

UEC

工
業
部
報
四
号

工学研究部 部報 第44号

もくじ

作者	題名	ページ
皆川 太志	時限爆弾を作つてみた	2
山岸 大騎	アーケードコントローラー制作	4
ポン☆デ☆リング☆村上	エレコン制作構想	5
大川 修平	最初の工研部報	7
池田 佳那	女装について	8
石岡 俊一	フルカラーコンサートライトの制作	10
志子田 敏	初めて (?) の電子工作	13
住澤 庄秀	今回こそはアンプを作るぞ！	14
Mark.IU	疑似科学を科学する～イオン水についての考察と効果～	17
Hagi	ぱっこんなっ！の紹介とか	20

時限爆弾を作つてみた

先進理工学科 3 年 皆川太志

1 はじめに

さて、本記事を手にとって早速 110 番へ連絡しようしてくれたそのアナタ。あなたの正義感は見上げたものですが、やめてください！これは決して、人に危害を加えるものではありません！本作品は「時限爆弾解除の感覚を楽しむ玩具」です。モチーフは映画とかで電線を切断することによって時限爆弾を解除する感じです。図 1 が完成した時限爆弾です。

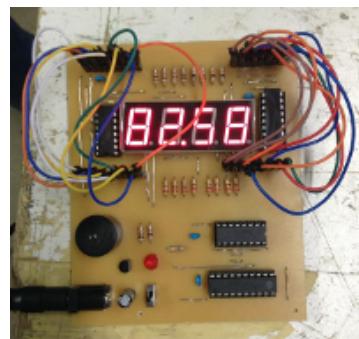


図 1 時限爆弾

2 遊び方

電源となる AC アダプタを差し込み、電源スライドスイッチを ON すると時限爆弾が起動します。起動後はピンソケットからジャンパー線を引き抜くことで、爆弾を解除していきます。表示されている残り時間が 0 になる前に、当たりのジャンパー線を引きぬいてタイマーを止める事が出来れば爆弾解除成功となります。もちろん、一発アウトや残り時間が 10 秒減るといったハズレ線も多数存在しているのでご用心を。もし解除に失敗した場合は、白い閃光と轟音とともに・・・ということではなく、けたたましくブザーが鳴り解除失敗を知らせます。

3 機構説明

図 2 が時限爆弾の構成の簡略図です。

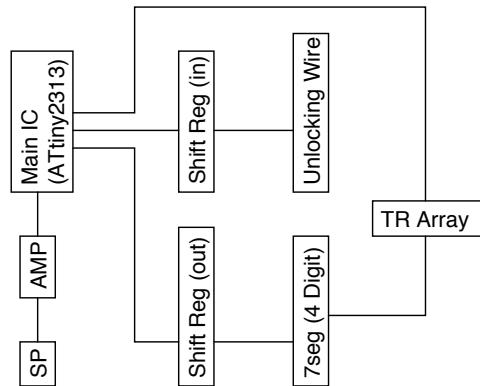


図 2 簡略図

メイン IC は ATtiny2313 です。出力用シフトレジスタは 4 枚 7 セグの文字の表示のデータを ATtiny2313 から受けて、7 セグを光らせます。入力用シフトレジスタは解除線の切断判定を ATtiny2313 に送ります。トランジスタアレイは四枚 7 セグのどの桁を光らせるのかを選択し、ダイナミック点灯させます。スピーカーは爆弾爆発時にブザー音を鳴らします。

4 感想（苦労したこととかいろいろ）

いまいの一言で製作が始まった時限爆弾ですが、いろいろあって完成が遅くなってしまいました。いやーほんとすいません。

苦労したことは、まず ATtiny2313 には ATmega328 のようにハードウェア SPI(Serial Peripheral Interface) が内蔵されていないので、他の機能で代替する必要がありました。じやあ ATmega328 を使えよって思うかもしれません、茨の道を通るのが工研部員です。今回は ATtiny2313 にもハードウェア内蔵されている USI(Universal Serial Interface) を 3 線同期駆動することで、SPI 同じ動作をさせてみました。

結果として、ソフトウェアで実現する方が開発が早かったと思います。でもこれで ATtiny2313 でもハードウェアを使って高速に SPI 通信できるようになったよ！

アーケードコントローラー制作

始めに

最近（といっても結構経ちますが）、某先輩の影響でSTGを始めることがなったので、なにかにかこつけて物を作りたがる私はアケコンを作ろうと思いました。そこで今回は、

- ・現行で最も普及しているVIEWLIX[TAITO]のボタン配列と、未だに稼働数の多いBLAST CITY[SEGA]他、旧筐体のボタン配列どちらにも対応出来る筐体

であることを念頭に作り始めました。

仕様

- ・ボタンレイアウトの置換可能 (VIEWLIX/ プラスト / 他)
- ・レイシリーズを遊べるために、対応ハードは少なくともPS/SS
- ・新作のためにXbox360対応

筐体設計

ボタン配置する部分を別パーツにして制作します。他の形は、大体VIEWLIXを参考にしています。

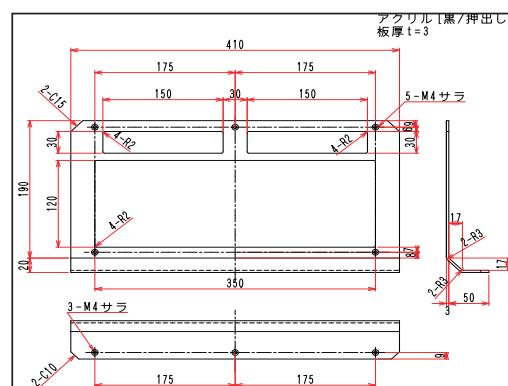
ボタンメンテナンス時に、ボタンへアクセスするために開けるネジを少なくするために、トルクヒンジを使って開閉式にしています。

また触った時の冷たい感じを無くすため（いつものレーザー加工機を使うために）、素材はアクリル板で作ります。

便利です。レーザー加工機。



外観



天板図面



パネル展開図

あとがき

あれっ？回路は？と思った方ごめんなさい。まだできてません。

回路の方は、過去部報に書いてあるPS互換コントローラー制作などを見ていただけると、だいたいこんな感じのものが出来上がると言うのが解っていただけると思います。

とりあえず、決まっていることは、PS対応->自作 / SS対応->自作 / Xbox360対応->手持ちの360基板使用することです。PSはSPI通信with Acknowledge。SSはページ切り替え方式と、自作が簡単なのですが、Xbox360はただのHIDジョイパッドというわけではないので、実装が難しいです。（そもそも自分でUSBの実装出来ませんが）なので、基板を乗っけようと思います。

まあ、プレイに関係する筐体までは出来上がっているので良し。

エレコン制作構想

1S ポン☆デ☆リング☆村上

平成 25 年 8 月 19 日

概要

エレクトロニクスコンテスト参加に向けて創作物の概要を記す。なお本記事は全て構想段階であり脳内設計であるため制作の変更は大いに起こり得る。

1 目的

自分がこれまで培った工作技術を復習、統合し、今後身につけるべきスキルを確認するためにエレクトロニクスコンテストに参加する。植物の栽培には定期的に水やりをすることが望ましいが、忙しい生活故、忘れてしまうことがある。そこで、自動で散水し、またデータ取得をすることで効率的な栽培を行うロボットを作成する。

2 経緯

自分の過去の主な制作物に以下のものがある。エレクトロニクスコンテストに向けて受験勉強によるブランクを払拭すべく、以下を踏まえた新作を制作する。

2.1 CANSAT

高校 2 年の夏、能代宇宙イベント・カンサット甲子園参加にあたり、電装班の一員として制作に参加した。自分はキャリアに搭載する小型データロガーの制作に携わったが、過剰な小型化に自身のスキルが追いつかず、その結果未完成となり非常に悔しい結果となった。カンサット甲子園で自分の成果物と言えるものは存在しないが、初めてプロジェクトに参加して制作を行ったこのイベントでの経験は有意義なものであり、自分の創作活動の原点となっている。

