

# AITCシニア技術者勉強会 第三回

## 高度なセンサを使ってみよう 温度、加速度

2017年01月21日

先端IT活用推進コンソーシアム

クラウド・テクノロジー活用部会 リーダー

アドソル日進株式会社 荒本道隆

# 本日のゴール

- **第一回**

- アナログ入力: 照度センサ
- デジタル出力: LED
- デジタル入力: ボタン

前回、消化不良だった人は、再度チャレンジ

- **第二回**

- アナログ入力: マイク、距離センサ
- 高度なデジタル出力: フルカラーLED
- デジタル入力: なし

センサ購入時には、  
▪ Arduinoの事例  
▪ 他部品が必要か？  
を必ずチェック

- **第三回**

- 高度なアナログ入力: 加速度センサ
- 高度なデジタル出力: なし
- 高度なデジタル入力: 温度センサ

様々な入出力ができるようになったはず

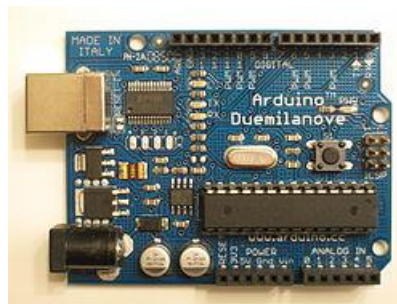
# 前回までの復習

前回参加者は、P32までを自習  
もしくは、持ち込んだセンサを試す

# Arduino とは

Arduino はスタンドアロン型のインタラクティブデバイス開発だけでなく、ホストコンピュータ上のソフトウェア（例えば、[Adobe Flash](#)、[Processing](#)、[Max/MSP](#)、[Pure Data](#)、[SuperCollider](#)）で制御することもできる。[オープンソースハードウェア](#)でありハードウェア設計情報の[EAGLE](#)ファイルは無料で公開されており、組み立て済みの基板を購入することもできるほか、誰でも自分の手でArduinoを組み立てることができる。

Arduinoプロジェクトは2005年に[イタリア](#)で始まり、当時入手可能だった他の学生向けのロボット製造用コントロールデバイスよりも安価なプロトタイピング・システムを製造することを目的にスタートした。設計グループは多くの競合製品よりも遥かに安価で簡単に使用できるプラットフォームの開発に成功した。Arduinoボードは、[2008年10月](#)までに5万ユニット以上<sup>[3]</sup>が、[2011年2月](#)で約15万台<sup>[4]</sup>販売されている。Arduinoプロジェクトは2006年度の[アルス・エレクトロニカ賞](#)で名誉言及を受けている。<sup>[5][6][7]</sup>



ウィキペディアより

# Arduinoの特徴

- **アナログ・デジタル**の入出力が複数ある
  - そこにセンサやリレーを簡単に接続できる
- 豊富なシールド
  - イーサネット, GPS, LCD, モーター制御
  - <http://ideahack.me/article/147>
- センサ以外は、使い慣れた技術・用語が多い
  - USB, シリアル, HTTP, TCP/IP
- 取り扱いが容易
  - 不器用な私でも何とかあった
    - 無線シールドの半田付けに失敗し、3つほど捨てたけど
  - そんなに高くない
    - 壊してしまっても、大人なら平気

色々と追加すると、Raspberry PIの方が安上がりな場合も

# 機器購入時の注意事項

- 無線LAN, Bluetoothは、技適が付いているか？
  - 海外の無線シールドには、技適が付いていない
  - 技適が付いていないものを使うデメリット
    - 発表時に、構成を詳しく言えない
    - **電波法違反** = 「1年以下の懲役又は100万円以下の罰金刑に処せられる」
  - 有線→無線変換が簡単
- 配置時
  - 電源の確保
    - スマホの充電で使うUSBアダプタが大活躍
    - センサだけなら、電池でも結構持つ
      - 無線LANを電池で使いたいなら、省電力のものを選択

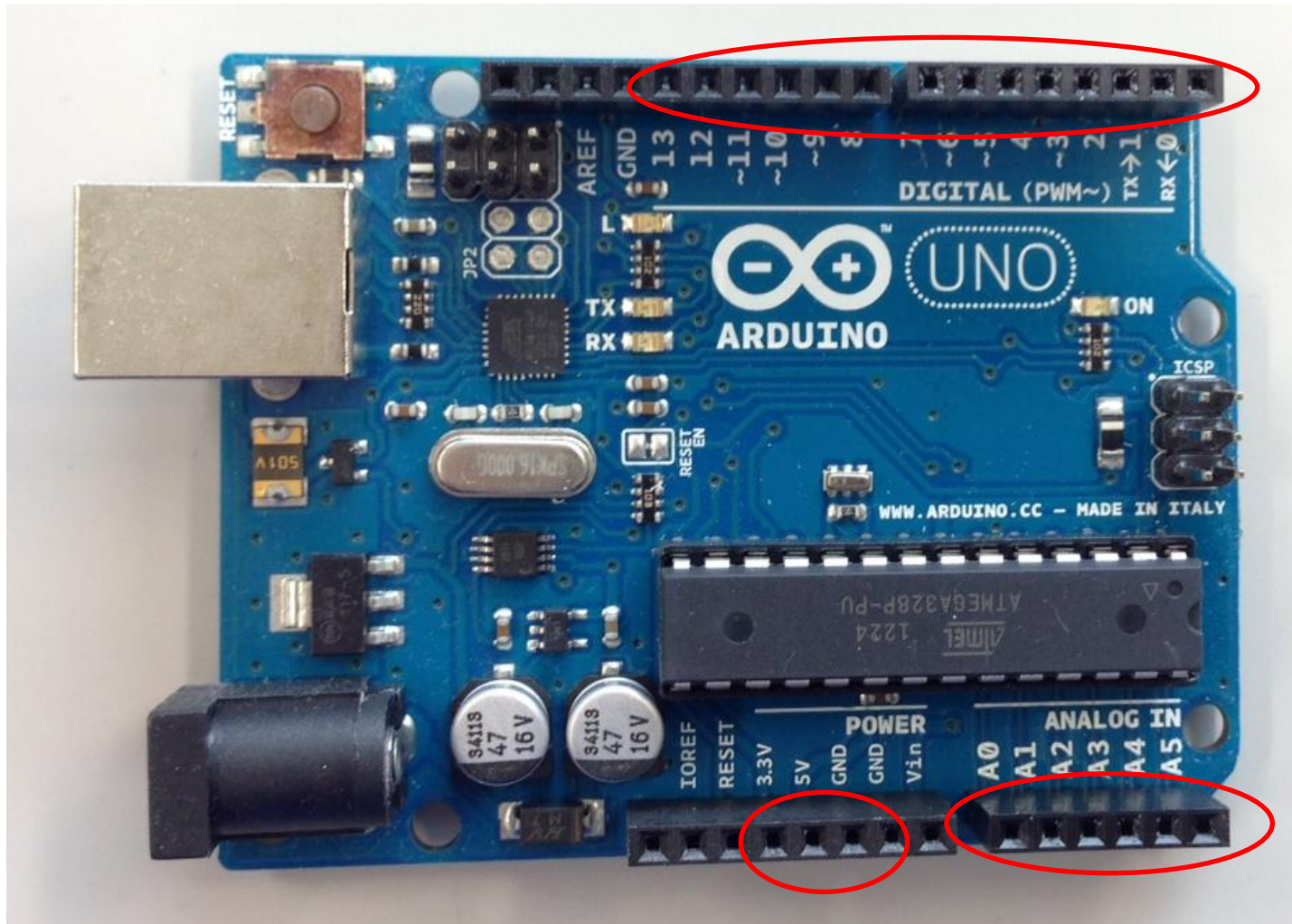


Amazon「PLANEX 充電万能  
2ポートUSB充電器 ホワイト」  
¥1,002-



# Arduinoの概要

デジタル入出力(プログラムで切り替え)

