mbed 講習会

2014 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 2013.12.28

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

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．

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．

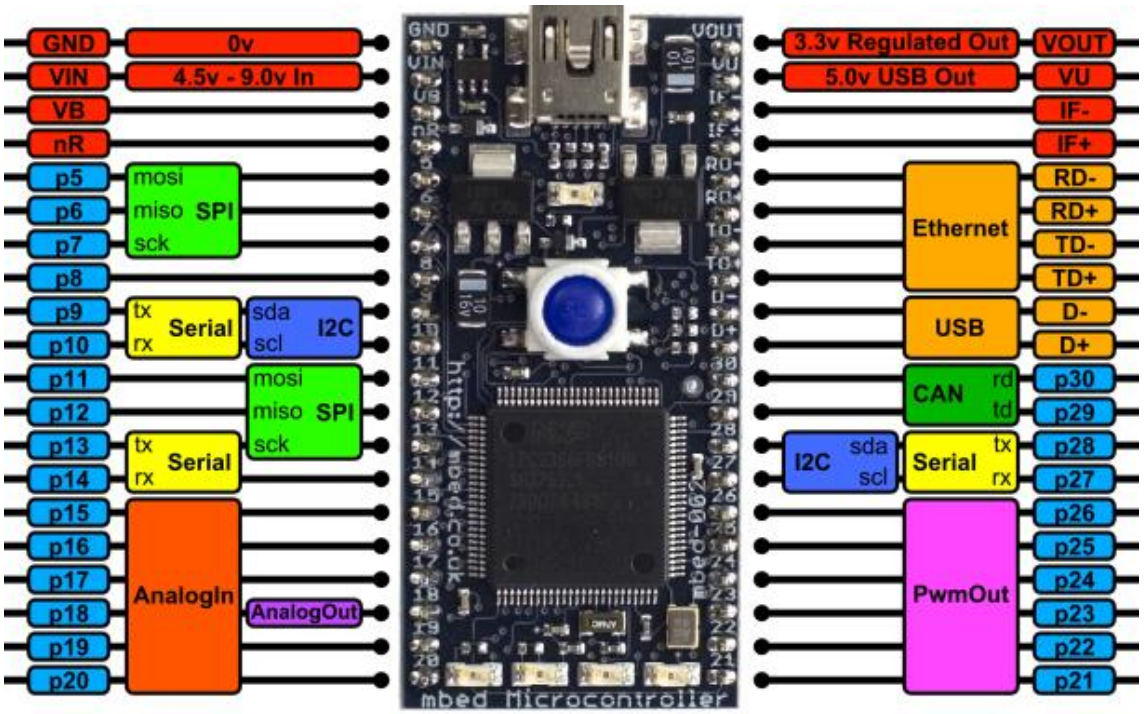


図 1　mbed ピン配置(http://mbed.org/handbook/mbed-NXP-LPC1768)

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．その後，個々のテーマに関しては選択課題として振り分けて行います．発表会はこの選択課題について行います．

全員が行うのは以下のとおりです．これは最初の一週間で行います．

- 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「同期性を要求される状況への対処」: mbed の入出力と CG 描画など要求される時間粒度が異なる場合への対処を行います．

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 9 章
二週目最後〜三週目：自由課題
というペースで進めます．

選択課題は以下のとおりです.

- 10.1「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.
- 10.2「通信機能を持つセンサモジュールの使用」: 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.
- 10.3「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.
- 10.4「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.
- 10.5「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.

以下は今年は選択しない課題です.

- 10.6「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.
- 10.7「振動呈示型データグローブ」
- 10.8「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**参照機能およびアウトライン定義**は必ず行なって下さい． 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）

2012 年にやってみた感じでは次の指導が必要です．ほうっておい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（オートセット，CH1 をトリガとする方法）

- 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，９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）
  - ファイル経由のプログラム間通信の説明
- ２週間目中頃
  - テーマ割り振り，部品配布
  - 基板間の接続をする課題が多いので，幾つかのコネクタの使い方の説明．熱収縮チューブの使い方，フラットケーブル用端子の使い方を教える．

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．特に自主課題になると講習資料は不完全ですので，マニュアルの隅々まで徹底理解しないと出来ません．多くの人はそうした体験は初めてで，ほぼ 100%できないはずです．**マニュアルをしっかり読めばできるという体験に持っていく**のは指導者の力です．苦労させつつ無駄な苦労はさせない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，最初はほぼ 100%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」，次にターミナルソフトで表示して「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**を体得することが必須です．実際**初めてだと，テスタやオシロスコープが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．

また自主課題は一種のプロジェクトなので，**プロジェクトマネジメントの観点**からの指導をしてください．はじめはほぼ 100%プロジェクトマネジメントの感覚がなく，「回路を全

部作り」→「プログラムを全部書き」→「全然動かなくてハマる」のが通例です．例えば8ch 動かす回路だったら，1ch 分だけ回路を作り，その部分のプログラムを書いてみて，という感覚は必ず身につけさせてください（なおプロジェクトマネジメントのできていない学生は，自分が何をやっているか理解していない確率が高いです．心細いのでとりあえず回路づくりの作業に打ち込んでいるのかもしれません．早期発見してください）
プログラムだけをとっても，このプロジェクトマネジメントが行われているかに注意してください．if 文，for 文，while 文を完璧に理解していても，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通例です（これは典型的な型をまだ知らないからでもあります）．これはフローチャートを書けというのではなく，**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．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 は徹底的に糾弾してください**．研究の基本ですが，研究のディスカッションの場で糾弾しても，うるさい先輩と思われがちです．しかし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，本講習は mbed の紹介を超えて

