電気通信大学

工学研究部報

76号

目次

Ubuntu で Hyprland

22 もやし

タイル型ウィンドウマネージャの Hyprland を Ubuntu 上で動かしてみました。

1　ビルドは挫折

Hyprland をソースからビルドするのは無理そうだったので下記の方法でビルドから逃げます。

1.1　方法 1

ubuntu ベースで Hyprland を採用しているディストリビューション PikaOS のリポジトリを追加することで、apt コマンドから Hyprland をインストールします。

PikaOS の git から pika-sources.deb なるパッケージをダウンロードして Ubuntu にインストールすることで、pikaOS のリポジトリが apt に登録されます。目当てのパッケージリストはおそらく.deb 内部の usr/share/apt-get/apt/source.list.d/system.sources のようですが、
/etc/apt/sources.list.d/system.sources にただコピーするだけでは公開鍵の登録がされずにうまく行きません。鍵の手動登録はよくわからないので.deb もろともインストールしました。Ubuntu24.04LTS で動作確認。

1.2　方法 2

Hyprland が Ubuntu24.10 の apt リポジトリに追加されているので、Ubuntu24.10 のベータ版を導入すると apt コマンドから Hyprland をインストールできます。

2　インストール

どちらの方法でも
sudo apt update
sudo apt install hyprland
でインストールすることができました。環境によっては GDM3 が Hyprland を認識しないことがあります。

図 1　Ubuntu24.10 ベータ版での Hyprland

3　まとめ

Ubuntu24.10 の正式リリースを待つのが良さそうです。

へるーれっと！押すだけ！

へるくん

優柔不断でよく何かしらで迷うことないですか。例えば、夕飯はマックにするかケンタッキーにするか、とか。そんなあなたに「へるーれっと」。抽選項目を追加してルーレットを回すだけで、優柔不断な状態から簡単に脱却することが可能です。

1　そうだ、web ルーレットを自作しよう

自分はよく優柔不断で、「旅行のお土産で何買うか」とか「飲食店で何を食べるか」で長時間悩んだるすることが多々ある。このような状態から脱却したいと思った私は、何かしらで悩んだ時 (進路などの人生が絡んでいる場合を除く) は web ルーレットを使用して意志決定するようになった。

ある日、お酒を飲んだ際に「自分が使いやすいような web ルーレットでも作ってみるか」と思ったので自作することになり、web ルーレットであるへるーれっとが誕生した。

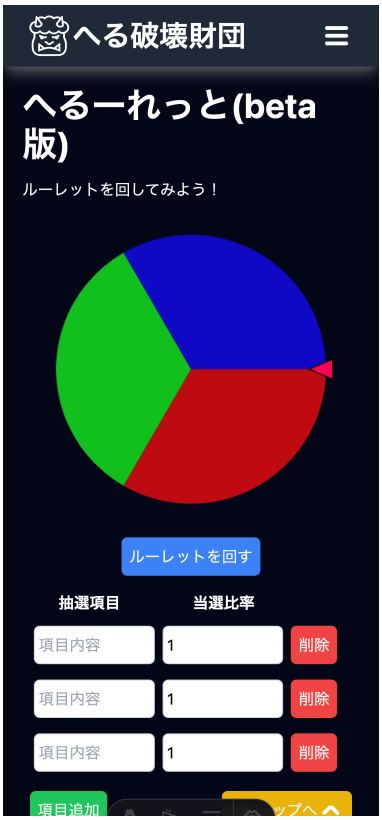


図 1　へるーれっと画面

2　完成図

実際に触ってみたい方は `https://helkun.dev/works/webRoulette` にアクセスしてほしい。

項目は項目追加ボタンを押すことで追加できる。また、各項目ごとに項目内容と、確率比を設定することが可能となっている。「ルーレットを回す」ボタンでルーレットを

回すことができる。

3　使用技術

へるーれっとは Vue を使用している。Astro 製の個人サイトの中に Vue のコンポーネントとして web ルーレットを製作した (今思えば別のプロジェクトとしてリポジトリを完全に分けてしまっても良かったかもしれない...)。ルーレットの円は canvas を使用している。ルーレットが回るアニメーションは requestAnimationFrame を使用。ルーレットの回転角を 3600 + Math.random() * 360 とすることで、ランダムに決まるようになっている。ルーレットが止まった段階でどのくらいの角度回転したかを元に結果が表示されるようになっている。抽選結果を表示するポップアップ画面も Vue コンポーネントとして製作し、使用している。

4　今後

バグを発見次第直していく予定。また、項目内容入力後に Enter キーを押すことで新たな項目を追加できるようにしていく予定である。

5　ソースコード

https://github.com/hel-kun/helkun.dev/tree/main/src/components/app/Roulette に掲載。部報のページ数かさ増しのため、ここにも掲載する。

Listing 1　Roulette.vue

```
<template>
    <h1 class="font-bold text-4xl mbへるーれっと-4">(版 beta)</h1>
    <p class="mbルーレットを回してみよう！-4"></p>

    <div class="m-4">
        <canvas ref="canvas" width="500" height="500"></canvas>
    </div>

    <PopupCard ref="popup" />

    <button class="spinButton" type="button" @click="spin" :disabled="
        isSpinning" ルーレットを回す
        ></button>

    <div class="tableTitle">
        <p抽選項目></p>
        <p当選比率></p>
```

```
16      </div>
17      <form class="label">
18          <ul>
19              <li v-for="(entry, index) in items" :key="index">
20                  <input class="text-black" v-model="entry.label" @blur="
                        contentEvent" type="text" placeholder項目内容
                        ="" :disabled="isSpinning" />
21                  <input class="text-black" v-model.number="entry.rate"
                        @blur="rateEvent(index) " type="number" placeholder比
                        率="" :disabled="isSpinning" />
22                  <button class="deleteButton" type="button" @click="
                        removeItem(index)" :disabled="isSpinning" 削除
                        ></button>
23              </li>
24          </ul>
25      </form>
26
27      <div class="flex justify-between">
28          <button class="addButton" type="button" @click="addItem" :
                disabled="isSpinning" 項目追加
                ></button>
29          <button class="scrollButton" type="button" @click="moveToTop">
30              <div class="flex flex-col justify-center items-center px-2">
31                  <span class="bar1"></span>
32                  <span class="bar2"></span>
33              </div>
34              <pトップへ></p>
35              <div class="flex flex-col justify-center items-center px-2">
36                  <span class="bar1"></span>
37                  <span class="bar2"></span>
38              </div>
39          </button>
40      </div>
41  </template>
42
43  <script>
44  import PopupCard from '@/components/app/Roulette/PopupCard.vue';
45  export default {
46      components: {
```

```
47          PopupCard
48      },
49      data() {
50          return {
51              items: [{ label: '', rate: 1 }],
52              currentRotation: 0,
53              spinDuration: 3000,
54              isSpinning: false,
55          };
56      },
57      mounted() {
58          this.drawCanvas();
59      },
60      methods: {
61          addItem() {
62              console.log('Add item!');
63              this.items.push({ label: '', rate: 1 });
64              this.drawCanvas();
65          },
66          contentEvent() {
67              this.drawCanvas();
68          },
69          rateEvent(index) {
70              //が数字じゃないもしくは負の時の処理 rate
71              if(typeof this.items[index].rate != 'number' || this.items[
                    index].rate<0 ){
72                  this.items[index].rate=1;
73              }
74              // 全てのがの時の処理 rate0
75              let totalRate = this.items.reduce((sum, item) => sum + item.
                    rate, 0);
76              if(totalRate === 0){
77                  this.items[index].rate=1;
78              }
79              this.drawCanvas();
80          },
81          removeItem(index) {
82              if(this.items.length === 1){
```

```
83                 window.alert最後の項目は削除できません。('');
84             }else{
85                 console.log('Remove item!');
86                 this.items.splice(index, 1);
87                 this.drawCanvas();
88             }
89         },
90         moveToTop(){
91             window.scrollTo({
92                 top: 0,
93                 behavior: 'smooth'
94             });
95         },
96         spin() {
97             this.isSpinning = true;
98
99             const spinToAngle = 3600 + Math.random() * 360;
100            const startTime = Date.now();
101            const duration = this.spinDuration;
102
103            const animate = () => {
104                const elapsedTime = Date.now() - startTime;
105                if (elapsedTime < duration) {
106                    const easeOut = 1 - ((1 - elapsedTime / duration) **
                         3); 少しずつ遅くする
                        //
107                    this.currentRotation = easeOut * spinToAngle; //をかけ
                        ることで遅くなる easeOut
108                    this.drawCanvas(this.currentRotation); //を描画 canvas
109                    requestAnimationFrame(animate);
110                } else {
111                    this.currentRotation = spinToAngle; 最終的な角度に設定
                        //
112                    this.drawCanvas(this.currentRotation); //を描画 canvas
113                    this.selectItem(this.currentRotation % 360); 選ばれた
                        項目を表示//
114                    this.isSpinning = false;
115                }
116            };
```

```
117
118             requestAnimationFrame(animate);
119         },
120         selectItem(finalAngle) {
121             console.log(finalAngle);
122
123             let totalRate = this.items.reduce((sum, item) => sum + item.
                    rate, 0);
124             let startAngle = -finalAngle;
125             let pointAngle = 0;
126
127             for (let item of this.items) {
128                 let sliceAngle = (item.rate / totalRate) * 360;
129                 if (startAngle < pointAngle && pointAngle <= startAngle +
                        sliceAngle) {
130                     this.$refs.popup.show(item.label);
131                     break;
132                 }
133                 startAngle += sliceAngle;
134             }
135         },
136         drawCanvas(rotation = 0) {
137             let canvas = this.$refs.canvas;
138             let ctx = canvas.getContext('2d');
139             ctx.clearRect(0, 0, canvas.width, canvas.height);
140             let centerX = canvas.width / 2;
141             let centerY = canvas.height / 2;
142             let radius = Math.min(centerX, centerY) - 30;
143             let totalRate = this.items.reduce((sum, item) => sum + item.
                    rate, 0);
144             let startAngle = -rotation * Math.PI / 180;
145
146             this.items.forEach((item, index) => {
147                 if(item.rate === 0) return;
148
149                 let sliceAngle = (item.rate / totalRate) * Math.PI * 2;
150                 let color = `hsla(${360 * (index / this.items.length)},
                        100%, 50%, 75%)`;円
```

```
                弧
151
152             //
153             ctx.beginPath();
154             ctx.fillStyle = color;
155             ctx.moveTo(centerX, centerY);
156             ctx.arc(centerX, centerY, radius, startAngle, startAngle +
                    sliceAngle);
157             ctx.closePath();
158             ctx.fill();
159
160             // ラベル
161             ctx.font = totalRate > 12 ? "18px Arial" : "28px Arial";
162             ctx.textAlign = "center";
163             let textAngle = startAngle + sliceAngle / 2;
164             let textX = centerX + (radius / 2) * Math.cos(textAngle);
165             let textY = centerY + (radius / 2) * Math.sin(textAngle);
166             ctx.save();
167             ctx.translate(textX, textY);
168             ctx.rotate(textAngle);
169             ctx.fillStyle = "#fff";
170             ctx.strokeStyle = "#000";
171             ctx.lineWidth = 6;
172             ctx.strokeText(item.label, 0, 0, canvas.width /2 - 40);
173             ctx.fillText(item.label, 0, 0, canvas.width /2 - 40);
174             ctx.restore();
175
176             startAngle += sliceAngle;
177         });
178
179         // 三角矢印
180         ctx.beginPath();
181         ctx.moveTo(centerX + radius + 10, centerY - 15);
182         ctx.lineTo(centerX + radius + 10, centerY + 15);
183         ctx.lineTo(centerX + radius - 25, centerY);
184         ctx.closePath();
185         ctx.lineWidth = 4;
186         ctx.stroke();
```

```
            ctx.fillStyle = "#ff0055";
            ctx.fill();
        }
    },
}
</script>
```

Listing 2　PopupCard.vue

```
<template>
    <div v-if="visible" class="popup">
      <div class="content" >
        <p> 選ばれたのは...</p>
        <h2 class="message">{{ message }}</h2>
        <button class="closeButton" @click="hide閉じる"></button>
      </div>
    </div>
</template>

<script>
export default {
    data() {
        return {
            message: '',
            visible: false,
        };
    },
    methods: {
        show(message) {
            this.message = message;
            this.visible = true;
        },
        hide() {
            this.visible = false;
        }
    }
};
</script>
```

四足歩行ロボットを完成させた

23 わらび

2024/10/2

前回のサーボ用ブラケットの派生版

1　概要

前回の部報において四足歩行ロボットの為に、sg90 サーボモーター用のブラケットを作成した旨を記した。この記事ではその続きとして四足歩行ロボットを作成した。(作成にあたる動機は前回の部報を参照)

2　用意したもの

主な構成要素を以下に記す。

- サーボモーター装着ブラケット八個
- 72 × 95mm 基盤
- esp32 devkitv1
- 市販のモバイルバッテリー
- 電池ケース単三電池 3 本用× 2
- マスキングテープ、輪ゴム
- 0.15m ケーブル (A to MicroB)

3　作成経緯

3d プリンターでボディを作成した。これにサーボブラケットを装着し、ボディとなる部分を作った。

ボディの上部にモバイルバッテリーと電池

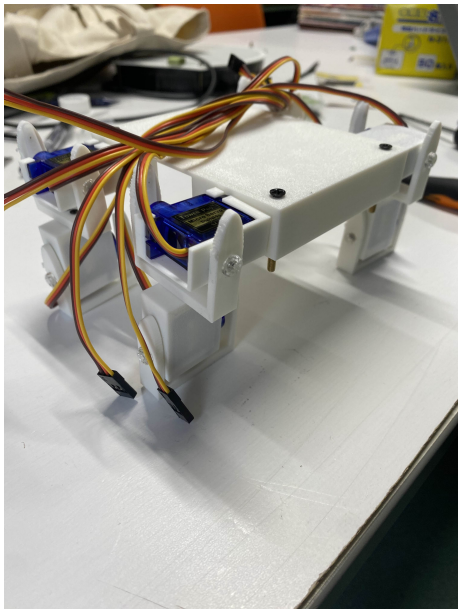


図 1　ボディ

ボックス二つを載せて輪ゴムとマスキングテープで固定し、下部にサーボに流す電源と通信の回路と esp32 を接続する基盤を取り付けた。

モバイルバッテリーからケーブルで esp32 を給電し、それぞれサーボの角度を指定したところ、垂直、平行な角度よりも若干のズレ

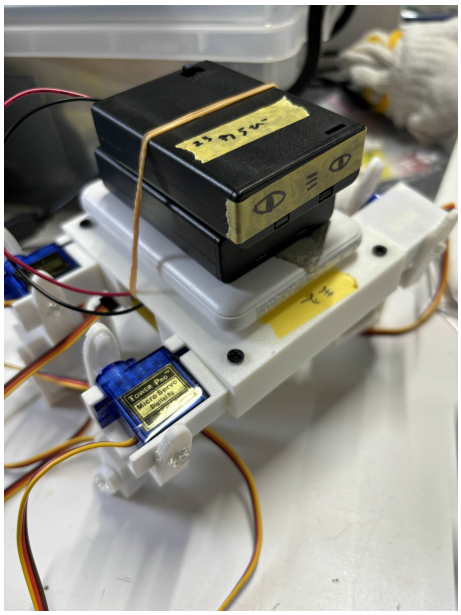


図 2　完成品

があった。すべてのサーボの動きに問題はなかったため、適当な電気が流れていることが認められた。地面に近い側のサーボによる足の移動距離がボディ側よりも短かった。

4　考察・展望

基盤や電池の取り外し等を可能とするデザインとしたため、今後のデバッグや取り換え等の作業の簡易化ができたと考えられる。サーボの数値と垂直平行の一致作業はサーボホーンの取付時に正確にならないことがあらかじめわかっていたため、今後は歩行するコードの前に角度指定を行うことが優先になる。
移動距離が短くなっている対象のサーボを機能すべく、足の取り換えが必要だが、電池ボックスやモバイルバッテリーといったロボットの重量の大部分を占めるものが上部にあるため、硬さや重量バランスを考慮する必要がある。

5　感想、余談

これを工研コンテストで発表したところ、デバッグできているで賞としてラバーダッグを頂きました!!!!!!!!　これからもデバッグしつつ進捗を生みたいです!!(その後講義の課題でケアレスミスしました.....)

アームクレーンゲーム開発進捗報告 ~MONAKO-KEN に向けて~

Hogaraka, Donn, Warabi

工学研究部では今年、調布祭で MONAKO-KEN を行う企画をしています。そこで私たちは MONAKO-KEN に向けて何かつくりたいと考えていたので、クレーンゲームを開発することに。しかし、クレーンと言ったらクレーン車とかのクレーンもありますよね。なので釣り下げ式ではなく、フルのアームで作ったクレーンゲームがあれば面白いのでは？と思い、今回の作品を作るに至りました。なおデモンストレーションは工学研究部の Youtube から見られるのでぜひ見て行ってください。

1　全体の構想と設計

発案当初の設計予定ではアーム部分の地面の平行化と機動、クレーンの爪、地面に平行な面での回転を、サーボモーターを用いて構成する予定でした。しかし、使用するサーボモーターが全回転することを考慮するため、可変抵抗によるポテンショメータを取り付け、更にアームが地面に押し込んだことを感知するスイッチも取り付け、今の構成に至りました。現在は SG90 や FS90、FS90R などの小型サーボモーターを使用する方針ですが、トルクの大きさや耐久性の問題から別のサーボモーターを使用するかもしれません。

2　機構に関する開発とこだわり

2.1　減速機の実装によるコストカット

普通、高いトルクを出すには高いサーボモーターが要るのですが、今回は減速機を用いることでモーターのトルクを増幅させ、低予算での作成を試みました。そしてギアを Fusion360 という 3DCAD ソフトでモデル設計しました。ちなみにこのギアは Fusion360 内のアドインである SperGear を用いることで簡単に作成することができます。

2.2　ウォームギア機構の作成

先のギアと異なり、ウォームギアにはアドインがなかったので一から CAD を操作して作成しました。中央のウォーム部分が回転することによって側面の白色のウォームホイールが回ります。ウォームギア機構は、摩擦の関係でウォームホイールからウォームへ回転が伝わらないセルフロック効果を持ちます。そのため、掴んだものを簡単に離してしまうようでは困るこのハンド部分にウォームギア機構を採用しました。ハンド部分に図のようにスイッチをつけて, ハンド部分が接地したのを感知できるようにしました。スイッチは図のようにカバーの裏に付いてあって、ハンド部分が地面についてこれ以上下がらなくなるとスイッチが押されるようにしました。スイッチが押されると電流が流れるようになっていて、これを

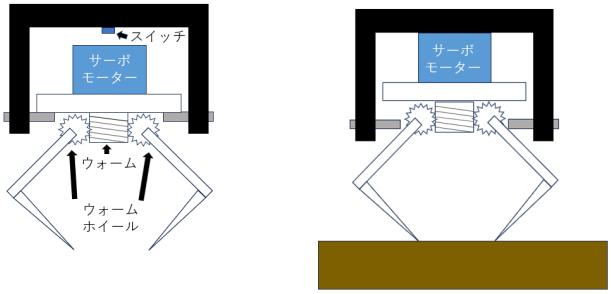


図 1: ハンド部分のウォームギアとスイッチ

ESP32 で読み取るようにしています。

2.3 平行クランク機構

平行クランク機構を採用することで先頭のハンドの部分が平行に移動させることができるようになりました。これによってハンドの部分が変な向きに移動することなく操作することができます。

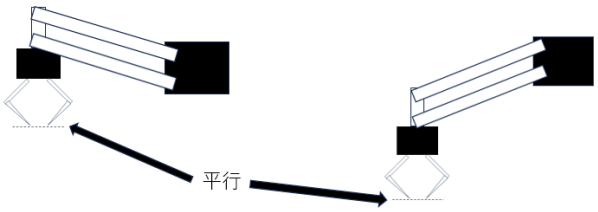


図 2: 平行クランク機構

3 プログラムの説明とこだわり

github にて、プログラムを公開していますので、よかったらご覧ください。

3.1 ステートマシン

左の図がクレーン全体の状態管理図です。また中央と右の図は、各ステートの内部のフローチャートです。今回用いたステートマシンの考えでは、まず各ステートごとに入る条件を設定します。それとは別にステート内部の条件処理を設定します。これにより、普通にひたすら条件分岐でプログラムするよりメンテナンス性が上がります。次に各ステートについて。Cont ステートでは Blynk からの入力を受け取ってステートを移動させたり、サーボモーターを回転させたりします。Auto ステートでは Blynk からのスイッチの入力を受け取り、自動で地面についたときにハンドを掴んでから、ゴールの座標上でハンドを離します。

図 3: プログラム (これを読み取ると GitHub に行けるので、そこでプログラムを見ることができます。)