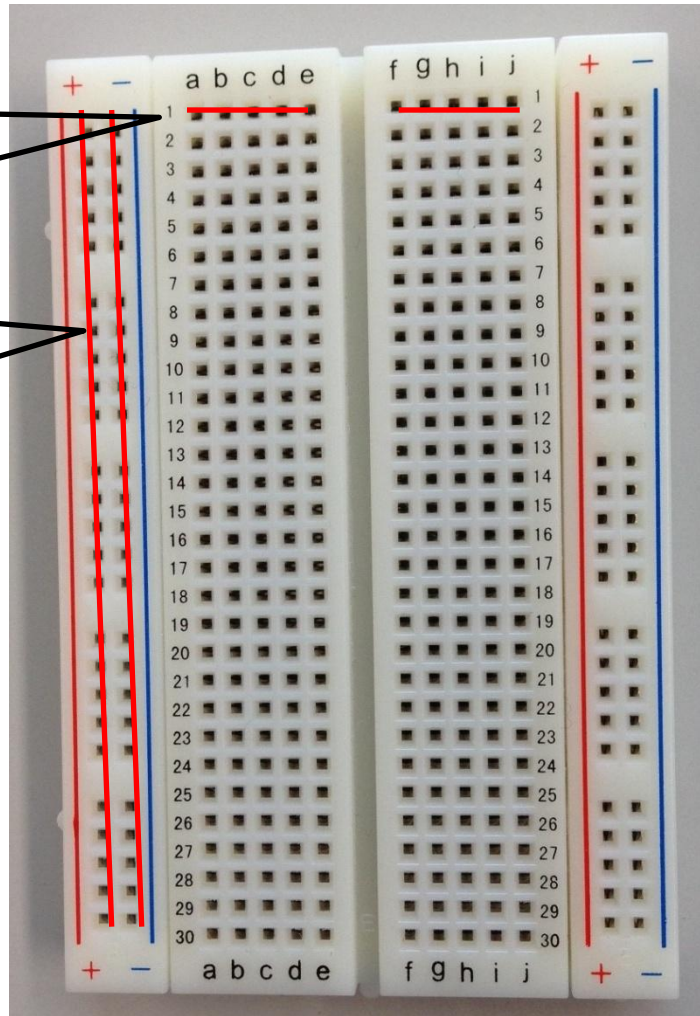


出力にすると  
5V, 40mA

電源

アナログ入力(0~1023の範囲)

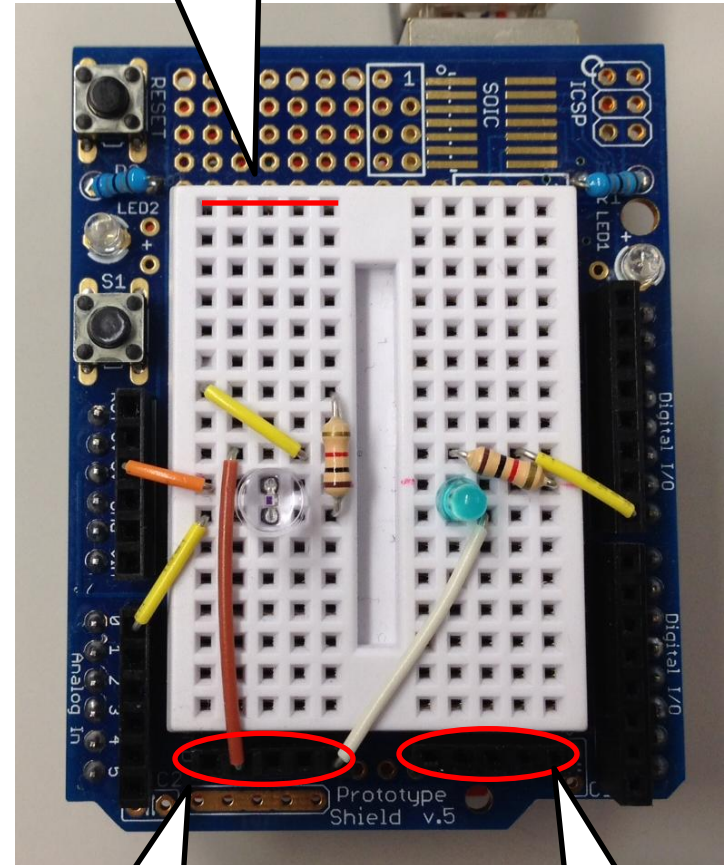
# ブレッドボードの概要



横につながっている  
a~e, f~j

+-だけ縦につながっている

横につながっている

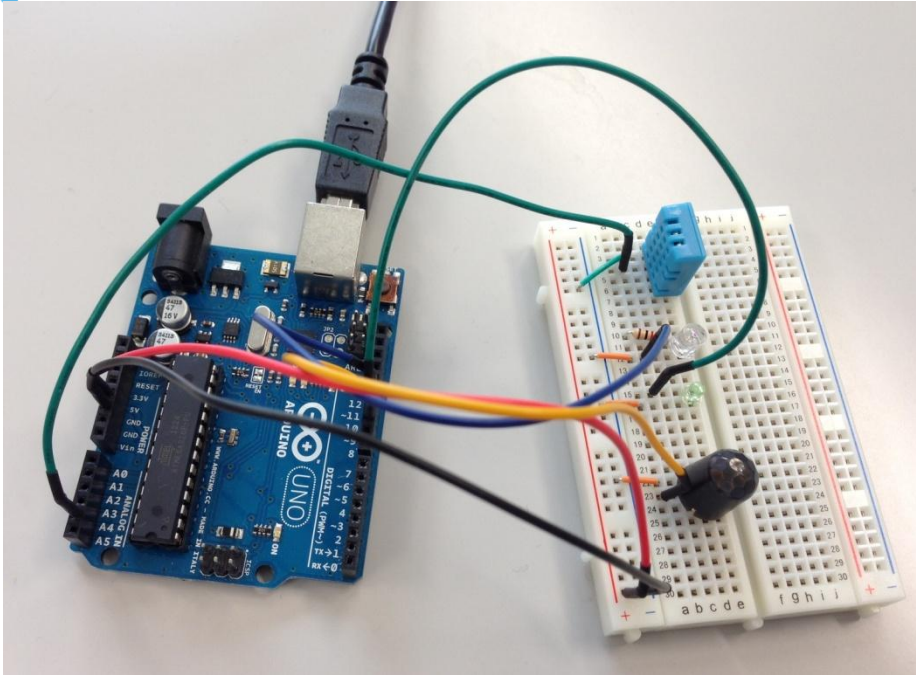


GND

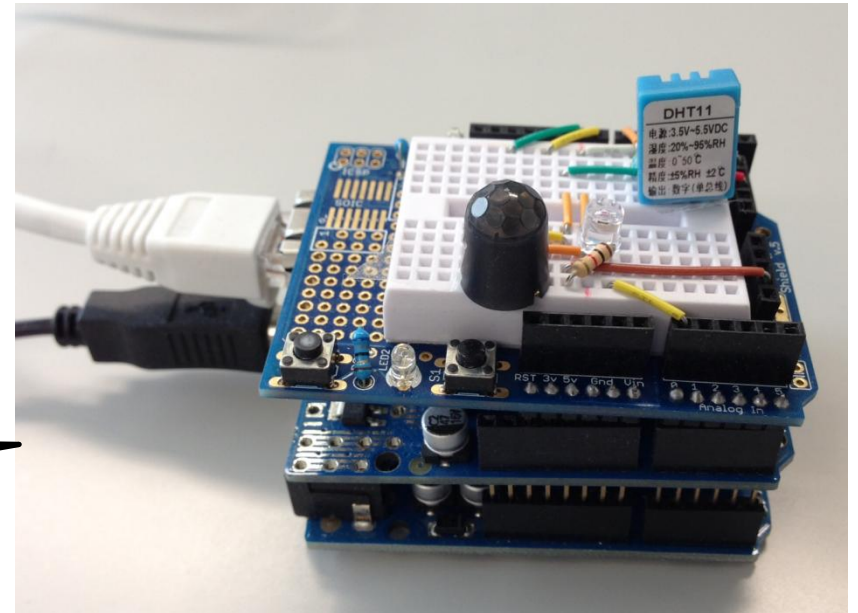
5V



# プロトタイピング



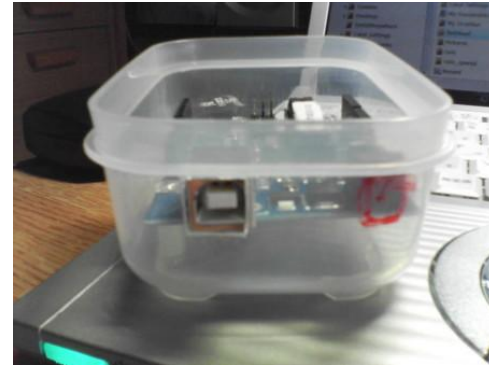
開発時



テスト配置時

# 注意事項

- 回路変更時には、必ず電源を抜く
- 抵抗値の計算について
  - よく分からなければ、計算用サイトを利用
    - <http://diy.tommy-bright.com/>
- Arduinoの電流量は貧弱
  - 5V, 40mA
    - 比較例: 単三電池は1.5V, 100mA
  - サーボモータなど大電流が必要な物は、別電源が必要
- 24時間運転する場合は
  - ちゃんとケースに入れましょう