- **マニュアル，ソース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**

を体得することを目的としています．

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

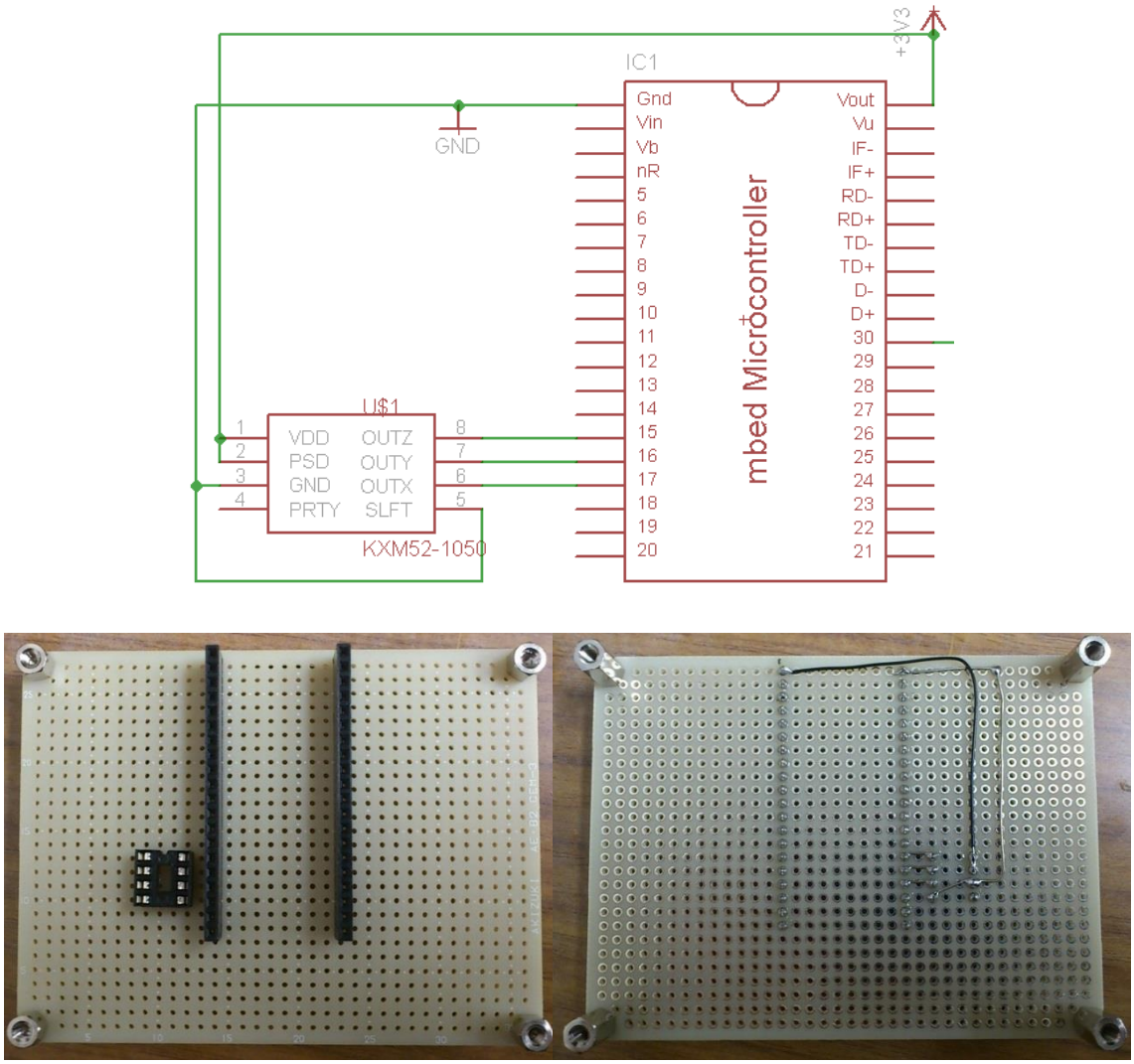


図 2　加速度センサ回路

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

回路構成上の注意：タクトスイッチの向きに注意．90 度間違えることがある．ボタンを押したときにどこが導通するかはテスタでチェックする．

web のサンプルを使って mbed 上の LED を光らせてみる．ただし入力ピンは p5 ではなく（P5 は後で使う），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, RS232c など）

RealTerm http://realterm.sourceforge.net/

Rs232c http://www.vector.co.jp/soft/win95/hardware/se369900.html?ds

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

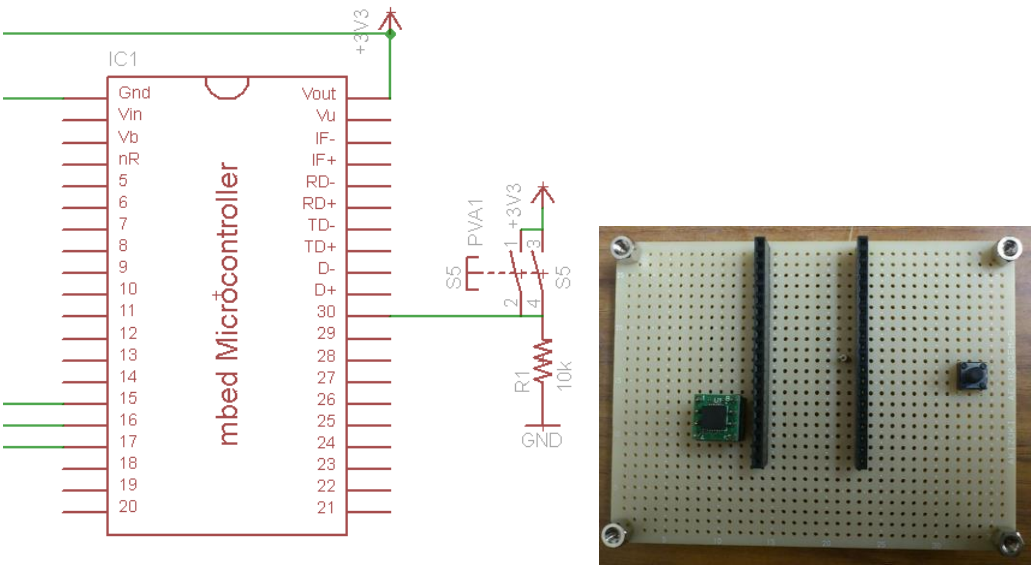


図 3　タクトスイッチ回路部分

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こと．「電子工作の素」などを参照．

[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する）．どれだけの電流が流れていることになるか計算する．その値は計算と比較して妥当か．

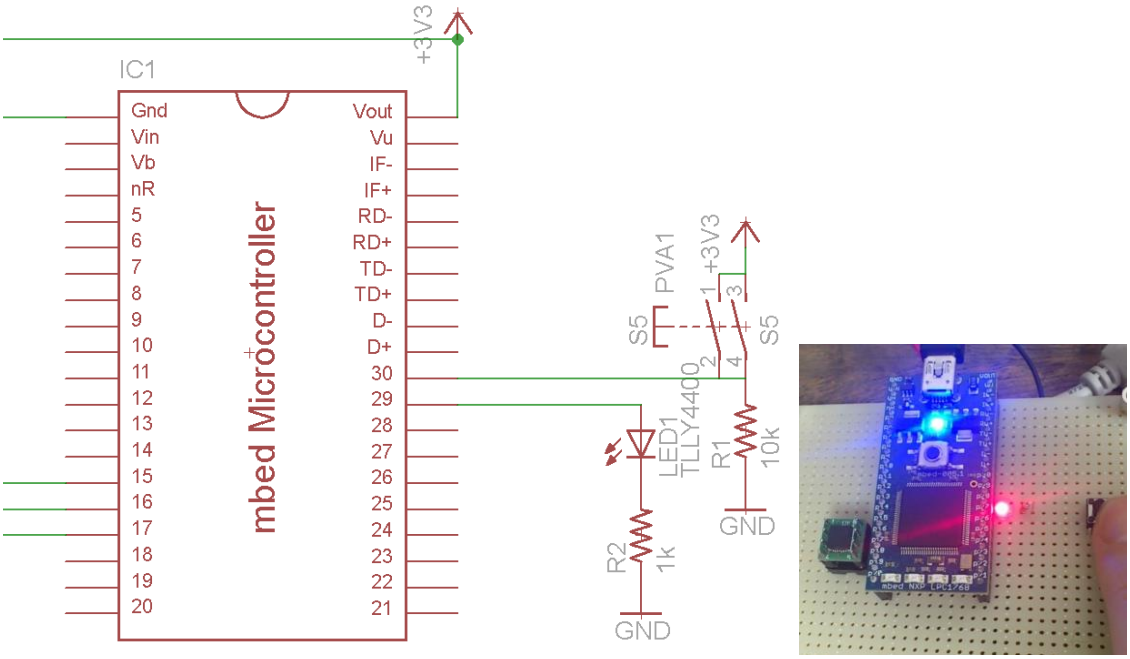


図 4　LED 回路（タクトスイッチを押すと光る）

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．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．

