情報工学実験（3 学年）ライントレーサの製作挑戦

１．はじめに

（１）ライントレーサとは，図１．１のように平面上に引かれた線の上を自動追尾して移動する車輪駆動ロボットである。平面は白色，線は黒色。線幅 18mm 程度とする。この課題では 2 つのモータを左右に持ち，その回転数を制御して，進行方向を制御する車輪ロボットを製作する。

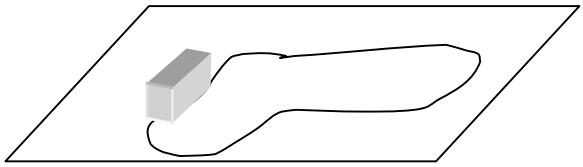


図１．１　ライントレーサ

（２）この課題は授業で習ったことだけで実現できるようになっていない。挑戦する心を育てる課題である。知識や技術を与えてもらうという受身ではなく，自分のコンピュータスキルをアップする機会を得たという位置付けが大事である。授業で習っていない技術も使うことがあるので，心得ておくこと。

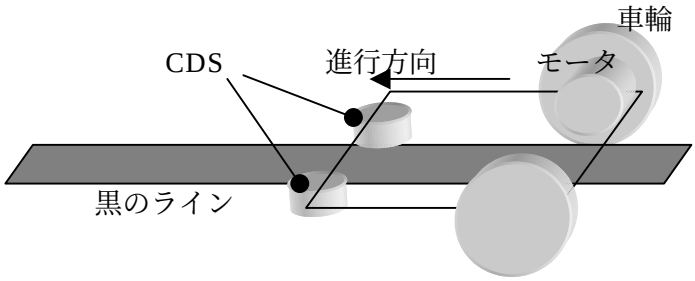


図２．１　ライントレーサの構想

（３）製作は 4 名で 1 台のライントレーサを製作し，コースでのタイムレースを行ない，最後は発表会を行なう。

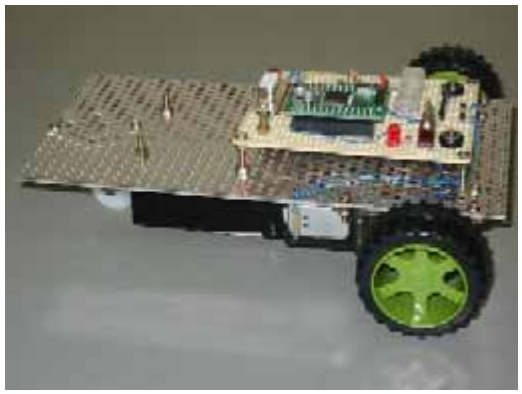


図２．２　ライントレーサ製作例

（４）製作にあたっては，以下の点に注意すること
１）部品をなくさない。
２）回路が出来上がったら，電源から見て回路がショートしていないか，信号線が電源やグランドとショートしていないか確認してから電源を入れる。
３）回路製作にあたっては，実装図を作ってから行なう。
４）はんだ付けは確実に行なう

２．ライントレーサ制御のアイディア

（１）図２．１のように CDS センサを車体下部の２つ付け，黒線の近くかどうかを検出し，これを電圧の変化に変換する。
（２）電圧の変化をマイクロコンピュータで読み取る。（AD コンバータ内蔵マイクロコンピュータを利用する）左右の光センサがどのくらい黒線に近いのか知ることができる。
（３）プログラムで２つのモータに与える指令値を算出する。進行方向左車輪のモータへの指令値が大きければ車体は右に曲がる。
（４）2 つのモータ指令値を２つの PWM 信号に変換する。（μコンピュータに内蔵されている PWM ユニット利用）
（５）２つのパワーMOSFET で２つのモータを駆動する。

図２．２にライントレーサ制作例を示す。

参考
（１）CDS センサ

CdS とは硫化カドミウムの事を指しており、この物質は可視光線でもっとも感度の良い光伝導体である。光制御の可変抵抗であり，光があたると抵抗が小さくなる。ここでは，照明用 LED を実際のセンサは CdS 粉末に $CdCl_2$ と $CuCl_2$ を少量加えて水で溶かしたものを、セラミックまたはガラスの基板に塗布し、空気中または不活性気体中て 500～700℃で焼結したものの上に、導電性ペーストでくしの歯状の電極を描いたものである。[参考　電子工学基礎論　和田正信 著]

（2）PWM 信号

PWM（Pulse Width Modulation：パルス幅変調）と呼ばれる信号で，モータなどの制御に使われる。モータに電池とスイッチを直列でつなぐことを考える。そして，スイッチを手で ON－OFF を繰り返す。スイッチ ON でモータは回り，OFF で止まる。この作業を手で高速に行なうと，１秒間に３回から５回はできるであろう。モータが高速に動いたり止まったりすることを目で見ることができる。マイコンにつながれたスイッチング素子を使うと，１秒間に１０００回くらいの ON-OFF 動作ができる。このくらいの ON-OFF を行うと，モータは滑らかに動いているように見える。しかも，ON になっている時間幅と OFF になっている時間幅を自由に変更できる。例えば 0.9msON にして 0.1msOFF にするとか，0.2msON にして 0.8msOFF にすることが可能である。

３．使うもの

ライントレーサ車体（モータ２，駆動輪２，キャスタ１，パンチトメタル）
光センサ CDS（明るさに応じて抵抗値が変化する素子），照明用 LED
オペレーショナルアンプリファイア（オペアンプ，電圧増幅，インピーダンス変換）
パワーMOSFET（電流増幅）
μコンピュータ H8-3664（AD コンバータ，PWM 信号発生）
電源をトレーサには積まずに，外部からケーブルで与える。制御回路用５V，モータ駆動用３V
ケーブル，抵抗，可変抵抗他

４．詳細な周辺回路設計（ブレッドボードを用いて予備実験を行なう）

（１）光センサ CDS（明るさに応じて抵抗値が変化する素子）と照明用 LED

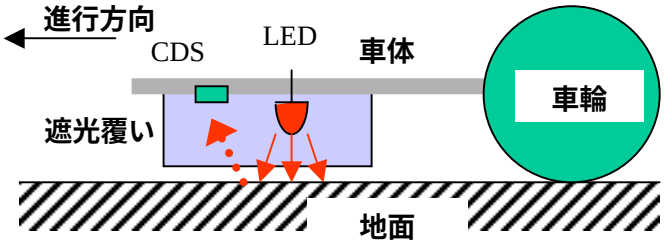


図４．１　光センサ CDS と照明用 LED

ＣＤＳは周辺の明るさの影響を受けるため，遮光覆い（カバー）をつけて，自然光の影響をなくして使う。一定の光源が必要なため，LED を照明用光源とする。図４．１のように照明用 LED を５V 電源で点灯させる。
LED には 15mA程度の電流を流すことにして図４．２のような回路を用いる。（左右 2 系統が必要となる）
LED は極性があるため，±を，間違えないこと。

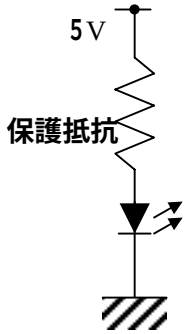


図４．２　照明用 LED

この時の保護抵抗は何オームにすればよいかブレッドボード上で検討しなさい。また＋側の見分け方を調べなさい

検討の結果（　　Ω）
LED＋側の目印（　　）

（２）ＬＥＤの地面反射光（競技面の白地，黒地）によるＣＤＳのみかけの抵抗値

次に，このＬＥＤの地面反射光（競技面の白地，黒地）によってＣＤＳの，みかけの抵抗値はどのような範囲で変化するかブレッドボード上でテストして調べなさい。（周りの光を覆って調べること，R0 の値には依存しない）

