PIC 講習

2007 年 4 月 梶本研究室
2009 年 3 月 改定

0 ChangeLog

2009 年 3 月

- コンパイラをフリーの PICC-Lite に変更.
- 書き込みツールを PICKIT2 に変更．PICKIT3 は未だ日本語の紹介が少ないので時期尚早か.

1 はじめに ～なぜ PIC か～

研究室では PIC マイコン, H8 マイコン, インタフェースボードを接続した PC という 3 つの環境を学びます. なぜ 3 つも知る必要があるのでしょうか？

PIC マイコンは安価で非常に小さなシステムを作ることができる半面，計算能力，プログラムメモリ量ともに貧弱です．高級なことをやろうと思ったら PC と接続して PC の周辺機器となる必要があります.
つまり PIC のメリットとは「安く小さく作れること」であり，デメリットはその他のすべて（！）であるとすら言えます．ですから本研究室での研究上の流れとしては，正統的には PC＋インタフェースボードか，H8 マイコンで試作・評価した後，小型化が必要となった時に PIC に移行することになります.

しかし時には小型であることがはじめから必須である場合もあります．無線タグの研究などはその例で，そうした場合ははじめから小型マイコンを使う必要があります．また，PC と PIC では得意分野が異なる場合もあり，例えば LED を 10kHz で点滅させるという場合， PC では大変ですが PIC だと一瞬です.

このように PIC マイコンは，少し癖はあるものの使いこなせれば強力な武器となるものです.

小型のマイコンは PIC 以外にも沢山あります．正直どれでもかまいません．海外では AVR が導入としてメジャーです．ただ PIC は国内で利用例が多く，日本語の情報が入手しやすいというメリットがあります．コンパイラも含めたフリーの開発環境もそろっています．また PIC は種類が多く，ピン数や速度など，用途に最適なものを選ぶことができます.

以上が PIC を学習する理由です.

研究として重要なのは，個別の開発スキルはもちろんですが,

- **開発環境にこだわりをもたず**
- **目的にあった最適な開発ターゲットをその都度選択・適応しつつ利用できる**

ことがより重要です．さらに時間軸の概念を加えて,

- **将来の拡張性，他人のメンテナンス性**

まで考えられるとなお良いです．企業の場合はさらに将来の入手性まで考えます.

2　自習のための情報源

「トランジスタ技術 2009 年 4 月号特集～これならわかる！！PIC マイコン」

今回用いる MPLAB ver.8.2(統合開発環境)，PICC- Lite(C コンパイラ)，PICKIT2(書き込みツール)のインストール，セットアップに関して最も良い情報源．**講習では教科書的扱い．**

「C 言語で始める PIC マイコン～フリーの C コンパイラではじめよう」

今回用いる PICC- Lite（C コンパイラ)を用いた入門書．このコンパイラに関する詳しい書籍は珍しく，また使用しているマイコン PIC12F675 は今回用いる PIC12F683 とほぼ同じ（メモリ容量のみ異なる）なので，最も良い情報源．ただしコンパイラのバージョンの違いからそのまま信じられないことも時々．**講習では教科書的扱い．**

「PIC とセンサの電子工作」

PIC 周辺の回路について特にセンサを中心に扱った良書．電子回路初心者には為になるのでぜひ買って読みこなしてほしい（ただしやや玄人好みか)．後閑さんの本とは異なり PICC- Lite を使っている点で研究室の方針と合致．

「電子工作の素」

後閑本その１．電子工作を初めて行う人はぜひ買って読みこなしてほしい．

「8 ピン PIC マイコンで始める作る，できる電子工作入門」
「C 言語による PIC プログラミング入門」
「電子制御のための PIC 応用ガイドブック」

後閑本その２３４．初めの二つが入門で最後が応用．入門編は特に標準的な内容だがコンパイラが CCS 社のものであるため研究室の現状とは合わない（共有 PC にはインストールされている)．ただし脳内変換して読める人には非常に有益．

将来的には自分で PIC マイコンのハードウエアマニュアルと PICC のソフトウエアマニュアルとヘッダファイルを読みこなせるのが理想です．結局すべての情報はマニュアルとソースにあります．

Web 上の情報

http://www.picfun.com/
後閑さんの有名なページ．ほとんどが CCSC ベース．

http://www.ne.jp/asahi/air/variable/picmel/index.htm
PICC-Lite ベースの開発のページ

http://www.geocities.jp/jk1brk/MISC/PIC/PIC_Index.htm
PICC Lite と PIC16F84A,PIC16F6??A による組み込み C 言語

3　予定

1 日目：LED 点滅

2 日目：シリアル通信(1)　PC とシリアル通信

3 日目：シリアル通信(2)　7 セグメント LED に数字を表示

4 日目：加速度センサかモータ

5 日目：無線通信と PC 間通信

4　1 日目：LED 点滅

開発環境を整備し，最も簡単なプログラムを動かしてみる．

4.1　　環境整備

「トランジスタ技術 2009 年 4 月号特集～これならわかる！！PIC マイコン」第 2 章を読みながら，MPLAB の最新版を Microchip 社のページからダウンロードし，インストールする．同時にコンパイラや USB デバイスドライバがインストールされる．MPLAB を立ち上げ，HITECH 社の PICC-Lite の設定をおこなう．
これらは時間がかかるので，次の回路製作は並行して行う．

4.2　　回路製作

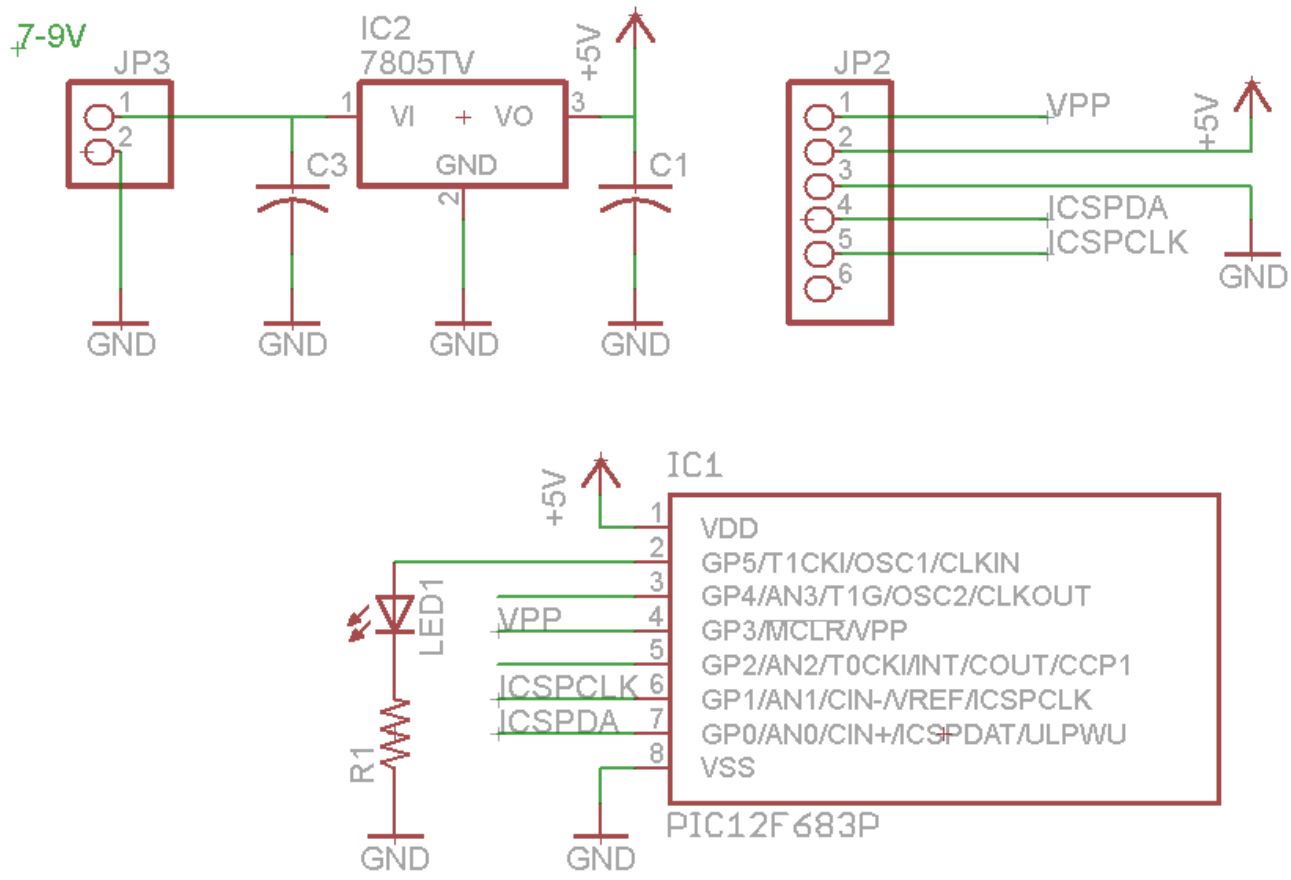


図 1　一日目回路図：LED の点滅

表 1 部品リスト

R1	470～1k 程度	C1	0.1μ 程度
C3	2.2μF～10μ 程度	LED1	赤色 LED
IC1	PIC12F683	IC2	5V 用三端子レギュレータ 7805
JP2	プログラム書き込み端子	JP3	電源端子

- VPP，ICSPCLK，GND などはネット名と呼ばれ，繋がっていることを表す．（回路図の煩雑化防止）
- 3 端子レギュレータは 5V 出力なら何でもよい．講習なのでなるべく安いものを使う．5V のスイッチン

グタイプ AC アダプタでもよい.

- JP3 には電池(9V)を接続するか，電源から **7V 以上**を供給する（3 端子レギュレータの種類によっては 6V でよいこともある）．15V 程度までは問題ない.
- LED と 3 端子レギュレータの極性に注意．LED については外見からは判断できないので，テスターのダイオード測定モードでチェック.
- PIC は 8pin ソケットを介して刺す．熱による故障を避けるため，半田付けはソケットだけで行う.
- 次回以降部品が追加されるので詰めておく．電源系(5V, GND)は配線量が多いので伸ばしておくと良い.
- PIC の VPP ピン（ここでは 4 番ピン）には，プログラム書き込み時に **13V** の電圧が加えられる．このため周辺回路はこのピンに高電圧が加えられても壊れないように設計する必要がある.

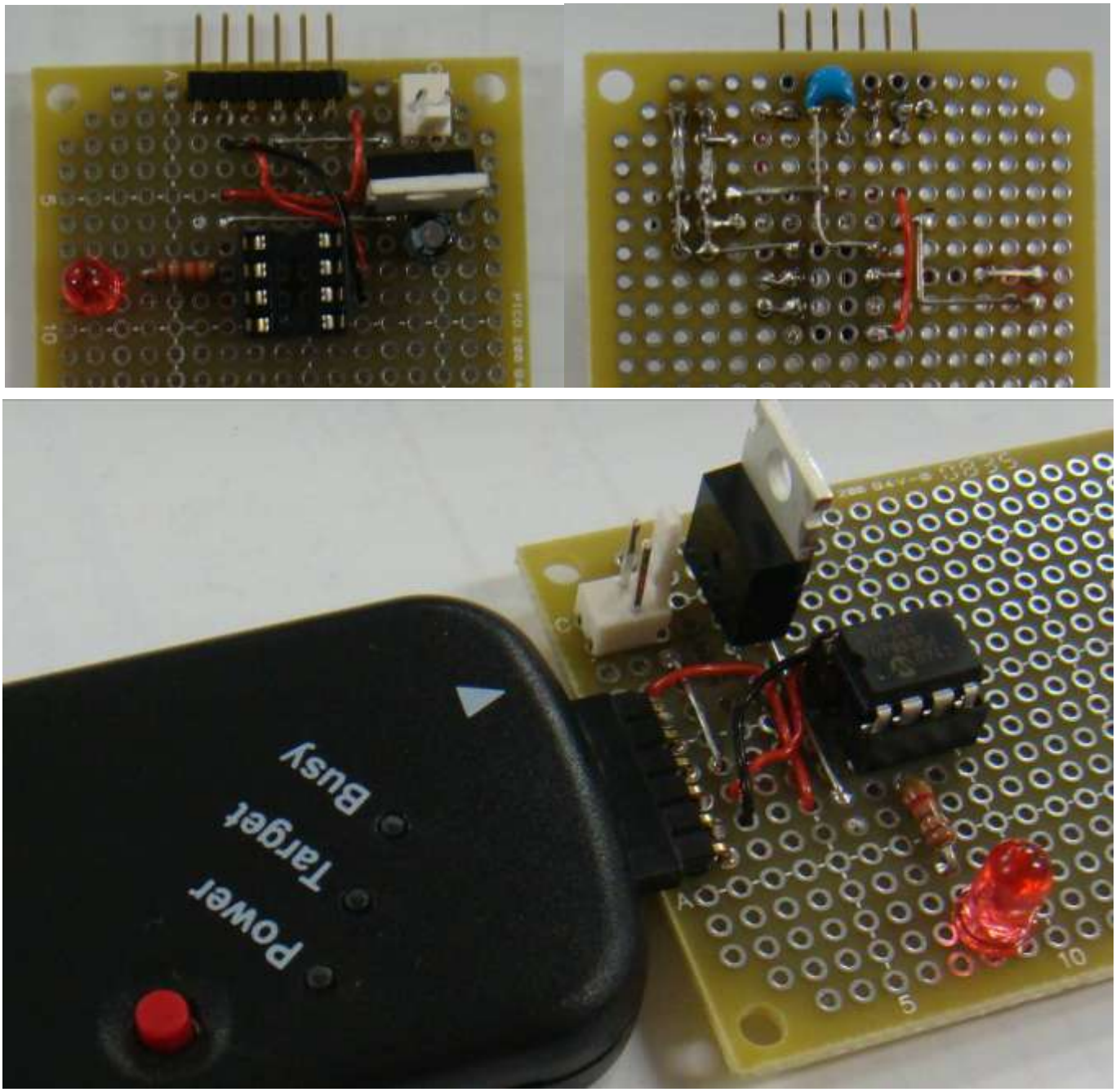


図 2　LED 点滅サンプル回路．LED と抵抗をつなぐ順番が回路と逆になっている（特に意味なし）．コンデンサを裏につける必要はない.

