Arduino・Suwano をはじめよう@KYOTO-U　No.2

2012 年 5 月 2 日

京都大学 学術情報メディアセンター

喜多　一

今回の学習目標

- ブレッドボードの使い方を覚える.
- ブレッドボード上で LED とコンタクトスイッチを使った回路作成し Arduino で制御する.（デジタルの入力と出力）
- 変数と条件分岐のプログラミングを覚える.

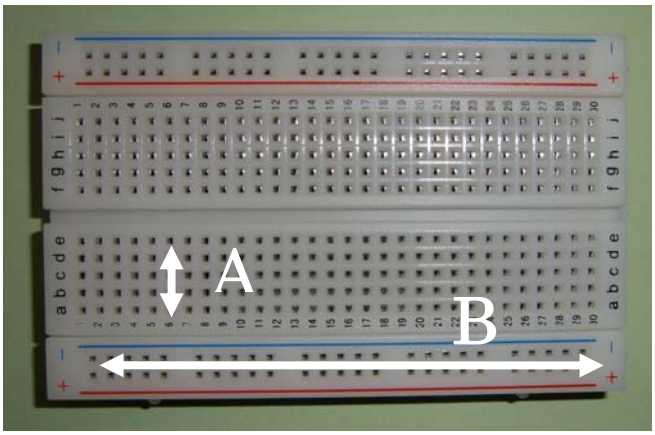


写真１ ブレッドボード

１．　ブレッドボードを使う

ブレッドボードは電子部品を差し込んで回路を試作するための配線用のボードです。
写真１のブレッドボードでは，中央の部分は A 方向に相互が導通してします.
周辺部の２列は B 方向に相互に導通しています．ここは電源線（赤色）や ０ V の線（GND，青色）などに使います.

今回の実習ではコンタクトスイッチ（写真２）の押し下げによって LED の点灯，消灯を制御する回路を作ってみます．コンタクトスイッチには４本の足が出ていますが，B 方向の２本は内部で接続されています．スイッチを押し下げると A 方向の２本が導通します.

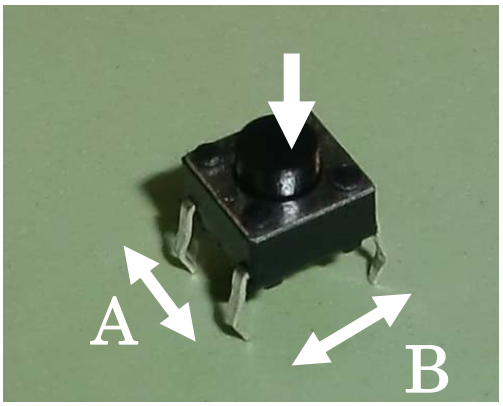


写真２　コンタクトスイッチ

２．　ブレッドボード上の LED とコンタクトスイッチを使った回路の制御

2.1 ブレッドボード上での配線

(Suwano ではドーターボードのスイッチ(SW2)と LED(LED2) を使って同じ実習ができます．)

Arduino でコンタクトスイッチの押し下げ状態を検出するために図１のような回路をブレッドボードを使って作成します．以下の手順に従ってください．

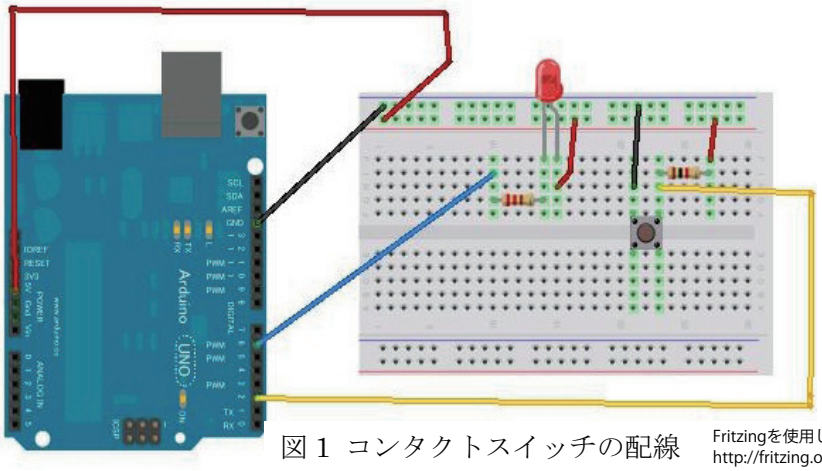


図１ コンタクトスイッチの配線　Fritzingを使用して作成 http://fritzing.org/download/

注意！回路を構成している最中は Arduino を USB ケーブルから外しておいてください．これは電源などを短絡して Arduino や電子部品を壊さないようにするためです．

