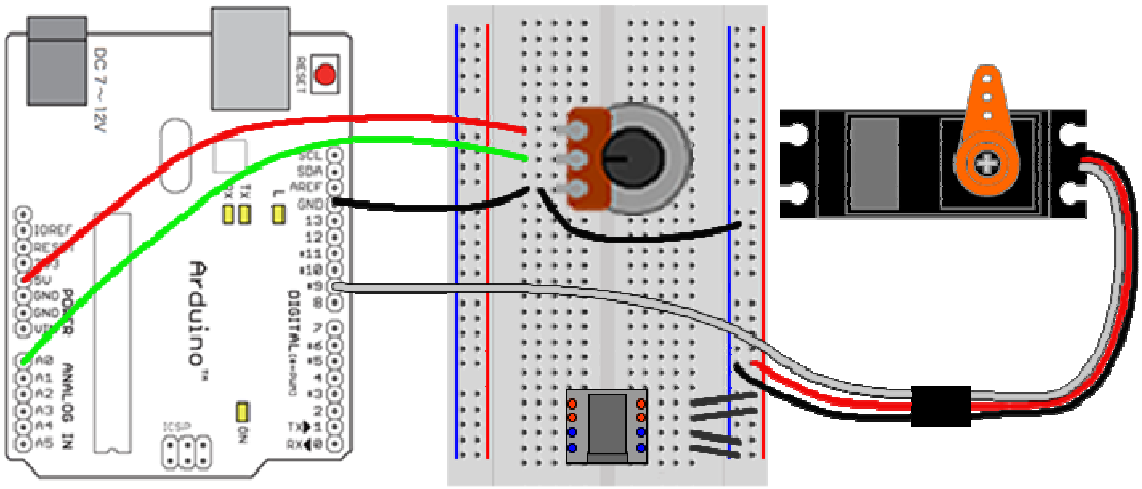
Arduino であそぼう



1. 目　標

Arduino を使って遊べるようになる。

・スケッチを書けるようになる。

・電子回路を配線できるようになる。

2. Arduino とは

Arduino マイコンボードと Arduino IDE（プログラム開発環境）の２つがセットになったものです。

（1）Arduino マイコンボード

図 1 は Arduino マイコンボード Arduino UNO R3 です。

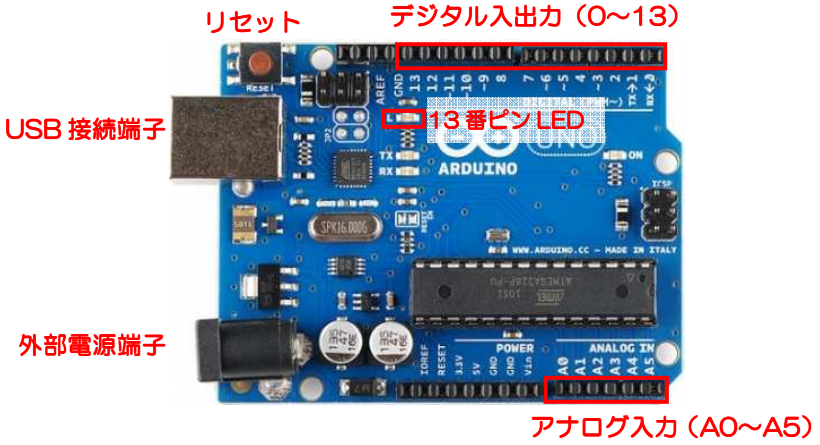


図 1

① デジタル入出力端子，アナログ入力端子

LED、センサ、アクチュエータ、その他部品を接続して色々な動作をさせることができます。

② USB 接続端子

パソコンとの通信を行います。USB 接続がされているときパソコンから５V 電源がボードに供給されます。

③ 外部電源端子

Arduino マイコンボードをパソコンから切り離して単独で動かす場合は外部電源端子にバッテリや AC アダプタ（6～20V／推奨は 7～12V）などを接続します。差し込むプラグは内径 2.1mm、外形 5.5mm で、中心軸がプラス極です。

⑵ Arduino IDE

Arduino IDE はフリーのプログラム開発環境で arduino.cc からダウンロードできます。Windows 用，Linux 用，Mac-OS 用が用意されています。図２は Arduino IDE の操作画面です。

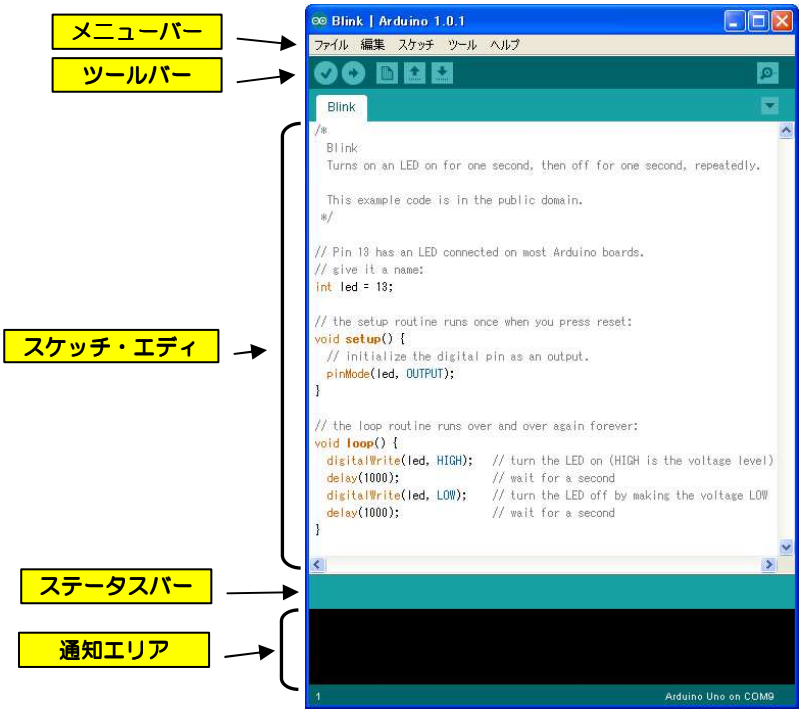


図 ２

① メニューバー：（説明省略）

② ツールバー：ボタンが配置されておりそれぞれの機能は次のとおりです。

	検証	スケッチのシンタックス（文法）チェックを行います。
	アップロード	スケッチをコンパイルして Arduino マイコンボードにアップロードします。
	新規作成	スケッチの新規作成画面を開きます。
	開く	既存のスケッチのファイルを開きます。
	保存	編集中のスケッチを保存します。
	通信モニタ	パソコンと Arduino マイコンボードのシリアル通信をモニタします。