図 2 に作成した回路と PICKIT2 への接続の様子を示す．これらは実際の配線の雰囲気を伝えるためであり，部品を同じ場所に配置する必要は全くない．**むしろ回路図を読み解いて配線する訓練をするため，異なる配置にして欲しい．**

4.3　　プログラム

「トランジスタ技術 2009 年 4 月号特集」p102 からの説明に従い，プロジェクトを作成する．

- プロジェクトのファイル名，フォルダ名に日本語が入らないようにすること．
- PIC デバイス選択画面（図 9 -(b)）は PIC12F683 を選択
- ソースファイルはあらかじめ用意されている led.c を使用する．
- プログラムの詳細については下記を参照し，**すべての行を理解する（重要）**．
  - ✓ 「トランジスタ技術 2009 年 4 月号特集」の第 3 章．P109～113.
    - ➢ 使用している型番が違うため，I/O 周りの変数名が多少異なる．
  - ✓ 「C 言語で始める PIC マイコン」の P69～81.
    - ➢ ただしコンパイラのバージョンが違うため，例えば DelayMs 関数などは無い．
  - ✓ ヘッダファイル¥¥Program Files¥HI-TECH Software¥PICC¥PRO¥9.60¥include¥pic12f683.h. pic.h がこれを呼び出している．pic12f683.h を見るとピン名(GPIO0 など)の定義が分かる．

4.4　　書き込み

「トランジスタ技術 2009 年 4 月号特集」p102 からの説明に従い，プロジェクトのコンパイルと書き込みを行う．念のため電源に関しては次のようにする．

- ➢ 書き込み時は電源を OFF して PICKIT2 の内部電源だけを使い，
- ➢ 動かすときには PICKIT2 を外してから電源を入れる．

課題1　　点滅間隔を早くしたり遅くしたりしてみる．

課題2　　__delay_ms マクロ関数を使って正確に 1 秒刻みで点滅させる

正確な時間待ちをするためのマクロ関数__delay_ms と__delay__us が用意されています

__delay_ms(ミリ秒);

__delay_us(マイクロ秒);

中の部分の数字は unsigned long で指定できます。**アンダーバーは最初が二つ，後が一つ**です．

使い方は最初に周波数のセットを行う宣言をします。例えば 4MHz(PIC12F683)動作の場合

#define _XTAL_FREQ 4000000

と定義したうえで使います．指定できる時間には上限の制約があります．4MHz の場合 197ms が設定できる最大値となります．

表 2 動作周波数と設定できる最大値

動作周波数	最大時間	__delay_ms(x)	__delay_us(x)
20MHz	39,424us	__delay_ms(39);	__delay_us(39424);
16MHz	49,280us	__delay_ms(49);	__delay_us(49280);

10MHz	78848us	__delay_ms(78);	__delay_us(78848);
8MHz	98560us	__delay_ms(98);	__delay_us(98560);
4MHz	197120us	__delay_ms(197);	__delay_us(197120);

これ以上の大きいタイマーが必要な場合はループ処理などで対応します.
なおこのマクロは変数が使えません．プログラミングの段階で固定した定数を入れるしかないので，利用には多少工夫が必要です．（可変の引数をとる関数を作るにはどうしたらよいでしょう？）
(参考) http://www.ne.jp/asahi/air/variable/picmel/other_info/index.htm (サンプルもあり)
(参考 2) ¥¥HITECH インストールフォルダ ¥PICC¥PRO¥9.60¥docs¥manual.pdf (コンパイラマニュアル)

課題3　(宿題)「トランジスタ技術 2009 年 4 月号特集」の第一章を読み，PIC マイコンの概要を知る.

課題4　(宿題)LED の制限抵抗の求め方を調べる．3 端子レギュレータの働きを調べる．参考文献に挙げた本などで調べてみる.

課題5　(宿題)実験で使う CVCC 電源の特徴を調べる．単なる AC アダプタや可変電圧電源にくらべて何がよいのか？
http://okwave.jp/qa4105361.html
http://www.kikusui.co.jp/knowledgeplaza/powersupply1/powersupply1_j.html

ICSP と PIC ライタについて補足
今回のように回路中にマイコンを装着したままプログラムを書き込むことを ICSP(In Circuit Serial Programming)と呼びます．特に表面実装型のマイコンでは半田付けしてしまうため必須です．H8 マイコンでも同じだったので違和感はないでしょう．これを使った PC によるデバッグ等も可能です.
しかし今回のような少数ピンの DIP では，メリットよりデメリットのほうが多いかもしれません．書き込み時に加わる電圧を意識して設計する必要がある，回路面積が少しでも大きくなるなど.
だから例えば今回の回路を「書き込み用のアダプタ（PIC ライタ)」として使い，実際のアプリケーション回路では書き込み部を省略するのも一つの方法です．というよりもむしろ普通の方法．PICKIT2 が PIC ライタとして非常に優秀なため講習で採用しただけです.

5　2 日目：シリアル通信(1)：PIC から PC に数値を送る．PC から LED を点滅させる

目標：シリアルライブラリの使い方，RealTerm，PC 側のプログラムを使った通信

5.1　　回路製作

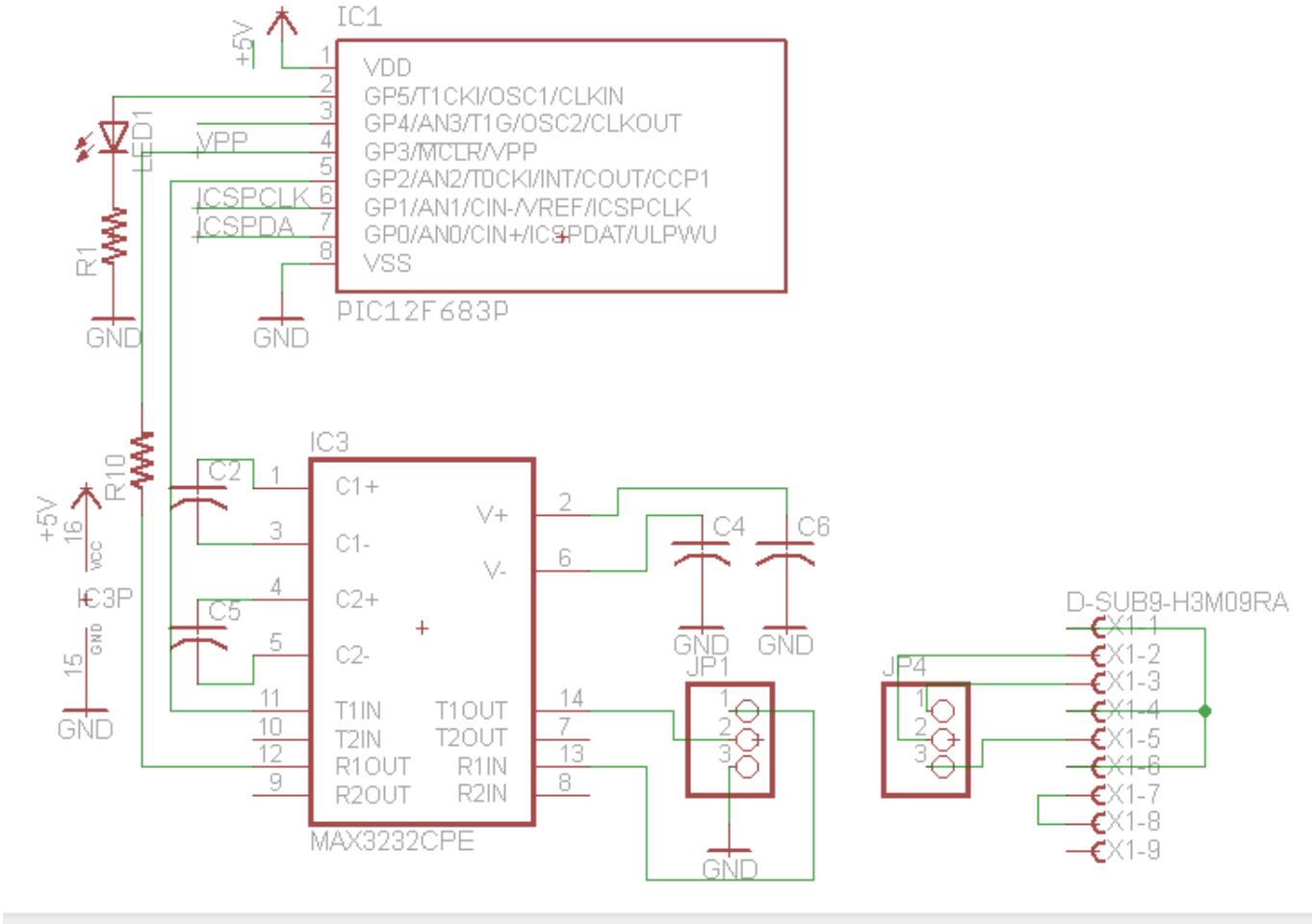


図 3　二日目課題：シリアル通信（電源，書き込みコネクタ部は省略）

表 3 追加部品リスト

IC3	**シリアル通信用電圧レベル変換 IC MAX3232，ADM3202 など**	**D-SUB9**	**シリアルコネクタ（メス）**
C2,4, 5, 6	**0.1μ**	**JP1,4**	**シリアル接続用コネクタ**
R10	**10k 程度**		

- シリアルコネクタへのケーブル配線は半田付け後に熱収縮チューブで保護する．
- 回路図中の IC3 の電源配線に注意．回路 CAD 特有の表現．
- R10 は保護抵抗．プログラム書き込みの際に Vpp 端子には 13V が加わるため，そこに繋がっている他の素子（ここでは MAX3232）が破損するかもしれないため．
- MAX3232 は電圧レベルを変換するだけの IC．似た型番で各社から出ている．

