講師用

ロボット博士養成講座

ロボティクスプロフェッサーコース

不思議アイテムⅠ－１②

（第3回/第4回テキスト）

必ず、生徒に授業日と自分の名前を記入させるようご指導をお願いいたします。

第３回授業日 2024年　　月　　日

第４回授業日 2024年　　月　　日

名　前

Human ヒューマンアカデミー ジュニア STE"AMスクール

ロボット博士養成講座 ロボティクスプロフェッサーコース

2024年8月授業分

1B2K_24.08

ロボット博士養成講座

ロボティクスプロフェッサーコース

不思議アイテムI－1②

第3回

マイコンで演奏（えんそう）してみよう

講師用

目　　次

○　授業開始にあたって

　授業のはじめは、着席させ、大きな声であいさつしてから始めます。

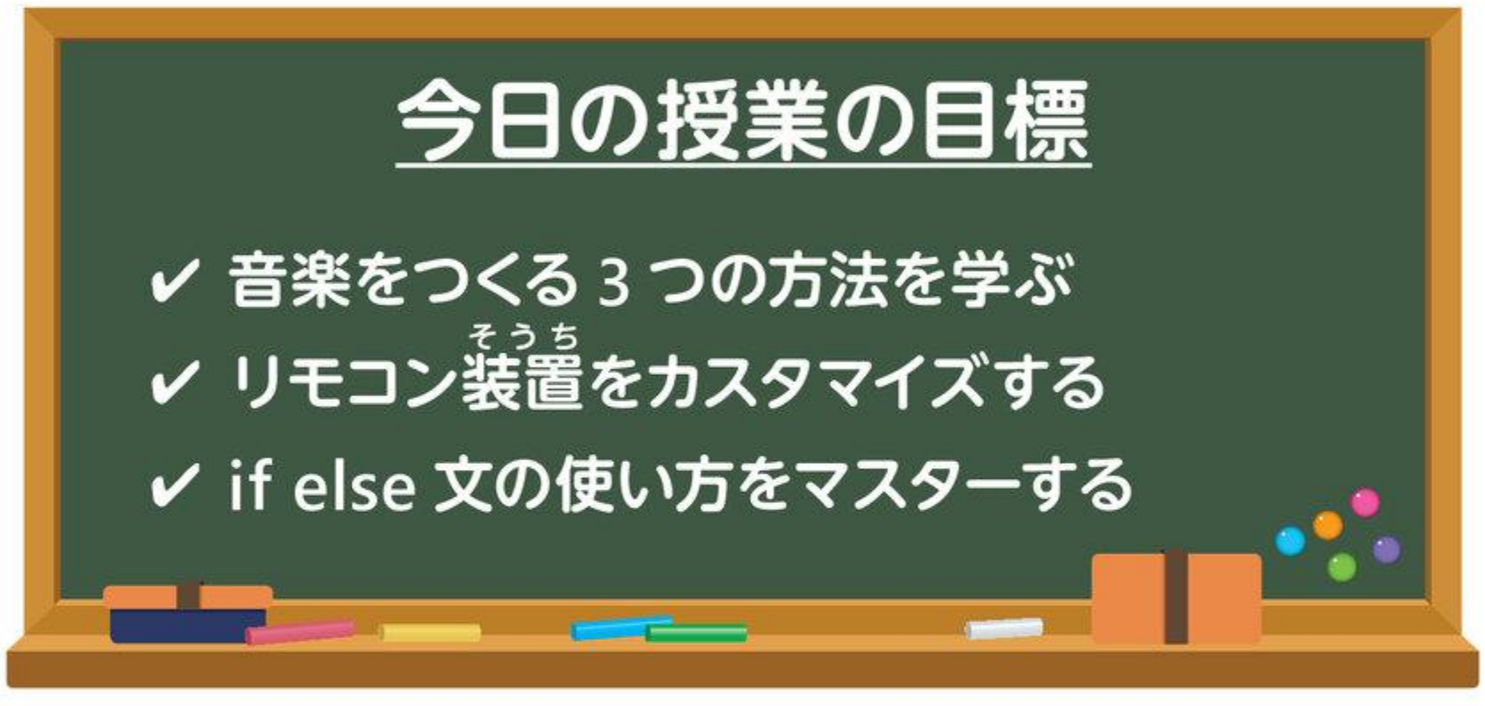
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

　（授業の目標を明確化することは大変重要なことですので、生徒によく理解させます）

目安時間は授業時間 120 分のうち、休憩 10 分程度を取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

0. マイコンで演奏(えんそう)してみよう（目安 5 分）

0.0. 「マイコンで演奏(えんそう)してみよう」でやること



第２回では、タッチセンサーをスイッチとして利用して、マトリクスLEDに表示されたキャラクターを動かしたり、スピーカーから音を出したりしました。しかし、いろいろな音楽を聞くなかで、「自分だったらもっとカッコいい曲をつくれるのに」とか、「自分がいつも聞いているあの曲を流してみたい」などと思いませんでしたか？
そこで、今回は、自分だけの曲を流してみましょう。そのためには少々面倒(めんどう)ですが、1音1音パソコンに打ち込(こ)んでいかなければなりません。その事前の知識として、そもそも「音」とは何なのかを考えましょう。
また、音楽にはライブ演奏(えんそう)というのもありますね。コンサート会場で、ミュージシャンたちが息を合わせて楽器を演奏(えんそう)するように、マイコンを楽器にして、みんなで演奏(えんそう)してみましょう。
さらに今回は、コントローラーを自分専用のリモコン装置(そうち)につくりかえます。これを「カスタマイズ」といいます。コントローラーを使って、ライブ演奏(えんそう)しましょう。パソコンからはなれて操作できるので、ダンスをしながらでも演奏(えんそう)できます。

0.1. 必要なもの

マイコンボード、ロボプロシールド、マトリクスLEDシールド、マトリクスLEDの順番に積み重ねたユニットと、タッチセンサーとスピーカーを引き続き今回も使います。
さらにマイコンユニットを遠隔(えんかく)操作する「コントローラー」と、マイコンユニットとコントローラーを通信させる「無線受信モジュール」も使います。コントローラーは単4電池を３本使用します。あらかじめセットしておきましょう。

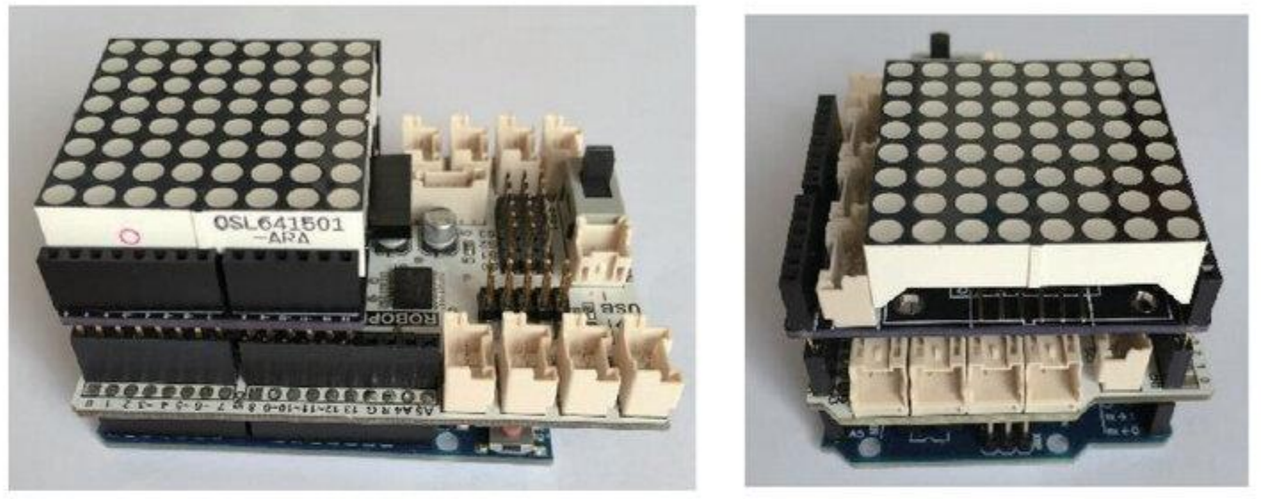


図0-0　前回までに組み立てたもの

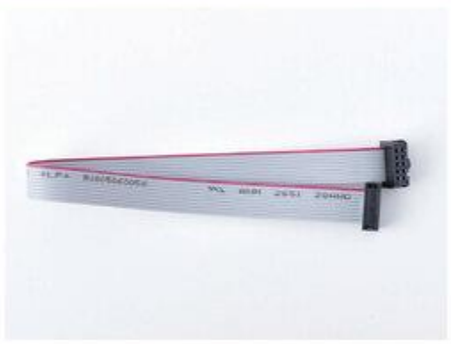
USBケーブル 1	マイコンボード 1	ロボプロシールド 1	リボンケーブル 1
			
コントローラー 1	**無線受信モジュール 1**	**タッチセンサー 2**	**マトリクスLEDシールド 1**
			
マトリクスLED 1	**スピーカー 1**		
			

図0-1　必要なもの

1. 曲をつくってみよう（目安 60 分）

1.0. マイコンとスピーカーで音を鳴らしてみよう

今回は、音階や音程など学校で習った音楽の勉強内容を思い出しながら、マイコンとプログラミングで音を奏でる方法を学びます。音楽をつくる方法はいくつかありますが、ここでは以下の3つの方法を学びましょう。

POINT

- 「RTTTL」をベースにしてつくる
- 「周波数」をベースにしてつくる
- 「ミュージカル・ノート」をベースにしてつくる

1.1. タッチセンサーとスピーカーの取り付け

ロボプロシールドにタッチセンサー2個とスピーカーを接続し、前回の終了時と同じ状態にします。**図1-0**のように、タッチセンサーのケーブルを D0 と D1 のコネクターに、スピーカーのケーブルを D2 のコネクターにさし込みます。