2.2 花の水やりき

初めて自身で設計を行った作品である。高校 2 年次の文化祭で展示した。これはカメラ、エアポンプ、LED ライトを AVR マイコンで制御し、周期的に撮影、水やり、ライト照射を行う機械である。マイコンプログラミングや電子工作の初歩を習得、および自分の生活をより便利にするために制作を行った。

2.3 ハイブリッドロケット

高校 2 年の冬から翌年の 6 月頃にかけて、ロケットガール＆ボーイ養成講座に参加し、電装班としてハイブリッドロケット内部電装及びペイロードの制作に携わった。ロケット内部電装の作成は当初サポートして下さった大学院の方の設計の下、基盤設計、切削、工作、

PIC プログラミング等を行なっていたが、打上げ 2 週間程前に急遽設計を我々が再設計することになり、そこでは SunSPOT を用いてデータロガーの制作を行った。また、ロケットの搭載物としてローバーの制作を行った。これはロケットから放出後、減速機構を用いて安全に着地、キャリアから離脱し地上を走行し、その後本体に内蔵された FM ピーコンを八木・宇田アンテナで受信し、回収を行うのが一連のミッションであった。自分はそこで、ローバー制作の主担当として各担当との調整やローバー内部電装の制作を行った。

3 ミッションシークエンス

図 1 に本製作物における動作の流れ図を示す。

まず、拠点にて気圧、湿度センサ等を用いて天候データを取得し、晴天／雨天で出動判定を行う。出動した場合、目的地まで GPS、方位センサ等を用いて誘導する。途中、障害物がある場合、距離センサを用いて検

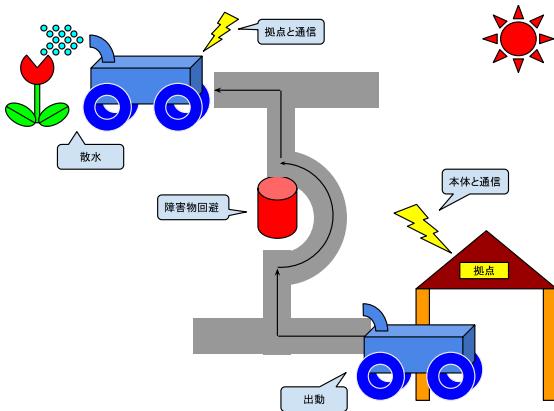


図1: ミッションシークエンス

出し、回避する。目的地に到着後、写真撮影、温度、湿度、気圧等のデータ取得後、散水し、同様に帰還する。拠点にて収集したデータを蓄積、解析し、効率的な栽培に役立てる。また、インターネットを用いてスマートフォンから遠隔操作、監視を行う。

4 サクセスレベル

前項のミッションをすべて達成するのは難しい。そこで、個別にサクセスレベルを設定する。サクセスレベルが低いものから順に制作を行う。

4.1 サクセスレベル1

- 走行
- 散水
- 物理データ取得

4.2 サクセスレベル2

- 自動走行
- 障害物回避
- 出勤判定

4.3 サクセスレベル3

- 拠点製作
- 無線通信
- データ解析

4.4 サクセスレベル4

- オンライン化
- スマートフォンアプリ製作

5 製作

現在、製作環境の構築段階であり、実際に製作が始まるのは9月頃を予定している。完成品はエレクトロニクスコンテストに出展する予定であるが、機体製作以外にもプレゼン作成等の必要があるため、そちらにも注力していきたい。

最初の工研部報

1年M科 大川修平

1. 1年生前期の活動

自分は過去に電子工作をしたことが数回しかなく、技術はまだほとんどないので、今のところ工研内で作ったものといえば、新入生講習で教えてもらったマイコンライターぐらいです。ただ、技術を身に着けるために学内の電子工作の講座に行って、まだケースが完成してませんが、オーディオアンプを作成しました。外部電源方式のアンプです。この講座で主にパソコンを使った回路設計の方法、オシロスコープなどの計測機器の使い方、旋盤の注意点などを学習しました。

あと Raspberry Pi を購入しました。なにかといろいろと便利に使えそうなので。今のところ小さなモニタと 12V 出力対応のモバイルバッテリーや、小型キーボードなどを合わせて購入し楽に持ち運べるようにしたいと思っています。

2. エレコン

最初は個人でやろうと思ってましたが、他のサークルや学業との兼ね合いを考えた結果、他のチームに混ぜてもらうことにしました。エレコンに参加するにあたって、とにかく聞いたアイデアをどんどん貯めていたので、そのチームで作るものも自分のアイデアの中の一つに決定しました。そのせいというか、その流れでチーフリーダーになってしまったのは

3. 今後の活動の展望

今後の活動は、エレコンの作業や他のサークル

の活動などを行っていくのはもちろんですが、ほかにも作りたいものがいっぱいあるし、エレコンのアイデアでボツになったものにもいいものがあるので、それらのものも細々と作っていきたいと思います。まあその前に今作ってるオーディオアンプを完成させることが先決ですが。

以下、湧き出たアイデアを箇条書きで列挙。

- ・超小型マウス
- ・トレース台
- ・モノクマ人形（ギミックを内蔵）
- ・一人用食洗器
- ・自動ベッドメイク装置
- ・本棚を整理する装置
- ・人形枕アラーム（膝枕）
- ・ロケット花火ランチャー
- ・個人用階段降機
- ・三人称視点カメラ
- ・多機能作業台（特に消しカスを何とかする）
- ・虫潰し（物理）
- ・エコー装置
- ・背後カメラ
- ・蚊の音のするアラーム
- ・拍手をすると時刻が音で分かる時計
- ・キャタピラ式移動装置
- ・集光機

自分で言うのもなんだが、はつきり言って1つでもできたらすごいと思う。

女装について

1年知能機械工学科 池田佳那

1 はじめに

今回、44号の工研部報を書くにあたり、女装を題材にした。

電通大の調布祭には、男子が女装して出場するコンテストが存在する。他の大学では中々見ることのできない行事である。このことからもわかる通り、電通大は比較的女装に対して寛容な環境が整っている。せっかく与えられたこの環境を生かさない訳にはいかないだろう。そこで、持論を交えて女装について記述しようと思う。

2 女装というジャンルについて

そもそも何故女装というジャンルが生まれたのか。私なりに考察してみることにする。

まず、女装の定義について Wikipedia から引用する。(Wikipedia ぐらいにしか定義が載っていなかった。辞書で調べても一文程度しか説明がなかった。激おこ。)

女装（じょそう）とは、それぞれの文化によって「女性用」と規定されている衣服・装飾品を男性が身につけ、これによって外見の衣装上は女性の姿になることを云う。男性の異性装である。

男性が女性ものの服を着る。それだけの話である。

人は非日常に憧れる傾向がある。普段できないことをしてみたいという欲望は多くの人が持っているだろう。女装もそのうちの一つと言える。ジェットコースターに乗りたい、バンジージャンプをしたいといったことと同じことだ。女装というジャンルが確立されたことにより、多くの人が女装という行為を認識することができ、堂々と女装をすることができるようになったのである。

3 女装のやり方

女装するためには、様々な準備をする必要がある。どうせ女装をするのならば、本気で楽しんだ方が良いだろう。そのため、いくつか準備すべきことを紹介しよう。

1. 服装について

一番重要なものはやはり服装である。女装だから当たり前だが。服装の準備はネットや専門店で購入・レンタルする自分で作る（もしくは人に頼む）この二つが主であろう。のデメリットは、値段が高いものが多いことである。特に、アニメなどのキャラクターの服だとセーラー服といった一般的な女装服よりも高くなる傾向がある（当社比）。一般的な女装服ならば、（私が今知る限りでは）ドン・キ○ーテあたりが安い。は、時間はかかるが購入するより

