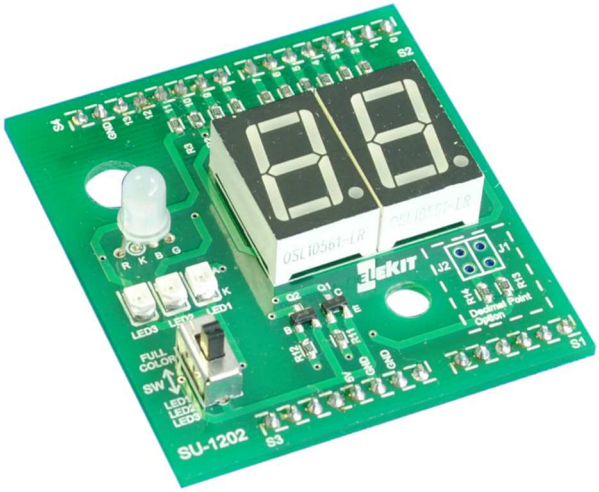
Arduino ビギナーのための

LED 表示制御入門



株式会社　イーケイジャパン

はじめに

電子回路を使って何かを作ろうと思ったとき、マイコンと電子部品を使うととても便利になります。今までは多くの電子部品を並べてつないで、それぞれの動作を理解して回路を設計し、希望の機能を作る必要がありました。さらに機能を変更したいときには、大変な手間をかけて回路を修正しなければなりませんでした。しかし、マイコンの登場で高度な動きをわずか数点の電子部品で実現できるようになり、動作の修正も簡単になりました。だだし、マイコンを使いこなすには専門的な知識や言語の理解、専用の機器を用意する必要があり、ハードルが高かったのですが、最近では例えば Arduino のようなマイコンボードが登場し、比較的簡単な言語で、希望の動作を得られるようになりました。

マイコンで電子部品を制御できるようになると、思いついたことをどんどん試すことができます。しかし、「電子部品」を使うためには、どうしてもその特徴や使い方を知っておく必要があります。電子部品を知るための近道は、やはり、実際に触って動かしてみることです。さらに、電子部品の特徴やポイントを知っておくと、余計なことに悩まされることなく、自分の作りたいモノに集中できるようになります。

本製品は、実際の部品の動作を確認しながら、電子部品の特徴や解説を読むことで効率的に学習できるセットです。マイコンには Arduino ボードを利用します。説明を読みながら、実際に動作させ、結果を確認していくことで、プログラムで電子部品を動かすときのポイントがわかるようになっています。本製品があなたのアイデアを創造する手助けになればと思います。

重要

本機(SU-1202：表示シールド)を動作させるためには、Arduino 基板が必要です。
Arduino 基板には多くの種類が発売されていますが、本書の説明には Arduino-UNO(R3)を使っています。
また、Arduino 基板の接続にはご使用の環境に合わせた USB ケーブルを準備してください。

●おことわり

本書では Arduino 基板の説明や、プログラム開発環境の入手～セットアップ方法、プログラム用命令の説明については詳しく記載していません。本格的に Arduino のプログラムを学びたい場合は、Arduino のサイト http://www.arduino.cc/や、インターネット、書籍などの説明や解説を参考にして学習してください。

もくじ

４．LCD を制御する

Arduino 基板と Arduino-IDE

（3）　Arduino の使い方

準備

●用意するもの

（１）　Arduino-UNO (R3)　　表示シールド(SU-1202)を動作させるために必要です。

（２）　表示シールド　　SU-1202

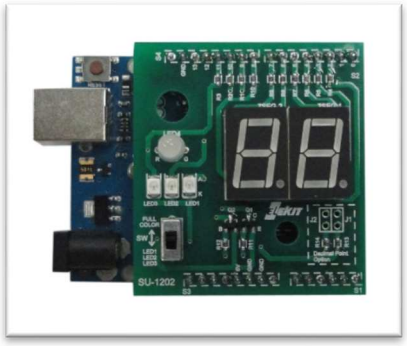
（３）　USB ケーブル　　Arduino 基板とパソコンを接続します。

（４）　パソコン　　Arduino プログラム開発環境がインストールされているパソコン

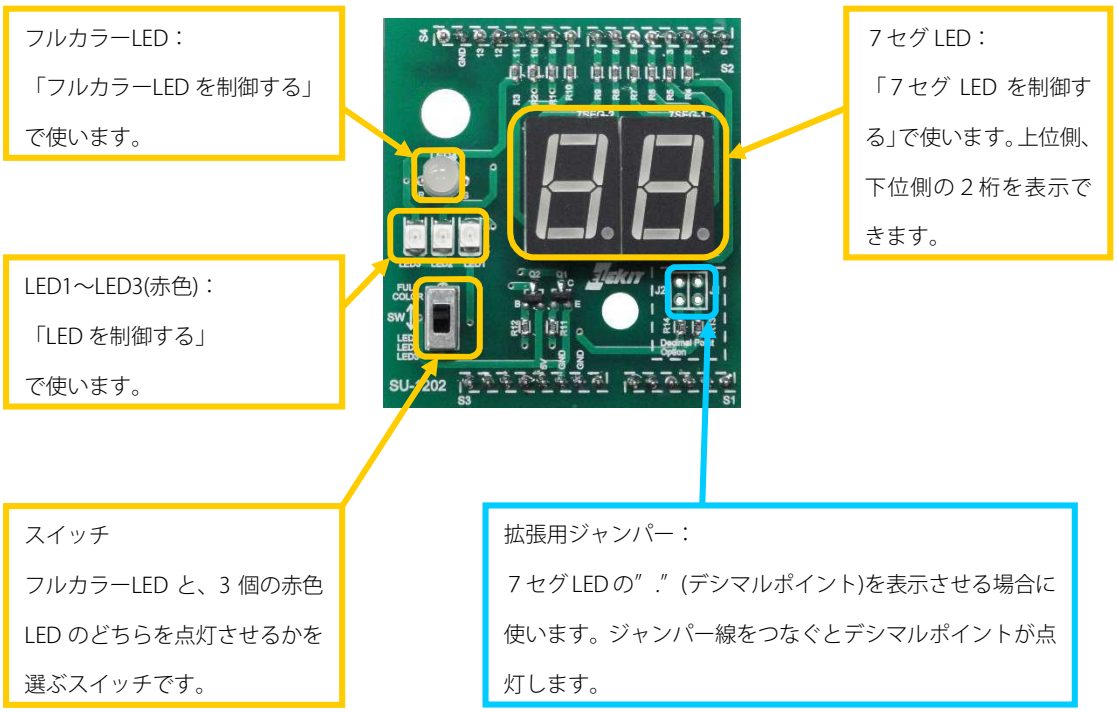
（※プログラム開発環境の入手～インストールがまだの場合は、巻末をご参照の上インストールしてください。）

●Arduino と表示シールドの接続

Arduino のコネクタに表示シールドのピンを差し込みます。
向きを確認し、ピンの位置を合わせて、ピンが曲がらないように
注意して差し込んでください。



●表示シールドについて

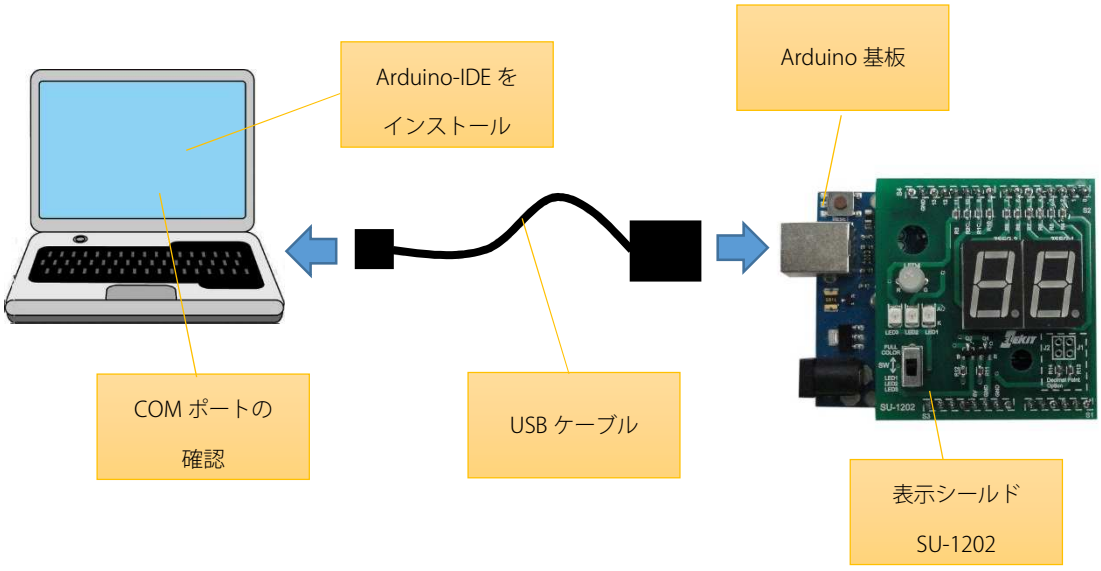


※電源には Arduino 基板の 5V を利用しています。

●パソコンとの接続

本書の「Arduino 基板と Arduino-IDE」を参考に、パソコンに Arduino の開発環境 Arduino-IDE をインストールし、USB ケーブルでパソコンと Arduino 基板を接続します。

COM ポートの設定・確認も忘れないようにします。



1．LED を制御する

（1）　『LED を光らせる』

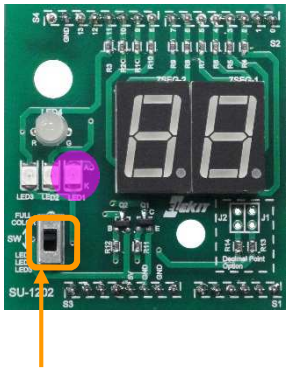
最初の一歩として、LED を光らせます。単純なことですが、LED を光らせる場面は多くあります。LED をランプとして使うこともありますが、電子工作やプログラムでは、動作の確認やプログラムが意図したとおりに動いているか確認表示として使うことも多いでしょう。LED の使い方や特性を知っておくと、より便利になります。

●プログラム

次のプログラムを作成して、マイコンにアップロードします。

※アップロードの方法は巻末を参照ください。

```
void setup(){
   pinMode(9, OUTPUT);
}
void loop(){
   digitalWrite(9, HIGH);
}
```

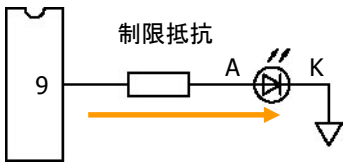


この SW は LED 1～3 側にしておきます。

プログラムをアップロードすると LED1 が光ります。

●解説

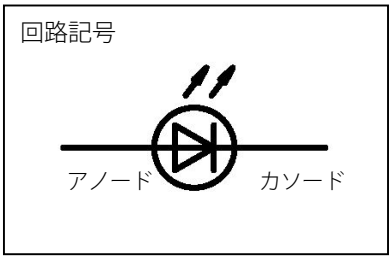
プログラムに digitalWrite（9,HIGH）と書くことで、マイコンの 9 番ピンが H になります。すると、9 番ピンに接続された LED に電流が流れて LED が点灯します。

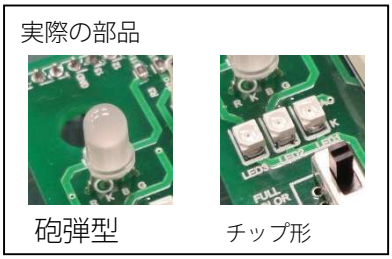


LED には極性があり、電流を流す向きを間違えると点灯しません。アノードからカソードに向かって電流が流れるように接続します。また、LED にはこれ以上流してはいけない電流の値（＝最大定格電流）が決められており、定格を越えないように抵抗を入れて電流を調整しています。電流の量を制限するので制限抵抗といわれます。

●LED とは

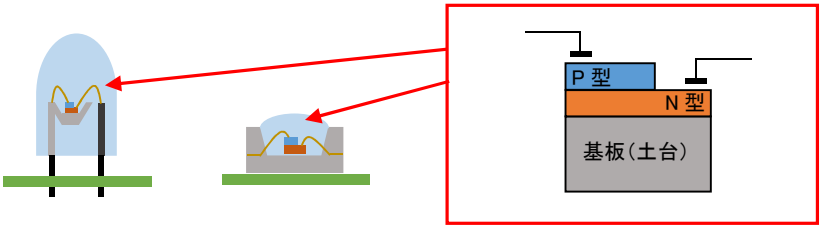
・回路記号



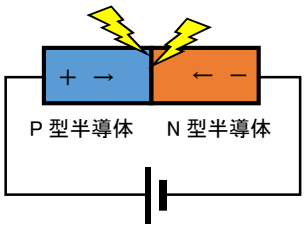


・LED 発光のしくみ

LED の内部は図のような構造になっています。特性の異なった 2 つの半導体（P 型半導体と N 型半導体）が接合された「PN 接合」で構成されます。これを LED チップといいます。



