# 自作デバイスボード製作・データ作成・データ登録方法

Rev.A

2017/07/29

Mille-feuille のプロジェクトに自作のデバイスを追加して自分で販売したい方向けの情報です。誰でも参加可能なコミュニティにしていきたいと思っています。

大まかな流れは以下の通りです。基本的には回路設計がわかっている方か、回路設計に十分慣れてからデバイスボードを作るようにしてください。デバイスボード部分はオープンソースハードになっているので、自作したものを必要な部品ライブラリやファームウェアを公開し、デバイス基板を自由に販売してもらって結構です。どこかに委託しても結構です。ユーザーに知らせるため販売場所などを教えてください。

**１、　配線方法の決定**

作りたいデバイスボードとマイコン側への配線方法を決定して、回路図と部品ライブラリを描く。

**２、　回路図を参考に回路スクリプトを書く。**

**３、　対応するファームウェアを書く。**

一部 Mille-feuille のライブラリと合わせる必要があります。

**４、　ＧＰＩＯボードでテストしてみる。**

**５、　サイトに登録**

完成したらサイトに生成される回路を登録して、自分でデバイスボードを販売できます。販売先のリンクも登録できます。ユーザーが使うためのファームと部品ライブラリを同時に公開してください。デバイスボードはミルフィーユの回路図を自動で作れるという性質から、オープンソースハードウェアとしてください。

## １、　配線方法の決定

モジュールボードの配線は図 1 のようになっています。これに合わせて配線を考えてください。

EEPROM は microchip 社の 24AA01 SOT23-5 を使ってください。

フレキシブルケーブルのコネクタは 0.5mm ピッチ 10pin の物を使います。コネクタは SFV10R-2STE1HLF かその互換品で、すべて top contact（コネクタの上に電極がついている）ようになっていることに注意してください。

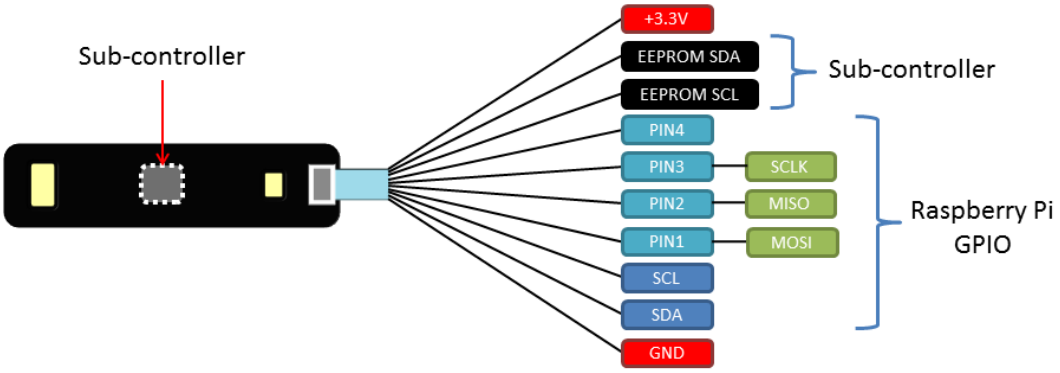


**図１　mille-feuille のモジュール基板の配線。**

デバイスボードとの端子の並ぶ方向に注意して設計してください。

ここでは例として mille-feuille 用デバイスボードの近接センサーを作ってみます。近接センサーは OSRAM SFH9206 を使用しています。部品ライブラリは mille-feuille の Eagle CAD ライブラリに入っています。図２のような配置で回路を組みました。この回路図をスクリプトで書くことになります。

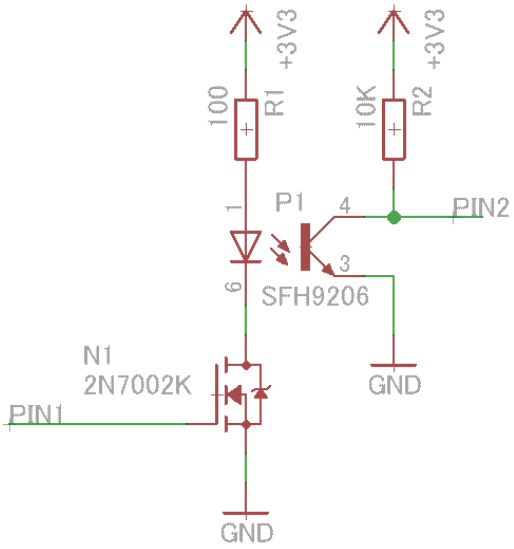


**図２　近接センサーの回路図**

次に mille-feuille に対応させるための図３のようにフラットケーブルコネクタとアドレス認識用の EEPROM の回路も用意しておきます。図３の JP1 のようにピンヘッダのパターンを組んでおいてください、後々テストが非常に楽になります。

