自作キーボード班
後期活動報告書

立命館コンピュータクラブ
2019 年度通年プロジェクト活動

-Leader-
青木雅典 *

-Members-
B1
阿部竜也
B2
芹澤拓也　西見元希　廣田公大朗
堀越俊行
B3
伊藤聡子　玄元奏　野崎弘晃
藤原浩一　山岡聖弥　吉田享平

2020/2/10

* 理工学部　電子情報工学科　三回生

目次

1　活動概要

文責：吉田享平

本プロジェクトでは，前期に引き続き，キーボードや，そのインタフェースを学び，実際に自ら基盤制作を行う目標のもと活動を行った．後期に行った活動は以下の通りである．

- キーボードの回路設計
- キーボード基板の制作
- キーボードの組み立て
- ファームウェアの実装

キーボードの回路設計は班員各自で好きな回路を設計し，回路をもとにパターンを設計した．キーボード基板はパターンをもとに切削機を用いて制作した．組み立てははんだこてを用いた．さらにファームウェアはArduino 互換ボードに対応した実装を行った．

2　活動内容

2.1　回路設計

文責：玄元奏

回路の設計はオープンソースの基板 CAD である KiCad[1]の回路図エディタ Eeschema を用いて行った．

はじめに，回路図記号であるシンボルを配置し，それらに配線を行って電気回路を構成する．その後，回路図上でリファレンス番号が振られていないシンボルに対して採番 (annotation) を行い，基板上に配置されるパッドの大きさや形状等のデータを持つフットプリントを各シンボルに対応付ける．

最後に，ERC(Electrical Rule Check) を用いて配線のエラーをチェックする．以上が KiCad を用いた回路設計の具体的な手順である．

今回は自作キーボードの作成を目的としているため，プッシュスイッチと電流の逆流を防ぐためのダイオードをマトリックス配線し，Arduino 互換ボードである ProMicro での制御を行った．マトリックス配線とは，図 1 のように複数のキースイッチの各端子を格子状に配線したものである．格子状に配線することによって，入力の状態を各行または各列ごとに切り替えながら読み取らなければならないという手間が発生するが，入力用のピンを節約できるという利点がある．図 1 においては 8 個のキー配置であり，通常であれば 8 本の入力ピンが必要であるが，2x4 のマトリックス配線にすることにより 6 本のピンで事足りている．この例であれば 2 本の節約にしかなっていないが，キーの数が多くなるにつれて節約できるピンの数は増大していき，メリットはより顕著になる．

[1] https://kicad-pcb.org/

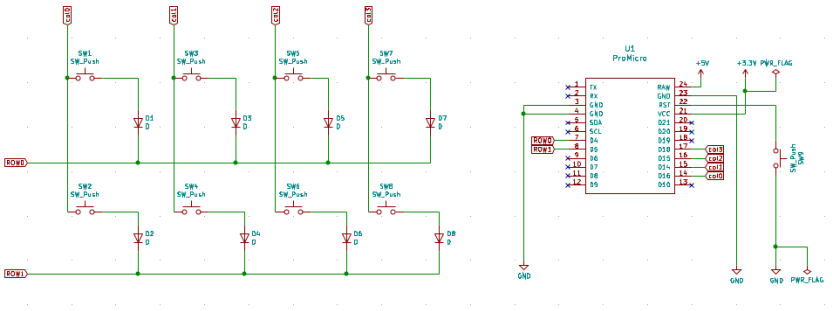

図 1　2*4 のキー配列と ProMicro の回路図

2.2　パターン設計

文責：山岡聖弥

設計した回路をもとに，パターンを設計した．回路の設計が完成してから KiCad に搭載されている PCB レイアウト エディタである Pcbnew を開き，回路図から基盤を更新する機能を用いると，回路設計のときに用いたシンボルに対応する部品と回路設計で行った配線に応じた配線すべき部品間を示す白線が表示される．パターン設計では，表示されたこれらの部品を移動させて，各部品を事前に決定したキー配列に従って配置し，配線を行った．

配線は，他の配線と干渉しないように注意しつつ，配線すべきすべての部品間の配線を行う必要がある．ここで，配線は基板の表面と裏面に行う両面配線が可能である．しかしその場合，実物の基板に配線を行うのにかなりの手間が生じる．したがって，なるべく配線は片面のみを利用する片面配線が推奨される．今回の作業においても，部品の配置や配線の回し方などを工夫することにより，なるべく片面配線となるようにパターン配線を行った．

