Arduino/ Suwano をはじめよう@KYOTO-U　No.3

2012 年 5 月 2 日

京都大学 学術情報メディアセンター

喜多　一

今回の学習目標

- 音の出し方を知る.
- 配列の使い方を知る.
- 関数を定義して使う.
- プログラムの改変に挑戦する.

１.　回路の準備.

ブレッドボードを用いて図１のような回路を作成してください．圧電ブザーの電極に極性はありません.

Suwano ではドーターボード上のスピーカーと LED 2 を用いて実習できます.

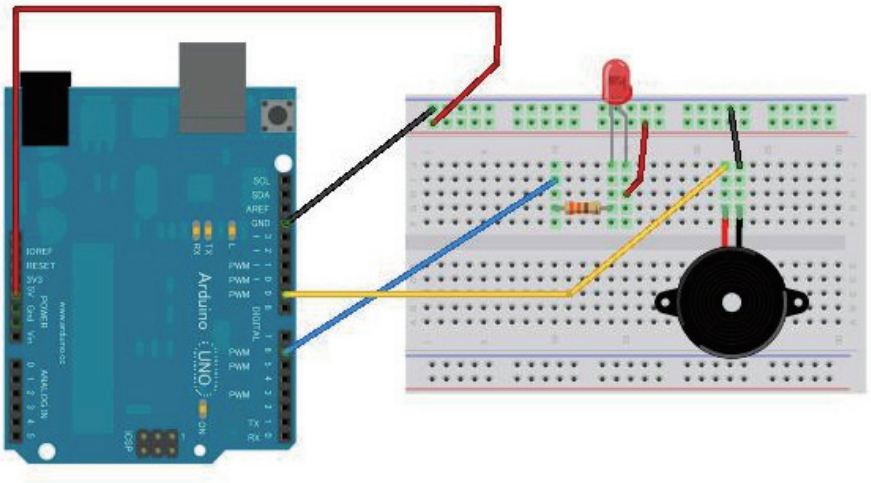


図 1 作成する回路（実体図）

Fritzingを使用して作成
http://fritzing.org/download/

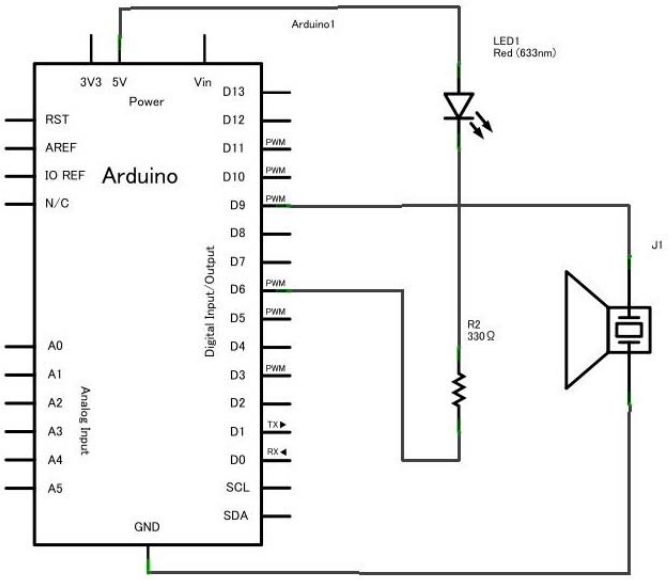


図 2　作成する回路（回路図）

２．　音の出し方を知る

次のプログラムを作成し，実行してみてください．

行番号	ソースコード	説明
1	#define SPEAKER 9	9 番ピンを SPEAKER と，
2	#define LED 6	6 番ピンを LED とします．
3	int freq[] =	初期値を持った配列の宣言
4	{440, 466, 494, 523, 554, 587, 622, 659, 698, 740,	音階に相当する周波数
5	784, 831, 880, 932, 988, 1047, 1109, 1175, 1245,	
6	1319, 1397, 1480, 1568, 1661, 1760};	
7	int TONE_MIN = 0;	音階の添え字の最小値
8	int TONE_MAX = 24;	音階の添え字の最大値
9	int melody[] = {3, 5, 7, 8, 10, 12, 14, 15, -1};	曲の音程．
10	int M_LENGTH = 9;	曲の長さ
11	int mIndex = 0;	曲の場所を指す変数
12	int DURATION = 750;	１音の長さ(750ms)
13	void rewind(void);	関数 rewind() と
14	void makeNextTone(void);	makeNextTone() のプロトタ
15		イプ宣言

```
16  void makeNextTone() {                                  次の音を出す関数
17    int fIndex = melody[mIndex];                         音程を取り出す
18    if ((fIndex>=TONE_MIN)&&(fIndex<TONE_MAX)){          音階の範囲にあれば
19      tone(SPEAKER,freq[fIndex],DURATION);               tone 関数で音を出す
20      delay(DURATION);                                   音の長さだけ待つ
21    } else {
22      digitalWrite(LED,LOW);                             そうでなきえればLEDを点灯
23      delay(DURATION);                                   750msec 待って
24      digitalWrite(LED,HIGH);                            LED を消灯
25    }
26    mIndex = mIndex + 1;                                 曲の場所を進める
27    if (mIndex >= M_LENGTH) {                            範囲に達したら
28      mIndex = 0;                                        最初にもどす
29    }
30  }
31
32  void rewind() {                                        曲の先頭にもどる関数
33    mIndex = 0;
34  }
35
36  void setup() {
37    pinMode(SPEAKER, OUTPUT);                            ピンの設定
38    pinMode(LED, OUTPUT);
39    digitalWrite(LED,HIGH);                              LED を消灯
40    rewind();                                            曲の先頭に
41  }
42
43  void loop() {
44    makeNextTone();                                      次の音を出す
45  }
```

