

部報

第48号

電気通信大学
工学研究部

kōken



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。

電気通信大学工学研究部 部報

第 48 号

目次

部長挨拶	2
	横田
半自動卓-ダイス付き点数計算表示器	3
	石関・筒井
アンプ IC を用いたお手軽オーディオアンプ作成	5
	聖徳たいし
月刊 ゲーム制作 -7 月号-描画編	7
	横田
ラーメンタイマーを作った	9
	粟島
ガジェットと生きる	10
	村上
Intel Edison を触ってみた	11
	樋口
工研で広がる修理の輪	15
	菅野
シンセサイザーをつくろう	17
	まひる
BS 地デジ分波分配器の製作	18
	ふな

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。

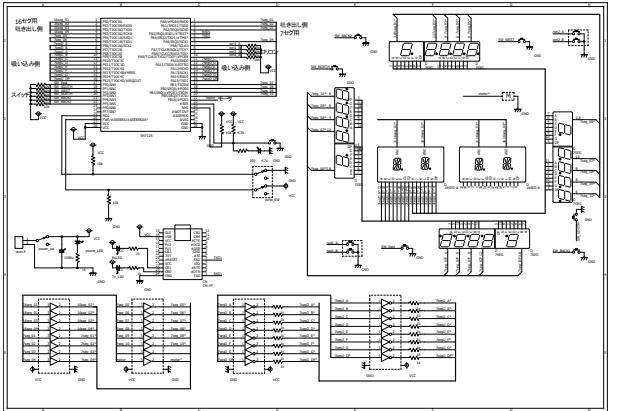


図 4: 回路図

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。

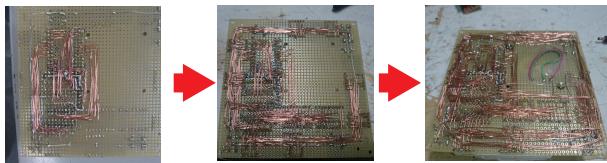


図 5: ハンダ漬け

更にハンダ付けします。

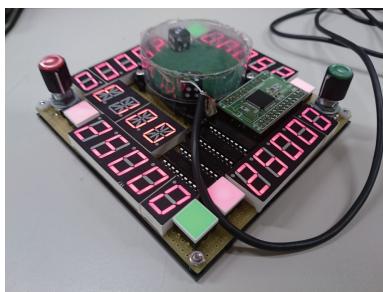


図 6: 回路完成

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。

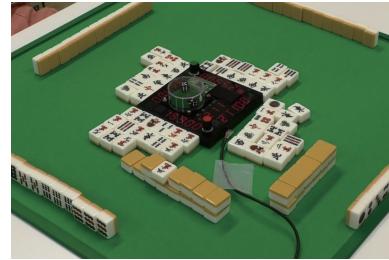


図 7: デバッブの様子

各部名称と主な機能	
共用表示部	表示部
表示部	表示部は、プレイヤーの点数を表示する。また、各色ボタンを押すと、現在の点数を表示する。
赤色ボタン	赤色ボタン
緑色ボタン	緑色ボタン
白色ボタン	白色ボタン
ロータリードials	ロータリードials
青色ボタン	青色ボタン
緑色ボタン	緑色ボタン
ローラードローブ	ローラードローブ

ダイスロールフェイズ
窓の赤滅している赤色ボタンを短押しすることでダイスロールします。

リーチフェイズ
赤色ボタンは数を表示している
リーチマークを表示する
窓の赤滅している赤色ボタンを短押しする
和了した際に赤色ボタンを長押しする(和局した際)

和了フェイズ
ロンボタンは次の場合
白らの表示部が、翻と符の表示に切り替わる
1. ローテリードialsを回し、符を入力する
(ルーテリードialsを回すと、表示部が白らになります)
2. 正しい朝が入力されたら、その赤色ボタンを押す
3. ローテリードialsを回し、符を入力する
(ルーテリードialsを回すと、表示部が白らになります)
4. 正しい符だた場合
ソモなう自分の赤色ボタン
ソモなう振り込んだ人の赤色ボタン

流局だった場合
1. ルーテリードialsを回すと、符を入力します。
2. ドンバーの人の赤色ボタンを押すと、点滅から点灯に貢ります。
3. 赤色ボタンを長押しで決定します。

精算
ダイスロールフェイズ・リーチフェイズ・和了フェイズで各ボタンを同時に長押しすることで
精算する。各ボタンの表示部が同時に点滅する。

点滅を確認したいときは両方の緑色ボタンを長押しします。

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。

おまけに、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！

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成してみました。音質に対して部品点数も少なく安くてお手軽でそこそこのいい音です。

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 に収められているため、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。

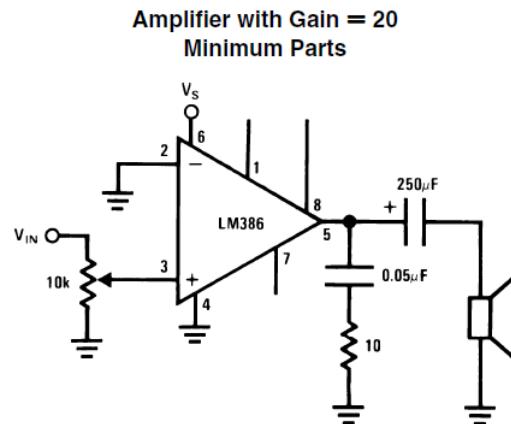


図 2: 推奨回路図

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。

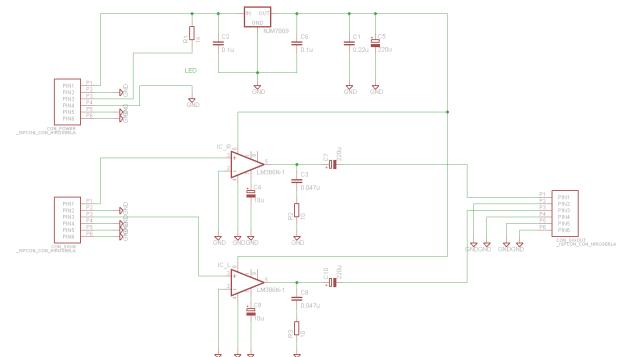


図 3: 設計した回路図

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)