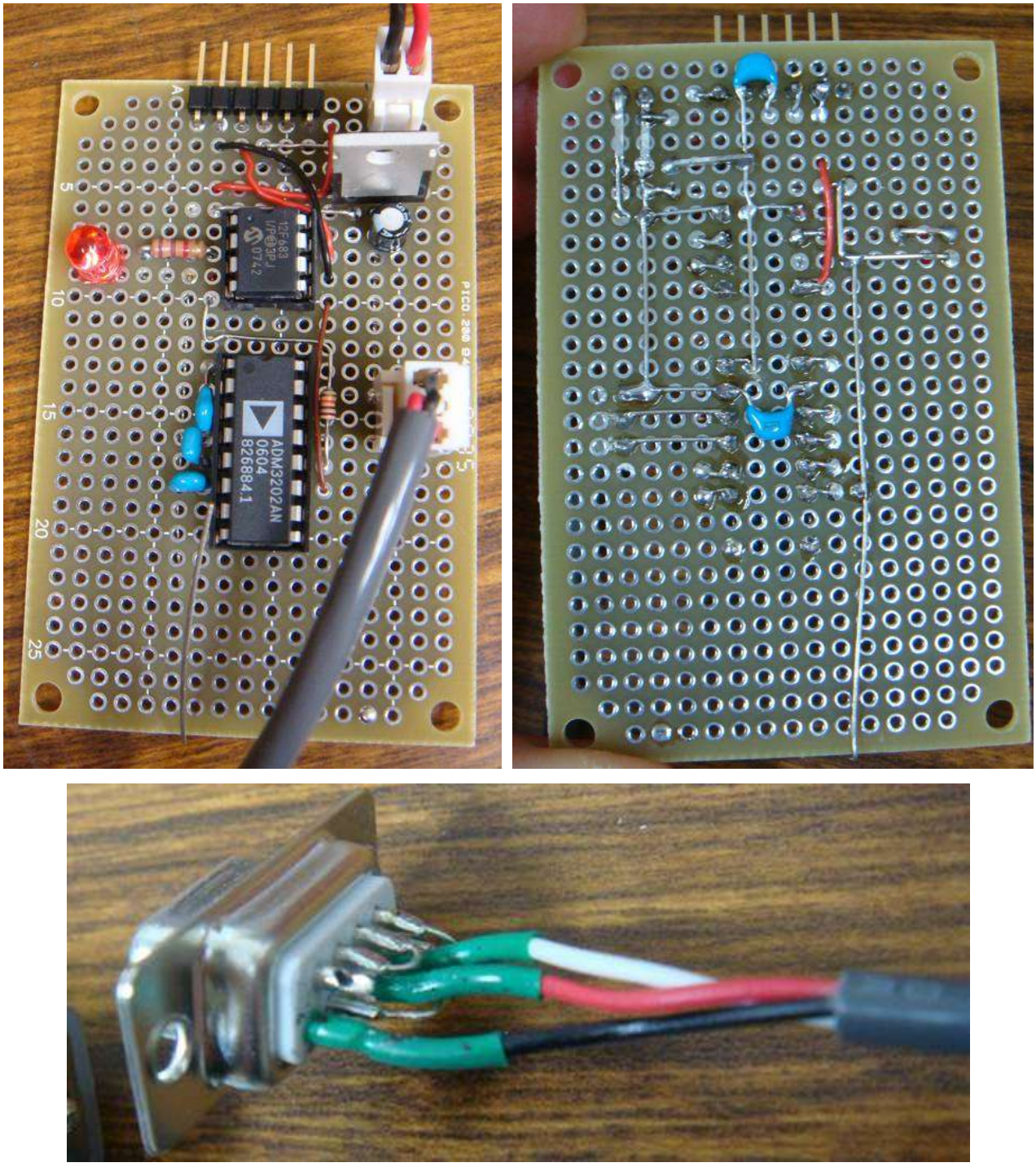


図 4　シリアル通信サンプル回路とシリアルケーブルの熱収縮チューブによる保護.

5.2　　プログラミング

(1) プロジェクトを作成. C ソースファイルは serial_snd.c と sci.c, ヘッダファイルとして sci.h を追加する.

(2) 動作をターミナルソフト（RealTerm 等）で確認.

(3) なおシリアル通信と PICKIT2 によるプログラム書き込みは干渉するので，**動作させるときには必ず PICKIT2 を抜く**こと.（今回の回路では特に PC→PIC の信号が干渉する）

課題6　シリアル通信の送信データをオシロスコープで観察する．観察しやすいように送信データを 0b10101010(二進表現．10 進では 170)として putch を連続して行う.

・PIC のどのピンを観察すればよいか？
・観察の結果パルス幅は何μs だったか？
・その値はシリアル通信速度(9600bps)と比較して妥当か？何%程度の誤差があるか.
・またそのパルス幅は sci.c 中の関数 putch でどのように実現されているか.

課題7　逆に PC からのデータを PIC で受信するプログラム serial_rcv.c を動作させてみる．これは sci.c 中で定義されている関数 getch を使い，PC から PIC に送られたデータによって LED の点灯，消灯ができる．動作を理解する.

課題8　getch の代わりに getche 関数を使い，PC から PIC に送られたデータが PC にエコーバックされるようにする.

課題9　3 年生後期実験で使った Visual Studio のシリアル受信サンプルプログラムを用い，PIC からのデータを PC 側のプログラムで受信して表示する.

課題10　（宿題）書籍等でシリアル通信の規格について調べ，次のキーワードを理解すると共に観察した波形を理解する．また sci.c 中の関数 putch 中の実装を理解する.
スタートビット，ストップビット，パリティビット

(付記１)　今回のシリアル通信用関数はパルス幅の誤差がやや大きい．また実験的に求めた値を用いており，他の型番の PIC を使用する際にはパラメータチューニングが必要．その際には上記課題で行ったようなオシロスコープによる観察が必要となる.

(付記２)　本格的にシリアル通信を行う場合，(1)ソースコードのアセンブラ化，(2)タイマー割り込みによる正確なタイミング生成，が必要となるかもしれない．下記を参照（PIC 工作実験室）
http://www.picfun.com/serial21.html

(付記 3)　IC によってはシリアル通信をハードウエアでサポートしている場合も多い

6　3 日目：シリアル通信(2)：ロジック IC と 7 セグメント LED による数字表示

目標：少数のピンしかないマイコンを拡張する方法を学ぶ.

6.1　　回路製作

ここでは 7 セグメント LED(シャープ製 GL9A040G)で数字を表示させる．7 セグと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 5)．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.

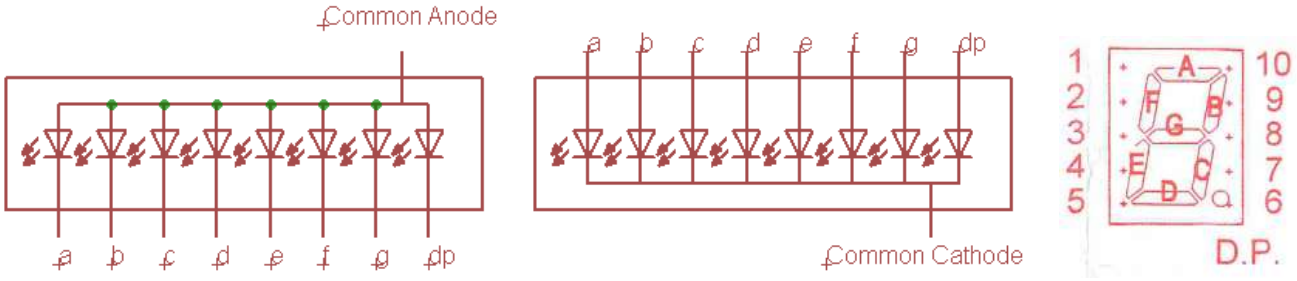


図 5　7 セグメント LED の内部構造．（左）アノードコモン，（中）カソードコモン，（右）LED の配置

8 つの LED を光らせる必要があるため，8 本のピンが必要である．しかし PIC12F683 は限られたポートしか持たない．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.
PIC の二つのポートから「クロック信号」「データ信号」を送出し，データをシフトレジスタで移動していく．これによりソフトウエア的に 8 つの LED を二つの信号線で駆動できる．(74HC164 にはラッチが無いため，データ移動の最中にも多少 LED が点灯することに注意)

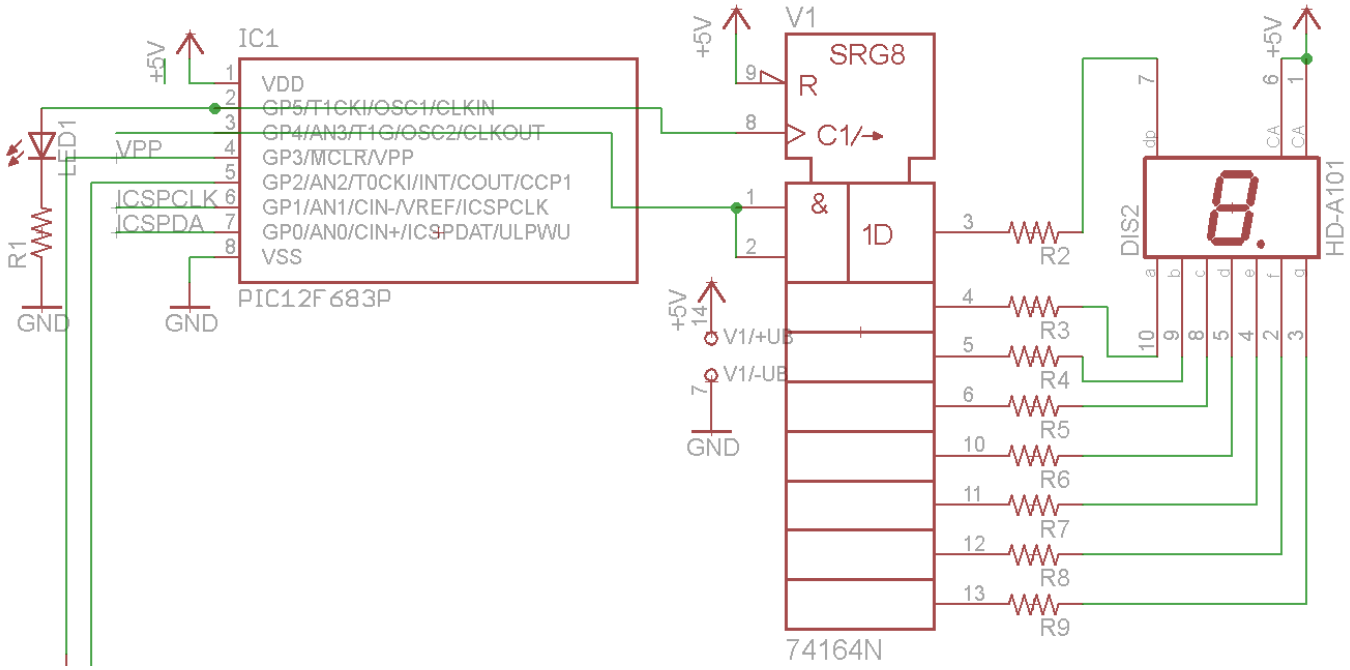


図 6 三日目課題：7 セグ LED 点灯回路（電源，書き込み，シリアル通信部は省略）

表 4 追加部品リスト

V1	74HC164	R2-9	1k～2k 程度
DIS1	GL9A040G		

- 回路保護のため，74HC164 と 7 セグメント LED は IC ソケットを介して半田付けする．
- 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っている．

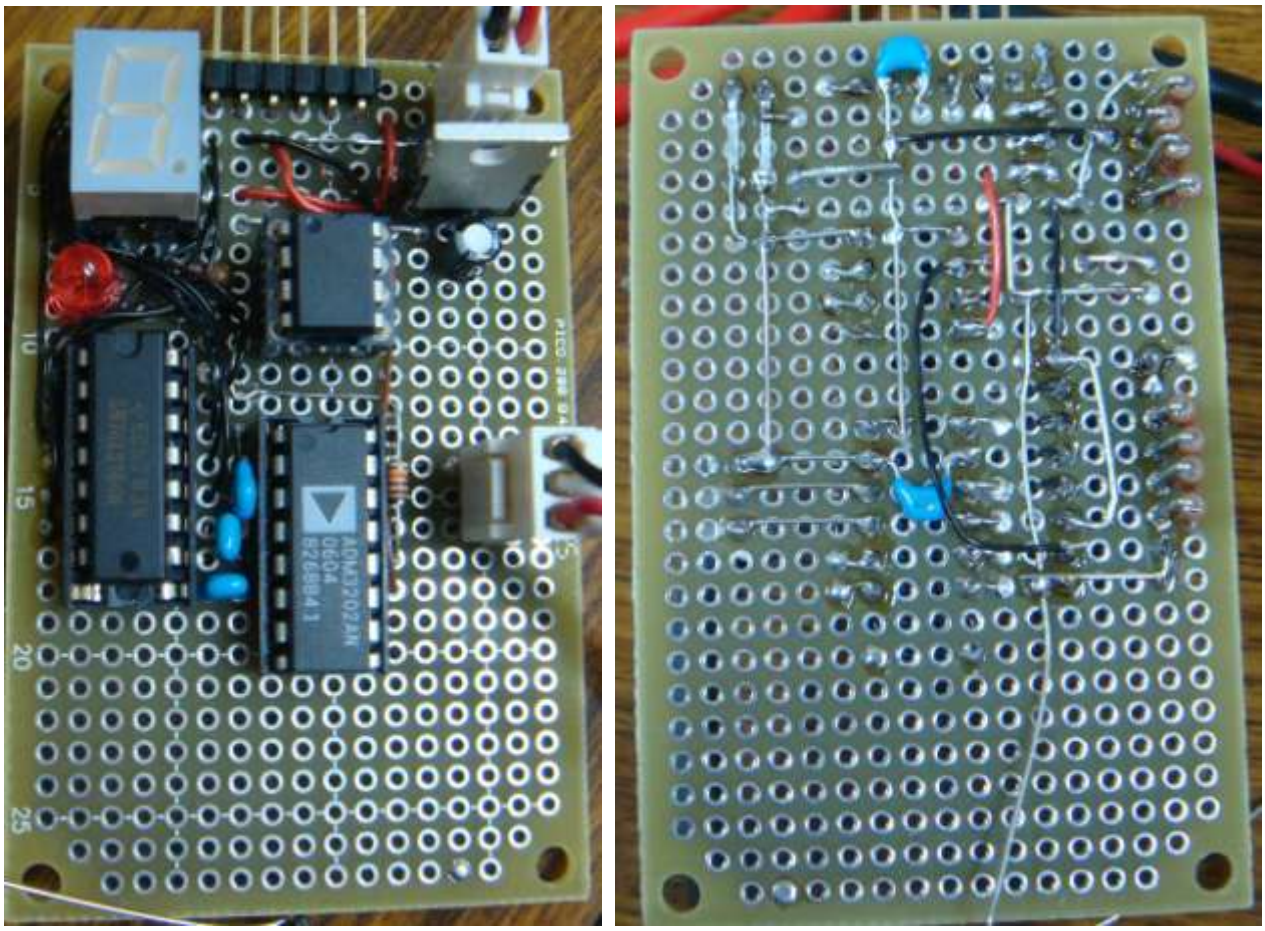


図 7　7 セグメント LED 回路. 基板が足りなくなったので抵抗を裏面に配置している. 74HC164 は 14 ピンだが, 16 ピンのソケットしかなかったため余っている. また 74HC164 を上下逆にしたほうが LED への配線は楽だった. 抵抗を裏につける必要はない.