注意！電子部品の端子は柔らかくて曲がったり折れたりしやすいものです．ブレッドボードに差し込む際には無理に押し込んだりせず，慎重に作業してください．

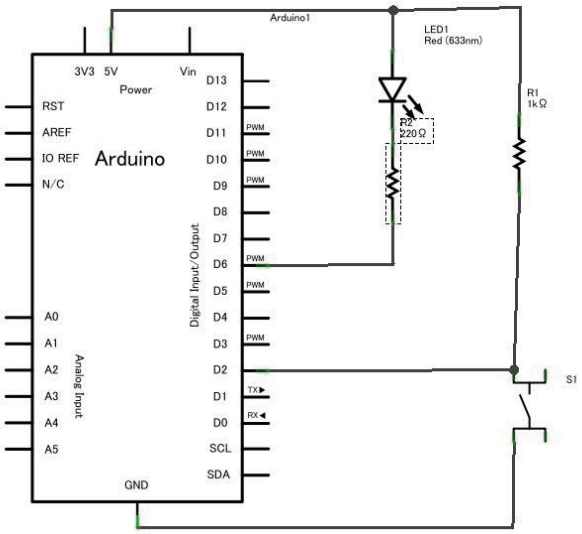


図２　回路図

1. まず Arduino の 5V と GND からの配線を行ないます．5V には赤の，GND には黒のジャンパ線を使います．
2. 次に LED と抵抗器からなる回路を作成してください．
   - 5V の電源から LED のアノード（足の長いほう）へ，LED のカソード（足の短いほう）から抵抗器（330Ω，橙，橙，茶，金のカラーコード）を介して Arduino の D6 に配線します．
3. 次にコンタクトスイッチの回路を作成します．
   - 5V の電源から抵抗器（10kΩ，茶，黒，橙，金のカラーコード）を介してコンタクトスイッチの片側に，コンタクトスイッチの反対側を GND に配線します．
   - コンタクトスイッチと抵抗器が接続している部分から Arduino の D2 ピンに配線します．

この回路では D6 が 0V（LOW）のとき LED が点灯し，5V（HIGH）のとき LED は消灯します．またスイッチが押されているときは D2 には 0V（LOW）が，押されていないときには 5V（HIGH）の電圧がかかります．

回路図で表したものが図２です．回路図では電源線(5V）を上側に，GND(0V）を下側に描くことが多いです．

注意！配線は電子回路における「プログラム」です．誤っていると動作しませんし，部品を壊す恐れもあります．動作させる前に必ず点検するようにしてください．

2.2 コンタクトスイッチからの入力の処理（その１）

スイッチを押している間，LED が点灯するするプログラムを作成します．次のプログラム（スケッチ）を作成して実行してみてください．

行番号	ソースコード	説明
1	#define LED 6	LED と書いて 6 と，
2	#define BUTTON 2	BUTTON と書いて 2 を表す定義
3	int val = 0;	ボタンの状態を格納する整数型
4		の変数
5	void setup()	
6	{	
7	pinMode(LED, OUTPUT);	
8	pinMode(BUTTON, INPUT);	2 番ピンを入力用に使います
9	}	
10		
11	void loop() {	
12	val = digitalRead(BUTTON);	2 番ピンの状態を読み込みます
13	if (val == HIGH) {	状態が HIGH なら
14	digitalWrite(LED, HIGH);	6 番ピンを高い電圧にします
15	} else {	そうでなければ
16	digitalWrite(LED, LOW);	6 番ピンを低い電圧にします
17	}	
18	delay(10);	少し待ちます
19	}	