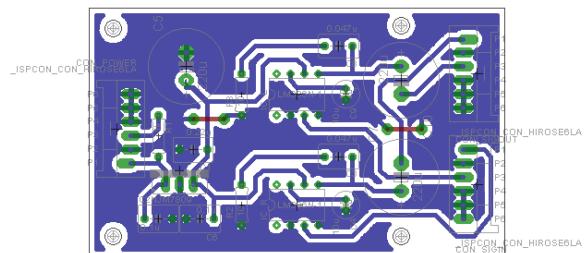


図 4: パターン図

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。

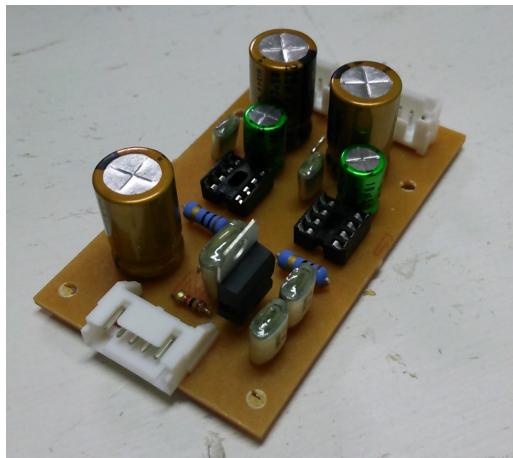


図 5: 実装

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。

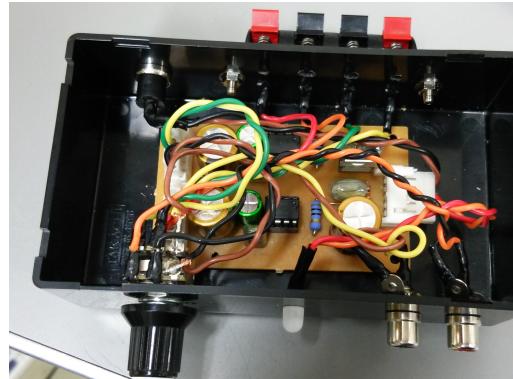


図 6: ケースに基板を格納

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。

月刊 ゲーム制作 -7月号-描画編

横田 嶺

2014年7月

1 概要

今回はゲームの構成要素として欠かせないグラフィックについて解説します。前回までにWindows上にゲームウィンドウを表示するところまで出来たと思います。そこでゲームウィンドウ内にキャラクターや得点など様々なモノを表示させないとゲームとして成り立ちません。

2 画面の描画

私たちに見えている画面というのはどのように表示されているのでしょうか。画面というのはピクセルという点を最小単位として、その点の集まりで構成しています。

ピクセルは光の三原色である赤、緑、青(RGB)の情報を持っておりRGBの配合の割合で色を表現しています。何段階で赤や緑、青を表すかをビット深度と言い1原色あたり8bitならば1ピクセルで24bitの深度があり $2^8 \times 2^8 \times 2^8 = 16777216$ 色を表現できます。例えばフルHDならば1920×1080ピクセルといったようにピクセル数が多いほど情報量が多く、すなわち解像度が高いということになります。

描画とはすなわちピクセルの色を変えることです。画面に敷きつめられたピクセルを場所ごとに色を変えることで写真や絵を表示できます。また、時間変化させれば動画ということですね。DXライブラリのウィ

ンドウモードの場合640×480ピクセルあります。左上を(0,0)として座標で位置指定を行います。

3 画像データの読み込み

まず表示させたい画像を用意します。表示させる画像はHDDなどの2次記憶装置に置いてあります。ここからデータを読み出すこともできますが同じ画像を何度も呼び出すときはHDDとのアクセスが増え処理が重くなります。そのため通常、画像を読み込む際はメモリに格納します。

4 画像の表示

実際にどのようにプログラムを書いていくか見てみましょう。表示させたい画像は適切な場所に保存しておいてください。私の場合ソースファイルと同じ場所に保存しました。

まず画像データを格納する変数を用意します。その変数にLoadGraph()という関数を用いて画像データを代入します。

今後もたくさんの関数を用いるので使い方が分からぬ場合、その都度リファレンスのページを参照してください。

次にDrawGraph()という関数で表示します。第一引数と第二引数で画像の左上の座標を指定できます。

今回は画像が表示できたらOKです。

5 次号

今回の表示方法では実は1つ問題があります。次回はその問題を解決し画像を動かしましょう。

6 画像の表示テスト

画像をウィンドウ内に表示できたら成功です。

```
#include "DxLib.h"

int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int){
    ChangeWindowMode(TRUE);           // ウィンドウモードにする
    DxLib_Init();                   // DX ライブラリ初期化処理

    int handle;                     // 画像ハンドル格納用変数
    handle=LoadGraph("ball.png");   // 画像をメモリにロード
    DrawGraph(10,20,handle,true);   // 画像ハンドルを使って画像を表示

    WaitKey();                      // キー入力待ち
    DxLib_End();                   // DX ライブラリ終了処理
    return 0;
}
```



図 1: 画像をウィンドウに表示

参考文献

- [1] DX ライブラリ・リファレンスのページ <http://homepage2.nifty.com/natupaji/DxLib/dxfunc.html>
- [2] 新・ゲームプログラミングの館, <http://dixq.net/g/>