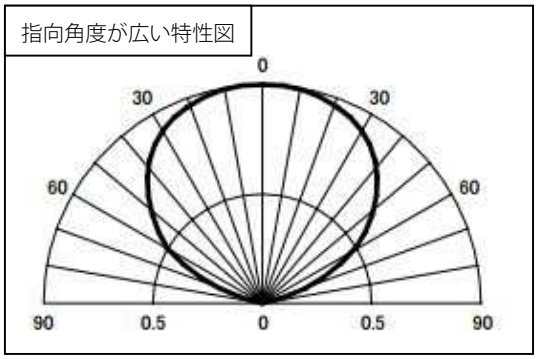
LED チップに決まった方向から電圧をかけると、
LED チップの中で「再結合」という現象が起き、
その時に生じた余分なエネルギーが
光のエネルギーとなり発光します。

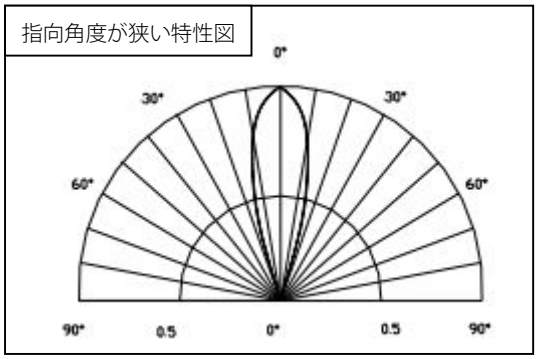


●使い方、ポイント

・指向角度、レンズ色

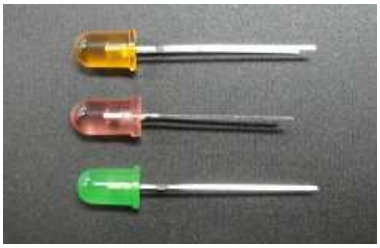
LED には様々な指向角度やレンズ色のものがあります。使用する用途に合うものを選びます。
指向角度は（下図）光りの広がり方のことで、指向角度が狭いと LED を真正面から見たときには明るく、遠くからでも光りを見ることが出来ます。しかし、LED を横側見たときは、光っていることをほとんど確認できません。





広い範囲で見ることが必要なときには、レンズ色が透明なものよりレンズに色の付いた拡散タイプ（写真）のものが使いやすいときがあります。

拡散タイプの LED の例



・発光色と順方向電圧

LED には色々な発光色があり、色によって動作させる電圧が違っています。点灯させるための電圧は「順方向電圧」を目安とします。一般的に、赤、黄は 1.8V 程度、青や白は 3.5V 程度のことが多いようです。そのため例えば乾電池で LED を光らせるときは、赤・黄の LED は 2 本で光りますが、青・白の LED には 3 本以上が必要になります。乾電池 1 本では LED は光らないと思った方がよいでしょう。

・制限抵抗の値の決め方

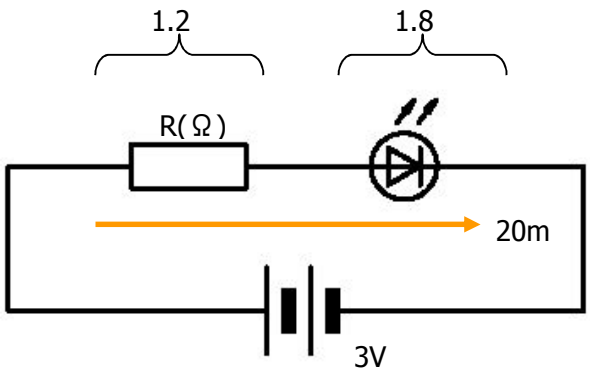
LED を光らせるときは、カタログやデータシートに記載されている、「順方向電流」と「順方向電圧」を目安にして流す電流を決めます。具体的にデータシートには、

順方向電圧　V_F=1.8V (IF=20mA のとき)

というように記載されています。この順方向の電圧値は LED の色や種類で変わります。

例として、電池 2 本(3V)を使って 1.8V（IF=20mA 時）と書かれている LED に 20mA を流すときの制限抵抗の値を求めてみます。

まず電流のことを考えます。LED に 20mA を流すということは、同じ電線上にある抵抗にも 20mA が流れます。次に電圧のことを考えます。先ほど説明したように、LED に 20mA 流れているときは LED の両端に 1.8V の電圧が掛かります。ここまでを整理すると、以下の図のようになります



ここで、電源が 3V であり、LED が 1.8V 使っていますので、残りの 1.2V が抵抗に掛かることが分かります。
抵抗に流れる電流値と電圧が分かりましたので、オームの法則を使えば抵抗の値が分かります。

オームの法則は　R(Ω)　＝　E (V) ÷ I (A) ですので、

R(Ω)　＝　1.2 (V) ÷ 0.02(A)

＝　60 (Ω)　　となり、

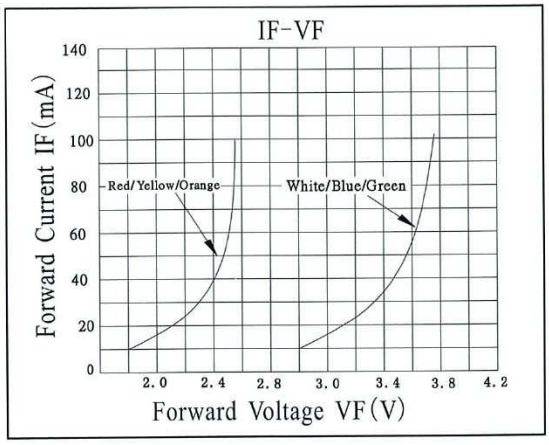
60Ω の抵抗を使えばよいことが分かります。
実際には 60Ω ちょうどの値の抵抗は売っていないので、値の近い、68Ω や 82Ω を使うことになります。

・最大定格

最大定格とは、一瞬でもそれ以上の電流・電圧を加えると部品が壊れてしまうという値です。LED の場合は、焼き切れてしまうか、すぐには壊れなくても、使用しているうちにだんだん暗くなるなど、寿命が著しく短くなります。部品は最大定格の範囲内で使わなければいけません。
日本語では最大定格、英語では Absolute Maximum Rating と書かれています。
もう一つ気を付けたいのは、その最大定格が適用される温度が「Ta=25℃」というように書かれていることです。
これは部品が動いているときの周囲温度が 25℃の時の最大定格ということです。一般的に部品の周囲温度が上がると最大定格はより厳しくなります。例えば Ta=25℃のときの最大定格が 100mA ならば、周囲温度が 80℃になると、流してよい電流は 50mA になるようなイメージです。
発熱が少ないといわれている LED であっても、電流を流すと発熱して温度が上がりますので、最大定格ギリギリで使用していると、最大定格を越えることになります。
実際に回路を設計して流れる電流を決めるような場合は、温度上昇も考慮してその部品の最大定格の 50%～70%程度にするように気を付けるとよいでしょう。

・順方向電圧、順方向電流

LED には極性があり、LED を光らせるためには、アノードに＋を、カソードにーを接続します。このように LED が光る向きの電圧のことを順方向電圧といいます。
順方向電流とは、その名前の通り順方向に電圧が掛かったときに流れる電流のことです。一般的には図のような特性になります。図では、赤色の LED の順方向に 20mA の電流が流れている場合は、LED の両端に発生する順方向電圧は 2.1V になり、30mA の電流が流れると順方向電圧は 2.3V になることが分かります。



●コラム１

「コイン電池で光るライトに抵抗が入っていないのは何故？」

電池には、「内部抵抗」という抵抗があります。この抵抗は実際に抵抗が入っているのではなく、電池を作るときにどうしてもできてしまう電気の流れを妨げる成分のことです。この「内部抵抗」は電池の構造、電池の種類によって、抵抗の値の大きさは違っています。

一般的に、乾電池は、数 Ω、コイン電池は数十 Ω といわれています。

コイン電池の内部抵抗によりある程度電流が制限されるので、制限抵抗を使わずに、LED を光らせているものがあるのです。

内部抵抗は、電池の消耗具合によっても変わりますし、ちゃんと管理されたものではないので、実験用でいつ壊れても OK だというように割り切った用途に使うのであればよいかもしれませんが、壊れないように長くしっかり動かしたいというような用途の場合は、制限抵抗を入れなければいけません。

●コラム２

「オームの法則を使って抵抗を決めるなんてめんどうくさい。」

電圧が変わっても一定の電流しか流さない「定電流ダイオード」という部品もあります。CRD といわれることもあります。本来 LED は一定の電流で使うのが理想なので、非常に相性のよい部品です。

LED と CRD を図のように接続するだけです。電圧が少々変わっても常に一定の電流を流します。注意点としては、逆方向に使う単なる線と同じように電流を流してしまうこと、部品が機能するために余計に電圧が必要なこと。そして、価格が抵抗よりも随分高いことです。

●コラム３

「マイコンの H、L とは？」

よく、「マイコンのポートが H になります。」などと表現されますが、H になる、または L になるとは具体的にはどういうことでしょうか。マイコンが H を出力する、L を出力するときに、実際にマイコンの内部で何が起こっているのか知っていると、電子部品を安全に確実に取り扱うことが出来るようになります。

マイコンには多くの端子（＝ピン）があります。電源ピンや特殊なピンを除く信号の入出力を行うピンを一般的に「ポート」と呼び、このポートに電子部品を接続して使うことになります。

このポートから H や L を出力したり、または入力ポートにすることで、電子部品に期待する動作をさせたり、接続されているセンサーの値をマイコンに取り込んだりします。

Arduino の場合、マイコンから H や L を出力するための命令は簡単で、

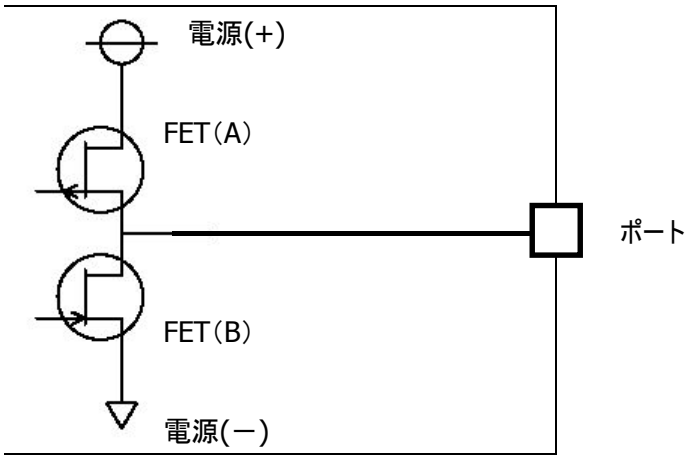
```
digitalWrite(13,HIGH);
```

と書けば、マイコンの 13 番ピンから H が出力され、

```
digitalWrite(13,LOW);
```

と書けば、マイコンの 13 番ピンから L が出力されます。

マイコンによって多少の違いはありますが、ポートは一般的には図のような仕組みになっています。



ポートには FET が 2 つつながっており、図のように

FET(A)は電源に、FET(B)は GND につながっています。

digitalWrite(ポートのピン番号,HIGH)　と書くと、

FET(A)が ON に、FET(B)が OFF になります。

FET(A)は電源につながっているのでポートには電源から電流が流れます。

つまりポートが電源と同じ電圧になるということであり、この状態が H を出力するということです。

また、**digitalWrite(ポートのピン番号,LOW)**　と書くと、

FET(A)が OFF に、FET(B)が ON になります。FET(B)は GND につながっているので、ポートから GND に電流が流れます。つまりポートが GND と同じ電圧になるということであり、この状態が L を出力するということです。

ここで大事なのは、「H または L を出力する」といっても「電源や GND と全く同じではない。」ということです。図で示したように、電源または GND とポートの間には FET があります。つまり、FET の流せる電流以上に電流を取り出すことはできず、それ以上流してしまうと FET が壊れてしまう、つまりマイコンが壊れてしまうということです。
この流して良い電流は、マイコンの最大定格電流という値で決められています。ここでは、マイコンには定格電流というものがあり、その範囲内で使わなければならないとだけ覚えておきましょう。

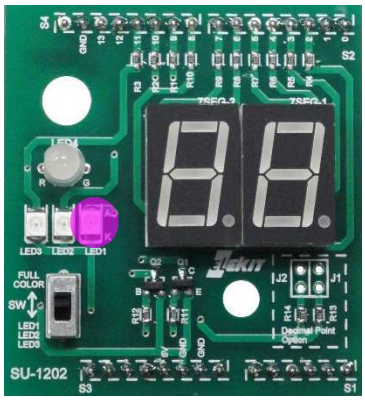
（2）　『点滅させる』

1 個の LED を点灯させることができたら、次は点滅させてみます。点滅も自分でプログラムを作って確認するときにはよく使う方法です。

●プログラム

次のプログラムを作成して、マイコンにアップロードします。

```
void setup(){
        pinMode(9, OUTPUT);
}
void loop(){
        digitalWrite(9,HIGH);
        delay(500);     <――――――――(1)
        digitalWrite(9,LOW);
        delay(500);     <――――――――(2)
}
```