6.2　　プログラミング

プロジェクトを作成．サンプルプログラム 7seg_led.c を書き込み，LED が点灯，移動することを確認する．

課題11　オシロスコープで観察する.

オシロスコープの入力を 2ch 用い，データ信号(PIC の 3 番ピン)とクロック信号(PIC の 2 番ピン)を同時に観察し，データとクロックのタイミングがシフトレジスタの仕様に合っていることを確認する．

課題12　数字を表示するプログラムを作成する.

ある数字を表示するにはどの LED を点灯したらよいか．またそのためには，シフトレジスタにはどのようなデータを流し込めばよいか．関数化するとよい．例えば Display(0)とすると 0 が表示されるようにする．

課題13　RealTerm によって PC から PIC に数字を送り，PIC 側でその数字を表示する

sci.h，sci.c をプロジェクトに追加し，メイン関数にシリアル通信用の初期化プロセスを加える（I/O 方向の設定に注意）．さらに getch 関数を用いて PC からのデータを受信するように改変する．

課題14　（宿題）汎用ロジック IC について本などで自習する．特にシフトレジスタの働き（クロック信号とデータ信号のタイミング）については 74HC164 のマニュアル中のタイミングチャートを読むことで理解する．さらに余裕があれば CPLD，FPGA についても自習するとよい.

（参考）汎用ロジック IC の代表である 74 シリーズは，情報理論基礎で学ぶようなデジタル回路の基本論理回路（AND，OR，Flip-Flop など）が入っているもの．マイコンの高性能化，低価格化に伴い相対的に学習の機会が減っているが，高速性（クロックに左右されないので 10～1000 倍程度まで可能）が求められたり，マイコンの限られたピン数を増やしたりする場合に必須. 今回用いたシフトレジスタは通信線を減らせることから多くの外部モジュールに組み込まれており，応用範囲が非常に広い.

課題15　（宿題）少数ピンで多数の素子を駆動する別の方法として，74HC138(3bit→8 ライン・デコーダ)を用いることも考えられる．また同じシフトレジスタでも，ラッチ付きの 74HC595 を用いるのも一般的. それぞれ機能を調べ，今回の場合と比較して得失を考察する．ポイントは必要ピン数，拡張性など.

4 日目は二人一組となり，以下の 2 つのうち一つをそれぞれ行う

- 加速度センサ：傾きを計測し，PC に送る．
- モータ駆動：トランジスタによる H ブリッジ回路を作成し，PC のキーボードで操作してみる．

7　4 日目(1)：加速度センサ

7.1　　加速度センサ回路の製作

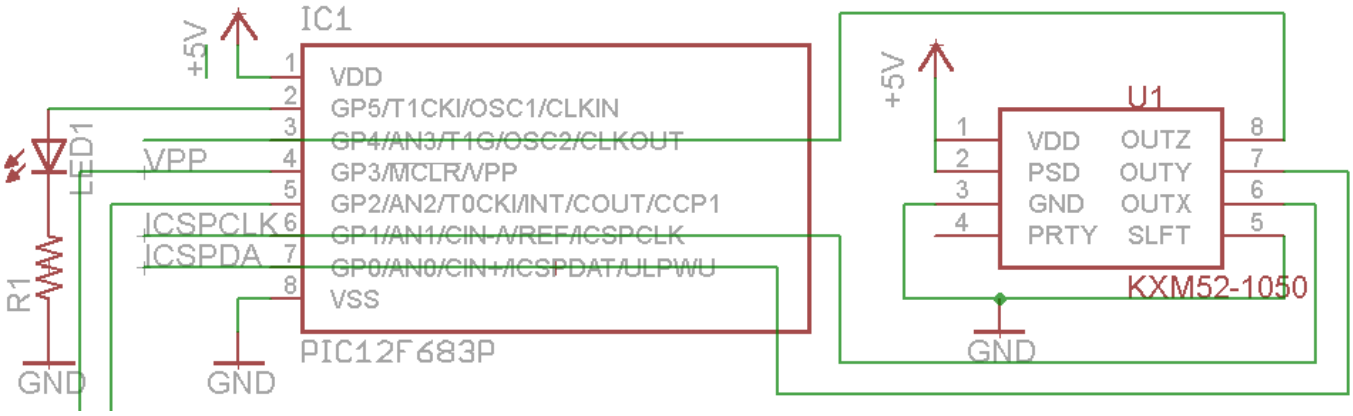


図 8 四日目課題①：加速度センサの読み取り（電源，書き込み，7 セグ，シリアル通信部は省略）

表 5 追加部品リスト

U1	秋月製加速度センサユニット KXM52-1050

- 加速度センサは高価（一個 1100 円！），かつプログラム書き込み時は（今回に限り）はずす必要があるため，**IC ソケットを履かせて使う**．
- シフトレジスタのデータ線と加速度センサの入力が同じポートを使っている．シフトレジスタを IC ソケットからはずす．

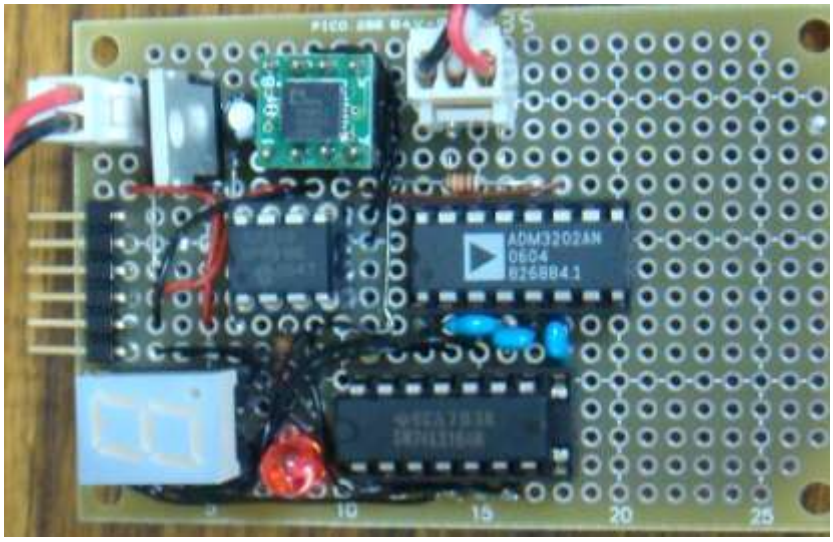


図 9　加速度センサを取り付けた様子（7 セグメント LED とシフトレジスタは本来不要．抜いておいてよい）

7.2　　プログラム

PIC のプログラム ad_conv.c，sci.c，sci.h でプロジェクトを作成し，PC 上の RealTerm で傾き信号を読んでみる．

※書き込みの注意：今回使用している PIC の AD ポートには加速度センサの出力が繋がれており，それが邪魔をして書き込みができない．書き込みの際には加速度センサをはずし，終了後取り付ける．

課題16　PC のプログラムで傾きを読む．現在のプログラムでは AD 変換の結果は 8bit と粗く，どのチャンネルから送信されたものかもわからない．PC, PIC 双方のプログラムを改良することで，10bit のデータを，チャンネル識別も含めて伝達できるようにする．

例えば 10bit のデータを上位 5bit，下位 5bit に分割し，それぞれ 1byte データとして送信する．このとき 3bit が余るから，そこにチャンネル番号(0～3 の 2bit)と上位下位の識別信号（1bit）を加える．PC 側のプログラムでは 1byte ずつ受け取るたびに解読する(図 10)．

なおこの考え方は一例であり，データ送信の順番がはっきりしていればデータ送信開始を識別する信号（例えば 2byte 連続 0xFF）だけで十分．

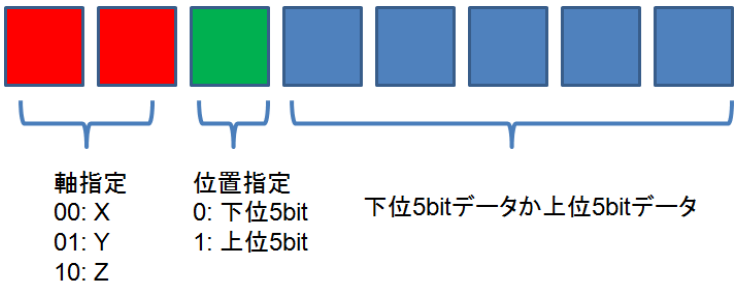


図 10　10bit データの送信例．加速度センサの軸情報も入れる場合．

課題17　(宿題)PICKIT2 と周辺回路の干渉の現象を知ったところで，下記の PICKIT2 に関する解説を読む．回路を自分で設計する際に必要になります．

http://www.ne.jp/asahi/air/variable/picmel/integration/write/pickit2/index.html

（補足）本来 ICSP(In circuit serial programming)は，マイコンを基板に刺したままで書き込み，動作，デバッグまでできることがメリット．今回のような状況は本来望ましくありません．

8　4 日目(2)：H ブリッジによるモータ駆動

8.1　　モータ駆動回路の製作

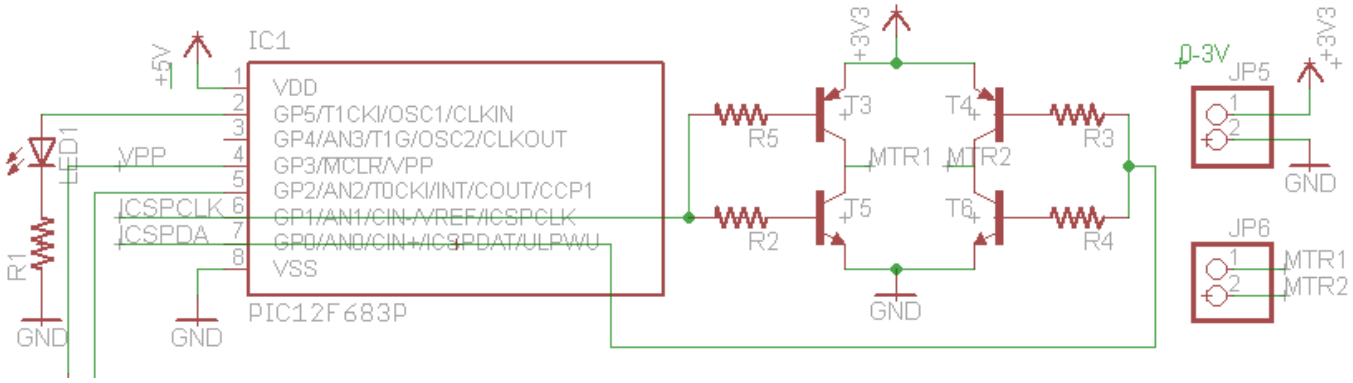


図 11　四日目課題②：モータドライバ（電源，書き込み，7 セグ，シリアル通信部は省略）

表 6 追加部品リスト

T3,4	PNP トランジスタ 2SA1015	T5,6	NPN トランジスタ 2SC1815
R2,3,4,5	1kΩ	JP6	モータ接続用コネクタ
JP5	モータ駆動用電源コネクタ		小型のマブチモータ(RF500T)