実験の結果　（　　kΩ～　　kΩ）

参考　1.0[kΩ]～3.7[kΩ]という実験値がある。

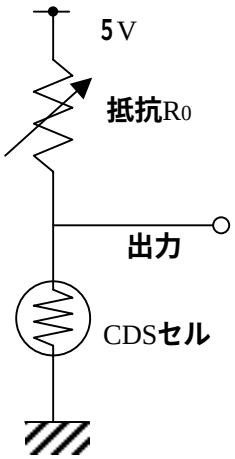


図４．３　光センサ

（３）オペアンプ（電圧増幅，インピーダンス変換）
図４．３のような回路で CDS を用いることにする。例えば抵抗値 R0 に１.5kΩを使ったとすると，CDS の見かけの抵抗が 1.0kΩの時出力は 1.0／(1.0+1.5)×5.0 = 2V となり，また CDS の見かけの抵抗が 3.0kΩの時出力は 3.0／(3.0+1.5)×5.0 = 3.3V となるはずである。
この電圧をマイクロコンピュータの AD コンバータに入力すると，その時の電圧を読み取ることができる。

AD コンバータとは，入力電圧をコンピュータの扱える数値に変換する素子であり，これから使用する H8/3664 には内蔵されている。０V から５V までの入力が，０から 65535 までの数値（設定によっては０から２５５の数値）に変換される。

ところが, CDS の見かけの抵抗値は使用状況によって変化するので，増幅率可変の電圧増幅を行なうようにしておくと，後の調整が楽になる。そこで，オペアンプを用いた電圧増幅を５倍程度行なうことにする。また中央点は 5V の半分の 2.5V とし，図４．４に示す回路とする。Rb を調節すると増幅率が変化する。

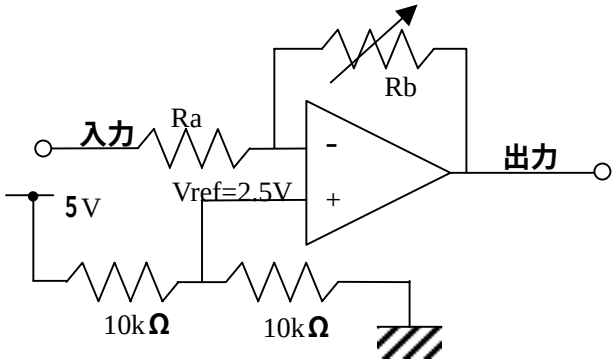


図４．４　中央点調節機能付で可変増幅率の反転増幅器

また, CDS 周辺の抵抗値は大きいため, 出力先の抵抗値に出力値が影響を受けやすいため，図４．５に示す見かけ上の入力抵抗が大きく，増幅率１倍のボルテイジフォロア回路を挿入する。

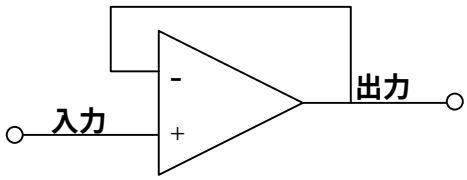


図４．５　見かけ上の入力抵抗が大きく，増幅率１倍のボルテイジフォロア回路

参考　オペアンプを用いた反転増幅器は，図４．６のように２つの抵抗を持っている。この回路の増幅率は－Rb／Ra であり，±が反転している。
図４．４の反転増幅器も増幅率は－Rb／Ra であるが，中央電圧が 0V でなく電圧 Vref を中心として電圧増幅が行なわれる。
入力電圧を Vin，出力電圧を Vout とすると図４．６，図４．４ではそれぞれ

$$V_{out} = -\frac{R_b}{R_a} V_{in} \qquad V_{out} - V_{ref} = -\frac{R_b}{R_a}\left(V_{in} - V_{ref}\right)$$

となる。

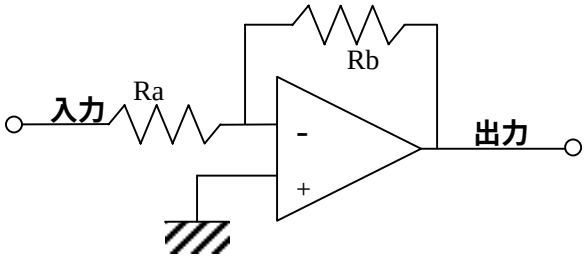


図４．６　基本的な反転増幅器

ここまでの光センサ部をまとめると図４．７に示すようになる。この回路で，CDS でラインを検出して，出力信号が 0V から 5V にあり，また，できるだけ大きく電圧が変化するように調節しておけば, AD コンバータへの光センサ信号入力が楽にできるようになる。

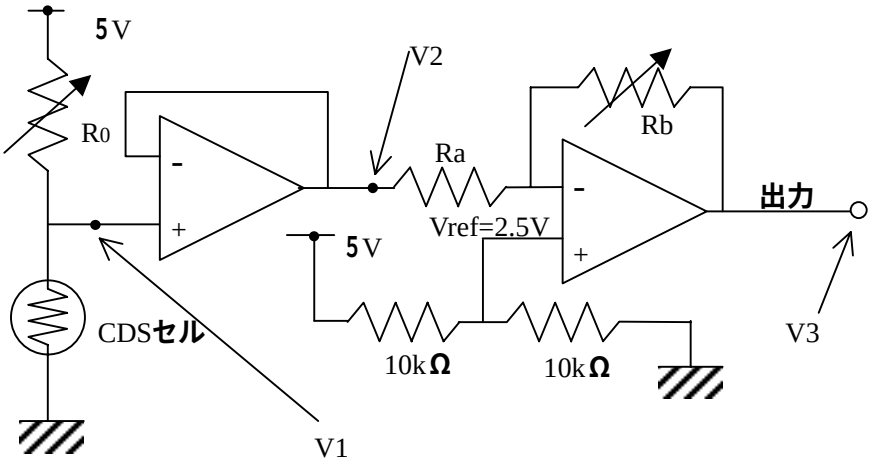


図４．７　光センサの完成

設定抵抗値の例
R0＝10ｋΩVR
Ra＝18ｋΩ
Rb=100ｋΩVR

例えばオペアンプ LM660 で図４．８のようになっていて，４つのアンプが１つの DIP パッケージに格納されている。ここで
V－には GND を
V+には５V を
接続すればよい。

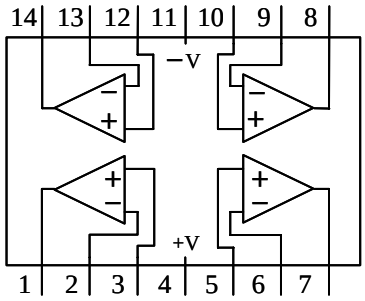


図４．８　オペアンプの例
LM660

光センサ回路の出力の測定
ブレッドボードで適当な抵抗値を用いて以下の測定を行いなさい。
設定のポイント
（１)図４．７において V1 が 2.5V を中心として等幅に振れるように R0 を設定する。
（２）図４．７において，V2=V1 となり，V3 が２．５V を中心に 0.5V～4.5V 程度に振れるように Ra，Rb を設定する。

設定抵抗値
R0＝（　　，　　ｋΩVR）
Ra＝（　　ｋΩ）
Rb=（　　，　　ｋΩVR）

黒地　（V1=　　V，V2=　　V，V3=　　V）
白地　（V1=　　V，V2=　　V，V3=　　V）

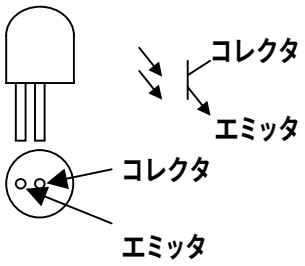


図４．１０　フォトトランジスタ TPS603A

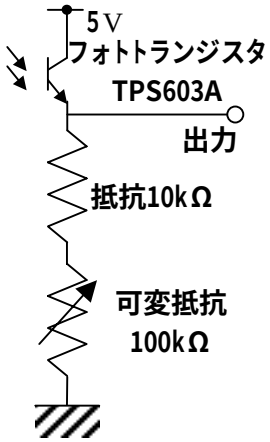
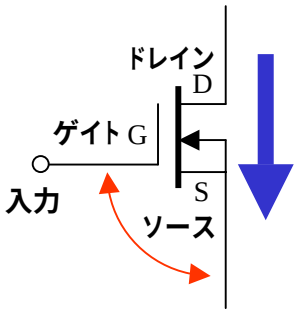