このプログラムで新しく学ぶこと：マクロ定義と変数の利用

マクロ定義:1 行目の「#define LED 6」と 2 行目の「#define BUTTON 2」の #define（define は定義という意味）は次に現れる名前をその次に現れる定義（文字の列）の代わりに使えるようにする命令（マクロ定義）です．

#define 名前 定義

プログラムの中に直接 6 や 2 という数字を埋め込む代わりに LED とか BUTTON と書けば，ベリファイの際に 6 や 2 に置き換えてくれます．たとえば 7 行目の

pinMode(LED, OUTPUT)

は pinMode(6, OUTPUT) に置き換えられます．こうすることにより，直接，数字を書くより意味が分かりやすくなりプログラムが分かりやすくなります．

変数の宣言:2行目の「int val = 0;」は変数の宣言です．

- 変数には値を入れたり（書き換えたり），入っている値を参照したりすることができます．プログラムの中で情報を保持するために基本的な仕組みです．

- この行では整数(int, integer の略)型の変数で名前が val というものを宣言し，その内容を 0 に初期化しています.
  - int 型の整数は Arduino では -2^{15} (-32768) から $2^{15}-1$ (32767) までの整数を扱うことができます．桁数が限られていることに注意してください.
  - より桁数の多い数値の利用には long 型（-2,147,483,648 から 2,147,483,647 まで扱えます） を，
  - また小数点以下の値を持つ場合には float 型（floating point number は浮動小数点数という意味）を用います.
- 12 行目の

  ```
  val = digitalRead(BUTTON);
  ```

  の = は変数に値を代入（設定）する命令（演算子）で，
  - まず右辺の値(ここでは関数 digitalRead(BUTTON) で得られるスイッチの値)を計算し，
  - これを左辺の変数 val に設定します.
- 13 行目の

  ```
  if (val == HIGH) {
  ```

  の if 文（後述します）の中で val == HIGH とありますが，「==」は左辺と右辺の値が等しいかどうかを検査する命令（演算子）です．左辺に val とありますが，これにより val という変数に設定している値を読み出し（評価し），右辺の定数 HIGH と等しいかどうか比較しています.
- setup() や loop() といった**関数の定義の外側で定義される変数**は**「外部変数」**と呼びます.
  - このプログラムでは loop() の中で val に値を代入したり，値を評価したりしていますが，
  - 外部変数はどの関数の中でも代入や評価ができます.
  - また外部変数の値はプログラムが起動すれば，常に値を保持しています.

2.3 コンタクトスイッチからの入力の処理（その２）

今度はスイッチを押す度に点灯と消灯を切り替えるプログラムを作ります．そのためには

- スイッチが押されていない状態から押された状態への変化を検出し，
- その度に消灯すべきか，点灯すべきかを切り替えなければなりません．

プログラムは少し複雑になります．

行番号	ソースコード	説明
1	#define LED 6	
2	#define BUTTON 2	BUTTON と書いて 2 を表す
3	int val = 1;	ボタンの状態を格納する変数
4	int oldVal = 1;	直前のボタンの状態用の変数
5	int lampOn = 0;	LED の状態用の変数，点灯時に
6		1，消灯時に 0 とします．
7	void setup()	
8	{	
9	pinMode(LED, OUTPUT);	
10	pinMode(BUTTON, INPUT);	2 番ピンを入力用に使います
11	digitalWrite(LED, HIGH);	最初，消灯しておきます．
12	}	
13		
14	void loop() {	
15	val = digitalRead(BUTTON);	2 番ピンの状態を読み込みます
16	if ((val == LOW)&&(oldVal == HIGH)) {	今の状態が LOW で前の状態が
17	if (lampOn == 0) {	HIGH ならスイッチが押された
18	lampOn = 1;	ときなので lampOn の値を反転
19	} else {	します．
20	lampOn = 0;	
21	}	
22	}	
23	if (lampOn == 1) {	lampOn の値に応じて LED を点
24	digitalWrite(LED, LOW);	灯，消灯します．
25	} else {	
26	digitalWrite(LED, HIGH);	
27	}	
28	delay(10);	少し待ちます
29	oldVal = val;	今の状態を前の状態として保存
30	}	します．