音をスピーカー(圧電ブザーと呼びます. あまり電流を消費しないので Arduino のポートに直接つなぐことができます)から出すための関数が

tone(int ポート, int 周波数, int 長さ)

です. 周波数は Hz, 長さは ms (ミリ秒)です. この関数は音を鳴らしながら, 平行してプログラムの続きを実行するので同じ長さだけ, delay() 関数で待つようにしています.

３．　配列と関数

3.1 配列

2 節のプログラムでは音階を表す周波数と曲の音程をそれぞれ

```
int freq[] =
 {440, 466, 494, 523, 554, 587, 622, 659, 698, 740,
  784, 831, 880, 932, 988, 1047, 1109, 1175, 1245,
  1319, 1397, 1480, 1568, 1661, 1760};

int melody[] = {3, 5, 7, 8, 10, 12, 14, 15, -1};
```

と表現しています. このように多くのデータを一括してプログラムで扱う手段として配列があります.
これらの例では int 型の配列 freq や melody を初期データとともに外部変数として宣言しています.
一般には

配列要素の型 配列名[] = {初期値データ, …, 初期値データ};

という形式になります. このほかに, 配列の大きさを指定して宣言する方法として

配列要素の型 配列名[要素数];

という方法もあります. 例えば

int melody[9];

とすると, int 型の要素を 9 つ持つ配列 melody を用意します.

配列の添え字は C 言語では 0 から始まります. 先の例の melody の配列要素と初期値は以下のようになります.

melody[0] → 3, melody[1] → 5, melody[2] → 7, melody[3] → 8, melody[4] → 10, melody[5] → 12, melody[6] → 14, melody[7] → 15, melody[8] → -1

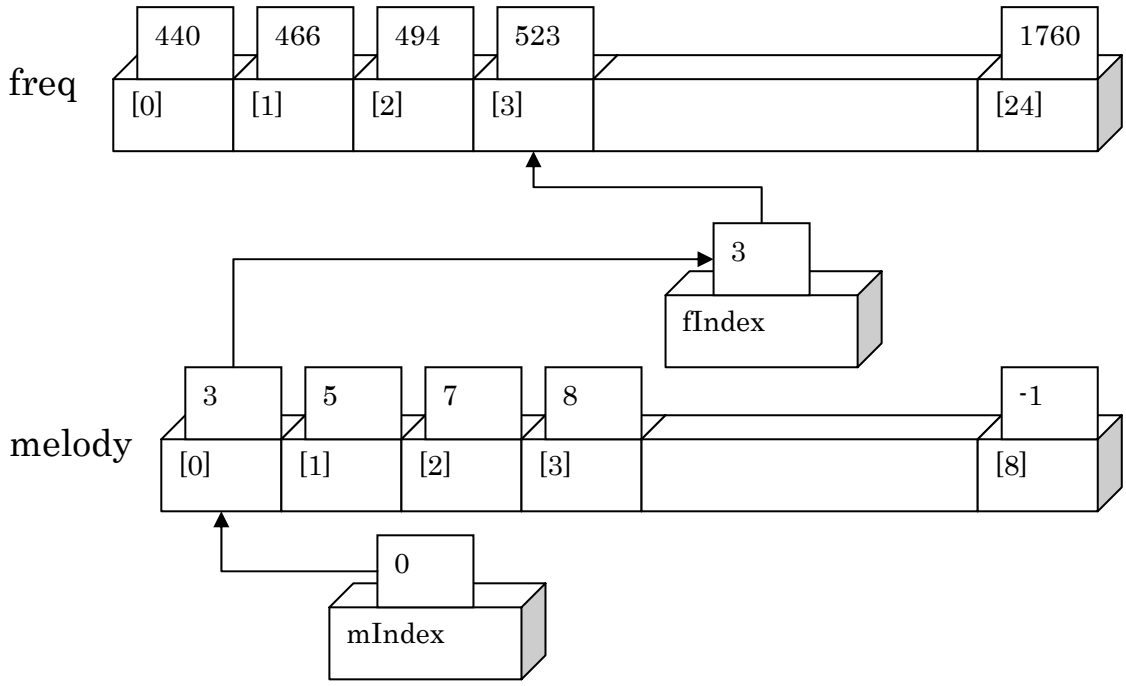
配列の要素は[]内に要素の添え字を指定すれば後は普通の変数のように内容を得たり, 値を代入したりすることができます.
ここでは曲の先頭にもどる関数 rewind() と次の音を出す関数 makeNextTone() を定義して呼び出しています. いずれも引数も戻り値もありません. ただし, どちらの関数も外部変数 mIndex の値を変更します.
先のプログラムでは

```
int fIndex = melody[mIndex];
if ((fIndex>=TONE_MIN)&&(fIndex<TONE_MAX)){
  tone(SPEAKER,freq[fIndex],DURATION);
  delay(DURATION);
} else {
```

