

部報

第49号

電気通信大学  
工学研究部

kōken



# 電気通信大学工学研究部 部報

## 第 49 号

### 目次

---

部長挨拶	2
	横田
オトマトペ	3
	GGE
エレキししおどしの製作	4
	和三盆
2SC1815 と 2SA1015 でオペアンプ自作	5
	横田
エコランカーの作り方（簡易版）	9
	本多
Sikuli を使って GUI アプリケーションを自動化し隊 2	10
	しゅんりゅー
応用情報技術者試験受験記	15
	木村
ミニギターアンプを作ろう	18
	桑原
アンプ IC を用いたお手軽オーディオアンプ作成	21
	聖徳たいし
半自動卓-ダイス付き点数計算表示器-	23
	石関・筒井

## 部長挨拶

---

部長の横田です。

工学研究部（以下、工研）ではイベントに向けて部報を発行しています。

部報は部員がそれぞれ自由に記事を書き

それをまとめたもので、工研には30年以上の歴史があります。

どうぞ部員が丹精を込めて書いた記事をお読みください。

過去の部報は当サークルのホームページに電子版がいくつかあります。

<http://www.koken.club.uec.ac.jp/doc.html>

以下は内部的な話になります。

今年度の工研は活動のテコ入れをいくつかしています。

部報に関しては伝統を守りつつ2つ変更を行いました。

1つ目は技術系の記事であるという制約を設けたことです。

近年では部員の自由性を尊重しすぎたあまり

質の低下を招いてしまいました。

そのため技術系記事を基本としたことで内容の充実と

部員の技術力の向上をはかりました。

2つ目は記事を毎月集めるということです。

以前は年に数回のペースで寄稿してもらっていましたが

記事数が年々減っていました。

定期的に記事を提出する機会があることにより気軽に記事を書くことができ、

年間の投稿数が増えるのではないかという狙いがありました。

今回の部報はこれらの変更がどう記事に反映されたかという

点に着目しても面白いかもしれません。

# オトマトペ

GGE：桑原(知能機械工学科1年), 伊林(知能機械工学科1年)

緒方(知能機械工学科1年), 鈴木(先進理工学科1年)

## 1. オトマトペとは

音符を模した LED を五線譜上のソケットにはめてスイッチを押すと、その音階を奏でます。LED を適当にはめてみた図が以下の通りです。音符の部分は LED の向きを間違えないためにも、プラスチックのボタンに LED を差し込み音符感を出しました。



## 2. 開発動機

音楽を習う子供が最初に躊躇するのが、楽譜に並ぶ無数の音符です。楽器で遊べるおもちゃはあるのに、楽譜で遊べるおもちゃってないよね、という事で作ってみました。

## 3. 回路

回路はマトリクススイッチ回路を用いました。基板上にモジュールを置くことで回路が導通し、マイコンで導通した部分を判断して音を鳴らす、という流れです。蓋になっているアクリル板がペラペラだったので、

そこに穴をあけてソケットをつけ、ナポリタンでソケットを下の基盤をつなげるようしました。すると以下のように、美味しいようになりました。



## 4. 展望

現段階では、五線譜上に LED を差して、出来あがった楽譜通りの音が鳴る仕組みしかありません。しかも、音の種類も一種類で、ビープ音のみです。今後は、MP3 音声再生モジュールを組み入れる等して音の種類を増やしていきたい考えです。いずれは音ゲーみたいな事が出来たら楽しそう…。

## 5. 終わりに

読んでいただきありがとうございました。工研に入部してから初めての物作りだったので、中も外も試行錯誤した雰囲気がとても出ました。助言をいただいた先輩方に最大の感謝を込めてもう一度！ありがとうございました！！

# エレキしおどしの制作

先進理工学科一年 粟島, 知能機械工学科一年 戸田, 情報・通信工学科一年 吉村

## 1 導入

私達は電気通信大学調布祭で行われた第17回エレクトロニクスコンテストに参加しました。その際に制作した「エレキしおどし」の概要及び仕組み等について説明します。

## 2 エレキしおどしについて

エレキしおどしとはしおどしの動作と音出しを電気的に制御するものであり、詳しい仕組みは後述しますが動作の流れはマイコン制御により筒を動作させ筒後部があたる位置のスイッチが押されることでしおどしの音を鳴らすというものです。

## 3 筒動作部

竹筒の動作を筒に取り付けた紐をサーボモータで引くことで制御しました。マイコン制御では2つのモードを用意し、モード1ではサーボモータをゆっくり動かし少しづつ紐を引っ張ることでしおどしから水が少しづつ流れていく際の動作を行わせて次のモード2ではサーボモータが急激に元の位置に戻ることで筒の重心を利用して後部が台座(スイッチ)にあたるようになっています。モードの切り替えはタイマー制御を用いており一定の周期で往復するようになっています。

## 4 音声出力部

筒によってスイッチが押されることでICが起動し音を出力します。ICは共立エレショップで購入した「組み込みタイプ・MP3音声再生モジュール」を使用しました。このICは再生時のスイッチ入力を電圧変化の下降端で認識するようになっており、そのまま使用するとスイッチを押してから離す際に音が鳴るようになっています。これでは筒が台座に

あたって離れる際に音がなる、という動作になってしまふため、ICではなくマイコンでスイッチ入力を受けるようにし割り込み処理でパルスをICに送ることによって求めていた動作であるスイッチを押し込んだ瞬間に音を再生するということを実現しました。このICではmp3もしくはwavファイルの再生が可能です。

## 5 外装

外装は100均ショップで売っているフィギア展示用ケースを用い、筒はホームセンターで購入した竹筒を加工しました。

## 6 まとめ

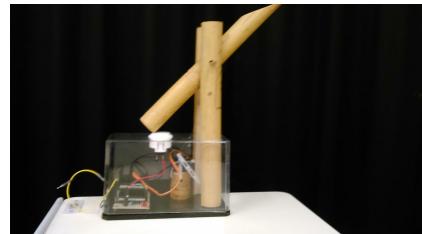


図1 完成作品

### 主に使った部品について

部品	入手場所
atmega328P	秋月電子
サーボモータ	Arduino 初心者キット
mp3再生 IC	共立エレショップ
竹筒	ホームセンター
ケース	百均
スイッチ	三和電子

以下のURLで動画を載せてます(諸事情あって音無し)

<https://www.youtube.com/watch?v=QdFhv1rlpw>  
最後に、エレクトロニクスコンテストの結果では入賞はできませんでした。

# 2SC1815と2SA1015でオペアンプ自作

横田 嶺

2014年12月

## 1 はじめに

オペアンプ(演算増幅器)とはその名の通りアナログで演算することのできる増幅器で通常はブラックボックスとして扱います。しかし中身はトランジスタで構成されているのでディスクリートでオペアンプを作れるのではないかでしょうか。幸いにも世に溢れているオペアンプICのデータシートには等価回路が記載されています。これを参考に秋葉原で手に入る電子工作で一般的なNPNトランジスタの2SC1815とPNPトランジスタの2SA1015でオペアンプを制作しようと思います。

