD/Keio**: A greenhouse and a large industrial pump and piping system.



IoT的なサービスの例

- なんでもかんでも IoT 化

- 環境制御 – スマートシティ / タウン / ビルディング / ハウス
- エネルギー制御 – HEMS / BEMS • HVAC 制御 • 系統電力制御
- 交通 – GPS • 自動運転 • 交通制御 • 環境改善
- 犯罪防止, セキュリティ – 警報 • ライト • カメラ
- 災害 – モニタリング • 避難指示
- 医療健康 – 遠隔診察
- 農業 – モニタリング • リコメンデーション • 農薬散布
- 設計 • 住居構造 – 最適化, 生命化建築

極めて SD 的

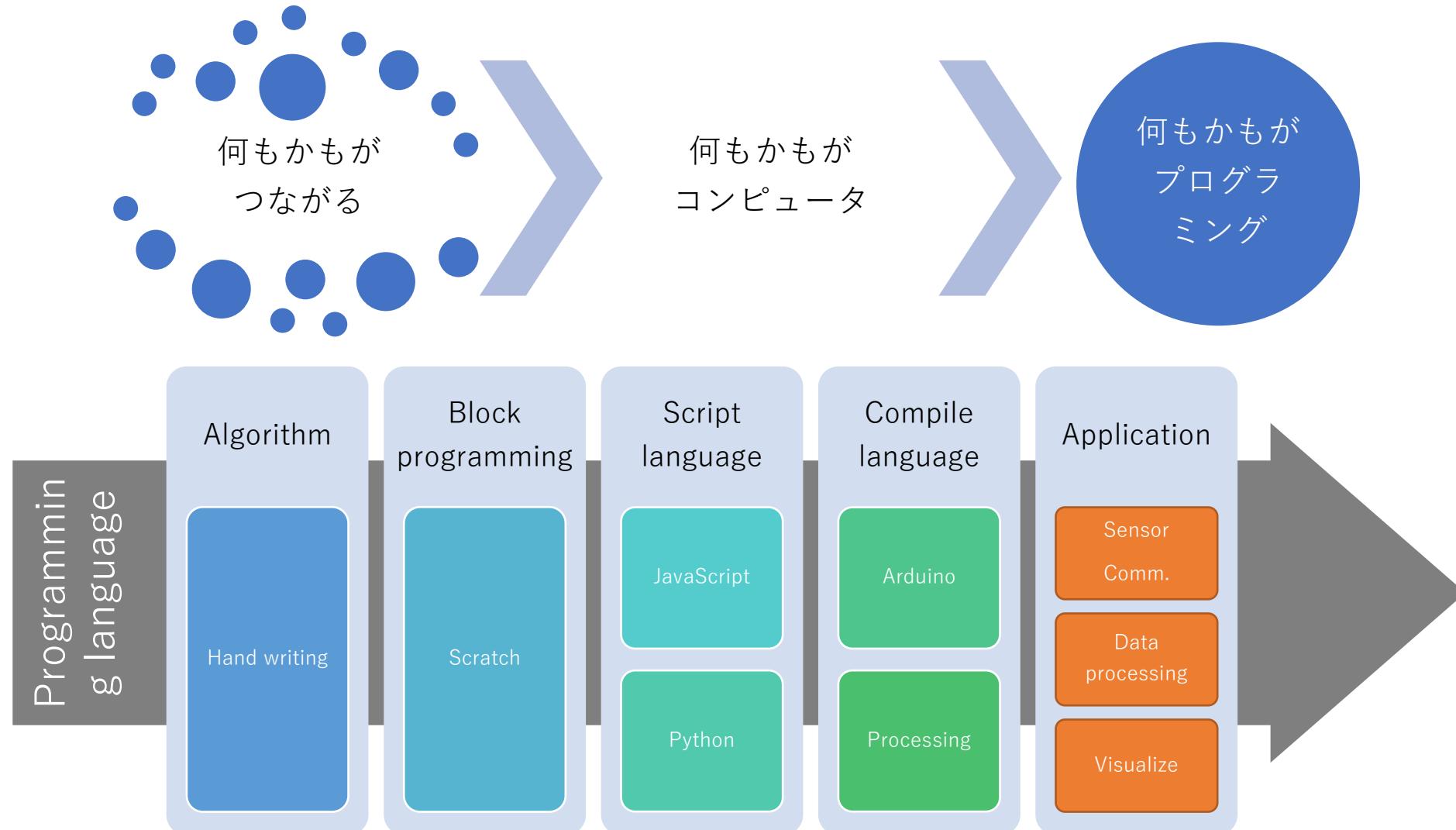


Images: The Internet of Everything, Aamer Azeemi, Cisco,
2013



全体像・IoTとプログラミングとの関係

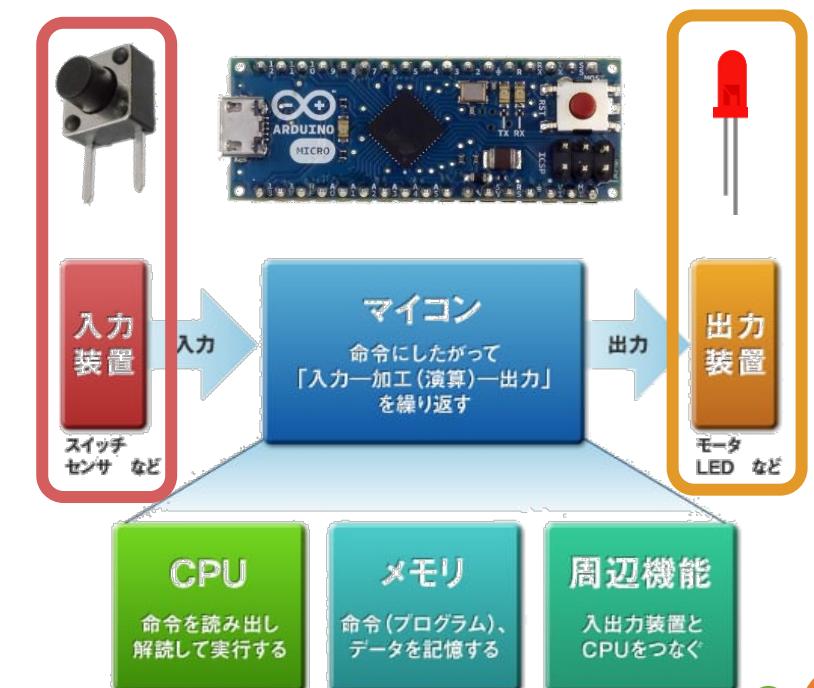
12



マイクロコントローラArduino

13

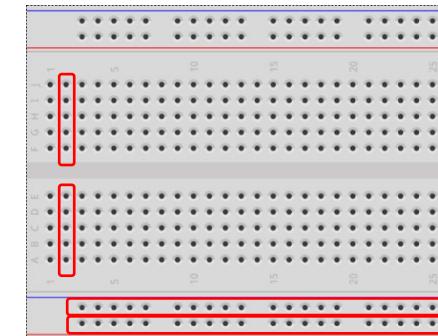
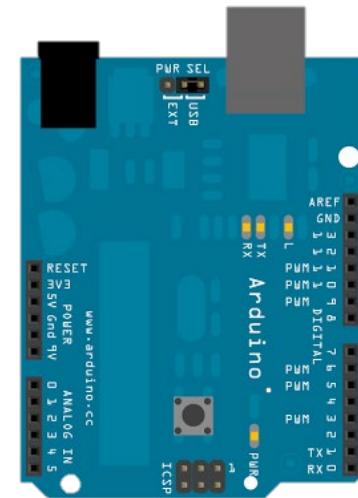
- マイコンの一種
 - マイコン=マイクロ・コンピュータ マイクロ・コントローラ
- マイコンチップは500円程度で購入可能
 - ATMegaシリーズは家電やロボットなどあらゆる電気機器で大量に利用されている
 - 周辺デバイスを合わせて安いもので1,500程度から
 - 20年前のPCよりも高性能
- 本来はマシン語（数値だけの言語）で設計
 - 高級言語による簡単設計ができるように工夫されている
- デバイス本来は駆動能力が弱く直接素子を扱えない
 - 使いやすいようにI/O周辺が強化されている
 - 動作周波数はトレードオフで遅いが、IoT用途では充分な性能
- 低消費電力化が可能
 - スリープモードなどはきちんと搭載
 - 様々な電圧に対応できるように工夫・そして壊れにくい





回路を作るために

- Arduino上のピン:各ピンに役割があるため適切なピンへ配線
- ブレッドボード:ワイヤの抜き差しをするだけで回路を作成
- LEDなどの素子には極性がある
- 電源の配線はプラスが赤, マイナスが黒
 - プラスマイナスを逆にすると素子などを壊しかねないので注意
- 研究資材なので丁寧に扱って下さい (基本西研持ち出し品です)
 - 壊したらそれなりの対応が必要になります (予算の性質上)
- 前準備 (すべてシミュレーターで確認できます)
 - Arduinoの基礎と設計方法を学ぶ
 - LEDをチカチカさせる
 - スイッチやボタンを押すと点灯・点滅する回路
 - 可変抵抗でぼんやり光る回路
 - サーボモータを動かす回路



内部はすべて結線されている



Tinkercad.com Arduino Simulator

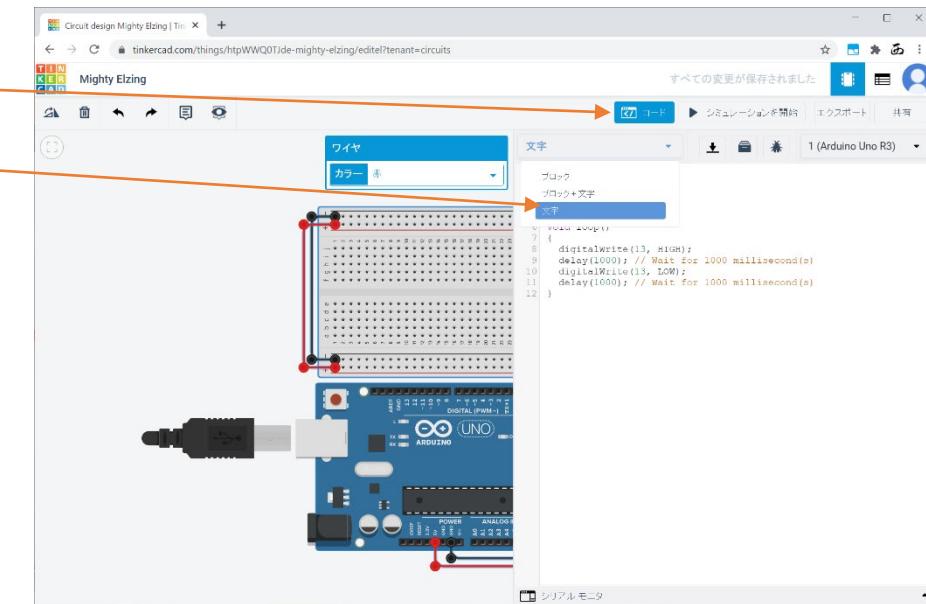
- Autocadの公式サイト (<https://www.tinkercad.com/>) にアクセス
- 右上の[今すぐ参加]をクリック
 - すでにアカウントを持っている場合は[サインイン]よりログイン
- [パーソナルアカウントで作成]をクリック
- Googleでサインインを選択
- Keio.jpアカウントでログイン
- 回路を選択
- 新しい回路を選択
- スターターからArduinoを選択
- 一番上のブレッドボードを選択して配置



Arduino Simulatorでコードを書く

16

- コードを選択
- 文字を選択するとプログラムが記述できる
- 実際には実機を複数個用いて実験する
 - 事前学習ではシミュレータが扱えれば充分
- プログラミング言語はC++由来ですが独自です
 - マルチメディアデザインで用いるProcessingも同様です
文法体系とインターフェースが同じです
 - 実はArduino IDEはProcessingの設計環境を基に作られた
 - ifやforなど基本はプログラミング演習を習得していれば問題なく理解できるはず
 - C言語系であるため、行の最後にセミコロン;が必要
 - 今回はアルゴリズムを学ぶのではなく、手順を学ぶので、複雑な構文は用いない
 - 情報系では、「プログラミング演習」と「データシステムの知能化とデザイン」でPythonを、「SD実験ディジタル回路」と「マルチメディアデザイン」でProcessing由来のJavaライクな言語を扱う

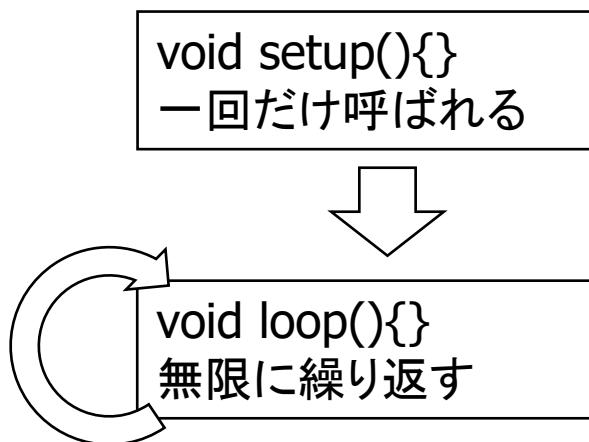




(1) LEDをチカチカ、いわゆるLチカの設計

17

- 右図のように配線、抵抗は 330Ω に設定、抵抗値によって明るさが変わる
カラーコードは（橙橙茶）
- Arduinoのコードも入力する
- シミュレーションを開始すると
毎秒点滅する
 - エラーの場合はプログラムを確認
 - 回路のエラーは教えてくれない



TINKER CAD LED-Blink

すべての変更が保存されました

コード シミュレーションを開始 エクスポート 共有

1 (Arduino Uno R3)

文字

```
int led = 2;
void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000) ;
```

int led = 2;
void setup() {
 pinMode(led, OUTPUT);
}
void loop() {
 digitalWrite(led, HIGH);
 delay(1000);
 digitalWrite(led, LOW);
 delay(1000) ;

The screenshot shows the TinkerCAD interface with a breadboard simulation and an Arduino Uno R3 board. A red box highlights the connection from digital pin 2 to the LED. The breadboard shows a 330 ohm resistor connected between the LED and ground. The Arduino board is connected to power and ground. The code editor on the right contains the standard LED-blink example code.





Lチカの流れ

18

int led = 2; 初期設定

ピンledをOUTPUT
(出力) にする

↓
pinMode(led, OUTPUT);

digitalWrite(led, HIGH);

ピンledで
HIGH(5V)出力

1000ms待つ

delay(1000);

↑

ピンledで
LOW(0V)出力

1000ms待つ

delay(1000);

↑

digitalWrite(led, HIGH);

The screenshot shows the Tinkercad interface for a "LED-Blink" project. On the left is a breadboard simulation with a red LED connected to digital pin 13 via a 220 ohm resistor. Pin 13 is connected to a breadboard power rail through a red wire. Pin 13 is also connected to the Arduino Uno's GND pin via a black wire. The Arduino Uno board is shown at the bottom, with its pins labeled. The code in the center-right panel is:

```
int led = 2;  
void setup() {  
    pinMode(led, OUTPUT);  
}  
void loop() {  
    digitalWrite(led, HIGH);  
    delay(1000);  
    digitalWrite(led, LOW);  
    delay(1000);  
}
```

The status bar at the bottom right indicates "シリアル モニタ".