図４．９　フォトトランジスタの利用



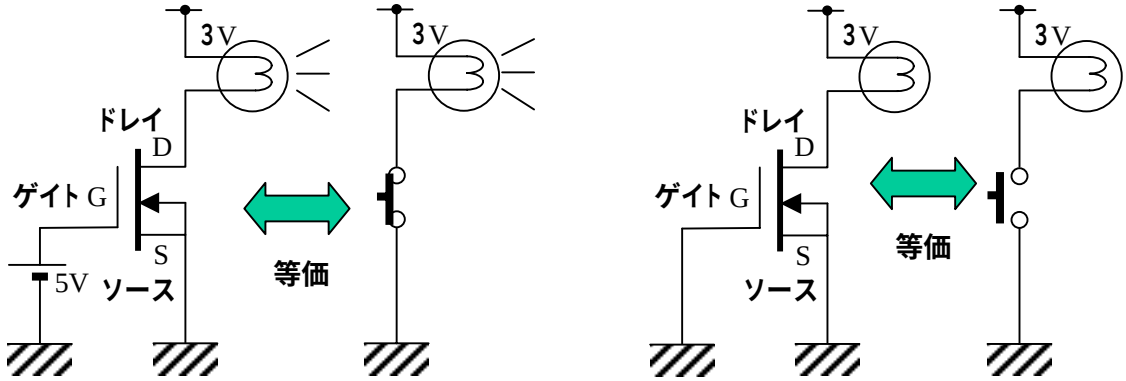
電圧で制御されるスイッチと考えればよい
ソースに対してゲート電圧が約３V 以上になると ON，0V では OFF になる

図４．１１　MOSFET の動作

研究　CDS 光センサは使いやすいが, 光の強さに対する応答が遅いとされている。フォトトランジスタは応答が速いといわれている。図４．９のように使うと，何らかの出力があるはずである。敏感なライントレースを実現するためにはフォトトランジスタの方が良いらしい。トライしたい学生は申し出ること。

（3）パワーMOSFET によるモータの PWM 制御

マイクロコンピュータ H8/3664 は，所定の端子から高速に H（約 5V）L（約 0V）を出力し，しかも H と L の時間間隔と周期をプログラムで設定できる機能を持っている。この機能でモータを駆動するには，パワーMOSFET によって電流増幅する必要がある。モータは小型なので電源電圧３V で駆動する。



ゲイトに５V を加え，MOSFET を ON　　ゲイトに 0V を加え，MOSFET を OFF

図４．１２　MOSFET はゲイト電圧制御のスイッチだ

パワーMOSFET は 3 本足の素子で，各足は図４．１１のように「ドレイン」「ゲート」「ソース」と呼ばれている。
この素子はゲートードレイン間にかかる電圧で制御されるスイッチと考えればよく，**ソースに対してゲート電圧が約３V 以上になると ON，０V では OFF になる。**

図４．１２には，ゲイトに電圧を加えたり加えなかったりして，豆電球のＯＮ－ＯＦＦをしているところを示す。この回路で豆電球を図４．１３のようにモータに置き換えるとモータのＯＮ－ＯＦＦ制御ができるようになる。

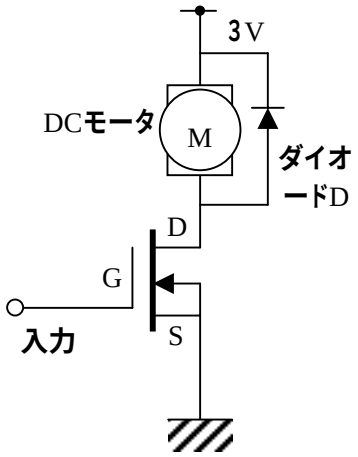
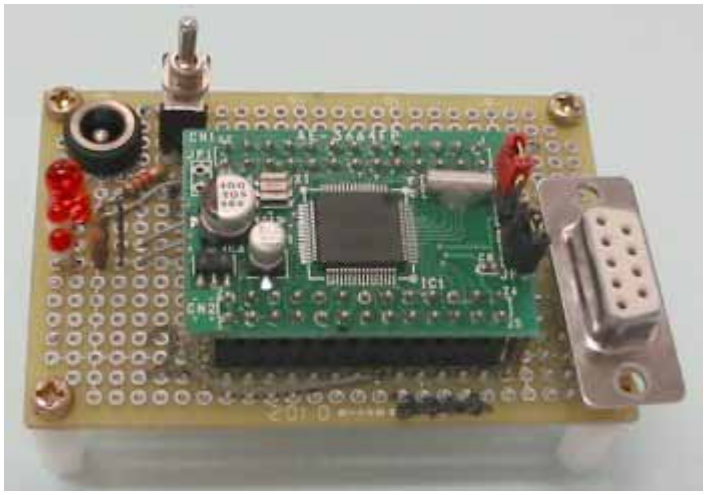


図４．１３　MOSFET を用いたモータ駆動部

５．マイクロコンピュータ H8/3664F

日立のマイクロコンピュータ H8 シリーズの中で安価に提供されているマイクロコンピュータ H8/3664F（図５．１）を制御に用いる。通常のマイクロコンピュータは ROM に機械語を書き込んで利用するが，このマイクロコンピュータでは機械語プログラムをフラッシュメモリに書き込むため，ROM が不要である。



H8/3664fCPU カードがマザーボードに載っている
これは，実装の一例であり，今回製作するものではない

図５．１　マイクロコンピュータ H8/3664F

H8/3664F の機能で何を使うか
（１）AD コンバータ２ｃｈ
（２）PWM 信号発生２ｃｈ
（３）タイマ割り込み
（４）LED（ハードウェア動作確認　シリアル通信動作確認）

どのように使うか

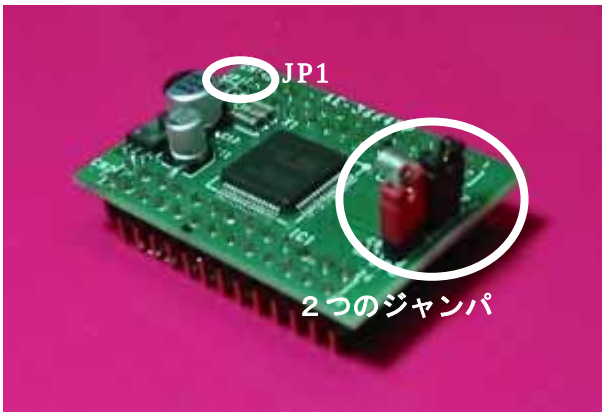
（1）windowsPC のエディタを用いて C 言語でプログラムを書く。
（2）クロス C コンパイラ（C プログラムを別の CPU 上で実行する機械語プログラムに変換するコンパイラをクロスコンパイラという）でコンパイルする。
（3）H8/3664CPU へシリアル通信で機械語プログラムを送り，フラッシュメモリ（ROM に相当）に書き込む。
（4）一度フラッシュメモリに書き込まれた機械語プログラムは電源を OFF にしても消えることは無い。

６．マイクロコンピュータ H8/3664F の入門

いきなり，H8/3664F でライントレーサを制御しようとしても出来ないので，まず H8/3664 を製作して，テストプログラムで動作をチェックし，AD コンバータテストプログラム，PWM テストプログラムを動作させてから，ライントレーサの制御を行なうこととする。

６．１　マイクロコンピュータ H8/3664F の製作

（1）H8/3664fCPU カード（図６．１，図６．２）を作る。CN1，CN２と JP2，JP3 をはんだ付けする。はんだ付けを終えたら図6．3の JP1 の部分のパターンをカットする。