## 2 オペアンプとは

オペアンプは主に電圧を用いて演算することができる増幅器です。オペアンプの基本は2入力1出力で入力端子の電位差を増幅して出力することです。これを式で表すと非反転入力端子の電圧を $V_+$ 、反転入力端子の電圧を $V_-$ 、増幅率 $A$ として出力電圧 $V_{out}$ は

$$V_{out} = A(V_+ - V_-) \quad (1)$$

となります。この理想オペアンプの等価回路を示します。

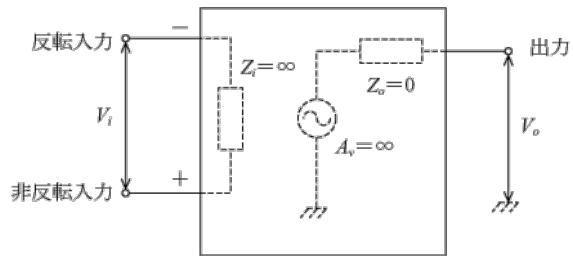


図1: 理想オペアンプの等価回路

ここで理想オペアンプは

- 入力インピーダンス無限大
- 出力インピーダンス0
- 差動利得無限大
- 同相利得0
- 同相信号除去比無限大
- 周波数帯域無限大
- 内部雑音0

などのようになっていますが実際のオペアンプはそれぞれ有限の値を持ちます。

それぞれの回路は割愛しますが外部に抵抗やコンデンサを付けることで加減算や微分積分演算ができます。

## 3 オペアンプ自作

ここで本題ですが理想オペアンプの等価回路からも分かるように、「差動入力」、「と

ても大きな利得」、「高入力インピーダンス」、「低出力インピーダンス」という方針を立てて設計していきます。

ここで差動入力、高入力インピーダンスを実現するためにトランジスタで「差動増幅回路」を入力段にすることにしましょう。次に利得を稼ぐために「エミッタ接地増幅回路」で振幅を大きくしたのち、出力インピーダンスを下げるために「プッシュプルエミッタフォロワ出力」にしようと思います。

では実際に作られているオペアンプ IC の等価回路はどうなっているのか見てみましょう。

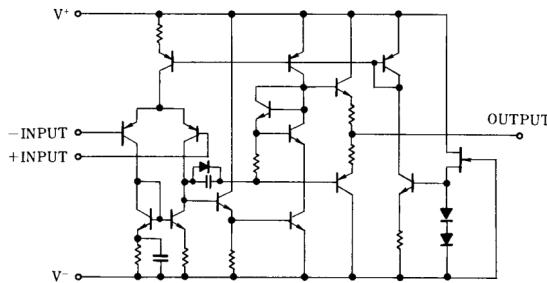


図 2: オペアンプ NJM4580 の等価回路

このオペアンプはオーディオ向けですが汎用でも使えるものです。このままではよく分からないので少し簡略化したものを図示します。

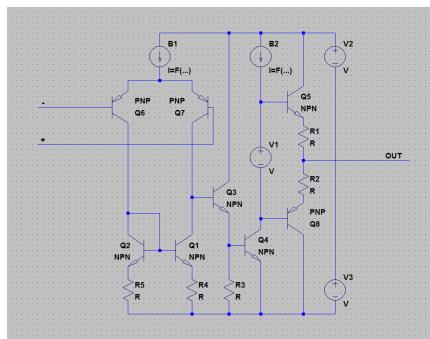


図 3: 簡略化した等価回路

こう見ると「差動増幅回路」、「エミッタフォロワ」、「エミッタ接地回路」、「プッシュプル出力」となっています。間にエミッタフォロワが 1 段入っているのは差動増幅回路の出力インピーダンスを下げ次段の入力容量とローパスフィルタを形成しないようになります。

次にそれぞれの段の構成を考えてみます。

### 3.1 差動増幅回路

オペアンプの肝とも言える差動増幅回路を考えます。差動増幅回路は定電流源によって 2 つのエミッタ電流の和が一定になるような回路です。2 つのエミッタ電流を  $I_{E1}, I_{E2}$  として定電流源の電流が  $2I_E$  とするとエミッタ電流の変化分は同じで増減の方向だけが異なる回路になります。

$$\begin{aligned} 2I_E &= I_{E1} + I_{E2} \\ &= I_E + I_E \\ &= (I_E + \Delta I_E) + (I_E - \Delta I_E) \end{aligned}$$

したがってベースエミッタ間電圧  $V_{BE}$  はそれぞれ

$$\begin{aligned} V_{BE1} &= V_{BE} + \Delta V_{BE} \\ V_{BE2} &= V_{BE} - \Delta V_{BE} \end{aligned}$$

となりコレクタ電位は振幅が同じで位相が反転します。ここで  $2\Delta V_{BE}$  が 2 つの入力の電位差になりますから差動増幅回路はエミッタ接地回路の半分の利得になります。

差動増幅回路の負荷として接続されているのはカレントミラー回路という定電流回路です。これは両側の電流が等しくなるもので電流源を負荷として繋ぐとインピーダンスが非常に高くなり出力として取り出せる電圧が上がり利得が大きくなります。

### 3.2 エミッタフォロワ回路

コレクタ接地回路とも言い、電流を増幅してくれる回路です。この場合は出力イン

ピーダンスを下げるバッファアンプとして動作します。

### 3.3 エミッタ接地回路

お馴染みの電圧を増幅する回路です。こちらも出力インピーダンスが高く電流も小さいため次段にプッシュプルエミッタフォワード回路を付けて出力します。

### 3.4 プッシュプル出力回路

NPNトランジスタとPNPトランジスタを上下に接続することで出力インピーダンスを小さくします。ベース電位が高いときは吐き出し、低いときは吸い込むという動作をするためプッシュプルと言います。

## 4 部品の選定

この回路では特性の一致したNPNトランジスタとPNPトランジスタが必要です。今回はNPNトランジスタに2SC1815, PNPトランジスタに2SA1015を用いることにします。

## 5 シミュレーション

シミュレーションに用いた回路を示します。オペアンプは非常に大きな利得を持っているためそのまま入力に信号を入れただけでは出力を見ることは出来ません。そこで帰還回路を外につけることが必要になります。帰還回路によって全体の利得を調整することができます。ここでは利得5倍の反転増幅回路をシミュレーションしてみました。

