

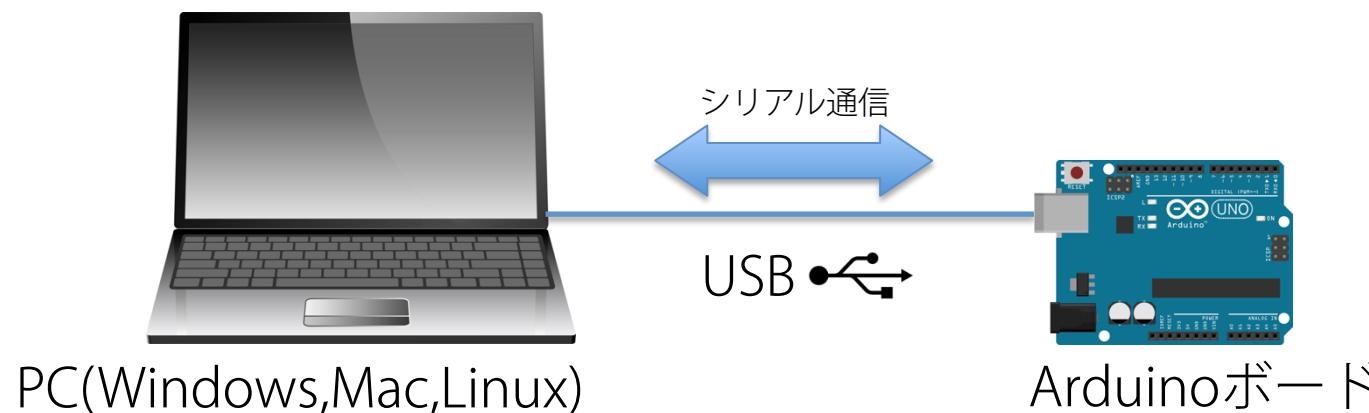
Arduino入門勉強会 #3

【外部PCとの連携】

平成27年7月21日
ソフトピアジャパン ドリーム・コア1F ネクストコア

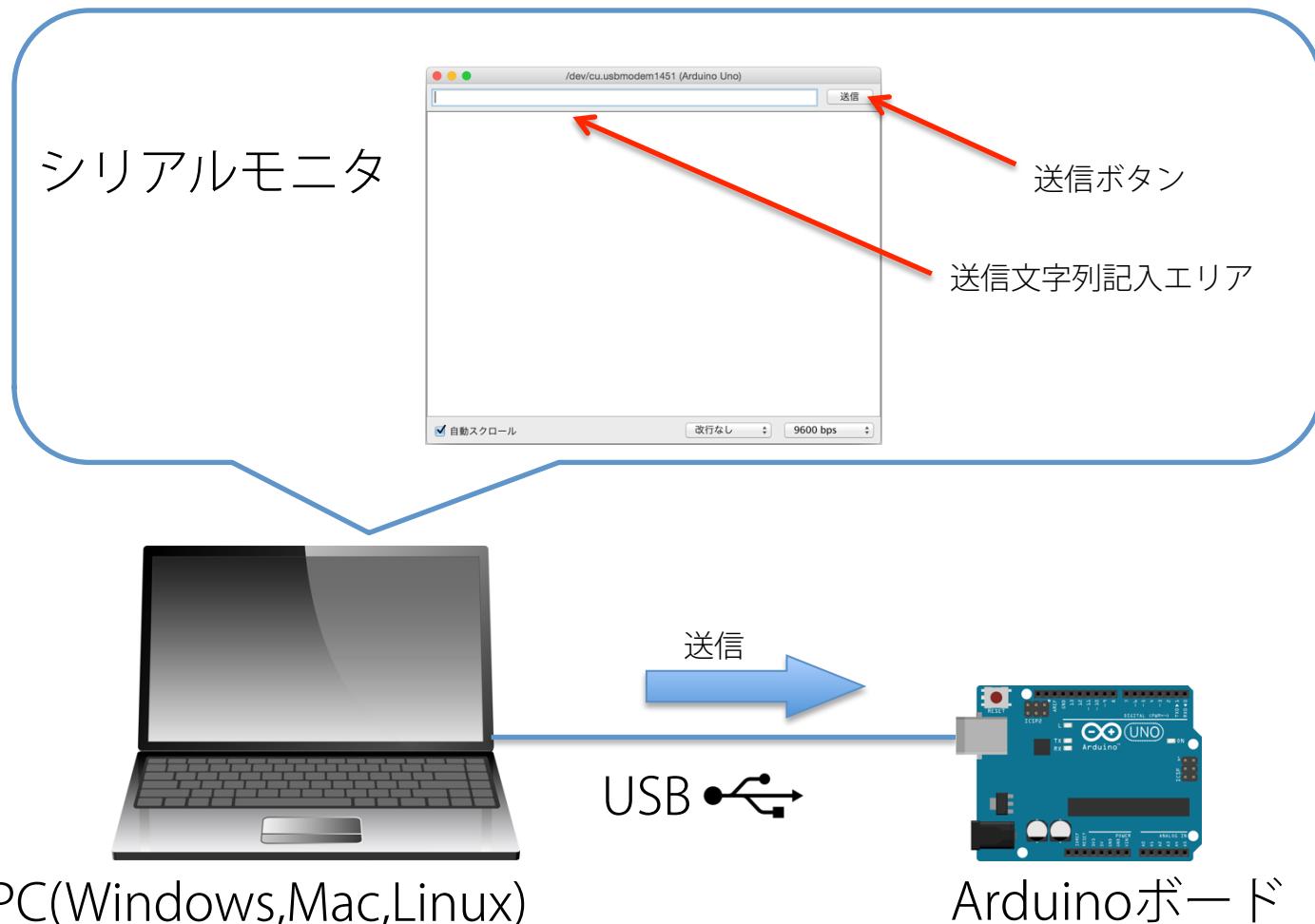
外部PCとの連携：シリアル通信

Arduinoにはシリアル通信機能が備えられており、PCとデータのやり取りを行いながら動作することができます。



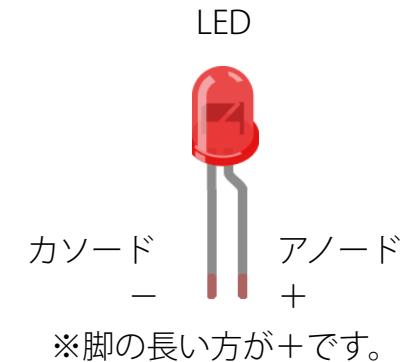
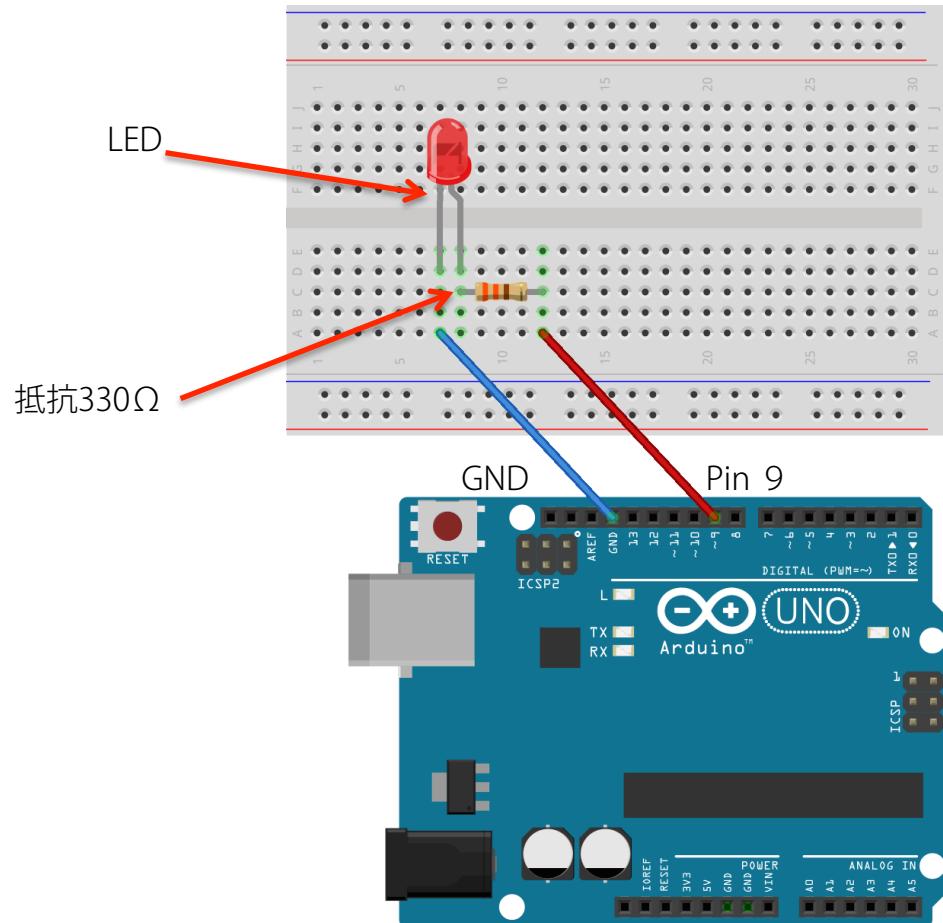
シリアルモニタからの送信