プログラムをアップロードすると LED1 が 1 秒間隔で光ります。

●解説

`digitalWrite(9,HIGH)`で LED1 が点灯します、`digitalWrite(9,LOW)`と書くと 9 ピンが L になり、LED は点灯しません。つまり消灯状態になります。`delay(500)`は、500 ミリ秒待って次に進む命令です。例えば（1）の `delay` のときは 9 ピンから H を出力した状態のまま 500 ミリ秒待つということです。

全体としては、LED を点灯して 500 ミリ秒まち、次に LED を消灯して 500 ミリ秒まちを繰り返すことで 1 秒間隔で点滅することになります。

●使い方、ポイント

・よくあるミス

よくやってしまう失敗例に以下があります。

```
void loop(){
        digitalWrite(9,HIGH);
        digitalWrite(9,LOW);
}
```

一見上手く行きそうですが、実際に行ってみると LED は光りっぱなしに見えてしまいます。

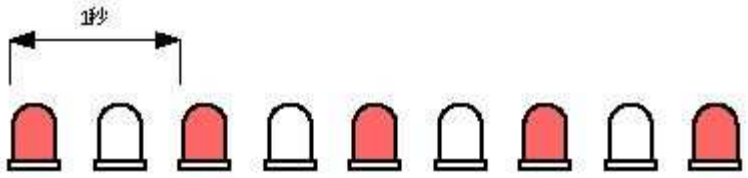
このプログラムでは、LED を光らせるとすぐに次の命令を実行し LED を消灯し、さらにすぐに LED を点灯・・・というように LED の ON、OFF をとても速いサイクルでくり返してしまい、結果として、点灯しっぱなしのように見えてしまいます。

・周期と周波数

1 秒間隔で点滅することを「点滅の周波数は 1Hz である。」といいます。この″Hz″（ヘルツ）は点滅させるときや、ON-OFF を繰り返すときによく使われる単位です。周波数を計算する式は、

f （Hz）＝1／T（秒）

という式です。「周波数」はマイコンではよく使われる言葉ですので覚えておくと良いでしょう。



例えば、図のように点灯から点灯までの時間（これを周期といいます）が 1 秒の場合は、式にあてはめると、周波数（ｆ）＝1 Hz となります。またもしも、周期が 0.01 秒だったとすると、100Hz ということになります。

（3）　『順番に点灯』

電飾系のガジェットを作成したときは、LED の光りを流れるように制御したいことが良くあります。

●プログラム

```
void setup(){
pinMode(9, OUTPUT);
      pinMode(10, OUTPUT);
      pinMode(11, OUTPUT);
}
void loop(){
      digitalWrite(9, HIGH);
      digitalWrite(10, LOW);
      digitalWrite(11, LOW);
      delay(300);
      digitalWrite(9, LOW);
      digitalWrite(10, HIGH);
      digitalWrite(11, LOW);
      delay(300);
      digitalWrite(9, LOW);
      digitalWrite(10, LOW);
      digitalWrite(11, HIGH);
      delay(300);
}
```

1　2　3

● ● ●

LED1 は点灯
LED2 は消灯
LED3 は消灯

● ● ●

LED1 は消灯
LED2 は点灯
LED3 は消灯

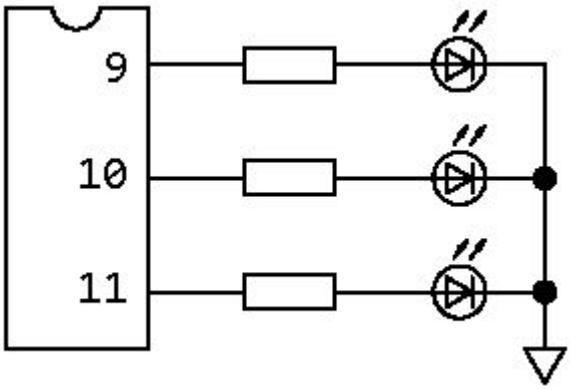
● ● ●

LED1 は消灯
LED2 は消灯
LED3 は点灯

●解説

基本的には光らせたい LED のピンを HIGH にします。LED を複数使う場合に大事なのは、点灯させる LED のことだけでなく、それ以外の LED の状態をどうしておくかということまで考えることです。

LED1、LED2、LED3 はマイコンの 9 ピン、10 ピン、11 ピンにつながっています。LED1～LED3 の光り方とプログラムを見比べて確認しましょう。



●使い方、ポイント

「点滅させる」でも説明したように、よく `delay` を入れるのを忘れてしまうので注意します。`delay` を入れないと 3 つの LED が全て点灯したように見えてしまいます。

（4）　『調光する』

LED の点灯はできましたが、もしも LED の光が明るすぎる場合はどうしたらよいでしょうか。

LED の制限抵抗の値を大きくすると流れる電流が少なくなり LED が暗くなりますが、毎回抵抗を差し替えたり、またはボリュームを回して値を変更するのは、あまりスマートではありません。スマートな方法としてプログラムで LED の明るさを調整する方法があります。

LED の明るさをプログラムで調節、つまり調光する方法として、LED をとても速い間隔で ON/OFF させる方法があります。速い間隔で LED を ON/OFF させると、人間の目には点滅に見えずに、明るさが変わっているように見えます。光っている時間が消えている時間に比べて長いと明るく、逆に光っている時間が短いと暗く見えます。光っている時間と消えている時間の割合を調節することで好みの明るさで光らせることができます。

ON と OFF を素早く切り替える方法には、大きく 2 の方法があります。その 2 つは

（1）点灯時間と消灯時間をプログラムにそのまま書いていく方法。

（2）マイコンの機能を利用して自動的に ON/OFF する信号を出力する方法。

の 2 つです。（1）の方法は「ソフト PWM」、（2）の方法は「ハード PWM」といわれています。

●プログラム 1 （ソフト PWM で調光する）

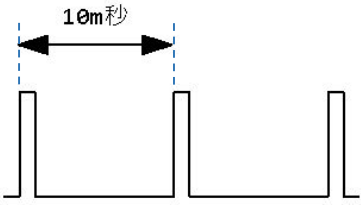
```
void setup(){
        pinMode(9, OUTPUT);
}
void loop(){
        digitalWrite(9, HIGH);
        delay(1);               ……(a)
        digitalWrite(9, LOW);
        delay(9);               ……(b)
}
```

●解説

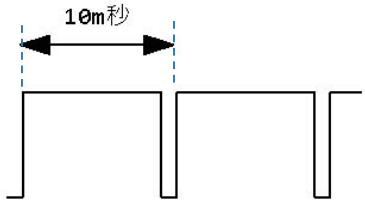
プログラムは前項目で説明した、「点滅させる」と同じですが、違う点は、**delay** の中の数字です。

このプログラムは、1 ミリ秒間 LED を点灯し、9 ミリ秒間 LED を消灯します。結果として 10 ミリ秒ごとに LED が点灯します。点灯時間を長くすると LED が明るく光り、点灯時間を短くすると LED が暗くなります。

図に示すと以下のようになります。



1 ミリ秒間”H”を出力し、
9 ミリ秒間”L”を出力しています。
“H”の期間つまり LED が点灯する時間が消えている時間に比べて短いので、暗く光ります。



9 ミリ秒間”H”を出力し、
1 ミリ秒間”L”を出力しています。
“H”の期間つまり LED が点灯する時間が消えている時間に比べて長いので、明るく光ります。

このように、ある一定の間隔の中で H と L の割合を変えることを、
パルス幅変調（Pulse Width Modulation、または PWM）
といいます。

●プログラム２（ハード PWM で調光する）

```
void setup(){
        analogWrite(9,50);
}
void loop(){
}
```

●解説

ハード PWM はマイコンの PWM 機能を使う方法です。Arduino にも PWM 機能があり、その機能を使うには、

`analogWrite(9, 50)`と書きます。

プログラムに H を出力する期間の値（ここでは **50** の部分）を書くだけで自動的に 9 ピンから H と L を切り替えながら出力し続けます。この H を出力する期間の値を変えることで明るさを調節することができます。

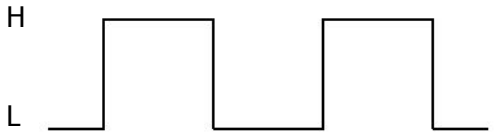
PWM の値の設定と実際の動作については次の使い方、ポイントで説明しています。

●使い方、ポイント

Arduino の PWM 機能を使うと以下の図のような出力になります。

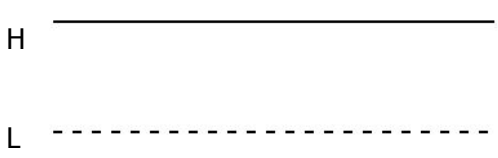
図の中の用語は PWM を使うときに良くでてくる用語です。マイコンを考える際に役立ちますのであわせて覚えておくと良いでしょう。

数値に 128 を指定すると、ちょうど半分ずつになります
このような状態を Duty 比が 50%といいます



数値に 64 を設定すると、H が 1/4 のこりが L になります。
このとき Duty 比が 25%といいます。

数値に 255 を設定すると、ずっと H になります。
このときの Duty 比は 100%です。



・Arduino の PWM 機能について

Arduino では PWM 機能が使えるピンは決まっており、それ以外のピンでは使うことができません。例えば Arduino-UNO（R3）の場合は、3,5,6,9,10,11 の 6 つのピンを PWM として使用できます。

さらにややこしいことに、そのピンによって PWM の周期（H から次の H までの時間）が違っていますので、PWM の周期が大事な回路をつくるような場合は、しっかりと確認しなければなりません。しかし単に LED を調光するのであれば特に問題はありません。

Arduino では、間隔は固定されており、基本的には変更できませんが、その他多くのマイコンの PWM 機能では間隔も設定できます。

●コラム 4

「ちらつき」

PWM の周期（H から次の H までの期間）をどんどん長くしていくと、だんだんと点滅しているのが見えるようになります。明るさを調節したいのに、点滅してしまってはいけませんね。このことを「ちらつき」といっています。

人間の目でちらつきを感じだす周波数は 50～60Hz といわれています。およそ 0.02 秒間隔で LED を ON/OFF させている場合は、点滅に見えてしまうということです。調光でちらつきを感じないようにするには、先ほどの周波数の倍程度あれば良いとされていますので、100Hz 以上の周波数、つまり、0.01 秒間隔より短い間隔で点滅させるようにしましょう。

2．フルカラーLED を制御する

フルカラーLED は赤・青・緑の LED が一つのパッケージに入った形になっています。光の 3 原色である赤・青・緑の光の重ね合わせの原理を使うことで、いろんな色で光らせることができます。

●プログラム 1

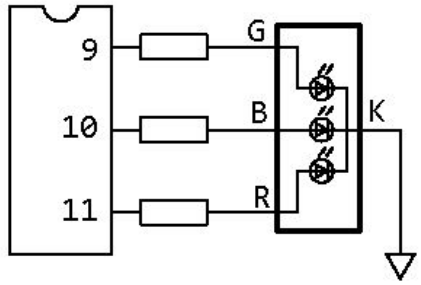
フルカラーLED が、赤→黄色→白→消灯と変化するプログラム

```
void setup(){
        pinMode(9,OTPUT);
        pinMode(10,OTPUT);
        pinMode(11,OTPUT);
}
void loop(){
        digitalWrite(11,HIGH);・・・・・・//(a)○●● →赤
        delay(1000);
        digitalWrite(9,HIGH); ・・・・・・//(b)○○● →赤＋緑＝黄色
        delay(1000);
        digitalWrite(10,HIGH); ・・・・・・//(c)○○○→赤＋緑＋青＝白色
        delay(1000);
        digitalWrite(9,LOW);
        digitalWrite(10,LOW); ・・・・・・・//(d)●●●→消灯
        digitalWrite(11,LOW);
}
```

●解説

表示シールドのフルカラーを使うためには、スイッチを”FULLCOLOR”の方にしてください。

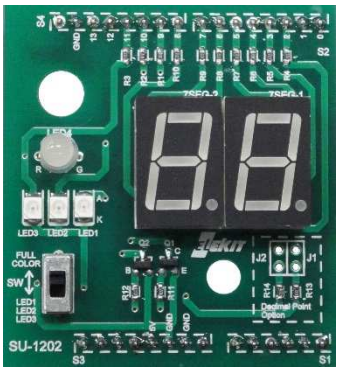
LED は下の回路のように 9,10,11 ピンとつながっています。



digitalWrite(11,HIGH)と書くと、実験基板のフルカラーLED の赤色が点灯します。同様に、

digitalWrite(10,HIGH)と書くと、実験基板のフルカラーLED の青色が点灯します。また、**digitalWrite(9,HIGH)**と書くと、実験基板のフルカラーLED の緑色が点灯します。

(a)で、11 ピンを HIGH にするとフルカラーLED が赤色に光ります。次に(b)で 9 ピンが HIGH になりフルカラーLED の緑色が点灯します。ここでは赤色と緑色の LED が点灯しているので、フルカラーLED は黄色に光って見えます。次に(c)で 10 を HIGH にするとフルカラーLED の青が点灯します。ここでは赤、緑、青の LED が 3 つとも光るのでフルカラーLED は白色に近い色で光って見えます。