とありますが, fIndex = melody[mIndex]; で曲を表す配列 melody の添え字 mIndex が指す要素の内容（音程）を読み出して音程の添え字 fIndex を得ています.

次に tone(SPEAKER,freq[fIndex],DURATION)； の中で音程を表す周波数の配列 freq から添え字 fIndex の示す要素の内容（周波数）を得て，音を出す関数 tone() の引数として与えています.

変数は「値」を書いた札を入れる箱，配列はこのような箱が連なって，配列名と添え字で参照できるものに例えることができます. 変数 mIndex, fIndex を使って配列 melody[], freq[] の値を参照している様子を図解したものが下の図です.



3.2 関数

2 節のプログラムでは rewind() と makeNextTone() という２つの関数を定義し、呼び出しています。

これまで関数として定義したものは

```
void setup()
void loop()
```

でした。これらはどちらも Arduino でのプログラムに必須のものとして想定されているものを利用者が記述したものです。

一方、利用者が呼び出したライブラリ関数としては

```
void pinMode(pin, mode)
```

```
void digitalWrite(pin, value)
void delay(ms)
int digitalRead(pin)
void tone(pin, frequency, duration)
```

があります.

2 節のプログラムではプログラム内に新たに関数 makeNextTone() と rewind() を定義し, 関数 setup() や loop() の中で呼び出しています.

- **関数プロトタイプ宣言**:13, 14 行目は「関数プロトタイプの宣言」と言い, 定義する関数について予め「返り値の型」,「関数名」,「引数の型」を実際の関数の定義やこの関数を呼び出す関数の定義の前に書きます. どちらの関数も引数も返り値もないのでともに void と書かれています. 最後にセミコロン(;) が付くことに注意してください.
- **関数の定義**:16 ～ 30 行目と 32～34 行目がそれぞれ関数 makeNextTone() と rewind() の定義です. 関数の返り値の型, 関数名, (引数の型と引数名) のあと { } で囲まれた部分が関数の定義です.
- **関数の呼び出し**: 40 行目と 44 行目でそれぞれ rewind() と makeNextTone() を呼出してます.

引数や返り値のある関数

例題:先の例では1音の長さを 750 msec に固定し, マクロ定義でこれを DURATION という文字列で表せるようにしていました. 音楽ではテンポは1 分間の拍数で定めることが多いので, 拍数から待ち時間を計算する関数 duration() を作ってみましょう.

関数プロトタイプ宣言は

```
unsigned long duration(unsigned long tempo);
```

となります。

関数定義本体は

```
unsigned long duration(unsigned long tempo){
  if (tempo <= 0) {
      return 0;
   }
  return (unsigned long)60*1000/tempo;
}
```

とします. 引数 tempo は 0 だったり, 負の数だったりしてはいけないのですが, そのことを検査して, tempo の値が適切でない場合は値 0 を戻すことにします. そうでない場合は拍数に対して1拍の長さをミリ秒単位で計算して return 文で返します. 計算は int 型では桁数が足りないので unsigned long 型の演算を行なったうえで、呼び出す時は例えば 1 分間に 80 拍の場合は duration(80) と呼び出せばよいので, tone 関数の中では以下のように使います.

```
tone(SPEAKER, freq[fIndex], duration(80));
```

４．　プログラムを改造してみる

1節の回路，2節のプログラムを改造して次の機能を付加してください．

1．コンタクトスイッチを回路に加えて D2 ピンでオンオフを読み取るようにしてください．
2．コンタクトスイッチを押すことで音楽の再生，停止を行なえるようにしてください．
3．現在は曲については音程しか与えていませんが，音符の長さを同時に与えられるようにしてください．8 分音符から全音符程度を表現できるようにしてください．音程は melody[] に 12 音階で表現したものを格納し，音程に相当する周波数へは freq[] という配列を参照して変換しています．同様の考え方に基づいて音符での表現とそれから時間の長さへの変換を行なうようにしてください．