- 単電源で DC モータに加える電圧の向きを変えられる回路(モータドライバ)として最も基本的なものが H ブリッジである．図 12 のように 4 つのスイッチから構成され，スイッチの切替えによってモータに流す電流の向きを反転可能にしている．名前はスイッチの配置が H 型をしていることに由来する．
- 今回は NPN トランジスタと PNP トランジスタで H ブリッジを構成している．
- NPN トランジスタとして 2SC1815，PNP トランジスタとして 2SA1015 を用いる．トランジスタの最大定格は 150mA なので，モータを ON し続けると発熱して壊れる．よって駆動は 1 分程度とすること．(本来はその辺も考慮してトランジスタを選定する)
- 実用上は H ブリッジの内蔵されたモータドライバ IC を使う．安いモータドライバ IC はスイッチ切り替えの速度が遅く，PWM 駆動が出来ない事が多いので注意．研究室では東芝 TA8440 を標準としている．
- **モータ駆動用の電源電圧は適正な値がわからないので 0V から最大 5V まで徐々に上げていくこと．**
- モータ駆動用の電源を別に取っている．別に一緒で良い場合もある．別々にしている理由は次のとおり
  - モータ駆動電圧とロジック回路の電圧が異なるため
  - ノイズを分離したいから

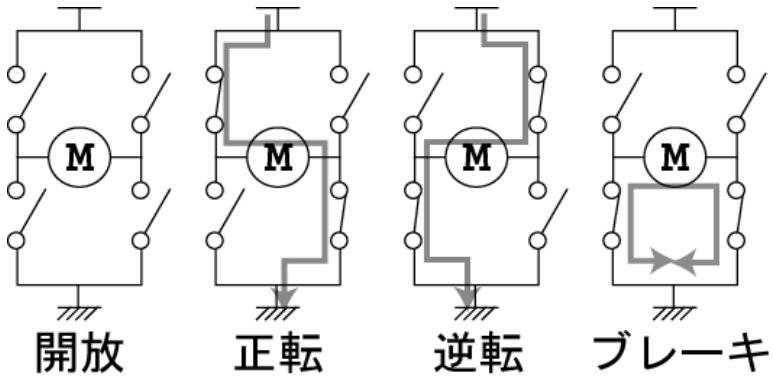


図 12　H ブリッジ回路の基本動作

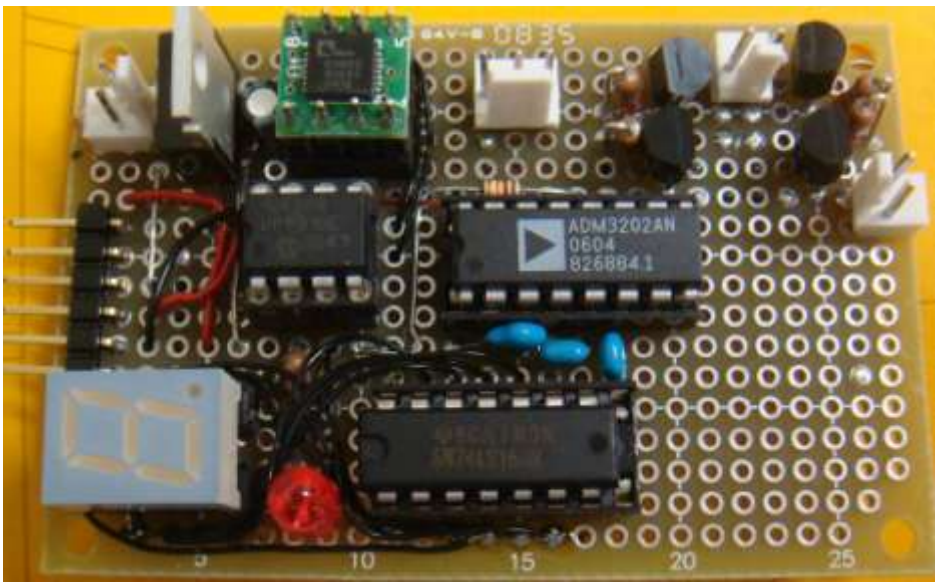


図 13　H ブリッジ回路（右上）．テキスト製作の流れで加速度センサも付いているが不要（電気回路的に干渉するので外す必要あり）

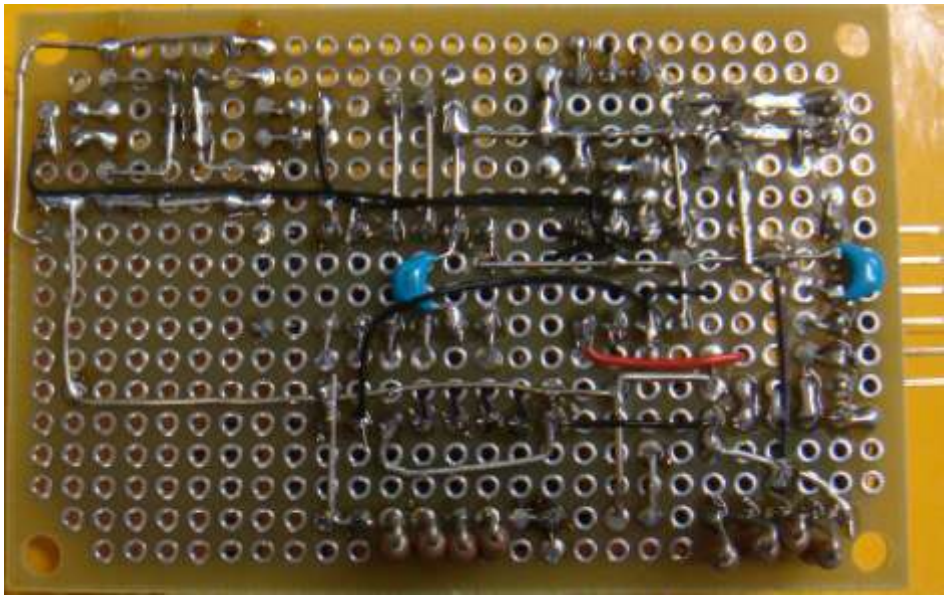


図 14　裏側の様子

8.2　　プログラミング

まず h_bridge.c でプロジェクトをつくり，動かしてみる．1 秒ごとに動作が変化するプログラムになっている．モータの電源電圧を徐々に上げていき，動作を確認する．

課題18　オシロスコープで H ブリッジの入力(PIC マイコンの 6,7 番ピン)の電圧を観察し，それとモータの動作を比較して，H ブリッジの動作を確認する．

次に h_bridge_serial.c，sci.h，sci.c でプロジェクトを作り，シリアルコンソールから動作させてみる．

このプログラムは割り込みを使ったシリアル通信データの受信の例になっている．

PIC12F683 はシリアル通信が純粋にソフトウエアで実現されているため，いつ PC からデータが来ても受け止められるようにするには，常時シリアル入力ポートを監視しなければならない（より上位機種では専用のメモリを持っている）．
しかしそうするとモータを動かせなくなってしまう．だからシリアル入力ポートの電圧が下がる（＝スタートビット）イベントを割り込み要因とすることで，常時監視の負荷を無くしている．

今回のサンプルではモータの PWM 制御はメイン関数で記述している．PWM 制御は専用のモジュールを使うことができる（H8 マイコンでやったように）ので試してみるとよい．またタイマー割り込みを使うことも考えられる．しかし PICC-Lite では割り込み関数は一つしか用意できない（割り込み要因別に処理を変えることは可能）ため，「タイマー割り込みの処理中にシリアルデータが来た」ような場合，シリアル受信の処理はタイマー割り込みが終了した後に行われてしまう．これが今回の PWM がメイン関数中で書かれている理由である．（上位機種では割り込みの優先順位をつけられるものがある）

課題19　オシロスコープでモータの両端の電圧を測り，所望のパルスが（電流の方向も含めて）出ていることを確認する．

課題20　PC のプログラムからシリアル通信でモータを動作させる．Unsigned char(0～255)型でデータを送り，値に対応して速度を変えて 2 方向に回転するようにする．

課題21　課題 13 で作成した「PC から PIC にデータを送信して 7 セグメント LED に数値を表示する」プログラムを，ディジタル入力割り込みを使ったものに変更する．（これまでのものは時々うまく通信できなかったはず）

課題22　タイマー割り込みを使ってみる．必要な情報はほぼ init_interrupt 関数中に書いてある．「C 言語ではじめる...」の 3.5 節に記載のサンプルを参考に，1ms ごとに LED を点滅させ，オシロスコープで観察してみる．
（「C 言語ではじめる...」はコンパイラのバージョンの違いからレジスタ名が若干異なることがある．ヘッダファイルを参照すること．

9　5 日目：無線通信と PC 間通信

最終日は，前日まで作ったものに対して Bluetooth によってシリアル通信を無線化する．さらに Windows のファイル共有で, 相手の PC 上にあらかじめ作成しておいた共有ファイルを読み出すことで PC 間通信を行う．これにより，加速度センサとモータ駆動を連携し，遠隔操縦を実現する．

図 15　無線コントローラ―PC－PC－ラジコンのイメージ．

9.1　　無線用回路（その 1 ：設定用）

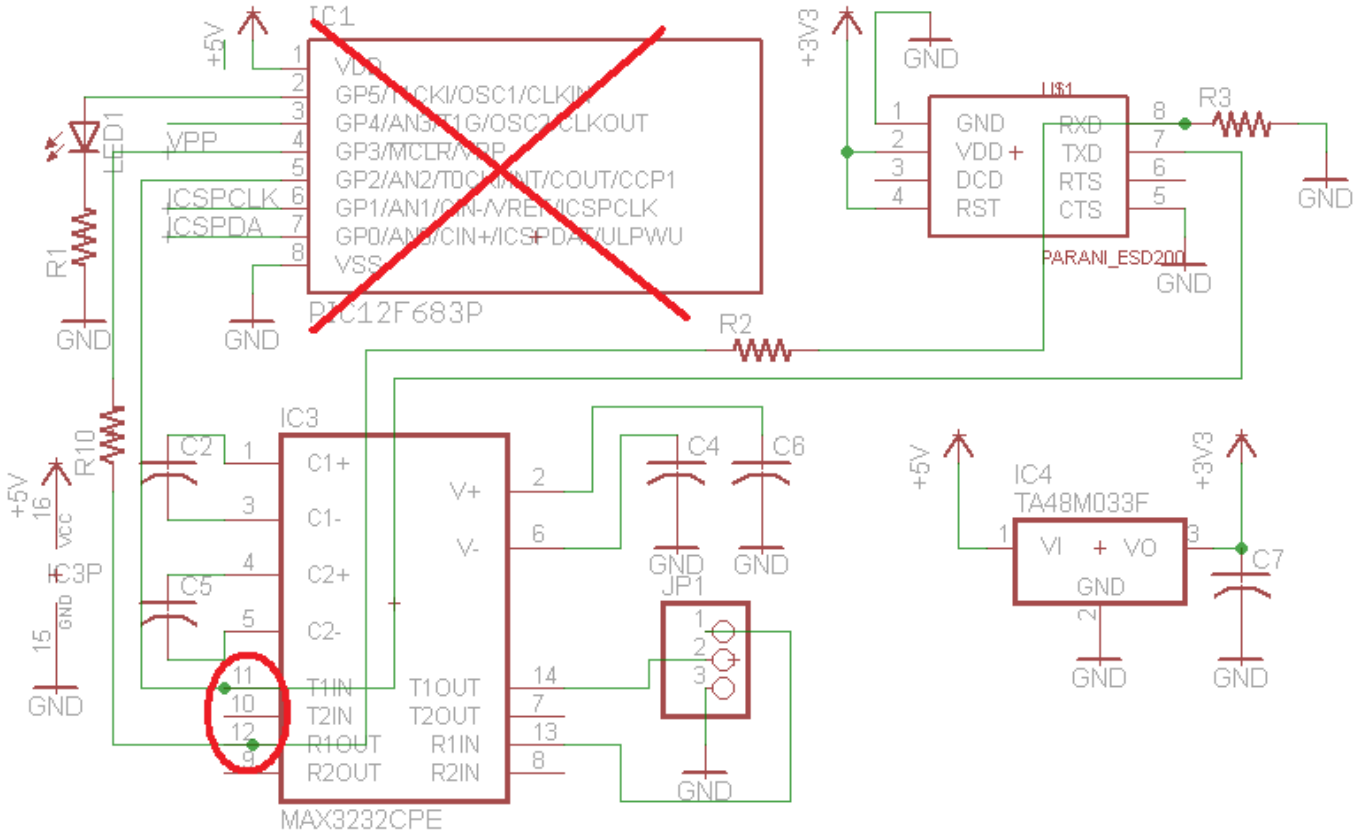


図 16　無線モジュール設定用回路．PIC はソケットから外す．

表 7 追加部品リスト

U$1	PARANI ESD200	IC4	3.3V 用 3 端子レギュレータ TA48M033F
R2	68kΩ	C7	40μF 以上の電解コンデンサ

R3	110kΩ		

- 無線化にはシリアル通信を Bluetooth 化するモジュール(Parani ESD200)を用いる．1 万円以上するモジュールなので下記の回路上の注意をよく読み，理解したうえで細心の注意で扱う．
- まず無線モジュールをシリアル通信用電圧変換 IC(ADM3202)を介して PC に接続．これは無線モジュールの設定を変更するためで，一回やってしまえば（通信速度を変えない限り）やらなくてよい作業．
- 通常は本通信モジュール専用のジグ基板でこの設定操作を行うが，今回はジグ基板を買っていないので PIC 回路を流用．ただし PIC は外す．
- モジュールの 8 番ピンと 7 番ピンはそれぞれモジュールから見て入力，出力．PIC の場合と同様に ADM3202 に接続．
- 無線化モジュールは 3.3V の電源で動く．このため 3.3V の 3 端子レギュレータを新たに設置．今回は東芝製の TA48M033F を用いる．この三端子レギュレータの入力は直接外部から 7V を加えても問題ない．
- 今回はすでに作成している 5V を加える．低ドロップタイプなので電圧的にも問題なし．この三端子レギュレータは出力段にコンデンサをつける必要があることに注意．
- ADM3202 は 5V で駆動されているのに対して無線化モジュールは 3.3V で駆動されているため，電圧レベルの変換が必要．具体的には無線化モジュールへの入力信号は電圧を下げる必要．
- このため，R2 と R3 で分圧．R2=68k，R3=110k だから，PIC からの 5V の信号は無線化モジュールで約 3V となって受け取られる仕組み（図 17）．