RC サーボモータの制御方法については下記を読むこと．

- 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
- MiniStudio の小型 RC サーボ RB50 の配線
  http://www.ministudio.co.jp/Cgi-bin/Order-JP/DetailJp.asp?GoodsNum=202
  http://www.ministudio.co.jp/Japanese/Goods-RB50-1.htm

（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となっているか（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）

（参考）シリアル通信で PC からの入力を待つ方法
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．

（注意）適切なウエイト
RC サーボモータの制御信号は，一般には 20ms 周期のパルス幅変調された信号です．プログラムのループ周期がこれよりも早く，PWM 信号がループ毎に設定され続ける場合，結果として適切な 20ms 周期の信号にならなくなることが有ります．これを回避するには例えば長い wait をはさみループ周期を遅くする必要があります．ただし今回の場合，キーボード入力があった場合だけ PWM 信号の設定を変えるようにすればこの問題はほぼ生じません．

```
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）
    }
}
```

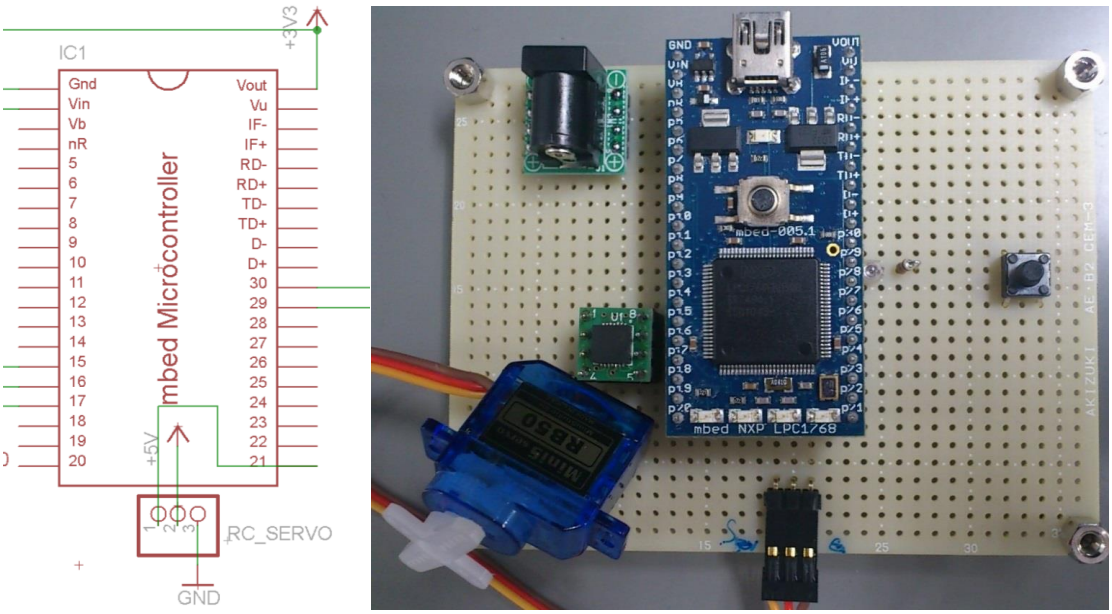


図 5 RC サーボ回路（部分）

6　mbed でアナログ出力（＋オペアンプ）

mbed は 1ch，10bit の DA および 6ch，12bit の AD を持っている（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，出力波形を確認する．
http://mbed.org/handbook/AnalogOut

オペアンプを使ってスピーカを駆動する．回路図（図 6）を参照のこと．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．

（注意）

- 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．

（参考）
オペアンプおよび電圧フォロア回路については下記を読むこと．
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スピーカの音を制御する．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．オシロスコープで波形の変化を確認する．（正弦波を出力するには？sin 関数を用いた場合とテーブルを用いた場合を比較する）