ラーメンタイマーを作った

先進理工学科一年 粟島裕大

1. はじめに

マイコンでタイマー割り込みを学習するためにラーメンタイマーを作成してみました。使用マイコンはAVRマイコンのatmega328Pです。

2. タイマー割り込み

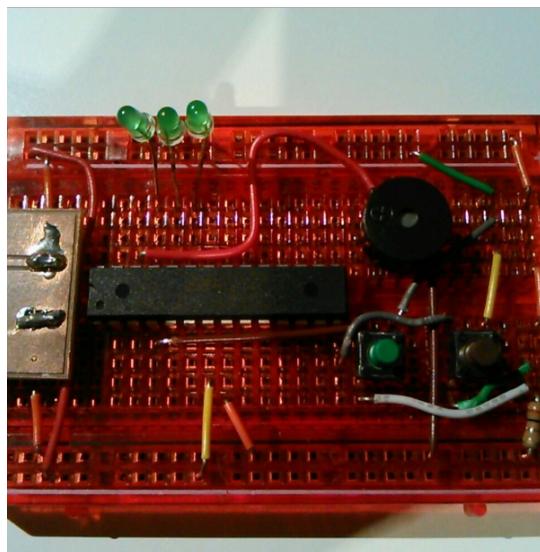
タイマー割り込みの細かい説明はここでは省きますが、これができると予め決めておいた間隔で決められた動作をマイコンにさせることができます。今回は「1秒ごとにある変数を1ずつカウントアップしていく」というものです

3. 初期設定

タイマー割り込みを使用するためには初期設定をする必要があります。

```
TCCR1A = 0b00000000; // CTC動作 タイマー1
TCCR1B = 0b00001011; // 64分周動作
TCCR1C = 0b00000000;
OCR1AH = 61;
OCR1AL = 8; // 比較値 15624, 1秒で割り込み
TIMSK1 = 0b00000010; // タイマー1A割り込み許可
今日はこの設定で行いました。
```

4. 完成品



2つスイッチを使っていて1つがスタート用、もう1つがリセット用になっています。1分ごとに右からLEDが点灯して進んで行くようになっており、例えば2分x秒なら右のLEDが点灯、真ん中が1秒おきに点滅、左が消灯となります。3分が経つとブザーで知らせるようになっています。

5. 感想

3分きっちり測ったラーメンはいつもより少し美味しい気がしました。LEDのところを7セグに変えたい。

ガジェットと生きる

先進理工学科2年 村上 大和

2014年11月6日

1 はじめに

情報端末、いわゆるガジェット、デジモノの技術は日々進化している。インターネットニュースサイトでは毎日のように新製品のレビューや発表会の速報が配信されている。それらは私を刺激して止まない。

2 スマートフォン

現在、スマートフォンといえばiOSとAndroidの二大巨頭といってもいいだろう。iPhone6やNexus6の発売は記憶に新しい。一方で、Kindle Fire PhoneやWindows Phone, FireFox Phone, Blackberryなども一定の評価を得ている。

2.1 iPhone

iPhoneの素晴らしいところに常に時代の一歩先を行く革新的なデザインであることが挙げられる。iPhoneの発売前後で携帯電話端末の形状だけでなく操作方法も大きく変わった。携帯電話に当たり前のようにあったダイヤルパッドを撤廃し、マルチタッチ操作を基本としたインターフェースは、今やどのスマートフォンにも見られる。

近年のiOS, MacOSの動向から察するところ、MacとiPhoneの通知を共通化するなどしてよりAppleの統一したユーザーエクスペリエンスの強化を押し出している。いわゆる、独自規格による囲い込みであるが、モバイル、デスクトップのハードウェアとソフトウェアのどちらも自社で開発するAppleならではの強みである。

2.2 Android

AndroidはGoogleが開発したオープンソースのOSである。この、オープンソースというところがミソで、各メーカーがハードに合わせてカスタマイズするだけでなく、Amazon KindleやFireFoxなどのOSのベースになっていたり、ユーザーによるOSのカスタマイズも活発に行われている。

というのも、最新AndroidOSに発売から18ヶ月間は対応することが保障されていることである。Android端末は1万円程度の低価格から豊富に販売されている。中でもやはりGoogle公認のNexusシリーズが安定性や信頼性の面で群を抜いている。メーカーによってはアップデートが一切ない場合や、同一モデルでもグローバルモデル向けのアップデートはあるが、日本向けモデルにはアップデートされないことある。ハードウェア制約により、アップデートによる恩恵が少ない場合があるが、最新のOSが触れるという魅力は大きい。もちろん、中には各メーカーもアップデート対応はあるのだが、現在大手メーカーのほとんどが来年春以降の配信となっているため大きな差があり、その頃には次世代のレビューが始まっていることがある。また、NexusシリーズはSIMフリー端末であるため、海外利用や各社MVNOサービスを利用できる。一方で、3大キャリアから直接販売されてないため、設定などはすべて自分で行う必要があるため、わからないことは自分で調べるなどの当り前の努力が必要である。

3 あとがき

締め切りの都合上、本当は書きたかったMVNOやタブレット、中華端末について書きたかったのだが、2大スマホの紹介で終わってしまった。それはまた今度の話に。

Intel Edison を触ってみた (工研部報)

1313144 樋口孔明

1 Intel Edison とは

Intel Edison は、米の Intel 社が 9 月 10 日に発表した Atom 搭載の超小型コンピュータです。日本では 10 月 25 日に発売が開始されました。どのようなものなのかは、
https://communities.intel.com/servlet/JiveServlet/downloadBody/23139-102-2-27268/edison_PB_331179-001.pdf より引用すると

The Intel(R) Edison development platform is designed to lower the barriers to entry for a range of inventors, entrepreneurs, and consumer product designers to rapidly prototype and produce “Internet of Things” (IoT) and wearable computing products.