シリアルモニタからのデータ送信をトリガにしてArduinoを動作させます。



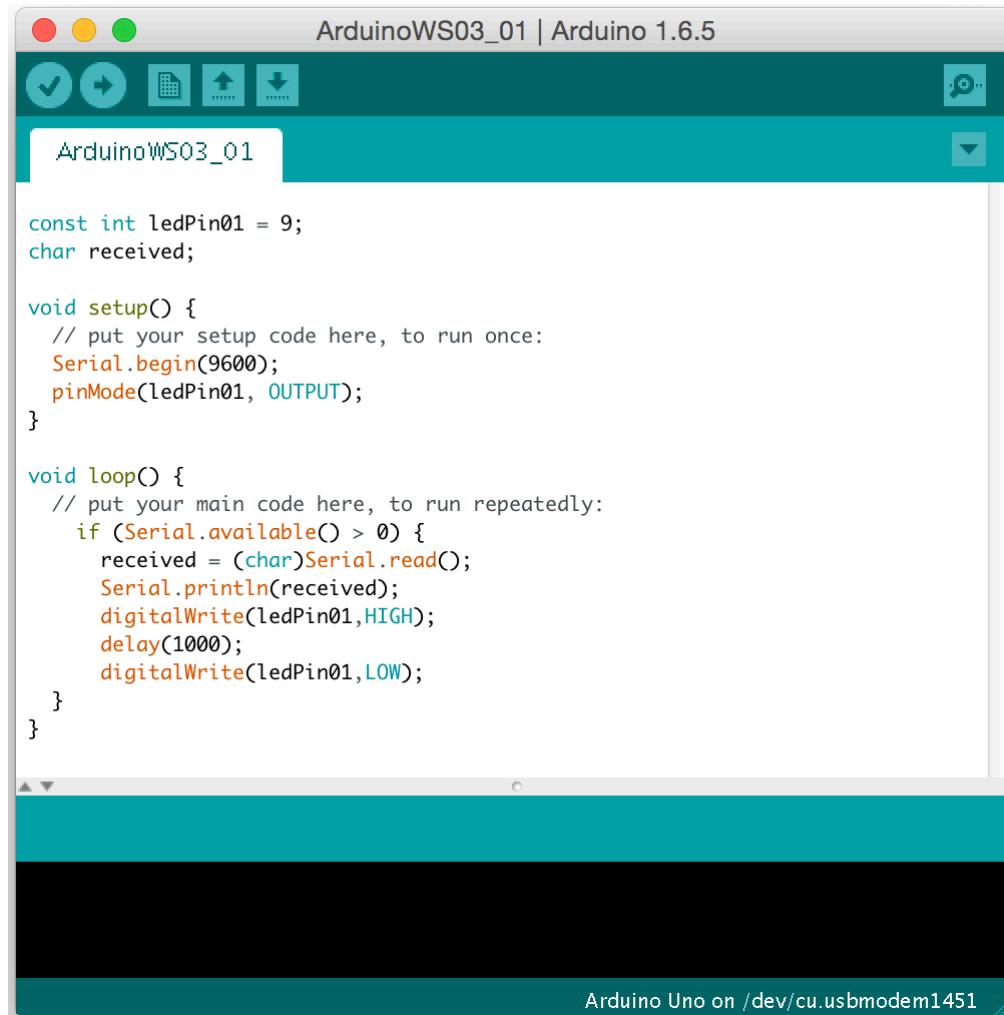
シリアルモニタから送信:回路

シリアルモニタからの受信を確認するために単純なLED点灯回路を作成します。



シリアルモニタから送信：コード

以下のコードを入力して実行します。



The screenshot shows the Arduino IDE interface with the title bar "ArduinoWS03_01 | Arduino 1.6.5". The main window displays the following code:

```
const int ledPin01 = 9;
char received;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(ledPin01, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (Serial.available() > 0) {
    received = (char)Serial.read();
    Serial.println(received);
    digitalWrite(ledPin01,HIGH);
    delay(1000);
    digitalWrite(ledPin01,LOW);
  }
}
```

The status bar at the bottom indicates "Arduino Uno on /dev/cu.usbmodem1451".

ArduinoWS03_01.ino

シリアルモニタから送信：コード

定数の宣言
変数の宣言

```
{ const int ledPin01 = 9; // LEDに使うデジタルピンの番号の定義  
char received; // シリアルで読み取った文字列を格納する変数
```

初期化

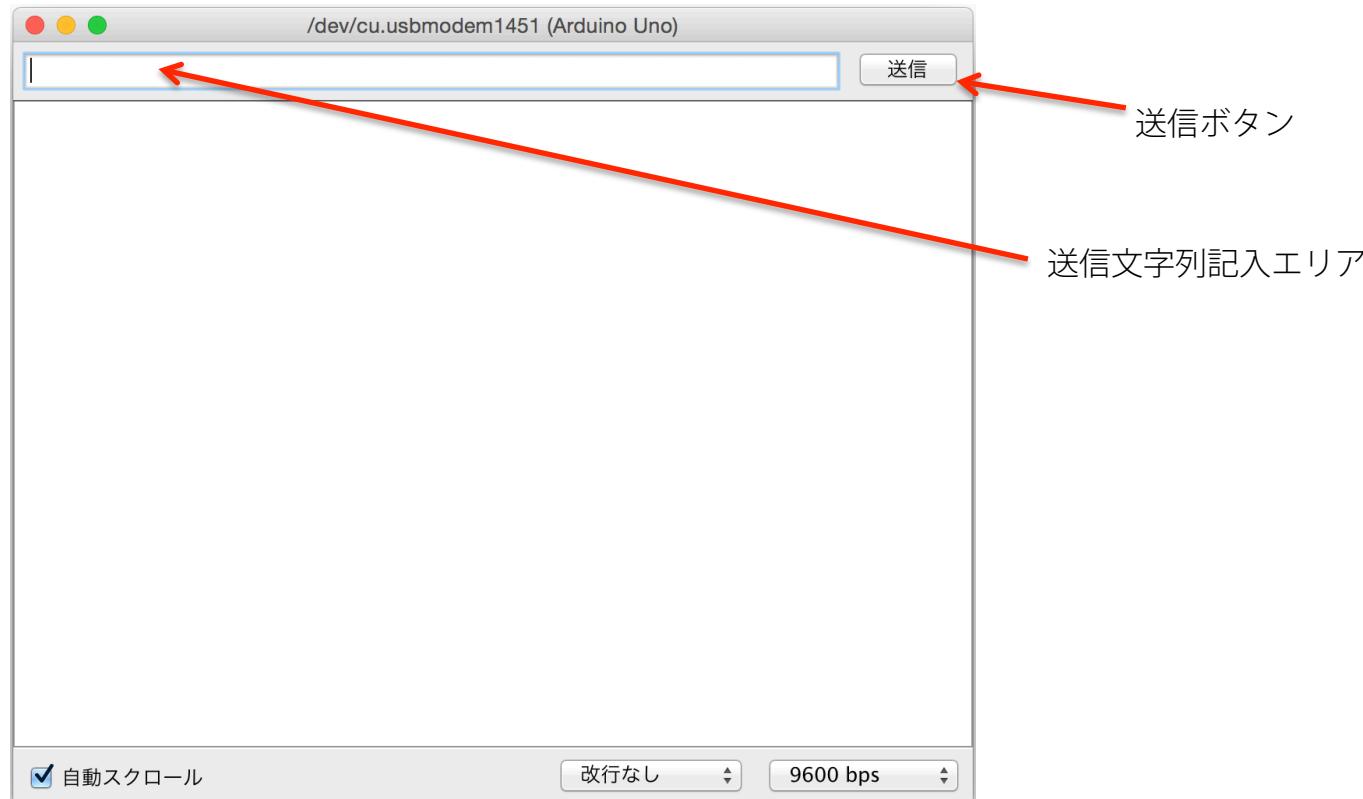
```
{ void setup() {  
    // put your setup code here, to run once:  
    Serial.begin(9600); // シリアル通信開始  
    pinMode(ledPin01, OUTPUT); // LED用のピンを出力に設定  
}
```

