Arduino 入門

2025/03/14 版(v1.0)

配布元: https://n.mtng.org/ele/arduino/

著者: 光永 法明

https://n.mtng.org/

1 マイコンボードの概要(端子(ピン)と LED とリセットスイッチ)

マイコンボード(ここでは秋月電子の ATMEGA168/328 用マイコンボード（Ｉ／Ｏボード）(図 1-1)http://akizukidenshi.com/catalog/g/gP-04399/ , Arduino Uno R3(図 1-2)と機能がほぼ同じ)には USB コネクタ(USB mini B)、DC ジャック（AC アダプタをつなぐ端子）と、他の回路や電子部品とつなぐための横に並んだ端子の列が 4 つあります。端子の列はそのまま挿せるシールドと呼ばれる基板（市販されています。自作もできます。）を挿したり、ジャンプワイヤーとよばれる電線を挿してほかの回路や電子部品とつなぎます。端子の列の各端子の役割は基板上に印刷されています。各端子はマイコンのピン（足）につながっているので、ピンとよぶことがあり、以下でもそう呼びます。

マイコンボードには **4 つの LED** があります。PWR または ON とあるのは電源が入っていると点灯する LED です。USB 端子で PC とつなぐと点灯します。TX, RX と表記のある LED は PC とマイコンが通信をしているときに点滅する LED です。**AREF の近くにある LED** はプログラムから点灯/消灯できる LED です。マイコンボード上にあるスイッチは**リセットスイッチ**です。マイコンの電源を入れたときと同じようにプログラムを最初から実行するときに使います。

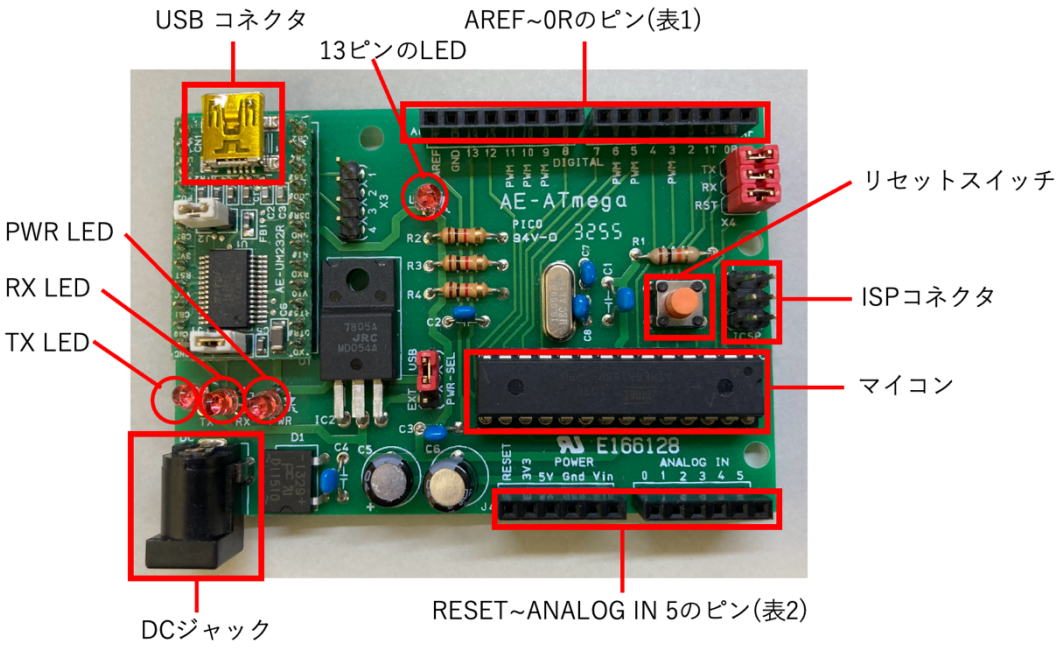


図 1-1 秋月電子の ATMEGA168/328 用マイコンボード

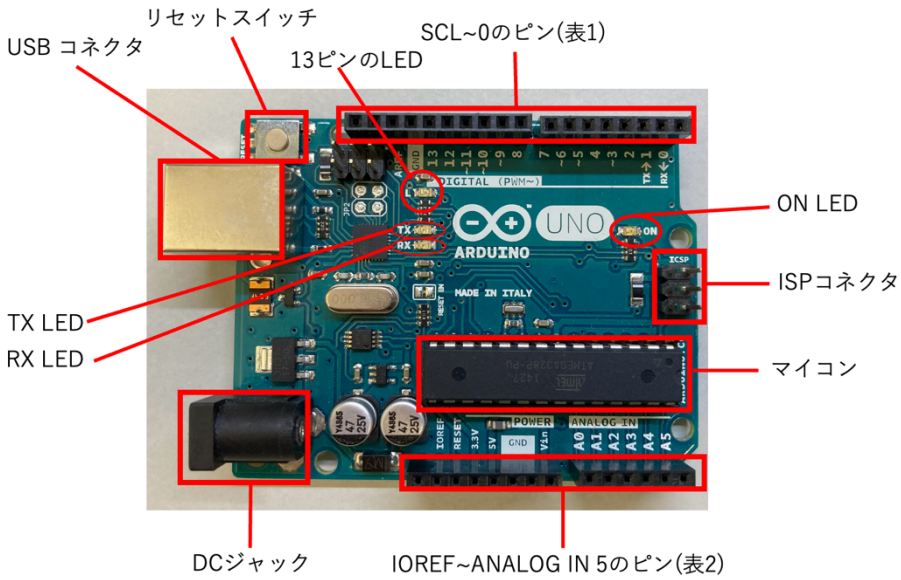


図 1-2 Arduino Uno R3 マイコンボード

各ピンの基板上での表記と機能、プログラム上での番号は表 1- 1 と表 1- 2 となっています。表から分かるようにデジタル出力に使えるピンには 0～13 番のピンと ANALOG IN(アナログ入力)にある 0～5 の合計 19 本あります。0～13 番のピンを D0, D1 ピンと、アナログ入力のピンは A0, A1 ピンと呼ぶことがあります。0 番 1 番とアナログ入力のピンは少し注意がいるので最初は 2 番から 13 番のピンを使うとよいでしょう。また 13 番のピンは基板上の LED(図 1-1, 図 1-2 で 13 ピンの LED と書いている LED)につながっています。

マイコンは電源を切っても書き込んだプログラムを憶えています。そのためプログラムが完成した後はパソコンとの USB 接続を外し、DC ジャックに AC アダプタをつなぐだけで動作します。もちろん USB コネクタにモバイルバッテリー等をつないでも動作します。ISP(In-circuit Serial Programming)コネクタはマイコン上のブートローダとよばれるプログラムを書き換えるときなどに使います。

マイコンボードで何ができるかは次のページを見て下さい。マイコンボードを使った作品の動画集になっています：

https://n.mtng.org/ele/arduino/samples/sample90.html

表1-1 DIGITAL と表示のある側のピンの機能とプログラム上での番号

基板上の表記	機能	プログラム上での番号/表記
0R / 0	デジタル入力/デジタル出力/シリアルポート	0
1T / 1	デジタル入力/デジタル出力/シリアルポート	1
2	デジタル入力/デジタル出力	2
3	デジタル入力/デジタル出力/PWM	3
4	デジタル入力/デジタル出力	4
5	デジタル入力/デジタル出力/PWM	5
6	デジタル入力/デジタル出力/PWM	6
7	デジタル入力/デジタル出力	7
8	デジタル入力/デジタル出力	8
9	デジタル入力/デジタル出力/PWM	9
10	デジタル入力/デジタル出力/PWM	10
11	デジタル入力/デジタル出力/PWM	11
12	デジタル入力/デジタル出力	12
13	デジタル入力/デジタル出力	13
GND	グラウンド(電源のマイナス側)	なし
AREF	A/D 変換の基準電圧	なし
SDA	I2C の SDA (Arduino UNO R3 のみ, A4 ピンに接続)	A4
SCL	I2C の SCL (Arduino UNO R3 のみ, A5 ピンに接続)	A5

表 1-2 POWER, ANALOG IN と表示のある側のピンの機能とプログラム上での番号

基板上の表記	機能	プログラム上での番号/表記方法
IOREF	マイコン駆動電圧(Arduino UNO R3 のみ，基板上で 5V に接続)	なし
RESET	マイコンのリセットピン	なし
3V3	3.3V 電源	なし
5V	5V 電源	なし
Gnd	グラウンド（電源のマイナス側）	なし
Gnd	グラウンド（電源のマイナス側）	なし
Vin	USB の 5V または AC アダプタのプラス側	なし
0	デジタル入力/デジタル出力/アナログ入力	A0
1	デジタル入力/デジタル出力/アナログ入力	A1
2	デジタル入力/デジタル出力/アナログ入力	A2
3	デジタル入力/デジタル出力/アナログ入力	A3
4	デジタル入力/デジタル出力/アナログ入力	A4
5	デジタル入力/デジタル出力/アナログ入力	A5

Arduino IDE をインストールして起動すると図 2-1 の画面が表示されます。プログラムを書く前に「マイコンボードとシリアルポートの選択」をクリックして正しいものを選びましょう。マイコンボードの名前は ATMEGA168/328 用マイコンボードでは Arduino Duemilanove or Diecimila を選びます（Uno のブートローダを書き込んでいる場合には Uno を選びます）。Arduino Uno R3 の場合は Arduino Uno を選びます。プログラムを書いてコンパイルできるか確かめるには「検証ボタン」をクリックします。書いたプログラムをコンパイルしてマイコンボードに書き込むには「書き込みボタン」をクリックします。コンパイルのときや書き込みの時には「コンパイラ等からのメッセージ」のところに途中経過などが表示されます。赤文字の時はエラーなので確認しましょう。Windows の場合には**パソコンの USB コネクタを変えるとシリアルポートが変わる**ので注意してください。シリアルポートが変わったら書き込めないので再度「マイコンボードとシリアルポートの選択」をクリックして設定します。

Arduino IDE ではプログラムのことをスケッチとよびます。これはアイデアスケッチを絵で描くようにプログラムも気軽に書いて欲しいという思いからだそうです。スケッチはパソコンのドキュメントフォルダ内に Arduino フォルダを作って保存するように初期設定ではなっています。そこの保存したファイルは、「ファイル」メニューの「スケッチブック」のところから開くことができます。また「ファイル」メニューの「スケッチ例」のからサンプルプログラムを開くことができます。

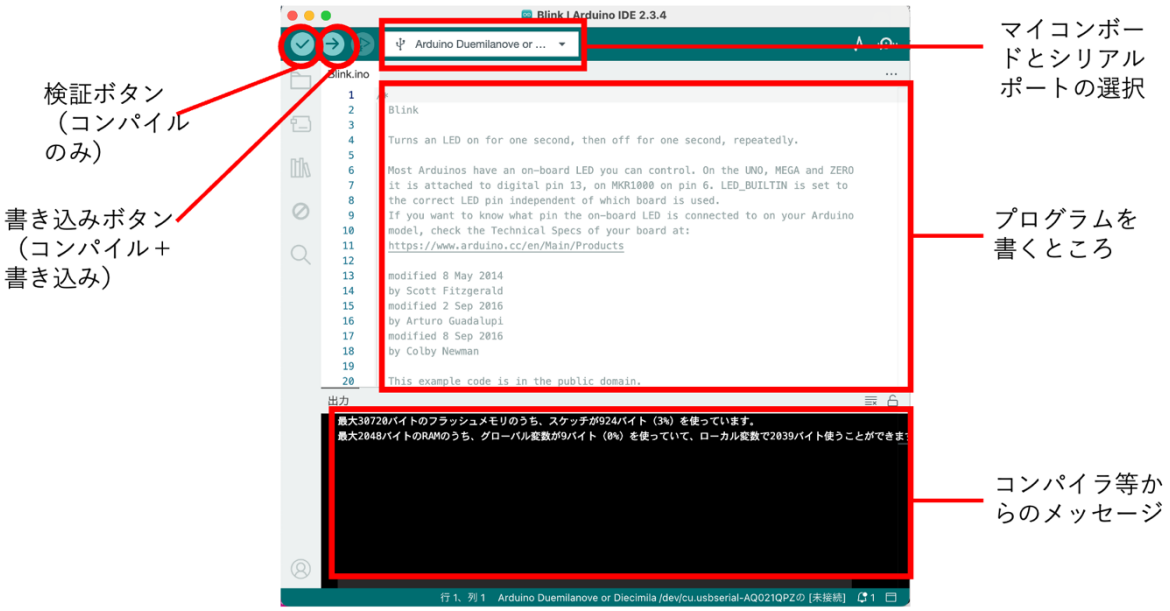


図 2-1 Arduino IDE の画面（macOS 14.7, Arduino IDE 2.3.4）

3 プログラムを書き込んでみよう

Arduino IDE を起動して、「ファイル」メニューから「スケッチ例」の中の「01.Basics」の「Blink」を開いてください。マイコンボード上の LED を点滅させるプログラムです。マイコンボードとシリアルポートを正しく選択します(図 3-1)。マイコンボード（ボード）はたくさんあるので「ボードを検索」のところに due, uno などをいれて候補を絞るとよいです。シリアルポートが複数あるときには順番に試すか、マイコンボードをパソコンにつなぐ前に様子を確認しておき、つないでから増えたポートを選びます。そして「書き込み」ボタンをクリックルとコンパイルが始まり、プログラムがマイコンボードに書き込まれます。書き込み中には RX, TX の LED が点滅します。正しくコンパイルして書き込みができるときの Arduino IDE の表示、書き込み時の Arduino IDE の表示と LED の点滅の様子を見て憶えておくとよいでしょう。

うまく書き込みができたら Blink プログラムの数字（プログラムリスト 3-1）を 100, 500, 2000 などと変えて（片方だけ両方ともなど組み合わせも変えて）「書き込み」ボタンをクリックしてみましょう。LED の点滅の様子が変わるのがわかると思います。

図 3-1 マイコンボードとシリアルポートの選択（macOS 14.7, Arduino IDE 2.3.4）

プログラムリスト 3-1 Blink プログラム抜粋

```
31  // the loop function runs over and over again forever
32  void loop() {
33    digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
34    delay(1000);                      // wait for a second
35    digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
36    delay(1000);                      // wait for a second
37  }
```

Arduino IDE を起動して「ファイル」メニューから「スケッチ例」の中の「01.Basics」の「BareMinimum」を開いてください。プログラムリスト 4-1 のスケッチが開きます。これが最小の Arduino のスケッチ（プログラム、以下プログラムと書きます）となります。// で始まる行はコメント行です。コメントというのはプログラムの動作には関係しないメモのようなものです。プログラムリスト 2 ではコメントが 2 行あります。英語で書かれていますが、put your setup code here, to run once は「ここに初期化のプログラムを書いて下さい」put your main code here, to run repeatedly は「ここに繰り返すプログラムを書いて下さい」と言う意味です。Arduino のプログラムを書くということは setup 関数と loop 関数のプログラム（命令文）を書くことになります。プログラムを書くと図 4-1 のフローチャートのように実行します。setup 関数のプログラムは setup()につづく中括弧{ }の中に loop 関数のプログラムは loop()につづく中括弧の中に書きます。setup()関数、loop 関数共にプログラム中に 1 つだけ書けます。**複数あるとエラー**になります。

「スケッチ例」の中の「01.Basics」の「Blink」のプログラムからコメントを省くとプログラムリスト 4-1 となります。setup 関数は 1 行、loop 関数は 4 行のプログラムとなっています。プログラムは文（命令文）を並べて書きます。**文の区切りは「;」（セミコロン）**です。このプログラムのフローチャートを書くと図 4-2 になります。loop 関数内のプログラムがずっと繰り返すのでマイコンボード上の LED が点滅します。

プログラムリスト 4-1　BareMinimum スケッチ

```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

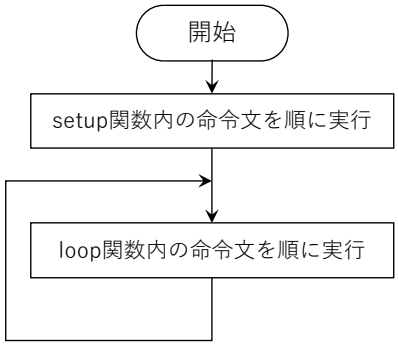


図 4-1　Arduino のプログラムの基本動作

プログラムリスト 4-1　Blink スケッチからの抜粋

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

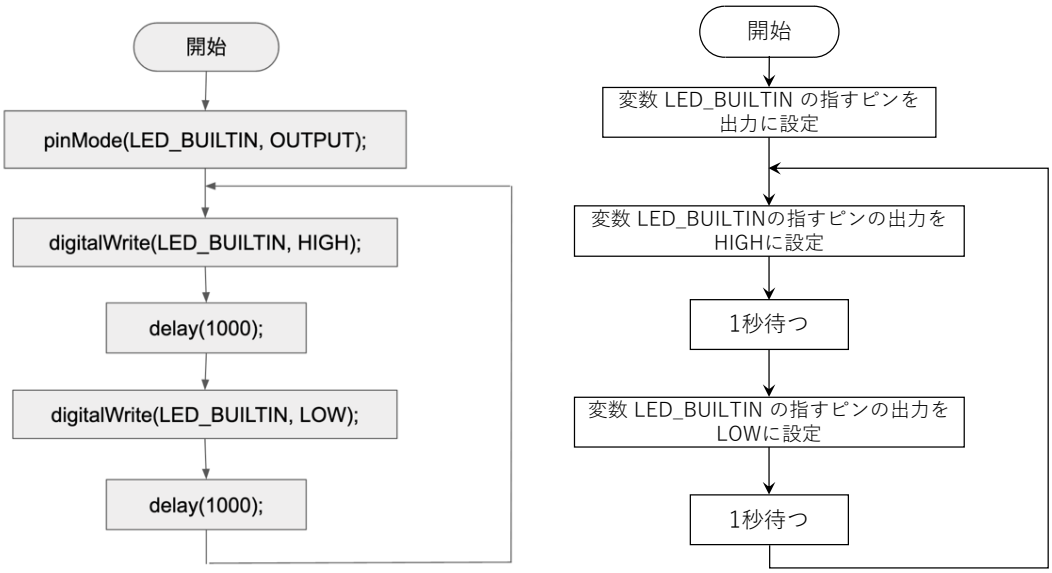


図 4-2　Blink スケッチのフローチャート(左：プログラムの命令文で表記、右：動作で表記)

フローチャートに使われる代表的な記号は 5 つあります（図 4.1-1）。フローチャートは必ず「開始」の記号で始めます。また終わりがある流れを表す場合は「終了」の記号で終わります。たとえばプログラムリスト 4-1 の loop 関数の中身を表すのであれば図 4.1-2 のように書きます。**この二つの記号は半円と上下の線**でできています。楕円ではないことに注意しましょう。

処理の記号（長方形）は図 4-2 のようにプログラムの命令文や、そこで実行する内容を書きます。一つのフローチャートの中では命令文を書くのか、内容を書くのか統一しましょう。**記号の間は矢印でつなぎ**ます。フローチャートは上にも右にも広げて書いてよいので矢印がないと順番が分からなくなります。分岐の記号（菱形）は **2 つに分かれる場合**に使います。3 つ以上の分岐では使いません（別の記号があります）。また分岐の記号は上から矢印が入り、下と横（左右のどちらか）に次の処理（もしくは分岐）への矢印を出します。

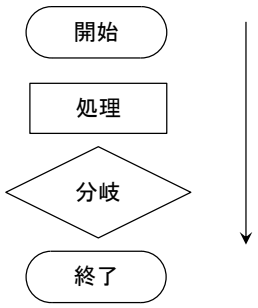


図 4.1-1 フローチャートの記号

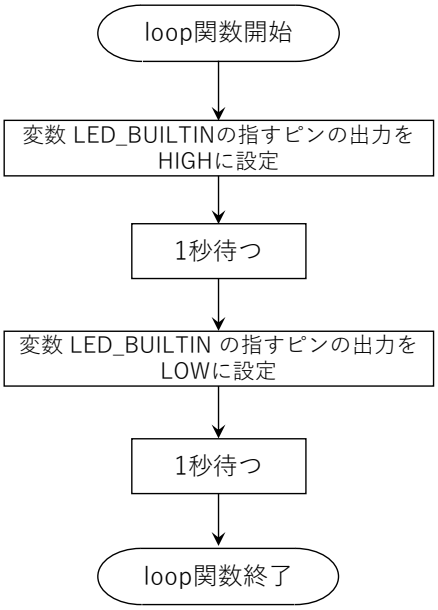


図 4.1-2 プログラムリスト 3 の loop 関数の動作を表すフローチャート

練習問題 4.1

1. プログラムリスト 3 の setup 関数の動作を表すフローチャートを書いてみよう。

4.2 文内の空白

文に入っている空白（スペース、改行、タブ）は無視されます。見やすいようにいれて構いません。また長い行は複数行に分けても構いません。たとえば