簡単にいえば手のひらサイズの IoT 的な感じだと思います(違っていたらごめんなさい)。性能は書いてるときりがないので上の pdf を見てください。簡単に言ってしまえば Intel Atom Processor が乗ってる SoC です。今回は Intel Edison に加えて、その拡張ボードである Intel Edison Breakout Board を購入していじってみました。母艦は Mac+OS X 10.10 Yosemite で行いました(今思うと、Mac ってこういうの使いづらすぎるなーと思ったり)。

2

とりあえずは起動してみないと何も始まらないので、Intel 公式の Getting Start(<https://communities.intel.com/community/makers/edison/getting-started>) を呼んでから始めることにしました。…なのですが、Breakout Board に関する説明が全くない。これっぽっちも見つからない。おまけに初心者は Arduino 互換の Board を使えと書いてあるのと他の Board に関しては自分で調べろと書いてあるではないですか…。頑張って探してみるがなかなか見つからない(探し方が下手なだけだったというオチ)。結局公式では無くここのサイト(<http://qiita.com/yoneken/items/1b24f0dd8ae00579a0c2>) を参考に進めました。

大まかにやったことは、

1. ディストリビューションをストレージに焼く
- 2.シリアル接続をする
3. 初期設定
4. Hello, Edison!!!

という感じです。

2.1 ディストリビューション焼き

ディストリビューションを落とす前に”\$ brew install dfu-util coreutils gnu getopt”で必要なパッケージをインストールしておきます。brewがない人はググって入れるなり何なりしましょう(brew以外のやり方は知らないです)。

入れ終わったら、<https://communities.intel.com/docs/DOC-23242>からYocto LinuxのComplete Imageを落としてきます。展開すると、.binファイルやら.imgファイルがたっぷりはいったフォルダがある中に”flashall.sh”というシェルスクリプトがあるのでEdisonをさした状態で実行してみます。

すると、”Please plug and reboot the board”と言われるので電源を一旦切ってから入れ直します。しばらくほうっておくとU-Bootだったりカーネルだったりを焼き始めるので5分程度待ちましょう。終わるといろいろメッセージが出た後に”Please unplug it for 2 minutes”と言われるので2分ちょっと待ちましょう。待つ意味があるのかどうか走りませんが。

これだけでイメージの焼き作業は終わりです。ddコマンドなんて使わなくていいから楽ちん！

2.2 シリアル接続

どのOSを使っていてもここは”\$ screen /dev/cu.usbserial”で通信ができるようですが(screenでこういうことできることを知らなかつたマン)、自分の環境だとなぜか出来ませんでした。半日潰して調べてもなんせ情報量が全くないのでコミュニティにすら引っかかる有り様…。どうしようかなー俺もなーとか思っていてふと”\$ minicom -D /dev/cu.usbserial”とするとあっさり繋がり、(^_-;)状態。結局なんだったのかは今もわからない感じです(原因すらわからない)。

2.3 初期設定

Yocto Linux, Gentoo Linux, Arch LinuxなどのLinuxディストリビューションは、あまりコンソールに馴染みのない人にとってはこういう設定がすごくつらい感じがしてならないです(UbuntuだったりDebianは除く)。しかし、Edison版Yocto Linuxでは素晴らしいことに”\$ confiure_edison”という素晴らしいコマンドが付いているではありませんか！！これを使えばクソみたいに面倒なwifiの設定だって自動でやってくれます！！！(WPAにも対応している模様)

ということなので有り難みを感じつつ”\$ configue_edison --setup”をしていきましょう。このコマンドでは3つについて聞かれます。

1. hostname
2. root password
3. wifiへの接続の有無と設定

この 3 つについて対話式にわかりやすく設定してくれるのでさいつつつつつつこうに Cool なのです (wifi の設定とかややこしすぎて自分には出来ないです)。とはいっても他にも設定しなければならない項目は多々あるわけあります、それらに関してはコンソールと戦闘しながら設定していかなければならぬのです (ヘタするとカーネル消し飛ばすので慎重に)。

コンソール上で設定することは

1. locltime
2. opkg

の 2 つです。あんまり難しくはない (むしろ普段からやってる人にとっては朝飯前でしょう) と思います。慣れれば。

locltime はそこまで面倒ではなく、

- ”\$ rm /etc/localtime”
- ”\$ ln -s /usr/share/zoneinfo/Asia/Tokyo /etc/localtime”

とすれば終わりです。rm はわかると思います。ln -s はシンボリックリンクを張っているだけです。ね、簡単でしょう？

もう一つの opkg の設定は少しややこしいですが、まあそこまででもないです。まず、/etc/opkg/base-feeds.conf ファイルを vi(これ以外でもいいですが、入っていないような気がします)で開き、

```
src/gz all http://repo.opkg.net/edison/repo/all  
src/gz edison http://repo.opkg.net/edison/repo/edison  
src/gz core2-32 http://repo.opkg.net/edison/repo/core2-32
```

と記入、その後/etc/opkg/intel-iotdk.conf を同じく vi で開き、

```
src intel-iotdk http://iotdk.intel.com/repos/1.1/intelgalactic  
src intel-all http://iotdk.intel.com/repos/1.1/iotdk/all  
src intel-i586 http://iotdk.intel.com/repos/1.1/iotdk/i586  
src intel-x86 http://iotdk.intel.com/repos/1.1/iotdk/x86
```

と記入。最後に”\$ opkg update”、”\$ opkg upgrade”すれば終わりです。ようはパッケージがどこにあるかを指定して最新版に更新しているだけです。ここまでややこしくはないし、。

ここで初期設定は終わり、新しいとびらは開けたと思いますが、まだやっておいたいいことがあります。”\$ df -h”してもらうとわかるのですが、”/”の容量がほとんど残っていない代わりに、”/home”に空き容量が割り当てられています。これでは新しいパッケージが入れられないで少しいじることにします。

”\$ vi ~/.profile”と打ち、export PATH=~/local/bin:\$PATH と入力して保存し、”\$ source ~/.profile”と入力します。こうすることで、インストール先を”/bin”と”/home/root/.local”の 2 つに増やすことが出来ます。

ついでに、”/etc/opkg/opkg.conf”に”dest local /home/root/.local”という文も追加しておきましょう。こうすることで”\$ opkg install git -d local”とすることで git を”/home/root/.local”以下にインストールすることが出来ます。