※Arduino では慣例的にプログラムのことを『スケッチ』と呼んでいます。

③ スケッチ・エディタ：ここにスケッチを記述します。

④ ステータスバー：Arduino IDE の動作状況が表示されます。

⑤ 通知エリア：Arduino IDE からのメッセージが表示されます。

3. 最初の一歩（Ｌチカ）

課　　題：	ボード上の１３番ピン LED を点滅させる。
内　　容：	Arduino IDE の基本操作 スケッチの基本構造 デジタル出力
キーワード：	setup, loop, pinMode, digitalWrite, delay, int, OUTPUT, HIGH, LOW, 関数, 引数（ひきすう）

(1) Arduino IDE の基本操作

① Arduino マイコンボードの機種設定

Arduino UNO を USB に接続する。

「ツール」→「マイコンボード」→「Arduino UNO」

② シリアルポートの設定

「ツール」→「シリアルポート」→「COM△」(Windows の場合)

③ スケッチを開く

サンプルスケッチの Blink を開きます。

「開く」ボタン→「01.Basics」→「Blink」

④ 検証

「検証」ボタンをクリックします。

ステータスバーに「スケッチをコンパイルしています」→「コンパイル終了」

メッセージエリアに

「コンパイル後のスケッチのサイズ：1,084 バイト（最大容量 32,256 バイト）」と表示されます。

⑤ アップロード

「アップロード」ボタンをクリックします。

ステータスバーに「スケッチをコンパイルしています」→「コンパイル終了」→

「マイコンボードに書き込んでいます」→「マイコンボードへの書込みが完了しました」

Arduino ボードの LED(L)が点滅します。

⑥ エラーがあったとき

スケッチの 10 行目 int led = 13; の行末の ;（セミコロン）を削除しておいて,「検証」ボタンをクリックします。

ステータスバーとメッセージエリアにエラーが表示されます。

（エラーが出ることを確かめたら９行目をもとにもどしておきます）

⑵ スケッチの基本構造

Blink スケッチを見ると

/* と */ で挟まれた行 → コメント(注釈)

先頭が//の行 → 1 行コメント

末尾が；の行 → ステートメント（命令文）

○○○( ){ と } で挟まれた部分 → 関数

に分類できます。

コメントにはスケッチの動作や注意事項が記述されています。

ステートメントは Arduino ボードを動かす命令です。行末には必ず；を付けなければなりません。

関数はステートメントの集まりでひとつのまとまった動作を記述します。

関数の先頭には 関数名 と { を，最後には } を記述します。

⑶ setup 関数，loop 関数

setup 関数は Arduino ボードが起動したとき最初に 1 回だけ実行される関数です。ここには通常 Arduino ボードや周辺の回路の初期設定をするステートメントを記述しておきます。

loop 関数は setup 関数が 1 回実行された後に実行されます。

ステートメントは上から下へ順に実行され最後のステートメントの実行が終わると loop 関数の最初のステートメントにジャンプします。つまり，loop 関数の先頭行～最終行の間のステートメントが無限に実行されます。

⑷ デジタル出力

digitalWrite 関数を用いてデジタル出力― HIGH(5V)か LOW(0V)をデジタル入出力ピン(端子)から出力します。

20 行目 digitalWrite(led, HIGH);で HIGH を led ピンに出力しています。

ここで，led は 10 行目で int led = 13;と定義されているので，digitalWrite(**13**, HIGH); として働き 13 番ピンに 5V が出力されます。13 番ピンにはボード上の LED がつながっていますので，これが点灯します。

【重要】

デジタル入出力ピンを使う場合は，そのピンを出力で使うか入力で使うかあらかじめ設定しておかなければなりません。Blink スケッチでは setup 関数内で

pinMode(led, OUTPUT);

と記述し，led ――13 番ピンを OUTPUT(出力)と設定しています。

⑸ delay 関数

21 行目，23 行目の delay(1000);は( )の中に記述された値の時間だけ待ちます。時間単位はミリ秒(1000 分の 1 秒)です。delay(1000);は 1 秒間待ちます。

01.BASIC - Blink

```
1  /*
2  Blink
3  Turns on an LED on for one second, then off for one second, repeatedly.
4
5  This example code is in the public domain.
6  */
7
8  // Pin 13 has an LED connected on most Arduino boards.
9  // give it a name:
10 int led = 13;
11
12 // the setup routine runs once when you press reset:
13 void setup() {
14    // initialize the digital pin as an output.
15    pinMode(led, OUTPUT);
16 }
17
18 // the loop routine runs over and over again forever:
19 void loop() {
20    digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
21    delay(1000);               // wait for a second
22    digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
23    delay(1000);               // wait for a second
24 }
```

【練習１】

Blink スケッチを変更して点滅周期をいろいろに変えてください。

4. ブレッドボートを使う

課　　題：	デジタル出力９番ピンにつないだ LED を点滅させる
内　　容：	回路図の読み方 ブレッドボードの使い方・配線のし方 LED と電流制限抵抗の計算のし方
キーワード：	**ブレッドボード，アノード，カソード，電流制限抵抗，オームの法則**

Arduino ボードのデジタル入出力９番ピンに LED を接続して点滅させます。

（1）回路図

図３はデジタル入出力９番ピン(D9)に LED を接続する回路図です。９番ピンから抵抗を通して LED に接続されています。抵抗の働きについては後述します。
回路図は端子や部品を表す抽象的な記号―図記号を用いて電気的にどことどこが接続されているかを描いた図です。

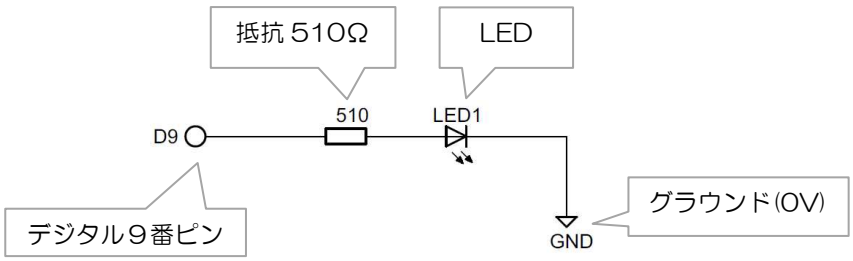


図 ３

(2) LED（発光ダイオード）

LED は電流を一方向にしか流さない特性があるため、＋と－を電源に対して正しくつながないと発光しません。図４のように、通常二つの端子（アノードとカソード）があります。アノード側をプラスに、カソード側をマイナスに接続します。

