Arduino/ Suwano をはじめよう@KYOTO-U　No.5

2012 年 5 月 31 日

京都大学 学術情報メディアセンター

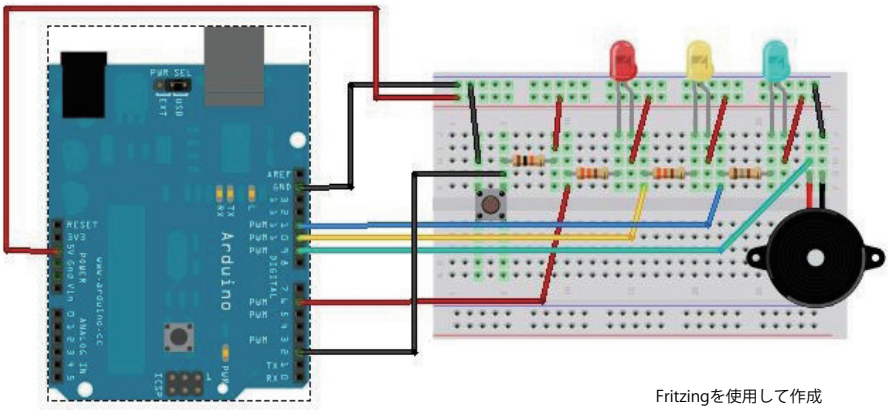
喜多　一

今回の学習目標

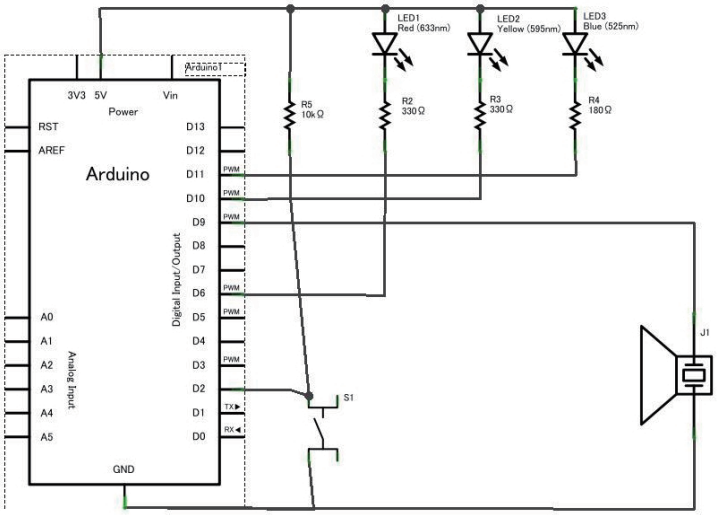
・ 演習課題についてのコードの検討

１．　回路の想定

- LED はピン 6，10，11 を使うものとし、これまで同様、抵抗器を直列に入れてアノード側を＋5V に、カソード側を Arduino のデジタル入出力ピンに接続します。
- スピーカは 9 番ピンと GND につなぎます。
- コンタクトスイッチは +5V 側に直列に抵抗器を入れる、抵抗器側にのピンを Arduino の 2 番ピンに配線し、反対側を GND に配線します。



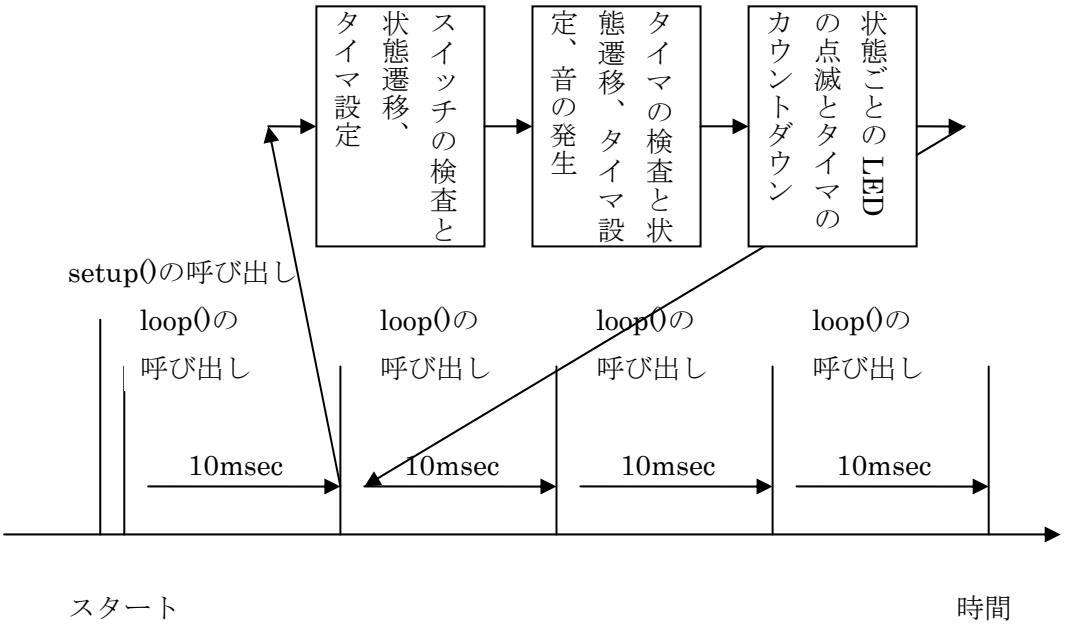
Fritzingを使用して作成
http://fritzing.org/download/



