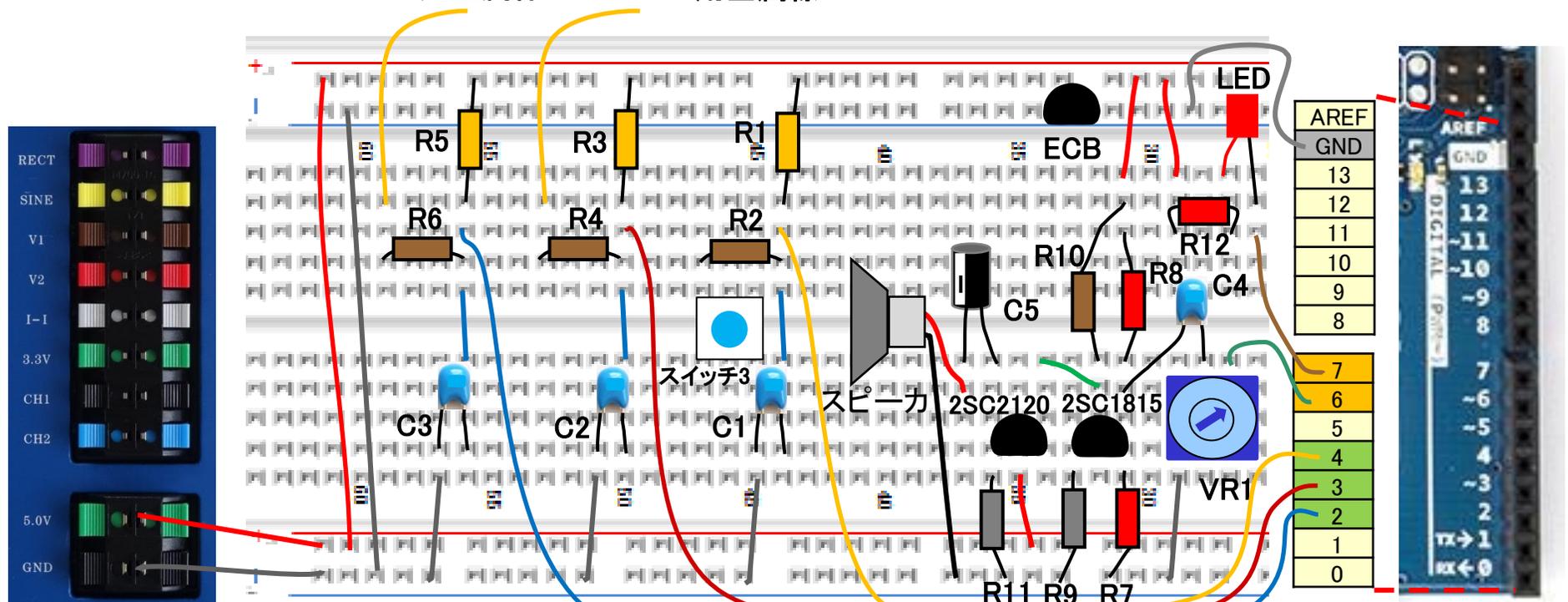


(a)回路図

ゴール用金属線へ コース用金属線へ



(b)接続図

```

107 /* Arduino UNO setup ST */
108 void setup()
109 {
110     Serial.begin(9600);
111     pinMode(SW_START, INPUT_PULLUP);
112     pinMode(SW_TOUCH, INPUT_PULLUP);
113     pinMode(SW_GOAL, INPUT_PULLUP);
114     pinMode(LED_OUT, OUTPUT);
115     digitalWrite(LED_OUT, HIGH);
116     TOUCH_STATE = true;
117     /* Initialize the MML and note module. */
118     mml_init(&mml, mml_callback, 0);
119     MML_OPTION_INITIALIZER_DEFAULT(&mml_opt);
120     int tempo_default = 160;
121     note_init(&note, tempo_default, mml_opt.bticks);
122 }
123 /* Arduino UNO setup ED */
124 /* Loop function ST */
125 void loop()
126 {
127     if(digitalRead(SW_TOUCH) == LOW && TOUCH_STATE == false)
128     {
129         TOUCH();
130     }
131     if(digitalRead(SW_START) == LOW && TOUCH_STATE == true)
132     {
133         START();
134     }
135     if(digitalRead(SW_GOAL) == LOW && TOUCH_STATE == false)
136     {
137         GOAL();
138     }
139     else