Arduino ボードのデジタルピンなどに直結すると過電流により LED または Arduino ボードが壊れる恐れがあるので抵抗を通して接続します。

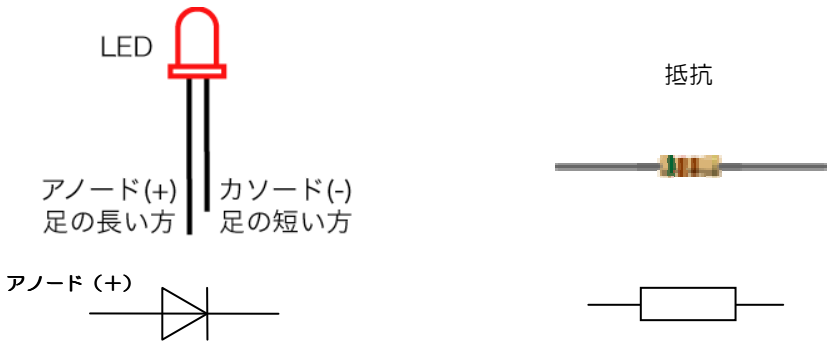


図 ４

⑶ 接続図

図５は実際の部品をどう配線接続するか表した接続図です。

ここで，配線図には無い LED の GND と Arduino の GND を結ぶ配線（赤い部分）がありますが，これは――

回路図では GND と GND を結ぶ配線は省略する。

実際に回路を組み立てるときは GND と GND は接続する。

という，暗黙の了解によるものです。

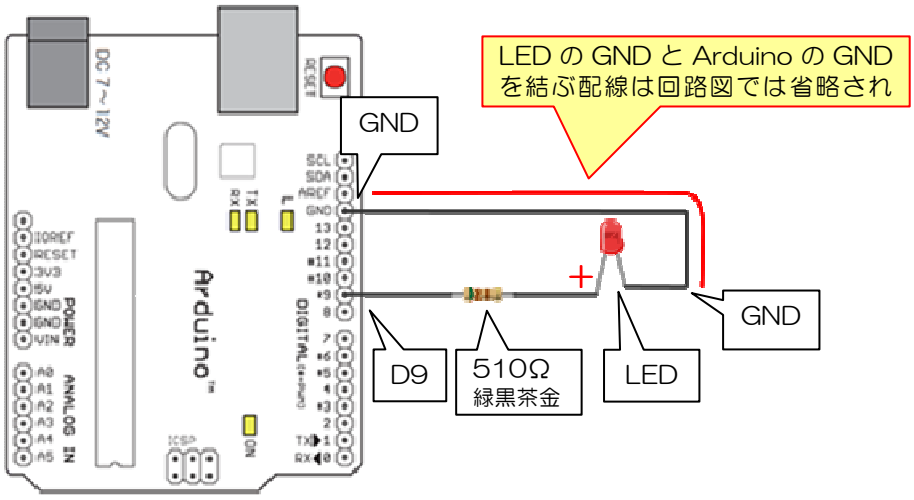


図 5

⑷ ブレッドボード

ブレッドボードは、はんだ付けせずに電子回路を組み立てることができます。

ブレッドボードの穴の下には金具（電極）が埋め込まれており、穴に挿入された部品やリード線をキャッチすると同時に電線として働きます。

図６は実態配線図をもとにブレッドボードでの配線を示したものです。。

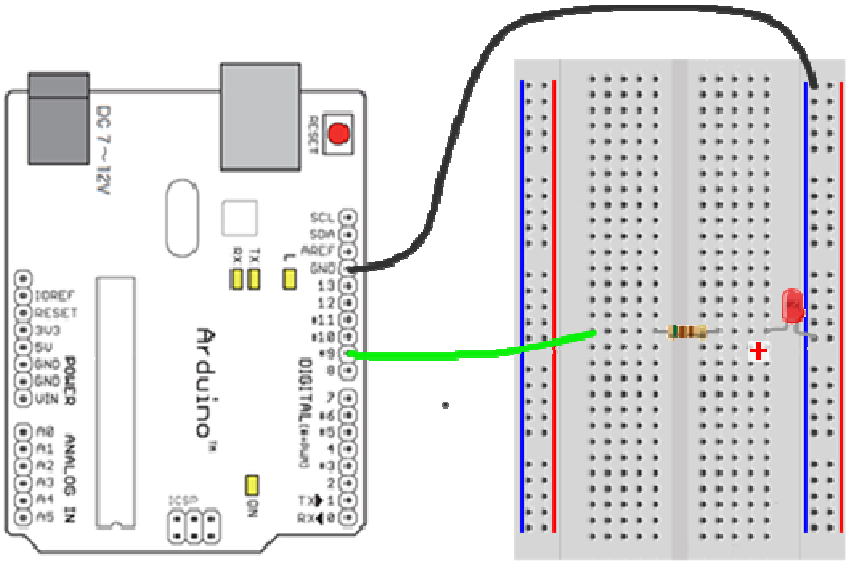


図 6

⑸ スケッチの修正と実行

Blink スケッチの 10 行目を `int led = 9;` と変更しアップロードするとブレッドボードに配線した LED が点滅するはずです。

⑹ スケッチの保存

「ファイル」→「名前を付けて保存」と進みます。ファイル保存のダイアログが開くので新しい名前を付けて保存します。

⑺ 抵抗（電気抵抗器）

抵抗はオームの法則にしたがった電流を流す部品です。この回路では LED に適切な電流を流すために挿入しています。抵抗には色のついた縞模様が描かれています。これはカラーコードといって何オームの抵抗かを色の組み合わせで表しています。
緑・黒・茶・金は 510Ω を表します。

オームの法則　　$電流 = \frac{電圧}{抵抗}$ [A]

degitalWrite の HIGH 電圧は約 5V なので，LED に流れる電流は

$$\frac{5\,V}{510\,\Omega} = 0.0098\ [A] = 9.8\,[mA] \qquad (0.001\,[A] = 1\,[mA])$$

【練習２】

図７の回路を Arduino―ブレッドボードに配線して，４個の LED を点灯制御してください。

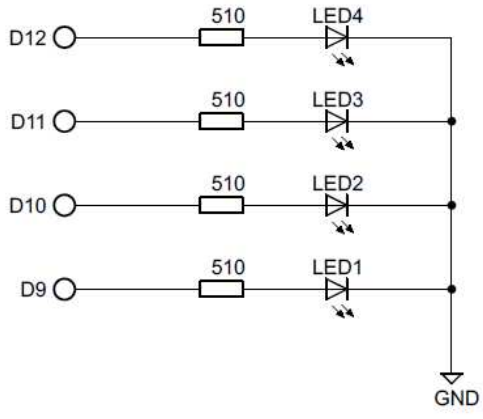


図 7

5. ふわふわ点灯(PWM)