２．　プログラムの考え方

音の鳴らすことの時間を別に loop() 関数では 10msec 待つ（関数 delay(10);を使用)ことし、毎回、

1. 最初にスイッチの変化を検査し、状態の変化とタイマの設定を必要に応じてします。またスイッチの値を「前回の値」用の変数に格納します。
2. 次に、タイマで状態変化しなければならない状態でタイマの時間切れを検査し、必要な状態変化、音の発生とタイマの設定をします。
3. そして各状態に応じて LED を点滅させるとともに、タイマのカウントダウンを行います。
4. 最後に loop() の終わりで 10msec 待ちます。



３．　プログラム

行	ソースコード	説明
1	#define LED1 6	
2	#define LED2 10	
3	#define LED3 11	
4	#define SPEAKER 9	
5	#define BUTTON 2	
6		
7	int state = 0; // 状態, 0->1->2->3	状態を保持する変数
8	int timerLength[] = {10,5};	タイマ長（秒）
9	int toneHight[] = {500,1000};	ブザーの音の高さ
10	int val = HIGH;	スイッチ読み取り変数
11	int oldVal = HIGH;	初期値は押されていない(HIGH)
12	long timerCounter = 0;	タイマカウンタ変数
13	void setup() {	(長時間計るので long 型)
14	pinMode(LED1,OUTPUT);	ピンのセット
15	pinMode(LED2,OUTPUT);	
16	pinMode(LED3,OUTPUT);	
17	pinMode(SPEAKER,OUTPUT);	
18	pinMode(BUTTON,INPUT);	
19	digitalWrite(LED1, HIGH);	LED の消灯
20	digitalWrite(LED2, HIGH);	
21	digitalWrite(LED3, HIGH);	
22	state = 0;	状態の初期化
23	}	
24	void loop() {	
25	/* ボタンの読み込みと押し下げによる状態変化の処理 */	
26	val = digitalRead(BUTTON);	
27	if ((val==LOW)&&(oldVal== HIGH)) {	スイッチの変化の検出
28	/* state change by button press */	
29	if (state == 0) {	状態が 0 なら 1 に
30	state = 1;	
31	timerCounter = timerLength[0]*100;	状態の更新, タイマの設定
32	} else {	それ以外は 0 に
33	state = 0;	
34	}	
35	}	
36	oldVal = val;	直近のスイッチの読みの保存

37	/* タイマーが 0 になったことによる状態変化の処理 */	
38	if ((state ==1) || (state ==2)) {	
39	if (timerCounter == 0) {	タイマの確認
40	if (state == 1) {	
41	state = 2;	状態ごとの遷移とタイマの設定
42	tone(SPEAKER,toneHight[0],50);	音の発生
43	timerCounter = timerLength[1]*100;	
44	} else if (state == 2) {	
45	state = 3;	
46	tone(SPEAKER,toneHight[1],50);	
47	delay(100);	
48	tone(SPEAKER,toneHight[1],50);	この 60msec はタイマに反映さ
49	}	れない
50	}	
51	}	
52	/* 各状態のときにする仕事, LED の点灯, 消灯, タイマの	
52	カウントダウン */	
54	if (state == 0) {	状態ごとの LED の点灯
55	digitalWrite(LED1,HIGH);	
56	digitalWrite(LED2,HIGH);	
57	digitalWrite(LED3,HIGH);	
58	} else if (state == 1) {	状態が 1 なら LED1 を点灯、タ
59	digitalWrite(LED1,LOW);	イマをカウントダウン
60	digitalWrite(LED2,HIGH);	
61	digitalWrite(LED3,HIGH);	
62	timerCounter--;	
62	} else if (state == 2) {	状態が２なら LED2 を点灯、タ
64	digitalWrite(LED1,HIGH);	イマをカウントダウン
65	digitalWrite(LED2,LOW);	
66	digitalWrite(LED3,HIGH);	
67	timerCounter--;	
68	} else if (state == 3) {	状態が３なら LED3 を点灯
69	digitalWrite(LED1,HIGH);	
70	digitalWrite(LED2,HIGH);	
71	digitalWrite(LED3,LOW);	
72	}	
73	delay(10);	10msec 時間待つ
74	}	