●プログラム 2

フルカラーLED の色をだんだんと変化させるプログラム

```
int valRed=0;
int updown=1;

void setup(){
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
}
void loop(){
  analogWrite(9,valRed);
  analogWrite(10,255);
  analogWrite(11,255);
  delay(5);

  if(valRed >=255){
    updown = -1;
  }else if(valRed <=0){
    updown = 1;
  }
  valRed = valRed + updown;
}
```

●解説

フルカラーLED の色をだんだん変化させます。この例では、赤色だけを変化させています。

analogWrite(PIN,PWM の値**)**で指定できる PWM の値は 0～255 までですので、0 から 1 ずつ加えて赤色の LED が光る時間を長くしていき、255 になると、今度は 1 ずつ減らして、光る時間を短くしていきます。

同じ調整方法を青色 LED と緑色 LED にも使ったプログラムがプログラム 3 です。ちょっとプログラムの量は多くなりますが、是非実際にプログラムして実験してみてください。

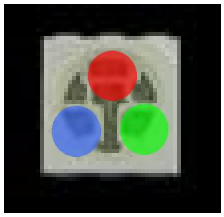
```
int valRed  =255;
int valBLUE =0;
int valGREEN=128;
int updown=1;
int updownBLUE=1;
int updownGREEN=1;

void setup(){
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
}
void loop(){
  analogWrite(9, valRed);
  analogWrite(10,valBLUE);
  analogWrite(11,valGREEN);
  delay(5);
//RED LED
  if(valRed >=255){
    updown = -1;
  }else if(valRed <=0){
    updown = 1;
  }
  valRed = valRed + updown;
//BLUE LED
  if(valBLUE >=255){
    updownBLUE = -1;
  }else if(valBLUE <=0){
    updownBLUE = 1;
  }
  valBLUE = valBLUE + updownBLUE;
//GREEN LED
  if(valGREEN >=255){
    updownGREEN = -1;
  }else if(valGREEN <=0){
    updownGREEN = 1;
  }
  valGREEN = valGREEN + updownGREEN;
}
```

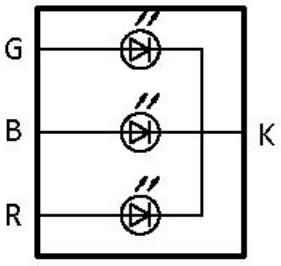
●使い方、ポイント

・フルカラーLED のしくみ

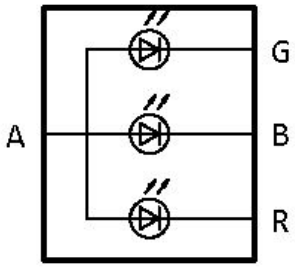
写真のように赤・青・緑の LED が一つのパッケージに入った形になっています。ただし、その仕組み上、近くで見るとそれぞれの色が光の点として見えてしまい、綺麗な一つの色で光らないことがあります。また、見る角度によって違う色に見えてしまうことがあります。



アノード同士が共通になったものや、逆にカソード同士が共通になったもの、それぞれ単独になっているものなどがあります。アノードが共通になっているものはアノードコモン、カソードが共通になっているものはカソードコモンと呼ばれています。



カソードコモン



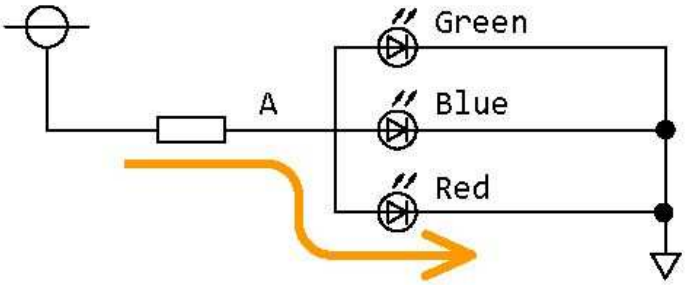
アノードコモン

・フルカラーLED の制限抵抗

3 つの LED がまとめられているだけですので、それぞれの LED に対して制限抵抗を付けることになります。制限抵抗の考え方は 1 つの LED を使うときと同じです。

「抵抗は 1 個でいいのではないか？」と考えたくなりますが、フルカラーLED の場合は各色の LED それぞれに制限抵抗を付ける必要があります。それには理由があります。

例えば図は、制限抵抗を一つだけつけたときの回路です。



この回路に電池をつなぐと、それぞれの LED に電流が流れ始めます。

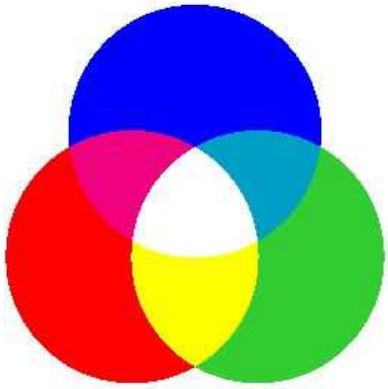
まず、LED の光り始める電圧が低い赤色 LED が点灯します。その時、赤色 LED の両端の電圧は、順方向電圧である約 2V になります。つまり、赤色の LED が点灯するということは、(A)の電圧はいつも 2V というこ

とになってしまい、青色や緑色の LED が光り始める電圧の 3.5V にはならないこととなり、いつまでたっても青と緑の LED は光り出さないことになります。

・光の 3 原色

やみくもに LED を光らせて好みの色を探すのは大変ですので、この色とこの色を混ぜるとどんな色になるのがある程度の目安を知っておくために、光の 3 原色について知っておくとよいでしょう。
次の図は光の 3 原色と重ね合わせたときの色の図です。例えば赤＋緑は黄色になるということが分かります。

各色の明るさは、電流により変わります。
例えば、3 色を光らせて綺麗な白色を出すには、
それぞれの色を電流で調節、または PWM で
調光して、合わせるのが一般的です。



3. 7セグ LED を制御する

電子工作で時計やカウンターなど、数字を表示したいことがあります。基板に数字の形に LED を取り付けて表示器を作る方法もありますが、配線が結構面倒です。単に数字を表示させたいだけなら、あらかじめ数字の形に作ってある「7 セグ LED」という部品を使うと便利です。

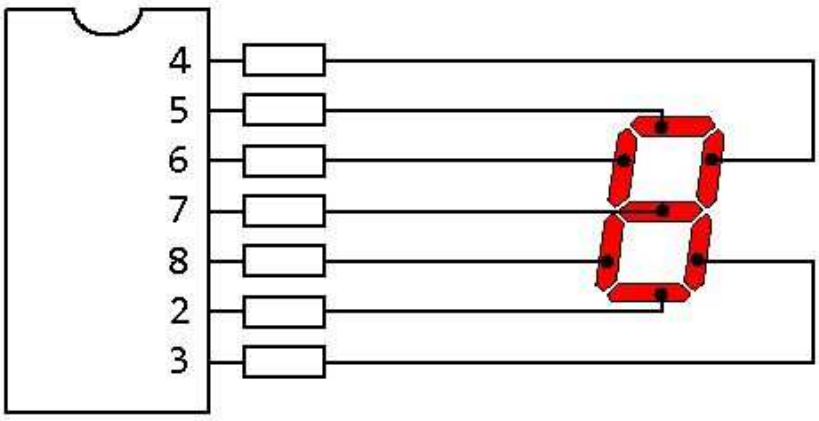
●プログラム 1

次のプログラムは、7 セグ LED に″1″を表示します。

```
void setup(){
        pinMode(2,OUTPUT);      pinMode(3,OUTPUT);
        pinMode(4,OUTPUT);      pinMode(5,OUTPUT);
        pinMode(6,OUTPUT);      pinMode(7,OUTPUT);
        pinMode(8,OUTPUT);
}
void loop(){
        digitalWrite(4,HIGH);
        digitalWrite(3,HIGH);
}
```

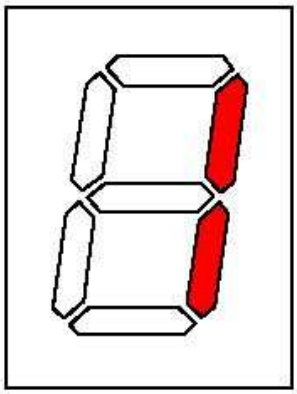
●解説

実験基板上の 7 セグ LED とマイコンのピンは、以下の接続図のようになっています。



7 セグ LED の光らせたい LED に接続されているピンを H にするだけで OK です。

例えば「1」を表示させたければ、右側の上下の LED を光らせます。



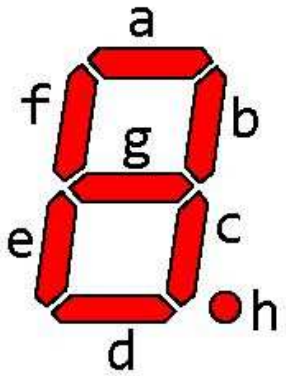
●使い方、ポイント

・7 セグ LED のしくみ

7 セグ LED は、7 つの LED が数字を表示しやすいようにあらかじめ配置されて 1 つのパッケージに収められている部品です。1 つ 1 つの場所のことをセグ（＝セグメント）と呼びます。数字を表示するには 7 つのセグがあればよいのですが、小数点を表示するとき使えるように、もう一つセグが付いているものもあります。この場合は全部で 8 セグになるのですが、普通はひとまとめに 7 セグ LED といわれます。

プログラムするときや、説明するときに、「1 を表示するときの上の部分のセグ。」というのでは不便なので、各セグメントは場所を示す記号で表されることが多く、一般的には、図のように、a,b,c,d,e,f,g(,h)のようになっています。h については DP（デシマルポイント）といわれることもあります。

7 セグ LED は 7 つの LED がまとめられているだけですので、それぞれの LED に対して制限抵抗を付けることになります。制限抵抗の考え方は 1 つの LED を使うときと同じです。



●プログラム 2

プログラム 1 では 1 つの 7 セグ LED に 7 本のマイコンのピンを使って数字を表示しましたが、2 桁や 4 桁で数字を表示する場合はどうしたらよいのでしょうか。

単純に増やすとしたら、7 セグ LED を 1 個に 7 つのピンを使うので、2 桁だと 14 ピン、4 桁だと 28 ピン必要になります。

しかし、Arduino で使えるピンは最大でも 18 ピンなので、この方法は使えません。このような場合は、ダイナミック点灯という方法を使います。プログラム 2 はダイナミック点灯で 2 桁の表示をさせる例です。

```
#define segA 5
#define segB 4
#define segC 3
#define segD 2
#define segE 8
#define segF 6
#define segG 7
#define segLeft 0
#define segRight 1
void setup(){
        pinMode(2, OUTPUT);    pinMode(3, OUTPUT);
        pinMode(4, OUTPUT);    pinMode(5, OUTPUT);
        pinMode(6, OUTPUT);    pinMode(7, OUTPUT);
        pinMode(8, OUTPUT);
        pinMode(segLeft,OUTPUT);
        pinMode(segRight,OUTPUT);
}
void loop(){
        digitalWrite(segLeft,HIGH);
        digitalWrite(segRight,LOW);

        digitalWrite(SEGA,HIGH);
        digitalWrite(SEGB,HIGH);
        digitalWrite(SEGC,HIGH);
        digitalWrite(SEGD,HIGH);
        digitalWrite(SEGE,LOW);
        digitalWrite(SEGF,LOW);
```

```
        digitalWrite(segG,HIGH);
        delay(10);

        digitalWrite(segLeft,LOW);
        digitalWrite(segRight,HIGH);

        digitalWrite(segA,HIGH);
        digitalWrite(segB,LOW);
        digitalWrite(segC,HIGH);
        digitalWrite(segD,HIGH);
        digitalWrite(segE,LOW);
        digitalWrite(segF,HIGH);
        digitalWrite(segG,HIGH);
        delay(10);
}
```

●解説

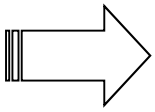
7 セグ LED に数字を表示する考え方はプログラム 1 と同じですが、ポイントはトランジスタを切り替えてダイナミック点灯で動かしているところです。ダイナミック点灯については、次の使い方、ポイントで説明しています。

●コラム 5

#define 文

数字を表示するには、多くのピンを HIGH にしたり LOW にしたりしないといけません。プログラムでピンの番号を書いていると間違いが起こりやすくなります。

```
digitalWrite(3, HIGH)
digitalWrite(2, LOW)
digitalWrite(8, LOW)
digitalWrite(6, LOW)
digitalWrite(0, LOW)
```



```
digitalWrite(segC, HIGH)
digitalWrite(segD, LOW)
digitalWrite(segE, LOW)
digitalWrite(segF, LOW)
digitalWrite(segLeft, LOW)
```