```
digitalWrite(LED_BUILTIN, HIGH);
digitalWrite(LED_BUILTIN,          HIGH);
digitalWrite(LED_BUILTIN,
    HIGH);
```

は、同じ文としてコンパイルされます。一方で改行をしても「;」がないと文が繋がっているとみなされるので**「;」忘れ**に注意してください。また英語で別単語となるように digitalWrite と digital Write は別物として扱われます。**大文字と小文字、全角と半角は違う文字として扱われる**ので注意してください。

4.3 コメントの書き方

先頭に//がある行は// より後の行末までがコメントとして扱われます。また /* と */ で囲まれた部分もコメントとして扱われます。/* と */ の間が複数の行になっても構いません。同じ行の /* より前と */ 以降はプログラムの一部となります。また

```
digital Write(/* コメント */ LED_BUILTIN, /* コメント */ HIGH);
```

のように行中に何回も /* */ で囲んだコメントを書いても構いません。ただし、/* */ のなかに /* */ は書けません。書くとと一つ目の */ 以降がコメントでなくなってしまいます。

4.4 ピンの機能の切り替え（pinMode 関数）

マイコンの電源が入った直後やリセットした直後には各ピンはデジタル入力またはアナログ入力になっています（つないだ回路がマイコンに信号を出力する回路で、マイコン側も出力になっていると信号の衝突が起きてマイコン又は回路を壊す危険があるため）。デジタル出力に切り替えるにはプログラムで

```
pinMode(<ピン番号>, OUTPUT);
```

デジタル入力に切り替えるにはプログラムで

```
pinMode(<ピン番号>, INPUT);
```

と書きます。デジタル入力で内蔵プルアップを有効にするには

```
pinMode(<ピン番号>, INPUT_PULLUP);
```

と書きます。<ピン番号>のところには直接数値を書いたり、変数や計算式を書きます。2～13 ピンでデジタル入力を使う場合と、アナログ入力ピン A0～A5 でアナログ入力を使う場合は pinMode を書かなくて構いません。

例1)　pinMode(2, OUTPUT);　// 2 番ピンを出力モードにする

例2)　pinMode(i, OUTPUT);　// 変数 i の値が指すピンを出力モードにする

例3)　for (int i=2; i<=13; i ++) {　// 2 番ピンから 13 番ピンを出力モードにする

```
    pinMode (i, OUTPUT);
}
```

例4)　for (int i=0; i<10; i ++) { // 2 番ピンから 10 個(11 番ピンまで)を出力モードにする

```
    pinMode (i+2, OUTPUT);
}
```

デジタル出力を使うとマイコンから電圧を出力することができます。電圧の状態は LOW（または 0）と HIGH(または 1)で表します。正確には少し違いますが図 7 のように LOW に設定すると電源のグラウンド側（マイナス側）に HIGH にするとプラス側(ATMEGA168/328 用マイコンボードと Arduino Uno R3 では 5V)にスイッチが切り替わると思って下さい。つまりデジタル出力はプログラムから制御できる電源になります。デジタル出力を電源として使う場合は、マイコン内のスイッチや配線が扱える電流に制限があるため、1 ピンあたり 10mA までにしておくのが無難です。そのため LED を点灯することはできますが、模型用のモータを直接つなぐことはできません。

デジタル出力の値を変えるには

```
digitalWrite(<式>, HIGH);
```

又は

```
digitalWrite(<式>, LOW);
```

と書きます。一度 digitalWrite の文を実行すると次に実行するまで**ピンの値(HIGH/LOW)はそのまま**です。式のところには数字を書いても構いませんし、変数や式を使っても構いません。

```
例5)  digitalWrite(2, HIGH);   // 2 番ピンを HIGH にする
例6)  digitalWrite(i, LOW);    //  変数 i の値が指すピンを LOW にする
例7)  for (int i=2; i<=13; i ++) {   // 2 番ピンから 13 番ピンを HIGH にする
        digitalWrite (i, HIGH);
      }
例8)  for (int i=0; i<10; i ++) { // 2 番ピンから 10 個(11 番ピンまで)を HIGH にする
        digitalWrite (i+2, LOW);
      }
```

LED を点灯する回路(図 4.5-1)を例に考えてみましょう。図 4.5-2 のように抵抗と LED と電源とスイッチをつなぐとスイッチで LED の点灯/消灯ができます。図 4.5-2 では電源に手動のスイッチが付いていますが、これをマイコンのピンに置き換えます(図 4.5-3)。図 4.5-3 ではマイコンの 2 番ピンに抵抗と LED をつないでいます。電源のグラウンドと GND の端子はマイコンボード内でつながっていますので、プログラムからスイッチを HIGH 側につなぐと LED に電流が流れて点灯し、LOW 側につなぐと消灯します。プログラムでは

```
digitalWrite(2, HIGH);
```

または

```
digitalWrite(2, LOW);
```

と書きます。マイコンの 13 番ピンにはボード上で図 4.5-3 のように LED がつながっているので配線することなくプログラムの動作を確かめられます。プログラムリスト 3 の LED_BUILTIN は 13 と書く代わりです（ほかのマイコンボードで 13 でないときにも対応できるようにしています）。また図 4.5-3 の回路は図 4.5-4 のように書きます(別の表記をすることもあります)。

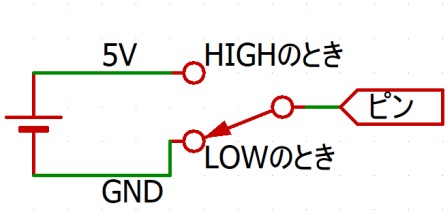


図 4.5-1 デジタル出力の概念図

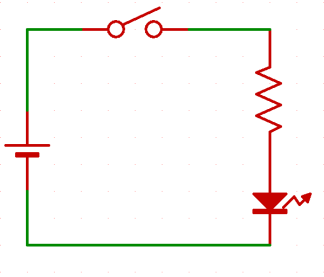


図 4.5-2 LED をスイッチで点滅する回路

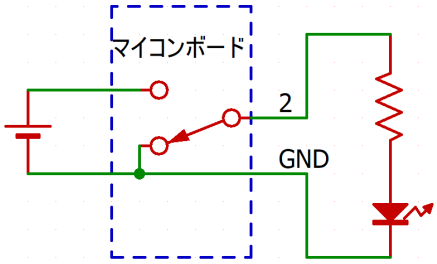


図 4.5-3 マイコンの 2 番ピンに LED つなぎ点灯制御する回路（概念図）

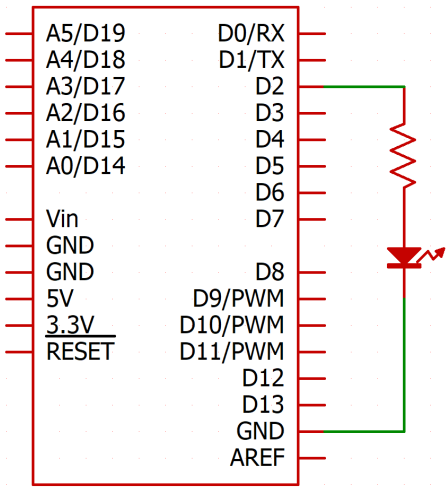


図 4.5-4 マイコンの 2 番ピンに LED をつなぎ点灯制御する回路

ここで使うマイコンはマシン語（機械語ともいう）だと 1 秒間に 1600 万もの命令を実行できる性能をもっています。Arduino のプログラムでも 1 秒間に 1 万から 10 万ぐらいの命令文を実行できます（命令文によります）。LED の点灯の方法と消灯の方法は分かったので図 4.5-3 の LED を点滅させようと

```
digitalWrite(2, HIGH);
digitalWrite(2, LOW);
```

を loop 関数に書いても、点灯と消灯の間隔が短すぎて人の目には点滅しているか分かりません。そこで一定時間なにもしない（待つだけの）命令文を書くために delay 関数があります。

```
delay(<式>);
```

と書くと<式>の計算結果の数値×1/1000 秒(数値の表す ms（ミリ秒）)だけ次の命令文を実行するのを待ちます。書き方の例は次のようになります：

```
例9）  delay (1000);          // 1 秒間待つ
例10） dealy(10*1000);        // 10 秒間待つ
例11） dealy(t);              // 変数 t の値×1/1000 秒間待つ
```

4.7 マイコンボード上の LED を点滅するプログラム

ここまでの pinMode, digitalWrite, delay を使うとマイコンボード上の LED(13 番につながっている)を 1 秒点灯、1 秒消灯することを繰り返すプログラムはプログラムリスト 4.7-1 のように書けます。プログラムリスト 4.7-2 では最初に led という名前の値を変更できない変数を宣言して 13 を代入しています。プログラム内で led と書くと変数の値が 13 なので直接 13 を書いたのと同じことになります。わざわざ宣言しているのは、13 ではなく led と書くことで意味をわかりやすくし、必要なら 1 カ所を書き換えれば別の数字にできるようにするためです。たとえば基板上の LED ではなく 2 番ピンにつないだ LED を点滅したくなったときにプログラムリスト 4.7-1 では 3 カ所書き換える必要がありますが、プログラムリスト 4.7-2 では 1 カ所だけ書き換えればよいのでミスを防げます。

なおプログラムリスト 4.7-1, 4.7-2 のプログラムは Arduino IDE のファイルメニューから「スケッチ例」→「01 BASICS」→「Blink」の例(プログラムリスト 4-1)と同じ動作をします。違いは LED_BUILTIN, 13, led のいずれを pinMode, digitalWrite に書くかというころだけです。LED_BUILTIN はあらかじめ Arduino に用意されていて、マイコンボード毎の違いを吸収するものです。

練習問題 4.7

1. プログラムリスト 4-1, 4.7-1, 4.7-2 が同じ動作をすることをマイコンボードに書き込んで確かめよう。
2. 2 つの delay 関数の行の先頭に // を書いてコメントアウトしたプログラムを書き込み、最高速度で LED を点滅させて何が起こるか確認しよう。
3. loop 関数の中にある 4 行をすべて setup 関数の pinMode の次の行に移動して見ましょう。どうなるか予想してからプログラムを書き込んで動作を確認しよう。またプログラムを書き込んで動作が終わったらリセットボタンを押して何が起こるか確認しよう。

プログラムリスト 4.7-1 基板上の LED を点滅するプログラム(1)

```
void setup() {
        pinMode(13, OUTPUT);
}

void loop() {
        digitalWrite(13, HIGH);
        delay(1000);
        digitalWrite(13, LOW);
        delay(1000);
}
```

プログラムリスト 4.7-2 基板上の LED を点滅するプログラム(2)

```
const int led = 13;
void setup() {
        pinMode(led, OUTPUT);
}
void loop() {
        digitalWrite(led, HIGH);
        delay(1000);
        digitalWrite(led, LOW);
        delay(1000);
}
```

マイコンボードで作品を作るときに便利なシステムに Seeedstudio 製の Grove モジュールがあります[1](図 5-1)。Grove モジュールはマイコンボードに Grove ベースシールドを挿し、必要な Grove モジュールとベースシールドを Grove ケーブルでつなぎます。Grove ベースシールド（図 5-2）の左下にスイッチがありモジュールの電源電圧を 3.3V と 5V で選択できるようになっています。ここで使っているマイコンボードとモジュールの場合はスイッチを 5V 側にして使います。

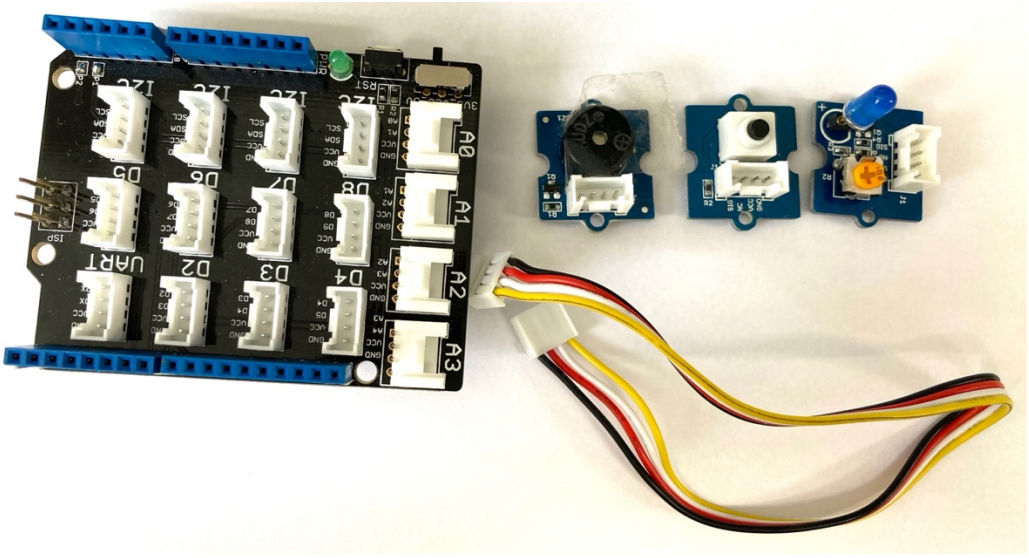


図 5-1 Grove ベースシールド（左）と Grove モジュール(右上、左からブザー、ボタン（スイッチ）、LED)、Grove ケーブル（右下）

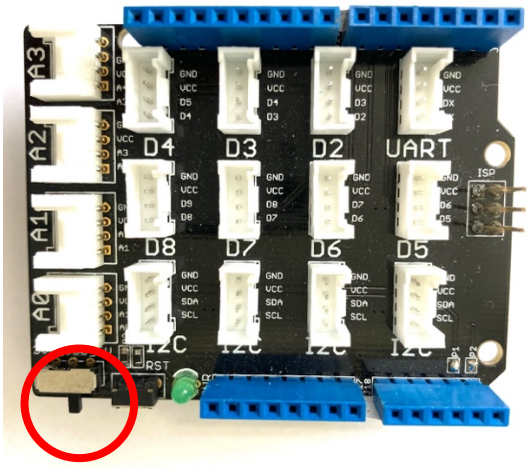


図 5-2 Grove ベースシールド。左下のスイッチ（赤丸内）は 5V で使う。

[1] https://lab.seeed.co.jp/entry/2019/10/25/120432

Grove ベースシールドには UART, D2～D8, A0～A3 のコネクタ（白いコネクタ）, I2C(4 つ)とマイコンボードと同じ位置にピンソケット（青）があります。白いコネクタは Grove モジュールをつなぐときに、ピンソケットはブレッドボードを使うときなどに使います。UART は非同期通信用です。D2～D8 はデジタルピン用、A0～A3 はアナログ入力用（デジタルピンとしても使える）、I2C は I2C 通信用です。I2C の 4 つのコネクタは内部で繋がっていてどれに挿しても同じです。

Grove コネクタは 4 つの端子があります(図 5-3)。GND は電源のグラウンド(電源の基準で 0V)、VCC は電源で 5V（先述のスイッチで決まる）です。残りの 2 つの端子が信号の端子です。D4 のコネクタには D4 と D5 が、D8 のコネクタには D8 と D9 の端子が出ています。I2C コネクタには電源の GND と VCC の他に SDA と SCL の端子が出ています。

ここで紹介するブザー、ボタン（スイッチ）、LED のモジュールの場合は 2 本の信号の内、若い側の信号だけ使うので D2～D8 コネクタにつなぐと 2～8 ピンのデジタル入出力でプログラムから操作できます。たとえば D2 つなぐとプログラム中では

```
pinMode(2, OUTPUT);
digitalWrite(2, HIGH);
digitalRead(2)
```

と D をとった数字で操作します。

A0～A3 だとプログラム上でも A0～A3 になります。たとえば A0 につなぐと

```
pinMode(A0, OUTPUT);
digitalWrite(A0, HIGH);
digitalRead(A0)
```

です。I2C コネクタは I2C 液晶モジュールに使います。

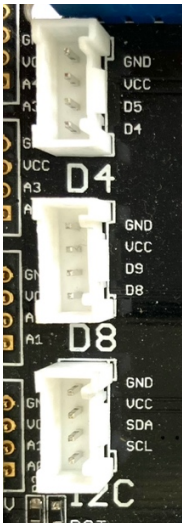


図 5-3 Grove ベースシールドの Grove コネクタ部の拡大

LED モジュール(図 5.1-1)は LED が 1 個載っています。D2～D8 の端子あるいは A0～A3 の端子に Grove ケーブルでつないで使います。モジュール上の半固定抵抗（図 xx では黄色の部品）を精密ドライバーで優しく回すと LED の明るさを変えられます。それでは LED モジュールを D2 に配線して使ってみましょう（図 5.1-2）。配線するときはマイコンボードにつながった**USB ケーブルを PC から抜いて**おきます。また Grove シールドはピンの数が多いので真っ直ぐ抜き差しします。一度挿したらあまり抜かない方がいいでしょう。抜くときは慎重に真っ直ぐ引っ張らないと足が曲がるので注意します。

プログラムリスト 4.7-1 または 4.7-2 を書き換えて 2 番ピンを初期化して、digitalWrite で HIGH/LOW を切り替えるようにしてみましょう。書き換えたプログラムはプログラムリスト 5.1-1 または 5.1-2 になります。プログラムを書き込んでみましょう。

図 5.1-1 Grove LED モジュール

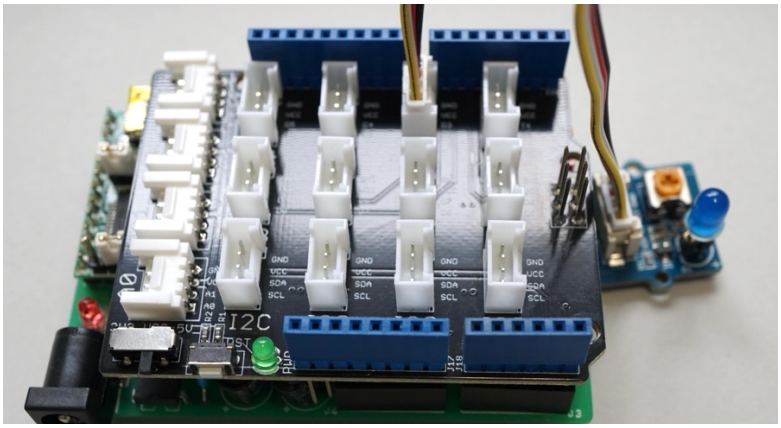


図 5.1-2 Grove LED モジュールを Grove シールドの D2 に挿した様子

プログラムリスト 5.1-1 ブレッドボード上の LED を点滅するプログラム(1)

```
void setup() {
        pinMode(2, OUTPUT);
}

void loop() {
        digitalWrite(2, HIGH);
        delay(1000);
        digitalWrite(2, LOW);
        delay(1000);
}
```

プログラムリスト 5.1-2 ブレッドボード上の LED を点滅するプログラム(2)

```
const int led = 2;
void setup() {
        pinMode(led, OUTPUT);
}
void loop() {
        digitalWrite(led, HIGH);
        delay(1000);
        digitalWrite(led, LOW);
        delay(1000);
}
```

練習問題 5.1

1. シールド上のコネクタを D3～D8 のどれかに変え、プログラムも変更して回路とプログラムの対応関係を理解したか確認しましょう。
2. D2 に LED モジュールを繋いで次のプログラムを動かすと「短い間隔で 2 回 LED を光らせたら、1.1 秒休む」をずっと繰り返します。では三三七拍子にするには、どのように変えたらいいでしょうか？プログラムを書いてみましょう。

```
const int led = 2;
void setup() {
        pinMode(led, OUTPUT);
}
void loop() {
        digitalWrite(led, HIGH);
        delay(100);
        digitalWrite(led, LOW);
        delay(100);
        digitalWrite(led, HIGH);
        delay(100);
        digitalWrite(led, LOW);
        delay(100);
        delay(1000);
}
```

5.2 LED の数を増やす

LED モジュールを追加すると LED の数を増やせます。LED モジュールを D2, D3, D4 につないでみましょう。プログラムにはプログラムリスト 5.2-1, 5.2-2 のように増やした LED の文の命令文（pinMode の文と digitalWrite の文）を追加します。プログラムリスト 5.2-1 と 5.2-2 は増やした LED を順番に点灯する例です。プログラムリスト 5.2-1 では LED のピンの番号を表す変数名を led1 から led にしていて、プログラムリスト 5.2-2 では white, green, red としています。このように自分にとって分かりやすい変数名をつけて構いません。

プログラムリスト 5.2-3 は LED を D2 から D6 に 5 つつないだ例です。setup 関数に見慣れない for 文がありますが、これは中括弧{}の中を i を 2 から 6 まで 1 ずつ増やしながら繰り返すということを書いています。2 ピンから 6 ピンまでなので 2 と 6 です。2 から 8 までにするなら、2 と 8 にすれば OK です。

練習問題 5.2

1. プログラムのフローチャートを書いてみよう。
2. 点灯する LED がだんだん増えていくようにするにはどうしたらいいでしょうか。考えてプログラムを書いてみましょう。
3. LED の数を 7 個に増やしてみましょう。

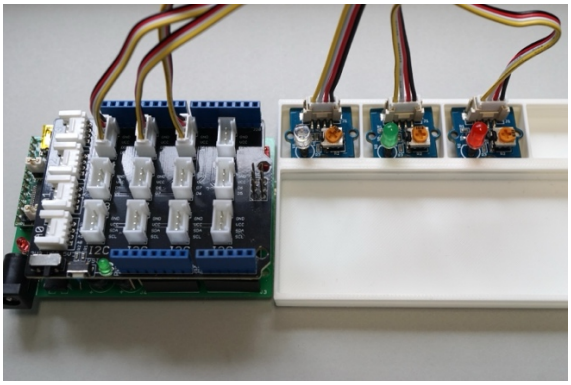


図 5.2-1　D2, D3, D4 コネクタに LED モジュールを繋いだ様子

プログラムリスト 5.2-1　3 個の LED を順番に点滅させる

