はじめての

# ぴゅんぴゅんマシン

Mixer

Ardunino
Discrete

(仮)ぴゅんぴゅんマシン製作所

# 目次

## はじめに

### ディスク内容

製作に使用したファイルは以下のフォルダに入れてあります。

| | |
|---|---|
| ぴゅんぴゅん１号 | PyunPyunNo1 |
| ぴゅんぴゅん２号 | PyunPyunNo2 |
| ミキサー | Mixer |

### ぴゅんぴゅんマシンとは

レゲエのおっさんが使ってるあれだ。と言ってもあやしいハーブではなくてちょいちょい飛び道具としてライブで使われるあれである。めっちゃ情けない音がするあれ。

サイレン・マシンとかダブ・サイレンと呼ばれることもある。

もし、ハードでもソフトでもシンセをお持ちであればすぐにエミュレート可能。

**LFO＋VCO＋Delay**

これで OK。

OK なんだが、これではぴゅんぴゅんマシンの一番大事な特性を無視していると言わざるを得ない。

一番大事な特性とはなにか？

それは…

**ラリってもライブで気持よく音が出せる。**

**めちゃくちゃないじり方でもなんとなく様になっちゃう。**

これだ。

ライブって言っても別にクラブに行かなくても、自宅で一人で曲かけていじってると結構楽しいのだ。

## ぴゅんぴゅんマシンはレゲエだけではもったいない

四つ打ち系の曲でも意外と合う。
というか、レゲエよりむしろエレクトロの方が新たな発見があって面白い。
ベースとして使ったりエフェクトとして使ったりとダーティなテクノやサイケならそのまま使える。

DTM の音源は大概 12 音階に縛られているので、最近流行りの DUB STEP は音程をずらして気持ちのいい外れ具合を追求している。

が、**ぴゅんぴゅんマシンはそもそも音階がない。**

## ぴゅんぴゅん１号


## むしろ零号機

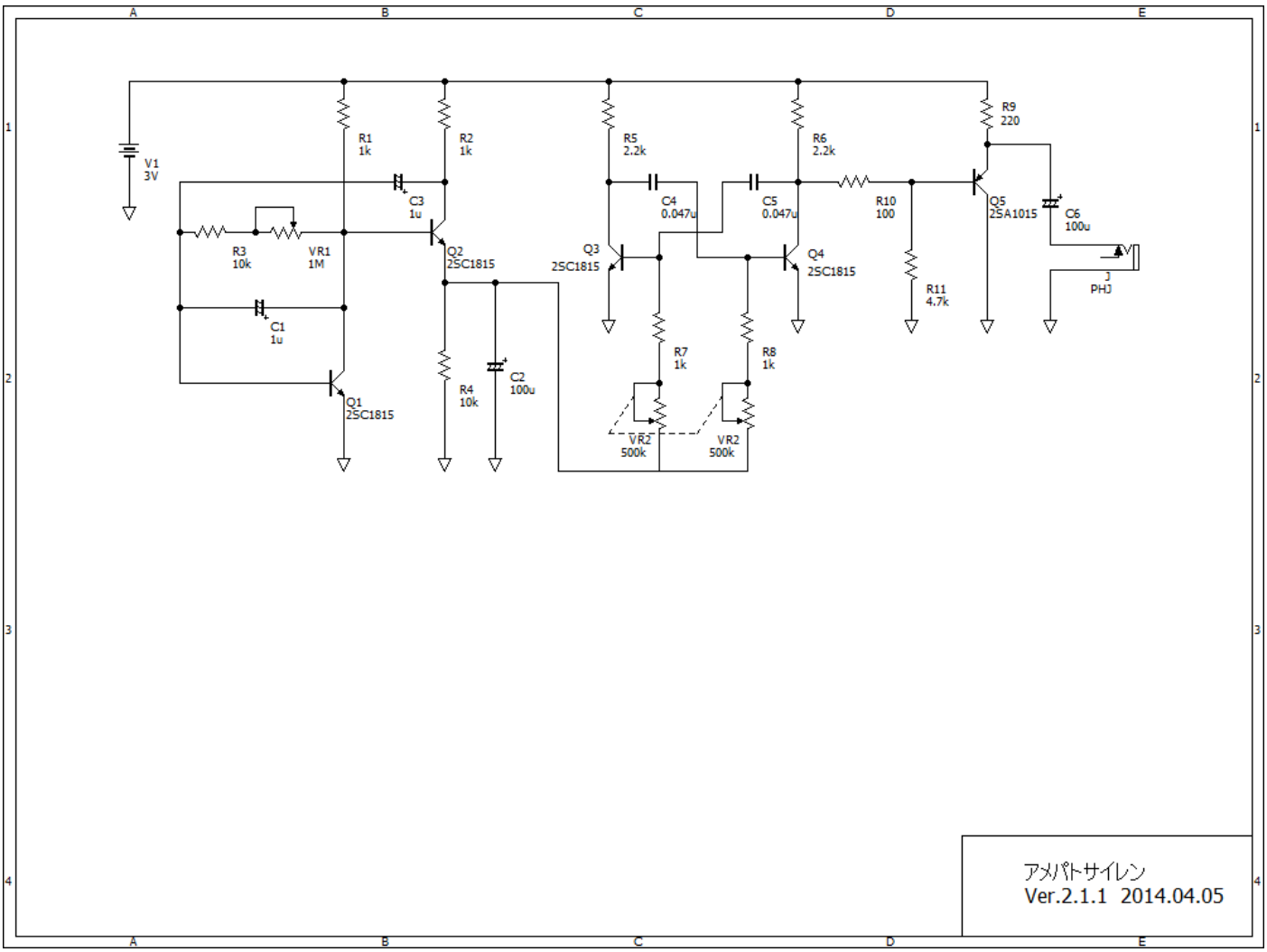
というわけで何十年かぶりにはんだごてを握って作った。製作過程の詳しいことはもう覚えてない。

たしか Web にあったアメパトサイレンの回路図をそのまま引っ張ってきて抵抗を可変にしたような気がする。まさに行き当たりばったり。まあいいや。

# それでも意外となんとかなるもんだ。

## 回路図

基板の配線図だけは残ってたので BSch(http://www.suigyodo.com/online/schsoft.htm)で回路図を書き直した。



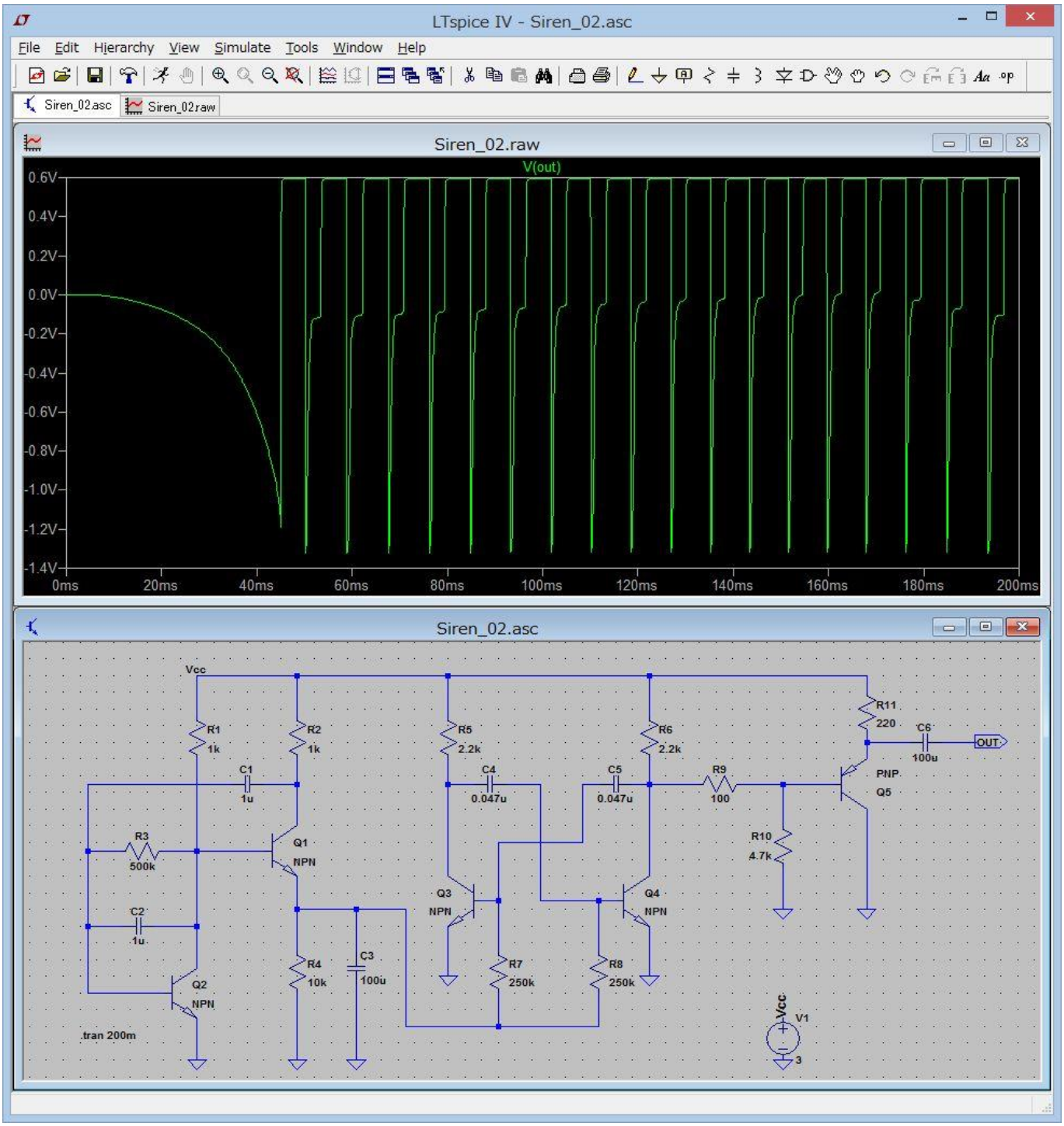

右側の波形生成部分はトランジスタの無安定マルチバイブレータだが、VR2 に流す電流で発振周波数を制御している（はず）。

左側の LFO に当たる部分は正直言ってよくわからないので、改めて単体でシミュレーションしてみることにした。

## シミュレーション

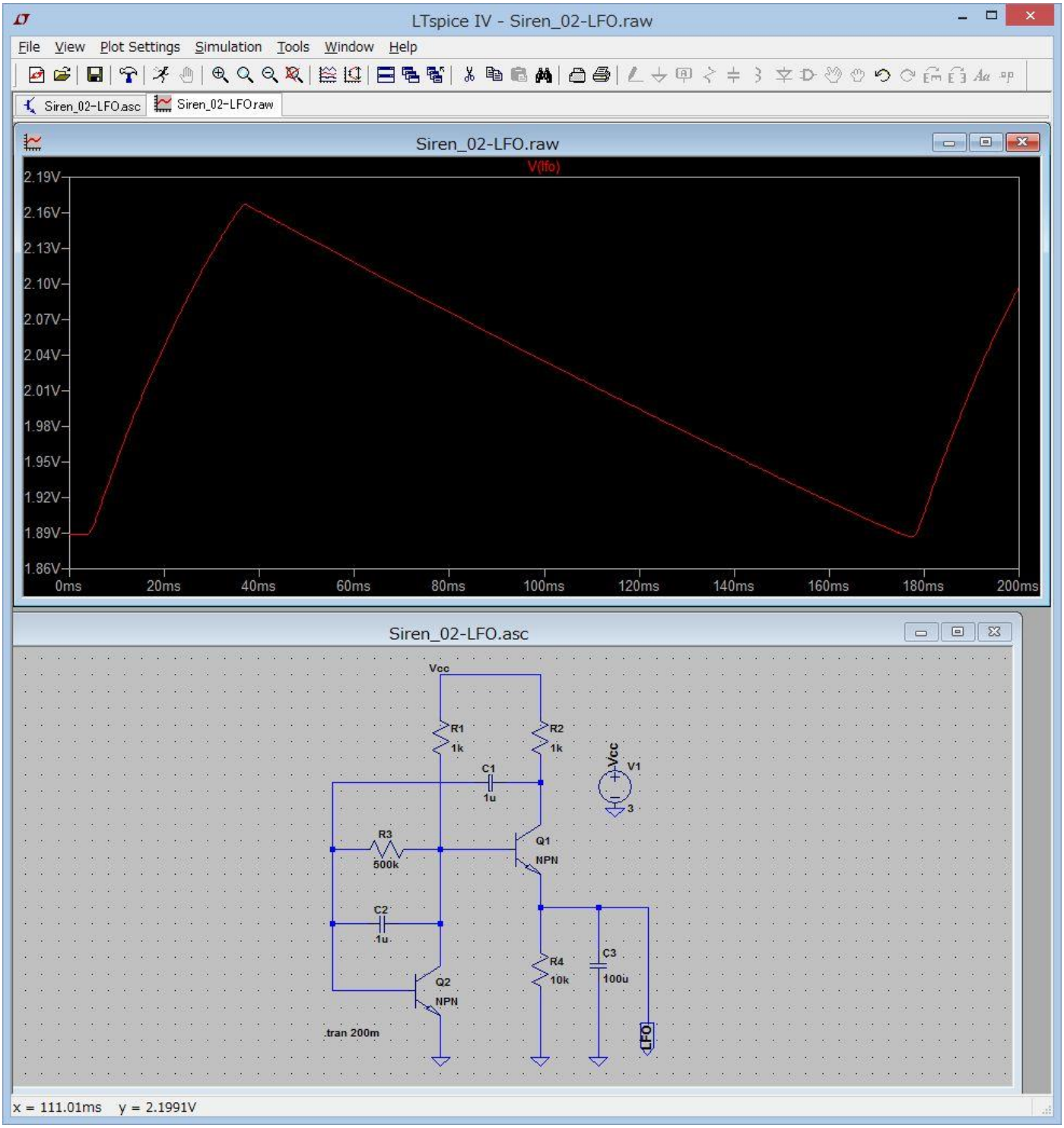
LTSpice(http://www.linear-tech.co.jp/designtools/software/)でシミュレーションしてみた。

