mbed 講習会

2017 年度生用
梶本研究室
ver.1 2012.3.25
ver.2 2012.4.11
ver.3 2013.1.4
ver.4 2014.2.18
ver.5 2015.5.22
ver.6 2017.3.23
ver.7 2017.4.08

Change Log
Ver2　全般的な回路図の微修正，指導方法について追記
Ver3　4 章で割り込みの追加，課題を変更
Ver4　DC モータの制御の追加，課題を変更
Ver5　DC モータのエンコーダ回路にプルアップ抵抗を追加
Ver6　全体の見直し．AD 変換の精度の追記．Ticker 等のタイマの使用，モータドライバ変更，　PC 側のプログラムを Processing に変更，ロギングの追加.
Ver 7 Processing のシリアル通信の解説を追記.

1　はじめに

３年生の宿題（DxLib+mbed）で，ブレッドボードを用いた mbed の基本的な導入については済んでいるものとして，ここでは mbed をより本格的に活用していきます．PC 側のプログラムは Processing を用います.
ここで作成したモジュールは研究の中で標準的なインタフェースとして使用しますので，きれいに作りましょう.

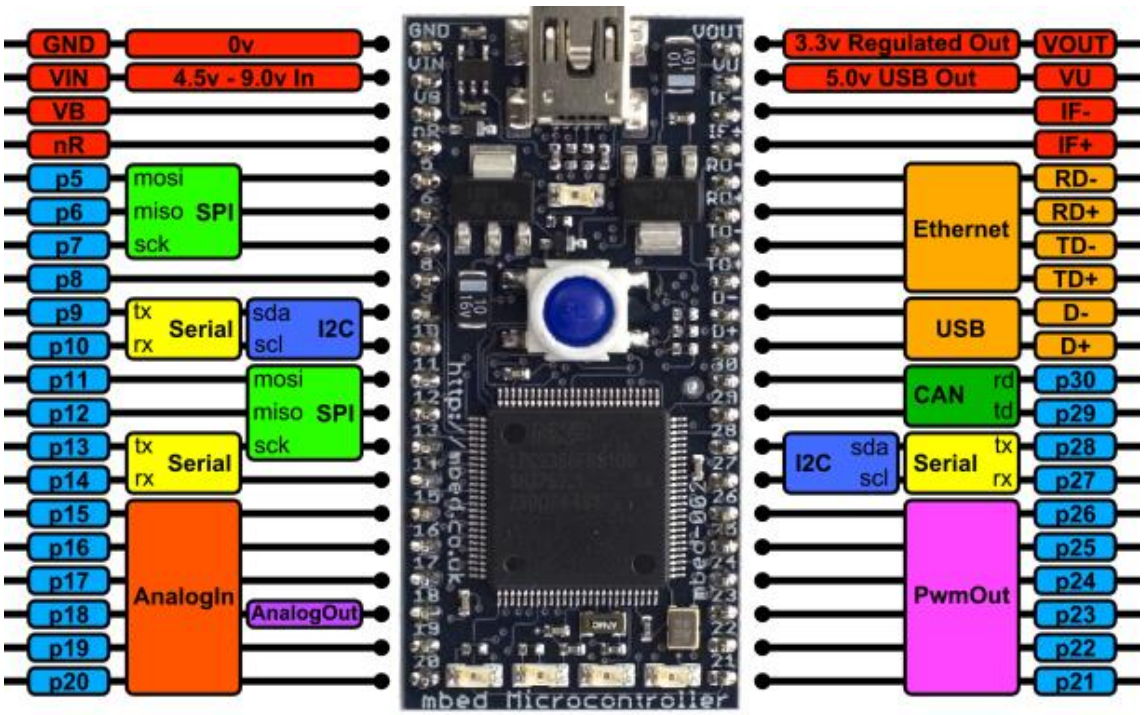


図 1　mbed ピン配置(http://mbed.org/handbook/mbed-NXP-LPC1768)

1.1 手元において欲しい本

電子工作に関する参考書は沢山ありますが，標準的な研究室新入生（ハンダ付け経験なし）には「電子工作の素」を勧めます．

1.2 課題の進め方

mbed はできることが多いので，まず研究室標準のインタフェースとして使えるところまで皆で行います．時間がある場合は選択課題を行ってください．

- 2 章「回路の準備」: ブレッドボードで行ったことを自作の基板で行います．
- 3 章「外付けスイッチ」: 外付けスイッチによる入力を行います．割込みの練習を兼ねます

- 4 章「LED の駆動」: LED を点灯させます.
- 5 章「RC サーボモータの駆動」: RC（ラジコン）サーボモータを動かします.
- 6 章「アナログ出力」: 内蔵の DA 変換器を使います.
- 7 章「SPI 通信」: SPI 通信によるポートの拡張例として DA 変換ポートの増設を行います.
- 8 章「DC モータ制御」: H ブリッジ IC とエンコーダで DC モータを PD 制御します.
- 9 章「データロギング」: Processing を用い，実用的なデータロギングを行います.
-

一週目：一日目 2～3 章，二日目 4～5 章，三日目 6 章，４日目 7 章，５日目予備日
二週目：一日目～二日目 8 章，三日目～四日目 9 章
というペースで進めます.

選択課題は以下のとおりです．時間があったらやってください.

- 11.1「７セグメント LED」: ７セグメント LED を動かします．シフトレジスタを使う例題にもなっています.
- 11.2「通信機能を持つセンサモジュールの使用１」: I2C 通信機能を持つセンサを使います．ここでは加速度センサを使って衝撃を計測，振動出力することによって叩いた対象の材質感を表現する事を試みます.
- 11.3「通信機能を持つセンサモジュールの使用１」: 同様に I2C 通信機能を持つセンサを使います．ここでは９軸センサを使って頭の姿勢を計測し，インタフェースとして使うことを試みます.
- 11.4「超音波センサの使用」:割込みプログラムの実例として超音波センサを使います．２つ使うことで擬似的にテルミンを作ります.
- 11.5　「超音波センサの作成」: 超音波距離センサを自作します.
- 11.6「外付け AD」: 12bit，8ch の AD 変換器を二つ用い，またフィルム状力センサを用いて力分布を測定します.
- 11.7「リニアアクチュエータの駆動」: リニアアクチュエータの先端に力センサを取り付け，柔らかさ感を変えていきます.
- 11.8「Wii コントローラと bluetooth 通信」: mbed をマスタとして Wii コントローラを入力機器として利用します.
- 11.9「振動呈示型データグローブ」
- 11.10「USBAudio 機器を作る」

1.3 レポート

最後にレポートを書いてください．添削します.

- Word を標準とします．Word 講習を兼ねていますので，図表番号への参照などの**参照**

機能は必ず使って下さい． Word 使い方講習は講習会の途中に行います．

- 実験中に取得した図（特にオシロスコープの画面キャプチャ）を貼って下さい．
- 回路図など，テキストにある内容を繰り返す必要はありません．
- コツはレポートを書きながら実験を行うことです．これは研究の場面でも同じです．データはリアルタイムに文書化していきましょう．
- 提出されたレポートの添削と回覧は研究室全体で行います．

1.4 指導方法（指導者向け）

次の指導が必要です．放置して野生力をつけるという方針もありえますが，この講習に限って前半は集中的に指導し，後半も適切な苦労をさせてください．

- 初日冒頭，2～5 章の説明．
  - ハンダ付けの方法
  - スイッチ回路の動作の原理
  - LED 回路の抵抗計算の原理（LED の特性），テスタで LED の極性を知る方法
  - RC サーボの駆動の前提となる PWM の説明
  - オシロスコープによる測定（トリガの概念，外部トリガを用いる方法）
- ３日目冒頭，6～7 章の説明
  - オペアンプ入門
  - 圧着端子の使い方説明
  - SPI 入門，シフトレジスタの説明から，LTC1660 のマニュアルを見ながらタイミングチャートと送信データ構造の理解
  - オシロスコープによる測定（外部トリガを使う方法）
- ２週間目冒頭，8 章の説明
  - エンコーダの説明，モータドライバ IC の説明
  - アルミ加工の解説（これは個別に教える）
- ２週間目中頃，9 章の説明
  - Processing の簡単な説明．通信遅延の重要さについて解説．

この講習は研究室的技能の中心となりますので，集中的に指導しましょう．講習資料は不完全ですので，プログラミング言語のヘルプや IC のマニュアルを隅々まで理解しないと出来ません．しっかり理解すればできる（あいまいな理解ではできない）という体験に持っていくようコントロールしてください．

問題解決（デバッグ）の効率的な手順も指導してください．例えば望ましい値が計測値として出ない場合，最初は必ずフリーズします．これを，まずテスタで確かに電圧が出てい

ることを確認して「ハードの問題かソフトの問題かを切り分ける」，次に通信ソフトで表示して「通信の問題かそれ以外かを切り分ける」，という**二分岐による可能性探索**を体験させてください．**初めてだとテスタやオシロが開発／デバグに使うものという概念がありません**（レポート用のデータを取るものという認識）．**「テスタ／オシロで測ってみた？」は最初に聞く**ようにしてください．

プログラムに関しては， if 文，for 文，while 文は理解していても，ある課題に対してどのようなプログラム構造を作るべきかという大きな視点は持っていないのが普通です．**ホワイトボードで擬似的なプログラムを書くという感じで議論**しながら，プログラムの**構造を作っていく**感覚を身につけさせてください．これはフローチャートを書くというのではありません（ソースコードそのものが論理構造なのでフローチャートは迂遠すぎます）．ホワイトボードでの議論に慣れさせるためにも意図的にたくさん使ってください．

最後に**わかっていない YES は徹底的に問い詰めてください**．講習会の場で徹底追求すれば，それは課題を解くことに直結するので利益が見えやすいです．つまり**わかっていないことをごまかさないことが実際に有益であることを体験できる機会**になります．

以上のように本講習の指導の際は

- **プログラムソースとハードウエアマニュアルを完全に理解する必要があるという感覚**
- **デバグ（二分岐）の概念，テスタやオシロがデバグ用ツールであるという常識**
- **プロジェクトマネジメントの概念，ソースコードは構造の設計が大切という体験**
- **わかっていないことをごまかさない習慣**