2.4 Hello, Edison!!!

これで Intel Edison の新しい扉を開くことが出来るようになったのです！おめでとう！いつも使っている PC に比べると性能は見劣りしますが、なによりも携帯性に優れ、その携帯性に見合わないぐらいの性能をもつ Edison は、本当に夢のあるデバイスだと思います。

3 おわりに

今年の 4 月に入ってから初めて部報を書いたので、色々とアレだったかもしれません、これをみて Edison 買ってくれる人が増えるといいなあとちょっと思ったりしています。

今回あまり期間がなかったので導入しか出来ませんでしたが、もう少しいじって GUI 環境とかも作れるといいなあと思っています。

ということでみなさん Edison を買っていじくりまわしましょう！UART や I2C もできるよ！！

工研で広がる修理の輪

情報・通信工学科 1年

菅野 翔太

1. 概要



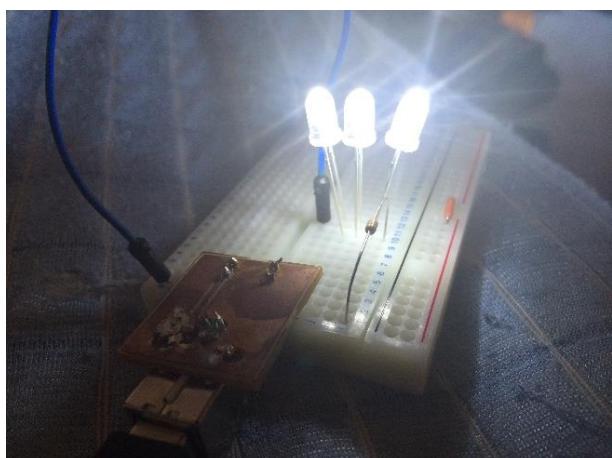
まず始めに、当部報は技術的というよりも日常的であり、かなり簡単な内容であることについて断りを入れておく。

愛用のスタンドライト「SL-104」（図1）が壊れた。詳しく言うと2つあるLED電球が焼けきってしまったのか、最初は片側、最終的には両方共点かなくなってしまった。最初はやはり買い替えを考慮に入れ、財布がまた寒くなるのを憂いていた。

しかし、せっかく手元に工具や電子部品があるので、自分の手で修理して安く仕上げようじゃないかという考えに息行き着き、基板レベルでの修理に踏み切った。

図1. SL-104

2. 準備物



- ・高輝度白色 LED 3 個
- ・ 150Ω 抵抗 1 個
- ・ユニバーサル基板 1 枚
- ・ブレッドボード 1 枚
- ・USB 電源 1 個

必要に応じて、ジャンパー線等の補助部品を用意するとよい。

図2. ブレッドボード上でのテスト接続

3. 修理の実行

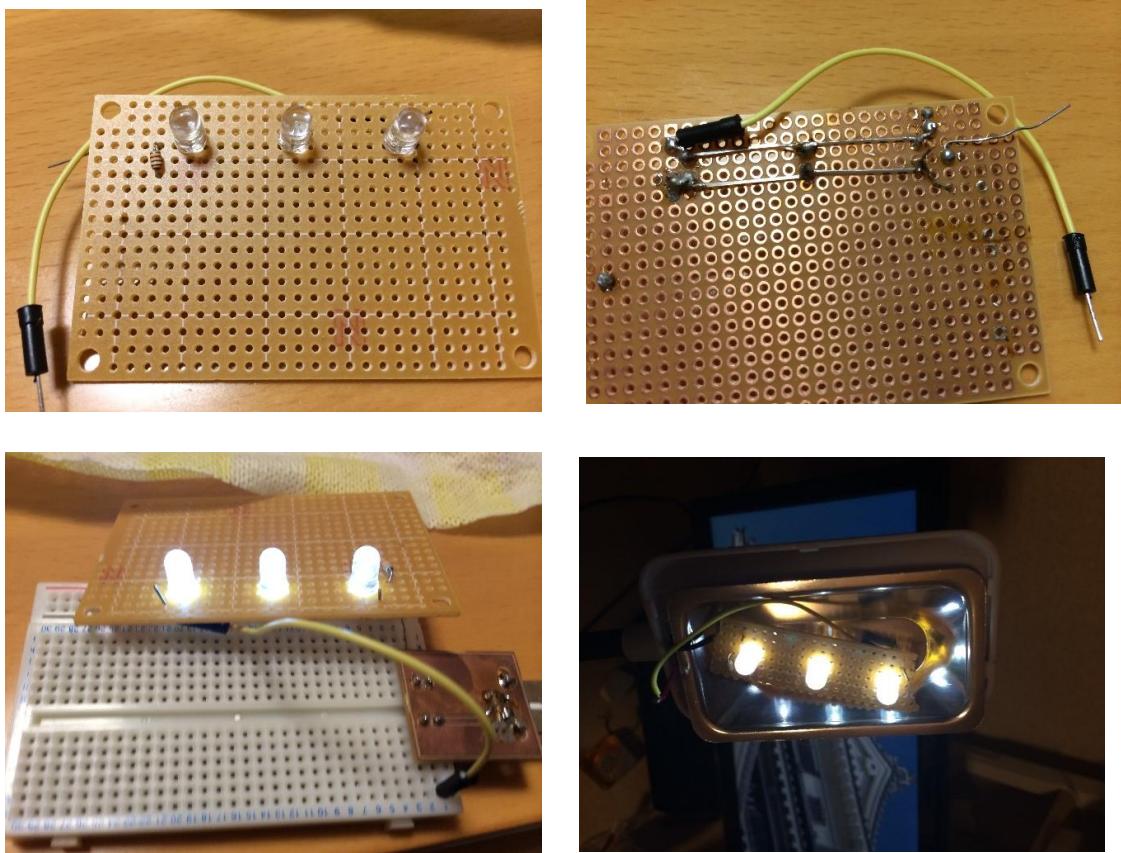


図3～6. 修理工程

ユニバーサル基板はプリント基板と異なり、自分で回路を設ける必要がある。私の場合、LED や抵抗の足の切れ端をハンダ付けして回路を作った。正極、負極を伸ばし、ストップライトの電極と結びつけて電源を入れたところ、無事 LED が点灯し、本来の役割を果たせるようになった。