出力波形




基本波形はこういう**矩形波くずれのクソ波形**だ。これを LFO 段で揺らしてる。

LFO 波形



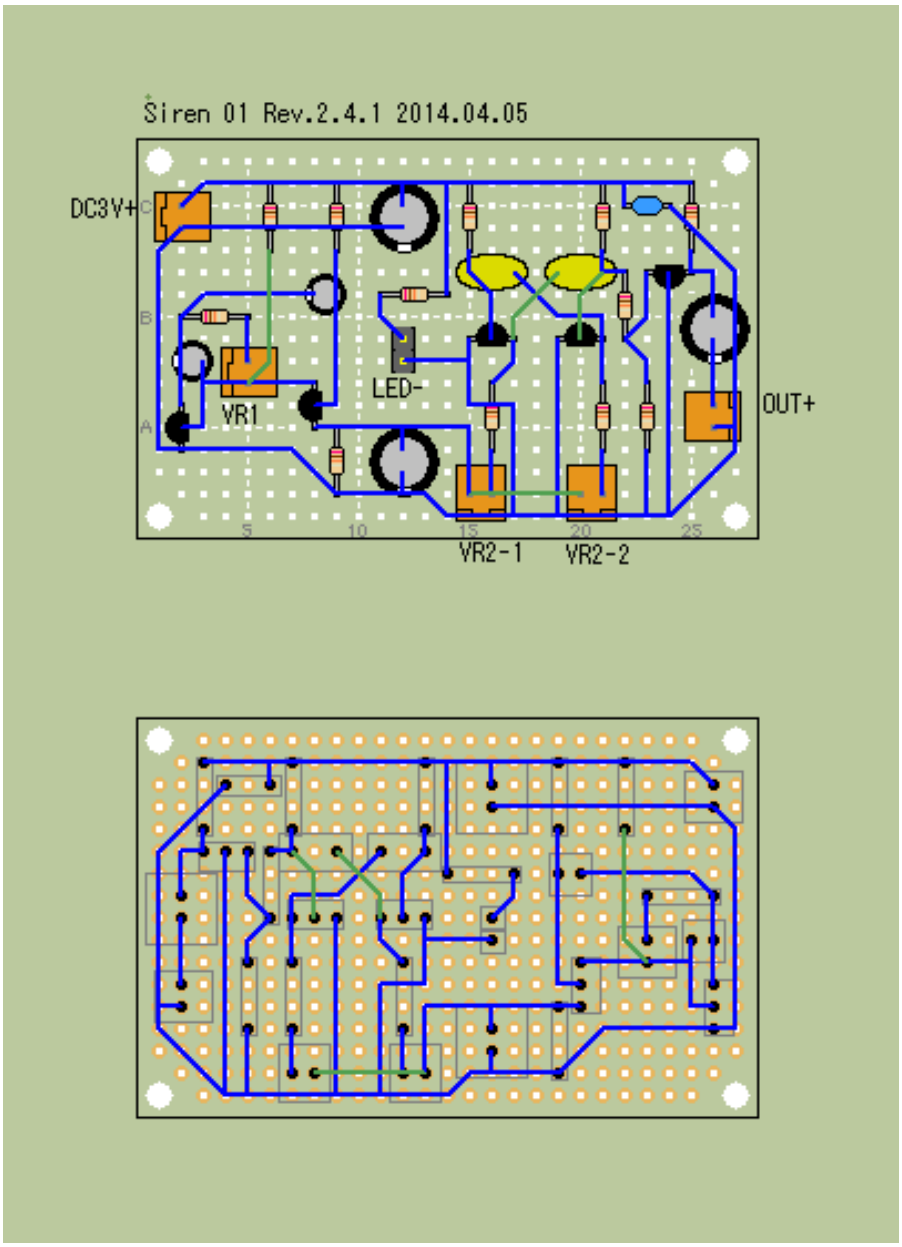

LFO 部分だけでシミュレーションしてみた。多分ランプ波の発振回路なんだろう。あんまり見たことがない回路だ。

ここの回路を変えてやれば揺れ方のバリエーションも増える。たぶん。

## 製作

PasS(http://www.geocities.jp/uaubn/pass/)でユニバーサル基板の配線図を描いた。

**基板配線図**

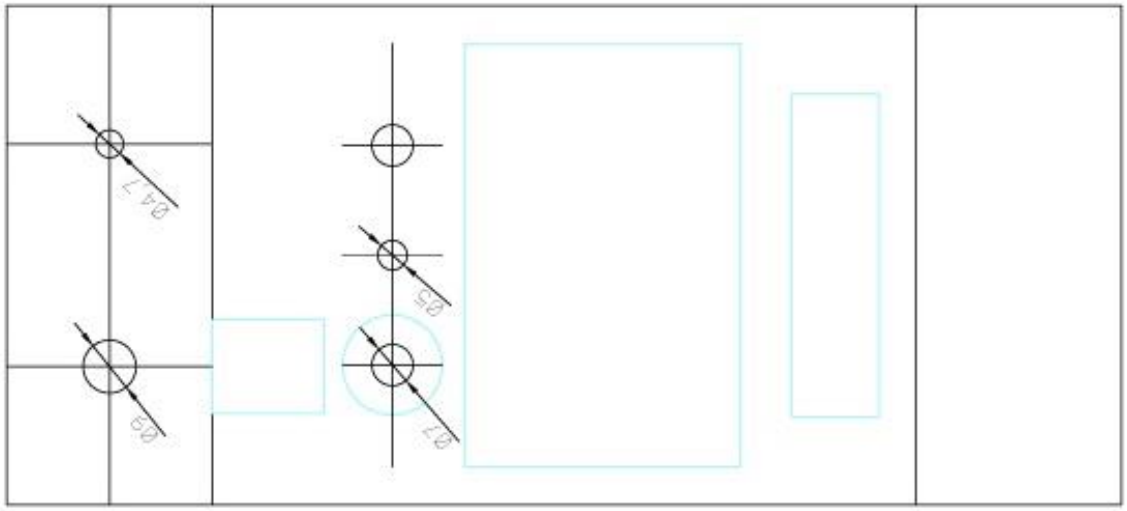



PasS はユニバーサル基板に特化していて部品もグラフィカルに表示されて便利なんだが、Undo 機能がないので終盤になって修正したくなるとひどい目に合う。特に GND や電源ラインは修正しようと思うと全体を消去しないといけなくなるので泣きそうになる。

それでも手書きで配線図を書くよりははるかに楽だし記録として残せるので、まずはこのソフトを使いたくなる。モチベーションがあがればユニバーサル基板専用の CAD ソフトも作ってみたいんだが。

でも、Openware のブレッドボード用で Flitzing というのもあるので(http://fritzing.org/projects/)、こっちにコントリビュートしたほうがいいのかなぁ…