(2) ボタンで明滅を制御（設計）

- 右図のように配線
 - 抵抗はすべて330Ω
- Arduinoのコードは後述
- シミュレーションを開始
- 仕様上はボタンを押す毎に、点灯と消灯を繰り返すはず
 - だが、実際にはうまく動作しない
 - 実機でもうまく動作しない
 - このうまく動作しないということをきちんとシミュレートする



TINKER CAD LED+SW

保存済み コード シミュレーションを開始 エクスポート 共有 文字 1 (Arduino Uno R3)

```
1 int led = 2;
2 int button = 3;
3 int state = 0;
4 void setup() {
5     pinMode(button, INPUT);
6     pinMode(led, OUTPUT);
7 }
8 void loop() {
9     if (digitalRead(button) == HIGH) {
10         if (state == 0) {
11             state = 1;
12             digitalWrite(led, HIGH);
13         } else {
14             state = 0;
15             digitalWrite(led, LOW);
16         }
17     }
18 }
```

シリアル モニタ



コードと動作

```
int led = 2;
int button = 3;
int state = 0;
void setup() {
    pinMode(button,INPUT);
    pinMode(led,OUTPUT);
}
void loop() {
    if (digitalRead(button) == HIGH) {
        if (state == 0) {
            state = 1;
            digitalWrite(led, HIGH);
        } else {
            state = 0;
            digitalWrite(led, LOW);
        }
    }
}
```

The screenshot shows a Tinkercad workspace titled "Circuit design LED+SW | Tinkercad". The circuit is built on a breadboard with an Arduino Uno at the bottom. A digital button is connected to pin 3 (labeled "button") through a pull-down resistor. A digital LED is connected to pin 2 (labeled "led"). The Arduino's ground pin is connected to the breadboard ground rail. The Arduino's 5V pin is connected to the breadboard power rail. The Arduino's TX pin is connected to a serial monitor module. The code editor on the right shows the following Arduino sketch:

```
int led = 2;
int button = 3;
int state = 0;
void setup() {
    pinMode(button,INPUT);
    pinMode(led,OUTPUT);
}
void loop() {
    if (digitalRead(button) == HIGH) {
        if (state == 0) {
            state = 1;
            digitalWrite(led, HIGH);
        } else {
            state = 0;
            digitalWrite(led, LOW);
        }
    }
}
```

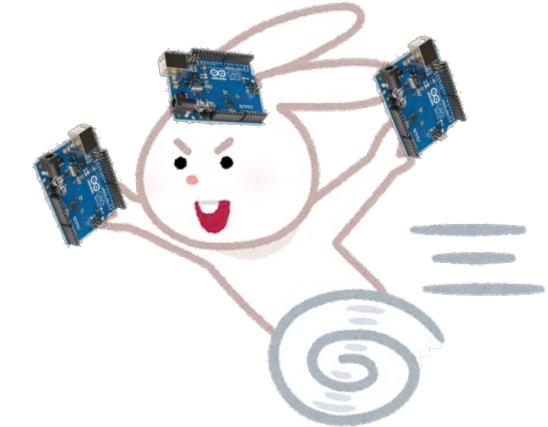
The "シリアル モニタ" (Serial Monitor) window is visible at the bottom.



なぜうまく動作しないのか？

21

- Arduinoマイコンの動作速度が速すぎるから
 - 搭載されているAVRマイコンATMegaの動作周波数は16MHz
 - 1600万回計算/秒、これでも非常に遅いほうで、PCは3GHzを簡単に超える
 - すると、チャタリング現象が発生する
 - ボタンを押す
 - StateとLEDの明滅が変わる
 - 人間は1600万回/秒で操作できないため、まだボタンが押されている
 - しばらく2と3を繰り返す
- ありきたりな解決法として、ボタンを押してから少し待つとよい（遅延させる方法）
- 待つ命令はdelayで実行すると指定時間何もしない
 - `delay(N);` → N ms待つという意味。 (1000ms=1秒)
 - Nのとるべき値について考え、さらに`delay(N);`を正しい場所に追加してチャタリングを防止する
 - 各自演習課題として試しておくこと（次のページ）
- より優れた方法がある
 - エッジ検出する方法もあるが、結局追加でメカニカルなチャタリングを防ぐため遅延は必要





演習問題

- チャタリングを防止する
 - 先のソースコードに例えばdelay(500);を適切な位置に入れてチャタリングを防ぐ
 - どこに入れるとよいだろうか?
 - ボタンが押されたら待つ、と考えれば?
- ボタンでモードを切り替える
 - ボタンを押すたびに、点滅を行うモードと、消灯モードを切り替える
 - こちらはチャタリングが発生するだろうか?実行する前にコードを見て考えよう
- 作って試してみること

```
int led = 13;
int button = 3;
int state = 0;
void setup() {
    pinMode(button,INPUT);
    pinMode(led,OUTPUT);
}
void loop() {
    if (digitalRead(button) == HIGH) {
        state = ~state;
    }
    if (state == 0) {
        digitalWrite(led, LOW);
        delay(500) ;
    } else {
        digitalWrite(led, HIGH);
        delay(250) ;
        digitalWrite(led, LOW);
        delay(250) ;
    }
}
```





(3) サーボを動かす

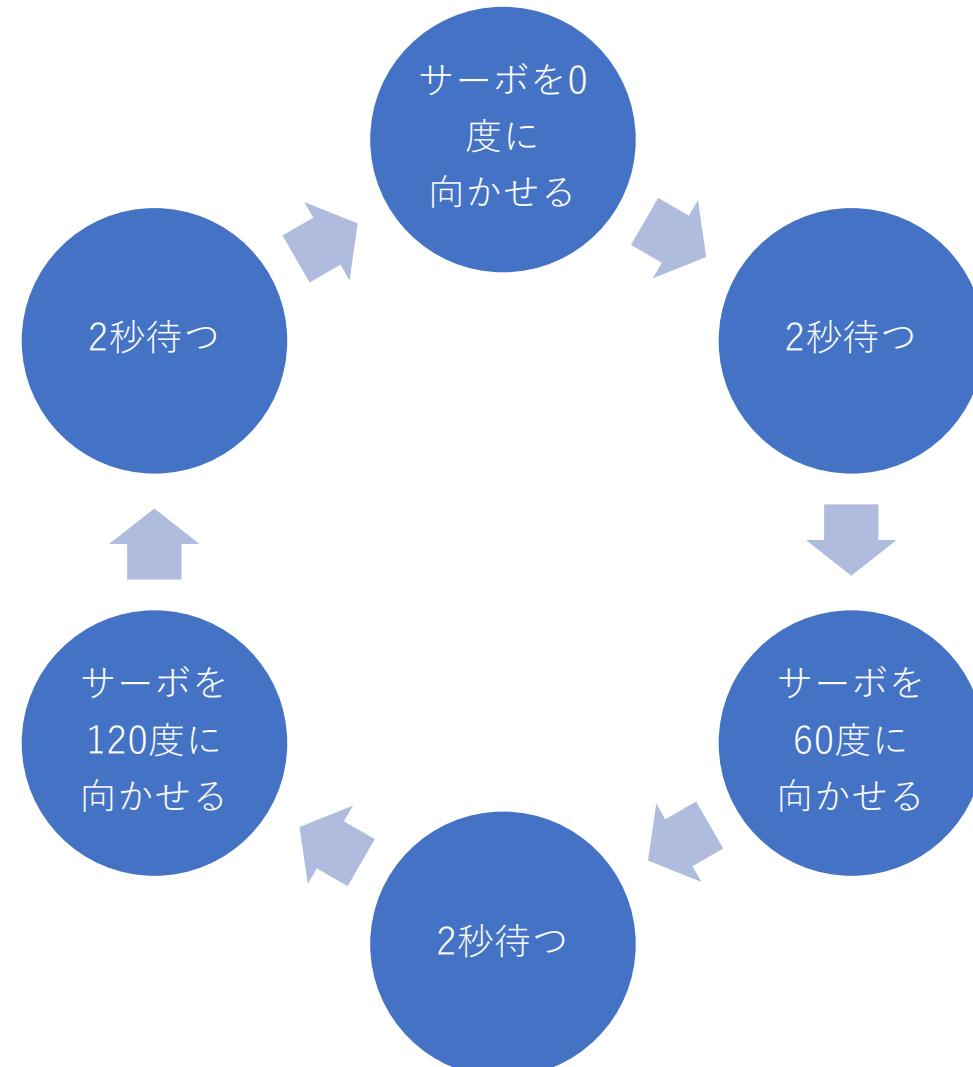
- サーボモータ
 - 本来は制御できるモータの意味、狭義ではラジコンなどで搭載されているPWM位置制御のモータ
 - 2種類ある
 - 回転サーボモータ：指定した速度で回り続ける
 - 位置サーボモータ：指定した位置（角度）まで移動して停止（魔改造で回転サーボになる）
- 実際のサーボと配線
 - 実際のサーボはギア機構がよく壊れる
手では回してはいけない
電源とGNDを間違えて接続しない
無茶な角度指令値を入れない
(ムギュと音を立てて回らない位置まで無理やり動こうとして壊れる)
- 3つの配線
 - プラス電源：赤色
 - マイナス電源：黒色や茶色など
 - 制御信号線：それ以外の黄色やオレンジなど





サーボ動作の流れ

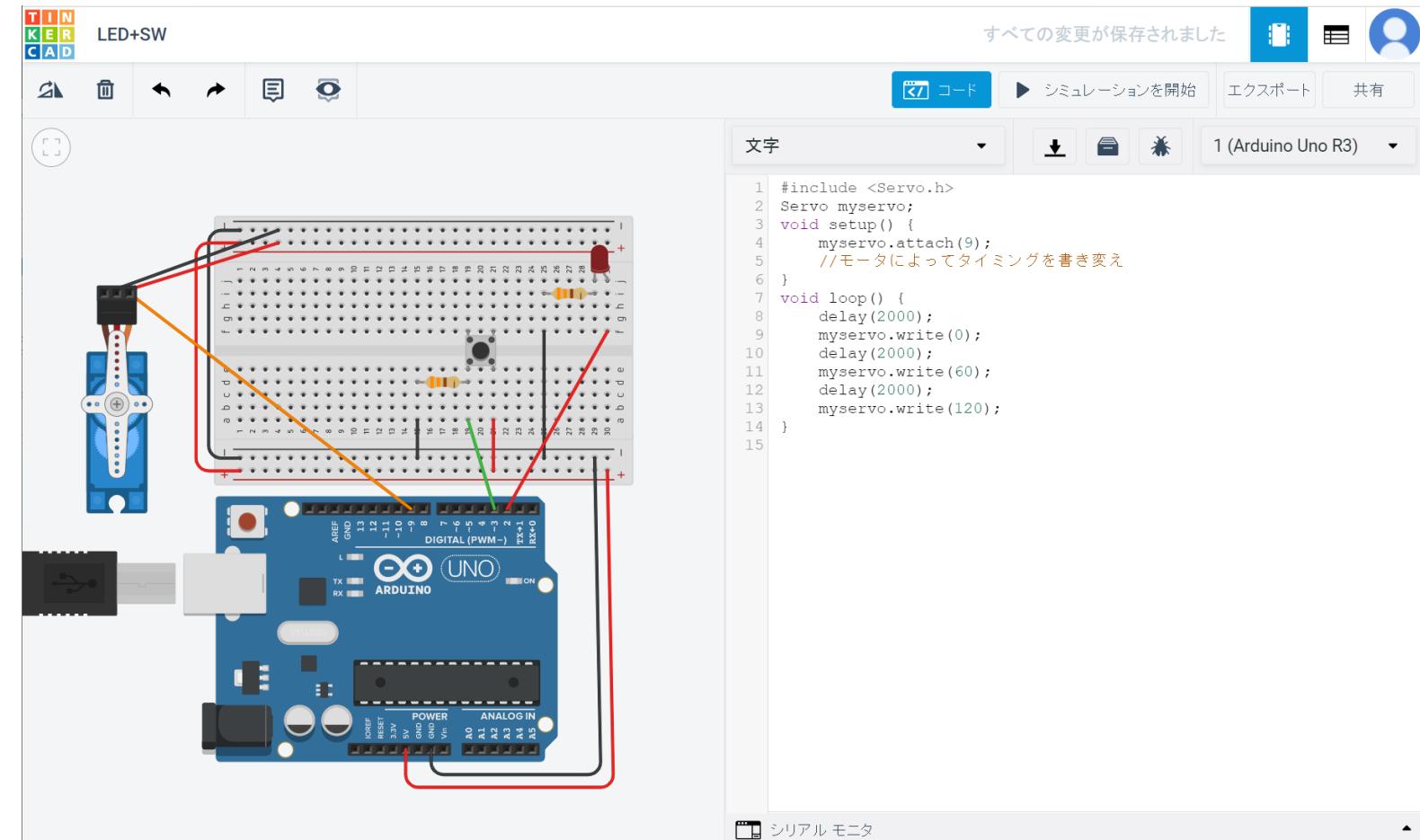
24



回路設計

25

- 右図のように配線
- 過去の回路は削除していない
- 残しておくと後で再利用できる
- Arduinoのコードは後述
- シミュレーションを開始
- 先ほどの説明通りの動作を行う





コードと動作

26

```
#include <Servo.h>
Servo myservo;
void setup() {
    myservo.attach(9);
    //モータによってタイミングを調整
    //例えば myservo.attach(9, 1500, 1900);
}
void loop() {
    delay(2000);
    myservo.write(0);
    delay(2000);
    myservo.write(60);
    delay(2000);
    myservo.write(120);
}
```

TINKER CAD LED+SW

すべての変更が保存されました

コード シミュレーションを開始 エクスポート 共有

文字

```
1 #include <Servo.h>
2 Servo myservo;
3 void setup() {
4     myservo.attach(9);
5     //モータによってタイミングを書き換え
6 }
7 void loop() {
8     delay(2000);
9     myservo.write(0);
10    delay(2000);
11    myservo.write(60);
12    delay(2000);
13    myservo.write(120);
14 }
```

シリアル モニタ





(4) 音を鳴らす

27

- 音を鳴らす関数
 - `tone(pin, frequency);`
- 音を止める関数
 - `noTone();`
- 周波数と音階の関係は自分で調べること
 - 例えばラは440
 - ちゃんと音もなります
- 回路は消さずに追加している
- 例えば
 - 2つ並べて和音を作る
 - 曲を奏でさせる

The screenshot shows the Tinkercad interface with the project titled "LED+SW". On the left is the breadboard setup, and on the right is the code editor.

Code (Arduino Uno R3):

```
int speaker = 11;
void setup() {
  pinMode(speaker, OUTPUT);
}
void loop() {
  tone(speaker, 440);
  delay(1000);
  noTone(speaker);
  delay(2000);
}
```



動作

28

The screenshot shows a Tinkercad workspace titled "Circuit design LED+SW | Tinkercad". The circuit is labeled "LED+SW". On the left, there's a breadboard with various components: a speaker, a push button, resistors, and jumper wires. On the right, an Arduino Uno R3 board is connected to the breadboard. The code editor on the right contains the following Arduino sketch:

```
1 int speaker = 11;
2 void setup() {
3     pinMode(speaker, OUTPUT);
4 }
5 void loop() {
6     tone(speaker, 440);
7     delay(1000);
8     noTone(speaker);
9     delay(2000);
10}
```

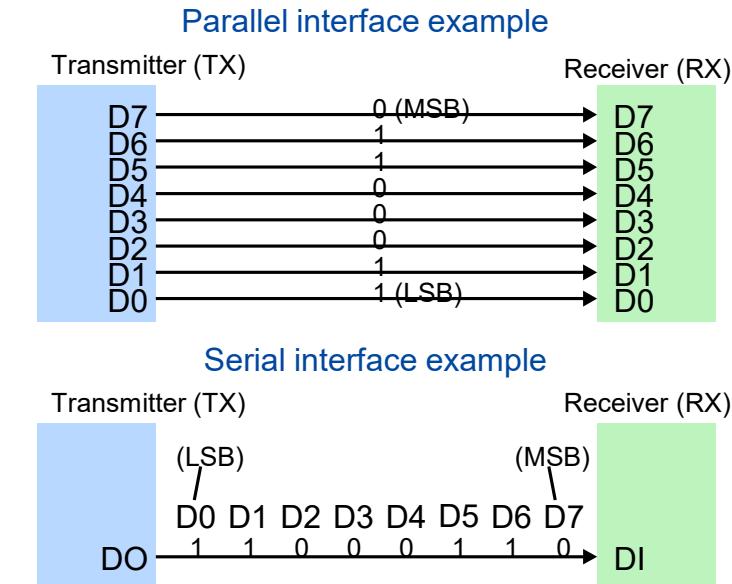
The "シリアル モニタ" (Serial Monitor) window at the bottom is currently closed.