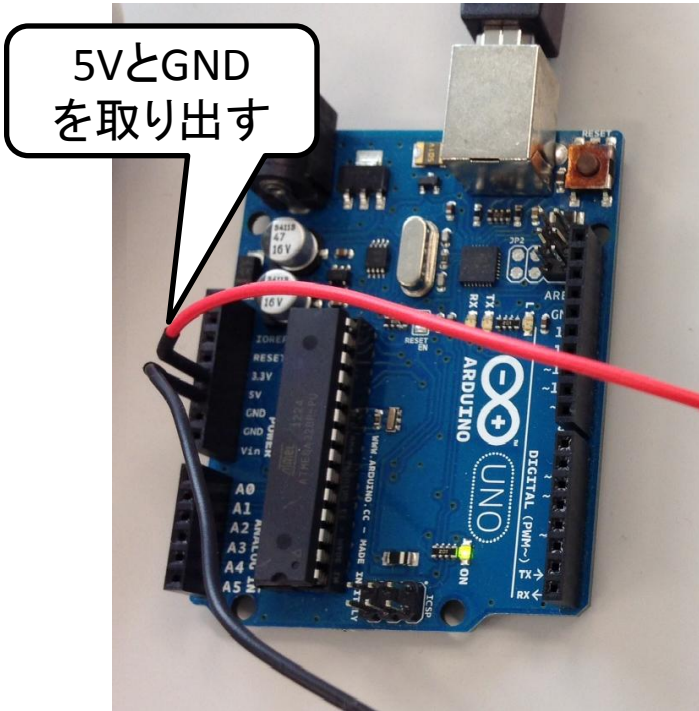
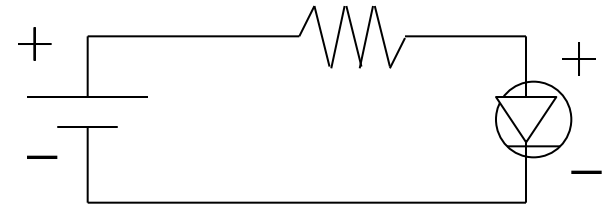


<http://d.hatena.ne.jp/koki-h/comment/20090407/1239090406>

- ダウンロード
  - <http://arduino.cc/en/main/software>
- Windows
  - 「Windows ZIP file」をダウンロードして、解凍
  - drivers¥arduino.inf を右クリックして「インストール」
  - Arduino を接続
  - arduino.exe でIDEを起動
  - メニューの「ツール」から
    - →「シリアルポート」→「COM3」(PCによって違う)を選択
    - →「マイコンボード」→「Arduino Uno」を選択
- Mac
  - 「Mac OS X」をダウンロードして、解凍
  - Arduino を接続
  - arduino でIDEを起動
  - メニューの「ツール」から
    - →「シリアルポート」→「/dev/tty.usbmodem3d11」を選択
    - →「マイコンボード」→「Arduino Uno」を選択

# 練習問題

- LEDを点ける
  - 状況が目視できるようになる

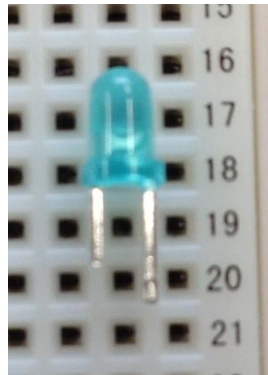


5VとGND  
を取り出す

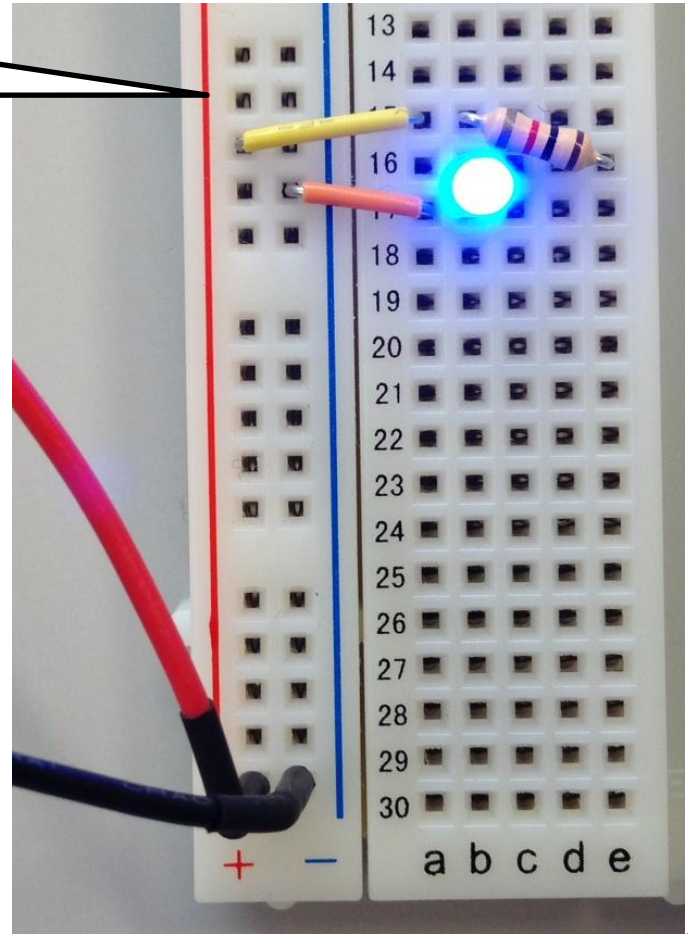
5V側に抵抗  
を入れる

LEDの仕様で  
抵抗値は違う

抵抗無しだと  
焼き切れる事も



足の長い方が+  
挿し易いように、カット済み

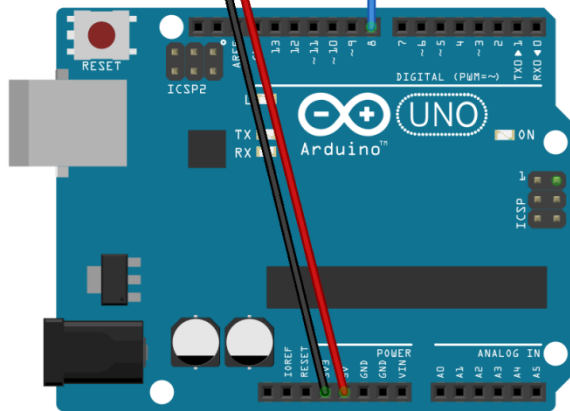
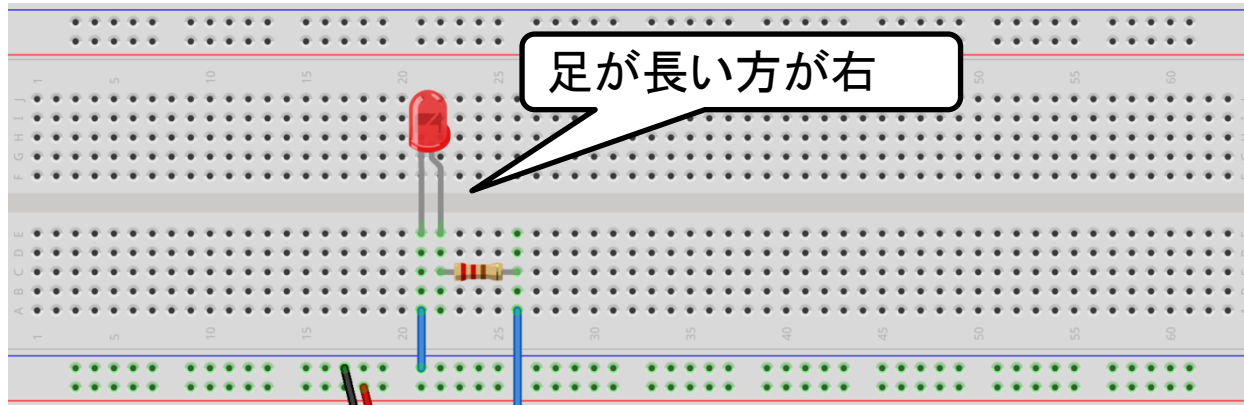
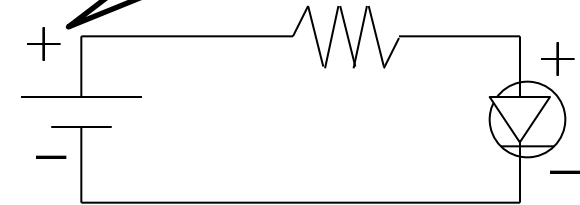