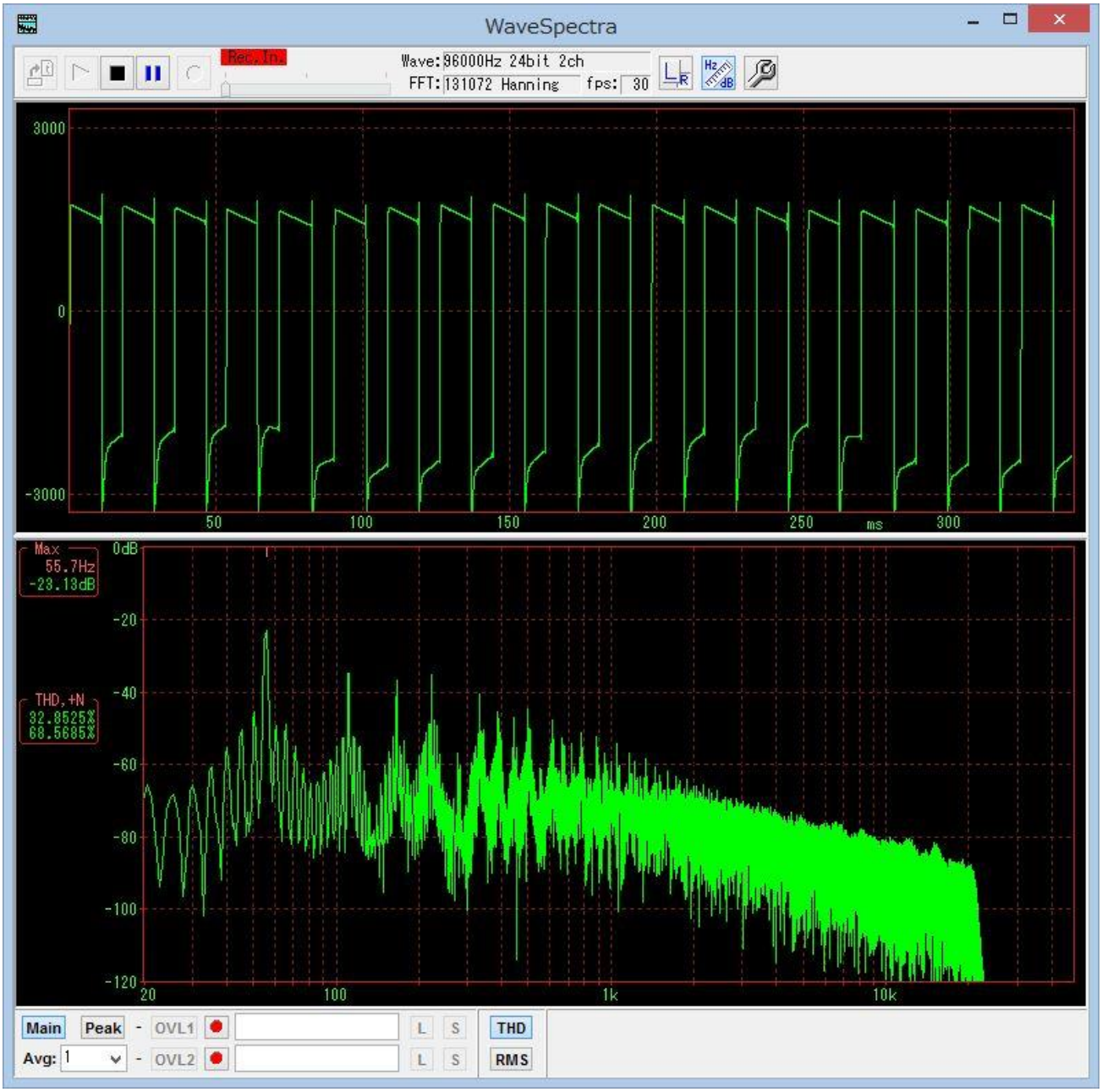
穴あけ加工図




ケースはタカチの TD9-12-4N を使用。
まじめに作ればもう 1 サイズ小さいケースに収められると思う。

**実際に出てくる波形**

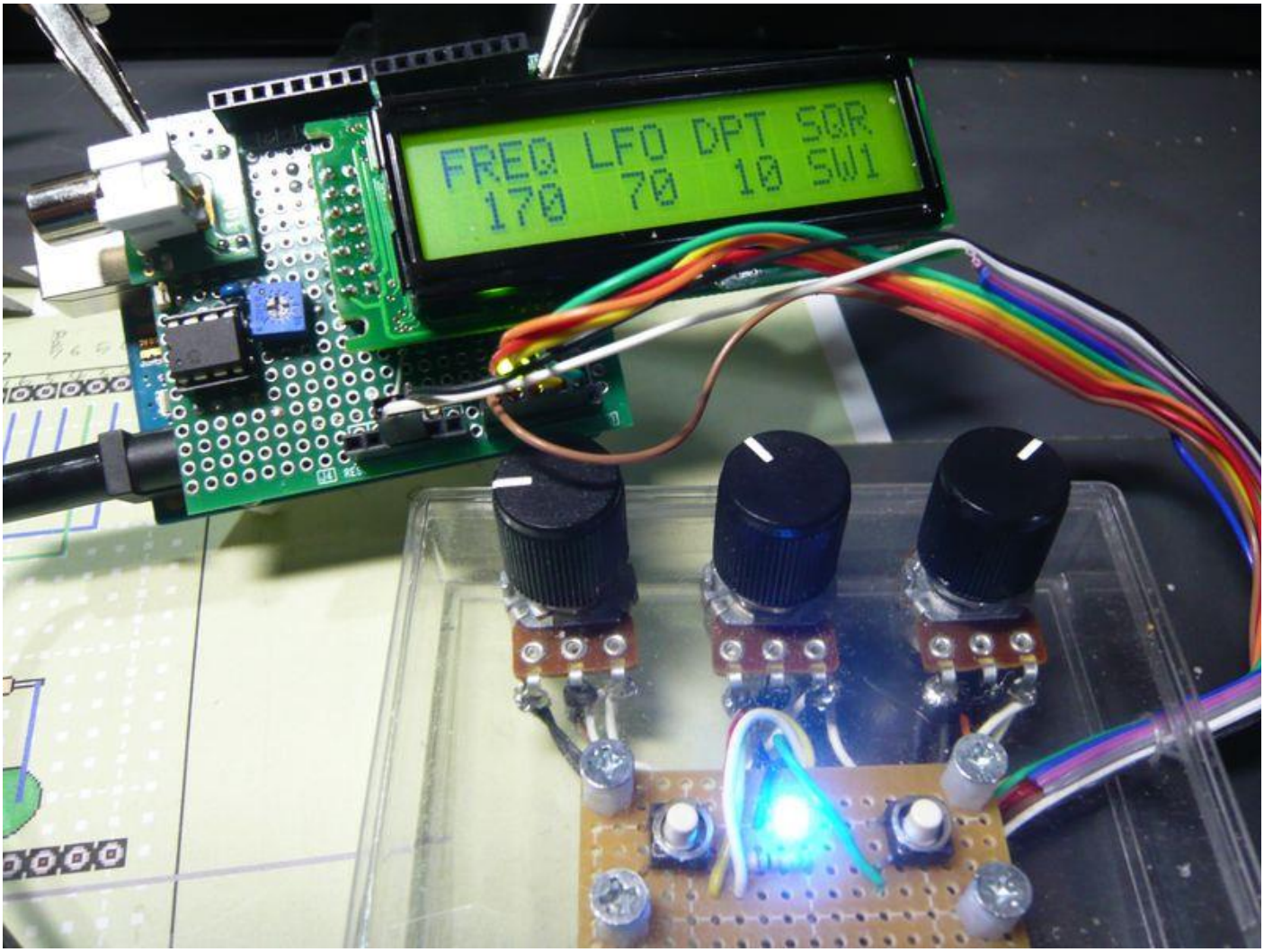
シミュレーションとはだいぶ違うがこんな波形になった。




アメパトっぽい音にするには左のつまみを９時方向に右のつまみを３時方向に回す。

でも、POT の位置によって出てくる音がころころ変わるので、適当なビートを流して適当に音出しするのが一番気持ちいい。音量を調節できるボリュームがあったらいいなとか、一瞬ブレイクさせるボタンがあったらいいなとか、不満点はいろいろあるが後々のお楽しみだ。

## ぴゅんぴゅん２号




## Arduino を使ってみた

正直言ってアナログは理論が難しすぎて思ったような波形をだすのに苦労する。日々勉強。
なので、まだ少しはわかるデジタルでなんとか処理してみたくなった。

と言っても PC やスマホでプログラミングするのもなんか仕事みたいで嫌だ。

なので巷で噂の Arduino というやつを使ってみた。

Arduino の MPU は 16Bit の AVR なので MS-DOS 世代としては悪くない選択かなと思った。

しばらく Arduino を使ってみたが、どうも Arduino の命令セットだけでは大したことができないことがわかった。

大事なところは AVR を直に叩くしかない。

しかたないんだが、データシートを読んでも AVR の参考書を読んでもピンとこない。さっぱりわからん。そこで、Web に上がってる Arduino のシンセの作例を参考にして（コピペとも言う）ぴゅんぴゅん２号を作ることにした。

**主に参考にしたサイト**

Arduino DDS Sinewave Generator

http://interface.khm.de/index.php/lab/experiments/arduino-dds-sinewave-generator/

Turn your Arduino into a 4 voice wavetable synth with only a few components...

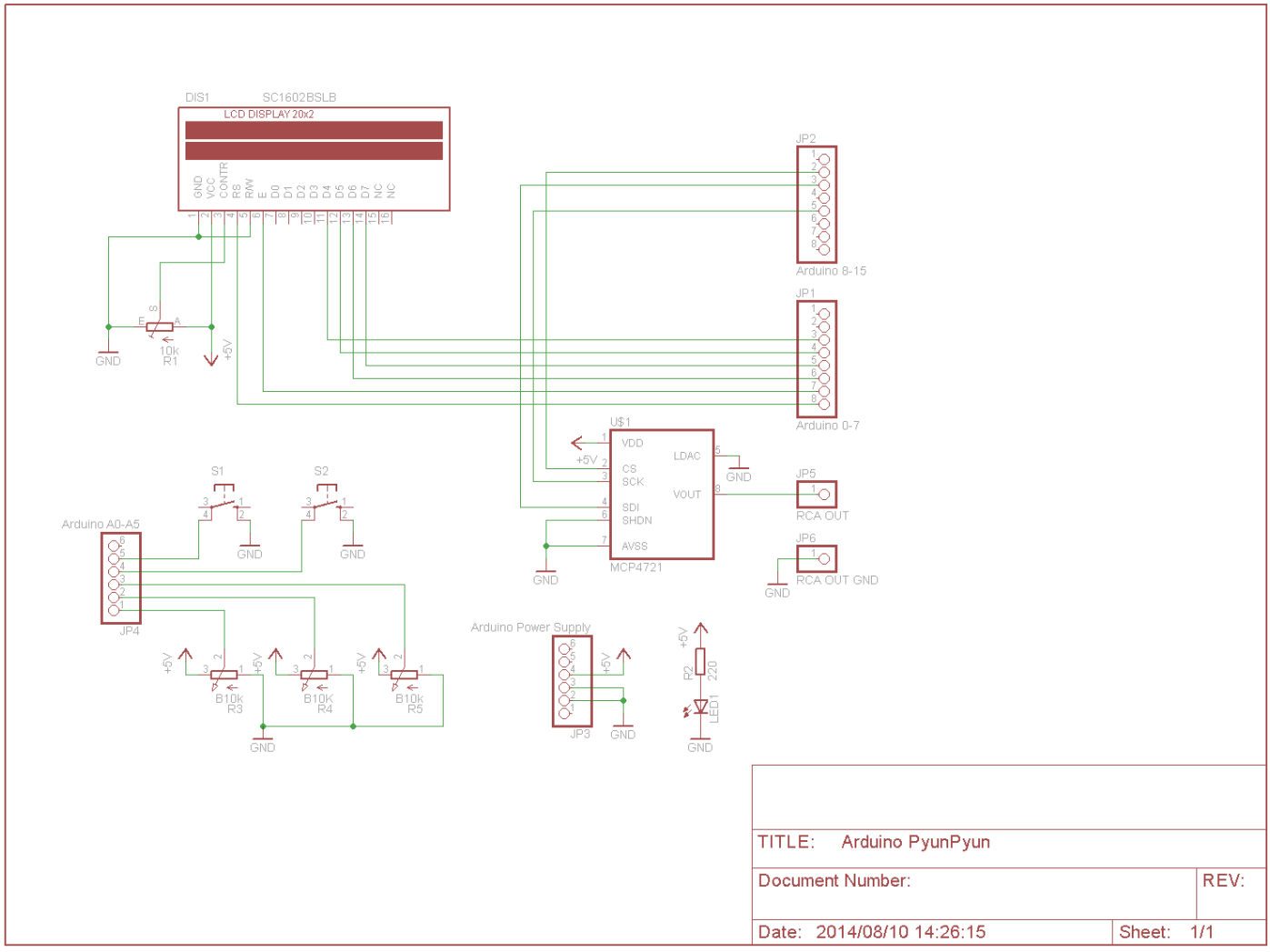
http://www.instructables.com/id/Turn-your-Arduino-into-a-4-voice-wavetable-synth-w/?lang=ja