WestLab SD/Keio



(5) シリアル通信（実機のみで動作）

- 配線一本で情報を送る通信方式
 - 電圧がHIGHつまり5Vならば1、LOWつまり0Vならば0
- 実際にはいろいろケアするべき点がある
 - 電圧は相対的なため、別途グラウンドを伝える必要がある
 - 信号を相手に伝える出力と、相手からもらう入力が一般的には必要
 - 1と0をバタバタしただけでは、正しく値が取得できないので、ボーレート（転送速度）をあらかじめ決めておく
 - 例えば、0000111100001111と0101の区別はつくだろうか？
 - 0から始まるデータの始まりがわからないので、スタートビットを設ける
 - ノイズに弱く、エラーが発生しやすいため、パリティーをつける
 - 処理できないうちに次々にデータが送られてこないように、バックプレッシャーを設ける
 - などなど
- USBも元々はシリアル(ユニバーサルシリアルバス)





(6) Hello,Worldの設計

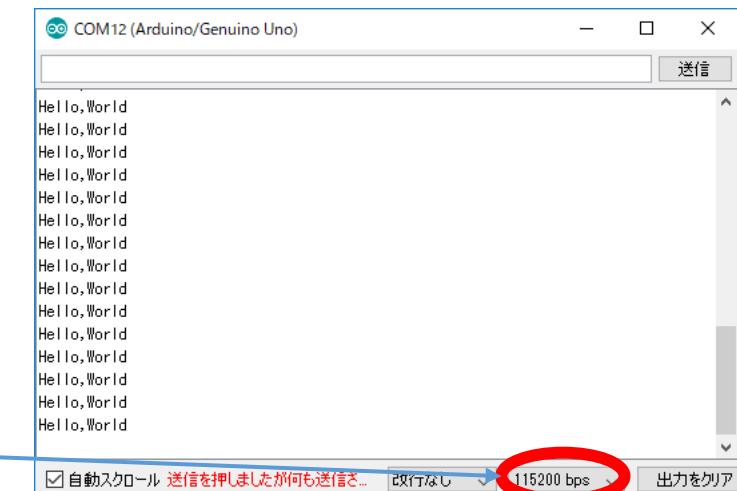
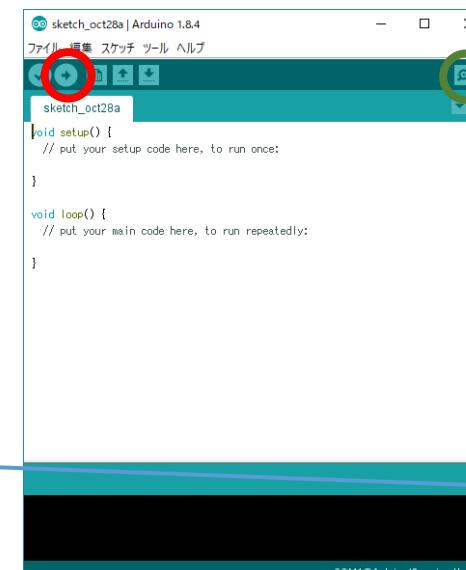
- ArduinoとPC間で通信する
- 手順
 - ArduinoとPCを繋げる
 - IDEを立ち上げる
 - ツールをクリック
 - ボードで「Arduino nano」を選択
 - ポートを選択
 - ()にArduinoの記載がある項目を選択
 - プログラムを記入
 - を押す
 - プログラムをArduinoに書き込む
 - ツールでポートを設定
 - Arduinoが接続されているポートを指定
 - をクリック
 - 通信速度を115200bpsに変更

```
void setup() {  
    Serial.begin(115200);  
}  
  
void loop() {  
    Serial.print("Hello");  
    Serial.println(", world");  
    delay(1000);  
}
```

シリアル通信を115200bpsで開始
その他の設定はデフォルトのまま

シリアルで"Hello"と出力

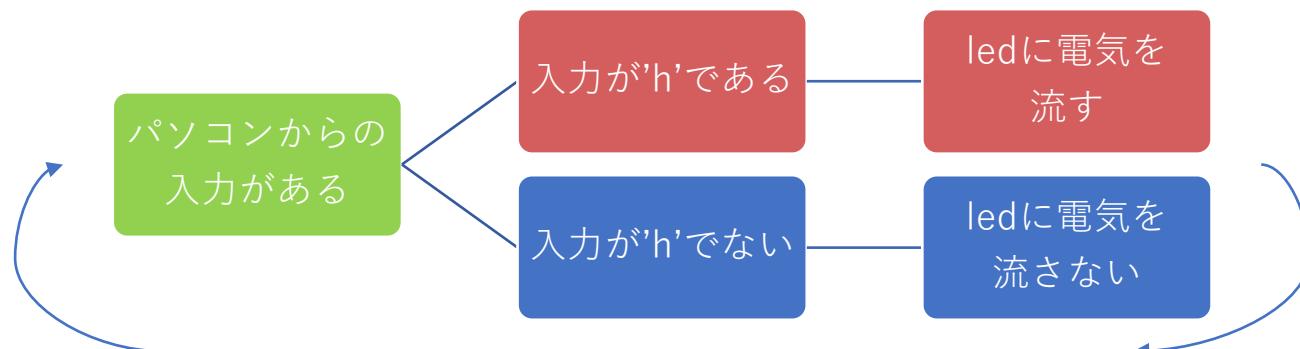
シリアルで", world"と出力 + 改行



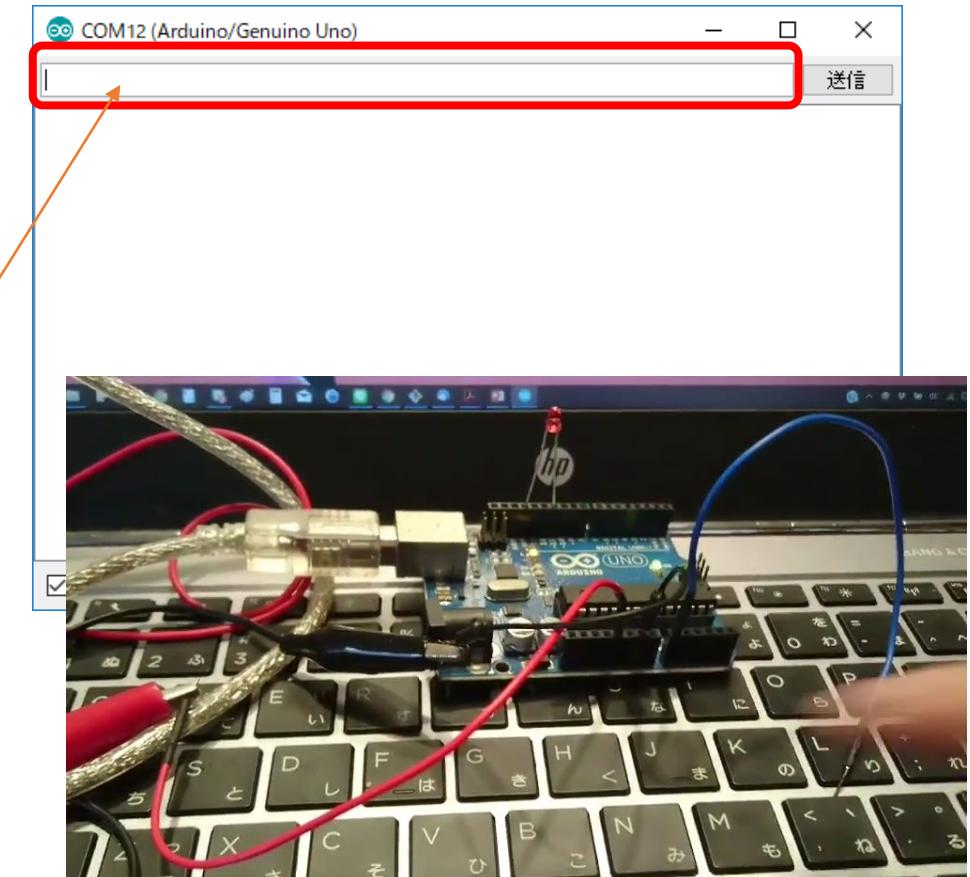
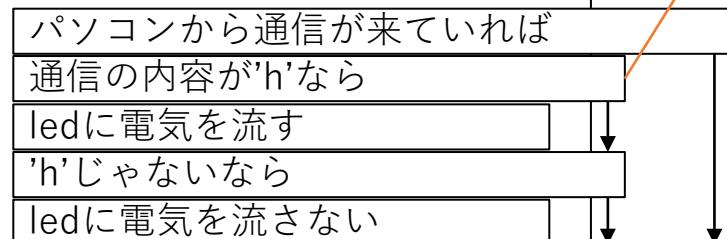


PCの指令でLEDを点灯・消灯 (ON/OFF制御)

31



```
int led = 2;  
void setup() {  
    pinMode(led, OUTPUT);  
    Serial.begin(115200);  
}  
void loop() {  
    if (Serial.available() > 0) {  
        if(Serial.read() == 'h'){  
            digitalWrite(led, HIGH);  
        }else{  
            digitalWrite(led, LOW);  
        }  
    }  
}
```



設計と動作内容

32

TIN
KER
CAD

LED+SW+SM+PB+SER

すべての変更が保存されました

コード シミュレーションを開始 エクスポート 共有

文字

```
1 int led = 2;
2 void setup() {
3   pinMode(led, OUTPUT);
4   Serial.begin(115200);
5 }
6 void loop() {
7   if (Serial.available() > 0) {
8     if(Serial.read() == 'h'){
9       digitalWrite(led, HIGH);
10    }else{
11      digitalWrite(led, LOW);
12    }
13  }
14 }
15
```

シリアル モニタ

TIN
KER
CAD

LED+SW+SM+PB+SER

すべての変更が保存されました

コード シミュレーションを開始 エクスポート 共有

同期していません

文字

```
1 int led = 2;
2 void setup() {
3   pinMode(led, OUTPUT);
4   Serial.begin(115200);
5 }
6 void loop() {
7   if (Serial.available() > 0) {
8     if(Serial.read() == 'h'){
9       digitalWrite(led, HIGH);
10    }else{
11      digitalWrite(led, LOW);
12    }
13  }
14 }
15
```

シリアル モニタ





Arduino言語

<http://www.musashinodenpa.com/arduino/ref/>

33

- Arduino言語 = avr-gcc + Wiring用ライブラリ + Arduino用ライブラリ
 - C++に準拠(SDではすでにC++は教えておらず、難解言語の一つ)

```
if(条件文) {
  // 条件に該当時実行
}
```

```
if (a < 500) {
  // 動作A
} else if (a >= 1000) {
  // 動作B
} else {
  // 動作C
}
```

```
x == y (xとyは等しい)
x != y (xとyは異なる)
x < y (xはyより小さい)
x > y (xはyより大きい)
x <= y (xはy以下)
x >= y (xはy以上)
```

```
&& : and, || : or, ! : not
```

```
switch (var) {
  case 1:
    // varが1のとき実行
    break;
  case 2:
    // varが2のとき実行
    break;
  default: // (省略可能)
    // 不一致のとき実行
}
```

break と **continue**
break : ブロック離脱
continue : 再実行

return と **goto**
return : 関数から戻る
 戻り値の追加も可
goto : ラベル(文字列:) ヘジャンプ

```
for (初期化; 条件式; 加算) {
  // 実行される文;
}
```

```
while(条件式){
  // 実行される文
}
```

```
do {
  // 実行される文
} while (条件式);
```

boolean : 二値, **char** : 1文字
整数 (**unsigned**で符号なし)
Byte : 8bit, **int** : 16bit
word : 32bit, **long** : 64bit
小数
float : 32bit, **double** : 64bit
形無し : **void**
高機能文字列クラス : **String**

pinMode(pin, mode)
digitalWrite(pin, value)
digitalRead(pin)

analogRead(pin)
analogWrite(pin, value)
analogReference(type)
analogReadResolution(bits)
analogWriteResolution(bits)

pulseIn(pin, value, timeout)
tone(pin, frequency)
noTone(pin)
delay(ms)

Serial.begin(speed)
Serial.available()
Serial.read()
Serial.flush()
Serial.print(data, format)
Serial.println(data, format)
Serial.write(val)





言語について簡単な知識を得る

- Arduino IDEをインストールする
 - <https://www.arduino.cc/en/software>
 - Arduino IDE 2.0のリリース待ち
- ファイル→スケッチ例に様々な例がある
- シミュレータでの動作には制限があるので注意する
- 文法そのものは難しくない
 - Pythonを知っていればセミコロンや波括弧でのブロック表記さえ慣れれば問題ないはず
 - 多くの人がArduinoを利用しているため、Google先生も大変役に立つ

The screenshot shows the Arduino IDE interface with the title bar "mote | Arduino 1.8.13 (Windows Store 1.8.42.0)". The menu bar includes "ファイル" (File), "編集" (Edit), "スケッチ" (Sketch), "ツール" (Tools), and "ヘルプ" (Help). The "Sketch" menu is expanded, showing sub-options like "新規ファイル" (New File), "開く..." (Open...), "最近使った項目を開く" (Open Recent), "スケッチブック" (Sketchbook), and "スケッチ例" (Sketch Examples). The "スケッチ例" submenu is further expanded, showing categories such as "内蔵のスケッチ例" (Built-in Sketch Examples) and "あらゆるボード用のスケッチ例" (Sketch Examples for All Boards). Under "Built-in Sketch Examples", "Blink" is highlighted. The code for the "Blink" sketch is displayed in the main editor area:

```
#define BUTTON_RED_PIN 10
#define BUTTON_GREEN_PIN 11
#define BUTTON_BLUE_PIN 12

int oldR = 0, oldG = 0, oldB
#define CNT_MAX 2
int cntR = 0, cntG = 0, cntB
int colorR = 0, colorG = 0, colorB = 0

#include "source.h"
MyXBee myxbee;
unsigned long pastMillis = millis();
#define SEND_INTERVAL 1000
#define LED_HEADER 'L'
#define ID_PACKET_OFFSET '0'

int clickDetection(int _pin,
    int buttonClicked = 0;
    int newButtonState = digitalRead(_pin);
    if (*_oldstate == HIGH && *_oldstate != newButtonState)
        *_oldstate = newButtonState;
    return buttonClicked;
}

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

pinMode(LED_RED_PIN, OUTPUT);
pinMode(LED_GREEN_PIN, OUTPUT);
pinMode(LED_BLUE_PIN, OUTPUT);

pinMode(BUTTON_RED_PIN, INPUT);
pinMode(BUTTON_GREEN_PIN, INPUT);
pinMode(BUTTON_BLUE_PIN, INPUT);

void loop() {
    if (clickDetection(BUTTON_RED_PIN))
        cntR = cntR + 1;
        if (cntR >= CNT_MAX) cntR = 0;
    if (clickDetection(BUTTON_GREEN_PIN))
        cntG = cntG + 1;
        if (cntG >= CNT_MAX) cntG = 0;
    if (clickDetection(BUTTON_BLUE_PIN))
        cntB = cntB + 1;
        if (cntB >= CNT_MAX) cntB = 0;
    colorR = map(cntR, 0, CNT_MAX, 0, 255);
    colorG = map(cntG, 0, CNT_MAX, 0, 255);
    colorB = map(cntB, 0, CNT_MAX, 0, 255);
    myxbee.sendData(LED_HEADER, colorR, colorG, colorB);
}
```

The status bar at the bottom right shows "COM1のArduino Uno". The page number "12" is located at the bottom center.



IoTを作る

35

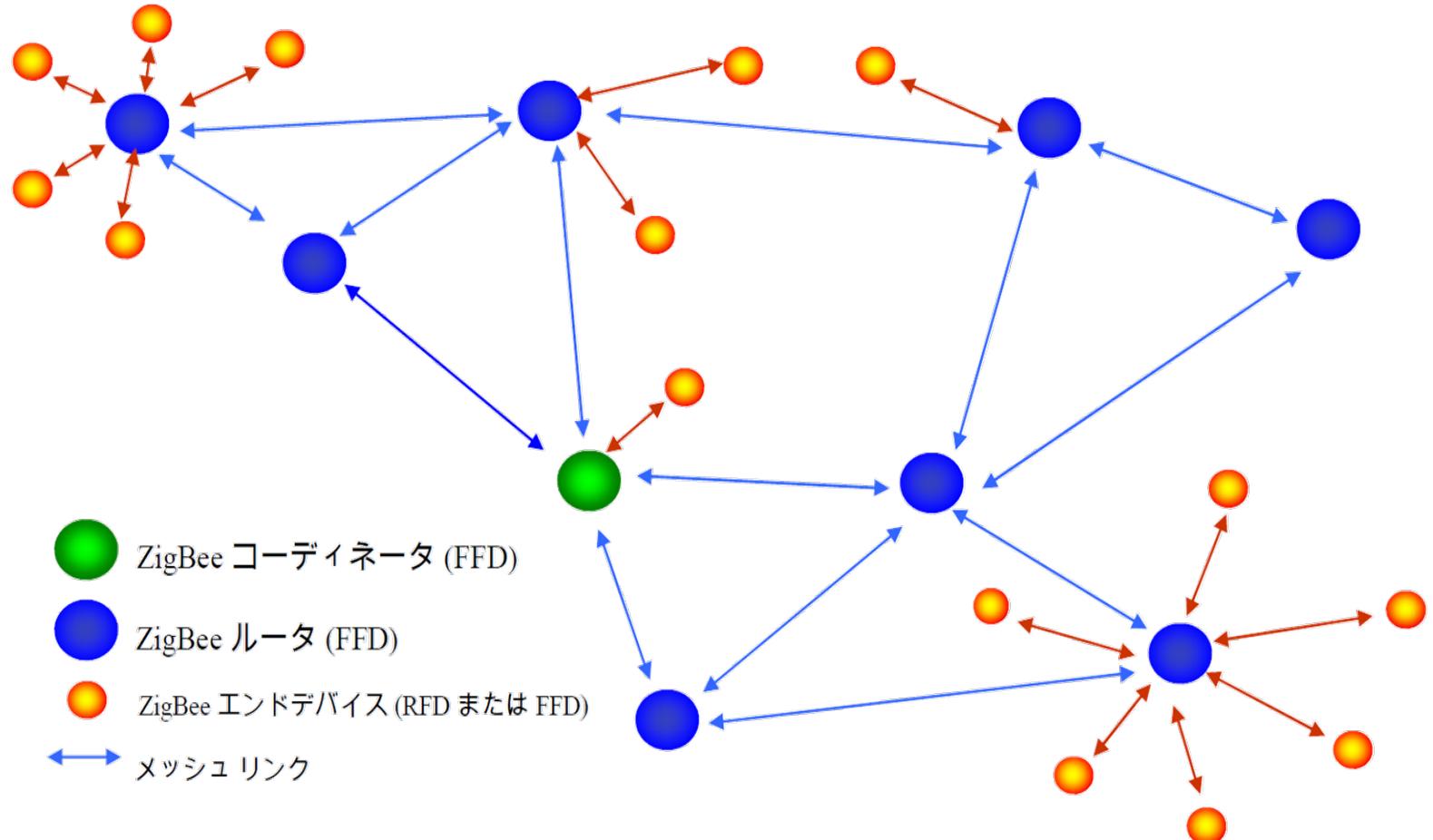
- ZigBeeと呼ばれる無線センサネットワークデバイスを利用してセンサネットワークを構築し、IoTシステムを実装する
- 実験では、ここから先の内容を実際に作成、実験内で仕様拡張を行う
- ここから先はシミュレータでは動作しないので注意すること





