# 気象モニターを作ろう（発展編）
## －誰でもできるプロトタイピング－

## 第11回 気象モニタの活用(1)

徳島大学大学院理工学研究部総合技術センター
辻　明典
E-mail: a-tsuji@is.tokushima-u.ac.jp

# 概要

## 気象モニターをつくろう－誰でもできるプロトタイピング－

講師：川上 博（徳島大学名誉教授）
　　辻 明典（徳島大学大学院理工学研究部総合技術センター）

曜日・時間：土曜日　10時00分～11時30分

スケジュール：

① 10／ 1　開発環境セットアップ，前期の復習
② 10／ 8　概要と動作確認
③ 10／15　Processingによるデータ可視化(1)
④ 10／22　Processingによるデータ可視化(2)
⑤ 11／ 5　気象モニタの活用(1)
⑥ 11／12　気象モニタの活用(2)

# 気象モニター（全体構成）

## データ収集

### Zigbee無線センサネットワーク

センサノード(子機)

センサノード
（親機）

センサノード(子機)

## データ蓄積・分析

### クラウドサービス

Serial

MQTT Broker
Web Server
DB Server

## データ監視・制御

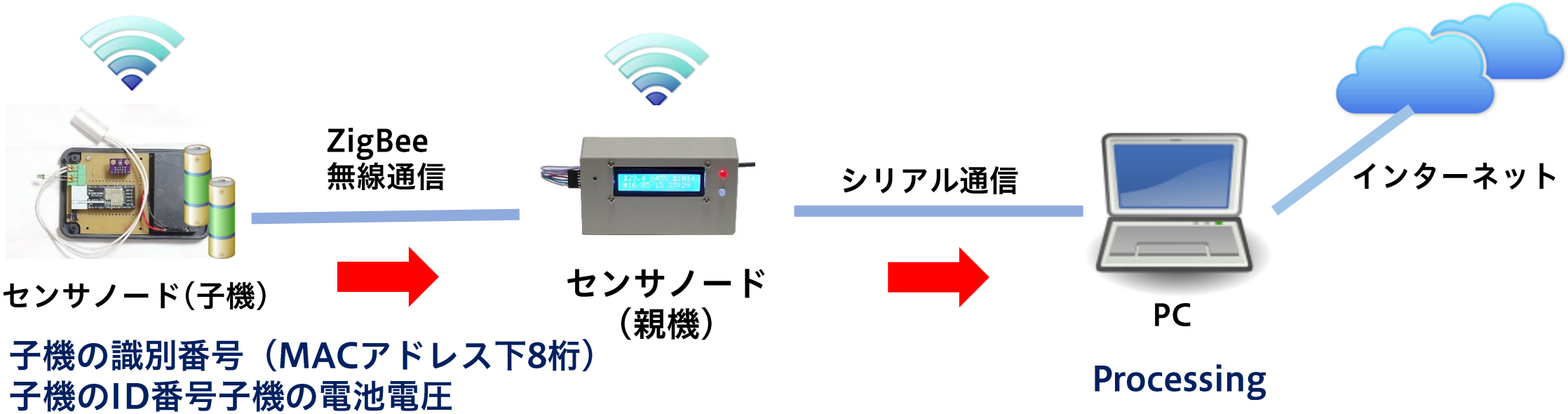
### モバイルサービス

Internet

制御端末
携帯端末
スマートフォン

# 気象モニター(取り扱う情報)

- 気象モニターに必要な情報の取得
- 親機－子機，親機－PC，PC－インターネット間で通信
- あらかじめ決められた形式のデータを通信に利用




子機の識別番号（MACアドレス下8桁）
子機のID番号子機の電池電圧
磁気リードスイッチ
照度センサ
温度，湿度，大気圧

親機に，Example1101Aを書き込んでおく。

# Example1101A

- 子機から親機へZigBee無線通信，親機からPCへシリアル通信

```
#include <SoftwareSerial.h>

SoftwareSerial xbee(10, 11); // RX, TX

void setup() {
  Serial.begin(19200);
  xbee.begin(19200);
}

void loop() {
  if (xbee.available()) { // 無線子機からのデータ受信
    String str = xbee.readStringUntil('¥n'); // 改行コードまで読み込む
    Serial.println(str);
  }
}
```