プログラムを見やすくするために、#define を使います。

例えば#define　segC 3 と書くとそれ以降、segC という文字が出てくるとプログラムが自動的に″3″という数字に置きかえて処理してくれます。

●使い方、ポイント

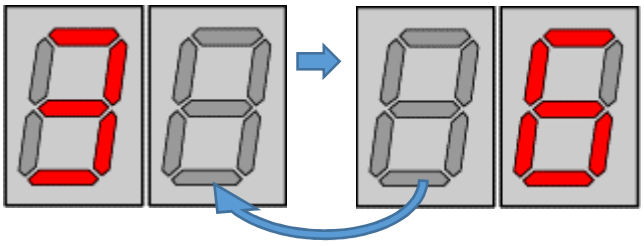
・ダイナミック点灯

LED1 個を点灯させるためには 1 本の信号、3 個点灯させるためには 3 本の信号・・・5 個点灯させるためには 5 本の信号が必要になります。このように 1 つの信号で 1 個の LED の点灯消灯を制御する方式のことをスタティック点灯といいます。7 セグ LED を 2 つ使った場合は 14 個の LED を制御する信号が必要になります。ところが実験基板では、9 本の信号で 14 個の LED を制御しています。その方法がダイナミック点灯という方法です。

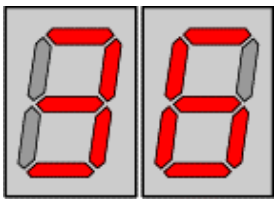
ダイナミック点灯とは、人間の目の残像現象を利用しています。人間の目は 1 度光をみるとその光が消えたあと 0.1 秒くらいはまだ点灯しているように感じています（これを残像現象といいます）。つまり LED が消灯したあと、だんだん暗くなり、0.1 秒後に完全に消えてしまうように見えるということです。この現象を利用して、LED を高速に点滅つまり 0.1 秒点灯して、0.1 秒消灯するという動作を繰り返すと、人間の目には LED が消灯せずに、連続で点灯しているように見えます。ただし、0.1 秒ではまだ、明るくなったり、暗くなったりするのを感じてしまい、光がチラついて見えます。そこで、コラムの「ちらつき」でも説明したように、その点滅の間隔をもっと早く、およそ 0.01 秒間隔より早くすると、人間の目はチラつきを感じなくなります。

実験基板には 2 つの 7 セグ LED があり、ちらつきなく見えるようにするには、0.01 秒間隔で点滅させるので、1 つ目の 7 セグ LED を 0.005 秒、2 つ目の 7 セグ LED を 0.005 秒交互に表示することを繰り返せば、全体として 0.01 秒間隔で光るので、ちらつきを感じずに、2 桁の表示がいつも点灯しているよう見えます。

7 セグ LED の動きをスロー再生すると・・・



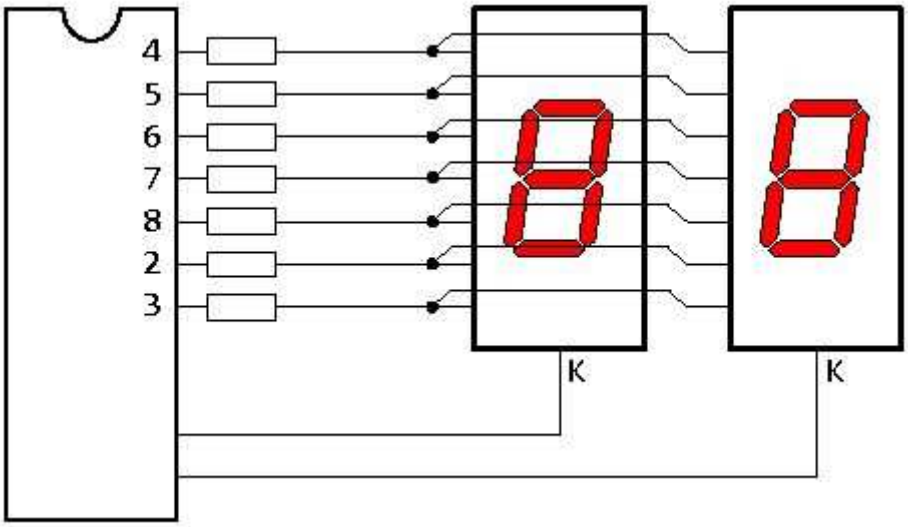
この動作を高速にすると、残像現象により「36」の数字に見えます。



実際に制御する方法を考えてみます。

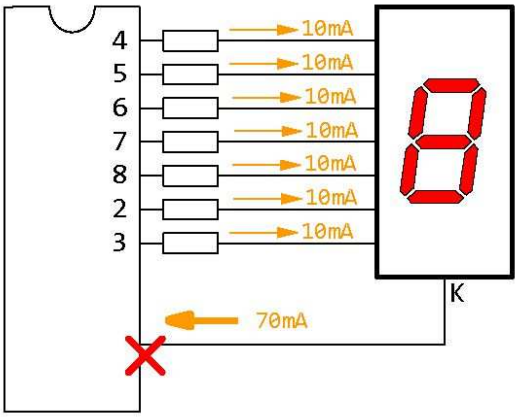
片方の 7 セグ LED が点灯しているときは、もう片方の 7 セグ LED の表示を消しておく必要があります。そのための方法として、7 セグ LED のカソードをー（マイナス）につなぐのでなく、マイコンのポートに接

続する方法があります。こうすることで、ポートが L のときは、電流が流れることができ LED が光りますが、ポートが H のときは、電流が流れず LED が光らないことになります。
このカソードにつないだポートを素早く切り替えることで、ダイナミック点灯を実現できます。



しかし、この方法は注意しなければいけない点があります。それは「ポートに流れる電流」です。
例えば 7 セグ LED の 1 つの LED に 10mA の電流を流すとします。すると、7 セグを全部点灯したときは 70mA の電流が必要になります。

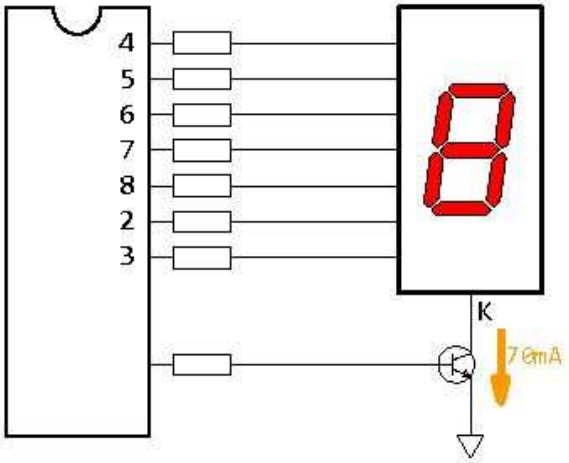
ここで、マイコン（ここでは Arduino マイコン）の 1 つのピンに流せる電流は最大定格で 40mA となっています。
つまり図のような回路の場合、ピンに 70mA の電流が流れてしまい、マイコンを壊してしまいます。



そこで、電流がマイコンのピンに直接流れないように、電流ブースト回路をつかう必要があります。電流ブースト回路は次で説明しています。

・電流ブースト回路

下図のように、7 セグ LED の共通カソード部分にトランジスタを接続します。そのトランジスタの ON/OFF はマイコンのポートで行います。ポートを H にするとトランジスタが ON になり電流は流れることができますので、7 セグ LED のアノードの状態に応じて LED が点灯します。ポートを L にすると、トランジスタが OFF し電流は流れなくなるため、アノードの状態が H/L どちらの状態であっても LED は点灯しません。



電流は、トランジスタを流れてー（マイナス）に流れていきますので、マイコンのポートに負担は掛からなくなります。

4．LCD を制御する

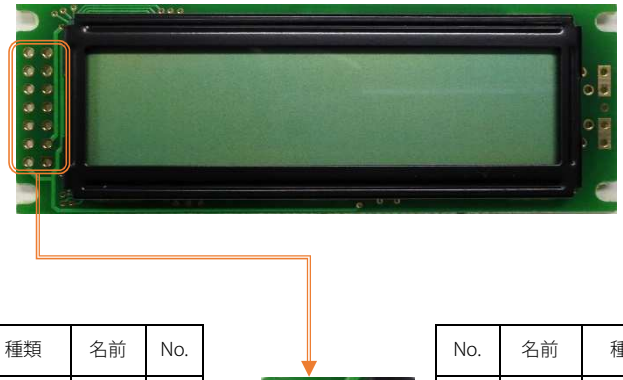
この学習セットには LCD モジュールはセットされていませんが、LCD モジュールの使い方を簡単ですが説明しています。せっかくマイコンを使うのですから、数字や文字を表示させて、値を確認したり、メッセージを表示する機器を作りたくなります。そんなときには LCD モジュールが入手もしやすく、手頃で便利なのですが、表示プログラムの全てを自分で作ろうとすると大変です。しかし、Arduino には LCD モジュールに文字を表示させるための「ライブラリ」が用意されており、このライブラリを使うと簡単に表示させることができるようになります。
今すぐに使う予定がなくても知識として知っておくとよいでしょう。

●LCD モジュールについて

LCD モジュールには色んな種類のものがあります。ここでは、「日立 HD44780 コンパチブル」といわれるモジュールを使っています。コンパチブルであっても、種類により端子の配置が違っていますので、次に説明する配線を参考に、入手した LCD モジュールに合った配線を行ってください。
なお、HD44780 コンパチブル(互換)と書かれていないモジュールの場合は、ここで説明するプログラムでは動かない可能性があります。

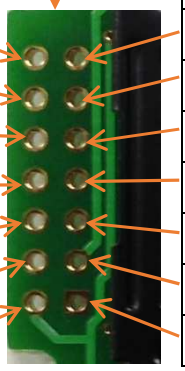
・本書で使用している LCD モジュール

SC1602BS(-XA-GB-K)　　16 文字×2 行バックライト無しタイプ



端子の説明

説明	種類	名前	No.
データビット 7	入出力	DB7	14
データビット 5	入出力	DB5	12
データビット 3	入出力	DB3	10
データビット 1	入出力	DB1	8
イネーブル	入力	E	6
レジスタ選択	入力	RS	4
電源グラウンド	電源	Vss	2



No.	名前	種類	説明
13	DB6	入出力	データビット 6
11	DB4	入出力	データビット 4
9	DB2	入出力	データビット 2
7	DB0	入出力	データビット 0
5	R/W	入力	リードライト
3	Vo	入力	コントラスト
1	Vdd	電源	電源＋5 V

・Arduino への配線

LCD モジュールを使うためには、LCD モジュールと Arduino 基板を接続しなければなりません。LCD モジュールの機能により配線の方法はいくつかあるのですが、ここでは一番シンプルな配線方法を使います。この方法では LCD モジュールの端子と Arduino ピンは、以下の配線表のように接続しています。

配線のポイント 1

配線には少々面倒に感じても、ピンソケットを使うことをお勧めします。LCD モジュールに直接コードをはんだ付けしてしまうと、将来別のものに LCD を使おうと思ったときに、はんだ付けの修正が発生して思わぬトラブルになります。LCD モジュールの端子に合うピンソケットを用意しましょう。

配線のポイント 2

Arduino 基板にはあらかじめピンソケットが取り付けられていますので、ソケットにそのまま差し込むことができるジャンパーコードを使用します。何度でも配線を変えることができ、テストに便利です。色も一色だけでなく、何色か用意しておくと配線ミス防止に役立ちます。

ジャンパーコードを 10 本用意しましょう。10cm 以上の長さのものが使いやすいと思います。

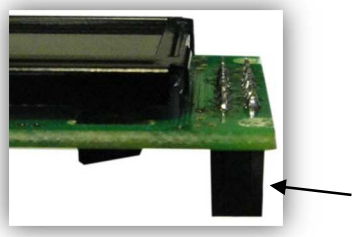
配線のポイント 3

本来は、LCD モジュールには液晶のコントラスト調整用端子に半固定抵抗を取り付けることが必要です。この端子に半固定抵抗を付けて適切な電圧を与えないと、液晶が真っ黒か真っ白にしか表示されず、正常に動作しているのか分からなくなります。

しかし今回は、LCD の動作実験が目的なので面倒で時間の掛かる配線やはんだ付けを極力減らすために、この端子を Arduino の PWM 出力端子に接続して、擬似的なアナログ電圧を加えます。

配線の手順 1

LCD モジュールにピンソケットをはんだ付けします。

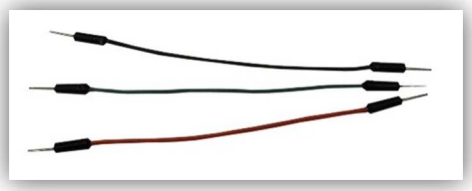


写真の向きに取り付けて
はんだ付けします。

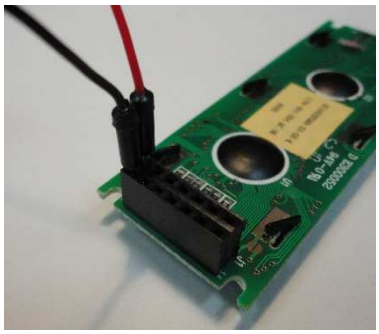
配線の手順 2

LCD モジュールのピンソケットに配線します。

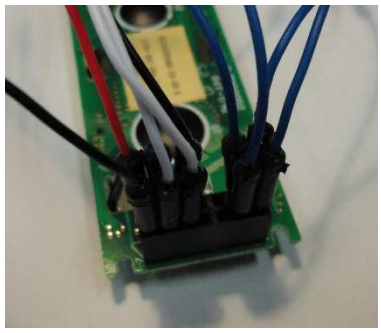
今回は、電源に赤と黒、制御信号に白、コントラスト調整に緑、データ信号に青という具合に 5 色のジャンパーコードを使いました。