## 回路図

いま見てみたら２号も回路図書いてなかった（笑）

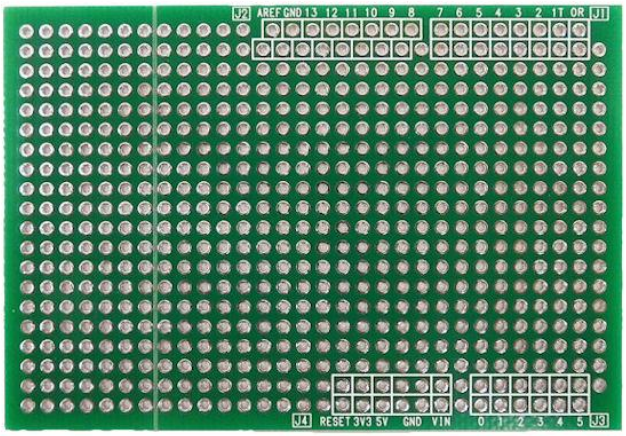
ブレッドボードで試作してそのまま基板の配線図を起こした模様。

あらためて回路図を描いた。

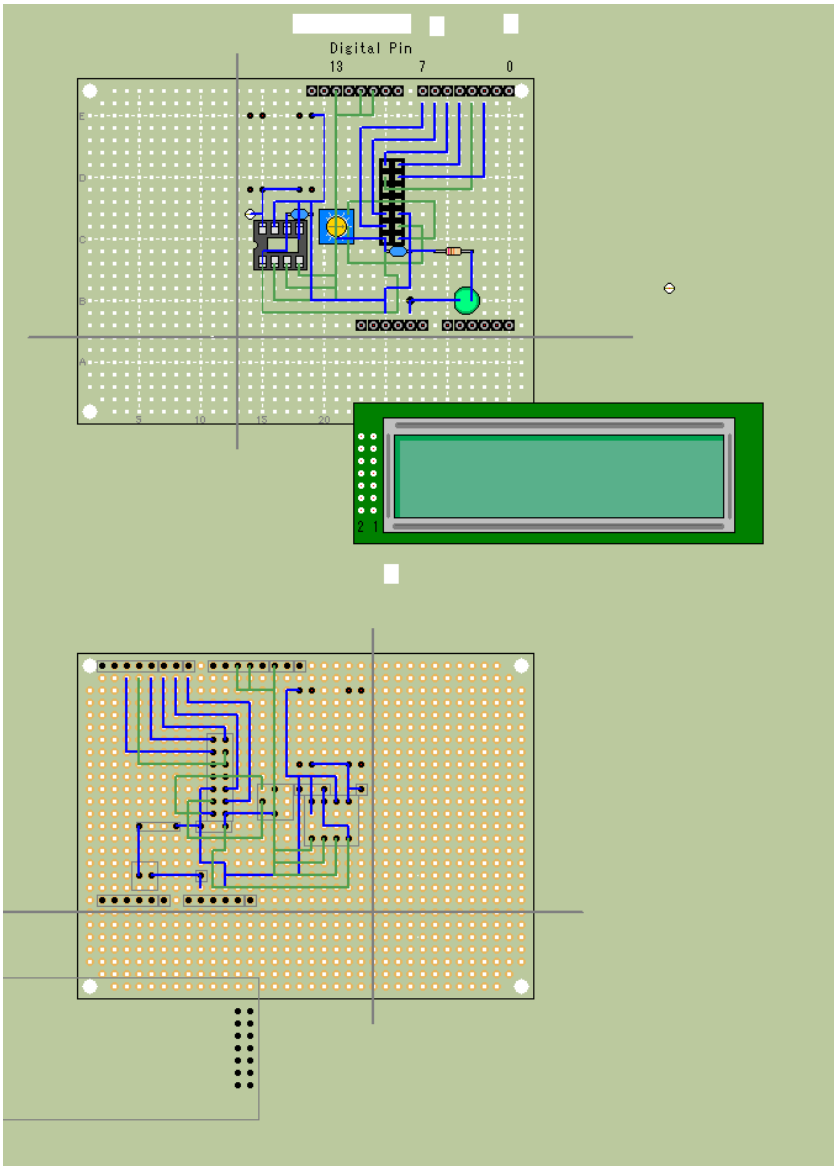


## 製作

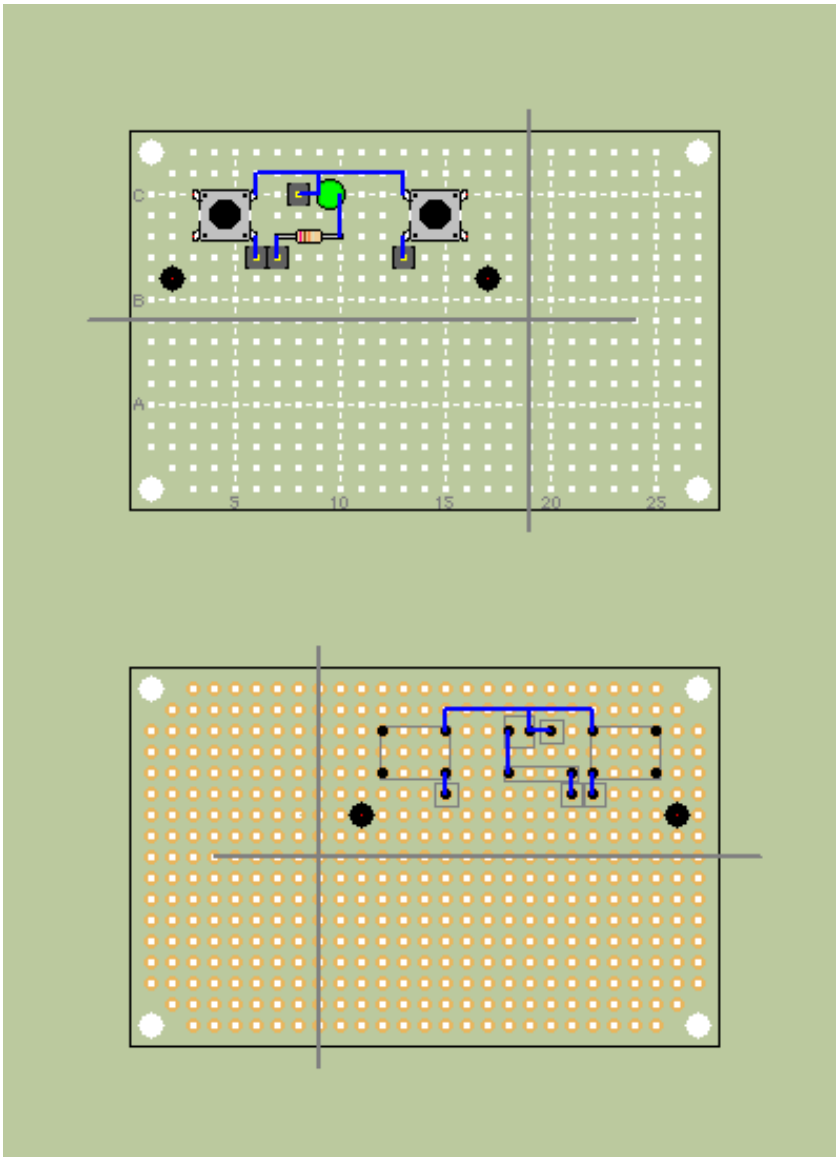
秋月の Arduino 用のシールド基板で組んだ。



### 基板配線図

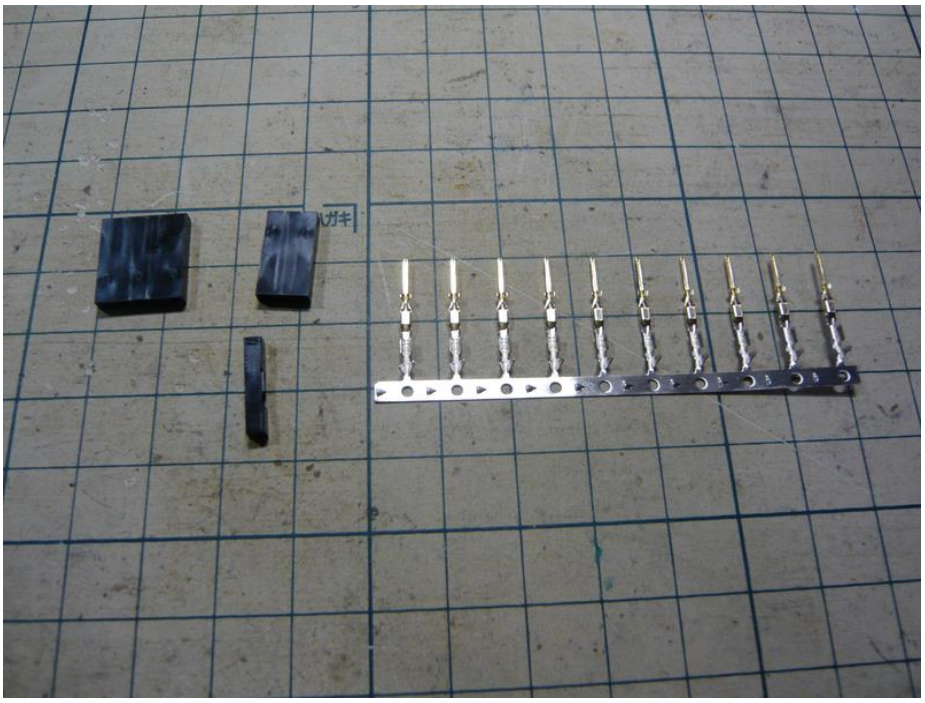


コントローラーの配線図

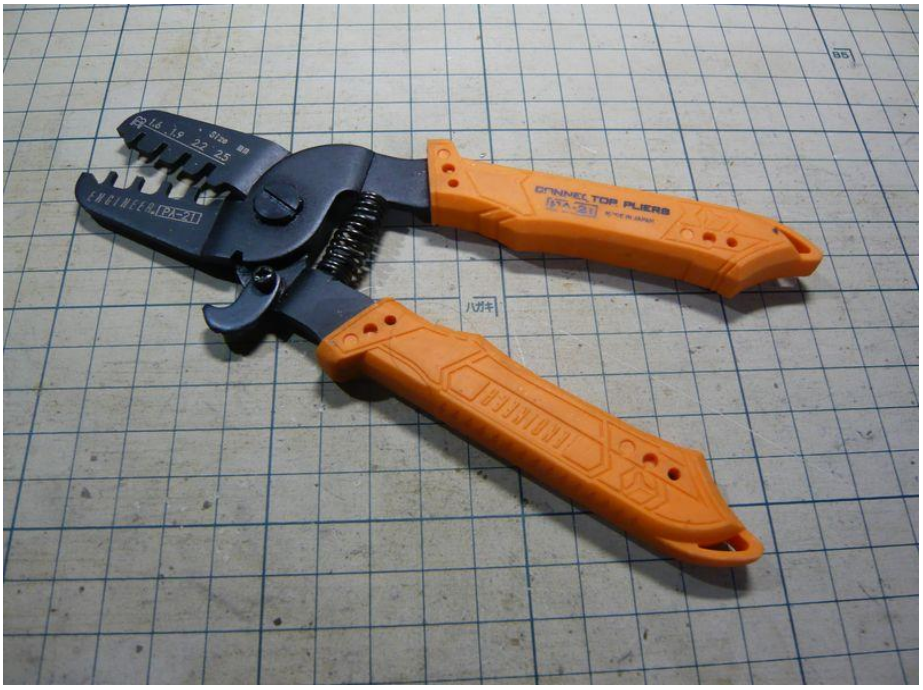



コントローラーは秋月のポリカーボネートのケースに入れた。

コントローラーと Aruduino のコネクタには共立で売ってた QI コネクタというやつを使ってみた。
ピンをケーブルに圧着してケースを被せる。



圧着にはエンジニアの PA-21 という工具を使っている。




圧着のコツを掴むのは結構大変だが、必要なピン数のケーブルが作れるのでそれなりに便利だと思う。
本番のアナログ回路に使う気はしないが、ピンコネクタだけではなくブレッドボードにもぶっさせる。

http://blog.digit-parts.com/archives/51796900.html

## ファームウエア

波形の生成には DDS(Direct Digital Synthesizer)を使った。

これはサンプラーやソフトウェア・シンセ、アナログ・モデリング・シンセで使われている方式だと思う。

利点は、少ない計算量で綺麗で多様な波形が生成できる点。

欠点は、演算ビット長、メモリ量、クロック数等それなりのハードウェア・リソースが必要とされる点。

Arduino ではさすがにスペック不足と言わざるを得ないが、逆に制約された中で波形を生成するのも面白い。DAW のエフェクトでわざわざデジタル歪をつくるより、最初から歪んでいたほうが何かとあきらめもつく。

演算速度の都合上、サンプリングレートは 15,625Hz にした。これでもまじめに LFO の重みを掛け算すると演算処理が追いつかないので、シフト演算で処理している。

DDS 以外では、DAC の制御に<MCPDAC>、LCD の制御に<LiquidCrystal>のライブラリを使ってみた。また、スイッチのチャタリング対策に直書きでタイマー処理の中でデバウンス処理を入れた。

