Ruby x Arduino でシンセサイザーを作ってみた

*　　2015/03/28 浜松 Ruby 会議 01

*　　Hamamatsu.rb 石垣 良

デモ（Ruby シンセ）

---

*　<http://risgk.github.io/ruby-arduino-synth/ruby-arduino-synth-op.mp3>

Thank you for listening!

---

*　ご清聴ありがとうございました

本日の発表では

---

* マイコンボード「Arduino」用シンセサイザーのプロトタイプとして

* Ruby でソフトシンセを作った経験についてお話しします

自己紹介

---

* 石垣 良 @risgk

* Hamamatsu.rb メンバー

* 組込みソフトウェア開発の仕事

* Ruby は小さなプログラムを書くのに使用（経験 10 年とか言えない...）

いま、電子工作・DIY が熱い！

---

* Maker Faire は世界 100 か所、53 万人（2013 年、Mini を含む）

* FabLab という市民工房の広がり（2014 年 FabLab 浜松オープン）

* 2014 年 勤務先でクラブ活動「Yara:Makers（やらめいかー）」が開始

* <http://yaramakers.tumblr.com/>

* 注：遠州弁「やらまいか」は「一緒にやろう」「やってやろう」の意味

さて、私は何を作ろうか？

---

* Arduino Uno を使いたい（流行っている、一度触ってみたい）

* Ruby を使いたい（好きだから）

* シンセサイザーを作ろう（過去に DTM 経験、「音楽のまち」浜松にいる）

* Yara:Makers のモットー：「手段のためなら、目的を選ばない。」

「Arduino（アルデュイーノ）」とは？

---

* オープンソースの電子工作プラットフォーム

* 豊富なライブラリがあり、誰でもインタラクティブなデバイスが作れる

* C++ライクな「Arduino 言語」でプログラミング

* 基本となるマイコンボードが「Arduino Uno」（約 3000 円）

Arduino Uno のスペック

---

* CPU：Atmel AVR 16 MHz（8 ビット CPU）

* ROM：32 KB、RAM：2 KB

* AVR は Z80 などよりは遥かに強力（例：8 ビット乗算器の搭載）

* PWM 出力で 1 ビット・オーディオが再生可能

* しかし、mruby を動かすのは、かなり難しそう...

Ruby でプロトタイピングするのはどうか？

---

* Arduino の実機上で試行錯誤するのは効率が悪い

* Ruby で音響プログラミングの実験ができる（WAV ファイルを出力）

* Ruby は Arduino 用のコード生成にも役立つ

やってみた

---

* Ruby でソフトシンセを開発（リアルタイム再生は JRuby、Windows 専用）

* Arduino Uno に移植（C++のインライン展開機能を多用）

* Ruby 版と Arduino 版で「だいたい」同じ音が出せた

* JavaScript と Web MIDI API を使って、音色エディターを作成

完成品（Digital Synth VRA8）のスペック

---

* <https://github.com/risgk/DigitalSynthVRA8>

* サンプリングレート：15625 Hz、量子化ビット数：8、モノ（単音）

* MIDI 入力：gem unimidi／USB シリアル 38400 bps

* オーディオ出力：gem win32-sound／PWM 出力

* 出力には、Java Sound API や PortAudio を使えばよかったかも

アナログシンセサイザーの仕組み

---

* VCO（電圧制御オシレーター）：基本波形を生成、音の高さを変化

* VCF（電圧制御フィルター）：音色を変化

* VCA（電圧制御アンプ）：音量を変化

* EG（エンベロープジェネレーター）：音量や音色を時間変化（ADSR）

* LFO（低周波数オシレーター）：ビブラートなどの変調に使用

VRA8 の構成

```
                              [EG]
                               :
                               +........+
    [VCO 1]-+                  :        :
            |                  V        V
    [VCO 2]-+->[Mixer]--->[VCF]--->[VCA]--->
            |
    [VCO 3]-+
```

* LFO は存在しない

* 複数の VCO から「厚い音」や「デチューン効果」が得られる

`synth.rb` より

```
---------------
     class Synth
       def clock
         level = $mixer.clock($vco_1.clock, $vco_2.clock,
                              $vco_3.clock)
         eg_output = $eg.clock
         level = $vcf.clock(level, eg_output)
         level = $vca.clock(level, eg_output)
       end
     end
```

* サンプリング周期（15625 分の 1 秒）毎に呼ばれるメソッド

* コードを一部編集して引用（以下も同様）

`generate_wave_table.rb` より

```
    def generate_wave_table_sawtooth(max)
      generate_wave_table(max, "sawtooth", 1.0) do |n, k|
        (2.0 / Math::PI) * Math::sin((2.0 * Math::PI) *
        ((n + 0.5) / 256.0) * k) / k
      end
    end
```

* 倍音成分（正弦波）を加算して、ノコギリ波などを一周期分合成

* 音の高さによって、最大倍音を制限（エイリアスノイズ対策）

`wave_table.rb` より

```
--------------------
    $wave_table_sawtooth_m63 = [
       +38,  +87,  +89,  +73,  +71,  +80,  +80,  +73,
       ...
       -73,  -80,  -80,  -71,  -73,  -89,  -87,  -38,
    ]

    $wave_tables_sawtooth = [
      $wave_table_sawtooth_m63,
      ...
      $wave_table_sawtooth_m3,
    ]

*    波形テーブルはそれぞれ 256 バイト（8 ビット×256 サンプル）
```

`vco.rb` より

```
-------------
        @phase += @freq
        @phase &= 0xFFFF
        ...
        curr_index = high_byte(@phase)
        ...
        curr_data = wave_table[curr_index]
        ...
          level = high_byte((curr_data * curr_weight) +
                            (next_data * next_weight))
```

* 高音ほど高周波数、速く再生（周波数テーブル参照、ピッチベンド非対応）

* 位相の値を使って、波形テーブルの「現在」と「次」のデータを線形補間

`vcf.rb` より

```
-------------
        tmp = -(a2_over_a0 * @y2);
        tmp += b2_over_a0 * x0;
        tmp += (b2_over_a0 << 1) * @x1;
        tmp += b2_over_a0 * @x2;
        tmp += a1_over_a0_i * @y1;
```

*　　たった 5 回の乗算でローパスフィルターを実現（不安定なところはある）

*　　<http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt>

`vca.rb` と `VCA.h` より

```ruby
# Ruby
def clock(a, k)
  high_byte(a * (k << 1))
end
```

```cpp
// Arduino (C++)
static int8_t clock(int8_t a, uint8_t k)
{
  return highByte(a * (uint8_t) (k << (uint8_t) 1));
}
```

デモ（Ruby シンセのリアルタイム再生）

---

* Chrome から JRuby 上のソフトシンセに、仮想 MIDI でデータ送信

* 音が途切れないように、バッファサイズを調整（発音遅延の原因...）

* なお、Arduino 版は発音遅延が非常に短く、音が途切れない

まとめ・感想

- - - - - - - - - - - - -

* 組込みソフトウェア開発でも Ruby は役立つ

* 音響プログラミングは面白い（Ruby でのリアルタイム処理は大変だけど）

* あまり Ruby らしくないコードになってしまった（移植を意識しすぎた）

* Ruby と C++のダブルメンテが必要になってしまった（仕方ないところもある）

エンディング

---

* <http://risgk.github.io/ruby-arduino-synth/ruby-arduino-synth-ed.mp3>

* Ruby を使った「アルゴリズム作曲」の実験より

* <https://github.com/risgk/algorithmic-composition-trial/blob/master/trial-4.rb>

Enjoy programming and making!

ご清聴ありがとうございました