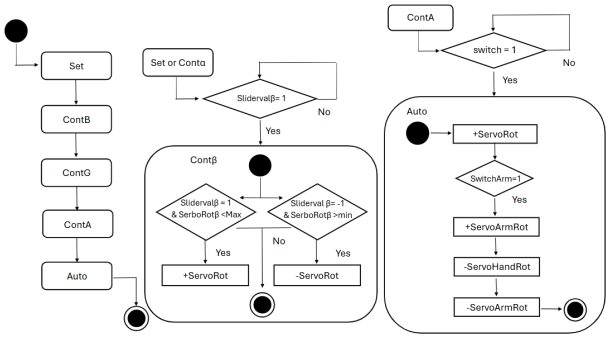


図 4: プログラムのフローチャート

4　今後の展望

4.1　ポテンショメータ

さらにコストを抑えるため、360 度回転サーボモーターをアーム部分の 2 箇所に採用しています。そのため、現状、一定時間経過するか UI でアームを動かすスイッチを切らない限りサーボモーターが動き続けてしまいます。構造上アームの動く範囲は結局限られてはいますが、それだと広すぎるのとギアへの負荷が大きいので何とかしたいところです。そこで、可変抵抗を付けて可動域を任意に調整できるようにしようとしています。可変抵抗のアナログ信号を図のような回路でデジタル化して ESP32 で読み取るようにする予定です。

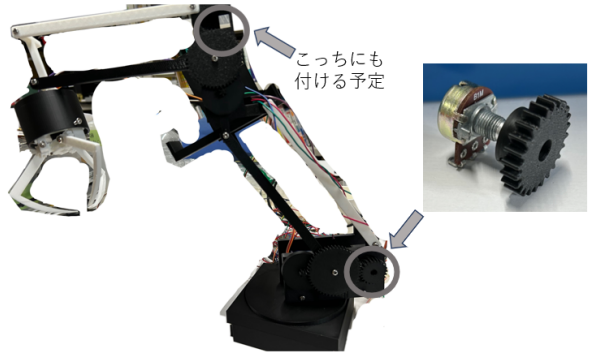


図 5: ポテンショメータの位置

4.1.1　回路

可変抵抗にギアを付けてアームと連動させて、アームの動きと可変抵抗の 2 番端子にかかる電圧を紐づけています。また、同じ電源から別口で電流を引っ張ってきて抵抗を適当につけて電圧を調整しておきます。この、2 番端子にかかる電圧と抵抗で調整済みの電圧をコンパレータで比較するようにしています。つまり後者は、前者のポテンショメータの電圧を比較する際の基準ということです。アームの動きと 2 番端子にかかる電圧が紐づいているので、アームの動きは当然電圧で監視することになります。よって電圧の閾値を設定する必要がありますが、高電圧側の閾値はコンパレータの (+) をポテンショメータの (入力) 電圧にし (-) 側を調整済みの基準電圧にすることで設定しています。低電圧側はその逆です。もちろん、ESP32 でアナログ信号を読み取ることもできて、図のような回路を実装せずプログラムコードで電圧の範囲を設定するのも良いのかもしれません。しかし、実装後の微調整のしやすさやプログラムコードが少し簡単になることなどから回路がある方を採用する方向で制作しています。

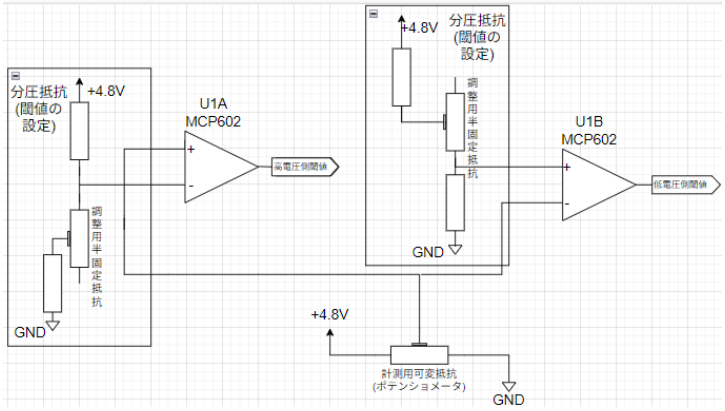


図 6: 回路の概略図

参考文献

[a] 日本産業規格,「JIS B 1723:2014 円筒ウォームギアの寸法」,https://kikakurui.com/b1/B1723-2014-01.html (2024/10/3 閲覧)

N100 マザーボードをセミファンレス運用してみた

22 あづみ

低消費電力かつ必要十分な性能を持つ CPU として、Intel N100 が有名である。主にミニ PC に搭載されているが、ノート PC やデスクトップ PC 用マザーボードにも採用例がある。今回は、ファイルサーバ用の PC として N100 マザーボードを導入し、セミファンレス運用を試みてみた。

1. 環境

ファイルサーバ PC 用に ASRock N100DC-ITX を導入した。N100 をオンボード搭載した Mini-ITX マザーボードで、この手のものでは珍しく SATA ポートを 2 口備えている。N100 の TDP は 6W と非常に小さく、N100DC-ITX ではファンレス運用を想定してCPU クーラーの代わりに大きめのヒートシンクが搭載されている。ケースは ATX のメーカー製 PC のものを流用し、背面と HDD ベイの直後にそれぞれ 120 mm ファンを搭載している。

この PC は自室に設置するが、一般の家庭なので空調をつけっぱなしにするわけにはいかず、さらに無駄に日当たりがいいため、自分が外出中の夏の昼間は人間が存在できない程度の高温になる。さすがに 40 ℃ 近くある空間でのファンレス運用は厳しいため、必要なときにのみファンを回すセミファンレス運用を試みる。

2. ファンの制御方式

PC ファンの制御方式には大きく分けて電圧制御と PWM 制御の 2 種類ある。電圧制御はそのままの意味で、電源電圧を変えることでファンの回転数を制御する。一方、PWM 制御では、電源は常に 12 V を与え、それとは別に PWM 信号を送ることで、ファン側で回転数が制御される。

PWM 制御はほとんどすべてのマザーボードで利用できるが、ファン側でも対応している必要がある。電圧制御は PWM 制御に対応していないファン(3 ピンファン)でも利用できるが、マザーボードによっては対応していないことがある。また、制御するうえで注意しなければならない点がいくつかある。

基本的には PWM 制御のほうが扱いやすいが、セミファンレス運用をしたいとなると話が変わってくる。PWM 制御では 30% 未満での挙動は「30% での回転数以下」としか規定されておらず[1]、ほとんどの場合は 30% での速度を維持する。つまり、完

全停止まで持っていくことはできない。しかし、電圧制御であれば、電源供給を断ってしまい強制的にファンを止めることができる。

3. 制御の設定

3.1. 制御方式の設定

幸いなことに、N100DC-ITX は電圧制御に対応している。UEFI の設定画面で図 1 のように「DC Mode」を選択することで電圧制御となる。回転数の制御も UEFI 上から行うことができるが、今回は OS 側で制御したいため、こっちの設定は最大速度固定にしておいた。

図 1: UEFI の設定画面

ちなみに、PWM 制御にしか対応していないマザーボードでも、PWM 信号によって電源をスイッチングする簡単な回路を組めば、電圧制御と同じようなことができる[2]。

3.2. 制御アプリの設定

設定ができたら、Windows 上に制御アプリをインストールする。今回は Fan Control を採用した。(あまり一般名詞を固有名詞として使うのは良くないと思うのだが……。)winget に登録されているため、`winget install FanControl` でインストールできる。初回起動時に指示値とファンの回転数を対応させるキャリブレーションが提案されるが、時間がかかるわりにあまり有益でないのでスキップして問題ない。どの出力がどの回転数入力に対応しているかのみ設定すればよい。

電圧制御特有の設定として、「Start %」と「Stop %」がある。Start % は止まっているファンが始動するために必要な出力で、Stop % は回っているファンが回転を維持できずに停止してしまう出力である。たとえば、始動には 50% の電圧が必要だが、30% までは下げても回転を維持できるというファンが存在するとき、それぞれ設定しておくと、30-50%の範囲の指示がされたときに 50% を 2 秒間出力して始動させたあと、指定された出力で運転を続けるという動作をしてくれる。また、30% 以下の指示のときには 0% に切り捨てることで、ロックしたモータに電流を流して損傷させることを防いでくれる。なお、PWM 制御ではこのような操作をファン側で行うよう定められている[1]。

手動設定を有効にして、回転数を見ながら出力を上げたり下げたりすることで、このような出力値を見つけて設定していく。Start % に関してはとりあえず100%としておいても問題はない。本来はこの値も自動キャリブレーションできるはずなのだが、あまり正しい値にならなかった。

3.3. 制御内容の設定

以上の設定でファンを回したり止めたりすることができるようになったため、あとはいつどのくらい回すか回さないかを設定する。制御の基準は無難に CPU 温度とする。まず、空調が効いていてファンを回していないときのCPU 温度を確認する。負荷が平常なときには 45 °C から 50 °C 程度、CPU 負荷を 100% にしているときの温度は 70 °C 前後であった。これをもとに、50 °C で 0%、80 °C で 100% という単純な制御カーブ(直線だが?)を図 2 のように作成した。

図 2: ファン制御曲線

このままでは空調が効いていても負荷がかかるとすぐにファンが回ってしまうため、応答時間を 60 秒と長めに設定し、また、温度の低下時のみヒステリシスを適用する設定を外すことで、自然放熱を待つようにした。さらに、最大温度を 80 °C に設定し、最大値と最小値でヒステリシスを無視する設定を有効にすることで、高温になりすぎたときには即座にファンを回すようにした。

制御対象はケースファンのほか、HDD の背後のファンも CPU 温度に連動して回すようにしている。本当は HDD 温度に連動させたいところだが、HDD の温度を取得していると HDD がアイドル時にもスリープしてくれず、そのせいで騒音と発熱が増えるという本末転倒な状態になってしまうためである。アクセスがあるときのみの稼働なら灼熱の部屋でも過熱することはないようなので、ひとまずこれで妥協しておく。

4. スタートアップ設定

Windows の起動とともにファン制御を開始してほしいので、Fan Control をスタートアップに登録する。アプリ内の設定から行えるが、電源を入れてもログインせずに放置するサーバ運用の PC においては、これだけでは不十分。このスタートアップは「任意のユーザーのログオン時」にタスクスケジューラによって実行されるようになっているため、ログインしなければ実行されない。`netplwiz` などを用いて自動ログインを設定しておく。また、ログインしたままではあまり気持ちよくないので、`rundll32.exe user32.dll, LockWorkStation` へのショートカットをスタートアップフォルダ(`shell:startup`)に置いて自動でロックされるようにしておく。なお、タスクスケジューラでは「スタートアップ時」をトリガーにすることもできるが、この手の GUI

アプリはログイン後でないと正常に実行できないので使えない。

5. N100 はいいぞ

N100DC-ITX 導入前は Sandy Bridge-E のメーカー製 PC をファイルサーバとしていたが、電源ファンも CPU ファンもケースファンも爆音を発していたのでとても自室でつけっぱなしにすることができず、HDD にアクセスしたいときのみ電源をつける面倒な運用をしていた。N100 マザボに置き換えてセミファンレス化することで、平時は無音で常時起動できるすばらしい PC になった。ちなみに、N100DC-ITX の電源は DC ジャックで、ELECOM の PD トリガーケーブルで 65 W の PD 充電器から給電している。当然音はしない。

N100 はこのようにファンレス運用できる低消費電力の CPU で、性能も個人的なファイルサーバ程度であれば余裕である。さらに、軽いブラウジングも試してみたところ、これもストレスなく行うことができ、サーマルスロットルングが起きるほど発熱することもなかった。さすがにゲームをしようと思うと厳しいが、うるさく熱くせずに軽い作業をしたいのであれば、かなりよい選択肢といえる。SATA や PCIeなどの拡張性を求めないのであればミニ PC がちょうどいいかもしれない。工研でも共有 PC として導入するとかなんとか。

6. 今後

N100DC-ITX は部報のネタにすることを見込んで導入したものではあるが、当初はファン制御について書く予定ではなかった。Mini-ITX マザーボードは 170 mm 四方とコンパクトであり、N100DC-ITX では電源装置も要らないため、3D プリンタ(工研の P1S では 256 mm 立方刷れる)でケースを自作しようと考えていた。欲張って 120 mm ファンを CPU 用と HDD 用とで 2 つ横並びに配置しようなどと試みたが、16 mm しか残らない余白ではなにもできず挫折した。とりあえず ATX ケースにぶちこんで運用してみた結果生まれたネタが今回の記事である。

実際に運用してみて、他が静かになると HDD の回転音が意外とやかましいのでスリープは続行したほうがよい、そのため HDD の個別冷却は不要、つまりファンは 1 つで十分、という知見を得られたため、ファンを 1 つにしてケースの設計を再度試みてみるのもいいかもしれないと思う。HDD2 台をねじこむだけでも大変だが……。

参照文献

[1] https://www.intel.com/content/dam/support/us/en/documents/intel-nuc/intel-4wire-pwm-fans-specs.pdf

[2] https://azumino.pages.dev/blog/20240010-6pin-fan-pwm/

ジャンクタブレット PC のバッテリー交換をした

22 ゆい

2024 年 10 月 6 日

1 はじめに

22 のゆいです。今回はジャンクで入手した Windows タブレット PC のバッテリー交換をしたのでその話を書きます。

2 経緯

2023 年 12 月に秋葉原にある PC ショップ、イオシスアキバ路地裏店でジャンクの Windows タブレットを購入しました。購入したのは Arrows Q736/P というモデルで、13.3 型のタブレットとしては割と大きめの液晶、CPU は Core i5-5200U、メモリは 4GB、ストレージは 128GB の SSD を搭載しています。価格は 5,000 円でした。その場で軽く動作確認して破損の少ない個体を選んだつもりでしたが、家に持ち帰って AC アダプタに接続してみると 15% までしか充電できませんでした。バッテリーのリフレッシュをしても変わらなかったのでバッテリーが劣化していると判断し、バッテリーを交換することにしました。

3 バッテリー購入

有名な機種であれば交換用のバッテリーは Amazon やその他通販サイトで購入できます。この機種のバッテリーも Amazon に 8,000〜15,000 円程度で販売されています。Aliexpress には 4,500 円程度のものがあったので興味本位で Aliexpress で購入しました。購入日は 2024 年 5 月 6 日、到着日は 5 月 29 日で届くのに 1 か月くらいかかりました。空輸できないからかなぁ。早くほしい人はちょっと高いけど Amazon での購入をおすすめします。
商品 URL: `https://ja.aliexpress.com/item/1005004476332740.html`

4 バッテリー交換

届いたバッテリーを組み込みます。この機種は防水使用なので、裏蓋を開けると防水フィルムで基板が保護されています。自分が購入したものは USB の防水フタがなかったので防水はあきらめて、フィルムを剥がしてバッテリーを交換しました。フィルムはピンセットでなるべく丁寧に剥がしました。バッテリーの交換自体は非常に簡単で、バッテリーを固定しているねじと端子を外すだけです。iPhone のように接着剤が使われたりはしていませんでした。細かい作業に慣れている人なら 10 分程度で交換できると思います。

図 1　バッテリー本体

図 2　バッテリー交換後の本体

5　結果

作業は無事に完了し、100% まで充電できるようになりました。ただ、現在の FullChargeCapacity が確認できなくなりました。使えているからまあいいやってことにしておきます。ちなみにバッテリーの情報は Windows の PowerShell を開いて、以下のコマンドを実行して確認できます。

```
powercfg /batteryreport
```

これでバッテリーの状態を正確に判定するのは難しいです、実際、交換前のバッテリーは FullChargeCapacity が 8 割ほど残っていましたが壊れていたので…

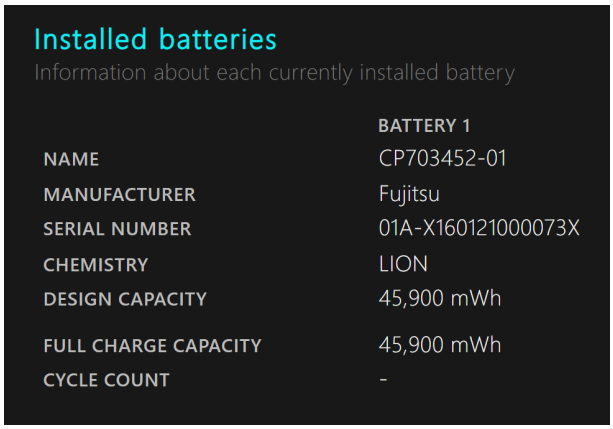


図 3　バッテリーの情報

6　おわりに

最後まで読んでいただきありがとうございました。このタブレット、授業のノートを取ったり資料閲覧に使っていますが、まだまだ使えそうです。高くて iPad に手が出ないというそこのあなた、Windows タブレットはいかがでしょうか。1 万円で大きめ画面とペン、タッチが使えるデバイスを入手できますよ。ノート PC やタブレット、スマホのジャンクは安いですがバッテリーが劣化しておりまともに使えないことも多いです。バッテリー交換すると古い端末でもまだまだ活躍できるかもしれません。ぜひお試しください。

M5stickcplus でスイカゲームをつくる 後編

22 しぐれ

1 はじめに

去年 (2023 年) の秋に行われた工研コンテスト用に、M5stickcplus でスイカゲームをつくろうと思い、始めました。本家のものと同じように、同じ大きさのものがくっつくと消え、スコアが加算されるところまで実装できました！ しかし完成してから半年ほど経ってからこの記事を書いているためだいぶうろ覚えです。そのため解説的なのはかけませんでした。土下座。