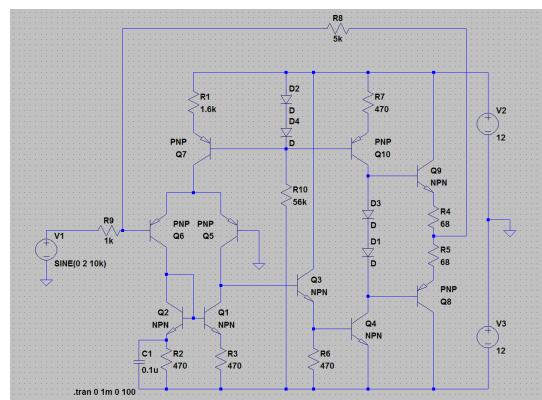


図4: ディスクリートオペアンプによる反転増幅回路

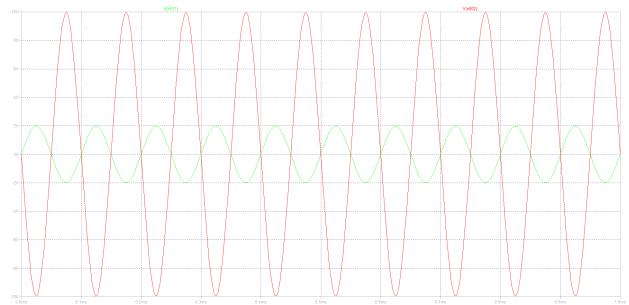


図5: 反転増幅回路の入出力波形

見づらいですが入力  $V_{pp} = 4V$  に対し出力が  $V_{pp} = 10V$  で位相が反転していることが分かります。

次に周波数特性をシミュレーションしてみました。

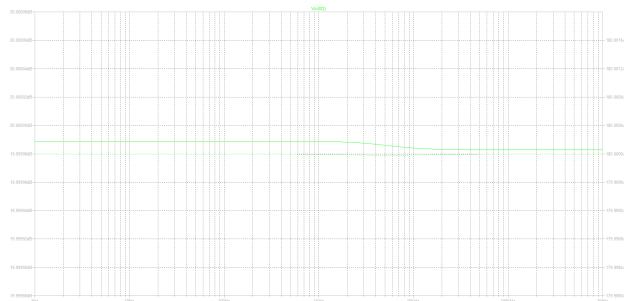


図6: ディスクリートオペアンプの周波数特性

シミュレーションでは理想的な素子を使ったため周波数特性がほとんど変化しない素晴らしいオペアンプが出来てしまいました。私がまだ回路シミュレータ LTspice に不慣れなだけなのですが、トランジスタやダイオード、抵抗のモデルをきちんと作ってやれば現実的な結果が出ると思います。

## 6 回路作成

この回路とシミュレーション結果を受けて実際に基板をおこしディスクリートでオペアンプを自作してみようと思います。

Coming Soon...

## 7 感想

私が1年生の頃オペアンプの等価回路を見たとき電流源が含まれていたりトランジスタの使い方も複雑で全然理解できませんでした。トランジスタ回路やそれを構成する半導体のことを学び、等価回路をいくつかに分解することでオペアンプを多少理解することができ感慨深いものがあります。

今回の部報を書くにあたり回路シミュレータを導入してみました。回路シミュレータは以前も使ったことがありますが回路設計に役立てようとしたのは今回が初めてです。使ってみて感じたのはアナログの解析が強力で今回のような複雑なアナログ回路の設計には非常に有用であるということです。マイコンなどのデジタル回路では動作の想像がつきやすいのですが、アナログでは電圧だけでなく電流の変動なども考える必要があり手計算で求めるのが困難な場合シミュレーションが力を発揮するのだなと思いました。

これから基板を設計して実際に作りシミュレーションとの比較をし、オペアンプについてさらに理解を深めたいと思います。

## 参考文献

- [1] 鈴木雅臣「定本トランジスタ回路の設計」1991,CQ出版社
- [2] 公益社団法人日本電気技術協会ウェブページ
- [3] NJM4580 データシート, 新日本無線

## エコランカーの作り方（簡易版）

M科 3年 本多寿矢

## 1. 発案

どのような車体を作りたいか（フレーム構造 or モノコック構造など）、手書きでおおまかなスケッチをまとめた。

## 2. データ収集

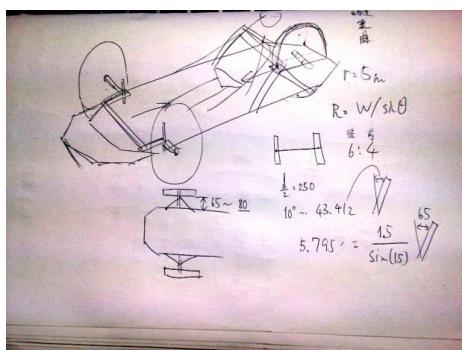


図 1 車体スケッチ

使用するタイヤの径、材料の種類、搭乗者のサイズなどを調べた。

## 3. 設計

大潟村スポーツラインのコーナー半径が 7m なので、次の式から必要なホイールベースを求めた。

$$R = \frac{W}{\sin \theta}$$

R: 回転半径, W: ホイールベース,

$\theta$ : タイヤの切り角

また、切り角はタイヤと車体の幅をどれだけ離すかを考えて決定した。タイヤの半径がおよそ 250mm だったので、車体との幅は  $250 \times \sin 15^\circ \approx 65\text{mm}$  となった。

## 4. 制作

アルミ角パイプとアルミ板を購入した。設計図に従ってけがき、切断、穴あけを行った。その後仮組みを行い、各部の修正をして、組み立てを行った。



図 2 完成した車体

## 5. 出場

走行試験を行い、微調整をしてから大会に出場した。



図 3 WEM の様子

## 6. 感想

次の機会があればもっと要領よく進めたい。また、溶接を使った車体の製作も試みたいと思う。

# Sikuli を使って GUI アプリケーションを自動化し隊2

著者：しゅんりゅー

## 1 実際に何かに使ってみよう

さて、毎月連載とか言いながらさっそく4月号はサボってしまいました。だって仕方ないじゃん、就活やエコランで大変だったんだもん……。気を取り直して、Sikuli の紹介を続けていきます。

前回で簡単な使い方は学べました。今回はもっと複雑な処理をさせてみたいものです。それに当たっては GUI 操作アプリケーションである以上、何か教材として使えそうなアプリケーションが欲しいところです。しかし、キーボード操作のできるようなアプリでは Sikuli を使う必要性が薄く、なかなか適切な教材が……



図 1: 艦隊これくしょん -艦これ-

ありました（真顔）。キーボード操作に対応していない flash アプリケーションであり、今の世を賑わせているこのゲーム。なかなか Sikuli の教材としては良いのではないでしょうか。

ということで、今回は艦これを使ってちょっと色々試していきます。

なお注意して欲しいのは、艦これはマクロなどの自動操作を公式で禁止している点です。今回は教材として少し利用するのに留めますが、間違っても自動出撃・自動遠征のスクリプトを組んで常に実行させることをしてはイケマセンヨ？（苦笑）

ここからは必然的に艦これをしている人向けの話になりますが、やったことない人はなんとなくやっていることを理解してもらえれば。

## 1.1 前回の補足：マウスが動かなかったよ、という人向け

そういえば、4月のXPサポート切れに合わせてOSを7に変えました。そして同じくSikuliを使ってみたのですが、最初マウス動作が機能せずに少し焦りました。その原因是、Windowsの管理者権限が関係しているようです。動かなかった人は、SikuliのIDEを立ち上げるとき、「runIDE.cmd」をダブルクリックではなく、右クリックから管理者として実行しましょう。うまくいくかもしれません。あるいはファイルのプロパティで、常に管理者権限で実行するよう設定しておくといいかもしれません。