```
int led1 = 2;
int led2 = 3;
int led3 = 4;
void setup() {
	pinMode(led1, OUTPUT);
	pinMode(led2, OUTPUT);
	pinMode(led3, OUTPUT);
}
void loop() {
	digitalWrite(led1, HIGH);
	delay(3000);
	digitalWrite(led1, LOW);
	digitalWrite(led2, HIGH);
	delay(100);
	digitalWrite(led2, LOW);
	digitalWrite(led3, HIGH);
	delay(3000);
	digitalWrite(led3, LOW);
}
```

プログラムリスト 5.2-2　3 個の LED を順番に点滅させる

```
int white = 2;
int green = 3;
int red = 4;
void setup() {
	pinMode(white, OUTPUT);
	pinMode(green, OUTPUT);
	pinMode(red, OUTPUT);
}
void loop() {
	digitalWrite(white, HIGH);
	delay(3000);
	digitalWrite(white, LOW);
	digitalWrite(green, HIGH);
	delay(100);
	digitalWrite(green, LOW);
	digitalWrite(red, HIGH);
	delay(3000);
	digitalWrite(red, LOW);
}
```

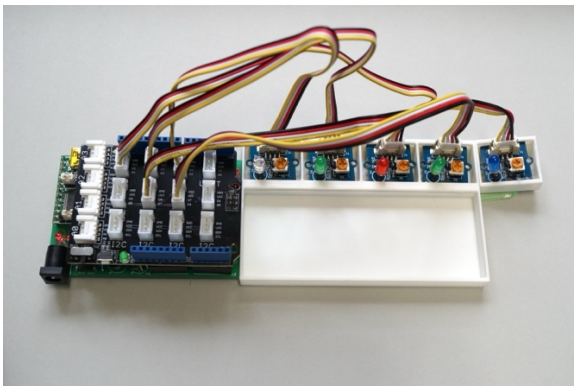


図 5.2-2 D2 から D6 コネクタに LED モジュールを繋いだ様子

プログラムリスト 5.2-3　5 個の LED を順番に点滅させる

```
void setup() {
        for (int i=2; i<=6; i++) {
                pinMode(i, OUTPUT);
}
}
void loop() {
        digitalWrite(2, HIGH);
        delay(500);
        digitalWrite(2, LOW);
        digitalWrite(3, HIGH);
        delay(500);
        digitalWrite(3, LOW);
        digitalWrite(4, HIGH);
        delay(500);
        digitalWrite(4, LOW);
        digitalWrite(5, HIGH);
        delay(500);
        digitalWrite(5, LOW);
        digitalWrite(6, HIGH);
        delay(500);
        digitalWrite(6, LOW);
}
```

同じことを繰り返すときには、for 文を使うとプログラムを並べる必要がありません。プログラムリスト 5.3-1 は、マイコンボード上の LED を 5 回点滅することを繰り返します。フローチャートは図 5.3-1 です。for 文の次の { から } までを繰り返していることが分かります。また変数 i を使って数を数えていて、最初は 0 で 1 ずつ足して、i の値が 5 を超えない間繰り返しているので 5 回繰り返します。i は繰り返す回数を数えているので**カウンタ変数**と呼びます。また for 文の中に書かれている、i=0, i<5, i++ がフローチャートで 3 カ所に分かれているのが分かります。

ひとまずは for (int i=0; i<繰り返す回数; i++) { } を憶えましょう。そして{}の中に繰り返したい文を書くとよいでしょう。

プログラムリスト 5.3-1　LED を 5 回点滅することを繰り返すプログラム

```
int led = 13;

void setup() {
pinMode(led, OUTPUT);
}

void loop() {
for (int i=0; i<5; i++) {
        digitalWrite(led, HIGH);
        delay(100);
        digitalWrite(led, LOW);
        delay(100);
}
delay(3000);
}
```

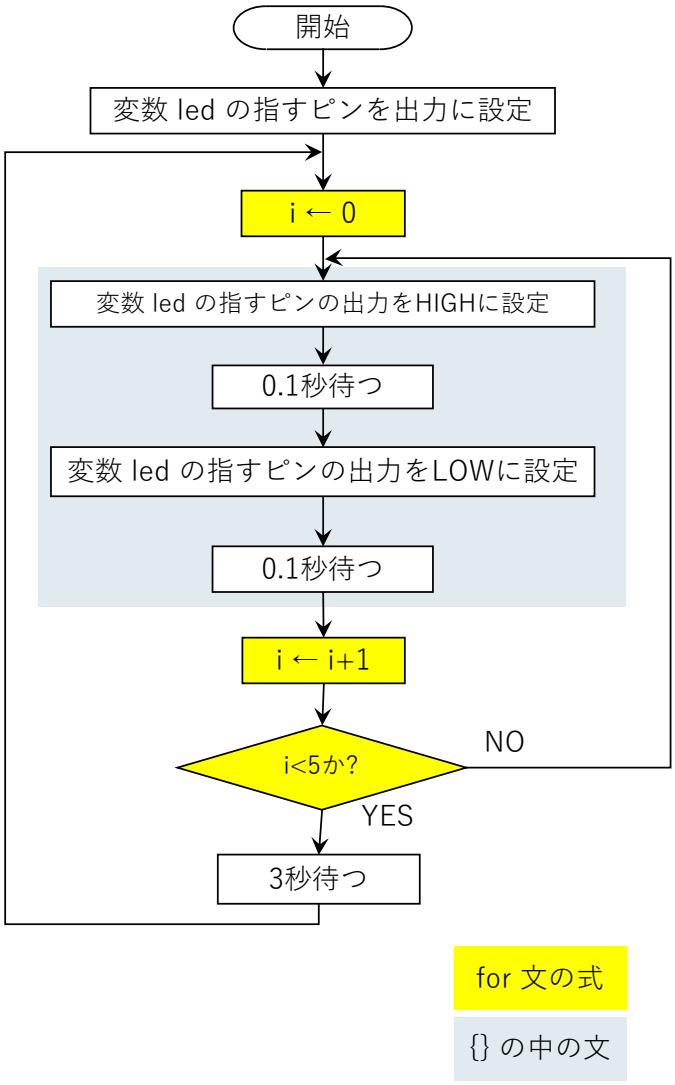


図 5.3-1 プログラムリスト 13 のフローチャート

練習問題 5.3

1. 練習問題 5.1 の 2 の 337 拍子のプログラムを for 文をうまく使って短く書き直してみましょう。
2. プログラムリスト 5.3-1 ではカウンタ変数 i をうまく使っています。プログラムを読んでみましょう。
3. 図 5.2-2 の回路で全ての LED を同時に点灯、消灯して点滅するプログラムを for 文とカウンタ変数をうまく使って書いてみましょう。

5.4 ブザーモジュール

ブザーモジュール(図 5.4-1)はブザーが 1 個載っています。D2～D8 の端子あるいは A0～A3 の端子に Grove ケーブルでつないで使います。ブザーは電源につなぐとブー、あるいはピーという音が鳴る部品です。音の高さや強さは変えられませんが使い方が簡単です。それではブザーモジュールを D2 に配線して使ってみましょう（図 5.4-2）。プログラムは LED モジュールの時と同じプログラムリスト 6 または 7 が使えます。プログラムを書き込んでみましょう。digitalWrite(2, HIGH);のところで LED が点灯する代わりに音が鳴ることがわかります。

図 5.4-1 Grove ブザーモジュール

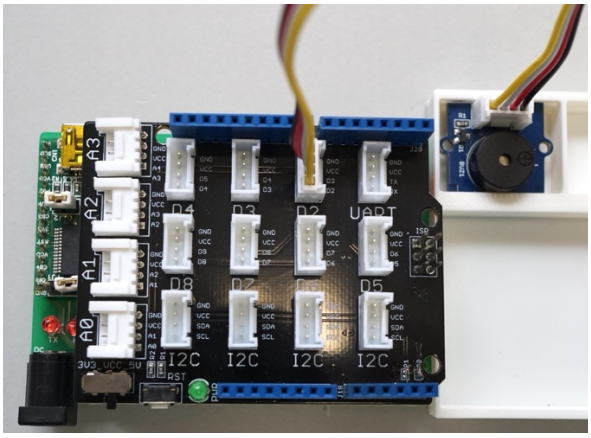


図 5.4-2 Grove ブザーモジュールを Grove シールドの D2 に挿した様子

練習問題 5.4

1. 練習問題 5.1 の 2 または練習問題 5.3 の 1 の 337 拍子のプログラムを動かしてみましょう。
2. LED モジュールとブザーモジュールの両方を使って LED が光るときに音が出るようにした 337 拍子のプログラムを作ってみましょう。

ボタンモジュール(図 5.5-1)はスイッチが 1 個載っています。D2～D8 の端子あるいは A0～A3 の端子に Grove ケーブルでつないで使います。スイッチを押すと挿した端子の信号の電圧が高く(HIGH に)なる回路が入っています。信号の電圧をプログラムから知るにはデジタル入力を使います。デジタル入力ではピンの電圧が高い(HIGH または 1 で表す)か低い(LOW または 0 で表す)を知ることができます。デジタル入力を使うには digitalRead 関数を使います。

```
digitalRead(<入力ピンの番号>)
```

とすると、ピンの番号の電圧が電源電圧である 5V の半分(2.5V)より高いと HIGH(1)が、低いと LOW（0）が戻ります。

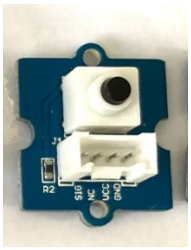


図 5.5-1 Grove ボタンモジュール

D2 に LED モジュールを D3 にボタンモジュールをつないで試してみましょう（図 5.5-2）。digitalRead と if 文を組み合わせて 3 番ピンの電圧を知って電圧が低いときに LED を点灯するプログラムはプログラムリスト 5.5-1 になります。if 文は条件が成立すると最初の中括弧{}の中の命令を実行し、不成立の場合は else の後の{}の中を実行します。動作をフローチャートで表すと図 5.5-3 になります。プログラムを書き込んでスイッチを押すと LED が点灯/消灯する様子を確認してみましょう。

ところで

```
digitalWrite(2, LOW);
```

の命令文は不要だと思いませんか。不要と思ったら消して書き込んでみましょう。フローチャートは図 5.5-4 に変わります。スイッチを一度押すと LED が点灯したままになったのではないでしょうか。これは digitalWrite の結果が保持されるので LED を消す文がないといつまで経っても点灯したままになるためです。

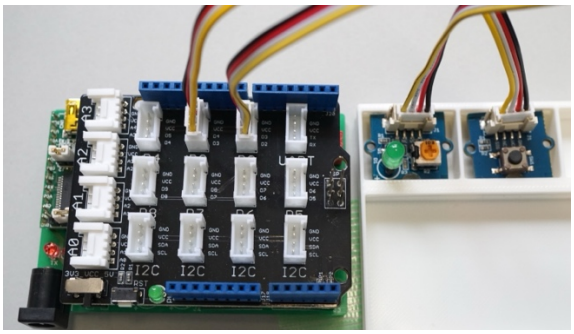


図 5.5-2 D2 に LED モジュール、D3 にボタンモジュールを繋いだ様子

プログラムリスト 5.5-1 入力で LED が点滅するプログラム

```
void setup() {
        pinMode(2, OUTPUT);
}
void loop() {
        if (digitalRead(3) == LOW) {
                digitalWrite(2, HIGH);
        } else {
                digitalWrite(2, LOW);
        }
}
```

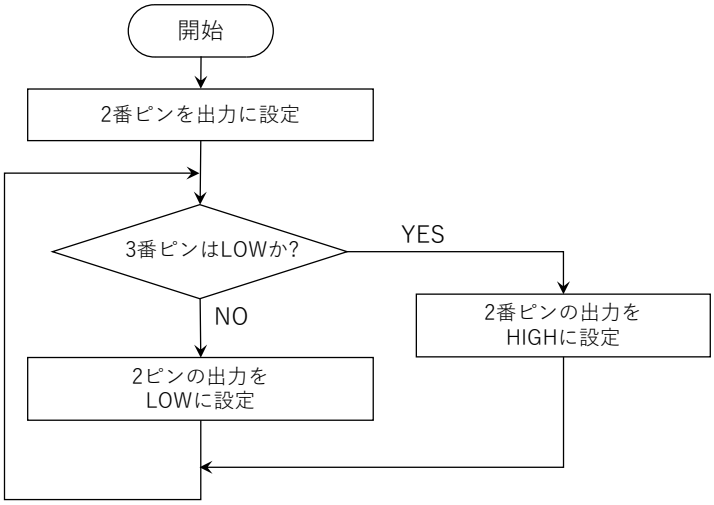


図 5.5-3 プログラムリスト 5.5-1 入力で LED が点滅するプログラムのフローチャート

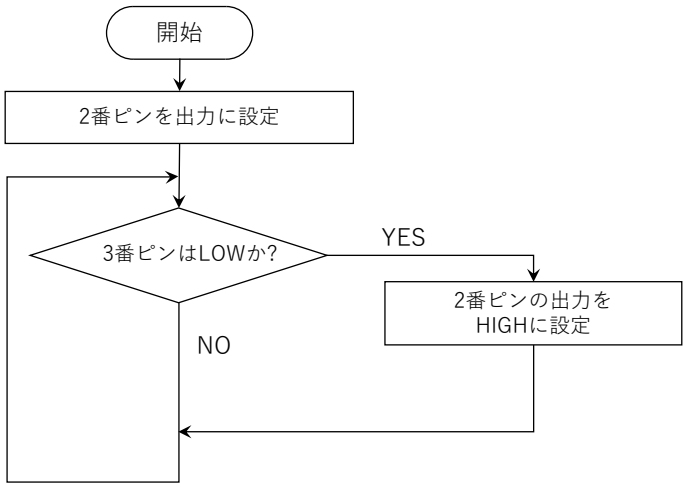


図 5.5-4 プログラムリスト 5.5-1 から digitalWrite(2, LOW);を消したときのフローチャート

練習問題 5.5

1. スイッチを押すとブザーから音が鳴るようにするにはどうしたらいいでしょうか？わかったら実際に作ってみましょう。
2. スイッチを押すと LED が光り、ブザーから音が鳴るようにするにはどうしたらいいでしょうか？わかったら実際に作ってみましょう。
3. スイッチを押していないときは、LED が点滅し、押すと音が鳴るようにするにはどうしたらいいでしょうか？わかったら実際に作ってみましょう。
4. スイッチを D3 ではなく D5 につなぐときは配線とプログラムをどう変えたらいいでしょうか？わかったら実際に作ってみましょう。

概要と準備

Grove I2C 液晶モジュール（図 5.6-1）は英数字とカタカナが横 16 文字×縦 2 文字表示できるモジュールです。このモジュールを使うときには図 5.6-2 のように I2C コネクタに繋ぎます。このモジュールを使うには Arduino IDE にライブラリの追加が必要です。まず https://n.mtng.org/ele/arduino/i2c.html で配付しているライブラリ（執筆時は I2CLiquidCrystal-1.6.1.zip が最新)をダウンロードしてください。ダウンロードができたら Arduino IDE のメニューの「スケッチ」→「ライブラリをインクルード」→「.ZIP 形式のライブラリをインストール...」をクリックし、開いたダイアログでダウンロードしたファイルを指定します(図 5.6-3)。

図 5.6-1 Grove I2C 液晶モジュール

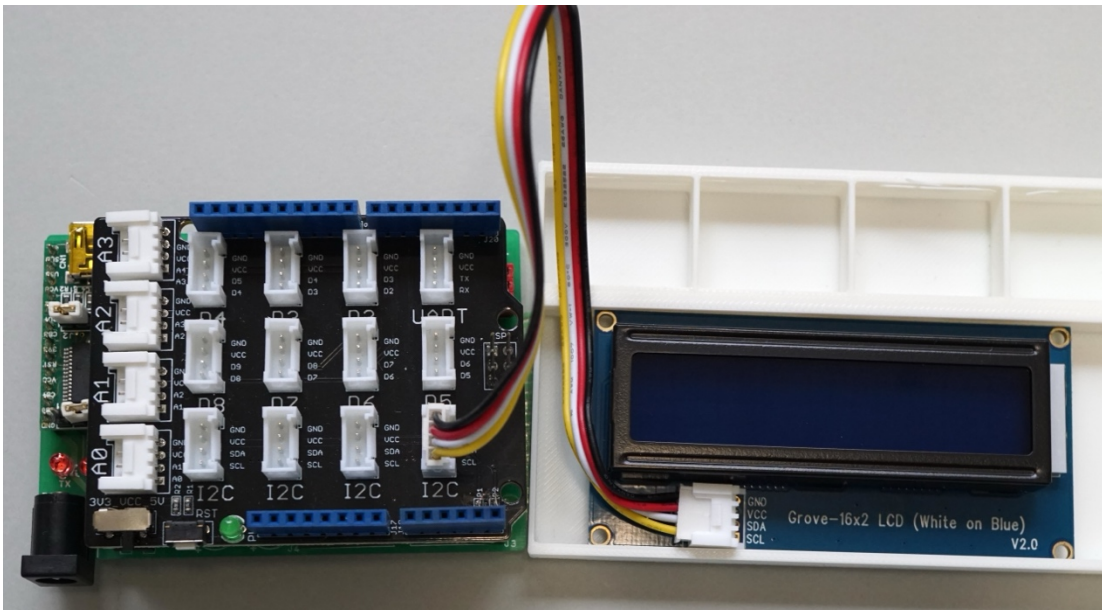


図 5.6-2 Grove I2C 液晶モジュールを I2C 端子に繋いだ様子

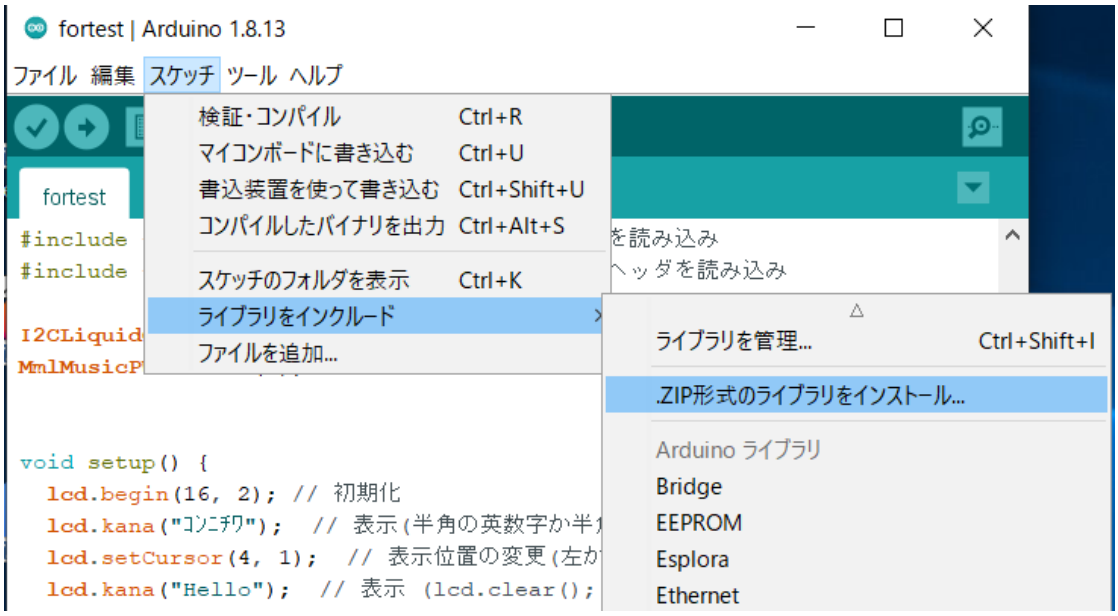


図 5.6-3 ライブラリの追加方法

液晶モジュールを使ってみる

プログラムリスト 5.6-1 を入力して書き込んでみましょう。液晶に「ｺﾝﾆﾁﾜ」と「Hello」が表示されます(図 5.6-4)。うまくいったら表示を変えて見ましょう。英数字、カタカナ共に半角で入力します。Windows IME の場合半角カタカナは、平仮名で入力した後に F7 キーを押して F8 を押すと半角カナになります。

プログラムリスト 5.6-1　Grove I2C 液晶モジュールを使うプログラム例