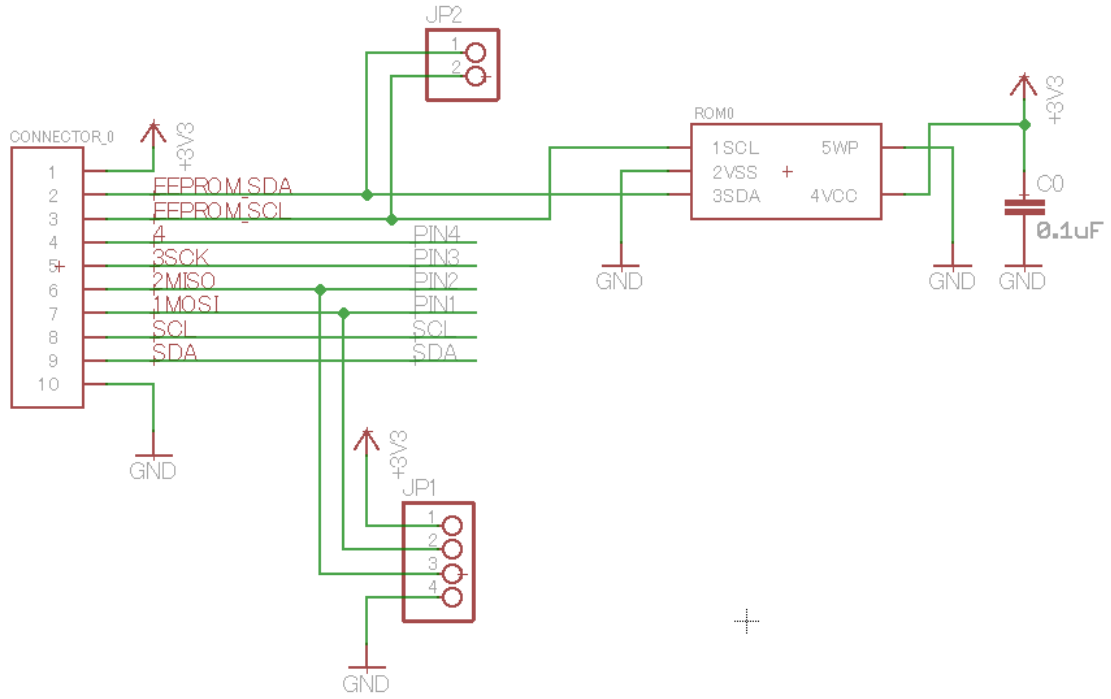


**図３　mille-feuille に対応させるためのコネクタと EEPROM**

## ２、　　回路図スクリプトを書く

回路図を生成するにはスクリプトファイルが必要になります。実際にスクリプトはどのように書くのかを説明します。

スクリプトは基本的には座標で部品や配線を指定します。

最初に単位を設定します。MIL にすると座標が書きやすいです。

```
GRID MIL;
```

部品の追加。ADD コマンド＋部品名@ライブラリ+(X 座標　Y 座標)

```
ADD SFH9206@mille-feuille ( 1100 1000 )
```

MIL 単位を指定したのでこの形になっています。**デバイス一個の回路図の最大サイズは 2017 年 7 月現在では(0 0)から(6600 6600)が最大値となっています。**このサイズをはみ出さないように図２のようなデバイスボードを作ってください。また、図３のコネクタと EEPROM はスクリプトで書く必要はありません。

端子番号の順にマイコンに接続される部分の配線を書く。必ず設定した端子番号の順番に書いてください。
今回の例では以下のように書きます。

```
NET SignalName ( 700 400 ) ( 100 400 )
LABEL ( 100 400 ) ( 100 400 );

NET SignalName ( 1300 1100 ) ( 1700 1100 )
LABEL ( 1700 1100 ) ( 1700 1100 )
```

配線は NET コマンド＋（信号名）（Ａ座標から）(B 座標まで)
最初の SignalName が PIN1 に相当し、２個目の SignalName が PIN2 に相当します。ここは mille-feuille の自動生成ツールとからんでくるので注意してください。

配線にラベルを付けるときは LABEL（書きたい座標)＋(書きたい座標)と２回座標を書きます。（もし２回書かなくてもよい方法があれば教えてください、よろしくお願いします。）

電源も ADD コマンドで同じように書けます。

```
ADD +3V3@Supply1 ( 900 1800 )
NET +3V3 ( 900 1700 ) ( 900 1600 )
```

NET コマンドで名前を書くのは基本的には任意なので、電源や特定の信号線以外は書く必要はないかと思います。

NET コマンドで配線を連続で書くときは、座標を連続で書くとどんどんつないでくれます。

```
ADD R-EU_R0402@resistor R270 ( 900 1400 )
VALUE 100 ( 900 1400 ) ( 900 1400 )
```

部品を回転して値を入れる例です。ここでは resistor ライブラリの R-EU_R0402 の面実装のチップ抵抗を書いています。R270 は回転角度です（抵抗値ではありません)。
抵抗値は VALUE コマンドで 100Ω を想定して書いています、LABEL コマンドのように座標は２回書かないとうまく設定されません。（もし２回書かなくてもよい方法があれば教えてください、よろしくお願いします。）