（ピンの向きを間違えない，ジャンパをなくさない）
図６．１　H8/3664F 表側

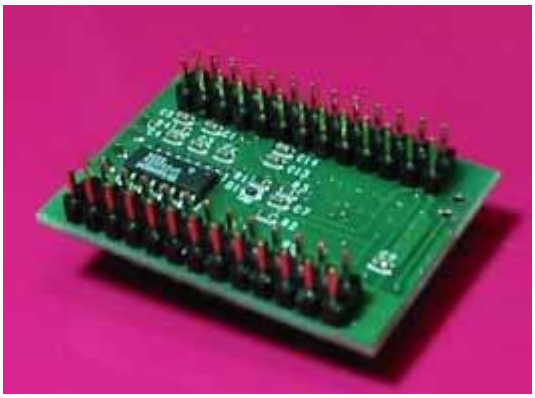


図６．２　H8/3664F 裏側

はんだ付けのポイント　多くの足を持つ部品をはんだ付けする時は，1 個所だけはんだ付けして，部品が基板から浮いていないかどうかチェックし，浮いていたらはんだ付け個所を再加熱して直す。そして残りの足のはんだ付けを行なう。

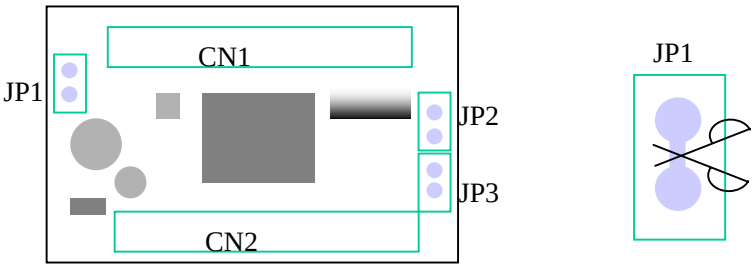


図６．３　H8/3664F のジャンパ JP1 カット

（2）マザーボードの製作
マザーボードを２つの段階に分けて作る。この段階でのマザーボードは図６．４のように製作するが，その役割は次のとおり。
1）電源部（電源用 DC ジャック５V と３V（モータ用））と LED（電源が入っていることを示す）
2）RS232C 通信部（DSUB9P）
3）動作チェック用 LED（約２mA で点灯させる。保護抵抗は 1.2～1.5kΩ）

製作にあたっての留意点
1）実装図を各自描いてから製作に入ること。部品の配置を考えながら実装図を方眼紙に描いて担

当教員にチェックを受けること。
2）後でこのマザーボードには回路を拡張するので，この段階の回路は端に寄せて製作すること。
3）4 隅には固定用の 3 ミリねじがつくので余裕を残すこと。
4）DSUB９のソケットには RS232C ケーブルが接続できるように余裕を残す。（図６．５参照）
5）DC ジャックは 2 ヶ所あるが，サイズが異なるので注意すること。３V 用にはフェルトペンで印がついているはず。（図６．８参照）
6）３V 用 DC ジャックはここでは使わないが，後で使うので，取り付けだけ済ませておく。
7）H8/3664fCPU カードのピンコネクタ CN2 の周りは，1 列以上あけておく。
8）５V 電源部と GND 部はこれから多くの配線があるので，島を作っておくと良い。
9）GND 部は，表面にピンを立てておくと，オシロスコープやテスタを用いた，トラブル時のチェックに役立つ。

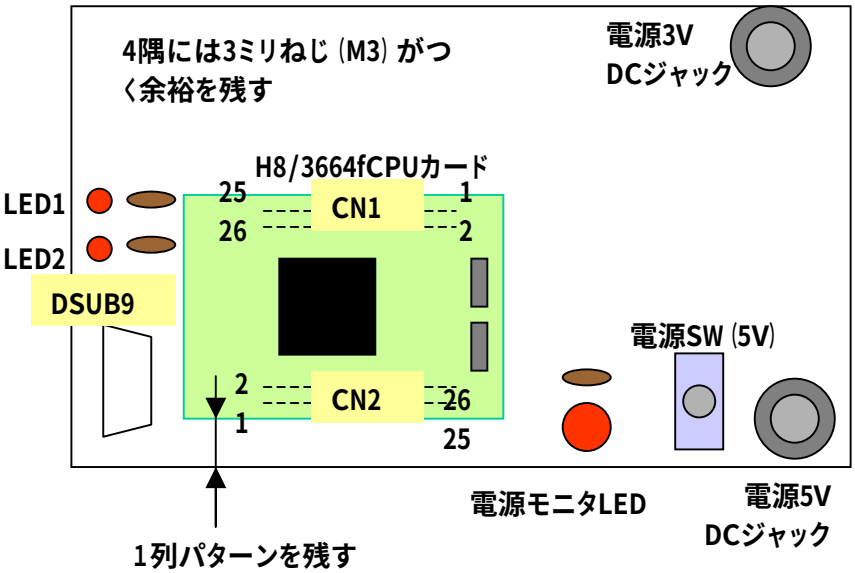


図６．４　H8/3664F のマザーボード製作

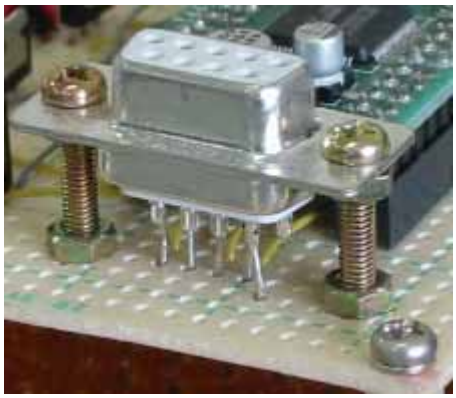


図６．５　DSUB9 の取り付けの工夫

回路図を図６．６，６．７に示す。

注意　マイコンキット付属の説明書では電源を CN1-24 につなぐように指示しているが，これは 7～9V の電源をつなぎ，内部の 3 端子レギュレータで 5V を作る場合であり，本製作では，CN1-23 につなぐ。

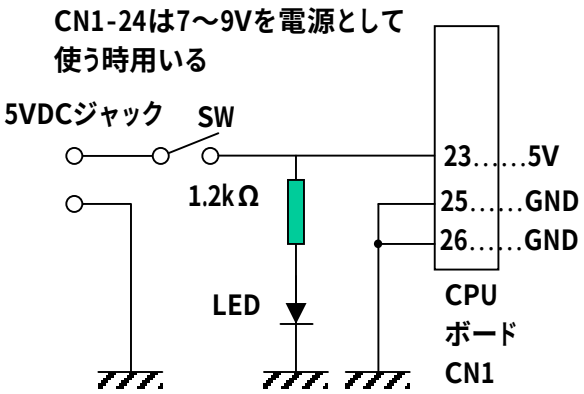


図６．６　電源部とパイロット LED

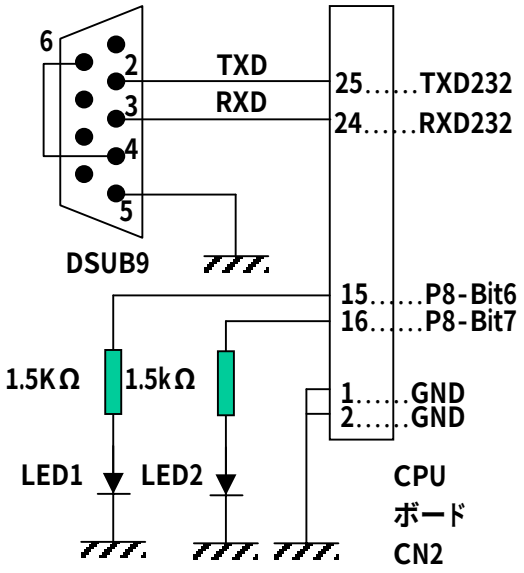
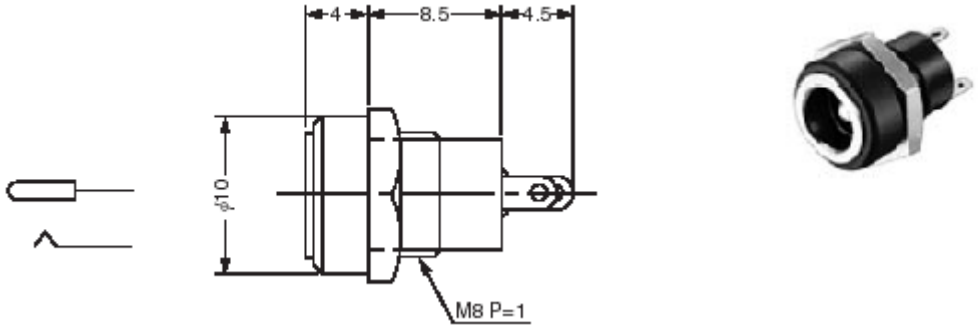