2 メインコード

```
#include <Arduino.h>
#include <M5StickCPlus.h>
#include <box2d/box2d.h>
#include "TFTDebugDraw.h"
#include "b2_setting.h"
#include <list>
#include <algorithm>
#include <unordered_map>
#include <unordered_set>
#include "efont.h"
#include "efontM5StickCPlus.h"
#include "efontEnableJa.h"


TFT_eSprite tft = TFT_eSprite(&M5.Lcd);
TFT_eSprite sp(&tft);
b2World *myWorld;
unsigned long lastMs = 0;


bool btn = false; // ボタンAが押されたかフラグ
bool btnb = false;
bool creatnewBall = false;

double newx;
double newy;
double newr;

float ax;
float ay;
float az;

float tilt_x;
float value;
float tiltbox2d_x;
float ballsize;

float score = 0;
char str[256];

int randomint = 1;


struct MyData{
  int objkind; //オブジェクトの種類
  int deleteobj;
};

std::unordered_map<int,double> Fruits={//0番は壁
  {1, 0.60},
  {2, 0.80},
  {3, 1.0},
  {4, 1.3},
  {5, 1.5},
  {6, 1.7},
  {7, 2.0}
};

int mapsize = Fruits.size()- 3;

void createSomeBalls(double xp, double yp, int objkindnum) { //objkindnum Fruits連想配列の左の数字

  b2BodyDef bodyDef; //body定義
  bodyDef.type = b2_dynamicBody; //動的らしい
  bodyDef.allowSleep = true; //falseだと本体がスリープにならないらしいよ
  bodyDef.position.Set(xp, yp); //物体の初期位置らしい

  b2Body* body = myWorld->CreateBody(&bodyDef); //物体を生成
```

```
    MyData* balldata = new MyData{
        objkindnum,0};
    b2BodyUserData& userData = body->
        GetUserData();
    userData.pointer = reinterpret_cast<
        uintptr_t>(balldata);

    b2CircleShape shape; //物体の形状を円形
        にする
    shape.m_radius = Fruits[objkindnum];

    b2FixtureDef f; //物理特性や形状を設定
        するためのデータ構造
    f.shape = &shape; //形状
    f.density = 1.0; //密度
    f.friction = 0.4; //摩擦係数
    f.restitution = 0.5; //反発係数

    body->CreateFixture(&f); //上記の設定を
        追加する
    body->ApplyForce(b2Vec2(0, 0), b2Vec2
        (1, 1), true); //物体に力を適用する
        関数
    //引数１：適用する力のベクトル　引数
        ２：力を適用する点の位置（この場合
        物体の中心）　引数３：物体が非アク
        ティブな場合にアクティブに切り替え
        るらしい
}

class MyContactListener : public
    b2ContactListener{
  void BeginContact(b2Contact* contact){

    b2Fixture* fixtureA = contact->
        GetFixtureA();
    b2Fixture* fixtureB = contact->
        GetFixtureB();
    b2Body* bodyA = fixtureA->GetBody();
    b2Body* bodyB = fixtureB->GetBody();

    b2BodyUserData& userDataA = bodyA->
        GetUserData();
    b2BodyUserData& userDataB = bodyB->
        GetUserData();
    uintptr_t dataApt = userDataA.pointer
        ;//mydataのアドレス
    uintptr_t dataBpt = userDataB.pointer
        ;

    int dataAptint = static_cast<int>(
        dataApt);//ポインタをintに変換
    int dataBptint = static_cast<int>(
        dataBpt);


    MyData* dataAstruct =
        reinterpret_cast<MyData*>(dataApt
        );//構造体
    MyData* dataBstruct =
        reinterpret_cast<MyData*>(dataBpt
        );

    int dataAint = dataAstruct->objkind;
    int dataBint = dataBstruct->objkind;

    if(dataAint == dataBint && dataAint
        >= 1){ //どっちもボールだったら

      MyData* balldata = new MyData{
          dataAint,1};
      userDataA.pointer =
          reinterpret_cast<uintptr_t>(
          balldata);
      userDataB.pointer =
          reinterpret_cast<uintptr_t>(
          balldata);


      b2Vec2 positionA = bodyA->
          GetPosition();
      b2Vec2 positionB = bodyB->
          GetPosition();

      float radiusA = ( (b2CircleShape*)
          fixtureA->GetShape())->m_radius
          ;
      float radiusB = ( (b2CircleShape*)
          fixtureB->GetShape())->m_radius
          ;


      b2Vec2 newPosition = (positionA +
          positionB);
      newPosition *= 0.5;

      newx = newPosition.x;
      newy = newPosition.y;
      newr = dataAint +1;

      creatnewBall = true;

      myWorld->SetAllowSleeping(true);
    }


  }
};


void createSomeWorld() {
  MyData* worlddata = new MyData{0,0};
  b2Vec2 gravity(0, 15); //y軸方向には重
      力が10かかっているということ
  myWorld = new b2World(gravity); //
      b2worldはgravityを受けるよてことら
      しい

  b2Draw *draw = new TFTDebugDraw();
  draw->SetFlags(1);
  myWorld->SetDebugDraw(draw);

  b2BodyDef groundBodyDef;
  groundBodyDef.type = b2_dynamicBody;
  b2Body *groundBody = myWorld->
      CreateBody(&groundBodyDef); //物理
      シミュレーション内に物体をつくって
      る
  b2EdgeShape groundBox; //多角形の形状を
      設定してるらしい
  b2BodyUserData& userData = groundBody->
      GetUserData();
  userData.pointer = reinterpret_cast<
      uintptr_t>(worlddata);
```

```
  b2Vec2 hidariue = b2Vec2(1, 20);
  b2Vec2 migiue = b2Vec2(11, 20);
  b2Vec2 hidarisita = b2Vec2(1, 3);
  b2Vec2 migisita = b2Vec2(11, 3);


  groundBox.SetTwoSided(hidariue, migiue
      ); //(16,0)に幅16高さ0.05の長方形を
      定義（地面の上部）
  groundBody->CreateFixture(&groundBox,
      1.0f);//上のやつの密度

  groundBox.SetTwoSided(hidariue,
      hidarisita);
  groundBody->CreateFixture(&groundBox,
      1.0f);

  groundBox.SetTwoSided(migiue, migisita
      );
  groundBody->CreateFixture(&groundBox,
      1.0f);



}



MyContactListener myContactlistener;

void setup() {

  M5.begin();
  tft.init();
  M5.Lcd.fillScreen(BLACK);
  Serial.begin(115200);
  sp.createSprite(130, 230);
  createSomeWorld();
  myWorld->SetContactListener(&
      myContactlistener);
  myWorld->SetAllowSleeping(false);
  M5.Imu.Init();
}

void loop() {
  M5.update();
  myWorld->Step(0.05, 10, 10);
  sp.fillRect(0, 0, 135, 240, BLACK);
  myWorld->DebugDraw();
  M5.IMU.getAccelData(&ax, &ay, &az);
  tilt_x = -ax*130 + 70;
  sp.drawCircle(tilt_x, 10, Fruits[
      randomint]*10,RED);
  sprintf(str, "スコア：%f",score);
  printEfont(str, 15, 220);
  sp.pushSprite(0, 0);

  if (millis() - lastMs >= 100) {
    lastMs = millis();

    if(M5.BtnA.isPressed()){
      btn = true;
    }

    if(btn){
      tiltbox2d_x = tilt_x / 10;
      createSomeBalls(tiltbox2d_x,3.0f,
          randomint);
      btn = false;
      randomint = std::rand() % 5;
      Serial.println(value);
    }

  }

  for(b2Body* bodyA = myWorld->
      GetBodyList(); bodyA != nullptr;
      bodyA = bodyA->GetNext()){

    b2BodyUserData& userDataA = bodyA->
        GetUserData();
    uintptr_t dataApt = userDataA.pointer
        ;//mydataのアドレス
    int dataAptint = static_cast<int>(
        dataApt);//ポインタをintに変換
    MyData* dataAstruct =
        reinterpret_cast<MyData*>(dataApt
        );//構造体
    int dataAint = dataAstruct->deleteobj
        ;
    if (dataAint == 1)
    {
      myWorld->DestroyBody(bodyA);
    }

  }
  if(creatnewBall){
    createSomeBalls(newx, newy, newr);
    score += newr;
    creatnewBall = false;

  }

}
```

3 box2d の設定関係

```
#ifndef B2_SETTINGS_H
#define B2_SETTINGS_H

#include "b2_types.h"
#include "b2_api.h"


#ifdef B2_USER_SETTINGS

#include "b2_user_settings.h"

#else

#include <stdarg.h>
#include <stdint.h>

// Tunable Constants

#define b2_lengthUnitsPerMeter 1.0f

#define b2_maxPolygonVertices   8

// User data

struct B2_API b2BodyUserData
```

```
  {
      b2BodyUserData()
      {
          pointer = 0;

      }

      uintptr_t pointer;
  };

  struct B2_API b2FixtureUserData
  {
      b2FixtureUserData()
      {
          pointer = 0;
      }

      uintptr_t pointer;
  };

  struct B2_API b2JointUserData
  {
      b2JointUserData()
      {
          pointer = 0;
      }

      uintptr_t pointer;
  };

  // Memory Allocation

  B2_API void* b2Alloc_Default(int32 size);
  B2_API void b2Free_Default(void* mem);

  inline void* b2Alloc(int32 size)
  {
      return b2Alloc_Default(size);
  }

  inline void b2Free(void* mem)
  {
      b2Free_Default(mem);
  }

  B2_API void b2Log_Default(const char*
      string, va_list args);

  inline void b2Log(const char* string, ...)
  {
      va_list args;
      va_start(args, string);
      b2Log_Default(string, args);
      va_end(args);
  }

  #endif // B2_USER_SETTINGS

  #include "b2_common.h"

  #endif
```

4 デバッグ描画のライブラリ

```
  #ifndef __TFT_DEBUG_DRAW__
#define __TFT_DEBUG_DRAW__

#include <Arduino.h>
#include <M5StickCPlus.h>
#include <box2d/box2d.h>

extern TFT_eSprite sp;

class TFTDebugDraw : public b2Draw {

  void DrawPolygon(const b2Vec2 *vertices,
      int32 vertexCount,
                   const b2Color &color) {
    for (int i = 0; i < vertexCount; i++) {
      auto p1 = vertices[i];
      auto p2 = vertices[(i + 1) %
          vertexCount];
      sp.drawLine(p1.x * 10, p1.y * 10, p2.x
          * 10, p2.y * 10, TFT_GREEN);
    }
  }

  void DrawSolidPolygon(const b2Vec2 *
      vertices, int32 vertexCount,
                        const b2Color &color)
                             {
    for (int i = 0; i < vertexCount; i++) {
      auto p1 = vertices[i];
      auto p2 = vertices[(i + 1) %
          vertexCount];
      sp.drawLine(p1.x * 10, p1.y * 10, p2.x
          * 10, p2.y * 10, YELLOW);
    }
  }

  void DrawCircle(const b2Vec2 &center, float
       radius, const b2Color &color) {
    sp.drawCircle(center.x * 10, center.y *
        10, (int)(radius * 10), TFT_PINK);
  }

  void DrawSolidCircle(const b2Vec2 &center,
      float radius, const b2Vec2 &axis,
                       const b2Color &color)
                            {
    sp.drawCircle(center.x * 10, center.y *
        10, (int)(radius * 10), TFT_PINK);
  }

  void DrawSegment(const b2Vec2 &p1, const
      b2Vec2 &p2, const b2Color &color) {
    sp.drawLine(p1.x * 10, p1.y * 10, p2.x *
        10, p2.y * 10, TFT_GREEN);
  }

  void DrawTransform(const b2Transform &xf)
      {}

  void DrawPoint(const b2Vec2 &p, float size,
       const b2Color &color) {
    sp.drawPixel(p.x * 10, p.y * 10, TFT_RED
        );
  }
};

#endif
```

参考文献

[1]《合成大西瓜》重制版！（朕机版在做了）, https://juejin.cn/post/7109867733260435469

[2] Box2D C++ tutorials - Introduction, https://www.iforce2d.net/b2dtut/

CH32V203K8T6 の開発ボードを作った

21 Valkyrie

CH32V203K8T6 というマイコンがあります。今後使っていきたいので、開発ボードを作りました。

1 目的

CH32V203K8T6 という WCH 社が提供しているマイコンがあります。秋月電子で 120 円と激安であり、USB から直接書き込めるため USB シリアル変換 IC が不要です。このマイコンを使うためにブレッドボード上に USB Type-C コネクタを実装して、PD 用の抵抗を接続して、レギュレーターで 3.3V に電源電圧を落として、EN と Reset にスイッチを接続していくのは面倒になってきたので、全部プリント基板にまとめます。

2 回路

図 1 に回路図を示します。

CH32V203 に USB から書き込むためには BOOT0 を HIGH にした状態で電源を入れる必要があります[*1]。電源を入れなおさなくても NRST を LOW にしてリセットしてもいいのでそれぞれスイッチを接続しています。保護回路には短絡保護用にリセッタブルヒューズを、開発ボードに接続した回路から USB の VBUS に電流が流れ込むのを防ぐために理想

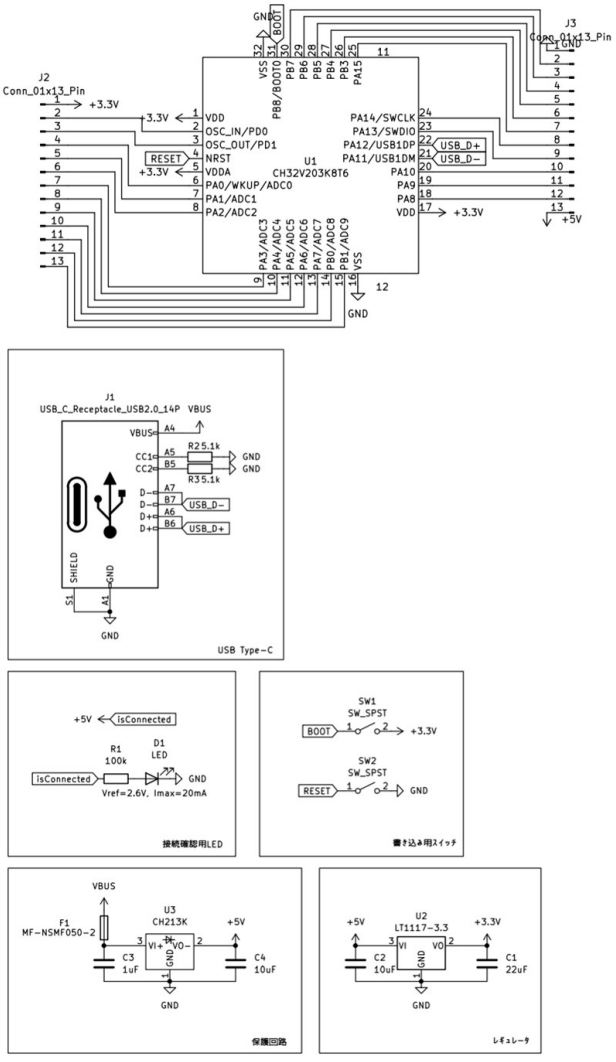


図 1 CH32V203K8T6 開発ボードの回路図

[*1] 本当は BOOT1 も LOW にしておく必要がありますが、CH32V203K8T6 は最初から BOOT1 が GND に落とされています。

ダイオードを接続しています[*2]。

3　結果

JLCPCB で発注した基板に部品を実装したものが図 2 です。

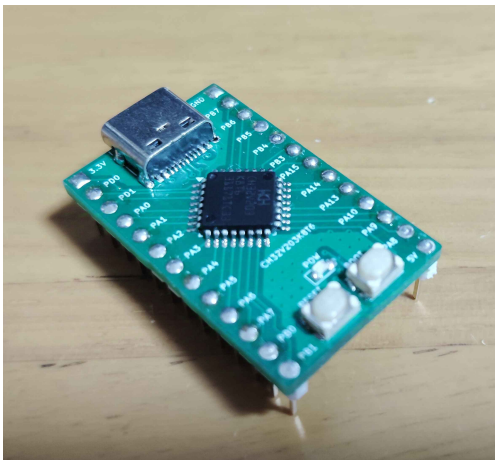


図 2　CH32V203K8T6 開発ボード

USB から書き込んでの動作が確認できました。

4　おまけ

SPI でディスプレイモジュールを繋いでライフゲームを実装しました。図 3 はその様子です。

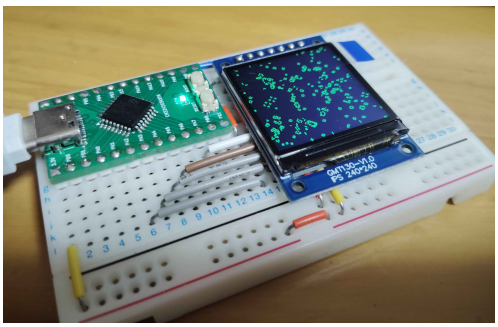


図 3　ライフゲーム動作中の様子

240×240 ピクセルのディスプレイにセル 1 つのサイズが 2×2 ピクセルなので全部で 120×120 セルのライフゲームが表示できます。120×120=14400 セルの状態を管理する変数のサイズは 1 セルあたり 1 ビットで、実装の都合上前回のセルの状態も保存するため、合計で 14400×2=28800 ビット =3600 バイト ≃3.5KB の RAM が必要です。CH32V203K8T6 の SRAM は 20KB あるので問題ありません。なんならセルサイズを 1×1 ピクセルにしてセルの数を 240×240=57600 セルに増やしても問題ありませんが、SPI での通信頻度が増えるので画面がちらつくようになります。

[*2] 別に普通のダイオードでもよかったのではないかと今は思っています。

YMF825FM 音源モジュールで東方の曲を作りたい

terraria

2024/10/7

好きな曲はエレクトリックヘリテージ

1　概要

今回は YMF825 という FM 音源モジュールを使って東方の曲を作った。今年は東風谷早苗がテーマということで、「信仰は儚き人間のために」を作った。midi を読み込めるようにしたかったが、バイナリを読んでもわからずあきらめてプログラムを手打ちした。

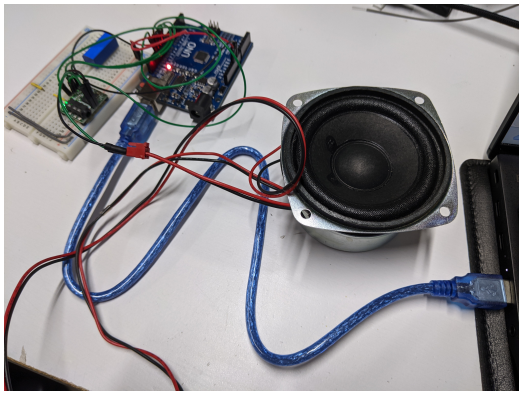


図 1　全体図

2　YMF825 について

YMF825 はヤマハが製作した、最大 16 音を出力することができる FM 音源モジュールである。具体的な特徴は SPI 通信を用いて音に関するデータを送信し、そのデータをもとにモジュールに内蔵されたアンプを通して音を出力できる。

■ ブロック図

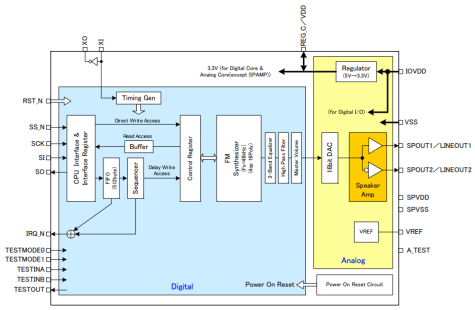


図 2　YMF825 のブロック図

モジュールの特徴 [1]

- 4 オペレーション・16 和音に対応した豊かなサウンドの生成に対応
- Arduino や Raspberry Pi と SPI で接続し簡単なコマンドで発音させることが可能
- 「YMF825」内蔵のアンプでスピーカを直接駆動可能
- ヤマハ公式 GitHub アカウントで、制御用サンプルプログラムを随時公開予定
- 3 バンドのイコライザを内蔵
- クロック同期式シリアルインタフェースを内蔵 (SPI 互換)

- 3.5mm ヘッドフォン出力
- IOVDD(5V、3.3V 切替)

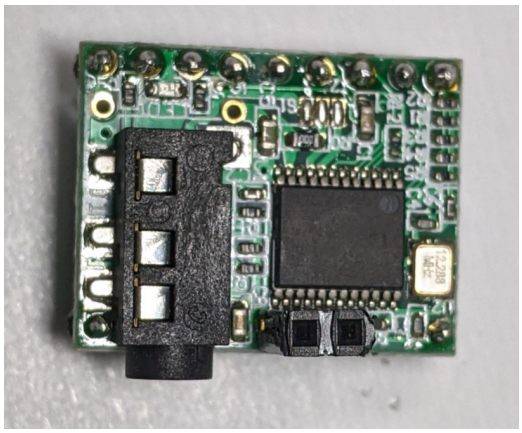


図 3　YMF825

3　プログラム

3.1　spi 通信用のプログラムを探す

spi 通信をするために arduino にプログラムを書く必要があるが、正直めんどくさい。そこで github から YMF825 のプログラムを探す。秋月の商品紹介ページによるとヤマハの github に上がっている……はずなんですが、github の存在が消えています。一応、Qiita にアンオフィシャルガイド [2] があるものの、プログラムを 1 から書くのは大変なので、lipoyang さんの github に公開されている SimpleYMF825 を用いて曲を打ち込んだ。

3.2　変更箇所

SimpleYMF825 は初期状態でドレミファソラシドを各種音源で再生できるように構成されている。また音の出力用の関数と、停止用の関数があり音の出力用の関数は
YMF825.keyon(音源, オクターブ, 音階);
停止用の関数は
YMF825.keyoff(音源);
によって定義されている。また音階は SimpleYMF825.h で定義されており、

表 1　音階のプログラム

KEY_C	0
KEY_C_SHARP	1
KEY_D	2
KEY_D_SHARP	3
KEY_E	4
KEY_F	5
KEY_F_SHARP	6
KEY_G	7
KEY_G_SHARP	8
KEY_A	9
KEY_A_SHARP	10
KEY_B	11

プログラムが長いと手打ちするときに大変なので、下記のように変更した

表 2　変更後の音階のプログラム

C	0
CS	1
D	2
DS	3
E	4
F	5
FS	6
G	7
GS	8
A	9
AS	10
B	11

半音上がるのを基準に#を S としているが半音下がるのを基準に b(フラット) を用いても良い。

4　曲の作製

とりあえず midi に耳コピした曲を書いていく。そのうえで、midi から arduino にプログラムを書く方法を考える。

4.1　バイナリファイルを読む

midi はバイナリによって構成されており、MThd から始まるファイルが midi にあたる。初めの数行は分解能やトラック数に関するデータが入っており、トラック自体や音に関するデータは MTrk から始まるデータで構成されている。これを解析することによりプログラムを生成できるが、バイナリデータを読んでもプログラムに変換するのは非常に大変で、仮に出力ができたとしてもプログラムの問題で、1 種類の音源につき、1 つの音しか出ない。そのため、midi から変換するのをあきらめて、プログラムを自力で打つことにした。実際に midi から出力するときは tone.js などの midi 解析ソフトを使うことをお勧めします。

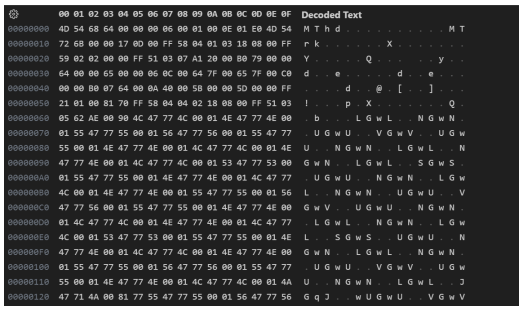


図 4　midi のバイナリデータ

4.2　作成したプログラム

基本的な構造としては、音を出す→ delay → 音を止めるという構造になっており、音の長さの最小単位を事前に設定し、その長さを変数 x でおくと bpm の調整が簡単にできる。また、1 種類の音源につき 1 音しか出せないため、複数の音源を使い、メロディーとコード進行を出力している。表 3 は各音源の特徴である。