最終的には以下のようなスクリプトが書かれます。

```
GRID MIL;
ADD SFH9206@mille-feuille ( 1100 1000 )
ADD 2N7002K@mille-feuille ( 800 500 )

NET SignalName ( 700 400 ) ( 100 400 )
LABEL ( 100 400 ) ( 100 400 );

NET SignalName ( 1300 1100 ) ( 1700 1100 )
LABEL ( 1700 1100 ) ( 1700 1100 )

ADD +3V3@Supply1 ( 900 1800 )
NET +3V3 ( 900 1700 ) ( 900 1600 )
ADD R-EU_R0402@resistor R270 ( 900 1400 )
VALUE 100 ( 900 1400 ) ( 900 1400 );
NET ( 900 800 ) ( 900 700 )
NET GND ( 900 300 ) ( 900 200 )
ADD GND@Supply1 ( 900 100 )

ADD +3V3@Supply1 ( 1400 1800 )
NET +3V3 ( 1400 1700 ) ( 1400 1600 )
ADD R-EU_R0402@resistor R270 ( 1400 1400 )
VALUE 10K ( 1400 1400 ) ( 1400 1400 );
NET ( 1400 1200 ) ( 1400 1100 )

NET GND ( 1300 900 ) ( 1400 900 ) ( 1400 700 )
ADD GND@Supply1 ( 1400 600 )
```

他にも多くのコマンドが存在しますので、EAGLE CAD のヘルプを参考にしてみてください。
http://web.mit.edu/xavid/arch/i386_rhel4/help/

## ３、　　対応するファームウェアを書く

【Raspberry Pi 版】

ポイントを赤文字で書きます。

```
#!/usr/bin/python
from ..mil import mil
from ..mil import p
from ..mil import wiringdata
```

各種インポートしてください

```
moduleAddress1 = 0x8000
moduleAddress2 = 0x0001
moduleAddress = 0x80000001
```

アドレスは最後に登録が終わった時点で発行されるので、GPIO ライブラリのアドレスを指定しておきます。

デバイスの情報を渡すための関数です。引数は mille-feuille のベースボードのコネクタ番号を意味しています。

```
def getInfo(Number):
        if ((Number >= 0) and (Number <= 3)):
                address = moduleAddress + Number
                address2 = moduleAddress2 + Number
                address1 = moduleAddress1
                #check address
                testaddr = (address1<<16) + address2
                if address != testaddr:
                        print "ERROR: Device address is not correct!"
                        address1 = -1
                        address2 = -1

        else:
                address = -1

        IOdata = wiringdata.getWiring(address)
        datas = [address1,address2]
        datas.extend(IOdata)
        return datas
```

デバイスのアドレスが何に設定されているかを見ることができる関数です。特に不要であれば入れなくても問題ありません。

```
def getAddr(Number):
        address = -1
        if ((Number >= 0) and (Number <= 3)):
                address = moduleAddress2 + Number
        else:
                address = -1

        return moduleAddress1,moduleAddress2
```

IO の設定を wiringdata.py から引き出してくる関数です。

```
def getIOs():
        IOdata = wiringdata.getWiring(moduleAddress)
        return IOdata
```

以下、デバイスの動作にかんする関数です。

```
def power(milModClass, OnOff):
        wiringdata.IOout(milModClass.pinData[0],OnOff)
```

通常のデジタル入出力は wiringdata.IOout を使います。端子番号が回路図生成時に変わるので、 main のファイルで定義したモジュールのインスタンス milModClass.pinData[0]と指定しています。

```
def read(milModClass):
        return wiringdata.IOin(milModClass.pinData[1])
```

入出力を保持したままの接続方法に関する関数です。必要なときは入れてください。

```
def holdConnect(milModClass):
        milModClass.connect() #connect before Hold off
        milModClass.HoldOff() #hold but not connect

def holdDisconnect(milModClass):
        milModClass.HoldOn(0x01)
        #milModClass.disconnect() #connect before Hold off
        milModClass.cutWire()
```

ＧＰＩＯボードと自作したデバイスボードを接続し、動かしてみてください。
動くようでしたら、あとは回路スクリプトデータをサーバーに登録することと、アドレスの発行、ファームウェアや販売場所へのリンクなどをこちらで必要な情報として公開しておきます。

**その他、お知らせ・注意点など**

・Arduino 系マイコンボード、mbed 系マイコンボードのプログラムは 2017 年 7 月 29 日現在制作中です。こちらの方は python ではなく、C/C++で書かれる予定です。メモリー容量を考慮して Python よりずっと軽くなる予定です。完成までしばらくお待ちください。

・EEPROM への書き込みは mbed 版でライブラリを用意しています。(LPC1768、LPC824 でテストしました。)
https://developer.mbed.org/users/Info/code/EEPROM24LC01/

・デバイスのアドレスは数億通り以上登録できるので、他のユーザーとデバイスが被った場合でも登録ができます。

・連絡がつかない物や販売が終了したと判断されたものは公開情報の所に「販売が終了しています」等と表示されます。再販する場合は問合せフォームからご連絡ください。

・登録したデバイスと販売しているデバイスはご自分で整合性を保つようにしてください。こちらですべてを把握することはできません。また、回路図と実物の基板が両方動くことはこちらで動作保証しません。よろしくお願いします。