をつけるように促すことが大切になります．

2　回路の準備

春休みにブレッドボードを用いて作った加速度センサ回路を作成する．ただし回路は配線しやすいように微修正してある．

配線は錫メッキ線，あるいはジュンフロン線（非常に細い）を用いる．無用に太い線は使わない（ただしモータを駆動する部分や電源部分には太い線を用いる）．加速度センサは後で外すので 8 ピンの IC ソケットを用いる．mbed 用には 20 ピンのシングルソケットを 2 つ用いる．

ハンダ付けの方法は下記リンクや「電子工作の素」などを参照する．

ハンダ付けの方法　http://www.youtube.com/watch?v=S5f7jueQHr8

良いハンダ付け形状など　http://homepage1.nifty.com/x6/elecmake/solder.htm

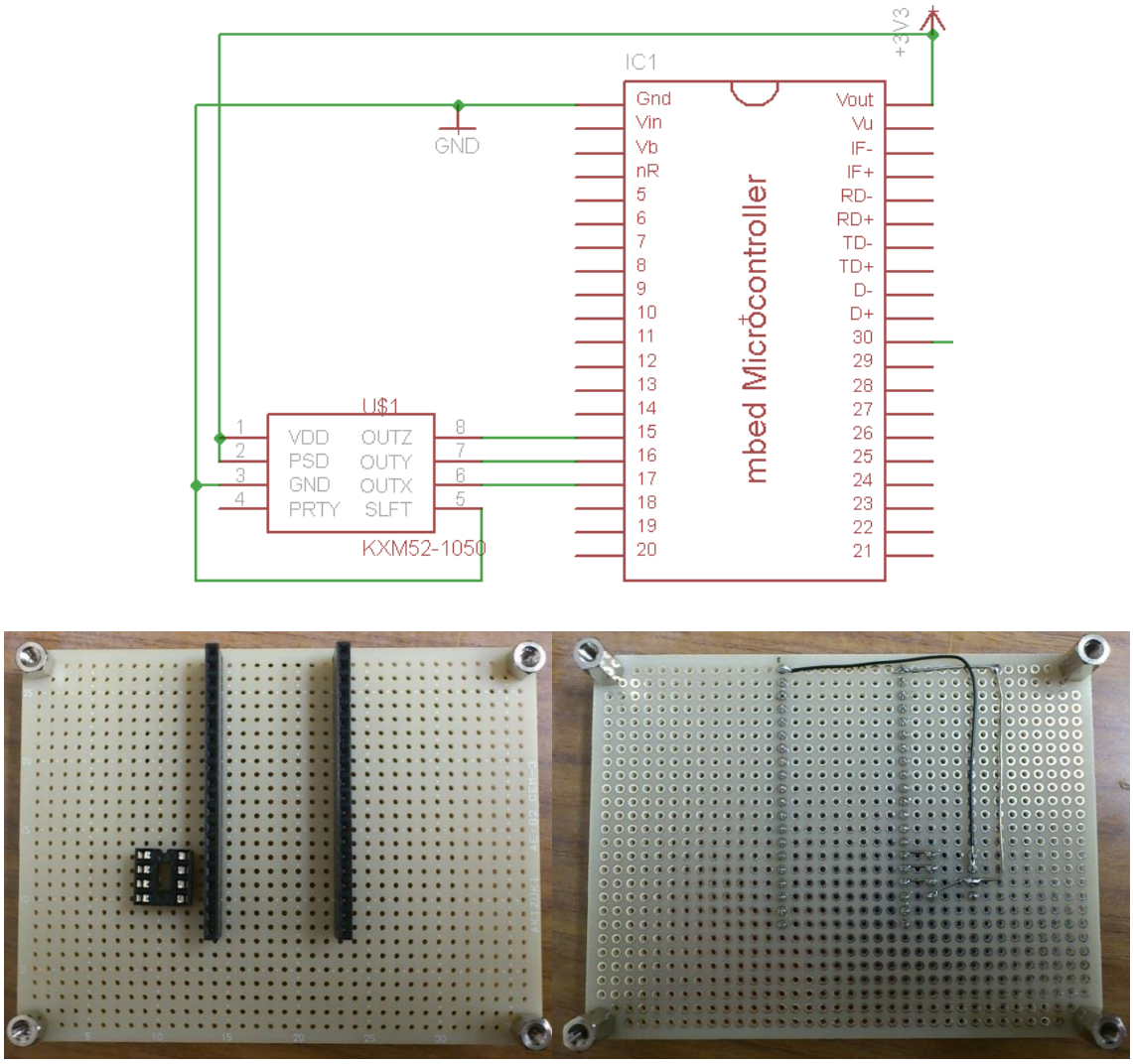


図 2　加速度センサ回路

3　外付けスイッチ

DigitalIn を用いて，外部からスイッチ入力を受け付けるようにする
http://mbed.org/handbook/DigitalIn

ボタンを押したときにどこが導通するかテスタでチェックしたうえで使用すること．web のサンプルを使って mbed 上の LED を光らせる．ただし入力ピンは p5 ではなく（p5 は後で使用），p30 とする．

[課題1]　スイッチのステータスをシリアル通信で PC に伝えるようにする．例えばスイッチが押されている間は”a”を，押されていない間は”b”を送信し続けるなど．PC 側はシリアル通信用ソフトを用いる（RealTerm, TeraTerm など）

RealTerm http://realterm.sourceforge.net/

TeraTerm http://sourceforge.jp/projects/ttssh2/

[課題2]　スイッチ回路に抵抗が必要な理由を考察せよ

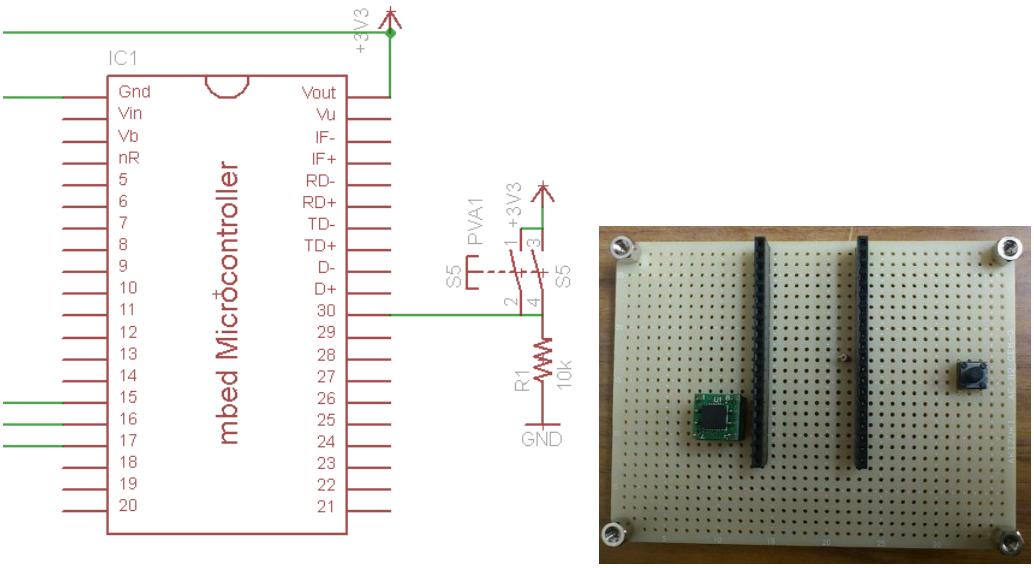


図 3　タクトスイッチ回路部分

4　mbed で LED 駆動

外付け LED を駆動する．ここでは p29 に接続する．

- LED の極性はテスタで確認すること．
- LED に直列に接続する抵抗（制限抵抗）はここでは 1kΩ としているが，電源電圧が変わる場合等もあるので計算できなければならない．抵抗値の計算方法は下記リンクなどに記載されているので自習する．大体 1mA～10mA 流れるようにする．
  http://www.rank-a.com/html/led.html
  http://www.ops.dti.ne.jp/~ishijima/sei/letselec/letselec11.htm
- 抵抗のカラーコードは読めるようになるとよい．

[課題3]　タクトスイッチを押すと LED が光るようにする

[課題4]　割込みプログラミングについて自習し，割込みを用いたプログラムに改変する．スイッチを押すたびに LED の状態が切り替わるようにする．

[課題5]　LED を流れる電流を計測する（抵抗間の電圧を計測し，オームの法則により算出する）．その値は前述の計算方法で求めた予想と比較して妥当か．

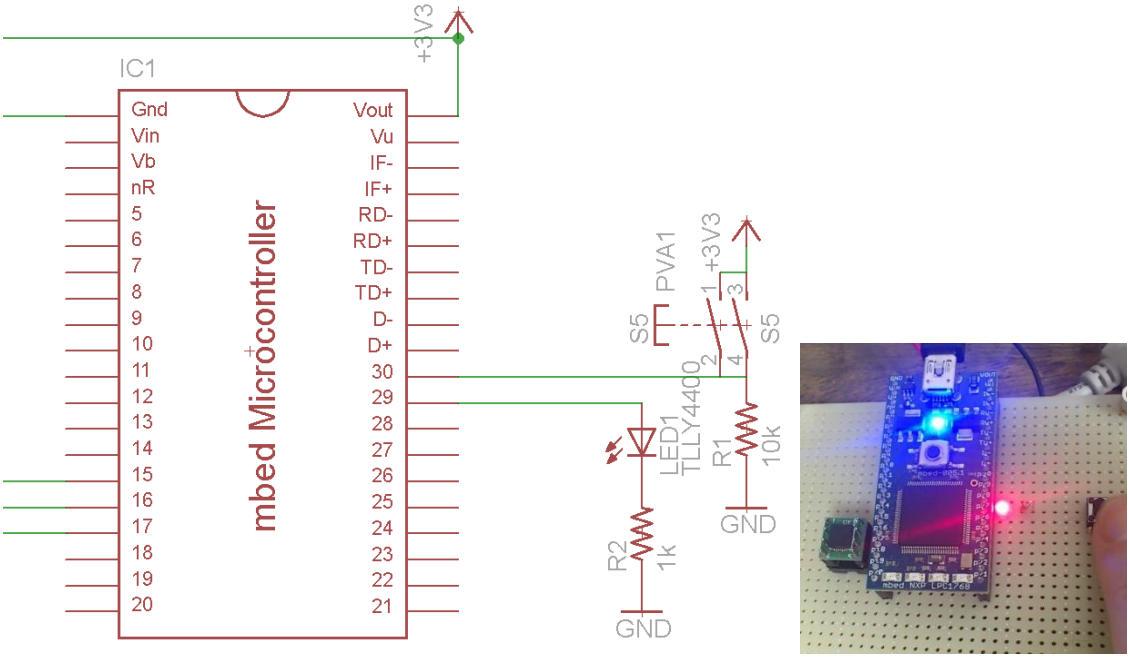


図 4　LED 回路（タクトスイッチを押すと光る）

5　mbed で RC サーボモータ駆動（PWM）

p21 から p26 までが PWM 出力可能なピンである．PWM 出力を利用して RC サーボモータを動かすことが出来る．ここでは RC サーボモータの信号線を p21 につなげる．
RC サーボモータの駆動には大電流が必要でこれまでの USB 給電では限界があるため外部電源（5V）をつなげる．RC サーボの電源はここから取る．