図６．７　RS232C 通信用 DSUB9 コネクタ部と動作チェック用 LED



５V 用：ピン径 2.1mm（中心が+5V，周りが GND）
３V 用：ピン径 2.5mm（中心が+3V，周りが GND）
図６．８　電源ジャック

マイコン H8/3664f カードおよびマザーボードが完成して，すぐに電源を投入してはならない。
電源側から見て 5V と GND がショートしていないかテスタを用いて確認すること。また RS232C 通信部 DSUB9 の各端子も 5V および GND に対してショートしていないことをテスタを用いて確認すること。

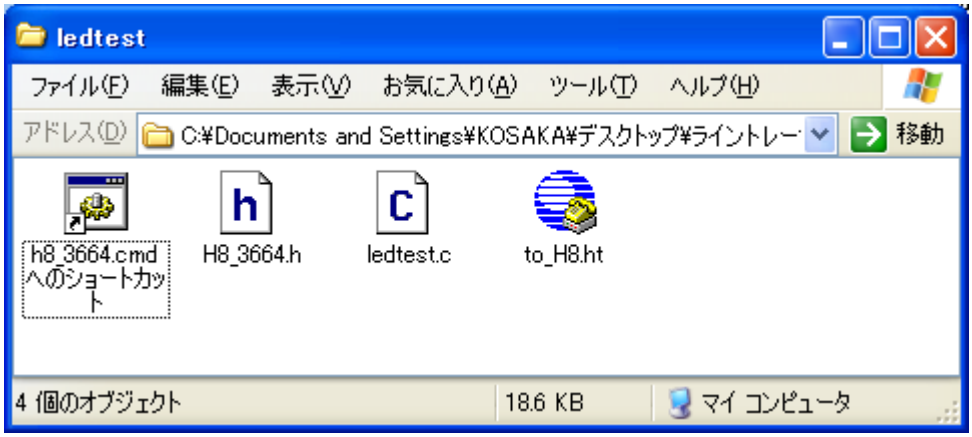
６．２　テストプログラム
出来上がった マイコンをテストしてみよう。
現時点で動作が確認できるのは，LED とシリアル通信である。

（１）LED をつける
図６．７に示す２つの LED がマザーボードについているが，これらは，H8/6334fCPU のポート８の第 6 ビットと第 7 ビットにつながっている。プログラム側から見るとマクロ定義された変数 P8.BIT.B6 と P8.BIT.B7 に見える。この２つの変数は 1 ビット変数であり，値は 0 か 1 しかない。P8 というのはポート 8 のことであり，ポート 8 の第 6 ビットと第 7 ビットを出力として用いるには，初期化が必要である。ポート 8 の初期化の詳細はヘッダファイル「H8_3664.h」により隠蔽されており，機能を実現するための関数を呼び出すことにより，プログラミングが可能である。
各関数は「H8_3664.h」内に説明と定義があるので，興味のある学生は H8/3664 のハードウェアマニュアルとあわせて読んでみるとよい。
なお，使用しているコンパイラで，int は short int を意味する。
「H8_3664.h」を用いると LED を 0.5 秒ごとに点滅させるプログラムは次のようになる。

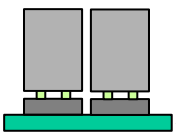
```
/*LED 点灯プログラム*/
#include "H8_3664.h"

void msecwait(int msec)
/*mesc 間なにもしない時間稼ぎ関数*/
{
    int i,j;
    for (i=0;i<msec;i++) {
        for (j=0;j<1588;j++);     /*1588 は実測によって求めた値*/
    }
}

main()
```

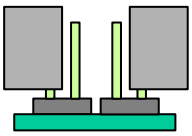
```
{
    initLed(); /*LED の初期化*/
    while (1) {
        turnOnLed(0); /*LED0 の点灯*/
        turnOffLed(1); /*LED1 の消灯*/
        msecwait(500);
        turnOnLed(1); /*LED1 の点灯*/
        turnOffLed(0); /*LED0 の消灯*/
        msecwait(500);
    }
}
```

このプログラムはコンパイルコマンドファイルと一緒にファイルで供給される。



コンパイルから実行までの手順は次の通りである。

	パソコン側	マイコン H8 側 注意：　５V の AC アダプタを電源として用いるが，電源電圧５V，内側が＋極性であることを確認して接続すること。わからなかったらテスタで極性，電圧を測定すること。
(1)	もしハイパーターミナルなど COM ポートを使用しているソフトがパソコン上で動作している時はそれらのソフトを中止します。	H8 は，電源スイッチ ON の瞬間に，状態選択スイッチの状態を検査しますので，必ず<1><2><3>の手順が必要です。 <1>AKI-H8/3664f 用の 5V の電源スイッチを OFF にします。 <2>AKI-H8/3664f カード上の状態選択ショートバー（２個）をショートし，ライト（Write）モードにします。ライトモードとはマイコン H8/3664f がパソコンからプログラムコードを受け取り，フラッシュメモリに書き込むモードのことです。  ←図６．２の２つのジャンパ <3>AKI-H8/3664f 用の 5V の電源スイッチを ON にします。

(2)	ソースファイル（ledtest.c）のアイコンを「h8_3664.cmd へのショートカット」のアイコン上にドラッグアンドドロップします。 「コンパイル」―「リンク」―「コンバート」が行なわれた後に，H8 のフラッシュメモリへ，実行プログラムの書き込みが行なわれます。	
(3)		<1>転送が終了したら、AKI-H8/3664f 用の 5V の電源スイッチを OFF にします。 <2>AKI-H8/3664f カード上の状態選択ショートバー（２個）をオープンし，ラン（Run）モードにします。  ←図６．２の２つのジャンパ
(4)	必要ならパソコン側で「to_H8.ht」をダブルクリックしてターミナルアプリケーション「ハイパーターミナル」を立ち上げます。	
(5)		AKI-H8/3664f 用の 5V の電源スイッチを ON にします。

練習１　「LED１を 2 回，LED２を 2 回点滅させた後，両方の LED を 2 回点滅させる」動作を無限に繰り返すプログラムを作成しなさい。ただし点滅の時間は 0.5 秒とする。（0.5 秒ごとに状態が変化する。）

練習２　LED１を点灯したまま，LED２を 0.01 秒間隔で点滅させなさい。（人間の目では LED２の点滅を確認できないが，暗く点灯しているように見える。）

（２）RS232C 通信プログラム

PC と H8/3664 が RS232C 通信（シリアル通信）してデータのやり取りを行なう。H8/3664 が動作する時には PC の方ではハイパーターミナルを動作させておく。通信の規則は次の通りである。