ジャンパーコード



電源端子に配線したところ



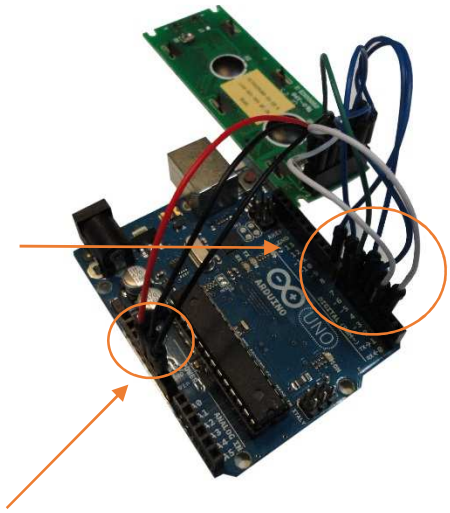
必要な端子全てに配線したところ

配線の手順 4

Arduino 基板に差し込むピンソケットに配線します。

特に電源の＋(Vcc)とー（GND）は間違えないように十分注意します。

LCD 端子	Arduino 端子	色
DB7	Digital 5	青
DB6	Digital 8	青
DB5	Digital 4	青
DB4	Digital 7	青
E	Digital 3	白
RS	Digital 2	白
Vo(コントラスト)	Digital 6	緑
R/W	GND	白
Vcc(5V)	5V	赤
Vss(GND)	GND	黒



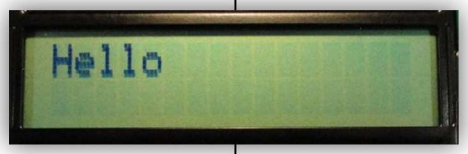
※コードの色は参考例です。

・LCD に表示させる

配線が無事完了したら、動作チェックを兼ねて LCD に文字を表示させてみます。

●プログラム 1

LCD 表示に　Hello　を表示させるプログラムです。




```
#include <LiquidCrystal.h>
LiquidCrystal  lcd(2,3,7,4,8,5);

void setup(){
  lcd.begin(16,2);          //LCD モジュールのサイズを宣言
  analogWrite(6,80);        //Vo にコントラスト用の電圧を加える処理
}

void loop(){
  lcd.setCursor(0,0);       //文字を書き始める場所の指定
  lcd.print("Hello");       //Hello を表示させる
}
```

● 解説

プログラムの最初に、「このプログラムでは LCD 用のライブラリを使いますよ。」という「宣言」を書かなければいけません。その宣言が、`#include < LiquidCrystal.h >` です。

続いて、LCD モジュールの配線が Arduino のどのピンに配線されているかをプログラムします。
そのための命令が、`LiquidCrystal  lcd（2,3,7,4,8,5）`です。
この `lcd` につづくカッコ（）の中に書く番号が Arduino のピンの番号で、書く順番も重要な意味があります。これについては使い方、ポイントで説明しています。

次に、LCD モジュールの画面のサイズを書きます。ここで使っている LCD 画面は 16 桁で 2 行のものなので、`lcd.begin（16,2）`と書きます。これは `setup` の中に書くのがよいでしょう。
次の `analogWrite` 命令で 6 番ピンに Duty30%程度の PWM を出力します。6 番ピンは LCD モジュールのコントラスト調整端子に接続されており、擬似的なアナログ電圧を入力して、コントラストを調整しています。（※正しくコントラストを調整する場合は、この端子に半固定抵抗を接続しますが、今回は実験用の方法として、配線を簡単にすることを優先しています。）

宣言をおこない、LCD モジュールと Arduino がどのように配線接続されているかも書き、使っている LCD の画面サイズも書きましたので、あとは LCD 画面に文字や数字を表示させるだけです。LCD 画面に表示す

るときに考える必要があるのは、主に 2 つで、どこに表示させるのか、何を表示させるのかです。

まず、どこに表示させるのかを決めるための命令が、`lcd.setCursor(0,0)`です。`0,0` と書くと、`0` 桁目の `0` 行目つまり、LCD 画面の左上から書き始めますという命令になります。
次に、何を表示させるのかを決めるための命令が `lcd.print("Hello")`です。この命令で、LCD 画面には Hello が表示されます。

●使い方、ポイント

・`LiquidCrystal lcd（RS,ENABLE,D4,D5,D6,D7）`について
LCD モジュールの端子に Arduino の何番ピンを接続したかを書くための命令です。カッコの中は、（`RS,ENABLE,D4,D5,D6,D7`）の順番で書かなければいけない決まりです。例えば、今回の例では LCD モジュールの RS 端子は、Arduino の D2 ピンに接続、LCD モジュールの ENABLE 端子は Arduino の D3 ピンに、LCD モジュールの D4～D7 端子は、それぞれ Arduino の D4～D7 に接続しましたので、`lcd（2,3,7,4,8,5）`と書きます。
もしも、配線を変えた場合は、この命令のカッコの中で、変更した配線の通りにピン番号を変えればよいので大変便利に使えます。

●プログラム 2

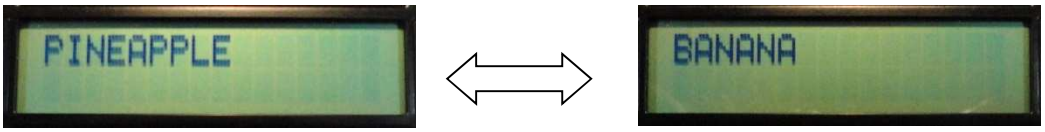
2 つの言葉を、1 秒ごとに表示するプログラム

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(3,2,7,4,8,5);


void setup(){
  lcd.begin(16,2);
  analogWrite(6,80);
}
void loop(){
  lcd.setCursor(0,0);
  lcd.print("PINEAPPLE");
  delay(1000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("BANANA");
  delay(1000);
}
```

●解説

LCD 画面に、PINEAPPLE と、BANANA という文字を 1 秒ごとに交互に表示します。



●使い方、ポイント

このプログラムのポイントは 2 つです。一つは `delay` を使っていること、もう一つは、`lcd.clear()`を使っていることです。

`delay` を使わないと、とても高速に交互に文字が表示されます。結果として、何を表示しているのかほとんど分からなくなります。(写真)　人間の目に分かる程度の時間しっかり表示させるために `delay` を使って PINEAPPLE を 1 秒間表示して、その後、BANANA を 1 秒間表示させています。

次ぎに `lcd.clear()`の役割です。これは LCD 画面の表示を消してしまうという命令です。もしもこの命令を入れないと、写真のように、PINEAPPLE と、BANANA が重なったような変な文字になります。

これは PINEAPPLE という 9 文字の前半の 6 文字は BANANA に上書きされますが、後半の 3 文字の表示が残ったままになってしまうためです。

用途、表示に応じて `lcd.clear()`を入れるようにしましょう。

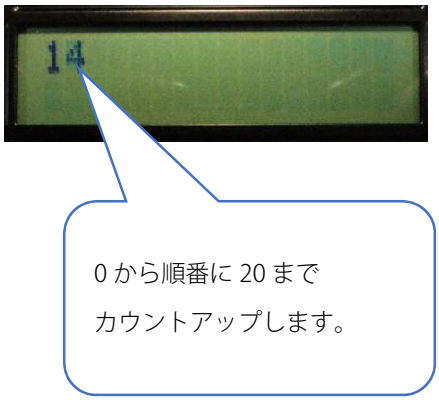
●プログラム 3

決められた文字を表示するだけでなく、刻々と変化していく数値を表示させる例。

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(3,2,7,4,8,5);
int val=0;
void setup(){
  lcd.begin(16,2);
  analogWrite(6,80);
}
void loop(){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(val);
  delay(500);
  ++val;
  if(val >= 21) val = 0;
}
```

●解説

上記のプログラムでは変数 `val` が `0` から `20` まで変化します。`lcd.print(val)`と書くと LCD には、`val` の値つまり、`0` から順番に `20` までの数字が表示されます。



●使い方、ポイント

ライブラリを使うと、簡単に LCD に表示を行えますが、実際にいろいろな表示をさせるときには、「表示が横にスクロールしていくといいなあ。」とか、「オリジナルの文字を表示させたいなあ。」などの希望がでてきます。そんなときのために、LCD のライブラリには、多くの命令が用意してあります。

LCD モジュールを使って高度なことがしたくなったら、Arduino のサイトを訪ねて、LCD ライブラリの情報を見てみるとよいでしょう。LCD の表示についての多くの命令や、サンプルなどが掲載してあります。

http://arduino.cc/en/Reference/Libraries

●コラム６

「ライブラリ」とは

プログラムを書くときの、決まった手続きや、似たような手続きをまとめて専用の命令としていつでも使えるようにしたものを「ライブラリ」と呼んでいます。

1 回だけ動かすのであれば長いプログラムになっても特に気になりませんが、LCD に文字をいろいろ変えて表示させたいときは、何度も何度も同じような命令を書くことになります。ここでその「決まった手続き」を一つの命令にまとめたものがあれば、プログラムの作成が便利になり、間違いも少なくなります。

Arduino には多くの関数があらかじめ用意されています。

新しく使いたい部品やモジュールを動かすためのライブラリが用意されていないかチェックしてみると良いでしょう。

Arduino 基板と Arduino-IDE

（1）　Arduino 基板

●いろいろな Arduino

市販されている Arduino には多くの種類があります。Arduino の最も標準的な Arduino-UNO に加えて、ブレッドボード上でのテストに便利な Nano、衣服に縫い付けることを考えられた LilyPad、PC との通信部分を取り外して安価にした Pro、入出力が強化された Mega などがあります。

これら全ての Arduino は、プログラムを作る仕組みが同じで、同じ Arduino-IDE といわれるプログラム開発ソフト（総合開発環境といいます）で作成することができます。

プログラムの書き換えに特別な機器は必要としないので、プログラムの書き込みが手軽ですぐに使い始められます。

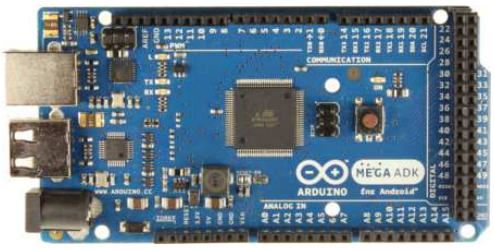
Arduino の例

Arduino-UNO

Arduino-Nano

LilyPad

Arduino-Pro



Arduino-Mega

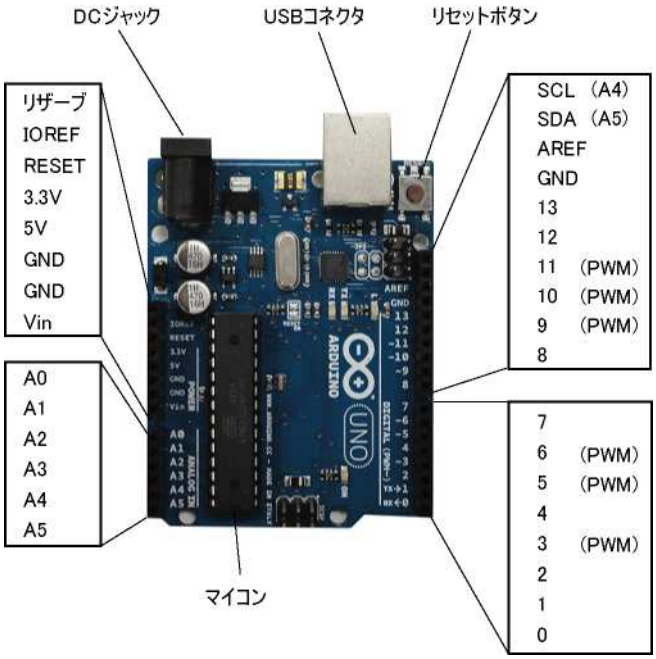
本書で説明に使用しているシールドは、**Arduino-UNO(R3)**と接続します。

●機能説明

【ピン配置図】

Arduino-UNO(R3)
の場合

※ Arduino-UNO(R3) 以外ではピン数、機能が異なる場合があります。



【ピンの機能】

機能	説明
リザーブ	将来の拡張用に予約されているピンで使うことはできません。
IOREF	ArdunoUNOではVccに接続されています。
RESET	リセットボタンに接続されています。リセットボタンの追加に使用できます。
3.3V	3.3Vの電源になります。
5V	5Vの電源になります。シールド基板の電源はここから供給しています。
GND	グラウンドです。シールド基板の電源のマイナスはここにつながっています。
GND	
Vin	DCジャックから入力された電源につながっています。 （内部でダイオードを経由しますので、電圧が少し低くなります。）

機能1	機能2	説明
A0		アナログ入力ポートです。 センサーなどの接続に向いています。 デジタル入出力ピンとして使うこともできます。
A1		
A2		
A3		
A4	SCL	I2C(Wire)通信を使うときには、SCL/SDAとしてここを使います。
A5	SDA	