メイン処理
の記述

```
{ void loop() {  
    // put your main code here, to run repeatedly:  
    if(Serial.available() > 0) { // 条件分岐：もしも、シリアルに何か届いていたら  
        received = (char)Serial.read(); // 1バイト読み込みして文字に変換  
        Serial.println(received); // 読み込んだ文字をPCに送り返す  
        digitalWrite(ledPin01, HIGH); // LEDを点ける(LEDピンの電圧をHIGHにする)  
        delay(1000); // 1000ミリ秒待機  
        digitalWrite(ledPin01, LOW); // LEDを消す(LEDピンの電圧をLOWにする)  
    }  
}
```

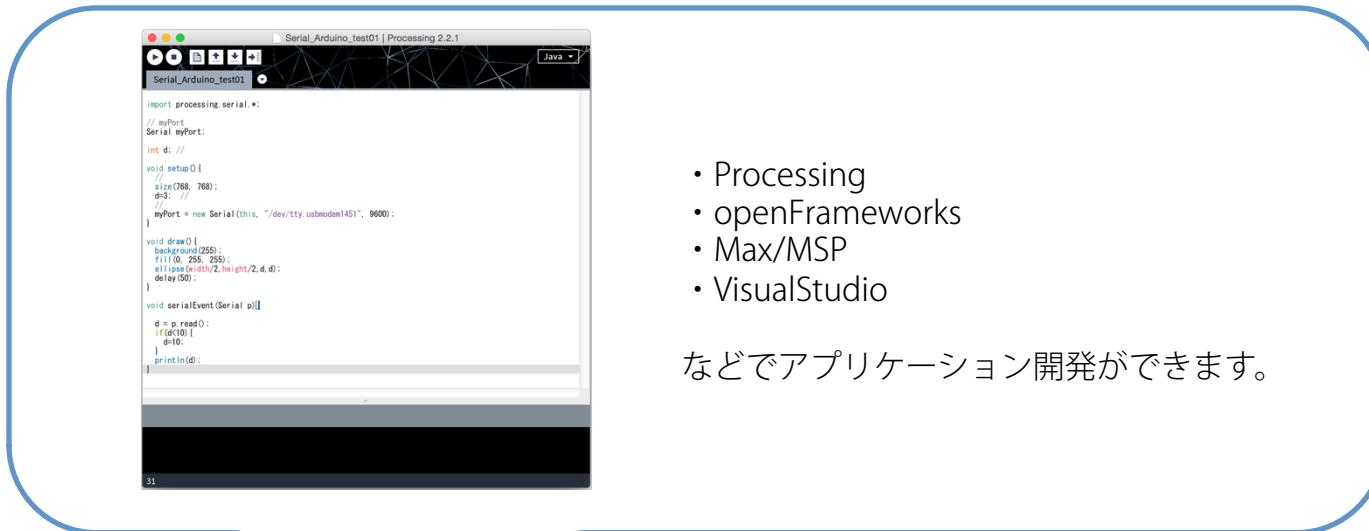
シリアルモニタから送信

送信文字列記入エリアに何か文字を入力して送信ボタンを押します。

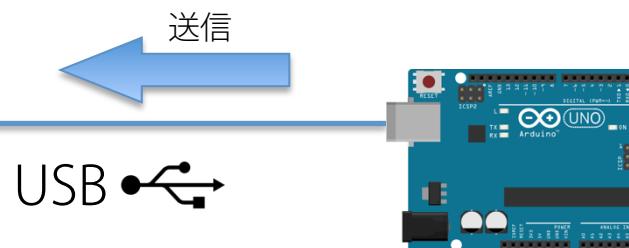


受信データをPC側で利用する

Arduinoから送信されたデータをPCで受信し、PC上のアプリケーションで利用します。



PC(Windows,Mac,Linux)



Arduinoボード

Processing

デザイナーやアーティストなどの情報処理の専門家出ない人向けに開発されたプログラミング言語で、シンプルな統合開発環境を備えています。視覚的なフィードバックが即座に得られるため、初心者がプログラミングを学習するのに適しています。Java をベースにシンプル化した言語です。

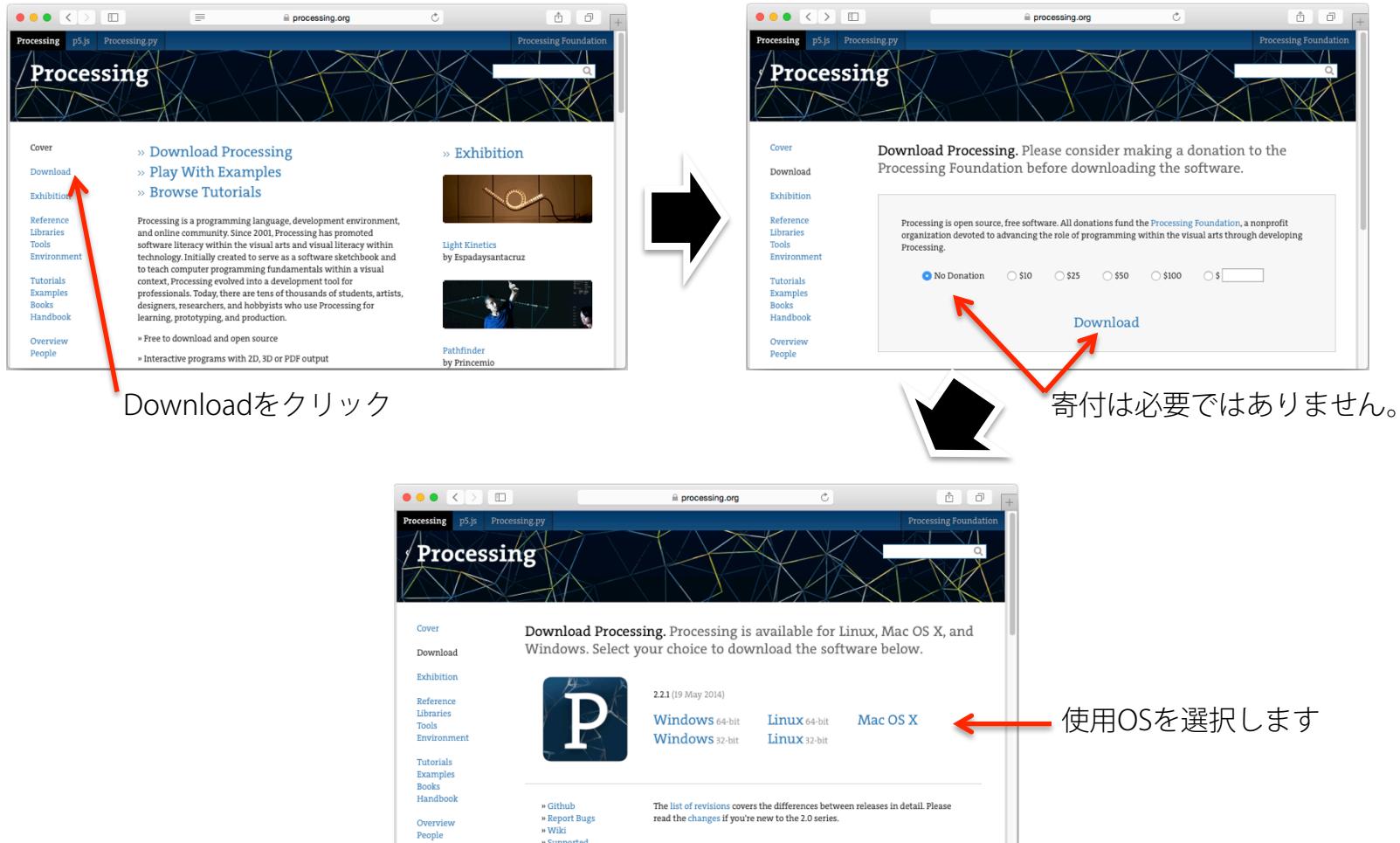


```
Serial_Arduino_test01 | Processing 2.2.1
Java ▾

import processing.serial.*;
// myPort
Serial myPort;
int d; // 
void setup() {
    // 
    size(768, 768);
    d=3; // 
    //
    myPort = new Serial(this, "/dev/tty.usbmodem1451", 9600);
}
void draw() {
    background(255);
    fill(0, 255, 255);
    ellipse(width/2, height/2, d, d);
    delay(50);
}
```