# ステップ1-1

デジタルの8番

- LEDを1秒ごとに点滅させる
  - デジタルの8番をLEDの+に接続
    - さっきまでの5Vの線は外す



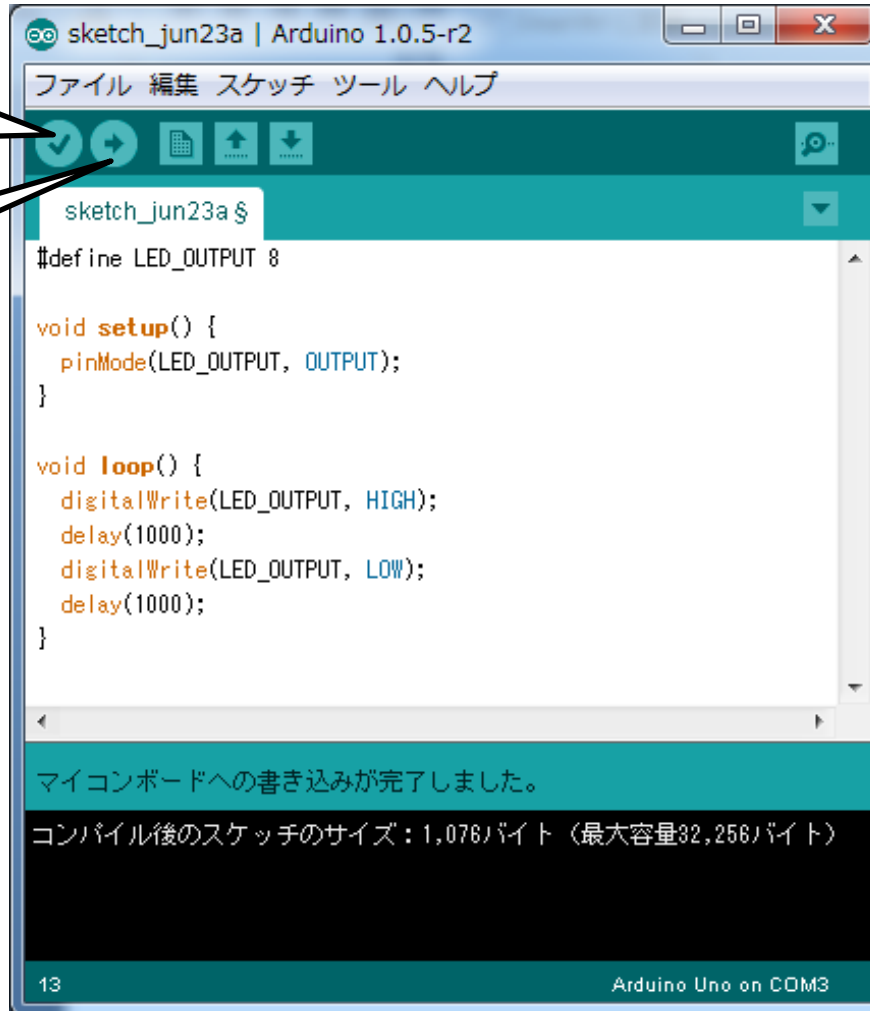
# ステップ1-2

- LEDを1秒ごとに点滅させる  
- 次にプログラムを作成

1. コンパイル

2. 書き込み

書き込んだら、  
自動で実行開始



```

sketch_jun23a | Arduino 1.0.5-r2
ファイル 編集 スケッチ ツール ヘルプ
sketch_jun23a $
#define LED_OUTPUT 8

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

void loop() {
  digitalWrite(LED_OUTPUT, HIGH);
  delay(1000);
  digitalWrite(LED_OUTPUT, LOW);
  delay(1000);
}

マイコンボードへの書き込みが完了しました。
コンパイル後のスケッチのサイズ: 1,076バイト (最大容量32,256バイト)
13 Arduino Uno on COM3
  
```

おまけ  
高速に点灯と消灯を  
繰り返せば、明るさ  
を調節できる

ちょっと発展形

```

#define LED_OUTPUT 8

boolean led = false;

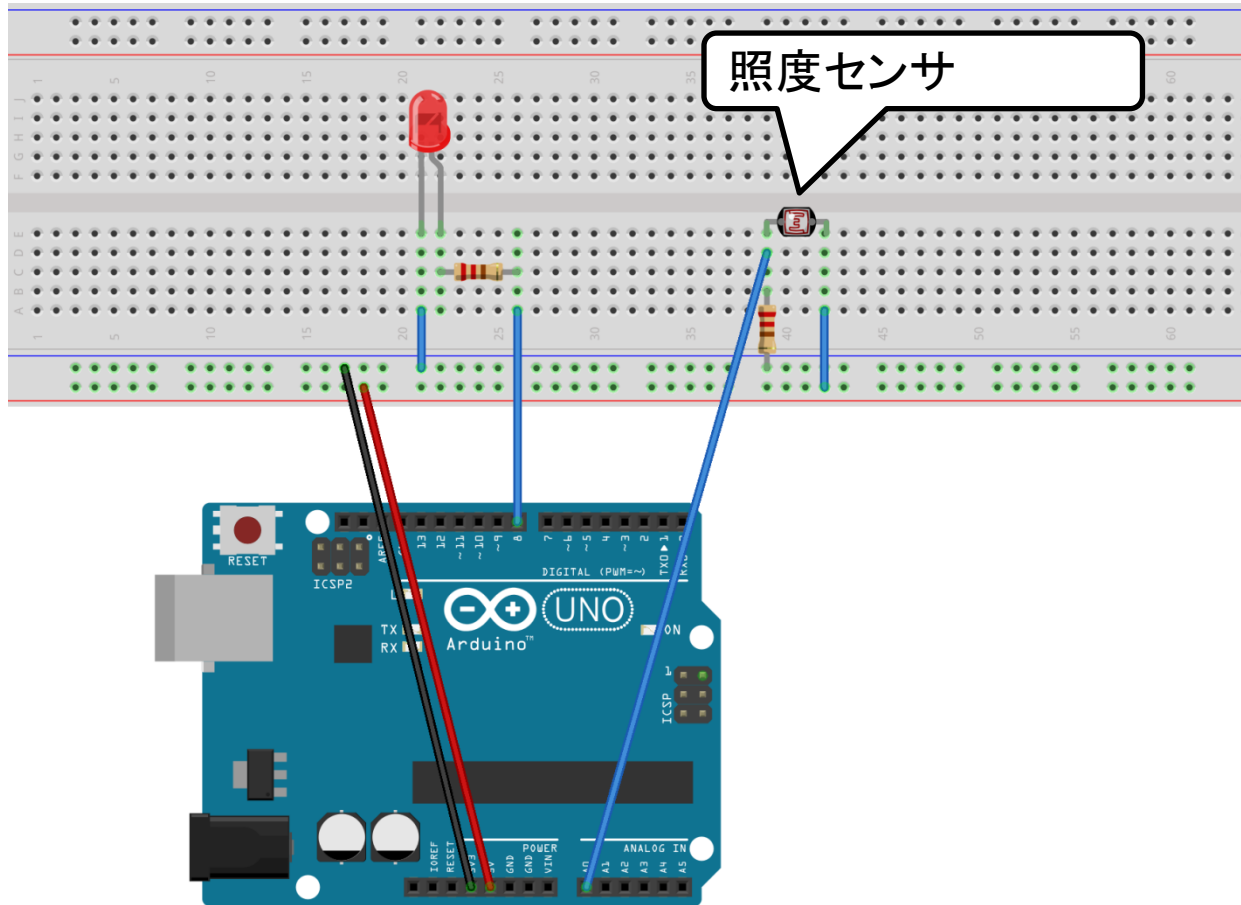
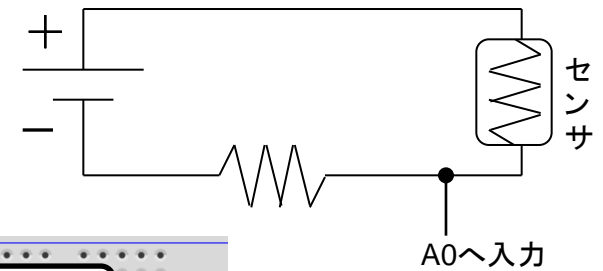
void setup() {
  pinMode(LED_OUTPUT, OUTPUT);
}

void loop() {
  led = !led; // 反転
  digitalWrite(LED_OUTPUT, led);
  delay(1000);
}
  
```

sample1\_1.txt

# ステップ2-1

- 照度センサの値をPCで参照
  - 照度センサをアナログの0番に入力
    - 向きが重要なものもある



# ステップ2-2

- 照度センサの値をPCで参照
  - プログラムを作成
  - 実行後は、「ツール」→「シリアルモニタ」で確認

```
#define LED_OUTPUT 8
#define CDS_INPUT 0

boolean led = true;

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

void loop() {
  int val = analogRead(CDS_INPUT);

  Serial.print("CdS :");
  Serial.print(val);
  Serial.println();
  delay(1000);
}
```