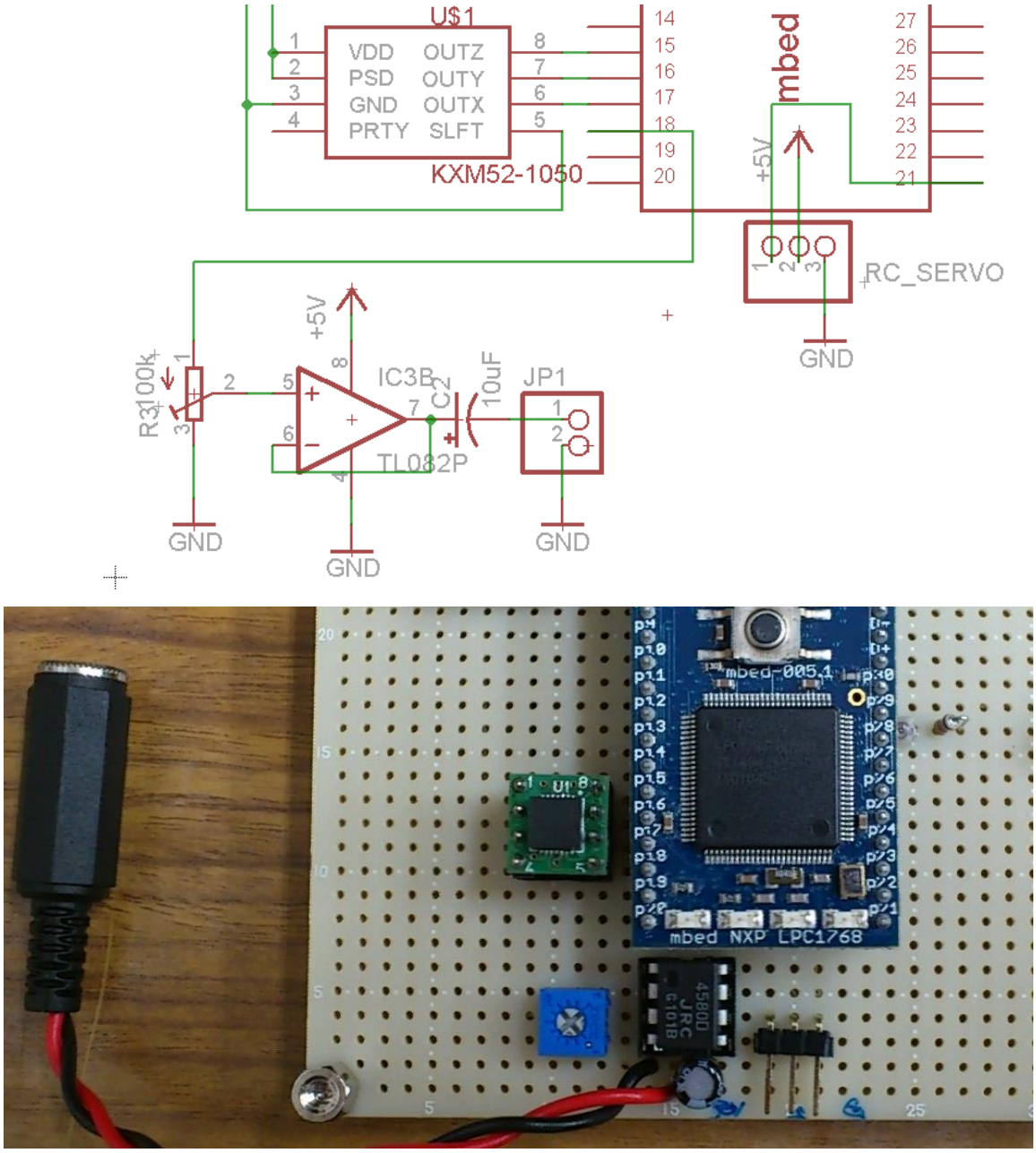


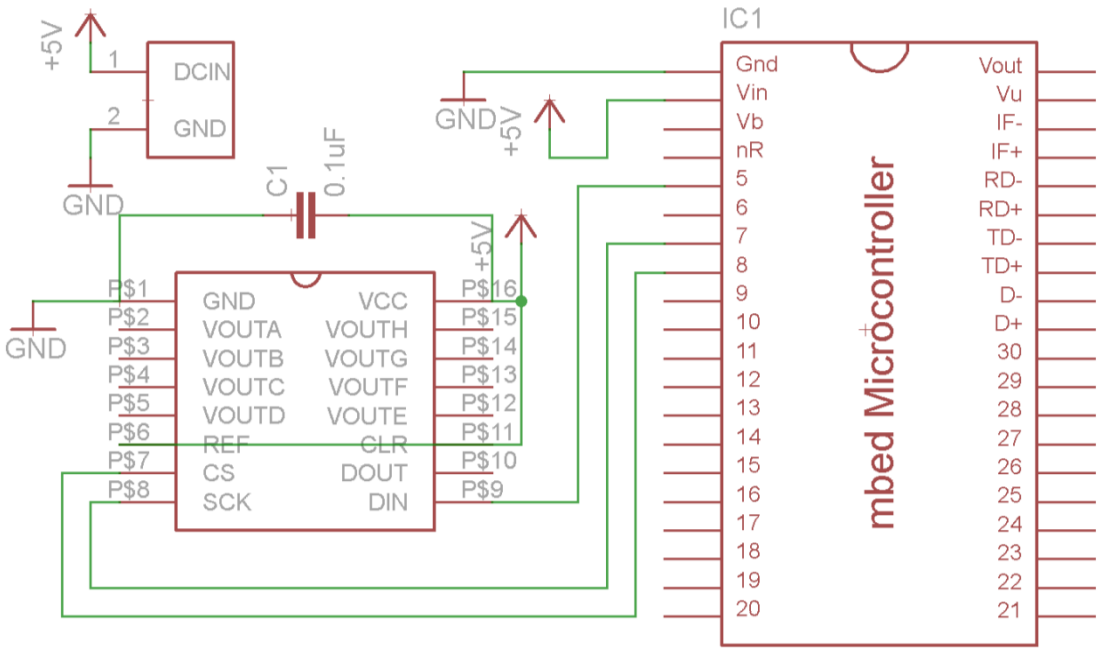
図 6　アナログ出力⇒オペアンプ回路（電圧フォロア）TODO:圧着端子版に変更

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こと．「電子工作の素」などを参照.