## 2 廃棄選択を効率化しよう

保有できる装備品には上限がありますから、艦これをやっている人は当然、装備品の廃棄をしたことがあると思います。しかしこれが面倒くさい。図2にありますように、左側のチェックボックスを一つ一つ手で選択して、「廃棄する」ボタンを押さなければなりません。「補給」項目と違ってページごとの一括選択ボタンもないで、すべて手で押さなければいけない。廃棄する装備品がたくさんあると、正直苦行です。

こんな悩みを、Sikuliを使って解決しましょう！



図2: 廃棄装備の選択画面

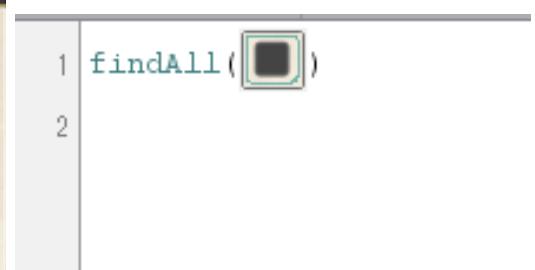


図3: findAllの選択結果

### 2.1 手順1：押すべきチェックボックスを探す

まず、押したいチェックボックスを画像検索で探します。このような時に使うのは、Sikuliメインウィンドウ左の、「find」「findAll」関数です。詳しい説明は前号のおまけページを参照してもらうとして、簡単に説明すると、「find」はスクリーン上で指定された画像を検索する関数で、「findAll」はその強化版、スクリーン上で一致する画像を全て探します。

今回、まずは押すべきチェックボックスを全て探したいので、「findAll」を使います。前回の「click」と同じく、「findAll」ボタンを押すと、画像選択モードになりますので、チェックボックス部分を一つ選択しましょう。図3のような感じになります。

findAll関数は引数として渡した画像と一致した全画像の、Regionオブジェクト（位置のようなもの）のリストを返します。返されたリストを、まずは適当な変数に保存しておきます。今回はCheckBoxListという変数に保存しておくことにして、「findAll」の左側に「CheckBoxList=」と書きます。

## 2.2 手順 2 : for で走査

### 2.2.1 foreachってなに？

次の手順に移る前に、まず foreach という構文について説明します。これは C 言語にはない構文なので、C しか使ったことのない人は見慣れない構文だと思います。C に実装されていないだけで、C 以外では割と色々な言語に実装されている構文です。

foreach はループ構文の 1 種で、配列やリストの全要素に対して何かしらの演算を行うときに使います。C で for 構文を使って配列の要素に対する演算をするときは、次のように書くと思います。

```
-----  
int Array[10]={0,1,2,3,4,5,6,7,8,9}; // 配列の宣言  
int i,hoge;  
for(i=0; i<10; i++){  
    hoge = Array[i];  
    ...  
}  
-----
```

しかし、ここで前提が一つ覆ると大きな問題が発生します。それは、【もし Array が大きさ 10 とは限らなかつたら？】というものです。今上で書いたコードは、宣言時に大きさ 10 としているので、上記コードで問題ありません。しかし、このような事実がソースコード上から簡単に読み取れないとき、for 構文を使う場合は何回ループすればいいのかを調べなければなりません。

そのような時こそ、foreach 構文の登場です。上記コードを C# の foreach 構文で書き直すと、次のようにになります。

```
-----  
int[] Array = new int[] {0,1,2,3,4,5,6,7,8,9}; // 配列の宣言  
foreach(int hoge in Array){  
    ...  
}  
-----
```

このコードは先ほどの C で書かれたコードと例と本質的に等価です。このようにすると、Array に含まれる要素を 1 番目から最後までを順に変数 hoge に代入しつつループします。1 回目のループでは hoge に、Array の 1 番目要素である「0」が代入されたループになり、2 回目のループでは hoge に「1」の代入されたループになり、その変数 hoge をループ内で使用することができます。

ここで重要なのは、for 構文と違ってループ回数を指定していないことです。foreach 構文ではループの回数は、構文に渡した配列の要素数によって決定されます。配列の要素数が 10 なら 10 回ループしますし、20 なら 20 回になります。オブジェクト指向を有する言語では非常に有用な構文なので、覚えておきましょう。

さて、今回扱う Pythonにおいては、foreach 構文はないのですが、for 構文が foreach 構文と等価になっています。そのため逆に Python では、C ライクな for の書き方 ( for(i=0; i < n; i++) 的な ) ができません。その代替法はあるのですが、まあそれは置いといて。上記の構文を Python で書くと、次のようになります。

```
-----  
Array = [0,1,2,3,4,5,6,7,8,9] # 配列の宣言  
for hoge in Array :  
    ...  
-----
```

## 2.2.2 本題

さて話を戻します。先ほど、チェックボックスの全部の位置を CheckBoxList という変数に保存しました。これは、チェックボックスの位置が全て保存されたリストになっています。今回、このチェックボックスの全てをクリックしたいので、リストの全要素に対する操作であり、foreach 構文の出番になります。適当なループ一時変数として ChaecckBoxPosition を用意し、そこに順にチェックボックス情報を格納してループさせます。さらに、このチェックボックス位置をクリックさせたいのですから、click 関数にこの変数を渡します。最終的なコードは、図4のようになります。

```
1 CheckBoxList = findAll()
2 for CheckBoxPosition in CheckBoxList:
3     click(CheckBoxPosition)
```

図 4: 組みあがったコード (1)

話を忘れてましたが、Python におけるブロック（C における {} で区切られた箇所）はインデント（コードを書き始める位置）で区別されます。同じ位置から書き始められたコードは同じブロックとみなされます。

## 3 実行！からのデバック

では出来上がったコードを実行してみます。艦これの装備廃棄画面を出した状態で、前回同様 Sikuli のスタートボタンを押して実行します。

……で、如何でしたか？問題なければ、マウスカーソルが勝手に動いて、全部のチェックボックスを押してくれたはずです。

しかし、誰もが「クッソ遅えwww」と思ったことでしょう。こんな速度では島風ちゃんに「おっそーいっ！」と笑われてしまいますがね。

この遅さは耐え難いので、マウスの移動速度を上げてやることにしましょう。Sikuli には、動作にかかる環境パラメータとして Setting クラスがあります。ここではマウスの動作速度、画面スキャンの頻度、画像スキャンの精度などを設定することができます。詳しくは Sikuli の説明ページを見てもらうとして、マウスの動作速度を決めるパラメータは「Settings.MoveMouseDelay」になります。ここに代入した値で動作速度が前後し、デフォルトでは 0.5 秒です。これをとりあえず 0.05 秒くらいにして、プログラムの文頭に書いておきます。