「38400baud,8bit,noparity,1stopbit」

「H8_3664.h」に用意された関数郡を用いると通信プログラムは次のようになる。ここで H8/3664 から PC へのデータ送信は printf 文もどきの関数「SCI_printf」と「putCharSCI」, PC から H8/3664 へのデータ転送は「getCharSCI」「chkgetCharSCI」「getIntSCI」を用いる。

```
/*シリアル通信でホストと交信する*/
#include "H8_3664.h"

main()
{
    int ch;
    initSCI(); /*シリアル通信インタフェイスの初期化
                38400baud, Ascnc, 8bit , NoParity, stop1*/
    SCI_printf("****** SCI test ******¥n");
    SCI_printf("Hit any character key¥n");
```

```
    while (1) {
        ch=getCharSCI(); /*1 文字取得*/
        putCharSCI(ch+1); /*1 文字送信*/
        SCI_printf("  test %d %d %x¥n",1234,9876,ch);
        SCI_printf("  %c %s %lx¥n",'#',"hello",0x80804040L);
        SCI_printf("  %s %c %lx¥n","hello",'#',0x80804040L);
    }
}
```

練習１　「LED を 2 回点滅させ，シリアル出力（RS232C 出力）へ「Hello world!」を出力する」を無限に繰り返すプログラムを作りなさい。

練習 2　「シリアル入力から 1 から 10 までの整数値 n を受け取り，LED をｎ回点滅させる」を無限に繰り返すプログラムを作りなさい。(「getIntSCI」を用いる)

練習３　シリアル入力から文字 A を受け取ったら LED1 を点滅させ,文字 B を受け取ったら LED2 を点滅させ，文字 C を受け取ったら両方とも消灯するプログラムを作りなさい。なお文字の受け取りは何回でも可能で，支持の通りに LED を駆動するものとする。(「chkgetCharSCI」を用いる)

（３）　タイマ割り込み

通常の関数は関数が呼び出された時に起動するが，割り込み関数は要求信号により起動し，割り込み関数が終了すると先ほどまで行なっていた処理の続きに戻るという動作となる。
タイマ割り込みとは，CPU のタイマユニットにより，一定時間間隔で割り込み要求信号を発生し，割り込み関数を実行する機能である。タイマ割り込みを用いると，例えば約 33msec ごとに割り込み関数を実行することができる。
割り込み関数内で，シリアル通信機能などの処理時間の長い処理を行なってはいけない。
次のプログラムでは，タイマ割り込みを用いて，LED の点滅を行なう。

```
/*タイマー割り込みを起動し，LED の点滅制御を行なう*/
#include "H8_3664.h"

volatile int cnt=0;

main()
{
    initLed();/*LED の初期化*/
    initTimerAInt(2); /*タイマ A による約 30Hz 割り込み*/
    E_INT(); /*CPU 割り込み動作許可*/
    startTimerAInt(); /*タイマ A 起動*/
    while (1) {
        if (cnt<15) {
            turnOnLed(0);
            turnOffLed(1);
        } else {
            turnOnLed(1);
            turnOffLed(0);
        }
    }
}
```

```
/*割り込み関数　関数名は変更してはならない*/
void interrupt_cfunc(void)
{
    cnt++;
    if (cnt==30) cnt=0;
}
```

練習１　リスト３のプログラムで，約 2kHz のタイマ割り込みに変更しなさい。「H8_3664.h」に書いてある説明を参考にして，初期化の引数を変えればよい。

練習２　リスト３のプログラムでは LED0 の ON の時間は 15/30 に固定である。シリアル入力で 0～30 の整数を変数 duration に受け取り，LED0 の ON の時間を 0/30～30/30 に変更できるようにしなさい。なおこの変更はプログラム起動時に 1 回だけでなく，プログラム動作中に何回でも変更できるようにしなさい。タイマ割り込みは約 2kHz にしなさい。

７　光センサおよびモータドライブのマイコン周辺回路

ここで光センサおよびモータドライブのマイコン周辺回路を完成させる。
回路製作にあたっては，実装図を描いてからはんだ付けを行なうこと。

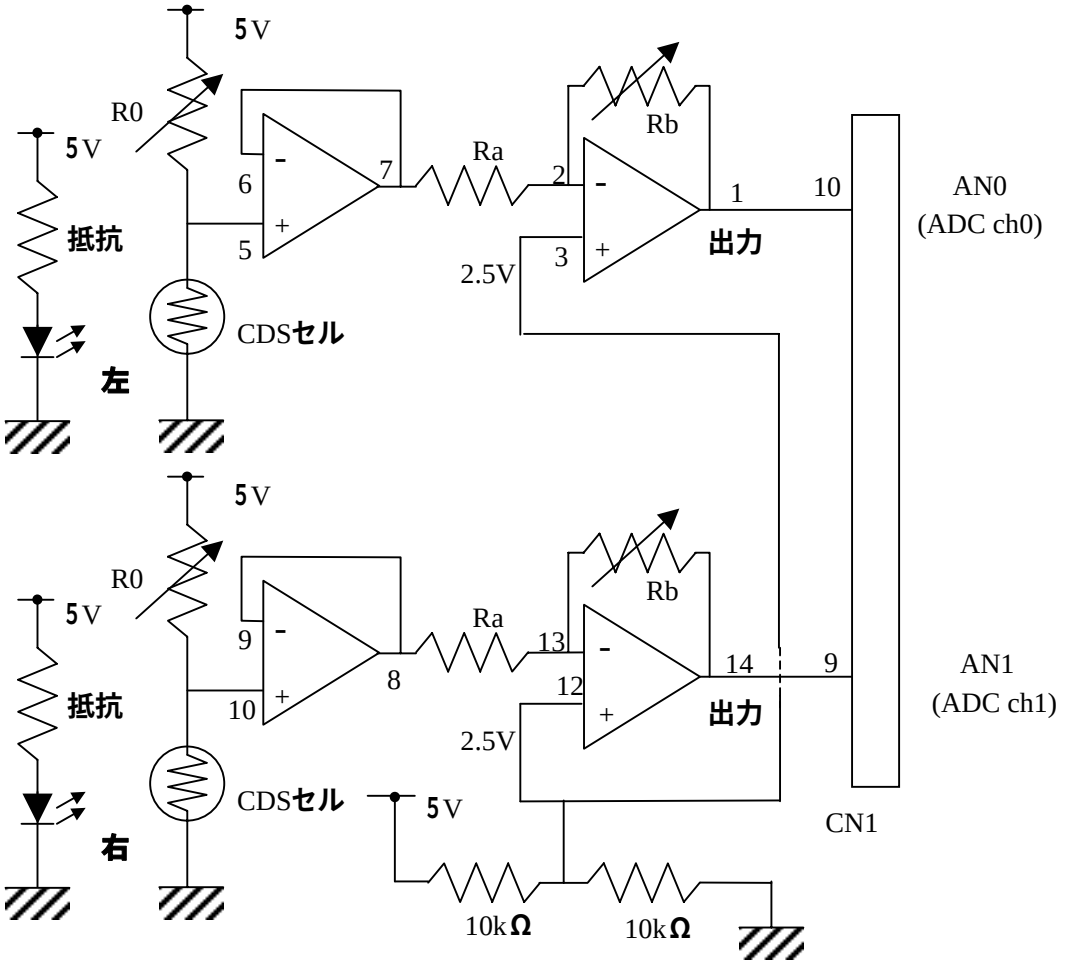


図７．１　光センサ回路

（1）LED および光センサ部
照明用 LED および光センサは左右 2ch 分を LED 光軸が CDS 直下になるように小基板に実装する。光センサ部にフォトトランジスタを使用する研究が終っている場合は，フォトトランジスタを用いた回路を使用してもよい。センサ回路の出力を CPU の AD コンバータ入力に接続する。図７．１，図７．２，図７．３，図７．４を参照すること。
なお CN1 の９，１０の位置に，テスタの針で触れやすいように，表側にピンを引き出しておくと良い。