表 3　各音源の特徴

grand piano	普通のピアノ少しノイズが入る
e piano	電子ピアノきれいな音が出る
tenor sax	うるさいノイズの温床
pick bass	音は小さめ音の出力と同時にパンッと鳴る
tnkl bell	ほぼノイズ、高音がうるさい
new age pd	tnkl bell とほぼ同じ
bright piano	grand piano との差が分からない
vibes	うるさい
church orgun	低音がうるさくて曲が分からない
frute	正弦波、非常に使いやすい
square lead	矩形波、非常に使いやすい
nylon guiter	パツパツと音が鳴りスピーカーと耳にダメージ
saw lead	のこぎり波、うるさい
harpsichode	長い時間音は出せない
harmonica	ダントツでうるさい

4.3　コード進行について

前章で何度も言っている通り、このプログラムでは、同一の音源から複数音出すことができないため、コード進行のような同一の音源から複数の音を出す必要があるものは作れない。そこで、コードの代わりとして、曲自体を工夫することでコード進行を再現している。今回は 8 分音符を基準として、コードを構成する音を連続的に出力させている。以下はその例になる[*1]

```
YMF825.keyon(channel, 3, E);
delay(x);
YMF825.keyoff(channel);
YMF825.keyon(channel, 3, G);
delay(x);
YMF825.keyoff(channel);
YMF825.keyon(channel, 4, C);
delay(x);
YMF825.keyoff(channel);
YMF825.keyon(channel, 3, E);
delay(x);
YMF825.keyoff(channel);
YMF825.keyon(channel, 3, G);
delay(x);
YMF825.keyoff(channel);
YMF825.keyon(channel, 4, C);
delay(x);
YMF825.keyoff(channel);
YMF825.keyon(channel, 3, E);
delay(x);
YMF825.keyoff(channel);
YMF825.keyon(channel, 3, G);
delay(x);
```

[*1] このプログラムのコード構成は C,E,G である

YMF825.keyoff(channel);

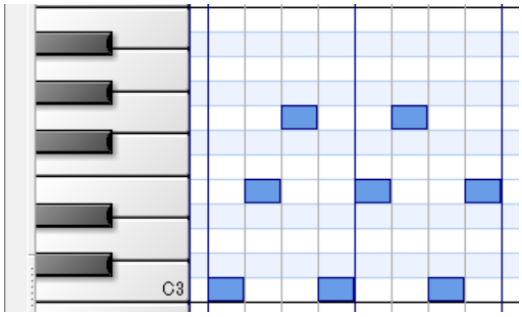


図 5　コードの打ち方の例

4.4　プログラムの全容

作製したプログラムは以下のようになった。[*2]

```
#include ”SimpleYMF825.h”
SimpleYMF825 YMF825;
int channel = 9; //音源定義
int ch = 12; //音源定義
int c = 1; //音源定義
int channel1 = 12; //音源定義
int x = 180; //bpm 調整
int y = 25; //個別音量調整
void setup()
{
//各種音源
YMF825.begin(IOVDD5V);
YMF825.setTone( 0, GRAND_PIANO );
YMF825.setTone( 1, E_PIANO );
YMF825.setTone( 2, TENOR_SAX );
YMF825.setTone( 3, PICK_BASS );
YMF825.setTone( 4, TNKL_BELL );
YMF825.setTone( 5, NEW_AGE_PD );
YMF825.setTone( 6, BRIGHT_PIANO );
YMF825.setTone( 7, VIBES );
YMF825.setTone( 8, CHURCH_ORGAN );
YMF825.setTone( 9, FLUTE );
YMF825.setTone(10, ROCK_ORGAN );
YMF825.setTone(11, NYLON_GUITER );
YMF825.setTone(12, SQUARE_LEAD );
YMF825.setTone(13, SAW_LEAD );
YMF825.setTone(14, HARPSICHORD );
YMF825.setTone(15, HARMONICA );
//マスターボリューム調整
YMF825.setMasterVolume(32);
//ボリューム調整
for(int ch=0;ch<16;ch++){
YMF825.setVolume(ch, 25);
}
}
void loop()
{
YMF825.keyon(channel1, 3, DS, y); //音の出力
YMF825.keyon(c, 5, CS); //音の出力
delay(x); //音を流す時間
YMF825.keyoff(c); //音の停止
```

5　感想

実は YMF825 って秋月では販売終了しているんですよね・・・
もし IC を持っているのであれば、github からダウンロードして遊んでみてください。
次回は YMZ285 を用いて作成しようと考えています。

[*2] 実際のプログラムは 2000 行あるのでここには一部を掲載している。プログラムは github にて公開中 [4]

参考文献

[1] 秋月電子通商, ヤマハ製 YMF825 使用 FM 音源 LSI モジュール, https://akizukidenshi.com/catalog/g/g112414/

[2] YMF825 アンオフィシャルガイド, https://qiita.com/Shigosen/items/a087546ba2322b82288f

[3] SimpleYMF825, lipoyang, https://github.com/lipoyang/SimpleYMF825

[4] 著者の github アカウント, https://github.com/Terraria012

美少女アルコールチェッカーを作る Part1

elmer

1 はじめに

秋月電子でアルコールセンサーを見つけ、esp32 を使いアルコールチェッカーを作ることにした。今回はセンサーで得た数値をスマホに表示させる所まで作った。最終目標は呼気中のアルコール濃度によってスマホに表示されるキャラクターの表情が変わるものを作ることである。

2 回路図

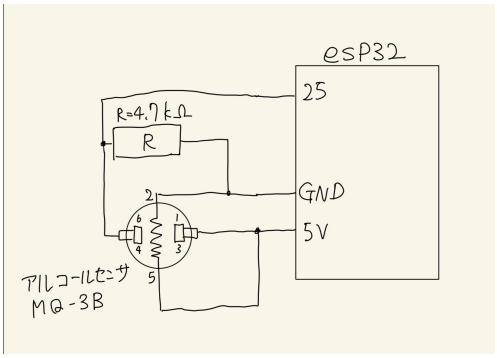


図 1 回路図

3 Blynk

Blynk Iot というスマホアプリをインストールし、esp32 から送られたデータを表示できるように設定する。[1] を参照

4 プログラム

センサーと Wifi 接続を同時に行うとセンサーの値が読み込めなくなってしまうため、最初にセンサーを動かしその時のデータを保存した後 Blynk に値を送信するようにした。

Listing 1 Python Code Example

```
#define BLYNK_TEMPLATE_NAME アルコールチェッカー""
#define BLYNK_TEMPLATE_ID ""
#define BLYNK_AUTH_TOKEN ""

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
// Wi-接続情報Fi
const char* ssid = "";
const char* password = "";
// 接続情報Blynk
const char* auth = "";

const int VOL_PIN = 25;
float savedVolt = 0;  // 保存する電圧値の変数

void setup(){
    Serial.begin(9600);

    // 最初にセンサーの値を取得し、保存
    int value = analogRead(
        VOL_PIN);
    savedVolt = value * 5.0 /
        10.0 - 180;
```

```
23
24        Serial.print("Initial Sensor
              Value: ");
25        Serial.print(value);
26        Serial.print("  Saved Volt: "
              );
27        Serial.println(savedVolt);
28
29        // Wi-接続と接続FiBlynk
30        Blynk.begin(auth, ssid,
              password);
31    }
32
33    void loop(){
34        Blynk.run();
35
36        // 保存した値をに送信Blynk
37        Blynk.virtualWrite(V0,
              savedVolt);
38
39        Serial.print("Saved Volt sent
               to Blynk: ");
40        Serial.println(savedVolt);
41
42        delay(500);  // 500ごとにへ送信
              msBlynk
43    }
```

図 2　スマホアプリの画面

5　結果

上手くいけば図 2 のように値が表示される。

6　これからやりたいこと

- データシート [2] のグラフを参考にして得られた値をアルコール濃度として表示するために計算式を修正する。
- 得られた値によって表示する画像を変える。
- 3D プリンターでアルコールチェッカーのケースを作りコンパクトにする。

参考文献

[1] 温湿度センサー (DHT11) のアウトプットを Blynk アプリに表示 (ESP - Blynk), https://dandydot.no-ip.biz/~dot/presen/advtech/seminars/iot_workshop/blynk_run_dht11/README.md.html, 2024 年 10 月 4 日最終アクセス

[2] Alcohol Gas Sensor(Model：MQ-3B)Manual, https://akizukidenshi.com/goodsaffix/mq-3b.pdf, 2024 年 10 月 4 日最終アクセス

Web ブラウザで動く クリッカーをつくる

24　かなえ

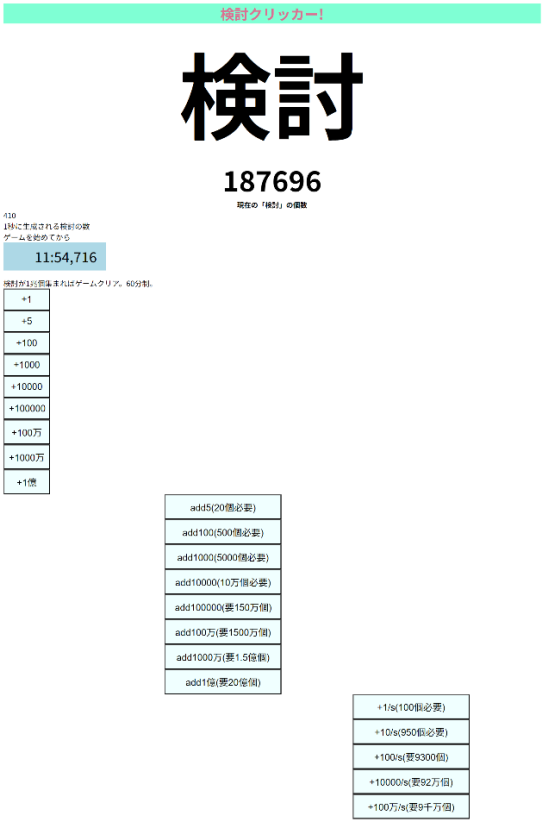
1 はじめに

Web ブラウザで動くゲームを作ってみようと思い、プログラムを書いたことのない人でも簡単にできるものはないかと考えた結果クリッカーゲームなら簡単に作れるのではないかと思い作り始めました。

2 できたこと

このような感じのクリッカーゲームを作ることができました。

具体的には、ボタンを押すとそのボタンに表示された数だけ検討の個数が増えること、ボタンを押すことで使えるボタンの数を増やすこと、あるボタンをおすことで自動でボタンを増やすこと、あとタイマーを作ることとかができました。



```
1
2  //1秒ごとにplusamountpersec分amountKを追加する
3  const autoplusing = setInterval(function(){
4      console.log(autoplusing);
5  amountK = amountK + plusamountpersec;
6  document.getElementById('amountK').
7  textContent = amountK;
8  }, 1000);
```

自動で 1 秒ごとに検討を追加するコードです。setInterval は一定時間ごとに処理を行うものです。この場合は 1000 ミリ秒ごとに

・amountK を 1 秒当たりの追加量分足す

・textContent(画面中央に表示される検討の個数の表示)を更新する

という処理をしています。

```
const plus5 = function(){
   if (p5 == 1){
    amountK = amountK +5 ; //5増やす
    document.getElementById('amountK').
    textContent=amountK;
   }
}
```

手動で検討を増やすコードです。(+5 以外のボタンもありますが、ほぼ同じ内容のコピペです。)

```
const addp5 = function(){
  if (amountK > 20){
       if(p5 ==0){
    amountK = amountK -20; //20減らす
    p5 = 1;  //+5機能をオンにする
    document.getElementById('amountK').
    textContent=amountK;
  }
  }
}

```

Add5 という+5 を開放するためのボタンのコードです。

```
<div id =switch>
   <div id="addp5">
   <form id = "faddp5" name ="test_form"
        onsubmit="return AddTable();">
      <input id="fadd5b" type="button"
          value="add5(20個必要)" onclick="addp5()">
   </form>
   </div>
```

タイマーのコードは【初心者向け】JavaScript でカウントアップを実装するコードを解説　のコピペです。

リンクは GitHub で公開します。

3　まとめ

コードを書いたことがなかったので、混乱することも多かったですが、なんとかゲームを形にすることができたので良かったです。

参考文献

【初心者向け】JavaScript でカウントアップを実装するコードを解説

sora　2023/1/24

https://programming-engineer.com/javascript-countup/

等身大パネルをノリと勢いの突貫工事で作ってみた

19 もっちゃん

みなさん、こんにちは。……というかお久しぶりです。実はいなくなったかと思いきや修士 2 年でさらに研究に追われていて、SAN 値が底をつきそうになっているもっちゃんです。73 号から 75 号は落としました。そして 76 号は実は〆切を超過してから作製しています（部長のへるくんさんに脅されましたタスケテ……）（部報担当の方には奇襲攻撃的に部報を提出してしまい申し訳ございません）。

今回は、電気通信大学の学園祭である「調布祭」に工研のマスコットキャラクターである「工研太郎（たくみ けんたろう）」の等身大パネル欲しくない！？という突発的な思いつきで等身大パネルを作製しました。ハードウェアともソフトウェアとも言えないんですけど、工作ということでここは許してくださいませ……。

1　「工 研太郎」とは？

こいつです。

図 1: 工 研太郎

誰がつくったか忘れてしまったのですが、少なくとも 10 年前から存在している、工学研究部のキャラクターです。よく「工研 太郎（こうけん たろう）」と間違えられるのですが、「工 研太郎（たくみ けんたろう）」が正しい名前です。20 kg くらいあるアナログオシロスコープを軽々抱えているので、細マッチョなのでしょう。

等身大パネルをつくるにあたって、身長の情報は必要なのですが、そんなものはありません。そこで、なぜか部室に落ちているアナログオシロスコープの寸法を測り、イラストの縮尺や変形などを考慮して計測したところ、1.75 m（つまり 175 cm）程度では？という結論に至りました。この記事では研太郎は 175 cm として扱います。

2　制作背景

1 年半ぶりに部報を書くので、書き方をど忘れしていました。そういえば筆者の部報ではいつもこの欄がありましたね。

実は等身大パネルをつくろうと思ったきっかけは工研ではありませんでした。筆者は「電気通信大学 工学研究部（Twitter(現:X): @ueckoken）」の他に、「電気通信大学 東方 Project 同好会 電々。通信（でんでんまるつうしん）（Twitter(現:X): @uec_Touhou_）」というサークルにも所属しています。この東方サークルの活動のひとつとして、他大学の学園祭を訪れた時、東方 Project のキャラクターの等身大パネルを展示しているのを見つけました。話を聞く限り、何かのイベントの抽選に申し込んだらたまたま当たったから展示したとのこと。

……いやぁ、いいなあ。等身大パネルがあると見栄えがいいなあ。せや、自分らでも作れんとちゃうか！？（筆者は関東人なので怒られてしまえ！）というノリで、製作に取り掛かった次第です。

ちなみに、この後から画像がたくさん出てきますが、研太郎と同時に「古明地 こいし」という東方 Project のキャラクターも作っていたので、その画像が中心にでてきます。こいしちゃんの画像は電々。通信の 2019 年のポストカードの絵柄を利用しています。まあでも作り方は一緒なので、本記事の進行上は問題ないと思います。暖かい目で見ていただけると幸いです。（1 年も前のものなので、画像が乏しいのです……。）

3　準備したもの

準備したものは以下のとおりです。

- 基となるイラストと、それを拡大したもの、印刷したもの
- 発泡なんでも板 A2（3〜4 枚）（DAISO、図 2 、[1]）
- カラーボード（450 mm × 800 mm）（8 枚程度、色は不問）（DAISO、図 3 、[2]）
- ボンド（持ってた）
- カッター（持ってた）

- マスキングテープ（持ってた）
- 両面テープ（持ってた）
- 謎の情熱（湧いてきた）

図 2: 発泡なんでも板（[1] より引用）

図 3: カラーボード（[2] より引用）

耐久力はなさそうに見えるのですが、実際はまあある程度は頑丈です。そりゃ製品として売られているパネルに比べれば劣りますが……。

かかった金額については、ほとんど DAISO で仕入れることができたので、1000 円程度で作製することができました。本当は印刷代が 3000 円程度するはずなのですが、後述する理由で今回はかかっていません。

4　製作工程

4.1　画像データの作製

まずはキャラクターの画像を等身大の大きさまで拡大させる必要があります。

元々ポストカード程度の大きさしかなかったものを「175 cm× それ相応の横幅で、解像度 dpi を 350（200 程度でも問題ない気がするが未検証）とするわけなので、普通に拡大しただけだと粗い画像ができあがってしまいます。本来は Photoshop とか illustrator とかを使うのが良いのですが、あいにく筆者は持ち合わせていなかったので、Clip Studio（イラストエディタ）を用いて、補完機能を信頼しながら 4 度にわけて拡大作業を行いました。（ちなみに、無料のイラストエディタとかは 20000px までしか対応していない場合もあるので、注意。）

画像なので png、そして印刷するために pdf を作成しました。結果は成功したのですが、やはりくそでかすぎるがために、さまざまなアプリが何度かクラッシュする事態になりました。また、60MB 程度あったので、何をやるにしても動作がもっさりしていて、笑っちゃいました。

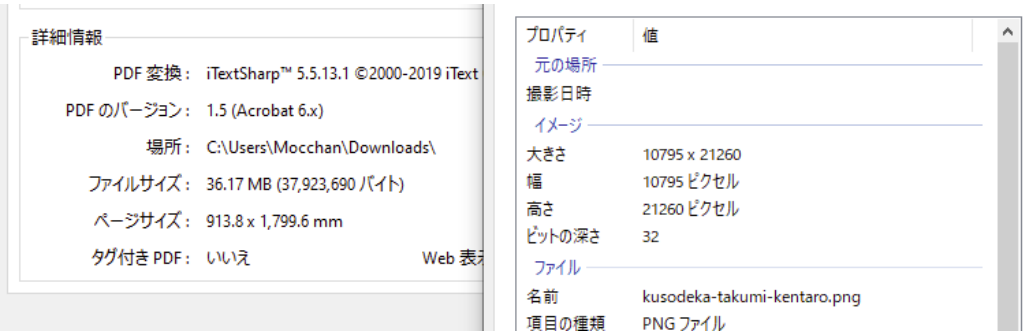


図 4: 明らかにくそでかなファイル情報

4.2　画像データの印刷

次に、出来上がったデータを印刷する必要があります。筆者は、大学構内にある大判印刷機を利用し、カラー印刷・光沢紙に印刷しました（だから印刷代は無料だったりする）。

研太郎の方は 1 枚でうまく印刷できたのですが、こいしちゃんの方は画像の構図的に大判印刷機の横幅 914 mm に収まりませんでした。そこで、お腹あたりから分割し、重なりを 3 cm 程度設けて印刷することで、接合したときに継ぎ目が目立たないようにすることができました。ちなみに、このような大きさなので 1 枚あたり 30 分程度印刷時間はかかりました。

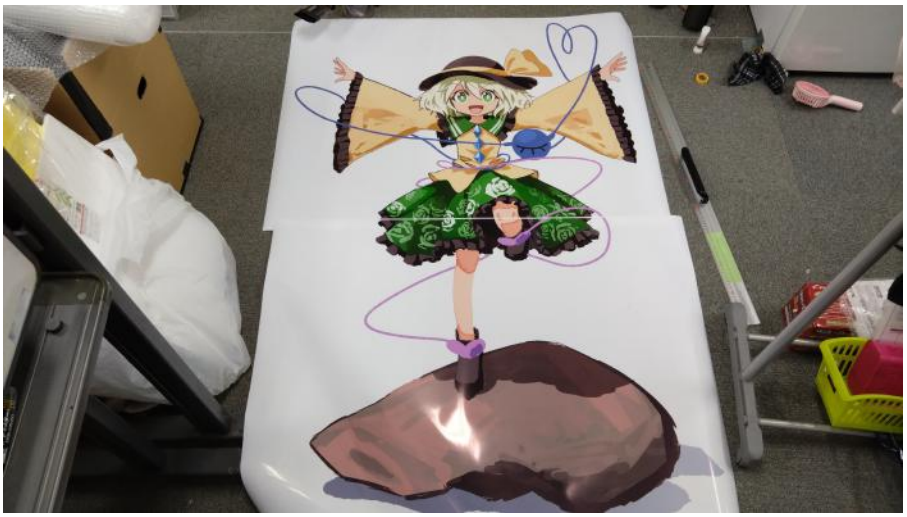


図 5: こいしちゃん。接合前なのでお腹のあたりで切れている。

4.3　トリミング

印刷した紙に対し、キャラクターの輪郭に沿うようにトリミングを行いました。本来は板に貼ってからトリミングしたほうが切る作業が二度手間にならずに済むのですが、裏の板はカラーボードを組み合わせて作る予定だったため、コストカットできるよう、紙だけで先に行いました。