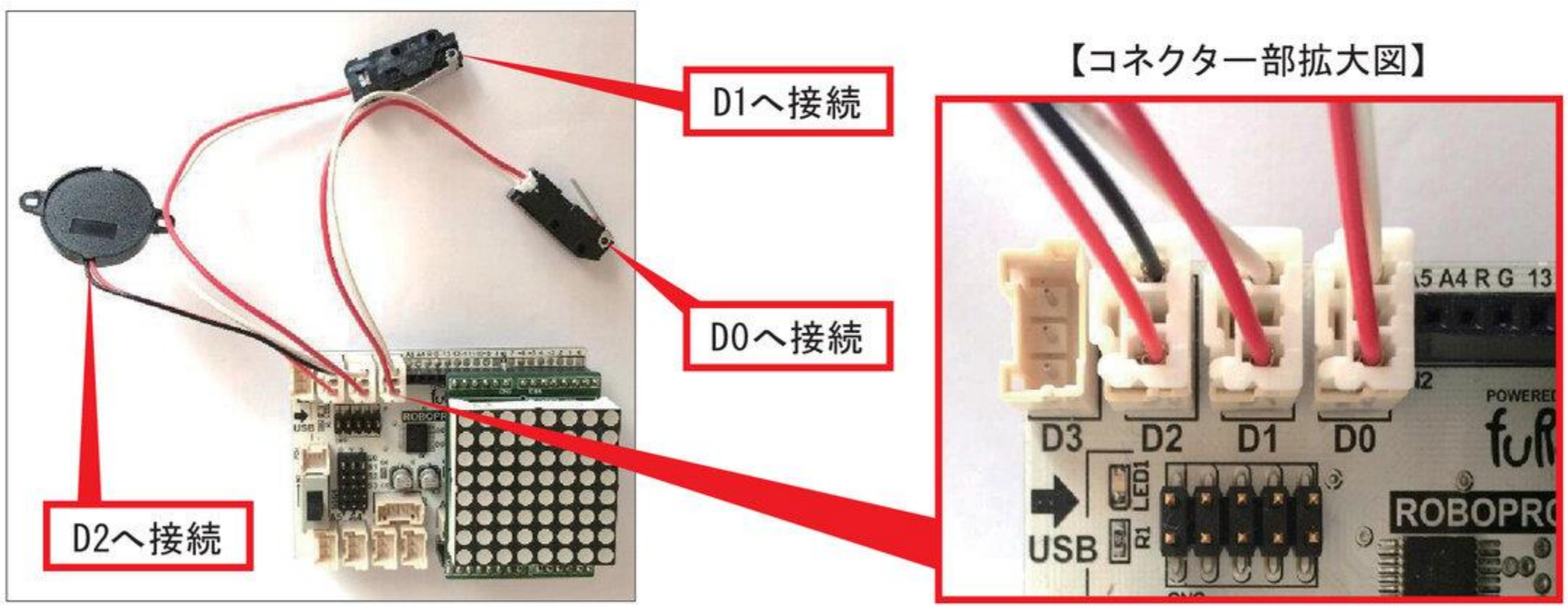


図1-0　ロボプロシールドとタッチセンサー、スピーカーの接続

1.2. 動作確認

タッチセンサーとスピーカーの接続が確認できたら、第2回で使ったプログラムで動作確認をしましょう。以下のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse1 > MagicItemA2 > MatrixSpriteMove2

2個のタッチセンサーを押したときに、十字マークが行ったり来たりすると同時に、「ピッ」「ピッ」という音が出れば動作確認終了です。

1.3.　RTTTL で曲をつくってみよう

1）RTTTLとは？

Ring Tone Text Transfer Language（ RTTTL ）とは、携帯電話に着信音を転送するために開発された言語で、一昔前の携帯電話の着信音に使われていました。まずは、このRTTTLで書かれたプログラムを使って音を鳴らしてみましょう。では、以下のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse1　>　MagicItemA3　>　RTTTL_Kaeru

実行結果：「カエルのうた」が演奏(えんそう)される。
演奏(えんそう)が終わったら、リセットボタンを押すとプログラムが最初から演奏(えんそう)されます。となりの人と協力して、演奏(えんそう)のタイミングをずらして輪唱しても面白いかもしれませんね。

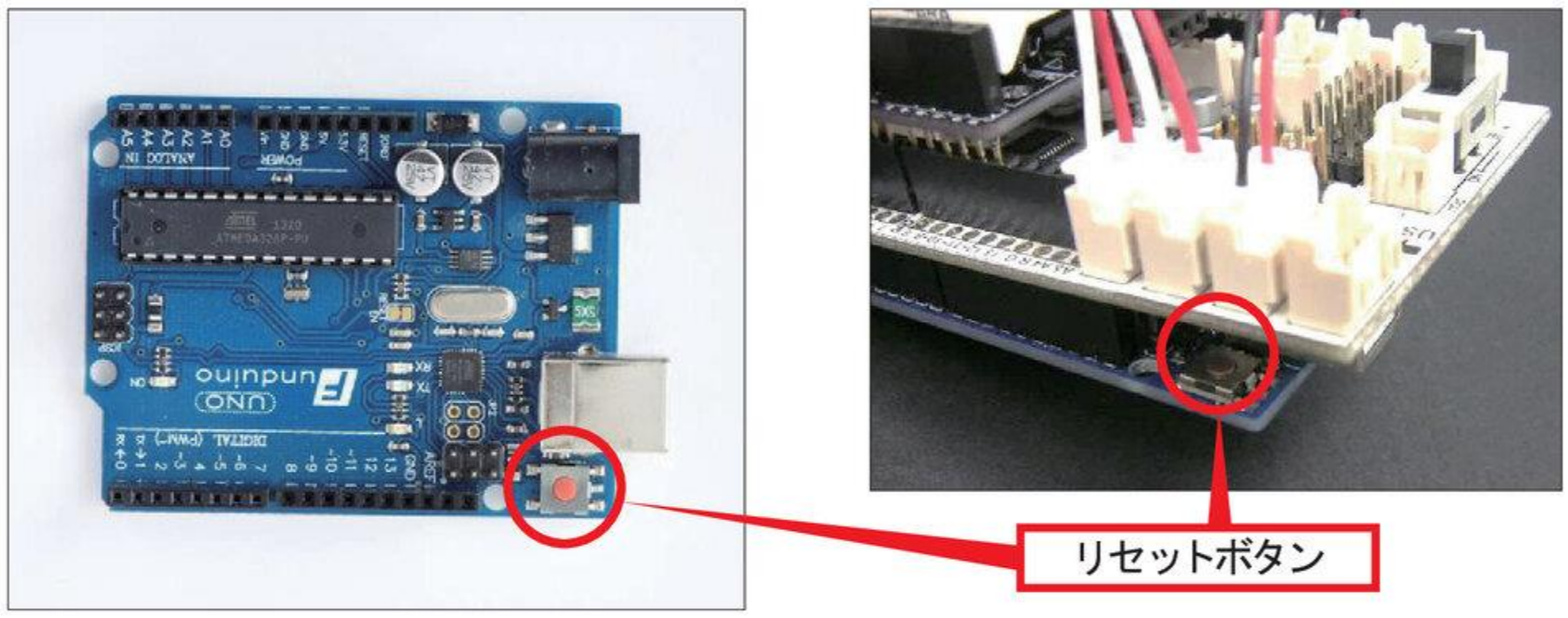


図 1-1　リセットボタン

2）楽譜の構成

さて、ここでプログラムの中を見てみましょう。

プログラム「RTTTL_Kaeru」より抜粋

```
// 楽譜データ
char *song = "kaeru:d=4,o=4,b=200:4c,4d,4e,4f,4e,4d,2c,4e,4f,4g,4a,4g,4f
,2e,4c,4p,4c,4p,4c,4p,4c,4p,8c,8c,8d,8d,8e,8e,8f,8f,4e,4d,1c";
```

プログラム中に上記のような記述（コード）がありますが、ここに「カエルのうた」の楽譜が入っています。さきほど説明したRTTTLで書かれています。

POINT

楽譜データは以下のように3つのブロックで構成されます。
1: 名前ブロック　2: 初期設定ブロック　3: 楽譜データブロック

```
// 楽譜データ
char *song = "kaeru  :d=4,o=4,b=200:
   1          2
4c,4d,4e,4f,4e,4d,2c,4e,4f,4g,4a,4g,4f,2e,4c,4p,4c,4p,4c,4p,4c,4p,
8c,8c,8d,8d,8e,8e,8f,8f,4e,4d,1c";
                                    3
```

1 のブロックは、この楽譜の名前です。「カエルのうた」なので「kaeru」です。

2 のブロックには、**「d」「o」「b」**という、3つの初期設定値が書かれます。カエルのうたの場合は、d=4,o=4,b=200 ですね。
dは「duration」の略で、音符の長さを意味します。
oは「octave」の略で、オクターブ、すなわち音程を意味します。
bは「beat」の略で、楽譜のはやさ、テンポを決めます。この値が小さいほどスローになります。1分間にいくつ拍をいれるかを示します。

3 のブロックは、楽譜のデータです。1音符をカンマ(,)で区切って使います。つまり、「8c」や「8d」といったかたまりの数だけ音符があるということになります。

char *song という部分は、前のテーマで登場した int i などのように、変数をつくるときの宣言です。 char が変数の種類（型）、song が変数の名前です。つまり、song という名前の変数をつくり、そこに楽譜のデータを入れているということになります。

あとは、「song の楽譜を使って曲を演奏して!」という命令を出すだけですね。

プログラム「RTTTL_Kaeru」より抜粋

```
tone1.play_rtttl(song);  // songという変数に入った曲を流す
```

命　　令 「play_rtttl」
実行結果：RTTTL形式でつくられた楽譜データを使い曲を流す
使 い 方：tone1.play_rtttl(song); // songと名付けた楽譜データの曲を流す

3）楽譜データブロックの中の音符の書き方

さらに、楽譜データブロックの中の音符の書き方を説明します。音符は通常、「16b6」「8e7」といったように、「①数字＋②英字＋③数字」で書かれます。この英数字がそれぞれ何を意味しているのかを確認してみましょう。

①音符と休符の長さ（最初の数字）

最初の数字は「音符の長さ」を表します。たとえば「16」と書かれていたら、それは16分音符を意味します。 先ほど出てきた初期設定ブロックの中の「d」（duration）と同じ数字の場合は、省略することができますが、今回の「RTTTL_Kaeru」はプログラムを見た人がわかりやすいように、あえて省略していません。

表1-0　音符と休符の長さ

表記	音符		休符	
	意味	記号（長さ）	意味	記号（長さ）
1	全音符（基準）		全休符（基準）	
2	2分音符（2分の1）		2分休符（2分の1）	
4	4分音符（4分の1）		4分休符（4分の1）	
8	8分音符（8分の1）		8分休符（8分の1）	
16	16分音符（16分の1）		16分休符（16分の1）	
32	32分音符（32分の1）		32分休符（32分の1）	

長い ↑ ↓ 短い

②音階（英字）

次の英字は、音階を表します。つまり、ピアノの鍵盤ですね。おなじみの「ドレミ」と照らし合わせるとわかりやすいです。

表1-1　音階

表記（プログラム）	音名（アメリカ式）	音名（イタリア式）
p	（休符）	（休符）
a	A	ラ
b	B	シ
c	C	ド
d	D	レ
e	E	ミ
f	F	ファ
g	G	ソ

③音程（最後の数字）

最後の数字は、オクターブを表します。この値が大きいと、同じ「ド」でもより高い「ド」が出ます。これも、音符の長さと同じように、初期設定の「o」（octave）と同じ場合は省略することができます。ここでは、スピーカーの仕様により、oは4～7の値しか取れません。

表1-2　音程

表記（プログラム）	音階Aの周波数
4	440Hz
5	880Hz
6	1760Hz
7	3520Hz

ここで、もう一度、「カエルのうた」の楽譜データを見てみましょう。

プログラム「RTTTL_Kaeru」より抜粋

```
// 楽譜データ
char *song = "kaeru:d=4,o=4,b=200:4c,4d,4e,4f,4e,4d,2c,4e,4f,4g,4a,4g,4f
,2e,4c,4p,4c,4p,4c,4p,4c,4p,8c,8c,8d,8d,8e,8e,8f,8f,4e,4d,1c";
```

