基礎演習 E 2020

[I/O workshop 2020] -Arduino-

2020.06.12. kazumi wada

ハードウェアで「スケッチ」しよう。 ー Arduino のススメ

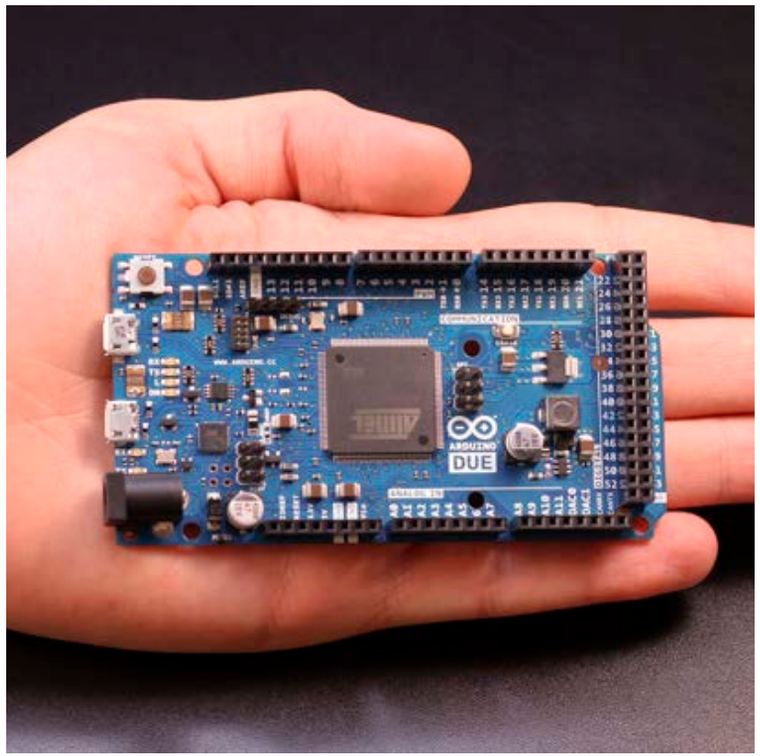
http://www.arduino.cc/

Arduino とは、フィジカルコンピューティングを実現する環境の一つ。
Arduino を利用することにより、センサやアクチュエータをパソコンに接続し、専用のプログラムを筐体にアップロードしてマイコンとしての制御が可能な他、Flash や Processing といった使い慣れているプログラミング環境からも利用できます。
Wiring や Arduino が登場した背景には、フィジカルコンピューティング (Physical Computing) という考え方があります。これは、ニューヨーク大学の ITP(InteractiveTelecommunicationsProgram、http://itp.nyu.edu/) でインタラクションデザインを教えるための方法の 1 つとして考案されたものです。現在では、『MakingThingsTalk』の著者としても知られる TomIgoe が中心的な役割を果たしています。
フィジカルコンピューティングは、コンピュータが理解したり反応したりできる人間のフィジカルな表現の幅をいかに増やすか、ということを目的とした教育プログラムで、デザインやアート教育の 1 つの分野として定着しています。エンジニアリングの専門教育を受けていない人に対して、コンピュータや電子回路に関する原理原則を教えるところから始め、「人々がいかにコンピュータとコミュニケーションし得るか？」について考え直すことを提案します。

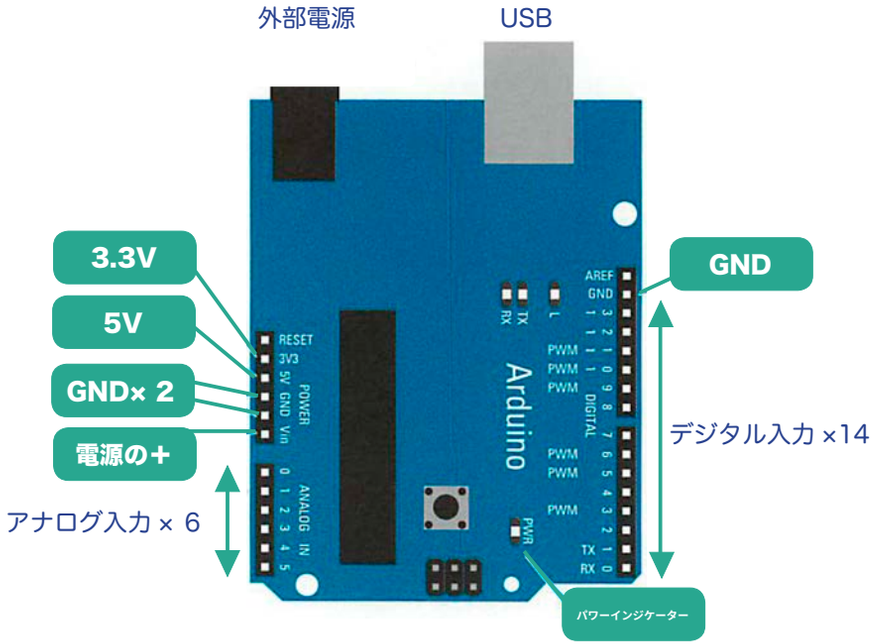
ハードウェアで「スケッチ」しよう。 ー Arduino のススメ

http://www.arduino.cc/

I/O モジュール



Arduino ボードの各部は次の図のようになっています。ボードには 6 個のアナログ入力ピン、14 個のデジタル入出力ピン、電源 (+5V、+3.3V と GND) などがあります。GND はボード上に全 3 カ所ありますが、いずれも働きとしては同じです。配線の都合で最も便利なところを使います。



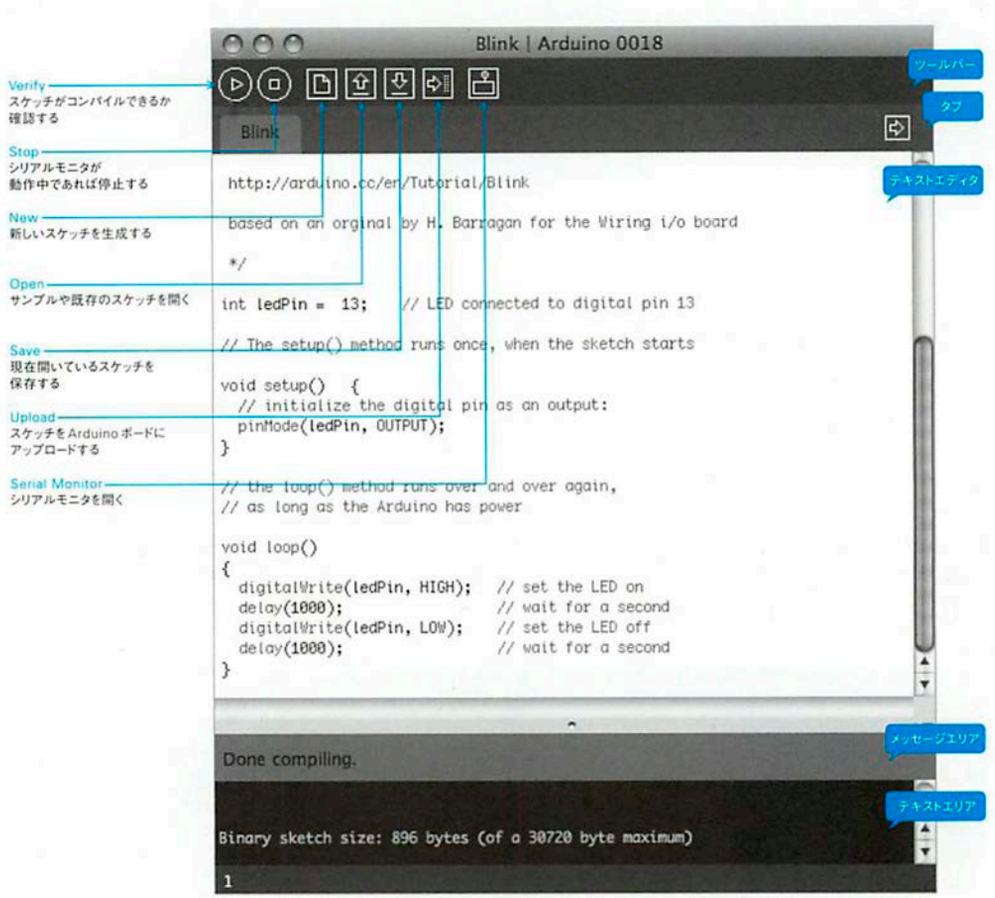
Arduino ボードの各部分の説明図

Arduino を制御する１：「Arduino IDE」

http://www.arduino.cc/

単独で動かす場合

「Arduino」app / exe のアイコン



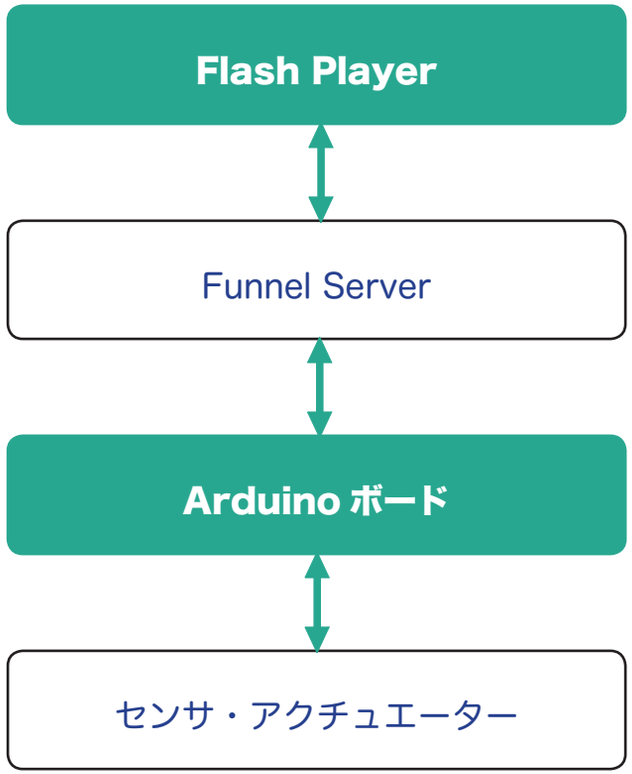
ARDUINO

Buy | Download | Getting Started | Learning | Reference | Products | FAQ | Contact Us

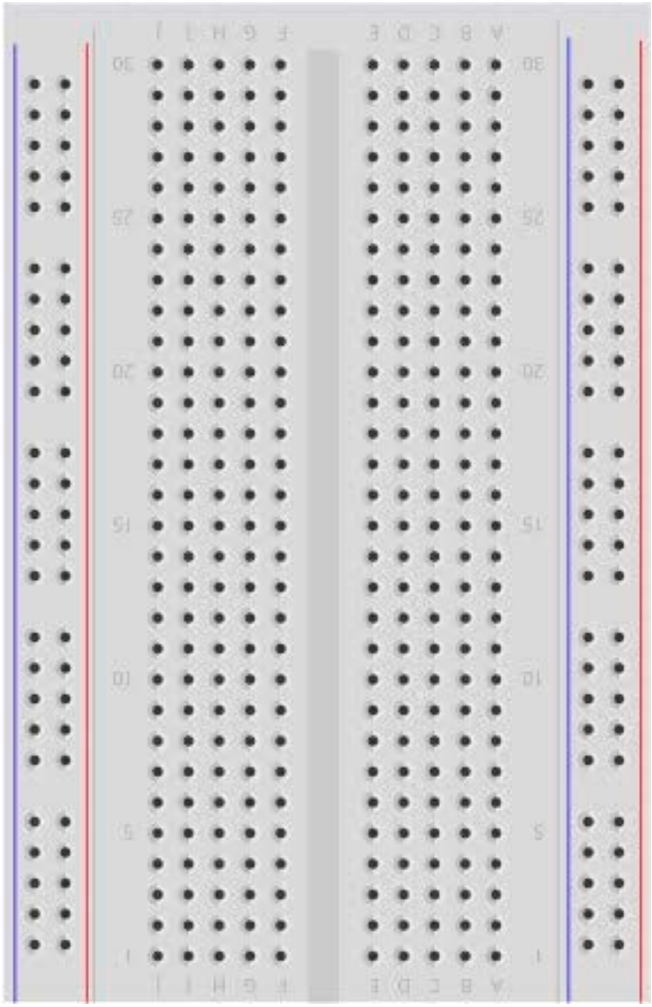
Arduino を制御する２：「Funnel」ライブラリ

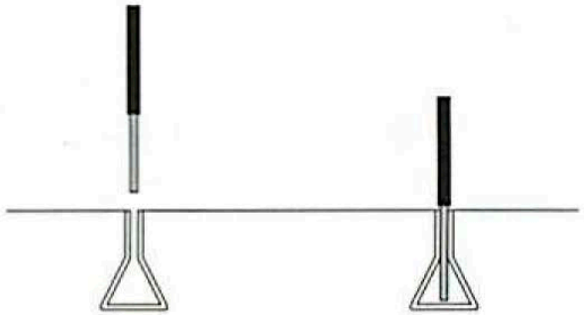
http://www.arduino.cc/

funnel-1.0-r801 /server /macosx.dmg Funnel Server
windows.zip



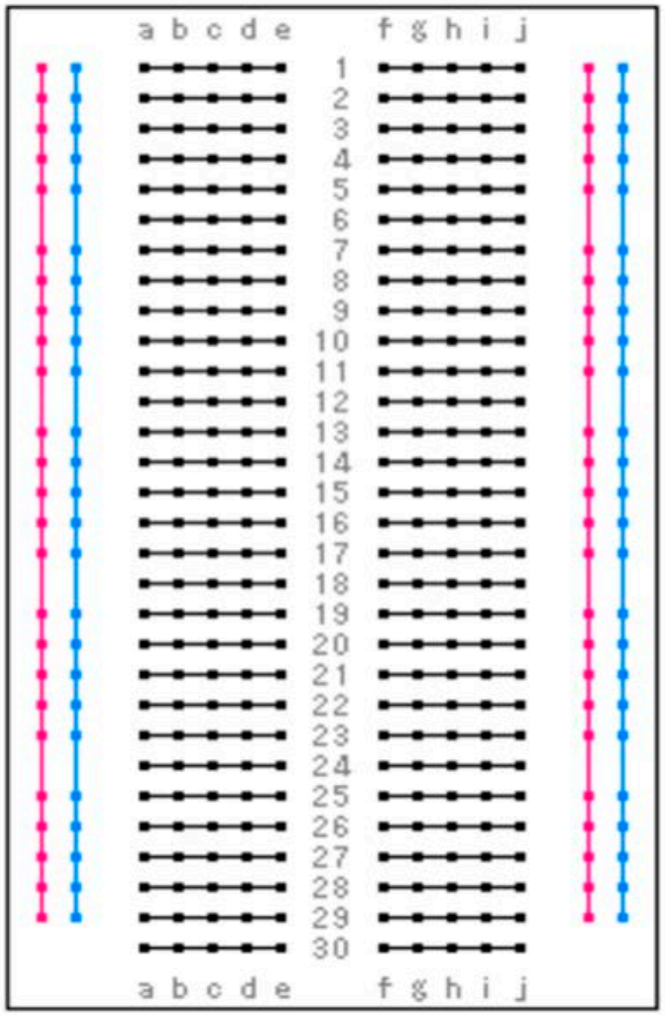
ブレッドボードとジャンプワイヤ。

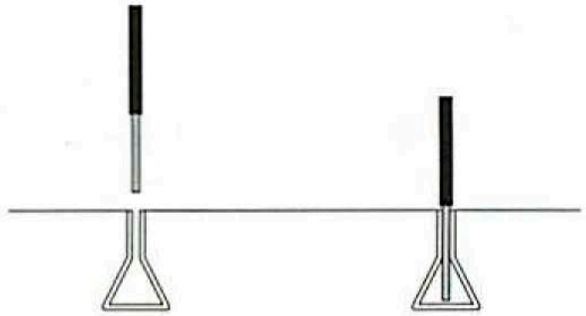




ブレッドボードの構造図。
ブレッドボードの各ピンがどのように接続されているかを上面から見た図 (左) と内部の構造図 (右)。
最も小型の EIC-15010 にはないが、EIC-801 など一般的なブレッドボードには左右に赤と青 (または赤と黒) で示される電源用のラインがある。