２．関数，switch 文などを利用した改良版

行	ソースコード	説明
1	#define LED1 6	
2	#define LED2 10	
3	#define LED3 11	
4	#define SPEAKER 9	
5	#define BUTTON 2	
6		
7	int state = 0; // 状態，0->1->2->3	
8	int timerLength[] = {10,5};	
9	int toneHight[] = {500,1000};	
10	int val = 0;	
11	int oldVal = 0;	
12	long timerCounter = 0;	long 型で精度を確保
13	int timerTick = 10;	タイマの刻み時間を明示化
14	/* 関数プロトタイプ */	
15	void setTimer(int sec);	
16	void countdownAndDelay();	
17	boolean isTimeUp();	
18	void onOffLED(int state);	
19		
20	/* タイマ関連の関数 */	タイマの設定，引数の単位は秒
21	void setTimer(int sec) {	
22	timerCounter = (long)sec * 1000/timerTick;	
23	}	
24		
25	void countdownAndDelay() {	カウントダウンとディレイの一
26	if (timerCounter>0) {	括処理
27	timerCounter--;	
28	}	
29	delay(timerTick);	
30	}	
31		
32	boolean isTimeUp() {	時間切れを確認する関数
33	return (timerCounter<=0);	
34	}	
35	/* ボタン関連の関数 */	
36	boolean isButtonPressed() {	ボタンの押し下げを検出する関

37	boolean isPressed;	数
38	val = digitalRead(BUTTON);	
39	isPressed = (val==LOW)&&(oldVal== HIGH);	
40	oldVal = val;	
41	return (isPressed);	
42	}	
43	/* LED 関連の関数 */	
44	void onOffLed(int state) {	状態ごとに必要な LED を点灯
45	digitalWrite(LED1,HIGH);	する
46	digitalWrite(LED2,HIGH);	
47	digitalWrite(LED3,HIGH);	
48	switch (state) {	Switch 文を使用
49	case 1: digitalWrite(LED1,LOW);	
50	break;	
51	case 2: digitalWrite(LED2,LOW);	
52	break;	
52	case 3: digitalWrite(LED3,LOW);	
54	break;	
55	}	
56	}	
57		
58	void setup() {	
59	pinMode(LED1,OUTPUT);	
60	pinMode(LED2,OUTPUT);	
61	pinMode(LED3,OUTPUT);	
62	pinMode(SPEAKER,OUTPUT);	
62	pinMode(BUTTON,INPUT);	
64	state = 0;	
65	onOffLed(state);	
66	}	
67		
68	void loop() {	
69	/* ボタンの読み込みと押し下げによる状態変化の処理 */	
70	if (isButtonPressed()) {	
71	if (state==0) {	
72	state = 1;	
73	setTimer(timerLength[0]);	
74	} else {	

```
75        state = 0;
76      }
77    }
78  /* タイマーが 0 になったことによる状態変化の処理 */
79   if (isTimeUp()) {
80      switch (state) {
81      case 1:
82          state = 2;
83          tone(SPEAKER,toneHight[0],50);
84          setTimer(timerLength[1]);
85          break;
86      case 2:
87          state = 3;
88          tone(SPEAKER,toneHight[1],50);
89          delay(100);
90          tone(SPEAKER,toneHight[1],50);
91          break;
92      }
93    }
94  /* 各状態のときにする仕事, LED の点灯, 消灯 */
95    onOffLed(state);
96  /* タイマのカウントダウンと時間待ち */
97    countdownAndDelay();
98  }
```

３．プログラム改善のポイント

・ 関数にして処理をカプセル化

関数を使うパターン

- 引数で情報を渡し，返り値で結果をもらう．(これが素直)
- 引数で情報を渡し，返り値は不要（処理は LED の点灯とか）
- 引数も返り値も使わない
  - ✧ 情報の受け渡しがない（決まりきった動作をする）
  - ✧ 外部変数を使って情報を受け渡しする．動作が分かりにくいの注意を要する
- 引数で情報を渡し，引数で返す．
  - ✧ 引数にポインタを使う必要がある．

・ switch 文の活用: 式の値によって処理を分ける．break 文を入れないとそのまま次の処理を行う（fall down）.

```
switch (整数の式) {
値: 処理;
    break;
値: 処理
    break;
default: 処理
    break;
}
```