プログラミング演習 II
フィジカルコンピューティング

第2回 シリアル通信、色々なデバイスを使う

中村，小松，小林，橋本

（監修：橋本直）

今日の内容

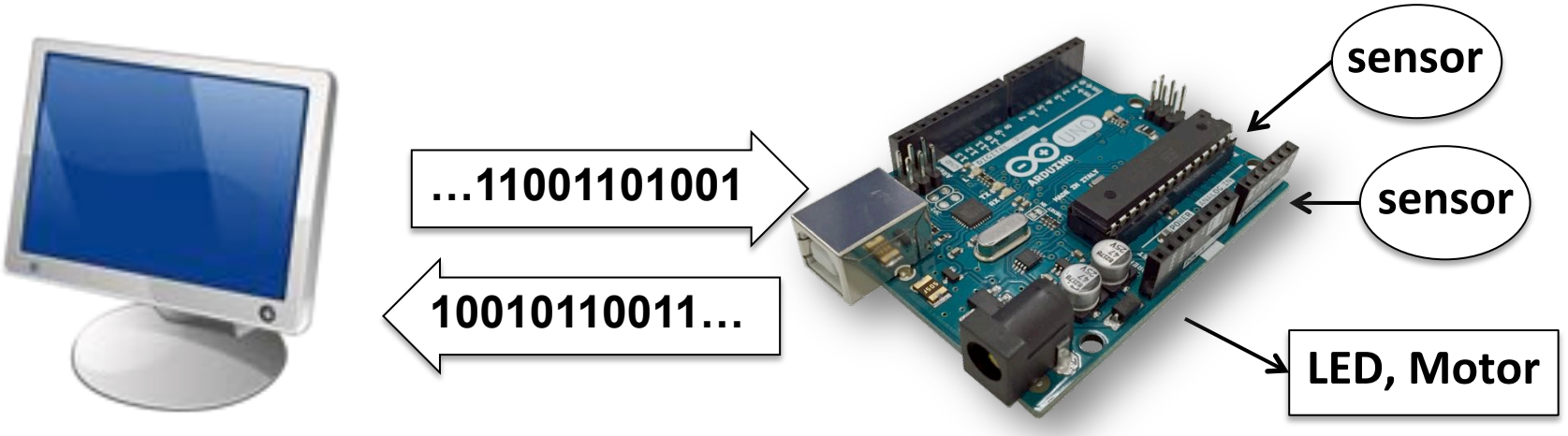
(1) シリアルモニタを使用したシリアル通信
  - センサからの入力値をPCに送る（Arduino→PC）
  - PCからの文字を受け取ってArduinoを制御する（PC→Arduino）

(3) カラーLEDを使う（PCから色を指定して光らせる）

(4) いろいろなセンサーを使う（シリアルモニターで値を見る）

(5) 液晶ディスプレイを使う（I2Cデバイス）

シリアル通信



- 10010110011…のようなデータ信号を1ビットずつ送る通信方式
- ArduinoとPCを接続して、互いに数値や文字列のデータをやりとりできる
- Arduinoで計測したセンサ情報をPCに送ったり、PCからのコマンド送信によってLEDやモータを制御したりできる