また，配線の線幅は初期値の 0.25mm よりも 4 倍太い 1mm に指定した．これにより，後工程である部品配置におけるはんだ付けのしやすさを向上させた．

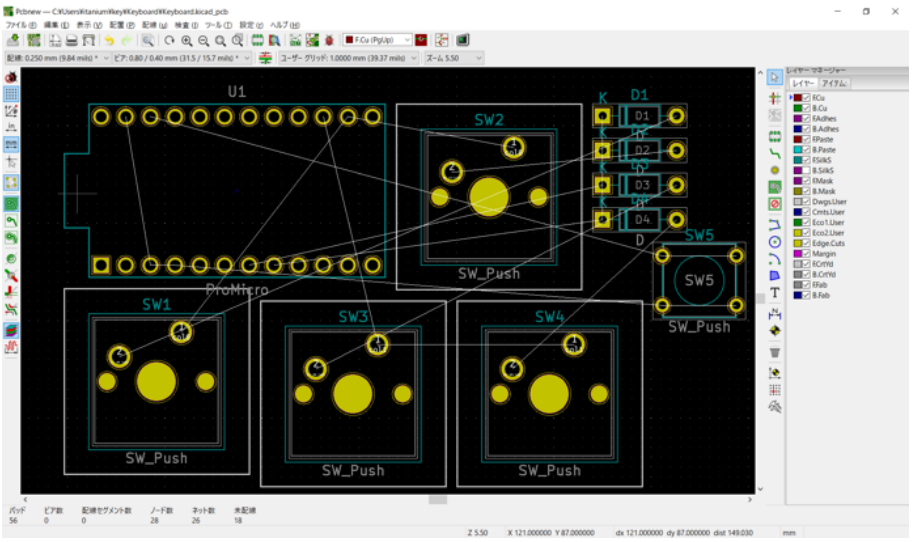

図 2　表面（パターン設計前）

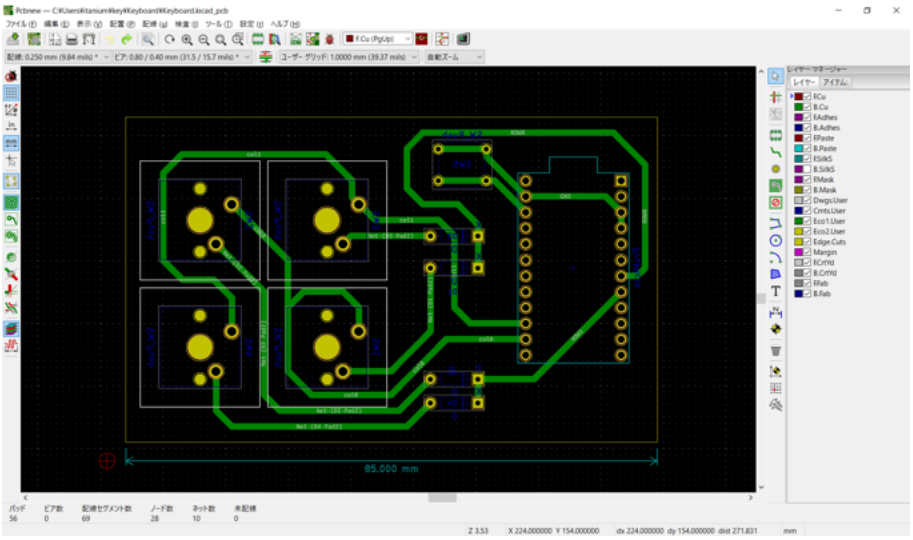

図 3　裏面（パターン設計後）

2.3　基板制作

文責：青木雅典

KiCad にて設計した基板データを元に，実際に回路基板を作成し部品の実装を行なった．

2.3.1　ガーバーデータの出力

KiCad での設計段階では，回路図や部品情報が相互に結びついており，環境依存性の強いデータとなっている．このため，一般的に基板製造を行う段階で「ガーバーデータ」と呼ばれるデータフォーマットへと書き

出す．ガーバーデータは，銅箔パターンやドリルなどの位置形状情報を保持しており，製造業者などに依頼する際は，適切な設定で出力することでデータ製作者と基板製造者の認識のズレを発生させない．