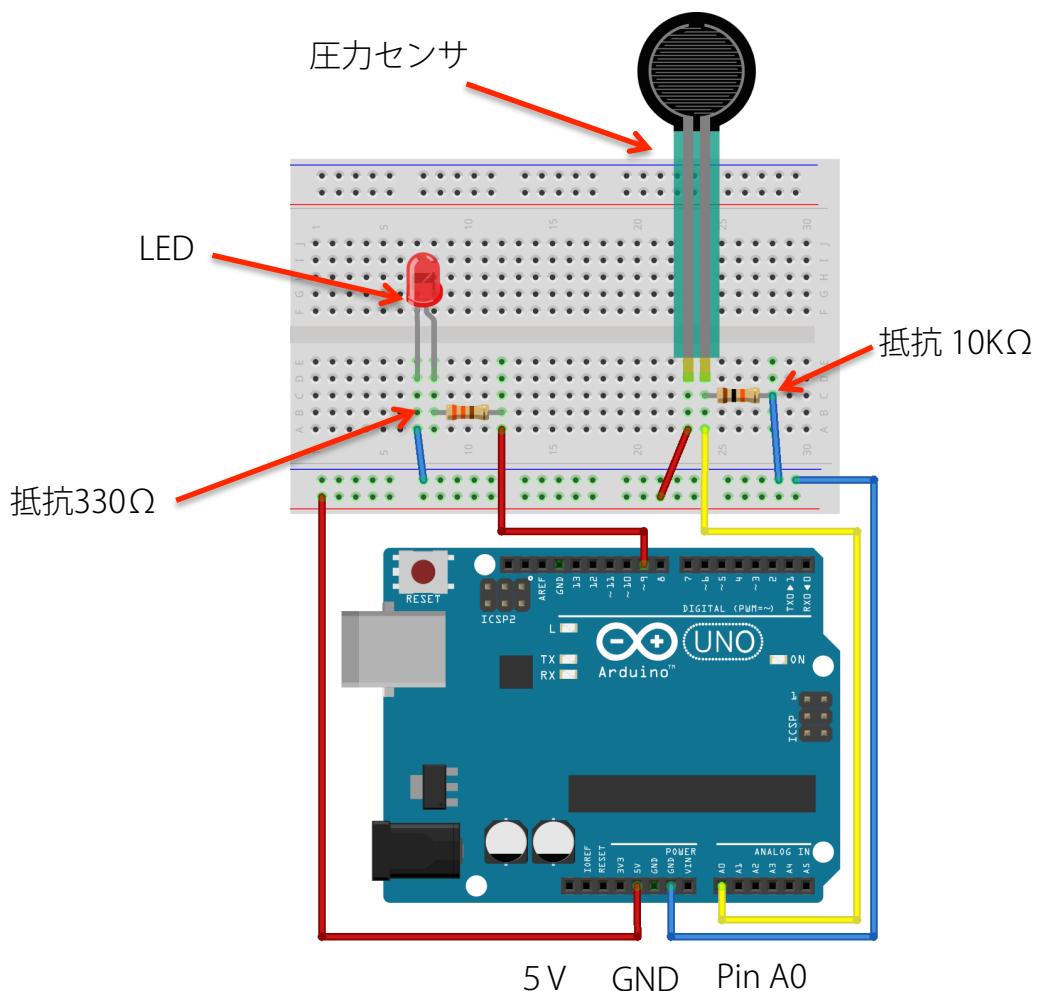
Processingのダウンロード

Processingの開発環境は「<https://processing.org>」からダウンロードします。



センサのビジュアライズ：回路

圧力センサの値を読み取るための回路を作成します。LEDはセンサ入力量の表示のものです。



LED

※脚の長い方が
+です。

カソード アノード
+ -

圧力センサ

※極性はありません

抵抗10KΩ



茶黒橙金

抵抗330Ω

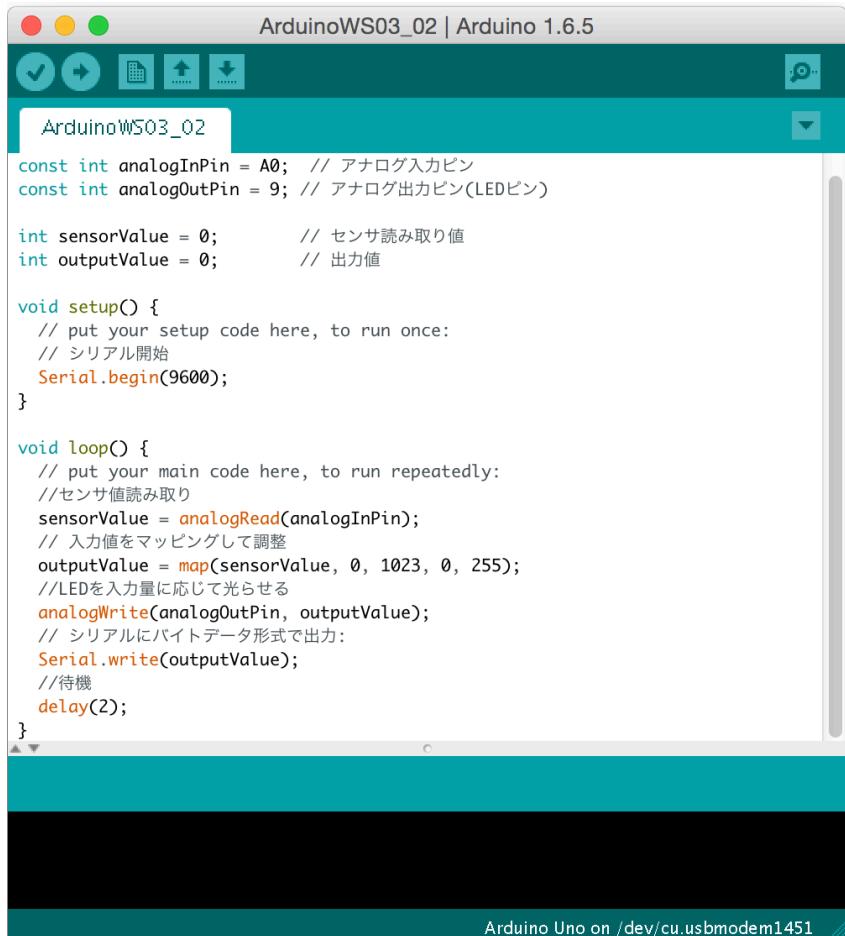


橙橙茶金

センサのビジュアライズ：コード

以下のコードを入力して実行します。

Arduino



```
ArduinoWS03_02 | Arduino 1.6.5

ArduinoWS03_02

const int analogInPin = A0; // アナログ入力ピン
const int analogOutPin = 9; // アナログ出力ピン(LEDピン)

int sensorValue = 0; // センサ読み取り値
int outputValue = 0; // 出力値

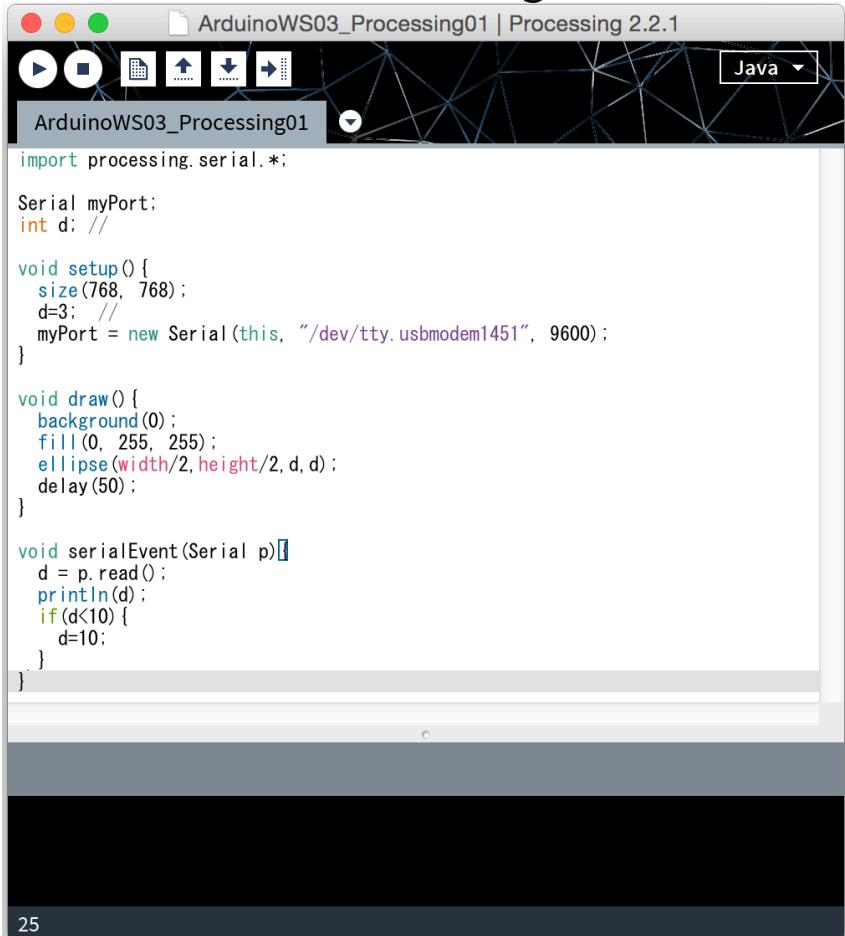
void setup() {
  // put your setup code here, to run once:
  // シリアル開始
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  // センサ値読み取り
  sensorValue = analogRead(analogInPin);
  // 入力値をマッピングして調整
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // LEDを入力量に応じて光らせる
  analogWrite(analogOutPin, outputValue);
  // シリアルにバイトデータ形式で出力:
  Serial.write(outputValue);
  // 待機
  delay(2);
}
```