ZigBeeとは

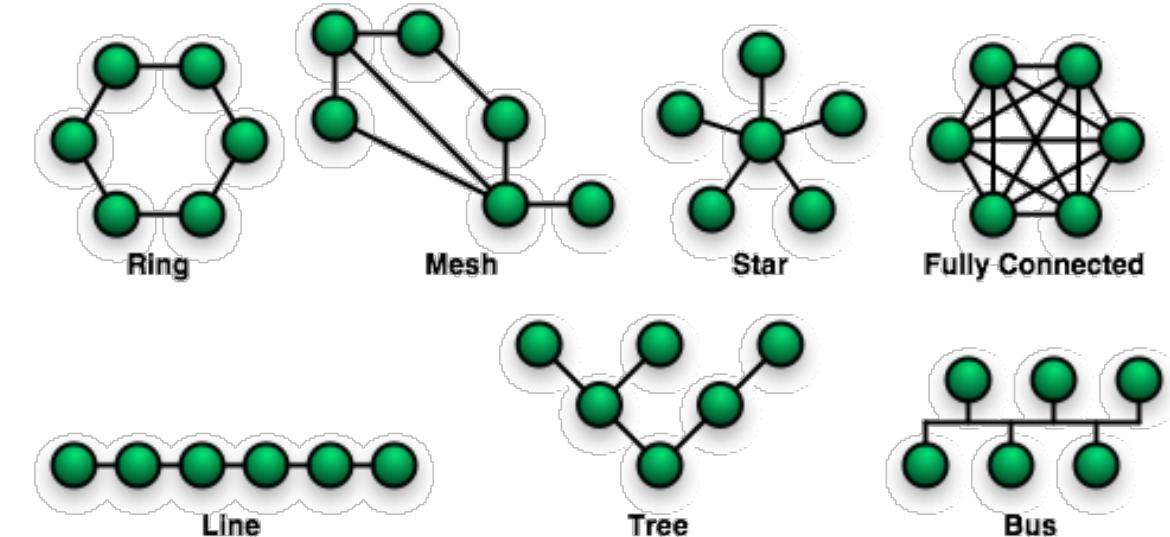
- ネットワーク通信モデル

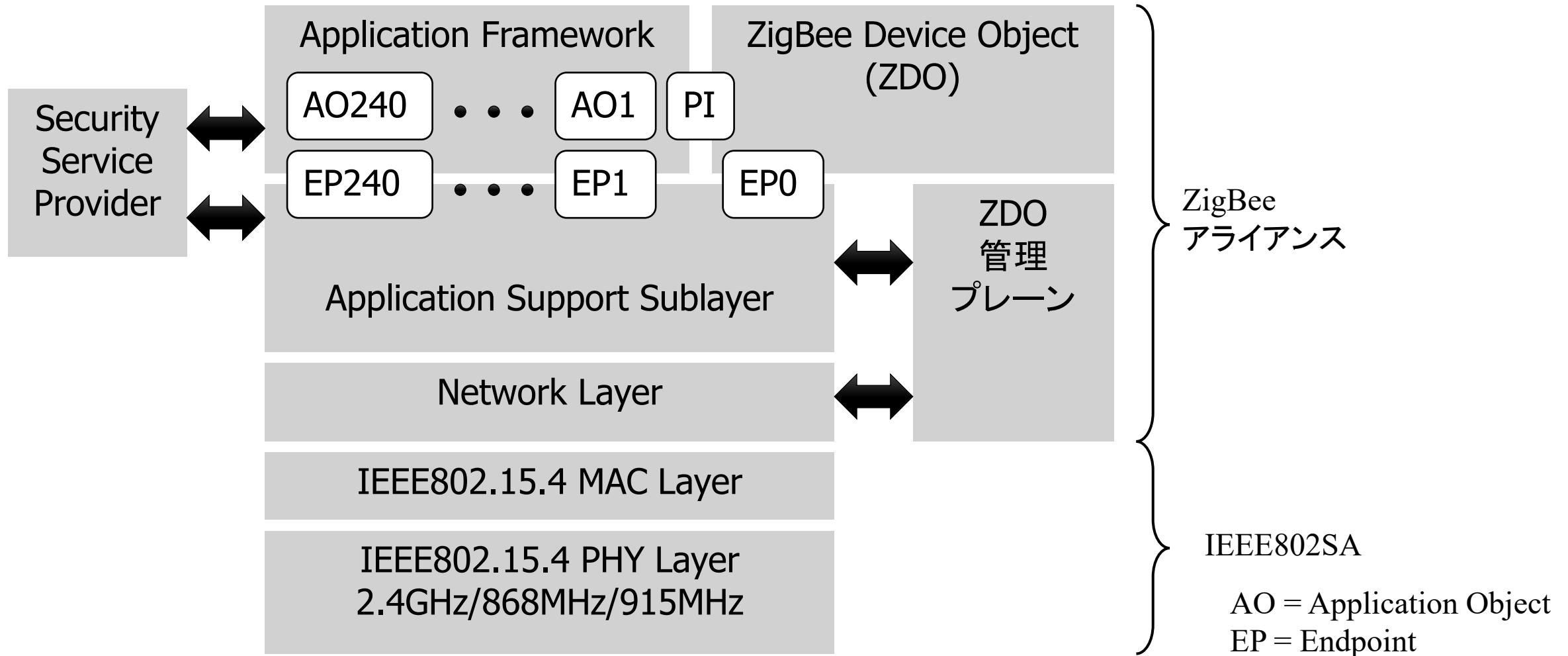


ネットワークトポロジ

37

- スター型ネットワーク
 - コーディネータ1つとエンドデバイス
 - 全てのZigBeeデバイスの省電力動作が可能
 - 通信エリアは狭い（最大半径70m程度）
 - ビーコンによる同期型PAN（パーソナルエリアネットワーク）を構築可能
- メッシュ型ネットワーク
 - ルータデバイスを配置することで通信エリア拡大可能
 - テーブルルーティングで複数の経路を使用可能であるため、障害に強い
 - 省電力化はエンドデバイスのみ可能（コーディネータとルータは基本的に常時オン）
- クラスタツリー型ネットワーク
 - ルータデバイスを配置することで通信エリア拡大可能
 - 全てのZigBeeデバイスの省電力動作が可能
 - PAN内の障害で通信不可が発生
 - ビーコンによる同期型PANが構築可能（ビーコン衝突による通信遅延あり）

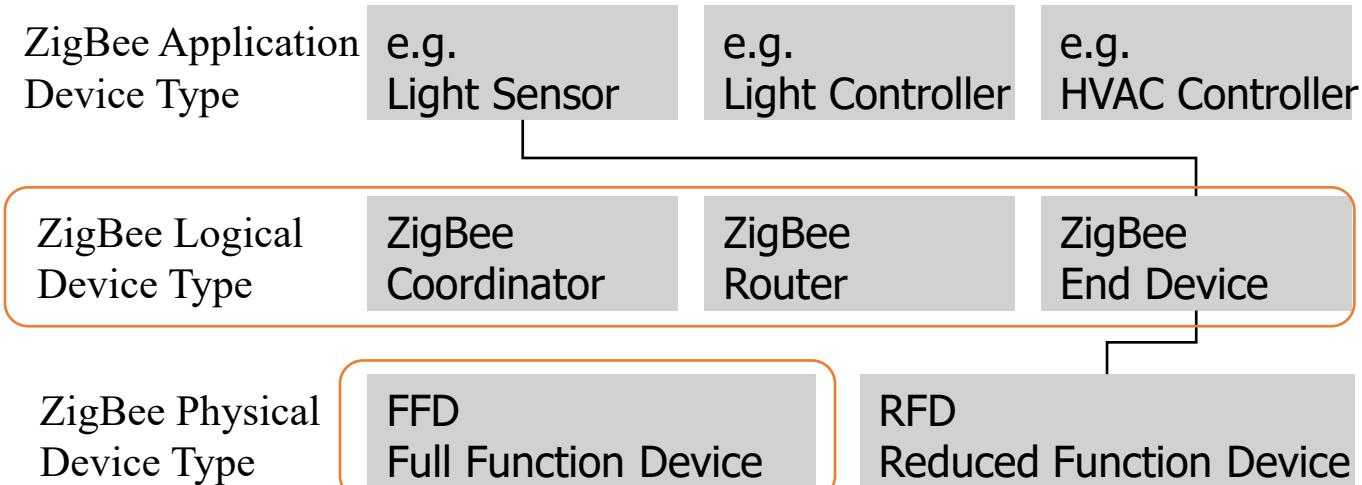






ZigBeeデバイスタイプ

- 各層は次のタイプに分類される
 - エンドユーザから見たデバイスタイプ
 - ネットワークで配置されるデバイスタイプ
 - ハードウェアプラットフォームのデバイスタイプ
 - 一般的には論理デバイスタイプで見る
- 通信形態には次の分類がある
 - ダイレクト通信
 - デバイス間直接通信でRFDはFFDとのみ通信可能 (RFD間不可)
 - 基本受け側は常にONであることが必要 (回避できなくはない)
 - ブロードキャストなどができる
 - インダイレクト通信
 - RFD間唯一の通信手段
 - 受信側デバイスの状態にかかわらず送信可能
 - 必要な時期にポーリングしてデータを受信できるため低消費電力化が可能
 - 相手先を指定する





ZigBee MAC層

- MAC層で規定されている機能

- スタート機能（ビーコン有り、または無し）
- 管理機能（Scan, PAN識別子衝突検出, Realignment）
- PAN参加（Association）、離脱（Dis-Association）
- データ通信機能（直接送信, 間接送信）
- 送達確認フレーム（ACKフレーム）のON/OFF
- 同期維持（Beacon Tracking）
- 同期外れ検出と再同期（Orphan Sync-loss）
- CSMA-CAによる衝突回避機能
- セキュリティ通信
- 信号電波のエネルギー検出
- デバイス間のリンク品質通達機能





ZigBee アプリケーション層

- アプリケーション層

- ZigBeeでは相互接続性確保のためにアプリケーションごとにプロファイルを定義
- プロファイル・オブジェクトはエンドポイント番号で識別、1つZigBeeデバイスは複数のプロファイルを持つ
- 1台のZigBeeデバイスに最大240のエンドポイント番号が付与可能
- 無線LANはアプリケーション層を規定していないが、規定することで「必ず繋がる」保証になる

タイプ	識別子	プロファイル名
Standard	0x0000	ZigBee device profile
0x0001… 0x00ff	(reserved)	
0x0100	Home Control, Lighting	
0x0101	Industrial Plant Monitoring	
0x0102	Health, Ventilation, Air Conditioning	
0x0103	Test Profile	
0x0104… 0x7fff	未定義	
Published	0x8000…… 0xbfff	未定義
Private	0xc000…… 0x7fff	ベンダ使用