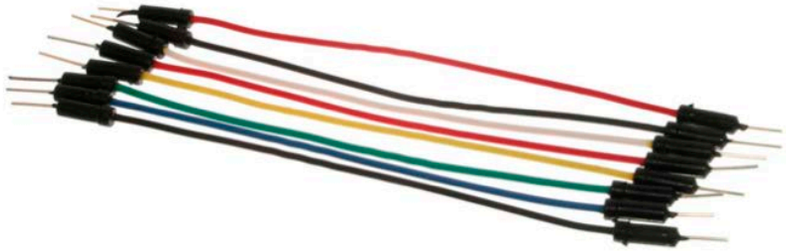
ブレッドボードとジャンプワイヤ。

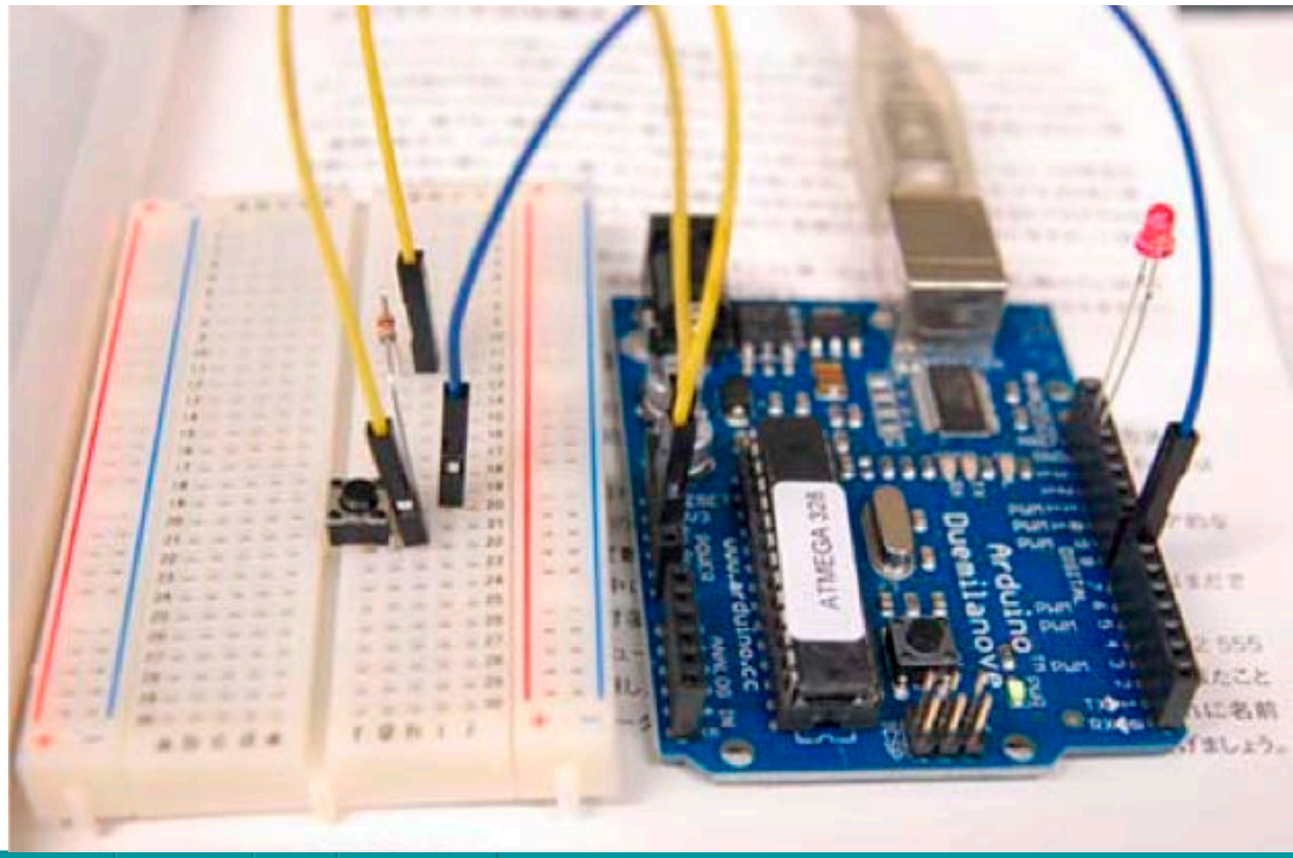




ブレッドボードの構造図。
ブレッドボードの各ピンがどのように接続されているかを上面から見た図 (左) と内部の構造図 (右)。
最も小型の EIC-15010 にはないが、EIC-801 など一般的なブレッドボードには左右に赤と青 (または赤と黒) で示される電源用のラインがある。

ブレッドボードとジャンプワイヤ。





間違えると何かしら壊れます。

必ず回路を組み終わってから、USB ケーブルを挿してください。正しく回路が組まれていれば､LED が点灯するはずです。もし､LED が点灯しない場合には接続が間違っています。その場合にはすぐに USB ケーブルをぬき、接続を確認してください。