トリミングの注意点として、あまり細かいところを忠実に切りすぎないことです。本当はそちらの方が見栄えが良いのでしょうが、等身大パネルの耐久度としては、太さが細いパーツが

たくさんあると弱くなってしまいます。そのため、頑張るところと手を抜くところをうまく切り分けて作業しました。

研太郎のほうはとてもやる気があったので、3 mm 程度のエッジでトリミングをすることができました。しかし、こいしちゃんのほうになってくると、~~なんか面倒になってきたし、~~構図的にしっかり切ってしまうと腕の耐久力がなさすぎると判断し、めちゃくちゃざっくり切り落としました。実際の等身大パネルはよく考えるとざっくりトリミングで作っているものも多いし、結果的にはよかったのかなと思います。

図 6: 研太郎のエッジ。やる気があった。

図 7: こいしのエッジ。戦略的撤退。

4.4　カラーボードの接着

すみません。それっぽい画像を撮影してなかったみたいです。4.6 章の図 10 に掲載されているので、そちらをご覧ください……。

カラーボードは [2] 程度の大きさだと圧倒的に面積がたりません。そこで、ボンドとマスキングテープを用いて、タイル状に接着しました。もちろんこれだけだと強度的に不安になるのですが、ボンドとマスキングテープをこれでもか！というくらいに充填・貼り付けることで、まあなんとかなったんじゃないですかね……（苦笑）。（実際、自立をしてくれるくらい頑丈になったので、ヨシ！）

4.5　カラーボードと印刷物の接合・カット作業

広いカラーボードの板ができたら、トリミングしたキャラクターの紙を両面テープで貼り付けます。ボンドを使うと紙がよれてしまうので、10 cm 間隔くらいのペースで貼り付けました。

貼り付けたらカラーボードのトリミング作業（カット作業）です。研太郎のほうはやる気のあるエッジにしたので、なるべく裏地となるカラーボードの板の色が見えないよう、紙の外枠に沿うように切り落としました。また、こいしちゃんのほうは戦略的撤退をしているので、縁取りをつけるという意味で 5 mm 程度板が見えるように切り落としました。

図 8: キレイにできた。しかし、まだ立てかけないと不安定。

4.6　支柱となる裏側構造の作製

本体はできあがったのですが、このままだと自立しません。そこで発泡なんでも板を使用して、自立してくれるような機構を作りました。

構造はシンプルで、本体の高さに対して、80° で自立してくれるように、図 9 のような三角形の板を 2 枚作製しました。本体との接着は、図 10 のように紙を蝶番のように利用し、紙に対して本体と三角板を固定しました。

そして、底面部と上部 3 分の 1 程度の位置に閂みたいな感じで横木（木じゃないけど）をはめ込みました。この横木は着脱可能とし、使用しない時は三角形の板を折りたためるようにしてあります。

これで、自立するようになり、収納のことまで考えられた等身大パネルの完成となります。

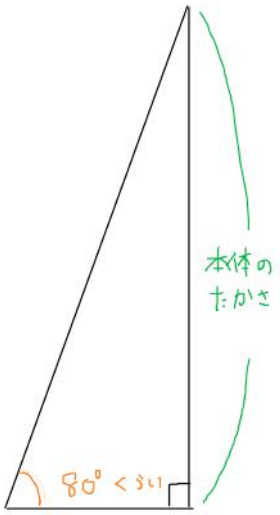


図 9: 雑なイメージ図

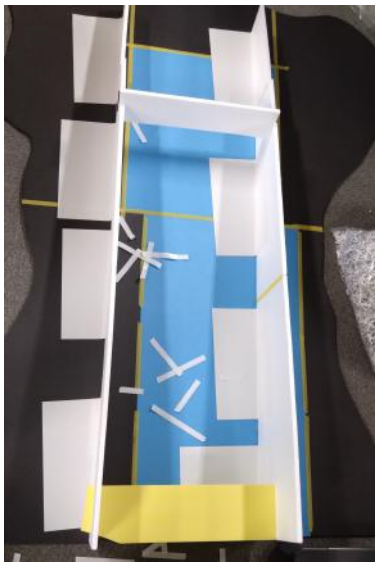


図 10: 横に刺さっている発泡板の部品で意外と安定する。

5　完成したもの

図 11: 完成品。研太郎の頭が見切れている……。

完成品は図 11 の通りです。

光沢感はないほうがよかったかなあ？と思いつつ、思いのほかしっかり自立してくれて、そしてしっかりできたので満足です。作業時間的には 2 つあわせて 6 時間程度なので、そこまで苦労もしていないといった感じです。

実際に 2023 年度の調布祭で飾った時の画像が図 12 と図 13 です。図 12 に関しては謎のコスプレをしている筆者がいますがこれは無視をしていただくとして、身長 181 cm の筆者と遜色ない大きさできちんと存在感がありました。この等身大パネルをいろいろな人に見ていただけて良かったです。図 13 に関しては左の企業作製のパネル（アイマスのキャラクター）や、右の企業に発注したオリジナルパネル（ボーカロイドのキャラクター）と比較しても、まったく見劣りしない質になっていて、来場した人の多くをだますことができました（別にだます意図はなかったんだけど、それくらい良かったということで）。どのように作ったか質問も受けるくらい来場者を引きつけることができました。

図 12: 工 研太郎（画像右）。本当に等身大で作っていることがわかる。

図 13: 古明地こいし（画像中央）。左右と比べても見劣りしない。

6　おわりに

いかがでしたでしょうか。ノリと勢いで始めた製作でしたが、思いのほかうまく作ることができて大満足の結果となりました。おそらくまだ大学に保管されていると思われるので、2024 年度の調布祭でも展示されているのではないかな……？もし気になったら足をお運びいただければと思います。

最後に、作業場所として研究室を利用しました。廊下をぶっ潰しての作業大変申し訳ございませんでした。また、真の〆切直前に唐突に提出ということを行い、大変迷惑をかけた部報担当者の方々、本当に申し訳ございませんでした。

そして、この記事を読んでくださったみなさま、今回もありがとうございました。卒業までにはもう１つくらい部報が出せるとよいなと思います……！

購入品詳細

[1] DAISO. 発泡なんでも板Ａ２. https://jp.daisonet.com/products/4984343618190. Last Accessed: 2024-10-09.

[2] DAISO. カラーボード（黒・白アソート、４５０mm×８４０mm）. https://jp.daisonet.com/products/4550480291758. Last Accessed: 2024-10-09.

汎用ロジック IC を用いたコンピュータの作成

terraria

2024/10/10

皆さんはじめまして電通大 24 生の terraria です

1　汎用ロジック IC とは？

汎用ロジック IC？ナニソレおいしいの？という人も多いだろう。
ズバリ汎用ロジック IC とはコンピュータの基礎となる演算処理を行うことができる IC なのである。

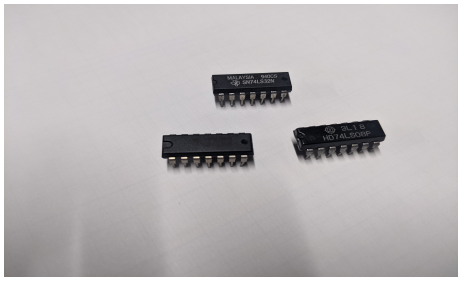


図 1　汎用ロジック IC

ここでいう基礎となる演算処理というのは、数学の基本計算である四則演算や比較、微積分等の事である。しかしながら、コンピュータは人間のように紙に計算式を書いて解を出すことはできず、電圧の大きさ (HIGH or LOW) で表される 2 進数によって計算しており、それらの制御には次に挙げる論理演算がかかわっている。

- NOT(否定)
- OR(論理和)
- AND(論理積)
- XOR(排他的論理和)

これらの論理演算子を組み合わせることによって数学の基本計算を行っていく。
ここからは具体的に各論理演算子の説明をする。

1.1　NOT(否定)

NOT 演算子は名前の通り入力信号を否定、つまり信号を反転させる。式では入力 A に対して $\bar{A}$ と表す。また、真理値表[*1]は以下のようになる。

表 1　NOT の真理値表

A	$\overline{A}$
0	1
1	0

[*1] 真理値表とは論理演算子のすべての入力パターンに対する結果の値を表にしたものである。

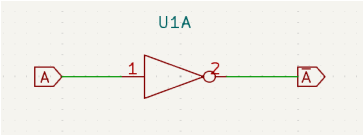


図 2　NOT 回路　回路図記号

1.2　OR(論理和)

OR 演算子は入力の和をとる。式では

$A + B = C$

と表す。また、真理値表は以下のようになる。

表 2　OR の真理値表

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

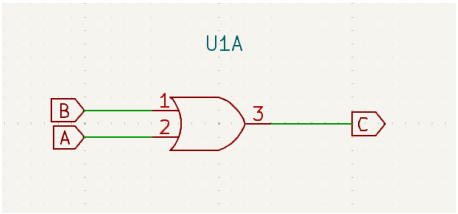


図 3　OR 回路　回路図記号

1.3　AND(論理積)

AND 演算子は入力の積をとる。式では

$A \cdot B = C$

と表す。また、真理値表は以下のようになる。

表 3　AND の真理値表

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

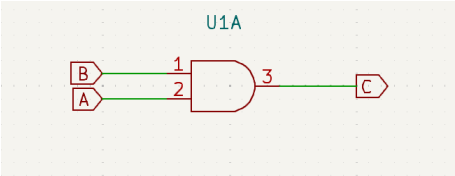


図 4　AND 回路　回路図記号

1.4　XOR(排他的論理和)

XOR 演算子は N 個の入力に対して信号の 1 の個数が奇数であるとき出力をする。式では

$A \oplus B = C$

と表す。また真理値表は以下のようになる。

表 4　XOR の真理値表

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

2　演算回路を組む

ここからはこれらの論理回路を用いて実際に計算をするための回路を考える。

計算において重要な四則演算及び、それらを

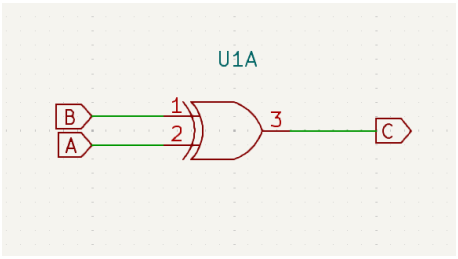


図 5　XOR 回路　回路図記号

制御する回路を考えていく。ここで 2 進数表記[*2]を導入してこれらの回路を考える。

2.1　加減算回路

まず加算回路は、繰り上げを考慮しない半加算器と繰り上げを考慮した全加算器から成る。

まず半加算器において、入力をそれぞれ A、B、足し算の結果を S 繰り上げの値を D とすると真理値表は以下のようになる。

表 5　半加算器の真理値表

A	B	S	D
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

これは、式を用いて表すと

$A \oplus B = S$

$A \cdot B = D$

よって S において XOR 回路,D において AND 回路を用いることにより実現が可能となる。

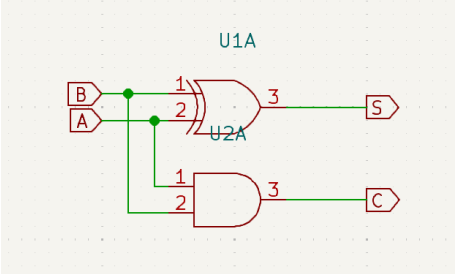


図 6　半加算器

次に全加算器において入力をそれぞれ A、B 下の桁の繰り上げを D_0、足し算の結果を S_1 繰り上げの値を D_1 とすると真理値表は次のようになる。

表 6　全加算器の真理値表

A	B	D_0	S_1	D_1
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

これは、式を用いて表すと

$(A \oplus B) \oplus D_0 = S_1$

$(A \cdot B) \oplus D_0 = D_1$

よりこちらも XOR 回路,AND 回路を用いることにより実現が可能となる。

[*2] 0bXXXX と表され、2 進数 (binary) で 00101 と書くとき、ここでは 0b00101 と表す。また、10 進数 (decimal) は 0d、16 進数 (hexadecimal) は 0h とする。

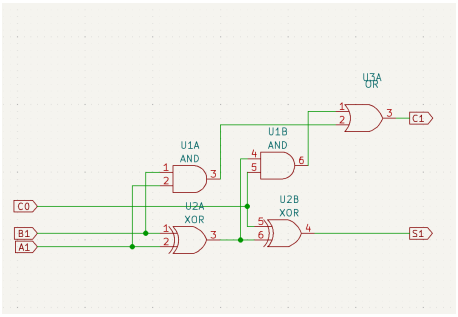


図 7　全加算器

2.2　減算

次に減算について、$A-B$ をするとき A に B の 2 の補数[*3]を足して最高位の 1 を無視すると $A-B$ と同様の結果が導かれる。例として $0d9-0d7$ について考えると、

$0d9=0b1001, 0d7=0b0111$

$0b0111$ について 2 の補数をとると、$0b1001$ となる。

$0b1001+0b1001=0b10010$

より、最高位の 1 を無視すると結果は $0b0010=0d2$ となり、

$0d9-0d7=0d2$

となることがわかる。次に結果が負になる場合について考える。例として $0d2-0d5$ について考えると、

$0d1=0b0010, 0d4=0b0101$

$0b0101$ について 2 の補数をとると、$0b1011$ となる。

$0b0010+0b1011=0b1101$

$0b1101$ において 2 の補数をとると、$0b0011$ となり、$0d2-0d5=-0d3$ となることがわかる。

[*3] 補数とは A に対して $\bar{A}$ となる数のことであり、2 の補数は補数にさらに 1 を加える操作のことである。

2.3　比較回路

次に、コンピュータの計算において重要な条件分岐を行うため、比較回路について考える。構造としては、各 bit の大小を比較した後、最高位 bit を優先して結果を反映すると比較が行える。

各桁の比較

A、B という 2 つの入力を考えるとき、大小関係を表す信号は 3 つ必要であるため、$A<B \rightarrow C$,$A=B \rightarrow D$,$A>B \rightarrow E$ とすると、真理値表は以下のようになる。

表 7　比較の真理値表

A	B	C	D	E
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

各 C,D,E において、式で表すと、

$$B \cdot (A \oplus B) = C \quad (1)$$

$$\overline{(A \oplus B)} = D \quad (2)$$

$$A \cdot (A \oplus B) = E \quad (3)$$

となる。

全体での比較

各桁で比較を行った後、最高位 bit を優先して結果を反映する回路が必要である。これは最高位 bit から順に値が同じなら下位の bit に移り、そうでないならその桁の比較の結果を反映する回路を作れば良い。

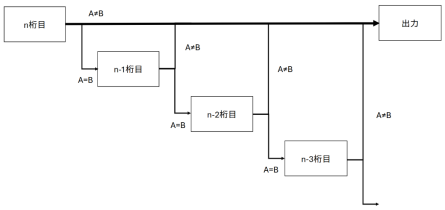


図 8　比較回路の構造

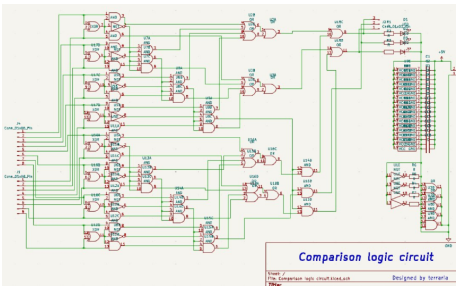


図 9　比較回路

3　感想

今回作成した回路は計算機の中でも初歩的な回路であり、これらを基礎に多くの回路を構成していく。この先は論理回路のみで構成していくと、とんでもない量の配線をして電子部品に何万もつぎ込む必要が出てくるの。そのため、一部の回路は IC 内部に組み込まれているものを使用するなどして、少し設計のレイヤーを上げて高度な計算ができるようにしていきたい。大学の後期に入った現在[*4]、秋月で購入した $1Mb$ の SRAM や I2C を用いた EEPROM など記憶回路を中心に設計をしています。課題が多く忙しすぎて、あまり制作に時間を使えていませんが・・・

NT 東京などに参加する予定でいるので、興味のある方は見に来ていただけると幸いです。

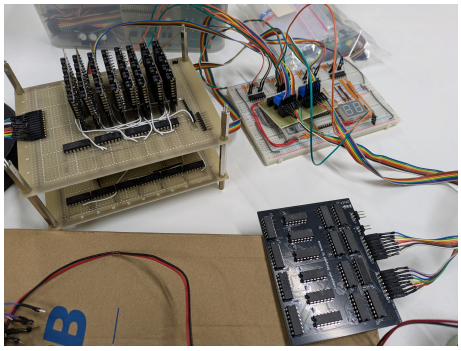


図 10　作成したもの

[*4] 2024/10/10

非絶縁型昇圧チョッパを作った話

はんかく

VVVF インバータを作っている途中の話

1　概要

交流を直流に変換する装置をコンバータと呼ぶのに対し、直流を交流に変換する装置のことをインバータと呼ぶ。一般には、交流を直流に変換する AC-DC コンバータと DC-AC インバータを組み合わせて任意の周波数と電圧に変換する回路または装置をインバータと呼んでいる。インバータは主にインバータ回路、ゲートドライブ回路、電源回路の 3 つの部分から成る。今回は 3 相誘導モータ（200v 定格 0.75kw）がなぜか部室に転がっていたので 3 相インバータ装置を作り回したいと思う。最終的には、5 インチゲージ電車を自作したいと考えているから電源は鉛蓄電池から取り 12v から 200V ほどまで昇圧することにした。

2　詳細

昇圧チョッパを作る。12v から 200v まで昇圧するように設計した。ic には dcdc コンバータ用 ic の NJM2360AD を使った。以下に回路図を示す。

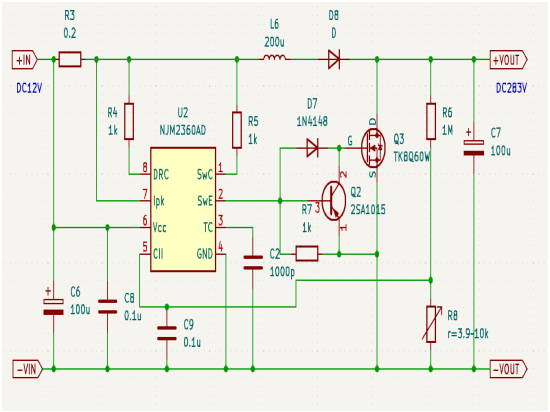


図 1　回路図

作成した回路で 283V が問題なく出せるか確認した。安定化電源の DC12V を入力して出力の開放電圧をテスターで測った。

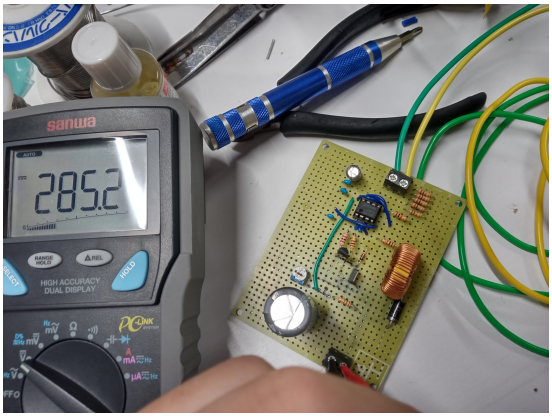


図 2　テスターで測定する様子

測定したところ開放電圧 283V が出力されていることが確認できた。

3　今後の展望

まずは dcdc コンバータを絶縁型にしたいと思う。要件を満たすトランス、コイルは市販されていないから自作していきたいと考えている。また蓄電池の出力電圧が下がったときに誤作動を防止する回路、過電流が流れたときに電流を遮断する回路、電池の逆接続を防止する回路を設計して組み込みたいと思う。

秋月フェーダーの特性測定と補正による特性改善

21　あふたむ

1. はじめに

はじめまして、もしくはお久しぶりです。工学研究部 4 年のあふたむです。毎年工学研究部っぽくない記事ばっかり書いていますが、今回は久々に電子工作の話を書きます。といっても、そんなに大したものではないですが……。

ここ数年、私は劇場の調光卓を模した装置を作っています。いくらかの話は部報 73 号に書きましたので興味ある方はそちらを参照いただきたいのですが（結局今は全てマイコンで制御しています）、簡単にいうと、フェーダー型の可変抵抗（スライドボリューム）を使って、0～100%のアナログ値を入力できる、というようなものです。音楽方面に知識のある方はミキサーを想像していただいてもいいです。

その中で秋月電子通商で売られているスライドボリュームを使っているのですが、これの特性があまりにも酷かったので、ちゃんと測定しよう&改善しようという記事になります。

……最初に結論を書いてしまいますが、千石電商にアルプスアルパインのフェーダーが売っています。45mm だと 1.3 倍ほどお値段がしますが、ケチらずにこちらを買うことをおすすめします。