シリアル通信をやってみよう

- 回路は光センシングの回路のまま
  ※第１回資料の「光センサで光の強さをセンシングする回路」のページを参照

【データの送信】（Arduino→PC）

– 光センサの入力値をPCに送信してモニタリング

【データの受信】（PC→Aruidno）

– PCからコマンドを送ってLEDをコントロール

【データの送信】
光センサの入力値をPCに送信するプログラム

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

void loop() {
    int value = analogRead(0);
    Serial.println(value);
}
```

シリアル通信を開始する
（通信速度：9600bps）

アナログの0番ピンを
読み取る

データをPCに送信する

命令の意味

- **Serial.begin(通信速度)**
  - シリアル通信を開始する。通信速度は以下のいずれか

    300bps, 1200bps, 2400bps, 4800bps, 9600bps, 19200bps, 57600bps, 115200bps

    単位: bps = bits per second（1秒間に送るビット数）

- **Serial.print(データ)　… 改行なし**

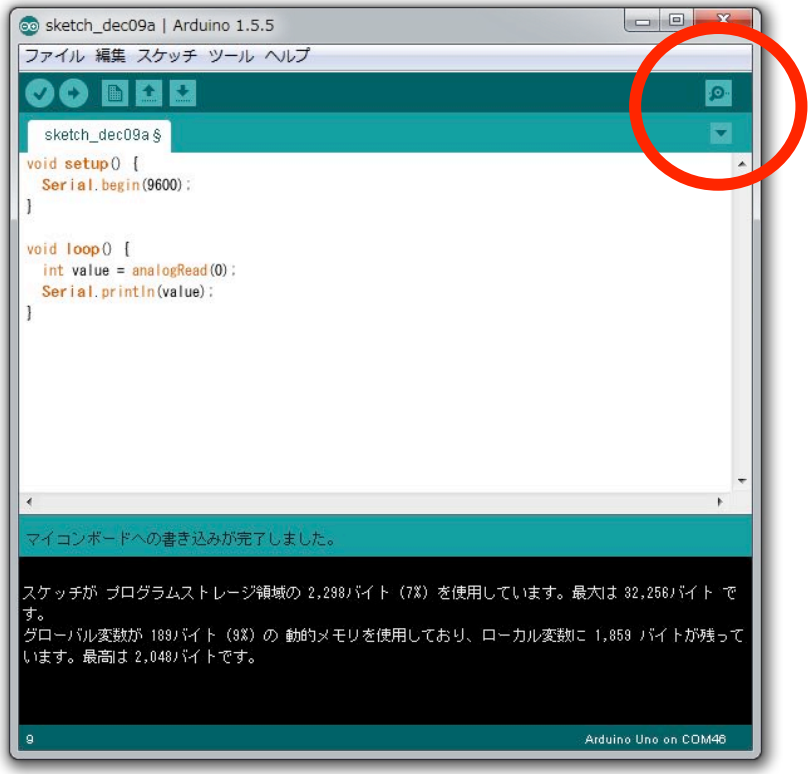
  Serial.println(データ)　… 改行あり
  - 数値や文字列を送れる
  - Arduinoでは「+」による文字列の結合はできないので注意

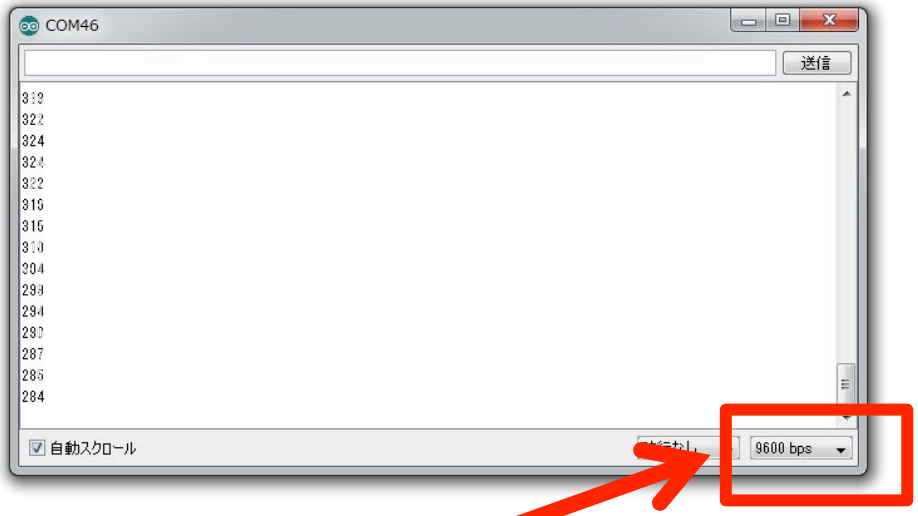
Serial.print()による文字の出力

- Serial.print(78) - “78”が出力される
- Serial.print(1.23456) - “1.23”が出力される
- Serial.print(‘N’) - “N”が出力される
- Serial.print(“Hello world.”) - “Hello world.”と出力される

データのモニタリング

- ウィンドウ右上にある「シリアルモニタ」ボタンをクリック
- シリアルモニタが起動してデータが流れてくる

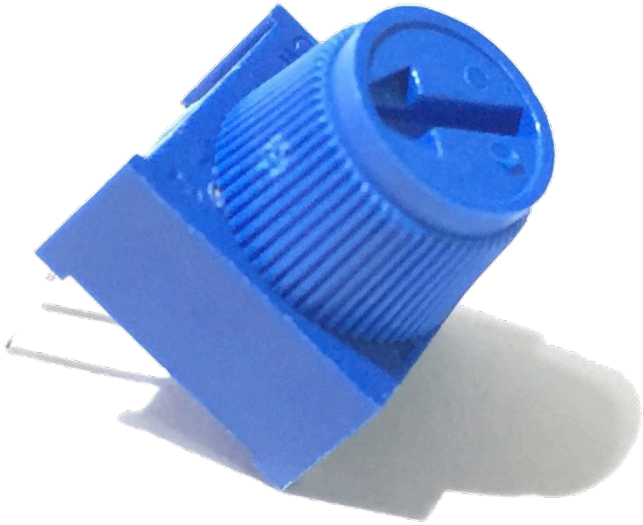


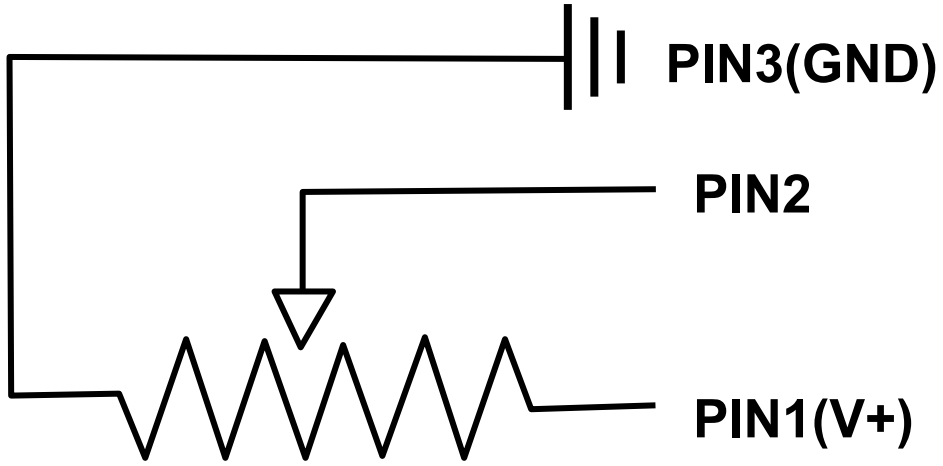


通信速度の設定を、データを送るプログラムとあわせる(9600bps)

半固定抵抗(ボリューム)

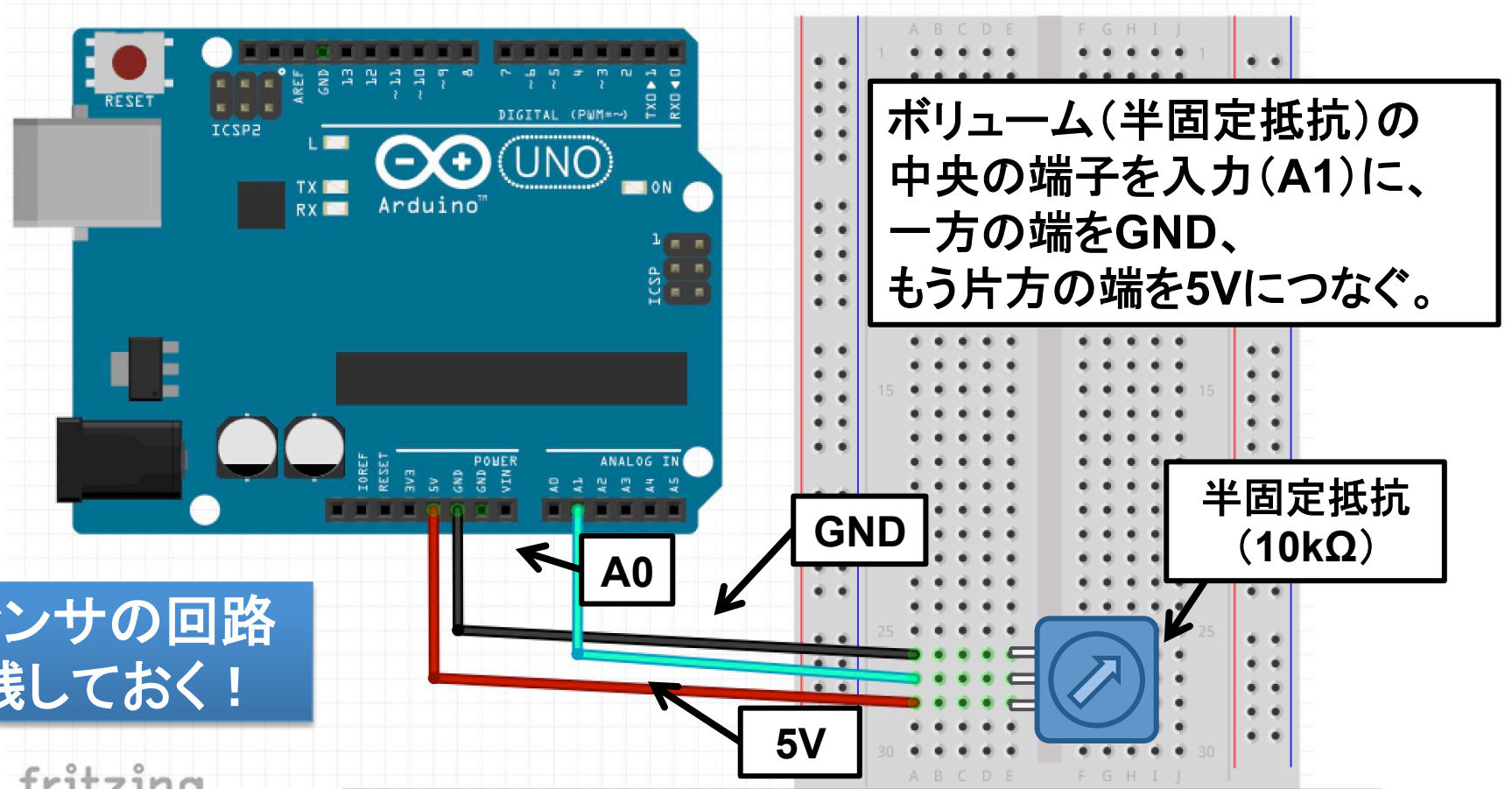
- つまみを回すと角度に応じて抵抗値が変化する可変抵抗器。
  - ある意味、センサ
  - 両側の端子に電源とGNDを接続すれば、真ん中の端子から角度に応じた電圧を取出せる。





ボリューム(半固定抵抗)の変化を読み取る回路

- 光センサと抵抗の回路を半固定抵抗に置き換えて、受信するデータを見てみよう。角度と数値の関係は？



A1を読むようにプログラムを変更する！

【データの受信】コマンドでLED制御