音符のデータが「4c」「4d」などとありますが、これはオクターブの記述を省略していて、それぞれ「4c4」「4d4」と同じ意味になります。

では、「カエルのうた」の楽譜データを音符と音階にした表で確認してみましょう。

なお、ドは正確には  ですが、ここでは説明のため  のように表記しています。

表1-3　カエルのうたの音符

4c	4d	4e	4f	4e	4d	2c	4e	4f	4g	4a
										
ド	レ	ミ	ファ	ミ	レ	ド	ミ	ファ	ソ	ラ
4g	4f	2e	4c	4p	4c	4p	4c	4p	4c	4p
										
ソ	ファ	ミ	ド	休	ド	休	ド	休	ド	休
8c	8c	8d	8d	8e	8e	8f	8f	4e	4d	1c
										
ド	ド	レ	レ	ミ	ミ	ファ	ファ	ミ	レ	ド

やってみよう！

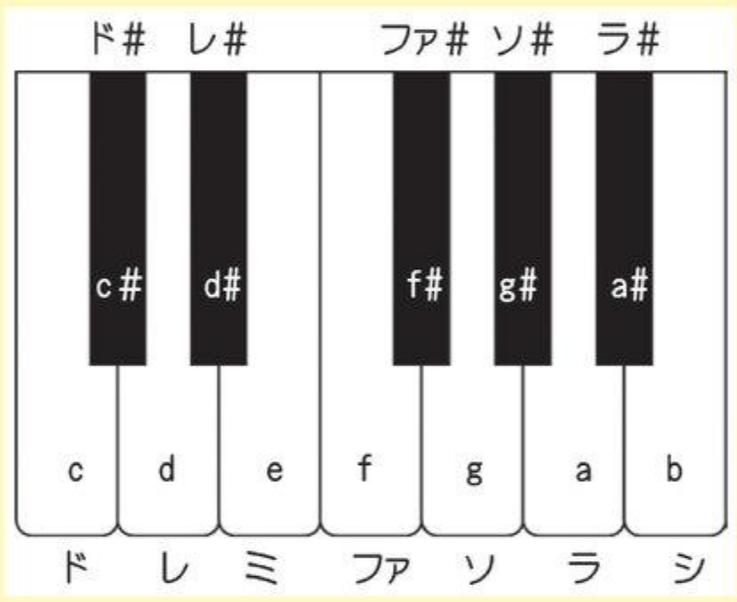
1. 「カエルのうた」の楽譜(がくふ)をアレンジしてみよう。音程やテンポを変えるとどうなるかな？具体的には、`o=4` や `b=200` の数字を変えてみよう。ただし、**o**は4～7、**b**は1～32767までの範囲しか取れないので注意しよう。

2. 「カエルのうた」の楽譜(がくふ)データから、`char *song = "kaeru:d=4,o=4,b=200:";` だけ残して、音符(おんぷ)データを全部消してしまおう。そして、音符(おんぷ)をはじめから打ち直してみよう。もとと同じ曲が流れたら、合格だ！

ヒント

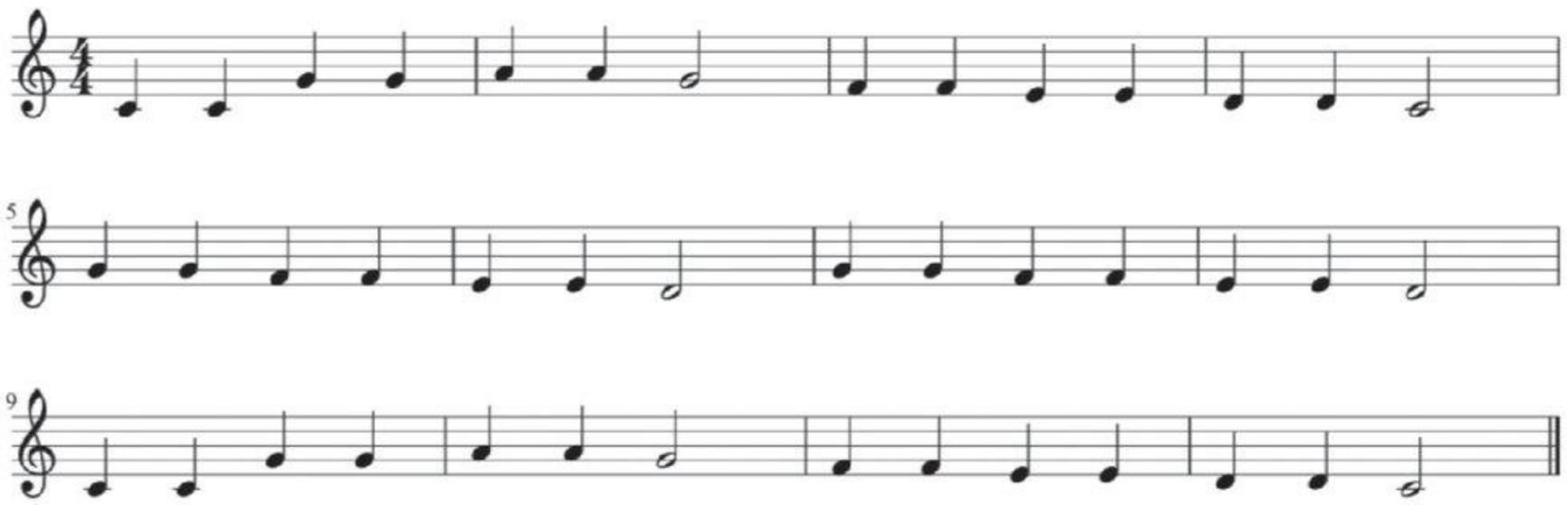
1. **o**は音程（octave）だよ。数字を大きくすると音程が高くなるよ。
   bはテンポ（beat）だよ。数字を大きくするとテンポが上がるよ。

2. 下の鍵盤(けんばん)の図と**表1-3**を参考にして曲をつくってみよう。



チャレンジ課題

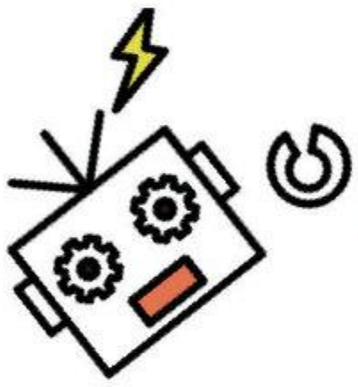
次の楽譜(がくふ)の曲名が何かわかるかな？ 楽譜(がくふ)を読み取って、プログラムに入力してみよう。
正しく入力できれば、何の曲かわかるはずだよ。



ヒント

楽譜(がくふ)に音の長さ（音符(おんぷ)）と高さをふっておくと、間違いが少なくなるよ。

4ド、4ド、4ソ、4ソ、4ラ、4ラ、2ソ、4ファ、4ファ、4ミ、4ミ、4レ、4レ、2ド
4ソ、4ソ、4ファ、4ファ、4ミ、4ミ、2レ、4ソ、4ソ、4ファ、4ファ、4ミ、4ミ、2レ
4ド、4ド、4ソ、4ソ、4ラ、4ラ、2ソ、4ファ、4ファ、4ミ、4ミ、4レ、4レ、2ド



RTTTLで曲をつくる方法をマスターできたかな？
次はちがう方法をマスターして曲をつくっていくよ！

講

解答は「きらきら星」です。解答プログラムは以下となります。
RoboticsProfessorCourse1 ＞ MagicItemA3 ＞ RTTTL_Challenge
なお、チャレンジ課題は、授業時間に余裕があったときに取り組むための応用問題です。実施する場合は、授業時間を考慮して制限時間を設けたうえで行ってください。チャレンジ課題に没頭するあまり、授業の後半パートが消化できないという事態にならないようご注意ください。

1.4. 周波数と音の関係を知ろう

1）プログラムの実行

RTTTL形式以外の命令も学んでおきましょう。
まずは、「周波数」をベースにしたものです。

プログラムの書き込み
RoboticsProfessorCourse1 ＞ MagicItemA3 ＞ Tone1

実行結果：音が1つだけ流れる。
特に変わったところのない、ただの音ですね。この、ただの音を使って、音が出るしくみについて考えてみます。
プログラムの中を見てみましょう。

プログラム「Tone1」より抜粋（ばっすい）

```
tone1.play(440, 500);                    // 440Hzの音を出す
```

この1行が、先ほどの音の正体です。「440」と「500」という数字が使われています。
「440」は「440Hz（ヘルツ）の音」であることを示しています。一方、「500」は音の長さを示しています。次のページでくわしく説明しますが、まずは好きにいじってみましょう。

やってみよう！

プログラム「Tone1」のHz（ヘルツ）の数字「440」を「880」に変えてみよう。
音はどのように変わるかな？　ほかにも数字をいろいろ変えて確かめてみよう。

講　1オクターブ上がることを確認させてください。なお、スピーカーの性能により、3520Hzまでしか音は出せません。

2）周波数について

さて、前のページで出てきた「Hz（ヘルツ）」について説明します。
「Hz」は、周波数に使われる単位です。スピーカーから音が出るときは、電圧のオンとオフを１秒間に何度もくり返して空気を振動させます。この1秒間に振動する回数のことを、周波数というのです。440Hzであれば、1秒間に440回オンとオフをくり返していることになります。
空気がはやく振動するほど、つまり周波数が大きいほど音は高くなります。
音と周波数の関係を一覧表にまとめました。

表1-4　音と周波数の関係

音名（アメリカ式）	音名（イタリア式）	周波数（Hz）
A	ラ	440
B	シ	494
C	ド	523
D	レ	587
E	ミ	659
F	ファ	698
G	ソ	784

440Hzの音は「ラ」ですが、その1オクターブ上の「ラ」は880Hzになります。つまり、1オクターブ上がるとき、周波数は2倍になっているということになります。

豆知識

ギターやバイオリンなどの弦楽器は細い弦ほど高音が出せますが、これは細い弦の方が軽く、すばやく振動させることができるためです。輪ゴムをはじくときも、輪ゴムがたるんでいるときより、ピンと張られているときのほうがすばやく振動できるため、高い音が出ますね。
ちなみに、人間の耳は20Hzから2万Hz程度の音を聞くことができます。

コラム　周波数と音階について

周波数とは、１秒間に何回振動するかを表したもので、単位はヘルツ（Hz）を用います。
20Hzから2万Hzぐらいの空気の振動が伝わる波を、人間の耳は「音」として聞いています。
2万Hzより周波数が大きい音波は「超音波」、20Hzより周波数が小さい音は「超低周波音」とよばれます。
ちなみに、五線譜の「ラ」の音の周波数はちょうど440Hz、1オクターブ上の「ラ」は880Hzになりますが、テレビなどでおなじみの「ピッ、ピッ、ピッ、ポーン」という時報には、この2種類の音からできているものもあります。興味がある人はプログラムを変更して時報音をつくってみましょう。

1.5. ミュージカル・ノートで曲をつくってみよう

周波数を入力して、うまく音を鳴らせましたか？ ただ、これから自由に作曲をしようにも、音名に対応する周波数をその都度入力するのは面倒(めんどう)だと思いませんか？ そう思った人にもっと良い方法を教えます。以下のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse1 > MagicItemA3 > Tone2