費用がかからずに済む可能性が高い。しかもオリジナリティを出せるのでこだわりを追求することができる。夢がひろがりんぐ。だが、裁縫という女子力が必要となってくる。

2. 化粧品

軽く女装したい人にはいらない項目かもしれない。そもそも私はそんなに化粧品について詳しくないのであまり書けない。周りの人で女装経験のある人を訪ね回って聞くのが一番早いと思う。

3. ヒゲや毛の処理

見た目がちゃんと女子になるように、このような手入れも重要である。腕毛やすね毛の処理をするのは恥ずかしいという人は、ロングスカートや長袖を着るなどしてうまく隠すのが良いだろう。

4. 写真の撮り方

女装したのだから、写真として残しておきたいという人もいるだろう。写真の撮り方の工夫次第で可愛さを増し増しすることが可能だ。いろんな角度・距離から撮るのを試して、一番格好が映える撮り方を探ろう。

5. キャラになりきる

キャラになりきるということも大事だ。アニメのキャラならば、そのキャラの登場する回をひたすら見てキャラの特徴（仕草や声など）を研究し、実際に習得できれば最高だ。この時、恥じらいを持ってはいけない。自分に自信を持ち、堂々としている方がよりキャラに近づくことができるはずだ。

4 最後に

女装する目的は人それぞれだろう。例えば、女子の気持ちを理解するため、男の娘になりたい、女子を装って男子を釣りたい、女装姿の自分を見て自己満足するなど、いくつか考えられる。目的はどうであれ、私は女装を推奨している。女装することは、自分の殻を破り、己を成長させるきっかけとなりうるかもしれない。自分に自信をつけることもでき得るだろう。女装は、多くの可能性を秘めているのである。

要するに、皆さん女装しましょう。

参考文献

- [1] 女装 - Wikipedia. <http://ja.wikipedia.org/wiki/女装>

フルカラーコンサートライトの製作

Creation of full-color concert light

石関 峻一
Shunichi ISHIZEKI

電気通信大学 工学研究部
(〒182-0021 東京都調布市調布ヶ丘 1-5-1, twitter id: ueckoken)

Abstract: It is one that is often used in live voice actor and full-color pen light. However, there is no agility and coordination very current. So, I, to produce a full-color study penlight with a corresponding property and coordination.

Key Words: LED, Full-color, Concert, Voice actor, cyalume

1. はじめに

コンサートライトとは、コンサートでよく用いられる長さ 25cm の LED により発光する棒状のアイテムである。主に声優系のライブ用いられ、公式グッズとしても、見ることが多々ある。2000 年代前半では、化学反応で発光するサイリュームやポケットネオンが主流であった。しかし現在は LED を用いたコンサートライトに主流が移った。ライブの現場（会場のこと）では、様々なコンサートライトを見ることが可能である。コンサートライトは単色ばかりではなく、フルカラー LED を用いたマルチカラーのコンサートライトもまた登場してきた。

そこで、なぜ自作するのかというと、市販されているフルカラーコンサートライトは、ボタンがひとつしかないため操作性にかけ、優れている構造とは呼べません。また、複数本を同時に持つなどをした時、個別に色を帰る必要があり、時間がかかり、コンサートを楽しむ時間の一部を、コンサートライトの色をかえる時間に奪われてしまう。よって、私は、複数本を目的の色に一操作で達することのできるフルカラーコンサートライトを開発を行う。

2. LED の明るさの制御方法

具体的な話をする前に LED の明るさを制御する方法について、語りたいと思います。LED の発光はほぼ、電流に比例します。

2.1 LED の基礎（制限抵抗）

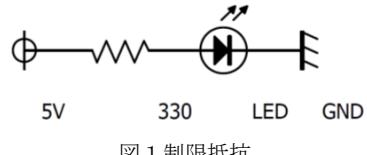


図 1 制限抵抗

みなさんは LED を光らせるときには必ずこの【制限抵抗】を入れていると思います。なぜなら LED は、ダイオードの一種であるため、電圧と電流が非線形性であり、ある程度の端子間電圧をかけると一気に大電流が流れ、素子を自ら破壊してしまうのだ。つまり、手間のかかるかわいい子なのだ。

一般的に LED の制限抵抗を決めるときの式は、

$$\text{制限抵抗 } R = \frac{(\text{電源電圧} - \text{LED の順方向電圧})}{\text{流したい電流値}} \quad (1)$$

となる式を、使っているはずである。

この式の表している意味は下図のようになる。

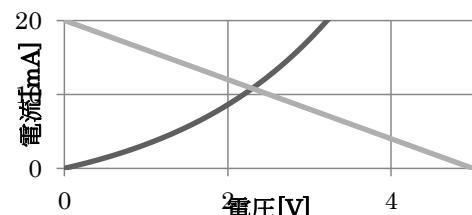


図 2 制限抵抗による LED の制御

0V から出ているのが、LED の I-V 特性、5V から出ているのが制限抵抗によって LED の逆起電力がその電圧だった時に流れる電流である。

つまり交点の電圧電流が LED にかかるのである。こんな感じで LED に流れる電流は決定され、LED の明るさが決まる。

この抵抗を入れるだけの方法の長所は、とても簡単な構成なことである。

欠点としては、LED は自らの温度によって I-V 特性が変化してしまうために、絶えず同じ電流が流れわけではなく明るさも変化してしまうこと。電源電圧が変化すると電流も変化すること。電力を抵抗でも消費するため、効率が悪いことがあげられる。

2.2 定電流回路

電源電圧の変化によって LED の明るさが変化したり、LED の定格電流を超えていたりするのは、問題である。それを解決する方法が定電流制御である。これはどの電源電圧であっても、電流値が一定になるようにする制御方法である。

2.2.1 CRD

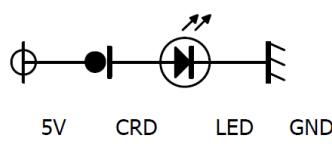


図 3 CRD

この CRD は定電流ダイオードというものである。ある一定の電圧がかかっている時にその CRD に流れる電流は一定となる。

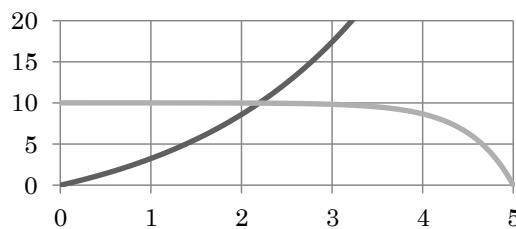


図 2 CRD

これにより、電源電圧が変化しても明るさが変化しない回路になった。

しかし、CRD には 3V~の端子間電圧をかけないと、一定の電流を流さないなどがある。さらに~30mA までの商品がほとんどである。

2.3 トランジスタ

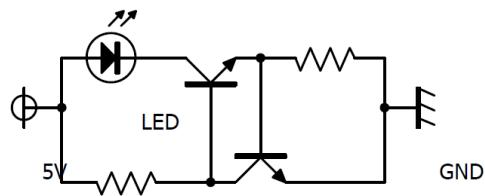


図 5 トランジスタを用いた定電流回路

この回路では、CRD では出来なかったパワーLED のドライブが可能になる。

2.4 PWM 制御

さて、これまで紹介してきた LED の光らせ方では、LED の明るさを変化させるときには、抵抗を取り替えるか、可変抵抗などを手で操作するしかない。

これでは LED をマイコンなどから明るさを変化させることは出来ない。そこで、PWM 制御という方法で行う。

PWM 制御とは、素早く LED を ON, OFF させ、ON の時間と OFF の時間の割合を変化させることで、見かけ上の LED の明るさを変化させる方法である。