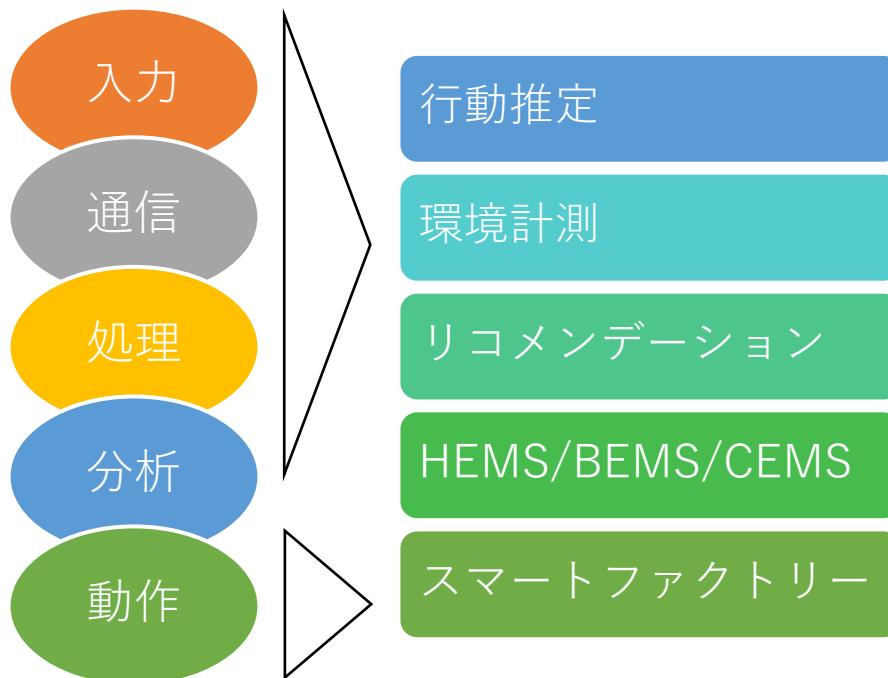


環境計測・制御システムの概要

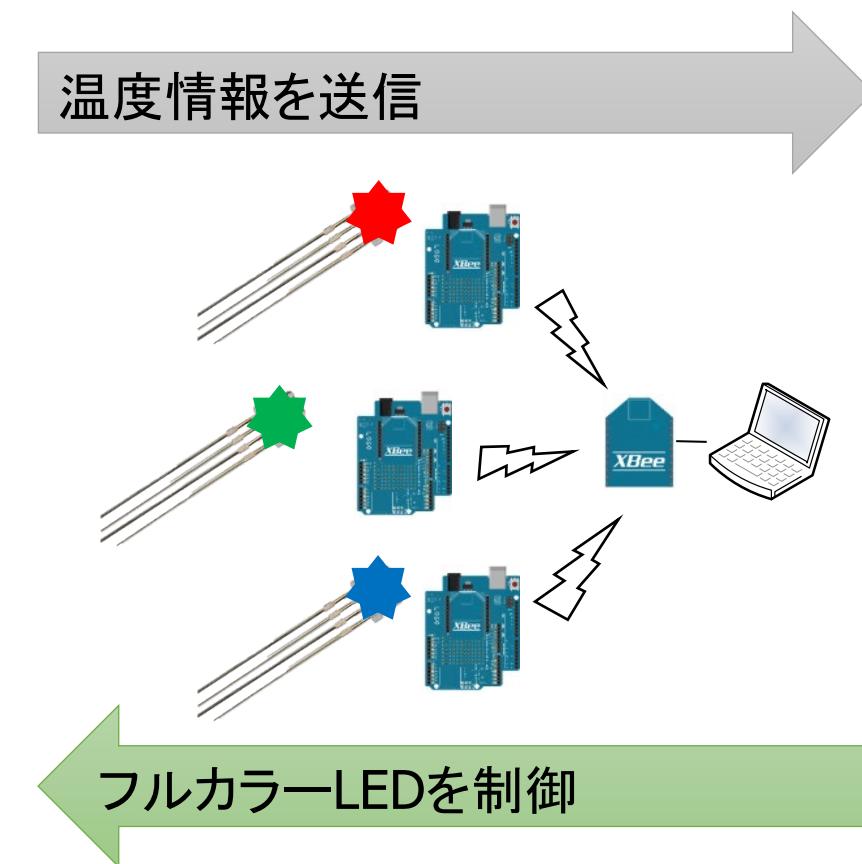
42

- データを無線で集め（センサを実際に設計）、視覚化し（著名フリーアプリで設計）、ノードを制御

- 応用例



WestLab SD/Keio





IDEの使い方と事前準備

・事前準備としてデスクトップの「SDrefresh.bat」を実行する

- D:\SD\日付 というディレクトリが生成されている。これが、今日使う作業場所



- Arduino IDEを起動する

- アイコンと機能



- コンパイルコードを確認（確認するだけ）



- スケッチをコンパイルしアップロードする



- 新しいエディタウィンドウを開く



- ファイルを開きます。



- スケッチを保存します。



- シリアルモニターを開く（主にデバッグに利用）



- タブを管理するオプションを表示（ほぼ使わない）

- Arduino UNOにプログラムを書き込み、通信させる

- USBが接続されており、ハブのスイッチがONであることを確認する

- ツールの、ボードとシリアルポート、両方にArduino UNOと表示させる

- ZigBeeシールドのスイッチが下(PCと接続)にする

```
BareMinimum
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

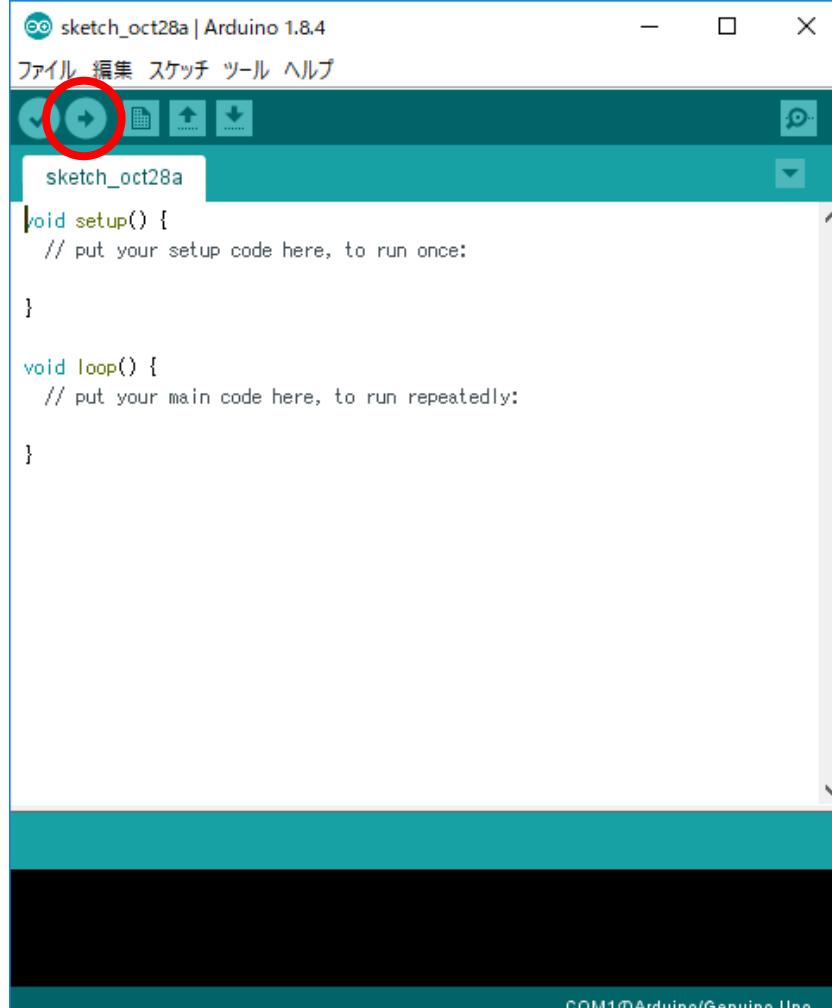
1 Intel® Galileo on /dev/cu.usbmodemfd141





PCとの接続と焼きこみ

- ArduinoとパソコンをUSBで繋げる
- Arduino IDEを起動する
- ツールをクリック
 - ボードで「Arduino UNO」を選択
 - ポートを選択 (Arduino UNO)
- 白枠内にプログラムを書く
-  を押す
 - プログラムがArduinoに書き込まれる
 - エラーがでたらコードを見直す
 - 当然ながら回路のエラーは教えてくれない



```
sketch_oct28a | Arduino 1.8.4
ファイル 編集 スケッチ ツール ヘルプ
sketch_oct28a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

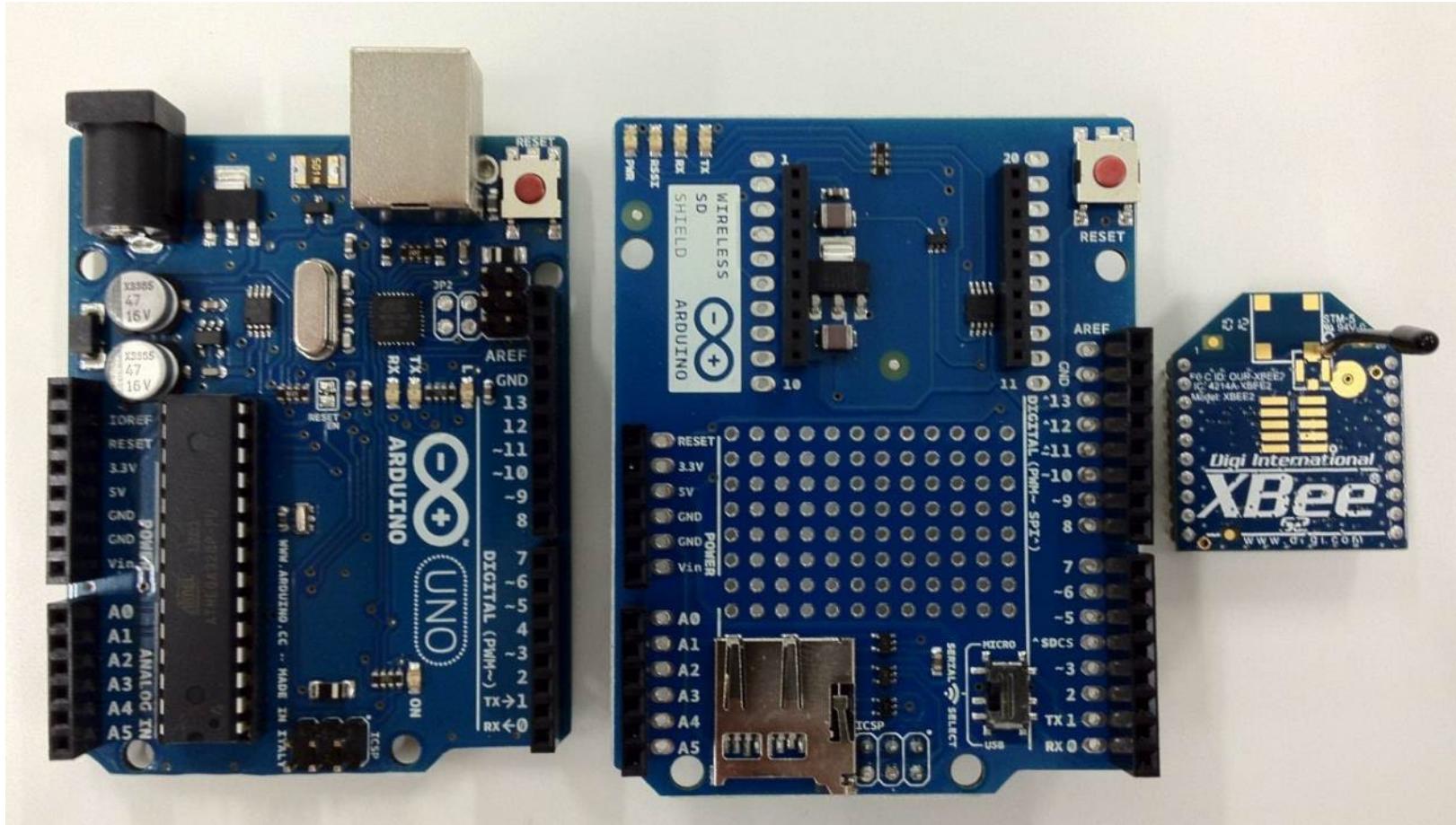
COM1のArduino/Genuino Uno



キットのパート 1/3

45

- Arduino UNO, Arduino shield, XBee

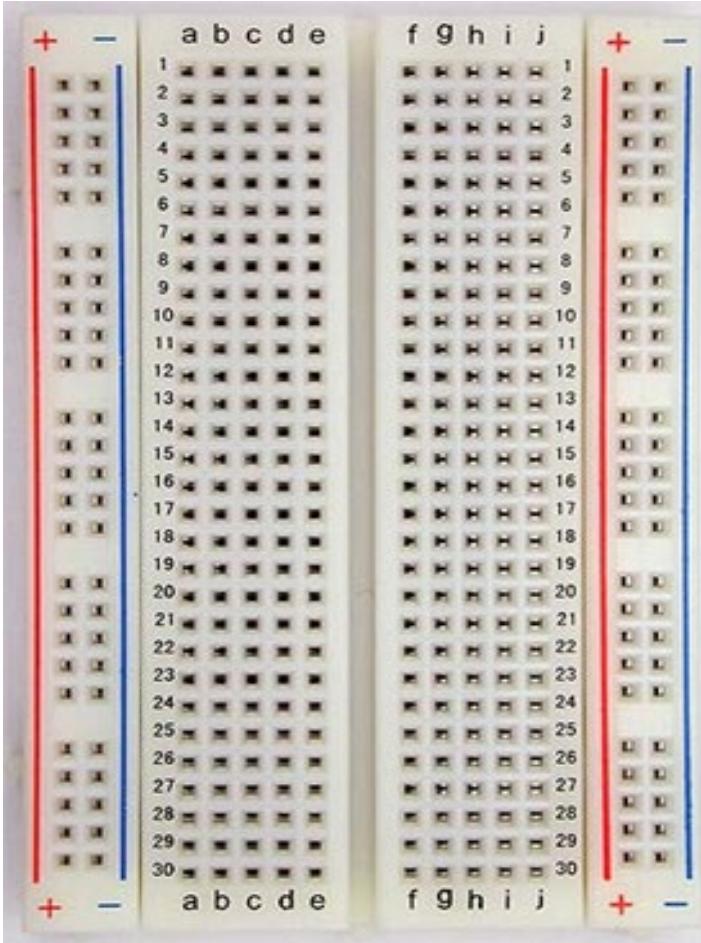




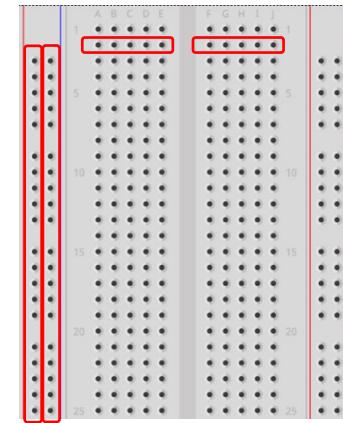
キットのパート 2/3

46

- ・ブレッドボード



内部はすべて結線されている



- ・ジャンパケーブル



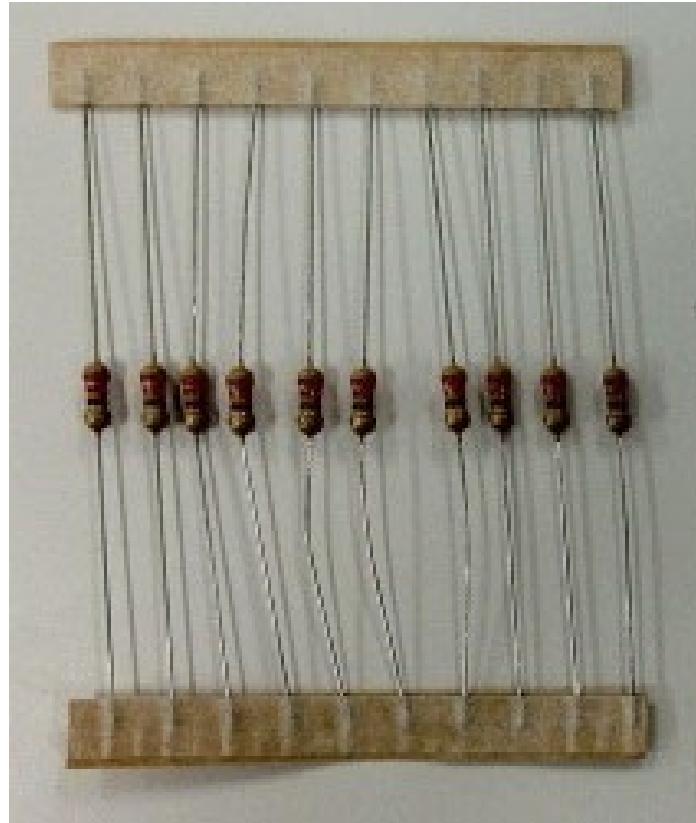


キットのパーツ 3/3

47

- 抵抗

- 今回は $10k\Omega$ (茶黒橙金)

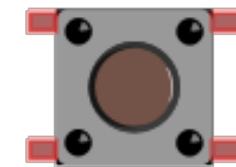


- フルカラーLED

- 最長は黄色ケーブルに、その他は抵抗に、つなぐ



- ボタン



- その他にもサーボや





可変抵抗

- 可変抵抗を読み取る

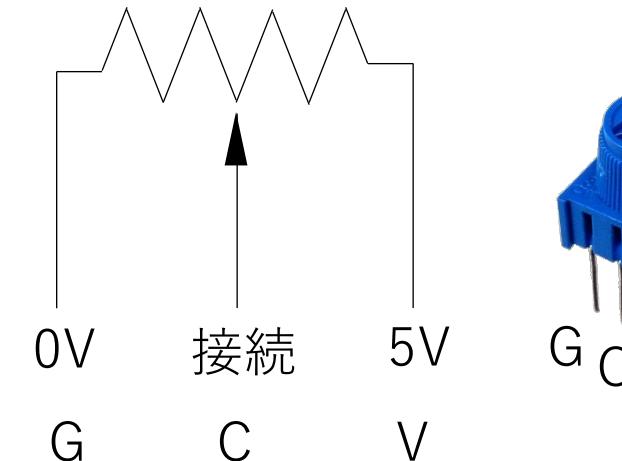
- 可変抵抗を使って、ピンに与える「電圧」を変化させ、その電圧を Arduino内蔵のA/Dコンバータで読み取る
- ArduinoにはA/Dコンバータが入っており、Analogと書かれた部分で読み取ることができる

```
int val; // 変数を定義  
val = analogRead(ピン番号); // 抵抗の電圧を読み取り
```

- DigitalとAnalogの同じ番号のピンは「同じ番号では」同時に利用できないので注意

- Arduino Unoの場合、アナログ入力ピンの0~5番をデジタル入出力ピンの14~19として利用できる
- さらにマクロ定義のA0~A5番を利用すると基板のシルク記載の通りに指定でき、次のような指定が可能である

```
pinMode(A0,OUTPUT);  
digitalWrite(A0,HIGH);
```





ピンのプルアップ、プルダウン

49

- INPUT_PULLUPを使うとプルアップ抵抗を省略出来る
- デジタル入出力ピンで押しボタンスイッチなどを繋ぐ場合、押されていない時の入力電圧を作るためプルダウンあるいはプルアップ抵抗が必要です
 - なぜ必要かは動画のActive LowやActive Highの議論が理解できている必要があります
 - プルアップ抵抗はマイコンチップ内部にすでに実装されており、これをピンモードの設定でINPUT_PULLUPキーワードを使用すると利用できるようになります
 - つまり、外部のプルアップ抵抗を省略できます
- 使用例
 - pinMode(2,INPUT_PULLUP);
 - 2番ピンがマイコン内部の抵抗を介して電源レベルに繋がるため、2番ピンに直接ボタンを繋ぎ、ボタンのもう片方をGNDにつなぐだけでよい
 - digitalRead(2)とすることで、ボタンが押されていない時にHIGH、押された時にLOWになる
 - つまり、この場合はActive Lowとなる、Pull Upだから当然ですよね





実装上の注意

- とにかく優しく扱うこと
 - チカラは必要なく、きちんと向きや方向があっていれば簡単に入る
- 予習は必須、動画およびこのテキストは事前に読んで理解しておくこと
 - 当日動画を見直す時間はない
- わからなかったら質問すること
 - 教室が分かれており、対応上空きになる可能性もありますが、いつでもオンライン質問できます
 - 質問することで共有され他の学生も助かります
- 小さいパートに気を付けること！本当に刺さります
 - 地面に落として踏まないように
- 最後に元通りに片づけますので、最初のおさまりを覚えておくと
 - 写メを撮っておくとよい
- パートはなくさないでください
 - すべて西研の所有物ですので、なくしたり、壊したりしたら、黙秘せず、すぐに連絡してください
 - 最も困ることは次の実験ができなくなることです





XbeeによるZigBee通信

- X-CTUでXBeeを初期設定する
 - 必要な場合は指示しますので、設定してください
 - 2台のXBeeを使用しArduinoとパソコンの双方から簡単な通信確認をとる
 - なお、すでに初期設定済みであるが、場合によっては修正が必要かもしれない
- XBeeの通信モード
 - ATモード（透過モード/Transparent mode）
有線シリアル通信で制御するためArduinoで一般的なシリアル通信をそのまま無線化できる
(シリアル通信は知っているかい?)
 - APIモード
APIフレームと呼ばれるデータ構造で通信を行う方法で高機能
- 実験では基本的にAPIモードを利用する