```
void setup() {
    Serial.begin( 9600 );
    pinMode( 7, OUTPUT );
}

void loop() {
    if ( Serial.available() > 0 ) {
      int data = Serial.read();
      if ( data == 'a' ) {
        digitalWrite( 7, HIGH );
      }
      else if ( data == 'b' ) {
        digitalWrite( 7, LOW );
      }
    }
}
```

シリアル通信を開始する

7番ピンを出力に設定

データがきているかチェック

データを1バイト読み込む

データが「a」だったら
LEDをONにする

データが「b」だったら
LEDをOFFにする

命令の意味

- **Serial.available()**
  - 受信したデータのバイト数を返す
  - 1以上の値を返せばデータが来ていることになる
  - Arduinoは一度に128バイトまで保持する

- **Serial.read()**
  - 受信したデータの最初の1バイトを返す
  - 戻り値は int型

PCからのデータの送信

- シリアルモニタを起動する

- 入力欄にコマンド（「a」または「b」）を入力して送信ボタン（またはエンターキー）を押す
- 「a」を送った時にLEDがONになり、「b」を送った時にLEDがOFFになる

（補足）文字「a」を送るということ

- 文字にはそれぞれ「文字コード」と呼ばれる識別番号が割り当てられている。

  （例）「a」は97、「A」は65、「1」は49

  参考： http://www9.plala.or.jp/sgwr-t/c_sub/ascii.html

- 「a」の文字コード（97）を2進数で表すと「01100001」

- この1/0の並びをON/OFFの電気信号に置き換えて伝送している。

（補足）int型での受信データの判定

- Serial.read()は受信データをint型で返す

- 文字を受信したときは？
  - 文字コードが受信される

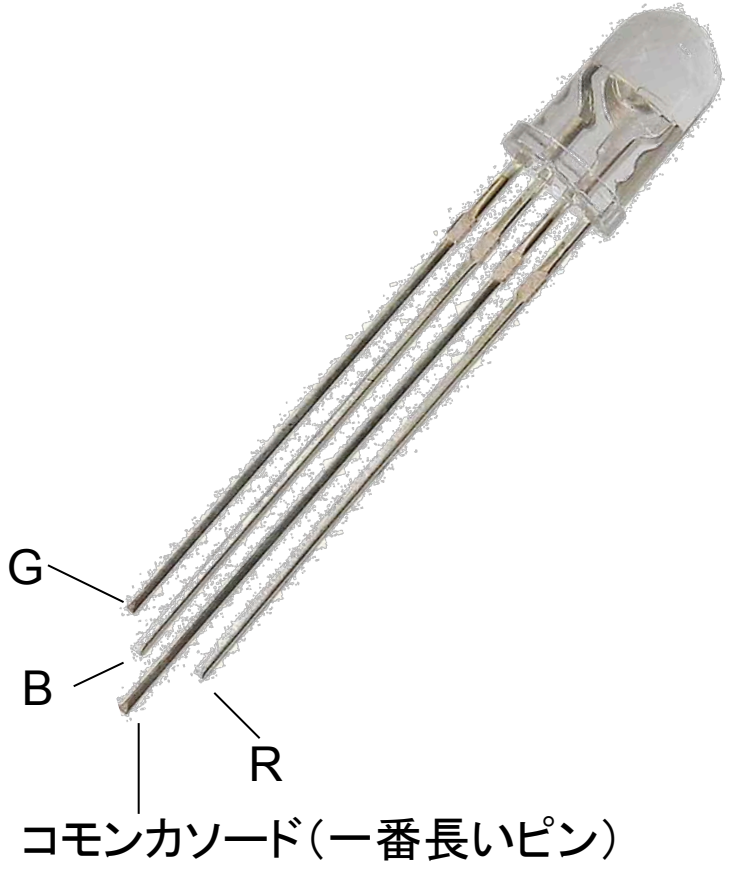
  - すなわち、「a」という文字がきた時は、97というint型の値を受け取っている

  - なので、このように判定することもできる（’a’ は 97 と置換可能）

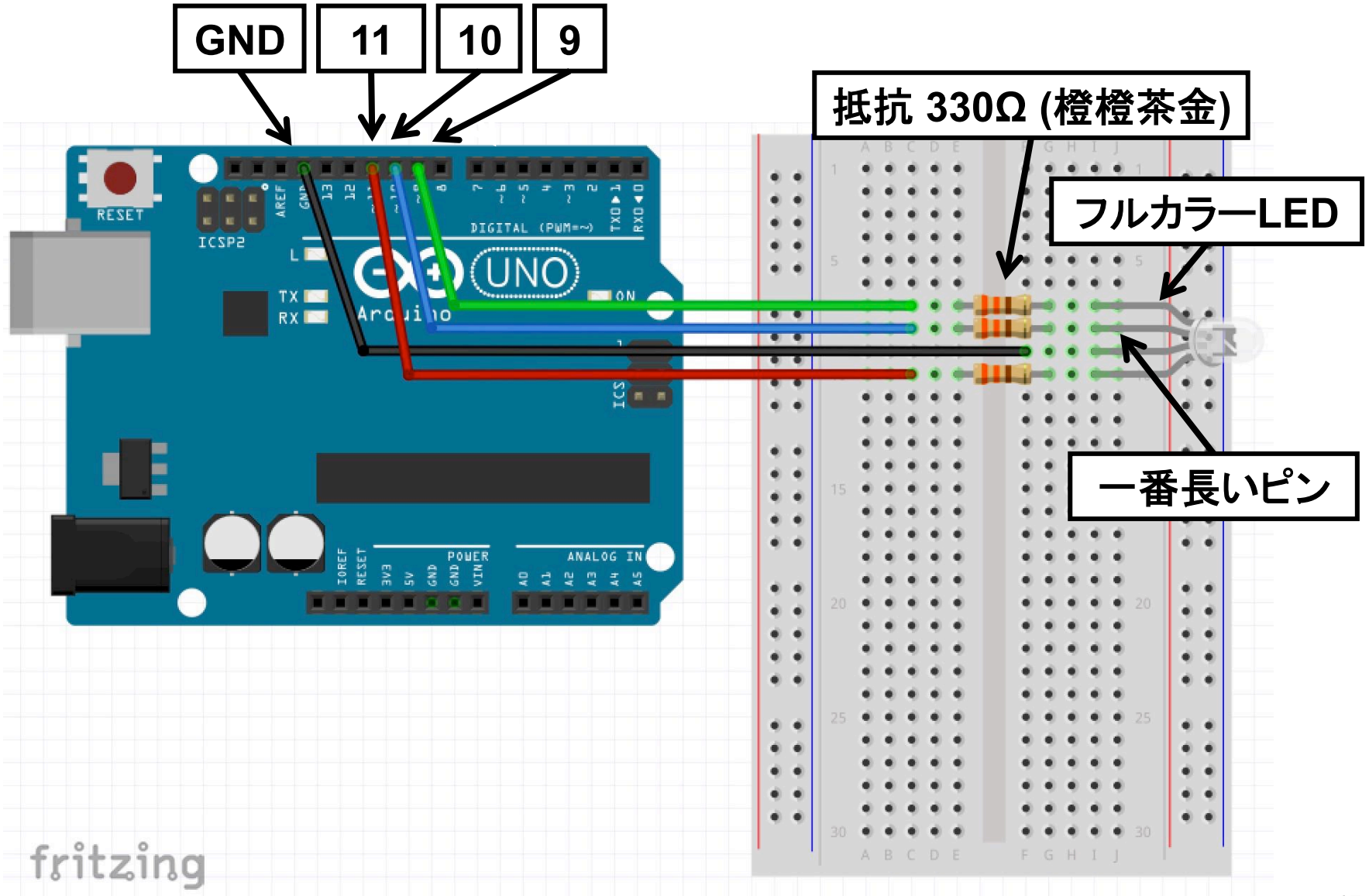
```
int data = Serial.read();
if ( data == 97 ) {
  digitalWrite( 9, HIGH );
}
```

フルカラーLED

- R,G,B 3色のLEDが1つになった素子。
- 各色の強度を調整して色々な色が出せる



フルカラーLEDを点灯する回路



フルカラーLED（アナログ出力の応用）

```
void setup() {
    pinMode( 9, OUTPUT );
    pinMode( 10, OUTPUT );
    pinMode( 11, OUTPUT );
}