今回は，基板製造業者に依頼せず班員がそれぞれ切削機で製造した為，銅箔面とドリル情報のみデータ出力を行なった．

図 4　ガーバーデータの出力設定 1

図 5　ガーバーデータの出力設定 2

2.3.2　G-code への変換

基板切削を行うにあたり，加工機の動作指令に必要な「G-code」と呼ばれるデータを作成した．今回は，変換ツールである「FlatCAM」を利用して，位置と形状を保持するガーバーデータから G-code を生成した．FlatCAM では，加工機の先端に取り付けるエンドミルの径を設定し，理想の形状を切削するためにどのような経路を移動すれば良いか計算する．また，銅箔面の切削やドリル穴加工において，エンドミルの種類によって G-code ファイルを分割出力することが可能である．

2.3.3　基板切削

紙フェノール板に銅薄膜が積層された，生基板と呼ばれる板をエンドミルで切削した．基板パターンの輪郭線を切削すると，導体である銅薄膜の島を作り出し，パターンが描ける．一般的なエッチングとは異なり，パターン以外の部分に銅箔が残ってしまうが，溶剤処理などの必要がない為，気軽に 1 枚単位で基板制作が可能となる．

今回は，オリジナルマインド社の「mini-CNC BLACK II」を使用して基板を製作した．また，細かなパターンを要求される為，エンドミルは同社の基板加工カッター「美濃昌典」を使用した．切削した基板を図に示す．

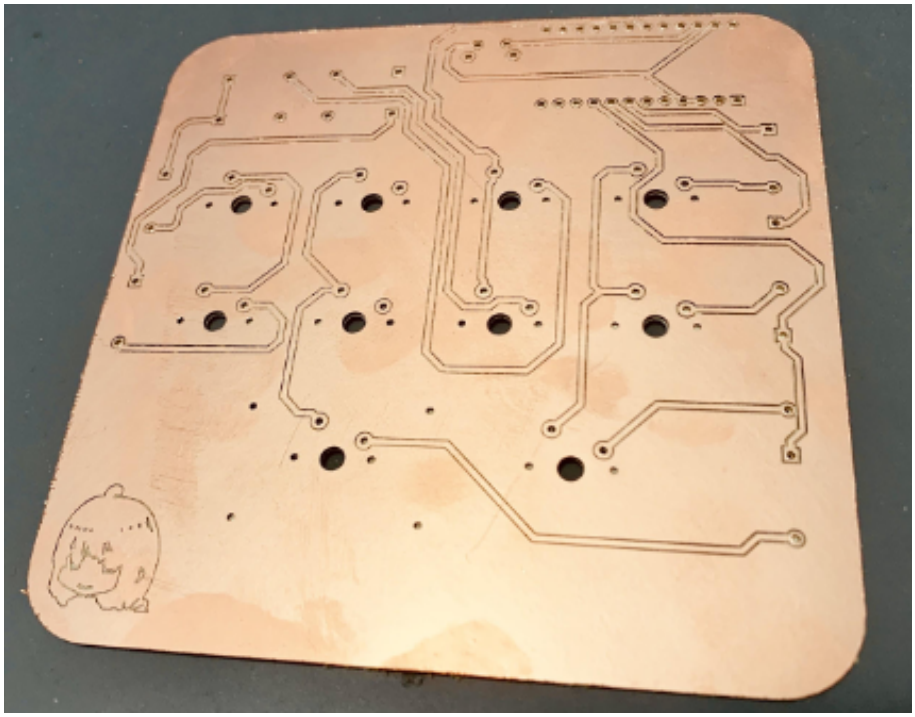

図 6　実際に切削して制作した基板

2.3.4　部品実装

製作した基板と回路図を元に，部品実装を行なった．制御には Arduino 互換ボードである「Pro Micro」，スイッチには MX 互換軸を使用し，その他ダイオードなどの部品を半田付けし，キーボード基板を完成させ

た．このとき，エンドミルの最大径が 3mm であったため，基板の穴径が不足しており，スイッチ下部の突起を切断してから実装した．

2.4　ファームウェア開発

文責：藤原浩一

制作した基板は，キーボードスイッチと抵抗がついているだけであるため，そのまま USB ケーブルでコンピュータに接続しても，入力することはできない．今回は，「Pro Micro」という Arduino 互換ボードを使用して，キーボードのファームウェアを開発した．　ProMicro は Arudino IDE で開発を行うことが可能である．なお，この報告書では開発環境の導入に関しては省略するが，参考にした Web サイトを記述しておく [1][2][3].

