制御システム工学演習
-リファレンスボード編-

1/4

http://www2.kaiyodai.ac.jp/~jtahar0/posts/post.html

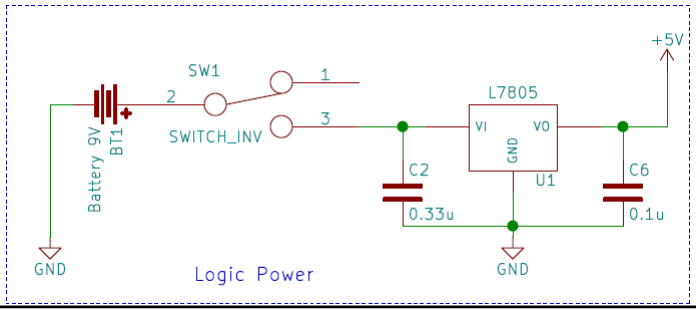
http://bit.ly/2x9HuL8

はじめに

- 前半4日でリファレンスボードを使い以下をマスターする
  - 回路図の読み方，書き方
  - ハードウエアの確認
  - ソフトウエアの書き込み方
  - デバッグ
  - ライントレースの基本部分
- 授業の資料は以下にまとまっています
  http://www2.kaiyodai.ac.jp/~jtahar0/posts/post.html
  - 必ず，確認してください．
- 持参するもの
  - テキスト，USBメモリ，ノート
- 持参した方が良い物
  - グラフ用紙，定規=>回路図作成
  - カラーペン=>回路図の確認用

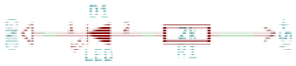
電源回路

- 今回は9Vから5Vの電源を作成する.
- 電気回路には必ずGNDがある
  - GNDが電源の基準となる
- 3端子レギュレータにより9Vを5Vにしている
  - 9V=>5Vに落とす
  - エネルギーは熱として放出される.



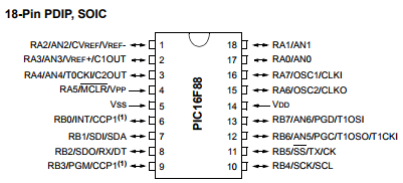
LED

- 発光ダイオードと呼ばれる
  - 足が長い方がアノード(+側),短い方がカソード(-側)
- 電流制限抵抗を必ず付ける事
  - R=(Vcc – Vf)/If



マイクロプロセッサ I

- PIC16F88
  - フラッシュ：4キロワード／7168バイト
  - データSRAM：368バイト
  - EEPROM：256バイト
  - クロック：最大20MHz
  - 電源電圧；4．0V（MAX20MHz）～5．5V（MAX20MHz）
  - I／O：16
  - 10ビットADコンバータ内蔵
  - PWM（CCP）、USART、コンパレータ、SSP機能



マイクロプロセッサ II

- 産業用の8bitマイコン
- 電源Vddに5V,VssにGNDをつなげば動作する
- PC用のCPUと異なる
  - 計算能力よりも．．．．．
  - 外部入出力
  - 省電力
  - 回路規模の小型化
- 各社特徴的なCPUを提供している
  - TI MSP430 超低消費電力
  - Microchip AVR ATMEGA Arduino
  - Arm 32bit,携帯電話

マイコンの回路構成

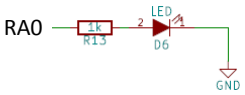
- 自分でマイコン回路をテストするのに最低限必要
- 1~5最低限必要
- 6 なるべく付けよう
- 7 rs232cが有るとPCと情報がやり取り出来る

1. マイコン
2. 電源
3. リセット回路
4. ISPプログラムライター接続端子
5. 確認用LED
6. 動作確認用SW
7. RS232C

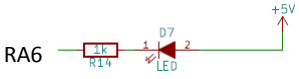
最後のページの回路図を
参考にしてください.

マイコンのDO制御回路
DO: Digital Output

- Hiレベルドライブ
  - RA0がON(5V)の時LEDが点灯
  - 電流はRA0=>GNDに流れる
  - 正論理



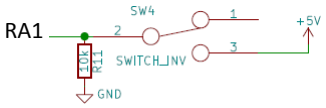
- Lowレベルドライブ
  - RA6がOFF(0V)の時にLEDが点灯
  - 電流はRA6<=5Vに流れる
  - 負論理



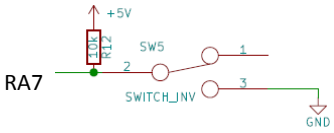
外部機器とつなぐ時は，正論理か負論理かを必ず確かめておくこと
産業用機器は通常負論理接続である

マイコンのDI回路
DI: Digital Input

- プルダウン
  - SWがONの時，RA1はON
  - 電流の流れる向きを記入せよ
  - 正論理



- プルアップ
  - SWがOFFの時RA7はON
  - 電流の流れるむきは？
  - 負論理



- プルアップ･プルダウン抵抗は必ず付ける事

プログラムの作成方法I

- アセンブラもしくはC言語を用いてプログラムを作る
- プログラムを作る前に，ハードウエアの接続先を確認
  - エクセルでピン配置表を作成すること
  - ライントレースロボを作る時は必ず必要になる.
    - 巻末に表があるので活用してください
- リファレンスボードでは以下の様になっている