```
#include <I2CLiquidCrystal.h>          // ライブラリのヘッダを読み込み
I2CLiquidCrystal lcd(LCD_GROVE_RGB);  // 宣言と初期化

void setup() {
  lcd.begin(16, 2);     // 初期化
  lcd.kana("ｺﾝﾆﾁﾜ");    // 表示(半角の英数字か半角のカタカナ)
  lcd.setCursor(4, 1);  // 表示位置の変更(左から 4 文字目上から 1 行目)
  lcd.kana("Hello");    // 表示 (lcd.clear(); と書くと表示が消える)
}

void loop() {
}
```

図 5.6-4 プログラムリスト 5.6-1 を実行したときの表示

プログラムの最初の 2 行はライブラリを使うための準備です。Grove I2C 液晶を使う場合にはプログラムの先頭に書いておきます(setup 関数より前)。setup 関数の最初の lcd.begin が液晶モジュールを使うための準備です。setup 関数の中で液晶を使う命令文の前に書きます。lcd.kana 関数が文字を表示するための関数です。決まった（プログラムで変更しない）文字列を表示するときには"(ダブルクオーテーション)で文字をくくって書きます。lcd.setCursor は文字を書く位置を変更する関数です。ほかに以下のような関数（書き方）があります。

```
lcd.print("Hello");     // 英数字のみ表示できる以外は lcd.kana と同じ
lcd.print(<式>);                 // 式の値を表示する
lcd.setCursor(x, y);   // カーソル（次に文字を書き始める位置）を x, y に
                                 // する。x の範囲は 0～15, y の範囲は 0～1。
lcd.clear();           // 表示を消す。カーソルは 0, 0 (左上)になる。
lcd.home();                      // カーソルは 0, 0 (左上)にする。
```

表示の一部を書き換えるときには lcd.setCusor を使って同じ位置に文字を書くとよいです。後から書く文字の文字数が前に書いた文字数より少ないことがある場合は後に空白文字(" ")を書くとよいです。15 からカウントダウン表示するプログラムの例をプログラムリスト 5.6-2 に示します。動作の様子は図 5.6-5 です。

図 5.6-5 15 からカウントダウンするプログラムの表示。左下の数字が変わる。

練習問題 5.6

1. プログラムリスト 5.6-2 の lcd.print(" "); をコメントアウトするか消して表示がどう変わるか確かめてください。

プログラムリスト 5.6-2　15 からカウントダウン表示するプログラム

```
#include <I2CLiquidCrystal.h>          // ライブラリのヘッダを読み込み
I2CLiquidCrystal lcd(LCD_GROVE_RGB);  // 宣言と初期化

void setup() {
  lcd.begin(16, 2);     // 初期化
  lcd.kana("ｺﾝﾆﾁﾜ");    // 表示（半角の英数字か半角のカタカナ）
  lcd.setCursor(4, 1);  // 表示位置の変更（左から 4 文字目上から 1 行目）
  lcd.kana("Hello");    // 表示 （lcd.clear(); と書くと表示が消える）
}

void loop() {
  for (int i = 15; i >= 0; i--) {
    lcd.setCursor(0, 1);
    lcd.print(i);
    lcd.print(" ");
    delay(1000);
  }
  delay(3000);
}
```

ブザーモジュールでは一つの音しか出せません。圧電スピーカーを使うと高さの違う音を出すことができます。モジュールとしては用意されていないのでブレッドボードを使うか、図 5.7-1,5.7-2 のようにピンソケットに挿して使います。あるいは Grove ケーブルに図 5.7-3 のように挿して、それをベースシールドに挿してもよいでしょう。プログラムと説明は 6.6, 6.7 を見て下さい。6.6 では 8 番ピンに圧電スピーカをつないで説明しています。それ以外のピンに挿したときには読み替えてください。

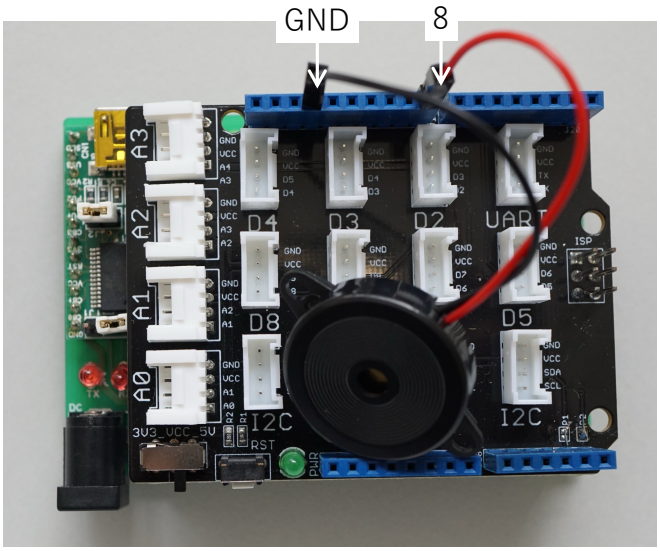


図 5.7-1 圧電スピーカーの繋ぎ方の例。電線が出ている圧電スピーカを 8 番ピンにつなぐ場合。GND にも忘れずつなぐ。

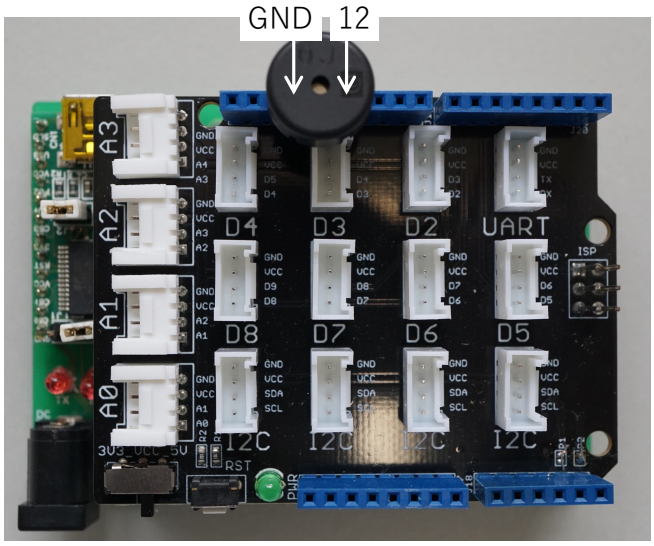


図 5.7-2 圧電スピーカーの繋ぎ方の例。端子が硬い圧電スピーカーの場合。足を無理に曲げないように GND と 12 番ピンに挿している。足の間隔が違う場合は部品に合わせる。

図 5.7-3 圧電スピーカを Grove ケーブルに挿しておくとモジュールのように簡単につなぎ替えられる。左：全体。右：挿し方。

乱数については 6.9 を見て下さい。

マイコンボードに LED と圧電スピーカーをつないで LED の数を増やしたり音を出してみましょう。試しに回路を作るにはブレッドボードが便利です[1]。ブレッドボード（図 6-1）は電子回路の実験によくつかわれるボードです。図 6-2 のようにブレッドボードの内部はつながっています。つなぎたい部品の足や導線（ジャンプワイヤー）をつながっている穴に差し込むことではんだ付けなしに接続できます。上下の+と-のところは横に長くつながっていて電源につないでおくと便利です。上下の+と-はつながっていないので、ジャンプワイヤーでつなぎます。ジャンプワイヤーというのはブレッドボードに使いやすいようになっている導線のことです（図 6-3）。柔らかいタイプと硬い（単線）のものが市販されています。柔らかい方は抜き差しがしやすく、硬い方はブレッドボード上にぴったりつけると配線がすっきり見えます。太さが AWG26（外径 0.4mm）の単心の電線や直径 0.6mm 程度のスズメッキ線を使って硬いジャンプワイヤーを作ることもできます。自分で作ると安価なのと長さが自由になります。

ブレッドボードを使うときに部品の足は長いままでも短くしても構いません。LED のように極性のある部品の足を短く切るときには、もともとのように長さを変えておくといいでしょう（LED の場合はアノード側が長い）。LED やタクトスイッチ、圧電スピーカーの足は**曲げずに**使う方がよいでしょう。部品の足を曲げるときには部品の根元に力がかからないように注意します。根元に力がかかると壊れることがあります。

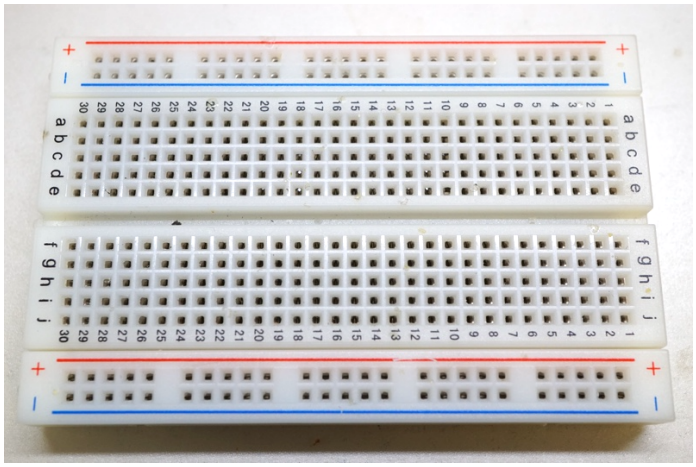


図 6-1 ブレッドボードの例（EIC-801 (E-CALL ENTERPRISE CO., LTD.製)）

[1] TinkerCAD (https://www.tinkercad.com/)で「回路」のシミュレーションをすると 6 章の内容を試せます。沖縄高専の先生の作られた資料: https://ic.okinawa-ct.ac.jp/advanced/tinker-iframe/document-for-tinkercad.pdf

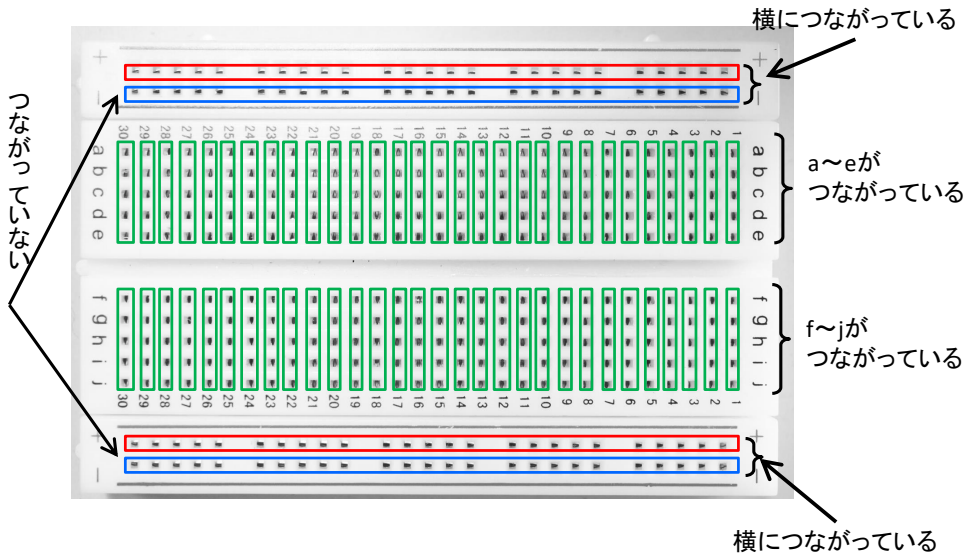


図 6-2 ブレッドボード EIC-801 は図のように内部で繋がっている

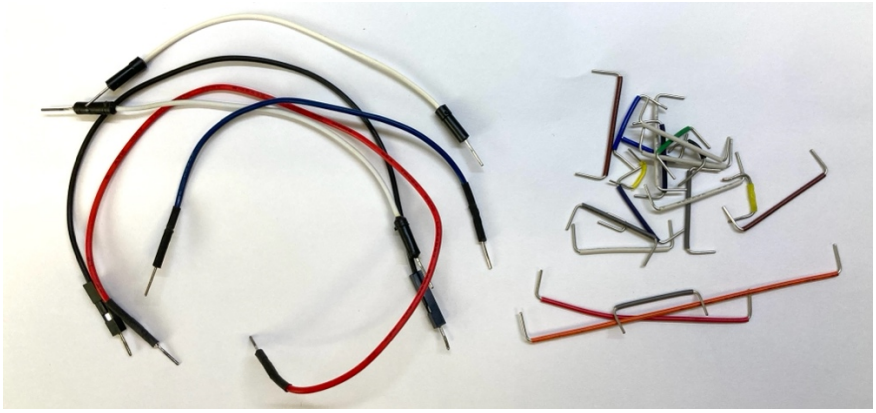


図 6-3 柔らかいジャンプワイヤー（左）と硬いジャンプワイヤー（右）

6.1 LED を点灯させる回路

LED は図 6.1-1 のような部品です。図 6.1-1 は砲弾型 LED とよばれるものです。図のように購入時は片側の足が長くなっています。足の長い方がアノード、短い方がカソードとよばれます。足が長すぎて切るときにも長さの差があるように切るといいでしょう。LED はアノード側をプラスに、カソード側をマイナスにして順方向電圧以上の電圧をかけると電流が流れて点灯します。ただ電圧が順方向電圧よりも高いと電流が流れすぎて壊れるので、抵抗を使って電流が流れすぎるのを防ぎます（このために使う抵抗のことを電流制限抵抗とよぶ）。これを回路図で表すと図 6.1-2 になります。DOUT はマイコンボードのデジタル出力へつなぎます。具体的なピンの番号が決まれば DOUT ではなく、D2, D3 などと表記します。GND はグラウンド（マイナス）のことです。抵抗値は 300Ω～10kΩぐらいがよいでしょう。LED の明るさは抵抗値が低いと明るく、高いと暗くなります。また抵抗値はカラーコードで表します。1kΩの抵抗の場合は図 6.1-3 のように茶黒赤金です。

図 6.1-1 砲弾型 LED の例。〇で囲った片側の足（アノード側）が長い。

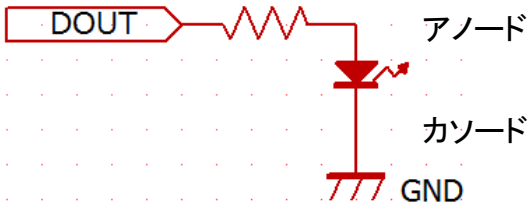


図 6.1-2 LED を点灯させる回路。DOUT はマイコンのデジタル出力へつなぐ。

図 6.1-3 1kΩの抵抗のカラーコード。茶黒赤金の順で並んでいる。

6.2 ブレッドボード上の LED を点滅する

図 6.1-2 の回路を実際に作ってみましょう。配線するときはマイコンボードにつながった **USB ケーブルを PC から抜いて**おきます。ブレッドボードとジャンプワイヤ 2 本、抵抗(1kΩ程度、300Ω～4.7kΩ程度ならよい)、LED（色は何でもよい）を用意して図 6.2-1 のように配線します。ブレッドボードの穴は縦 5 つがつながっています（図 6-2）。図 6.2-1 の通りの位置に配線する必要はありませんが、接続する LED の足とジャンプワイヤ、LED の足と抵抗の足、抵抗の足とジャンプワイヤをそれぞれ同じ縦 5 つのところに挿します。また LED には向きがあります。2 番ピン側（抵抗側）に LED の足の長い側を挿します。プログラムリスト 4.7-1 または 4.7-2 を書き換えて 2 番ピンを初期化して、digitalWrite で HIGH/LOW を切り替えるようにしてみましょう。書き換えたプログラムはプログラムリスト 6.2-1 または 6.2-2 になります。

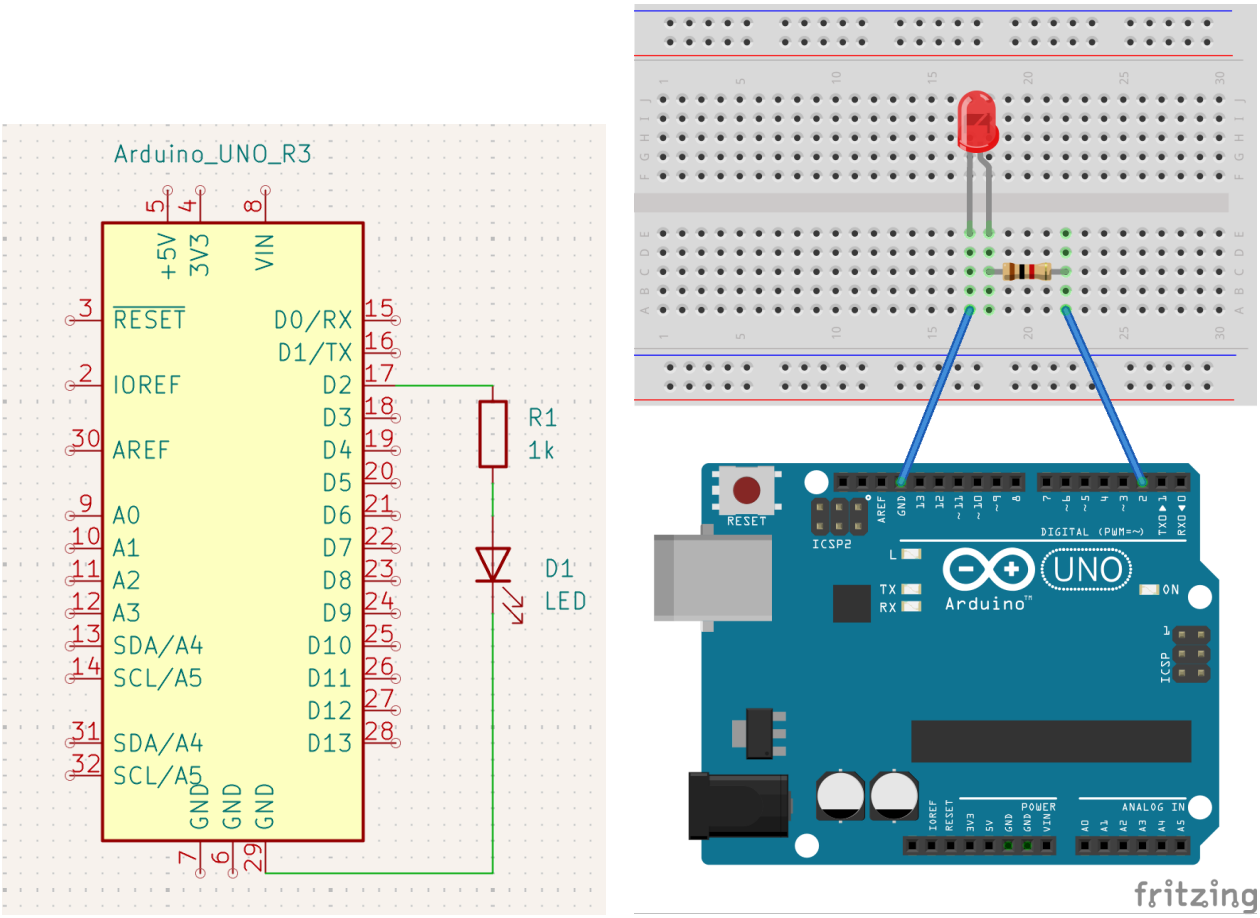


図 6.2-1 マイコンの 2 番ピンに LED をつなぐ回路（左：回路図、右：実体配線図）

プログラムリスト 6.2-1 ブレッドボード上の LED を点滅するプログラム(1)

```
void setup() {
        pinMode(2, OUTPUT);
}

void loop() {
        digitalWrite(2, HIGH);
        delay(1000);
        digitalWrite(2, LOW);
        delay(1000);
}
```

プログラムリスト 6.2-2 ブレッドボード上の LED を点滅するプログラム(2)

```
const int led = 2;
void setup() {
        pinMode(led, OUTPUT);
}
void loop() {
        digitalWrite(led, HIGH);
        delay(1000);
        digitalWrite(led, LOW);
        delay(1000);
}
```

練習問題 6.2

1. 図 6.2-1 のデジタルピンに挿しているジャンプワイヤーの端を 5V の端子、GND の端子に順に挿してみましょう。
2. ブレッドボード上の LED の位置を変えて配線に慣れましょう。
3. 配線とプログラムのデジタルピンを 3～13 の間で変えて見ましょう。
4. 図 6.2-1 の回路で次のプログラムを動かすと「短い間隔で 2 回 LED を光らせたら、1.1 秒休む」をずっと繰り返します。では三三七拍子にするには、どのように変えたらいいでしょうか？プログラムを書いてみましょう。