実行結果：「ドレミファソラシド」が流れる。

先ほどつくったプログラムと同じように動作しました。では、プログラムの中を見てみましょう。

プログラム「Tone2」より抜粋(ばっすい)

```
tone1.play(NOTE_C4, 500);     // C4(ドの音)を500ミリ秒出す
delay(500);
tone1.play(NOTE_D4, 500);     // D4(レの音)を500ミリ秒出す
delay(500);
tone1.play(NOTE_E4, 500);     // E4(ミの音)を500ミリ秒出す
delay(500);
tone1.play(NOTE_F4, 500);     // F4(ファの音)を500ミリ秒出す
delay(500);
tone1.play(NOTE_G4, 500);     // G4(ソの音)を500ミリ秒出す
delay(500);
tone1.play(NOTE_A4, 500);     // A4(ラの音)を500ミリ秒出す
delay(500);
tone1.play(NOTE_B4, 500);     // B4(シの音)を500ミリ秒出す
delay(500);
tone1.play(NOTE_C5, 500);     // C5(1オクターブ上のドの音)を500ミリ秒出す
delay(500);
```

周波数が入るべきところに、NOTE_C4 や NOTE_D4 が入っています。この暗号は、「ミュージカル・ノート」といい、自動的に周波数に置きかえてくれる、便利な記号なのです。記述のルールは簡単です。

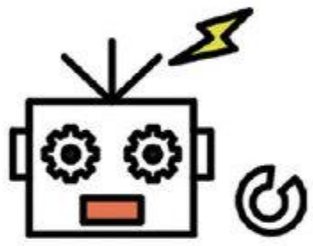
POINT

ミュージカル・ノートは（NOTE_音名とオクターブ番号,音の長さ）という順に記述します。

音と周波数の関係表に、ミュージカル・ノートを追加しました。周波数を記憶するより、感覚的に作曲できるでしょう。

表1-5　ミュージカル・ノート

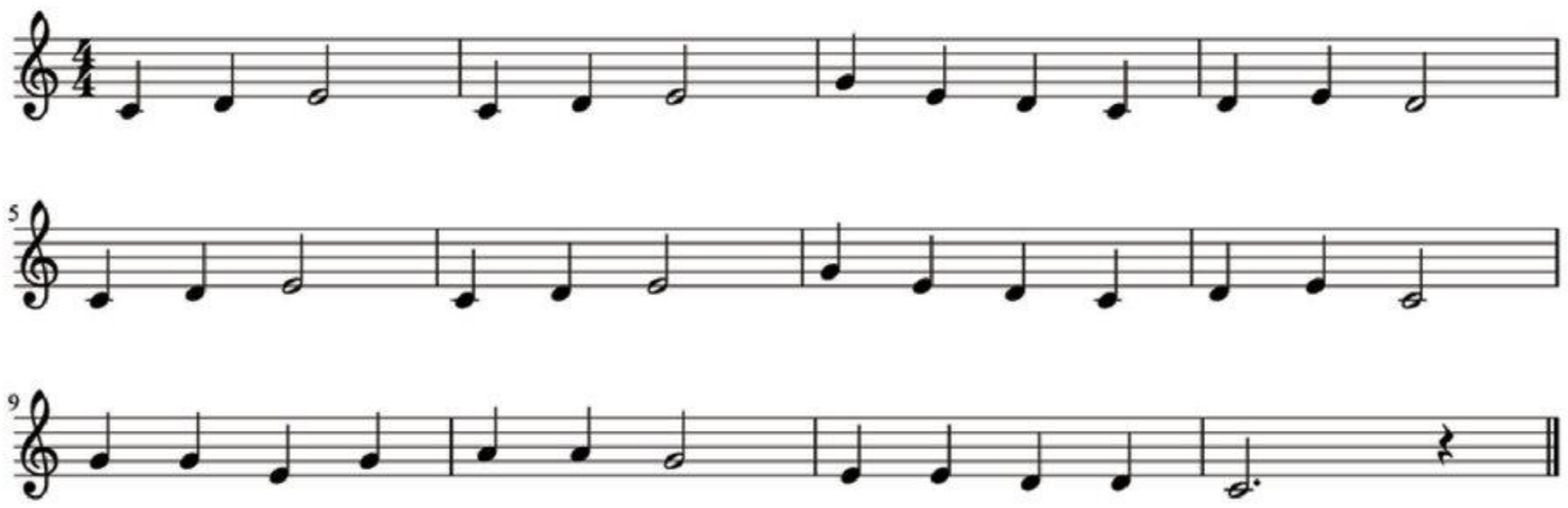
音名	音名	周波数（Hz）	ミュージカル・ノート
A	ラ	440	NOTE_A4
A#		466	NOTE_AS4
B	シ	494	NOTE_B4
C	ド	523	NOTE_C5
C#		554	NOTE_CS5
D	レ	587	NOTE_D5
D#		622	NOTE_DS5
E	ミ	659	NOTE_E5
F	ファ	698	NOTE_F5
F#		740	NOTE_FS5
G	ソ	784	NOTE_G5
G#		831	NOTE_GS5



なんだか音楽の授業みたいダヨー。でもロボットも歌いたいときもあるのさ。ぼくに歌を教えて！！

チャレンジ課題

ミュージカル・ノートを使って、楽譜から楽曲データ「チューリップ」を打ち込んでみよう。次の楽譜を参考にチャレンジしてみよう。



ヒント

楽譜に音の長さ（音符）と高さをふっておくと、間違いが少なくなるよ。

4ド、4レ、2ミ、4ド、4レ、2ミ、4ソ、4ミ、4レ、4ド、4レ、4ミ、2レ
4ド、4レ、2ミ、4ド、4レ、2ミ、4ソ、4ミ、4レ、4ド、4レ、4ミ、2ド
4ソ、4ソ、4ミ、4ソ、4ラ、4ラ、2ソ、4ミ、4ミ、4レ、4レ、2ド

講

時間制限を設けて、できるところまで実施させてください。解答プログラムは以下となります。
RoboticsProfessorCourse1 ＞ MagicItemA3 ＞ Tone2_Challenge
解答例は巻末に記載します。

ここまでで、曲を流すための命令を3種類学ぶことができました。
しかし、よく考えてみると、命令の書き方がちがうだけですね。たとえば、下の図はどれも同じ音を出す命令になっています。

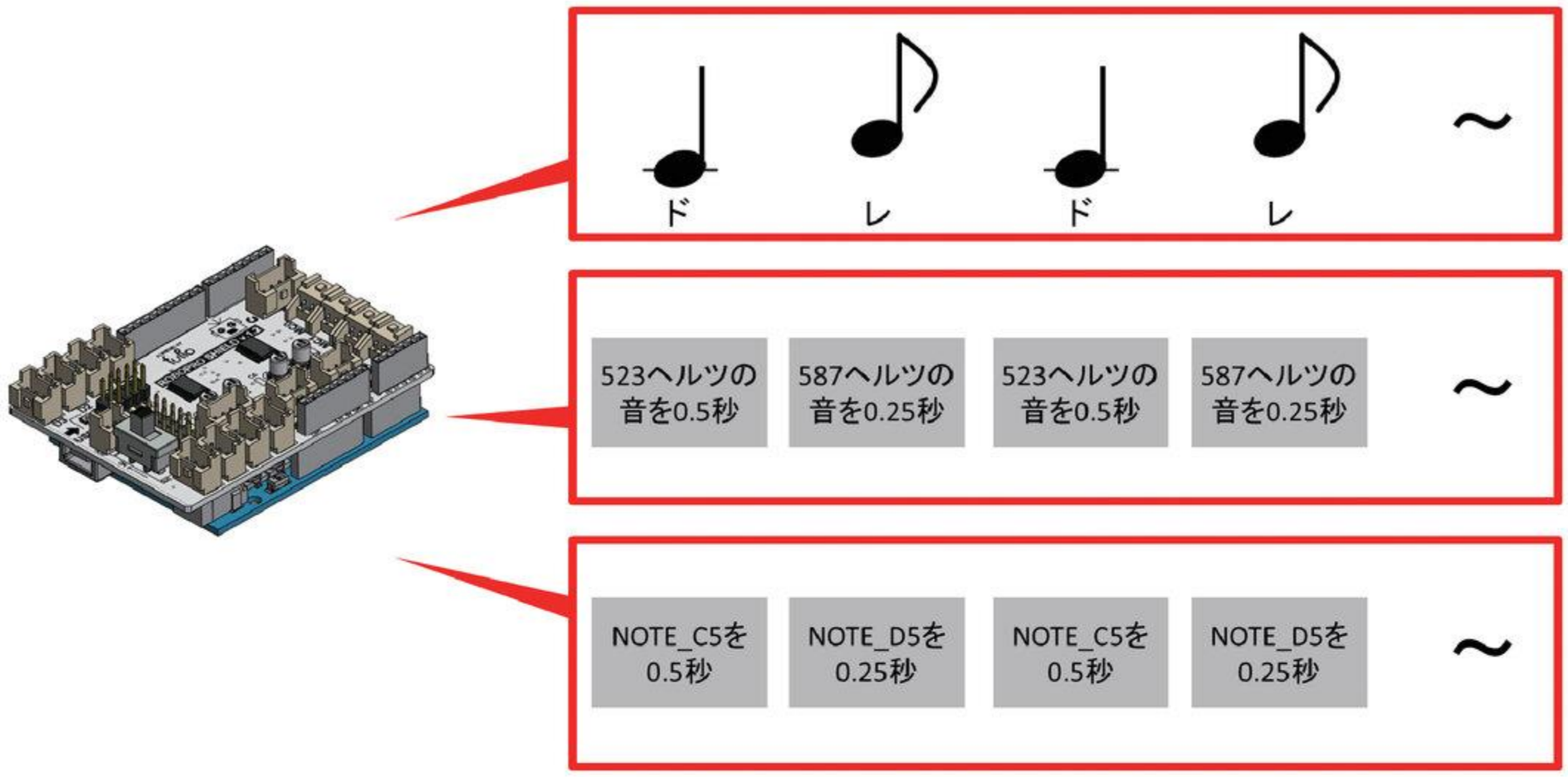


図1-2　曲を流すための3種類の命令

どの命令も、長所と短所があることが感じられたのではないでしょうか。
3つとも使えるようにしておいて、「今回はどの書き方がいちばん向いているかな?」と考えられるようになると、効率のいいプログラミングができますね。
このように、答えが1つでも、工夫しだいで途中の手間を減らすことができるのもプログラミングの特徴です。

1.6 . マイコンユニットを楽器（スイッチシンセサイザー）にしよう

1）タッチセンサーを使って音を鳴らそう

続いて、タッチセンサーを使いましょう。次のプログラムを実行してください。実行後、2個のタッチセンサーを押してみてください。今度は長めに押し続けてみましょう。

プログラムの書き込み
RoboticsProfessorCourse1 > MagicItemA3 > Tone3

実行結果：どちらのタッチセンサーも押している間、音が出続ける。
これでだいぶ、楽器らしくなってきました。それではプログラムの中を見てみましょう。

プログラム「Tone3」より抜粋（ばっすい）

```
if(SW1.isPressed()){                // もしSW1が押されたら
    tone1.play(NOTE_C4, 100);       // C4(ドの音)を100ミリ秒出す
    delay(100);
}
if(SW2.isPressed()){                // もしSW2が押されたら
    tone1.play(NOTE_D4, 100);       // D4(レの音)を100ミリ秒出す
    delay(100);
}
```