PWM については自習すること．RC サーボモータの制御方法については下記で自習する．

- RC サーボモータの制御方法について
  http://berry.sakura.ne.jp/technics/servo_control_p1.html

（サンプルソース）

- PWM 制御のサンプル．
  http://mbed.org/handbook/PwmOut
- RC サーボ用ライブラリ．こちらを使っても良い．
  http://mbed.org/cookbook/Servo

[課題6]　PC からのシリアル通信でサーボモータの姿勢を制御する．例えば 0-9 の数値を送る．PC 側はシリアル通信ソフトで良い．オシロスコープで PWM 信号を観察する．パルス幅，パルス間隔は計算どおりになっているか（オシロスコープで観察と書いてある場合はその画面をオシロの機能でキャプチャし，レポートに添付すること．以降も同様）

mbed のシリアル通信で PC からの入力を待つためには定石として readable 関数を用いる(http://mbed.org/handbook/Serial)．典型的には次のような構造になる．

```
Serial pc(USBTX, USBRX); // tx, rx

int main() {
    char c;
    while (1) {
        if(pc.readable()){
            c = pc.getc();
            switch(c){
                いろいろな条件処理
            }
            pc.printf(送信内容);
        }
        通常の処理
        適切なウエイト（必要なら）
    }
}
```

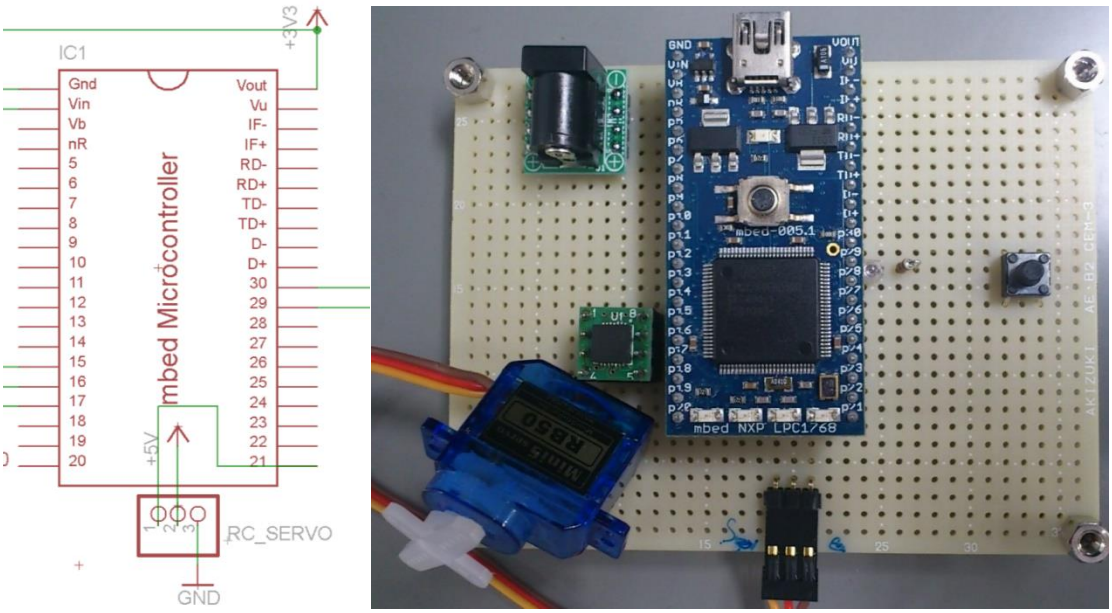


図 5 RC サーボ回路（部分）

6　mbed でアナログ出力（＋オペアンプ）

mbed は 10bit の DA を 1ch，　12bit の AD を 6ch 持っている（DA を使う場合は AD がひとつ減る）．すでに加速度センサを使う際にこの AD を 3ch 使用した．
サンプルソースに基づき DA 出力を行い，出力波形をオシロスコープで確認する．
http://mbed.org/handbook/AnalogOut

オペアンプを使ってイヤフォンを駆動する（図 6）．次のことが行われている．

① DA 信号を可変抵抗（10kΩ程度）で分圧．ボリュームの役割．
② オペアンプ（新日本無線製 NJM4580）の電圧フォロア（ボルテージフォロア）回路でインピーダンス変換．
③ コンデンサ（10uF 程度）によって直流成分をカット．
④ イヤフォンジャックからイヤフォンやスピーカに出力．

- 電解コンデンサの極性に注意．＋側をオペアンプの出力側とする（これはなぜか）
- 可変抵抗の 3 つの端子の使い方はテスタで確認する
- イヤフォンジャックは基板に直付けできるタイプのものもあるがここではケーブルタイプを用いる．圧着端子の使い方の講習も兼ねる．

オペアンプおよび電圧フォロア回路については下記で自習する．
http://picavr.uunyan.com/op_amp.html
http://kaji-lab.jp/ja/index.php?plugin=attach&pcmd=open&file=OpAmpIntro.pdf&refer=people%2Fkaji

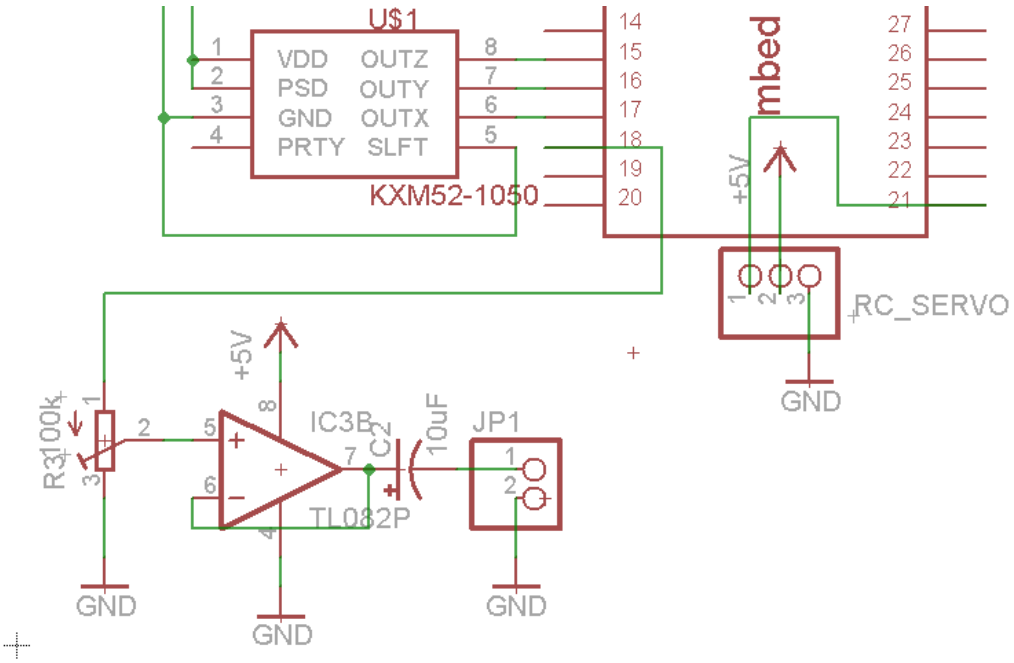
[課題7]　PC からのシリアル通信でイヤフォンの音を制御する．1-8 の数値を送ると「ドレミファソラシド」が鳴るようにする．PC 側はシリアル通信用ソフトで良い．[課題 6] のソースを改変していけば良い．
オシロスコープで波形の変化を確認する．
この課題ではまず，一周期（While ループの一周）が何 us かかるかをオシロスコープによって観察する．オシロスコープで波形を観察するとなめらかな正弦波ではなく量子化された「ギザギザ」の波形が出るはずである．このギザギザ一つ分が一周期に当たる．適当なデジタル出力ピンを用いて DA 変換のたびに状態を変化させて周期を読み取っても良い．
正弦波を出力するにはどうしたらよいか？sin 関数を用いた場合と，あらかじめテーブルを用いた場合を行い，一周期にかかる時間を比較すること．テーブルは例えば次のようなものである．
double mySine[64]={0, 0.099833417, 0.198669331, 0.295520207,…0.999573603};

このテーブルでは，sin(0)から sin(6.3≒2π)までを，0.1 刻みで計算してあらかじめ用意している．すると sin(x)は，fmod 関数を使うことにより，

mySine[(int)(fmod(x,2π)/0.1)]

等とすれば求めることが出来る．この処理の仕組みを説明せよ．

正確な周波数の波を実現するため，timer 関数を使う．プログラム開始時から timer を起動し，現在の時間を使って正確な周波数の音を出力する．例えば f[Hz]の周波数の音を出したい場合，現在の時間が t[s]であれば，sin(2πft)の波を出力すれば良い．