Arduino Uno on /dev/cu.usbmodem1451

ArduinoWS03_02.ino

Processing



```
ArduinoWS03_Processing01 | Processing 2.2.1

ArduinoWS03_Processing01

import processing.serial.*;

Serial myPort;
int d; // 

void setup() {
  size(768, 768);
  d=3; //
  myPort = new Serial(this, "/dev/tty.usbmodem1451", 9600);
}

void draw() {
  background(0);
  fill(0, 255, 255);
  ellipse(width/2, height/2, d, d);
  delay(50);
}

void serialEvent(Serial p) {
  d = p.read();
  println(d);
  if(d<10) {
    d=10;
  }
}
```

25

ArduinoWS03_Processing01.pde

シリアルポート名の指定

通信に使われているシリアルポートの名前をArduinoIDEで確認してProcessingのコードを修正します。



ArduinoIDEでシリアルポート名をチェック

The screenshot shows the Processing 2.2.1 interface with the title "ArduinoWS03_Processing01 | Processing 2.2.1". The code editor window contains the following Java code:

```
import processing.serial.*;

Serial myPort;
int d;

void setup() {
    size(768, 768);
    d=3; //
    myPort = new Serial(this, "/dev/tty.usbmodem1451", 9600);
}

void draw() {
```

A red arrow points from the text "実際の使用環境での名前に修正します。" to the line of code where the port name is specified: "/dev/tty.usbmodem1451".

Processingコード内のシリアルポート名を
実際の使用環境での名前に修正します。

Arduino側コード

定数の宣言

```
[ const int analogInPin = A0; // アナログ入力ピン  
const int analogOutPin = 9; // アナログ出力ピン(LEDピン)
```

センサ入力に使うピンの番号の定義

LEDに使うピンの番号の定義

変数の宣言

```
[ int sensorValue = 0; // センサ読み取り値  
int outputValue = 0; // 出力値
```

センサ読み取り値格納用変数

データ出力値格納用変数

初期化

```
[ void setup() {  
//シリアル開始  
Serial.begin(9600); }]
```

シリアル通信の開始

メイン処理
の記述

```
[ void loop() {  
//センサ値読み取り  
sensorValue = analogRead(analogInPin);  
//入力値をマッピングして調整  
outputValue = map(sensorValue, 0, 1023, 0, 255);  
//LEDを入力量に応じて光らせる  
analogWrite(analogOutPin, outputValue);  
//シリアルにバイトデータ形式で出力  
Serial.write(outputValue);  
//待機  
delay(2); }
```

センサのピンの値を読み取り

値の調整：0～1023のsensorValueを0～255に割り振ってoutputValueへ

LEDピンに出力

シリアルにバイトデータ形式で出力

Processing側コード

インポート `import processing.serial.*;` シリアルクラスをインポート(決まり文句)

変数の宣言 `Serial myPort;` シリアルポートの名前を定義
`int d; //` 円の直径

初期化 `void setup(){` 画面サイズ
`size(768, 768);`
`d=3; //` 円の直径の初期値
`myPort = new Serial(this, "/dev/tty.usbmodem1451", 9600);` シリアル通信の開始(決まり文句)
}

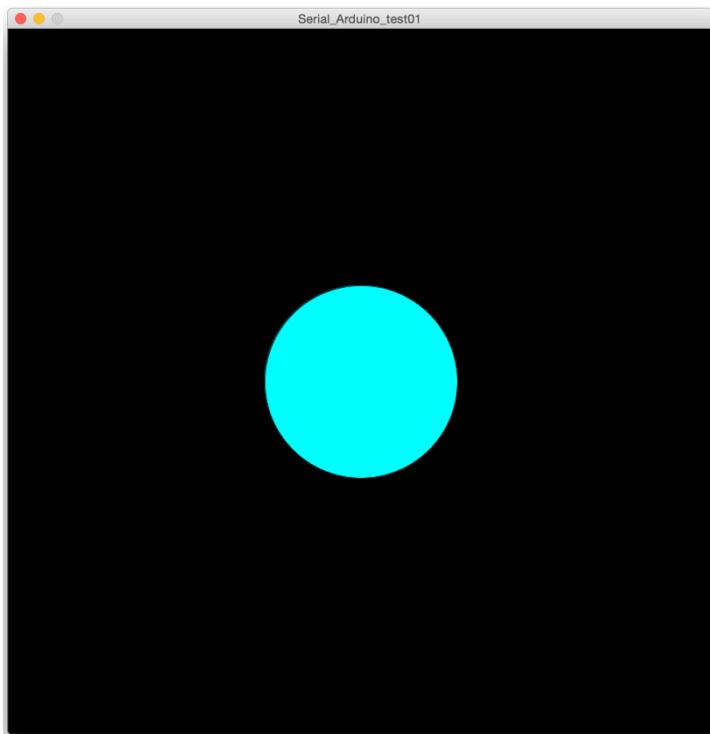
`void draw(){` シリアルポートの名前を記述
(ArduinoIDEで確認)

`background(0);` 背景色の設定(黒)
`fill(0, 255, 255);` 塗り潰し色の設定(R,G,B)
`ellipse(width/2,height/2,d,d);` 円を描画する(X座標 , Y座標 , X径 , Y径)
`delay(50);` 50ミリ秒待機
}

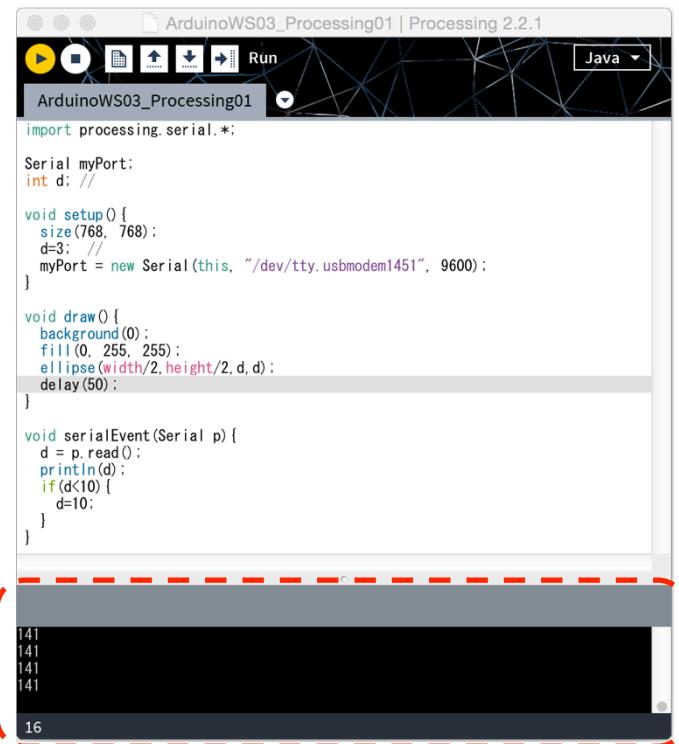
メイン処理 の記述

シリアル イベント メソッド `void serialEvent(Serial p){`
`d = p.read();` シリアルを1バイト読み込み直径に設定
`println(d);` デバッグ出力に直径を出力
`if(d<10){`
`d=10;` 直径が値が10未満なら直径を10に設定、10以上ならそのまま
}
}
} シリアルポートにデータを受信した際に自動的に呼び出されます。