前回の授業で学んだ、if文を思い出して、このプログラムを読み解いてみましょう。
SW1（D0 につながったタッチセンサー）が押されると、ドの音が出ます。また、SW2（D1 につながったタッチセンサー）が押されると、レの音が出ます。
isPressed()というのは、「押されている間中～」という意味です。

やってみよう！

プログラム「Tone3」のミュージカル・ノートの値を変えて、音階（おんかい）を変えてみよう。できたら、音の長さも変えてみよう。

2）音階をつけよう

では、次に音階をつけてみましょう。以下のプログラムを実行してください。

プログラムの書き込み

RoboticsProfessorCourse1 > MagicItemA3 > Tone4

実行結果：D0 につながったタッチセンサーを押すと、マトリクスLEDの表示が変わる。
D1 につながったタッチセンサーを押すと、表示されている音階の音が出る。

先ほどのプログラムTone3と似ていますが、タッチセンサー2個で、２音だけでなく8音も出せるプログラムです。D0 に接続されたタッチセンサーで音階を変えて、D1 に接続したタッチセンサーで音を出します。そして、わかりやすいように、マトリクスLEDにミュージカル・ノートのコードも表示されます。

チャレンジ課題

マトリクスLEDを隠して、D0 タッチセンサーを適当に連打して、D1 タッチセンサーを押そう。何の音かわかるかな？ 友達に音を聞かせてクイズを出してみよう。わかったら、絶対音感があるかもしれないぞ！

講

クラスの生徒全員参加で、クイズ形式で進めても楽しめます。
その場合は、先生が出題してください。

2. コントローラーで遊ぼう（目安 40 分）

2.0. コントローラーの接続と設定

続いては、コントローラーを使用します。

1）無線受信モジュールの取り付け

ロボプロシールドからタッチセンサー2個を外します（スピーカーは残します）。そして、ロボプロシールドの CN9 のコネクターにリボンケーブルを取り付けます。ケーブルの向きは、赤いラインがマトリクスLED側にくるようにしましょう。
リボンケーブルのもう一方の先には、無線受信モジュールを取り付けます。赤いラインは、**図2-1**の向きになるようにしましょう。

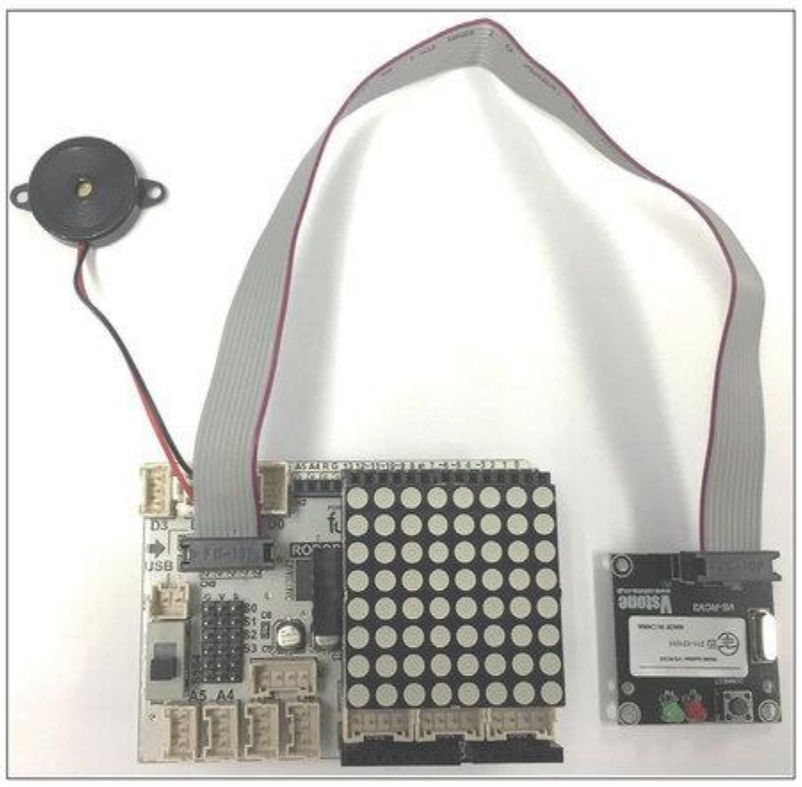


図2-0　ロボプロシールドと無線受信モジュールの接続完成図

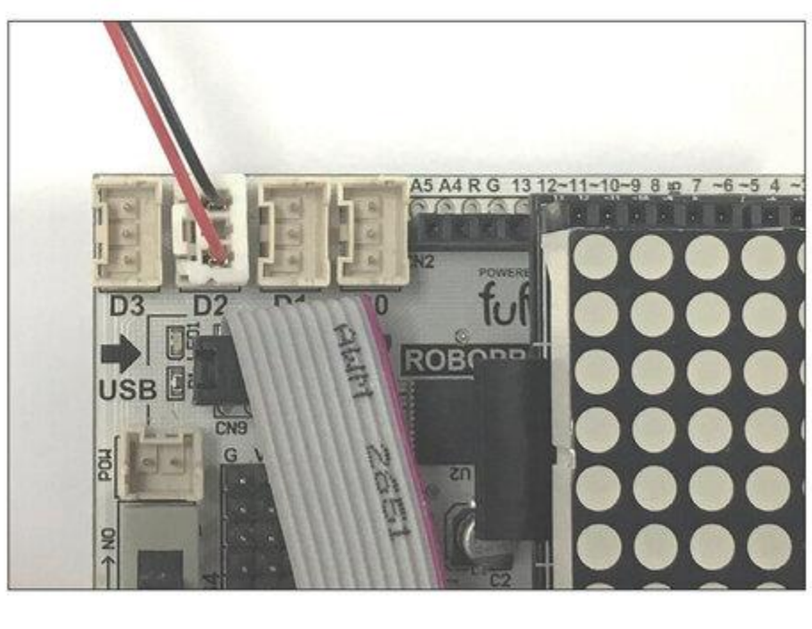


図2-1　リボンケーブルの向き

正しく接続されていれば、マイコンユニットをパソコンにつないだときに、無線受信モジュールの緑色LEDが点灯し、赤色LEDは点滅（てんめつ）しているはずです。

講　タッチセンサーを必ず取り外させてください。タッチセンサーがロボプロシールドに接続されたままだと正しく動作しません。

2）ペアリング

POINT

コントローラーと無線受信モジュールを1組のペアに設定し、通信ができるように設定することを、ペアリングといいます。

●ペアリングのやり方

まず､無線受信モジュールの黒い「CONNECT」ボタンを押します。次に､コントローラーの裏面の「POWER」をON にし、表面の「ANALOG」ボタンを押してください。

・無線受信モジュールの赤色LED が、点滅から点灯に変わる

・コントローラーの緑色LED と赤色LED が、点滅から点灯に変わる

状態になれば、ペアリング成功です。

●コントローラー使用時の注意点

コントローラーはしばらく放っておくと、自動的に接続が切れて、無線受信モジュールの赤色LEDが点滅に変わり､コントローラーのLED は消灯します。その場合は「START」ボタンを押せば、再びペアリングの状態になります。

2.1. コントローラー機能を試してみよう

ペアリングが完了したら、次のプログラムを書き込んでください。

プログラムの書き込み

RoboticsProfessorCourse1 > MagicItemA3 > MatrixPStest1

実行結果：コントローラーのボタンを押すと、マトリクスLEDに次のようなアルファベットが表示される。

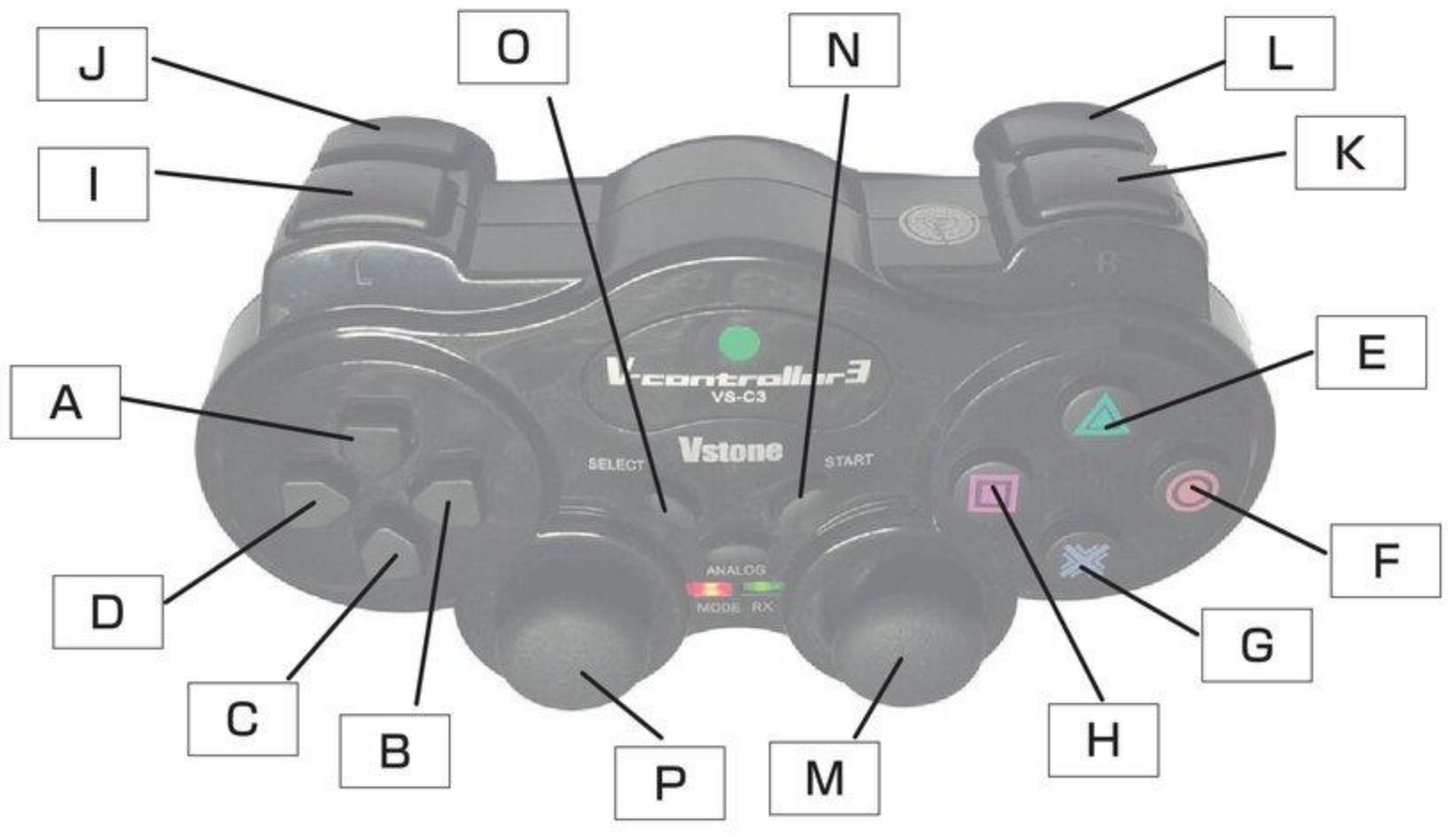


図2-2　アルファベットの表示

2.2. バイブレーション機能を試してみよう

次に、コントローラーの裏面の「VIBRATION」をONにし、以下のプログラムを実行してください。実行後にコントローラーの右のアナログスティックを上に動かしてみましょう。