シリアルモニタにて次のデータを5秒間隔で受信

:ed=8100D8F6:id=10:ba=2620:a1=2199:a2=0007
:tm=2965:hu=6535:at=1012

# 気象モニターによる計測

- 温度，湿度，大気圧(BME280)，照度(NJL7502L)，スイッチ
  - 温度：-40度～85度
  - 湿度：0～100%RH
  - 大気圧：300～1100 hPa
  - 照度：0～530 lux
  - スイッチ：ON/OFF (0/1)
- 気温・・・大気の温度（地上で計測）
  - ～時の気温・・・～時の前1分間の平均気温
  - 日平均気温・・・1日の平均気温（1時～24時の毎正時24回の気温の平均）
  - 日最高気温・・・1日（0時～24時）の最高気温
  - 日最低気温・・・1日（0時～24時）の最低気温

  - 月平均気温，最高気温，最低気温
  - 年平均気温，最高気温，最低気温

# データの形式について

- データ形式
  - 数値(26.2,50.2,1013, …) の羅列のみでは意味を持たない
  - データ形式によって数値に意味を持たせる
  - ワープロ(DOC), 画像(JPG)等，あらかじめ決められた形式に従って情報をやりとり
- JSON形式
  - ウェブブラウザ等で使用されるJavascriptのオブジェクトの表記法(JavaScript Object Notation)を基にした軽量のデータ記述言語
  - データ例 {“temp”: 26.2, “humid”, 50.2, “press”, 1013}
- データ変換
  - 子機から送られてくるデータをJSON形式に変換

:ed=8100D8F6:id=10:ba=2620:a1=2199:a2=0007:tm=2965:hu=6535:at=1012

{“ed”: 8100D8F6, “id”:10, “ba”:2.620, “a1”:2.119, “a2”:0.007, “tm”: 29.65, “hu”:65.35”, “at”: 1012}

# データの形式(詳細)

子機より得られる計測データ：
:ed=8100D8F6:id=10:ba=2620:a1=2199:a2=0007:tm=2965:hu=6535:at=1012

ed: 子機の識別番号（MACアドレス下8桁）
id: 子機のID番号子機の電池電圧
ba: 電池の電圧（mV）0-2400mV
a1: 磁気リードスイッチの電圧(mV) 0-2400mV
a2: 照度センサの電圧（mV）0-2400mV
tm: 温度(℃) -40℃～85℃
hu: 湿度(%RH) 0%～100%
at: 大気圧(hPa) 300hPa～1100hPa

# データ形式の変換（Example1101P）

例）ed=8100D8F6:id=10¥n（改行¥nは制御コードであり表示されない）

① まず，文字列を改行コードまで全て読み込む
```
String myString = myPort.readStringUntil('¥n');
```

② 文字列を':'で分離する
```
String Data1[] = split(myString, ':');
```
このとき，Data1[0] = “ed=8100D8F6”, Data1[1]=“id=10”となる

③ さらに文字列を分離する
```
String ed[] = split(Data1[0], '=');
String id[] = split(Data1[0], '=');
```
このとき，ed[0] = “ed”, ed[1]=“8100D8F6”, id[0]=“id”,id[1]=“10”となる