**sample1\_2.txt**



# ステップ3

- 「暗くなったら、LEDを灯す」を実現
  - ステップ2のプログラムを改良

```

#define LED_OUTPUT 8
#define CDS_INPUT 0

boolean led = LOW;

void setup() {
  pinMode(LED_OUTPUT, OUTPUT);
  // Serial.begin(9600);
}

void loop() {
  int val = analogRead(CDS_INPUT);
  if (val < 400) { // 暗ければ
    led = HIGH; // 点ける
  } else { // そうでなければ、
    led = LOW; // 消す
  }
  digitalWrite(LED_OUTPUT, led);
  delay(100); // 反応の遅延を減らす
}

```

閾値(400)は、  
場所に合わせて調整

**sample1\_3.txt**

# マイク、距離センサー を使ったLEDの操作

# 距離センサ

- シャープ距離モジュール
  - 測定したい距離によって、数種類ある
    - 10～80cm、20～150cm、1～5.5m
  - 各ピンの説明
    - 白色：距離出力 → アナログ0番へ接続して、LED操作
    - 赤色：クランド
    - 黒色：電源入植 (DC5V)

抵抗は不要

色が常識と違う！！

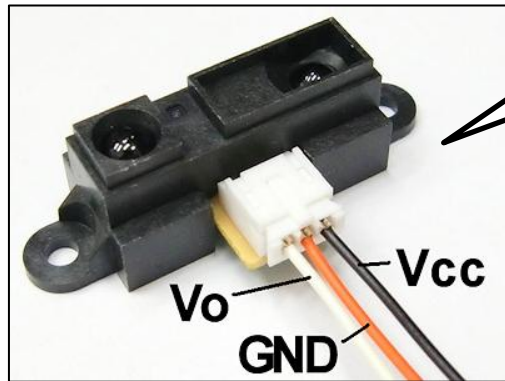
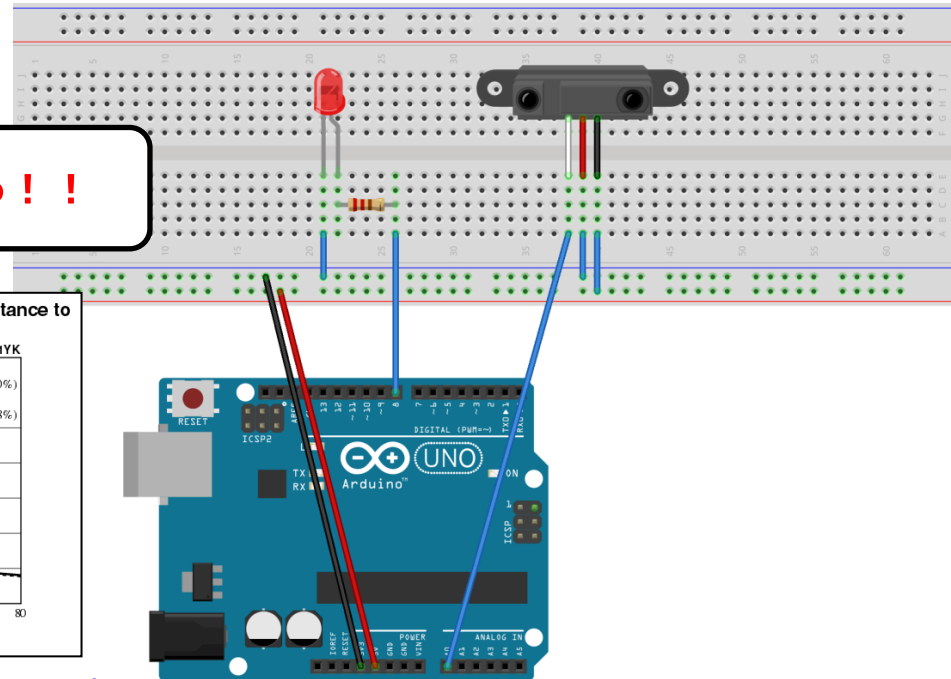
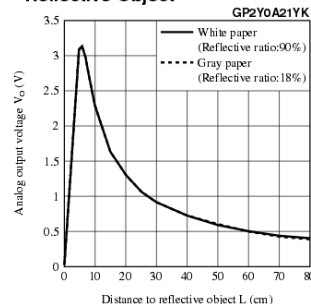


Fig.5 Analog Output Voltage vs. Distance to Reflective Object

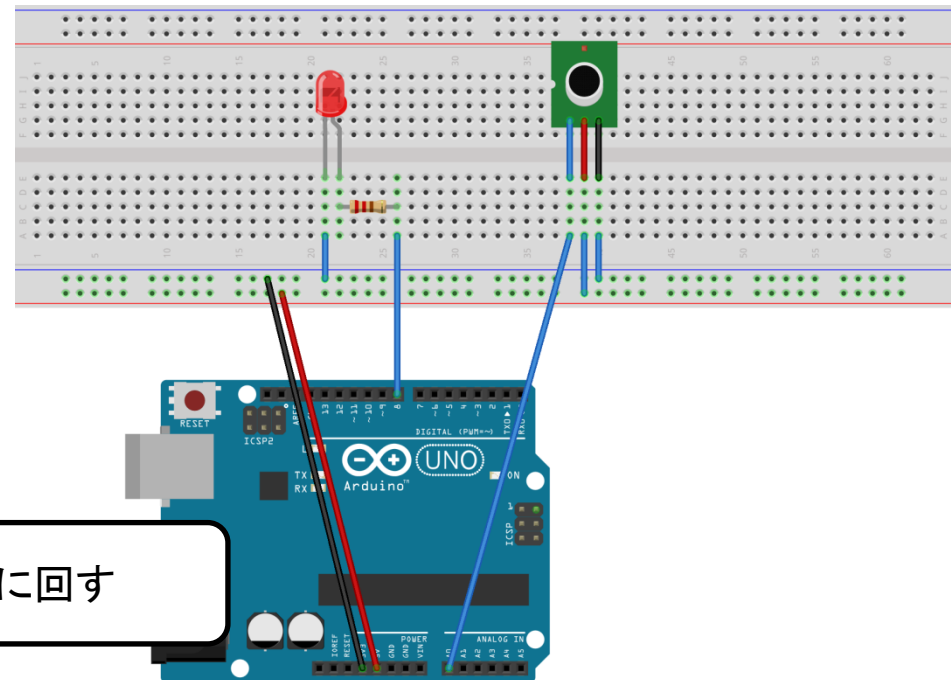
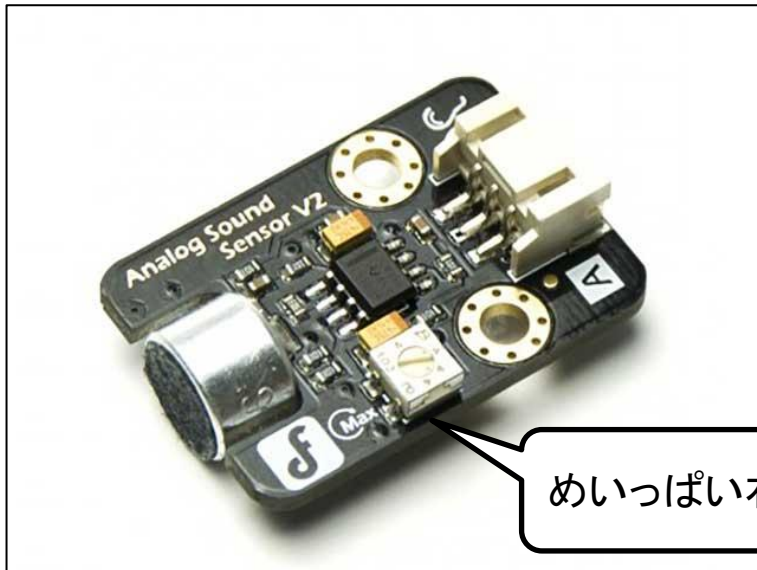


<http://akizukidenshi.com/catalog/g/gi-02551/>

# マイク

- アナログサウンドセンサモジュール
  - アンプが実装されているので、取り扱いが簡単
    - 注意: アンプが無いモノは扱いが難しいので注意
  - 各ピンの説明
    - 青色: 音声出力 → **アナログ0番へ接続して、LED操作**
    - 赤色: 電源入力(DC5V)
    - 黒色: グランド

抵抗は不要



<http://akizukidenshi.com/catalog/g/gM-07038/>



# マイク用の工夫

- 音に反応したら、1秒間、LEDを点灯させる
  - 課題：逆にして、音に反応して消灯させる

```

void setup() {
  pinMode(8, OUTPUT);
  Serial.begin(9600);
}

int count = 0;
void loop() {
  int val = analogRead(0);
  Serial.println(val);
  if (val > 10) { // 一定以上の音なら
    count = 100; // LEDを点灯させておく時間
  }
  if (count > 0) {
    digitalWrite(8, HIGH); // 点灯
    count--;
  } else {
    digitalWrite(8, LOW); // 消灯
  }
  delay(10);
}