機能1	機能2	説明
SCL	(A4)	A4,A5はここにもつながっています。
SDA	(A5)	
AREF		AD変換器の基準電圧のためのピンです。使用するためには特別な命令や配線が必要です。
GND		グラウンドです。
13		13～0はデジタル入出力ピンとして使用します。 11,10,9,6,5,3はPWM制御できるピンとして使用することもできます。
12		
11	PWM	
10	PWM	
9	PWM	
8		
7		
6	PWM	
5	PWM	
4		
3	PWM	
2		
1	TX	1,0はUSB経由でPCと通信するときに使用します。
0	RX	

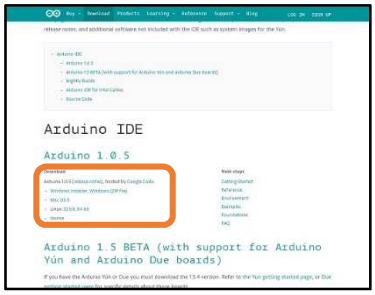
（2）　Arduino-IDE の準備

Arduino を制御するためには、プログラムを作成し、Arduino 基板へ書き込みを行う必要があります。このプログラムを作成するエディタとプログラム書き込み機能を持った開発環境のことを、Arduino-IDE といいます。

Arduino-IDE の入手と起動

1. まず Arduino のホームページにアクセスします。
   http://www.arduino.cc/ (英語のサイトです)

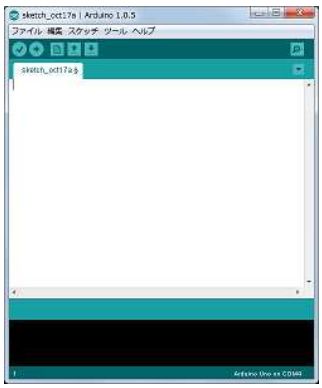
2. ダウンロード画面から、お使いの OS に対応したソフトを選択し、適当なフォルダーに保存します。



画面は変わる可能性があります
画面は 2013 年 10 月現在のものです。

3. ダウンロード後、保存したフォルダー内に、arduino-X.X.X-XXX が作成されます。このファイルは圧縮されているので解凍(展開)ソフトなどで解凍してくだい。
   ※XXX は使用する OS やソフトのバージョンによって異なります。
4. ファイルを解凍すると、Arduino-1.0.1 のように、バージョン番号が付いたフォルダーが作成されます。
5. 先ほど解凍したフォルダー内にある、arduino のアイコンをダブルクリックすると、arduino-IDE が起動し画面が表示されます。

↓フォルダーの中の arduino をクリック　　IDE 起動画面



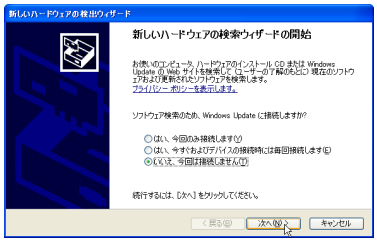
ドライバーのインストールとシリアルポート(COM ポート)の確認

Windows の場合

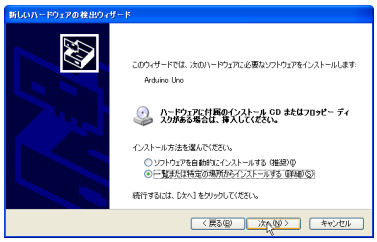
次に、Arduino 基板とパソコンが通信するために必要なドライバーをインストールします。

1. Arduino 基板とパソコンを USB ケーブルで接続します。
2. WindowsXP の場合：

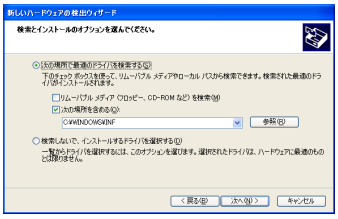
   接続後に「新しいハードウェアの検出」画面が表示されますので、「いいえ。今回は接続しません。」を選択して次へ進みます。

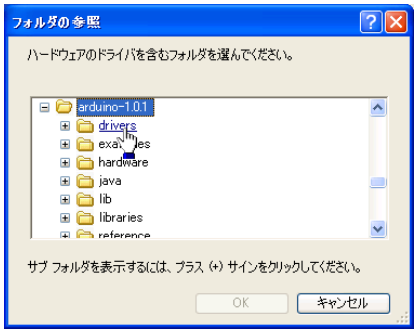


「一覧または特定の場所からインストールする。」を選択して次へ進みます。



「次の場所を含める。」で、参照のボタンをクリックして、先ほど解凍したフォルダーの中にある「Drivers」を選択して、次へ進みます。





ドライバーのインストールが始まり、自動的に完了します。

完了後に、もう一度ドライバーのインストールを求められる場合は、1 回目と同じように進めて完了させます。

WindowsVista/7 の場合：

「デバイスドライバーソフトウェアをインストールしています。」と表示され、しばらく待つと、パソコンが自動でドライバーを検索しインストールが完了します。もしも自動でインストールができない場合は手動でインストールする必要があります。

＊トラブルシューティング「ドライバーを手動でインストールする。」をお読みください。

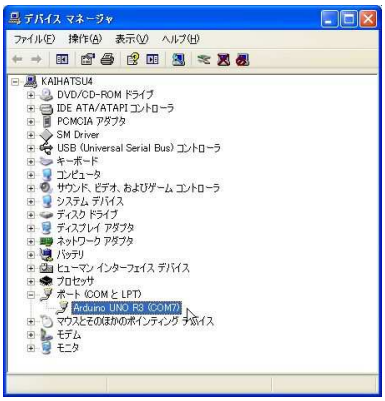
3. 次に Arduino 基板が、どのシリアルポート（COM ポート）に接続されたか確認します。

   WindowsXP では、マイコンピュータのアイコンを右クリックして「プロパティ」を選択し、「ハードウェア」のタブの中にある「デバイスマネージャー」をクリックします。

   WindowsVista/7 では、「コントロールパネル」→「システムとセキュリティ」→「デバイスマネージ

ャー」と進みます。

デバイスの一覧が表示されますので、「ポート（COM と LPT)」という項目の下を確認します。



Arduino 基板が接続されたポートが表示されます。

図では Arduino 基板が「COM7」に接続されていることが分かります。この COM の番号を覚えておきます。

4. Arduino-IDE 画面の「ツール」メニューから、「マイコンボード」を選択し、パソコンに接続した Arduino 基板を選択します。
5. 次に Arduino-IDE 画面の「ツール」メニューから、「シリアルポート」を選択し、先ほど確認した COM ポートの番号と同じものを選択します。

これで Arduino を使用する準備が整います。

ドライバーのインストールとシリアルポート（COM ポート）の確認

Mac OS X の場合

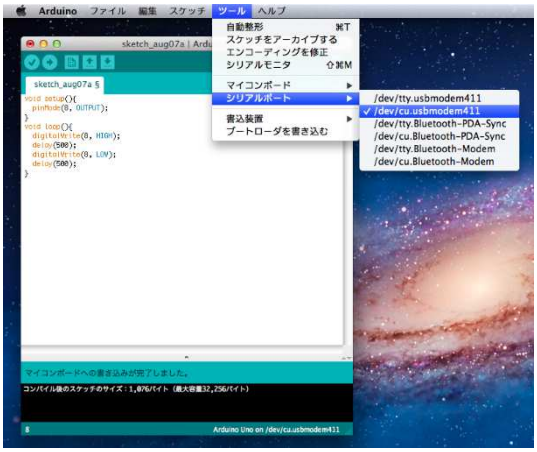
Mac OS X の場合、ダウンロードが完了すると自動的にドライバーがインストールされますので、表示されるメッセージに従い必要に応じて管理者パスワードの入力や再起動を行ってください。

Mac OS X でシリアルポートを選択する。

1.Arduino 基板をパソコンと接続します。

2.Arduino-IDE を起動し、画面の「ツール」メニューから、「マイコンボード」を選択し、パソコンに接続した Arduino 基板を選択します。

3.次に Arduino-IDE 画面の「ツール」メニューから、「シリアルポート」を選択し、「/dev/cu.usbmodem-」または「/dev/tty.usbmodem-」ではじまる項目を選んでください。



これで Arduino を使用する準備が整います。

起動画面

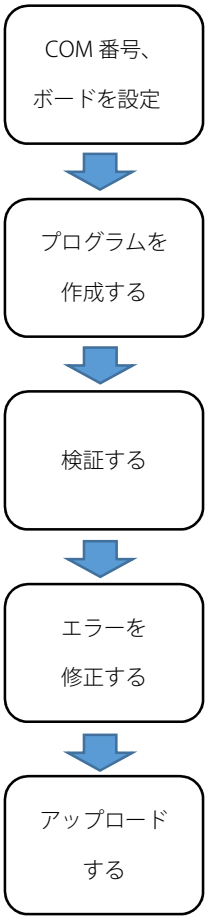
新規ファイル　：　新しいプログラム作成エリアを開きます。
開く　：　保存しているプログラムを開きます。
保存　：　作成したプログラムを保存します。
検証(コンパイル)　：　正しい文法でプログラムが作成されているかチェックします。
シリアルモニタ　：　シリアルデータを表示します。
（データに何を表示させるかあらかじめプログラムに書く必要があります。）
マイコンボード　：　プログラムを PC に接続している Arduino 基板に書き込みます。
に書き込む　アップロードと表示されている場合もあります。
プログラム作成エリア　：　プログラムを編集するエリアです。
メッセージエリア　：　操作に応じてメッセージやエラーが表示されます。
コンソール　：　メッセージやエラーの詳細が表示されます。(※)
ボード・COM 番号　：　Arduino-IDE で設定している、Arduino 基板の種類と COM ポート番号が表示されます。

※　ほとんどの場合、非常に分かりづらい意味不明の文字が表示されます。そんなときは、表示された文字や単語を使って検索サイトで調べることになります。

（3）　Arduino-IDE の使い方

●プログラムの作成からアップロードまでの流れ

プログラムを作成し、Arduino 基板に書き込むまでのおおよその流れは以下のようなイメージになります。



Arduino-IDE 起動後に一度だけ設定すれば OK です。
特に COM ポートの設定は忘れることが多いので注意しましょう。
Arduino 基板と PC は USB ケーブルでつないでおく必要があります。

プログラム作成エリアに、プログラムを作成します。

Arduino 基板に書き込む前に検証します。
エラーが表示された場合はエラーを修正します。
エラー表示は、どこが間違っているか的確に表示してくれることもありますが、ほとんどの場合、読んでもよく分からない表示になっています。それでも、エラーの発生している近くを指摘していることが多いので、エラー表示された部分の周辺をチェックすることから始めましょう。

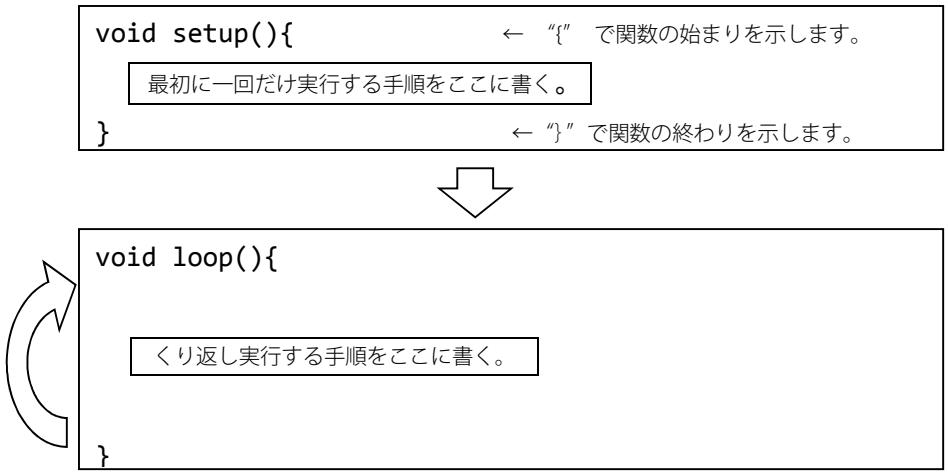
作成したプログラムを Arduino 基板に書き込みます。書き込み完了後はメッセージエリアに「書き込みが完了しました」と表示されます。

●Arduino のプログラムの基本

Arduino-IDE で作成したプログラムをアップロードすると、Arduino 基板のマイコンの中に書き込まれます。書き込まれたプログラムは、基板の電源を ON/OFF しても消えません。書き込まれたプログラムは、基板の電源を ON 後、またはリセット後に、自動で実行されます。

Arduino のプログラムには、「setup」と「loop」という 2 つの関数が必要です。
関数とは、Arduino 基板上のマイコンに、何をどんな順番で行わせるかを記述した手順書です。
電源 ON 後に、「setup」の中の手順を一回だけ実行し、続けて、「loop」の中の手順を、電源を切るまでくり返し実行するということを覚えてください。setup と loop は省略できません。

プログラムを実行するイメージ



void setup()、void loop() って何？と思うかもしれませんが、これは Arduino-IDE で、setup 関数と loop 関数を使うときの決められた形ですので、「この形で使うもの。」と覚えてください。

●よくやってしまうミス

しっかり注意しながらプログラムしたつもりでも、書き忘れやタイプミスをやってしまいます。とくに以下のようなミスはエラー表示でどこを示しているのか分かりづらいので十分注意しましょう。