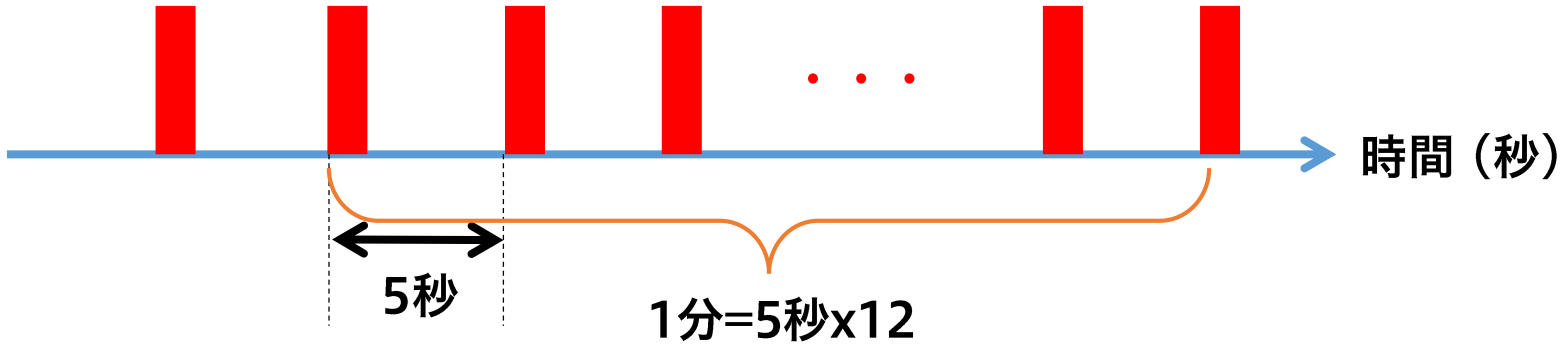
④ データの確認して，正しければ文字列を数値等に変換する
```
if (ed[0].equals(“ed”)) { ED=ed[1]; }
if (id[0].equals(“id”)) { ID=int(id[1]); }
```

# 1分間の平均(Example1102P)

```
float ave = 0; // 平均値
int n = 0; // 計測回数

void draw() {
  if (received == 1) {
    ave += TM;
    n++;
    if (n >= 12) {
      ave /= 12.0;
      print("average(min):");
      println(floatFormat(ave,2));
      ave = 0.0;
      n = 0;
    }
    received = 0;
  }
}
```

- 5秒間隔で計測結果(TM)を取得
- 12回のサンプリングで1分間のデータ




無線子機の動作：

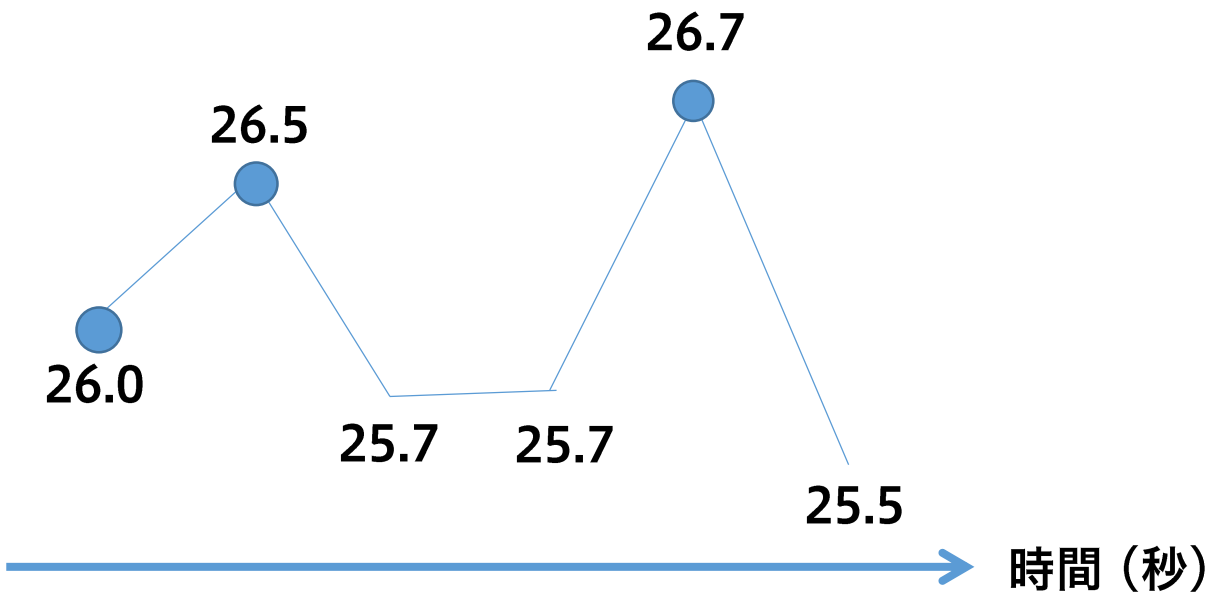
- 5秒間スリープをして待機(低消費電力)
- 5秒毎に起床して，計測データの送信