表示結果



圧力センサの押し具合で
円の大きさが変化します。



```
Serial myPort;
int d; //
```

```
void setup() {
    size(768, 768);
    d=3; // 
    myPort = new Serial(this, "/dev/tty.usbmodem1451", 9600);
}

void draw() {
    background(0);
    fill(0, 255, 255);
    ellipse(width/2, height/2, d, d);
    delay(50);
}

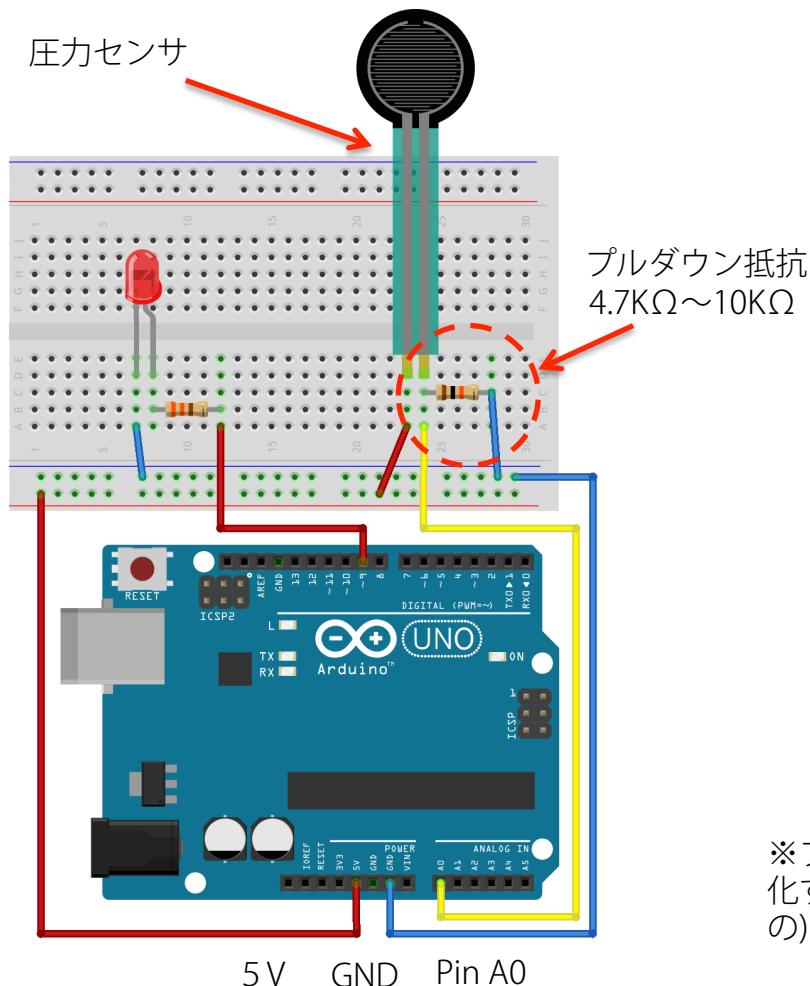
void serialEvent(Serial p) {
    d = p.read();
    println(d);
    if(d<10) {
        d=10;
    }
}
```

141
141
141
141
16

Arduinoから受信した数値が
表示されます。

プルダウン抵抗

圧力センサや、光センサ(CDS)などのセンサやタクトスイッチなどを使用する場合にセンサ入力値を安定させるために挿入する抵抗です。



入力ピンは何も繋がっていない場合や、繋がっているセンサの抵抗がMAXの場合(スイッチではOFFの場合)、本来入力電圧が0Vのはずですが、実際には0Vではなく周囲のノイズなどに影響されて不安定な値になっています。

入力ピンへの入力電圧を0Vに安定させるためには入力ピンをGNDへ繋げば良いのですが、単純にGNDと繋ぐと本来入力ピンへ流れるはずの電流がGNDピンへ逃げてしまうため、少し大きめの抵抗を介して入力ピンとGNDを接続します。この抵抗をプルダウン抵抗といいます。

※プルダウン抵抗が必要かどうかの目安としては、抵抗が変化するタイプのセンサで足が2本のもの(GNDピンの無いもの)には必要な場合が多いです。

プルダウン抵抗

プルダウン抵抗を外して入力値の不安定さを確認してみましょう。

```
import processing.serial.*;

Serial myPort;
int d; // 

void setup() {
  size(768, 768);
  d=3; //
  myPort = new Serial(this, "/dev/tty.usbmodem1451", 9600);
}

void draw() {
  background(0);
  fill(0, 255, 255);
  ellipse(width/2, height/2, d, d);
  delay(50);
}

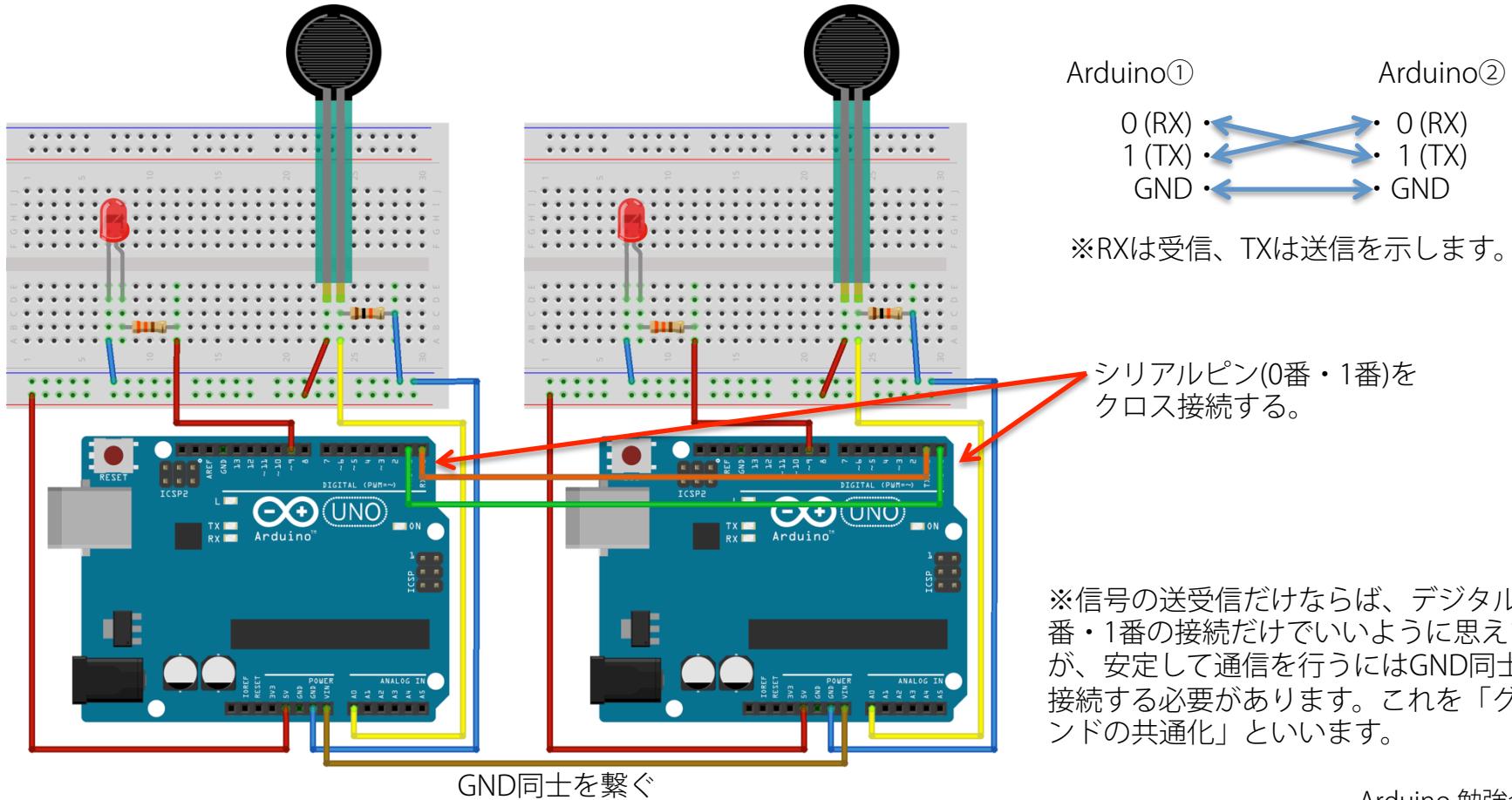
void serialEvent(Serial p) {
  d = p.read();
  println(d);
  if(d<10) {
    d=10;
  }
}
```

センサの入力値が激しく変動しているのが観察できます。

Arduino同士の通信：回路

Arduino同士でシリアル通信をしてみましょう。

デジタルの0番・1番ピンはシリアル通信のポートとして使用できます。前の例題で使用した回路を残した状態で、隣の人のArduinoと接続してみましょう。



Arduino同士の通信：コード

以下のコードを入力して実行します。



The screenshot shows the Arduino IDE interface with the title bar "ArduinoWS03_03 | Arduino 1.6.5". The main window displays the following C++ code:

```
const int analogInPin = A0; // アナログ入力ピン
const int analogOutPin = 9; // アナログ出力ピン(LEDピン)
int sensorValue = 0; // センサ読み取り値
int outputValue = 0; // 出力値
int inByte; //シリアル読み取り値

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); // シリアル開始
  Serial.write(0); // データ初回送信
}

void loop() {
  // put your main code here, to run repeatedly:
  if (Serial.available() > 0) { // もし、データを受信していたら
    // 1バイト読み込み:
    inByte = Serial.read();
    //LEDを読み込んだ値で光らせる
    analogWrite(analogOutPin, inByte);
    //センサ値読み取り
    sensorValue = analogRead(analogInPin);
    // 入力値をマッピングして調整
    outputValue = map(sensorValue, 0, 1023, 0, 255);
    // シリアルにバイトデータ形式で出力:
    Serial.write(outputValue);
  }
  delay(2); //待機
}
```

At the bottom of the IDE, it says "Arduino Uno on /dev/cu.usbmodem1411".