PyunPyun62.ino

```
/*
  Arduino Pyun Pyun Machine

 * CS   10
 * MOSI 11
 * SCK  13

 * Pot Freq      A0
 * Pot LFO Freq  A1
 * Pot LFO Depth A2
 * Button Wave Form A3 (D17)
 * Button LFO Wave  A4 (D18)

 * LCD RS pin to digital pin 7
 * LCD Enable pin to digital pin 6
 * LCD D4 pin to digital pin 2
 * LCD D5 pin to digital pin 3
 * LCD D6 pin to digital pin 4
 * LCD D7 pin to digital pin 5
 * LCD R/W pin to ground
 * 10K resistor:
```

```
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)

 2014.01.15 by gizmo
 2014.01.15 DAC 出力
 2014.01.27 WavForm 選択をボタンに
 2014.01.27 Debounce
 2014.02.01 LFO 表示を 3 桁に

 */

#include <SPI.h>
#include <MCPDAC.h>
#include <LiquidCrystal.h>
#include "avr/pgmspace.h"

// table of 256 values / one period / stored in flash memory
PROGMEM  prog_uchar sine256[]  = {

127,130,133,136,139,142,145,148,151,155,158,161,164,167,170,173,175,178,181,184,187,190,192,195
,198,200,203,205,208,210,212,215,

217,219,221,223,225,227,229,231,233,235,236,238,239,241,242,244,245,246,247,248,249,250,251,251
,252,253,253,254,254,254,254,254,

255,254,254,254,254,254,253,253,252,251,251,250,249,248,247,246,245,244,242,241,239,238,236,235
,233,231,229,227,225,223,221,219,

217,215,212,210,208,205,203,200,198,195,192,190,187,184,181,178,175,173,170,167,164,161,158,155
,151,148,145,142,139,136,133,130,

127,124,121,118,115,112,109,106,103,99,96,93,90,87,84,81,79,76,73,70,67,64,62,59,56,54,51,49,46
,44,42,39,
  37,35,33,31,29,27,25,23,21,19,18,16,15,13,12,10,9,8,7,6,5,4,3,3,2,1,1,0,0,0,0,0,
  0,0,0,0,0,0,1,1,2,3,3,4,5,6,7,8,9,10,12,13,15,16,18,19,21,23,25,27,29,31,33,35,

37,39,42,44,46,49,51,54,56,59,62,64,67,70,73,76,79,81,84,87,90,93,96,99,103,106,109,112,115,118
,121,124
};
PROGMEM prog_uchar tri256[] = {
```

```
128,130,132,134,136,138,140,142,144,146,148,150,152,154,156,158,160,162,164,166,168,170,172,174
,176,178,180,182,184,186,188,190,

192,194,196,198,200,202,204,206,208,210,212,214,216,218,220,222,224,226,228,230,232,234,236,238
,240,242,244,246,248,250,252,254,

255,254,252,250,248,246,244,242,240,238,236,234,232,230,228,226,224,222,220,218,216,214,212,210
,208,206,204,202,200,198,196,194,

192,190,188,186,184,182,180,178,176,174,172,170,168,166,164,162,160,158,156,154,152,150,148,146
,144,142,140,138,136,134,132,130,

128,126,124,122,120,118,116,114,112,110,108,106,104,102,100,98,96,94,92,90,88,86,84,82,80,78,76
,74,72,70,68,66,
  64,62,60,58,56,54,52,50,48,46,44,42,40,38,36,34,32,30,28,26,24,22,20,18,16,14,12,10,8,6,4,2,
  0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54,56,58,60,62,

64,66,68,70,72,74,76,78,80,82,84,86,88,90,92,94,96,98,100,102,104,106,108,110,112,114,116,118,1
20,122,124,126
};
PROGMEM prog_uchar saw1_256[] = {

255,254,253,252,251,250,249,248,247,246,245,244,243,242,241,240,239,238,237,236,235,234,233,232
,231,230,229,228,227,226,225,224,

223,222,221,220,219,218,217,216,215,214,213,212,211,210,209,208,207,206,205,204,203,202,201,200
,199,198,197,196,195,194,193,192,

191,190,189,188,187,186,185,184,183,182,181,180,179,178,177,176,175,174,173,172,171,170,169,168
,167,166,165,164,163,162,161,160,

159,158,157,156,155,154,153,152,151,150,149,148,147,146,145,144,143,142,141,140,139,138,137,136
,135,134,133,132,131,130,129,128,

127,126,125,124,123,122,121,120,119,118,117,116,115,114,113,112,111,110,109,108,107,106,105,104
,103,102,101,100,99,98,97,96,

95,94,93,92,91,90,89,88,87,86,85,84,83,82,81,80,79,78,77,76,75,74,73,72,71,70,69,68,67,66,65,64
,
```

```
63,62,61,60,59,58,57,56,55,54,53,52,51,50,49,48,47,46,45,44,43,42,41,40,39,38,37,36,35,34,33,32
,
  31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0
};
PROGMEM prog_uchar saw2_256[] = {
  0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,

32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63
,

64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95
,

96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120
,121,122,123,124,125,126,127,

128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151
,152,153,154,155,156,157,158,159,

160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,182,183
,184,185,186,187,188,189,190,191,

192,193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,210,211,212,213,214,215
,216,217,218,219,220,221,222,223,

224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247
,248,249,250,251,252,253,254,255
};
PROGMEM prog_uchar sqr256[] = {
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127
,127,127,127,127,127,127,127,127,

127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127
,127,127,127,127,127,127,127,127,
```

```
127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127
,127,127,127,127,127,127,127,127,

127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127
,127,127,127,127,127,127,127,127,
};

#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))

#define WAVEFORM_NUM  5
#define DEBOUNCE_CNT  1250         // Wait count for debounce 15625 / 12.5 Hz (80ms)

int ledPin = 13;                  // LED pin 13
//int testPin = 6;
//int testPin2 = 7;

double dfreq_wav;                 // wave frequency
double dfreq_lfo;                 // lfo frequency

const double refclk = 15625.0;      // = 16MHz / 8 / 128

// variables used inside interrupt service declared as voilatile
volatile uint8_t cnt_wav;           // var inside interrupt
volatile uint8_t cnt_lfo;           // var inside interrupt
volatile uint8_t v_lfo;

volatile uint32_t phaccu_wav;       // pahse accumulator
volatile uint32_t phaccu_lfo;       // for lfo
volatile uint32_t tword_m_wav;      // dds tuning word m
volatile uint32_t tword_m_lfo;      // for lfo

volatile uint8_t depth_lfo;
volatile uint8_t *waveforms[WAVEFORM_NUM];
volatile uint8_t waveform_wav;
volatile uint8_t waveform_lfo;

volatile uint16_t waveform_wav_pushed_cnt;
volatile uint16_t waveform_lfo_pushed_cnt;
```

```
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(7, 6, 2, 3, 4, 5);

const char *waveform_str[] = {
  "SIN", "TRI", "SW1", "SW2", "SQR" };

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
  Serial.begin(115200);         // connect to the serial port
  Serial.println("DDS Test");

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  //lcd.clear();

  //pinMode(6, OUTPUT);      // sets the digital pin as output
  //pinMode(7, OUTPUT);      // sets the digital pin as output

  pinMode(17, INPUT_PULLUP); // A3
  pinMode(18, INPUT_PULLUP); // A4

  // CS on pin 10, no LDAC pin (tie it to ground).
  MCPDAC.begin(10);

  // Set the gain to "LOW" mode
  MCPDAC.setGain(CHANNEL_A,GAIN_LOW);

  // Do not shut down channel A, but shut down channel B.
  MCPDAC.shutdown(CHANNEL_A,false);
  MCPDAC.shutdown(CHANNEL_B,true);

  Setup_timer2();

  // disable interrupts to avoid timing distortion
  sbi(TIMSK0,TOIE0);              // disable Timer0 !!! delay() is now not available
  sbi(TIMSK2,TOIE2);              // enable Timer2 Int.        errupt

  dfreq_wav = 1000.0;             // initial output frequency = 1000.o Hz
```

```
  dfreq_lfo = 1.0;              // initial lfo frequency = 1.0Hz
  tword_m_wav = pow(2, 32) * dfreq_wav / refclk;  // calulate DDS new tuning word
  tword_m_lfo = pow(2, 32) * dfreq_lfo / refclk;

  waveforms[0] = sine256;
  waveforms[1] = tri256;
  waveforms[2] = saw1_256;
  waveforms[3] = saw2_256;
  waveforms[4] = sqr256;

  waveform_wav = 0;
  waveform_lfo = 0;
  depth_lfo = 0;

  waveform_wav_pushed_cnt = 0;
  waveform_lfo_pushed_cnt = 0;

  sei();
}

void loop()
{
  char buff[20];

  while(1) {
//    sbi(PORTD,7);                        // Test / set PORTD,7 high to observe timing with a oscope

    dfreq_wav = analogRead(0);            // read Poti on analog pin 0 to adjust output frequency
from 0..1023 Hz
    dfreq_lfo = analogRead(1) / 50.0;     // 0..20.48 Hz
    depth_lfo = map(analogRead(2), 0, 1023, 16, 0);
    tword_m_wav = pow(2, 32) * dfreq_wav / refclk;  // calulate DDS new tuning word
    tword_m_lfo = pow(2, 32) * dfreq_lfo / refclk;
    sbi(TIMSK2, TOIE2);                          // enable Timer2 Interrupt

    if (waveform_wav_pushed_cnt == 0 && digitalRead(17) == LOW)
      waveform_wav_pushed_cnt = DEBOUNCE_CNT;
    if (waveform_lfo_pushed_cnt == 0 && digitalRead(18) == LOW)
      waveform_lfo_pushed_cnt = DEBOUNCE_CNT;
```

```
    lcd.setCursor(0, 0);
    sprintf(buff, "FREQ LFO DPT %s",  waveform_str[waveform_wav]);
    lcd.print(buff);
    sprintf(buff, "%4d %3d %3d %s", (int)dfreq_wav, (int)(dfreq_lfo * 10), 16 - depth_lfo,
waveform_str[waveform_lfo]);
    // sprintf(buff, "%4d %3d %3d %d", (int)dfreq_wav, (int)dfreq_lfo, 16 - depth_lfo,
waveform_lfo_pushed_cnt);
    lcd.setCursor(0, 1);
    lcd.print(buff);
  }
}

//************************************************************
// timer2 setup
// set prscaler to 8, PWM mode to phase correct PWM,  16000000 / 8 / 128 = 15625 Hz clock
void Setup_timer2() {

  // Timer2 PWM Mode set to Phase Correct PWM
  cbi (TCCR2A, COM2A0);
  cbi (TCCR2A, COM2A1);

  sbi (TCCR2A, WGM20);  // Mode 7 / Fast PWM
  sbi (TCCR2A, WGM21);
  sbi (TCCR2B, WGM22);

  OCR2A = 127;

  // Timer2 Clock Prescaler to : 8
  cbi (TCCR2B, CS20);
  sbi (TCCR2B, CS21);
  cbi (TCCR2B, CS22);
}

//************************************************************
// Timer2 Interrupt Service at 31372,550 KHz = 32uSec
// this is the timebase REFCLOCK for the DDS generator
// FOUT = (M (REFCLK)) / (2 exp 32)
// runtime : 8 microseconds ( inclusive push and pop)
ISR(TIMER2_OVF_vect) {
```

```
  //  sbi(PORTD,6);         // Test / set PORTD,7 high to observe timing with a oscope

  phaccu_lfo = phaccu_lfo + tword_m_lfo;
  cnt_lfo = phaccu_lfo >> 24;
  v_lfo = pgm_read_byte_near(waveforms[waveform_lfo] + cnt_lfo);

  phaccu_wav = phaccu_wav + tword_m_wav + (tword_m_wav >> depth_lfo) * v_lfo;
  cnt_wav = phaccu_wav >> 24;     // use upper 8 bits for phase accu as frequency information

  MCPDAC.setVoltage(CHANNEL_A, pgm_read_byte_near(waveforms[waveform_wav] + cnt_wav));

  // Debounce read buttons
    if (waveform_wav_pushed_cnt) {
      if (--waveform_wav_pushed_cnt == 0 && digitalRead(17) == LOW) {
        waveform_wav++;
        if (waveform_wav >= WAVEFORM_NUM)
          waveform_wav = 0;
      }
  }
  if (waveform_lfo_pushed_cnt) {
      if (--waveform_lfo_pushed_cnt == 0 && digitalRead(18) == LOW) {
        waveform_lfo++;
        if (waveform_lfo >= WAVEFORM_NUM)
          waveform_lfo = 0;
      }
  }

  //  cbi(PORTD,6);           // reset PORTD,7
}
```

※基本的にコピペで書いて動けばいいやという方針だったので、**ソースのコメント等は信用しないでください**（＾ｑ＾；

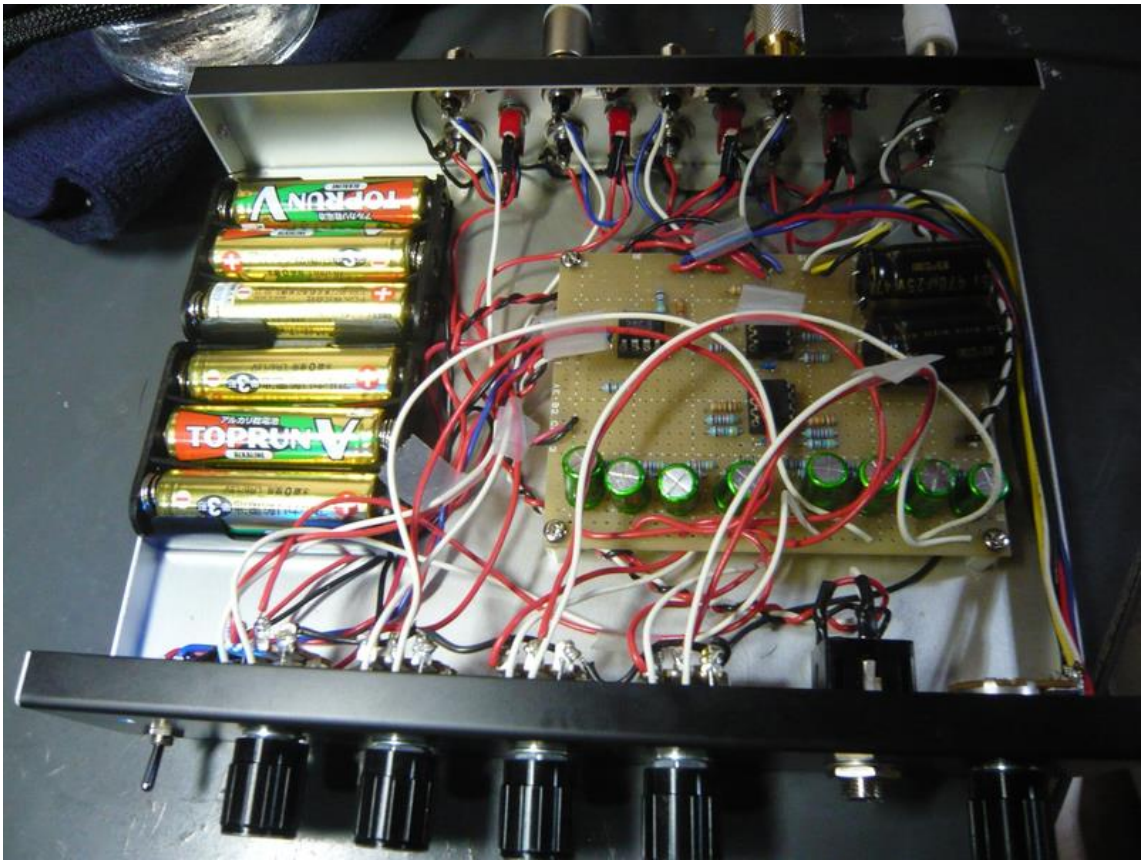
## 反省点

- Pin ジャックに刺しただけでは固定が甘くて LCD の収まり具合が悪い
- シールドの裏面の配線と Arduino の USB/DC ジャックとが干渉する
  - とりあえずシールドの裏面にビニールテープを貼って回避
- コントローラーの LED が明るすぎた

# ・とにかくファームウェアをがクソすぎる

不満点はいろいろあるが、随時更新する予定なのでブログ(http://dad8893.blogspot.jp/)とか Github(https://github.com/ryood)をチェックしてみてください。

## ミキサー



## 使い道

モニタスピーカーにはYAMAHAのMSP3を使っている。アクティブ・スピーカーで入力も２系統あり、ボリュームもそれぞれ調整できる。PCのオンボードサウンドとDTM用のAudio Interfaceを入力して１０年以上使っている。