void loop() {
    analogWrite( 9, random(0,255) );
    analogWrite( 10, random(0,255) );
    analogWrite( 11, random(0,255) );
    delay(1000);
}
```

9,10,11番ピンを出力に設定
どれも、~が付いているピン

赤の強度を設定
緑の強度を設定
青の強度を設定

いろいろなセンサーを使ってみよう

- 光センサー（先週）
- ボリューム
- 感圧センサ
- フォトリフレクタ
- 温度センサ

→アナログ電圧

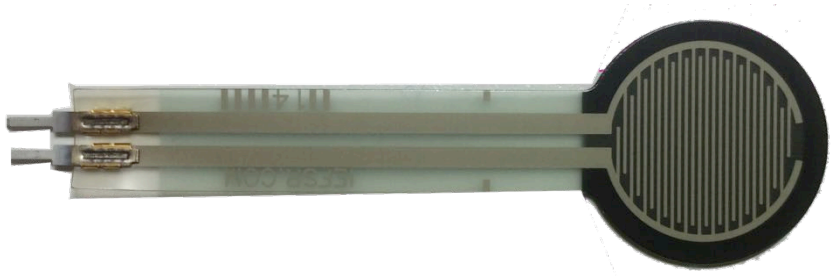
- 超音波距離センサ

→パルス幅

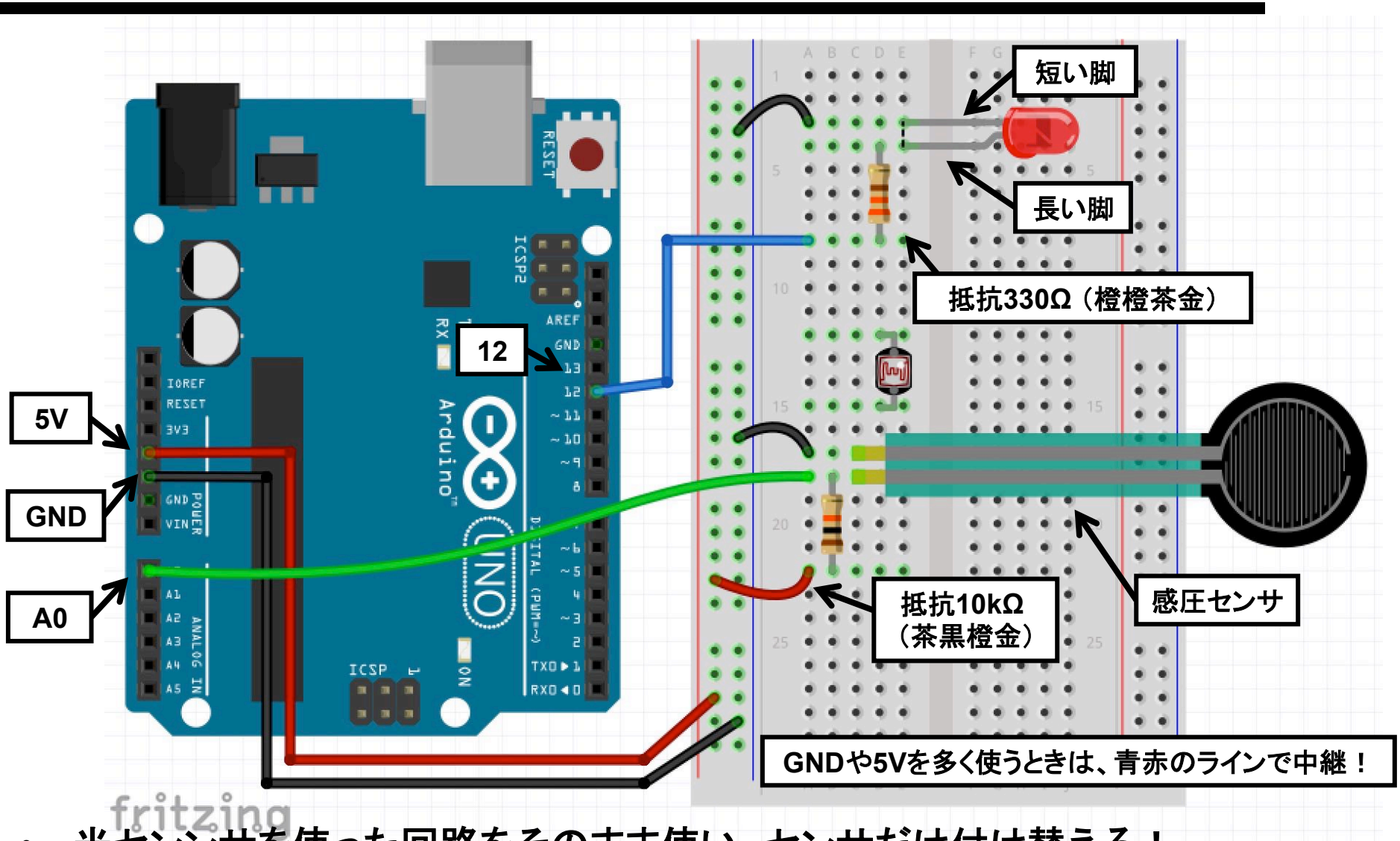
※他にシリアル通信でデータの受け渡しを行うセンサもある

感圧センサ

- 押す力に応じて抵抗値が変化するセンサ
- 光センサをはずして感圧センサで置き換えてみよう(根元が折れやすいので注意！ジャンパワイヤを使うのも良い)
- 力に応じて値はどのように変化するだろうか



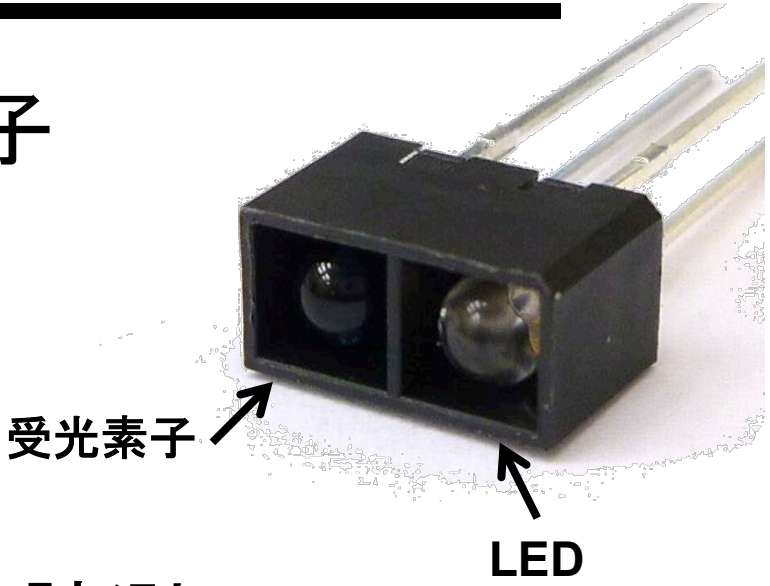
感圧センサで圧力をセンシングする回路



- 光センンサを使った回路をそのまま使い、センサだけ付け替える！
- プログラムは「光センサの入力値をPCに送信するプログラム」と同じ