プログラムの書き込み
RoboticsProfessorCourse1 > MagicItemA3 > MatrixPStest2

実行結果：コントローラーがふるえるとともに、マトリクスLEDに01～99の数字が表示される。スティックを倒す角度によって、数字が変化することがわかると思います。中途半端（ちゅうとはんぱ）な数字のまま、とどまることが難しいでしょう。これが、0/1のデジタルにはない、微妙（びみょう）なアナログ感です！

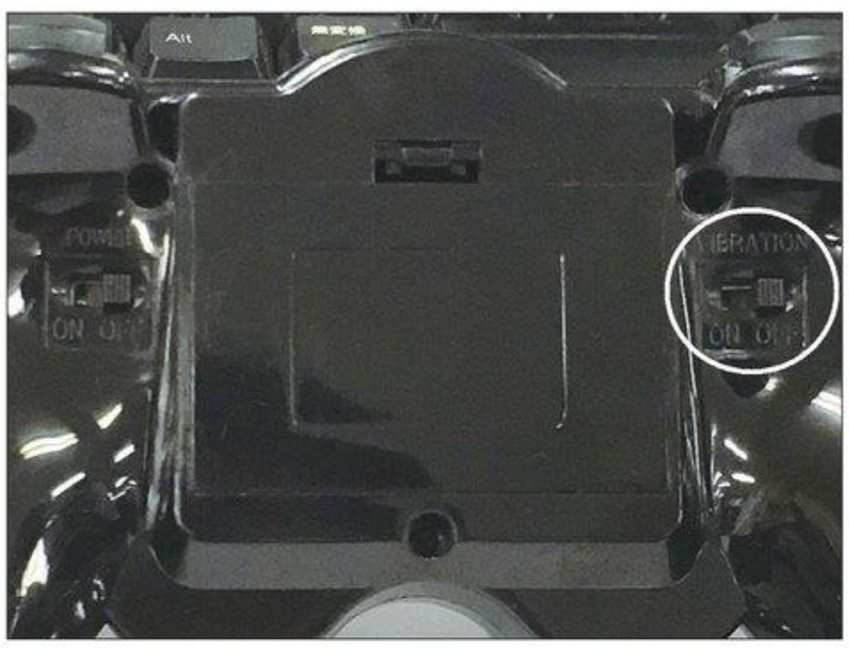


図2-3　コントローラーの裏面

図2-4　右アナログスティックの位置

では、次はどうでしょうか？ 以下のプログラムを実行して、同じく右のアナログスティックを上に動かしてみましょう。

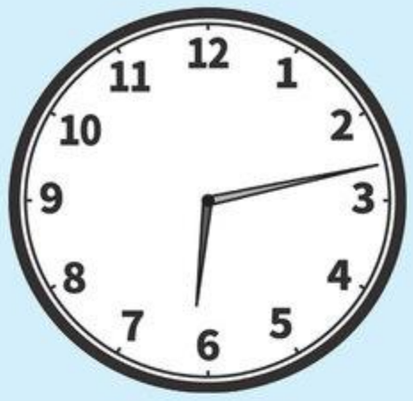
プログラムの書き込み
RoboticsProfessorCourse1 > MagicItemA3 > TonePS1

実行結果：「MatrixPStest2」の実行結果に音の効果が加わる。スティックを上に倒す角度によって音が変化する。

こちらの方が、数字とともに音も微妙（びみょう）に変化していくので、アナログ感がよりわかりやすいですね。

コラム　アナログとデジタルについて

「アナログとデジタルとは何か」を、アナログ時計とデジタル時計をもとに考えてみましょう。



デジタル時計は、秒の位がなければ「18時13分の何秒くらいか」は読み取れません。いっぽう、アナログ時計であれば秒針がなくても「13分と14分のちょうど間くらいに長針があるから、18時13分30秒くらいだな」と読み取ることができます。このように、デジタルは「ある値から次の値に一気に変化し、間は読み取れないもの（離散量といいます）」、アナログは「ある値から次の値に絶え間なく変化し、間が読み取れるもの（連続量といいます）」をさします。

「虹」は赤から少しずつ色が変わっていって、いつの間にかオレンジになっていますよね。つまり、正確には「虹の色は無限に存在する」ということになります。これは、アナログの考え方です。しかし「虹の絵をかきなさい」と言われたら、日本であれば７つの色にむりやり分別して虹をかく人が多いはずです。つまり、アナログだったはずの虹を、デジタルに変換したことになります。コンピューターもこれと同じように、本来アナログだったものをデジタルデータ化していることが多いのです。今回あつかった「音」もその一つで、音の高さは無限に分けることができるアナログのデータですが、コンピューターに取り込む際には何段階かの周波数にむりやり分別するなどしてデジタルデータに変換しています。音の種類が限られるのでデータ全体の容量が少なく済み、コピーがしやすいという利点がありますが、もととは違う音に変えられているので音楽としては「劣化」したことになります。

円盤に小さな凹凸をつけて回転させ、針を当ててひっかくことで音を鳴らす「レコード盤」を知っていますか。物理的な凹凸というアナログデータなので音が劣化しないことから、現在も「本来の音で音楽を楽しみたい」という人に親しまれています。また、そろばんは昔の道具でいかにもアナログなイメージがあるかもしれませんが、「玉が上がっているか、下がっているか」という2段階で区別をするため、実はデジタル機器といえます。

「アナログよりデジタルのほうが新しい、優れている」「デジタル機器は電気で動かす」と思っている人もいるかもしれませんが、実はそうとも限らないのです。先入観にとらわれず、いろいろな考え方を吸収していきましょう。

2.3. コントローラーで操作してみよう

1）十字キーを操作してみよう

続いて以下のプログラムを実行して、コントローラーの十字キーを動かしてみましょう。

プログラムの書き込み
RoboticsProfessorCourse1 > MagicItemA3 > LedMovePS1

実行結果：十字キーを押した方向に、マトリクスLED上の点が動く。

図2-5　十字キーの位置

では、次はどうでしょうか？ 以下のプログラムを実行してください。

プログラムの書き込み
RoboticsProfessorCourse1 > MagicItemA3 > LedMovePS2

実行結果：「LedMovePS1」の実行結果に音の効果が加わる。
点の動きに合わせて音が鳴りましたね。縦方向と横方向では、音がちがうようですね。

ゲームみたいになってきたって思った？ 実はゲームのプログラムとロボットのプログラムは似ているんダヨ。

では、先ほどのプログラムの中を見てみましょう。

プログラム「LedMovePS2」より抜粋（ばっすい）

```
ps2x.read_gamepad();                    // コントローラーから値を読み取る

if (ps2x.Button(PSB_PAD_UP)){           // もし十字ボタンの上が押されたら
    y--;
    tone1.play(NOTE_E5, 10);            // E5(ミの音)を10ミリ秒出す
}
if (ps2x.Button(PSB_PAD_RIGHT)){        // もし十字ボタンの右が押されたら
    x++;
    tone1.play(NOTE_E6, 10);            // E6(1オクターブ上のミの音)を10ミリ秒出す
}
if (ps2x.Button(PSB_PAD_LEFT)){         // もし十字ボタンの左が押されたら
    x--;
    tone1.play(NOTE_E6, 10);            // E6(1オクターブ上のミの音)を10ミリ秒出す
}
if (ps2x.Button(PSB_PAD_DOWN)){         // もし十字ボタンの下が押されたら
    y++;
    tone1.play(NOTE_E5, 10);            // E5(ミの音)を10ミリ秒出す
}
```

プログラムの中にいくつかのミュージカル・ノートを見つけることができます。十字キーを操作したときに出る音を決めているのがその場所です。

やってみよう！

プログラム「LedMovePS2」のミュージカル・ノートの値を変えて、自分の好きな音を出してみよう。上下左右、ちがう音を入れると、ちょっとした楽器になるよ。

2）constrain命令とは

さて、先ほどのプログラムをもっと下まで見ると、こんなコードがあります。

プログラム「LedMovePS2」より抜粋（ばっすい）

```
// xとyが画面からはみ出ないように制限する(constrain)
x = constrain(x, 0, 7);     // xの値を0から7の数字におさめる
y = constrain(y, 0, 7);     // yの値を0から7の数字におさめる
```

「constrain」には、「制約する」、「束縛（そくばく）する」、などといった意味があります。

命　　令「constrain」
実行結果：動く範囲を指定する
使 い 方：x = constrain(x, 0, 7);
　　// 変数xの値が0未満ならば0に、7より大きければ7に強制的に変更する

たとえば右キーを押すと変数 x の値が増えていきますが、マトリクスLEDのx座標は最大7なので、変数 x が8以上になってしまうと画面に表示できなくなってしまいますね。constrain を使うことで、表示する座標が0 ～ 7の範囲におさまるようにしています。

ステップアップ

プログラム「LedMovePS2」の最後にあるconstrain文を変更(へんこう)して、移動する点がマトリクスLEDの右半分の4列、および下の2行に行かないようにしてみよう。

講　解答例は以下の通りです。
「x = constrain(x, 0, 7);」を「x = constrain(x, 0, 3);」に、
「y = constrain(y, 0, 7);」を「y = constrain(y, 0, 5);」に変える。

2.4.　if else 文をマスターしよう

1）OR表現とは

さらにあたらしい文法を学びましょう。まずは、以下のプログラムを実行してください。
実行後に、先ほどと同じように十字キーを操作してみましょう。

プログラムの書き込み
RoboticsProfessorCourse1 > MagicItemA3 > SpriteMovePS1

実行結果：十字キーを押した方向にマトリクスLEDの「＋印」が動く。「＋印」が壁にぶつかると、コントローラーがふるえる。

講　コントローラー裏の「VIBRATION」スイッチを ON にする必要があります。

プログラムの中を見てみましょう。バイブレーションの動く範囲を決めているのは、以下の部分です。

プログラム「SpriteMovePS1」より抜粋

```
if (x > 5 || x < 0 || y > 5 || y < 0){
                                    // if文応用編（もし＋印の位置が指定範囲の外だったら）
    vibrate = 100;                  // コントローラーを振動させる
}
else {                              // そうでなければ
    vibrate = 0;                    // コントローラーの振動を止める
}
```

これは前回の授業で覚えたif文ですね。ただし、ifの条件が複雑になっています。「もし、○○だったら」の○○の部分が `x > 5 || x < 0 || y > 5 || y < 0` と、4つも条件らしき項目があります。「||」は「OR（または）」という意味です。
ですからこの部分は、「x>5またはx<0またはy>5またはy<0」になったときに、バイブレーションを動かす命令となっています（ここでのxとyは「＋印」の左上の点のx,y座標を指します）。
簡単にいうと、4つの条件のどれかが起きたときに実行してね、ということです。

2）elseについて

「else」は、「その他」とか「他の方法」といった意味で、ifと共に使われます。if文は、「もし、○○だったら、□□する」という命令でしたよね。「else」は逆に、「もしも○○ではなかったら、■■する」という命令を実現したいときに使われます。

やってみよう！

「else」は本当に必要なのかを確かめてみよう。
プログラム「SpriteMovePS1」の中の、`else {vibrate = 0;}` を消して、if文だけにして実行してみよう。何が起きるかな？