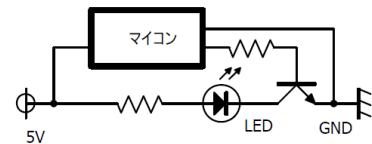


図 6 PWM 制御

この方法の長所は、マイコンと相性がよく、簡単に輝度制御が行えることです。また、抵抗で消費する電力を小さく出来ます。なぜかは考えてね。しかし、この方法でも抵抗などは使用しており、すべての電力を LED にまわせてはいません。

2.5 定電流制御

この方法では、抵抗器からの発熱は少ないので、とても高効率に電源からの電力をほぼ LED で消費する電力にすることが出来、発熱も LED からのみになります。

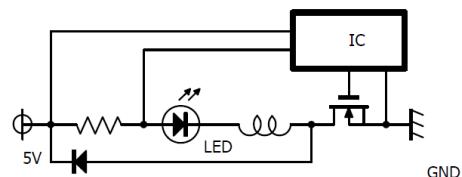


図 7 定電流制御

低抵抗でLEDに流れる電流値を測りFETをほどよくスイッチングすることで、一定の電流がLEDに流れます。

この方法のメリットは、常時LEDが光っているのでちらつかない。とても高効率で光らせることができる。短所は回路が複雑になるなどです。この制御法でこそLEDの持っている特性を十分に活かすことができます。

3. 機能

4本を同時に色切り替え

4段階明るさ切り替え

とても丈夫

とても明るい

明るさが変わらない

4. 回路構成

4.1 LED部

スイッチング制御により、定電流回路を作成した。抵抗だけでは効率や電流制限抵抗からの発熱が多くなってしまうため採用した。

4.2 制御部

AVRマイコンを使用し、全てのペンライトを同一マイコンから制御を行った。

4.3 入力部

スイッチマトリクスを用いて20個のスイッチによる入力がある。一つのボタンで一つの色であるため、一度の操作で、任意のプリセットカラーにすることができる。

さらにボリュームを3つ載せることにより、任意の色を登録し、使用することもできる。

5. 発光部構成

5.1 10WフルカラーLED

一般的なフルカラーペンライトは1.5Wなのでかなり明るい。定電流制御を行っている。定電流制御ICにはPT4115を用いた。

5.2 ヒートシンク

10WのLEDには放熱板が必要となる。

5.3 ファン

自然風冷だとヒートシンクの大きさが大きくなってしまうため、ファンを取り付けることにより強制空冷を行った。

5.4 発光部

アクリルパイプで作成を行った。アクリルパイプ内で光が反射し、棒全体が光る。

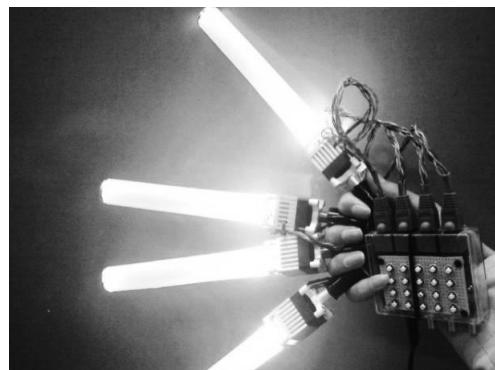


図8 完成品

6. 考察と残された課題

色合いと明るさにおいては、どの既成品よりもいいものが出来た。操作性においては、スイッチが手首の位置に来るため、もう片方の手を用いて操作を行う必要がある。重量が片腕で500gするため、ダンベルを振っているような運動となることである。次に12V以上の電源を必要とするため、ラジコンバッテリーなどを用いる必要があることである。

今後は全てのペンライトを別制御とし、グループの歌手を応援しやすいようにする。さらに出来れば軽量化を行う。またグラデーション機能などもあるといい感じた。

7. 参戦実績

- 7.1 NANA MIZUKI LIVE JOURNEY 2011 埼玉
- 7.2 NANA MIZUKI LIVE CASTLE 2011 -QUEEN'S NIGHT-
- 7.3 NANA MIZUKI LIVE CASTLE 2011 -KING'S NIGHT-
- 7.4 NANA MIZUKI LIVE UNION 2012 群馬
- 7.5 NANA MIZUKI LIVE UNION 2012 千葉
- 7.6 アニメロミックス presents NANA MIZUKI LIVE GRACE 2013 -OPUS II- supported by JOYSOUND Calbee ポテリッヂ 1日目
- 7.7 アニメロミックス presents NANA MIZUKI LIVE GRACE 2013 -OPUS II- supported by JOYSOUND Calbee ポテリッヂ 2日目
- 7.8 魔法少女リリカルなのは THE MOVIE 2nd A's BD&DVD 発売記念イベント リリカル☆パーティー V 1日目
- 7.9 魔法少女リリカルなのは THE MOVIE 2nd A's BD&DVD 発売記念イベント リリカル☆パーティー V 2日目
- 7.10 アニメロミックス presents 田村ゆかり LOVE LIVE 2013*Cute'n Cute'n Heart* supported by JOYSOUND 1日目

初めて（？）の電子工作

学部1年M科 志子田 毅

1 はじめに

皆さんこんにちは。2013年度前期試験にてギリギリで合格した、学部1年M科のシコダです。大学に来て初めて経験したことが多く、まだ色々不慣れな所もございますがよろしくお願いします。因みにこの部報も大学に来て初めて学んだLaTeXで作成しました（汗）。

2 作品紹介()

さて、今回の部報の本題となる作品紹介に移りたいと思います、、が、なんと、入部して以来製作したものがまだ一つしかありません（大汗）。工学研究部では毎年、「新入生講習」を行っているそうで、私はその講習で初めて（？）電子工作を経験しました。そして、何を作ったかと言うと、、マイコン(*microcontroler*)にプログラムを書き込む「ライター(*writer*)」と呼ばれる電子機器です、決して着火する方のモノではありませんw。

（本来はここに画像添付する予定でしたが、JPGファイルをEPSファイルに変換するコマンドが動作しなかったため添付出来ませんでしたorz）

3 実際に電子工作をしてみて…

今回の製作はあくまで「新入生講習」と言うコトなので、準備を0から始めたワケではありません。ですが約4年ぶりにはんだ付けをしたり、実際に電子部品に触れることで、電子工作に対する関心が高まりました。エレコンでは多種センサーを用いた音楽プレイヤーを4人で製作する予定です。

今回こそはアンプを作るぞ！

情報・通信工学科 2 年 住澤 桢秀

1. はじめに

去年は 2 程ほどアンプ作りに挑戦しましたが満足のいく物ができなかつたので、リベンジです。

最初に断っておきますが、私は電子回路に関しては初心者同然なので、あまり高度なことはしていません。

あと、部報を書いている 7 月末現在、完成しません……

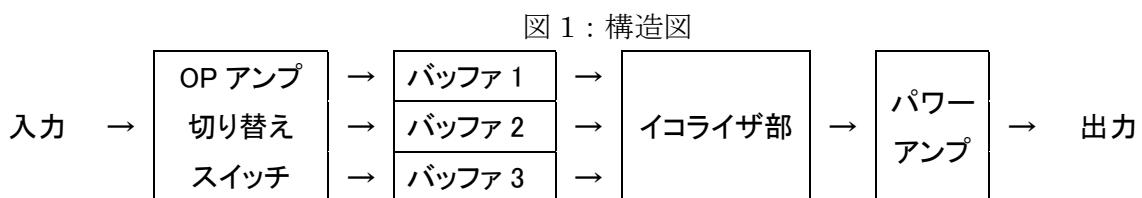
2. 構想

理想というか欲しい機能はこんなかんじです

- ・ オペアンプを何種類か用意しておいて、スイッチ一つで切り替える
 - ・ オペアンプは差し替えがラクにできる
 - ・ 低音に複数のチャンネルがあるイコライザ
 - ・ IC は使用しない
- などなど……