先のプログラムは 15-19,21-25 行目を変更して以下のように書いても同様に動作します．これは lampOn の値が 0 と 1 を取ること，HIGH と LOW の定義が 1 と 0 であることを使っています．プログラムはコンパクトになりますが，分かりにくくなります．

行番号	ソースコード	説明
1	#define LED 6	
2	#define BUTTON 2	BUTTON と書いて 2 を表す
3	int val = 1;	ボタンの状態を格納する変数
4	int oldVal = 1;	直前のボタンの状態用の変数
5	int lampOn = 0;	LED の状態用の変数
6	void setup()	
7	{	
8	pinMode(LED, OUTPUT);	
9	pinMode(BUTTON, INPUT);	2 番ピンを入力用に使います
10	digitalWrite(LED, HIGH);	
11	}	
12		
13	void loop() {	
14	val = digitalRead(BUTTON);	2 番ピンの状態を読み込みます
15	if ((val == LOW)&&(oldVal == HIGH)) {	今の状態が HIGH で前の状態が
16	lampOn = 1 - lampOn;	LOW なら lampOn の値を反転し
17	}	ます.
18	digitalWrite(LED, 1-lampOn);	lampOn の値で LED を点灯，消
19	delay(10);	灯し,少し待ちます
20	oldVal = val;	今の状態を前の状態として保存
21	}	します.

2.4 コンタクトスイッチからの入力の処理（その３）

今度はスイッチが押されたら暫くの間，LED を点灯するプログラムを作りましょう．

行番号	ソースコード	説明
1	#define LED 6	
2	#define BUTTON 2	BUTTON と書いて 2 を表す
3	#define DURATION 1000	継続時間
4	#define TICK 10	刻み幅
5	int val = 1;	ボタンの状態を格納する変数
6	int oldVal = 1;	直前のボタンの状態用の変数
7	int count = 0;	LED の点灯期間用の変数
8	void setup()	
9	{	
10	pinMode(LED, OUTPUT);	
11	pinMode(BUTTON, INPUT);	2 番ピンを入力用に使います
12	digitalWrite(LED,HIGH);	最初，消灯します．
13	}	
14		
15	void loop() {	
16	val = digitalRead(BUTTON);	2 番ピンの状態を読み込みます
17	if ((val == LOW)&&(oldVal == HIGH)) {	今の状態が HIGH で前の状態が
18	count = DURATION;	LOW なら count を 1000 にし
19	}	ます．
20	if (count > 0) {	count が正なら
21	digitalWrite(LED, LOW);	LED を点灯し
22	count = count - TICK;	count を TICK だけ減らします
23	} else {	
24	digitalWrite(LED, HIGH);	そうでなければ消灯
25	}	
26	delay(TICK);	TICK だけ少し待ちます
27	oldVal = val;	今の状態を前の状態として保存
28	}	します．

３．　変数，代入文と条件分岐のプログラミング

２．節のプログラムで新しく出てきたプログラムの概念に変数のほかに条件分岐があります.

3.1 代入文での変数の評価と代入

変数は代入文の右辺に現れれば，その値を評価し，左辺に現れれば右辺の値を代入することになります．例えば 2.4 節のプログラムで 22 行目に

```
count = count - TICK;
```

とありますが，例えば count に値 1000 が，TICK の定義が 10 に設定さているとすれば，この行は以下のように解釈，実行されます.

1. まず右辺を評価（します）
   (ア)右辺に現れた count では値を評価して 1000 を得ます.
   (イ)右辺を計算すると 1000 - 10 を評価して 990 という値（評価値）を得ます.
2. これを左辺に現れた変数 count に代入します.count の値は右辺の評価値である 990 に更新されます.

すなわち，この行は変数 count に代入されている値を１０だけ減らすプログラムです.

代入文は　「変数 = 式；」の形式をとる.
等号の右辺を計算して結果を左辺の変数に設定する.
右辺に現れる変数は代入されている値が使われる.

3.2 条件分岐

変数の値などによってプログラムの動作を変えるには if 文を用います.