あとは、押す順番がめちゃくちゃだったのが気になる人もいるかと思います。そういう人は、for 文の手前に「CheckBoxList = sorted(CheckBoxList, key=lambda m:m.y)」と書いておいてください。これは説明が面倒なのでおまじないと思って下さい。簡単に説明すると、Python のソート関数を使って画像の位置の y 座標順に並び替えてます。並び替えの順番定義にラムダ式（無名関数）という概念が使われています。これも詳しくは書かないで気になる人は 今のキーワードで調べてください おまけで後ろに書きました。

さて、コードの完成です。図5のようになります。同じく実行してみて、如何でしょうか。これなら島風ちゃんもびっくりのクリック速度です。廃棄が捲るよ！やったね島風ちゃん！おいやめろ

この例だと無条件で全選択を行いますから、廃棄するかは自分でよく見て決定してください。レア装備を廃棄しても私は責任取ません。

```

1 Settings.MoveMouseDelay = 0.05
2 CheckBoxList = findAll(
3 CheckBoxList = sorted(CheckBoxList, key=lambda m:m.y)
4 for CheckBoxPosition in CheckBoxList:
5     click(CheckBoxPosition)

```

図 5: 組みあがったコード (2)

## 4 おまけ: ラムダ式 (無名関数)

余白がとても余ったので雑談でも書いておきます。

先の sorted 関数は、引数に並び替え元となるリスト（仮称 List）と、その並び替えの順番を定義する関数（仮称 Func）を引数に取ります。その定義方法とは、「関数 Func に List の要素を渡したときの、その返り値の昇順」です。ややこしいので次の例を考えます。

---

```

def Func( hoge ) :    # 関数「Func」の宣言、引数 hoge
    return hoge        # hoge を返して終了
# ここからメインプログラム
List = [1,5,2,3,8,6]          # でたらめな数値列
List = sorted(List, key=Func)  # Func の返り値の昇順に並び替える

```

---

上記のように書くと、List 内の要素が昇順に並び替えられます。この例では「関数 Func に List の要素を渡したときの、その返り値」は、そのリストの要素そのものです。すなわちリストの 1 番目要素なら 1 を返し、2 番目要素なら 5 を返します。その返り値の昇順に並びかえますから、結果的にもとのリストを昇順に並び替えることができます。

では降順に並び替えるには？ それなら、引数として渡された値に-1 をかけた関数を定義すればいいです。すなわち上記例の、「return hoge」の部分を「return -hoge」にすればいいだけです。こうすると、1 を渡されれば-1 を返し、5 を渡されれば-5 かを返します。この返り値の昇順ですから、結果的にもとのリストを降順に並び替えることになります。

さて、本題のラムダ式ですが、上記のようにたった 1,2 行で終わるような関数をわざわざ用意するのは非常に冗長、ということで作られた構文です。これは上記の、並び替えの順番定義の関数を書くところ (key=Func) に、「もうここに直接関数を書いちゃえよ」というようなもののです。ここでしか使い道のない関数なので、名前をつける必要すらない。だから「無名関数」と呼ばれます。とは言っても関数であることをプログラムを見て分かるように、共通キーワードである「lambda (ラムダ)」という関数名を与えてやります。

先の Sikuli プログラム例は「CheckBoxList = sorted(CheckBoxList, key=lambda m:m.y)」でした。「lambda」は関数名であり、無名関数であることを示します。これに続けて書かれた「m」は、引数名です。上記 Func 関数の hoge に相当し、「CheckBoxList」の要素一つ一つ、Region オブジェクト（画像位置情報のようなもの）が格納されていきます。最後にこの lambda 関数の返り値として、「m.y」を返しています。これは、その画像位置情報の y 座標プロパティ数値を返している、ということです。

結果的にこれは、画像リストの y 座標昇順に並び替える、といった意味になります。こうしたラムダ式（無名関数）は、C# や C++ など多数の言語のほか、関数型言語でも非常によく使われます。こうした記法があることを覚えておくと、いずれ役に立つと思います。

……もう余白がない。最後に一言、金剛ちゃんは俺の嫁。榛名改二マダー？

# 応用情報技術者試験受験記

情報・通信工学科 3年 木村敢

今年の 4/20(日) に受験し、合格した応用技術者試験について書きたいと思います。

## 1 応用情報技術者とは

応用情報技術者とは情報処理技術者試験の中の一つの試験であり、独立行政法人情報処理推進機構(略称 IPA)が実施している国家資格試験です。

### 1.1 資格所持によるメリット

資格所持者は IPA より

高度 IT 人材となるために必要な応用的知識・技能をもち、高度 IT 人材としての方向性を確立した者

と認められます。また、この資格の有資格者は企業が採用の際に考慮する項目でもあります。さらに、他の国家資格試験(中小企業診断士、弁理士)の一部試験免除が得られます。教員採用先行試験の一部免除が受けられる県もあるそうです。また、一部県警においてコンピュータ犯罪捜査官やサイバー犯罪捜査官の応募資格もあります。

まあ、持っておいて不便はなく、就職に役に立つ資格です。

### 1.2 種類

情報処理技術者試験には以下のような資格があります。括弧の中は略称です。

- IT パスポート試験 (IP)
- 基本情報技術者 (FE)
- 応用情報技術者 (AP)
- IT ストラテジスト (ST)
- システムアーキテクト (SA)
- プロジェクトマネージャ (PM)
- ネットワークスペシャリスト (NW)

- データベーススペシャリスト (DB)
- エンベデッドシステムスペシャリスト (ES)
- 情報セキュリティスペシャリスト (SC)
- IT サービスマネージャ(SM)
- システム監査技術者 (AU)

FE は基本的な知識、技能を AP は応用的な知識、技能が問われます。IP は少し毛色が違い、IT を利活用する社会人に求められる基礎知識が問われます。また、これら以外の ST などの試験はスペシャリストと呼ばれ、高度な知識、技能が問われます。難易度は早い順に IP, FE, AP, スペシャリストとなります。詳しいことは IPA のホームページに書いてあるので興味がある人は見てみるといいでしょう。

### 1.3 試験日程

IP は随時実施されていますが、他の試験は違い毎年 4 月と 10 月の第三日曜日に行われます。基本情報技術者、応用情報技術者は両方とも試験が行われますが、スペシャリストの中には 4 月のみ若しくは 10 月のみしか行われないものもあります。

## 2 内容

応用情報技術者は以下の知識が問われます。

- テクノロジ系  
基礎理論や、コンピュータシステム、技術要素、開発技術について問われます
- マネジメント系  
プロジェクトマネジメント、サービスマネジメントについて問われます
- ストラテジ系  
システム戦略、経営戦略、企業と法務について問われます。

試験は午前と午後に分かれていて、午前は全問選択肢問題、午後は記述問題になっています。午前午後ともに 100 点満点です。

なお、合格点は午前午後共に 6 割の 60 点になっています。また、午前試験で 60 点に満たなかつた場合は午後は採点されません。

## 3 勉強方法

個人個人で合う勉強法は異なると思うので、私がした勉強法を書きます。

まず、応用情報技術者用の参考書を買います。私が購入したのは Ohm 社の”2013 年版 応用情報技術者標準教科書”です。正直あまり良くなかったので別の参考書を買うことをオススメします。ネットでオススメを探してもいいですし、実際に少し立ち読みなどをして決めるのがいいでしょう。