3. 概要

構造はこんなかんじ・・・



違う種類の OP アンプを入れたバッファを 3 つ用意して並列に並べ、スイッチ 1 つでどのバッファを経由するかを決める構造です。

イコライザ部には CR 型トーンコントロール回路を使いました。

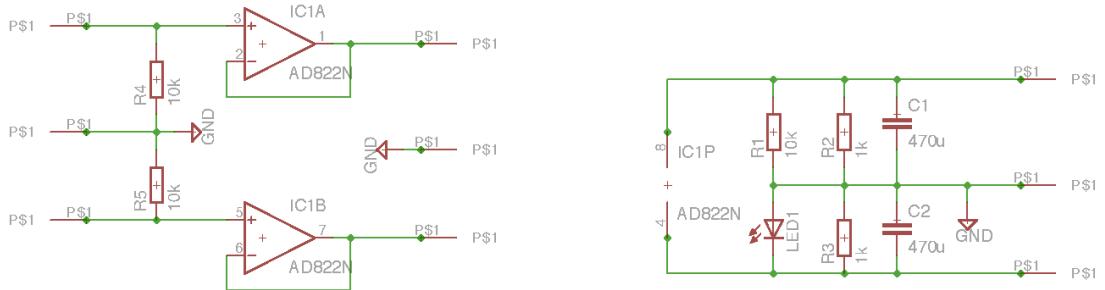
4. 作業

ということで、さっそく作業内容を書きます。回路はインターネットを参考にしました。

4.1 バッファ部

回路図はこんなかんじです。

図 2：バッファ部の回路図



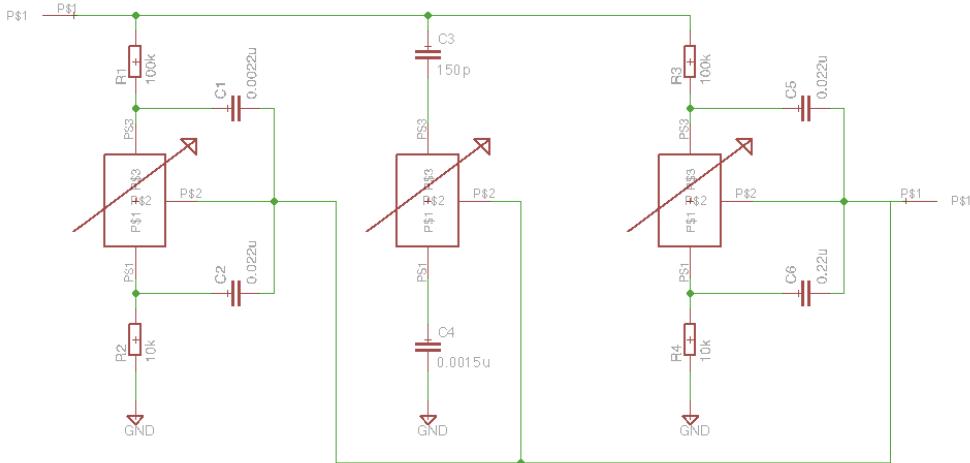
左側が本体で、左から入力して OP アンプを通して右に出力します。単純ですね。これと同じものを 3 つ並列に並べましたが、図が大きくなるので省略。電源部分（右側）の LED は付けないかもしれません。

バッファ部は、回路を設計して基盤を彫り、各部品を付け終わって動作確認まで行いました。

4.2 イコライザ部

回路図はこんなかんじです。ステレオなので、同じものをもう一つ作りました。

図 2：イコライザ部の回路図（片側）

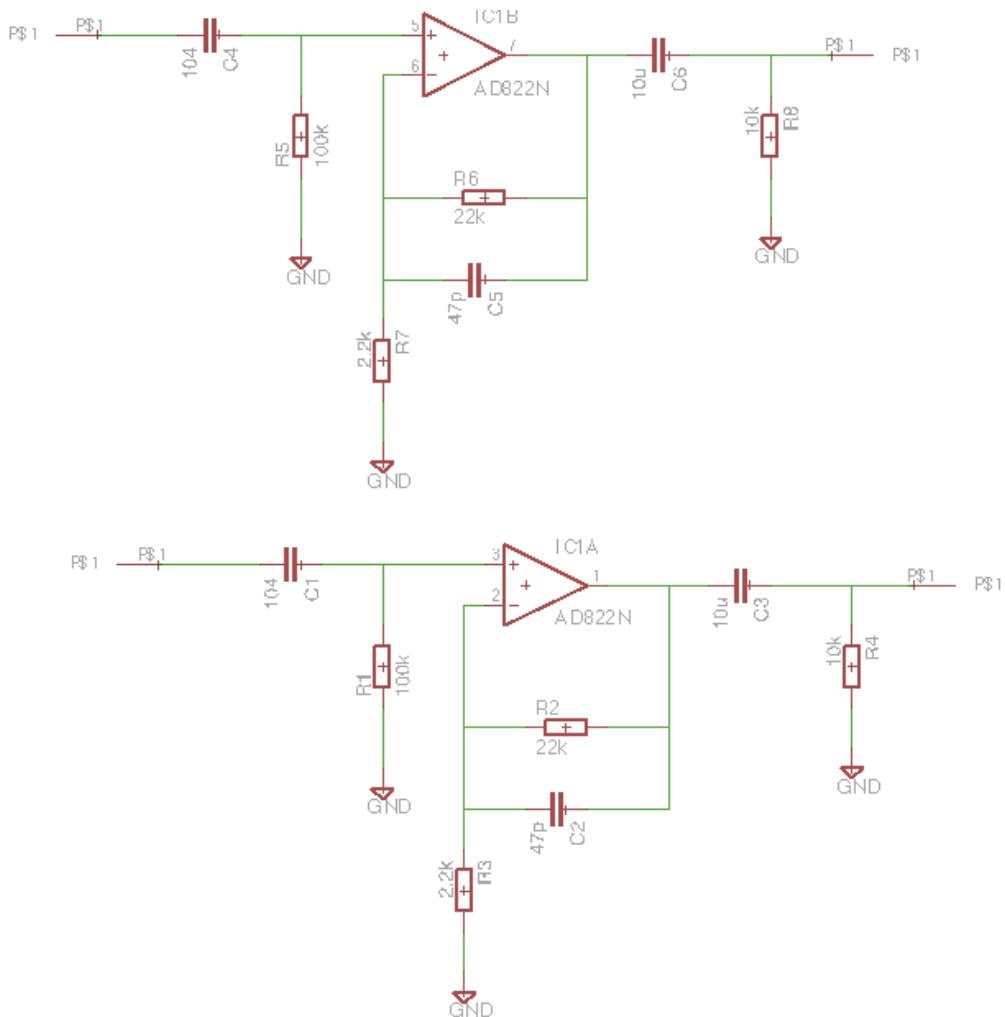


左上が入力で、3 つのチャンネルを通し、右側へ出力します。チャンネルは左から順に低音域（約 700Hz 以下）、高音域（1000Hz 以上）、超低音域（約 70Hz 以下）を調整します。

これはまだ基盤を彫っていません。

4.3 パワー・アンプ部

回路図です。



電源部分は省略します。左上から入力して、OP アンプで増幅させて右から出力するという、すごく単純な増幅回路です。ただ、この回路図で決定というわけではなくて、もうちょっと改良してから基盤を彫る予定です。

5. 感想・そのた

回路設計は楽しいですね！いかにコンパクトに部品を配置するかに悩みますが、それがまたおもしろいです。

がんばって夏休みの間になんとか完成させたいと思います。

6. 参考

マルツパーズ館 パーツまめ知識 「CR 型トーンコントロール回路（音質調整）」

<https://www.marutsu.co.jp/user/mame/155.htm>

疑似科学を科学する ～イオン水についての考察と効果～

Mark.II

世の中には科学的な根拠も何もなしに正しい、と信じられているものがあります。有名なもので言えば血液型占いや、ホメオパシーなどです。これらが所謂科学に見せかけた科学では無い「疑似科学」と呼ばれるものです。