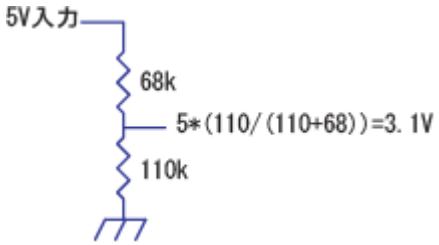


図 17　電圧レベル変換：5V→3.3V

このような「5V⇔3.3V」の電圧レベル変換の問題はディジタル回路で非常に多く見られる．

ただ実は，PIC12F683 も 3.3V で動き，ADM3202（シリアル変換）や 74HC164 も 3.3V で動く．だからはじめから，すべての電源供給を 3.3V で行っておけばよかった，というのが正解．その場合にはレベル変換が必要ないため，R2，R3 による分圧は不要．
（5V を基本とするものが多い（安い）ため実習としては 5V としていた，ということ．今後研究室も 3.3V 系に移行していくかも．）

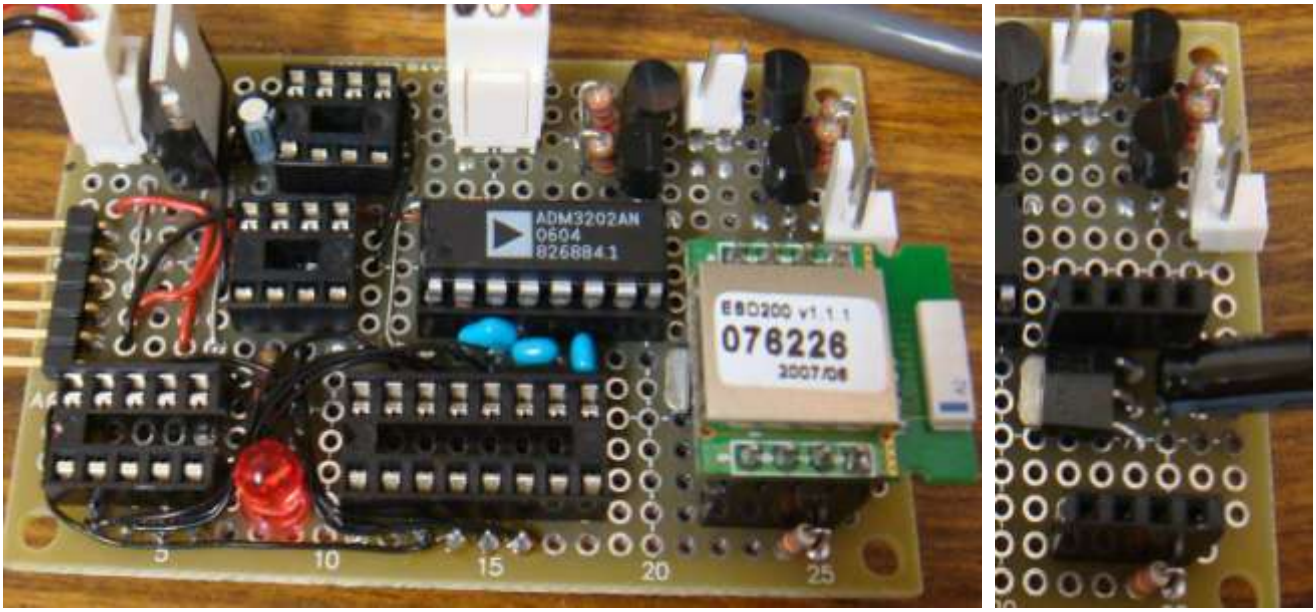


図 18　無線モジュール書き込み回路．(左) PIC は外す．(右) 無線モジュールの下に 3.3V 用 3 端子レギュレータ．

9.2　　無線モジュールの設定

- 無線モジュール供給元のページ http://www.sena.com/support/downloads/　から無線モジュール設定用のプログラム ParaniWIN をダウンロード，インストール．
- シリアルケーブルで先の回路をつなげ，ParaniWIN を起動し，接続されている COM ポートを選択すると，無線モジュールが自動的に認識される．（ESD200…という名前になるはず）

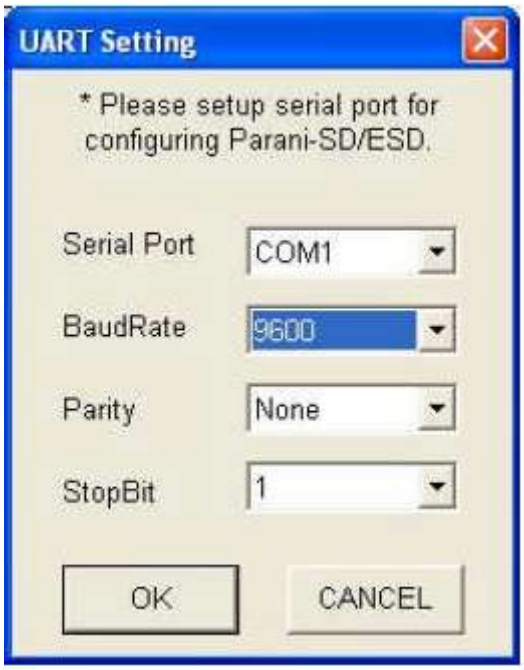


図 19　ParaniWIN 起動画面

- 次に画面左の「Device Setting」を選ぶ．この画面で次の設定を行う．
  - ✓ Operation Mode を「３」とする．
  - ✓ Security Option を「Authentication」とし，Pin Code を「0000」とする．
  - ✓ Hardware Flow Control を OFF にする．

✓ Command Response を OFF にする.

その後 Apply ボタンを押すとデバイスの設定完了．電源を抜いてもこの設定は保存される.

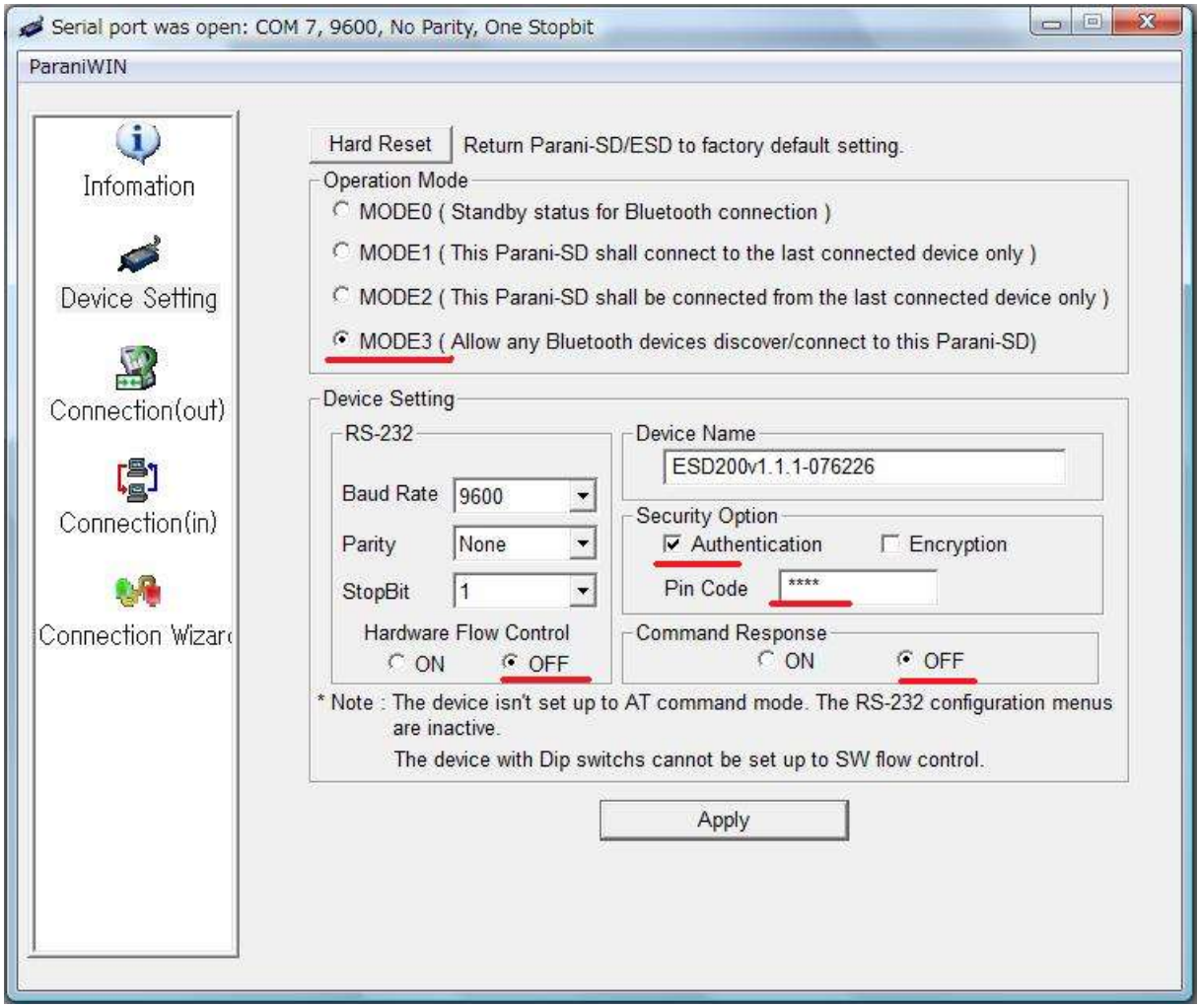


図 20　ParaniWIN の設定画面

9.3　　無線用回路（その２：通信用）

モジュールの設定が終わったので Bluetooth 通信ができるように回路を組む. ほとんど先ほどの回路と同じだが以下の点が異なる.

- 先ほどは PC と通信モジュールをつなげたが，今度は PIC と通信モジュールをつなげる．このとき **PIC の「入力・出力」に通信モジュールの「出力・入力」をつなげることになるので，先ほどと配線が逆になる**．図で丸で囲んだ部分が微修正箇所.
- 今度はシリアル通信用の IC が不要となるので，ADM3202 を外す.
- PIC を元に戻す.