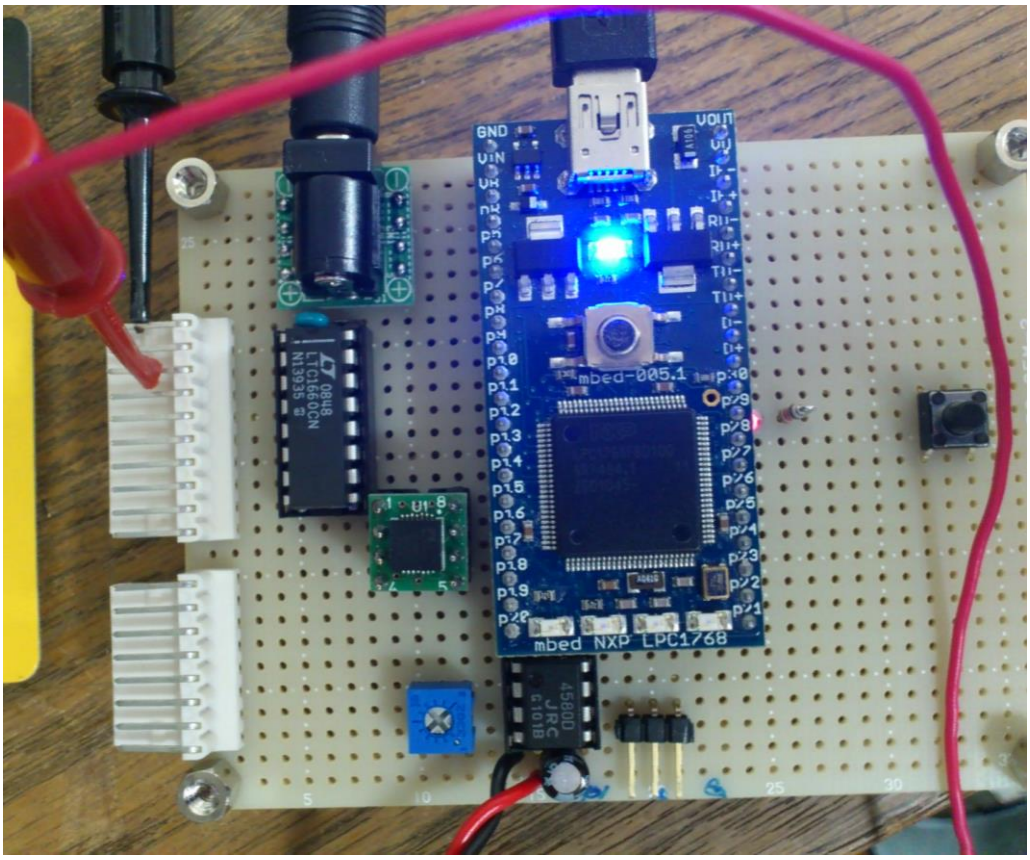


図 7　8ch DA ポート回路（10pin の端子を使い余った 2pin はグランドにしている．また この写真では AD6ch を 8pin 端子につなぎ，余った 2pin はグランドと 3.3V にしている）

```
#include "mbed.h"

SPI spi(p5, NC, p7);  // mosi（出力 master output slave input）, miso（入力）, sclk（クロック）．今回は入力不要なので NC(not connected)とする
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 を動かす最初のサンプルプログラム

[課題8] サンプルプログラムを動かし，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] 波形を適当な正弦波に変更，プログラム中のウエイトを無くしオシロスコープで観察する．データ更新一周期にどのくらいの時間がかかっているか．

[課題10] 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．

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 TA8429HQ 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．

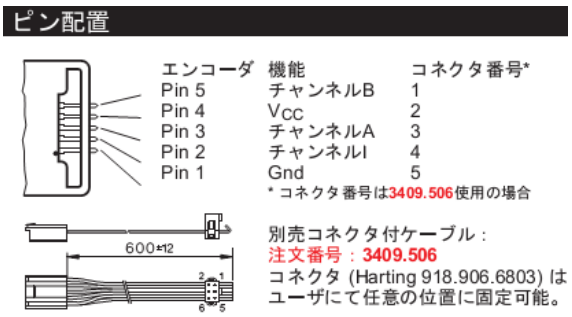


図 9 エンコーダピン配置（HEDS5540 のマニュアルより）

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);
    }
}
```

[課題11]　　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．

H ブリッジ回路については例えば下記で自習すること．

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，今回使用する TA8429 とは合致しないため PWM 制御により明示的に制御してみる．

TA8429 のマニュアルによれば駆動回路は図 10 のとおりである（ただし電源電圧は 24V ではなく 12V を使用する）．ロジック用電源（5V）と駆動用電源の電圧が異なることに注意．1,2 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．

コンデンサは電源電圧の 2 倍位上の耐圧のあるものを選ぶこと．GND ピン(6)を mbed のグランドに接続するのを忘れないこと．

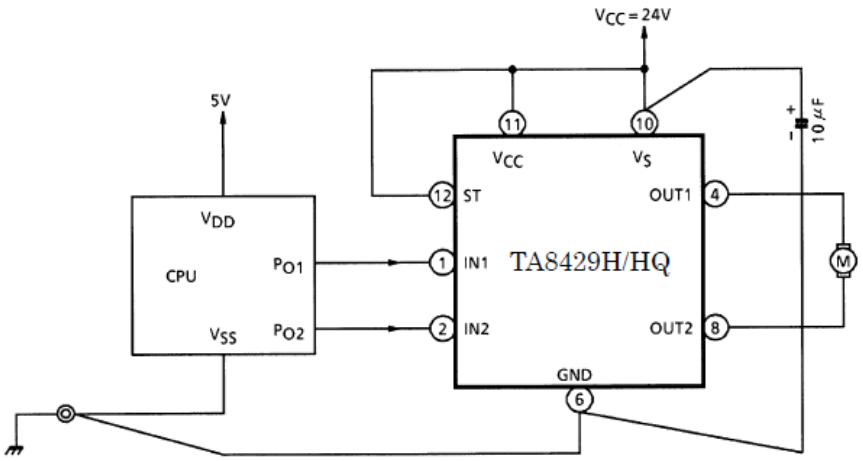


図 10　駆動回路(TA8429 マニュアルより)

電源は実験用の電源を用いる．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 0.5A と設定する．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こと．

http://okwave.jp/qa4105361.html

http://www.kikusui.co.jp/knowledgeplaza/powersupply1/powersupply1_j.html

各ピンへの入力と出力の関係は表 1 のとおりである．ST 端子は常に High とするので無視して，IN1 と IN2 を，{L，H} とすれば正転，{H，L} とすれば逆転する．{H，H} はブレーキであるが今回は使わない．{L，L} だとストップ，つまりモータは駆動されない．

表 1 ファンクション（TA8429 のマニュアルより）

入力			出力		出力モード
IN1	IN2	ST	OUT1	OUT2	
H	H	H	L	L	ブレーキ
L	H	H	L	H	逆転
H	L	H	H	L	正転
L	L	H	OFF (ハイインピーダンス)		ストップ
H / L	H / L	L	OFF (ハイインピーダンス)		スタンバイ

[課題12]　　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，IN1 と IN2 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
    }
}
```

上記でも基本的な動作は可能だが，TA8429 のマニュアルによれば，「注１：入力信号切替時（正転⇔逆転，正転／逆転⇔ブレーキ）は OFF-TIME を挿入してください（100μs 以上）」とある．これを素直に実装すると例えば次のように書ける．（実際にやってみるとこのウエイトは無くても問題はほぼ生じず，むしろ制御ループの周期が変わる副作用のほうが大きいかもしれない．ただし長期的な故障率や発熱を考えると入れておいたほうが良いかもしれない．）

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

#define CW 0            //正転
#define CCW 1           //逆転
void motor(float p)
{
    static char state=CW;   //回転状態を保存するstatic変数

    if(p>1.0) p=1.0;        //最大値を1.0に
    if(p<-1.0)p=-1.0;       //最小値を-1.0に

    if(p>0) {               //IN1をL，IN2を定期的にHにして正転
        IN1 = 0;
        if(state==CW) { //正転→逆転の場合に100usのウエイトを挟む
            state=CCW;
            IN2=0;
            wait_us(100);
        }
        IN2 = p;
    } else {              //IN2をL，IN1を定期的にHにして正転
        IN2=0;
        if(state==CCW) { //逆転→正転の場合に100usのウエイトを挟む
            state=CW;
            IN1=0;
            wait_us(100);
        }
        IN1=-p;         //マイナスを付けることに注意
    }
}
```

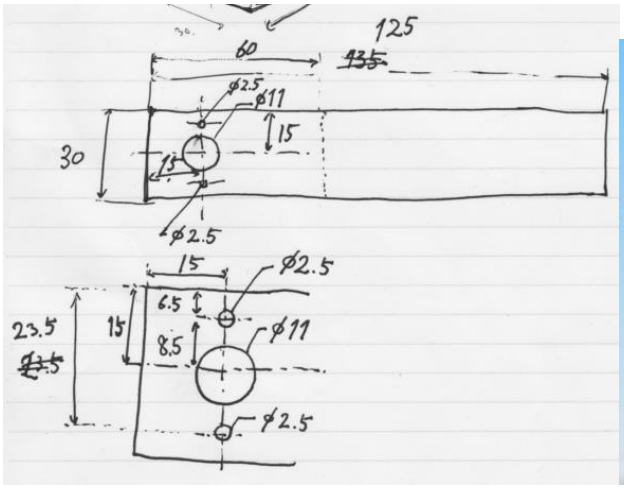
[課題13]　シリアル通信で指令を送り，例えば f(orward)，b(ackward)を入力すると正転／逆転が切り替わり，h(igh)，l(ow)を入力するたびに速度が変更されるようにせよ．PWM の制御周期を 0.02s(50Hz，デフォルト値)，1ms（1kHz)，50us(20kHz)，と変化させ，振る舞いの変化，モータから発する音を観察し，考察せよ．20kHz に設定し，２つの PWM 出力が予想通りの出力になっているかどうかをオシロスコープで 2ch 同時に計測して確認せよ（波形は画面を記録)．　この後は 20kHz の設定で用いる．