；　←セミコロンを忘れる。またはセミコロンではなく、：　←（コロン）を使ってしまっている。
，　←(コロン)ではなく、　．　←（ピリオド）を使ってしまっている。
｛　と　｝　←（波カッコ）が対になっていない。　｛ ｝が多重になるときは要注意です。
全角スペースを使っている。プログラムの中では全角文字は使用できません。

●Arduino の言語

ここに書いたものが全てではありませんが、Arduino のプログラムでよく使う文法や関数をまとめています。Arduino-IDE のその他の関数や詳細については書籍などで学習を行ってください。

基本となる文法

形式　　;　（セミコロン）

説明　　文の終わりに使います。基本的に Arduino では文の終わりにセミコロンが必要です。
セミコロンによって文がどこで区切られているかを示します。日本語の句点に似ています。

使用例
```
int redLED 9 ;
pinMode(9, OUTPUT);
delay(100);
```

形式　　{　}　　（波カッコ）

説明　　複数の文をまとめるために使用します。

使用例
```
if(x==1){
  文 1;
  文 2;
}
```

デジタル入出力

形式　　`pinMode(pin, mode)`

説明　　ピンの動作を入力または出力または、プルアップ機能を利用した入力に設定します。

パラメータ　`pin`　　設定したいピンの番号

`mode`　　`INPUT`　または　`OUTPUT` または　`INPUT_PULLUP`

使用例
```
pinMode(9, OUTPUT);          9 番ピンが出力になります。
pinMode(13,INPUT);           13 番ピンが入力になります。
pinMode(A0, INPUT_PULLUP);   A0 番ピンがプルアップされた入力になります。
```

形式　`digitalWrite(pin, value)`

説明　指定したピンを、`HIGH` または `LOW` にします。`HIGH` は 5V、`LOW` は 0V(GND)になります。

パラメータ　`pin`　設定したいピンの番号

`value`　`HIGH` か `LOW`　（または、`1`　か　`0`）

使用例　`digitalWrite(9, HIGH);`　9 番ピンが `HIGH` になります。

`digitalWrite(2, 0);`　2 番ピンが `LOW` になります。

形式　`digitalRead(pin)`

説明　指定したピンの状態を調べます。結果は `HIGH` か `LOW` になります。

パラメータ　`pin`　調べたいピンの番号

使用例　`x = digitalRead(9);`　x に 9 番ピンの状態が `HIGH` か `LOW` で記憶されます。

例えば 9 番ピンが `GND` に接続されている場合は `LOW` になります。

アナログ入出力

形式　`analogRead(pin)`

説明　指定したアナログピンの状態を調べます。結果は 0 から 5V の電圧範囲を 0 から 1023 の範囲に変換した値になります。Arduino-UNO では A0～A5 がアナログピンとして利用できます。

パラメータ　`pin`　調べたいアナログピンの番号

使用例　`x = analogRead(A0);`　x に A0 番ピンの状態が 0 から 1023 の整数値で記憶されます。

形式　`analogWrite(pin, value)`

説明　指定したピンから PWM 波を出力（アナログ出力）します。

`analogWrite` の前に `pinMode` で出力にする必要はありません。

Arduino-UNO では３，５，６，９，１０，１１番ピンがアナログ出力ピンとして利用できます。

`analogWrite` を実行すると、次に同じピンで `analogWrite` を実行するまで PWM 波が出力されます。

パラメータ　`pin`　出力に設定するピン番号

`value`　PWM 出力のデューティ比（0 から 255 の範囲で設定します。）

使用例　`analogWrite(3, 64);`　3 番ピンからデューティ比が 64 の波形を出力します。

`analogWrite(3, 0);`　3 番ピンからはデューティ比が 0(=0V)が出力されます。

`analogWrite(3, 255);`　3 番ピンからはデューティ比が 255(=5V)が出力されます。

制御文

形式　　if (条件) { 条件に一致したときに実行する文 }

説明　　カッコ内の条件が true つまり条件が満たされている場合は、次に続く波カッコ内の文を実行します。
false つまり条件が満たされていない場合は、波カッコ内の文を実行せずに次に進みます。
波カッコの中の文が 1 つだけの場合は波カッコを省略できます。

使用例
```
if ( x == 1 ) {
  digitalWrite(9, HIGH);
  delay(100);
}

if ( y < 500 ) digitalWrite(4, LOW);
```

注意点　比較に利用する　＝（等号）は　＝＝のように 2 つ書かなければいけません。
＝が 1 つの場合は代入となり、全く違った意味になり、結果としていつも true と判断されます。

形式　　if(条件){条件に一致したときに実行する} else {条件に一致しないときに実行する}

説明　　カッコ内の条件に一致したときに実行する文と、一致しないときに実行する文をそれぞれ分けて書くことができ、if よりも細かく制御できます。

使用例
```
if (x == 1) {
  digitalWrite(9, HIGH);
}else{
  digitalWrite(9, LOW);
}
```

形式　　for (初期化 ; 条件式 ; 加算) {実行する文}

説明　　条件を満たしている間、波カッコで囲まれた文を繰り返し実行します。
カッコの中は 3 つの部分から成り立っています。
動作は、まず初期化が一度だけ実行されます。次に、条件式がテストされ条件を満たしていれば、加算と波カッコ内の文が繰り返し実行されます。次に条件がテストされたときに条件を満たしていなければ、そこで繰り返しが終わります。

使用例
```
for ( i = 0; i <= 255; i++){
  analogWrite(9, i);
  delay(100);
}
```

形式　`while(条件) {条件に一致したときに実行する文}`

説明　カッコ内の条件を満たさなくなるまで（`false` になるまで）波カッコ内を永遠に繰り返します。

使用例
```
x = 1;
While( x < 50 ){
  x++;
}
```

時間

形式　`delay(ms)`

説明　カッコ内で指定された時間待って、次の文の処理へ進みます。単位はミリ秒です。

パラメータ　`ms`　次の処理までの待ち時間。単位はミリ秒。（1 ミリ秒は 1000 分の 1 秒です。）

使用例
```
digitalWrite(9,HIGH);
delay(500);
digitalWrite(9,LOW);
```

便利な機能

形式　`#define　定数名　値`

説明　検証時（コンパイル時）に自動的に定数名を値に変換します。プログラムを見やすく、間違いを少なくするために有効です。`#`を付けることを忘れないようにします。

`#define` 文の最後には；（セミコロン）は必要ありません。

使用例
```
#define ledPin 4
digitalWirte(ledPin, HIGH);
```
`ledPin` は検証時に自動的に `4` に置き換えられます。

形式　`#include　<ライブラリ名>`

説明　外部に用意されているライブラリをプログラムに取り入れたいときに使います。

ライブラリは Arduino-IDE にあらかじめ用意されているものや、インターネットから入手できるものなどがあります。

`#include` 文の最後には；（セミコロン）は必要ありません。

使用例　`#include　<LiquidCrystal.h>`　LCD 表示器のライブラリを使うことができるようになります。

形式　　// または /* ～ */

説明　　プログラムの中にメモしておきたい内容、つまりコメントを書くときに使います。
コメントは検証（コンパイル）から無視され、プログラムに影響しません。
コメントは、プログラムの動作内容を理解したり、思い出したりするために使われます。

使用例　x = 1;　//ここがコメントになる。　ダブルスラッシュの後ろはすべてコメントです。

```
/*      ここから複数行のコメントになります。
  digitalWrite(3,LOW);
  x = analogRead(A0);
*/      ここまでがコメントです。
```

シリアル通信

形式　　Serial.begin(speed)

説明　　シリアル通信の通信速度を設定します。

パラメータ　speed　通信スピードの値。一般的にパソコンと通信する場合は、
300,1200,2400,4800,9600,14400,28800,38400,57600,11520
から 1 つを選びます。

使用例　Serial.bigin(9600);

形式　　Serial.print(data, format)

説明　　シリアルポートに指定されたデータを出力します。

パラメータ　data　すべての整数型と String 型(*)
(※アルファベット、数字、区切り記号などの文字が連続したもの。
ダブルクオテーションで囲むことで String 型になります。)

format　data を変換する方法を指定します。省略可能です。
省略すると、data は 10 進数の数値として取り扱われます。

使用例　Serial.print(58);　58 を出力します。
Serial.print(58, HEX);　3A を出力します。
Serial.print(“Hello”);　Hello を出力します。

形式　　Serial.println(data, format)

説明　　データの最後に改行コードを付けて出力します。それ以外の動作は Serial.print と同じです。

使用例　Serial.println(58);

データ型

変数そのものや、変数に代入される値の範囲に合わせて、データ型を選ぶ必要があります。
取り扱える範囲を超えた計算を行うと、変数は期待している値と違ったものになるので注意が必要です。
データ型と、値の範囲は以下の表のようになります。

データ型	値の範囲
boolean	true か false のどちらかの値を取り扱います。
byte	0 から 255 までの値を取り扱います。負の値は扱えません。
int	-32,768 から 32.767 までの値を取り扱います。
long	-2,147,483,648 から 2,147,483,647 までの値を取り扱います。

※これ以外にもデータ型はあります。代表的なものだけ紹介しています。

使用例　`int x ;`　　x は int 型の範囲の数値を取り扱う変数になります。
　　`byte y = 128 ;`　　byte 型の変数 y に 128 を代入します。

演算子

形式　＝
説明　定数や計算結果を変数に記憶させるには、＝(等号) を使います。＝は代入演算子といわれます。
使用例　`a = 10;`　　a は 10 になります。
　　`b = a;`　　b は a と同じ値になります。

形式　`+ , - , * , /`
説明　加算、減算、乗算、除算を行います。これらは算術演算子といわれます。
　　データ型によって計算結果が期待しているものと違ってくることがありますので、
　　計算結果を格納するのに十分な大きさの型になっていることに注意が必要です。
　　※プログラムをスマートにする演算子は他にもあります。ここでは基本的なものだけです。
使用例　`a = 10 + 1;`　　a は 11 になります。
　　`a = 9 - 1;`　　a は 8 になります。
　　`a = 3 * 33;`　　a は 99 になります。
　　`a = 9 / 3;`　　a は 3 になります。
　　`a = 9 / 4;`　　a のデータ型によって変わります。a が int 型であれば、
　　a は 2 になります。
　　※少数が取り扱いたい場合は別のデータ型を選ぶ必要があります。

形式　　++　--

説明　　++をインクリメントといい、1 を足します。--をデクリメントといい、1 を引きます。

使用例　++a;　　a に 1 を足して a に記憶します。

　　--a;　　a から 1 を引いて a に記憶します。

比較演算子

形式　　== , != , < , > , <= , >=

説明　　2 つの変数や定数の関係を調べるには、==,!=,<,>,<=,>=を使います。

関係が正しい場合を真(true)といい、正しくない場合を偽(false)といいます。

演算子	使用例	結果
==	a==b	a と b が等しければ真
!=	a!=b	a と b が等しくなければ真
<	a<b	a が b より小さければ真
>	a>b	a が b より大きければ真
<=	a<=b	a が b 以下なら真
>=	a>=b	a が b 以上なら真

使用例　a = (1>2);　　a は false(=0)になる。

a = (1!=2);　　a は true(=1)になる。

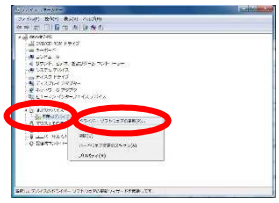
a = (1==2);　　a は false(=0)になる。

トラブルシューティング

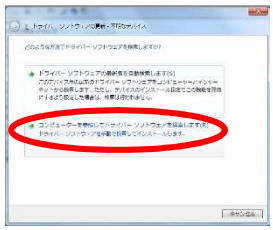
WindowsVista/7 に手動でドライバーをインストールする

（1）　コントロールパネル　→　　システムのセキュリティ　　→　デバイスマネージャー　と選んで、デバイスマネージャーの画面を開きます。

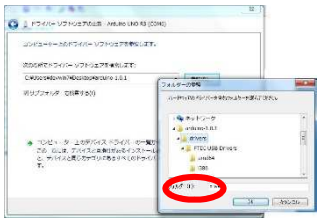
（2）　デバイスマネージャー画面の中に、「不明なデバイス」という表示がされていますので、その文字の上で右クリックして、「ドライバーソフトウェアの更新・・・」を選びます。



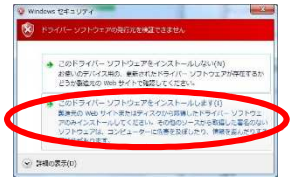
（3）　「ドライバーソフトウェアの更新」画面から、「コンピューターを参照してドライバーソフトウェアを検索します。」を選びます。



（4）　「次の場所でドライバーソフトウェアを検索します。」→「参照」を選び、最初に保存した Arduino のフォルダーの中から Drivers を選びます。



(5)　Windows セキュリティの画面が表れますので、「このドライバーソフトウェアをインストールする。」を選びます。



(6)　しばらく待つと、ドライバーのインストール完了画面が表示されます。

(7)　もう一度デバイスマネージャーを表示して、Arduino 基板が、ポート（COM と LPT）に登録されていることを確認してください。