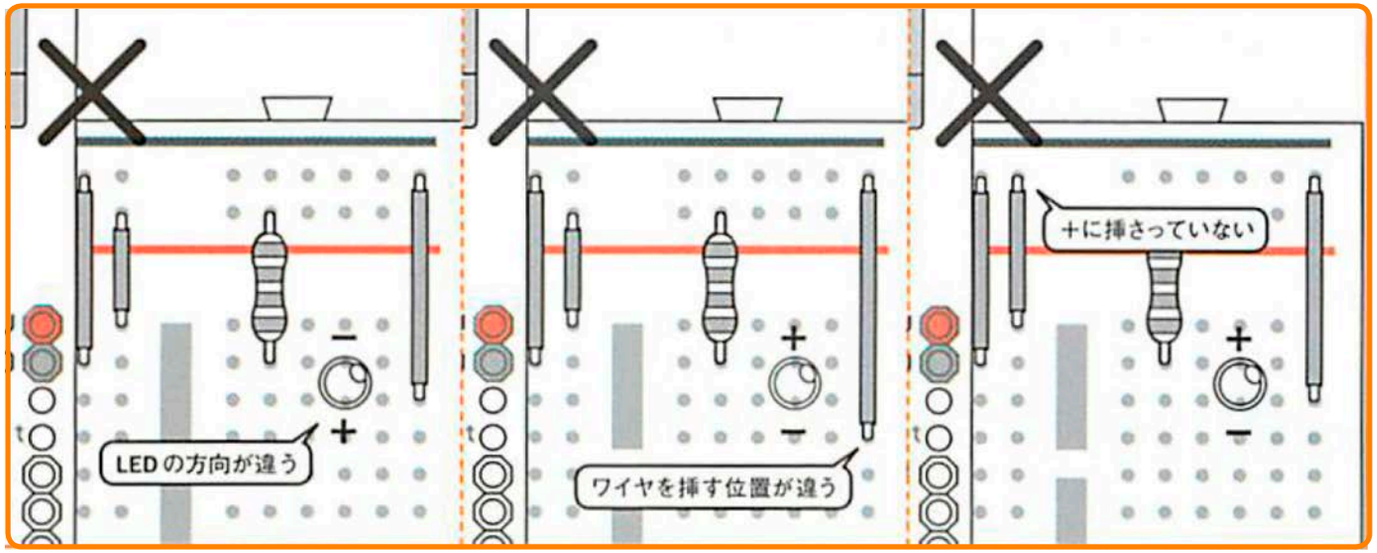
①電池の＋ー を直接接続しない

②部品の足同士が接触しないように位置関係を工夫する

③部品の足はブレッドボードにしっかり挿す

④壊れやすい部品の扱いに注意 ・部品の足を曲げる力を部品自体に伝えない

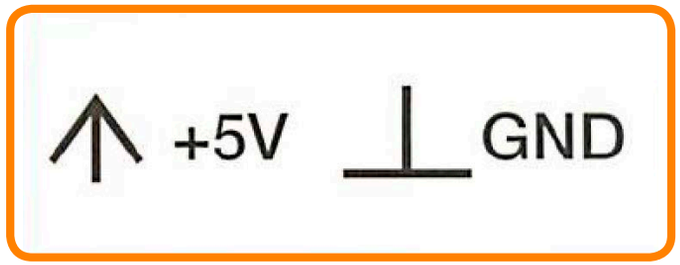
・部品の足を曲げたり伸ばしたりを繰り返さない



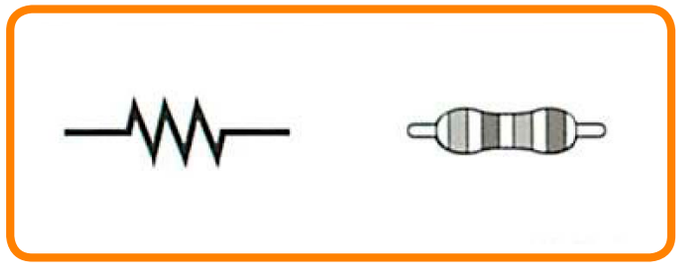
よくある失敗例

回路図の読み方。

■電源の回路シンボル

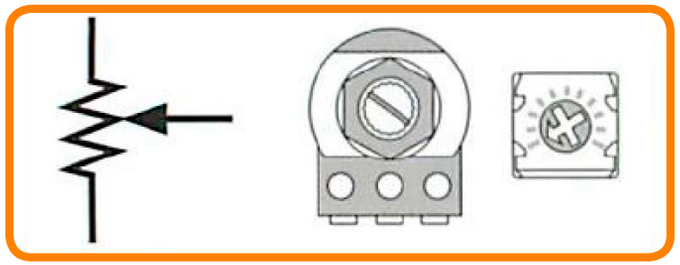


■抵抗器



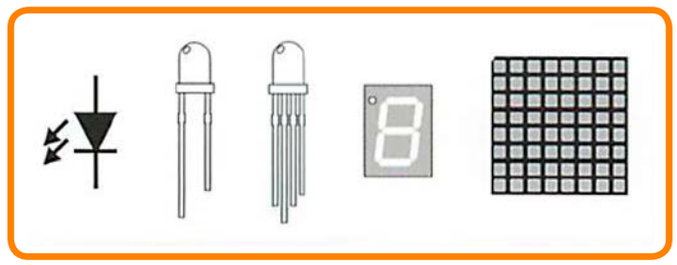
抵抗器の回路図シンボルと部品例

■可変抵抗器（ボリューム）



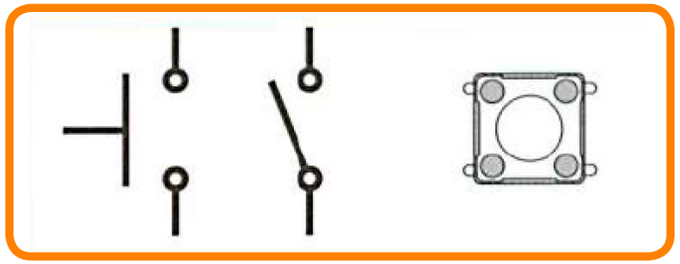
可変抵抗器（ボリューム）の回路図シンボルと部品例（丸形、半固定）

■ LED



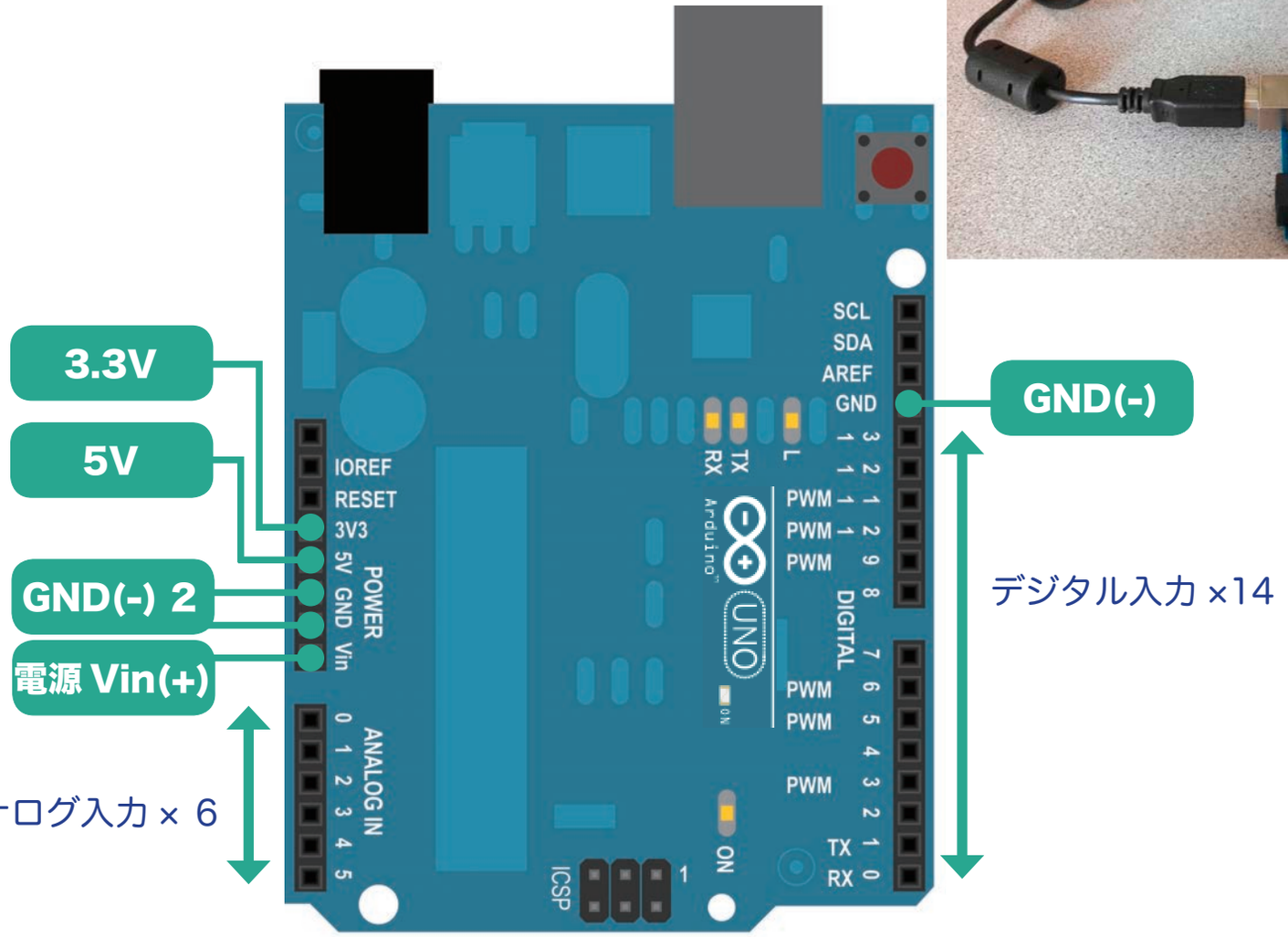
LED の回路図シンボルと部品例（単色、フルカラー、７セグ、マトリクス）

■スイッチ



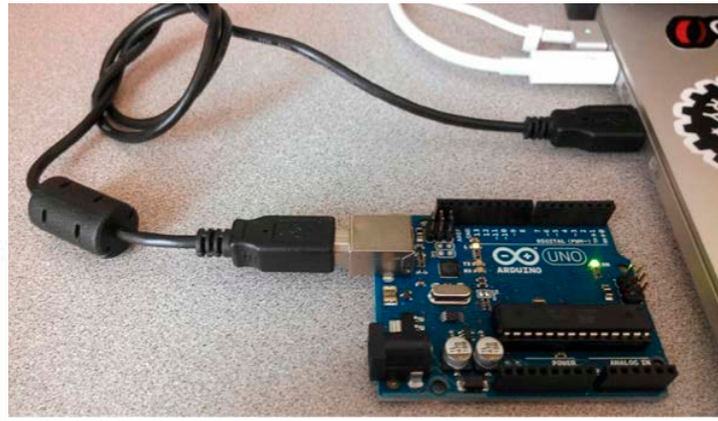
スイッチの回路図シンボルと部品例

電源の接続。



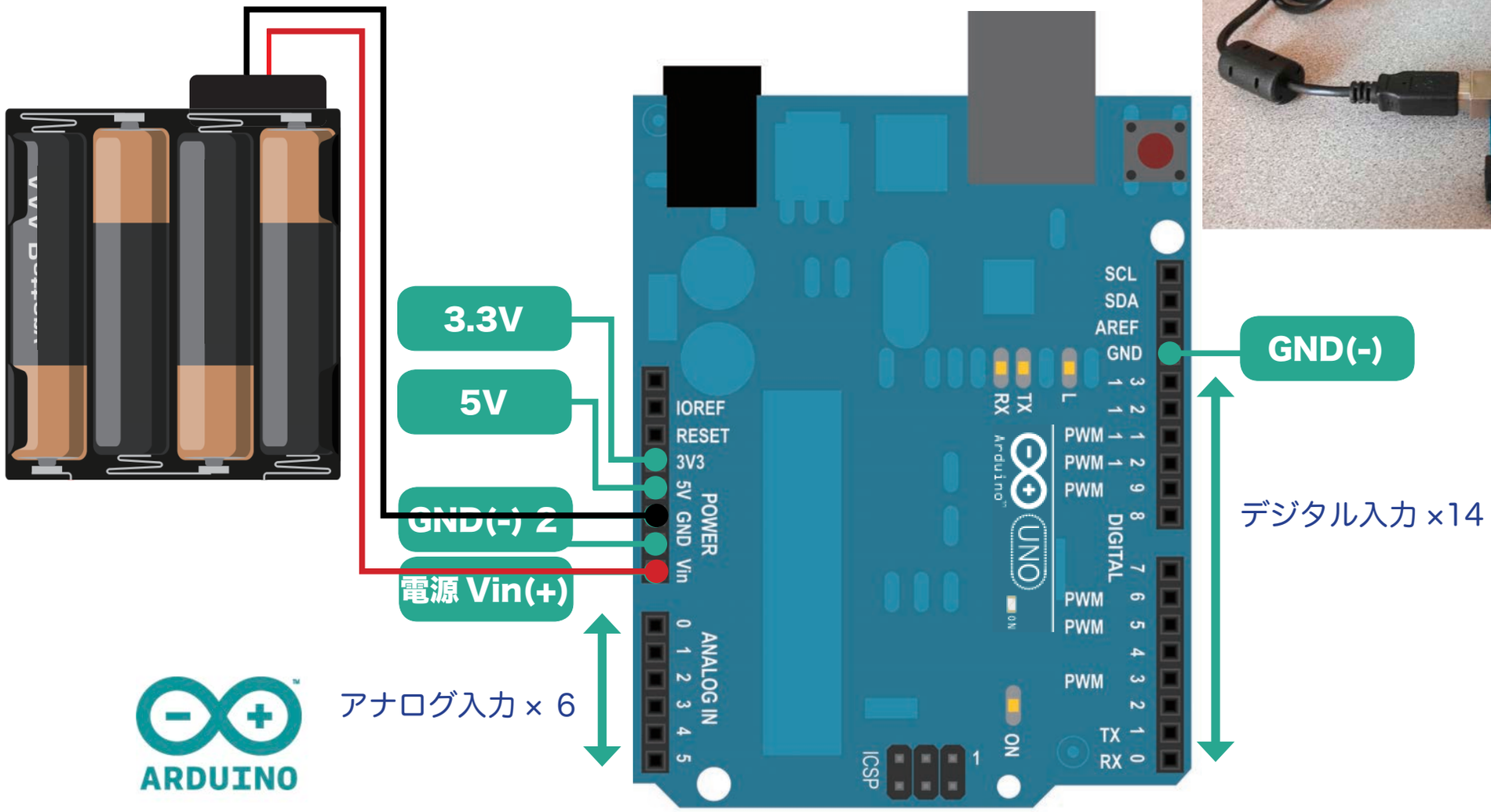
ARDUINO

電源の接続。



ARDUINO

電源の接続。


外部電源
USB ケーブル口
3.3V
5V
GND(-) 2
電源 Vin(+)
GND(-)
デジタル入力 ×14
アナログ入力 × 6
IOREF
RESET
3V3
5V
GND
Vin
POWER
ANALOG IN
SCL
SDA
AREF
GND
DIGITAL
PWM
TX
RX
L
ON
ICSP
UNO
ARDUINO

実際に回路を組んでみよう。

図１：LED を直接電源に接続する

◯ Arduino ボード：１個　◯ジャンプワイヤ：適量
◯ブレッドボード：１個　◯ LED：１個

USB ケーブルが抜かれていることを確認したら、Arduino ボードとブレッドボードを２本ジャンプワイヤでつなぎ、LED をつなぎます。LED には極性がありますが、足の長さで区別できます。足の長い方がプラス側、短い方がマイナス側ですので、この図では上側にプラス側が来るように差し込みます。次に、Arduino ボードと PC を USB ケーブルでつなぐと、LED が点灯するはずです。

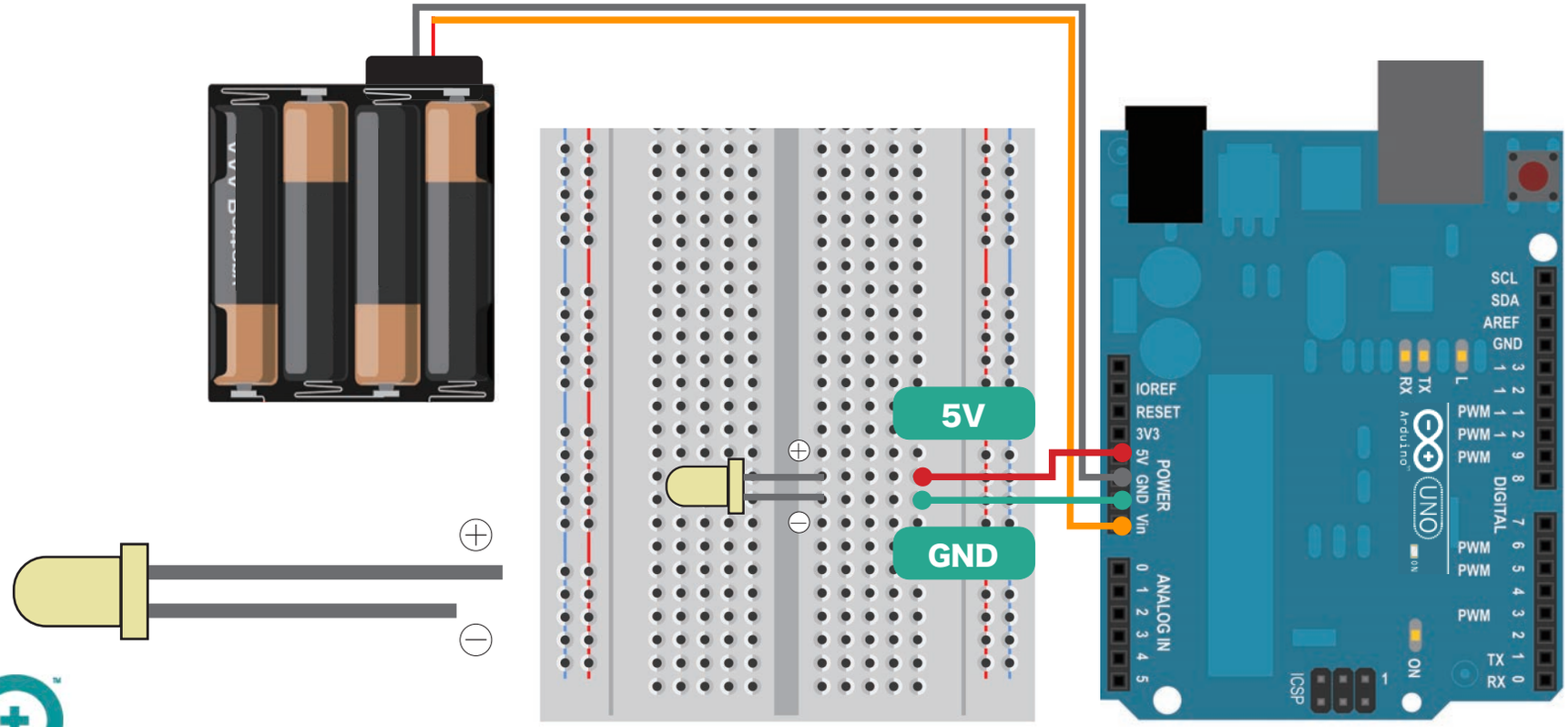


図１　Arduino ボードの電源に直接 LED を接続して点灯させる配線図。
LED は明るく点灯するが、短時間でかなり発熱して最終的には焼き切れてしまう.

実際に回路を組んでみよう。

図１：LED を直接電源に接続する

○ Arduino ボード：１個　○ジャンプワイヤ：適量
○ブレッドボード：１個　○ LED：１個

USB ケーブルが抜かれていることを確認したら、Arduino ボードとブレッドボードを２本ジャンプワイヤでつなぎ、LED をつなぎます。LED には極性がありますが、足の長さで区別できます。足の長い方がプラス側、短い方がマイナス側ですので、この図では上側にプラス側が来るように差し込みます。次に、Arduino ボードと PC を USB ケーブルでつなぐと、LED が点灯するはずです。

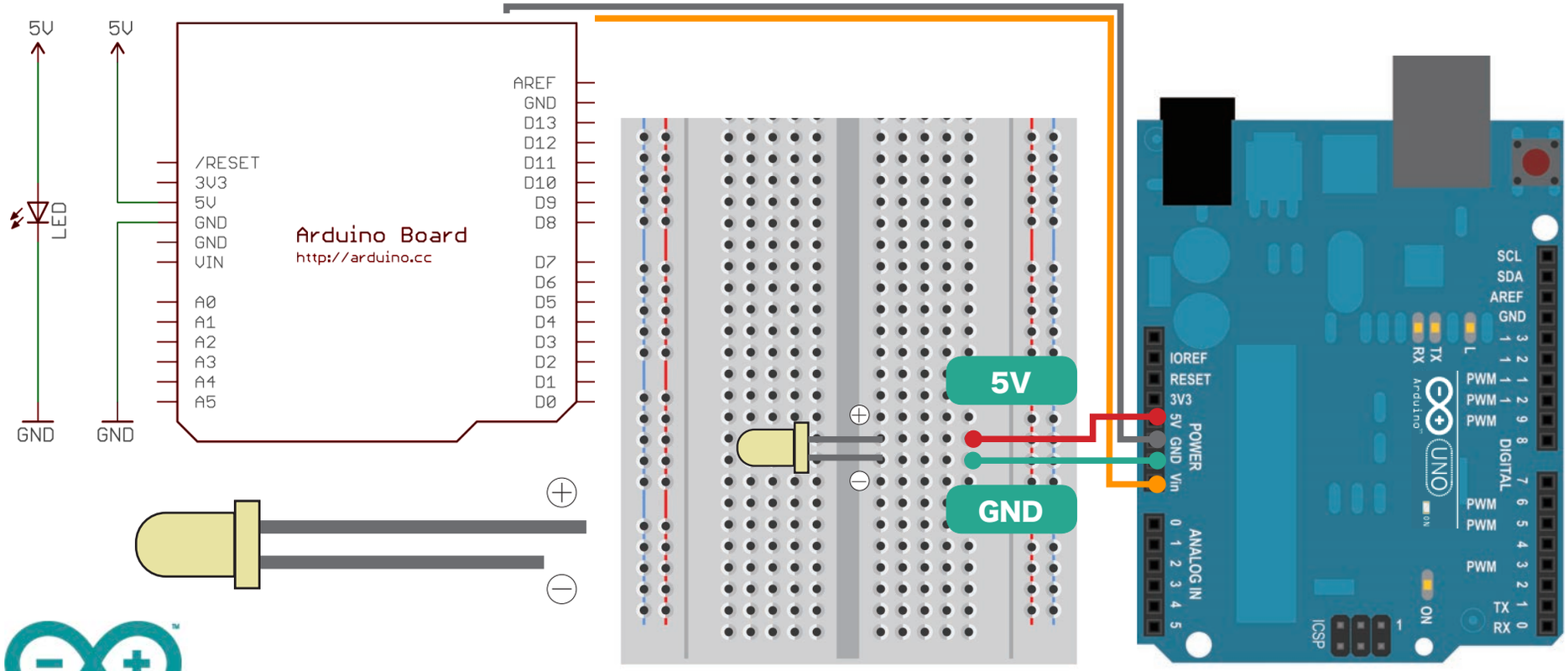


図１　Arduino ボードの電源に直接 LED を接続して点灯させる配線図。
LED は明るく点灯するが、短時間でかなり発熱して最終的には焼き切れてしまう.

実際に回路を組んでみよう。

図 1 ：LED を直接電源に接続する

○ Arduino ボード：１個　○ジャンプワイヤ：適量
○ブレッドボード：１個　○ LED：１個

次に LED に流れる電流が制限されるようにしましょう。具体的には、次の図にあるように 330Q の抵抗器 (燈燈茶金) を追加し、+5V →抵抗器 → LED → GND という流れになるようにします。
今度はきっと、１時間経っても２時間経っても LED は安定して点灯し続けていると思います。

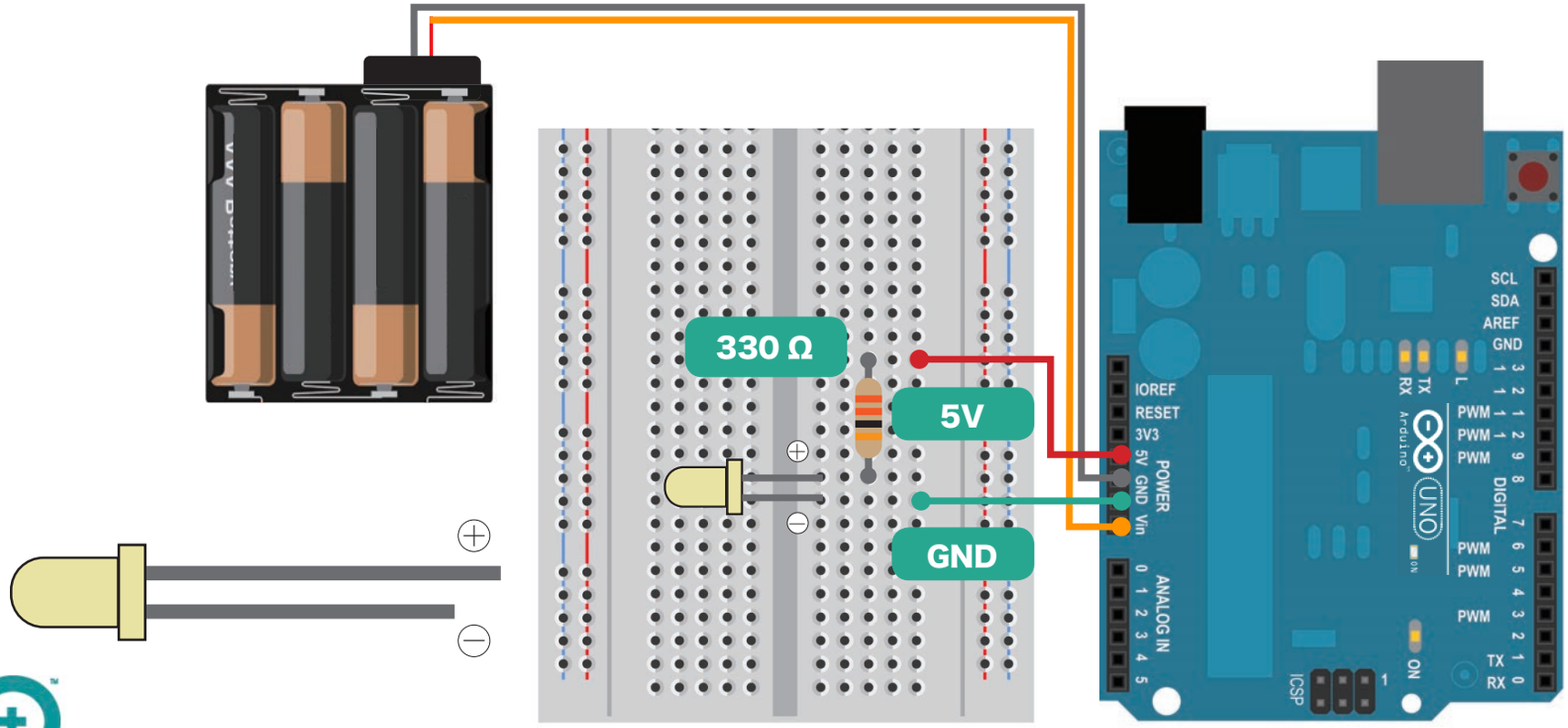


図 2　電流制限用の抵抗器を用いて LED を点灯させる配線図。抵抗器によって LED に流れる電流が適切に制限されることにより、LED が焼き切れることなく点灯し続ける。