ArduinoWS03_03.ino

Arduino同士の通信：コード

定数の宣言

```
[ const int analogInPin = A0; // アナログ入力ピン  
const int analogOutPin = 9; // アナログ出力ピン(LEDピン)
```

センサ入力に使うピンの番号の定義
LEDに使うピンの番号の定義

変数の宣言

```
[ int sensorValue = 0; // センサ読み取り値  
int outputValue = 0; // 出力値  
int inByte; //シリアル読み取り値
```

センサ読み取り値格納用変数
データ出力値格納用変数
シリアル通信で受信した値の格納用変数

初期化

```
[ void setup() {  
    //シリアル開始  
    Serial.begin(9600);  
    Serial.write(0);  
}
```

//シリアル開始
//データ初回送信
シリアル通信の開始
データ初回送信

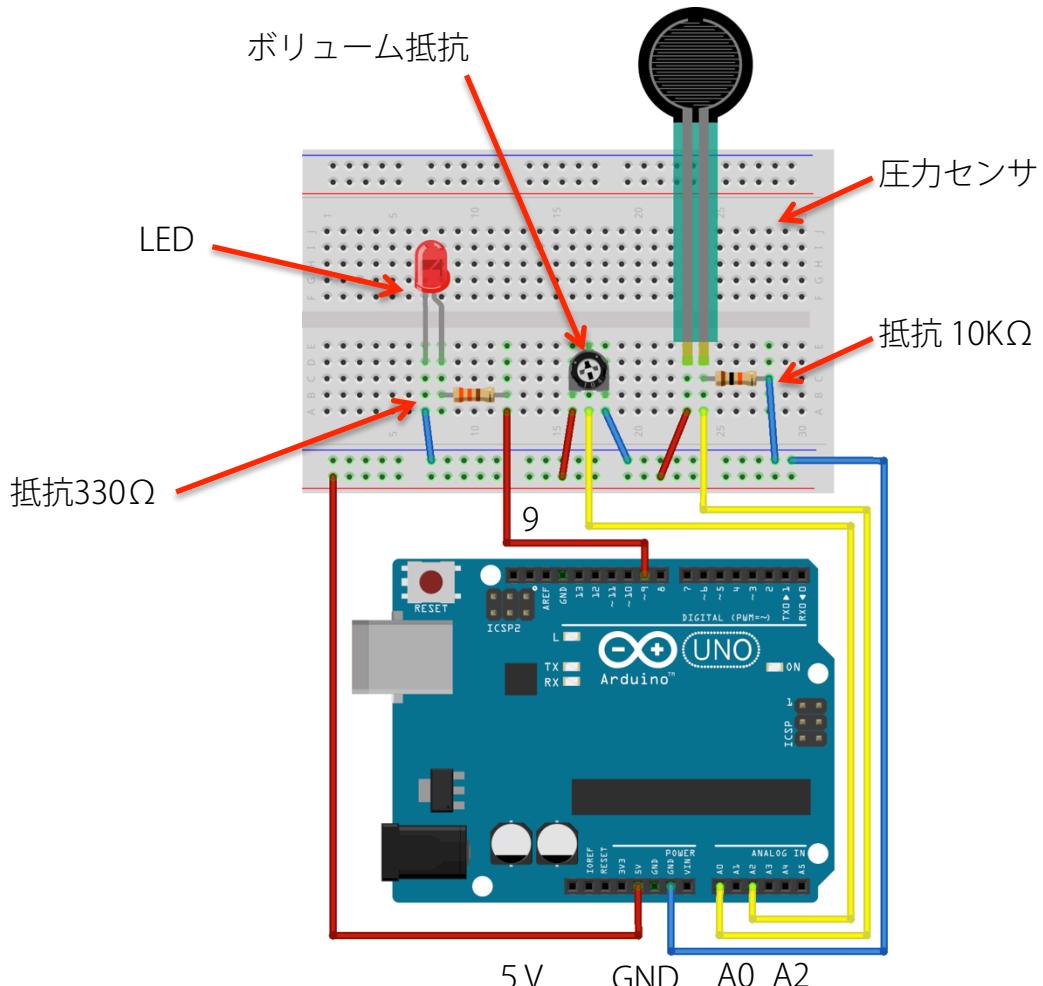
メイン処理の記述

```
[ void loop() {  
    if (Serial.available() > 0) { //もし、データを受信していたら  
        //1バイト読み込み  
        inByte = Serial.read();  
        //LEDを読み込んだ値で光らせる  
        analogWrite(analogOutPin, inByte);  
        //センサ値読み取り  
        sensorValue = analogRead(analogInPin);  
        //入力値をマッピングして調整  
        outputValue = map(sensorValue, 0, 1023, 0, 255);  
        //シリアルにバイトデータ形式で出力  
        Serial.write(outputValue);  
    }  
    delay(2); //待機
}
```

データ受信していた時だけ処理を行う
1バイト読み込み
受信した値でLEDピンに出力
センサのピンの値を読み取り
値の調整：0～1023のsensorValueを0～255に割り振ってoutputValueへ
シリアルにバイトデータ形式で出力

発展：複数のセンサを使う

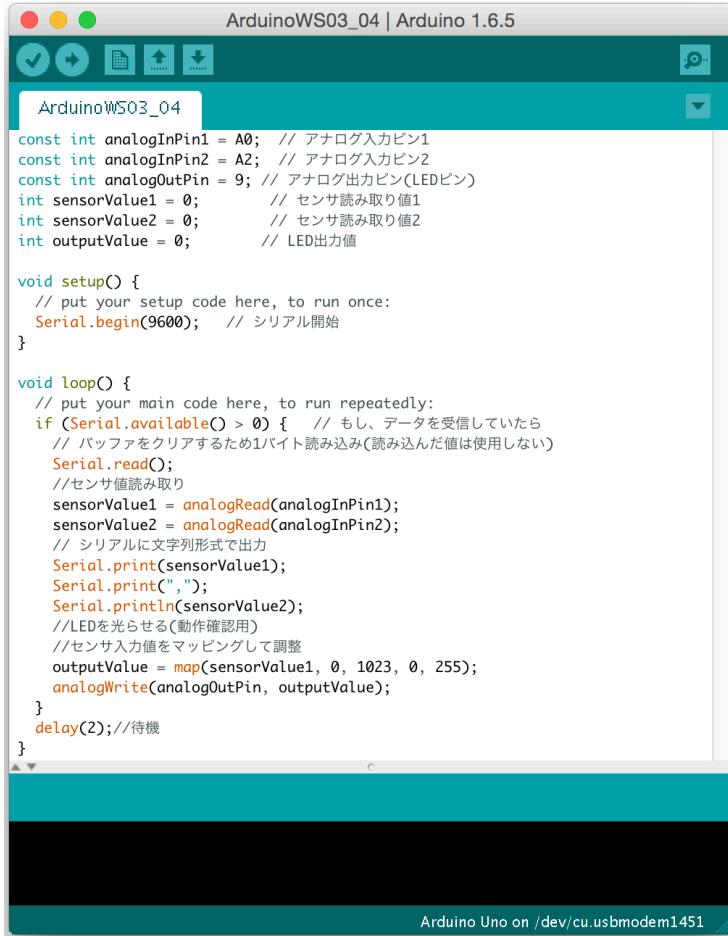
ボリューム抵抗を追加し、複数のセンサ入力をPCへ送信してみます。複数の値を送信する方法はいくつか有りますが、この例ではCSV形式の文字列送信の方法を紹介します。