課　　題：	D9 に接続した LED の明るさを制御する。

内　　容：	ＰWM制御のし方
キーワード：	anarogWrite, if, PWM

ＰWM（Puls Wide Moduration）で LED の明るさを制御します。

（1）回路図

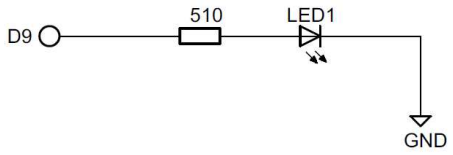


図 8

⑵ スケッチ

「開く」→「01.Basics」 →「Fade」

「アップロード」→ D9 に接続した LED がふわふわと明るさを増減して明滅します。

⑶ ＰWM（Puls Wide Modulation）

ＰWMとは周期は一定でパルス幅のデューティ比（パルス幅の HIGH と LOW の比）を変えて電力を制御する方法です。

図９の斜線部分がHIGHの部分で①に比べ②はデューティ比が大きいので②の方が電力が大きい――LED が明るく点きます。

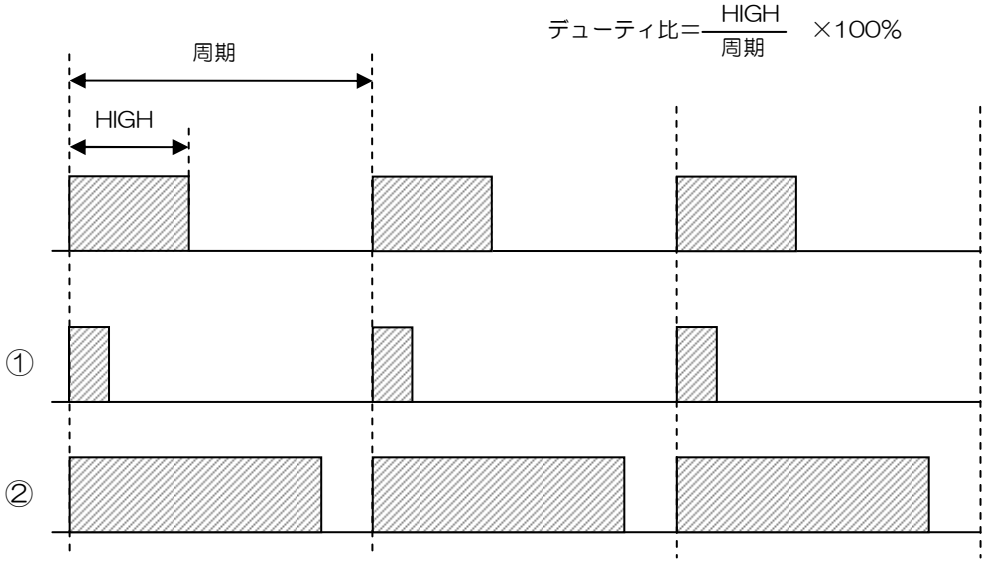


図 9

この節からステートメント（命令文）解説はインターネット上の「Arduino 日本語リファレンス」を参照します。Ｗｅｂブラウザを起動し次のサイトを開いてください。

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

⑷ アナログ入出力関数　analogWrire(pin,value)

指定したピンからアナログ値(PWM 波)を出力します。

pin: 出力に使うピンの番号

value: 0～255（デューティ比 0%～100%に相当）の 256 段階で PWM が可能

⑸ Fade スケッチ２６行目

```
brightness = brightness + fadeAmount;
```

は，変数 brightness に変数 fadeAmount の値を累積させる計算式です。

⑹ 制御文　ｉｆ

if 文は( )の中に与えられた条件式が満たされているかどうかをテストし,条件が満たされている（true である）とき，if 文に続く{　}内の文が実行されます。

条件式の例

x == y (x とｙは等しい)

x != y (x とｙは等しくない)

x < y (x はｙより小さい)

x > y (x はｙより大きい)

x <= y (x はｙ以下)

x >= y (x はｙ以上)

満たされている	→ true
満たされていない	→ false

⑺ ブール演算子　||（論理和）

```
brightness == 0 || brightness == 255
```

２つの値のどちらか一方でも true ならば全体で true となる。

⑻ Fade スケッチ３０行目

```
fadeAmount = -fadeAmount ;
```

は，変数 fadeAmount の符号（＋，－）を切り替える計算式です。

01.Basics - Fade

```
1  /*
2   Fade
3
4   This example shows how to fade an LED on pin 9
5   using the analogWrite() function.
6
7   This example code is in the public domain.
8   */
9
10 int led = 9;           // the pin that the LED is attached to
11 int brightness = 0;    // how bright the LED is
12 int fadeAmount = 5;    // how many points to fade the LED by
13
14 // the setup routine runs once when you press reset:
15 void setup()  {
16   // declare pin 9 to be an output:
17   pinMode(led, OUTPUT);
18 }
19
20 // the loop routine runs over and over again forever:
21 void loop()  {
22   // set the brightness of pin 9:


23   analogWrite(led, brightness);
24
25   // change the brightness for next time through the loop:
26   brightness = brightness + fadeAmount;
27
28   // reverse the direction of the fading at the ends of the fade:
29   if (brightness == 0 || brightness == 255) {
30     fadeAmount = -fadeAmount ;
31   }
32   // wait for 30 milliseconds to see the dimming effect
33   delay(30);
34 }
```

6. スイッチの読み取り

課　　題：	D7 に接続したスイッチでボード上の LED を ON-OFF する
内　　容：	スイッチの HIGH,LOW を読み取る
キーワード：	digitalRead, pinMode, ~(NOT)

(1) 回路図

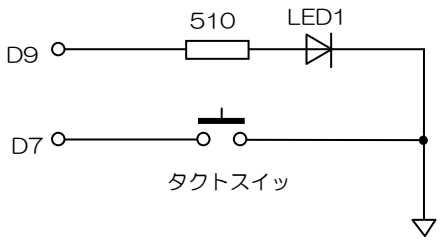


図 10

(2) ブレッドボード配線

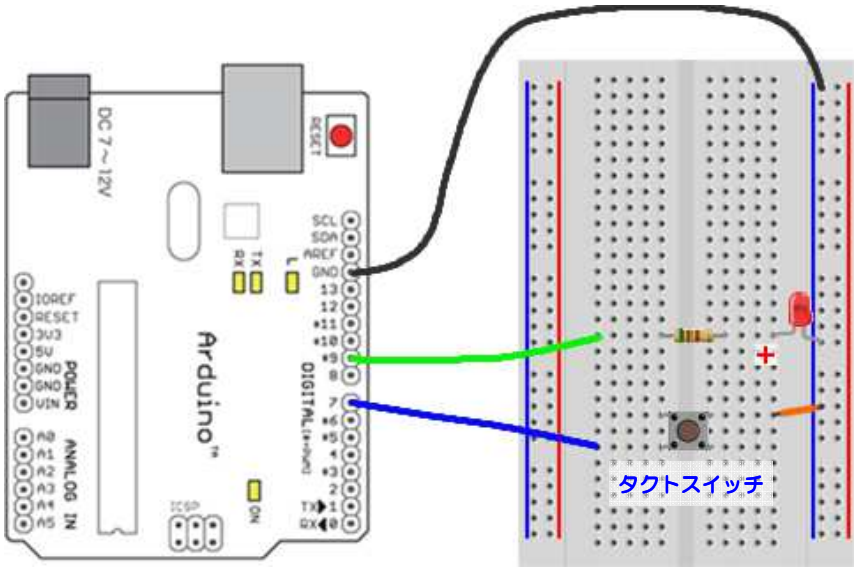


図 11

(3) デジタル入出力関数　　pinMode(pin, mode))