スイッチで LED をオン／オフする。

図３：LED をスイッチでコントールする

- ○ Arduino ボード：１個
- ○ LED：１個
- ○ブレッドボード：１個
- ○抵抗器：１本 (330 Ω)
- ○ジャンプワイヤ：適量
- ○タクトスイッチ：１個

回路にスイッチを追加してみましょう。
この回路では、スイッチを押したときだけ +5V から GND までの回路が接続され、LED が点灯します。スイッチを離しているときには、回路が途中で切断されているため、LED には電流が流れず、点灯しません。

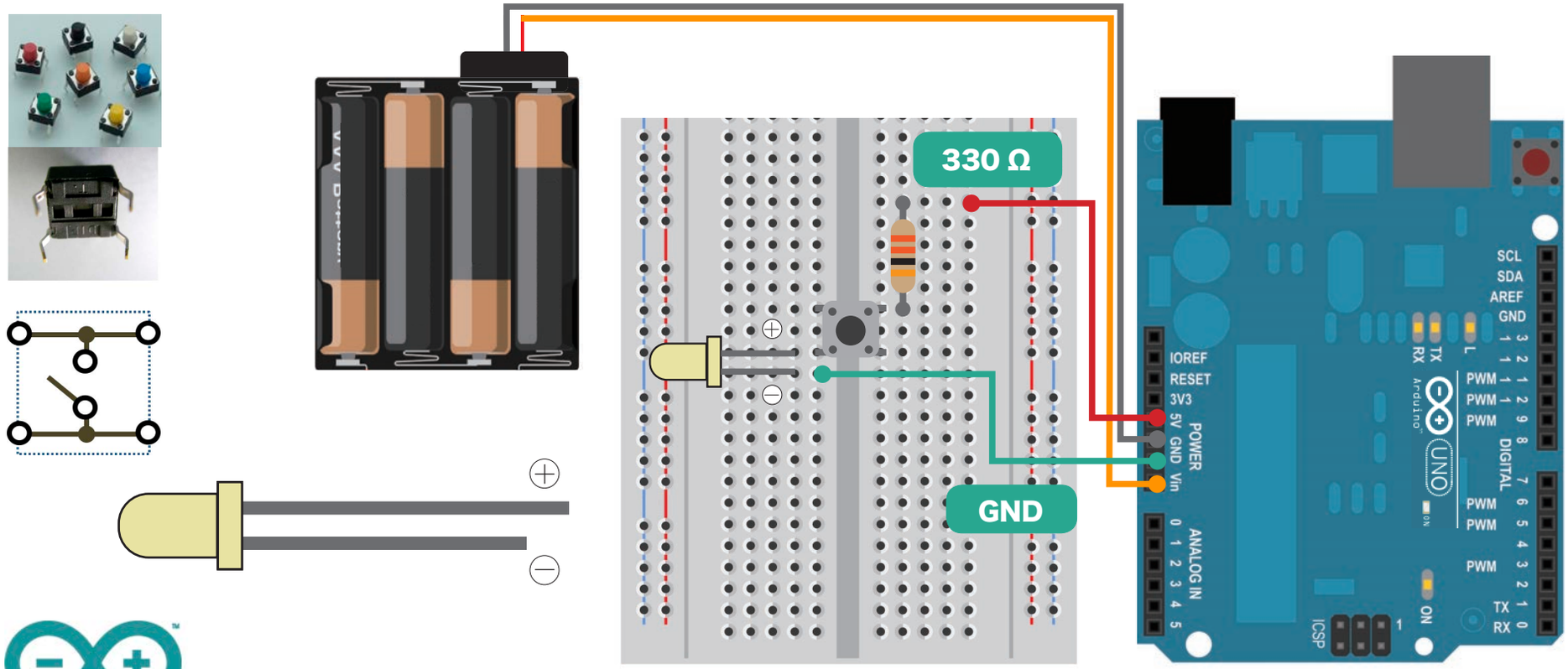


図３　図２の回路にタクトスイッチを追加した配線図。タクトスイッチは、カニのように左右に足が出ている状態でブレッドボードに差し込む正しく回路を組むと、スイッチを押している間だけ LED が点灯するようになる。

PWM によるアナログ出力。

図５：LED の明るさを変更する

- ○ Arduino ボード：１個
- ○ LED：１個
- ○ブレッドボード：１個
- ○抵抗器：１本 (330 Ω)
- ○ジャンプワイヤ：適量

02Analog_Fading/Fading

Arduino では、PWM を使って連続的に出力を変えるために analogWrite が用されています。digitalWrite はその名の通りオン／オフの２段階だけですが、analogWrite は 0 から 255 までの 256 段階で連続的に変化させることができます。これを使ったサンプルが次の Analog/Fading です。次のように配線してからアップロードしてどんなことが起きるか見てみましょう。

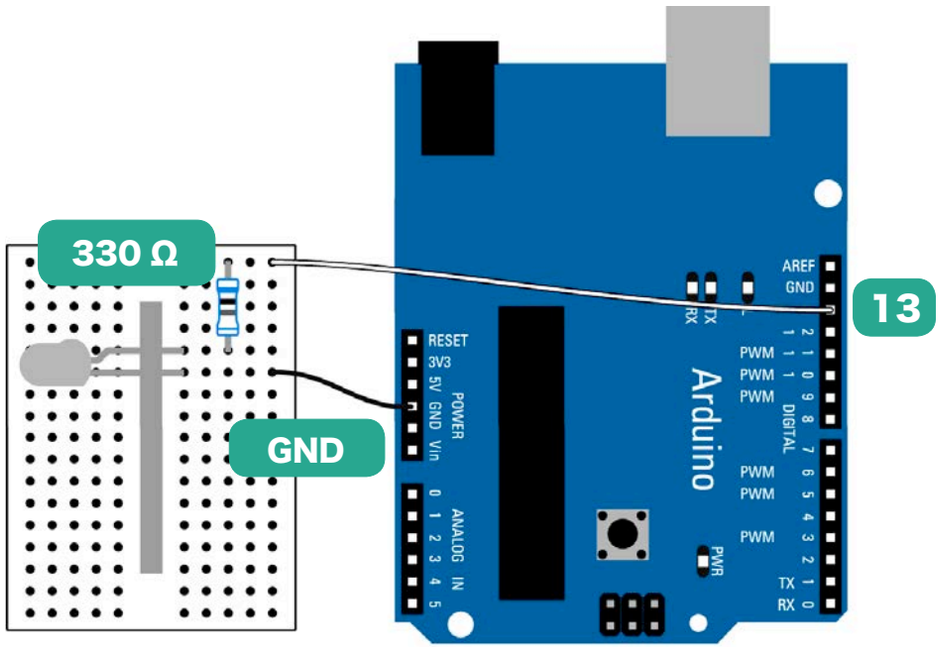


図５　デジタルピンの 13 番（D13）に抵抗器と LED を接続した配線図

Arduino によるデジタル制御。

図４：LED を点滅させる

○ Arduino ボード：１個　○ LED：１個
○ブレッドボード：１個　○抵抗器：１本 (330 Ω)
○ジャンプワイヤ：適量

01Digital_Blink/Blink

それでは、Arduino による制御を作業してみましょう。一旦 USB から Arduino をはずして、回路を繋ぎ変えます。スイッチをブレッドボードからはずして、Arduino 入力の右側の 13 番から LED の電気をもらうことにします。LED の足が長いプラス側、抵抗から 13 番につなぎ、LED の足が短いマイナス側はそのまま GND につなげておきます。
回路がつながったら、再び USB ケーブルを Arduino をつなぎます。

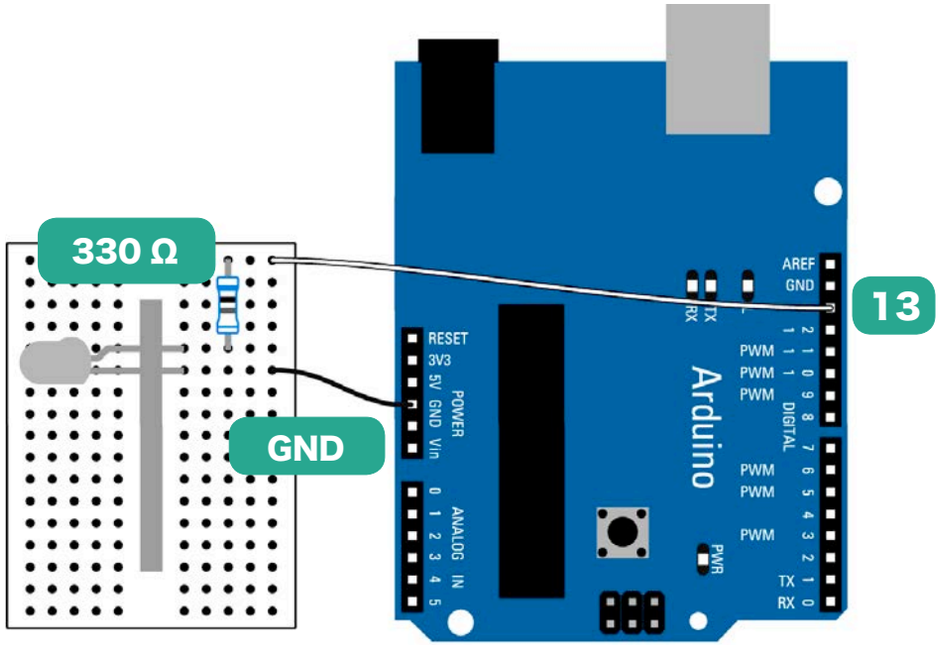
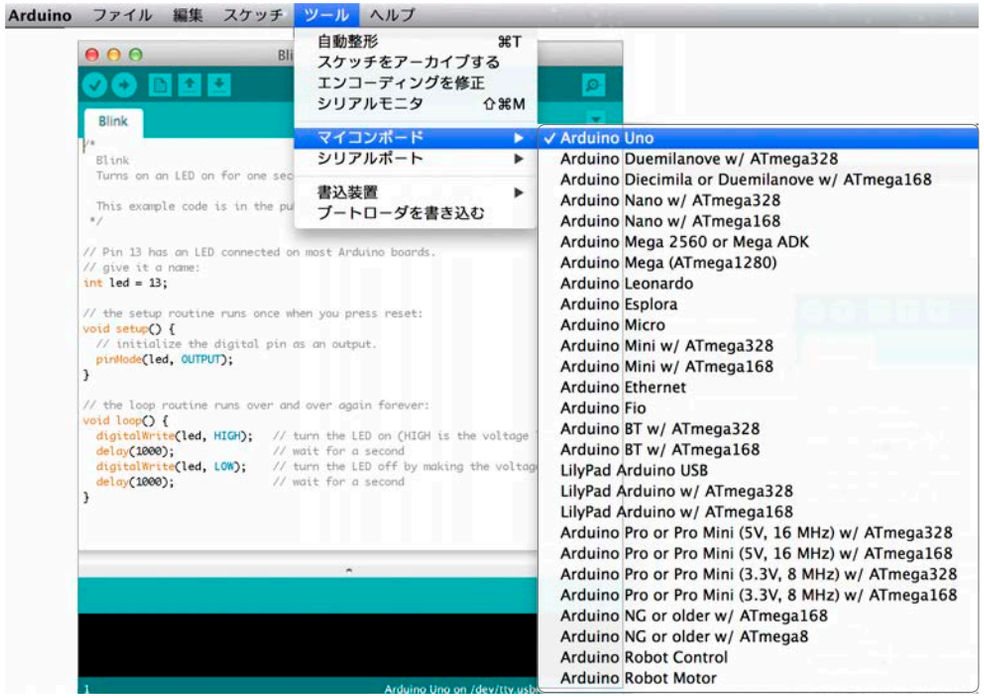
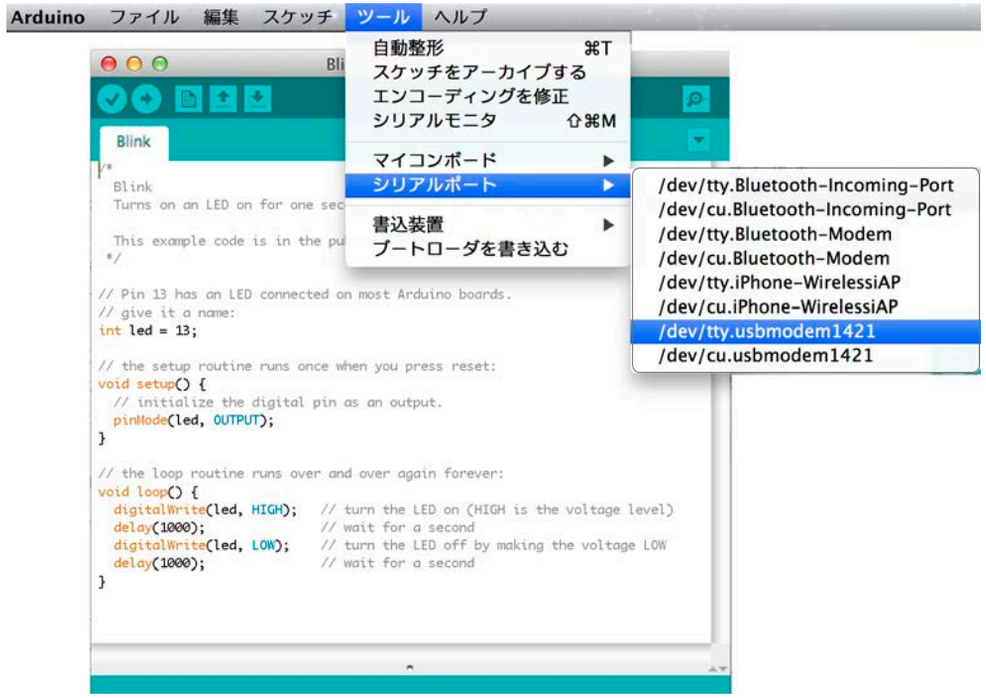


図４　デジタルピンの 13 番（D13）に抵抗器と LED を接続した配線図

マイコンボードとシリアルポートを確認。





USB ケーブルを Arduino につなげたら、プログラムを書き込むために、正しく PC が認識しているか確認します。ArduinoIDE の上のメニュー「ツール」からマイコンボードは「Arduino」にチェックが入り、シリアルポートは「dev/tty.usbmodem1421」といった USB ポートがチェックされているか確認します。チェックが入っていない場合や Bluetooth などがチェックされている場合は、USB ポートをチェックし直します。