複数のセンサを使う：コード

以下のコードを入力して実行します。

Arduino



```
ArduinoWS03_04 | Arduino 1.6.5

const int analogInPin1 = A0; // アナログ入力ピン1
const int analogInPin2 = A2; // アナログ入力ピン2
const int analogOutPin = 9; // アナログ出力ピン(LEDピン)
int sensorValue1 = 0; // センサ読み取り値1
int sensorValue2 = 0; // センサ読み取り値2
int outputValue = 0; // LED出力値

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600); //シリアル開始
}

void loop() {
    // put your main code here, to run repeatedly:
    if (Serial.available() > 0) { // もし、データを受信していたら
        // バッファをクリアするため1バイト読み込み(読み込んだ値は使用しない)
        Serial.read();
        //センサ値読み取り
        sensorValue1 = analogRead(analogInPin1);
        sensorValue2 = analogRead(analogInPin2);
        // シリアルに文字列形式で出力
        Serial.print(sensorValue1);
        Serial.print(",");
        Serial.println(sensorValue2);
        //LEDを光らせる(動作確認用)
        //センサ入力値をマッピングして調整
        outputValue = map(sensorValue1, 0, 1023, 0, 255);
        analogWrite(analogOutPin, outputValue);
    }
    delay(2); //待機
}
```

Arduino Uno on /dev/cu.usbmodem1451

ArduinoWS03_04.ino

Processing



```
ArduinoWS03_Processing02 | Processing 2.2.1

import processing.serial.*;
Serial myPort;
int d; //直径
int fillColor; //塗り潰し色

void setup() {
    // ウィンドウサイズの指定
    size(768, 768);
    //シリアル通信の初期化
    myPort = new Serial(this, "/dev/cu.usbmodem1451", 9600);
    myPort.clear(); //バッファのクリア
    //serialEvent()が発動するきっかけの文字列の指定
    myPort.bufferUntil('10'); //改行コード(ラインフィード)ASCIIコードで「10」
}

void draw() {
    background(0);
    fill(0,fillColor,fillColor);
    ellipse(width/2,height/2,d,d);
}

//シリアルイベント
void serialEvent(Serial p) {
    delay(10);
    //シリアルバッファーを読み込み
    String myString = myPort.readStringUntil('10');
    if(myString != null){
        println(myString);
        //空白文字などを余計な情報を消去
        myString = trim(myString);
        //コンマ区切りで複数の情報を読み込む
        int sensors[] = int(split(myString, ','));
        //読み込んだ情報の数だけ、配列に格納
        if (sensors.length >= 1) {
            d = (int)map(sensors[0], 0, 1023, 0, height);
            fillColor = (int)map(sensors[1], 0, 1023, 0, 255);
        }
        //読み込みが完了したら、次の情報を要求
        myPort.write('A');
    }
}

//マウスクリックイベント
void mousePressed() {
    //開始用データ送信
    myPort.write('A');
}
```

43

ArduinoWS03_Processing02.pde

複数センサ：Arduino側コード

定数の宣言

```
[ const int analogInPin1 = A0; // アナログ入力ピン1  
const int analogInPin2 = A2; // アナログ入力ピン2  
const int analogOutPin = 9; // アナログ出力ピン(LEDピン)
```

センサ入力に使うピンの番号の定義 x2

LEDに使うピンの番号の定義

変数の宣言

```
[ int sensorValue1 = 0; // センサ読み取り値1  
int sensorValue2 = 0; // センサ読み取り値2  
int outputValue = 0; // LED出力値
```

センサ読み取り値格納用変数 x2

LED出力値格納用変数

初期化

```
[ void setup() {  
    //シリアル開始  
    Serial.begin(9600); //シリアル開始  
}
```

シリアル通信の開始

メイン処理
の記述

```
[ void loop() {  
    // put your main code here, to run repeatedly:  
    if (Serial.available() > 0) { //もし、データを受信していたら  
        //バッファをクリアするため1バイト読み込み(読み込んだ値は使用しない)  
        Serial.read();  
        //センサ値読み取り  
        sensorValue1 = analogRead(analogInPin1);  
        sensorValue2 = analogRead(analogInPin2);  
        //シリアルに文字列形式で出力  
        Serial.print(sensorValue1);  
        Serial.print(",");  
        Serial.println(sensorValue2);  
        //LEDを光らせる(動作確認用)  
        //センサ入力値をマッピングして調整  
        outputValue = map(sensorValue1, 0, 1023, 0, 255);  
        analogWrite(analogOutPin, outputValue);  
    }  
    delay(2); //待機  
}
```

データ受信を切っ掛けとして処理を行う

PC側と同期のためのデータで
内容は無意味なので利用しない。

センサの値を読み取る x2

シリアルに文字列形式で出力

} センサ1値,センサ2値[改行]の形式
でPCに出力する。

値の調整：0～1023のsensorValueを0～255
に割り振ってoutputValueへ

LEDへ出力

複数センサ：Processing側コード①

```
import processing.serial.*;
Serial myPort;
int d; //直径
int fillColor; //塗り潰し色

void setup(){
    size(768, 768); //画面サイズの指定
    myPort = new Serial(this, "/dev/tty.usbmodem1451", 9600); //シリアルポートの名前を記述(ArduinoIDEで確認)
    myPort.clear(); //バッファのクリア
    //serialEvent()が発動するきっかけの文字列の指定
    myPort.bufferUntil(10); //改行コード(ラインフィード)ASCIIコードで「10」
}

void draw(){
    background(0); //背景色の設定(黒)
    fill(0, fillColor, fillColor); //塗り潰し色の設定(R,G,B)
    ellipse(width/2,height/2,d,d);
}
```

インポート import processing.serial.*; シリアルクラスをインポート(決まり文句)

変数の宣言 Serial myPort; シリアルポートの名前を定義

初期化 void setup(){ size(768, 768); //画面サイズの指定
myPort = new Serial(this, "/dev/tty.usbmodem1451", 9600); //シリアル通信の開始(決まり文句)
myPort.clear(); //バッファのクリア
//serialEvent()が発動するきっかけの文字列の指定
myPort.bufferUntil(10); //改行コード(ラインフィード)ASCIIコードで「10」 } シリアルポートの名前を記述(ArduinoIDEで確認)

メイン処理の記述 void draw(){ background(0); //背景色の設定(黒)
fill(0, fillColor, fillColor); //塗り潰し色の設定(R,G,B)
ellipse(width/2,height/2,d,d); }

シリアル通信の開始(決まり文句)

文字列を1行(改行コードまで)受信した場合にシリアルイベントが発生する。

円を描画する(X座標, Y座標, X径, Y径)

つづく

複数センサ：Processing側コード②

つづき

シリアル
イベント
メソッド

```
void serialEvent(Serial p){  
    delay(10); // おまじない：少し待機  
    //シリアルレバッファーを読み込み  
    String myString = myPort.readStringUntil(10);  
    if(myString != null){ // 条件分岐：読み込んだ文字列がnull(空っぽ)でなければ  
        println(myString); // デバッグ用にIDEへ出力  
        //空白文字など余計な情報を消去  
        myString = trim(myString); // データの必要な部分のみ取り出す。  
        //コンマ区切りで複数の情報を読み込む  
        int sensors[] = int(split(myString, ',')); // Split()メソッドは指定した文字(ここではカンマ)で  
        //読み込んだ情報の数だけ、配列に格納 // 分割して配列に収納する  
        if (sensors.length >= 1) { // データを収納した配列のインデックスが1以上(データ2個)以上の場合  
            d = (int)map(sensors[0], 0, 1023, 0, height); // データの数値を整理(0~1023を0~ウィンドウ高さ)に調整  
            fillColor = (int)map(sensors[1], 0, 1023, 0, 255); // 整数型にキャスト  
        }  
        //読み込みが完了したら、次の情報を要求  
        myPort.write('A'); // こちらから何か送信することがArduino側が動作する切っ掛けになっている  
    }  
}
```

マウス
クリック
イベント
メソッド

```
//マウスクリックイベント  
void mousePressed(){  
    //開始用データ送信  
    myPort.write('A'); // こちらから何か送信することがArduino側が動作する切っ掛けになっている
```

起動後マウスをクリックすると
データ通信が開始される

参考URL

Processing公式

- processing.org

<https://processing.org>

開発元です。ソフトウェアのダウンロードなどはここから行います。

- Processingリファレンス

http://tetraleaf.com/p5_reference_alpha/

Processing言語の仕様について書かれています。
ここで関数などが調べられます。

- Processingクイックリファレンス

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

Processing言語の仕様について逆引き形式で解説
しています。