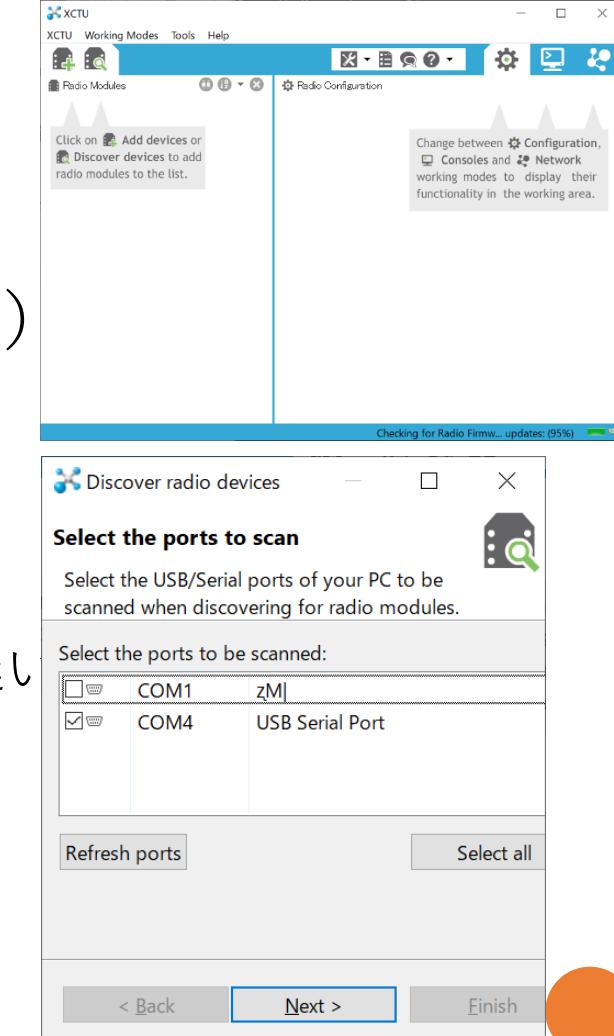




USBシリアルXbeeボードによる更新

52

- Digiが提供しているX-CUTをインストール
 - <https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu#productsupport-utilities>
 - 起動したら左上の+の「Add devices」もしくは🔍の「Discover devices」でXbeeを認識させる
 - USB Serial Portなどと表示されるので「これだけ」選択、デフォルトの設定ままでFinishを選択すると、発見してくれる
- 必要に応じてファームウェアを更新する（指示があった場合のみ）
 - 少々時間がかかるので、一つのマシンでまとめてやった方が早い
 - S2B(Xbee PRO)は古くS2Cでなければ接続できない？#HelpからLegacyを入れると認識はする
 - FirmwareをUpdateボタンで更新すること(現在4061が最新)
 - ScanすればほかのXbeeも発見でき、リモートでファームを更新できる(遅い)
- IDは全員違う番号になっている
 - 番号ごとにコーディネータが1つ存在
 - 番号が重なると正しく動作しない！



XBeeの設定における躊躇どころ

53

- Arduinoに乗ったままでは何もできない
 - XBeeをシリアルモジュールに搭載しPCにつなげて書き換える
- PAN ID
 - 2401から2415まで設定 それぞれにラベルが振られている
- APIモードにすること
- 次の設定パラメータの通りになっていれば問題ないはず
- X-CTUで見えない！
 - 解法1：X-CTUのシリアルコンソールを開いてresetを押し、+++と連打、OKと帰ったらすかさず、AT FS FORMAT CONFIRMと入力してファイルシステムを初期化する
 - 解法2：ファームウェアを更新する





設定パラメータ（指示があった場合のみ設定）

54

Networking

- ID: 242201-(順番)
- SC: 7FFF
- SD: 3
- ZS: 0
- NJ: FF
- NW: 0
- JV: 1
- JN: 0
- OP: 3320
- OI: A6D
- CH: 14
- NC: 14
- CE:
 - Router: 0
 - Coordinator: 1
- DO: 0
- DC: 0

Addressing

- SH: 13A200
- SL: 41637A8A
(P2P通信で必要)
- MY: C262
- MP: FFFE
- DH: 0
- DL: 0
- NI:
- NH: 1E
- BH: 0
- AR: FF
- DD: A0000
- NT: 3C
- NO: 0
- NP: FF
- CR: 3

ZigBee Addressing

- SE: E8
- DE: E8
- CI: 11
- TO: 0

RF Interfacing

- PL: 4
- PM: 1
- PP: 8

Security

- EE: 0
- EO: 0
- KY: 00
- NK: 00

Serial Interfacing

- BD: 3
- NB: 0
- SB: 0
- RO: 3
- D6: 0
- D7: 1
- AP: 2
 - Node-REDとCoordinatorをつなげる際の便宜を考え、0にする場合もあり
- AO: 0

AT Command Options

- CT: 64
- GT: 3E8
- CC: 2B

Sleep Modes

- SP: 20
- SN: 1
- SM: 0
- ST: 1388
- SO: 0
- WH: 0
- PO: 0

I/O Setting

I/O Sampling

- デフォルトから設定値が異なる場所について赤で示しており、X-CTUでは青色の右下三角マークが現れる
- 変更を施し、Writeする際に更新されるパラメータは、X-CTUにおいて緑色の右下三角マークが現れる



プログラムの概要

// Program for Arduino version KEIO SD Westlab 2021.3

// Author: Tada Matz, Kanami Yuyama, Hiro West

// 定義

#define ARD_LED 13

#define RECVINTERVAL 50

#define BTNO 10 // 10番ピン以降(12まで)を連続してボタン入力とする

#include "source.h" // 独自軽量化XBee関数

#include <MsTimer2.h> // タイマー割り込み

MyXBee myxbee;

// メッセージ受信用バッファ

String message;

int msglen;

// ボタン入力検出用関数

int cDflag[3];

int clickDetection(int _pin) {

int buttonClicked = 0;

int button = _pin - BTNO;

int newButtonState = digitalRead(_pin);

if (cDflag[button] == HIGH && newButtonState == LOW) buttonClicked = 1; // 状態がHIGHで新しくLOWなら

cDflag[button] = newButtonState; // 状態を更新(チャタリング防止)

return buttonClicked;

}

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** kit | Arduino 1.8.13 (Windows Store 1.8.42.0)
- File Menu:** ファイル
- Toolbars:** 編集 スケッチ ツール ヘルプ
- Tab Bar:** kit XBee.cpp XBee.h source.cpp source.h
- Code Area:** C++ code for Arduino. It includes defines for ARD_LED, RECVINTERVAL, and BTNO, and includes source.h and MsTimer2.h. The code implements button detection logic using MsTimer2 and sends messages via XBee.
- Serial Monitor:** Shows the message "保存しました。" (Saved.)
- Status Bar:** COM1のArduino Uno



通信部分

```
void sendmsg(String message) {  
    myxbee.broadcastXBeeData(message);  
}  
  
int recvret();  
void recvmsg(void){  
    String receiveStr = myxbee.receiveXBeeData();  
    MsTimer2::stop();  
    if(msglen > 0){  
        // 重複割り込み発生  
        Serial.println("ERR:Duplicated RCV Interrupt");  
    }  
    msglen = receiveStr.length();  
    if (msglen > 0){ // パケットを受信した  
        int messagelen = receiveStr.substring(0, 3).toInt();  
        int loc = 3;  
        message = receiveStr.substring(loc, loc+messagelen);  
        // [即時処理]変数代入など簡単な処理を記述、時間のかかる処理はアウト  
    }  
    MsTimer2::start();  
    // [通常処理]比較的時間のかかる処理はここで記述することができる。  
    if(msglen > 0){  
        // message;  
    }  
}
```

- 送信はいたってシンプル
- 相手を指定する場合は、sendXBeeData(ノードID, Message);とする
- タイマー割り込みを用いて受信データを読み込む
 - 本来はシリアル通信による割り込みを用いるがPCとのシリアル通信に利用されている
- 50msに一度確認している
- パケットを受信するとメッセージ長が0よりも大きくなることを利用
 - 現在の中身はあくまでも例





プログラムの本体

- loop()の中で主に設計する
- Node-REDの方で工夫して「ゴミ」を取るため、コンマで囲って送りたいデータを送ることにしている
 - スマートな方法はいろいろあるが、javascriptをなるべく必要としない方法としている

```
void setup() {  
    Serial.begin(9600);  
    myxbee.init(Serial);  
    pinMode(ARD_LED, OUTPUT);  
    MsTimer2::set(RECVINTERVAL, recvmsg); // タイマー割り込みをかける  
    MsTimer2::start();  
}  
  
void loop() {  
    sendmsg(",送りたい値,");  
    delay(1000); // データは頻繁に送信しないように  
}
```

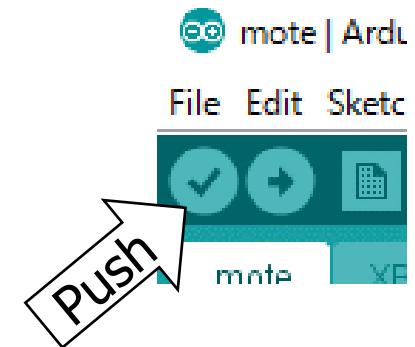




プログラムの変更とコンパイル

58

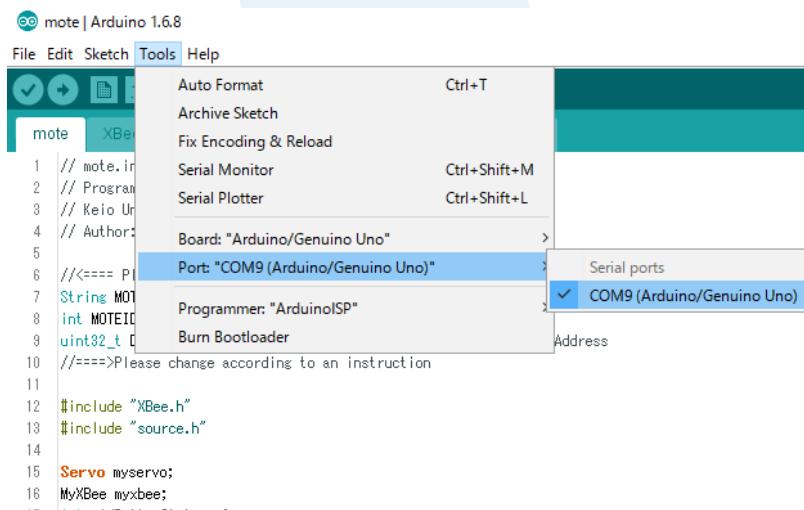
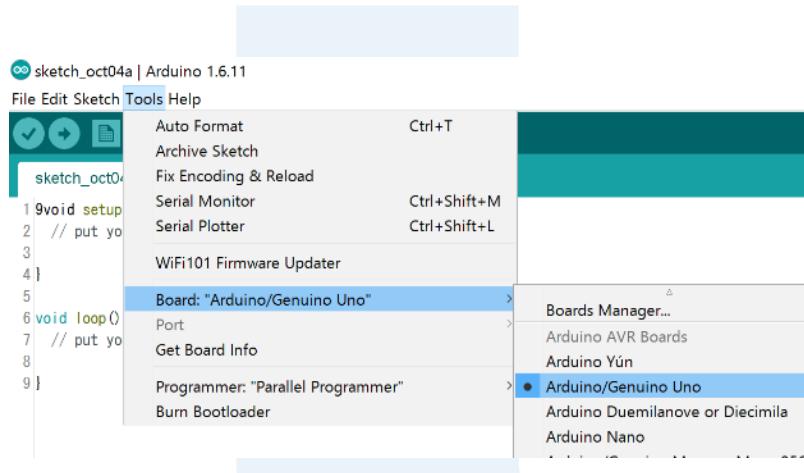
- 最初の行は各個人で異なる設定となる
 - セミコロン「;」消さないように！
- main.h, source.h, source.cpp, XBee.h, XBee.cppはkit.inoと同じ階層に入れること
- オリジナルは保存しておいたほうがよい
- “✓”ボタン押すとコンパイルが実行される
 - 書き込みは行わないので、記述があつていているかどうかを確認できる
- エラーの場合
 - エラーメッセージをよく読むこと
 - よくある間違い
 - セミコロン「;」が抜けている
 - String変数にダブルクオート「""」がついていない
 - インターネットで調べる
 - 得意な友達やTAに聞く



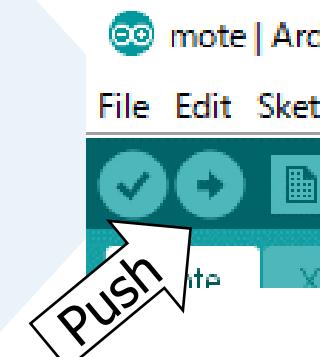
プログラム書き込み（毎回確認すること）

59

- Arduino – USB – PCと接続
- ボードはUNOを選択
- ポートを選択(複数の場合は注意)
- “**SERIAL SELECT**”を下“**USB**”(**書込モード**)
- “→”を押す
- 「正常に書き込まれた」というメッセージを確認
- SERIAL SELECTを上”**MICRO**”(**動作モード**)



書き込むときは
“**SERIAL SELECT**”
を“**USB**”に！！！



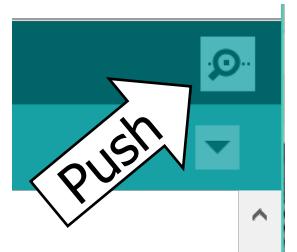


動作チェック

60

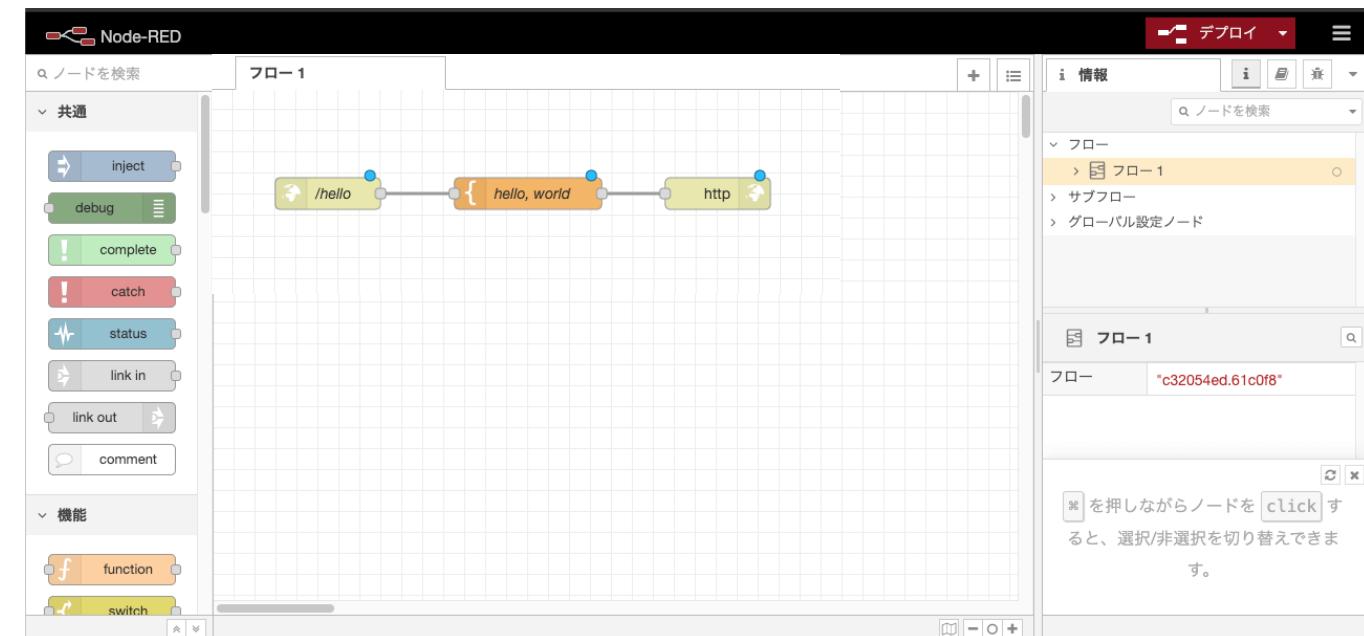
- シリアルモニターで通信内容を見る
 - PCとArduinoはシリアル通信しており、Xbeeとの通信も見ることができる
- よく見てみる
 - 送っているタイミングでスクロールするか確認
 - 送っている内容が含まれているか確認
 - 送っている内容が正しいか確認

The screenshot shows the Windows Serial Monitor window. At the top, it says "COM5 (Arduino/Genuino Uno)". The main area displays a series of received messages in a log format:
~\$□□□□L2255000000version3-2
□Send ZbTx
The log ends with an empty line and a checked "自動スクロール" (Auto Scroll) checkbox at the bottom.