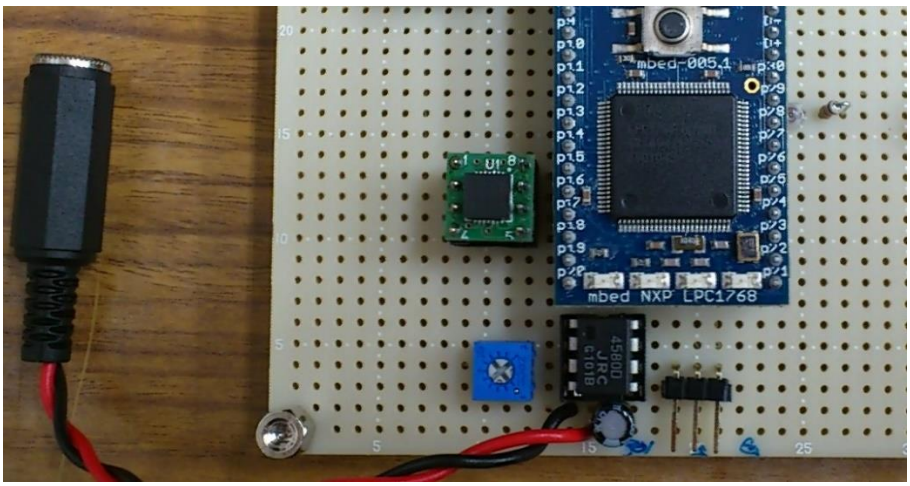


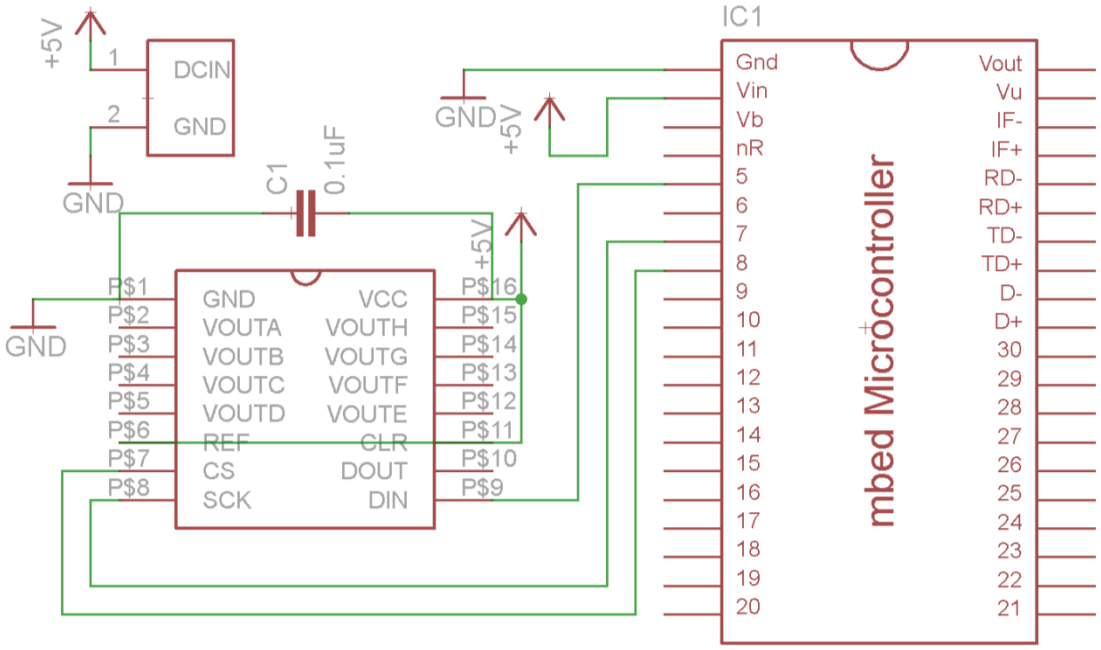
図 6　アナログ出力⇒オペアンプ回路（電圧フォロア）TODO:圧着端子版に変更

7　外付けの DA（SPI 通信によるポートの拡張）

mbed には SPI 通信モジュールが二つ用意されている．これを使うことでポートを拡張することが出来る．ここでは 8ch の D/A ポートを実現する．
SPI 通信，およびシフトレジスタについては自習すること．

外付けの DA 変換器としてオクタル 10bit D/A コンバータ LTC1660CN を用いる．特に**この課題はマニュアルを読む練習を兼ねている**．
http://akizukidenshi.com/catalog/g/gI-02794/
SPI 通信のサンプルソース
http://mbed.org/handbook/SPI
SPI 通信
http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

- DA 変換器の電源とグランドの間に電圧安定化のためのコンデンサを入れる．コンデンサの数値コードは読めるようになる必要がある．「電子工作の素」などを参照して自習．



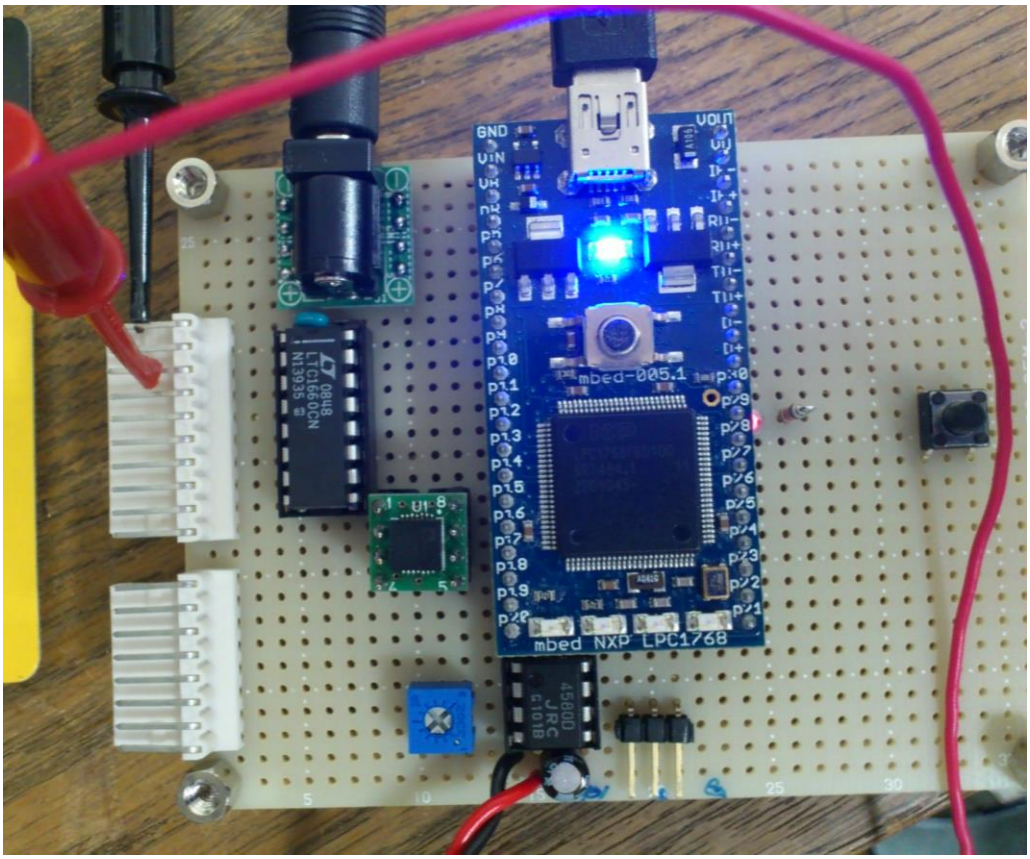


図 7　8ch DA ポート回路（10pin の端子を使い余った 2pin はグランドにしている．またこの写真では AD6ch を 8pin 端子につなぎ，余った 2pin はグランドと 3.3V にしている）

[課題8]　下記のサンプルプログラムを写経して動かし，オシロスコープを使って波形を観察する．つぎにクロック信号とデータ信号を同時に観察する（CS ピンをトリガとする．**オシロスコープのトリガの使い方の練習も兼ねている**ので調べること）．

二つの信号のタイミングはどのような関係にあるか．それらは LTC1660 のデータシートのタイミングチャートと比較して妥当か．

16bit のデータ信号はプログラムとどのような関係にあるか．またそれらは LTC1660 のデータシートと比較して妥当か．

[課題9]　チャンネルを指定して電圧を出力できる関数を作成する．

void DAout(char channel, float voltage)

LTC1660 のデータシートによると，チャンネルを指定するにはどうすれば良いか．また電圧を 0.0 から 5.0 までの float で指定するにはどうすれば良いか．

その上で，8ch それぞれから指定した異なる周波数の正弦波を出力し，オシロスコープで波形を観察する．正確な周波数で出力するために[課題 7] の方法を用いる．

```
#include "mbed.h"

SPI spi(p5, NC, p7);// mosi（出力）, miso（入力）, sclk（クロック）. 今回は入力不要
DigitalOut cs(p8);  //chip selectピン

Serial pc(USBTX, USBRX); // tx, rx

int main() {
    int t=0;
    short spiData,DA;
    char channel=0;
    // SPIのセットアップ. 16bit, 立ち上がりエッジ, 1MHz clock
    spi.format(16,0);
    spi.frequency(1000000);

    while (1) {
        t++;
        //矩形波
        if (t%2==0) {
            DA=0x3FF;
        } else {
            DA=0;
        }
        cs = 0; //clockを有効化
        //データの整形. マニュアル参照のこと
        spiData = (((channel&0x07)+1)<<12) | ((DA&0x3FF)<<2);
        spi.write(spiData);
        cs = 1; //clock を無効化し, データをロード
        wait_ms(1);
    }
}
```

図 8 8ch D/A 変換器 LTC1660 を動かすサンプル．タイマは使っていない

8　DC モータの制御

研究室標準の DC モータである MAXON 社製 DC モータ(RE25, 10W, 118746)を駆動する．使用する DC モータにはエンコーダ(HEDS5540，500CPR (CPR: count per revolution))が付いており，位置を計測しながら制御する事ができる．出力には H ブリッジ IC の BD6222HFP を使用する．

エンコーダについては例えば下記で自習すること．特に 4 逓倍モードについて理解する．
http://ednjapan.com/edn/articles/1203/16/news012_2.html
http://www.geocities.jp/horie_ryu/page010.html
エンコーダ用ライブラリは下記のものを用いる
QEI　http://mbed.org/cookbook/QEI

図 9 を参考に Vcc（5V），GND,チャンネル A,B を接続する．なお各出力チャンネルは 5V 電源と 3.3k オームの抵抗で接続すること（これをプルアップ抵抗という：自習すること）．これをしなくても動くが高速回転の際に不具合が生じることが多い．