ピンの動作を入力か出力に設定します。

pin: 設定したいピンの番号

mode: OUTPUT（出力）

INPUT（入力）

INPUT_PULLUP（入力 内部プルアップ抵抗有効）

(4) デジタル入出力関数　　digitalRead(pin

指定したピンの値を読み取ります。その結果は HIGH または LOW となります。

switch_1

```
1  int ledPin = 13; // LED を 13 番ピンに
2  int inPin = 7;   // デジタルピン 7 にタクトスイッチ
3  int val = 0;     // 読み取った値を保持する変数
4
5  void setup() {
6    pinMode(ledPin, OUTPUT);   // LED 用に出力に設定
7    pinMode(inPin, INPUT_PULLUP);      // スイッチ用に入力に設定
8  }
9
10 void loop() {
11   val = digitalRead(inPin);  // 入力ピンを読む
12   digitalWrite(ledPin, val); // LED のピンを読み取った値に変更
13 }
```

switch_2

```
1  int ledPin = 13; // LED を 13 番ピンに
2  int inPin = 7;   // デジタルピン 7 にタクトスイッチ
3  int val = 0;     // 読み取った値を保持する変数
4
5  void setup() {
6    pinMode(ledPin, OUTPUT);   // LED 用に出力に設定
7    pinMode(inPin, INPUT_PULLUP);      // スイッチ用に入力に設定
8  }
9
10 void loop() {
12   val = !digitalRead(inPin);  // 入力ピンを読んで反転
13   digitalWrite(ledPin, val); // LED のピンを読み取った値に変更
14 }
```

⑸ ブール演算子　　！(否定)

false ならば true を、true ならば false を返します。

入出力ピンの HIIGH は LOW に、LOW は HIGH の値になります。（反転）

【練習】３

タクトスイッチ４個で LED４個を ON-OFF してください。

7. アナログデータの読み取り

課　　題：	A0 に接続した可変抵抗器で LED の点滅周期を調整する
内　　容：	アナログデータ（電圧）の読み取り方
キーワード：	analogRead, map

(1) 回路図

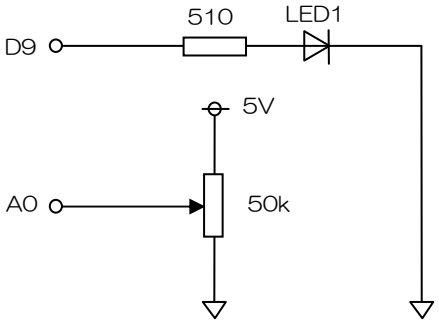


図 12

(2) ブレッドボード配線

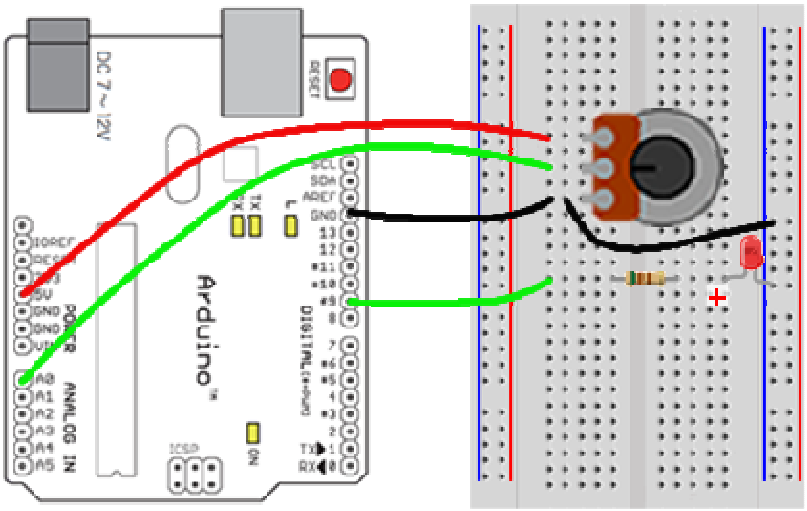


図 13

(2) アナログ入出力関数　analogRead(pin)

指定したアナログピンから値を読み取ります。0 から 5 ボルトの入力電圧を 0 から 1023 の数値に変換することができます。

03.Analog - Analoginput

```
/*
  Analog Input
 Demonstrates analog input by reading an analog sensor on analog pin 0 and
 turning on and off a light emitting diode(LED)  connected to digital pin 13.
 The amount of time the LED will be on and off depends on
 the value obtained by analogRead().
 */

int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:

  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

8. サーボモータを動かす

課　　題：	D にサーボモータを接続して動かす
内　　容：	Servo ライブラリの使い方
キーワード：	

(1) 接続図

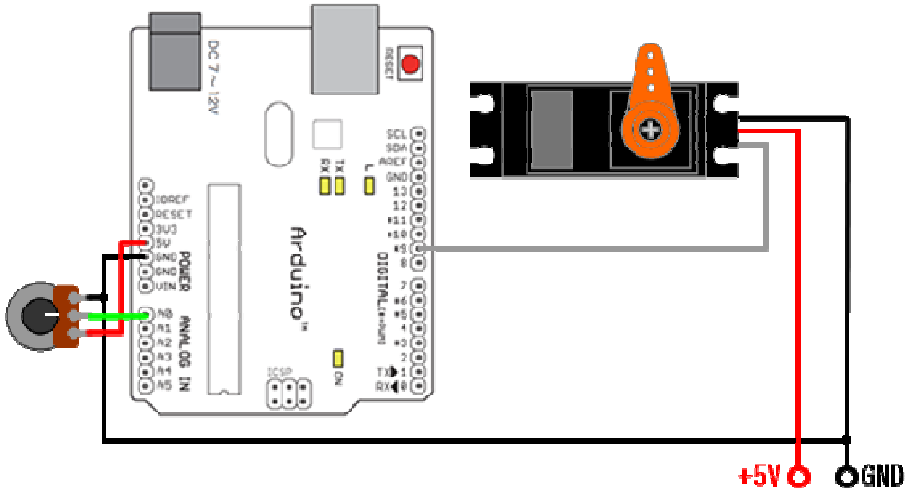


図 14

(2) ブレッドボード配線

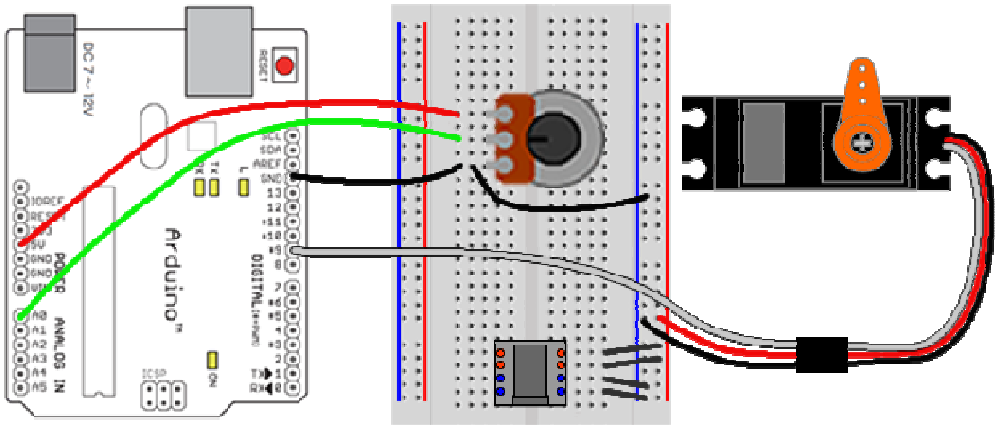


図 15

(3) ４行目　#include <Servo.h>

Servo ライブラリのファイル Servo.h を組み込みます。

(4) ６行目　Servo myservo

Servo 型の変数（オブジェクト）myservo を生成します。

(5) myservo.attach(pin)

サーボ変数をピンに割り当てます。

myservo: Servo 型の変数

pin: サーボを割り当てるピンの番号

(6) myservo.write(angle)

サーボの角度をセットし、シャフトをその方向に向けます。

myservo: Servo 型の変数

angle: サーボに与える値(0 から 180)

(7) 数学的な関数　map(value, fromLow, fromHigh, toLow, toHigh)

数値をある範囲から別の範囲に変換します。

value: 変換したい数値

fromLow: 現在の範囲の下限　　fromHigh: 現在の範囲の上限

toLow: 変換後の範囲の下限　　toHigh: 変換後の範囲の上限

Servo - Knob