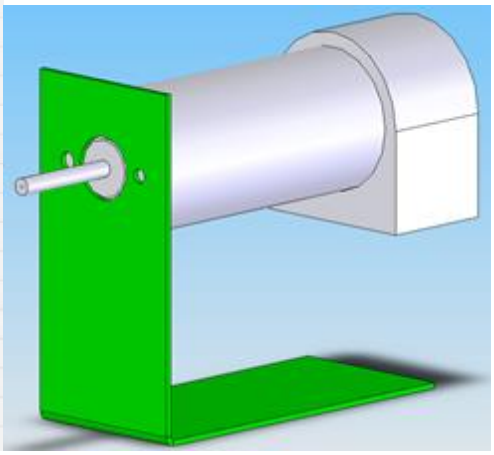
[課題14]　一方向に回転させ続けている時，電流はどの程度流れているか．またこのモータの回転を手で止めた時（気をつけること！)，電流はどの程度流れるか．この違いはなぜ生じるか説明せよ（DC モータ，逆起電力，のキーワードで調べる)．

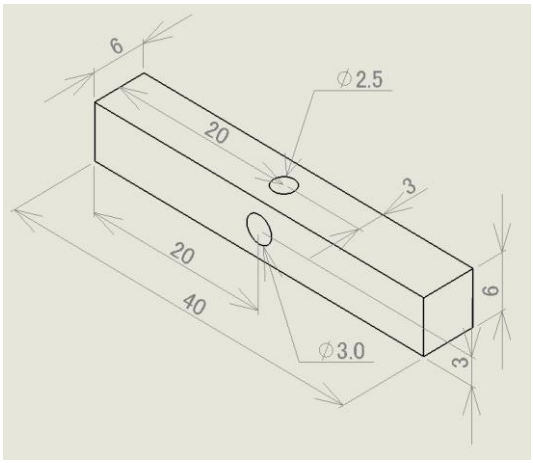
[課題15]　アルミ加工により，DC モータの軸に取手をつける．　必要ならヒートシンクも作成する．（この部分は下記に従い個別に指導します．また普通はヒートシンクは購入します)

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 11）.

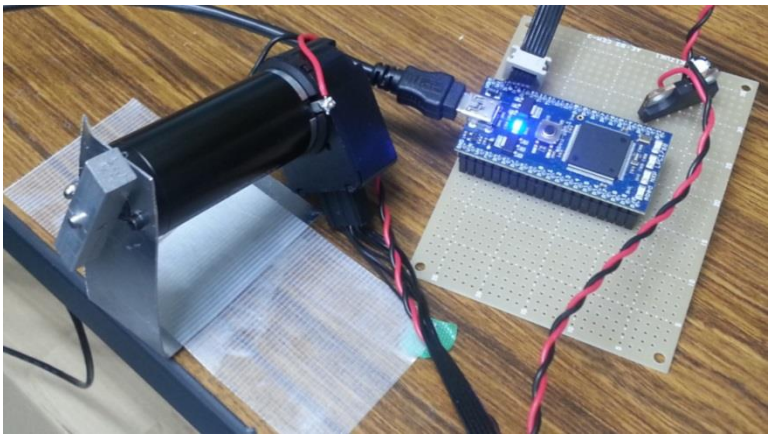


図 11　ノブ付きモータと制御基板

[課題16]　　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17]　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), t(op),b(ottom)），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．

```
void motor(float p){
    …
}

int main() {
    char c;
    while (1) {
        if(pc.readable()){
            c = pc.getc();
            switch(c){
                case ‘r’:
                    指令値変更
                    break;
                case ‘l’:
                    …
            }
        }
        //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
    }
}
```

[課題18]　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．なるべく心地よい感覚にするには．

9　同期性を要求される状況への対処

実際の研究プロジェクトでは，複数のセンサ，複数の感覚呈示（ディスプレイ，音，触覚，他）を同時に扱うため，その同期性の確保が重要となります．本章では同期性を要求されるプログラムを書く際の２つの Dirty Prototype 的方法を扱います．

9.1 ファイル共有による通信

次章の自主課題ではすべての課題で DxLib を用いていますが，実世界のセンシング情報と組み合わせるにはひとつ難しい点があります．DxLib のプログラムは通常，画面のリフレッシュレート（60Hz 等）でループを回しますが，mbed からのデータや mbed を介した制御は数百 Hz になることがあります．つまり，**必要なリアルタイム性のレベルが異なる**場合があるということです．　典型的な例はシリアル通信です．
このように異なる粒度の時間を同時に扱うには，一般にはひとつのプログラム中でスレッドを使えば出来ます．が，ここでは PC 側で二つのプログラムを動かし，共有ファイルを用いたプログラム間通信を行います（図 12）．研究室内の環境では他の場合への応用（例えば PC 同士の通信や，他のソフトウエアとの連携）が楽なのでこの方法を使います．力がついてきたらスレッドによる方法や他の一般的なプログラム間通信も試してみてください．（mbed の方もシリアル通信部分を割り込み処理にすれば完璧です）