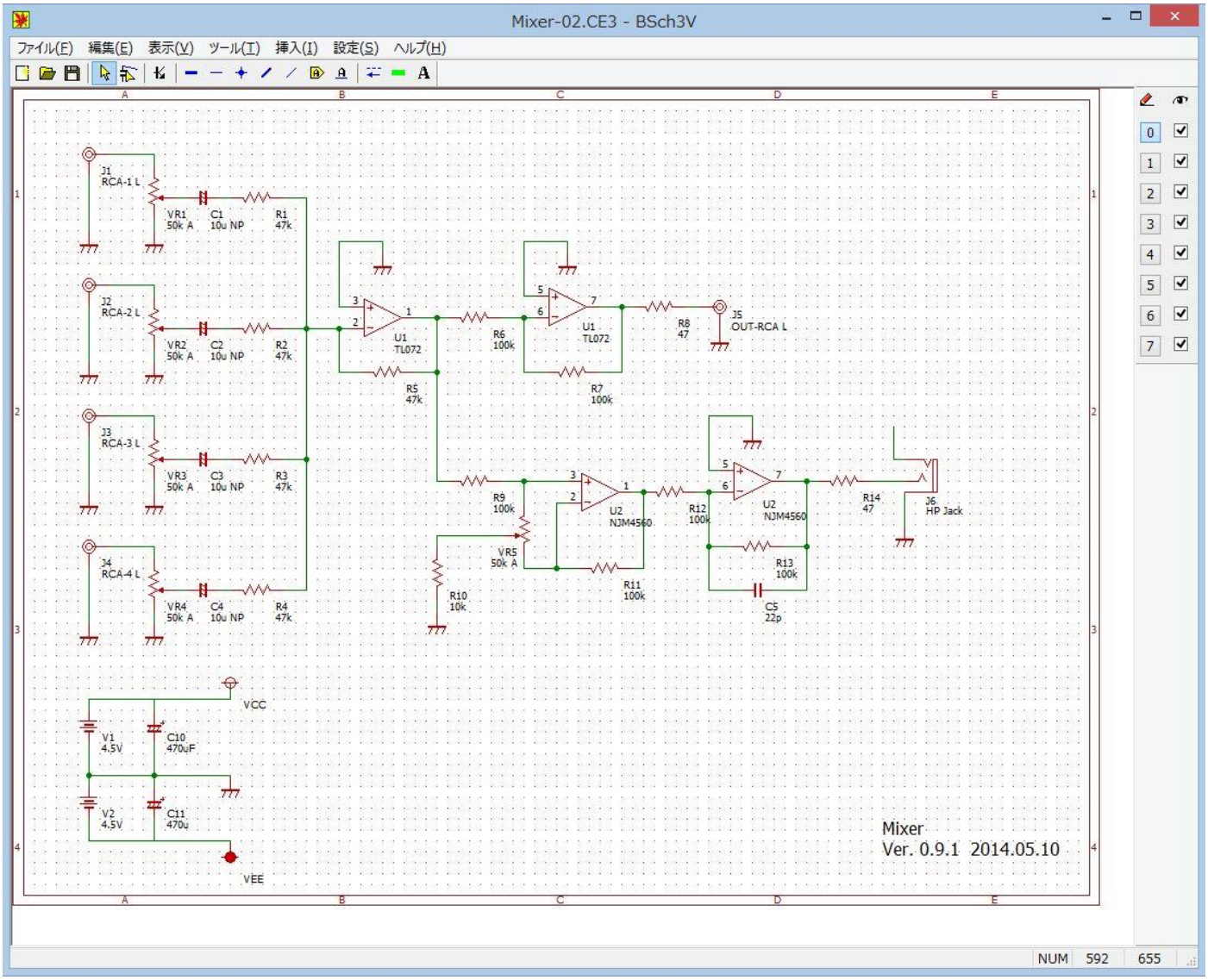
自作音源もこのスピーカーで鳴らしたいとこがあるが、入力端子が足りない。
Audio InterfaceのLine INに入力して音出ししているが、何分自作音源なのでいつ入力オーバーでAudio Interfaceを破壊するかわからない。

そこで、自作のミキサーを作ることにした。自作であれば故障しても自分で直せる。
音質に特に拘らなければOPAMP一個100円以下の修理費で済む。

ひとつあれば何かと便利だと思った。

## 回路図

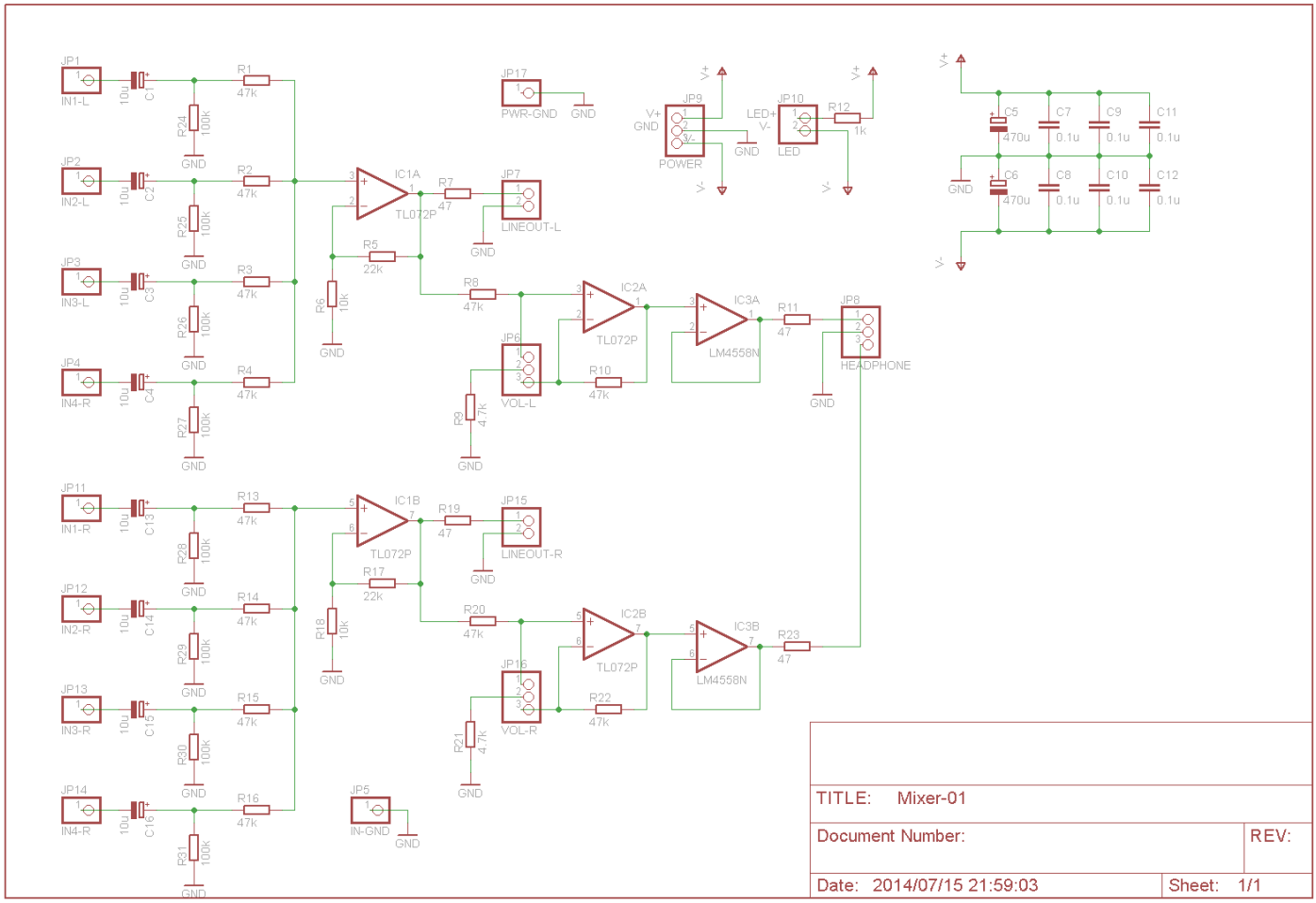
最初は Web の作例を参考にして反転加算回路で初段のミキサー部を構成した。




ブレッドボードで試作して実験してみるとちょっとした問題が発覚。電源をオフにしても入力が洩れるのだ。回路図を眺めてオペアンプの部分をオープンにして考えて見ると抵抗が何個か入ってはいるものの入力から出力まで筒抜けだ。受け側の入力インピーダンスが高いとかなり盛大に音漏れする。

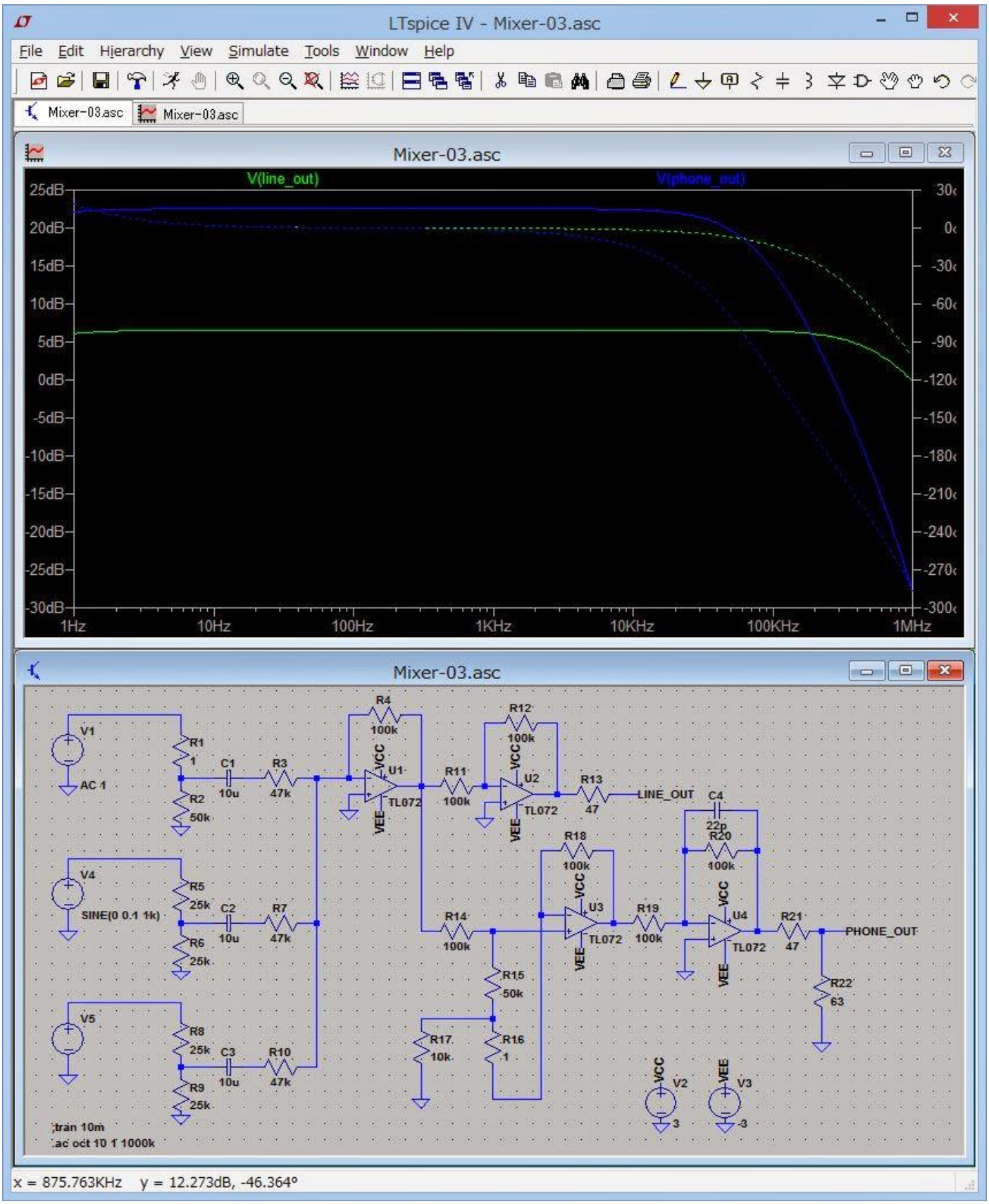
しかも位相が反転してるのを直すために 1 段反転回路が必要になる。
結果、部品数が増えるしノイズや歪が増えるのは確実。