```
1  // Controlling a servo position using a potentiometer (variable resistor)
2  // by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>
3
4  #include <Servo.h>
5
6  Servo myservo;  // create servo object to control a servo
7  int potpin = 0;  // analog pin used to connect the potentiometer
8  int val;    // variable to read the value from the analog pin
9
10 void setup()
11 {
12   myservo.attach(9);  // attaches the servo on pin 9 to the servo object
13 }
14
15 void loop()
16 {
17   val = analogRead(potpin);
18   val = map(val, 0, 1023, 0, 179);
19   myservo.write(val);   // sets the servo position according to the scaled
   value
20
21   delay(15);            // waits for the servo to get there
22 }
```

9. 音を出す

(1) 回路図

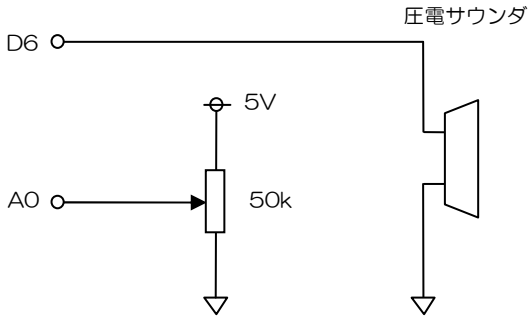


図 16

(2) ブレッドボード配線

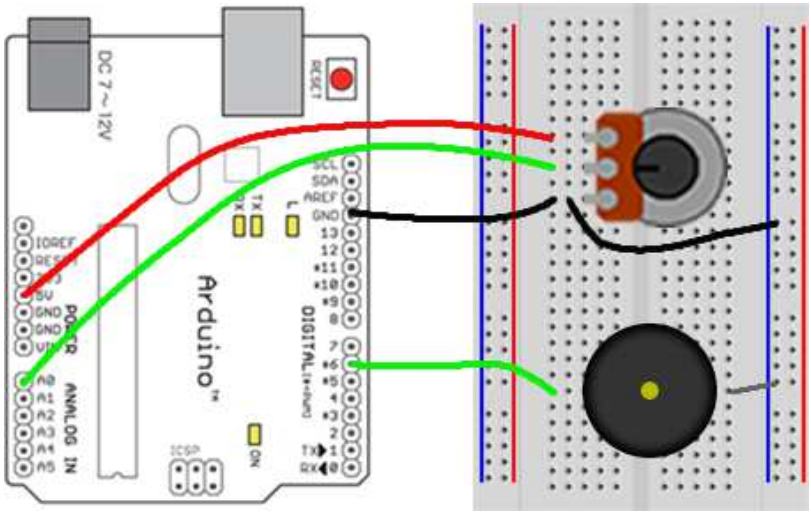


図 17

(3) tone(pin, frequency)

frequency で指定した周波数の矩形波(50%デューティ)を pin に出力します。時間(duration)を指定しなかった場合、noTone()を実行するまで動作を続けます。出力ピンに圧電ブザーやスピーカに接続することで、一定ピッチの音を再生できます。

同時に生成できるのは 1 音だけです。すでに他のピンで tone()が実行されている場合、次に実行した tone()は効果がありません。同じピンに対して tone()を実行した場合は周波数が変化します。

※この関数はピン 3 と 11 の PWM 出力を妨げます。

tone(pin, frequency, duration)

pin: トーンを出力するピン

frequency: 周波数(Hz)

duration: 出力する時間をミリ秒で指定できます(オプション)

oto

```
1  int sp = 6;
2  int potpin = 0;
3
4  void setup() {
5  }
6
7  void loop() {
8    int val;
9    val = analogRead(potpin);
10   val = map(val, 0, 1023, 100, 3000);
11   tone(sp,val);
12 }
```

oto2

```
1  int beat = 300;   // 音の長さを指定
2  int sp = 6 ;  // 圧電スピーカを接続したピン番号
3
4  void setup() {
5  }
6  void loop() {
7       tone(sp,262,beat) ;  // ド
8       delay(beat) ;
9       tone(sp,294,beat) ;  // レ
10      delay(beat) ;
11      tone(sp,330,beat) ;  // ミ
12      delay(beat) ;
13      tone(sp,349,beat) ;  // ファ
14      delay(beat) ;
16      tone(sp,392,beat) ;  // ソ
16      delay(beat) ;
17      tone(sp,440,beat) ;  // ラ
18      delay(beat) ;
19      tone(sp,494,beat) ;  // シ
20      delay(beat) ;
21      tone(sp,523,beat) ;  // ド
22      delay(2000) ;            // 2秒後に繰り返す
23 }
```

液晶表示

課　　題：	LCD にメッセージを表示させる。
内　　容：	LCD ライブラリの使い方
キーワード：	#include, ライブラリ, sprintf

(1) 接続図

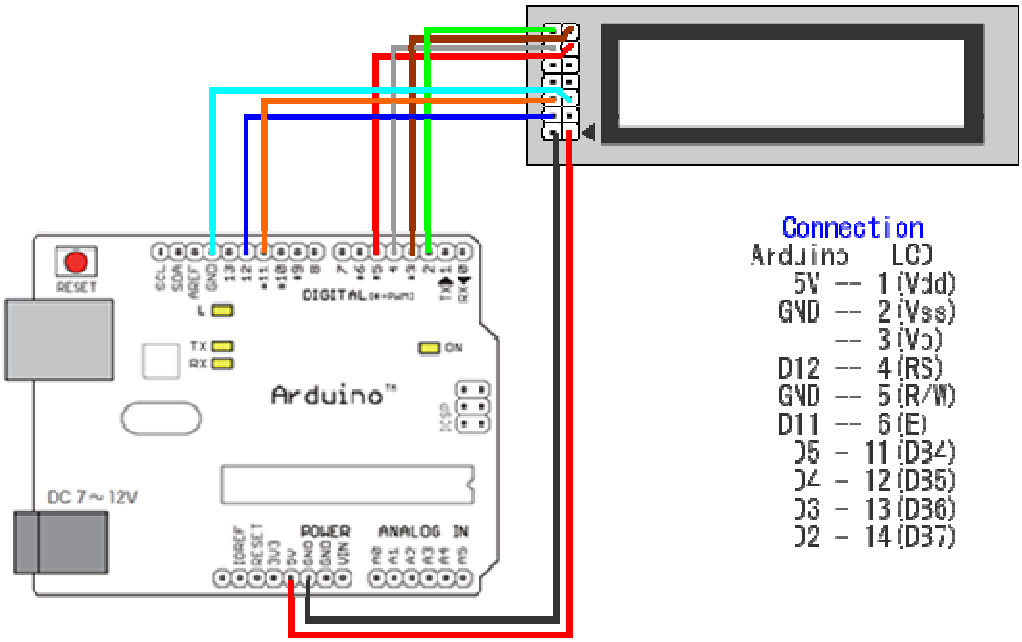


図 18

(2) LiquidCrystal

LiquidCrystal lcd(rs, enable, d4, d5, d6, d7);

LiquidCrystal 型の変数（オブジェクト）lcd を生成します。

(3) lcd.begin(cols, rows)

ディスプレイの桁数と行数を指定します。

cols: 桁数(横方向の字数)

rows: 行数

(4) lcd.print(data″)

テキストを LCD に表示します。

data: 表示したいデータ（char, byte, int, long, string の各型）

(5) lcd.setCursor(0, 1)

LiquidCrystal - Hello World