```
const int led = 2;
void setup() {
        pinMode(led, OUTPUT);
}
void loop() {
        digitalWrite(led, HIGH);
        delay(100);
        digitalWrite(led, LOW);
        delay(100);
        digitalWrite(led, HIGH);
        delay(100);
        digitalWrite(led, LOW);
        delay(100);
        delay(1000);
}
```

6.3 マイコンボードとブレッドボードの LED を点滅させる

マイコンボード上(13 番ピン)とブレッドボード上(2 番ピン)の LED を点滅させるにはどうしたらいいでしょうか。図 6.2-1 の配線のまま、プログラムを変えてみます。プログラムリスト 6.3-1 をみてください。2 番ピン用と 13 番ピン用の pinMode と digitalWrite の文が並んでいます。フローチャートは図 6.3-1 左になります。 2 つの文は**順番に実行**されますが、**プログラムは十分に速く動く**ので LED は**同時に点滅**して見えます。細かなタイミングを気にする必要がなければ、2 番と 13 番のどちらの文が先にあっても大丈夫です。

ではプログラムリスト 6.3-2 のプログラム（フローチャートは図 6.3-1 右）はどのような動作をするでしょうか。2 番ピンが HIGH のとき 13 番ピンを LOW に、2 番ピンが LOW のとき 13 番ピンを HIGH にするプログラムです。このプログラムの場合は 2 つの LED が交互に点滅します。

練習問題 6.3

1. マイコンボードの LED が点灯してから 0.5 秒後にブレッドボードの LED が点き、0.5 秒経ったらマイコンボードの LED が消灯し、さらに 0.5 秒経つとブレッドボードの LED が消灯して、両方消えてから 0.5 秒経って最初に戻るにはどうしたらいいか考えプログラムを書いて下さい。

プログラムリスト 6.3-1 マイコンボードとブレッドボードの LED を同時に点滅させる(1)

```
void setup() {
        pinMode(2, OUTPUT);
        pinMode(13, OUTPUT);
}
void loop() {
        digitalWrite(2, HIGH);
        digitalWrite(13, HIGH);
        delay(1000);
        digitalWrite(2, LOW);
        digitalWrite(13, LOW);
        delay(1000);
}
```

プログラムリスト 6.3-2 マイコンボードとブレッドボードの LED を同時に点滅させる(2)

```
void setup() {
        pinMode(2, OUTPUT);
        pinMode(13, OUTPUT);
}
void loop() {
        digitalWrite(2, HIGH);
        digitalWrite(13, LOW);
        delay(1000);
        digitalWrite(2, LOW);
        digitalWrite(13, HIGH);
        delay(1000);
}
```

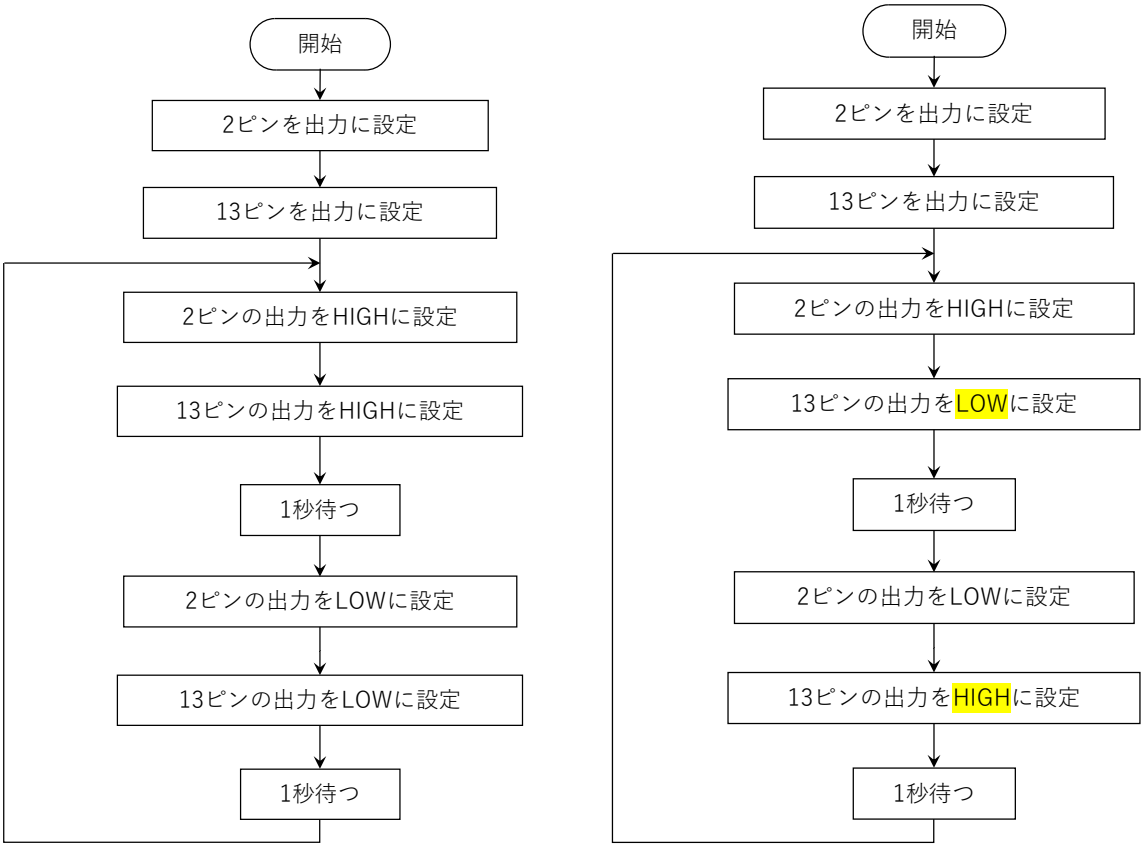


図 6.3-1 プログラムリスト 6.3-1 のフローチャート（左）とプログラムリスト 6.3-2 のフローチャート

6.4 もっと多くの LED を点滅させる

もっと多くの LED を点滅させるにはどうしたらいいでしょうか。LED の数だけ図 6.1-2 の回路を増やし、プログラムにはプログラムリスト 6.4-1, 6.4-2 のように増やした LED の文の命令文（pinMode の文と digitalWrite の文）を追加します。図 6.4-1, 図 6.4-2 は LED を 2 番ピンから 4 番ピンにそれぞれつないだ回路図と配線例です。LED と抵抗が 3 組並んでいます。マイコン側のピンはすべて違う端子に、グラウンドはすべて同じ端子に繋ぎます。グラウンドにはブレッドボードの電源用の端（図の上下）の穴が横にすべてつながっているところを利用します。図 6.4-2 では下のマイナスの列にグラウンド(GND)をつなぎ、そこから LED につないでいます。プログラムリスト 6.4-1 と 6.4-2 は増やした LED を順番に点灯する例です。プログラムリスト 6.4-1 では LED のピンの番号を表す変数名を led1 から led にしていて、プログラムリスト 6.4-2 では green, yellow, red としています。このように自分にとって分かりやすい変数名をつけて構いません。

図 6.4-3 とプログラムリスト 6.4-3 は LED を 2 番ピンから 6 番ピンにそれぞれつないだ回路図・配線例とプログラム例です。LED と抵抗が 5 組並んでいます。setup 関数に見慣れない for 文がありますが、これは中括弧{}の中を i を 2 から 6 まで 1 ずつ増やしながら繰り返すということを書いています。2 ピンから 6 ピンまでなので 2 と 6 です。2 から 8 までにするなら、2 と 8 にすれば OK です。

練習問題 6.4

1. 点灯する LED がだんだん増えていくようにするにはどうしたらいいでしょうか。考えてプログラムを書いてみましょう。
2. LED の数を 7 個に増やしてみてましょう。

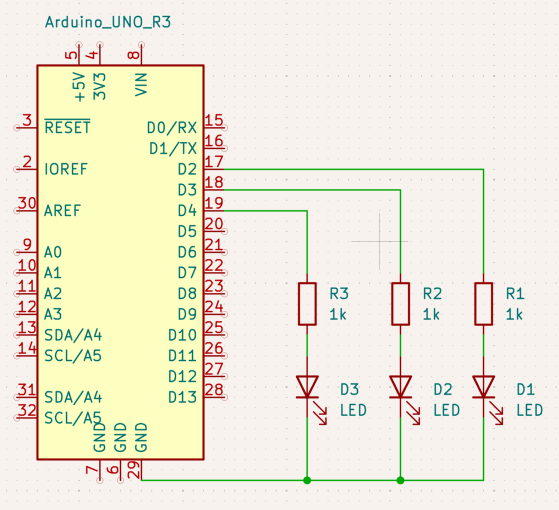


図 6.4-1 マイコンの 2 番ピンから 4 番ピンに LED をつなぐ回路

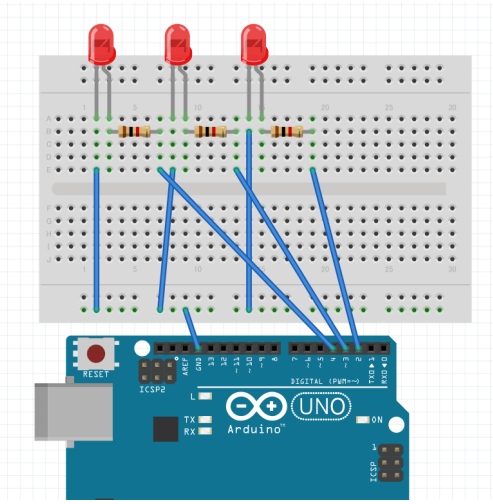


図 6.4-2 マイコンの 2 番ピンから 4 番ピンに LED をつなぐ回路の実体配線図

プログラムリスト 6.4-1　3 個の LED を順番に点滅させる

```
int led1 = 2;
int led2 = 3;
int led3 = 4;
void setup() {
    pinMode(led1, OUTPUT);
    pinMode(led2, OUTPUT);
    pinMode(led3, OUTPUT);
}
void loop() {
    digitalWrite(led1, HIGH);
    delay(3000);
    digitalWrite(led1, LOW);
    digitalWrite(led2, HIGH);
    delay(100);
    digitalWrite(led2, LOW);
    digitalWrite(led3, HIGH);
    delay(3000);
    digitalWrite(led3, LOW);
}
```

プログラムリスト 6.4-2　3 個の LED を順番に点滅させる

```
int green = 2;
int yellow = 3;
int red = 4;
void setup() {
    pinMode(green, OUTPUT);
    pinMode(yellow, OUTPUT);
    pinMode(red, OUTPUT);
}
void loop() {
    digitalWrite(green, HIGH);
    delay(3000);
    digitalWrite(green, LOW);
    digitalWrite(yellow, HIGH);
    delay(100);
    digitalWrite(yellow, LOW);
    digitalWrite(red, HIGH);
    delay(3000);
    digitalWrite(red, LOW);
}
```

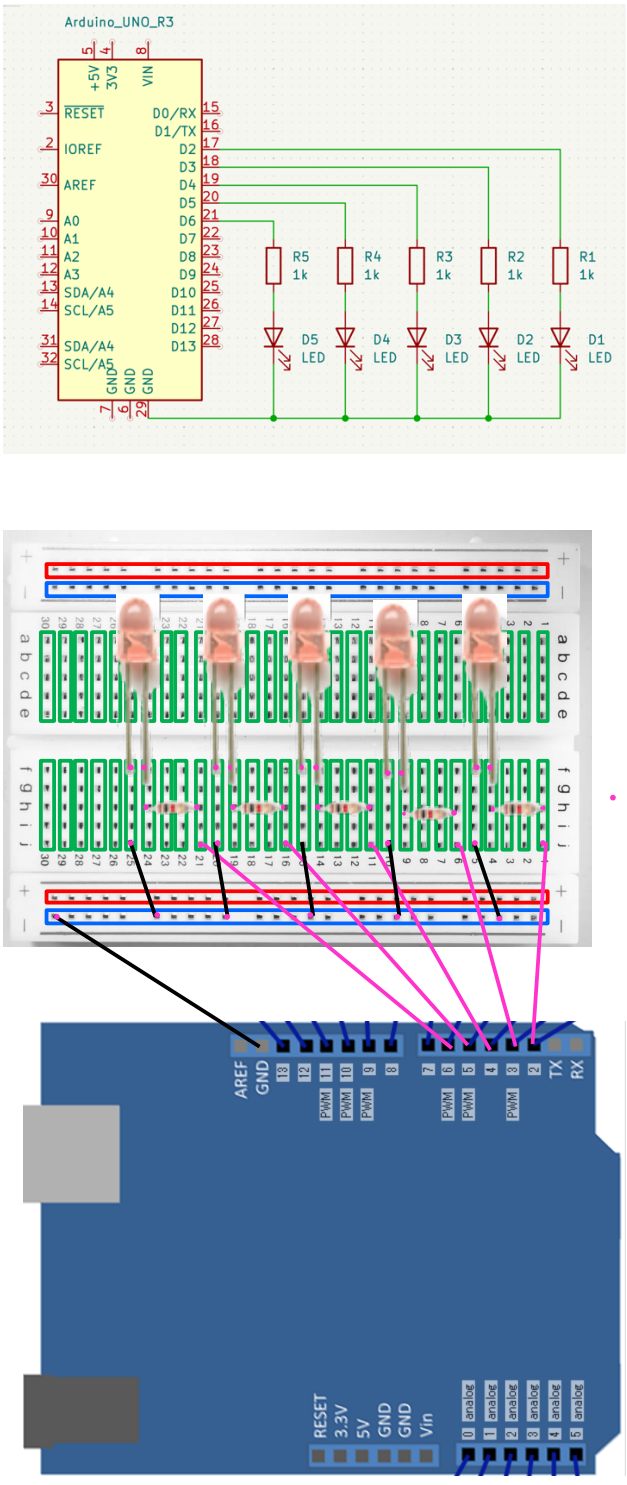


図 6.4-3 マイコンの 2 番ピンから 6 番ピンに LED をつなぐ回路の回路図（上）と実体配線図（下）

プログラムリスト 6.4-3　5 個の LED を順番に点滅させるプログラム

```
void setup() {
        for (int i=2; i<=6; i++) {
                pinMode(i, OUTPUT);
}
}
void loop() {
        digitalWrite(2, HIGH);
        delay(500);
        digitalWrite(2, LOW);
        digitalWrite(3, HIGH);
        delay(500);
        digitalWrite(3, LOW);
        digitalWrite(4, HIGH);
        delay(500);
        digitalWrite(4, LOW);
        digitalWrite(5, HIGH);
        delay(500);
        digitalWrite(5, LOW);
        digitalWrite(6, HIGH);
        delay(500);
        digitalWrite(6, LOW);
}
```

同じことを繰り返すときには、for 文を使うとプログラムを並べる必要がありません。プログラムリスト 6.5-1 は、マイコンボード上の LED を 5 回点滅することを繰り返します。フローチャートを書くと図 6.5-2 となります。for 文の次の { から } までを繰り返していることが分かります。また変数 i を使って数を数えていて、最初は 0 で 1 ずつ足して、i の値が 5 を超えない間繰り返しているので 5 回繰り返します。i は繰り返す回数を数えているので**カウンタ変数**と呼びます。また for 文の中に書かれている、i=0, i<5, i++ がフローチャートで 3 カ所に分かれているのが分かります。

ひとまずは for (int i=0; i<繰り返す回数; i++) { } を憶えましょう。そして{}の中に繰り返したい文を書くとよいでしょう。

プログラムリスト 6.5-1　LED を 5 回点滅することを繰り返すプログラム

```
int led = 13;

void setup() {
pinMode(led, OUTPUT);
}

void loop() {
for (int i=0; i<5; i++) {
        digitalWrite(led, HIGH);
        delay(100);
        digitalWrite(led, LOW);
        delay(100);
}
delay(3000);
}
```

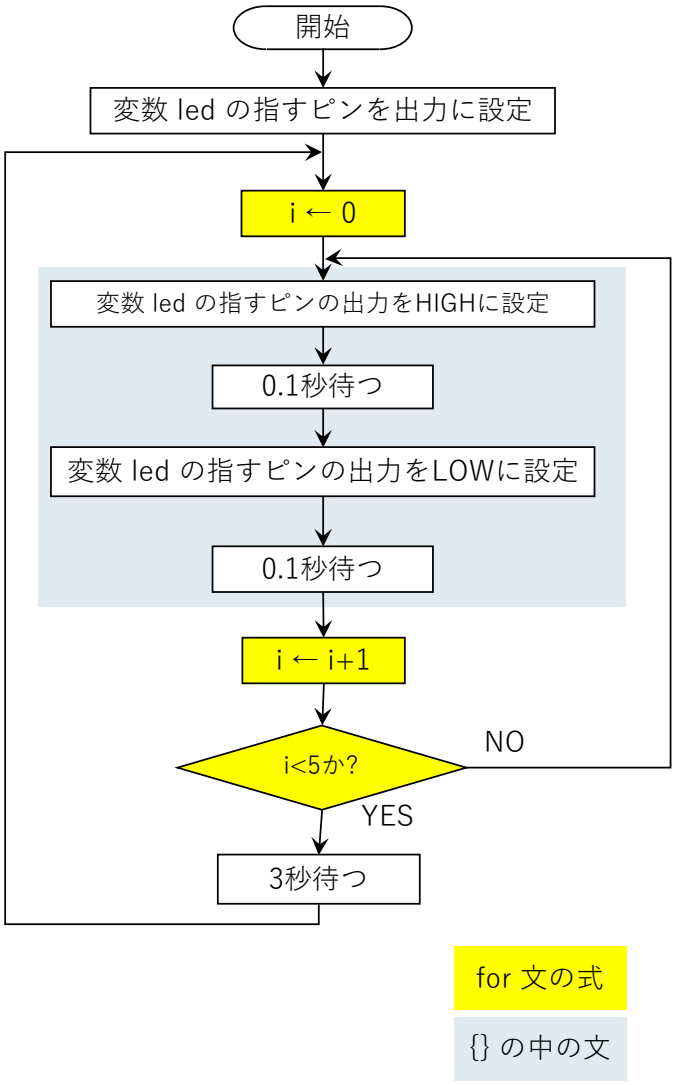


図 6.5-1 プログラムリスト 13 のフローチャート

練習問題 6.5

1. 練習問題 6.2 の 4 の 337 拍子のプログラムを for 文をうまく使って短く書き直してみましょう。
2. プログラムリスト 6.4-3 ではカウンタ変数 i をうまく使っています。プログラムを読んでみましょう。
3. 図 6.4-3 の回路で全ての LED を同時に点灯、消灯して点滅するプログラムを for 文とカウンタ変数をうまく使って書いてみましょう。

圧電スピーカー（圧電素子とよばれることもある）という部品(図 6.6-1)を使うと音を出せます。圧電スピーカーには 2 つ端子があります。端子は電線で出ている場合と硬い足で出ている場合があります。2 本の端子は（ここでは）区別する必要はありません。回路図（図 6.6-2）、実体配線図(図 6.6-3)のように片側をグラウンドに片側をデジタル出力のできるピン（図 6.6-3 では 8 番ピン）につなぎます。プログラムリスト 6.6-1 のように書くと電源が入るかリセットすると 440Hz の音が 0.5 秒(500 ミリ秒)でます。フローチャートは図 23 です。tone で音を出し始めることを命令します。

tone(<音を出すピンの番号>, <音の高さ>);

あるいは

tone(<音を出すピンの番号>, <音の高さ>, <音を出す時間(単位は ms)>);

と書きます。後者の場合は指定した時間だけ音が鳴ります。プログラムリスト 6.6-1 では 8 ピンから 440Hz の音を出し始めます。そして delay で 500 ミリ秒待ち noTone で音を止めます。

noTone(<音を出しているピンの番号>);

と書きます。時間を指定せずに noTone がないと**ずっと音がでたまま**です。

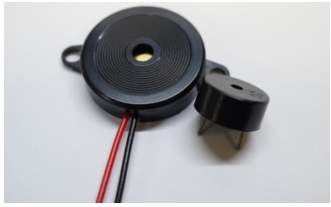


図 6.6-1 圧電スピーカーの例。左は電線が出ているタイプ。右は硬い足が出ているタイプ。

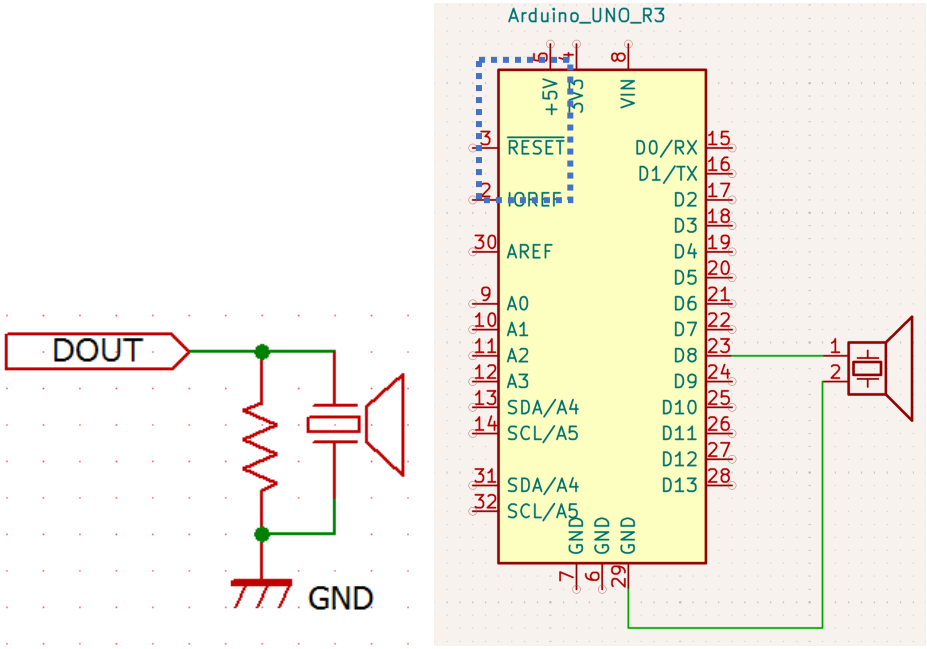


図 6.6-2 圧電スピーカーから音を出す回路。左の図の点線内の抵抗は省いてよい。DOUT と書いているがピン番号が決まれば D8 などと書く。右はマイコンボードを書くときの回路全体（8 番ピンに接続）

また tone 関数は音を出し始めるとすぐに終わって次の命令を実行します。そのため