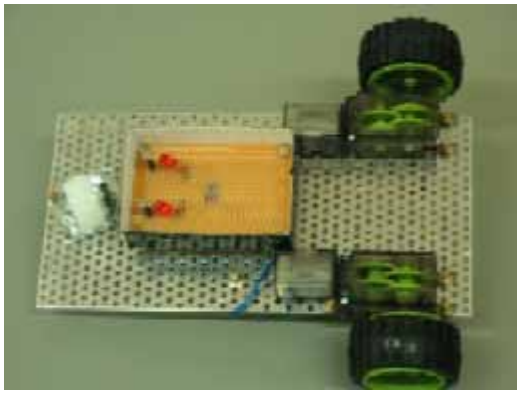


図７．２　車体裏側（左側が前）

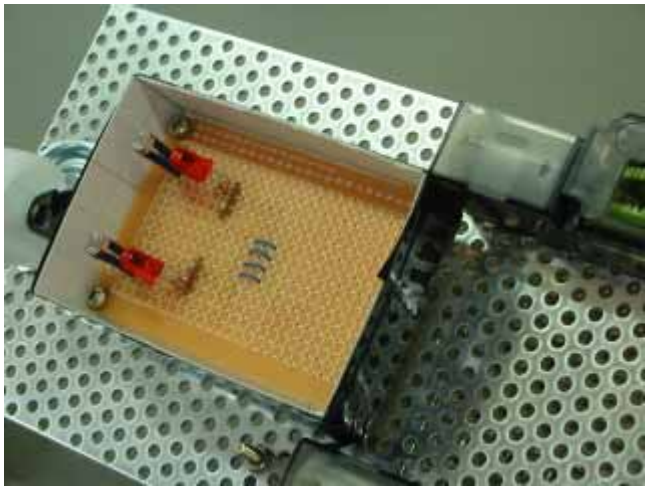


図７．３　光センサー部（遮光覆いがついている）

（2）モータドライバ回路
モータドライバへの入力を CPU の PWM 出力へ接続する。図７．５を参照のこと。
パワーMOSFET のゲイト入力に対しては電流が流れ込まないため, CPU の出力ピンを直接つなぐことができる。

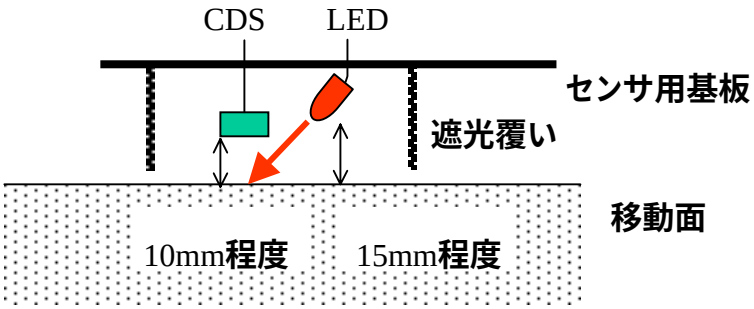


図７．４　照明 LED とセンサの移動面よりの高さ

なお CN2 の 11, 12 の位置に，オシロスコープの針で触れやすいように，表側にピンを引き出しておくと良い。

使用したパワーMOS FET の型番とソース，ゲイト，ドレインがどの足に対応するのか調べなさい。

FET 型番
（　　）
FET を前から見て
左側から（　　），
（　　），
（　　）　（ソ

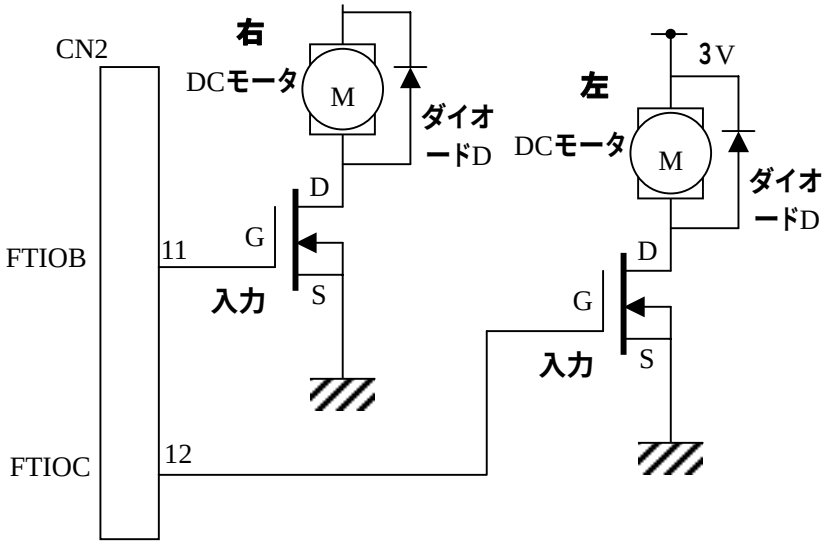


図７．５　モータドライバ回路

ース，ゲイト，ドレインの文字を埋めなさい）

使用したダイオードの型番を調べなさい。
（　　）

（3）シャーシとモータの取り付け
シャーシにモータ，制御回路を組み込み，ライントレーサを完成させなさい。
注意　出来た回路に電源を入れる前に電源の DC ジャックから見て，５V ラインと GND がショートしていないかテスタで検査しなさい。３V ラインと GND も同様に検査し，ショートの有無を確かめなさい。３V ラインと５V ライン間も同様にショートの有無を確かめなさい。

８．　光センサおよびモータドライブのマイコン周辺回路のテスト

（１）　光センサ回路テスト
（１．１）光センサ回路の調節
注意　ここでの調節で，テスタの針で，信号同士をショートさせないように気をつけなさい。
照明用 LED と遮光覆いをつけて，白紙上の黒いラインを検出させるようにした時，光センサ回路の出力（AD コンバータへの入力）をテスタで電圧を調べて，可変抵抗を調整しなさい。
調整の方法（CDS の場合）

（１）　調整したい方のセンサを黒ラインと白地に交互に置き，光センサ出力（オペアンプ初段への入力，図７．１のオペアンプ 5 番ピン 10 番ピン）が 2.5V を中心に振れるように Rc を調整する。（2.0V から 3.0V とか 1.8V から 3.2V のようにする）
（２）　調整したい方のセンサを黒ラインと白地に交互に置き，光センサ増幅回路の出力（AD コンバータへの入力位置，図７．１のオペアンプ１番ピン１４番ピン）が 2.5V を中心に振れ，最大で４．５V，最小で０．５V 程度になるように Rb を調整する。

調節結果の報告（オペアンプ初段への入力，光センサ増幅回路の出力を測定）

センサ位置	実測値（オペアンプ初段入り口）	実測値（光センサ増幅回路の出力）	CDS の場合の理想値
右センサ黒ラインの上	（　　）V	（　　）V	０～１V
右センサ白地の上	（　　）V	（　　）V	４～５V
左センサ黒ラインの上	（　　）V	（　　）V	０～１V
左センサ白地の上	（　　）V	（　　）V	４～５V

この時の結果を次に進む前に担当教員に報告しなさい。
これが出来たら動作は正常である。

（１．２）AD コンバータテストプログラム
H8/3664 の ADC 入力ピンに入力された電圧を，数値に変換する。AD コンバータへの入力が 0V の時，AD 変換値は約 0，AD コンバータへの入力が約５V の時（絶対に５V を超えてはならない），AD 変換値は約 65535 となる。

このためのサンプルプログラムは次のようになる。

```
/*シングルモードで AD コンバータから信号を得る*/
/*シングルモードでは初期化は不要*/
#include "H8_3664.h"

void msecwait(int msec)
/*mesc 間なにもしない時間稼ぎ関数*/
```

```
{
    int i,j;
    for (i=0;i<msec;i++) {
        for (j=0;j<1588;j++);     /*1588 は実測によって求めた値*/
    }
}

main()
{
    unsigned int v0,v1;
    int i;
    initSCI();
    putStringSCI("ADC single mode¥n");
    while (1) {
        v0=getADCValue(0); /*ADCch0 から入力*/
        v1=getADCValue(1); /*ADCch1 から入力*/
        SCI_printf("ch0,1 =");
        SCI_printf(" %4u",v0);
        SCI_printf(" %4u",v1);
        SCI_printf("¥n");
        msecwait(1000); /*1 秒間おやすみ*/
    }
}
```