2.2 節のプログラムでは 13 ～ 17 行目のプログラムで

```
if (val == HIGH) {
  digitalWrite(LED, HIGH);
} else {
  digitalWrite(LED, LOW);
}
```

のように記述されています．これは変数 val の値が HIGH と等しいかどうか(val == HIGH) を検査し，それが成り立っていたら LED を消灯（digitalWrite(LED, HIGH)）し，そうでなければ(else) LED を点灯する（digitalWrite(LED, LOW)）というプログラムです.

注意！等しいかどうかの検査には演算子 == を用います．等号が１つ (=) だと代入文になってしまうので注意が必要です.

これをフローチャートに描いたものが図 2 です. 条件判断には菱形の記号を使います.

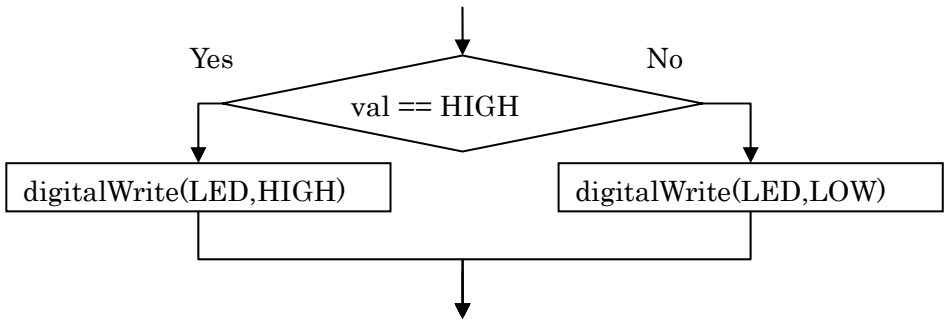


図２　条件分岐のフローチャート

複合的な条件を設定することもできます. 2.3 節のプログラムでは２つの等号がともに成立していることを判定するために

```
if ((val == LOW)&&(oldVal == HIGH)) {
…
}
```

という表記を用いています. ここで,「&&」は論理積(AND, 両方が成立しているときに成立)を表します. 論理和 (OR, どちらかが成立しているときに成立) には「||」を用います.

if 文の形式は以下のようになります.

```
if (条件) {
   成立している時の動作
} else {
   成立していない時の動作
}
```

成立していない時に何もしないなら, else { … } を省略できます.

```
if (条件) {
   成立している時の動作
}
```

if 文を入れ子にして使うこともできます. 先の複合的な条件設定は if 文の入れ子で次のようにプログラムすることもできます.

```
if (val == LOW) {
  if (oldVal == HIGH) {
    …
  }
}
```

コンタクトスイッチのプログラミング

コンタクトスイッチの状況は関数 digitalRead(BUTTON) を呼び出すことによって知ることができます.「スイッチが押されている」ことを検出するには，digitalRead(BUTTON) の値が HIGH かどうかを検査すればいいのですが,「スイッチが（新たに）押された」ことを検出するには

- 直前には押されていない
- 今の時点では押されている

という２つの条件がともに成り立っていることを確認する必要があります.
そこで

```
oldVal = digitalRead(BUTTON);
delay(10);
val = digitalRead(BUTTON);
if ((val == LOW)&&(oldVal == HIGH)) {
   成立したときの処理
}
```

などとして検査しなければなりません．ここで delay(10)を挿入しているのは機械的なスイッチは金属片を相互に接触させて回路を導通させるのですが，接触は単純に生じるのではなく，スイッチの投入時や切断時には短時間に何度が接触，非接触を繰り返すことがあります．これはバウンスと呼びます．このようなバウンスを無視するために 10 ミリ秒だけ（値は経験的に決める）待つようにプログラムするのです.