そこで、ミキサー部を非反転加算回路で構成することにした。これなら電源をオフにすると OPAMP で絶縁されるので音漏れはしない。




Headphone 出力にボルテージフォロワを入れてみた。ここの OPAMP を変更するとヘッドホンで聴いた時の音質の違いを試してみたりもできると思う。

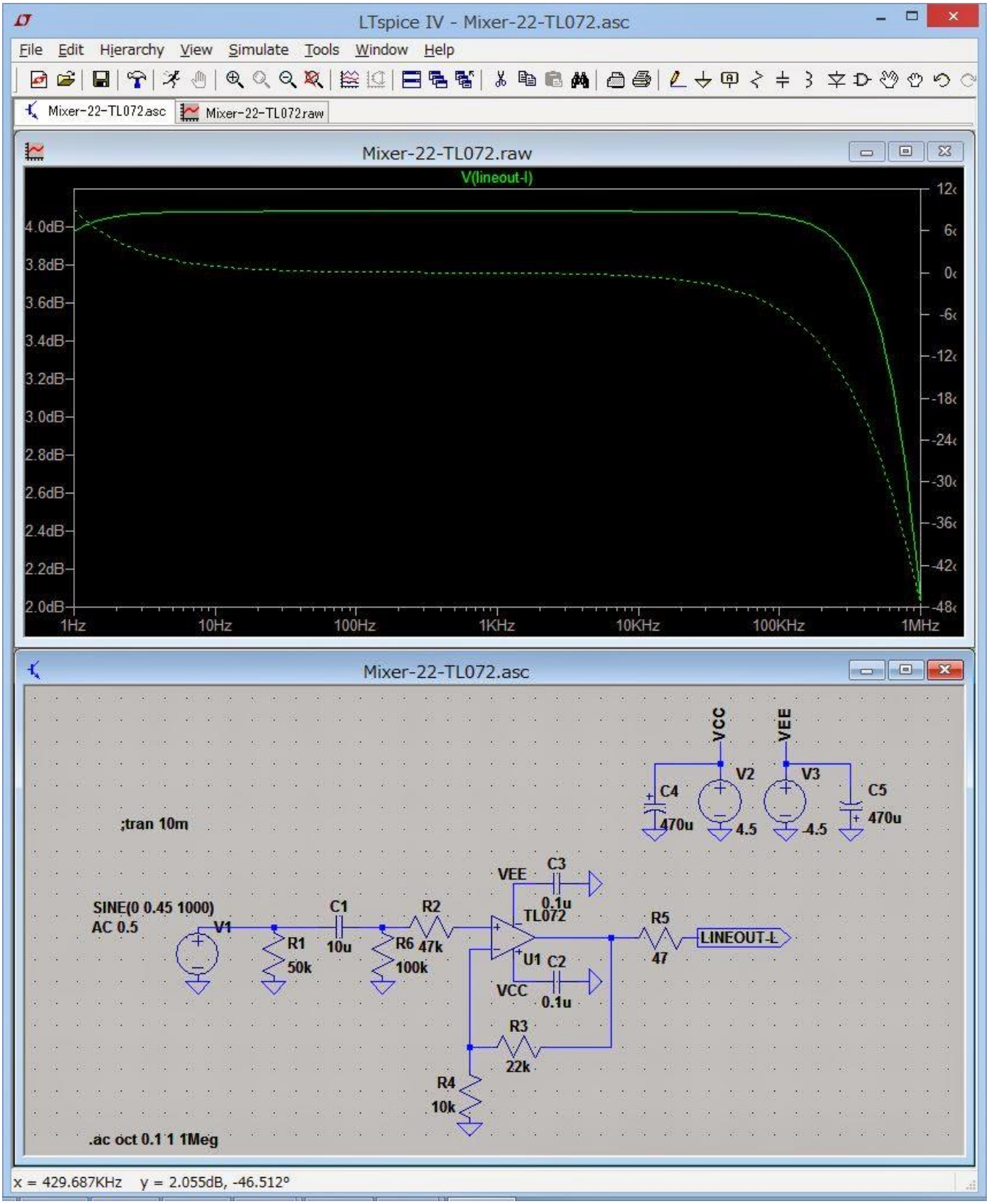
## シミュレーション




ブレッドボードで試作を繰り返していたので、全体のシミュレーションは行わなかった。

周波数特性が問題になりそうな初段のシミュレーションもしてみてみた。

入力のカップリングコンデンサは無極性電解コンデンサを使うことにして 10uF にした。
OPAMP の非反転側は 100kΩの抵抗で接地したので、ここはハイパスフィルターになる。



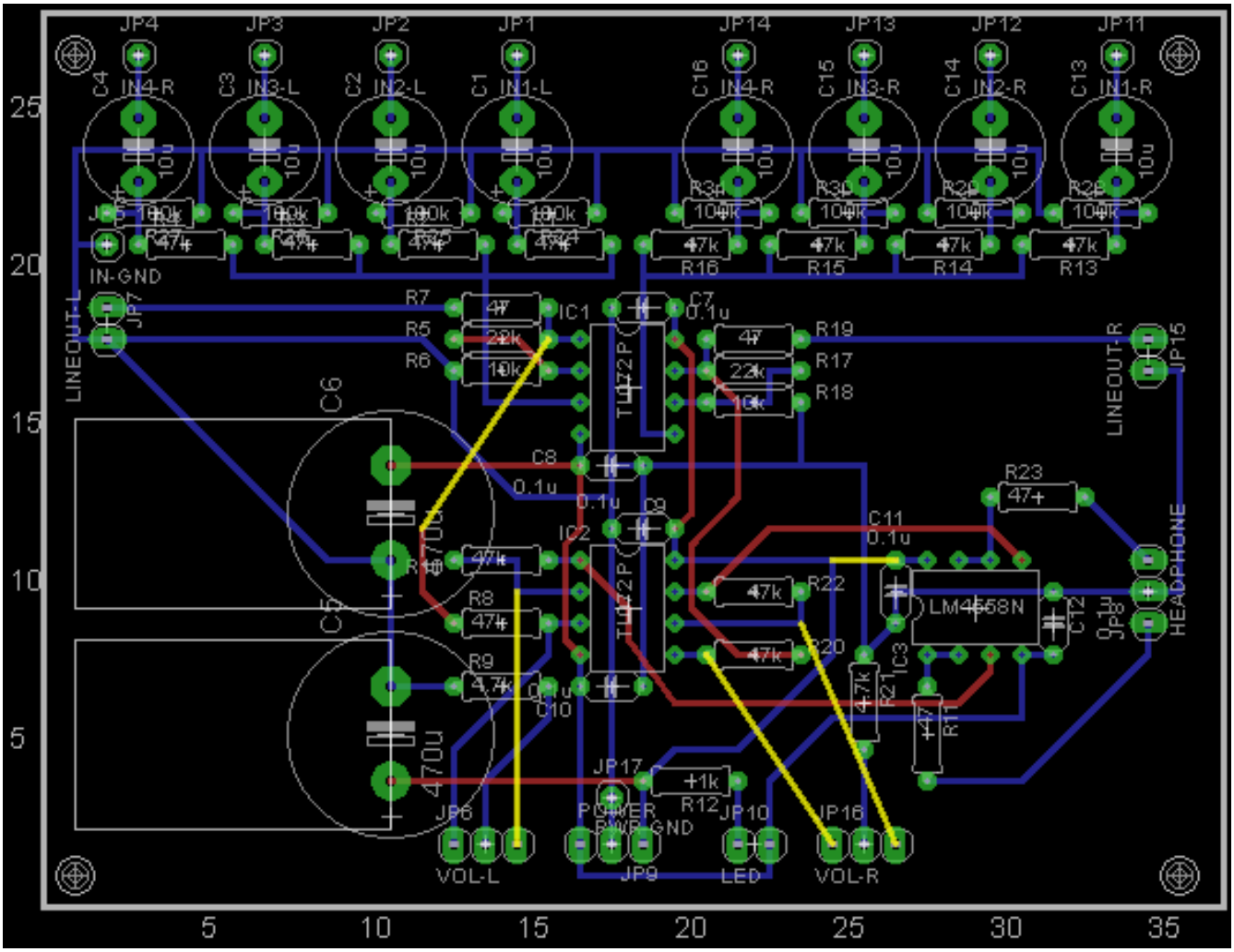
カットオフ周波数は

$$\frac{1}{2\pi CR} \cong 0.16\ [\mathrm{Hz}]$$

なので実用上問題無いはずだ。1uF でもカットオフ周波数は 1.6Hz なので、フィルムコンデンサにしてもいいかもしれない。

## 製作

基板配線図




回路は単純だが部品点数がそれないりに多いので PasS ではなく、はじめて Eagle(www.cadsoftusa.com/)を使った。複雑な回路を自動配線してプリント基板のデータを作成できるのが売りだが、手配線すればユニバーサル基板の配線図の作成にも使えた。

使い方がわからなくて最初は戸惑ったが、以下のサイトを参考にした。

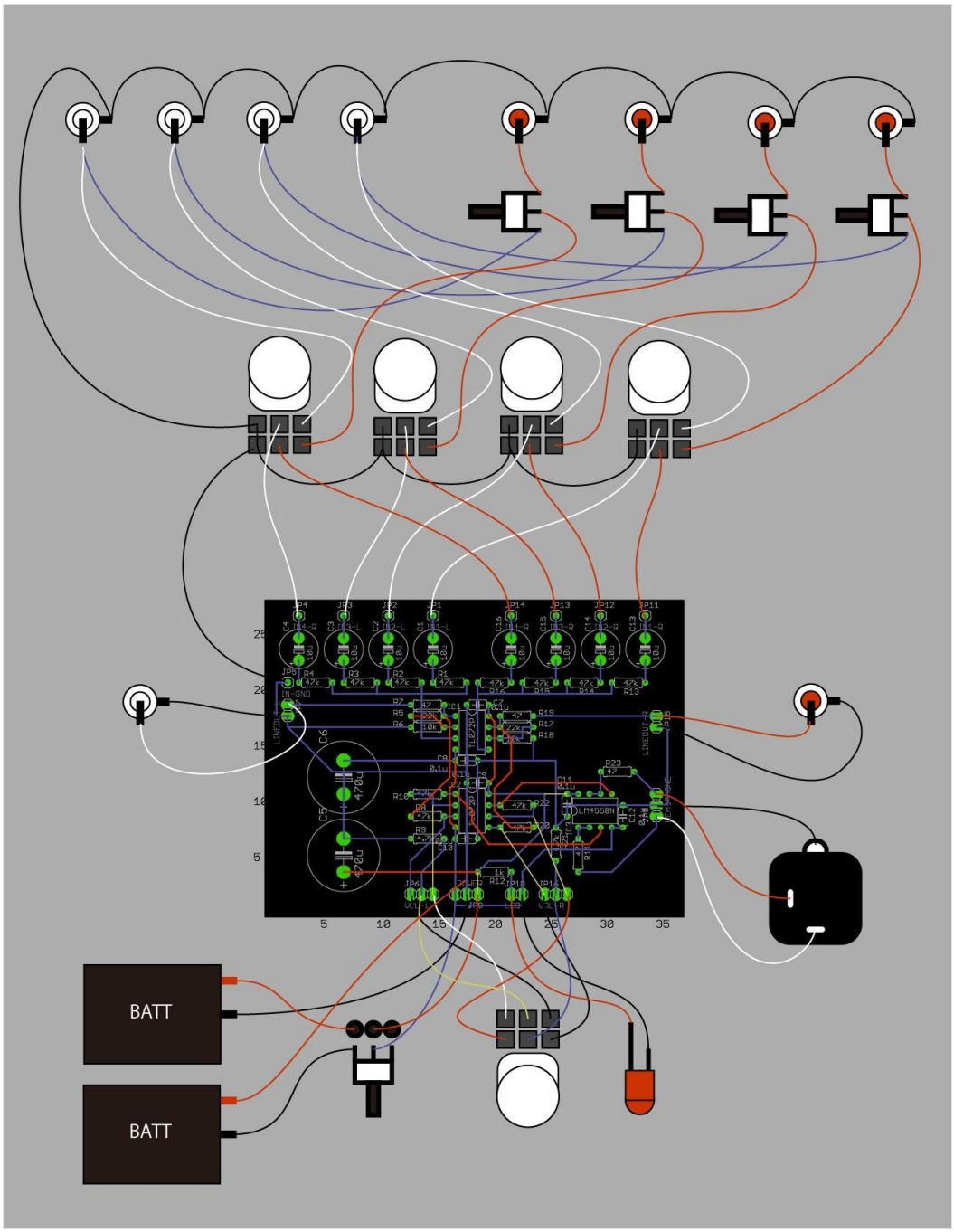
EAGLE6 tutorial JP.pdf
http://homepage3.nifty.com/circuitboards/v2_software/EAGLE/EAGLE6%20tutorial%20JP.pdf

DENSIKIT.COM
http://www.densikit.com/home/eagle

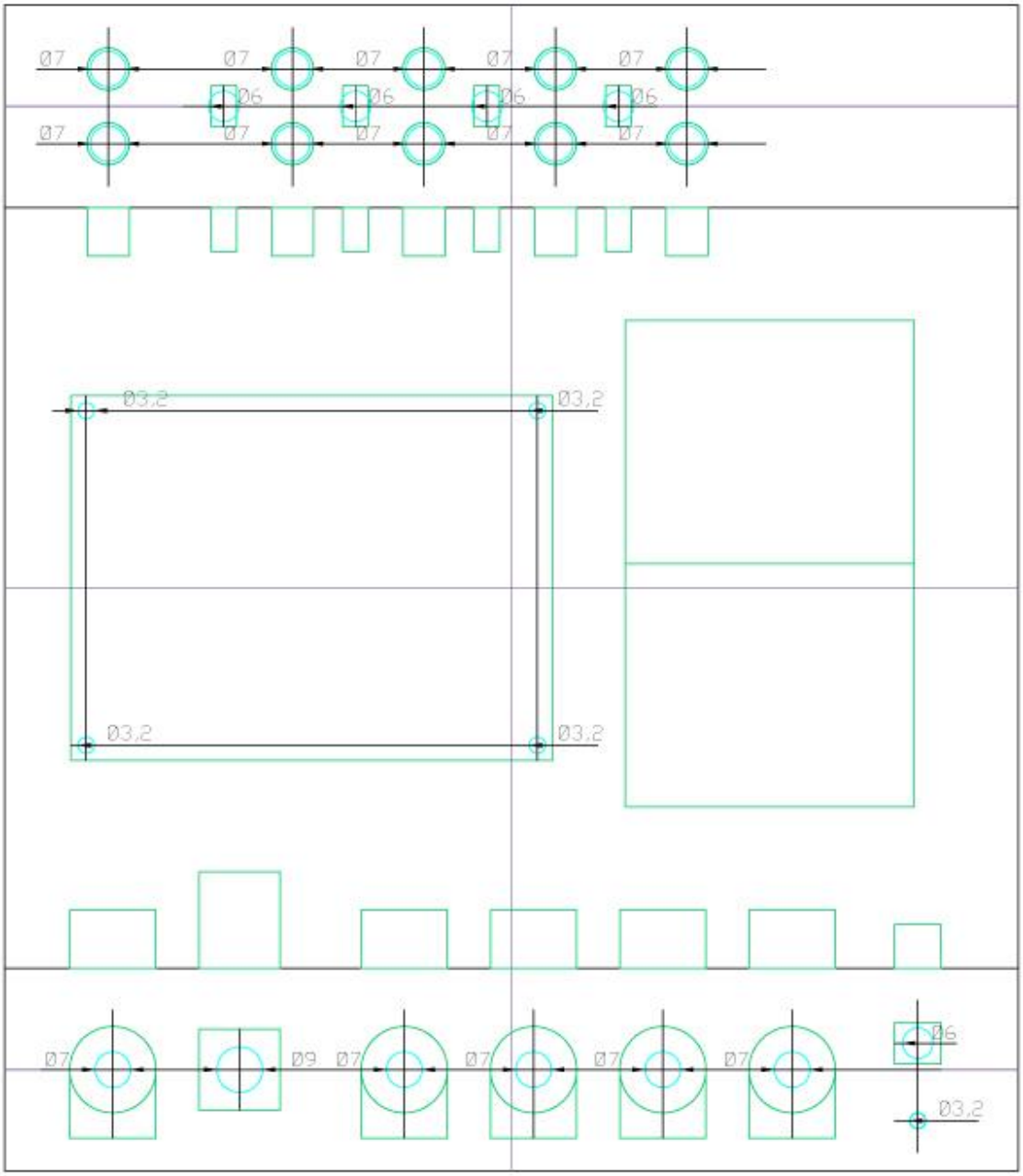
半日もあれば基本的な使い方はわかった。

## 結線図

基板上もさることながら、外付けの部品も多い。
なので、結線図も作成した。手書きだと修正が大変なので仕事で慣れてた Adobe Illustator で作成。
部品の図を描くのが結構面倒だったが次からは使い回しできるかな。