```

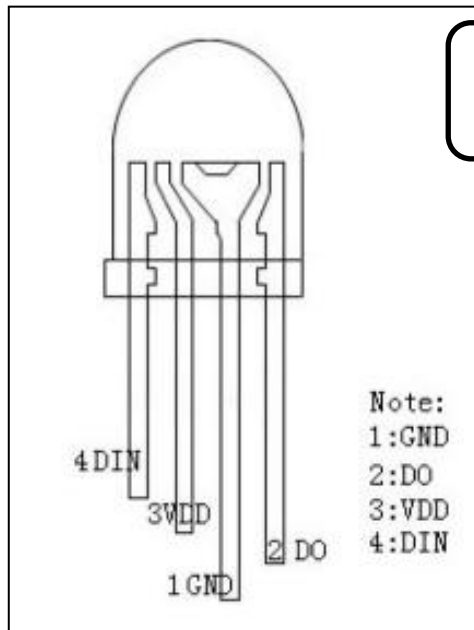
閾値(10)は、調整

**sample1\_4.txt**

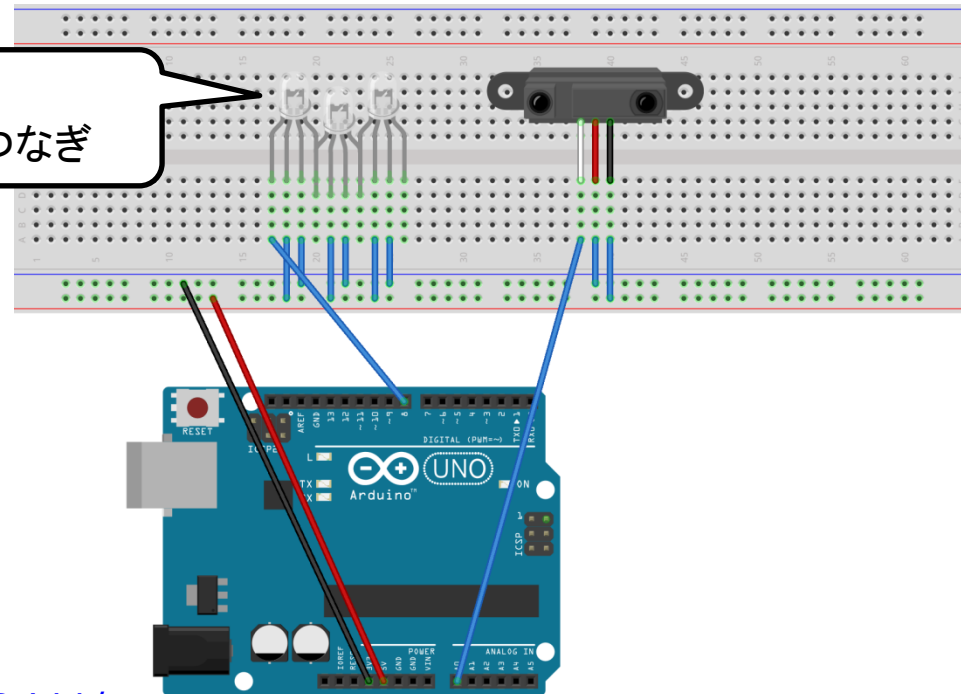
# フルカラーLEDを使った 色の作成

# フルカラーLED: 1

- マイコン内蔵RGB LED
  - 様々な形状のものがあり、RGB値で好きな色を作れる
- 複数を数珠つなぎにできる
  - 沢山のLEDを使う場合、**5VとGNDを別電源から取る**
  - Arduinoの電力が足りなくなると、動作が不安定になる



短い脚を左にし、  
DOとDINを数珠つなぎ

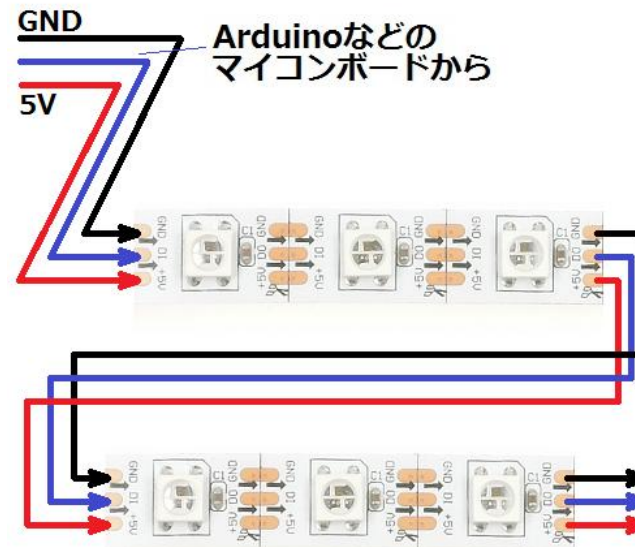


<http://akizukidenshi.com/catalog/g/gl-08411/>

# フルカラーLED: 2

- フルカラーシリアルLEDテープ
  - 接続がとても楽
    - ワニ口クリップを使う時は、ショートしないように注意
  - 1m版(3.18A)を使う時は、別電源から取る
  - マイコン内蔵RGB LEDと同じプログラムで動作
    - プログラムの修正点

Adafruit\_NeoPixel pixels = Adafruit\_NeoPixel(NUMPIXELS, PIN, **NEO\_GRB** + NEO\_KHZ800);



<https://www.switch-science.com/catalog/1400/>

次のモジュールへ



# フルカラーLEDを使う準備

- ライブラリをダウンロード
  - 使用するライブラリは、LEDによって違うので注意
  - ダウンロード先
    - [https://github.com/adafruit/Adafruit\\_NeoPixel](https://github.com/adafruit/Adafruit_NeoPixel)
    - 「Clone or download」→「Download ZIP」
- 開発環境に追加
  - 「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」
  - 『**Adafruit\_NeoPixel-master.zip**』を指定
- コントロールは、デジタル8番ピンを使用  
#define PIN           8

# フルカラーLEDを点灯ー1

- 0.5秒間隔に、点灯、消灯を繰り返す
  - 課題：RGBの組み合わせで、好きな色を作成

```
#include <Adafruit_NeoPixel.h>

#define PIN      8
#define NUMPIXELS  5

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_RGB + NEO_KHZ800);

void setup() {
  Serial.begin(9600);
  Serial.println ("START");
  pinMode(13, OUTPUT); digitalWrite(13, HIGH); // DIGITAL13を5Vとして使用
  pixels.begin(); // This initializes the NeoPixel library.
}

void loop() {
  // 点灯
  pixels.setPixelColor(0, pixels.Color(150, 0, 0)); // RED
  pixels.setPixelColor(1, pixels.Color( 0, 150, 0)); // GREEN
  pixels.show(); // 反映
  delay(500);   // ちょっと間をあける

  pixels.clear(); // 消灯
  pixels.show(); // 反映
  delay(500);   // ちょっと間をあける
}
```

- 色をランダムに変える

```
#include <Adafruit_NeoPixel.h>

#define PIN      8
#define NUMPIXELS 5

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_RGB +
NEO_KHZ800);

void setup() {
  Serial.begin(9600);
  Serial.println ("START");
  pinMode(13, OUTPUT); digitalWrite(13, HIGH); // DIGITAL13を5Vとして使用
  pixels.begin(); // This initializes the NeoPixel library.
}

void loop() {
  // ランダムで点灯
  for (int i = 0; i < NUMPIXELS; i++) {
    int c = random(1,8); // ランダムで1-7を発生させる
    pixels.setPixelColor(i, pixels.Color((c&1)*150, (c&2)*150, (c&4)*150));
  }
  pixels.show();
  delay(500); // ちょっと間をあける
}
```

- 炎を表現してみる

```
#include <Adafruit_NeoPixel.h>

#define PIN      8
#define NUMPIXELS  5

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_RGB +
NEO_KHZ800);

void setup() {
  Serial.begin(9600);
  Serial.println ("START");
  pinMode(13, OUTPUT); digitalWrite(13, HIGH); // DIGITAL13を5Vとして使用
  pixels.begin(); // This initializes the NeoPixel library.
}

void loop() {
  // 炎っぽさを表現
  for (int i = 0; i < NUMPIXELS; i++) {
    int c = random(10,100); // ランダムで10-99を発生させる
    pixels.setPixelColor(i, pixels.Color(c, 0, 0)); // 赤だけ使用
  }
  pixels.show();
  delay(random(10,100)); // 間隔もランダムで
}
```