2. 使用部品

- SL4515G-B103L15CM（販売コード 109242）
  秋月の 2 連 B10kΩフェーダーです。メーカーは SUPERTECH ELECTRONIC. CO., LTD. という台湾の会社です。
- RS6011Y-OC10-CO-PO-B103
  千石の B10kΩフェーダーです。メーカーはアルプスアルパイン。
- ESP32
- MCP3204-BI/P（販売コード 117732）
  Microchip 社の 12bit 4ch ADC です。

3. ESP32 ADC による測定

2 つのフェーダーを 5%ずつ動かして、ESP32 の AnalogRead で測定した結果を図 1 に示します。

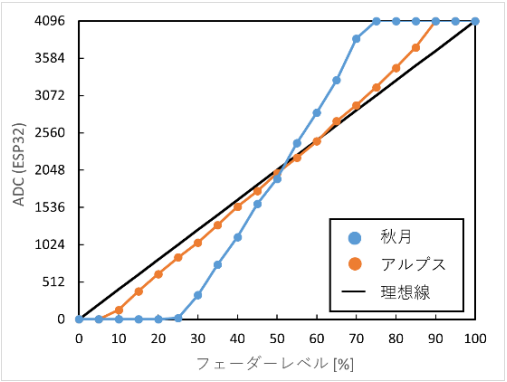


図 1　ESP32 ADC による測定

秋月フェーダーは上下 25%が 0 or 4095 に張り付いていて、B カーブを描いているのは実質中間の 50%しかありません。一方アルプスのフェーダーは中間 80%ぐらいでスイングしています。

……なんかおかしいなとは思っていたのですが、ここまで幅が狭いとは思いませんでした。

ESP32 の ADC ですが、上下 150mV 程度の範囲が上手く測定できないことが知られています[1]。そのため次はテスターで測定してみます。

4. テスターによる測定

Sanwa の PC700 というテスターを用いて、秋月フェーダーで抵抗分圧した電圧を測定した結果を図 2 に示します。また ESP32 で測定した結果を電圧に直してグラフに重ねます。

上下 25%の部分に注目すると、テスターでの測定結果は徐々に増加していることがわかります。また確かに上下 150mV の範囲内に収まっています。つまり、フェーダーの変化が小さい部分と ESP32 の ADC の欠陥が重なって、上下 25%が 0

と 4095 に張り付いていたことがわかりました。

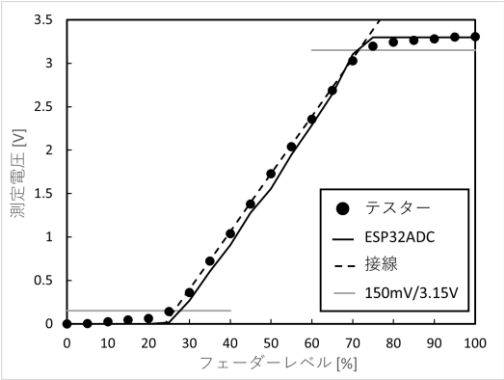


図 2　テスターによる測定

5. MCP320x による特性改善

この秋月フェーダーの特性が酷い問題を改善する方法としては、フェーダーを性能が良いものに変更するか、ADC を性能が良いものに変更してマイコンで補正するか、最初から 25～75%の範囲しか使わないように設計するかだと思います。今回は 12bit ADC の MCP3204 を使って補正をかけてみます。

補正は 0～27.5%、27.5～70%、70～100%の 3 領域に場合分けして行います。今回はアナログ入力値を LED の明るさに反映させるため、多少のズレは正直わからないということで、二次関数で大雑把にフィッティングしました。さらに、測定値から正しいフェーダーの移動量を知りたいため、逆関数を取って無理関数としています。

補正式を表 1 に、補正の様子を図 3 に、補正結果を図 4 に示します。

図 4 の横軸はフェーダーの移動量（0～100%）を 4096 段階で表したもので、縦軸は補正後の 4096 段階の値です。これが直線になっていれば、測定値を正しいフェーダーの移動量に補正できたことになります。二次関数でフィッティングした部分は多少ガタついてはいますが、それなりに補正できたのではないかと思います。

表 1　補正式（x: 12bit 測定値, y: 12bit 補正値）

領域	式
0～27.5%	$y = 66\sqrt{x}$
27.5～70%	$y = 0.48x + 982$
70～100%	$y = 4095 - 66\sqrt{4095 - x}$

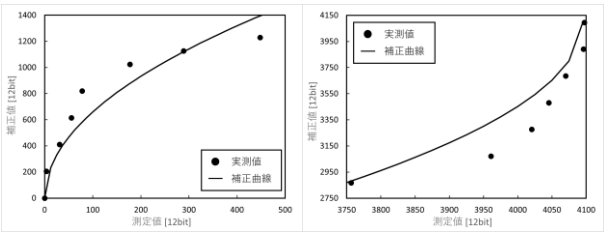


図 3 (a)0～27.5%の補正 (b) 70～100%の補正

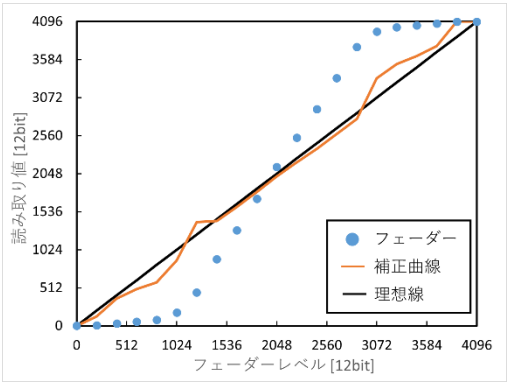


図 4 秋月フェーダーの補正結果

6. おわりに

今回の部報では秋月フェーダーの特性を ESP32 とテスターで測り、フェーダー自身の特性の悪さと ESP32 ADC の欠陥が重なって中間 50%しかスイングしなくなっていることを特定、別の 12bit ADC IC を用いて特性の補正を行いました。今の制作物はフェーダーを 64 個使うので買い替えを諦めこのような形で対応しようと思いますが、今から買う人は素直にちゃんとしたメーカーのものを買うことをおすすめします。

ということで今回はここまでにしたいと思います。次回の部報でまたお会いしましょう。お相手は私、あふたむでした！

参考文献

[1] Lang-ship「ESP32 の ADC 特性測定」, https://lang-ship.com/blog/work/esp32-adc/

JSで画像生成 QRコード編

22 McbeEringi

「本日は2024/10/6である。
ネタはまだない。
なにを以て部報を書けばよいか頓と見当がつかぬ。
何でも薄暗いじめじめした所でニャーニャー泣い」

この部法の提出締切日は10/4である。
いいから部報書け。
はい。

概要

前々回の部法に於いて、JavaScriptを用いたzip及びpngファイルの生成について執筆した。
今回は、昨今世間にて情報共有や決済の手段としてよく使われるQRコードについて理解を深めるべく、幅広いJS環境に於いて動作するQRコードのエンコーダの実装を1から行う。

生成手順

- エンコード対象の文字列郡を用意する
  - `[str,str_]`
- 文字種別を判定して使用するモードを決定する
  - 数字、英数字、8bit、漢字モードがある
- 決定したモードで文字列をエンコードする
  - 同時にヘッダーを付与する
  - ヘッダの一部はバージョンで長さが変動する
  - `[[header,str],[header,str_]]`
- エンコード済みデータの長さからバージョンを決定する
  - ユーザによるエラー訂正レベルの指定も必要である
  - 同時にデータ長が決定する
  - `{ver:N,err_lv:M,data:[[header,str],[header,str]]}`
- 埋め草bit,byteしてバージョンが要求するデータ長に合わせる
  - `{ver:N,ver:N,err_lv:M,data:[[header,str],[header,str],terminator,padding]}`
- バージョンに応じたブロック数に分割してブロック毎にRS符号を付与する
  - `{ver:N,ver:N,err_lv:M,data:[[data,err],[data,err]]}`
- 各ブロックから順番にデータを取り出す
  - 所謂インターリブ配置である
  - `{ver:N,ver:N,err_lv:M,data:interleaved}`
- 機能パターンモジュールを描画する
- 型番情報にBCH符号を付与して描画する
  - バージョン7以上の場合のみである
- インターリブ配置したデータを描画領域右下から順に配置する
- 8通りのマスクから最適なマスクを選択する
- 形式情報にBCH符号を付与して描画する
  - エラー訂正レベルと使用したマスクを含む
- 完成

day0 2024/1/29

Reed Solomon 符号

Reed Solomon符号、RS符号とはデータが欠損した場合において、欠損の検出及び元のデータの復号を可能にする追加のデータである。
以下はデータ本体を表現する配列wとRS符号の長さnを取り、RS符号を配列で返す関数rseである。

```
const
rse=(w,n)=>((
        {exp,log}=[...Array(255)].reduce((a,_,i)=>(a.exp[i]=a.x,a.log[a.x]=i,a.x*=2,(a.x>255)&&(a.x^=0x11d),a),
{x:1,exp:[],log:[]}),
        mul=(x,y)=>x&&y&&(x=log[x]+log[y],exp[x]||exp[x-255]),pow=(x,y)=>exp[(log[x]*y)%255],
        g=[...Array(n)].reduce((b,_,k)=>[1,pow(2,k)].reduce((a,y,j)=>(b.forEach((x,i)=>a[i+j]^=mul(x,y)),a),[]),
[1]).slice(1)
)=>w.reduce((a,_,i)=>(a[i]&&g.forEach((x,j)=>a[i+j+1]^=mul(x,a[i])),a),w.slice()).slice(-n))();
```

day1 10/6

文字列をエンコードしたデータを含むブロックを作成する。

```
mode 4bit | length 8~16bit | data Nbit
```

符号化

文字種別を判別する。
数字>英数字>漢字>8bitモードの優先順で決定する。
出力は整形のために複数の形式を含むオブジェクトである。
以下は数字モードのマップn、英数モードのマップa、そして漢字モードのマップkを含むmを含むdの宣言及び、文字列wを取りモードに対応する数字を返す関数modeである。

```
const
d={
        m:{
                enum:['NUM','ALPHANUM','BYTE','KANJI'],
                n:[...Array(10)].reduce((a,_,i)=>(a[i]=i,a),{}),
                a:[...'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:'].reduce((a,x,i)=>(a[x]=i,a),{}),
                k:[...Array(86)].reduce((a,y,_y)=>(y=(_y/2|0)+0x81+0x40*(61<_y),[...Array(_y==85?
33:94)].forEach((_,x)=>(x+=_y&1?0x9f:0x40+(62<x),
                        a[td_sjis.decode(new Uint8Array([y,x]))]=(y-(0x9f<y?0xc1:0x81))*0xc0+x-0x40
                )),a),{})
        }
},
mode=(w)=>(w=[...w].reduce((a,x)=>(Object.keys(a).forEach(i=>(x in d.m[i])||(a[i]=0)),a),{n:1,a:1,k:1}),w=w.n?
0:w.a?1:w.k?3:2,{x:1<<w,l:4,s:w});
```

次に文字種別からモードを決定したらモードに応じた手続きでエンコードする。
あとでbit単位で詰める必要があるのでデータ本体とbit単位のデータ長を持つオブジェクトの配列を返す。
以下は文字列群wを受け取り、対応するqrコードの画素データを返すことを目標に書かれた関数qrの一部である。
ここで、関数qrは文字列の配列wを受け取り、wに、文字列本体w、モードを表すオブジェクトm、そしてエンコード済みのデータdを含むオブジェクトを再代入している。

```
const
qr=w=>(
        w={
                d:w.map(w=>(
                        w={w,m:mode(w)},
                        w.d=([
                                _=>[...Array(Math.ceil(w.w.length/3))].map((x,i)=>(x=w.w.slice(i*3,++i*3),{x:
+x,l:[0,4,7,10][x.length]})),// NUM
                                _=>[...Array(Math.ceil(w.w.length/2))].map((x,i)=>(x=w.w.slice(i*2,++i*2),{x:
[...x].reduce((a,x)=>a=a*45+d.m.a[x],0),l:[0,6,11][x.length]})),// ALPHANUM
                                _=>[...te.encode(w.w)].map(x=>({x,l:8})),// BYTE
                                _=>[...w.w].map(x=>({x:d.m.k[x],l:13}))// KANJI
                        ][w.m.s])(),
```

最後にヘッダーを用意する。
ヘッダーに必要な情報はモードとデータ長とQRコードのバージョン(大きさ)である。
しかしQRコードのバージョンはヘッダーなしでは決定できないので、バージョン番号を引数に取りデータ全体の長さを返す関数で用意する。
再代入されたオブジェクトwに対して、バージョン番号を取りデータ長を表すオブジェクトを返す関数c、そしてバー

ジョン番号を取りヘッダを含むデータ長を返す関数lをメソッドとして宣言した。

```
                        w.c=v=>({x:(w['wd'[w.m.s>>1]].length),l:[[10,12,14],[9,11,13],[8,16,16],[8,10,12]][w.m.s]
[(9<v)+(26<v)]}),// データ長
                        w.l=v=>w.m.l+w.c(v).l+w.d.reduce((a,x)=>a+x.l,0),// ヘッダを含むデータ長
                        w
                ))
        }
);
```

day2 10/7

表1

表1とはQRコードの仕様書JISX0510 p17に記載されている、バージョン番号と対応するデータ容量についての表である。
表を定数としてそのまま書き写すのは美しくないと考える。
以下はこれをバージョン番号から導出するコード、具体的には
バージョン番号に対して表1に存在するパラメータを含むオブジェクトを返すオブジェクトd.vの宣言である。

```
d.v=[...Array(40)].reduce((a,x={},i)=>(
        // JISX0510:2018 p17 表1
        x.l=21+i*4,// モジュール数/辺
        x.fpm=(_=>(// 機能パターンモジュール
                _.ap=Math.max(0,_.apps**2-3)*25,// 位置合わせパターンモジュール
                _.tp=(x.l-16-Math.max(0,_.apps-2)*5)*2,// タイミングパターンモジュール
                _.pp+_.ap+_.tp
        ))({
                pp:192,// 位置検出及び分離パターンモジュール
                apps:i?((i+1)/7|0)+2:0// 位置合わせパターン/辺
        }),
        x.im=31+(5<i)*18*2,// 形式情報及び型番情報モジュール
        x.dm=x.l**2-x.fpm-x.im,// データモジュール
        x.dw=x.dm>>3,// データ容量
        x.dr=x.dm&7,// 残余ビット

        a[x.ver=i+1]=x,a
),{});
```

day3 10/8

表9で悩む

表9とはQRコードの仕様書JISX0510 p36に記載されている、バージョン番号と対応するブロック数、誤り訂正についての表である。
規則性が殆ど読み取れないので、これを実装するとはつまるところ表を書き写すのみとなるのである。
非常に不本意かつ苦しい行為である。
今日は休むのだ。

day4 10/9

表9の実装

表1の再実装と一緒に実装した。
同時に表E.1も実装した。
表E.1はバージョン番号に対してalignment moduleの座標を定義する表である。
以下はバージョン番号に対して、表1、表9、及び表E.1に存在するパラメータを含むオブジェクトを返すオブジェクトd.vの宣言である。
エラー訂正レベル毎の定義はバージョンに対するオブジェクトの内部にて行われる。

```
d.v=[// [...ec,...ap] ec[lv=0~3]:[short_data_l,short_blk_n(,long_blk_n)], ap:[6,...ap,l-7]
        [[19,1],[16,1],[13,1],[9,1]],[[34,1],[28,1],[22,1],[16,1]],
```

```
        [[55,1],[44,1],[17,2],[13,2]],[[80,1],[32,2],[24,2],[9,4]],
        [[108,1],[43,2],[15,2,2],[11,2,2]],[[68,2],[27,4],[19,4],[15,4]],
        [[78,2],[31,4],[14,2,4],[13,4,1],22],[[97,2],[38,2,2],[18,4,2],[14,4,2],24],
        [[116,2],[36,3,2],[16,4,4],[12,4,4],26],[[68,2,2],[43,4,1],[19,6,2],[15,6,2],28],
        [[81,4],[50,1,4],[22,4,4],[12,3,8],30],[[92,2,2],[36,6,2],[20,4,6],[14,7,4],32],
        [[107,4],[37,8,1],[20,8,4],[11,12,4],34],[[115,3,1],[40,4,5],[16,11,5],[12,11,5],26,46],
        [[87,5,1],[41,5,5],[24,5,7],[12,11,7],26,48],[[98,5,1],[45,7,3],[19,15,2],[15,3,13],26,50],
        [[107,1,5],[46,10,1],[22,1,15],[14,2,17],30,54],[[120,5,1],[43,9,4],[22,17,1],[14,2,19],30,56],
        [[113,3,4],[44,3,11],[21,17,4],[13,9,16],30,58],[[107,3,5],[41,3,13],[24,15,5],[15,15,10],34,62],
        [[116,4,4],[42,17],[22,17,6],[16,19,6],28,50,72],[[111,2,7],[46,17],[24,7,16],[13,34],26,50,74],
        [[121,4,5],[47,4,14],[24,11,14],[15,16,14],30,54,78],[[117,6,4],[45,6,14],[24,11,16],[16,30,2],28,54,80],
        [[106,8,4],[47,8,13],[24,7,22],[15,22,13],32,58,84],[[114,10,2],[46,19,4],[22,28,6],[16,33,4],30,58,86],
        [[122,8,4],[45,22,3],[23,8,26],[15,12,28],34,62,90],[[117,3,10],[45,3,23],[24,4,31],
[15,11,31],26,50,74,98],
        [[116,7,7],[45,21,7],[23,1,37],[15,19,26],30,54,78,102],[[115,5,10],[47,19,10],[24,15,25],
[15,23,25],26,52,78,104],
        [[115,13,3],[46,2,29],[24,42,1],[15,23,28],30,56,82,108],[[115,17],[46,10,23],[24,10,35],
[15,19,35],34,60,86,112],
        [[115,17,1],[46,14,21],[24,29,19],[15,11,46],30,58,86,114],[[115,13,6],[46,14,23],[24,44,7],
[16,59,1],34,62,90,118],
        [[121,12,7],[47,12,26],[24,39,14],[15,22,41],30,54,78,102,126],[[121,6,14],[47,6,34],[24,46,10],
[15,2,64],24,50,76,102,128],
        [[122,17,4],[46,29,14],[24,49,10],[15,24,46],28,54,80,106,132],[[122,4,18],[46,13,32],[24,48,14],
[15,42,32],32,58,84,110,136],
        [[117,20,4],[47,40,7],[24,43,22],[15,10,67],26,54,82,110,138],[[118,19,6],[47,18,31],[24,34,34],
[15,20,61],30,58,86,114,142]
].reduce((a,x,i)=>(
        x={_:x},x.l=21+i*4,x.ap=i?[6,...x._.slice(4),x.l-7]:[],// モジュール数/辺 位置合わせパターン座標
        x.de=(x.l**2-(192+Math.max(0,x.ap.length**2-3)*25+(x.l-16-Math.max(0,x.ap.length-2)*5)*2)-
(31+(5<i)*36))>>3,// データ容量 (size-(pos+align-timing)-info)/8 cf.p17表1

x.lv=x._.slice(0,4).map((y,lv)=>(y=y.slice(1).reduce((a,n,i)=>(a.b.push(Array(n).fill(y[0]+i)),a.d+=(y[0]+i)*n,a),
{b:[],d:0}),y.b=y.b.flat(),{lv,b:y.b,d:y.d,e:(x.de-y.d)/y.b.length})),// エラー訂正 cf.p36表9
        delete x._,a[x.v=i+1]=x,a
),{});
```

day5 10/10

バージョンの自動判定

表9からfindでデータ量が容量以下に収まるバージョンを探索する。
以下は前述にて定義されたd.vを用いて、渡された文字列のエンコード済みデータw.dを格納可能なバージョンw.vを算出し定義するコードである。

```
w.v=d.v[Object.values(d.v).find(x=>(w.d.reduce((a,y)=>a+y.l(x.v),0)<=x.lv[ecl].d<<3))];
```

パディング、ブロック分割

エンコード済みデータがバージョンが規定する所定のデータ容量に対して十分な余裕がある場合、末尾に終端パターン`0000`を追加する必要がある。
十分な余裕がない場合は、短縮及び省略が可能である。
バージョンが規定する所定のデータ容量を満たすため、これに対してデータ量が不足する場合は埋め草bit`[0]`及び埋め草byte`[0xec,0x11]`でパディングを行う。
以上の機能を実装した。
以下は前述にて算出されたエンコード済みデータw.dを、一度バイナリを表すStringに分解して、8bit毎に再構成し、終端パターン、埋め草bit,byte、を追加後、w.dを再定義するコードである。

```
w.d=(b=>[...Array(w.lv.d)].reduce((a,x,i)=>(x=b.slice(i*=8,i+8),a.a.push(x?+('0b'+x.padEnd(8,0)):(a.i^=1)?
236:17),a),{a:[],i:0}).a)(
        w.d.flatMap(x=>[x.m,x.c(w.v.v),...x.d].map(({x,l})=>x.toString(2).padStart(l,0))).join('')+'0000'
),
```