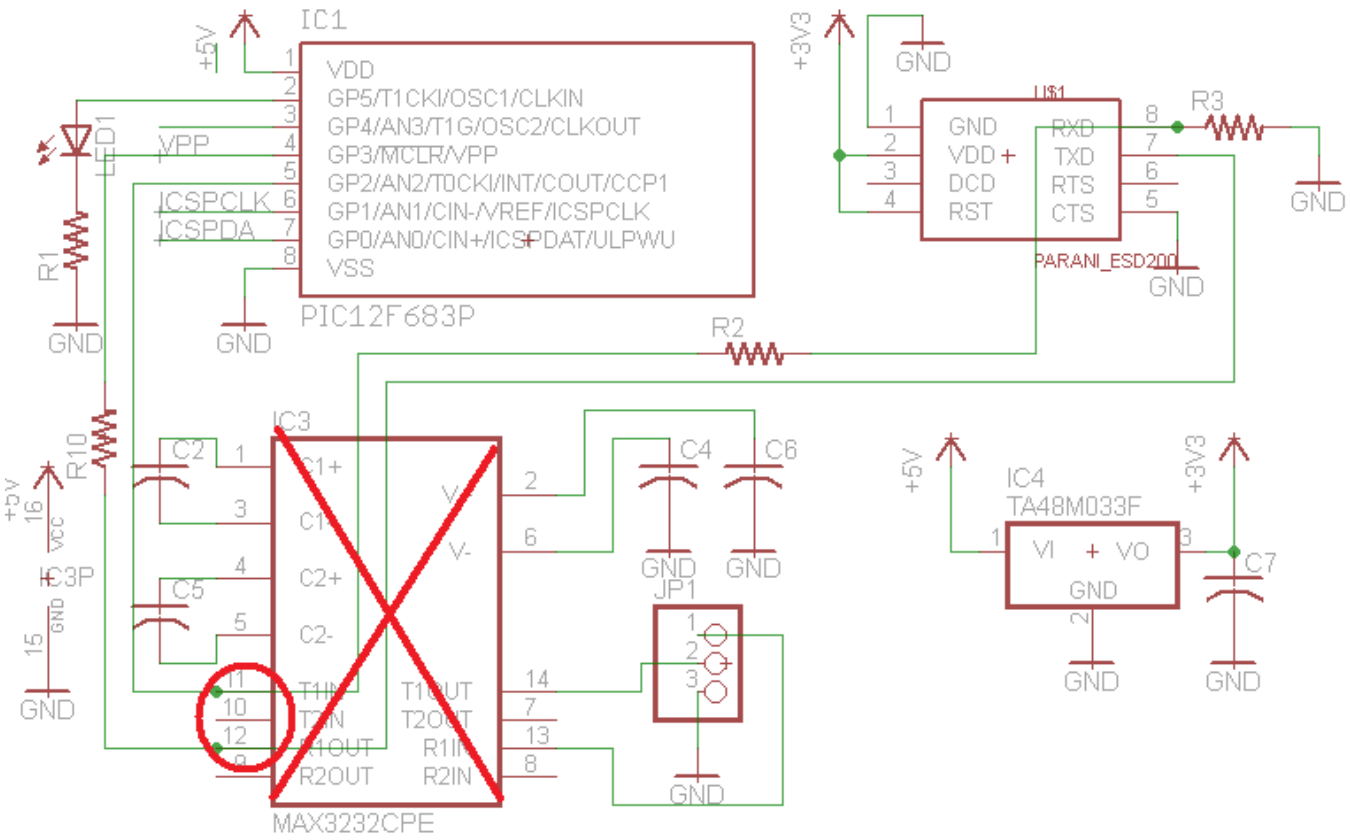


図 21　Bluetooth 通信回路．配線を修正し，シリアル通信用電圧コンバータ(ADM3202)は外す．

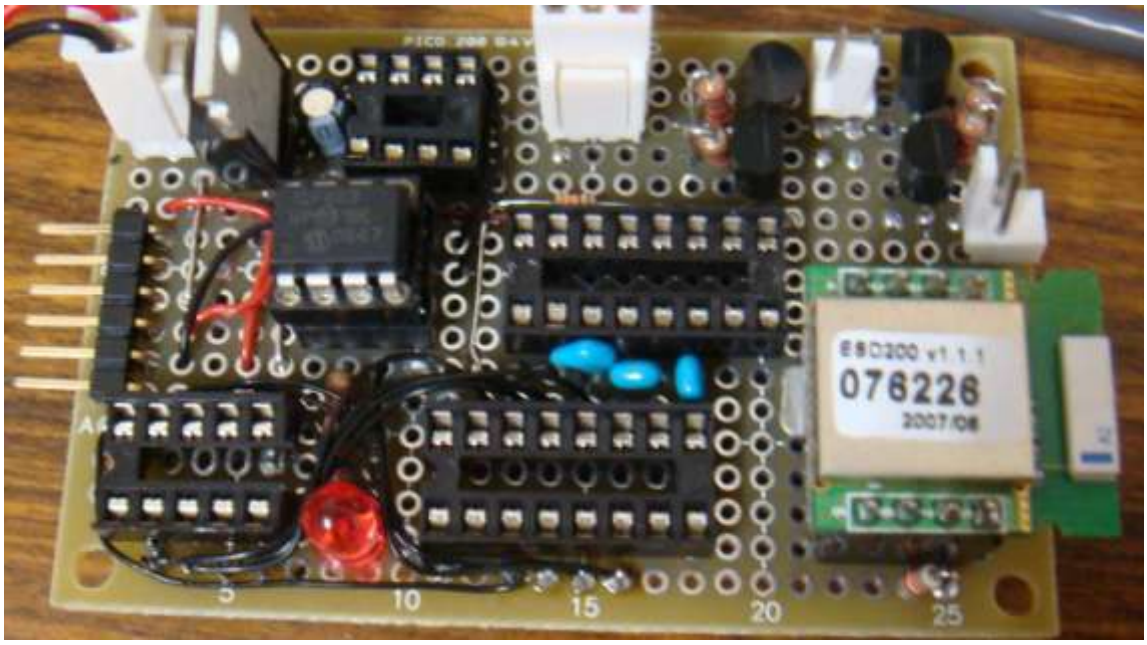


図 22　無線モジュールと PIC．シリアル通信用電圧コンバータ（ADM3202）は外す．

9.4　　プログラミング

まず PC 側の bluetooth 通信を準備する．研究室で用意した USB-Bluetooth 変換モジュールを，~~ドライバインストール後に~~ USB ポートに刺す．（追記：純正のドライバでは後のシリアル通信プログラムがうまくいかないため，補遺を参照して Windows の標準のドライバを入れること）

以前作成した適当なシリアル通信用プログラムを書き込んで試す. 加速度センサを使っている場合は書き込む際に外すのを忘れないこと.

（以後不具合発見のため削除．補遺を参照すること）~~書き込み後，PIC 側の電源を入れる．その後 PC のタスクトレイに現れている Bluetooth マネージャ（図 23）を右クリックし，「Bluetooth 設定」で，ESD200...を選択．パスキーは先ほど設定した「0000」を入力．（Bluetooth マネージャはなぜか二つ現れることがあるが，明るい色のほう）~~

~~すると，「○番のシリアルポートが追加されました」と表示されるので，その番号をメモ．これ以外に大量の bluetooth 用シリアルポートがシステムに追加されてしまうが，デバイスマネジャーで削除してかまわない.~~

~~図 23　Bluetooth マネージャ~~

Realterm で通信を試みる．Realterm は立ち上がりの際にすべてのシリアルポートをスキャンするため，立ち上がりに時間がかかる．図 24 のように警告が出ることがあるが Abort してかまわない.

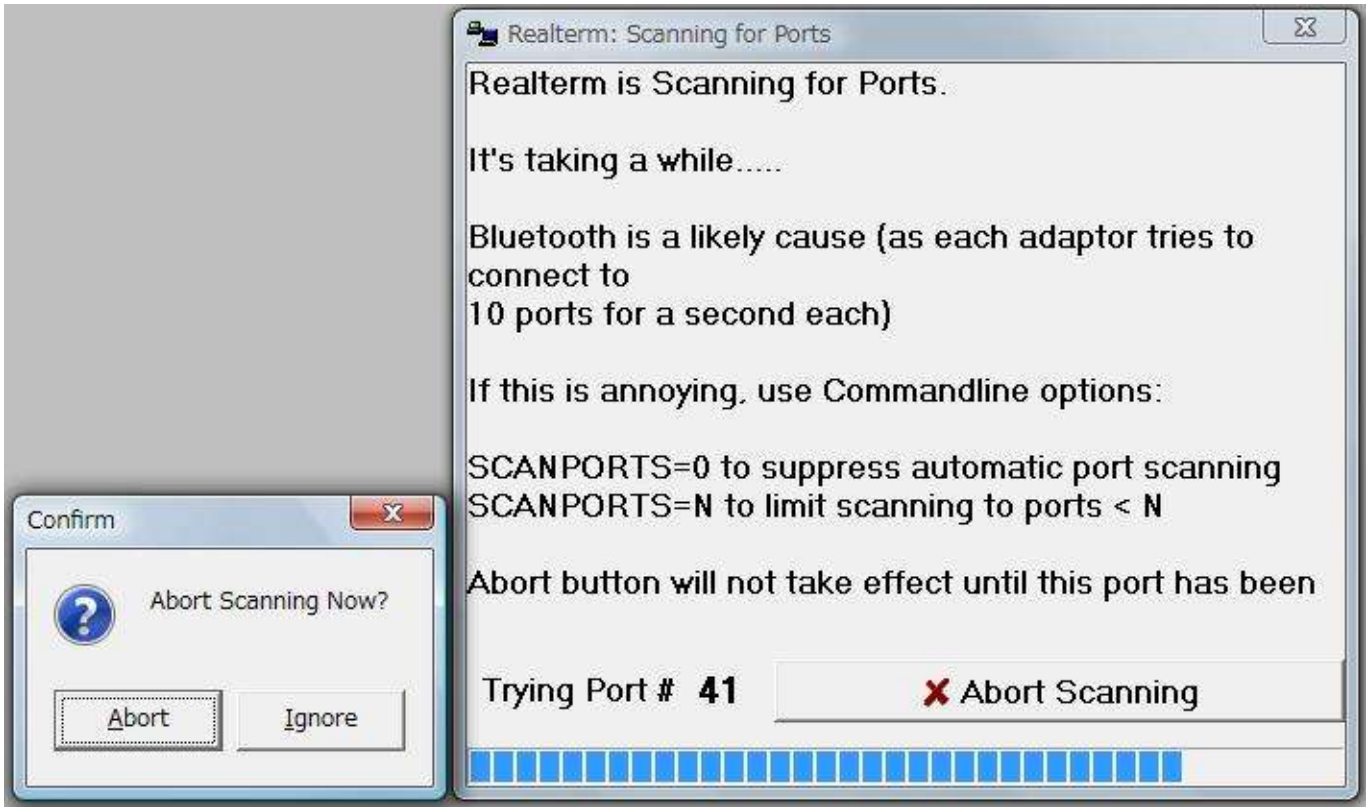


図 24　Realterm の警告

（補足）これで一応の Bluetooth を使ったシリアル通信ができた.

いったんここまでできれば，次からは無線通信モジュールの設定という手間は不要．場合によってはシリアル通信用の IC（ADM3202）を最初から省いてしまってもよい（デバッグ用に必要だが，小さくしたい場合は IC は外付けにして対応すれば 3 ピンを立てるだけですむ(RxD，TxD，GND)．
今回作成した回路は，無線モジュールの設定も含め色々な用途でデバッグ用に使えるので取っておく．

9.5　PC 間通信

すでに他の講習で経験している「ファイル共有を用いた PC 間通信」を行う．PC_COMMUNICATION フォルダ中のサンプルを使う．その後パートナーと組み，片方のマシンに共有フォルダを作成し，そこに作ったファイルを使って通信する．

ここまで出来たら，これまでの成果である加速度センサ，モータをそれぞれにつなぎ，PC 間で遠隔操縦する．

課題23　（自由課題）自由課題の時間がどうしても取れないようなので，5 日目の成果をなるべくブラッシュアップしたものを二人一組で発表会では見せてください．もちろん好きに発展，応用歓迎です．

課題24　(興味のある人)他の型番の PIC を使ってみる．例えば研究室では 16F88 を常備しています．
ピンの数は少し増えるだけだが，使用可能な機能は増え（シリアル通信，PWM などのハードウエアサポート），ポートの名前も変わる．このため今回のサンプルプログラムを隅々まで理解して変更しないと動かず，苦労する反面よい復習になるはず．新しいマイコンを自分で導入できるようになれば選択肢の幅は一気に広がる．
（本に載っている製作例も多いので，12 シリーズより 16 シリーズを研究室標準にしたほうがよかったかもしれない．誰かが 16 シリーズで講習会資料を作ってくれることを期待）