```
/*
  LiquidCrystal Library - Hello World

*/

// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis()/1000);
}
```

データの型

型指定	データ型	扱える数値の範囲
boolean	ブール型	true か false
char	文字型	-128～127
byte	byte 型	0～255
int	整数型	-32768～32767
long	倍長整数型	-2147483648～2147483647
float	単精度浮動小数点型	3.4E-38～3.4E+38
double	倍精度浮動小数点型	1.7E-308～1.7E+308
String	文字列型	

10.温度センサー

(1) ブレッドボード配線

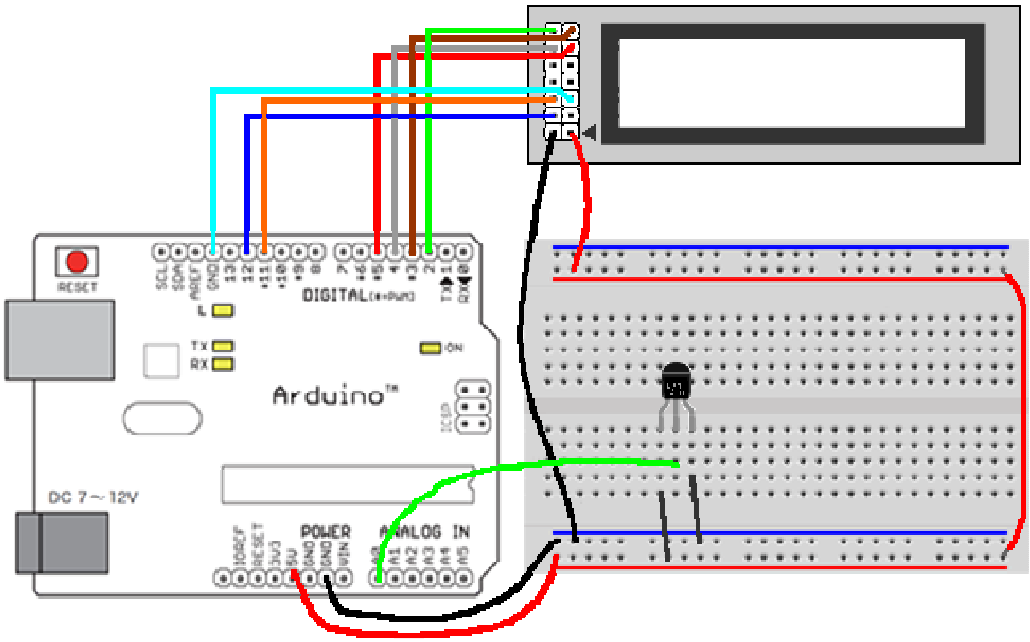


図 19

(2) 温度センサー部回路図

図 20

(3) 温度センサー　LM61BIZ

測定範囲：－２５℃～＋８５℃

温度係数：＋１０ｍＶ／℃

動作電圧範囲：＋２．７～＋１０Ｖ

(4) スケッチ ondo

(5) analogReference(type)

アナログ入力で使われる基準電圧を設定します。analogRead 関数は入力が基準電圧と同じとき 1023 を返します。

DEFAULT: 電源電圧(5V)が基準電圧となります。これがデフォルトです

INTERNAL: 内蔵基準電圧を用います。ATmega168 と 328P では 1.1V です

EXTERNAL: AREF ピンに供給される電圧(0V～5V)を基準電圧とします

11.光センサー

(1) ブレッドボード配線

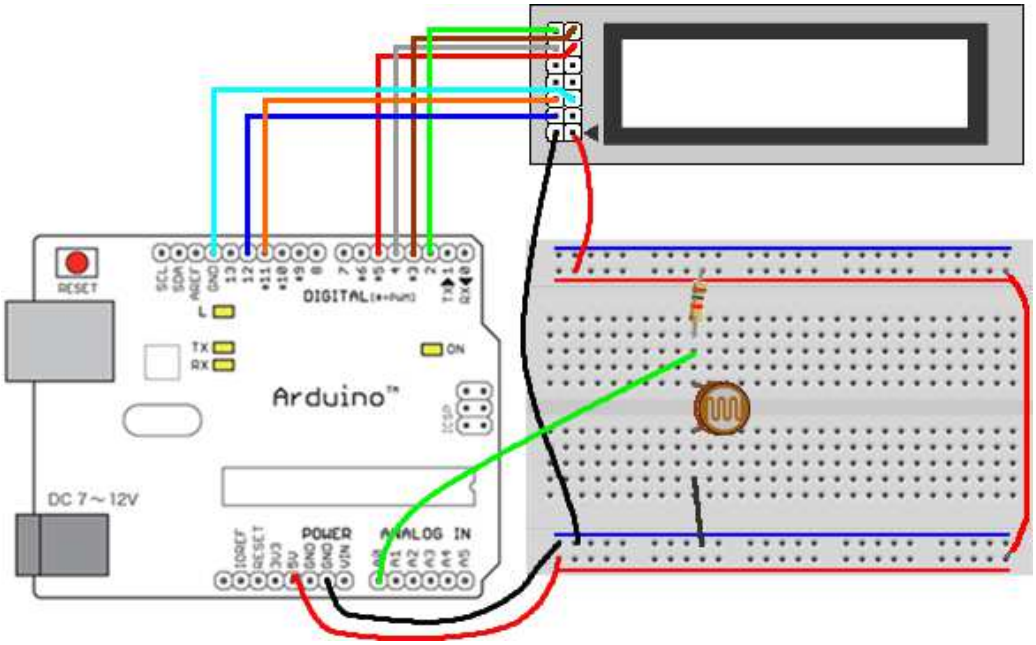


図 21

(2) 光センサー部回路図

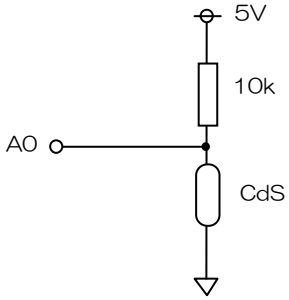


図 22

(3) スケッチ hikari

(4) CdS

CdS は光の強弱によって抵抗値が変化する光センサーです。

光弱――抵抗大

光強――抵抗小

12.算術演算子

(1) ＋　－　＊　／

　これらの演算子は、２つ以上の値の加算、減算、乗算、除算の結果を返します。その動作はデータの型に従います。たとえば、９と４が int(整数)であるとき、９ / ４の答えは２となります。これは、それぞれのデータ型に許されている値より大きな結果が得られたときに、オーバーフローが生じることも意味します。たとえば、int 型の 32,767 に 1 を足すと、-32,768 となります。

異なる型を組み合わせたときは、大きい方の型にもとづいて計算されます。また、一方の型が float か double のときは、浮動小数点演算が行われます。

答 = 値１ + 値２;
答 = 値１ - 値２;
答 = 値１＊ 値２;
答 = 値１ / 値２;

(2) ％　（剰余）

整数の割り算を行ったときの余りを返します。

答 = 値１％ 値２;
x = ７％５;　// x は２に
x = ９％５;　// x は４に
x = ５％５;　// x は０に
x = ４％５;　// x は４に

(3) ++ (加算) -- (減算)

変数に対し 1 を加算(++)・減算(--)します。
演算子を変数の左右どちらに置くかで意味がかわります。
x++; // x を返し、1 を加えます
++x; // 1 を加えた x を返します

x--; // x を返し、1 を引きます
--x; // 1 を引いた x を返します

13.制御文

Arduino で色々なものを動かすとき，センサーから取り込んだ値によって処理（仕事）を切り替える必要が出てきます。また，何回か同じことを繰り返したあと次の処理に進むようなスケッチが必要となる場合があります。このような判断分岐。繰返しを簡単に行わせる命令文を制御文といいます。

(1) if else

if else 文を使うと複数のテストをまとめることでき、単体の if より高度な制御が可能となります。

次の例は、アナログ入力の値が 500 より小さいときと、500 以上のときに分けて、それぞれ別の動作を行うものです。