フォトリフレクタ

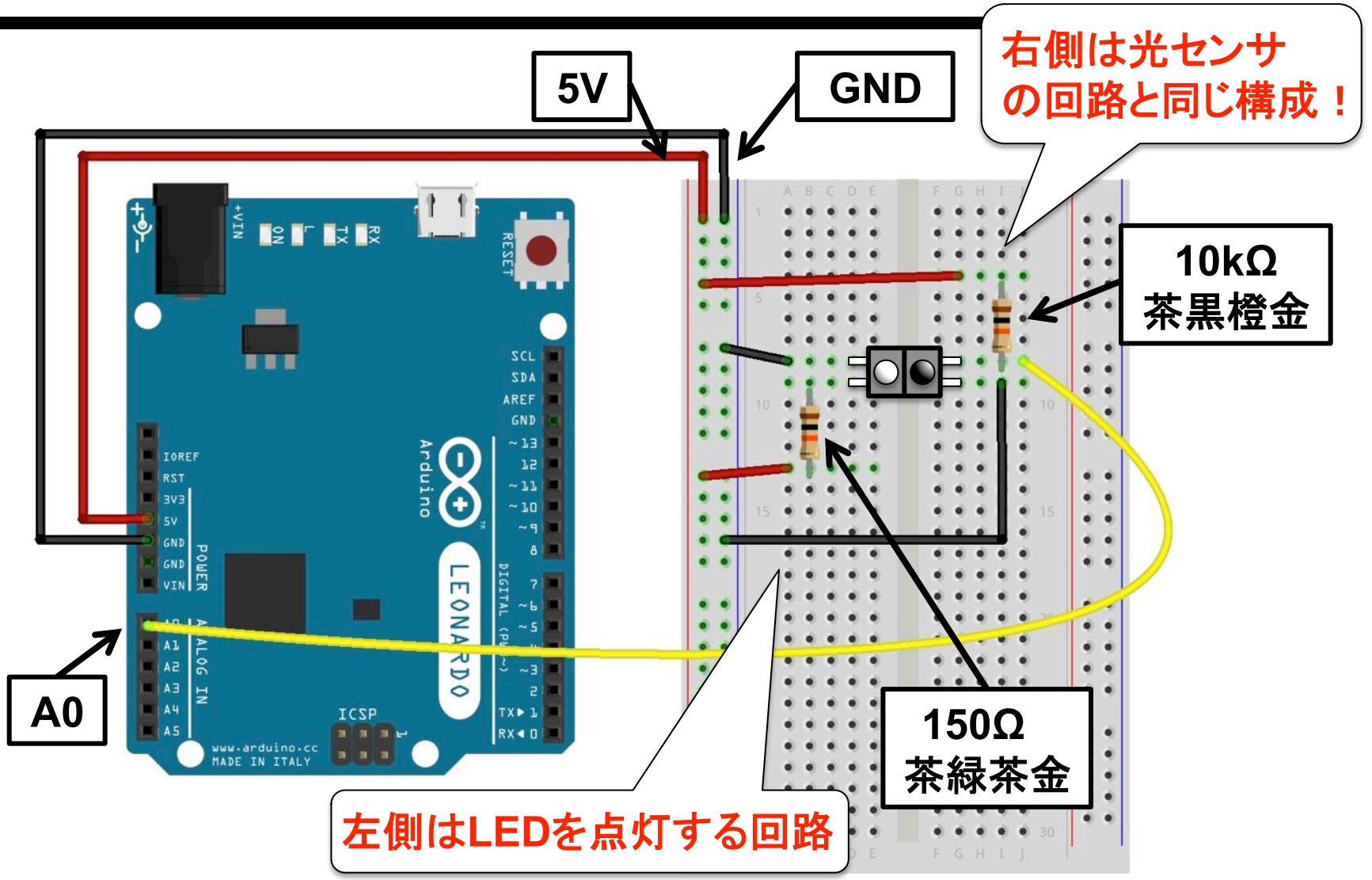
- 赤外線LEDと赤外線受光素子を合体したセンサ



- 赤外線の反射光の「強度」を計測
  - 紙の白黒を判別したり、かざした手との距離を測ったり、目の横に付けてまばたきを検出したり、クッションの中にいれて圧力を計測したり、指先の脈波を計測したり、etc…

- 参考ページ　http://kougaku-navi.net/sensact/

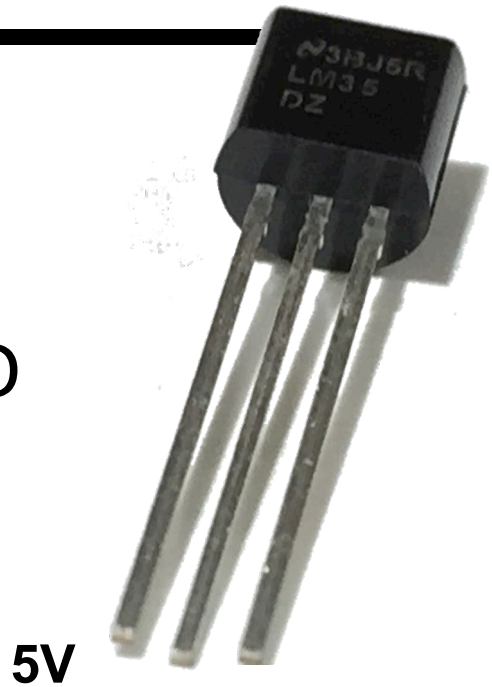
フォトリフレクタを使う回路



- アナログ入力を使って電圧の変化を読み取ってみよう
- プログラムは「光センサの入力値をPCに送信するプログラム」と同じ

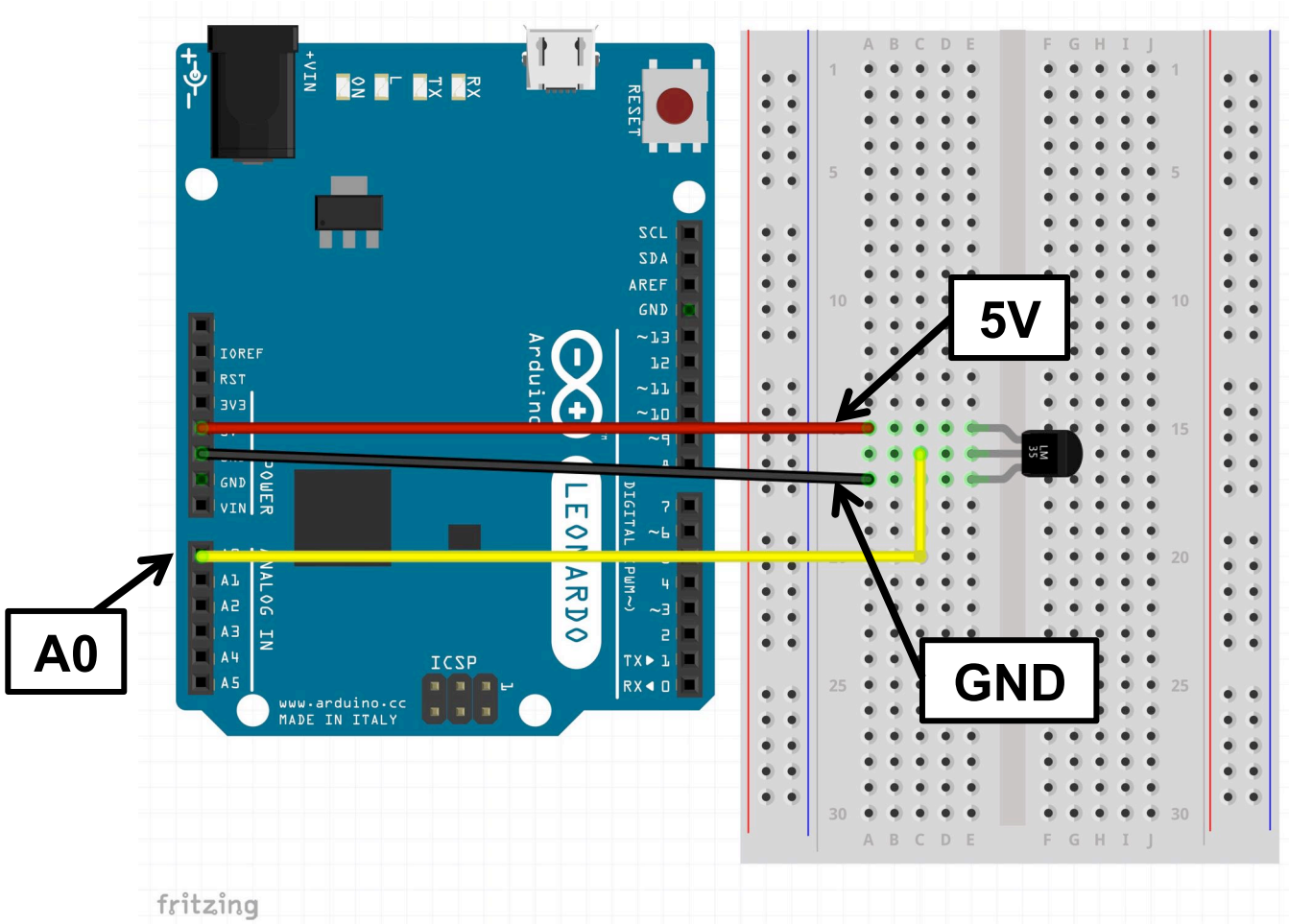
温度センサ（LM35）

- 温度を測るIC
  ※トランジスタと形が似ているので注意
- 左のピンに電源、右のピンにGND
  をつなぐと、中央のピンに温度
  に対応した電圧が出力される。



- 0℃で0V，1℃あたり10mVの変化となる仕様。
  - Arduinoのアナログ入力は0～5Vを0～1023の数値に変換するので、以下の式で得られる
    摂氏温度＝　(5*読取値/1024)*100

温度センサ（LM35）を使う回路



読み取った値を温度に変換してPCに送ってみよう

温度センサ（LM35）の値を読むプログラム