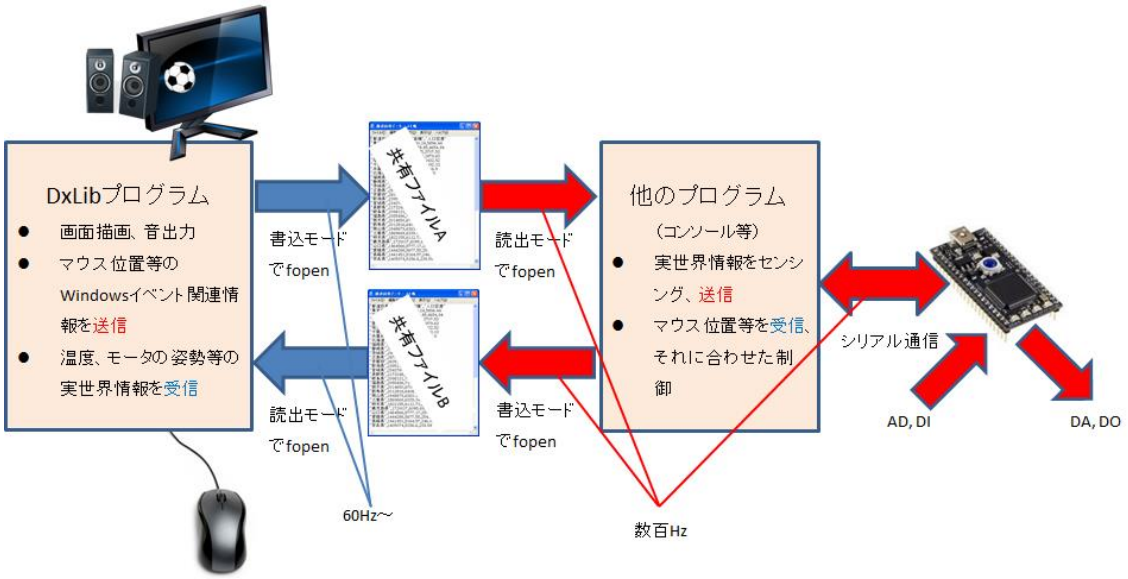


図 12　共有ファイルを用いた DxLib プログラムと他のプログラムの協調

```
void main(void)
{
    FILE *fp;
//  char SharedFileName[] = "¥¥¥¥192.168.0.2¥¥shared¥¥shareddata.txt";
//  char SharedFileName[] = "C:¥¥shareddata.txt";
    char SharedFileName[] = "..¥¥..¥¥shareddata.txt";
    char sendData[255]={0};
    int a=0;

    //ファイル書き込み専用で開く
    if((fp = fopen(SharedFileName,"wb"))== NULL){
        printf("%s : ファイルオープンに失敗しました¥n",SharedFileName);
        exit(0);
    }else{
        printf("書き込み用にファイルを開きました¥n");
    }

    while(!_kbhit()){
        sprintf(sendData,"書き込むデータ %d 読めますか",a);
        a++;
        //以下の2行が中核．fseekでファイル先頭に行き，fwriteでデータを書き込む
        //通常のfprintfなどではファイルに書き*足される*が，ここでは書き足す必
        //要が無い(最新のデータだけあればよい) のでこれでよい．
        fseek(fp,0,SEEK_SET);
        fwrite(sendData, sizeof(char),255,fp);
        printf("書き込みました：%s¥n",sendData);
        Sleep(500);//500ms
    }
}
```

図 13　送信側プログラム（部分）

```
void main(void)
{
    FILE *fp;
//  char SharedFileName[] = "¥¥¥¥192.168.0.2¥¥shared¥¥shareddata.txt";
//  char SharedFileName[] = "C:¥¥shareddata.txt";
    char SharedFileName[] = "..¥¥..¥¥shareddata.txt";

    char receiveData[255]={0};

    //ファイル読み出し専用で開く
    if((fp = fopen(SharedFileName,"rb"))== NULL){
        printf("%s : ファイルオープンに失敗しました¥n",SharedFileName);
        exit(0);
    }else{
        printf("読み出し用にファイルを開きました¥n");
    }

    while(!_kbhit()){
        //以下の2行が中核．fseekでファイル先頭に行き，fwriteでデータを書き込む
        //通常のfprintfなどではファイルに書き*足される*が，ここでは書き足す必
        //要が無い(最新のデータだけあればよい) のでこれでよい．
        fseek(fp,0,SEEK_SET);
        fread(receiveData, sizeof(char),255,fp);
        printf("読み出しました：%s¥n",receiveData);
        Sleep(500);//500ms

    }
}
```

図 14　受信側プログラム（部分）

[課題19]　　PC_COMMUNICATION のフォルダを開き，送信，受信の二つのプログラムを動かす．このサンプルは片方からもう片方に数値を送っている．動作を確かめた上，共有ファイルを増やし，反対側からもう片方にも別の数値を送るようにする（双方向

の通信を実現する).

fread，fwrite，fseek 関数を使っている．これらを使用している部分の動作を一行ずつ説明すること．なおこの方法で画像の送受信等も可能である．

[課題20]　[課題 19] のソースを使い，二人一組で PC 間通信を行う（ファイル名を変更するだけ．場合によっては共有ファイルを事前に作成する必要).

[課題 19] のプログラムの片方を DxLib プログラムとし，もう片方をシリアル通信用のコンソールプログラムとすれば，研究室で扱うたいていの課題に対応できます.

（参考）mbed のシリアル通信は USB を介しているため，規格上リフレッシュレートは 1kHz が上限です．リフレッシュレートが 1kHz でもオーディオ信号(40kHz 程度)が送れるのは，一回に大量のデータを送っているからで，データ転送量自体は最大約 1Mbps です．リアルタイム性という点ではあくまでも最大 1kHz であり，データの送受信だけで数 ms かかることに注意してください.

9.2 シリアル通信の遅延防止

リアルタイム性に関するもう一つの典型的な症状は，シリアル通信で生じます．mbed から送るデータの周期と，PC で読み取るデータの周期が異なると，PC の方が早ければデータが空の状態が続き（それを明示的に処理するプログラムにしないとバグになり)，PC の方が遅ければシリアルバッファにデータが「溜まり」，けっこうな時間遅れを感じることが生じます（春休みの宿題で苦労したはず).

色々な対処法が考えられます．前節で述べた，PC 側でシリアル通信担当を何らかの形で独立させるのはその一つ．また mbed→PC のデータ通信の際は，PC のデータ取得周期を短めにして，かつ，受信データが来るまで待つよう設定することも考えられます（逆に PC→mbed のデータ通信の場合は readable 関数で PC からのデータがあるかどうかを判断できます）http://mbed.org/handbook/Serial

もう少しちゃんとやるためには，「PC からデータ要求コマンドを送り，そのときだけ mbed からデータを送る」というやりとりを作ればよいです．mbed プログラムはすでに述べた readable 関数で待ち構える形になります（あるいは割り込みプログラムになります).

また PC から mbed に数バイトないし数十バイトのデータが送られる場合があり，この際はデータの開始がどこからかがわからないと問題が生じます．この場合送信データの最初に開始を示すヘッダがあれば，データの切れ目がはっきりします（下のサンプルコード参照).

このあたりを考えていくと，通信の規格を考えていくことになります．例えばシリアル通信で可変長のコマンドを扱う代表的な規格の一つは midi で，mbed でも使うことが出来ます．また例えば近藤科学のサーボモータ用シリアル通信規格なども参考になります（シリアル通信で動く機器はたいていこういう規格を自作しています).

・midi 通信規格　http://ja.wikipedia.org/wiki/MIDI

・近藤科学のサーボモータ用シリアル通信規格 http://bit.ly/1cSsLlA

```
#define HEAD_DATA 0xFF

while (1) {
   if (pc.readable()) {
      if (pc.getc() == HEAD_DATA) {
            PC からのデータを数十バイトまとめて読む
          メインの処理をする
      }
   }
}
```

10 選択課題

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 Dxlib 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．

10.1　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 15)．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．