う。買ったら、一周は読みます、内容は全部理解しないでいいのでとりあえず読みます。読んだら一度午前試験の過去問をやります。過去問は IPA が解答と共に公開しているのでそれを使います。やり方としては、解らない問題が出たら参考書を読みながら問題を解く方法で解いていきます。多分 4 割も取れないと思うので、少し焦ります。そこからわからなかったところを中心に参考書を読み直します。あとはひたすら演習です。演習にはこの”応用情報技術者ドットコム”と言うサイトがいいと思います。このサイトに”Web アプリ過去問道場”というものがあり、これでひたすら午前試験の過去問の演習ができます。問題の解説もついているのでかなりいいです。私が午前試験を通過できたのはこのサイトのおかげと行っても過言ではありません。

これで午前試験の対策は大丈夫です。午後試験は記述問題があるので自分の実力がしっかり出ます。なので一夜漬けで合格等は難しいと思います。午後試験は午前試験に比べて難しいので、ある程度午前試験の勉強が終わってから取り組むといいでしよう。勉強方法としては午前と同じでひたすら演習です。午後用の対策問題集などが出版されているので、それらを活用してもいいでしよう。

これで合格の下準備は完了です。あとは、頑張って本番で合格点を取りましょう。

## 4 最後に

このやり方でやったからといって必ず合格するわけではありません。合格率も 20% とあまり高くありません。むしろ低い方です。実際私も一回落ちてます。なので、恐れずにどんどん挑戦していきましょう。

## 5 参考文献

情報処理推進機構ホームページ <http://www.ipa.go.jp/index.html>  
応用情報技術者ドットコム <http://www.ap-siken.com/>

# ミニギターアンプを作ろう

知能機械工学科 1年 桑原圭佑

## 1. 背景

調布祭でのエレコンが終わり一段落ついたところで、今までを振り返ると…何も作ってない。そう、何も作ってないので。これでは一工研部員として失格であり、また来年入ってくるであろう新入生に対して示しがつかない。ということでいい加減物を作ろうと画策しました。

## 2. 目的

自分は趣味でギターをしているが、それに関するもので、また今後役に立ちそうなものはないかと考えた結果、まずは取り回しの良いエレキギターアンプを作ることにした。

## 3. 原理

そもそもエレキギターアンプというのは通常のオーディオアンプとは違い、大きな音で歪んだ音を出すことを目的としている。そのためオーディオアンプとは違い、あえて音を歪ませなければならないので回路もそれに対応している。今回は出来るだけ小さく作る、また電子工作の初心者である自分でも作りやすいオペアンプを用いたものになっている。用いたアペアンプはLM386N-1です。一個50円と安価であり、そこそこ良い音が出るとのことだったのでこいつにしました。