今回は数年前にテレビ等で騒がれた「イオン水」について科学的に考察してみたいと思います。

1 イオンとは何か

「イオン水」について考える前にまずイオンとは何か、について考えなければなりません。化学をやっていない人の一般的な考えは「何か電気を帯びているもの」、だと思います。これは強ち間違っていないのですが、正しいとは言えません。

イオンとは原子や分子が電荷を持っている状態を示し、これらは Na^+ や CH_3COO^- のように書きます。また、イオンには陽イオンと陰イオンが存在し、陽イオンは正電荷を帯びていて陰イオンは負電荷をお帶びています。 Na^+ は陽イオンで CH_3COO^- が陰イオンです。

2 酸とアルカリ（塩基）

イオンについて説明したので次に酸とアルカリ（塩基）について説明しましょう。酸とアルカリ（塩基）には複数の定義、アレニウスの定義やブレンステッド・ローリーの定義等があるので、今回は一般的に用いられているルイスの定義で説明します。

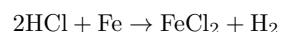
1. 酸とは

ルイスの定義では、酸とは電子を受け取る物質と定義されています。次のような

式（半反応式と呼びます）で表される物質が酸になります。



また、酸は主に金属などと反応し下式のように水素を発生させます。



また、酸がタンパク質と反応するとタンパク質が変性を起こし、様々な反応を示します。この「変性」は不可逆な反応なので、皮膚に触れた場合、触れた部分のタンパク質が火傷を起こしたような症状を示します。ただ、このような反応は塩酸のような強酸でしか起こりません。また、人の胃には rm HCl があり、胃酸と呼ばれています。 rm HCl は強酸なのに胃が痛まないのには理由があります。胃壁には粘膜がありその粘膜によって胃酸によって胃が荒らされるのを防いでいます。

2. アルカリ（塩基）とは

アルカリと塩基はほぼ同義なのでこれ以後は全て塩基に統一します。ルイスの定義では、塩基とは電子を与える物質と定義されています。次のような式で表される物質が塩基になります。



また、金属は基本的に塩基となります。NaOHのような強塩基はタンパク質を溶かす性質を持っています。なので指で強塩基に触れると指紋がとけてしまいます。さらに、目に入った場合などには失明する危険性もあります。ただ、このような反応はNaOHのような強塩基でしか起こりません。人の尿にもアンモニア NH₃ という弱塩基が含まれています。

また、水溶液を電気分解することによって陰極に陽イオン（塩基）、陽極に陰イオン（酸）を集めることができます。

3 イオン水とは

Wikipediaによると、イオン水とは「飲用アルカリ性電解水の通称」だそうです。電解水とはイオン交換膜などを用いて水道水を電気分解した水で、弱塩基性の物質を混入してから電気分解する場合もあるそうです。また、Panasonicのホームページによると「ろ過した浄水を電気分解」と書いてあります。

つまり水道水（塩や塩基性物質を混ぜることもある）を電離させた物のようです。

4 イオン水についての科学的考察

弱塩基性であるイオン水が本当に体にいいのかどうかは専門家ではないのでわかりませんが、私はあまり体に良くないのではないかと愚考します。理由としては、先に書いたように胃には胃酸があります。体にあまり良くない強酸がなぜ体内にあるかと言えば、胃の中にある消化酵素の為なのです。タンパク質等は強酸ではありません。そのため胃や腸の中には消化酵素があります。この消化酵素はまわりの pH (ペーハー) によって分解する速度が変わります。胃の中にある酵素は pH が低いところ

で活発に反応します。なので胃酸によって胃の中の pH を低くしているのです。しかしこに弱塩基性のイオン水が入ってしまうと pH が変わり、酵素の反応が鈍くなる可能性があります。なので体にあまり良くないのではないかと考えます。またアルカリイオン水の説明を読む限りだと、ただ水道水を電気分解してるとわかります。日本の水道を流れている水は軟水であり、軟水はミネラル（塩基性物質）をあまり含んでいません。すると、電気分解しても陰極に集まつてくる陽イオンは非常に少ないと考えられます。なので体に影響を与えられるだけのミネラルが集められるのか非常に疑問です。カルシウム等の塩基性物質を加えてから電気分解を行うのならば確かに何かしらの効果-それが良いか悪いかは置いておいて-は得られるかもしれません。

また、ただ設置するだけでイオン水ができると言う浄水器、これも正しいとは思えません。水道水に塩基性物質を加えた後に電気分解するタイプについてですが、買った当初は確かにアルカリイオン水ができるかもしれません。しかし、使い続ければいつかは必ず水道水に加える塩基性物質がなくなるはずです。しかしそのようなものを追加する必要がある、等とはどこにも書いてないのであります。そして水道水を電気分解するタイプは前述した通りです。

5 結論

私の結論としては「アルカリイオン水」による企業が謳っているような効果は望めない、でした。しかしこれはあくまであまりその分野について詳しくない私個人の考察なのでこれが正しい、というわけではありません。また日本には「信じるものは救われる」と言う言葉があるように信じることで心の安定を得てもいいのではないでしょうか。実際、人が信じれば影響ができるプラシーボ効果というものもあります。

私は信じませんが、信じる人は効果を感じても良いのではないでしょうか。

6 あとがき

科学的な考察をしてみると「いながら肝心なところは「専門家ではないのでわかりません」とぶん投げてしまいすみません。しかしにわか知識で答えを決め付けるよりはいいのだと思います。もう一度書きますが「効果がない」というのはあくまで私個人の考えなので鵜呑みにしないでください。ただ私が伝えたいのは、偉い人が言っているから正しい、ではなく自分で考えて正しいかどうかを決める事は非常に重要なことです。

この記事が様々なことを自分で調べたり考えたりするきっかけになれば非常に嬉しく思います。最後に、このくだらない文章を読んでいただきありがとうございます。

7 参考資料

- wikipedia -アルカリイオン水-
<http://ja.wikipedia.org/wiki/%E3%82%A2%E3%83%AB%E3%82%AB%E3%83%AA%E3%82%A4%E3%82%AA%E3%83%B3%E6%B0%B4>
- Panasonic ホームページ -アルカリイオン整水器-
<http://panasonic.jp/alkaline/aboutalkaline/mechanism/index.html>

ぴっこんなっ!の紹介とか

情報系男子院生 Hagi

URL: <http://www38.atwiki.jp/hagisoft/>

挨拶

こんにちは。情報系男子院生 Hagi と申します。せっかくなので工研部報に何か書こうということで、つい最近バージョンアップを行ったフリーソフト「ぴっこんなっ!」について、機能の紹介やプログラムについての話を書くことにしました。一部お見苦しい点があることをお許し下さい。

「ぴっこんなっ!」は、作成された経緯上使い方にかなり癖があるソフトです。実際に使ってみたい方は、<http://www38.atwiki.jp/hagisoft/pages/24.html> からダウンロード出来ます。よろしかったら使ってみてください。

1 背景

最近は様々な Web サービスで画像を投稿する機会も増えたり、また、様々なユーザーに画像を送る機会も増えたりしてきている。その際に、画像を縮小・他形式への変換を行う必要が出てくる場合がある。PC 上でこの作業を行う場合には、一般的に縮小ソフトやレタッチソフトで画像を縮小してから、その画像ファイルをアップロードする場合が多い。しかし、縮小するたびに保存するファイル名を指定しなければいけない場合や、アップロード時にアップロードするファイル名を指定するのに手間がかかる問題がある。また、アップロードしたファイルを手元に残す必要がない場合はそのファイルを削除するのに手間がかかったりする問題がある。