## ケース穴あけ加工図




ケースにはタカチの YM-200 を使った。今回は電池電源のみだがケースに余裕があるので安定化回路を入れて、AC アダプタからの給電もできるかもしれない。電池の減りがひどいようならエネループで使っていく予定。

## 計測

ソフトオシロとタブレットの Fucntion Generator を使って入出力を見た。

### 使用機材

Function Generator（もどき）

- ASUS MeMO Pad HD 7
- FuncGen Signal Generator

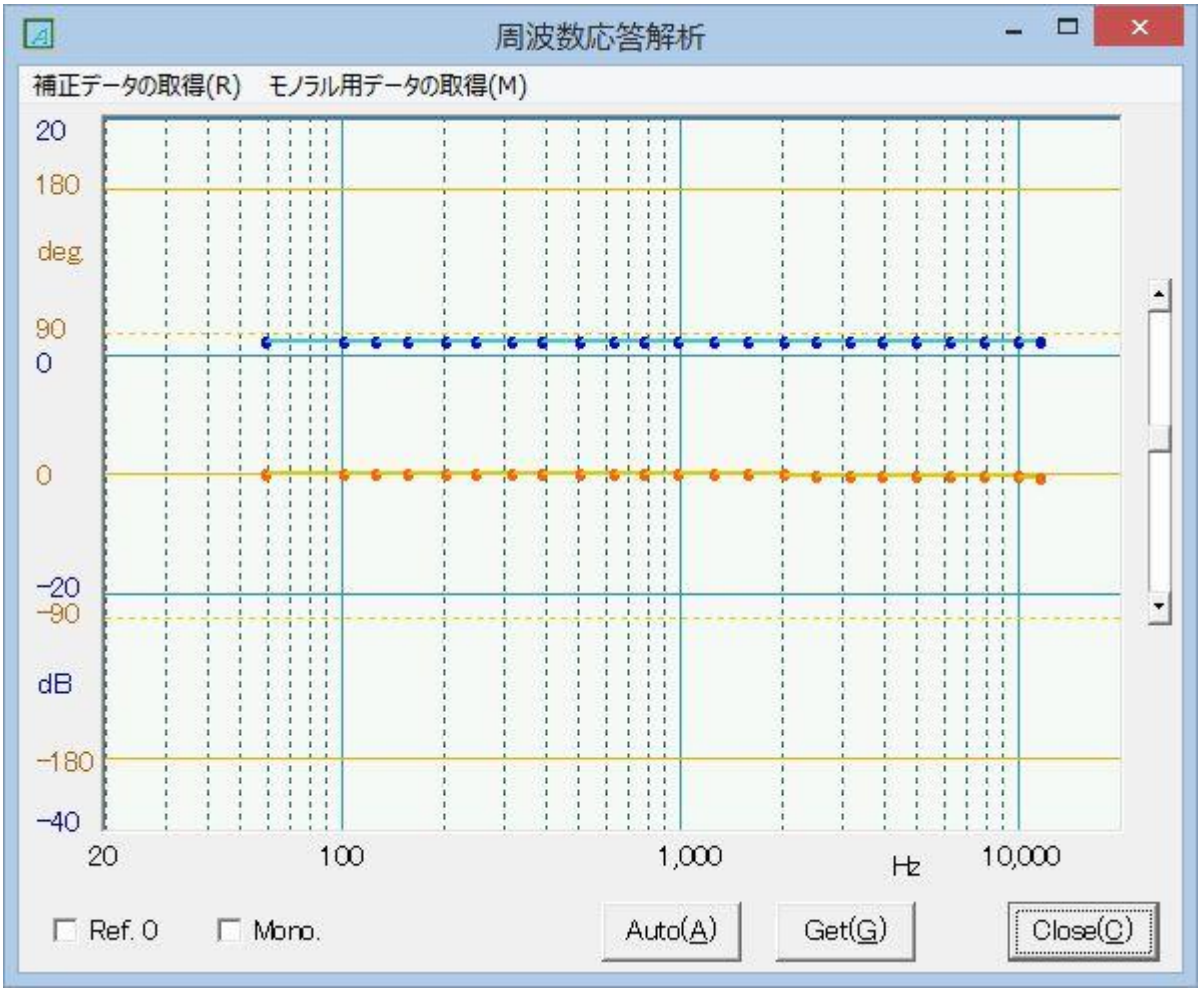
Audio Interface

- TASCAM US-144 MKII / GUITAR 入力（1MΩ）

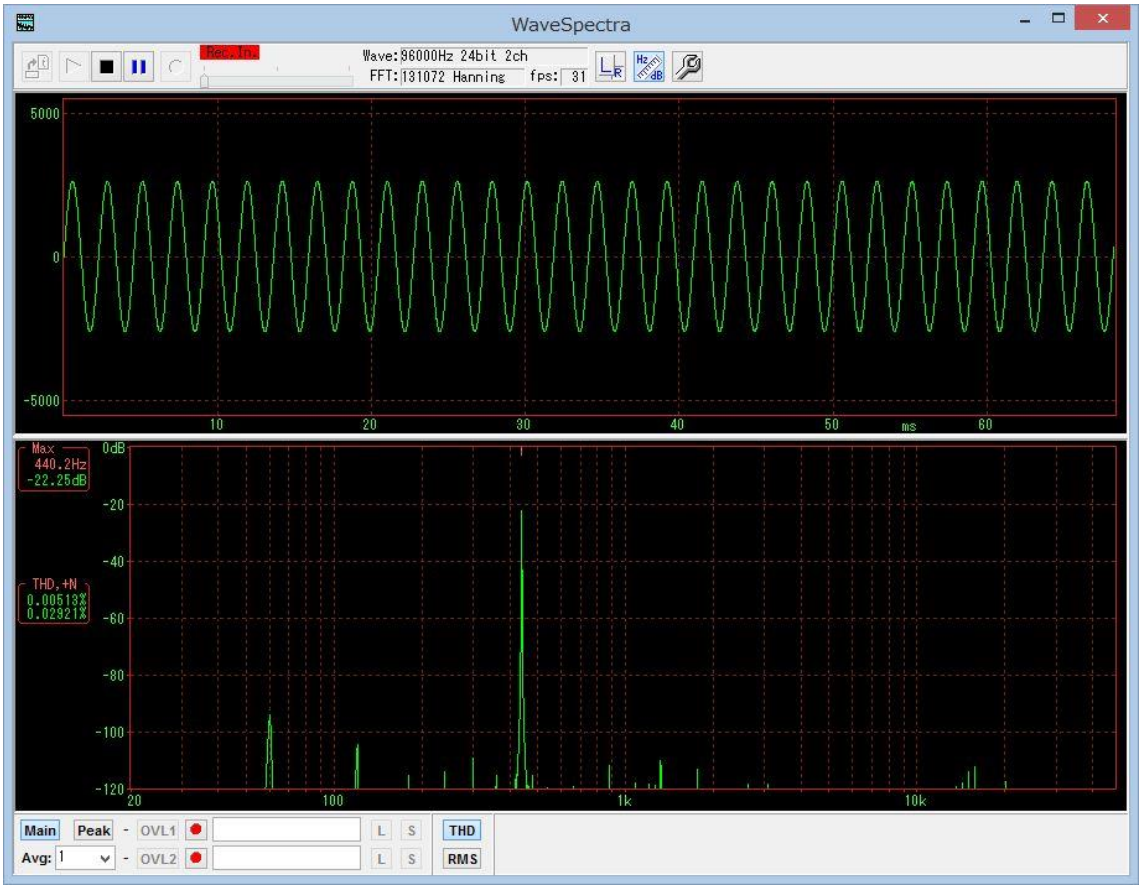
Software Ocillo Scope

- Wave Spectra (http://www.ne.jp/asahi/fa/efu/soft/ws/ws.html)

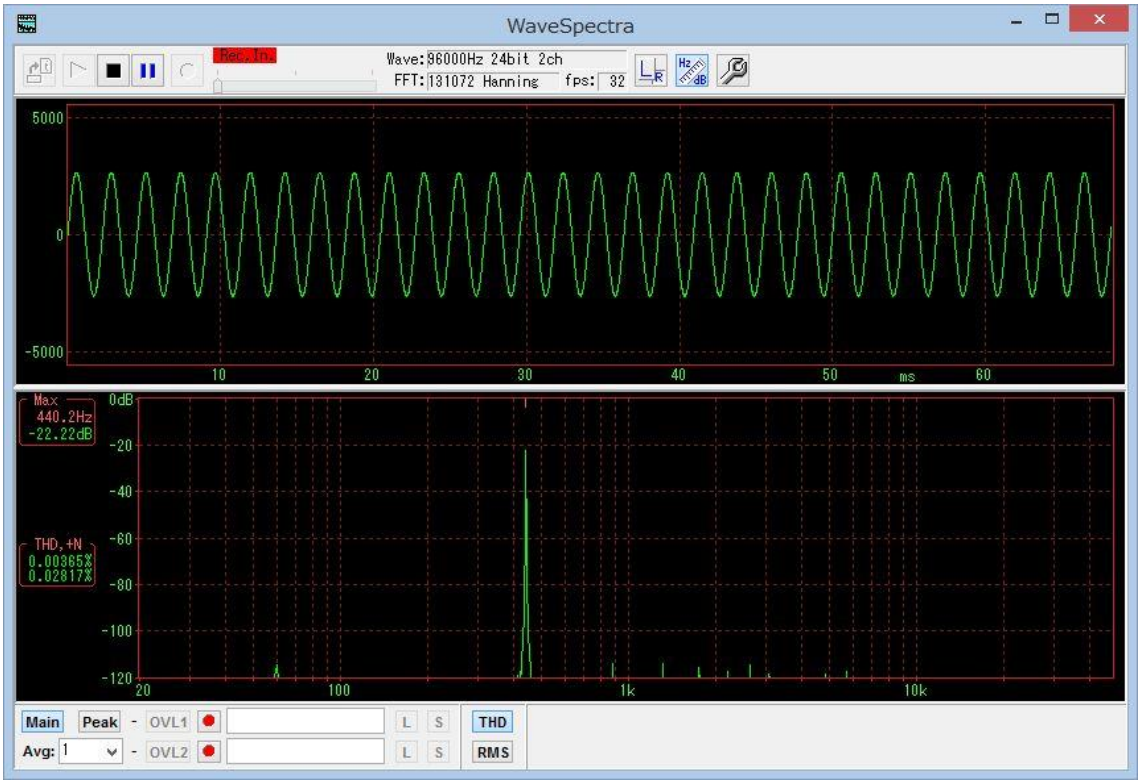
### 周波数特性



440Hz ケースなし




440Hz ケースに収納



アルミケースに入れたおかげでハム音は軽減できた。むき出しの状態だと-95dB ぐらい出てたが、ケースに入れたら-110dB ぐらいまでになった。15kHz ぐらいで出てた謎のノイズもなくなった。

## 反省点

配線材は基板から直出しにしたが、ハンダのし直しが困難。やはりピンヘッダで出しておいた方が無難だと思う。プリント基板なら配線の修正は簡単だがユニバーサル基板だと特にそう思う。

## ご連絡・お問い合せ

blog: http://dad8893.blogspot.com
e-mail: kawanabe@gmail.com

２０１４年8月１６日初版作成