講

解答：「+ 印」が壁にぶつかった後、コントローラーがずっとふるえます。
突然ふるえてもあわてる必要はありません。バイブレーション機能は、コントローラーの裏の「VIBRATION」スイッチを OFF にすれば止まります。

「else」がないと、大変なことになりました。if文だけでは、アクセルしかない自動車みたいなものです。ブレーキという「他の方法」もなければ、危険この上ないですよね。
条件による分岐(ぶんき)を示すプログラムには、if else文を使うようにしましょう。

命　　令　**「if( ){ A } else{ B }」**
実行結果：条件( )が満たされている場合、{ A }　内の処理を実行する
　　条件( )が満たされていない場合、{ B }　内の処理を実行する
使 い 方：if (条件) {
　　実行する内容 A
　　}
　　else {
　　実行する内容 B
　　}

POINT

if else文は、elseに続けてifを書くことで、2つの条件分岐(ぶんき)だけでなく、もっと多くの場合分けを行うことが可能になります。たとえば、このような構文になります。

```
if (条件①) {
実行する内容 A
}
else if (条件②) {
実行する内容 B
}
else {
実行する内容 C
}
```

チャレンジ課題

サンプルプログラム「SpriteMovePS1」のif else文の中に、もう一つif文（else if）を入れて、「＋印」が真ん中付近にきたときに、特別な音楽が流れるという効果を追加してみよう。

ヒント

たとえば、「もしx座標が0でy座標が6なら」というif文なら
`if(x == 0 && y == 6)` となるよ！
等号（イコール）が2つ連続することに注意しよう！

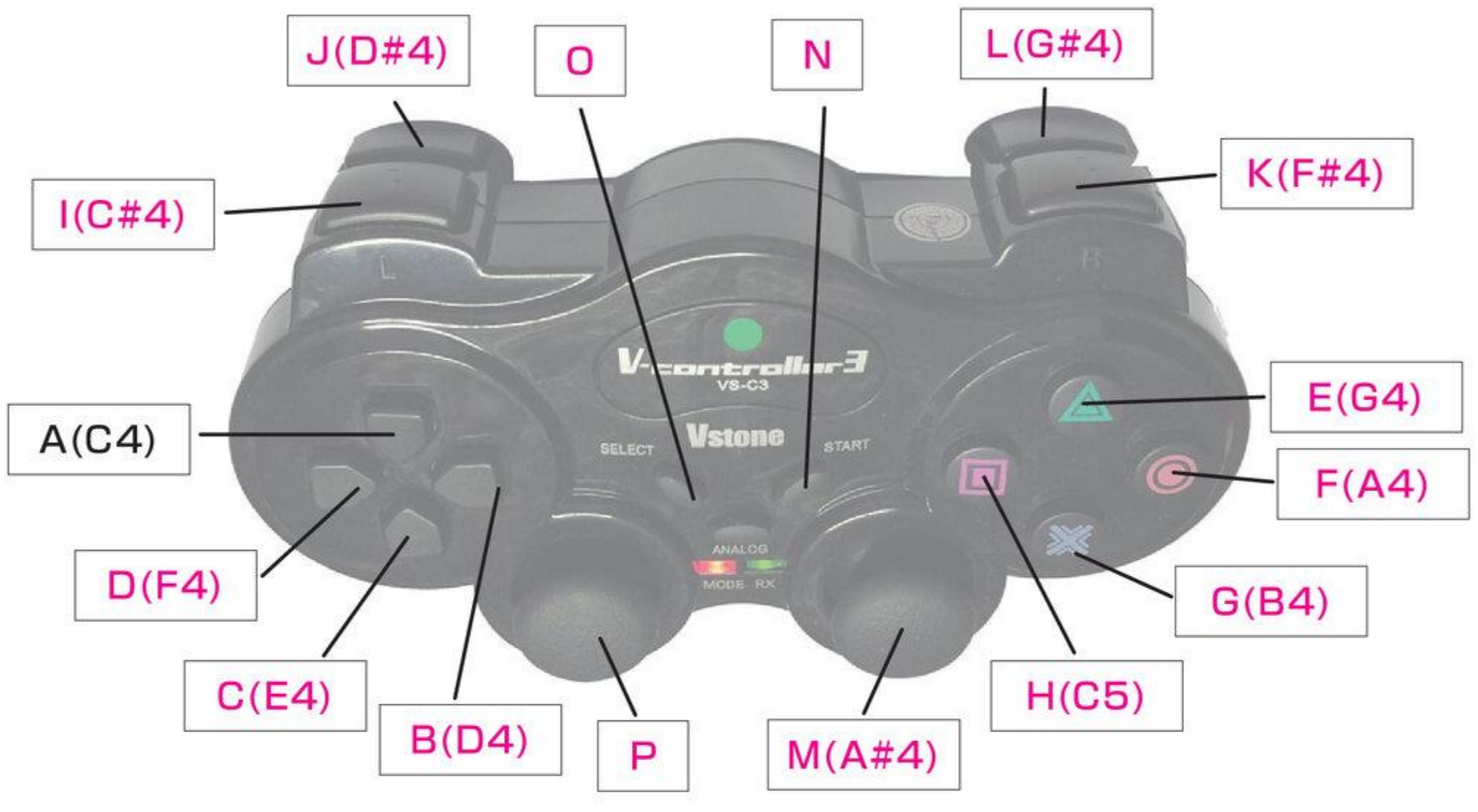
講　解答例プログラムは以下となります。
RoboticsProfessorCourse1 ＞ MagicItemA3 ＞ SpriteMovePS1_Challenge

チャレンジ課題

楽器として、コントローラーを使ってみよう。下のプログラムを実行して、コントローラーの各ボタンを押したときに、マトリクスLEDに表示されるアルファベットと、鳴る音階をそれぞれ次の図に書き込んでみよう。何の音階かわからないときは、プログラムの中を見てね。

プログラムの書き込み

RoboticsProfessorCourse1 > MagicItemA3 > TonePS2



講　左右のアナログスティックは押し込むことで反応します。
N、O、P のボタンは音は出ません。

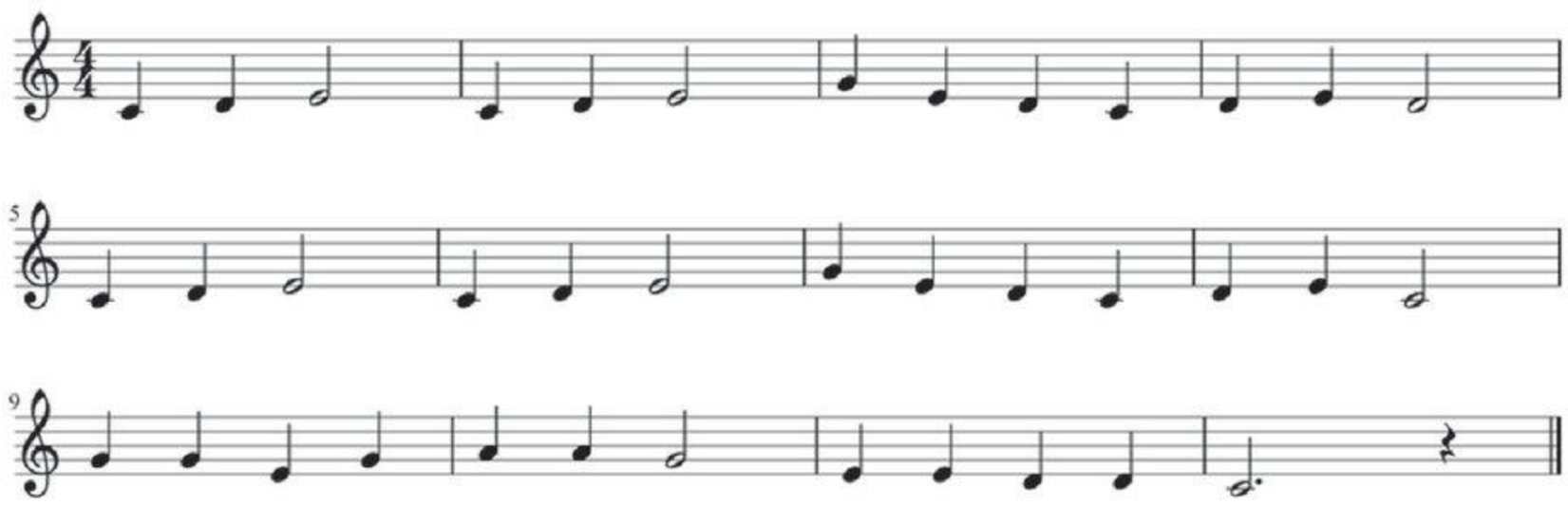
チャレンジ課題

コントローラーのボタンと音階の位置関係を変えて、自分好みの楽器につくりかえよう。自分の好きな音楽のコードを近くに配置すれば、アドリブも簡単になるよ。次のページの楽譜を使ってみてもいいよ。今まで自動演奏だった曲を自分でひいてみよう。

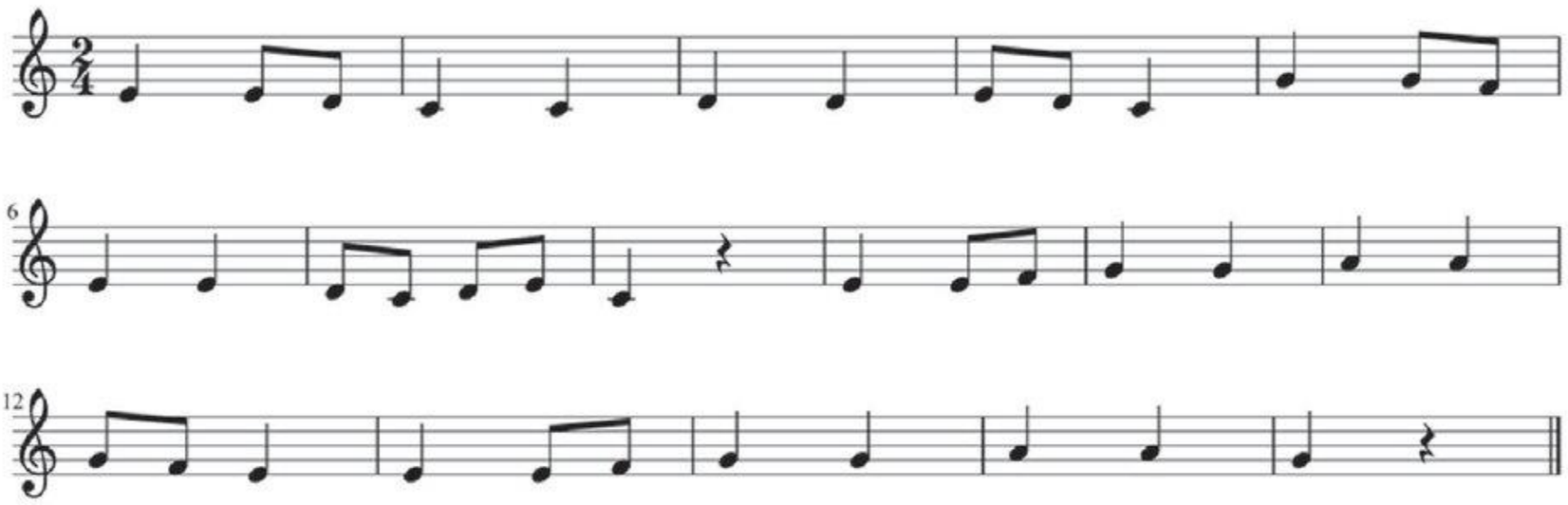
講　チャレンジ課題は、時間に余裕がある場合に実施してください。また問題は自由課題になりますので、時間を決めて実施してください。