そこで、画像縮小・変換時にファイル保存やアップロードのファイル名指定の手間を省きつつ、また、使用後の画像ファイルの削除の手間を省く事もできる方法として、フリーソフト「ぴっこんなっ!」を作成し、初版を 2011 年に公開した。

2 提案手法

本ソフトを設計するにあたり、背景で述べた問題を解決すべく、以下の機能をもたらせた。

2.1 ドラッグ&ドロップだけで実行できる画像の拡大・縮小機能

図 1 のウインドウで、画像のサイズ・形式を指定する。

ウインドウに画像ファイルをドラッグすると、すぐに変換が始まる。変換された画像は、予め決まった場所へ自動的に保存される。



図 1: 「ぴっこんなつ! Ver. 3.1」 画像を 1 枚変換した時の画面

2.2 変換したファイルのファイル名(フルパス)をクリップボードにコピー

図 1「最後に変換した画像のパス文字列」が有効な場合、フルパスがクリップボードにコピーされる。(複数を同時に変換した場合は最後に変換したファイルのフルパスがコピーされる。)

変換後に図 2 のようにブラウザ等でファイル選択画面のファイル名に変換後のパスを貼り付けると、変換したファイルをアップロード対象として直接指定することができる。

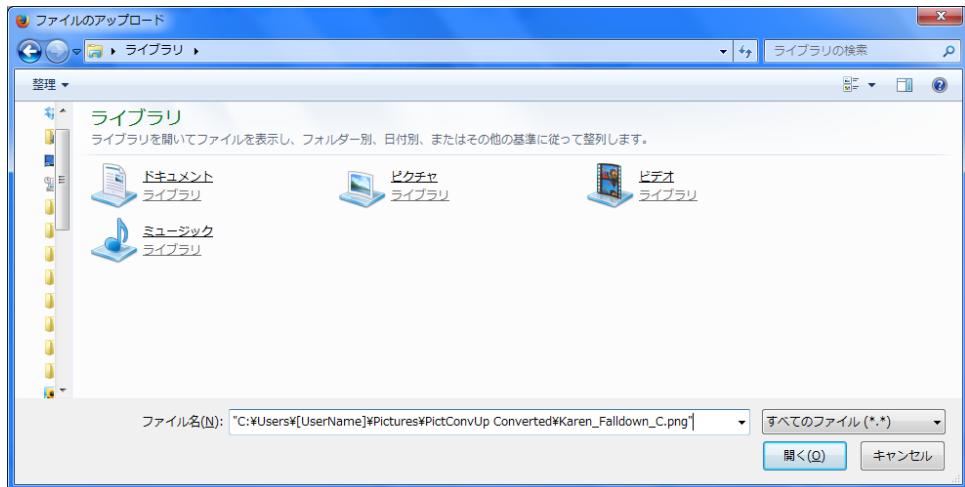


図 2: Firefox のファイルアップロード画面

2.3 変換した画像の自動削除

図 1「終了時に変換した画像を削除」が有効な場合、本アプリを終了時に変換後の画像ファイルを削除することができる。これによって、例えば、アップロードや他アプリへの取り込みなどが終わったあとに本アプリを終了すれば、変換後の画像を管理する必要がなくなる。

2.4 変換した画像をドラッグ&ドロップ

図1の変換した画像の一覧に表示されているアイテムをエクスプローラや他のアプリにドラッグ&ドロップすることにより、変換した画像を別の場所にコピーしたり、他のアプリで利用することができる。

3 設計

このソフトはC#で作成した。ソフトウェアの要となっている部分を幾つか紹介していく。

3.1 フォーム記述とドラッグ&ドロップ

このソフトでは、Windows Formsを利用してGUIを記述した。WPFも勉強してればよかった。

他のアプリ(エクスプローラとか)からドラッグ&ドロップを受け入れる場合は、メインフォームのクラスにDragEnterとDragDropイベントを設定する。[1]

また、変換した画像の一覧から画像をドラッグできるようにするには、ListViewクラスにItemDragイベントを設定し、その中で ListView.DoDragDropメソッドを呼び出す事により、指定したアイテム(ここでは画像)を他のアプリにドラッグ&ドロップできる。[2]

ここで、以下にドラッグ&ドロップのコードの一部を示す。興味のある方は参考文献[1, 2]やその他WEB上の情報も参考にしつつ読んでみてほしい。一部メンバやクラスの命名がおかしくなってる部分があるのはご容赦願いたい。

```
public partial class MainForm : Form
{
    /*中略*/
    //ListViewからドラッグされたらフラグを立てる
    //ListViewからドラッグされたアイテムをこのフォーム自身にドロップできないようにする
    bool thisDraging = false;

    //ドラッグされたデータのチェック
    private void MainForm_DragEnter(object sender, DragEventArgs e)
    {
        //ドラッグデータとフラグの確認
        if (e.Data.GetDataPresent(DataFormats.FileDrop) && !thisDraging)
            e.Effect = DragDropEffects.Copy;//正しいデータならコピーとして受け入れる。
        else
            e.Effect = DragDropEffects.None;//それ以外は受け入れない。
    }

    //ドラッグされたアイテムがドロップされた時の動作
    private void MainForm_DragDrop(object sender, DragEventArgs e)
    {
        //ドロップされたアイテムのファイル名を取得
        string[] fileName =
            (string[])e.Data.GetData(DataFormats.FileDrop, false);
        this.Activate();
    }
}
```

```

//入力されたファイルの変換を別スレッドで行う。
//メインスレッドの処理がこのメソッドを抜け出さないとエクスプローラが固まるため、
//時間がかかる操作はスレッドを分けている
BeginInvoke((MethodInvoker)delegate()
{
    //変換を行うメソッド
    ConvertPict(fileName, 0, fileName.Length);

});
}

//変換された画像の一覧から画像を変換した時の動作を記述
private void listViewConvertedHistory_ItemDrag
(object sender, ItemDragEventArgs e)
{
    //変換するファイルのリスト
    System.Collections.Generic.List<string> col =
        new System.Collections.Generic.List<string>();

    //選択されたアイテムをドラッグするリストに登録
    foreach (ListViewItem selected in listViewConvertedHistory.SelectedItems)
    {
        //変換された画像の情報を listViewConvertedHistory に登録する際、
        // ListViewItem.Tag プロパティに、別途以下に定義した ConvertedImageInfo オブジェ
        //クトを登録、
        //ConvertedImageInfo.ConvertedPath に変換された画像のフルパスを書き込んでおく
        ConvertedImageInfo info = selected.Tag as ConvertedImageInfo;
        if (info != null)
        {
            col.Add(info.ConvertedPath);
        }
    }
    //実際にドラッグするデータの準備
    DataObject dobj = new DataObject(DataFormats.FileDrop,col.ToArray());

    //ドラッグドロップのアクションを定義
    //DragDropEffects.Copy を設定し、エクスプローラにアイテムをドラッグするとコピーさ
    //れる
    //DragDropEffects.Move を設定してしまうと、ファイルが移動してしまうので設定しない
    //DragDropEffects.Link は、iTunes がドラッグを受け入れるのに必要
    DragDropEffects effect = DragDropEffects.Copy | DragDropEffects.Link;

    //フラグを立てることで、自分自身からドラッグされたアイテムを
    //自分自身が受け入れないようにする
    thisDraging = true;
}

```

```

//ドラッグドロップの実行
listViewConvertedHistory.DoDragDrop(dobj, effect);

}

//このアプリからドラッグしたアイテムをエクスプローラにドロップした時のイベント
private void listViewConvertedHistory_QueryContinueDrag
(object sender, QueryContinueDragEventArgs e)
{
    //ドラッグが完了したら
    if (e.Action != DragAction.Continue) {
        //フラグを降ろす
        this.Draging = false;
    }
}

//listViewConvertedHistory の定義とイベントハンドラの登録
//(通常は VisualStudio が自動的にやってくれる)
private System.Windows.Forms.ListView listViewConvertedHistory;
this.listViewConvertedHistory.ItemDrag +=
    this.listViewConvertedHistory_ItemDrag;
this.DragDrop += this.MainForm_DragDrop;
this.DragEnter += this.MainForm_DragEnter;
this.listViewConvertedHistory.QueryContinueDrag +=
    this.listViewConvertedHistory_QueryContinueDrag;

}

//変換された画像の情報
class ConvertedImageInfo : IDisposable {
    /*中略*/
    //変換後の画像のパス
    public string ConvertedPath
    {
        private set;
        get;
    }
}