# センサー＋フルカラーLED

# こんなモノを作ってみよう

- 距離センサー＋フルカラーLED
  - 距離を色と数で表現
    - 近づくと警告
    - 最適な距離を指示
- マイク＋フルカラーLED
  - 炎を表現。音があると消灯。リセットで復活
  - 音があると点灯。一定時間経過で元の状態に戻る
  - 過去最大音を色と数で表現
    - LEDが複数あれば、直近、10秒前、20秒前、と使い分ける
  - 拡張案: `delay();`の値を短くし、音の反応をよくする



- 過去最大音をLEDで表現

```

setup() までは省略

int max = 0;
void loop() {
  int val = analogRead(0);
  Serial.println (val);
  if (val > max){
    max = val;
  }

  if (max <= 10){
    // 低い
    int c = (max + 1) * 20;
    pixels.setPixelColor(0, pixels.Color(0, 0, c)); // 青
  } else if (max <= 20){
    int c = (max-10+1) * 20;
    pixels.setPixelColor(0, pixels.Color(c, c, 0)); // 黄
  } else {
    int c = (max-20+1) * 20;
    if (c > 255) c = 255;
    pixels.setPixelColor(0, pixels.Color(c, 0, 0)); // 赤
  }
  pixels.show();
  delay(10);
}

```

- イルミネーションを息で消す

```

int count = 0; // マイナスで消灯、プラスで点灯
void loop() {
  int val = analogRead(0);
  Serial.println (val);
  if (val > 10){
    count = -1000;
  }
  if (count < 0){
    pixels.clear(); // 全部消す
    pixels.show();
    delay(5);
    count++;
    return;
  }

  if (count > 100){
    for (int i = 0; i < NUMPIXELS; i++) {
      int c = random(1,8); // ランダムで1-7を発生させる
      pixels.setPixelColor(i, pixels.Color((c&1)*150, (c&2)*150, (c&4)*150));
    }
    pixels.show();
    count = 0;
  }
  delay(5); // 反応を良くするため、小さい値にする
  count++;
}

```

「音で点灯」もやってみよう

「近づいたら点灯」もやってみよう

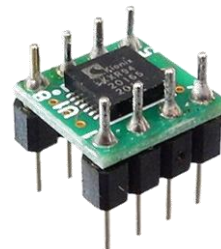
閾値(10)は、調整

# 高度なセンサを使ってみよう 温度、加速度

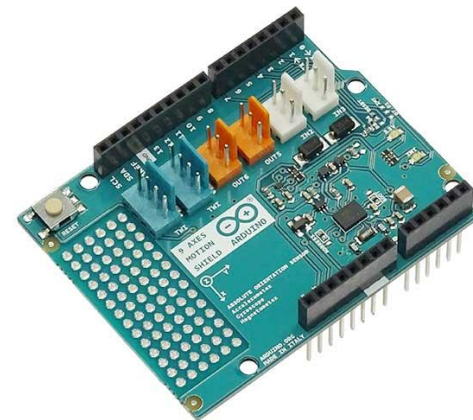
ここからが今日の本番です

# 加速度センサ

- 3軸加速度センサモジュール
  - <http://akizukidenshi.com/catalog/g/gM-05153/>
  - X, Y, Z軸の各加速度をアナログ (Arduinoは0~1023) で取得
  - 安い、簡単、扱い易い

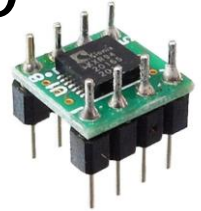
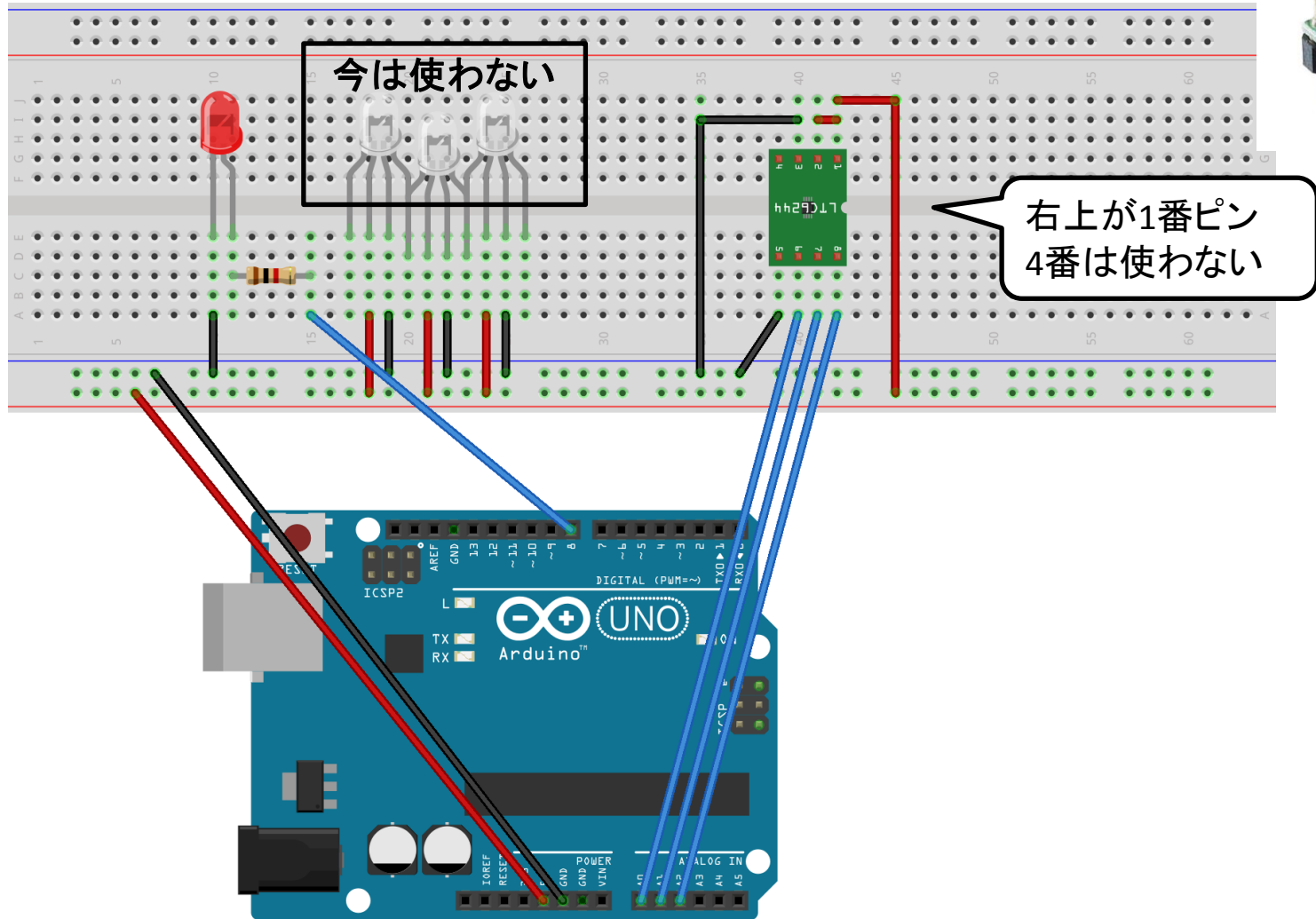


- Arduinoシールドの加速度センサもある
  - <http://akizukidenshi.com/catalog/g/gM-09400/>
  - 14bit精度 (0~ 16,383 )
  - 9軸 + 地磁気
  - Arduinoの上に乗せて、動かしやすい
  - 複雑な計算を内部でやってくれる



# 加速度センサ：配線

- 3軸加速度センサモジュール KXR94-2050



# 加速度センサを使う: 1

- x, y, z の各要素を表示

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

void loop() {
  int x = analogRead(0);
  int y = analogRead(1);
  int z = analogRead(2);

  Serial.println ("x:" + String(x) + " y:" + String(y) + " z:" + String(z));
  delay(100);
}
```

**sample3\_1.txt**

- このセンサの制限事項
  - 精度が低い
  - analogRead()が0.1msecかかるので、x,y,zを同時に読めない



# 加速度センサを使う: 2

- 初期状態からの傾きを表示

```
int startX = 0;
int startY = 0;
int startZ = 0;
void setup() {
  pinMode(8, OUTPUT); // LEDに接続
  pinMode(9, OUTPUT); // LEDに接続
  pinMode(10, OUTPUT); // LEDに接続

  Serial.begin(9600);

  startX = analogRead(0);
  startY = analogRead(1);
  startZ = analogRead(2);
}
```