「楽譜（がくふ）サンプル」

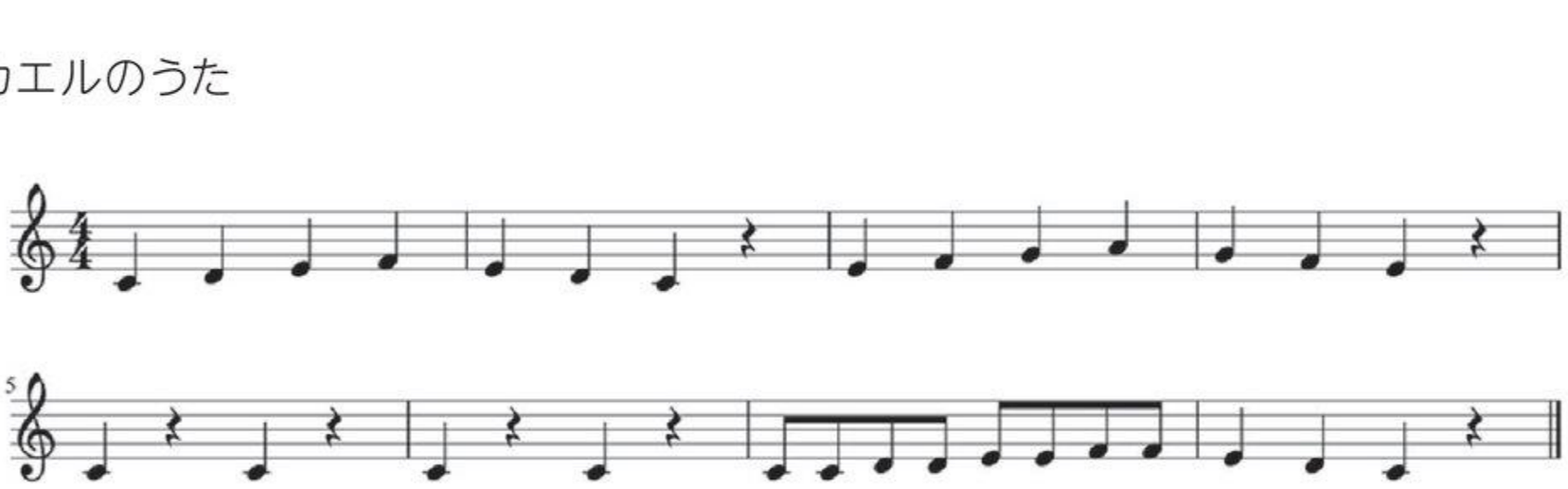
●チューリップ



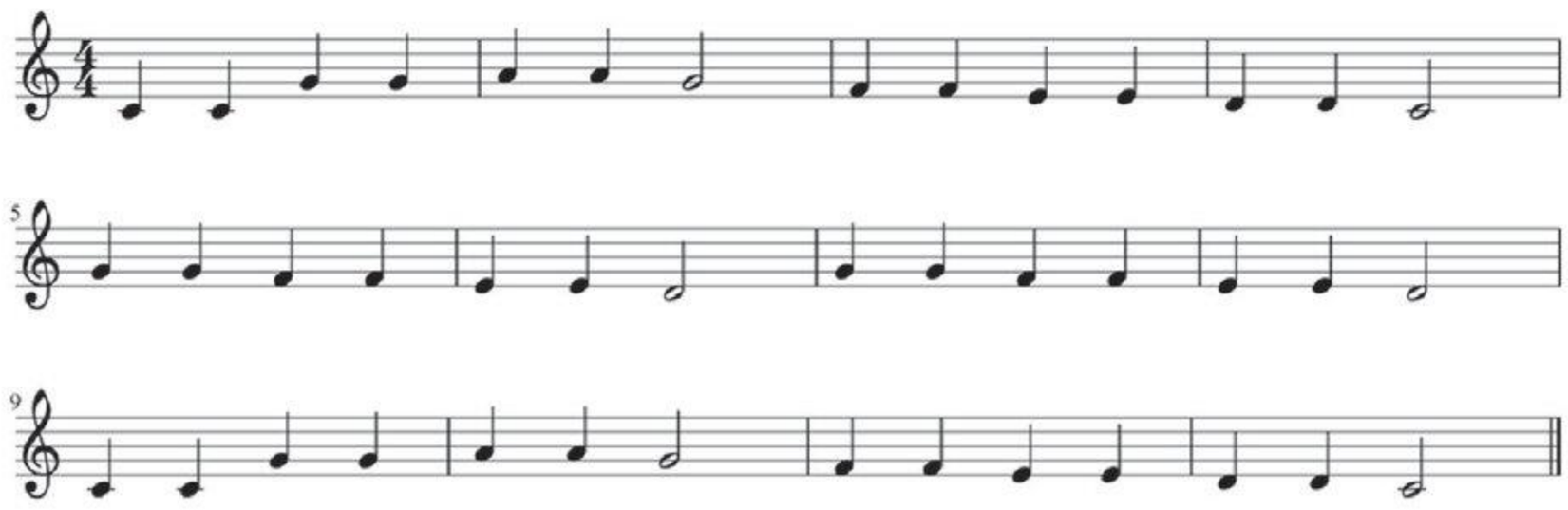
●むすんでひらいて



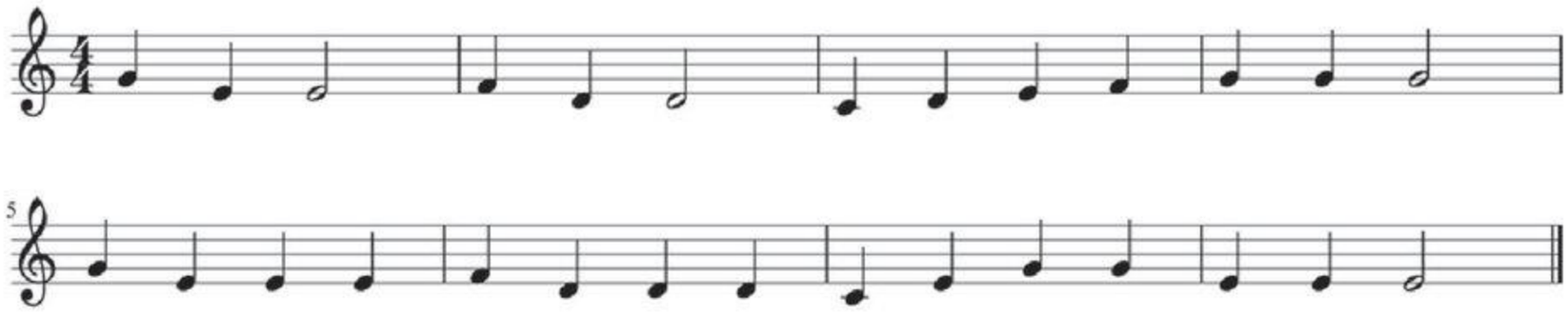
●カエルのうた



●きらきら星

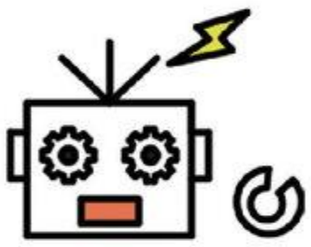


●ちょうちょう



3. まとめ（目安5分）

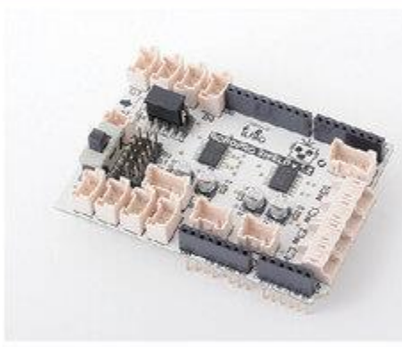
今回は、いろいろな音楽の演奏(えんそう)方法を学びました。音と周波数の関係を学んだり、ミュージカル・ノートでのプログラム方法も学習しました。また、コントローラーをまるで楽器のようにして演奏(えんそう)しました。
次回は、光を使って、アニメーションをつくりましょう。



光でつむがれるアニメーション。
なんて幻想的なセカイだろうか！

≪次回必要なもの≫

次回は、以下のパーツを持ってきてください。

USBケーブル 1	マイコンボード 1	ロボプロシールド 1	マトリクスLEDシールド 1
			

マトリクスLED 1


図3-0　次回必要なもの

講

○　以下の理解度を確認します。
・音楽をつくる 3 つの方法を学ぶ
・リモコン装置をカスタマイズする
・if else 文の使い方をマスターする

○　次回テーマは「CG（コンピューターグラフィックス）で遊ぶ」であることを告知します。引き続き、マイコンユニットを使います。

P.15　チャレンジ課題　解答例（void loop以下を抜粋）

```
void loop(){
    tone1.play(NOTE_C4, 500);    // C4(ドの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 500);    // D4(レの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_E4, 1000);   // E4(ミの音)を1000ミリ秒出す
    delay(1000);

    tone1.play(NOTE_C4, 500);    // C4(ドの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 500);    // D4(レの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_E4, 1000);   // E4(ミの音)を1000ミリ秒出す
    delay(1000);

    tone1.play(NOTE_G4, 500);    // G4(ソの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_E4, 500);    // E4(ミの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 500);    // D4(レの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_C4, 500);    // C4(ドの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 500);    // D4(レの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_E4, 500);    // E4(ミの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 1000);   // D4(レの音)を1000ミリ秒出す
    delay(1000);

    tone1.play(NOTE_C4, 500);    // C4(ドの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 500);    // D4(レの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_E4, 1000);   // E4(ミの音)を1000ミリ秒出す
    delay(1000);

    tone1.play(NOTE_C4, 500);    // C4(ドの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 500);    // D4(レの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_E4, 1000);   // E4(ミの音)を1000ミリ秒出す
    delay(1000);
```

```
    tone1.play(NOTE_G4, 500);    // G4(ソの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_E4, 500);    // E4(ミの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 500);    // D4(レの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_C4, 500);    // C4(ドの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 500);    // D4(レの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_E4, 500);    // E4(ミの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_C4, 1000);   // C4(ドの音)を1000ミリ秒出す
    delay(1000);

    tone1.play(NOTE_G4, 500);    // G4(ソの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_G4, 500);    // G4(ソの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_E4, 500);    // E4(ミの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_G4, 500);    // G4(ソの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_A4, 500);    // A4(ラの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_A4, 500);    // A4(ラの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_G4, 1000);   // G4(ソの音)を1000ミリ秒出す
    delay(1000);

    tone1.play(NOTE_E4, 500);    // E4(ミの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_E4, 500);    // E4(ミの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 500);    // D4(レの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_D4, 500);    // D4(レの音)を500ミリ秒出す
    delay(500);
    tone1.play(NOTE_C4, 1500);   // C4(ドの音)を1500ミリ秒出す
    delay(1500);
    while (1);                   // 終了(無限ループ)
}
```