# 最高気温(Example1103P)

```
float temp_max = -40; // 最低気温をセット

void draw() {
  if (received == 1) { // 5秒毎にデータ受信
    float tm = TM;
    if (tm > temp_max) { // 最大値を超えていたら
      temp_max = tm;
      print("temperature current(max):");
      println(floatFormat(temp_max,2));
    }
    received = 0;
  }
}
```

- 5秒間隔で計測結果(TM)を取得
- 最大値temp_maxを記憶し，現在値と比較




## if分の比較動作

TM > temp_max:

26.0>-40　25.7 > 26.5　25.7>26.5　25.5 > 26.7

26.5>26.0　26.7 > 26.5

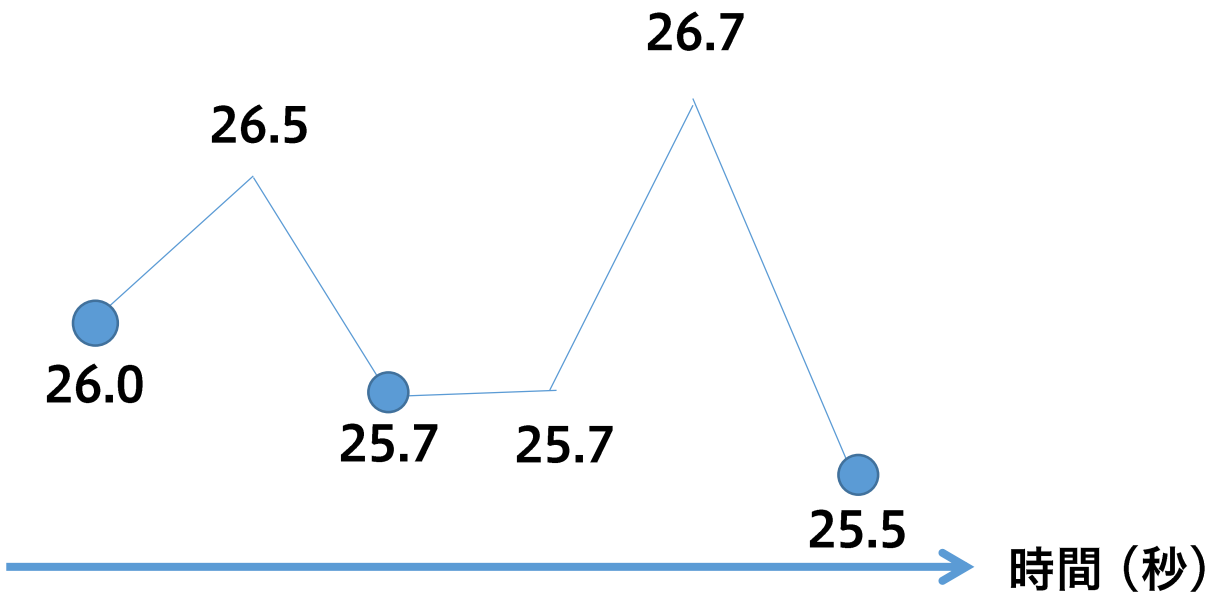
temp_maxの値:　26.0　26.5　26.5　26.5　26.7　26.7

# 最低気温(Example1105P)

```
float temp_min = 80; // 最高気温をセット

void draw() {
  if (received == 1) { // 5秒毎にデータ受信
    float tm = TM;
    if (tm < temp_min) { // 最小値を下回ったら
      temp_min = tm;
      print("temperature current(min):");
      println(temp_min);
    }
    received = 0;
  }
}
```

- 5秒間隔で計測結果(TM)を取得
- 最小値temp_minを記憶し，現在値と比較




## if分の比較動作

TM > temp_min:

26.0<80　25.7 < 26.0　25.7<25.7　26.7 < 25.5

26.5<26.0　26.7 > 25.7

temp_minの値:　26.0　26.0　25.7　25.7　25.7　25.5

# 計測結果のファイルへの書き込み(Example1106P)

- 計測したデータをCSV形式で保存
- CSV形式：’,’で区切られたデータ系列
  - <data1>,<data2>,<data3>,･･･
- プログラムを実行すると，201611041758.csvのような形式で保存される
  - Temperature, Humidity, Altitude
  - 26.2,62.3,1030
  - ・・・
- エクセルなどの表計算ソフトで開いてグラフを描画できる