```
tone(8, 440, 500);
tone(8, 880, 500);
```

と続けて書くと 1 つ目の tone 関数の音はせず、二つ目の音だけが鳴ります。そこで音が鳴り終わってから次の音を出すようにします。たとえば次のように書きます。

```
tone(8, 440, 500);
delay(500);
tone(8, 880, 500);
delay(500);
```

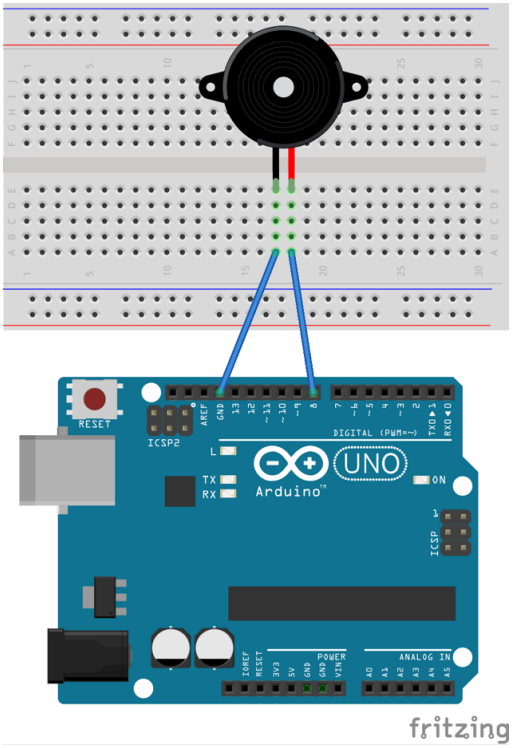
LED などと圧電スピーカを一緒に使うときには、LED と圧電スピーカのピンが同じにならないように決めます。例えば LED を 2 番から 6 番ピンにして圧電スピーカを 8 番ピンにします。圧電スピーカを 2 番ピンにして LED を 3 番から 6 番ピンにしても構いません。tone/noTone はデジタル出力のピンならどれでも使えます。

音階と音の周波数の関係は表 6.6-1 のとおりです。プログラムリスト 6.6-2 のように書くとドレミファソをずっと繰り返し出します。音を止めたいときは圧電スピーカーの配線を抜くか、USB ケーブルを抜いて電源を切ります。休符のように無音を作りたいときは音を出さずに delay で少し待ちます。また音と音の間を少し開けると歯切れがよく聞こえます。

また救急車のピーポーピーポーピーポーという音は周波数 960Hz の音が 0.5 秒(500ms)と周波数 770Hz の音 0.5 秒をずっと繰り返しています。横断歩道のカッコウの音は周波数が 1220Hz の音 200ms, 無音 200ms, 周波数 980Hz の音 600ms で作られています。NHK の時報の音（ぽっぽっぽっぴー）は 440Hz の音 100ms, 無音 900ms を 3 回繰り返して周波数 880Hz の音 1500ms で作られています。

練習問題 6.6

1. 説明にある端切れのよい音になるようプログラムリスト 6.6-2 を書き換えてください。
2. 救急車の音、横断歩道のカッコウの音、NHK の時報の音から一つ選んで音を鳴らしてください。
3. 音がなっているときにマイコンボード上の LED(13 ピン)が点灯するようにしてください。
4. ブレッドボードに LED を追加して、音がなっているときにその LED が点灯するようにしてください。
5. 4 の回路で 337 拍子のタイミングで LED が点灯して音が鳴るプログラムを作ってください。



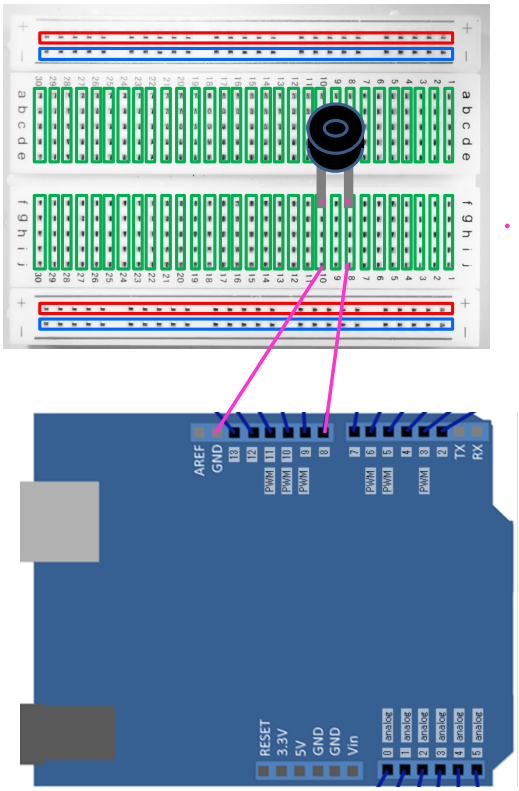


図 6.6-3 圧電スピーカーをマイコンボードにつなぐ
（上：電線の出ている圧電スピーカの場合、下：固い線が出ている圧電スピーカの場合）

プログラムリスト 6.6-1 音を出すプログラム

```
void setup()
{
        tone(8, 440);
        delay(500);
        noTone(8);
}

void loop()
{
}
```

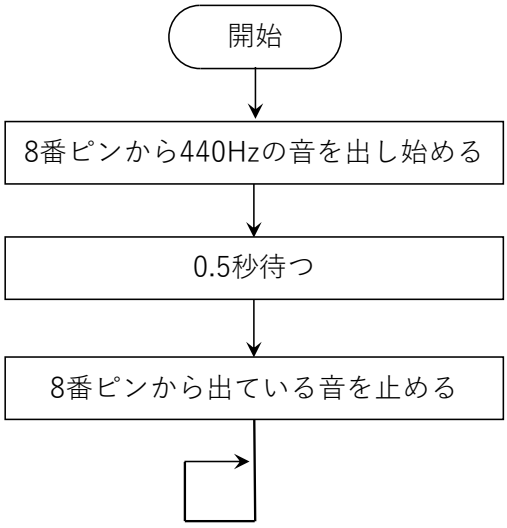


図 6.6-4 プログラムリスト 6.6-1 のフローチャート

表 6.6-1 音階と周波数の関係

音階	周波数[Hz]
ド	523
レ	587
ミ	659
ファ	698
ソ	784
ラ	880
シ	988
ド	1046

プログラムリスト 6.6-2 ドレミファソを繰り返すプログラム

```
void setup()
{
}
void loop()
{
        tone(8, 523);
        delay(500);
        noTone(8);

        tone(8, 587);
        delay(500);
        noTone(8);

        tone(8, 659);
        delay(500);
        noTone(8);

        tone(8, 698);
        delay(500);
        noTone(8);

        tone(8, 784);
        delay(500);
        noTone(8);

                delay(1000);
}
```

概要と準備

tone 関数を使うと圧電スピーカーから音を出すことができます。ただ長い音楽を出すのは少し大変です。MMLMusic ライブラリを使うとより簡単に音楽を出すことができます。まず必要なライブラリ MMLMusic, MMLMusicPWM の二つをダウンロードします。MMLMusic は https://github.com/maxint-rd/MmlMusic を開き、Code のボタンをクリックして開く中の Download ZIP をクリックしてダウンロードします(図 6.7-1)。MMLMusicPWM は https://github.com/maxint-rd/MmlMusicPWM から同様にダウンロードします。ダウンロードができたら Arduino IDE のメニューの「スケッチ」→「ライブラリをインクルード」→「.ZIP 形式のライブラリをインストール...」をクリックし、開いたダイアログでダウンロードしたファイルを指定します(図 6.7-2)。インストールの順は MMLMusic, MMLMusicPWM のどちらが先でも構いません。

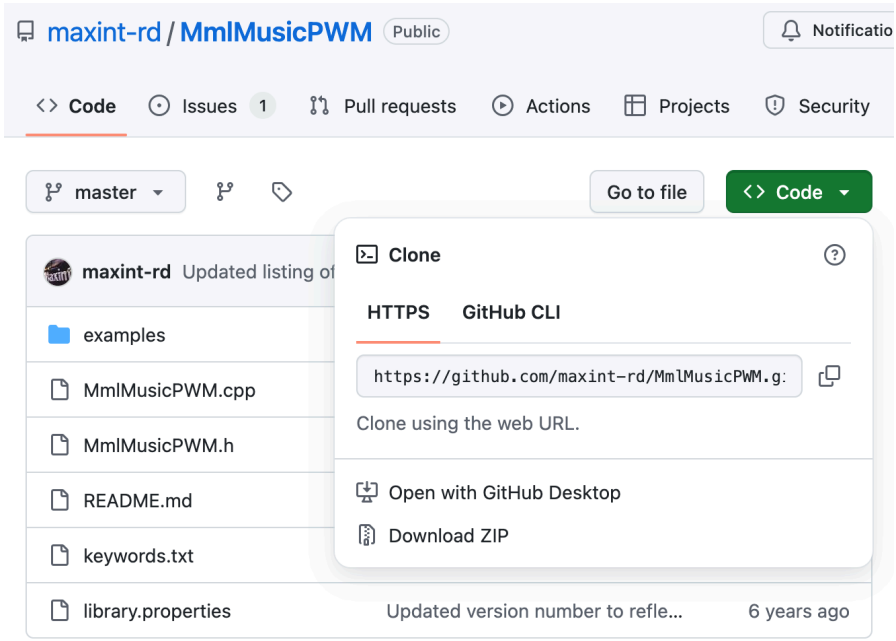


図 6.7-1 MMLMusic ライブラリのダウンロード

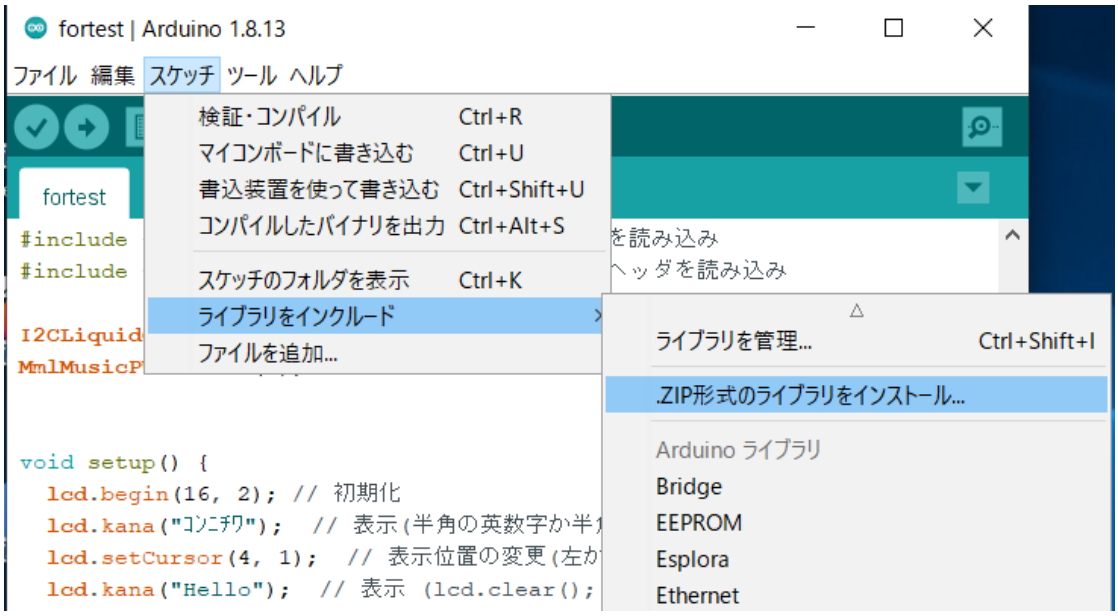


図 6.7-2 ライブラリの追加方法

MMLMusic ライブラリを使ってみる

プログラムリスト 6.7-1 を入力して実行してみましょう。圧電スピーカーは 2 番ピンにつないでいるとしています。別のピンにつないでいるときは 2 をピンの番号に変えてください。//以降はコメントなので省略しても動作します。プログラムを書き込むと「ドレミファソラシド・ドシラソファミレド」と音が流れます。もう 1 度聞きたい場合はリセットボタンを押すと setup 関数の先頭から開始するので再度音が鳴ります。

プログラムリスト 6.7-1 MMLMusic のプログラム例

```
#include <MmlMusicPWM.h> // ライブラリのヘッダを読み込み

MmlMusicPWM music(2);   // 宣言と初期化

void setup() {
  music.play("CDEFGAB>CRC8<B8A8G8F8E8D8C8RR"); // 音を鳴らす
  // A から G がラからソ、>が 1 オクターブ上げる、<が 1 オクターブ下げる
  // R は休符、音階の後ろの数字は音符の長さ(8は 8 分音符)
}

void loop() {
}
```

MML (Music Macro Language)

音楽（楽譜に相当）は MML(Music Macro Language)で書きます。音階は A〜G で表します（英語の音名表記）。ドレミファソラシがそれぞれ CDEFGAB に対応します。R は休符を表します。音階の後の数字を省略すると四分音符、数字を書くとその音符の長さの音が鳴ります。たとえば 8 にすると八分音符になります。>で 1 オクターブ上げることができ、<で 1 オクターブ下がります。T を使うとテンポを指定できます。♪=120 は T120 と書きます。残念ながら MMLMusic では和音は出せません。

MMLMusic ライブラリの動作

music.play()は音が鳴り始めるとすぐに次の命令文が実行されます。BGM のようなイメージです。そのため、音を鳴らしながら LED を点滅させるといったことが簡単にできます。音の鳴り終わりとプログラムの実行のタイミングを合わせたい場合は音が流れる時間をストップウォッチなどで測って、delay 関数でプログラムの動作を遅くするのが一つの方法です。もう一つ鳴り終わったことを確かめる music.isPlaying()関数があります。この関数は音が鳴っているときには true, 鳴っていないときには false となります。プログラムリスト 6.7-1 の loop 関数を

```
void loop() {
  if (music.isPlaying())
    digitalWrite(13, HIGH);
    else
    digitalWrite(13, LOW);
}
```

と書き換えると音が鳴っている間、マイコンボード上の LED が点灯します。あるいは

```
        while (music.isPlaying())
                ;
```

と書くと音が鳴り終わるまでプログラムの動作が止まります。

tone 関数との関係

MMLMusic を使うときには tone 関数が使えません。代わりに music.tone 関数を使います。使い方は tone 関数と同じです。

デジタル入力を使うとマイコンはピンの電圧が高い(HIGH または 1 で表す)か低い(LOW または 0 で表す)を知ることができます。ここではタクトスイッチ（図 6.8-1）を使ってみます。タクトスイッチには 2 本足と 4 本足のものが売られています。4 本足のものは向きがあり（足が出ている方向と出ていない方向がある）、図 6.8-2 のように内部で 2 本ずつ足が繋がっています。ですからブレッドボードに挿すときには向きに注意します。スイッチをマイコンボードに繋ぐときによく使う回路が図 6.8-3(配線例は図 6.8-4)です。スイッチを押していないときに DIN の電圧が高く(5V)なり、スイッチを押すと低く(0V)なります。抵抗の値は 1kΩ～100kΩぐらいをよく使います。10kΩか 1kΩを使うとよいでしょう。

図 6.8-1　2 本足のタクトスイッチ（左）と 4 本足のタクトスイッチ

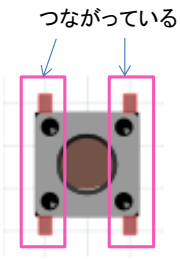


図 6.8-2 4 本足のタクトスイッチは内部で足 2 本ずつが繋がっている。向きを 90 度間違えるとスイッチを押しっぱなしと一緒になるので注意する。

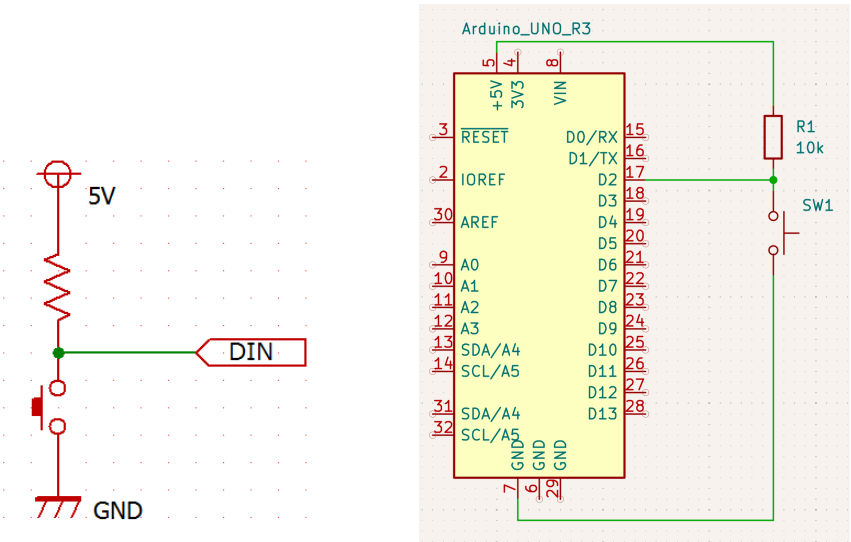


図 6.8-3 スイッチをマイコンボードに繋ぐ回路。左はピン番号を DIN と書いているがピン番号が決まれば D2 などと書く。右はマイコンボードを書くときの回路全体（スイッチは 2 番ピンに接続）

デジタル入力を使うには digitalRead 関数を使います。

digitalRead(<入力ピンの番号>)

とすると、ピンの番号の電圧が電源電圧である 5V の半分(2.5V)より高いと HIGH(1)が、低いと LOW（0）が戻ります。digitalRead と if 文を組み合わせて 2 番ピンの電圧を知って電圧が低いときに基板上の 13 番ピンにつながった LED を点灯するプログラムはプログラムリスト 6.8-1 のようになります。if 文は条件が成立すると最初の中括弧{}の中の命令を実行し、不成立の場合は else の後の{}の中を実行します。動作をフローチャートで表すと図 28 になります。

図 6.8-4 のように配線してプログラムリスト 6.8-1(プログラムのフローチャートは図 6.8-5)を入力し、スイッチを押すと LED が点灯/消灯する様子を確認してみましょう。またデジタル入力ピンになにもつながないと LOW になるか HIGH になるかは電気的なノイズによって決まります。ジャンプワイヤの片側を 2 番ピンに挿したまま、反対側をブレッドボードから抜いて手で触ると LED が点いたり消えたりするので、LOW になるときも HIGH になるときもあることが分かります。

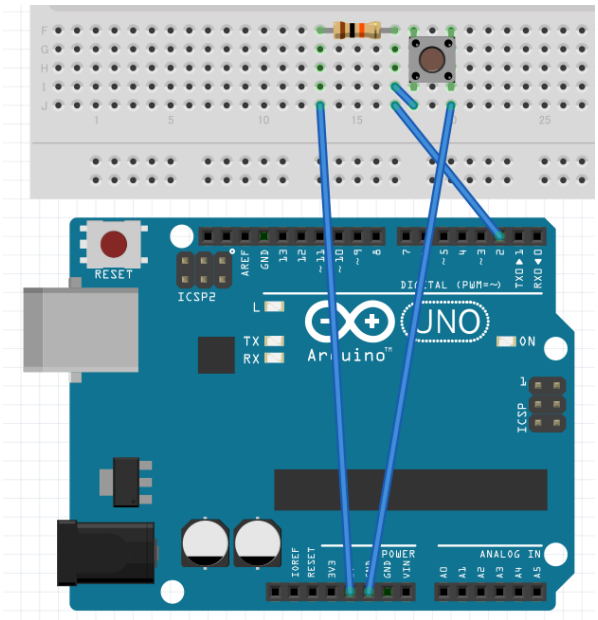


図 6.8-4 スイッチを使ってみる回路のブレッドボード上での配線例。図は 4 本足のタクトスイッチを使っている。4 本足のタクトスイッチを使うときは向きに注意する。

プログラムリスト 6.8-1 入力で LED が点滅するプログラム