次の実験を行ないどのような値に変換されたか調べなさい。

右側センサ位置	ADC チャンネル（　　）	ADC への入力電圧
黒ライン	変換値（　　）	（　　）V
白地	変換値（　　）	（　　）V
中間	変換値（　　）	（　　）V
左側センサ位置	ADC チャンネル（　　）	
黒ライン	変換値（　　）	（　　）V
白地	変換値（　　）	（　　）V
中間	変換値（　　）	（　　）V

これが出来たら動作は正常である。

練習　２つの ADC の値を読み取り, ADC0 の値の方が ADC1 の値より大きかったら LED0 を点灯させ,ADC1 の値の方が ADC0 の値より大きかったら LED1 を点灯するプログラムを作りなさい。この動作は最速ループで行なうことにします。

（２）　PWM テストプログラム

モータの電源３V はまだ供給しないこと。
次のプログラムを動作させなさい。

```
/*「TimerW」を用いた２系統 PWM 出力*/
#include "H8_3664.h"

void msecwait(int msec)
```

```
/*msec 間なにもしない時間稼ぎ関数*/
{
    int i,j;
    for (i=0;i<msec;i++) {
        for (j=0;j<1588;j++);     /*1588 は実測によって求めた値*/
    }
}

main()
{
    initSCI();
    initTimerWPWM(1000); /*TimerW の２系統 PWM 出力の初期化*/
    /*約 2kHz の周期の PWM 信号が発生される 16MHz／8÷1000*/
    while (1) {
        setPWMValueB(1000); /*B 系統に 1000/1000*/
        setPWMValueC(1000); /*C 系統に 1000/1000*/
        putStringSCI("PWM 1000/1000¥n");
        msecwait(3000);
        setPWMValueB(1000); /*B 系統に 1000/1000*/
        setPWMValueC(0);    /*C 系統に 0/1000*/
        putStringSCI("PWM 1000,0/1000¥n");
        msecwait(3000);
        setPWMValueB(0);    /*B 系統に 0/1000*/
        setPWMValueC(1000); /*C 系統に 1000/1000*/
        putStringSCI("PWM 0,1000/1000¥n");
        msecwait(3000);
        setPWMValueB(700); /*B 系統に 700/1000*/
        setPWMValueC(700); /*C 系統に 700/1000*/
        putStringSCI("PWM 700/1000¥n");
        msecwait(3000);
        setPWMValueB(400); /*B 系統に 400/1000*/
        setPWMValueC(400); /*C 系統に 400/1000*/
        putStringSCI("PWM 400/1000¥n");
        msecwait(3000);
    }
}
```

動作したら，ピン間のショートに気をつけながら，CN2 の 11，12 の位置に，オシロスコープの針で触れて，約 2KHz で PWM 信号が出ている様子を観察しなさい。
約 3 秒間隔で PWM のデューティ（1 周期中の H 信号の長さ）が 100%，70%，40%と変化しているのがわかると思う。これが出来たら動作は正常である。

次にモータ用の電源３V をつなぐと，モータが約 3 秒間隔で，高速回転，中速回転，低速回転するのがわかるであろう。

もしモータの回転方向が逆だったら，モータとドライバをつなぐ配線を逆にすればよい。

練習　シリアル通信で２つの整数値を変数 p1,p2 に受け取り，右モータに p1 の値を PWM 値とし

て与え，左モータに p2 の値を PWM 値として与えるプログラムを作りなさい。永久ループのプログラムとし，絶えず新しい整数値を入力できるようにしなさい。

９．　ライントレーサプログラミング

ハードウェアが完成したのであとはソフトウェアで。ライントレースを実現しよう。
２つの光センサの ADC の読み取り値により，センサと黒ラインとの位置関係がわかる。また PWM への出力により，モータの可変速も可能である。あとは自分で考えて制御プログラムを書きなさい。

シンプルなアウトラインは次のようになるであろう。

```
#include "H8_3664.h"

main()
{
    int adc0=0; /*ADC0  左側センサの値*/
    int adc1=0; /*ADC1  右側センサの値*/
    int pwmB=0; /*PWMB  右側モータへの指令値*/
    int pwmC=0; /*PWMC  左側モータへの指令値*/

    initTimerWPWM(1000); /*TimerW の２系統 PWM 出力の初期化*/
    /*約 16kHz の周期の PWM 信号が発生される 16MHz÷1000*/

    while (1) {
        adc0=getADCValue(0);
        adc1=getADCValue(1);
        /*---------------------------------------*/
        /*ここで２つのモータへの出力値を計算する*/
        /* adc0,adc1 → pwmB,pwmC */
        /*---------------------------------------*/
        setPWMValueB(pwmB);
        setPWMValueC(pwmC);
    }
}
```

光センサ出力値の時間変化率をも考慮すると例えば次のようなアウトラインになるであろう。(改善の余地がある)

```
#include "H8_3664.h"

volatile int adc0=0; /*ADC0  左側センサの値*/
volatile int adc1=0; /*ADC1  右側センサの値*/
volatile int d_adc0=0; /*D_ADC0  右側センサの値の時間変化率*/
volatile int d_adc1=0; /*D_ADC1  左側センサの値の時間変化率*/

main()
{
    int pwmB=0; /*PWMB  右側モータへの指令値*/
    int pwmC=0; /*PWMC  左側モータへの指令値*/
    initLed();/*LED の初期化*/
```

```
    initTimerWPWM(1000); /*TimerW の２系統 PWM 出力の初期化*/
    /*約 16kHz の周期の PWM 信号が発生される 16MHz÷1000*/
    initTimerAInt(3); /*タイマ A による約 120Hz 割り込み*/
    E_INT(); /*CPU 割り込み動作許可*/
    startTimerAInt(); /*タイマ A 起動*/
    while (1) {
        /*-------------------------------------*/
        /*ここで２つのモータへの出力値を計算する*/
        /* adc0,adc1,d_adc0,d_adc1 → pwmB,pwmC */
        /*-------------------------------------*/
        setPWMValueB(pwmB);
        setPWMValueC(pwmC);
    }
}

/*割り込み関数　関数名は変更してはならない*/
void interrupt_cfunc(void)
{
    static int old_adc0=0; /*前回の ADC0 の値*/
    static int old_adc1=0; /*前回の ADC1 の値*/
    adc0=getADCValue(0);
    adc1=getADCValue(1);
    d_adc0=adc0-old_adc0;
    d_adc1=adc1-old_adc1;
    old_adc0=adc0;
    old_adc1=adc1;
}
```

あまりこれらのプログラムにとらわれずにいろいろな考え方を提案しなさい。

１０．レポートとタイムアタック競技，プレゼンテーション

（１）各自ライントレーサ製作に関するレポートを提出すること。ただし，レポート採点者はこの指導書の存在は知らない立場で採点する。自分がどの部分に貢献したのかを含めること。レポートは MicrosoftWord を用いて作成し，指示する方法で電子提出すること
（２）タイムアタック競技を行なう。
（３） レポートをもとにプレゼンテーションを行なう。

１１．資料

小坂の Web サイト http://tnct20.tokyo-ct.ac.jp/~kosaka/index.html（学内のみ）または
http://www2.tokyo-ct.ac.jp/users/j/staff/kosaka/index.html （自宅からも OK）
から「for students」に入り「工学実験」のところの
http://tnct20.tokyo-ct.ac.jp/~kosaka/for_students/lineTrace/lineTrace.html（学内のみ）または
http://www2.tokyo-ct.ac.jp/users/j/staff/kosaka/for_students/lineTrace/lineTrace.html
を参照しなさい。
なおこの Web ページ「lineTrace.html」の「１１．資料」にはこのプリントには入っていない資料（素子のマニュアルの pdf ファイルなど）がある。