スケッチの基本的な構造とデジタル出力。

01Digital_Blink/Blink

サンプル 14.1　Arduino(ファイル / スケッチの例 /Digital/Blink)

```
                                        //LED に接続したピンの番号
int ledPin=13:

                                        //setup は最初に 1 度だけ実行される
void setup(){
                                //LED に接続したピンのモードを OUTPUT( 出力 ) にセット
pinMode(ledPin,OUTPUT);
}
                        //loop は Arduino ボードの電源がオンであるかぎり、繰り返し実行される
void loop(){
                                //LED に接続したピンの値を HIGH にｾｯﾄして LED を点灯
digitalWrite(ledPin,HIGH);

                                                        //1000ms(1s) 待つ
delay(1000);

                                //LED に接続したピンの値を LOW にセットして LED を消灯
digitalWrite(ledPin,LOW);

                                                        //1000ms(1s) 待つ
delay(1000);
}
```

次に Arduino にプログラムを書き込んでみましょう。
今日の日付のフォルダの中から「01Digital Blink」 →「Blink」を開き、中の「Blink.ico」ファイルを開きます。
Arduino のウインドウ左上にある「Play」ボタンを押すと、Arduino 上の LED が点滅して、プログラムが書き込まれます。
書き込みが終了すると、LED が素早く点滅し、自動的にプログラムの動作を始めます。
このプログラムはデジタル制御なので、LED はスイッチのオン／オフのような、カチカチとはっきりした点滅を繰り返します。

デジタル出力とアナログ出力の違い。

02Analog Fading/Fading

```
int ledPin = 13;                                    //LED に接続したピンの番号
void setup()  {
                                                    //setup は最初に 1 度だけ実行される
}                                                   //setup では特に何もしません
void loop()  {
                    // 変数 fadeValue は 0 から始まって、255 になるまで 5 ずつ加算され
                                                    その間繰り返し処理されます。
  for(int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
                //analogWrite というメソッドで、ledPin(13) に 0 ～ 255 が送られます。
    analogWrite(ledPin, fadeValue);
    delay(30);                                      //30ms(0.03s) 待つ
  }
                    // 変数 fadeValue は 255 から始まって、0 になるまで 5 ずつ減算され
                                                    その間繰り返し処理されます。
  for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
                                  //analogWrite で、ledPin(13) に 255 ～ 0 が送られます。
    analogWrite(ledPin, fadeValue);
    delay(30);                                      //30ms(0.03s) 待つ
  }
}
```

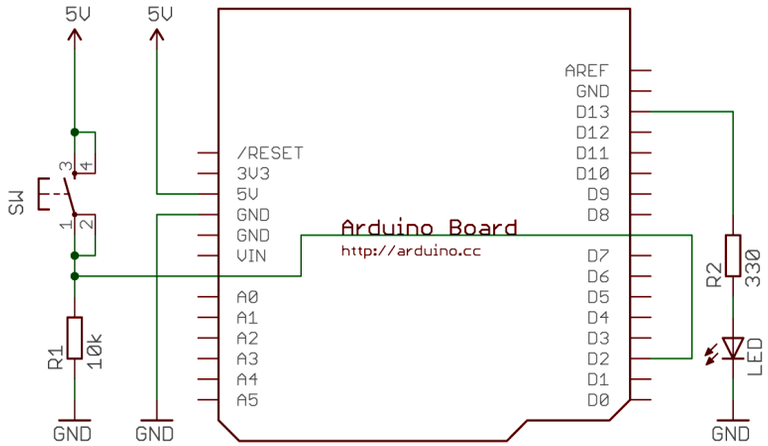
デジタル出力の場合は「digitalWrite」というメソッドで「HIGH」か「LOW」という 2 つの値を LED に送りますが、アナログ出力では for 文のように繰り返し処理を使って、その間 0 ～ 255 の細かく刻んだ値を「analogWrite」というメソッドで LED に送るので、徐々に明るくなったり、暗くなったりする動作になります。

デジタル入力。

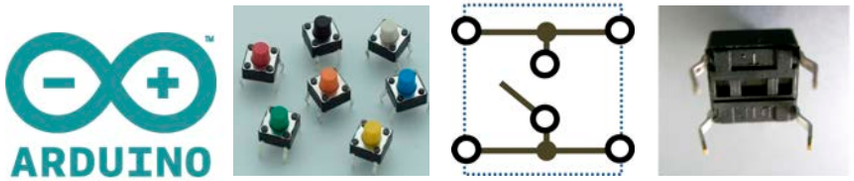
今度は入力を使ってみましょう。入力のサンプルは Digital/Button です。次のように回路を組んだら、スケッチをアップロードして動かしてみましょう。

図 6 ：ボタンで LED のオン／オフをコントールする

- ○ Arduino ボード：１個
- ○ LED：１個
- ○ブレッドボード：１個
- ○抵抗器：２本 (330 Ωと 10k Ω)
- ○ジャンプワイヤ：適量
- ○タクトスイッチ：１個



03Digital_Button/Button



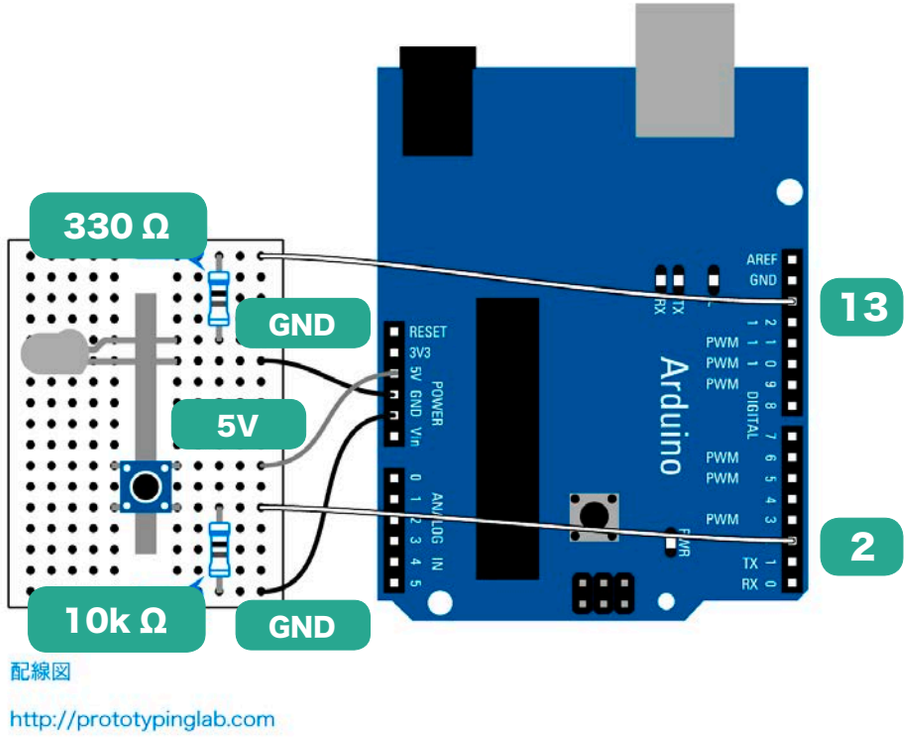


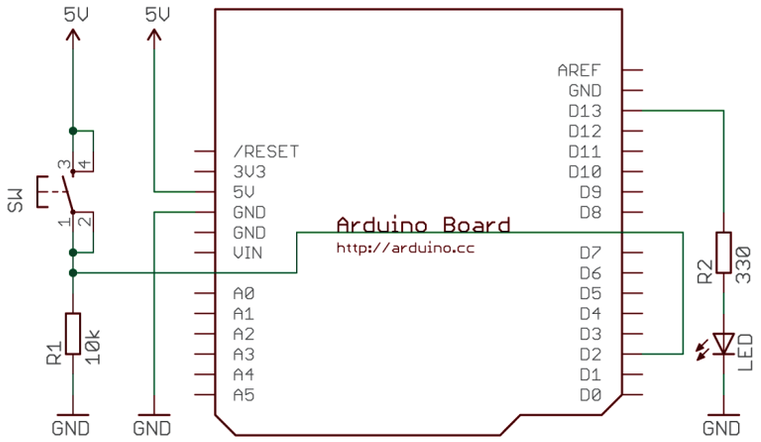
図 6　D2 にタクトスイッチと抵抗器を、D13 に抵抗器と LED を接続した配線図。タクトスイッチを押している間だけ LED が点灯する。

デジタル入力。

19 の回路が成功したら、スイッチをリードスイッチに変更してみましょう。
ON/OFF は、同じ形の磁石でできるようになります。

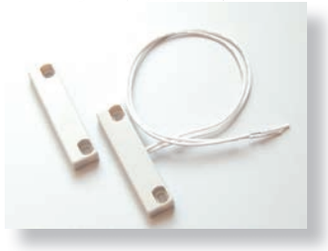
図 7 ：ボタンで LED のオン／オフをコントールする

- ○ Arduino ボード：１個
- ○ LED：１個
- ○ブレッドボード：１個
- ○抵抗器：２本 (330 Ωと 10k Ω)
- ○ジャンプワイヤ：適量
- ○リードスイッチ：１セット



03Digital_Button/Button

リードスイッチ



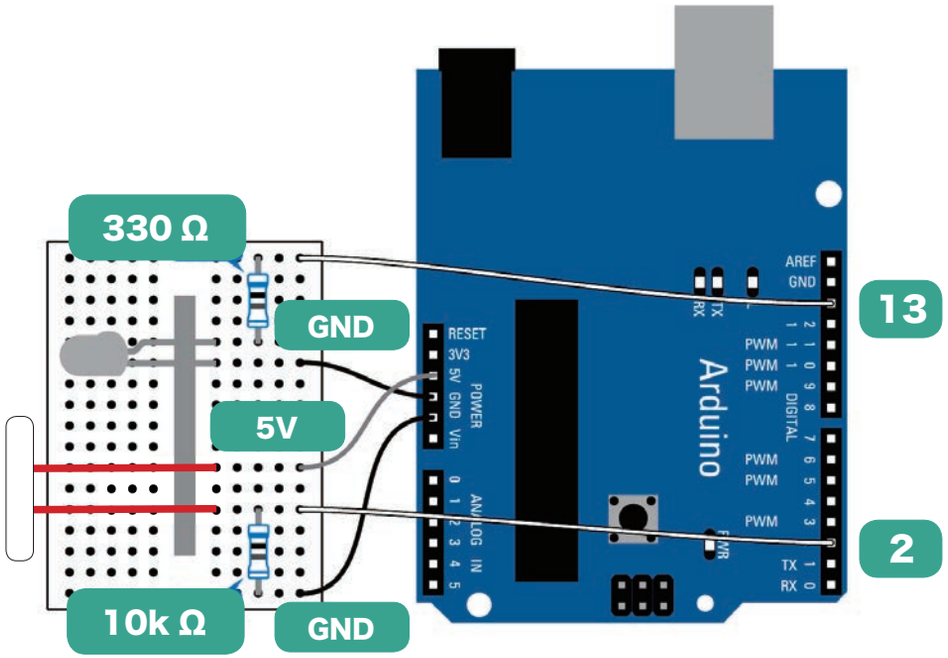
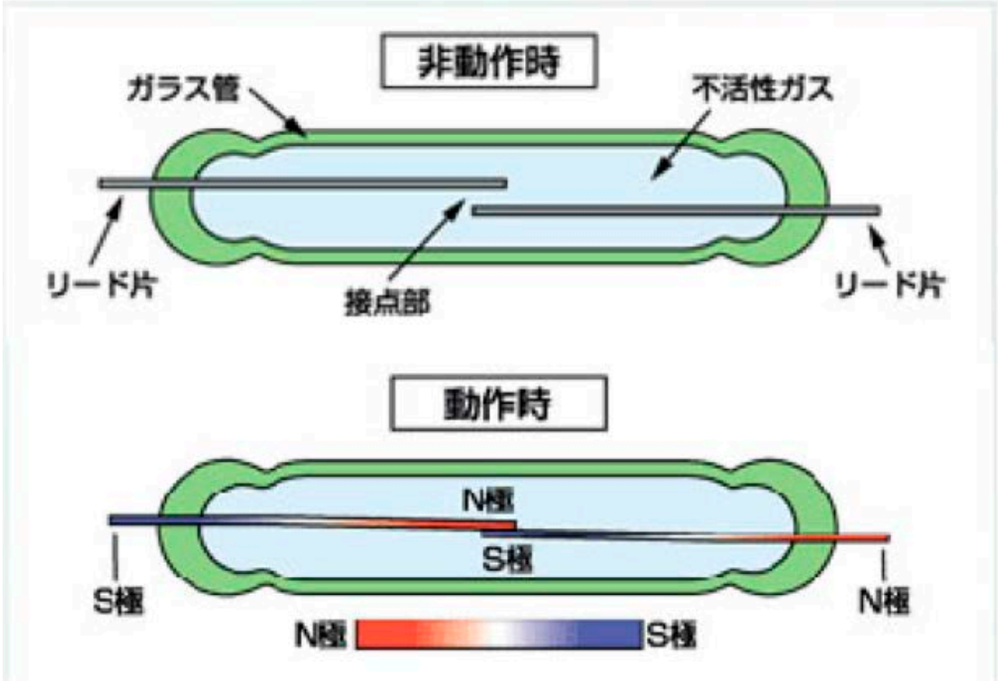
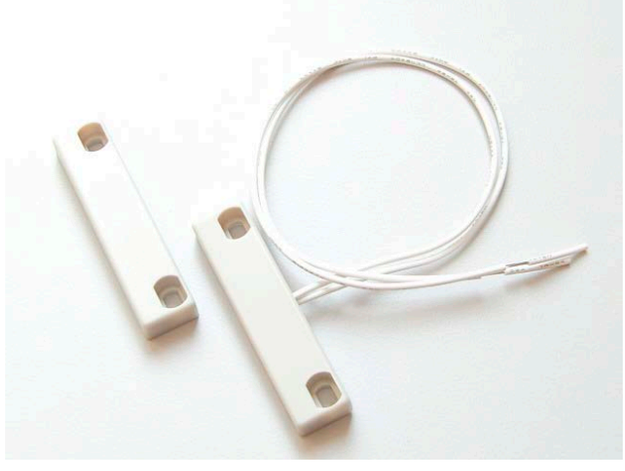