```
void setup() {
        pinMode(13, OUTPUT);
}
void loop() {
        if (digitalRead(2) == LOW) {
                digitalWrite(13, HIGH);
        } else {
                digitalWrite(13, LOW);
        }
}
```

ところで

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

の命令文は不要だと思いませんか。不要と思ったら消して書き込んでみましょう。フローチャートは図 6.8-6 に変わります。スイッチを一度押すと LED が点灯したままになったのではないでしょうか。これは digitalWrite の結果が保持されるので LED を消す文がないといつまで経っても点灯したままになるためです。

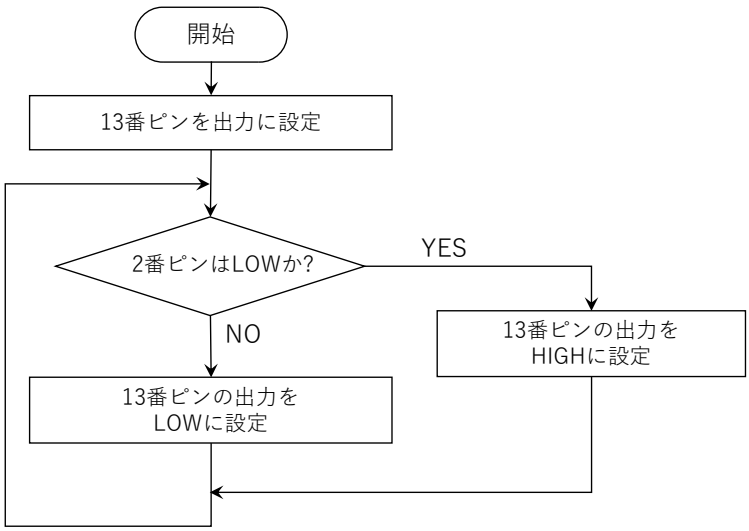


図 6.8-5 プログラムリスト 13 入力で LED が点滅するプログラムのフローチャート

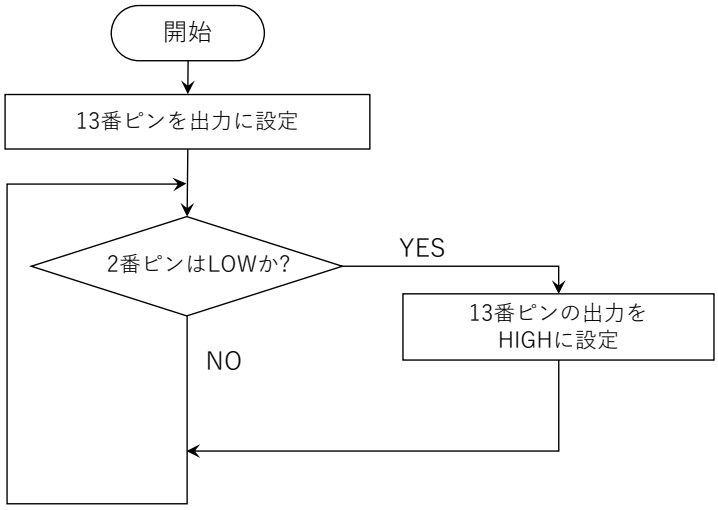


図 6.8-6 プログラムリスト 13 から digitalWrite(13, LOW);を消したときのフローチャート

練習問題 6.8

1. スイッチを押すと音が鳴るようにするにはどうしたらいいでしょうか？わかったら実際に作ってみましょう。
2. スイッチを押していないときは、LED が点滅し、押すと音が鳴るようにするにはどうしたらいいでしょうか？わかったら実際に作ってみましょう。
3. スイッチを D2 ではなく D5 につなぐときは配線とプログラムをどう変えたらいいでしょうか？わかったら実際に作ってみましょう。
4. ブレッドボード上の LED をスイッチで点滅するように回路とプログラムを変更してみよう。

アナログ入力を使うとマイコンはピンの電圧を知ることができます。ここで使うマイコンボードの場合には 0V～5V の間の電圧を 0～1023 の数値で知ることができます[2]。電圧とプログラム上の値の関係は

（プログラム上での値）＝（ピンの電圧）/ 5 V ×1024

で表されます。値は正の整数です。

明るさセンサに CdS セル（硫化カドミウムセル）とよばれるものがあります(図 6.9-1)。CdS は図 6.9-1 の赤い波線部分が受光部で明るさに応じて抵抗値が変わる安価なセンサです[3]。暗いときには抵抗値が大きく、明るくなると抵抗値が小さくなります。2 本の足の区別（向き）はありません。図 6.9-2 のように CdS と抵抗 1 本をつなぐと明るさに応じて AIN のところの電圧が変化します。この回路では明るくなると AIN の電圧が低くなります。AIN をマイコンボードの A0 から A5 のいずれかのピンにつなぐと明るさをマイコンが知ることができます。明抵抗 10kΩ程度の CdS（たとえば秋月電子通商で売っている Macron International Group Ltd.製の MI527）の場合は、抵抗の値は 10kΩ(茶黒橙金，図 6.9-3)でよいでしょう。

図 6.9-4 のように配線してプログラムリスト 6.9-1(フローチャートは図 6.9-5)を入力しましょう。CdS にあたる光を遮って暗くすると LED が点灯します。500 の数値を変えると点灯する明るさの閾値を変えることが出来ます。周囲の明るさによっては点灯しなかったり、消灯しないので、その場合には数値を変更して試してみてください。

図 6.9-1 CdS セルの例(MI527)

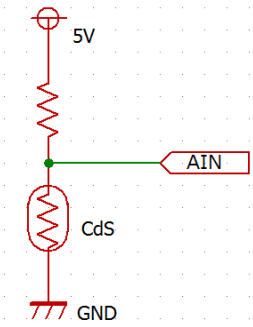
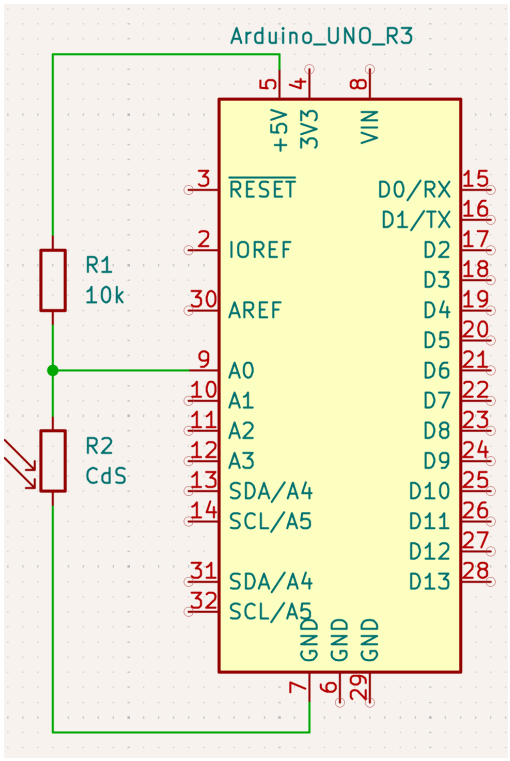


図 6.9-2 CdS をマイコンボードに繋ぐ回路

図 6.9-3 10kΩの抵抗の例（カラーコードは茶黒橙金）

[2] アナログ参照電圧 AREF を変更しない場合

[3] 重金属の一種であるカドミウムを使うことから使われなくなりつつある



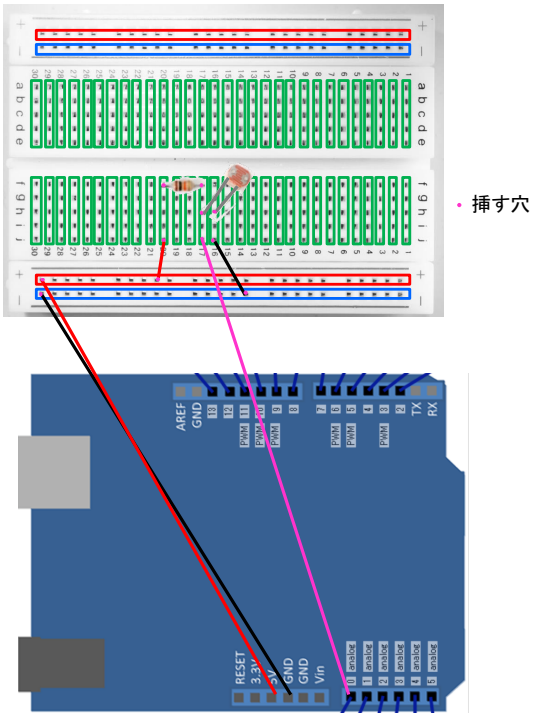


図 6.9-4 CdS を使った回路の例（上:回路図、下：実体配線図）

プログラムリスト 6.9-1　暗くなると LED が点灯するプログラム

```
void setup() {
        pinMode(13, OUTPUT);
}
void loop() {
        if (analogRead(A0) > 500) {
                digitalWrite(13, HIGH);
        } else {
                digitalWrite(13, LOW);
        }
}
```

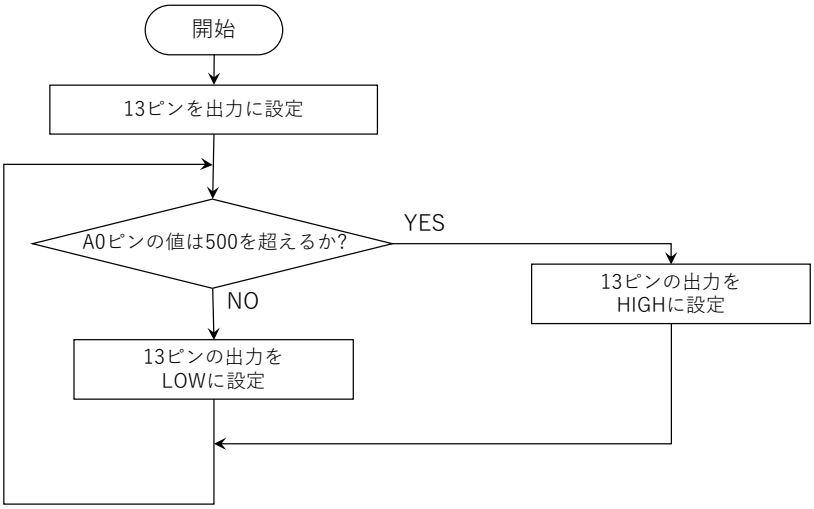


図 6.9-5 プログラムリスト 6.9-1 暗くなると LED が点灯するプログラムの動作

練習問題 6.9

1. 明るくなると音が鳴るようにするにはどうしたらいいでしょうか？分かったら作ってみましょう。
2. CdS を A0 ではなく A2 につなぐときは配線とプログラムをどう変えたらいいでしょうか？分かったら作ってみましょう。
3. 暗いときは LED が点滅し、明るくなると音が鳴るようにするにはどうしたらいいでしょうか？分かったら作ってみましょう。
4. 特定の明るさの時（ある明るさの範囲）だけ音が鳴るようにするにはどうしたらいいでしょうか？分かったら作ってみましょう。

6.10 乱数（random 関数）

乱数というのはサイコロを振って出てくる目のように予想のつかない（ランダムな）数です。random 関数を使うと乱数(疑似乱数)を発生できます。使い方は二通りです：

random(<生成する乱数の上限+1>);

random(<生成する乱数の下限>, <生成する乱数の上限+1>);

下限を指定しない場合、下限は 0 になります。

プログラムリスト 6.10-1 は図 6.4-3(Grove LED モジュールを使う場合は図 5.2-2)の回路で 5 つの内の 1 つの LED だけ点灯させるプログラムです。random 関数の下限を 2、上限+1 を 6+1 の 7 とすることで 2 から 6 の範囲で乱数を発生させています。setup 関数の最初にある randomSeed 関数は乱数の計算の基になる値を決める関数です。コンピュータのプログラムでは実は毎回同じことしかできません。乱数(疑似乱数)の計算も同じです。そこで randomSeed 関数で乱数の計算の基になる値を A0 ピンの値で変えることで予想がつかないようにしています。A0 ピンに何も繋いでいないと値（電圧）が何になるかは分かりません。それを利用しているのです。

練習問題 6.10

1. プログラムリスト xx の randomSeed 関数をコメントアウトして書き込み、毎回同じ LED が点灯することを確かめなさい。

プログラムリスト 6.10-1 乱数で 5 つの内の LED 一つを点灯するプログラム

```
void setup() {
    randomSeed(analogRead(A0));

    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);

    digitalWrite(random(2, 6+1), HIGH);
  }
void loop() {
}
```

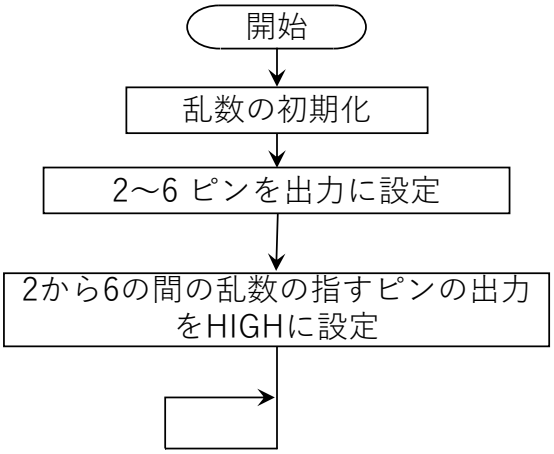


図 6.10-1 プログラムリスト xx のフローチャート

ブレッドボードの使い方のコツ

ブレッドボード上で回路を作る前に上下の穴が電源として使えるよう図 6.11-1 のように配線しておくといいでしょう。

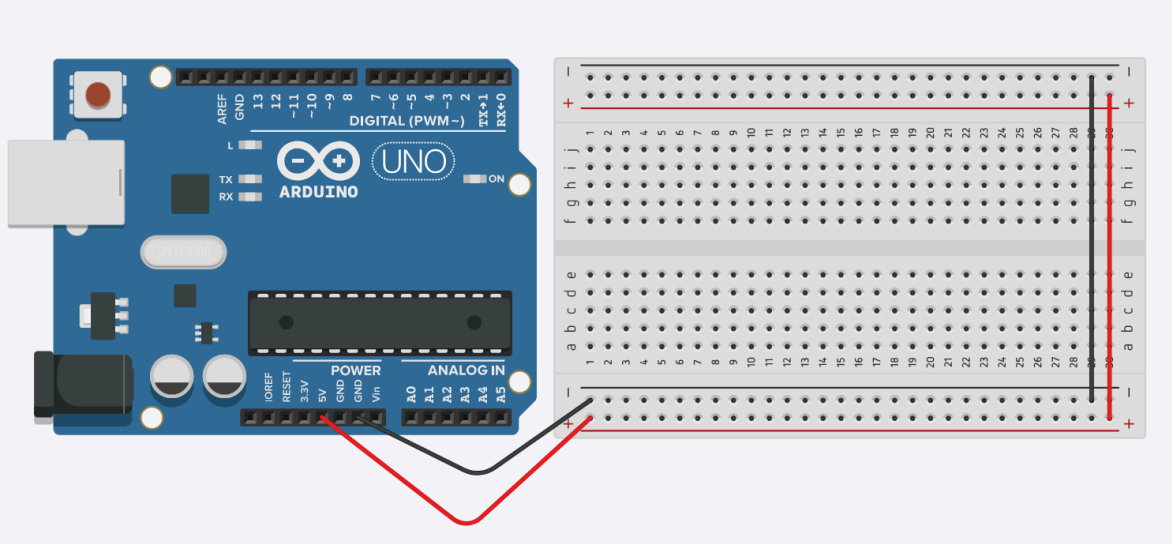


図 6.11-1 マイコンボードの+5V と GND をブレッドボードにあらかじめ配線しておく

LED については 6.1 で説明した回路ではなく図 6.11-2 の回路で配線する方がブレッドボード上の配線が少なく（配線例は図 6.11-3）なります。

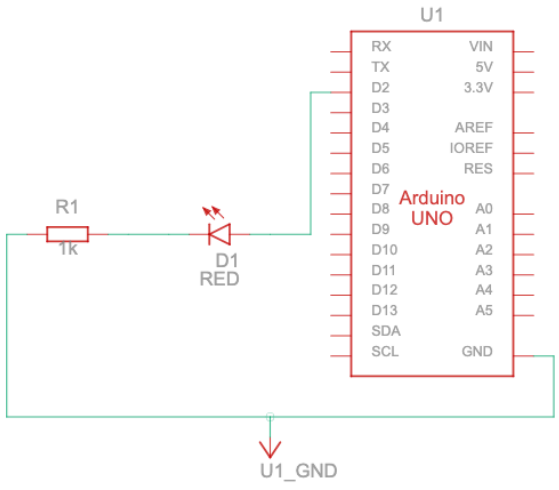


図 6.11-2 ブレッドボード上で配線が少なくなる LED を点灯させる回路

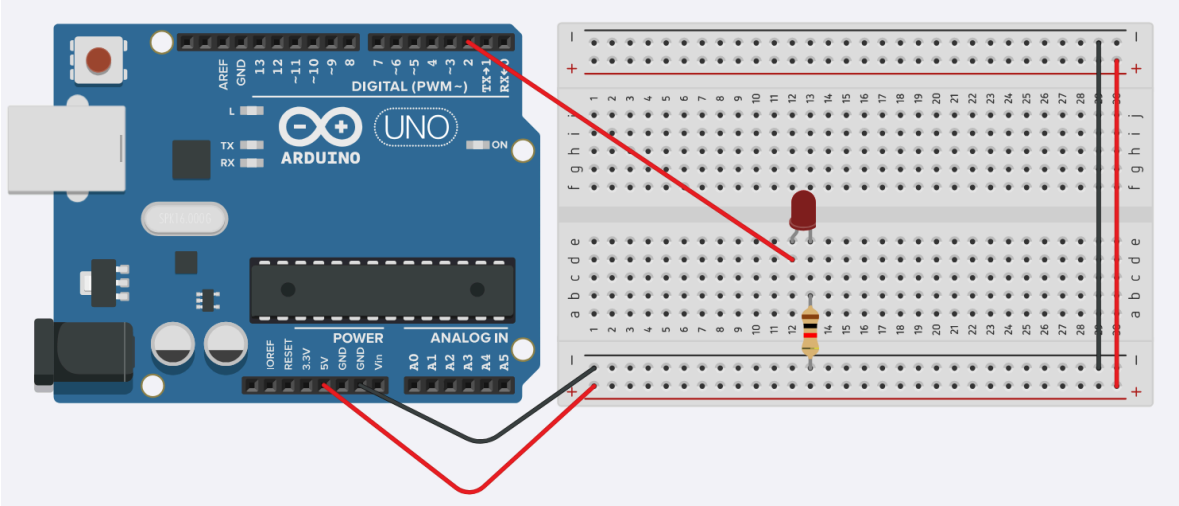


図 6.11-3 ブレッドボード上で配線が少なくなる LED を点灯させる回路の配線例

スイッチで動作を開始するプログラムの書き方

スイッチで動作を開始するプログラムを作るときには return 文を使うとプログラムが少し簡単になります。return 文は関数を終わる（戻るといいます）ときに使う文で、loop 関数に使うと以降の命令を無視して loop 関数の最初から繰り返すことになります。そこで図 6.8-4 のように 2 番ピンにスイッチを繋いでいるとき、loop 関数を次のように書くと

```
void loop() {
        if (digitalRead(2) == HIGH) {
                return;
        }