```

3.2 画像の変換

.NET Framework では、簡単な画像の処理ができるライブラリが標準で含まれている。このアプリでは、.NET で使用出来る Image オブジェクトやそれを継承した Bitmap オブジェクトで描画キャンバスを定義し、その上で Graphics オブジェクトを用いて縮小後の画像を描画している。縮小された画像は、Image.Save メソッドで、PNG、JPG、BMP 形式などで保存することができる。

なお、画像を DrawImage で縮小描画する際、そのまま描画させると変換モードによっては縮小後に

画像の外周が乱れてしまう。それを防ぐために、図3のように、大きめの描画キャンバス（一時キャンバス）の中央に一旦画像を描画し、その周囲を画像の外周部と同じ色で塗っておき、縮小時に、大きめの描画キャンバスに描いた中央の元画像の部分を縮小後のキャンバスに描画するという手法をとっている。



図3: 一時キャンバスの描画

```
//インスタンスは作らない すべて static メソッド
static class PictureShrinker {
    /*中略*/

    //指定したサイズに画像を縮小(拡大縮小方法も指定)
    //拡大縮小時は、まずこのメソッドを呼び出す
    //返り値の Bitmap(Image クラスを継承) は呼び出し元で保存
    public static Bitmap Convert(Image toConvert, Size outputSize,
        System.Drawing.Drawing2D.InterpolationMode interpolation)
    {
        if (toConvert == null)
        {
            throw new ArgumentNullException("toConvert");
        }
        Bitmap newImg =
            new Bitmap(outputSize.Width, outputSize.Height, toConvert.PixelFormat);
        try
        { //縮小後の画像の描画キャンバスを用意
            DrawToNewImage(toConvert, newImg, interpolation);
        }
        catch (Exception)
        {
            if (newImg != null)
            {
                newImg.Dispose();
            }
        }
    }
}
```

```

//PixelFormat を 24bitにしてやり直し
newImg = new Bitmap
    (outputSize.Width, outputSize.Height,
     System.Drawing.Imaging.PixelFormat.Format24bppRgb);
//縮小後の画像を描画する
DrawToNewImage(toConvert, newImg, interpolation);
}
return newImg;
}

//newImg に toConvert の縮小イメージを描画させる
private static void DrawToNewImage
(Image toConvert, Bitmap newImg,
System.Drawing.Drawing2D.InterpolationMode interpolation)
{
    //一時キャンバスをもとの画像よりどれだけ大きくするか
    //縦クッションサイズ
    const int cussionSizeV = 100;
    //横クッションサイズ
    const int cussionSizeH = 100;
    using (Bitmap cussionBmp =
        new Bitmap(
            toConvert.Width + cussionSizeH*2, toConvert.Height + cussionSizeV*2)
        )
    {
        using (Graphics g = Graphics.FromImage(cussionBmp))
        {
            g.PixelOffsetMode = System.Drawing.Drawing2D.PixelOffsetMode.Half;
            g.CompositingMode =
                System.Drawing.Drawing2D.CompositingMode.SourceCopy;
            g.InterpolationMode =
                System.Drawing.Drawing2D.InterpolationMode.NearestNeighbor;

            //画像本体
            g.DrawImage(toConvert, new Rectangle(
                cussionSizeH, cussionSizeV, toConvert.Width, toConvert.Height));

            //以下で、画像の周囲を元画像の縁と同じ色で描画しておく
            //上
            g.DrawImage(toConvert,
                new Rectangle(cussionSizeH, 0, toConvert.Width, cussionSizeV),
                new Rectangle(0, 0, toConvert.Width, 1), GraphicsUnit.Pixel);
            //下
            g.DrawImage(toConvert, new Rectangle
                (cussionSizeH, cussionBmp.Height - cussionSizeV,
                 toConvert.Width, cussionSizeV),

```

```

        new Rectangle(
            0, toConvert.Height - 1, toConvert.Width, 1), GraphicsUnit.Pixel);
        //左
        g.DrawImage(toConvert,
        new Rectangle(0, cussionSizeV, cussionSizeH, toConvert.Height),
        new Rectangle(0, 0, 1, toConvert.Height), GraphicsUnit.Pixel);
        //右
        g.DrawImage(
        toConvert, new Rectangle(
        cussionBmp.Width - cussionSizeH, cussionSizeV,
        cussionSizeH, toConvert.Height),
        new Rectangle(toConvert.Width - 1, 0, 1, toConvert.Height),
        GraphicsUnit.Pixel);
        //左上
        g.DrawImage(toConvert, new Rectangle(0, 0, cussionSizeH, cussionSizeV),
        new Rectangle(0, 0, 1, 1), GraphicsUnit.Pixel);
        //右上
        g.DrawImage(toConvert, new Rectangle(
        cussionBmp.Width - cussionSizeH, 0, cussionSizeH, cussionSizeV),
        new Rectangle(toConvert.Width - 1, 0, 1, 1), GraphicsUnit.Pixel);
        //左下
        g.DrawImage(toConvert, new Rectangle(
        0, cussionBmp.Height - cussionSizeV, cussionSizeH, cussionSizeV),
        new Rectangle(0, toConvert.Height - 1, 1, 1), GraphicsUnit.Pixel);
        //右下
        g.DrawImage(toConvert, new Rectangle(
        cussionBmp.Width - cussionSizeH, cussionBmp.Height - cussionSizeV,
        cussionSizeH, cussionSizeV),
        new Rectangle(toConvert.Width - 1, toConvert.Height - 1, 1, 1),
        GraphicsUnit.Pixel);

    }

#if DEBUG
    cussionBmp.Save("DebugCussionBmp.png");
#endif
//縮小画像の描画
using (Graphics g = Graphics.FromImage(newImg))
{
    g.CompositingMode =
        System.Drawing.Drawing2D.CompositingMode.SourceCopy;
    g.InterpolationMode = interpolation;
    g.PixelOffsetMode = System.Drawing.Drawing2D.PixelOffsetMode.Half;
    g.DrawImage(cussionBmp,
    //一時キャンバスの元画像の部分を縮小描画する
    new Rectangle(0, 0, newImg.Size.Width, newImg.Size.Height),
    new Rectangle(

```

```
        cussionSizeH, cussionSizeV, toConvert.Width, toConvert.Height),  
        GraphicsUnit.Pixel);  
  
    }  
}  
}  
}  
}
```

4 まとめと展望

今回は、「ぴっこんなっ!」の紹介や設計について簡単な説明を行った。
インデックスカラー(256色)への変換対応や透過GIFや透過PNGの処理(透過部が黒になる)の改善・その他の形式への変換は今後の課題である。

参考文献

- [1] Drag&Dropを行う: .NET Tips: C#, VB.NET,
<http://dobon.net/vb/dotnet/control/draganddrop.html>
- [2] ファイルやディレクトリをエクスプローラへドラッグ＆ドロップするには? - @ IT:,
<http://www.atmarkit.co.jp/fdotnet/dotnettips/384expdragdrop/expdragdrop.html>

国立大学法人 電気通信大学
工学研究部 部報 第44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
URL <http://delegate.uec.ac.jp:8081/club/koken/>
E-mail koken@koken.club.uec.ac.jp

発行 皆川 太志
編集人 住澤 拓秀
表紙 河村 大輝
発行 2013年8月20日
執筆 工学研究部 部員