```
float analogData;  // 0 - 1023
float tempC = 0;   // 摂氏温度

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

void loop(){
  analogData = analogRead(0);
  tempC = ((5 * analogData) / 1024) * 100;
  Serial.println( tempC );
  delay(1000);
}
```

シリアル通信を開始する

LM35の出力電圧を読む

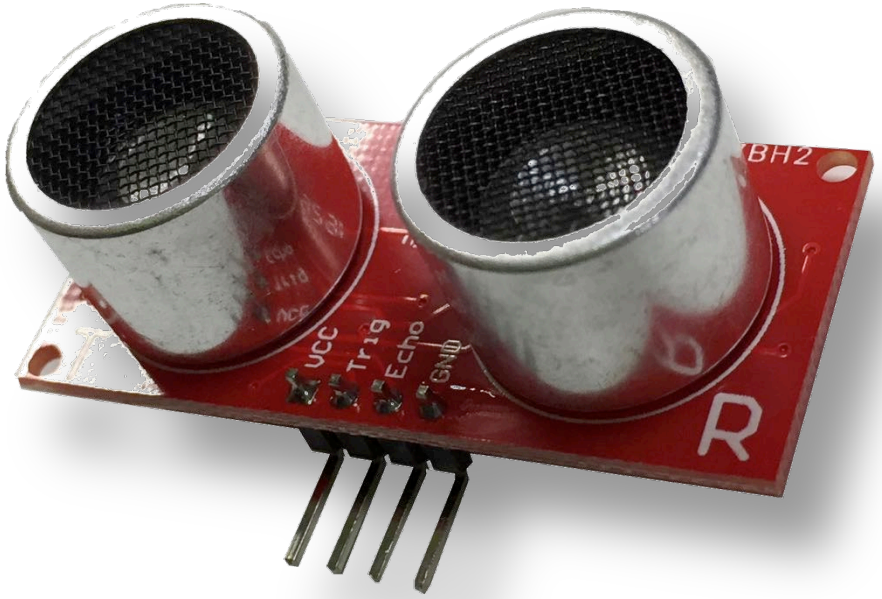
摂氏温度に変換する

- プログラムは「光センサの入力値をPCに送信するプログラム」とほとんど同じ。値を変換してから送る。

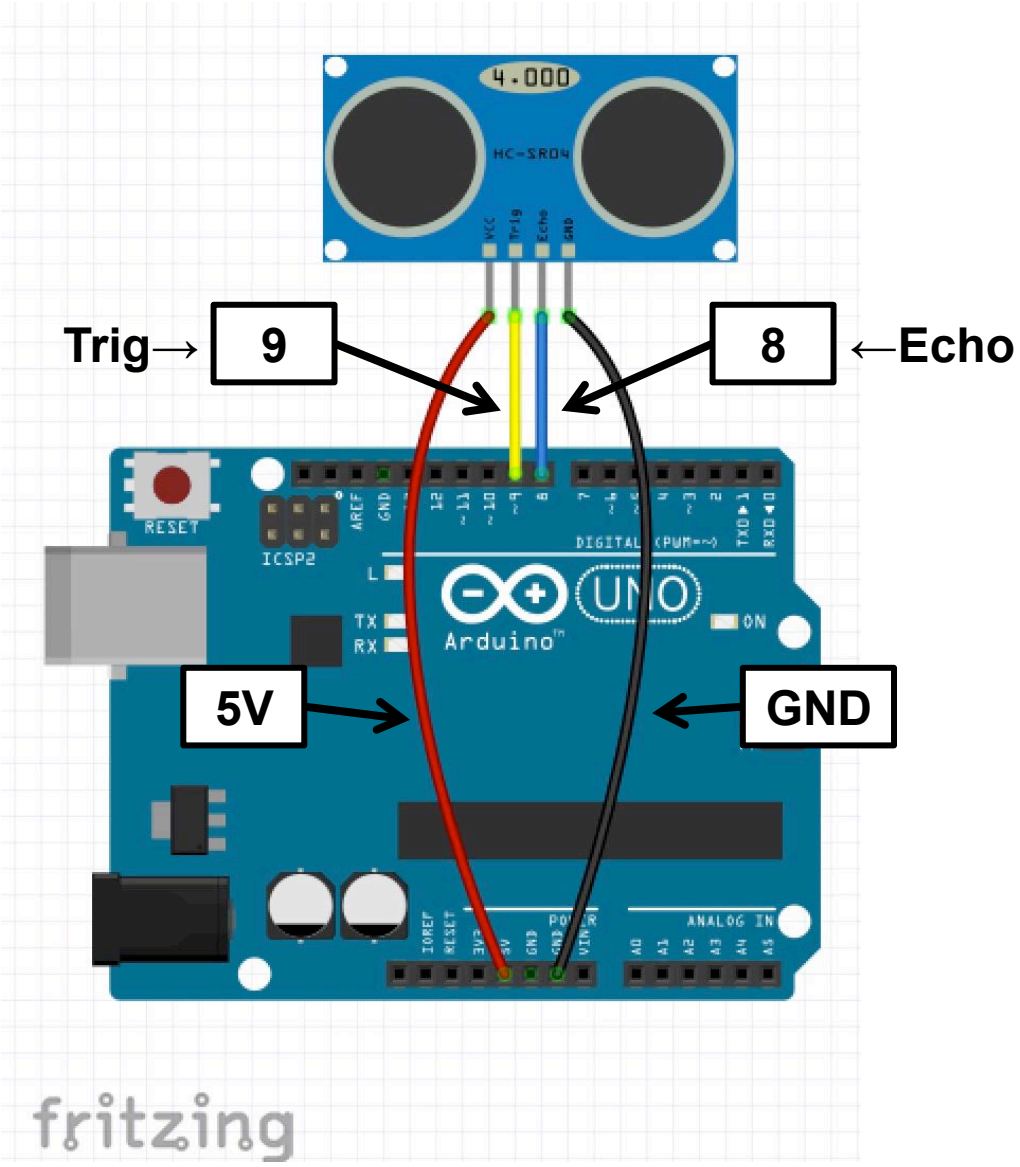
超音波距離センサ

- 超音波パルスを発射して、反射して戻るまでの時間を計ることで距離を計るセンサ
- 製品WEBサイトを見て使い方を確認

https://www.switch-science.com/catalog/2860/



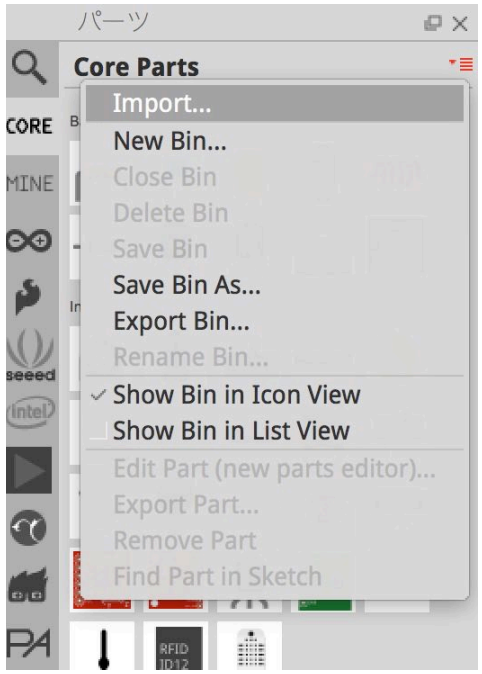
超音波距離センサを使う回路



補足：FRITZINGの部品を追加する方法

(1) http://fritzing.org/projects/hc-sr04-project から部品ファイルをダウンロード

(2) Fritzing で部品ファイルをインポート



超音波距離センサの値を読むプログラム

```
int Trig = 9;   // Trigger に接続したピン
int Echo = 8; // Echo に接続したピン
int Duration;
float Distance;

void setup() {
  Serial.begin(9600);
  pinMode(Trig,OUTPUT);
  pinMode(Echo,INPUT);
}
```

Triggerは出力，
Echoは入力に設定

パルス入力命令

- **pulseIn(** ピン番号, 値, timeout **)**
  - 値(HIGHまたはLOW）でパルスの種類を指定
  - 指定したピン届いたパルスの幅をミリ秒単位で返す

超音波距離センサの値を読むプログラム

（つづき）