2.4.1　キースイッチ押下検出

以下の図 7 は，この章の文責である私が作成したキーボードの回路図となっている．左半分がキースイッチ部分，右半分が電源と ProMicro に関する回路図である．キースイッチは SW1〜6 の 6 つが実装され，スイッチ 1 つにコンデンサ 1 つがセットとなっている．

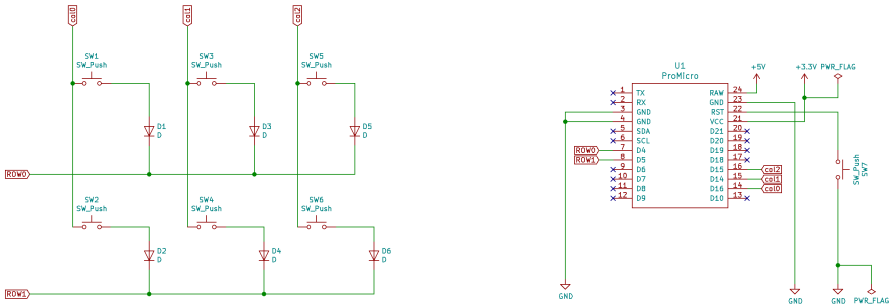

図 7　押下検出で取り扱う回路図

電気信号は col0〜3 から ROW0〜1 の方向に流れるため，col0 から順番に信号を流していき，ROW 側で電気信号を受け取ることが出れば，全てのスイッチの押下検出を行うことが可能である．以下のソースコードが簡易的に実装したものとなる．このプログラムを実行することによって，どのスイッチを押しているかがコンソールに出力される．

Listing 1　スイッチ押下検知のコードサンプル

```
#define col0 16
#define col1 14
#define col2 15

#define ROW0 4
#define ROW1 5

void setup() {
  Serial.begin(115200);
```

```
  pinMode(col0, OUTPUT);
  pinMode(col1, OUTPUT);
  pinMode(col2, OUTPUT);

  pinMode(ROW0, INPUT);
  pinMode(ROW1, INPUT);

  digitalWrite(col0, LOW);
  digitalWrite(col1, LOW);
  digitalWrite(col2, LOW);
}

void loop() {
  digitalWrite(col0,HIGH);
  digitalWrite(col1,LOW);
  digitalWrite(col2,LOW);

  delay(80);

  if(digitalRead(ROW0) == HIGH){
    Serial.write("SW1 ");
  }
  if(digitalRead(ROW1) == HIGH){
    Serial.write("SW2 ");
  }

  digitalWrite(col0,LOW);
  digitalWrite(col1,HIGH);
  digitalWrite(col2,LOW);

  delay(80);

  if(digitalRead(ROW0) == HIGH){
    Serial.write("SW3 ");
  }
  if(digitalRead(ROW1) == HIGH){
    Serial.write("SW4 ");
  }

  digitalWrite(col0,LOW);
  digitalWrite(col1,LOW);
  digitalWrite(col2,HIGH);

  delay(80);

  if(digitalRead(ROW0) == HIGH){
    Serial.write("SW5 ");
  }
  if(digitalRead(ROW1) == HIGH){
    Serial.write("SW6 ");
  }
  Serial.write("\n");
}
```

2.4.2　キーボード入力対応

前項でコンソール上においてスイッチの動作の確認が取れたため，続いてスイッチの信号をキーボード入力として扱えるようにした．今回は，Arduino ライブラリに格納されている「keyboard.h」を使用した．上記のライブラリを使用することで，単純に 1 文字入力することを始め，複数の特殊キーを組み合わせたショートカットキーなどを簡単に制作することが可能である．