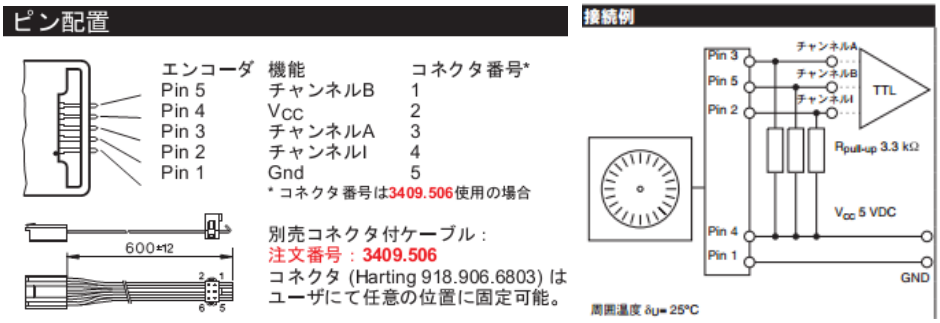


図 9 エンコーダピン配置（HEDS5540 のマニュアルより）

ソースコードは次のようになる(p29,30 はすでに使っていると思われるので別のピンにする)．　500 パルス／一周のエンコーダを 4 逓倍モードで利用するよう設定している．

```
#include "mbed.h"
#include "QEI.h"

Serial pc(USBTX, USBRX);
//Use X4 encoding.
QEI wheel(p29, p30, NC, 2000, QEI::X4_ENCODING);
//Use X2 encoding by default.
//QEI wheel (p29, p30, NC, 500);


int main()
{
    while(1) {
        pc.printf("Pulses is: %i\n", wheel.getPulses());
        wait(0.5);
    }
}
```

図 10　エンコーダを読むサンプルプログラム

[課題10]　　エンコーダ用ライブラリを用い，モータの回転角度を計測，シリアル通信によって PC 画面に表示し続けるようにする．角度はラジアンに変換して表示すること．正転・逆転方向に一周させて±2πの数値が表示されるか確認する．

（参考）mbed マイコンによるエンコーダ値計測は高速な回転になるとカウントの取りこぼしが発生し，ずれを生じることがある．特に複数のモータを用いる場合にはこの影響は大きい．研究で使用する場合はカウント専用 IC(LS7366 等)を用いるなどの対策が望ましい．

H ブリッジ回路については例えば下記で自習する．
http://www.picfun.com/motor03.html
モータ駆動（H ブリッジ駆動）用ライブラリは例えば http://mbed.org/cookbook/Motor があるが，今回は PWM 制御により明示的に制御してみる．
今回使用する BD6222HFP の足は 1.27mm ピッチなので通常の基板ピッチ（2.54mm）に変換する必要がある（図 11）．ピッチ変換基板を使う際は BD6222 の放熱面が各端子に接触しないようにすること（放熱面自体が GND に接続されている）

図 11　　BD6222HFP のピッチ変換

BD6222HFP の内部モジュールと端子名，真理値表は図 12 のとおりである．　Fin,Rin 番ピンに mbed からの入力を加えることで，正転，逆転，ブレーキ，ストップを実現する．

電源と GND の間に 10uF のコンデンサを挟む．コンデンサは電源電圧の 2 倍以上の耐圧のあるものを選ぶこと． 2 つの電源（mbed 用，モータ用）を使うことになるので，GND 同士を接続すること．

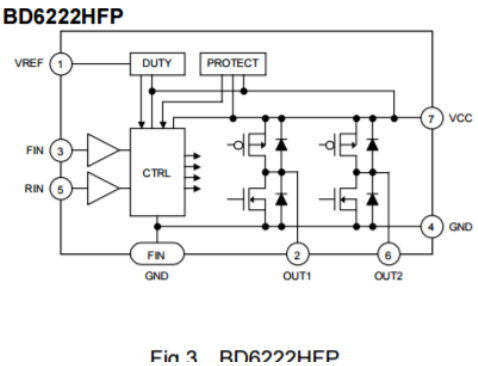


Fig 3 BD6222HFP

Table 2 BD6222HFP

番号	端子名	機能
1	VREF	VREF 可変電圧入力
2	OUT1	出力端子
3	FIN	制御入力(正)
4	GND	GND
5	RIN	制御入力(逆)
6	OUT2	出力端子
7	VCC	電源
FIN	GND	GND

Table 5 真理値表

	FIN	RIN	VREF	OUT1	OUT2	動作(OPERATION)
a	L	L	X	Hi-Z*	Hi-Z*	スタンバイ(空転)
b	H	L	VCC	H	L	正転(OUT1→OUT2)
c	L	H	VCC	L	H	逆転(OUT2→OUT1)
d	H	H	X	L	L	ブレーキ(停止)
e	PWM	L	VCC	H	$\overline{\text{PWM}}$	正転(PWM 制御 A)
f	L	PWM	VCC	$\overline{\text{PWM}}$	H	逆転(PWM 制御 A)
g	H	PWM	VCC	$\overline{\text{PWM}}$	L	正転(PWM 制御 B)
h	PWM	H	VCC	L	$\overline{\text{PWM}}$	逆転(PWM 制御 B)
i	H	L	Option	H	$\overline{\text{PWM}}$	正転(VREF 制御)
j	L	H	Option	$\overline{\text{PWM}}$	H	逆転(VREF 制御)

* Hi-Z とは、出力トランジスタが OFF の状態です。メカ・リレーとは異なり、ダイオードが接続された状態になっていますので、ご注意ください。
X : Don't care

図 12 内部モジュールと端子名，真理値表(BD6222 マニュアルより)

モータを駆動する部分の電源（図の Vcc）は実験用の安定化電源を用いる．この電源は電圧の設定，および電流のリミットの設定が可能である（**CVCC 電源**: Constant Voltage Constant Current power supply）．一般的な働きとして，「制限電流未満のときは定電圧電源として働き，それを上回ったときは定電流電源になる」と説明される．主な用途としては電流のリミットを低めに設定しておくことでショート等による回路の損傷を防ぐ．また電流リミット機能を積極的に使うことで，定電流源として使うこともある．

本実験では当初は電源電圧 12V，電流リミット 0.3A 程度に設定する．その後回路上の問題がなければ電流リミットを 1.5A 程度

に上げる．

CVCC 電源については例えば下記で自習する．
http://okwave.jp/qa4105361.html
http://www.kikusui.co.jp/knowledgeplaza/powersupply1/powersupply1_j.html

[課題11]　H ブリッジ回路におけるストップとブレーキの違いを，原理を含めて説明せよ．

このファンクションテーブルと，Fin と Rin に PWM 端子をつなげる事とを合わせて考えると，大まかに次のような関数によって出力を制御できると考えられる．

```
PwmOut IN1(p23);
PwmOut IN2(p24);

void motor(float p)
{
    if(p>0){            //IN1 を L，IN2 を定期的に H にして正転
        IN1 = 0;
        IN2 = p;
    }else{              //IN2 を L，IN1 を定期的に H にして正転
        IN2=0;
        IN1=-p;         //マイナスを付けることに注意
    }
}
```

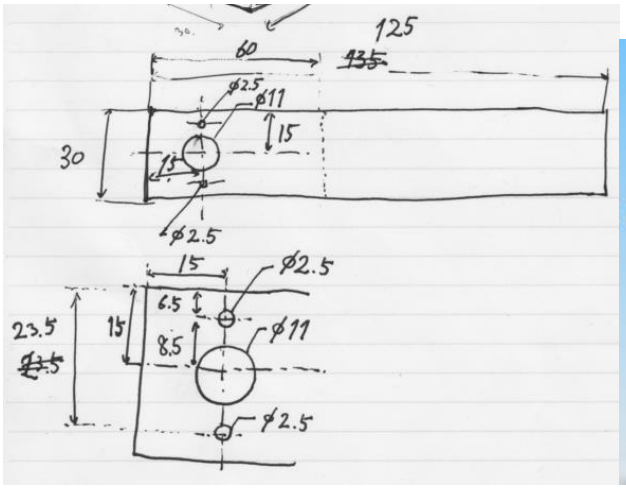
[課題12]　シリアル通信で指令を送り，例えば f(orward)，b(ackward)を入力すると正転／逆転が切り替わり，h(igh)，l(ow)を入力するたびに速度が変更されるようにせよ．PWM の制御周期を 0.02s(50Hz，デフォルト値)，1ms（1kHz），50us(20kHz)，と変化させ，振る舞いの変化，モータから発する音を観察し，考察せよ．20kHz に設定し，２つの PWM 出力が予想通りの出力になっているかどうかをオシロスコープで 2ch 同時に計測して確認せよ（波形は画面を記録）．　この後は 20kHz の設定で用いる．

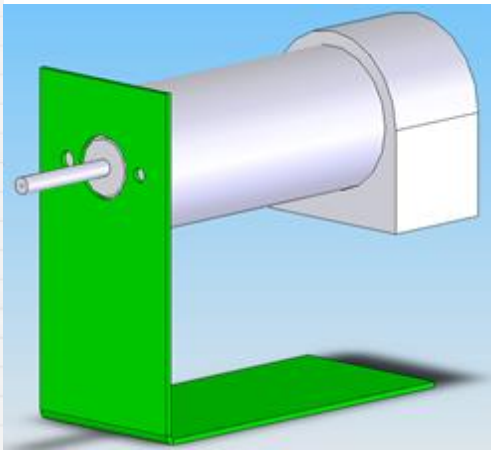
[課題13]　一方向に回転させ続けている時，電流はどの程度流れているか．またこのモータの回転を手で止めた時（気をつける！），電流はどの程度流れるか．この違いはなぜ生じるか調べ，説明せよ（キーワード：DC モータ，逆起電力）．

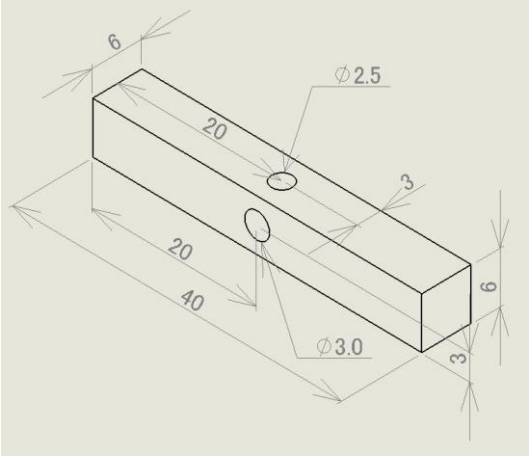
[課題14]　　アルミ加工により，DC モータの軸に取手をつける．（この部分は下記に従い個別に指導します.）

アルミ加工の講習内容

1. ケガキの仕方
    ① ノギスの使い方
    ② ハイトゲージの使い方
    ③ ポンチの打ち方
2. バンドソーの使い方
3. ドリルの使い方
    ① 台座の緩め方，上下させ方
    ② ドリルの交換方法
    ③ 万力による材料の固定
    ④ 穴あけ，小さい穴に関するバリとり
    ⑤ 特殊ドリル１：ステップドリルの使い方
    ⑥ 特殊ドリル２：バリ取り用ドリルの使い方
4. タップの切り方
5. 後片付け：掃除の仕方（工具周りを刷毛で．床を掃除機で）







これ以降はアルミ台座を机にガムテープで固定する（図 13）.