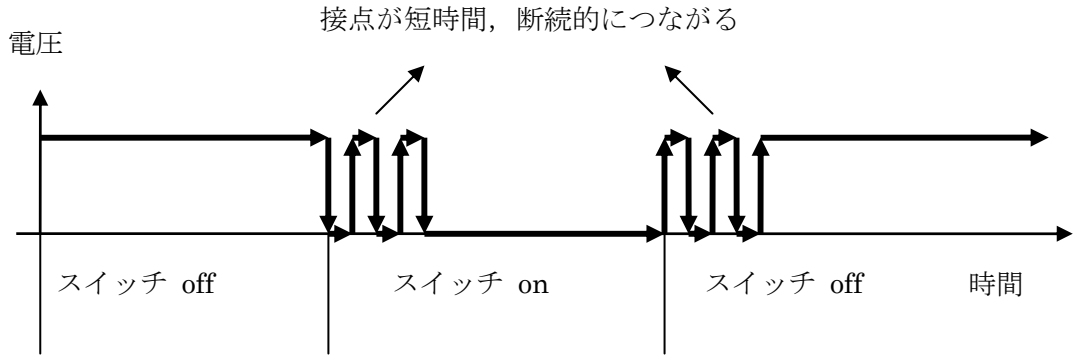
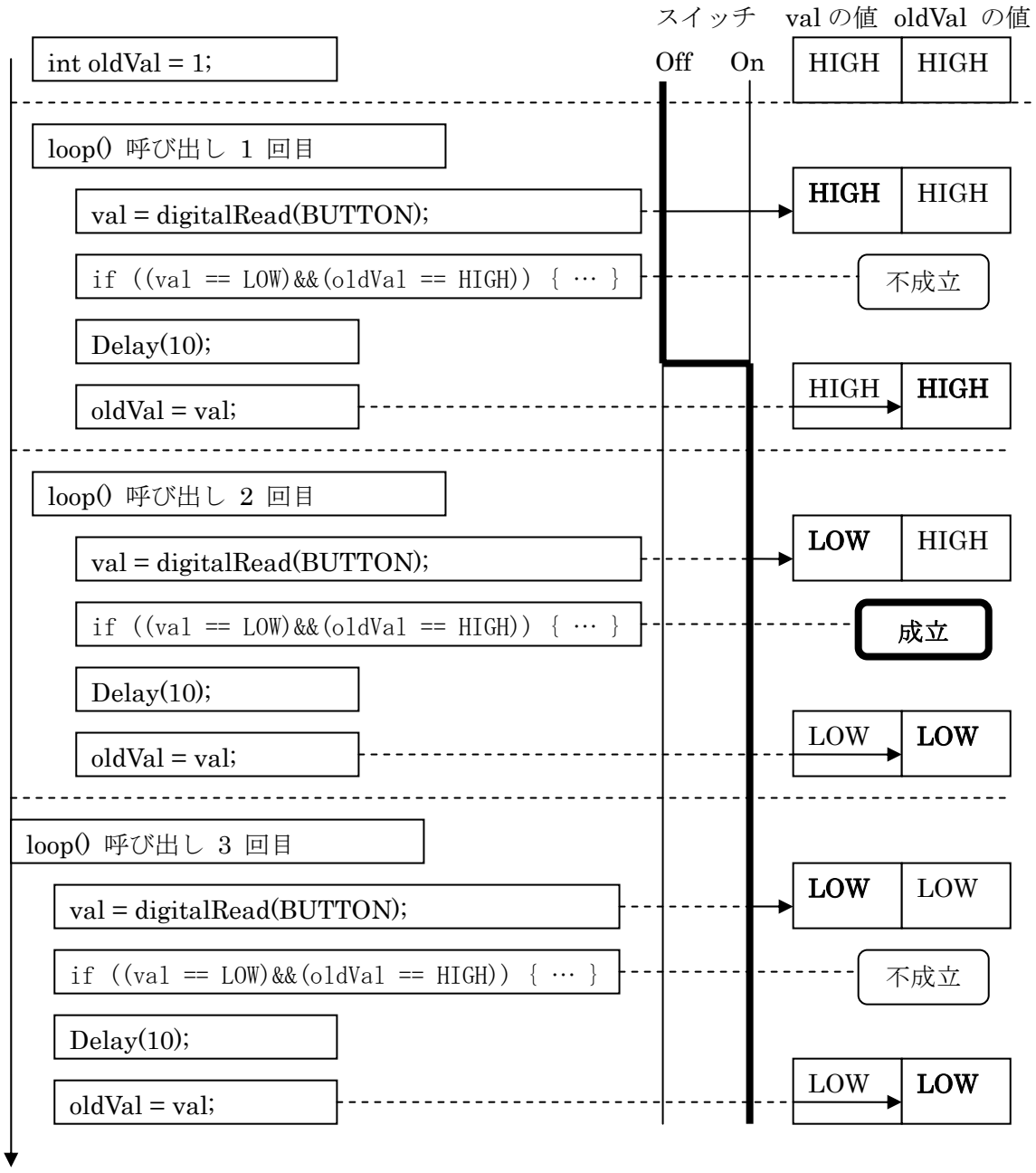