今回は以下の図 8 のようなキーマップで作成した．なお，右上のキーはアルファベットの「E」である．これを実装したソースコードが以下のようになる．

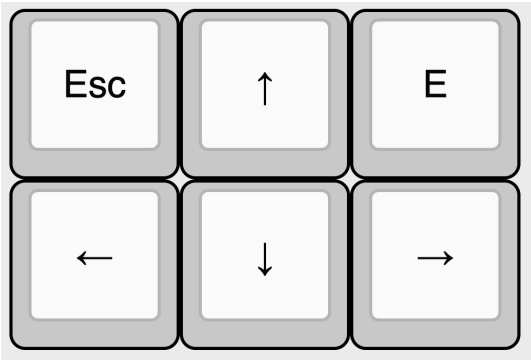

図 8　今回実装するキーマップ

Listing 2　キーボードデバイスとして動作される際のサンプルコード

```
#define col0 16
#define col1 14
#define col2 15

#define ROW0 4
#define ROW1 5

void setup() {
  Keyboard.begin();

  pinMode(col0, OUTPUT);
  pinMode(col1, OUTPUT);
  pinMode(col2, OUTPUT);

  pinMode(ROW0, INPUT);
  pinMode(ROW1, INPUT);

  digitalWrite(col0, LOW);
  digitalWrite(col1, LOW);
  digitalWrite(col2, LOW);
}

void loop() {
  digitalWrite(col0,HIGH);
  digitalWrite(col1,LOW);
  digitalWrite(col2,LOW);

  delay(80);

  if(digitalRead(ROW0) == HIGH){
    //SW1
    Keyboard.press(KEY_LEFT_ARROW);
  }
  if(digitalRead(ROW1) == HIGH){
    //SW2
    Keyboard.press(KEY_ESC);
  }

  digitalWrite(col0,LOW);
  digitalWrite(col1,HIGH);
  digitalWrite(col2,LOW);

  delay(80);

  if(digitalRead(ROW0) == HIGH){
    //SW3
    Keyboard.press(KEY_DOWN_ARROW);
  }
  if(digitalRead(ROW1) == HIGH){
    //SW4
    Keyboard.press(KEY_UP_ARROW);
  }

  digitalWrite(col0,LOW);
  digitalWrite(col1,LOW);
```

```
    digitalWrite(col2,HIGH);

    delay(80);

    if(digitalRead(ROW0) == HIGH){
      //SW5
      Keyboard.press(KEY_RIGHT_ARROW);
    }
    if(digitalRead(ROW1) == HIGH){
      //SW6
      Keyboard.press('(' e);
    }
  }
```

3　問題点と展望

文責：青木雅典

本プロジェクトで発生した顕著な問題点として，活動時間の不足が挙げられる．特に，後期活動期間では，開講期間中に活動が開催できなかった為，追い込み合宿での集中作業となった．また，作業時間の縮小により，本来発注する予定であった基板を手元で製造するなど，いくつかの実現手法の変更を余儀なくされた．

一方で，外部依存の作業が減少したことで，追い込み合宿期間中は比較的柔軟な作業が行われた．基板については先述の通り 1 枚ずつ参加者それぞれ作成し，キースイッチについても班員が代表して購入したことで，合宿中に大多数の実装を終わらせることができた．銅箔の薄型基板であるため，強度に不安の残る構造とはなったが，設計や実装作業は業者発注時と変わらないフローである為，班員が今後自身の独自基板を発注する際の目安になったと考えられる．

4　おわりに

文責：藤原浩一

本プロジェクトは，本会の 2019 年度通年プロジェクトとして活動を行った．前章で述べたように活動頻度こそ少ないものとなってしまったが，ハードウェア制作の経験は班員にとって非常に良い経験になったと感じている．また，コンピュータのキーボードという，本会会員には非常に縁のあるデバイスを一人一人カスタマイズすることは，ハードウェア制作へのハードルを低くしたと考えている．

このプロジェクトを機に，更にハードウェアに興味を持ってもらうことを期待し，以上を報告書とする．

5　参考文献

[1] (Arduino) Pro Micro https://ht-deko.com/arduino/promicro.html

[2] Arduino Pro Micro にキーパッドをつなげて PC の入力装置に http://okiraku-camera.tokyo/blog/?p=7988

[3] Arduino Leonardo(Pro Micro) の HID（キーボード）機能を使う（ショートカットキー実行，コマンド実行） https://qiita.com/MergeCells/items/17bdc1c1fb35949195b5

[4] ガーバーデータとは https://www.pban-a.com/knowledge/gerber-data.html