```
if (pinFiveInput < 500) {
  // 動作 A
} else {
  // 動作 B
}
```

else に続けて if を書くことで、さらに複数のテストを書くことができます。

```
if (pinFiveInput < 500) {
  // 動作 A
} else if (pinFiveInput >= 1000) {
  // 動作 B
} else {
  // 動作 C
}
```

true となる if 文にぶつかるまで、テストは続きます。true となった if 文の波カッコ内が実行されると、その if else 文全体が終了します。true となる if 文がひとつもない場合は、(if の付いていない)else 文が実行されます。

else if を使った分岐は好きなだけ繰り返して構いません。

似たことを実現する別の方法として、switch case 文があります。

(2) for

for 文は波カッコに囲まれたブロックを繰り返し実行します。様々な繰り返し処理に活用でき、データやピンの配列と組み合わせて使われることがあります。

for 文のヘッダは３つの部分から成り立っています。

```
for (初期化; 条件式; 加算) {
  //実行される文;
}
```

まず初期化が一度だけ行われます。処理が繰り返されるたびに条件式がテストされ、true ならば波カッコ内の処理が実行され、加算が行われます。次に条件式がテストされたときに false ならば、そこでループは終了します。

LED をふわふわと明滅させるサンプル。for ループの中で、PWM のパラメータを 0 から 255 まで 1 ずつ上げている。

```
int PWMpin = 9; // LED を 9 番ピンに 510Ωの抵抗を直列にして接続

void setup() {
  // 初期化不要
}

void loop() {
   for (int i=0; i <= 255; i++){
      analogWrite(PWMpin, i);
      delay(10);
   }
}
```

(3) while

while は繰り返しの処理に使います。カッコ内の式が false になるまで、処理は無限に繰り返されます。条件式で使われる変数は、while ループの中で、値を加えるとかセンサの値を読むといった処理により変化する必要があります。そうしないと、ループから抜け出すことができません。

```
while(条件式){
  // 実行される文
}
```

単純な繰り返しの例です。

```
var = 0;
while(var < 200){
  // この部分が 200 回繰り返される
  var++;
}
```

(4) switch case

switch case 文は if 文と同じようにプログラムの制御に使われ、場合分けの記述に適しています。switch()で指定された変数と一致する case の文が実行されます。

変数 var がテストしたい変数です。その値がどれかの case に一致すると、それに続く文が実行されます。default:は、どの case にも一致しなかったときに実行されます。処理が終わったら、break を使って swithc 文から抜け出す必要があります。break がないとそのまま続けて次の case が実行されてしまいます。

```
switch (var) {
  case 1:
    // var が 1 のとき実行される
    break;
  case 2:
    // var が 2 のとき実行される
    break;
  default:
    // どの case にも一致しなかったとき実行される
    // (default は省略可能)
}
```

14.作って見ましょう

⑴ デジタル表示温度計
温度センサーの値を温度に変換して LCD に表示

⑵ アナログ温度計
温度センサーの値を角度に変換してサーボを駆動

⑶ 暗くなると点灯，明るいと消灯する LED
光センサーの値がある値以上になったら点灯

⑷ LED の明るさを手動で調節する。
可変抵抗器の値で LED を PWM 制御。
または，Up と Down のスイッチ設けて，Up を押すと明るく Down を押すと暗くなる。

⑸ フルカラーLED イルミネーション
時間とともに色が変化するイルミネーション
フルカラーLEDは図23のように内部接続されています。

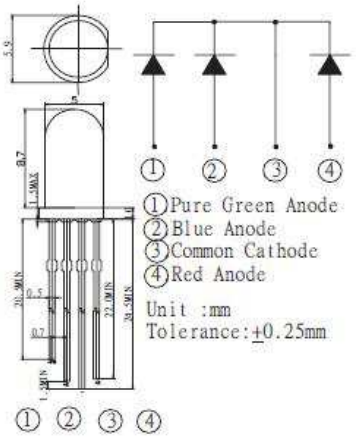


図 23

Arduino であそぼう

著者・発行者：小坂貴美男
kim.kosmac@gmail.com

初　版　2012 年7月 13 日