```
void loop() {
  int x = analogRead(0);
  int y = analogRead(1);
  int z = analogRead(2);

  digitalWrite(8, LOW); // いったん、消す
  digitalWrite(9, LOW); // いったん、消す
  digitalWrite(10, LOW); // いったん、消す

  Serial.print ("sx:" + String(startX) + " ");
  if ((int)(startX/10) > (int)(x/10)){ // 割った数で、感度を調整
    digitalWrite(8, HIGH);
    Serial.print ("> ");
  } else if ((int)(startX/10) == (int)(x/10)){
    digitalWrite(9, HIGH);
    Serial.print ("== ");
  } else if ((int)(startX/10) < (int)(x/10)){
    digitalWrite(10, HIGH);
    Serial.print ("< ");
  }

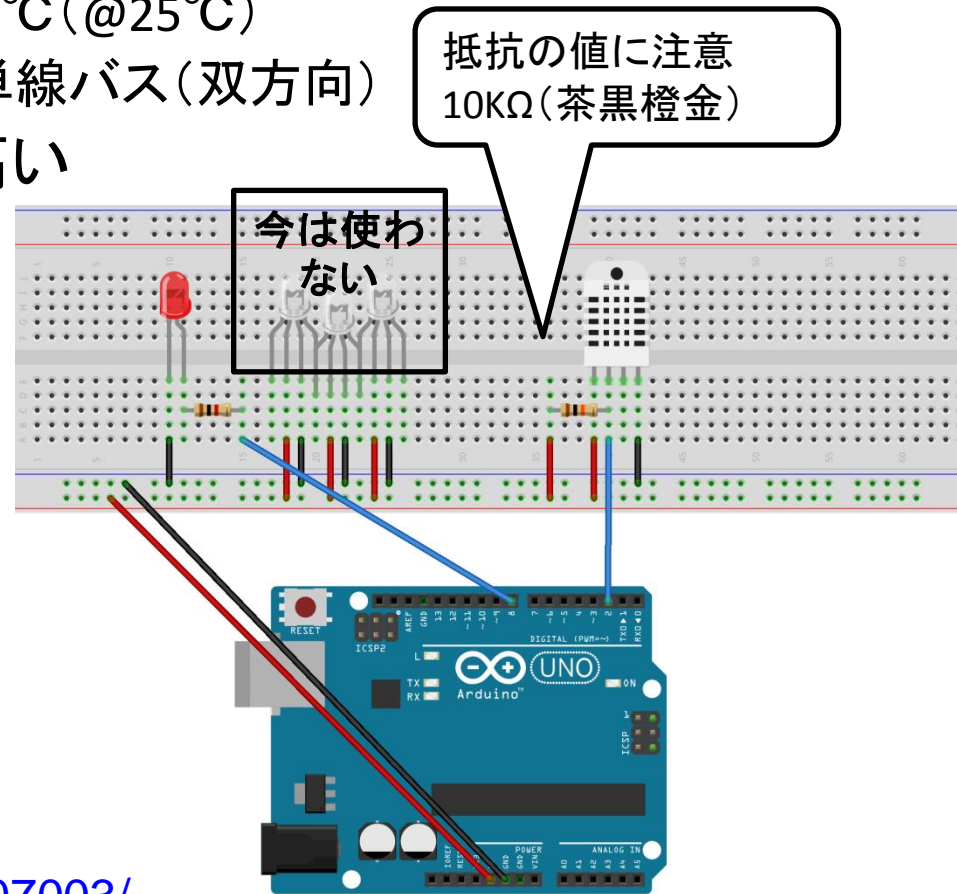
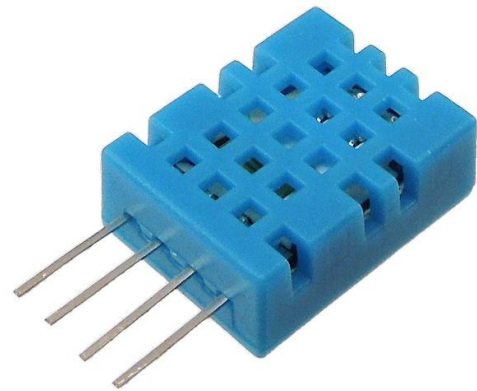
  Serial.println ("x:" + String(x) + " y:" + String(y) + " z:" + String(z));
  delay(100);
}
```

- LEDを3つにする
  - 8,9,10 に接続

**sample3\_2.txt**

# 温湿度センサ

- 温湿度センサ: DHT11
  - サンプルング間隔: 2秒以上
  - 湿度センサ部、精度:  $\pm 5\% \text{ RH} (@25^\circ\text{C})$
  - 温度センサ部、精度:  $\pm 2^\circ\text{C} (@25^\circ\text{C})$
  - シリアル通信部、形式: 単線バス(双方向)
- 高価なものは、精度も高い



<http://akizukidenshi.com/catalog/g/gM-07003/>

# 温湿度センサを使う

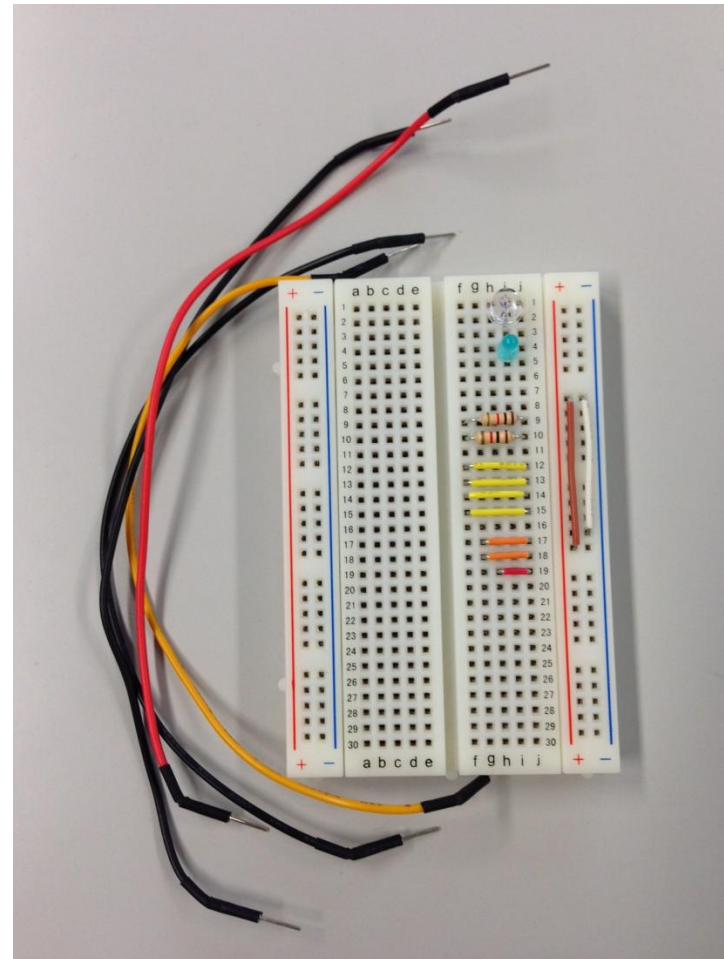
- ライブラリをダウンロード
  - 使用するライブラリは、センサによって違うので注意
  - ダウンロード先
    - <http://playground.arduino.cc/Main/DHT11Lib>
- 開発環境に追加
  - 「スケッチ」→「ライブラリをインクルード」  
→「ZIP形式のライブラリをインストール」  
→『**dht11.zip**』を指定
- サンプルコードを開く
  - 『sample3\_dht11.txt』を開いて、開発環境にコピーする

# こんなモノを作ってみよう

- 前回(第2回)に参加していない人
  - 距離センサ、フルカラーLEDを触ってみよう
- 加速度センサ
  - 傾きをLEDで可視化する
    - x, y, z の傾き具合をLEDで表現
    - フルカラーLEDで傾き度合いを表現(少:青、中:黄、大:赤)
    - ランダム(random(400, 600))で出した傾きを探すゲーム
  - グラスに付けて、グラスを傾けた回数をカウントする
    - アルコール:頻度が多いと警告
    - 飲料水:頻度が少ないと警告
  - 動きを検出し、LEDを点灯したままにする
- 温湿度センサ
  - 乾燥しているとLEDを点灯させる
  - 快適度をフルカラーLEDで表現する(乾燥:黄、暑:赤、湿気:青)

# 後片付け

- 借し出したものを返却してください
- 壊れたかな?と思ったら、言ってください



次回は、サーボモータと  
イーサネットシールドを使います

使ってみたいセンサや  
紹介できるセンサを  
アンケートに書いてください。



<http://aitc.jp>



<https://www.facebook.com/aitc.jp>



ハルミン

AITC非公式イメージキャラクター