- ・「データの流れ」をフローで描画しプログラムからある程度解放された形として記述
- ・データの流れを記述することで、データの変換や可視化などあらゆる処理を実現
 - ・3,000を超えるプラグインが存在、スマートスピーカ、データベース、メール、skype、LINE、Twitter、HomeKitなどIoT、AIもとにかく何でも「簡単なことならすぐにできる」ようになる
 - ・ブラウザ上で動作し、インターネットにつなげておけば世界のどこからでも操作できる
 - ・中身はNode.jsにより動作し、javascriptで記述するが、意識する必要はない
- ・試してみよう（必須ではない）
 - ・<https://users.sensetecnic.com/register>
 - ・FRED shortより無料で利用できる（FREDのサーバを間借りする）
 - ・制限はあるが充分利用できる
- ・なお、実験では利用できない
 - ・オンラインでArduinoと接続できない
 - ・Wi-Fiシールドならできる





Access to <https://fred.sensetecnic.com>

62

- フリーのNode-REDサーバを提供

The screenshot shows a browser window titled "Sense Tecnic Accounts" with the URL "users.sensetecnic.com/app/services". The main content area displays the "FRED" service under the "SERVICES" tab. The FRED section includes a logo, a brief description, and a link to "Click here to go to FRED". Below this, there is a "MQTT" section with a cloud icon and a brief description.

Sense Tecnic Accounts

FRED MQTT InfluxDB Manager

Plugfest west@keio

FRED

Front End for Node-RED (FRED) manages instances of Node-RED for multiple users in the cloud. Node-RED is a visual tool for wiring the Internet of Things developed by IBM Emerging Technology and the open source community. With Node-RED you can wire up input, output and processing nodes to create flows to prototype IoT applications.

Click here to go to FRED >

MQTT

The STS MQTT service allows you to send and receive data streams to your devices and FRED account easily using the standard MQTT protocol. Use the service to manage your client connections and access control for public and private topics.

<https://mqtt.sensetecnic.com>

Provide Feedback

The screenshot shows a browser window titled "FRED: Front End for Node-RED" with the URL "fred.sensetecnic.com". The page features a "sense tecnic" logo and a "LOGIN" button. The main heading is "Welcome to FRED" with the subtext "We're hosting Node-RED so you don't have to". Below this, there is a section titled "PAID PLANS" with four options: "FRED Tall" (\$9.99/mo), "FRED Grande" (\$49.99/mo - recommended), "FRED Venti" (\$249/mo), and "FRED Short" (Free). Each plan includes a checkbox for "No node limit (up to 500mb memory)" and a checkbox for "24x7 run time, always".

Resources

LOGIN

Welcome to FRED

We're hosting Node-RED so you don't have to

PAID PLANS

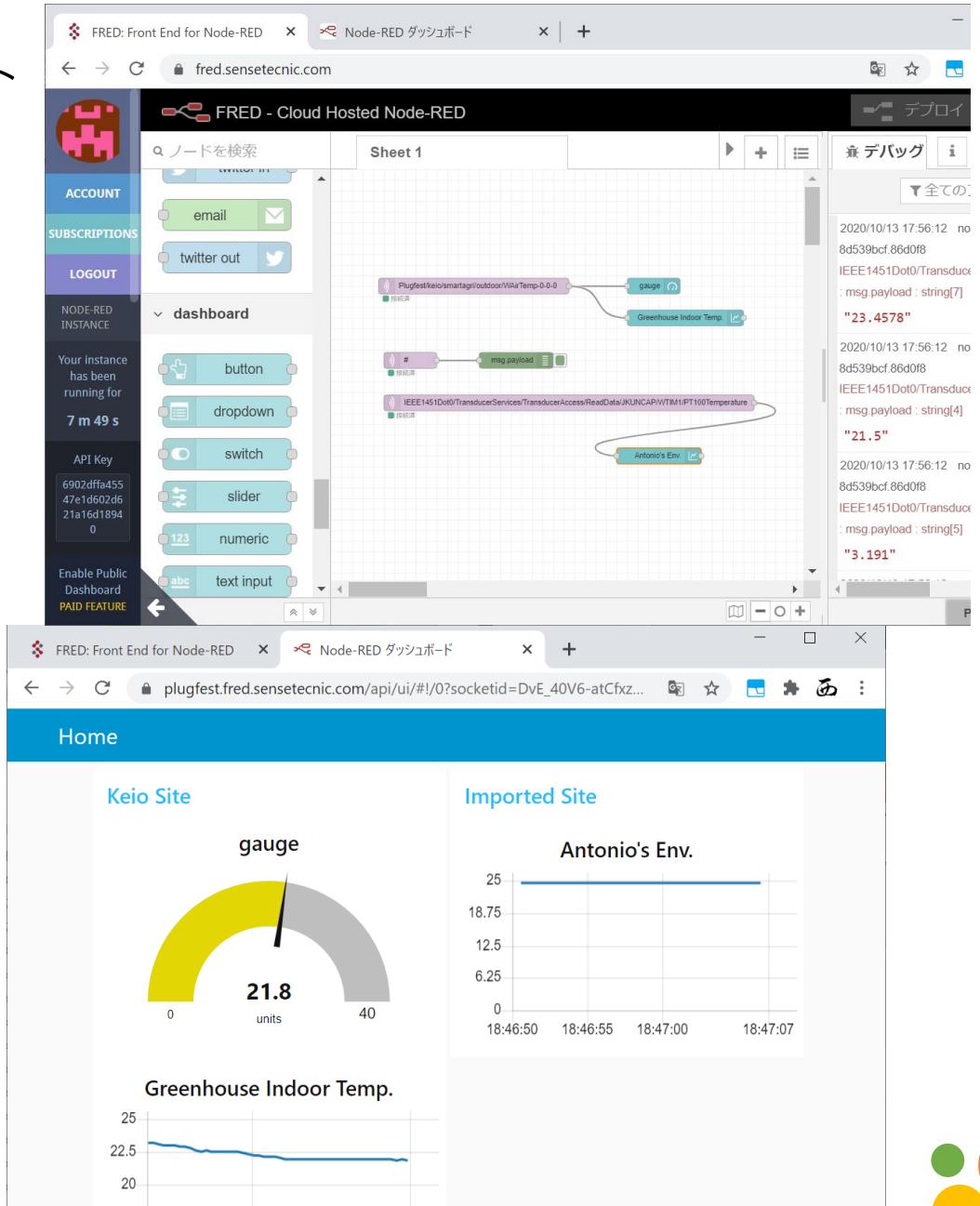
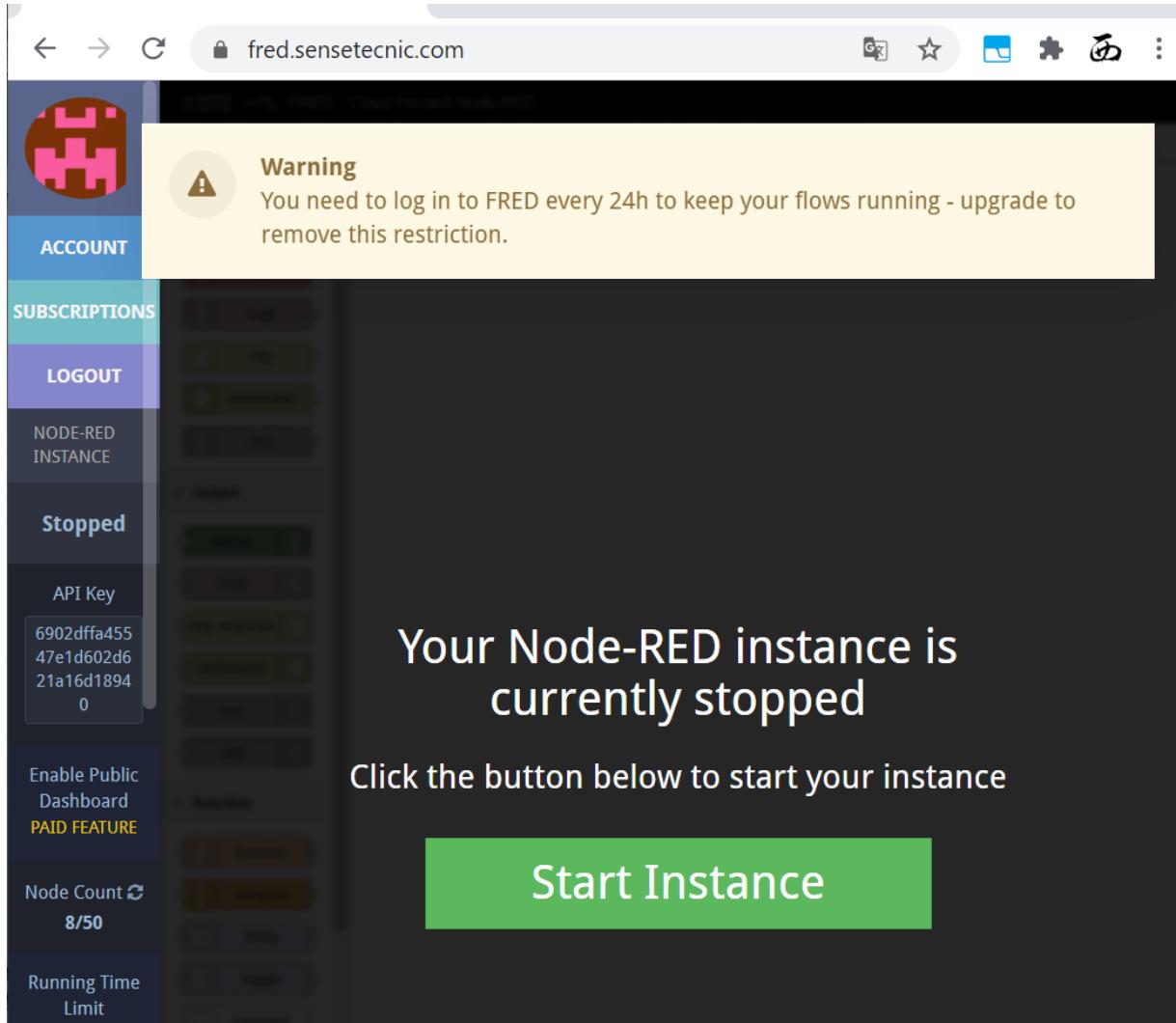
Plan	Price	Node Limit	Run Time
FRED Tall	\$9.99 /mo	<input checked="" type="checkbox"/> 150 node limit (limited memory)	<input checked="" type="checkbox"/> 24x7 run time, always
FRED Grande	\$49.99 /mo	<input checked="" type="checkbox"/> No node limit (up to 500mb memory)	<input checked="" type="checkbox"/> 24x7 run time, always
FRED Venti	\$249 /mo	<input checked="" type="checkbox"/> No node limit (up to 1gb memory)	<input checked="" type="checkbox"/> 24x7 run time, always
FRED Short	Free	<input checked="" type="checkbox"/> 50 node limit (limited memory)	<input checked="" type="checkbox"/> 24 hour run time



Node-REDで設計する

63

- Start Instanceが現れたらこれを押すとスタート
- 他の人と設計を共有することもできる





Node-REDを試してみよう

64

Sense Tecnic Accounts +

users.sensetecnic.com/login?return=https://fred.sensetecnic.com

Sign in to the SenseTecnic Platform

fred@e-west.info

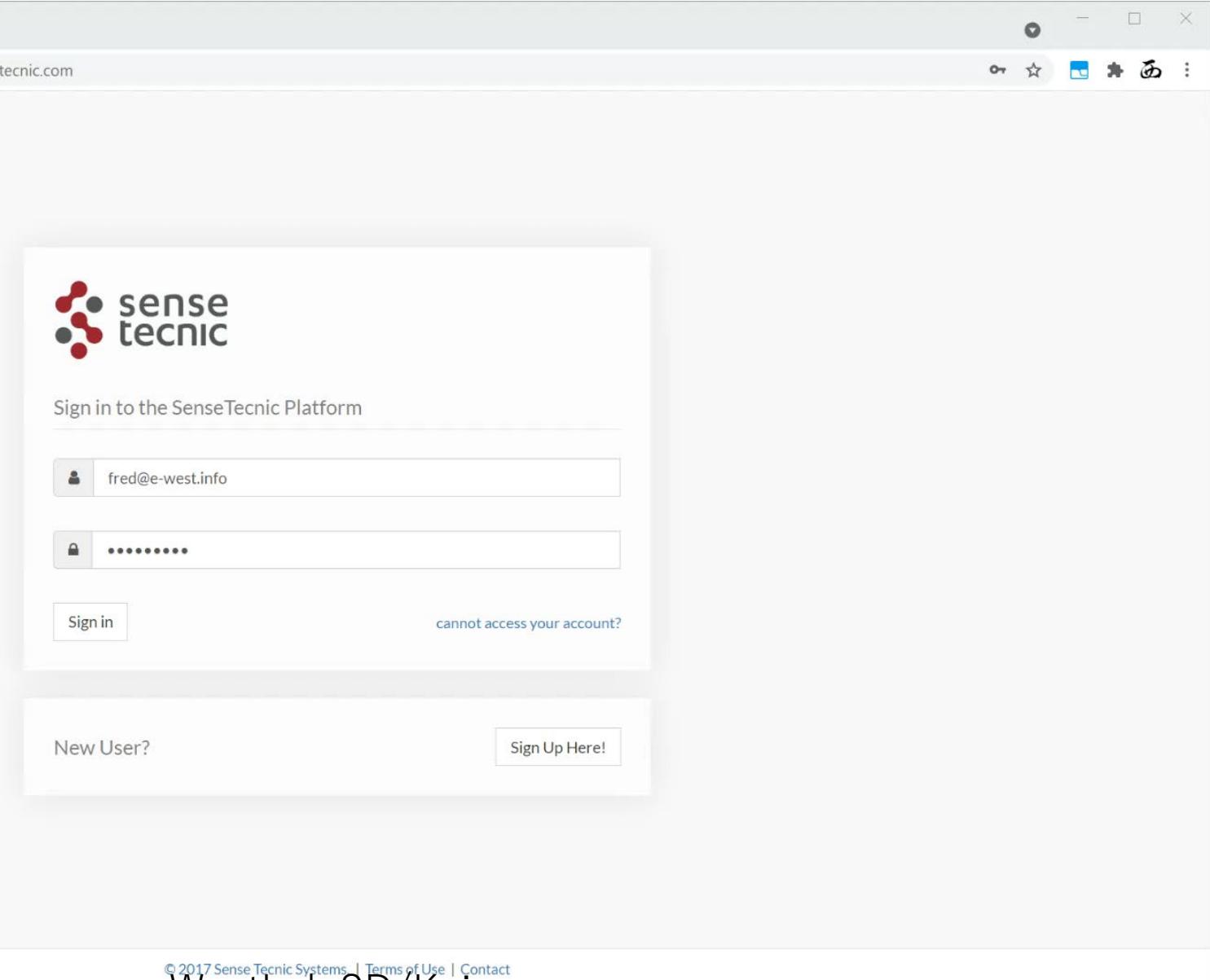
.....

[cannot access your account?](#)

[New User?](#) [Sign Up Here!](#)

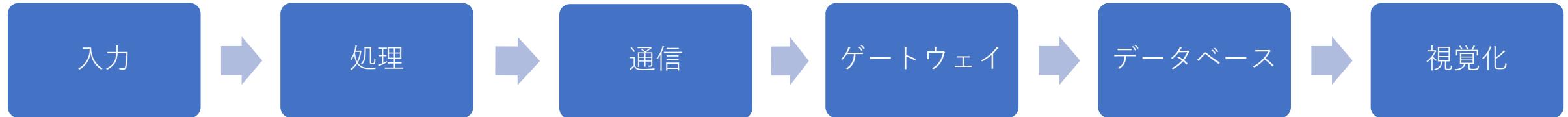
© 2017 Sense Tecnic Systems | [Terms of Use](#) | [Contact](#)

WestLab SD/Keio

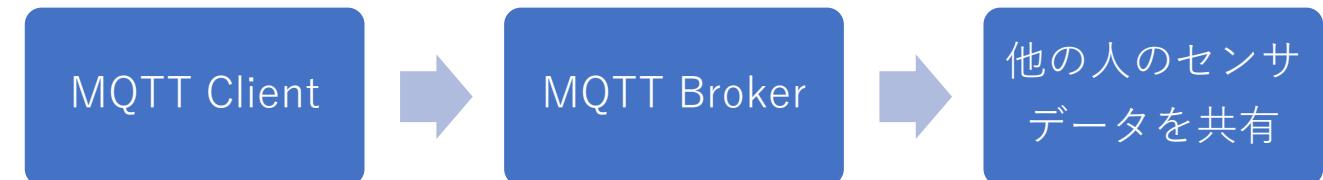


システムの流れ

65



可能であれば他の人のセンサの値も一緒に表示してみよう



- APIモードのため、**コマンドを含むすべての文字が見えててしまう**
 - APIモードのみ無線接続の確認や無線経由の設定が行える
 - 本来データのみを転送するモードを選ぶべきであるがあえてAPIモードにしている
- APIエスケープモードを利用しているため、通常のデータ以外はエスケープされる
 - エスケープとは、ここでは「**文字コードを邪魔にならない大きな値にすること**」
 - つまり、コマンドを意識せず、文字を見ることができる
 - 例えば10という数値データを受け取るとすると
?(エスケープによりなんだかわからない文字列)?10?(なんだかわからない文字列)
といった具合に、10だけきちんと読めるようになる
 - 10の部分だけ切り出せるようにすればよい
- ここでは、何かしらの文字を用いて数値部分を切り出して用いる
 - 例えば"/'スラッシュを用いて/10/として送信すれば、/より前と後はゴミと判断できる
 - Arduinoでは、例えばsが文字配列、aが値とするとString型の文字列変数に対して、`String s = "/" + String(a) + "/";`などとする



Node-REDによる視覚化

67

Node-REDによる視覚化

デプロイすると保存して実行

フロー 1

chart

msg.payload

COM9

split

switch

chart

gauge

プロパティ

図サイズ: 6 x 6

ラベル: chart

種類: 折れ線グラフ

ポイントを表示

X軸: 直近 20 秒 又は 1000 ポイン

X軸ラベル: HH:mm:ss UTCを

Y軸: 最小 0 最大 1024

凡例: 非表示 補完 直線

プロパティ

Group: [ホーム] デフォルト

Type: Gauge

Label: gauge

Value format: {{value}}

Units: units

Range: min 0 max 1024

プロパティ

名前: 名前

プロパティ: msg. payload

分割: /

メッセージのストリームとして処理

配列: 固定長 1

オブジェクト: 各key/valueペアのメッセージを送信

keyのコピー先: msg.