10　補遺：研究室の Bluetooth-USB アダプタのインストール方法

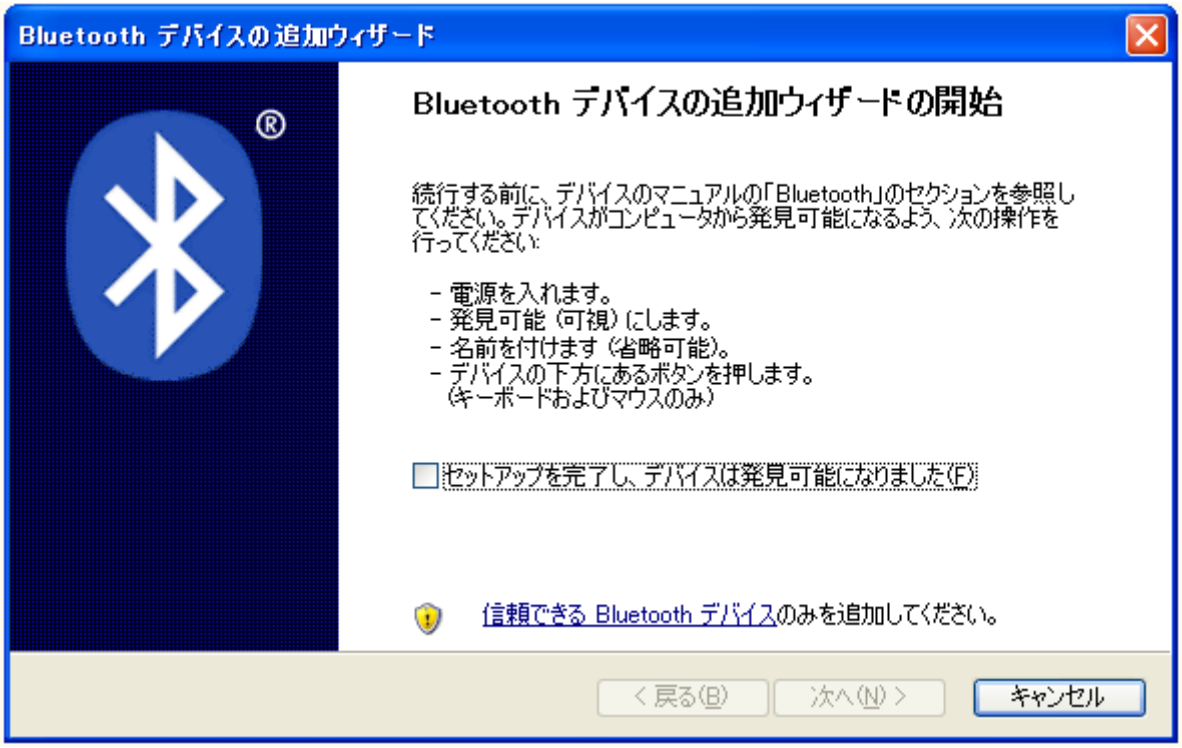
付属の CD でドライバをインストールした場合, シリアル通信プログラムがシリアルポートを発見できないことが分かったので，Windows 付属のドライバを使う．

東芝のドライバがインストールされている場合は，コントロールパネルの「プログラムの追加と削除」で「Bluetooth Stack for Windows by Toshiba」を削除する．「Bluetooth デバイスを削除しますか」と聞かれた場合は Yes とする．

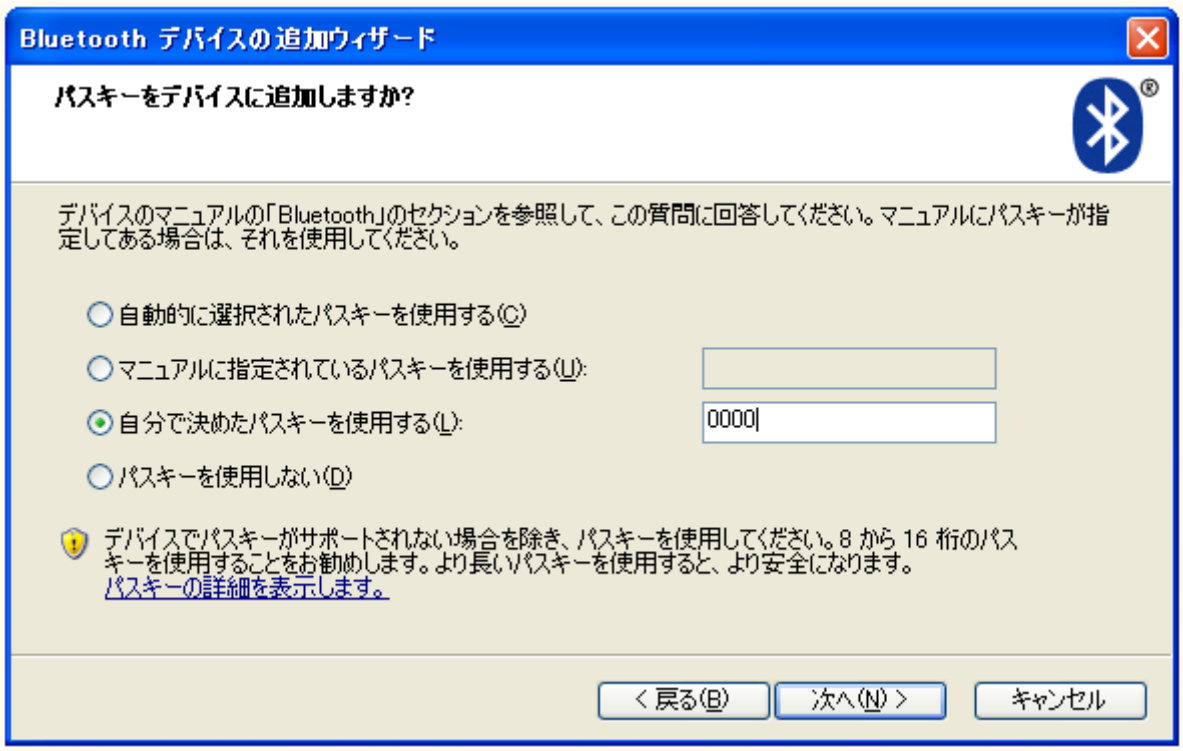
USB-Bluetooth モジュールを USB ポートに差し込む．Windows は自動的に標準のドライバをインストールする．

Windows の通知領域の「Bluetooth デバイス」を右クリックして，「Bluetooth デバイスの追加」
このアイコンは Bluetooth アダプタを入れることで表示されるが，もし表示されていなければ，コントロールパネル⇒プリンタとその他のハードウエア⇒Bluetooth デバイス

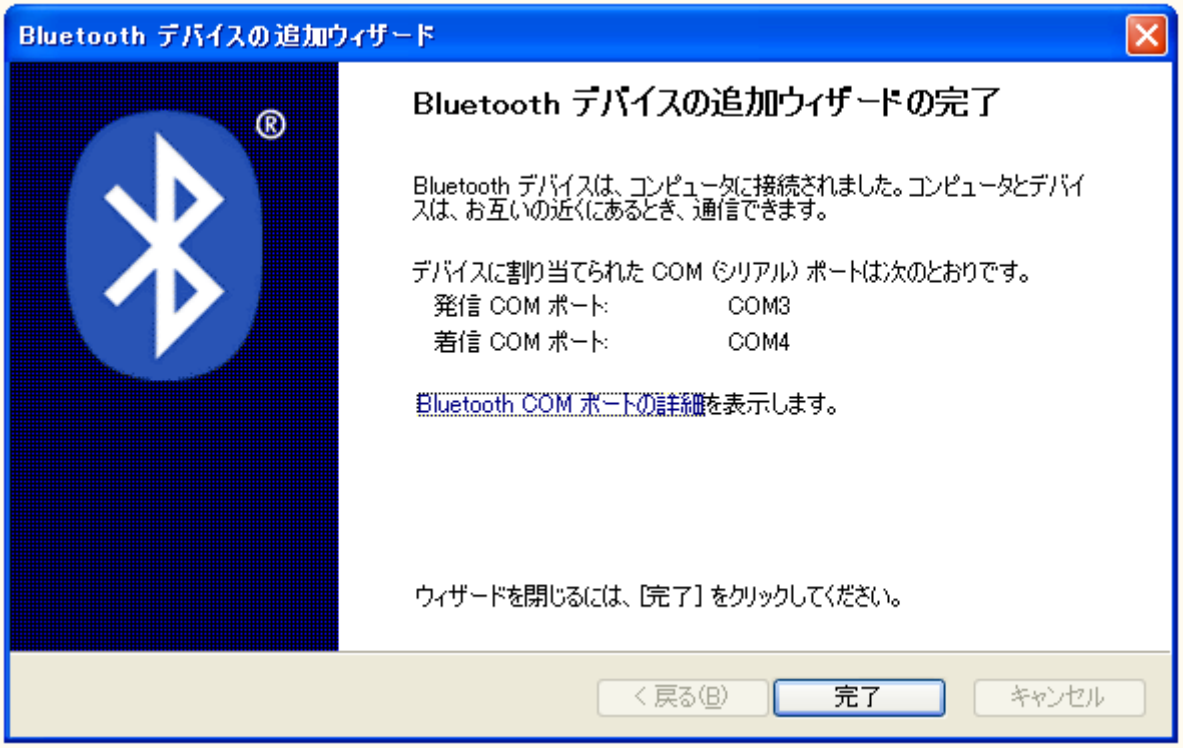
「セットアップを完了し，デバイスは発見可能」をチェックし，次へ

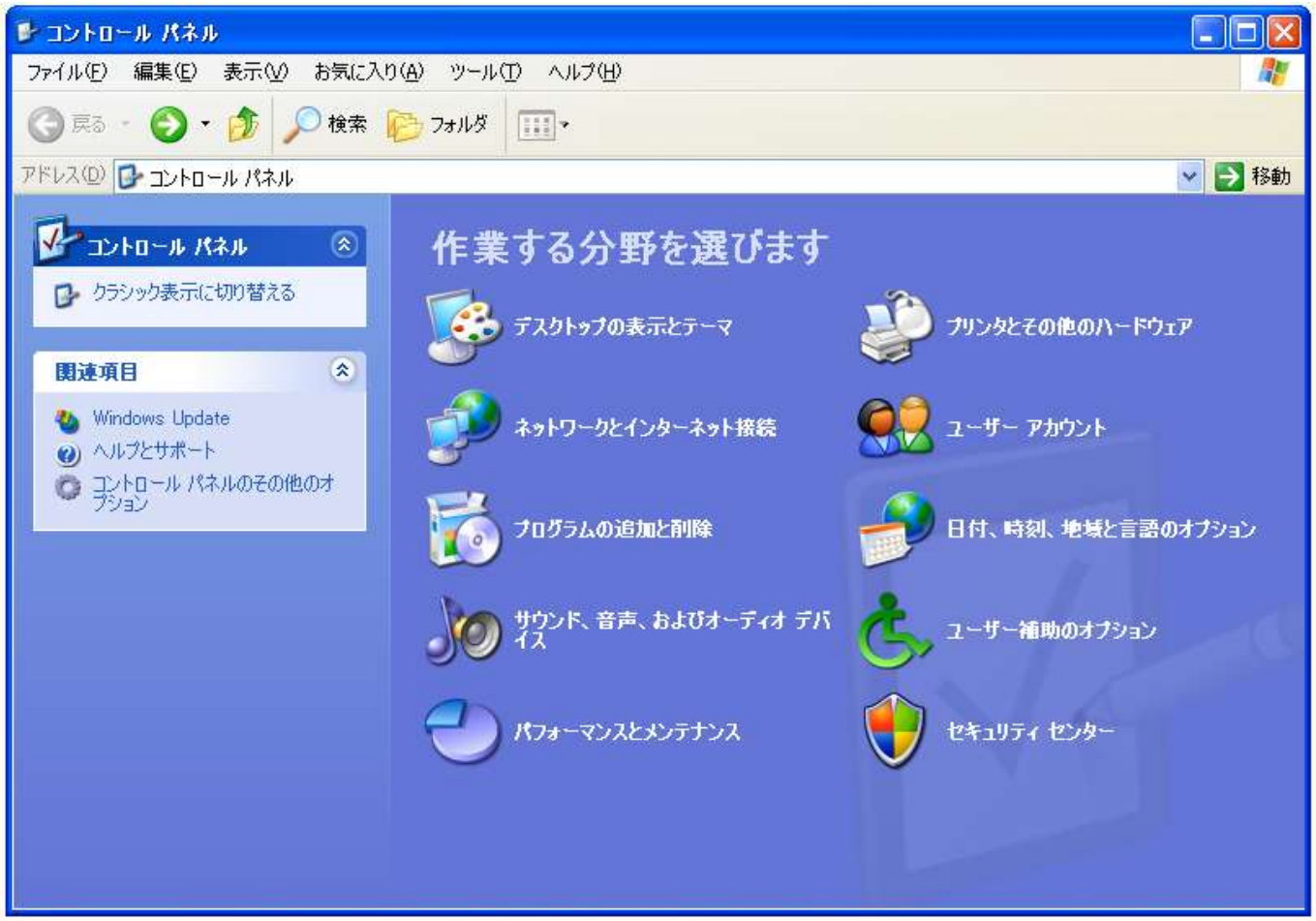


「自分で決めたパスキーを使用」：0000 を入力



発信 COM ポート，着信 COM ポートが表示される．**実際に使うのは「発信 COM ポート」**．この例では COM3





コントロールパネル⇒プリンタとその他のハードウエア⇒Bluetooth デバイスで確認すると，先ほどの発信 COM ポートが Generic Serial となっている