```
void loop() {
  digitalWrite(Trig,LOW);
  delayMicroseconds(1);
  digitalWrite(Trig,HIGH);
  delayMicroseconds(11);
  digitalWrite(Trig,LOW);
  Duration = pulseIn(Echo,HIGH);
  if (Duration>0) {
    Distance = (float)Duration/2*340*100/1000000;
    Serial.print(Distance);
    Serial.println(" cm");
  }
  delay(500);
}
```

TrigピンをLOW→HIGH→LOW
と変化させて超音波パルスを発射

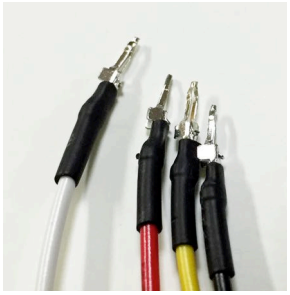
pulseIn関数でechoピンに入力
されるパルス幅をms単位で計測

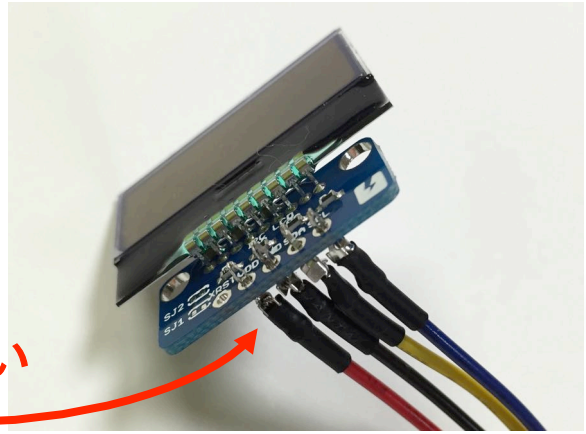
パルスの往復にかかった時間（ms）
を2で割り
音速(340m/s)をかけ、cm/secに換算

液晶ディスプレイ

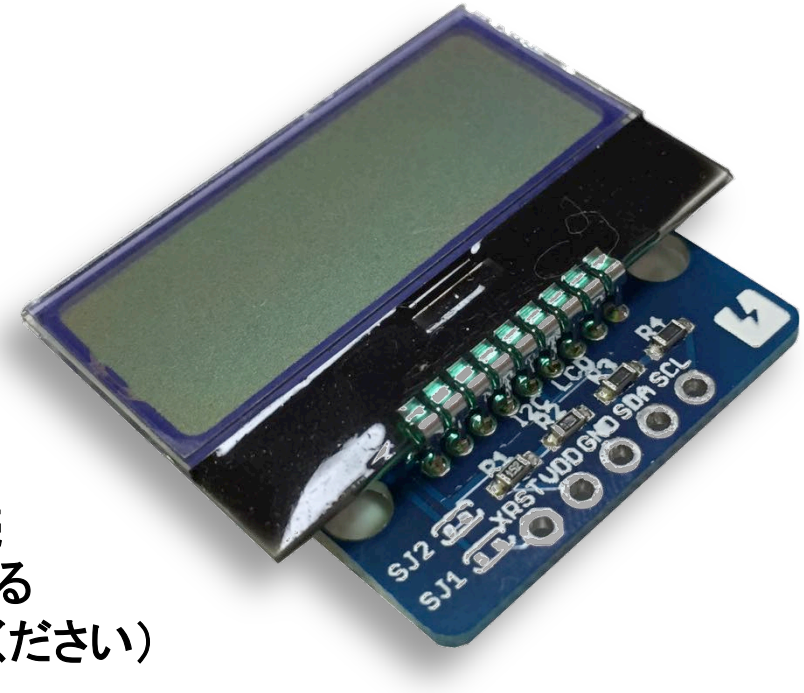
- 英数字、カタカナを表示するディスプレイ
- I2C接続
- 製品WEBサイトを見て使い方を確認

https://www.switch-science.com/catalog/1407/





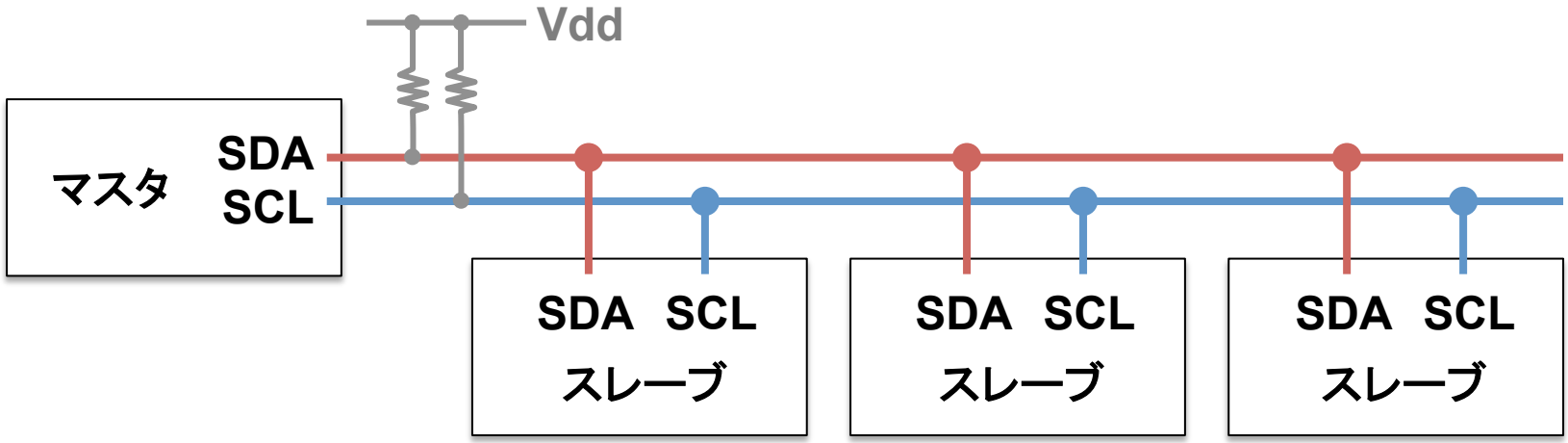
ショートさせない
ように注意！



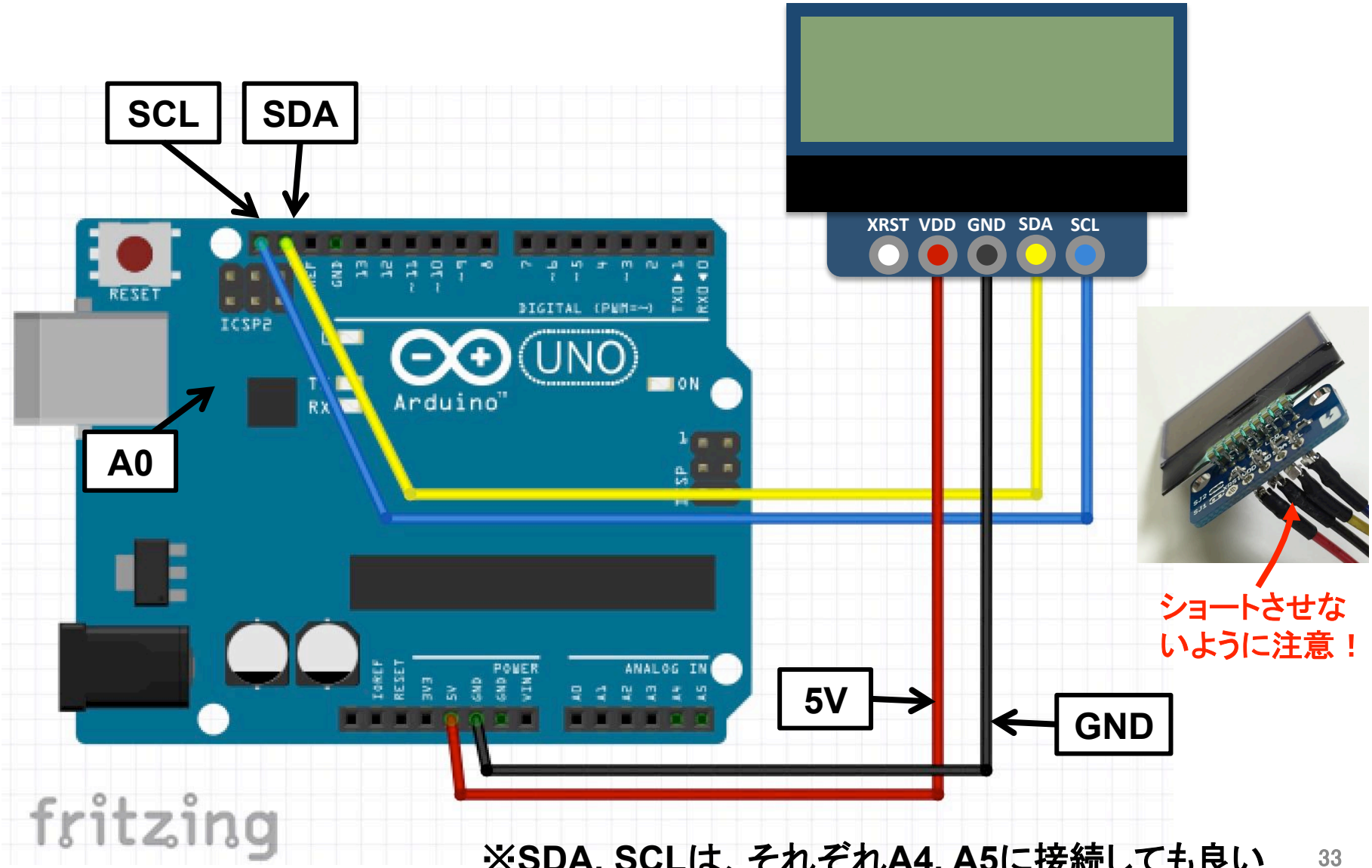
実習では「スルホール用テストワイヤ」で接続
普通は「ピンヘッダ」をハンダ付けして使用する
（できる人は安全に気をつけて挑戦してみてください）

I²Cデバイス

- I²C(Inter-Integrated Circuit)は、マイコンと周辺装置の間のシリアル通信方式
- 複数のセンサやディスプレイに対し、1組(2本)の信号線で入出力ができる
- Wireライブラリを使う；　#include <Wire.h>
- デバイスのアドレスを指定して、コマンドを送る



LCDを使う回路



※SDA, SCLは、それぞれA4, A5に接続しても良い

LCDに文字列・数値を表示するプログラム 1/3

```
#include <Wire.h>
#define I2Cadr 0x3e  //LCDのアドレス(固定）

// デフォルト設定で SDAは18番 SDAまたはA4
//                 SCLは19番 SCLまたはA5

void setup() {
  delay(500);
  Wire.begin();
  lcd_cmd(0x38);  lcd_cmd(0x39);  lcd_cmd(0x14);
  lcd_cmd(0x70);  lcd_cmd(0x56);  lcd_cmd(0x6c);
  delay(200);
  lcd_cmd(0x38);  lcd_cmd(0x0c); lcd_cmd(0x01);
  delay(2);
}
```

参考資料（次々ページにリンク掲載）にならって初期設定

※ lcd_cmd は次々ページで定義

LCDに文字列・数値を表示するプログラム


```
void loop() {
  // 文字列
  lcd_setCursor(0, 0);
  lcd_printStr("FMS");

  // 整数
  int  idata = 2016;
  char intStr[4];
  sprintf(intStr, "%4d", idata);
  lcd_setCursor(4, 0);
  lcd_printStr(intStr);