```

各ピンを入力・出力に設定

入力の状態をチェック

(c)

```

81 /* Start, Touch, Goal ST */
82 void START()
83 {
84     if(TOUCH_STATE == true)
85     {
86         TOUCH_STATE = false;
87         START_TIME = millis()*1.000;
88         digitalWrite(LED_OUT, LOW);
89         Serial.println("***** S T A R T !! (-v-) *****");
90         Serial.println("   Good luck ... ");
91         wave_start();
92         delay(500);
93         digitalWrite(SW_START, HIGH);
94     }
95 }
96
97 void TOUCH()
98 {
99     if(TOUCH_STATE == false)
100     {
101         TOUCH_STATE = true;
102         digitalWrite(LED_OUT, HIGH);
103         GOAL_TIME = float (millis() - START_TIME)/1000.000;
104         Serial.println("x x x C R A S H E D !! (T_T) x x x");
105         Serial.print("   Total time : ");
106         Serial.println(GOAL_TIME);
107         wave_touch();
108         digitalWrite(SW_TOUCH, HIGH);
109     }
110 }
111
112 void GOAL()
113 {

```

スタート時のArduino起動時間を  
Millis()を用いてSTART\_TIMEへ格納シリアルモニタにメッセージ表示  
スタート時のメロディを鳴らすクラッシュ時のArduino起動時間から  
START\_TIMEの差を計算シリアルモニタにメッセージ表示  
クラッシュ時のメロディを鳴らす

(b)

(※)当サンプルスケッチのコンパイルには、  
別途 ‘A tiny MML parser’ ライブラリ(後述)の導入が必要となります

(e)

```

stime  note.cpp  note.h  song.h
35 #include <avr/pgmspace.h>
36
37 // Super Mario Bros. Original music composed by KOJI KONDO
38 const char wave_start_text[] PROGMEM = "T240L8>E8R16E8R16R8E8R16R8C8R16E4R16G4R16R4<G4R4";
39 const char wave_touch_text[] PROGMEM = "T240L8>C32C+16D16R8R4<A8R16>F8R8F8R16F8R8E8R16D8R8C8R16<E8R8E8R16C8R8R4";
40 const char wave_goal_text[] PROGMEM = "T240L8<G83R16>C83R16E83R16G83R16>C83R16E83R16G4.R16E4R8<<A-83R16>C83R16E-83R16";
41
42 #endif
43

```

MML形式の記述データ

当サンプルスケッチでは、スタートボタン・コース用金属線・ゴール用金属線を3つのスイッチとして設定し、それぞれの入力をスタート・クラッシュ・ゴールとして判定しています(図c)  
LEDはゲーム開始から終了時まで点灯、スピーカーはスタート・クラッシュ・ゴールの判定時にメロディが流れます。

また、Arduino UNOをPCと接続し、シリアルモニタを開いてゲームを行うと計測タイムが表示されます。これはArduino起動時間を返すmillis()関数を用いてスタート時、終了時の時間差を計算して行っています。  
※E-Station・Arduino・PCを同時に接続する際は配線にご注意ください。

3種類のメロディはMML(ミュージック・マクロ・ランゲージ)という演奏データ用の記述言語で表現しており、記述通りのノート番号・テンポで再生をすることで楽譜通りのメロディの再現ができます。ArduinoでこのMMLを扱うためのライブラリ‘A tiny MML parser’を導入することにより、MMLの記述で簡単に好きなメロディの再生を行わせることができます。  
当サンプルスケッチでは‘A tiny MML parser’をMITライセンスに基づき利用しています。

‘A tiny MML parser’

Copyright (C) 2014-2015 Shinichiro Nakamura

<http://www.cubeatsystems.com/firmware/tinymml/>

Released under the MIT license

<http://opensource.org/licenses/mit-license.php>