図 7　D2 につなげたタクトスイッチを、リードスイッチに置き換えた配線図。

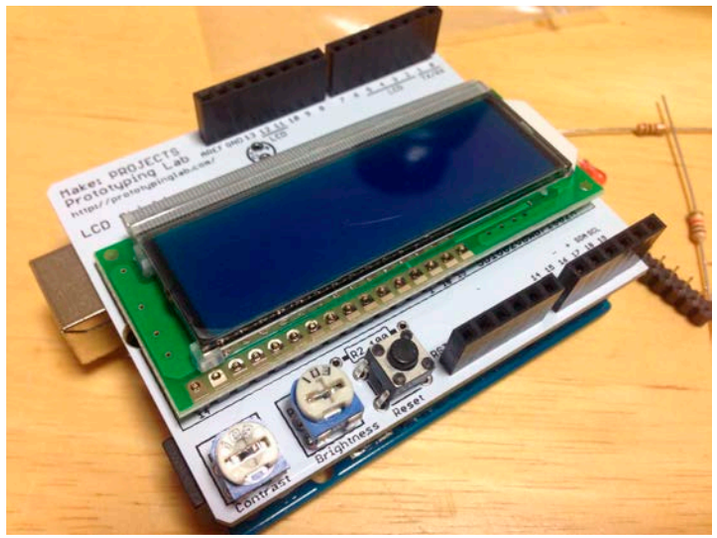

非動作時
ガラス管
不活性ガス
リード片
接点部
リード片
動作時
N極
S極
S極
N極
N極
S極

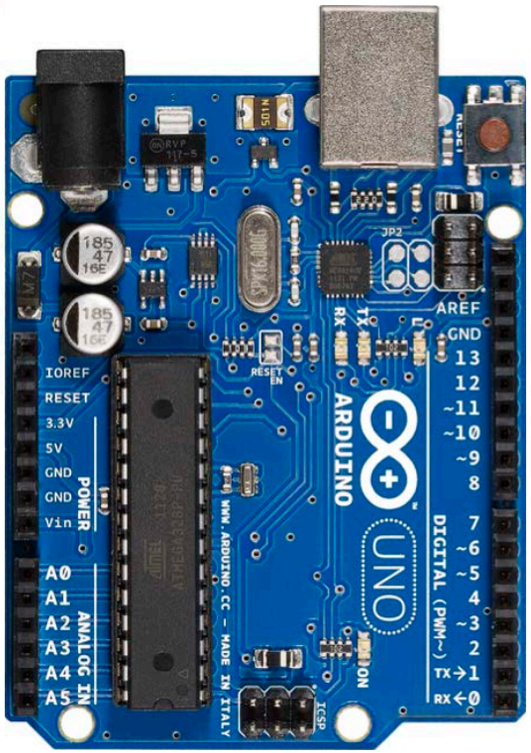


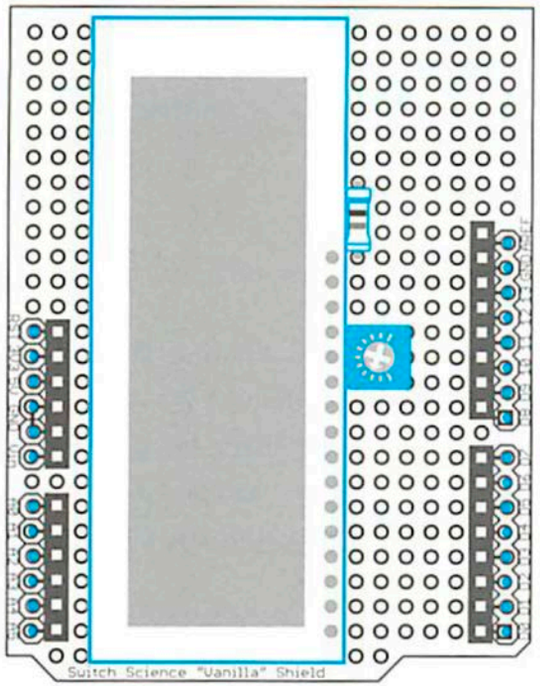
LCD を装着してパラメータを読む。

左右の入出力ピンの位置を合わせて装着。

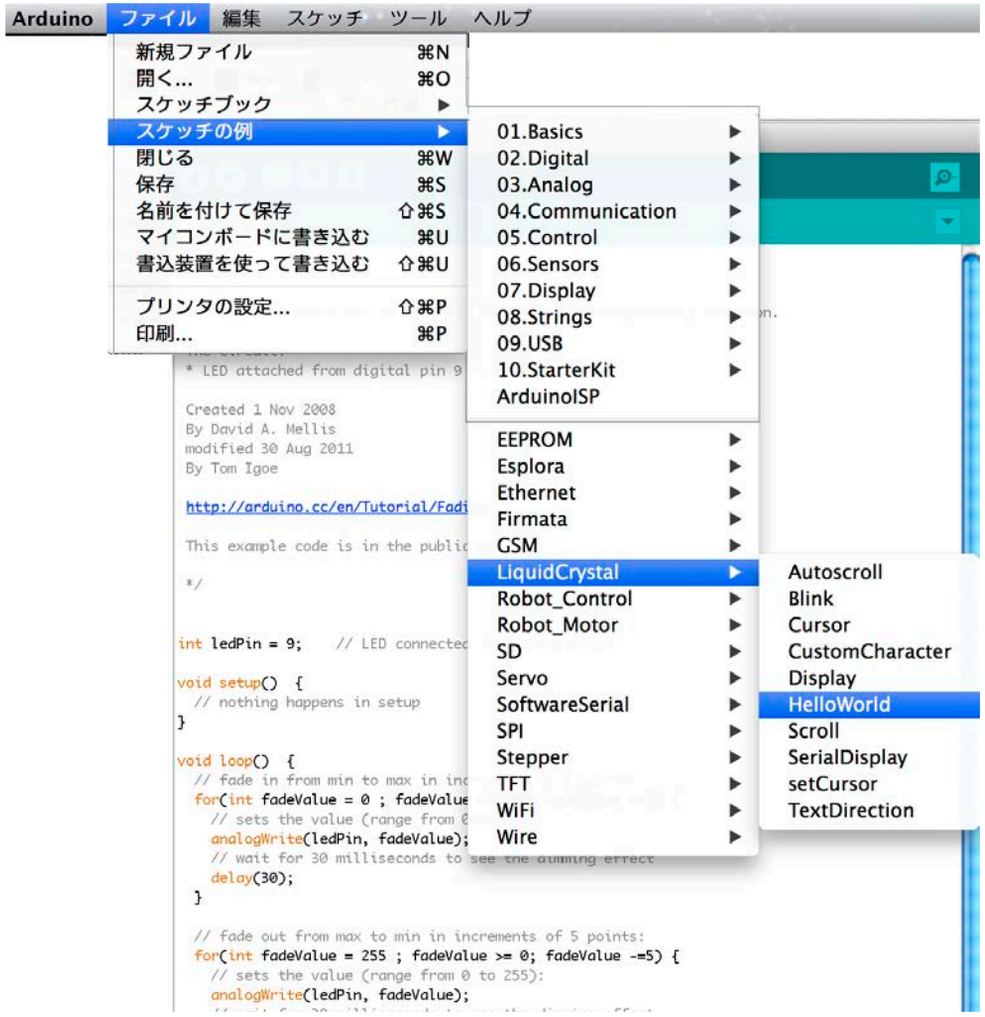
	Arduino 単体	Arduino＋PC
コスト	低	高
スタンドアロン	○	×
サイズ	小	大
サウンド	△	○
動画	×	○







Hallo World を書き出してみましょう。 04HelloWorld/HelloWorld



入力①

1 自然光の明るさを測りたい

材料
- Arduinoボード：1個
- ブレッドボード：1個
- ジャンプワイヤ：適量
- CdSセル：1個（5mm）
- 抵抗器：2本（330Ωと10kΩ）
- LED：1個
- LCDシールド：1個

CdSセルを用いることで自然光の明るさを簡単に計測できます。

Arduinoの場合

光センサに接続したピンの値を読み取り、環境光が明るければLEDの輝度を低く、暗ければ高くセットします。オプションのLCDを接続している場合には読み取った値とそこから求めた輝度を表示します。

05AnalodIN_LightSensor/MeasureBrightness

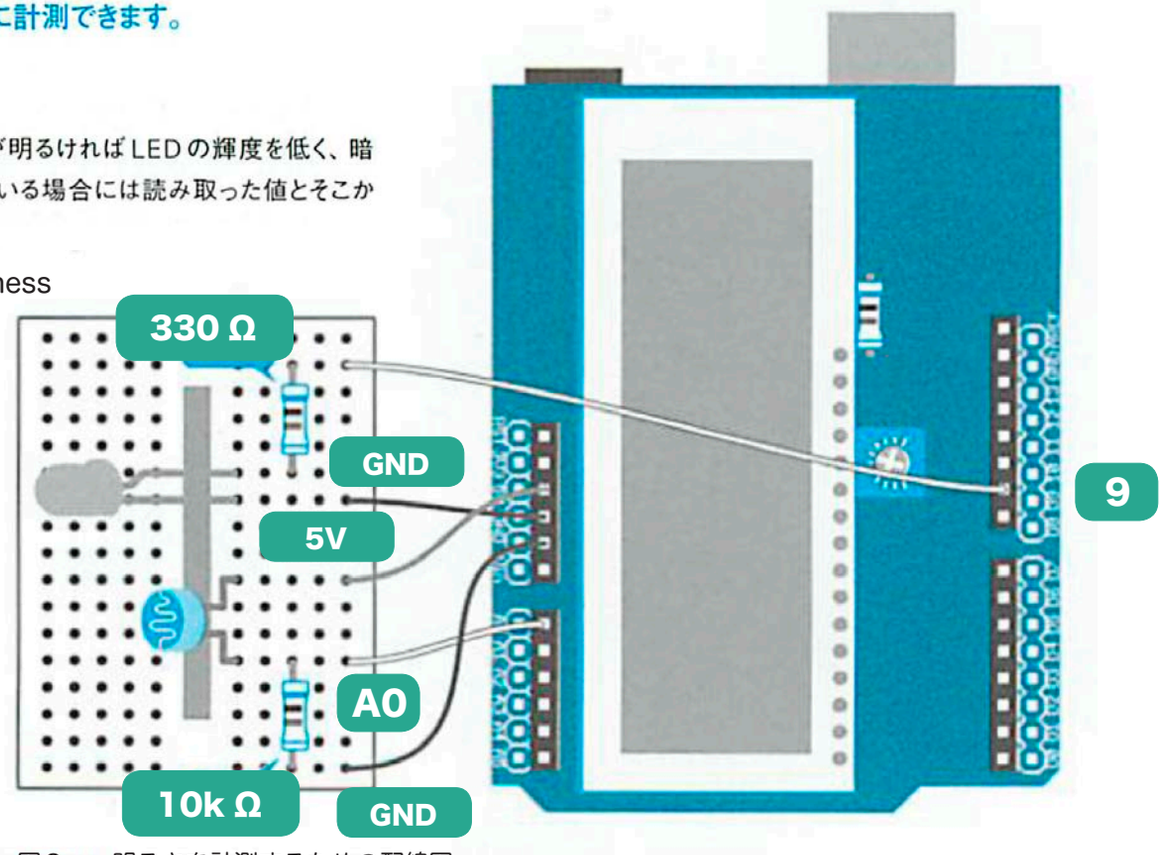


図8　明るさを計測するための配線図。
LCDを接続するとテキスト表示で確認できる。

2 距離を測りたい（赤外線センサ）

材料
- Arduinoボード：1個
- ブレッドボード：1個
- ジャンプワイヤ：適量
- PSD測距センサ用ハーネス：1個（共立エレショップの7C6312など）
- PSD測距センサ：1個（シャープ製GP2Y0A21YK）*1
- コンデンサ：1個（100μF程度の電解コンデンサ）*2
- LCDシールド：1個（98ページ参照）

赤外線方式の測距センサを用いると距離を測ることができる。

Arduinoの場合

06AnalogIN_DistanceSensor/DistanceByPSD

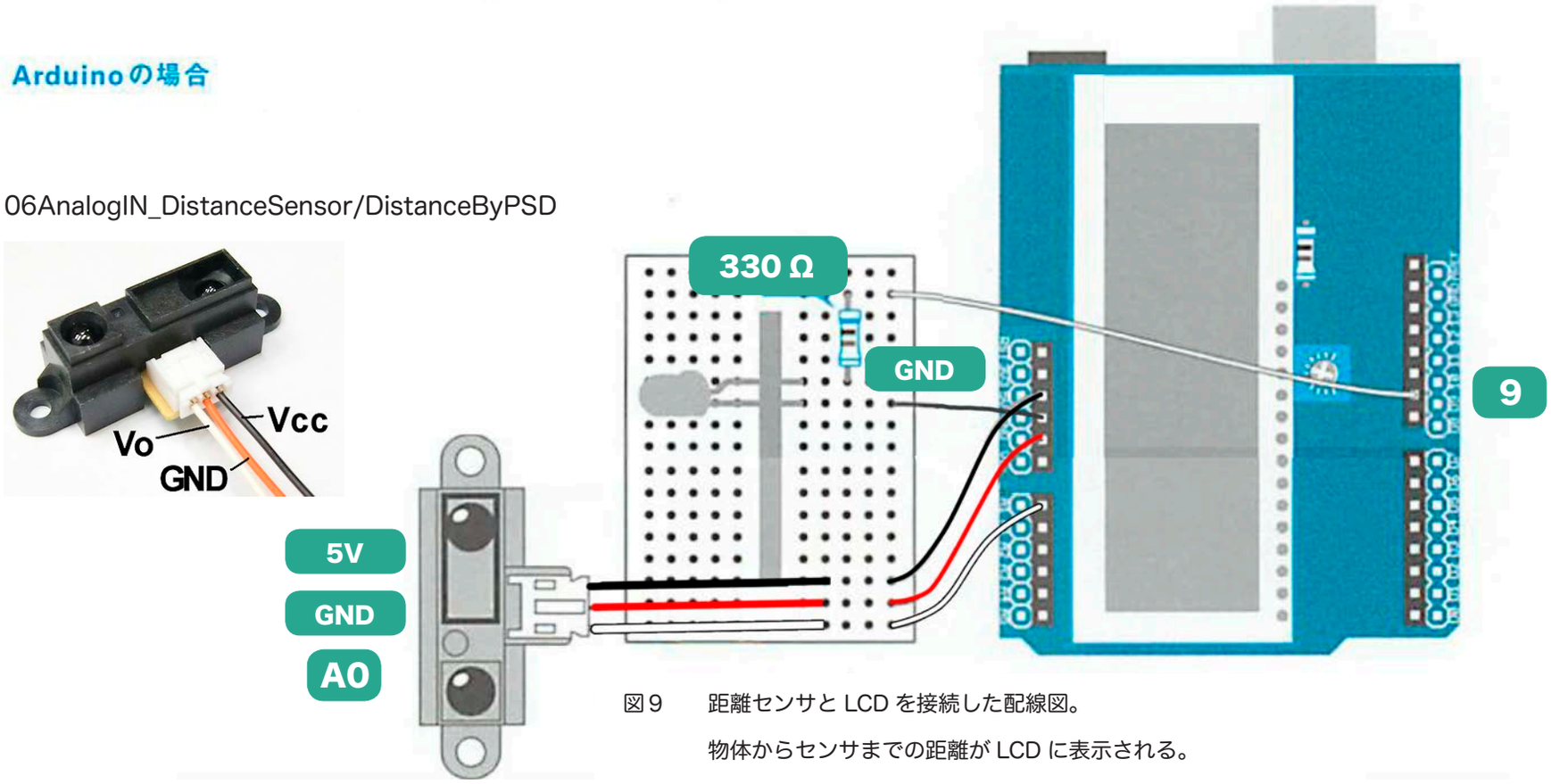


図9　距離センサとLCDを接続した配線図。
物体からセンサまでの距離がLCDに表示される。

4 振動を測りたい

材料
- Arduinoボード：1個
- ブレッドボード：1個
- ジャンプワイヤ：適量
- ピエゾ素子：1個
- 抵抗器：1本（1MΩ）
- ツェナーダイオード：1個（5.1V）

ピエゾ素子を使うことで振動を測ることができます。

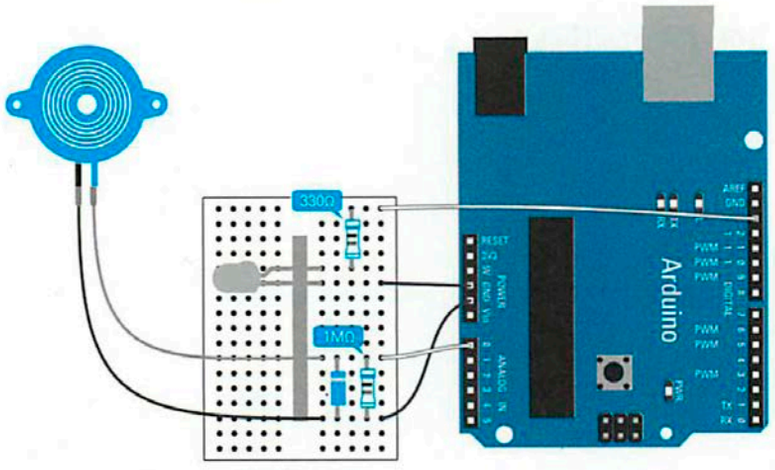


図 10　ピエゾ素子を接続した配線図。
振動を検出すると LED が一定時間点灯する。
ピエゾのワイヤは黒い方をグランド側に接続する。

Arduinoの場合

このサンプルは、ピエゾ素子を接続したアナログピンの値が一定の閾値を超えた時、LEDを一定時間点灯するものです。どこにピエゾ素子を取り付けるかによって得られる値は大きく変化するため、可変抵抗器を別のアナログピンに接続して、可変抵抗器で閾値を設定できるようにしてもよいでしょう。