  // 浮動小数点
  float fdata = 3.1415;
  char floatStr[8];
  dtostrf(fdata, 6, 2, floatStr);
  lcd_setCursor(0, 1);
  lcd_printStr(floatStr);
  delay(100);
}
```

カーソル位置設定（x座標, y座標）

文字列表示

整数を文字列に変換

浮動小数点数を文字列に変換

※ lcd_setCursor, lcd_printStr は次ページで定義

sprintf（文字配列,フォーマット文字列,変数）;

dtostrf（変数,全体の桁,小数点以下の桁,文字列配列）;

LCDに文字列・数値を表示するプログラム


```
// コマンドを送信する
void lcd_cmd(byte x) {
  Wire.beginTransmission(I2Cadr);
  Wire.write(0b00000000);
  Wire.write(x);
  Wire.endTransmission();
}
// 文字の表示
void lcd_printStr(const char *s) {
  Wire.beginTransmission(I2Cadr);
  while (*s) {
    if (*(s + 1)) {
      Wire.write(0b11000000);
      Wire.write(*s);
    } else {
      Wire.write(0b01000000);
      Wire.write(*s);
    }
    s++;
  }
  Wire.endTransmission();
}

// 表示位置の指定
void lcd_setCursor(byte x, byte y)
{
  lcd_cmd(0x80 | (y * 0x40 + x));
}
```

続きの文字がある場合

最後の文字の場合

送信文字のポインタを進める

参考資料：http://akizukidenshi.com/download/ds/xiamen/AQM0802.pdf
これを参考に、初期化やコマンド送信、データ送信の処理を書く。
製品Webサイトの製作事例、サンプルコードを参考にして使い方を把握する。

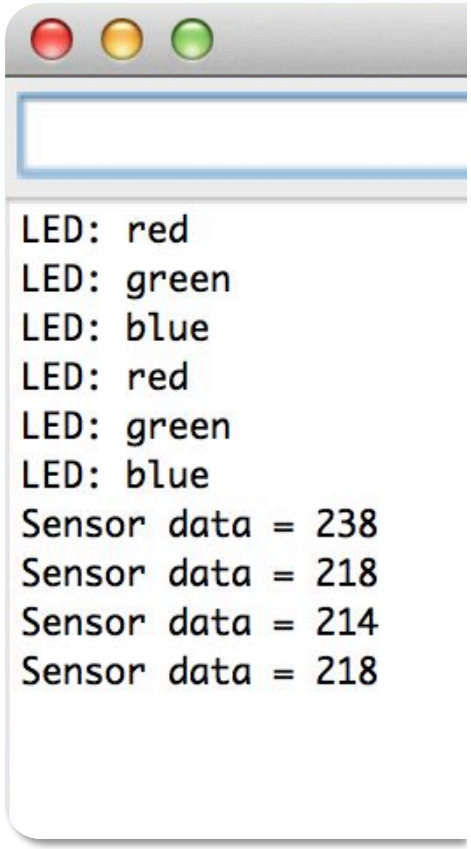
I2Cデバイスを扱う命令

- Wire.begin()
  - Wireライブラリを初期化してマスターとして接続
- Wire.beginTransmission(address)
- Wire.write(値)
- Wire.endTransmission()
  - addressで指定したデバイスに送信開始。write()でデータをキューに送り、endTransmission()で送信を実行
- Wire.requestFrom(address, count), Wire.available(), Wire.onReceive(handler), Wire.onRequest(handler), Wire.read()
  - デバイスからのデータの受信

※詳しくはリファレンスで確認

練習問題1

- コマンド入力によって以下の反応を返すプログラムにしてみよう
  - 「**r**」という文字を送信したらカラーLEDが赤く点灯して「LED: red」という文字列を返す
  - **同様に**「**g**」、「**b**」という文字を送信したらカラーLEDが緑、青に点灯して「LED: green」、「LED: blue」という文字列を返す
  - 「c」という文字を送信したら、光センサの値を「sensor data= ???」 という形式で返す
  - 「t」という文字を送信したら、摂氏温度を「temp= ???」 という形式で返す



「rgbrgbr」と送信すると、どうなるだろう？

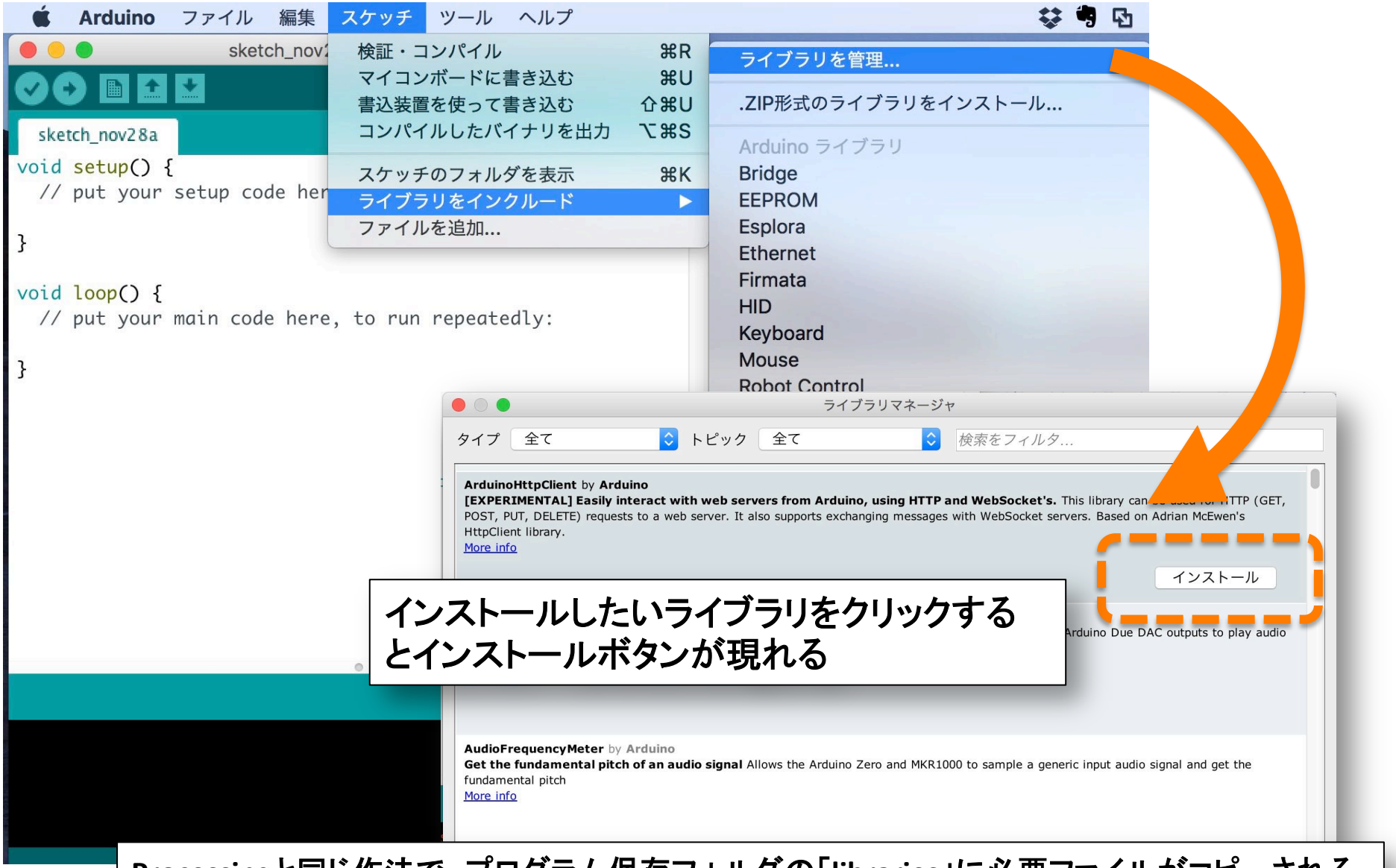
練習問題2

超音波距離センサーとLCDモジュールを組み合わせて、対象物までの距離を液晶ディスプレイに表示する装置を作ってみよう

- LCDモジュールを使うプログラムの loop() の中に、超音波距離センサで距離を計測する処理を追加する。
- 超音波距離センサ、LCDモジュールを使用するための初期化処理、loop()の中で行うべき処理を整理して並べる。

（おまけ）距離計ができたら、温度センサを追加して、温度も液晶に表示してみよう。

（参考）新しいライブラリの追加



インストールしたいライブラリをクリックするとインストールボタンが現れる

Processingと同じ作法で、プログラム保存フォルダの「libraries」に必要ファイルがコピーされる