RS符号の生成、インターリーブ

QRコードは一部が欠損している場合に於いて少しでもデコード可能な確率が増すようにRS符号の末尾追加とインターリーブ配置を行う。
以上の機能を実装した。
以下は二次元配列を入力に取り転置と展開を行った結果を一次元配列として返す関数flatTrの定義と、前述したRS符号生成関数rseとflatTrを用いて、w.dに対してその操作を行うコードである。

```
const
flatTr=w=>w[w.length-1].flatMap((_,i)=>w.reduce((a,x)=>(i in x&&a.push(x[i]),a),[]));
// ...
w.d=(({d,e})=>[d,e].flatMap(flatTr))
(w.lv.b.reduce((a,x)=>(a.d.push(x=w.d.slice(a.p,a.p+=x)),a.e.push(rse(x,w.lv.e)),a),{d:[],e:[],p:0})),
```

描画系

前々回の部報にて執筆したPNGエンコーダを活用するため、執筆直後にpng.mjsとしてライブラリにまとめた。
今回はこれを活用することで低レイヤなAPIのみを用いてPNGファイルの出力を行った。
以下は、以降画素情報が追加されるオブジェクトw.aからDataURL及びBlobを出力するメソッドを内包するオブジェクトを出力するメソッドw.toPNGの宣言である。

```
import{png}from{./png.mjs};
w.toPNG=({bg=0xffffffff,fg=0x000000ff,scale:s=4,padding:g=4}={})=>png({data:
[...Array(w.v.l+g*2)].flatMap((_,y)=>(y-=g,Array(s).fill([...Array(w.v.l+g*2)].flatMap((_,x)=>(x-=g,
        Array(s).fill(0<=x&&x<w.v.l&&0<=y&&y<w.v.l?w.a[[x,y]].x:0)
))).flat())),width:(w.v.l+g*2)*s,height:(w.v.l+g*2)*s,palette:[bg,fg],alpha:1});
```

PNGエンコーダの修正

png.mjsでは巨大なデータを扱うことは想定していなかったため、PNGのIDATチャンクの容量が65535Bを超過すると画像が破損する不具合が存在する。
これはdeflate streamのuncompressed blockのデータ長指定が16bitであることに由来するものであり、
uncompressed blockを複数用いることで回避可能である。
この修正を行うことで巨大なデータも正常にPNGとしてエンコードできるよう改良を行った。
String.fromCharCodeの引数の数に制限が存在したため、同様に複数回に処理を分割した。
以下に修正箇所の差分を示す。

```
...ch([73,68,65,84, 8,29,1, ...
((x,l=x.length)=>[l>>>0&255,l>>>8&255,~l>>>0&255,~l>>>8&255,...x,...be4(adler(x))])(
// ...
],{toDataURL(){return''+btoa(String.fromCharCode(...this));},toBlob(){return new Blob([new
Uint8Array(this)],{type:'image/png'})}}))();
```

↓

```
const
map=(x,f,n=65535)=>[...Array(Math.ceil(x.length/n))].flatMap((_,i,{length:l})=>f(x.slice(n*i,n*i+n),i,l));
// ...
        ...ch([73,68,65,84, 8,29, ...(x=>[...map(x,
(y,i,a,l=y.length)=>[i==a-1,l>>>0&255,l>>>8&255,~l>>>0&255,~l>>>8&255,...y]),...be4(adler(x))])(
// ...
],{toDataURL(){return''+btoa(map(this,x=>String.fromCharCode(...x)).join(''));},toBlob()
{return new Blob([new Uint8Array(this)],{type:'image/png'})}}))();
```

day6 10/11

機能パターン描画

描画にあたって画像をデータとしてどのように表現するかを考える必要がある。
今回はブラウザのAPIに頼らないことを目標としているのでCanvas APIは使わない。
結果、`w.a[[x座標,y座標].join(',')]={p:[x座標,y座標],x:画素};`とした。
これならObject.assignを用いて複数画素の上書きを容易に行うことができる。

まず、形式情報が格納される場所を予約する。
予約は01以外の値を画素として宣言することで行う。
以下は今回用いるフォーマットを扱いやすくするためのヘルパ関数群の宣言と、w.aを算出し宣言するコードである。

```
const
oa=Object.assign,
a2px=w=>w.reduce((a,[x,y,f])=>(~f&&(a[[x,y]]={p:[x,y],x:f}),a),{}),// 配列からフォーマット
px=({x,y,f})=>(~f?{[[x,y]]:{p:[x,y],x:f}}:{}),// 画素単体
rect=({x,y=x,w,h=w,f,s=f})=>[...Array(h)].reduce((a,_x,j)=>([...Array(w)].forEach((_y,i,_)=>(_=(!i||i==w-1||!j||
j==h-1)?s:f,~_&&(a[[_x=x+i,_y=y+j]]={p:[_x,_y],x:_}))),a),{});// 矩形領域 ストローク機能付き
// ...
w.a=oa(
```

```
        a2px([...Array(8)].flatMap((_,i)=>[i+(5<i),w.v.l-1-i].flatMap(x=>[[8,x,2],[x,8,2]]))),//reserve
// ...
```

次に機能パターン、つまりバージョンが確定された時点で描画可能なパターン、具体的には
位置検出パターン(3箇所ある大きな四角)、タイミングパターン(位置検出パターンの内側同士を結ぶ破線)、位置合わせパターン(格子状に配置された小さな四角)を描画する。

```
// ...
        (({l,ap})=>oa(// functional pattern module
                a2px([...Array(l)].flatMap((x,i)=>(x=(i+1)&1,[[6,i,x],[i,6,x]]))),// time
                oa(...[[0,0,0,0],[l-7,0,-1,0],[0,l-7,0,-1]].map(([x,y,i,j])=>oa(// pos

rect({x:0+x+i,y:0+y+j,w:8,f:0}),rect({x:0+x,y:0+y,w:7,f:-1,s:1}),rect({x:2+x,y:2+y,w:3,f:1})
                ))),
                oa({},...ap.flatMap((y,j)=>ap.map((x,i)=>(i==0&&(j==0||j==ap.length-1)||(i==ap.length-1&&j==0)?
{}:oa(// align
                        rect({x:x-2,y:y-2,w:5,f:1}),rect({x:x-1,y:y-1,w:3,f:-1,s:0})
                ))))),
                px({x:8,y:l-8,f:1})// dark
        ))(w.v),
// ...
```

BCH符号

次に、バージョン7以上の場合に於いて型番情報を算出、描画する。
BCH符号はビット列同士のxorを被除数の桁数回だけ試行することで求めることができる。
以下は桁数を指定可能な被除数及び除数を入力に取りBCH符号を返す関数bchの宣言と、それを用いた型番情報の算出のコードである。

```
const
bch=({x:x,l:a},{x:y,l:b})=>[...Array(a)].reduce((e,_,i)=>(i++,((e>>(a+b-i))&1)?e^(y<<(a-i)):e),x<<b);

// ...
        // (({v,l})=>6<v?oa(rect({x:l-11,y:0,w:3,h:6,f:2}),rect({x:0,y:l-11,w:6,h:3,f:2})):{})(w.v)// 予約 仮実装
        (({v,l})=>6<v?a2px([...((v<<12)|bch({x:v,l:6},
{x:7973,l:12})).toString(2).padStart(18,0)].flatMap((x,i)=>([[...(i=[l-9-i%3,5-(i/3|0)]),+x],[i[1],i[0],+x]])):
{})(w.v)
);
```

データ描画

QRコードでは、データを右下から幅2マスを保ったまま左方向に蛇腹状に配置する必要がある。
この際、既に描画又は予約済みのマス目を上書きしてはならない。
この実装は些か面倒に思える。
今回は描画済みか否か関係なしに蛇腹状のパターンを生成した後に、描画済みの領域を除去してソートする方針で実装を行った。
以下はデータ配置パターン、w.dmの算出及び宣言と、これを用いてw.aにw.dのデータを配置するコードである。

```
w.dm=(({l})=>[...Array(l)].flatMap((_,y)=>[...Array(l-1)].flatMap((i,x)=>(
        i=l*2*((l-2-x)>>1)+!(x&1)+((x>>1)&1?l-1-y:y)*2,x+=5<x,
        w.a[[x,y]]?[]:[{p:[x,y],i}]
))).sort(({i:a},{i:b})=>a-b))(w.v);

oa(
        w.a,
        a2px(w.dm.map(({p},i)=>[...p,(w.d[i>>3]>>(7-(i&7)))&1]))
);
```

マスク

QRコードでは安定した読み取りのために、8種類うち最適なパターンとxorを取ることでデータ中に出現する位置検出パターンに似たパターンを除外している。
最適なパターンは、8種類のマスクで実際にxorを取りパターンを読むことで算出したスコアから決定する。
今となっては一刻も早く完成に漕ぎ着けたい状況なのでマスク0を決め打ちした。
マスク0は`1&~(x+y)`である。
以下はマスク番号を決め打ちし、w.dmを用いてw.aに対してマスクを施すコードである。

```
w.mask=0;
w.dm.forEach(({p:[j,i]})=>w.a[[j,i]].x^=((i+j)&1)==0);
```

最後にエラー訂正レベルとマスク番号を形式情報として配置する。
型番情報と同じくBCH符号を末尾に追加する。
以下は形式情報をw.aに配置するコードである。

```
oa(
        w.a,
        (({l},{lv},x=(+'1032'[lv]<<3)|w.mask)=>a2px([...(((x<<10)|bch({x,l:5},
{x:1335,l:10}))^21522).toString(2).padStart(15,0)].flatMap((x,i)=>[[i+(5<i)+(6<i&&l-16),8,+x],[8,l-1-
(i+(8<i)+(6<i&&l-16)),+x]]))(w.v,w.lv)
),
```

完成

「では、己がcp(src)をしようと恨むまいな。己もそうしなければ、饑死をする体なのだ。」
McbeEringiは、すばやく、ソースコード全文をコピペした。
以下がソースコード全文である。

```
import{png}from'./png.mjs';
/*

thanks to
- [日本産業規格の簡易閲覧 - JISX0510:2018](https://kikakurui.com/x0/X0510-2018-01.html)
- [独極 - 独学QRコード](http://ik1-316-18424.vs.sakura.ne.jp/category/QRCode/index.html)
- [Thonky.com's QR Code Tutorial](https://www.thonky.com/qr-code-tutorial/)
- [Creating a QR Code step by step](https://www.nayuki.io/page/creating-a-qr-code-step-by-step)
- [wikiversity - Reed-Solomon codes for coders](https://en.wikiversity.org/wiki/Reed-Solomon_codes_for_coders)

*/
const
qr=(w,{ecl=0,v=0}={})=>((
        te=new TextEncoder(),td_sjis=new TextDecoder('sjis'),oa=Object.assign,
        d={
                m:{
                        enum:['NUM','ALPHANUM','BYTE','KANJI'],
                        n:[...Array(10)].reduce((a,_,i)=>(a[i]=i,a),{}),
                        a:[...'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:'].reduce((a,x,i)=>(a[x]=i,a),{}),
                        k:[...Array(86)].reduce((a,y,_y)=>(y=(_y/2|0)+0x81+0x40*(61<_y),[...Array(_y==85?
33:94)].forEach((_,x)=>(x+=_y&1?0x9f:0x40+(62<x),
                                a[td_sjis.decode(new Uint8Array([y,x]))]=(y-(0x9f<y?0xc1:0x81))*0xc0+x-0x40
                        )),a),{})
                },
                v:[// [...ec,...ap] ec[lv=0~3]:[short_data_1,short_blk_n(,long_blk_n)], ap:[6,...ap,1-7]
                        [[19,1],[16,1],[13,1],[9,1]],[[34,1],[28,1],[22,1],[16,1]],
                        [[55,1],[44,1],[17,2],[13,2]],[[80,1],[32,2],[24,2],[9,4]],
                        [[108,1],[43,2],[15,2,2],[11,2,2]],[[68,2],[27,4],[19,4],[15,4]],
                        [[78,2],[31,4],[14,2,4],[13,4,1],22],[[97,2],[38,2,2],[18,4,2],[14,4,2],24],
                        [[116,2],[36,3,2],[16,4,4],[12,4,4],26],[[68,2,2],[43,4,1],[19,6,2],[15,6,2],28],
                        [[81,4],[50,1,4],[22,4,4],[12,3,8],30],[[92,2,2],[36,6,2],[20,4,6],[14,7,4],32],
                        [[107,4],[37,8,1],[20,8,4],[11,12,4],34],[[115,3,1],[40,4,5],[16,11,5],[12,11,5],26,46],
                        [[87,5,1],[41,5,5],[24,5,7],[12,11,7],26,48],[[98,5,1],[45,7,3],[19,15,2],
[15,3,13],26,50],
                        [[107,1,5],[46,10,1],[22,1,15],[14,2,17],30,54],[[120,5,1],[43,9,4],[22,17,1],
[14,2,19],30,56],
                        [[113,3,4],[44,3,11],[21,17,4],[13,9,16],30,58],[[107,3,5],[41,3,13],[24,15,5],
[15,15,10],34,62],
                        [[116,4,4],[42,17],[22,17,6],[16,19,6],28,50,72],[[111,2,7],[46,17],[24,7,16],
[13,34],26,50,74],
                        [[121,4,5],[47,4,14],[24,11,14],[15,16,14],30,54,78],[[117,6,4],[45,6,14],[24,11,16],
[16,30,2],28,54,80],
                        [[106,8,4],[47,8,13],[24,7,22],[15,22,13],32,58,84],[[114,10,2],[46,19,4],[22,28,6],
[16,33,4],30,58,86],
                        [[122,8,4],[45,22,3],[23,8,26],[15,12,28],34,62,90],[[117,3,10],[45,3,23],[24,4,31],
[15,11,31],26,50,74,98],
                        [[116,7,7],[45,21,7],[23,1,37],[15,19,26],30,54,78,102],[[115,5,10],[47,19,10],
[24,15,25],[15,23,25],26,52,78,104],
                        [[115,13,3],[46,2,29],[24,42,1],[15,23,28],30,56,82,108],[[115,17],[46,10,23],[24,10,35],
[15,19,35],34,60,86,112],
                        [[115,17,1],[46,14,21],[24,29,19],[15,11,46],30,58,86,114],[[115,13,6],[46,14,23],
[24,44,7],[16,59,1],34,62,90,118],
                        [[121,12,7],[47,12,26],[24,39,14],[15,22,41],30,54,78,102,126],[[121,6,14],[47,6,34],
[24,46,10],[15,2,64],24,50,76,102,128],
                        [[122,17,4],[46,29,14],[24,49,10],[15,24,46],28,54,80,106,132],[[122,4,18],[46,13,32],
[24,48,14],[15,42,32],32,58,84,110,136],
                        [[117,20,4],[47,40,7],[24,43,22],[15,10,67],26,54,82,110,138],[[118,19,6],[47,18,31],
[24,34,34],[15,20,61],30,58,86,114,142]
                ].reduce((a,x,i)=>(
                        x={_:x},x.l=21+i*4,x.ap=i?[6,...x._.slice(4),x.l-7]:[],// モジュール数/辺 位置合わせパターン座標
                        x.de=(x.l**2-(192+Math.max(0,x.ap.length**2-3)*25+(x.l-16-
Math.max(0,x.ap.length-2)*5)*2)-(31+(5<i)*36))>>3,// データ容量 (size-(pos+align-timing)-info)/8 cf.p17表1
```

```
x.lv=x._.slice(0,4).map((y,lv)=>(y=y.slice(1).reduce((a,n,i)=>(a.b.push(Array(n).fill(y[0]+i)),a.d+=(y[0]+i)*n,a),
{b:[],d:0}),y.b=y.b.flat(),{lv,b:y.b,d:y.d,e:(x.de-y.d)/y.b.length})),// エラー訂正 cf.p36表9
                        delete x._,a[x.v=i+1]=x,a
                ),{})
        },
        mode=(w)=>(w=[...w].reduce((a,x)=>(Object.keys(a).forEach(i=>(x in d.m[i])||(a[i]=0)),a),
{n:1,a:1,k:1}),w=w.n?0:w.a?1:w.k?3:2,{x:1<<w,l:4,s:w}),
        rse=(w,n)=>((
                {exp,log}=[...Array(255)].reduce((a,_,i)=>(a.exp[i]=a.x,a.log[a.x]=i,a.x*=2,
(a.x>255)&&(a.x^=0x11d),a),{x:1,exp:[],log:[]}),
                mul=(x,y)=>x&&y&&(x=log[x]+log[y],exp[x]||exp[x-255]),pow=(x,y)=>exp[(log[x]*y)%255],

g=[...Array(n)].reduce((b,_,k)=>[1,pow(2,k)].reduce((a,y,j)=>(b.forEach((x,i)=>a[i+j]^=mul(x,y)),a),[]),
[1]).slice(1)
        )=>w.reduce((a,_,i)=>(a[i]&&g.forEach((x,j)=>a[i+j+1]^=mul(x,a[i])),a),w.slice()).slice(-n))(),
        bch=({x:x,l:a},{x:y,l:b})=>[...Array(a)].reduce((e,_,i)=>(i++,((e>>(a+b-i))&1)?e^(y<<(a-i)):e),x<<b),
        // bcha=(a,b)=>[...((a.x<<b.l)|bch(a,b)).toString(2).padStart(a.l+b.l,0)],
        flatTr=w=>w[w.length-1].flatMap((_,i)=>w.reduce((a,x)=>(i in x&&a.push(x[i]),a),[])),
        a2px=w=>w.reduce((a,[x,y,f])=>(~f&&(a[[x,y]]={p:[x,y],x:f}),a),{}),
        px=({x,y,f})=>(~f?{[[x,y]]:{p:[x,y],x:f}}:{}),
        rect=({x,y=x,w,h=w,f,s=f})=>[...Array(h)].reduce((a,_x,j)=>([...Array(w)].forEach((_y,i,_)=>(_=(!i||
i==w-1||!j||j==h-1)?s:f,~_&&(a[[_x=x+i,_y=y+j]]={p:[_x,_y],x:_}))),a),{})
)=>(
        w={
                d:w.map(w=>(
                        w={w,m:mode(w)},
                        w.d=([
                                _=>[...Array(Math.ceil(w.w.length/3))].map((x,i)=>(x=w.w.slice(i*3,++i*3),{x:
+x,l:[0,4,7,10][x.length]})),// NUM
                                _=>[...Array(Math.ceil(w.w.length/2))].map((x,i)=>(x=w.w.slice(i*2,++i*2),{x:
[...x].reduce((a,x)=>a=a*45+d.m.a[x],0),l:[0,6,11][x.length]})),// ALPHANUM
                                _=>[...te.encode(w.w)].map(x=>({x,l:8})),// BYTE
                                _=>[...w.w].map(x=>({x:d.m.k[x],l:13}))// KANJI
                        ][w.m.s])(),
                        w.c=v=>({x:(w['wd'[w.m.s>>1]].length),l:[[10,12,14],[9,11,13],[8,16,16],[8,10,12]][w.m.s]
[(9<v)+(26<v)]}),
                        w.l=v=>w.m.l+w.c(v).l+w.d.reduce((a,x)=>a+x.l,0),
                        w
                ))
        },
        w.v=d.v[Math.max(v,Object.values(d.v).find(x=>(w.d.reduce((a,y)=>a+y.l(x.v),0)<=x.lv[ecl].d<<3)).v)],
        w.lv=w.v.lv[ecl],
        w.m=w.d.map(x=>d.m.enum[x.m.s]),
        w.d=(b=>[...Array(w.lv.d)].reduce((a,x,i)=>(x=b.slice(i*=8,i+8),a.a.push(x?+('0b'+x.padEnd(8,0)):
(a.i^=1)?236:17),a),{a:[],i:0}).a)(
                w.d.flatMap(x=>[x.m,x.c(w.v.v),...x.d].map(({x,l})=>x.toString(2).padStart(l,0))).join('')+'0000'
        ),
        w.d=(({d,e})=>[d,e].flatMap(flatTr))
(w.lv.b.reduce((a,x)=>(a.d.push(x=w.d.slice(a.p,a.p+=x)),a.e.push(rse(x,w.lv.e)),a),{d:[],e:[],p:0})),

        console.log(w.d.map(x=>x.toString(16).padStart(2,0))),

        w.a=oa(
                a2px([...Array(8)].flatMap((_,i)=>[i+(5<i),w.v.l-1-i].flatMap(x=>[[8,x,2],[x,8,2]]))),//reserve
                (({l,ap})=>oa(// functional pattern module
                        a2px([...Array(l)].flatMap((x,i)=>(x=(i+1)&1,[[6,i,x],[i,6,x]]))),// time
                        oa(...[[0,0,0,0],[l-7,0,-1,0],[0,l-7,0,-1]].map(([x,y,i,j])=>oa(// pos

rect({x:0+x+i,y:0+y+j,w:8,f:0}),rect({x:0+x,y:0+y,w:7,f:-1,s:1}),rect({x:2+x,y:2+y,w:3,f:1})
                        ))),
                        oa({},...ap.flatMap((y,j)=>ap.map((x,i)=>(i==0&&(j==0||j==ap.length-1)||
(i==ap.length-1&&j==0)?{}:oa(// align
                                rect({x:x-2,y:y-2,w:5,f:1}),rect({x:x-1,y:y-1,w:3,f:-1,s:0})
                        ))))),
                        px({x:8,y:l-8,f:1})// dark
                ))(w.v),
                // (({v,l})=>6<v?oa(rect({x:l-11,y:0,w:3,h:6,f:2}),rect({x:0,y:l-11,w:6,h:3,f:2})):{})(w.v)
                (({v,l})=>6<v?a2px([...((v<<12)|bch({x:v,l:6},
{x:7973,l:12})).toString(2).padStart(18,0)].flatMap((x,i)=>([[...(i=[l-9-i%3,5-(i/3|0)]),+x],[i[1],i[0],+x]]))):
{})(w.v)

        ),
        w.dm=(({l})=>[...Array(l)].flatMap((_,y)=>[...Array(l-1)].flatMap((i,x)=>(
                i=l*2*((l-2-x)>>1)+!(x&1)+((x>>1)&1?l-1-y:y)*2,x+=5<x,
                w.a[[x,y]]?[]:[{p:[x,y],i}]
        ))).sort(({i:a},{i:b})=>a-b))(w.v),

        oa(
                w.a,
                a2px(w.dm.map(({p},i)=>[...p,(w.d[i>>3]>>(7-(i&7)))&1]))
        ),

        w.mask=0,
        w.dm.forEach(({p:[j,i]})=>w.a[[j,i]].x^=((i+j)&1)==0),
        oa(
```

```
                w.a,
                (({l},{lv},x=(+'1032'[lv]<<3)|w.mask)=>a2px([...(((x<<10)|bch({x,l:5},
{x:1335,l:10}))^21522).toString(2).padStart(15,0)].flatMap((x,i)=>[[i+(5<i)+(6<i&&l-16),8,+x],[8,l-1-
(i+(8<i)+(6<i&&l-16)),+x]]))(w.v,w.lv)
        ),



        w.toPNG=({bg=0xffffffff,fg=0x000000ff,scale:s=4,padding:g=4}={})=>png({data:
[...Array(w.v.l+g*2)].flatMap((_,y)=>(y-=g,Array(s).fill([...Array(w.v.l+g*2)].flatMap((_,x)=>(x-=g,
                Array(s).fill(0<=x&&x<w.v.l&&0<=y&&y<w.v.l?w.a[[x,y]].x:0)
        ))).flat())),width:(w.v.l+g*2)*s,height:(w.v.l+g*2)*s,palette:[bg,fg],alpha:1}),

        w
))();

export{qr};
```