4. 感想

このように、工研で得た知識は専門的な工作でなくとも、日常的に起こりうる家電製品の故障にも対応できる。さらに、モノをすぐに捨てずに修理することは、お金の節約になるばかりでなく廃棄するゴミの量を減らすことにもつながる。

シンセサイザーをつくろう

情報・通信工学科 1年 まひる

1 動機を書こう

唐突に簡単なシンセサイザーを作つてみたいなと思ったので、とりあえずデジタルでどうすれば作れるかなと考えてみることにしました。

2 出す音を考えよう

単純に正弦波を得たい場合は、時刻を t/s 、周波数を f/hz として、振幅を A としておくと

$$x_{\sin}(t) = A \sin 2\pi ft, (0 \leq A < 1) \quad (1)$$

でいいわけですが、合成してないのにシンセサイザーも何もあったもんじゃない気がするので、せめて鋸波くらいは得たいものです。[/ / /] みたいなギザギザ形の波形になるやつです。理想的な鋸波は次の式で表せます。

$$x_{\text{saw}}(t) = \frac{2}{\pi} \sum_{k=1}^{\infty} \frac{x_{\sin(kt)}}{k} \quad (2)$$

ただし、可聴域の範囲があれば十分でしょうから、 k は $fk < 2 \times 10^4$ の範囲、つまり $k < (2/f) \times 10^4$ 程度でよさそうです。

抑揚の無い音を得てもつまらなさうなので、次は減衰させるようにしましょう。これは単純に減衰の速さの度合いを γ とでもおいて、

$$x_{\text{out}}(t) = e^{-\gamma t} x_{\text{saw}}(t), (0 \leq \gamma) \quad (3)$$

とでもしておけばよさそうです。

3 PCM データを作ろう

シンセサイザーを作つうとしてるのに wav ファイルに出力しようとしてる時点ではどうなのという感じがしますが、とりあえず簡単なところから。ヘッ

ダ情報は置いておくとして波形データについて考えてみます。一般的な CD の音質ではサンプリング周波数 44.1kHz で符号付き 16bit なので、さっきの $x_{\text{out}}(t)$ を使い、 s をサンプリング数として、

$$x_{\text{PCM}}(s) = 2^{15} \times x_{\text{out}} \left(\frac{s}{44100} \right) \quad (4)$$

を計算して int 型の配列に突っ込めば PCM データになります。これをバイナリで出力すればなんなくいけそうですね。…いけるのか？

4 PWM 出力しよう

先ほど作った PCM データから PWM 信号を作るのも容易そうです、多分。44.1kHz ですから、1 サンプルあたりの時間は $1/44100$ 秒になります。同じ PWM 信号を流し続ける時間である PWM 周期をこれより十分小さくとればよいので、仮にこの $1/20$ 程度に PWM 周期 T/s をとると、

$$T = \left(\frac{1}{44100} \right) \frac{1}{20} = 1.13 \times 10^{-6} \quad (5)$$

程度になります。更に、あるサンプル s に対する PWM デューティー比 $D(s)$ は、

$$D(s) = \frac{x_{\text{PCM}}(s)}{2^{15}} \quad (6)$$

として、 $D(s)$ の符号で場合分けして出力すればいいそうです。…いけるのか？

5 おしまい

合ってるのかはなはだ不安な内容になってしましましたが、実際に作つたりはしていないのでとりあえずここまで終わります。

1 はじまり

私が工研を去ってから約3年半ぶりに記事を書かせていただくのですが、みなさまいかがお過ごしでしょうか？

一昨年のことですが、私は SDR-Kits が販売している小型ネットワークアナライザ VNWA3E (図1) を購入しました。VNWA は小型の USB 機器ですが、1kHz~1.3GHz までの S パラメータを測定することができます。しかしながら数 100MHz を超える回路を扱う機会はなかなかなく、VNWA の性能をフルで使ってあげられていませんでした。そんな中、ことは私の友人の DAC 氏から PC 用チューナーカードを頂いたところから始まります。



図 1: ネットワークアナライザ (VNWA3E)

BS 放送が見れる一般的な家庭では、図2に示すように地デジアンテナが受信した信号と BS アンテナが出力する BS-IF 信号を混合器により混合し、1 本のアンテナ線に流すようになっています。最近のテレビなどは、室内のアンテナ線をそのまま接続可能なものが多いのですが、入手したチューナーカードや、一部のレコーダなどでは図のように機器の入力が BS と地デジでわかれています。前段に分波器と呼ばれる装置を必要とするもののが存在します（直接、地デジ BS-IF 混合信号を入力できる機器も内部に分波回路を持っている）。

私は分波器を持っていなかったため、巷で購入するか自作する必要がありました。分波器は安いもので数 100 円程度から入手可能なのですが、せっかくネットワークアナライザを購入したので、自作することにしました。

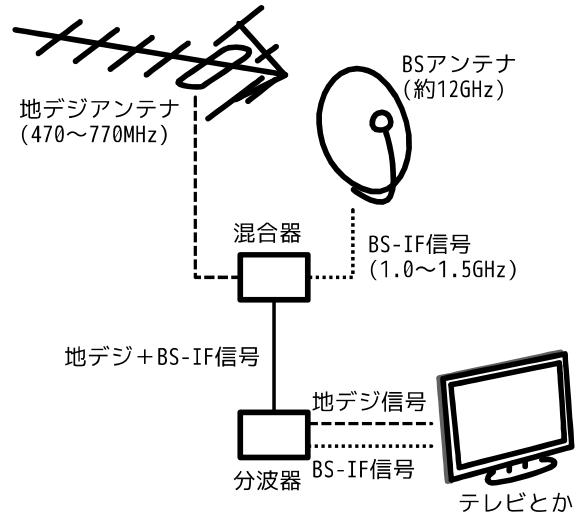


図 2: 一般的なアンテナ配線

2 仕様とか

自作する BS 地デジ分波分配器の構成を図3に、仕様を表1にそれぞれ示します。今回接続する PC チューナーカードは地デジ×2、BS × 2 の入力端子を持つので、分波分配器により入力を4分配します。挿入損失はなるべく少なくしたいので、LPF と HPF を用いた分波回路により入力信号を地デジ周波数帯と BS-IF 周波数帯に分離した後、分配回路により2分配します。また、使用する部品は分布定数線路とチップキャパシタのみに限定し、伝統的なフィルタ設計理論を使って設計を行います。古典フィルタ理論を使った LPF や HPF で広帯域特性を実現することは難しいので、CS 周波数帯はあきらめて 1300MHz までにします。