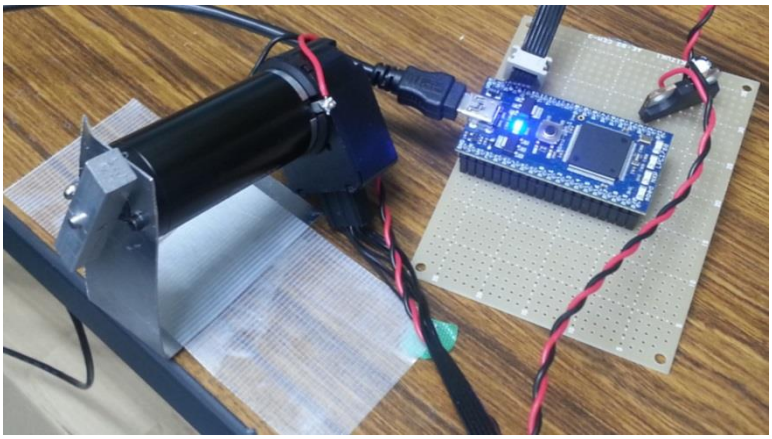


図 13　ノブ付きモータと制御基板

[課題15]　　P 制御により目的の位置（初期位置で良い）に向かうようにし，電源を付け，取手を手で回転させてみる．どうなるか．ゲインを上げ，なるべく早く戻るようにする．

[課題16]　　PD 制御により目的の位置（初期位置で良い）に向かうようにし，電源を付け，取手を手で回転させてみる．どうなるか．また PC から位置指令を送り（例えば r(ight), l(eft)），キーボードを押すたびに移動するようにせよ．プログラムの構造は下記を参照すること．繰返しタイマ割り込み ticker を使っている点に注意．

```
Ticker myticker;

void motor(float p){
    PWM 指令値の変更によりモータ出力
}

void PDcontrol(){//1ms ごとに呼び出される関数
    指令値に基づいた PD 制御.
    現在の位置を取得
    現在の位置と過去の位置から速度を求める
    目標位置との差分からモータへの出力を決定する
    motor(p); //モータ制御関数を呼ぶ
}

int main() {
    char c;
    myticker.attach(&PDcontrol, 0.001); //繰り返しタイマー割り込みを 1ms に設定.
    while (1) {
        if(pc.readable()){
            c = pc.getc();
            switch(c){
                case 'r':
                    指令値変更
                    break;
                case 'l':
                    …
            }
        }
    }
}
```

[課題17]　DC モータの制御により「カチカチ感」のあるロータリースイッチを擬似的に再現する．カチカチの正体を周期的な抵抗感とすれば，これは PD 制御の目標位置を周期的に設定することで実現できると考えられる．

9　mbed でアナログ入力・データロギング・Processing の導入

mbed にはアナログ入力端子が 6pin あり，簡単なプロトタイピングに利用できる．すでに加速度センサの３軸の値を読み取ることまで別課題で習得した．本章では実用的なデータロギングの方法を考える．

ここからは PC 側のプログラミング環境として Processing を用いる（https://processing.org/）．研究室の実験環境でも多用するため自習のこと．

[課題18]　　下記のサンプルプログラムを写経して mbed に接続された加速度センサの電圧をシリアル通信にて PC で表示する．

その後，Processing のプログラムを変更し，データを CSV ファイル形式で保存する．PrintWriter クラスを使用する．Processing のヘルプを参照．キーボード関係の関数も調べ，どれかキーを押したら記録を開始し，100 回記録したら保存して終了するようにする．

```
#include "mbed.h"

Serial pc(USBTX, USBRX); // tx, rx

AnalogIn gX(p15);
AnalogIn gY(p16);
AnalogIn gZ(p17);

int main() {
    unsigned short sX,sY,sZ;
    double t=0;

    pc.baud(921600);

    while (1) {
        //read AD values
        sX=gX.read_u16();
        sY=gY.read_u16();
        sZ=gZ.read_u16();
        //prepare for sending. pack to 6 bytes
        pc.putc(sX>>8); //upper 8bit
        pc.putc(sX&0xFF); //lower 8bit
        pc.putc(sY>>8); //upper 8bit
        pc.putc(sY&0xFF); //lower 8bit
        pc.putc(sZ>>8); //upper 8bit
        pc.putc(sZ&0xFF); //lower 8bit
        //wait for 20 ms
        wait_ms(20);
    }
}
```

図 14　加速度センササンプルプログラム（mbed 側）

```
import processing.serial.*;

Serial myPort;
String COM_PORT="COM6"; //COM番号. 変更する
int time=0;

void setup() {  // setup() runs once
  size(256, 256);
  frameRate(60);
  //シリアルポートに接続. 速度は921.6kbpsに設定
  myPort = new Serial(this, COM_PORT, 921600);
}

void draw() {  // draw() loops forever, until stopped
  int ax=0,ay=0,az=0;
  color cx = color(255,0,0);
  color cy = color(0,255,0);
  color cz = color(0,0,255);

  //シリアル通信
  if(myPort.available()>=6){
    ax = myPort.read() * 256;
    ax = ax + myPort.read();
    ay = myPort.read() * 256;
    ay = ay + myPort.read();
    az = myPort.read() * 256;
    az = az + myPort.read();
    println(ax,",",ay,",",az);
  }

  //描画
  time = (time+1)%256;
  background(0,0,0);//背景消去
  fill(cx);ellipse(time,ax/256, 5, 5);//x座標
  fill(cy);ellipse(time,ay/256, 5, 5);//y座標
  fill(cz);ellipse(time,az/256, 5, 5);//z座標
}
```

図 15　加速度センサ値読み出しサンプルプログラム（Processing 側）

上記サンプルでは Processing 中の Draw 関数中でシリアル通信を行った．しかし Draw 関数は通常 60fps で読み出される関数である（変更は可能）．より高頻度ないし正確な通信をする場合には限界がある．

Processing のシリアルライブラリはこうした場合に有用な仕組みをもっているので利用する．まず serialEvent 関数を定義する．また bufferUntil 関数を用いて，特定の文字が来た時にだけ serialEvent が発生するようにする．これらは関数のヘルプで使い方を調べること．この仕組を使うためには，データの末尾に特定の文字（以後フッタと呼ぶ）を指定する必要があるため，一回あたりの送受信量は 1byte 増える（図 16）．

またこの時，フッタを示す数値がデータ中に偶然現れることを避ける必要がある．今回の AD 変換は 1ch あたり 12bit であり，16bit に変換されて出力されているので，これを 4bit 右シフトし，上位，下位 6bit ずつ送信すれば，データはすべて 0-63 の範囲に収まる．フッタの数値を例えば 0xFF(8bit すべて 1)とすれば，データと指定文字が重なる心配はなくなる．

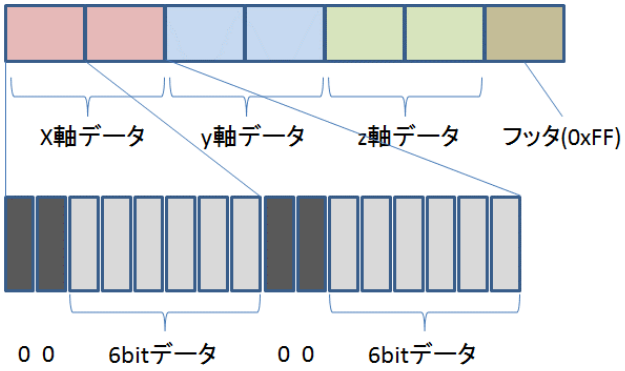


図 16　データ列とフッタの例

複数バイトのデータを読み出すに際して，データの「切れ目」が正しいことを保証する事が重要である． mbed マイコンからのデータは，mbed マイコンを起動するタイミング，PC のプログラムを起動するタイミング，などにより，Y 軸データから始まっているかもしれないし，Z 軸データの 2 バイト目から始まっている可能性もある．この問題は PC のプログラム側でシリアルバッファをクリアしても完全には解決しない（それはなぜか）．
この問題を解決するためには，第一に PC 側からデータ送信のリクエスト信号を mbed 側に送るという方法が考えられる．これが最も標準的であり推奨されるが，次のように PC 側のプログラムでフッタを明示的に探索するという方法もある．これによって初回の読み出し時にデータの切れ目が正しく認識される．

```
void serialEvent(Serial mp){
    ax= mp.read()*64;          //2 バイト×3 のデータ読み出し
    ax+= mp.read();
    …
    az+= mp.read();
    while(mp.read()!=0xFF);  //明示的なフッタの探索
    …  //ファイル保存などの処理
}
```

また bufferUntil による割り込み関数の定義は，こうした割り込み関数の常として，serialEvent 関数がいつ呼び出されるかわからないために再現性のないバグを生むことがある．例えば serialEvent 関数は，bufferUntil が呼ばれた次の瞬間に（まだ Processing の setup 関数が終わらないうちに）呼ばれるかもしれない．例えばこれによって，まだ書き込むファイルが準備されていない状態でファイル書き込みが生じる等のバグが生じる．こうしたバグは典型的に「何回かに一回」発生する．逆に再現性のないバグに対してアドホックな解決は絶対に取らないこと．