図３　スイッチのバウンシング

2.3 節や 2.4 節のプログラムでは loop() 関数が繰り返し呼ばれることを利用して以下の動作をしています。

- oldVal は初期値として 1 (HIGH, 押されていない)としている。
- loop() 関数の最初に val = digitalRead(BUTTON); としてスイッチの値を読み取っている。
- if ((val == LOW)&&(oldVal == HIGH)) { … } によりスイッチが「(新たに) 押された」ことを検出して処理している。
- delay(10) などで適当な時間待っている。
  次回の loop() の呼び出しに備えて今回読み込んだスイッチの値（val に代入されている）を oldVal に代入（oldVal = val;）して保存する。



４．プログラムを少し変えてみる

先の回路に LED をもう一つ加え，10 番ピンからの出力で制御するようにします．片方は 2.2 節のプログラムと同じように，他方は 2.3 節のプログラムと同じように動作するプログラムを作成してみなさい．回路図を図４に示します．
Suwano ではドーターボードの LED3 を追加して使うことになります．

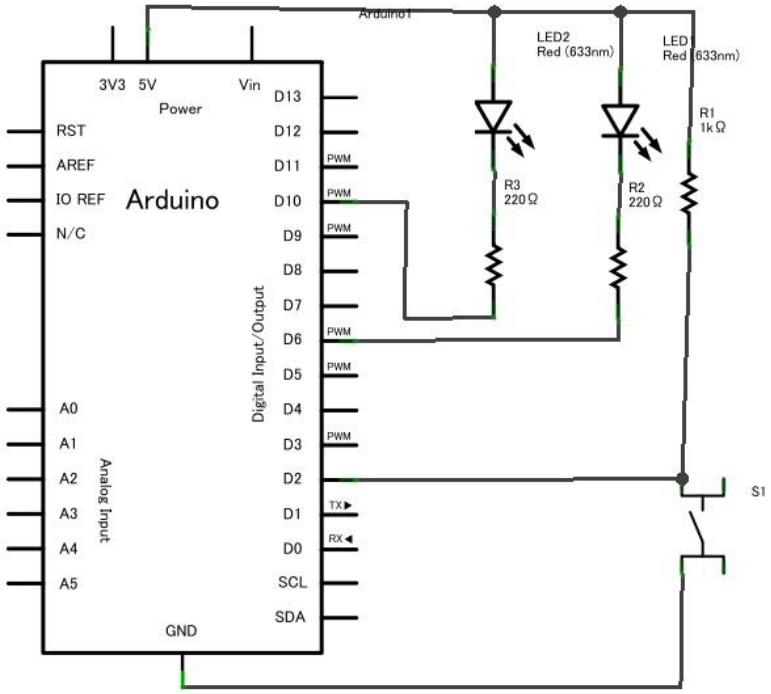


図４　LED を２つ使う回路

５．　LED の使い方

LED は適正な電圧，電流で使わなければなりません．

- LED の足の長いほうがアノード，短いほうの足がカソードと呼ばれ，点灯する際にはアノード側を高い電圧にします．
- 赤色の LED が点灯する電圧は 1.8V，緑や青色の LED が点灯する電圧は 3.5V 程度です．
- また砲弾型の LED に流すべき電流は 10mA 程度です．

このため，LED を直接，5V の電源線と GND につなぐと電流が流れすぎて LED を破損します．
適切な電流，電圧で使用するためには LED に直列に 150 Ω（緑，青），330 Ω（赤）といった値の抵抗器を直列につなぎます．
例えば 330 Ωの抵抗器に 10mA の電流を流すと電圧降下は 3.3 V になります．したがって 5V の電源からの差が 1.7 V になり，想定した電流値で概ね適正な電圧が LED に加わることになります．