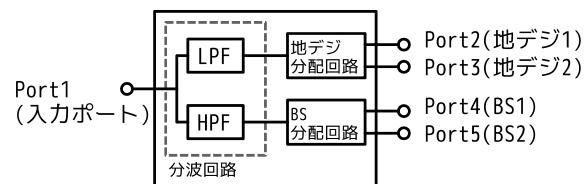


図 3: BS 地デジ分波分配器の構成

表 1: BS 地デジ分波分配器の仕様

項目	仕様
出力構成	地デジ×2、BS × 2
周波数	470~770MHz、1000~1300MHz
挿入損失	-6dB 以内
回路構成	LPF+HPF
フィルタ特性	バターワース

3 分波器と分配器

ここで少し分波器と分配器の違いについて触れておきたいと思います。分配器と分波器は、どちらも入力した信号を2つ以上のポートに分配する回路です。分配器と分波器の入力信号と出力信号の関係を大まかに表すと下図のようになります。分配器は周波数に関係なく信号を分配するため、受信信号をテレビとレコーダに同時に分配する用途などに用いられます。一方で分波器は入力された信号を、地デジとBSというように、周波数で分離するために用いられます。受信信号をテレビとレコーダに分配するために分波器を用いた場合、テレビとレコーダは地デジとBS、それぞれどちらか一方しか見れなくなってしまいます。

分配器は入力電力を単純に各ポートに分配するため、出力される電力は $1/2, 1/3, 1/4 \dots$ と減っていきます。一方で理想的な分波器は、周波数を分離するだけなので、単位周波数あたりの電力は減少しません。今回は入力を4分配しますので、すべて分配器で構成した場合電力は無損失の回路であっても-6dBとなります。図3に示した回路構成を用いた場合-3dBに抑えることが可能ですが(実際には回路の損失があるのでどちらの構成もさらに損失が増加する)。

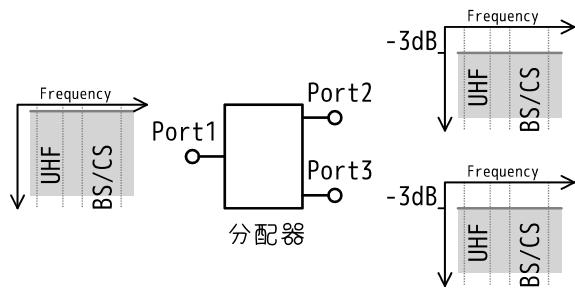


図4: 分配器

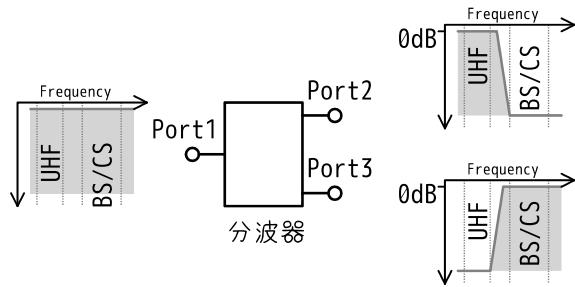


図5: 分波器

4 回路の設計

まず、分波回路を構成するLPFとHPFについて設計を行います。LPFとHPFはそれぞれ図6(a)(b)に示す集中定数回路をもとにインバータ変換と呼ばれる変換計算により分布定数回路に近似します。インバータとは図7に示すように、インバータを介して接続された任意のインピーダンスをその逆数に変換する回路のことであり、インダクタをキャパシタに交換し、分布定数線路とキャパシタのみから構成される回路に変換することができます。また、90度の位相を持つ伝送線路はインバータとして機能することが知られています。図8に伝送線路を用いたインバータ変換により、変換されたLPFとHPFの回路構成を示します。

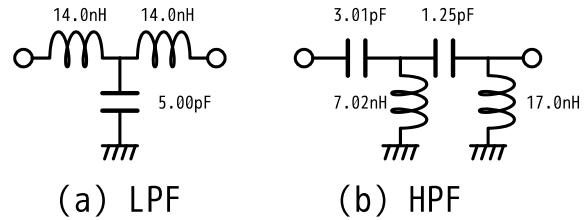


図6: LPFとHPFの集中定数回路

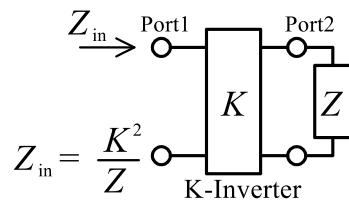


図7: インバータ

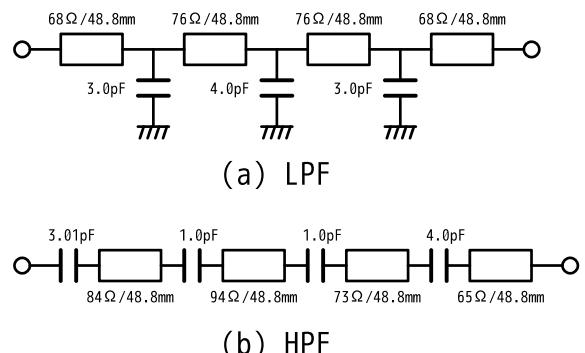


図8: LPFとHPFの分布定数回路

次に分波回路後段に接続する分配器について説明します。今回分配器としては図9に示すウィルキンソン分配回路を用いました。2等分配のウィルキンソン分配器は、 $\lambda/4$ の伝送線路2本と、出力ポート間をつなぐ抵抗から構成された線対称回路です。この分配器は、信号が出力ポート(ポート2またはポート3)に入力された場合に、 $\lambda/4$ の伝送線路×2を通って位相が反転した信号と、抵抗を通った信号とが打ち消しあうようになっており、出力ポート間のアイソレーションが取れるようになっています。