        // スイッチを押されたら実行したい文を書く
}
```

スイッチが押されていないとすぐに loop 関数の先頭に戻り、スイッチが押されていると続きの命令文が実行されるようになります。Grove ボタンモジュールの時には digitalRead(2) == HIGH ではなく digitalRead(2) == LOW としてください。

作品テーマのヒント

ここまでのプログラム・回路を応用すると次のようなことができます。楽しんで作って見て下さい。

- ラーメンタイマ
  - 3 分間 LED が点滅し、3 分経つとチャルメラのメロディを繰り返す
  - 1 分経過ごとに音を鳴らすと実用的です。

- カウントダウンタイマ
  - ラーメンタイマの一種です。LED で残り時間を表示します。
- 明るくなるとメロディが流れる
  - 箱の中に回路を入れておくと、蓋を開けると明るくなります。それにあわせてメロディを鳴らすと、誕生日プレゼントなどに使えます。それを意識してプログラムを作ってもいいでしょう。
  - 重要なものを入れた引き出しや冷蔵庫に入れておくと、開けていることが音で分かります。
- 暗くすると音が鳴る
  - 回路を暗くすると音が鳴るように仕掛けます。触られたくないものの手前に置いておくと、警報になります。
- 電飾・メロディ
  - クリスマスツリーなど LED を点滅させたり、音を鳴らすと楽しいでしょう。
  - CdS に手をかざすと明るさが変化するので、それに合わせて点滅や音を変化させると面白くなります。
- 信号機・踏切シミュレータ
  - 信号機や踏切の点滅変化を LED で再現してみます
  - 盲人用信号の音楽や、踏切の音を再現してもいいでしょう。
- 電子占い、電子サイコロ
  - スイッチを押すと毎回違う目を表示します（サイコロの目はサイコロのように 7 個 LED を並べると表せる）
  - しばらく目が変わる様子を見せても面白いでしょう。音を出しても面白いです。
- 電子ルーレット
  - LED を円形に並べます。スイッチを押すと LED が順番に点灯し、最後にどこかで止まります。
  - 点灯する LED の位置の変わり方を、最初は速く、だんだん遅くすると雰囲気が出ます。点灯する LED の位置が変わるときに音を出すと面白くなります。

ラーメンタイマのフローチャート

ヒントとしてラーメンタイマのフローチャートを図 6.11-4 に示します。黄色い背景のところは for 文のところです。薄青背景のところを工夫することで自分なりのオリジナルになりますよ。

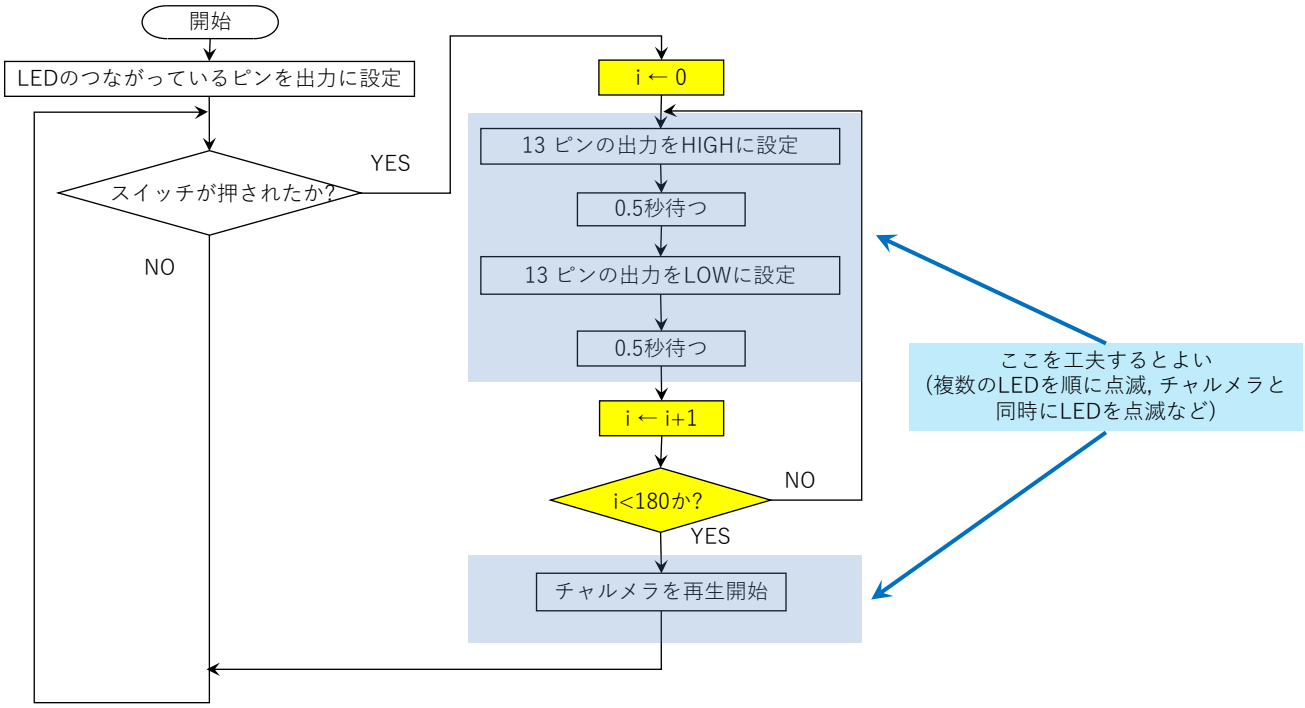


図 6.11-4 ラーメンタイマーのフローチャート

動画集と回路とプログラムの例

マイコンボードを使った作品の動画集を用意しています。上記の作品のヒントであげたテーマを作った例も多くあります：

https://n.mtng.org/ele/arduino/samples/sample90.html

本書で取り上げたものと重なるところはありますが Arduino を使った回路とプログラムを解説した文章です：

http://n.mtng.org/ele/arduino/samples/

7 Arduino 言語の文法

Arduino 言語は C 言語, C++言語を基にしているので文法はそれらとほぼ同じです。以下では 4 章説明していない算術演算子、関係演算子、論理演算子、定数、変数、配列変数、制御文について順に簡単に説明します。詳細な内容は Arduino 日本語リファレンス(http://www.musashinodenpa.com/arduino/ref/)を見るとよいでしょう。

7.1 算術演算子

四則演算（加減乗除）で使う記号は表 7.1-1 の通りです。乗除算の記号が見慣れないかもしれません。また括弧は中括弧、大括弧は使わず () だけを使います。また整数の計算で余りを求める記号%があります。ほかに 1 を足して代入する ++ と 1 を引いて代入する -- があります。

i ++, ++ i

または

i --, -- i

と書きます。変数の前に書くか後に書くかで少し動作が変わります（ここでは説明を省きます）。関数の引数は定数や変数だけでなく、演算子を使った計算式をいれることもできます。

表 7.1-1 四則演算の記号

	数学の記号	Arduino 言語での表記
加算	＋	+
減算	−	−
乗算	×	*
除算	÷,	/
余り		%
括弧	(), {}, []	()

7.2 関係演算子

二つの式の値の関係を調べる関係演算子は表 7.2-1 の通りです。統合を含む場合には>, < に続けて = を書きます。等しいかどうかを調べるには==と書きます。=が一つだと別の意味になるので注意します。二つの式の値が式の関係を満たす（式が成立する）ときに値が真(1, true)に満たさないときに偽(0, false)になります。

表 7.2-1 比較の記号

数学の記号	Arduino 言語での表記
＞	>
≧	>=
＜	<
≦	<=
=	==
≠	!=

7.3 論理演算子

式の真偽を反転する（否定する）!、二つの式が共に真であるときに真となる論理積の&&、二つの式のいずれかが真であるときに真となる論理和の|| の 3 つの論理演算子があります。

Arduino 言語では数学のように**3 つの式の比較はできません**。そこで論理演算子を使います。たとえば a<b<c が成り立つかを調べたいときには a<b && b<c と書きます。数学で a<10, 12<a と書く範囲に a の値が入っているかを調べるときには a<10 || 12<a と書きます。

7.4 定数

定数はプログラムを実行中に変わらない値のことです。Arduino 言語では 10 進数のほかに 2 進数と 16 進数が書けます。書き方は次の通りです：

- 10 進数（整数）の例： 123
- 2 進数（あたまに B をつける）の例：B1010
- 16 進数（あたまに 0x をつける）の例： 0x123
- 10 進数（浮動小数点、. をつける）の例：123.0, 1.23
- 10 進数（指数表現、 e を使う）の例： 2.4×10^5 を表したければ 2.4e5

また以下の記号が使えます：

- デジタル入出力でよく使うもの： HIGH, LOW
- デジタル入出力の向きを示すもの： INPUT, OUTPUT
- 真と偽： true, false
- マイコンボード上の LED のピン番号： LED_BUILTIN

変数とはプログラム上で数値などを憶えておくためのものです。変数の名前は既に使われていなければ（たとえば setup や loop, LOW, HIGH, for, while などはダメです）。自分で自由に決められます。名前の長さは任意で英文字で始まれば、数字や_を使っても構いません。大文字と小文字は区別されます。

変数名の例）　　a, b, a1, a2, a3, a4, LEDpin, LED_pin1

変数を使う前には宣言が必要です。宣言というのはコンパイラに変数の型（扱う数値の範囲など）を知らせるものです。型には char, int, short, long, float, double などがあります。とりあえずは int 型を使っておくとよいでしょう。宣言は

型＜空白＞名前;

または

型＜空白＞名前1, 名前2,…;

と書きます。たとえば

```
int a;
int a, b;
```

と書きます。上記のように書くと変数の初期値は不定（決まっていない）です。初期値を決めるには

```
int a; a = 0;
int a=2;
```

といった書き方があります。上は宣言の後で代入をしていて、下は宣言と初期値の代入を一つの文で書いています。

変数に数値を憶えさせる（代入という）には ＝ を使います。たとえば

```
a = 2;
b = 100;
```

と書きます。数学の=（等号）と代入の記号は意味が違います。次のようなプログラムを書いたとします。

```
a = 2;
b = 3;
c = a + b;
a = 100;
```

4 行目で c の値は 2+3 の 5 になります。5 行目で a の値は 100 になりますが、c の値は 5 のまま変わりません。＝ は変数間の関係を表す（方程式を書く）ものではなく、代入の記号だと言うことを憶えて置いてください。

7.6 配列変数

配列は数学の数列(a_1, a_2, a_3, …)のようなものです。配列は使う前に大きさを宣言して使います。たとえば

```
int a[10];
```

として宣言します。この例では型は int ですが、ほかの型でも構いません。配列の添え字は 0 から始まります。したがって上の例のように大きさを 10 で宣言すると、有効な添え字は 0 から 9 となります。宣言した大きさを超えた添え字を使うとするとうまく動作しません（一見動作して見えることがありますが何かおかしくなります）。プログラム上では添え字は[ ]で表します。たとえば

```
a[0] = a[1];
a[i+1] = a[i] + 1;
```

と書きます。

7.7 制御文

Arduino 言語ではプログラムは命令文を書かれた順に実行していきますが、その流れを変えるのが制御文です。制御文には条件分岐の if 文、繰り返しの while 文、do-while 文、for 文があります。以下では if 文、while 文、for 文を説明します。

7.7.1 if 文

if 文は

```
if (条件式) {
        〈条件式が成り立つ（真のとき）に実行する文〉
} else {
        〈条件式が成り立たない（条件式の値が偽のとき）に実行する文〉
}
```

と書きます。フローチャートで示すと図 7.7-1 となります。条件式は digitalRead 関数、analogRead 関数や 7.2, 7.3 で説明した演算子を使って書きます。もし偽のときに実行する文がなければ

```
if (条件式) {
        〈条件式が成り立つ（真のとき）に実行する文〉
}
```

と書いても構いません。また実行する文が 1 つしかないときは{ }を省略して

```
if (条件式)
        〈条件式が成り立つ（真のとき）に実行する文〉
else
        〈条件式が成り立たない（条件式の値が偽のとき）に実行する文〉
```

または

```
if (条件式)
        <条件式が成り立つ(真のとき)に実行する文>
```

と書いても構いません。

if 文を使ったプログラムの例（部分）です

```
if (digitalRead(2) == LOW)
        digitalWrite(13, HIGH);
else
        digitalWrite(13,LOW);
```

この例では 2 番ピンの値が LOW なら 13 番ピンを HIGH にし、そうでなければ LOW にします。2 番ピンにスイッチをつけるとスイッチでマイコンボード上の LED が点滅するプログラムです。このプログラムのフローチャートは図 7.7-2 となります。

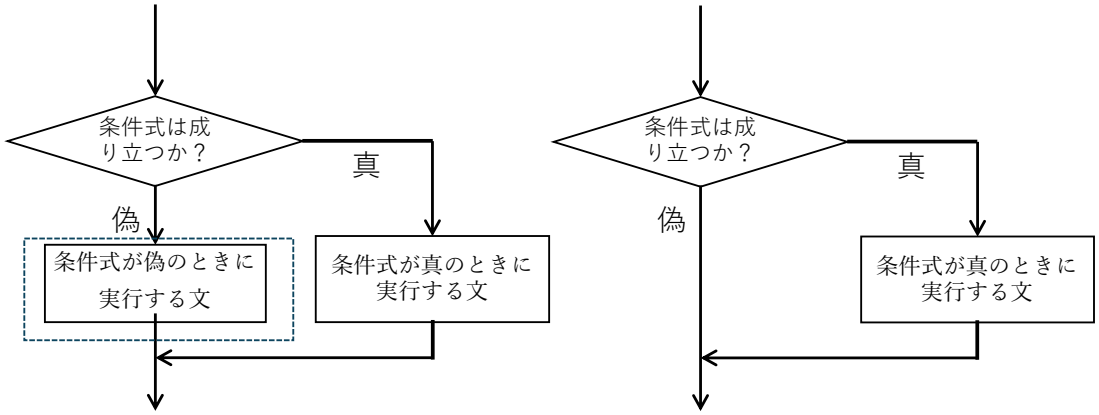


図 7.7-1 if 文の部分のフローチャート。右は else 以下を省略したとき。

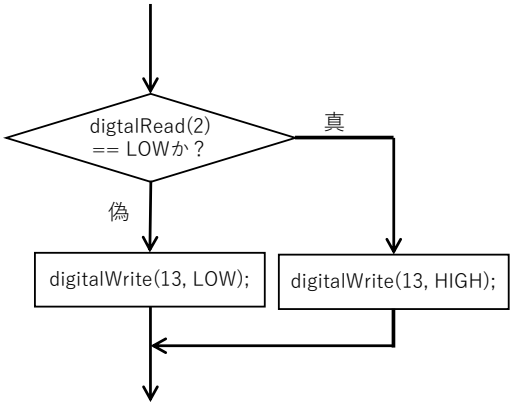


図 7.7-2 if 文の例のフローチャート

7.7.2 while 文

while 文は条件式が成り立っている間、繰り返す文です。次のように書きます：

```
while (条件式) {
        〈条件式が成り立つ（真の）ときに実行する文〉
}
```

動作をフローチャートで表すと図 7.7-3 となります。フローチャートをみると分かるように、最初に条件が成り立っていないときには一度も実行しません。また条件式を確認して、繰り返す文を実行することを繰り返すので、

- 繰り返す文を途中で中断できません（必要なら中断するプログラムを書く必要があります）
- スイッチを条件式で確認するプログラムでは、プログラムが条件式で判断する的確なタイミングにスイッチを押していないと反応しないプログラムになる場合があります。

このあたりの解消方法は色々あるので探してみてください。

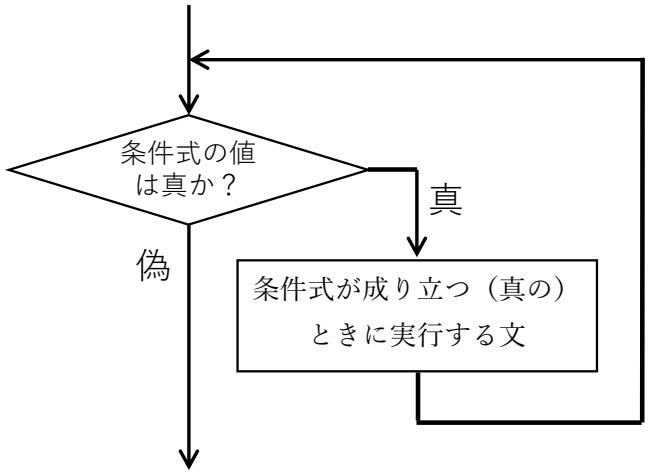


図 7.7-3 while 文の動作

プログラム例）

```
while (digitalRead(2) == HIGH)
        ;
```

このプログラムは図 6.8-4 の回路でスイッチが押されて 2 番ピンの値が HIGH になるまでずっと待つという動作をします(フローチャートは図 7.7-4)。「;」だけの文は何もしないという命令文になります。

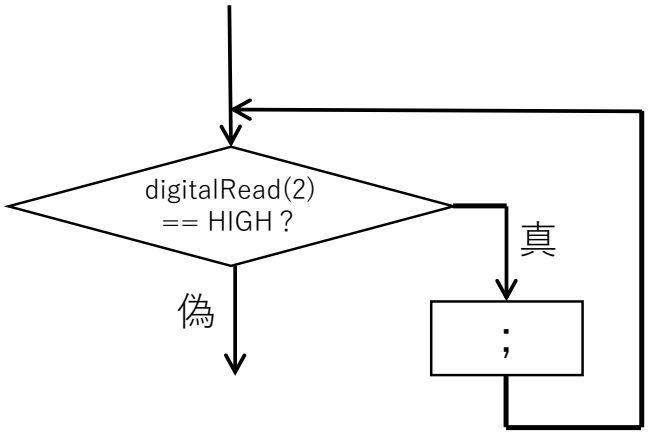


図 7.7-4 while 文のプログラム例のフローチャート

7.7.3 for 文

for 文は回数を決めた繰り返しによく使われます。次のように書きます：

```
for(文 1; 条件式; 文 2) {
        〈繰り返す文〉
}
```

フローチャートは図 7.7-5 となります。最初に文 1 を実行してから条件式を確認し、条件が成立していれば繰り返す文を実行し、文 2 を実行します。そして条件式を確認して成立していれば繰り返します。文 1 と文 2 がある以外は while 文と同じ動作です。

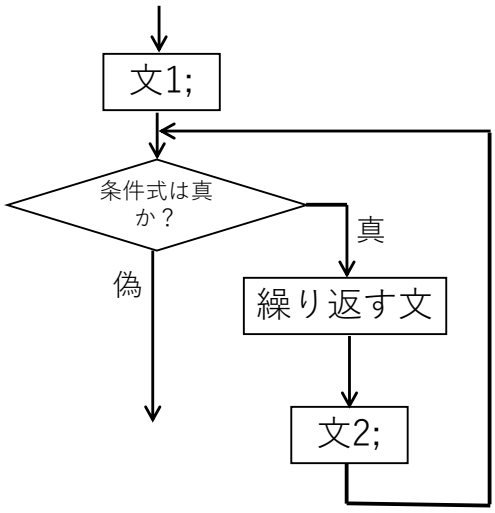


図 7.7-5 for 文のフローチャート

プログラム例）

```
for (int i=0; i<10; i++)
        delay(1000);
```

このプログラム例のフローチャートは図 7.7-6 です。最初に i を宣言して初期値を 0 にしています。条件式 i<10 が成り立っているので、1000ms 待ち、i に 1 を足します。i は 1 になりますが i<10 が成り立っているので、1000ms 待ち i に 1 を足します。これを繰り返すので、結果として 1000ms×10 回待つプログラムとなっています。

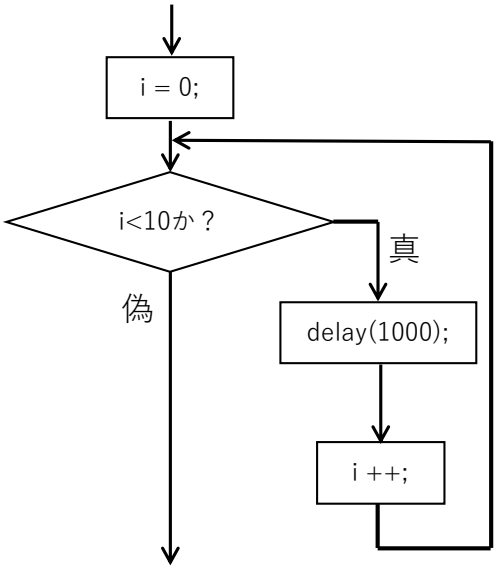


図 7.7-6 for 文のプログラム例のフローチャート

7.7.4 複文

中括弧 { } で囲まれた複数の命令文は一つの命令文扱いになります。{ }で囲んだ文を複文と呼びます。制御文は複文でなければ一つの命令文しか実行しません。7.7.1～7.7.3 の説明で、一つの文の場合は { } を省略できると書きましたが、正確には制御文は 1 つの文のみを実行するようになっているのを、複文にすることで複数の文を実行できるようにしていています。