依存関係にあるqng.mjsのコード全文は以下の通り。

```
const// https://qiita.com/McbeEringi/items/9928a9bc05798924e68c
png=({data:d,width:w,height:h,palette:p,alpha:a})=>((
        crc=(t=>(buf,crc=0)=>~buf.reduce((c,x)=>t[(c^x)&0xff]^(c>>>8),~crc))
([...Array(256)].map((_,n)=>[...Array(8)].reduce(c=>(c&1)?0xedb88320^(c>>>1):c>>>1,n))),// https://www.rfc-
editor.org/rfc/rfc1952
        adler=data=>{let a=1,b=0,len=data.length,tlen,i=0;while(len>0){len-
=(tlen=Math.min(1024,len));do{b+=(a+=data[i++]);}while(--tlen);a%=65521;b%=65521;}return(b<<16)|a;},

be4=x=>[x>>>24&255,x>>>16&255,x>>>8&255,x>>>0&255],ch=x=>[...be4(x.length-4),...x,...be4(crc(x))],bd=[1,2,4,8][p?
Math.ceil(Math.log2(Math.ceil(Math.log2(p.length)))):3],bdi=8/bd,
        map=(x,f,n=65535)=>[...Array(Math.ceil(x.length/n))].flatMap((_,i,{length:l})=>f(x.slice(n*i,n*i+n),i,l))
)=>Object.assign([
        137,80,78,71,13,10,26,10,// header
        ...ch([73,72,68,82, ...be4(w),...be4(h), bd,p?3:alpha?6:2, 0,0,0]),// IHDR: w h bitDepth colType 0,0,0
        ...p?ch([80,76,84,69,...p.flatMap(x=>be4(a?x>>>8:x).slice(1))]):[],// PLTE: ...RGB
        ...p&&a?ch([116,82,78,83,...p.map(x=>x&255)]):[],// tRNS: ...alpha
        ...ch([73,68,65,84, 8,29, ...(x=>[...map(x,
(y,i,a,l=y.length)=>[i==a-1,l>>>0&255,l>>>8&255,~l>>>0&255,~l>>>8&255,...y]),...be4(adler(x))])(
                [...Array(h)].flatMap((_,i)=>(i=d.slice(w*i,w*++i),[0,...p?[...Array(Math.ceil(w/
bdi))].map((_,j)=>[...Object.assign(i.slice(bdi*j,bdi*++j),{length:bdi})].reduce((a,x)=>a<<bd|
(x&2**bd-1),0)):i]))
        )]),// DATA
        0,0,0,0,73,69,78,68,174,66,96,130// IEND
],{toDataURL(){return''+btoa(map(this,x=>String.fromCharCode(...x)).join(''));},toBlob()
{return new Blob([new Uint8Array(this)],{type:'image/png'})}}))();

export{png};
```

使用例は以下の通り。

```
import{qr}from'./qr.mjs';
console.log(qr(['こんにちは世界！']).toPNG().toDataURL());
```

ソースコードは以下にて公開している。
なお、このQRコードは今回作成したエンコーダで作成したものである。

RS符号の関数を実装して長らく放置していた題目が、こうして日の目を見たことに喜ぶばかりである。
時間に追われて実装したため、合理化できていない箇所が多数見受けられる。
今後数日は各所の修正を行いたい。

部報の執筆

時間無さ過ぎてやばいのである。
純アルコール量53.6gの飲酒を行い少々朦朧とした意識でこの記事を執筆していたはずだが、最早酔も冷めてしまいday7、01:52である。
「この記事が部報に掲載されるのか否かすら我々の科学力では分からないのだ。」
皆さんに於いては是非計画的な部法の執筆をされることを期待する。

Caps Lock の黙らせ方

鳩鴨 (Hatogamo)

現代の人類にとって Caps Lock 程目の敵にされているものはいない (はずだ)．今回はこのキーを Windows にて無効にする方法を紹介する．

1 Caps Lock の弊害

PC のキーボードには Caps Lock というキーがある．このキーと Shift キーを合わせて入力すると、英字入力時に小文字ではなく大文字が出力されるようになる [1]．たとえは、"apple orange banana"を入力すると"APPLE ORANGE BANANA"となる．再度 Shift と Caps Lock を入力することで元の状態に戻る．このように Caps Lock は、大文字で構成された英字文章の作成を容易にすることができる．しかし Caps Lock が Shift と隣接するように配置されていることが多いため、ユーザが Shift を押す際に誤って Caps Lock も一緒に押すことがある．この誤操作により英字を大文字でしか入力できなくなり、ユーザが困惑することになる．

2 Caps Lock の無効化

Caps Lock の災害を防ぐ方法としてキーキャップ外しがある．しかしこの方法では入力を検知するスイッチが残ってしまうため、確実に防ぐことができない．今回紹介する方法では、Windows が Caps Lock の入力を無効化するように設定する．この方法では Caps Lock の入力を Windows が無視するため、Caps Lock の入力を確実に防ぐことができる．

Caps Lock 入力の無効化には Windows のレジストリに適切な値を設定する．本記事では SharpKeys [2] というツールを用いたやり方を紹介する．このソフトはユーザに代わってレジストリを編集するため、レジストリ操作に自信がないユーザでも値を設定できる．また GUI を用いてレジストリの値を設定するため、キー入力の無効化を容易に設定できる．

3 SharpKeys の使い方

SharpKeys を起動すると図 1 のウィンドウが表示される．ウィンドウ上の表は入力されたキーとその変更先キーの組が掲載されており、入力キーが「Map this key」の列に、変更先のキーが「To this key」の列に置かれている．新しい組を追加するにはウィンドウ左下の「Add」ボタンを押し、表示された図 2 のウィンドウにてキーの組を設定する．Caps Lock を無効化するには図 2 のとおりに設定すればよい．ここで「Turn Key Off」はキー入力を無効化する．追加すると図 1 のウィンドウが図 3 のようになるため、ウィンドウ右下の「Write To Registry」を押した後に再起動を行うと Caps Lock が無効化される．

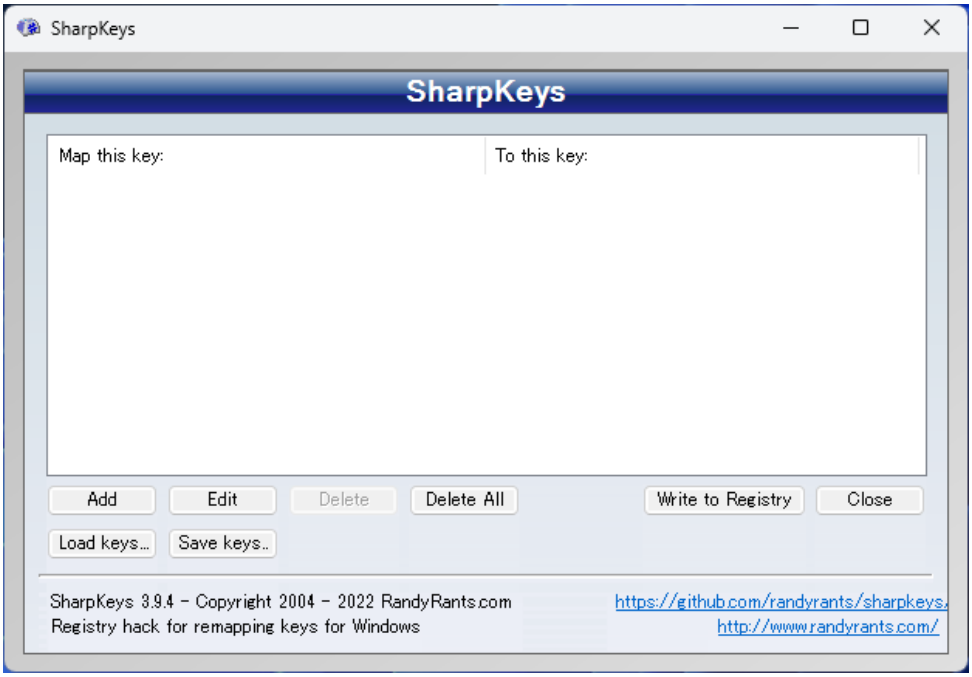


図 1: 入力キーとその変換先の一覧を表示するウィンドウ

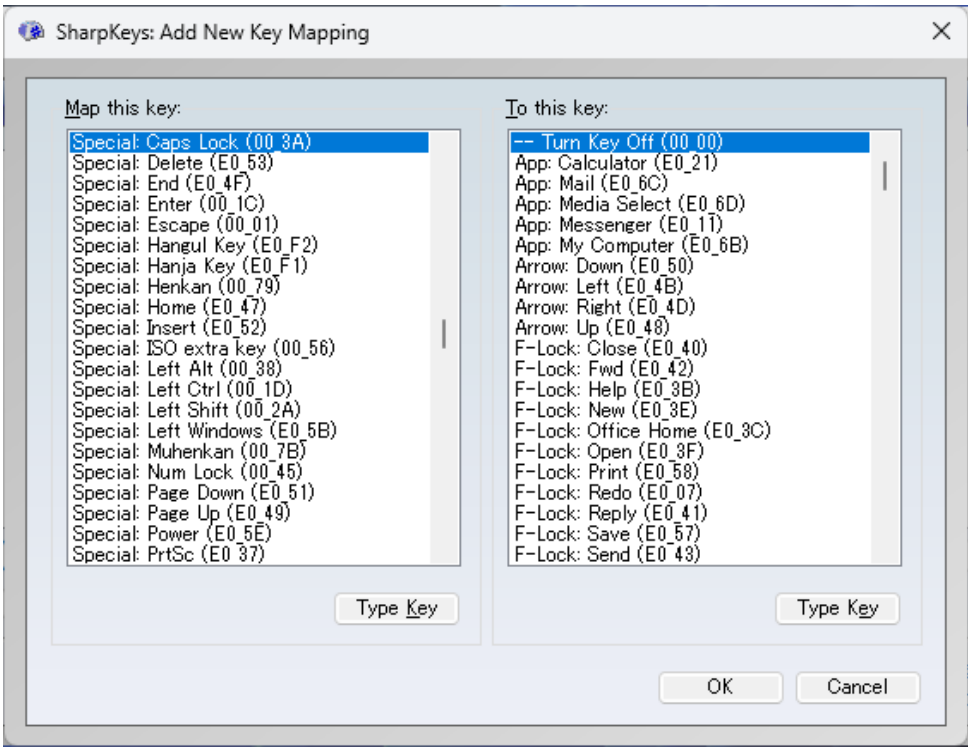


図 2: キー変更組設定画面

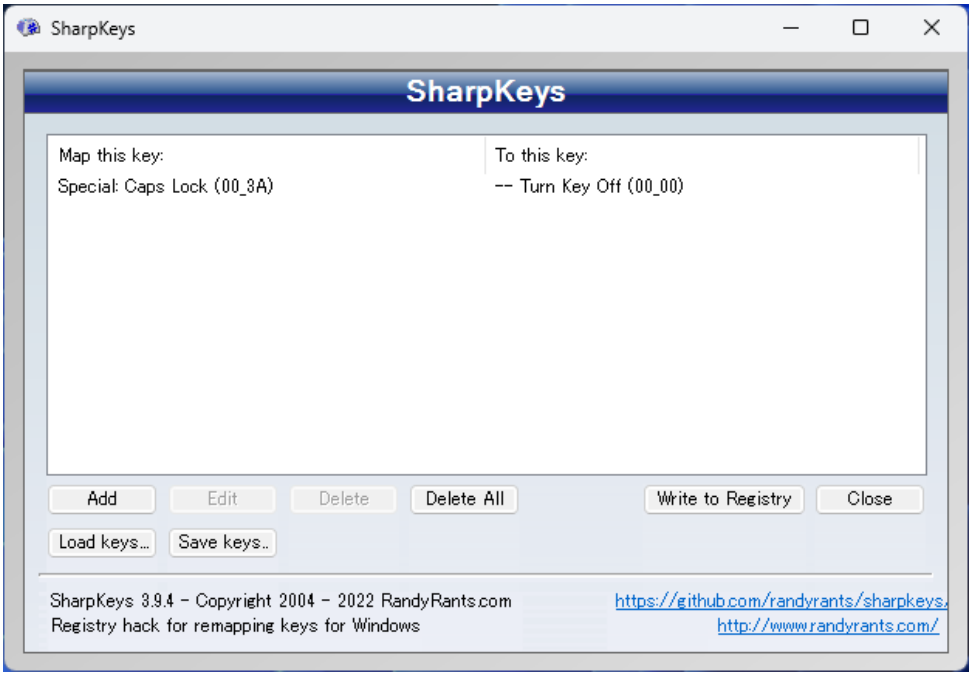


図 3: 設定後のキー変更確認ウィンドウ

4 おわりに

今回は Windows にて忌まわしき Caps Lock を無効化する方法を紹介した．この記事が読者のタイピングライフ向上に寄与することを願う．

参考文献

[1] NEC パーソナルコンピュータ株式会社. CapsLock キー (キャップスロックキー). https://support.nec-lavie.jp/support/pc/yougo/03_ei/ei080.htm

[2] RandyRants. SharpKeys. https://github.com/randyrants/sharpkeys

リモート環境の構築

Dr.Latency

夏休みで実家に帰省したり自動車免許の合宿に行ったりしたとしてもメイン PC を動かしたいと思い構築した。

1 環境

使用した OS 環境を以下に記す。

ホスト Windows10 Home
クライアント Windows11 Pro
VPN 用鯖 UbuntuServer 22.04 (Proxmox CT)

2 インストール

2.1 遠隔操作ソフト

ホスト側としたいマシンがライセンスの都合上 Windows リモートデスクトップのホストとはできない。機能は制限されるものの今回は無料版 TeamViewer を用いることとした。

ホスト側に TeamViewer Host、クライアント側に TeamViewer full client をインストール。利用開始のためアカウント登録し、 画面の指示に従い問題なく完了した。[1]

2.2 VPN など

折角なので電源も操作できるよう WakeOnLAN の環境も整備した。Ubuntu に wakeonlan をインストールし、ホスト PC の設定を行い、動作を確認した。

```
1    sudo apt-get install wakeonlan
2    wakeonlan XX:XX:XX:XX:XX:XX         #MAC Address of Host PC
```

本命である Tailscale もインストールして別のネットワークから接続を試みたところ、Proxmox の管理画面すら開けなかった。コンソールを確認したところそもそも Tailscale が正常に動作していなかった。[2]

2.3 原因

どうやらコンテナには Tailscale が必要とするファイルへのアクセス権が足りていないらしい。サイトに従い設定を書き換えたところ解決した。[3]

/etc/pve/lxc/112.conf

```
1    lxc.cgroup2.devices.allow: c 10:200 rwm
2    lxc.mount.entry: /dev/net/tun dev/net/tun none bind,create=file
```

3 結果

当初の計画の通り遠隔起動および操作ができた。画面を直接転送する関係で操作感はネットワーク速度に依存するが概ね良好だった。しかしゲームの類、特に Minecraft などカーソルの処理が少々特殊なものは操作ができなかった。

4 余談

4.1 代替手段

TeamViewer で出来なかった Minecraft だが、ホスト側で Steam に Minecraft を認識させることで出来てしまった。また、GoogleChrome の拡張機能にもリモート接続のサービスが存在するが Windows のロック解除はできない。

4.2 ある日…

リモート接続中にあるゲームのアップデートを走らせて放置していたところ、一定時間の無操作判定で切断したうえ再接続も再起動もかからなくなった。後日直接メイン機を操作したところゲーム用 SSD が逝去していた事が判明した。直接の原因は分からないが、OS ドライブでなかったこと、部分的にバックアップがあったこと、そこまで重要ではないファイルしか入っていなかったことが功を成し被害という被害はない。SSD は保証期間内なので時間があるときに修理に出そうと思っている。

今回のようなイレギュラーにも対応できるよう、電源ボタンを物理的に操作できるような手段を機会があれば製作してみたい。

参考文献

[1] TeamViewer 公式サイト (https://www.teamviewer.com/ja/)
[2] Download · Tailscale (https://tailscale.com/download/linux)
[3] Tailscale in LXC containers (https://tailscale.com/kb/1130/lxc-unprivileged)

3D プリンターで髪飾りを作った

elmer

1　はじめに

先日髪飾りを購入したが、サイズが小さく、いつ無くしてもおかしくないため予備で同じデザインの髪飾りを 3D プリンターで作った。

2　3d モデル

部室にある P1S プリンターで刷った。(図 1) サイズが 35mm × 11mm × 3mm と小さく、印刷に苦労した。(6/7 回失敗)

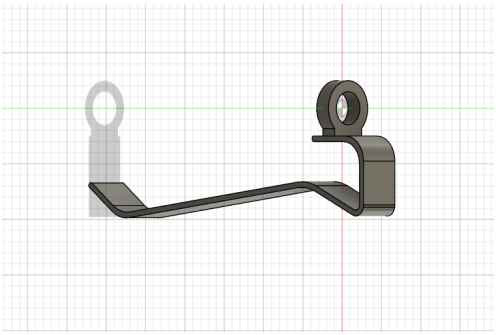


図 1　3d モデル

3　リボン

調布パルコ店にある mano creare という手芸屋さんでサテンリボンを購入した。10cm15 円でとても安い。

4　完成品

図 2 の左が作ったもの (製作費 30 円)、右が購入したもの (1000 円) 長い部分は購入品と比べ柔らかいか問題なく使用できる。リボンの色も簡単に変えられるので便利。

図 2　髪飾り

国立大学法人 電気通信大学
工学研究部 部報 第 76 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
Email ueckoken@gmail.com
URL http://www.koken.club.uec.ac.jp

発行責任者　長光 遥飛
編集者　内山 瑛太
加藤 滉太
印刷所　株式会社日光企画
執筆　工学研究部 部員