また、2本の伝送線路の特性インピーダンス Z_s と、出力ポート間の抵抗 R_p を適切に選ぶことで、ポート1の入力インピーダンスを基準インピーダンス Z_0 と整合させることができます。今回、導出方法については割愛しますが、 $Z_s = \sqrt{2}Z_0$ 、 $R_p = 2Z_0$ となります。

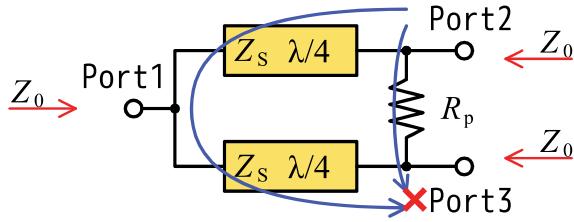


図 9: ウィルキンソン分配回路

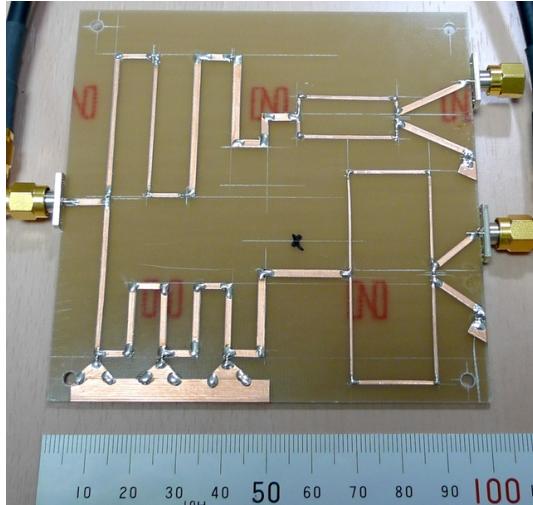


図 10: BS 地デジ分波分配器のパターン

5 製作と評価

4 節の設計をもとに銅箔テープと生基板で製作した回路を図 10 に示します。上半分が HPF、下半分が LPF になっています。基板パターン作成の際には、簡易定盤とハイトゲージによるけがきが非常に役に立ちました。同じように銅箔テープで正確なパターンを引く予定のある方は試してみてください。

続いて、図 11 に VNWA3E で測定した結果を示します。なお、破線はシミュレーション結果を表しています。測定の結果、いちおう BS 地デジ分波分配器として機能していることがわかります。また、挿入損失もぎりぎり-6dB 以内を確保することができました。

最後に製作した基板を入れるためのアルミケース



図 12: 板金加工により製作したアルミケース

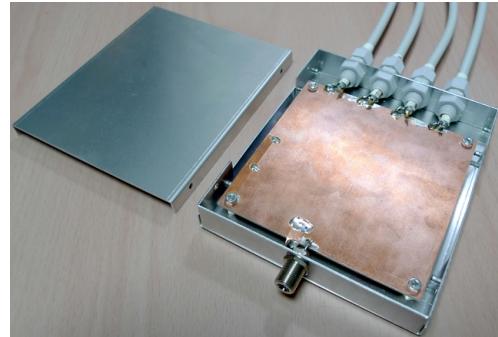


図 13: 基板をアルミケースに組み込んだ状態

の製作を行いました。ケースは板金加工により図 12 に示すようなパーツを製作し、リベットにより組み立てました。図 13 に組み立てたアルミケースに基板を組み込んだ状態を示します。

6 おわりに

製作した BS 地デジ分波分配器は、現在順調に稼働中です。今回は触れませんでしたが、LPF と HPF を組み合わせて分波回路を作成する場合、マッチングの調整が必要になります。製作した回路では、スマスチャートを見ながら LPF 入力部の伝送線路長と HPF 入力部のキャパシタ容量の調整を行い、マッチングを調整しました。

また、回路を製作するにあたり、実は開発参考品として市販の安価な分配器と分波器を購入して特性を測定しています。量産品というものは、なかなかよくできていますが、安いながらも 1.3GHz 以上の特性もそれほど悪くなく、回路サイズも小型です。一方、製作した回路は分布定数線路の長さが長いため、1.3GHz 以上の領域ではあまり良い特性は得られません。また機会があれば、再設計してみようかと思います。

余談ですが、3 年半ぶりに大学に戻ってきました。短期間ではありますが、工研部員の皆様、よろしくお願いします。

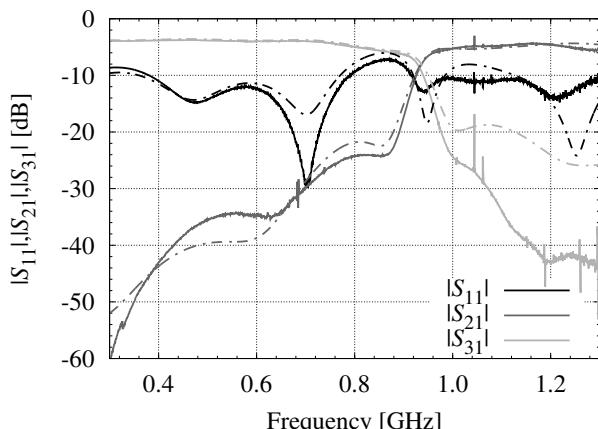


図 11: BS 地デジ分波分配器の特性

国立大学法人 電気通信大学
工学研究部 部報 第48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11月21日
執筆 工学研究部 部員