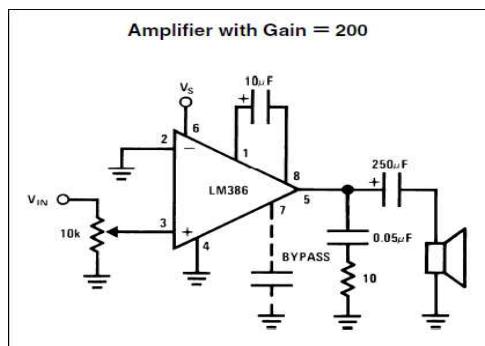


図1. リファレンスデータシート

#### 4. 製作

回路に関してはネットのものを参考にして以下のものを作りました。

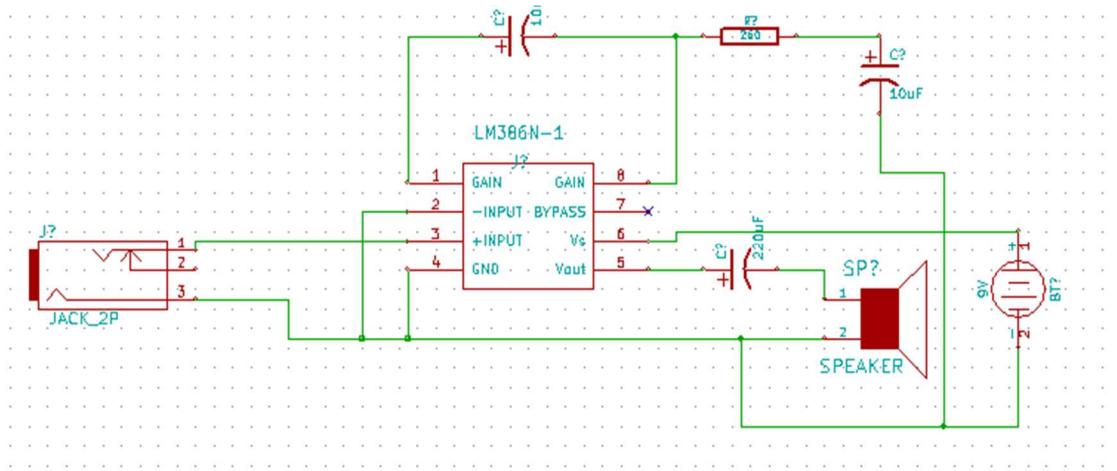


図2. 回路図

非常にシンプルですね。あとはこれをユニバーサル基板に実装して終わりです。

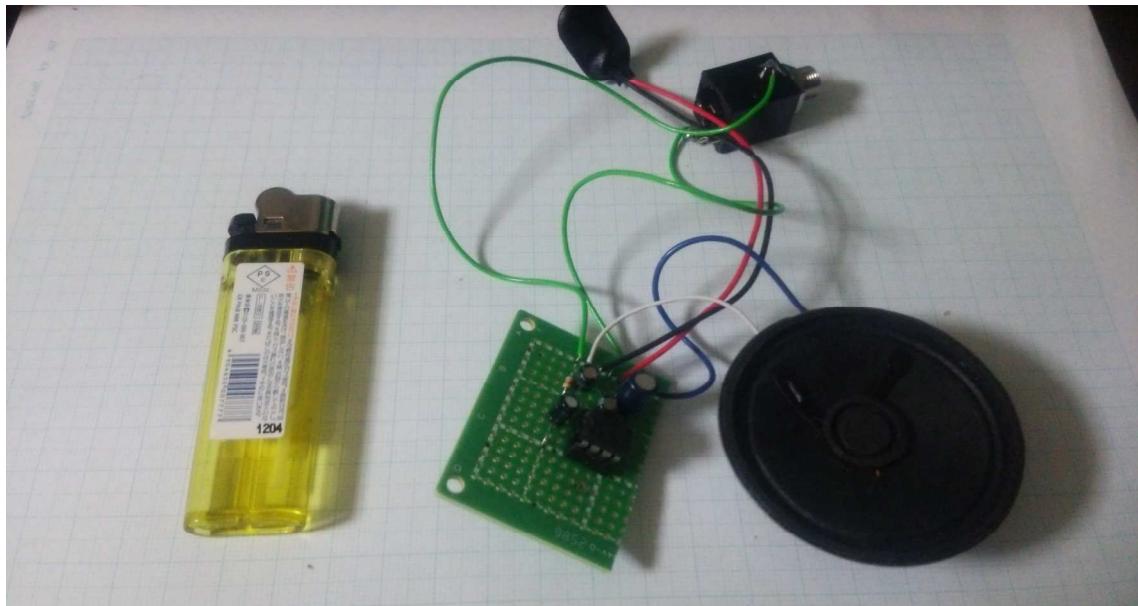


図3. 完成品

## 5. 感想

肝心な音のほうですが…そこそこといった感じです。今回は抵抗値を430Ωの物を使用したのですが、適度に歪んだオーバードライブサウンドとなった。抵抗値を低くするとディストーションサウンド、大きくするとクリーンサウンドとなります、自分の好みで変えるのも良いでしょう（もちろん抵抗値が大きくなると音量も小さくなるので注意）。問題点としては小さくすることに力を入れたため機能面で貧弱であることが挙げられる（例としては電源スイッチ無し、歪の量が変えられない、音量も調節できない,etc.）本当はタバコの箱に入れるようにしたかったのですが時間が無かったのでむき出しというのもいただけないですね。今後は追加機能を実装したり、基板を彫ってより小さいものが出来るようしたい。冬休みに頑張ろう。

# アンプ IC を用いたお手軽オーディオアンプ作成

聖徳たいし

## 1 はじめに

今回はアンプ IC を使って小型で持ち運びの簡単なオーディオアンプを作ってみました。音質に対して部品点数も少なく安くお手軽でそこそこの音です。

## 2 LM386

アンプ IC には AB 級や D 級、200mW から 150W の大電力に対応するものまで様々なものが有ります。今回はその中でもバージョンによって 325mW から 1000mW に対応する LM386(図 1)を用いました。電源電圧は 4~12V です。それでも通常の室内でスピーカーを駆動すれば十分な音量が得られます。



図 1: LM386

アンプ IC はオペアンプなどと違い、オーディオアンプそのものが IC に收められているため、最小限の外付け部品でそこそこの良音質なオーディオアンプが作れるところが素晴らしいです。

## 3 回路設計

データシート推奨回路は図 2 です。外付け部品はボリュームとコンデンサだけでもスピーカーを鳴らせます。今回は工研部室という劣悪な環境での使用を考えて、推奨回路に発振防止の為のコンデンサを付け加えました。

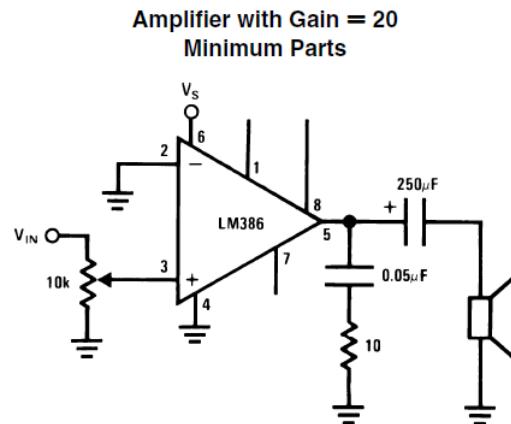


図 2: 推奨回路図

また、部室に転がっているようなスピーカーでもいい音に聞こえるように、低域の特性を改善する目的でカップリングコンデンサの値を  $220\mu\text{F}$  から  $470\mu\text{F}$  に変更しました。

そうして出来上がった回路が図 3 です。ケースを作ることを想定して、電源 LED、入出力をコネクタにしています。電源は AC アダプタ入力で、回路内部でレギュレータで 9V に降圧しています。

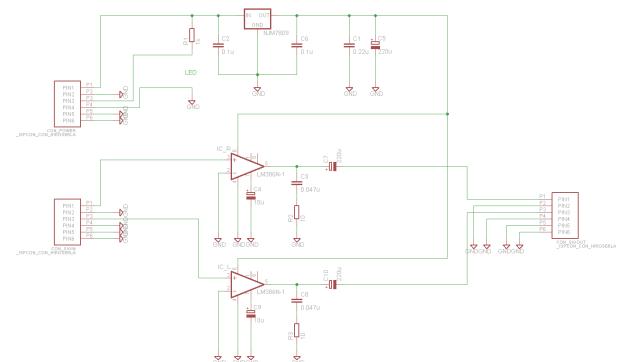


図 3: 設計した回路図

## 4 パターン設計

オーディオアンプなのでなるべく対称になるように適当に配線しました。図 4 にパターン図を示します。とりあえず高級部品使えばいいだろ的な感じでカップリングコンデンサと発振防止コンデンサは MUSE、その他はマイラーコンデンサで設計してみました。(高音質になる根拠はないです、ただの気休めです)

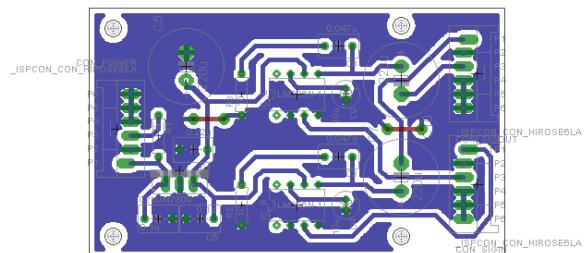


図 4: パターン図

## 5 実装

あとは基板を彫って実装です。図 5 のようになりました。手のひらサイズに收まりました。

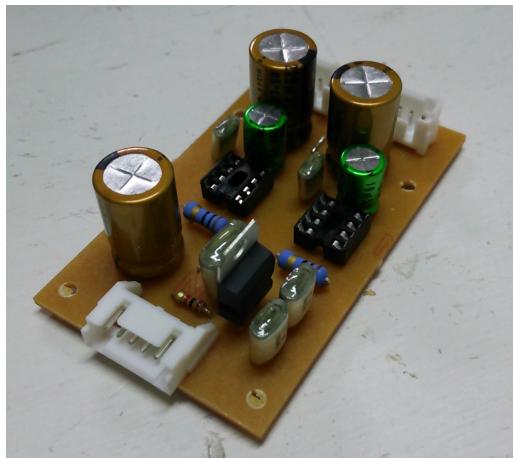


図 5: 実装

次にケース加工をしました。TAKACHI のプラケースに、電源コネクタ、オーディオ端子、LED、ボリュームのための穴をけがき、ボール盤等で加工しました。

内部配線を処理します。基板がケースに収まったところが図 6 です。



図 6: ケースに基板を格納

蓋を閉じて、保護シールを除去すれば完成です。電源投入したところが図 7 です。



図 7: 完成

## 6 おわりに

超簡単でした。お金も 3000 円弱でお手軽です。音は味付けの少ないピュアな感じです。部室で使っていますが低音も結構出ます。ただ、1W 級なのであまり音を大きくすると音が割れてしまいます。実用範囲では問題ないです。時間があれば周波数特性なども測定したかったですが、今回はここまで。