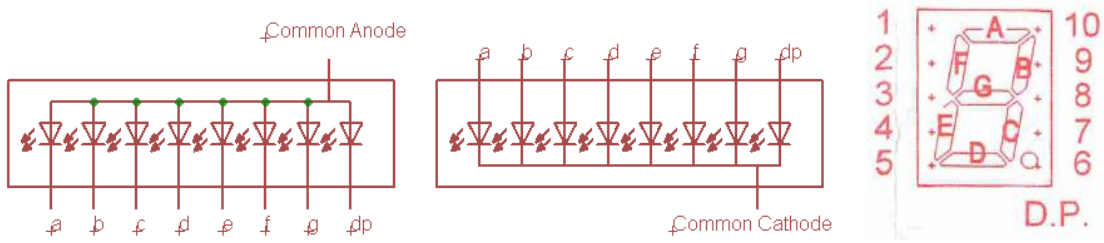


図 15　7 セグメント LED の内部構造．(左) アノードコモン，(中) カソードコモン，(右) LED の配置

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）．

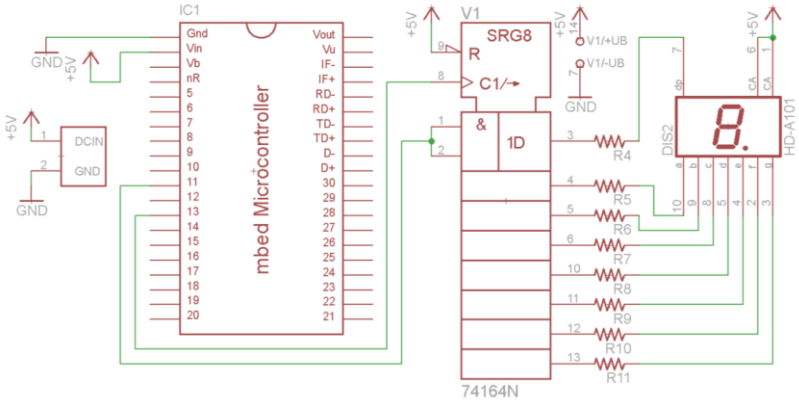


図 16 7 セグメント LED 点灯回路

- 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21]　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 DxLib を用いて数値をグラフィカルに表示する．

10.2　通信機能を持つセンサモジュールの使用（中級～上級）

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22]　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くようにする．

10.3　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23]　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．

10.4　外付けの AD を使う（中級～上級）

mbed には 6ch もの 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．この場合も，SPI 通信で外付けの AD を増やすことが出来る．DA の場合と違って入力．

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．

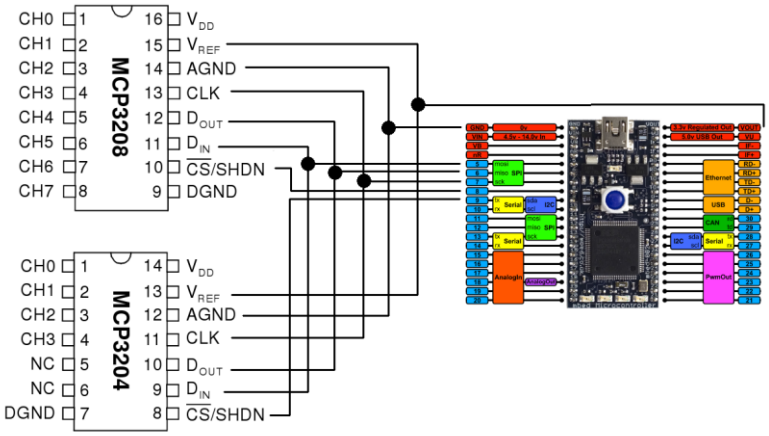


図 17　複数個の AD コンバータをつなぐ例．Vdd は 5V に接続．CS によって IC を選ぶ．
http://mbed.org/cookbook/SPI-communication-with-external-ADC-MCP3 より

[課題24]　　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 DxLib で画像表示する．回路は図 19 の曲げセンサ回路と同じで良い．

10.5　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 18）．

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

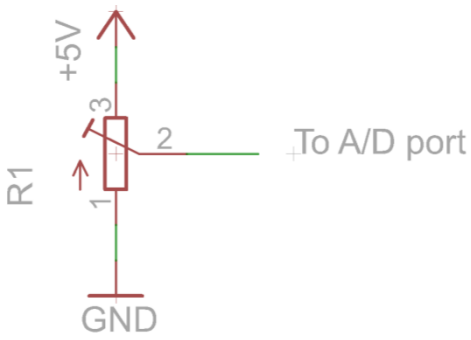


図 18　ポテンショメータ回路

[課題25]　　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 DxLib で表示する．

以上により mbed を研究室の研究で使ういろいろな場面に対応出来ました．この講習では省いた他の機能としては，インターネットへの接続，USB デバイスなどがあります．インターネットへの接続は研究室的にはデバイスの無線化に力を発揮します．USB デバイスへの対応は研究室的にはUSBオーディオデバイスやUSBマウス等で使う可能性があります．ここまで出来たなら対応は容易でしょう．

～以降の課題は今回は選ばないこと～

10.6　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

10.7　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．

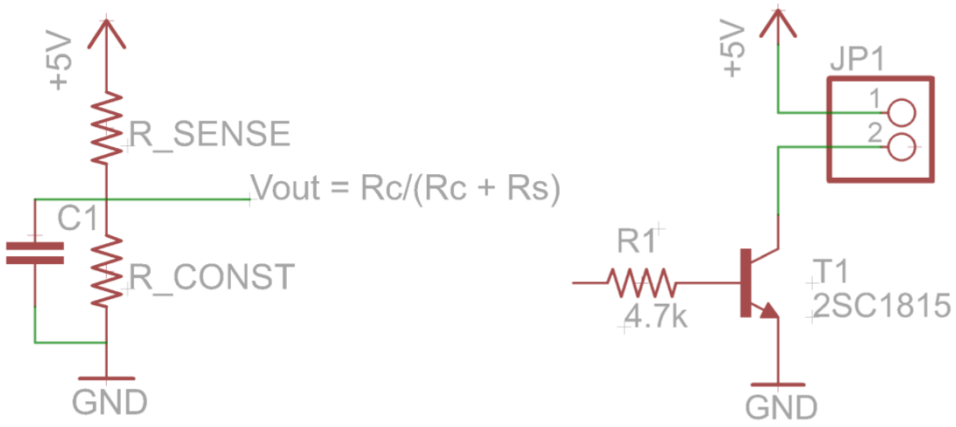


図 19　曲げセンサ回路（簡易版）と振動モータ回路

- 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26]　　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（DxLib）で何が出来るかも検討する．最低限，状態の表示は行う．

10.8　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。**エラー! 参照元が見つかりません。**章を参照のこと。

http://mbed.org/handbook/USBAudio

[課題1] 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/

10.9 複数スピーカの駆動と波形ファイルの用意（初級～中級）

8ch の DA 出力にオーディオ用パワーオペアンプ 8 個を接続、8ch のスピーカを駆動する。アンプは LM675 等を用い、外部電源 12V で駆動する。

LM675 の回路はマニュアルを参照すること。

[課題2] DxLib と連動して音源位置を動かすようにする。例えばスピーカを自分の周囲に配置し、マウスカーソルを動かすと音源位置が移動するようにする。さらに適当な音源（例えばピアノの音や人の声）をテキストファイルに変換し、mbed のヘッダファイルに配列として定義することによって、その音が出るようにする（wave ファイルからテキストへの変換ソフトは検索すると沢山出てくる。また実は mbed に wave プレイヤのサンプルが沢山ある。その場合は外付けの SD カードが必要なようである）

11 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．