[課題19]　加速度センサの３軸の値を 1kHz で記録したい．上記の解説を参考に，mbed 側から 1kHz（1ms 周期）で AD 変換，シリアル通信を行い，データを記録せよ．mbed 側のデータ取得間隔を正確にするため，ticker による繰り返しタイマ割り込みによる AD 変換を行うこと．また 1ms 周期で計測できていることを確認するため，適当なデジタル出力端子の状態を ticker 関数中で毎回反転させ，オシロスコープで観察すること．

さらに，解説を参考に，明示的なフッタの探索を行う場合と行わないで 7 バイトずつ読み出す場合で結果が変わることを確認すること．

上記は Processing 側の Draw 関数の周期に依存しない計測の方法だが，そもそもシリアル通信を USB2.0 経由で行なっているため，通信は実際には 500Hz 程度の頻度で行われている．このためマイコンの段階では 1kHz でサンプリングができているとしても，PC での取得時刻は正確な 1kHz ループではない．PC を入れた制御ループを構成する際には問題となる．高速な制御ループはマイコン内で行うべきである．

振動子の応答特性計測などではより速い計測が必要となるが，リアルタイム性が必要でない場合には，mbed 上のメモリに保存し，あとで USB メモリ上のデータとして PC から取得することもよく行われる．

また mbed の通常のシリアル通信関数では一回に PC に送れるデータ量には限界があり，大きなデータを送信するには分割して送る必要がある．

[課題20]　mbed(LPC1768)の AD 入力はノイズが大きいという問題が知られている (https://developer.mbed.org/users/chris/notebook/Getting-best-ADC-performance/). これがどの程度かを観察する．加速度センサを外し，同じポートを安定化電源に接続し，1.65V を加える．これまでの方法で電圧を csv ファイル形式で取得し，グラフにしてみる．グラフの縦軸は電圧[V]，横軸は時刻[秒]とすること．ノイズは何 V 程度か．

また上記引用ページには対策として「使用していない AD ポートをすべて DigitalOut ピンに設定する」「AD ポートと GND の間に 0.001uF のコンデンサを入れる」という対策が述べられている．これらの効果を確認する．

[課題21]　（Processing 課題）複数の PC プログラムに個別の役割を与え，プログラム間の通信でデータのやり取りをする場合がある．例えばセンサ情報を常に取得する Processing プログラムと，3D 視覚環境を構築する Unity といった形である．こうした場合にはプロセス間通信を行う（なお Unity 内部でもシリアル通信のプログラムを書くことは出来る）．

Processing の Network ライブラリ中の Client，Server サンプルを使い，２つの

Processing プログラム間で通信を行うこと．送信側が加速度センサの値を読み，受信側でその値に対応して画面の色を変化させる(R,G,B を加速度の X,Y,Z の値に対応させる)．同時に 3 つの値を送受信するため工夫が必要だがシリアル通信の場合と比較して類推すれば容易に理解できる．

[課題22]　　(最終課題) 二人一組で, [課題 16] を元に，2 台の PC 間での力伝送を行う．Processing の Network 機能を用いて相手のモータの姿勢を目標値として自分のモータを制御する．すると 2 つのモータは常に同じ姿勢を取るようになるため，結果として力を伝達することが出来る．これを対称型のバイラテラル制御とよぶので自習のこと．通信部分については神戸大学西田先生の下記のページ等を参照すると良い．この枠組で何か面白いことをする．最終課題は研究室内で発表会を行う．
http://www2.kobe-u.ac.jp/~tnishida/course/2012/programming/ServerClient.pdf

10 その他研究室内 TIPS

- LPC1768 は最も標準的な mbed マイコンであるが価格が高く，大抵の研究室内用途にオーバースペックであり，かつ大きい．このため，小型かつ多数使う場合には LPC1114 を用いる（書き込みツール LPC1114FN28）．特に小型化したい場合は表面実装版を用いる．
- LPC1768 と同等の性能が必要で，ピン数はそれほど必要ない場合には同一コアの mbed マイコンを検討する．Bluetooth 通信が必要な場合は通信機能付きの mbed マイコンを検討する．
- これらはウエアラブル化，ポータブル化の際に必要となるので先輩に相談し，プロジェクトにとって最適なマイコンを用いる．
- ウエアラブル化の際には電源周りも重要となる．電池関連の TIPS も先輩に相談．ケーブル有りと無しはデモの説得力に大きな差が出る．
- コンパイラは当座は標準の web コンパイラで問題ないが，web は比較的頻繁に落ちる．買っているものがあるので研究室内開発環境については先輩に相談．
- 本格的な小型化や本格的な高速化が必要な場合 mbed のシリーズに拘る必要はない．

11 選択課題

以下は 2017 年度は行わない参考資料.

一人 1 課題を行う．すべての課題がある種のインタフェース拡張であり，最終的に PC との通信を行う．PC 側は Processing のプログラムを動かし，ユーザからの何らかの入力を反映して mbed 側も変化するようにする．
説明は完全では無い（回路図も未検証で間違っているかもしれない）ので，必要であれば部品のマニュアルを読む.

11.1　7 セグメント LED を動かす（初級～中級）

7 セグメント LED(シャープ製 GL9A040G)で数字を表示させる．これは数字を表示するための 8 つの LED の集合であり，アノードコモン，カソードコモンの 2 種類がある(図 17).
今回使用するのはアノードコモンであり，「吸い込み」電流によって点灯させることを前提としている.

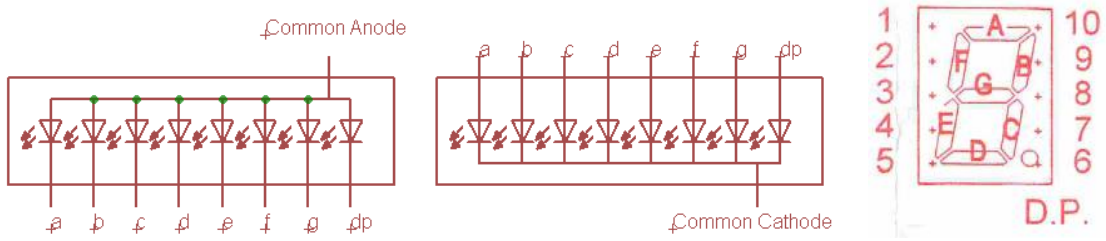


図 17　7 セグメント LED の内部構造.（左）アノードコモン，（中）カソードコモン，（右）LED の配置

8 つの LED を光らせる必要があるために 8 本のピンが必要である．そこでポート拡張のためシフトレジスタ（今回は 74HC164）を用いる．SPI 通信を使ってデータを送る．74HC164 にはラッチが無いため，データ移動の最中にも LED が点灯することに注意．ラッチを持つ IC もある（例えば 74HC595．またすでに扱った D/A 変換 IC も）.

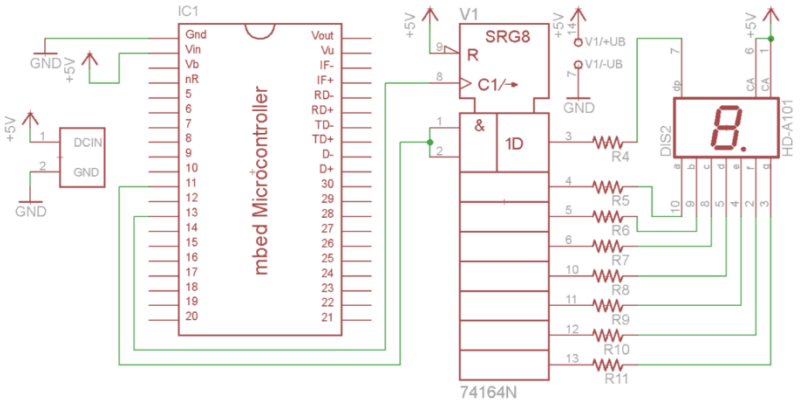


図 18 7 セグメント LED 点灯回路

- 回路保護のため 74HC164 と 7 セグメント LED は IC ソケットで半田付けすること．
- 74HC シリーズは入出力ピン一つあたり最大 4mA までしか電流を流すことができない．このため LED 用の抵抗 R2-9 はやや大きめのものを使う（計算すること）．

[課題23]　PC から送られてきた 0～9 の数値を表示する．さらにシフトレジスタを二段連結して二つの 7 セグメント LED を駆動し，PC から送られてきた文字を 16 進数で表示する．（シフトレジスタの連結は前段の最後の出力を次段の入力とすれば良い）．PC 上でも Processing を用いて数値をグラフィカルに表示する．

11.2　通信機能を持つセンサモジュールの使用 1（中級）

これまでに扱ったセンサモジュールのほとんどはアナログ電圧出力をするものであるが，多くが SPI 通信または I2C 通信機能を備えている．ここではその例としてボッシュ社の加速度センサ BMA180 を取り上げる．この加速度センサはこれまで使ってきたものよりもダイナミックレンジが広く，例えばバチを叩いた時の衝撃を計測することもできる．

http://www.switch-science.com/products/detail.php?product_id=386

このモジュールと mbed を接続する(BMA180 用ライブラリは非常にたくさん見つかる). さらに mbed の DA 出力にオーディオ用パワーオペアンプ(LM675 等)を接続し，振動子 ForceReactor を駆動する．LM675 の回路はマニュアルを参照すること．

[課題24]　ここでは HACHIStick（あるいは製品では Dayon）の模擬を行なう．加速度センサモジュールと振動子を棒に取り付け，ある閾値以上の衝撃が加わった時に振動子が駆動されるようにすることで，叩いた対象の材質感を表現する．ハウリングを防ぐにはどうしたらよいか．PC 画面にシンバル等の楽器を表示し，画面を実際に叩いた際に楽器も揺れ動き，音が出るようにする．