# 半自動卓-ダイス付き点数計算表示器-

回路担当 知能機械工学科 4 年 石関 峻一  
ソフト担当 知能機械工学科 4 年 筒井 悠平

## 0.1 背景

私事なんんですけど、最近麻雀を覚えました。まぁ、とある作品の影響(図 1)です。

この作品は数年前から好きで読んでいたのですが、麻雀のルールさっぱりわからなかったんです。それでも楽しめてたのですが、なんか機会もあいまって、先日ついに始めました。



図 1: とある作品の一例

そこで最近麻雀を打つ機会が毎週あるのですけど、その時に点数のやりとりって大変じゃありませんか。どうにか和了ったはいいけど、どの程度の点数かっていう点が難しい。まず、慣れてないと点数の計算に戸惑います。慣れていても本場や親と子を間違えたりします。

また点数棒のやりとりが面倒です。1000 点棒がなくなって両替してもらったり、百点棒を大量に持ってプレーすることになります。また、相手の点数が気になつた時にわざわざプレー中に聞くのは億劫です。

なら自分で麻雀専用の点数計算機をつくってはどうかなと思いました。そうすることで、点数に関わる煩わしさがかなり減ると期待しました。

## 1. 作品概要

というわけで、図 2 のように完成しました。



図 2: 全体図

表 1: 部品機能対応表

部品	機能
ロータリーエンコーダ	符・翻の入力
赤色 LED 付きスイッチ	ダイス・ツモなど
緑色 LED 付きスイッチ	決定
7 セグ	点数表示
16 セグ	場・局・本場・供託
サイコロ	ダイス

と言った部品で構成されています。使用した金額などを図 2 に示します。

表 2: 明細

製作期間	9/20 ~ 11/4
製作時間	不明
飛ばしたジャンパ	計 157 本
費用	7,000 円
失った単位	0

## 2. 制作の様子

さて、どのように制作をしていくか考えたところ、点棒を出す必要をなくすんだったら、ダイスも出したくないよね。ってことでダイスを回す機構を考えることにしました。



図 3: 構想図

この図 3 のような感じで、PC ファンでダイスロールを行えるのでは無いかと考え、試したところ、うまくいきました。

ということで回路図を考えます。制御には SH7125 を使用します。

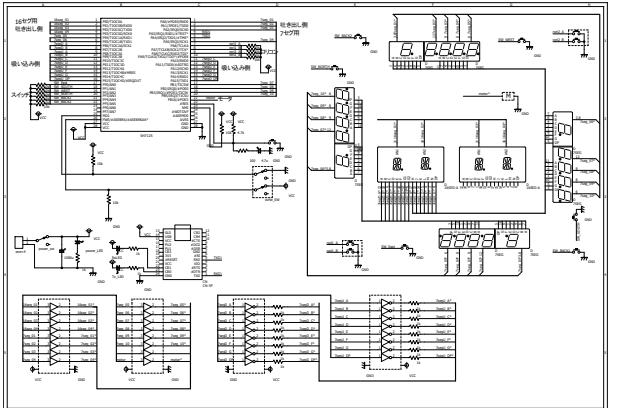


図 4: 回路図

とりあえず、ダイナミック点灯を行えるようにちょちょいと（ピン足りないので RXD は兼用にしちゃえ）回路図を引きます。

部品をはんだづけします。

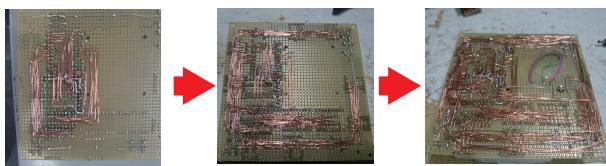


図 5: ハンダ漬け

更にハンダ付けします。

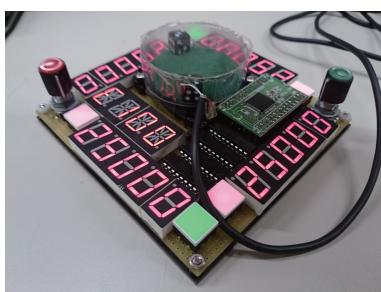


図 6: 回路完成

ここで、スペシャルサンクス すこーんが麻雀アルゴリズムからプログラムを作成してくれました。まじ、感謝。

次に、ケースをどこのご家庭にもあるレーザー加工機で制作して、完成です。

大事なステップ忘れてました。完成したらデバッブ(麻雀)をしました。

この麻雀はデバッブだから――――――(図 7)。

最後に仕様書(図 8)を仕上げます。

### 3. 感想

今回最初は夏休みのラスト 3 日でなにか工作したいなってことから始まったのですが、すでに 11 月、もう秋というより冬の足音が聞こえます。もう少し早く完成する予定だったのですが。

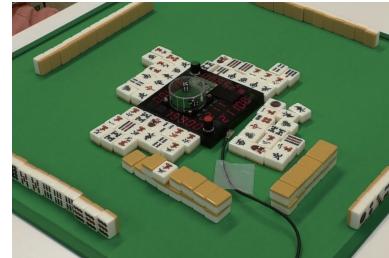


図 7: デバッブの様子

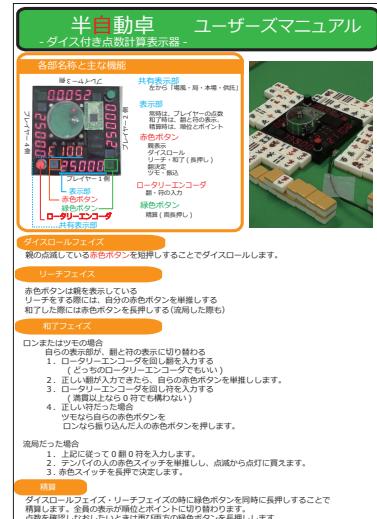


図 8: 仕様書(仮)

気になる使い心地ですが、有線の取り回し以外はとても便利です。わざわざ対局する前に点棒を配らなくてもいいし、点数を数えて渡す必要もありません。

また、誰が何点持っているのか意識して打つことが何より便利です。となりから誰が勝ってるの？って聞かれることも少なくなりますしね。

おかげに、サイコロを回すときの音が気持ち良いです。ただの PC のファンでこんなにうまく回るとは思ってませんでした。

これから麻雀をするときには必ず持って行きたいです。ではでは、次の工作で！

国立大学法人 電気通信大学  
工学研究部 部報 第49号

発行所 国立大学法人 電気通信大学工学研究部  
〒182-8585 東京都調布市調布ヶ丘 1-5-1 サークル棟2階  
E-mail koken@koken.club.uec.jp  
URL <http://www.koken.club.uec.ac.jp/>

発行 横田嶺  
編集人 吉村英幸  
発行日 2014年12月30日  
執筆 工学研究部 部員