Arduino＋PCの場合

このレシピでも他のレシピと同様にStandardFirmataを利用していますが、サンプリング間隔を変更しています。デフォルトのサンプリング間隔は33ms（1秒間に30回サンプリング）ですが、ピエゾ素子を叩いた際の変化は数ms程度の間しか持続しないため、このままでは取りこぼしてしまいます。そこで、間隔を10ms（1秒間に100回サンプリング）にしています。レシピ24で詳しく紹介しているSetPointフィルタを用いることで、変化が起きた瞬間をとらえるようにしています。

サンプルコード：	ディレクトリ・フォルダ名：
Arduino	(PrototypingLabExamples_r738/arduino/Sample_04_1_DetectVibration)
Processing	(PrototypingLabExamples_r738/processing/Sample_04_2_DetectVibration)
ActionScript3.0	(wonderfl_AS30/exAS30_wonderfl/lenl_Recipe4.3)

8 温度を測りたい

材料
- Arduinoボード：1個
- ブレッドボード：1個
- ジャンプワイヤ：適量
- 温度センサ：1個（LM35DZなど）
- LCDシールド：1個（98ページ参照）

温度センサを用いることにより気温などを測ることができる。

Arduinoの場合

LM35DZには3本の足がありますが、それぞれ+5V、A0、GNDに接続します。スケッチをアップロードしたら、温度センサを指ではさむ、あるいは氷やお湯の入ったカップを近づけるなどして変化を確認してみましょう。なお、そうしたテストをする場合には回路部分に水がかからないよう注意しましょう。

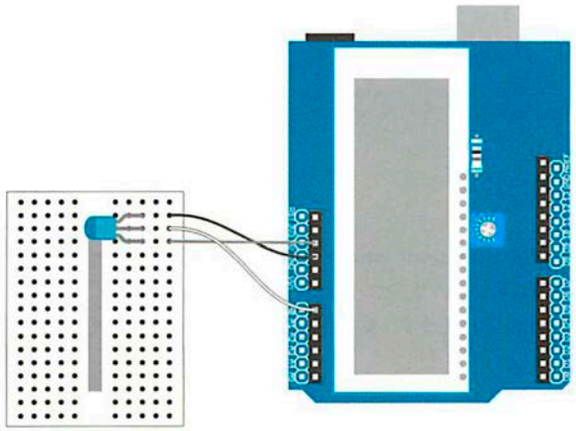


図11　温度センサをArduinoボードに接続した配線図。

Arduino＋PCの場合

Arduinoボードにはあらかじめ StandardFirmata をアップロードしておきます。入力される値は0～1が0～5Vに対応しますので、ピンに対してスケーラを追加することで0～0.2までの値を0～100にスケーリングします。実行すると、計測した温度がPCのディスプレイ上に表示されます。

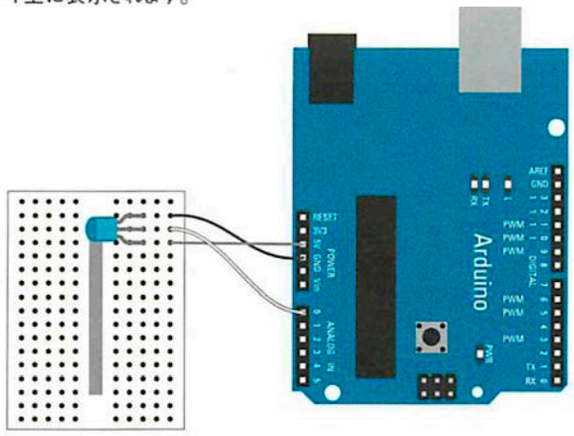


図8-2　温度センサをボードに接続した配線図。
LCDがない以外は図8-1と同じ。

サンプルコード：	ディレクトリ・フォルダ名：
Arduino	(PrototypingLabExamples_r738/arduino/Sample_08_1_MeasureTemperature)
Processing	(PrototypingLabExamples_r738/processing/Sample_08_2_MeasureTemperature)
ActionScript3.0	(wonderfl_AS30/exAS30_wonderfl/bMEY_Recipe8.3)

9 傾きを測りたい

傾きセンサや加速度センサを使うと傾きを測ることができます。

Arduinoの場合（傾きセンサ）

このサンプルは、傾きセンサが傾きを検知するとLEDが点灯するサンプルです。傾きセンサはスイッチと同様に扱うことができるため、コードはとてもシンプルになります。なお、傾きセンサの構造上、一定以上傾いた際、金属球が接点の上でバウンドし、何度もオン／オフを繰り返す場合があります。今回のように単に結果をLEDで表示する場合には問題になりませんが、一定の角度以上に傾いた瞬間に音を鳴らす、というような用途で用いる場合には、レシピ25のようにデバウンス処理を行う必要があります。逆に、ある程度の振動でオン／オフが繰り返されるため、一定時間内に変化のあった回数を計測することで、振動を検出する目的にも利用できます。

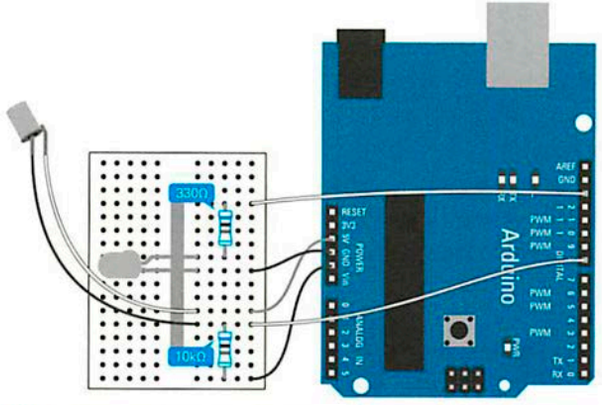


図9-2 傾きセンサとLEDを接続した配線図。
一定以上傾くとLEDが点灯する。

材料

[加速度センサの場合]

- ○Arduinoボード：1個
- ○ブレッドボード：1個
- ○ジャンプワイヤ：適量
- ○加速度センサ：1個（秋月電子のKXM52-1050など）
- ○LCDシールド：1個（98ページ参照）

Arduino＋PCの場合

このサンプルは、加速度センサが検出した傾きに応じて、画面の中に表示した立方体をコントロールするものです。Processing用サンプルの大部分は、標準配布パッケージに含まれているExamples/3D/Form/RGBCubeと同じで、もともとマウスでコントロールするようにしていた部分を、加速度センサのx軸とy軸の値でコントロールするように変更したものです。

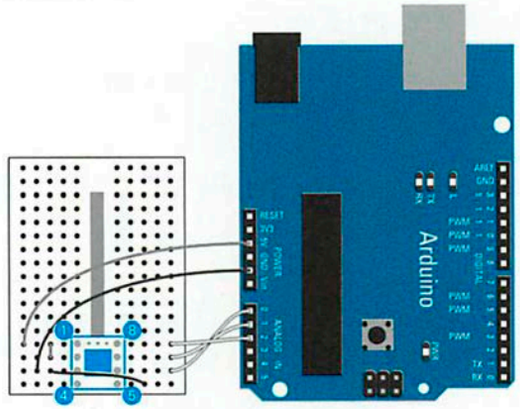


図9-4 加速度センサを接続した配線図。
加速度センサの6、7、8はそれぞれA0、A1、A2に接続する。
LCDがない以外はArduino単体の場合と同じ。

サンプルコード：	ディレクトリ・フォルダ名：
Arduino	(PrototypingLabExamples_r738/arduino/Sample_09_1_MeasureTilt)
Processing	(PrototypingLabExamples_r738/processing/Sample_09_3_MeasureTilt)
ActionScript3.0	(wonderfl_AS30/exAS30_wonderfl/8wmt_Recipe9.4)

10 動きを検出したい

材料
- Arduinoボード：1個
- ブレッドボード：1個
- ジャンプワイヤ：適量
- 加速度センサ：1個（秋月電子のKXM52-1050など）
- 抵抗器：1本（330Ω）
- LED：1個

加速度センサの急な変化を検出することで動きを検出できる。

Arduinoの場合

レシピ23で紹介するMeanフィルタを用いて、フィルタによってスムージングした値と元の値の差を求め、その差が閾値以上であればLEDを点灯させます。オーディオのレベルメータのような動作にするため、LEDの輝度は徐々に減衰させています。真っ暗な部屋で手にセンサとArduinoボードを持って動きながら長時間露光すると面白い写真が撮れるかもしれません。ただし、激しく動きすぎてUSBケーブルが抜けてしまうことのないように注意してください。

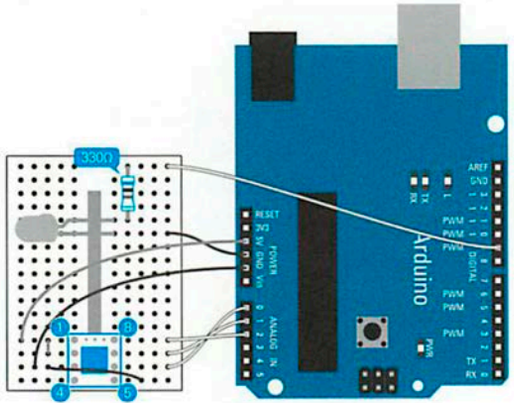


図10-2　加速度センサとLEDを接続した配線図
急な動きを検出するとLEDが一定時間点灯する。

Arduino＋PCの場合

Arduino＋PCの場合にも基本的には同じです。Meanフィルタをかけた値とセンサの値の差を求め、その差が閾値を超えた時にLEDの輝度をコントロールするオシレータをスタートさせ、LEDをオーディオのレベルメータのように点灯させています（配線図は図10-1と同じです）。

サンプルコード：	ディレクトリ・フォルダ名：
Arduino	(PrototypingLabExamples_r738/arduino/Sample_10_1_DetectMovement)
Processing	(PrototypingLabExamples_r738/processing/Sample_10_2_DetectMovement)
ActionScript3.0	(wonderfl_AS30/exAS30_wonderfl/7N9g_Recipe10.3)

12 人が動いたことを検知したい

材料
- Arduinoボード：1個
- ブレッドボード：1個
- ジャンプワイヤ：適量
- 焦電センサ：1個（NaPiOn AMN31112など）
- 抵抗器：2本（330Ωと10kΩ）
- LED：1個

07DigitalIN_MotionSensor/DetectMovementByMotionSensor

人体検知センサを使うと人や動物が動いたのを検知することができる。

Arduinoの場合

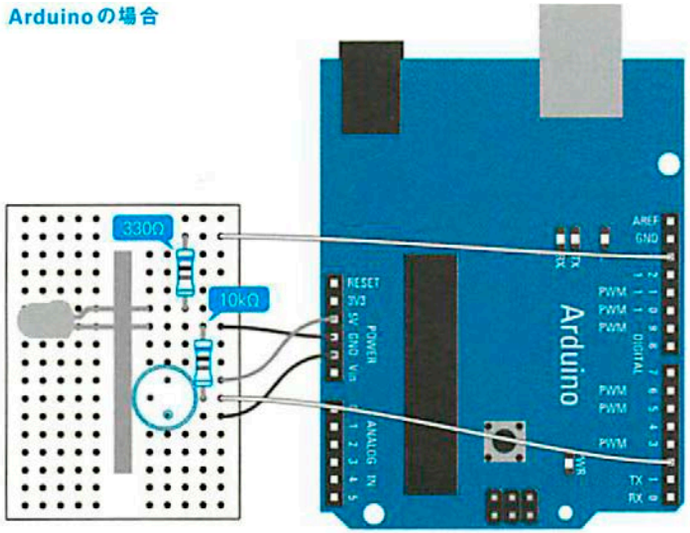


図12　Arduinoに焦電センサとLEDを接続した配線図。
焦電センサによって人体の 動きを検知すると、一定時間LEDが点灯する。

Arduino＋PCの場合

Arduino＋PCの場合も基本的にはArduino単体の場合と同じです。センサに接続したピンの立ち上がりと立ち下がりで処理を行います。配線図は、図12-2と同じです。

サンプルコード：	ディレクトリ・フォルダ名：
Arduino	(PrototypingLabExamples_r738/arduino/Sample_12_1_DetectMovementByMotionSensor)
Processing	(PrototypingLabExamples_r738/processing/Sample_12_2_DetectMovementByMotionSensor)
ActionScript3.0	(wonderfl_AS30/exAS30_wonderfl/geSC_Recipe12.3)

15 光をコントロールしたい

RGB LEDを用いることでさまざまな光をコントロールできる。

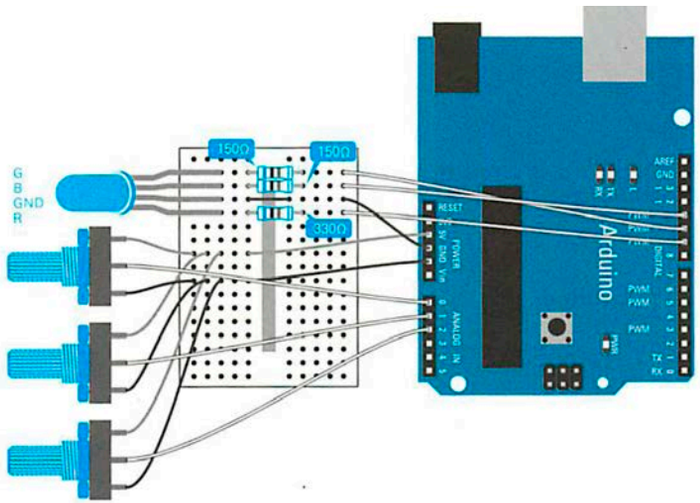


図15-1 カソードコモンの砲弾型RGB LEDを接続した配線図

材料

[砲弾型RGB LEDの場合]

- ○Arduinoボード:1個
- ○ブレッドボード:1個
- ○ジャンプワイヤ:適量
- ○砲弾型RGB LED:1個(秋月電子のI-02476など)
- ○抵抗器:3本(330Ω1本と150Ω2本、1/4W型)
- ○可変抵抗器:3個(10kΩ、Bカーブ)

Arduinoの場合

RGB LEDには、アノード側が共通のアノードコモン(英語ではcommon anode)のものと、カソード側が共通のカソードコモン(英語ではcommon cathode[*1])の2種類があります。例えば、今回材料としてあげている砲弾型RGB LEDはカソードコモンですが[*2]、どちらかと言えばアノードコモンが一般的です。カソードコモンの場合には、共通の端子をGND側に接続してピンからの吐き出し(ソース)でドライブするため、analogWriteで255にした時に最も明るくなります。これに対して、アノードコモンが一般的の場合には、共通の端子を電源のプラス側(5V)に接続してピンへの吸い込み(シンク)でドライブするため、analogWriteで0にした時に最も明るくなります。このサンプルはカソードコモンを想定したものですが、ドライブモードを定義している行を変更することで、アノードコモンでも同様の結果が得られるようになっています。

Arduino＋PCの場合

Arduino＋PCの場合も基本的にはArduino単体の場合と同じです。R、G、Bそれぞれに接続したピンのモードをPWMにセットし、可変抵抗器の値を読み取って出力ピンの値としてセットします。配線図は図15-1、15-2および15-3と同じです。

サンプルコード：	ディレクトリ・フォルダ名：
Arduino	(PrototypingLabExamples_r738/arduino/Sample_15_1_RGBLED)
Processing	(PrototypingLabExamples_r738/processing/Sample_15_2_RGBLED)
ActionScript3.0	(wonderfl_AS30/exAS30_wonderfl/fqEg_Recipe15.3)

17 | モノを振動させたい

RGB LEDを用いることでさまざまな光をコントロールできる。

材料
- Arduinoボード：1個
- ブレッドボード：1個
- ジャンプワイヤ：適量
- 振動モータ：1個
- 抵抗器：1本（2.2kΩ）
- トランジスタ：1個（2SC1815）
- 整流用ダイオード：1個（11EQS04など）
- 単3電池：2本（単4電池でもOK）
- 単3電池2本用電池ボックス：1個