最初に合致した条件で終了

メッセージ列の補正

リクエスト

デフォルトの応答タイムアウト: 10000 ミリ秒

有効: 1 個のノードが、この設定を使用しています

プロパティ

名前: 名前

プロパティ: msg. payload

分割: /

メッセージのストリームとして処理

配列: 固定長 1

オブジェクト: 各key/valueペアのメッセージを送信

keyのコピー先: msg.

最初に合致した条件で終了

メッセージ列の補正

有効: 1 個のノードが、この設定を使用しています

プロパティ

名前: 名前

プロパティ: msg. payload

分割: /

メッセージのストリームとして処理

配列: 固定長 1

オブジェクト: 各key/valueペアのメッセージを送信

keyのコピー先: msg.

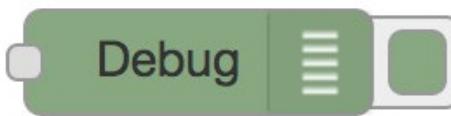
最初に合致した条件で終了

メッセージ列の補正

有効: 1 個のノードが、この設定を使用しています



Inject



Debug



Function

68

- Injectノードは、エディタ内でクリックすることで手動でフローを始動させることができ、また、一定間隔で自動的にフローを始動させることもできる
 - 送出メッセージに、payloadおよびtopicプロパティを設定できる
 - 通常はpayload
 - payloadには様々な型を設定することができます。
 - フローまたはグローバルコンテキストのプロパティ値
 - 文字列型、数値型、Boolean型、バッファ型、オブジェクト型
 - エポックミリ秒
- Debugノードは、Debugサイドバーにメッセージを表示させる
 - メッセージを受信した時刻とそのメッセージを送出したDebugノードの情報が含まれる
 - ソースノードIDをクリックするとワークスペース内のどのノードなのかがわかる
 - ノードのボタンで出力の有効化無効化を制御できる
 - ランタイムログにすべてのメッセージを出力したり、Debugノードの下に短い（32文字）のステータステキストを表示させることができる
- Functionノードは、JavaScriptコードを実行する
 - 例えば、長さを返すならば、`var newMsg = {payload: msg.payload.length}; return newMsg;`



- Changeノードは、メッセージプロパティ変更、コンテキストプロパティの設定が可能
 - 各ノードには、順番に適用される複数の操作を設定可能
 - 値の代入 - プロパティの代入、各種型を利用でき、既存のメッセージやコンテキストプロパティもOK
 - 値の置換 - メッセージプロパティの一部を検索、置換
 - 値の移動 - プロパティの移動や名称変更
 - 値の削除 - プロパティの削除 プロパティ設定に軽量クエリ変換言語JSONata形式を設定できる
 - Switchノードは各メッセージを評価し、メッセージを異なるフローに振り分ける
 - 評価するプロパティ（メッセージプロパティまたはコンテキストプロパティ）を設定
 - ルールには以下の4種類がある
 - Value ruleは設定されたプロパティに対して評価する
 - Sequence ruleはSplitノードによって生成されるようなメッセージシーケンスを利用できる
 - JSONata式はメッセージ全体を評価し trueの値を返した場合に一致したとみなす
 - その他は前述のルールのどれにも一致しなかった場合に一致する
 - 全一致ルールにメッセージを送出するか、一致ルールがあった時点で評価をやめるか指定可能
 - Templateノードは、メッセージプロパティからテキストを生成する(Mustache記法)
 - たとえば、This is the payload: {{payload}} !
 - Splitノードは、フローを分割、joinノードは結合する



動作のコツ

- シリアルのコンフリクトに注意すること
 - シリアルは複数を相手にできません。同じシリアルを複数で使うとコンフリクトします
 - ただし、実験では基本的にはシリアルはコンフリクトしません
 - PCはSerial AでArduinoと通信
 - Node-REDはSerial BでXbeeボードと通信
- XBeeの設定を変える場合などでコンフリクトする場合は、Node-REDを止める必要があります
 - Node-REDを動かしているコマンドラインでCtrl-Cとして止めると良いです
- エラーメッセージはよく読むこと
 - 書き込めない場合
 - 切り替えスイッチをまず見る
 - USBシリアルのポートや、Arduinoボードのタイプをよく見る





複数センサの値を表示

71

チャレンジ（1）

- 複数のArduinoを準備し、プログラムと設計とする
- 複数のArduinoのセンサ値が混ざってNode-REDで表示されることを確認する

チャレンジ（2）

- 複数のArduinoのセンサ値を異なるメータで表示させる
(ヒント) メッセージに識別子を付与し、Node-RED内でこれを認識、仕訳する



- MQTTとは
 - 普通の通信はPeer-to-Peerつまり、サーバクライアントモデル
 - サーバとクライアントの間で相互通信する
- 問題点
 - 例えば複数の相手に同時に投げたいし、相手も変えたいという場合に面倒で、負荷がかかる
 - 相手が変わったらサーバが変わる？設計が面倒
- MQTTはpublish-subscribeモデルである
 - トピックという名前でデータをブローカーに投げると、同じトピックを持っているクライアント全員にデータが届くという仕組みで、構造がシンプルなため組み込みでも簡単に実装できる
 - IoTでよく利用されるプロトコル
- Node-RED内の設計だけでデータ共有ができる！ペアを組んでやってみよう
 - MQTTブローカへのpublishノードで送り、subscribeノードで値を受け取って表示させればよい





MQTT brokerの場所

- mosquittoを利用します
- 皆さんのPC
 - コマンドラインプロンプトを開く(ウィンドウズアイコン→cmdと入力)
 - cd c:\Program Files\mosquitto
 - mosquitto -v
 - 実行させたコマンドラインプロンプトは閉じないでください
- 実験部屋に専用のMQTT broker mosquitoを設置します
 - そのIPアドレス(相手先の名前)は別途お伝えします
 - ログイン・パスワードは必要ありません
- フリーのオーブンはMQTT brokerも利用できます
 - broker.hivemq.com
 - パスワードもログインも不要で、1883番ポートにアクセスすれば、すぐに利用できます。
 - test.mosquitto.org
 - 同様の設定で利用できます。その他、Googleで検索すると様々な同様のサービスがあることがわかります。





片付け

74

- 元通りに片づけること
 - 何も無くさないこと、繰り返しますが私物です
 - 何も壊さないこと
 - 力を入れて無理やりやらないこと
- ちゃんと蓋を閉じる
 - 閉じない場合は入れ方が間違えている
 - 蓋を開けた時、写メをとって入れ方を覚えておくとよい





補足

75

- 本日の機材 計9000円ぐらい（大体）
 - Arduinoは約2500円
 - 無線デバイス Xbeeは約2000円
 - Arduinoシールドは約2500円
 - ケーブルやケースなどパーツが約2000円
- Raspberry Piもおすすめ
 - LinuxベースのOSが利用でき、Node-REDもArduino IDEも動作し設計できる
 - IoTノードとして必要十分な機能を持っており、Arduinoよりも性能が高い
 - カメラやディスプレイなども搭載でき、AIにも対応、もはやPC
 - 欠点は信頼性





片付け

- 元通りに片づけること
 - 何も無くさないこと、繰り返しますが私物です
 - 何も壊さないこと
 - 力を入れて無理やりやらないこと
- ちゃんと蓋を閉じる
 - 閉じない場合は入れ方が間違えている
 - 蓋を開けた時、写メをとって入れ方を覚えておくとよい



レポートと ディスカッション

- しっかりと学習内容をまとめよう





レポートとディスカッション

78

- レポートは手書き禁止
 - MS Wordなどを利用して書くこと（手書き文字は読みにくいのでスキャンも不可）
 - 回路図は、写メか、エミュレータで入力して全画面キャプチャしなさい（もしくは実行画面）
 - 設計した回路（もしくはコード）もすべてレポートのWordファイルに張り付けて、PDFで提出
 - 実験目的、実験理論、実験手順、実験方法、実験結果などは絶対に記述するな
 - 実験レポートは写経ではなく、たとえテキストであっても写すのは剽窃と同じである
 - コピペレポートは厳禁と指導するのに、テキストを写させるのは意味不明である
 - 与えられた問題のみ回答すること
 - たくさん解いても加点されないが、ミスがあると減点されるため得をしない
 - シミュレータで設計可能であり、その設計のスクショと動作結果について記述、考察すること
 - オリジナリティを最も重視します（LMSによる自動剽窃チェックを利用します）
 - ググっても参考文献をつけること、参考文献の言及があれば剽窃ではない
 - 逆に手の内を明かしているので、覚悟を持ってこの剽窃チェックに捕まらないようにコピペしなさい
 - 残念ながら、偶然にも一致してしまった場合でも減点の対象となります
 - これは、オリジナリティが不足したことによる原点という扱いになります
- ディスカッション
 - ディスカッションが必要と思われる場合のみSlackもしくはLMSから直接keio.jpアカウントに連絡が届きます





演習問題集（1）

1. ボタンを押すとLEDが1秒点灯し、同様に離すと別のLEDが1秒点灯するようにせよ
2. 可変抵抗とサーボモータを組み合わせて、可変抵抗でサーボを制御せよ
 - 要するに、リモートロボットを作る
 - <https://www.arduino.cc/en/Tutorial/Sweep>
3. タイマー関数とCallbackを用いて、ボタンを押したら2秒間点灯するようにせよ
4. 毎秒LEDが200ms点灯するようにせよ
 - 正し、正確に毎秒であること
 - sleepなどを用いると、sleep以外の処理時間が加算されていく
5. 足し算電卓を作成せよ
 - ボタンは0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, =があればよい
 - 答えはシリアル表示とする
6. シリアル足し算電卓を作成せよ
 - シリアルから、式を入力してもらい、シリアルで答えを返す
 - 受け付けるのは上記と同じ





演習問題集（2）

7. ビンゴ抽選機を作れ

- ・ 抽選ボタンを押すとビンゴ抽選を行い、ある数を「重複なく」シリアルで出す
- ・ 全ての球が数字ができったら、「終了」LEDを点灯させる
- ・ リセットボタンを押すと、初期状態に戻り、再び抽選できる
- ・ 抽選の数字は1から30としなさい

8. 2個のArduinoを1本で接続し、LEDを制御せよ

- ・ 片側のArduinoのボタンを押すと、もう片方のArduinoのLEDが1秒点灯し、同様に離すと別のLEDが1秒点灯する

9. 2個のArduinoをパラレル接続し、サーボを制御せよ

- ・ 片側のArduinoのつまみの角度に応じて、もう片方のArduinoのサーボが動作する

10. 2個のArduinoをシリアル接続し、送信したa-zの文字をシリアルで出すようにせよ

- ・ 配線電圧のHIGH、LOWを制御して伝えるようにすること

11. 2個のArduinoをシリアル接続し、サーボを制御せよ

- ・ シリアル通信で片側のArduinoのつまみの角度に応じて、もう片方のArduinoのサーボが動作する





演習問題集（3）

81

12. 3個のArduinoをカスケードシリアル接続しLEDを制御せよ

- プライマリとなるArduinoのボタンを押すと、残り全てのArduinoのLEDが1秒点灯する
- （ヒント）カスケードなので、プライマリ以外のArduinoは親からの情報の受信と子への送信を同時に使う

13. 3個のArduinoをカスケードシリアル接続しサーボを制御せよ

- プライマリArduinoのつまみの角度に応じて、残り全てのArduinoのサーボが動作する

14. 3個のArduinoをカスケードシリアル接続し送信したa-zの文字をシリアルで出すようにせよ

- 配線電圧のHIGH、LOWを制御して伝えるようにすること
- プライマリとなるArduinoから送信すると残り全てに伝わるようにする

15. 3個のArduinoをカスケードシリアル接続し送信したa-zの文字をシリアルで出すようにせよ

- マスターを規定せず、カスケード接続をループで構成して、指令を出したArduino以外のArduinoで文字が出るようにせよ
- 配線電圧のHIGH、LOWを制御して伝えるようにすること
- すべてのArduinoが送信・受信両方共できるようにする





演習問題集（4）

16. 3個のArduinoをバスシリアル接続しLEDを制御せよ

- どれか一つのArduinoのボタンを押すと、残りのArduinoのLEDが1秒点灯する、同様に離すと、別のLEDが1秒点灯する
- (ヒント)バスなので、一つが送信で後の1つは同時に受ける

17. 3個のArduinoをバスシリアル接続しサーボを制御せよ

- どれか一つのArduinoのつまみの角度に応じて、残りのArduinoのサーボが動作する
- 制御するArduinoを選択するためのボタンかスイッチも設けなさい
- アクティブHighに接続した配線に対してLowに繋ぐと、全体としてLowになるなどする

18. 3個のArduinoをバスシリアル接続し送信したa-zの文字をシリアルで出すようにせよ

- シリアル通信は、それぞれHIGH、LOWを制御して伝えるようにすること
- どれか一つのArduinoから送信すると残り全てに伝わるようにする
- アクティブHighに接続した配線に対してLowに繋ぐと、全体としてLowになるなどする





演習問題集（5）

19. 2個のArduinoをボーレートを合わす必要がないようにシリアル接続せよ

- 2本配線してよい

20. 2個のArduinoを次の仕様でシリアル接続せよ

- ボーレートを合わす必要がないようにすること(2本配線してよい)
- 通信中に配線を外しても、エラーとなって何も出ないようにする
- 配線外れを検出するわけではなく、通信中にエラーが発生しても問題ないようによること

21. 2個のArduinoをボーレートを合わす必要がないようにシリアル接続せよ

- 1本のみ配線するとし、100bps（動作限界は環境に依存する）までならばどのようなボーレートでもよいようにすること

22. 2個のArduinoを次の仕様でシリアル接続せよ

- ボーレートを合わす必要がないようにすること（1本のみ配線するとし、100bps（動作限界は環境に依存する）までならばどのようなボーレートでもよいようにすること）
- 通信中に配線を外しても、エラーとなって何も出ないようにする
- 配線外れを検出するわけではなく、通信中にエラーが発生しても問題ないようによること





第2回実験レポート

- 3つのArduinoを用いたセンサノードを構成する
 - それぞれに、温度センサとスイッチが搭載されており、温度情報とスイッチのON/OFF情報がコーディネータに送信され集められる
- Node-redがコーディネータで集められた情報を受け取り、温度情報を3つのノードそれぞれ別々に表示する
- また、温度情報の異常（異常となるスレッショルドは自由に設定してよい）および、スイッチが押されたことによる異常を通知する。
 - 通知する手法は自由度をもって取り組むこと。
 - 通知する際の優先順位はどのような設計でもよい。温度を優先しても、スイッチを優先しても、両方警告してもよい。できれば、それが「想定する利用からみて正しい優先順位である事」を説明しつつ、実装すること。
- Arduinoは、Tinckercadを用いて設計し、1つだけ設計すればよい。実機で実装したときでも動作することを想定し、シリアルで送信文字列を画面に表示させること。
- Node-redは、3つのArduinoに搭載されている温度センサとスイッチ、最低でも全部で $3 \times 2 = 6$ 個のInjectノードを用いて、Arduinoが送る文字列を投入、その後の動作を確認すること