11.3　通信機能を持つセンサモジュールの使用 2（上級）

同様のセンサモジュールとしていわゆる 9 軸センサを扱う．使用する IC は MPU-9150 であり，3 軸加速度＋3 軸ジャイロ＋3 軸コンパスを内蔵している．方位や角速度を計測できるため，加速度センサでは出来ない応用が可能となる．

https://strawberry-linux.com/catalog/items?code=12150

[課題25]　ここでは簡便なモーションキャプチャ装置を作る．ユーザの頭頂部にセンサモジュールを取り付け，加速度センサによって重力方向の傾きを，コンパスによって水平方向の回転を取得する．この値に基づき頭の姿勢を計算し，PC 画面中に表示した 3D の人の頭を同様に動かす．またジャイロセンサ等も用い何らかのコマンド入力出来るようにする．

11.4　超音波センサの使用（中級）

[課題 4] において，ピンの立ち上がりや立下りを検出する割込みプログラミングを行ったが，これを使う必要がある場面として超音波センサを取り上げる．ここでは浅草ギ研製超音波距離センサーPING)))を用いる．このセンサは距離情報をピン電圧が上がっている時間によって表しているためピンの監視が必要であり，割込みプログラミングが適している．

http://mbed.org/users/rosienej/code/Ping/

[課題26]　これを 2 つ使い，テルミンっぽいものを作る．音出力用アンプは LM675 などを用い，外部電源 12V で駆動すること．波形の切り替え（正弦波，矩形波，三角波など．テルミン自体についても調べること）は PC 側から行なう．他に PC ができることを考える．

11.5　超音波センサの作成（上級）

市販の超音波センサは応答時間が遅く，入力インタフェースとして使用するのは難しい．ここでは超音波トランスデューサを 2 つ用い，発信－受信を行い，受信の時間差を用いて計測する．　オペアンプによる増幅回路が必要となる．

[課題27]　超音波センサを自作し，回路のフィルタ特性について考察せよ．

11.6　外付けの AD を使う（中級～上級）

mbed には 6ch の AD 変換が用意されているが，それでは足りない場合もある．また内蔵 AD はノイズが大きい．そこで SPI 通信で外付けの AD を増やす．

12 ビット 8ch AD コンバータ MCP3208-CI/P を使う．

http://akizukidenshi.com/catalog/g/gI-00238/

↓サンプルソースの例

http://mbed.org/cookbook/SPI-communication-with-external-ADC-MCP3

↓専用クラスの例

http://mbed.org/users/Kemp/libraries/mcp3208/lsnbqh/docs/

この IC 1 つで 8ch のチャンネルを追加できる．さらに追加する場合には，SPI 通信のデータ，クロック線は共通にし，CS 信号を追加すれば良い．

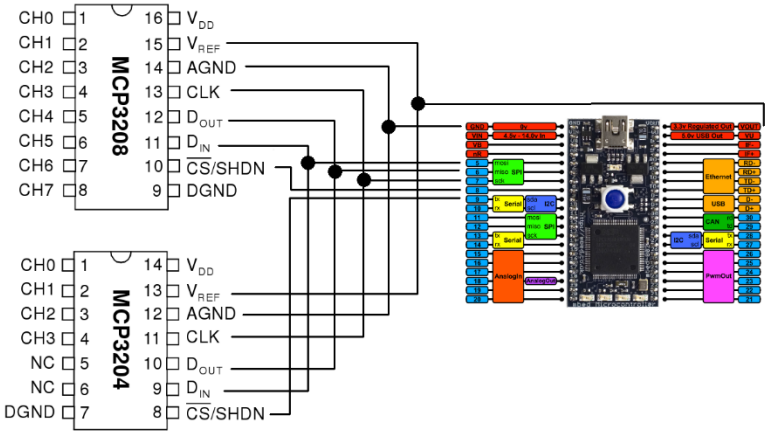


図 19　複数個の AD コンバータをつなぐ例．Vdd は 5V に接続．CS によって IC を選ぶ．http://mbed.org/cookbook/SPI-communication-with-external-ADC-MCP3 より

[課題28]　　フィルム状力センサを 16 枚用い，4x4 の分布型触覚センサを作る．力分布を Processing で画像表示する．回路は図 21 の曲げセンサ回路と同じで良い．

11.7　リニアアクチュエータの駆動（中級～上級）

研究室標準のリニアアクチュエータである Firgelli 社製 PQ12-P を駆動する．

AD によるポテンショメータの読み取り，H ブリッジ回路による制御＋PWM を行う．ポテンショ（可変抵抗）は分圧して A/D ポートで読み取る（図 20）．

PQ12-P http://www.firgelli.com/

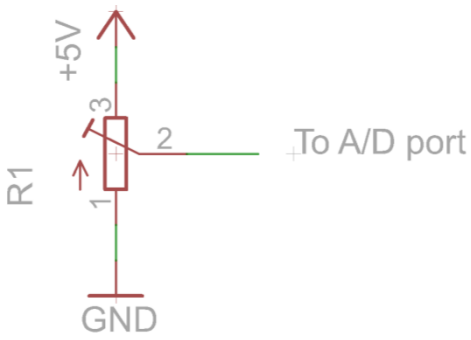


図 20　ポテンショメータ回路

[課題29]　　リニアモータの先端にフィルム状力センサを付け，加わる力も測定する．指で押した時に，リニアモータの振る舞いによって「柔らかさ」を再現する．またこの変形の様子を何らかの形で Processing で表示する．

11.8　Wii コントローラと Bluetooth 通信

mbed は USB のホスト機能を持つため, USB ドングルタイプの Bluetooth 通信モジュールを用いることが出来る．これを利用して，Wii リモコンを入力としてモータの回転角度を変更する．

http://mbed.org/cookbook/USBBluetoothHost

http://mbed.org/users/jksoft/notebook/BlueUSB_Pep/

http://d.hatena.ne.jp/o2mana/20101114/1289725665

http://luuuuga.seesaa.net/article/302322076.html

（これは今は Bluetooth モジュール付きの mbed を使うことを検討すべきである）

11.9　振動提示型データグローブ（初級～中級）

軍手に 5 つの曲げセンサ＋5 つの振動モータを取り付け，簡単なデータグローブとする．曲げセンサの値がある閾値を超えたら振動モータを駆動させることで，バーチャルな物体を触る状況を実現する．

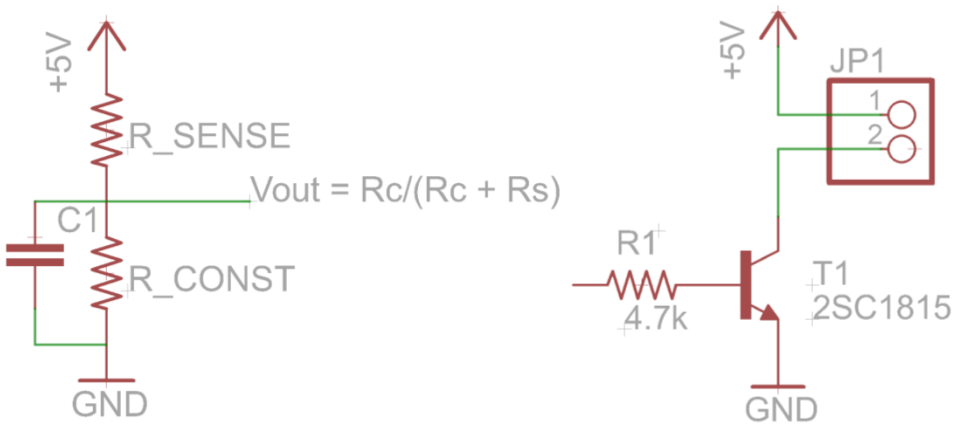


図 21　曲げセンサ回路（簡易版）と振動モータ回路

- 曲げセンサは曲がりが大きくなるほど抵抗値が下がる．この時図のように固定抵抗(Rc)との間で分圧回路を組むと，簡易的に曲げに応じた電圧出力が得られる．この出力をそのまま A/D 入力に入れれば良い．固定抵抗の値は実験的に適したものを用いるが，例えば 10kΩ～100kΩ程度である．またノイズを低減するために固定抵抗と並列に小さなコンデンサ（例えば 0.01uF～0.1uF）を入れる場合が多い．
- 振動モータはトランジスタによって駆動する．PWM 出力可能な P21～P26 から出力する．抵抗値は適当なので変える必要があるかもしれない．トランジスタ（ここでは NPN トランジスタ）について知らなければ自習のこと．

[課題30]　　指の曲げに応じて振動の強さを変える．5 つの振動モータそれぞれが PWM 出力可能であるので，例えば曲げが大きくなるにつれて振動の強さが大きくなるという制御が可能である．以上は mbed で完結するが，PC 側（Processing）で何が出来るかも検討する．最低限，状態の表示は行う．

11.10　USBAUDIO デバイスを作る（中級～上級）

PC 側からは普通の USB-Audio デバイスとして認識される USBAUDIO デバイスを作成する。新たに外付けの USB コネクタを取り付ける必要がある。

http://mbed.org/handbook/USBAudio

12bit、2ch の D/A コンバータ MCP4922-E/P を用いるなどして PC 接続の USB オーディオデバイスを作る。アンプは LM675 などを用い、外部電源 12V で駆動すること。http://akizukidenshi.com/catalog/g/gI-02090/

12 参考文献

文中で取り上げたもの以外で参考になるものを挙げます．大体研究室においてあります．

[1] 「オペアンプ基礎回路再入門」
標準的な教科書． オペアンプは一度ちゃんと勉強する必要があります．

[2] 「すぐ使える！オペアンプ回路図」

[3] トランジスタ技術 SPECIAL「OP アンプによる実用回路設計」

[4] トランジスタ技術 SPECIAL「OP アンプ IC 活用ノート」

[5] トランジスタ技術 SPECIAL「徹底図解 ディジタル・オシロスコープ活用ノート」
CQ 出版の SPECIAL シリーズ．内容は標準的で，入門を卒業した人の次のステップ用で，長く座右に置けるタイプです．これ以外にも研究室に置いてあるトランジスタ技術 SPECIAL はどれも良い参考書．

[6] トランジスタ技術
研究室で毎月購読しています．だんだん読めるようになるので分かるところだけ拾い読みしましょう．

[7] PIC とセンサの電子工作
PIC を使った電子工作本ですが，センサ関係の解説が豊富．

[8] 日経エレクトロニクス，日経サイエンス
研究室で毎月購読しています．意外にこの分野の人はみんな読んでいます．興味を持てるところを拾い読みしましょう．