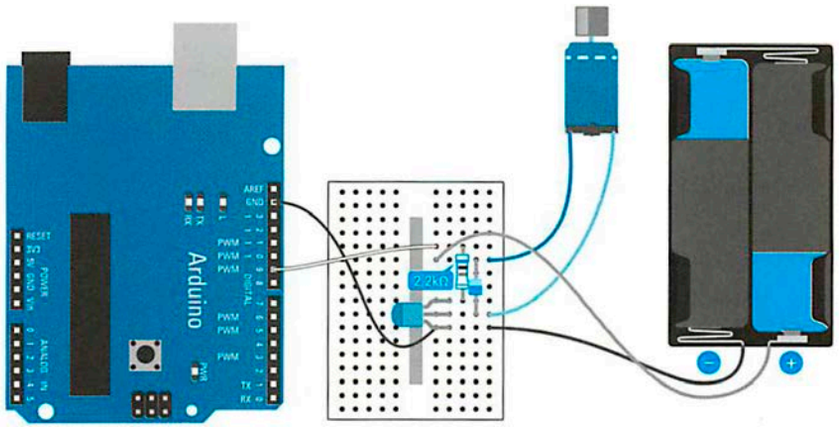
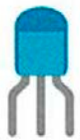


図17-2　振動モータとトランジスタを接続した配線図。
スケッチを実行すると、一定時間ごとにモータの回転と停止を繰り返す。



Emitter（エミッタ）
Collector（コレクタ）
Base（ベース）

図17-1　今回使用するバイポーラトランジスタ2SC1815の各端子の説明

Arduinoの場合

このサンプルは、2秒間隔で0.5秒間だけ振動モータを駆動するものです。もし、正しく回路を組んでいるつもりなのに振動モータが動かない場合には、USBケーブルと電池をいったん外し、トランジスタの向きや、接続している位置が配線図と同じになっているかどうかを1つずつ確認しましょう。

Arduino＋PCの場合

Arduinoボードにはあらかじめ StandardFirmata をアップロードしておきます。このサンプルでは2秒間隔で0.5秒間、振動モータを駆動します。Funnelライブラリには一定周期で出力をコントロールするオシレータクラス Osc が用意されているため、これを利用することで簡単に一定時間だけ振動モータをオンにすることができます。配線図は図17-2と同じです。

サンプルコード：	ディレクトリ・フォルダ名：
Arduino	(PrototypingLabExamples_r738/arduino/Sample_17_1_ControlVibrationMotor)
Processing	(PrototypingLabExamples_r738/processing/Sample_17_2_ControlVibrationMotor)
ActionScript3.0	(wonderfl_AS30/exAS30_wonderfl/nMKq_Recipe17.3)

出力③

19 | DCモータをコントロールしたい

材料
- Arduinoボード：1個
- ブレッドボード：1個
- モータドライバモジュール：1個（TA7291PまたはTB6612FNGモジュール）
- モータ：1個（LEGOパワーファンクションMモータなど）
- ACアダプタ：1個
- ジャンプワイヤ：適量
- 抵抗器：1本（10kΩ）

モータドライバなどを使うことでDCモータをコントロールできます。

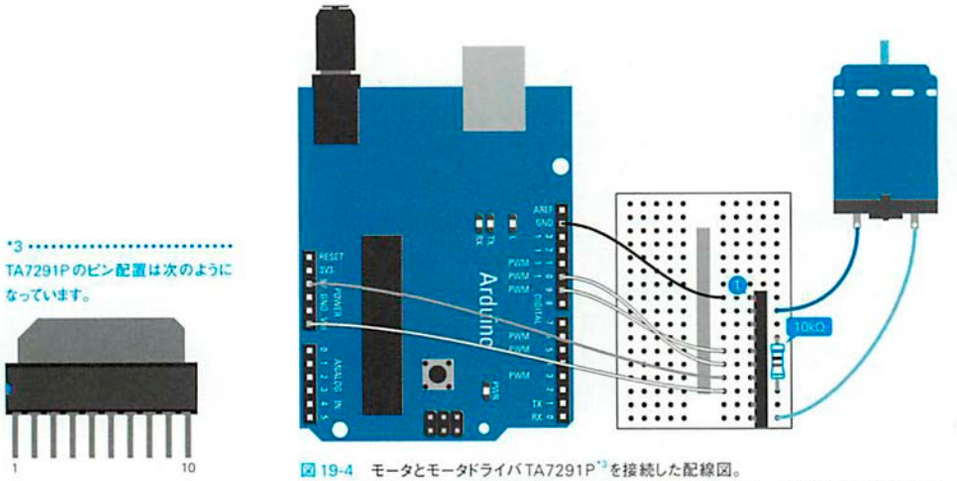


図19-4　モータとモータドライバTA7291P[*3]を接続した配線図。
スケッチを実行すると、一定時間ごとに、順方向に回転、停止（ブレーキあり）、逆方向に回転、停止（ブレーキなし）を繰り返す。

*3 TA7291Pのピン配置は次のようになっています。

ピン番号	ピンの機能	Arduino側のピン	備考
1	GND	GND	グラウンド
2	OUT1	-	モータ側端子の一方に接続
4	Vref	VIN	10kΩの抵抗器でプルアップ[*4]

Arduinoの場合

モータ駆動用の電源はArduinoボードのVIN端子から供給します。VIN端子は、ボード上の電源コネクタ（外形5.5mm、内径2.1mm、センタープラス）から逆接続防止用のダイオードを経由した電源がそのまま供給されます。今回は9VのACアダプタを接続していますので、VINの電圧は9V（実際にはダイオードによって電圧が若干降下します）となります。モータにより標準電圧が異なるため、6V以下を標準電圧とするモータを使用する場合には、VINではなくモータ用の電源と直接接続する必要があります[*2]。

このサンプルは、一定時間ごとに正回転と逆回転を繰り返すものです。モータなどの出力をコントロールする場合、コントロールの仕方に誤りがあると物理的に壊れてしまうことがあります。最初はモータにギアだけを取り付けた状態で動作を確認し、次にその他の機構部品と組み合わせて動作確認、というように段階を追って確認していくとよいでしょう。

Arduino＋PCの場合

Arduinoボードには StandardFirmata をアップロードしておきます。配線図は図19-4、図19-5と同じです。

サンプルコード：	ディレクトリ・フォルダ名：
Arduino	(PrototypingLabExamples_r738/arduino/Sample_19_1_ControlDCMotor)
Processing	(PrototypingLabExamples_r738/processing/Sample_19_2_ControlDCMotor)
ActionScript3.0	(wonderfl_AS30/exAS30_wonderfl/h7fx_Recipe19.3)

22 AC100V機器のオンオフをコントロールしたい

材料
- Arduinoボード：1個
- SSRモジュール：1個（秋月電子の通販コードK-00203など）
- ACコンセントコード：1本（自作またはエレキットのAP-903など）
- 電球など：1個（AC100Vで電源スイッチのみで動作するシンプルな機器）

SSRを使うとAC100V機器のオンオフをコントロールできる。

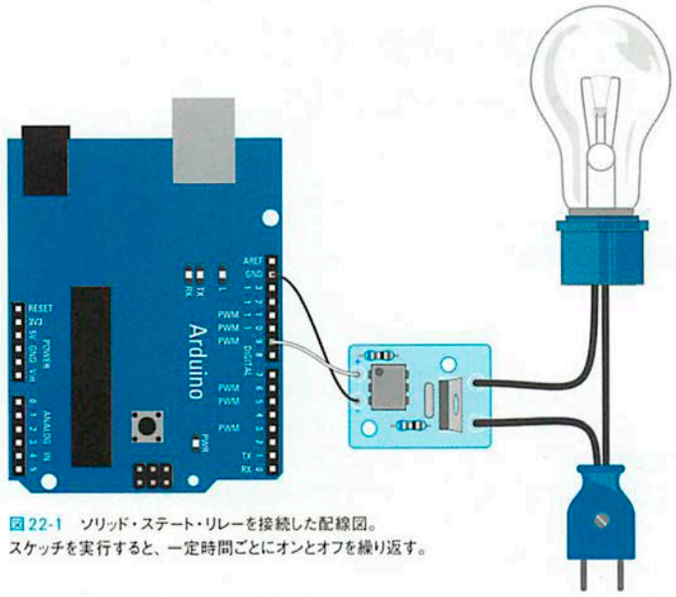


図22-1　ソリッド・ステート・リレーを接続した配線図。
スケッチを実行すると、一定時間ごとにオンとオフを繰り返す。

Arduinoの場合

まず最初に、ソリッド・ステート・リレーキットを説明書に従って組み立ててください。組み立てが終わったら、実体配線図を参考にArduinoボードのデジタル出力ピンをSSRモジュールの入力側の＋（プラス）端子に、GNDを－（マイナス）側を端子に接続します。SSRモジュールの出力側は、AC100V機器と電源プラグの間に接続します。

AC100V機器と接続するケーブルには、100円均一ショップなどで販売しているAC延長コードを加工して利用するとよいでしょう。加工が難しいと感じる場合には、加工した状態で販売されているACコンセントコード（エレキットAP-903など）を使うとよいでしょう。

スケッチはArduinoの動作確認でも用いたサンプルとほぼ同じで、1秒ごとにオンとオフを繰り返します。なお、オンにしている時間をあまりに短くしすぎるとAC100V機器が壊れてしまう場合もありますので、注意が必要です。

Arduino＋PCの場合

Arduino＋PCの場合もArduino単体の場合と基本的には同じです。Funnelライブラリには一定周期の波形を生成するOscクラスがありますので、これを利用して周波数0.5Hzの矩形波を生成することにより、Arduino単体のサンプルと同じ結果をよりシンプルなコードで実現しています。配線図は、図22-1と同じです。

サンプルコード：	ディレクトリ・フォルダ名：
Arduino	(PrototypingLabExamples_r738/arduino/Sample_22_1_ControlACDevice)
Processing	(PrototypingLabExamples_r738/processing/Sample_22_2_ControlACDevice)
ActionScript3.0	(wonderfl_AS30/exAS30_wonderfl/xEkF_Recipe22.3)

Flash 編での wondefl ソースコードの適用方法。

サンプルコード：(和田マニュアルでは p24 に掲載してます)

Introduction 4.10　…P103

wonderfl_AS30　/exAS30_wonderfl/IGFN_I4.10

/zip_filesl/GFN.zip

(1)AS ファイル「IGFN.as」を使えるように修正する

AS3.0 では public 宣言とファイル名と同じ名前にする必要があります。

このサンプルの場合は public class、function とも「AnalogIOExample」とついているので、「AnalogIOExample.as」という同名のファイルを作成します。

(2)「AnalogIOExample.as」を参照する Flash ファイルを作成する

flash ファイルを新規作成したら、プロパティパネルのクラスパスのボックスに「AnalogIOExample」と、拡張子なしで入力します。AS ファイルと同じ箇所に保存します。

(3) ライブラリパス、ソースパスを指定する

Arduino との通信のための他 AS ファイルを読み込めるように、ライブラリへのソースパス設定する必要があります。(和田マニュアル p05) パブリッシュ設定の ActionScript3.0 の詳細設定にて、ダウンロードしたパッケージ「libraries/actionscript3」というフォルダの中の src フォルダをソースパスとして指定します。

IGFN_I4.10.as

この名前の AS ファイルを作成

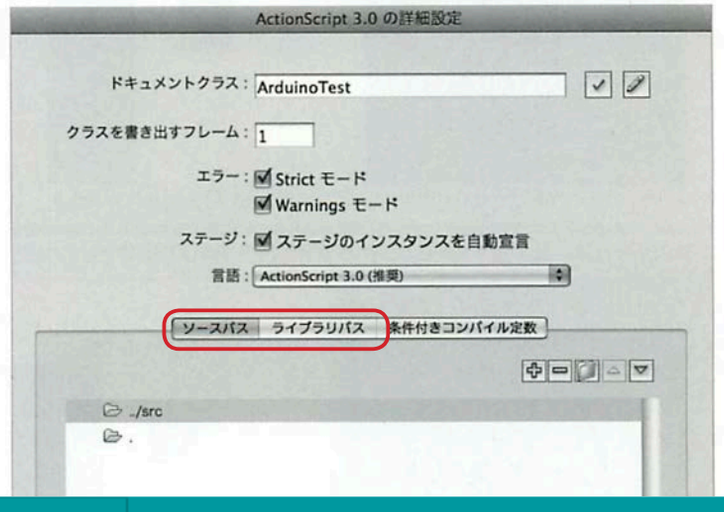
```
package {
  import flash.display.Sprite;
  import funnel.*;

  public class AnalogIOExample extends Sprite {
    private var arduino:Arduino;

    // 可変抵抗器に接続したピン
    private var sensorPin:Pin;

    // LED に接続したピン
    private var ledPin:Pin;

    public function AnalogIOExample() {
      // LED に接続したピン（D9）のモードを PWM にセット
      var config:Configuration = Arduino.FIRMATA;
      config.setDigitalPinMode(9, PWM);
      arduino = new Arduino(config);
```



2 | 距離を測りたい（赤外線センサ）

P.111

材料
- ○Arduinoボード：1個
- ○ブレッドボード：1個
- ○ジャンプワイヤ：適量
- ○PSD測距センサ用ハーネス：1個（共立エレショップの7C6312など）
- ○PSD測距センサ：1個（シャープ製GP2Y0A21YK）*1
- ○コンデンサ：1個（100μF程度の電解コンデンサ）*2
- ○LCDシールド：1個（98ページ参照）

赤外線方式の測距センサを用いると距離を測ることができる。

Arduinoの場合

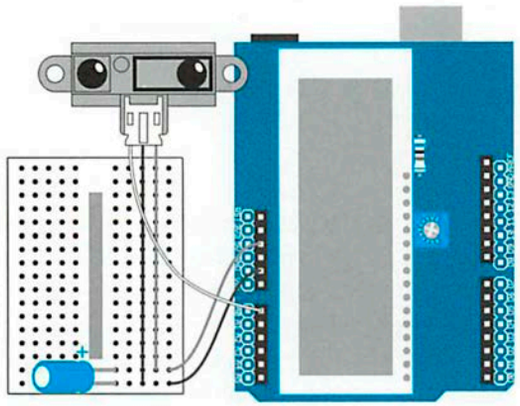


図2-2　PSDセンサとLCDを接続した配線図。
物体からセンサまでの距離がLCDに表示される。

Arduino＋PCの場合

Arduinoボードにはあらかじめ Standard Firmata をアップロードしておきます。Arduino＋PCの場合も基本的にはArduino単体の場合と同じです。ただし、Arduinoの場合にはアナログ入力の値を0～1023で表すのに対して、Funnelライブラリでは0～1で表します。このため、ピンの値に1023をかけ算することで同じ変換式を使えるようにしています。

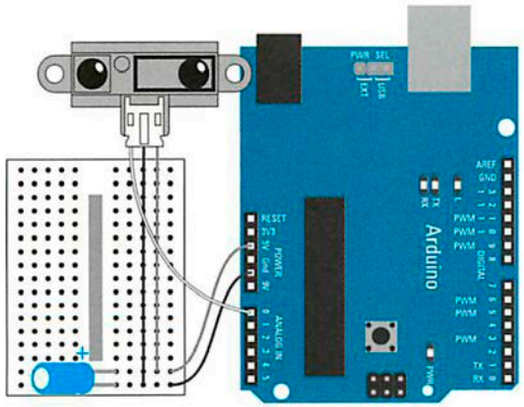


図2-3　PSDセンサとLCDを接続した配線図。
LCDがない以外はArduino単体の場合と同じ。

サンプルコード：	ディレクトリ・フォルダ名：
Arduino	(PrototypingLabExamples_r738/arduino/Sample_02_1_MeasureDistanceByPSD)
Processing	(PrototypingLabExamples_r738/processing/Sample_02_2_MeasureDistanceByPSD)
ActionScript3.0	(wonderfl_AS30/exAS30_wonderfl/7pvz_Recipe2.3)