設定	#define	接続先		PIC16F88			接続先	#define	設定
出力	M_IN1	H-Bridge: IN1	1	RA2	RA1	18	SWITCH1	SWITCH1	入力
出力	M_IN2	H-Bridge: IN2	2	RA3	RA0	17	True ON LED	LED1	出力
入力	VOLUME	VR1	3	RA4	RA7	16	SWITCH2	SWITCH2	入力
		Reset	4	RA5 (MCLR)	RA6	15	False ON LED	LED2	出力
		GND	5	Vss	Vdd	14	5V		
		Not Connect	6	RB0	RB7 (PGD)	13	PGD		
		Not Connect	7	RB1	RB6 (PGC)	12	PGC		
入力		RX	8	RB2 (RX)	RB5 (TX)	11	TX		出力
出力	TR	Transistor	9	RB3	RB4	10	Photosensor	SENSOR	入力

プログラムの作成方法II

- PICのプログラムで難しい所
- TRISA,TRISBの設定のしかた
- **TRISXはPICの入出力機能を決定するレジスターである**
- 入力としたいPINのBITを1にする．出力は0にする
  - 初めは，ここをマスターする

設定	#define	接続先		PIC16F88			接続先	#define	設定
出力	M_IN1	H-Bridge: IN1	1	RA2	RA1	18	SWITCH1	SWITCH1	入力
出力	M_IN2	H-Bridge: IN2	2	RA3	RA0	17	True ON LED	LED1	出力
入力	VOLUME	VR1	3	RA4	RA7	16	SWITCH2	SWITCH2	入力
		Reset	4	RA5 (MCLR)	RA6	15	False ON LED	LED2	出力
		GND	5	Vss	Vdd	14	5V		
		Not Connect	6	RB0	RB7 (PGD)	13	PGD		
		Not Connect	7	RB1	RB6 (PGC)	12	PGC		
入力		RX	8	RB2 (RX)	RB5 (TX)	11	TX		出力
出力	TR	Transistor	9	RB3	RB4	10	Photosensor	SENSOR	入力

TRISAの設定

ピン名	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
接続先	SW	LED	Reset	VR	H_Brd.	H_Brd.	SW	LED
入出力	入力	出力	-	入力	出力	出力	入力	出力
TRISA	1	0	0	1	0	0	1	0

プログラムの作成方III

- サンプルプログラムをダウンロードする．
- ブックマークから
  http://www2.kaiyodai.ac.jp/~jtahar0/posts/post.html
  へ移動
- サンプルプログラム　16F88_Ref3_Blink.X　をダウンロードし展開する．
- 展開したフォルダーを自分のUSBメモリーにコピーする
- USB上のサンプルを使うので2-13を参照
  - http://www2.kaiyodai.ac.jp/~jtahar0/posts/activity19.html
- コンパイル
  - エラーがないか？
- PICkit3でプログラムを書き込む
  - リファレンスボードで確認

制御システム工学演習
-リファレンスボード編-

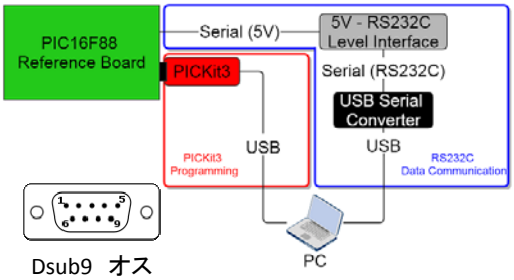
2/4

http://www2.kaiyodai.ac.jp/~jtahar0/posts/post.html

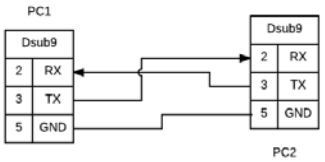
http://bit.ly/2x9HuL8

RS232Cによるデバッグ　I

- RS232Cとは
  - PCと産業用機器をつなぐためのインターフェイス
    - シーケンサー，モデム，ロガーの接続
  - TX(データ送信),RX(データ受信),GNDの3本の線があれば良い
    - クロス接続をする必要がある，TX(1)=>RX(2), RX(1)<=TX(2)
  - RS232C自体は±12Vで動作
    - マイクロプロセッサは0,5Vで動作(UART
    - 直接つなぐと故障するので注意



Dsub9　オス



ピンNo.	信号名	別名	入出力	内容
1	DCD	CD	IN	キャリア検出
2	RxD	RD	IN	受信データ
3	TxD	SD	OUT	送信データ
4	DTR	ER	OUT	データ端末レディ
5	GND	SG	–	グランド
6	DSR	DR	IN	データセットレディ
7	RTS	RS	OUT	送信リクエスト
8	CTS	CS	IN	送信可
9	RI	CI	IN	被呼表示

RS232Cによるデバッグ II

- ソフトウエアの変数を印字できる
  - SWの状態を画面に表示する

    printf(“Val %d¥n”,sw1);

- プログラムの状態を変える事が出来る
  - LEDのON/OFFをキーボードで出来る

```
If (key == 1) {
LED1=1;//LEDをONにする
}  else {
LED1=0; //LEDをOFFにする
}
```

マイクロプロセッサと対話処理が可能になる
例：センサーの値とモータの回転方向を表示し確認する

SWの入出力

- SWは入力として使うのでTRISA,TRISBを設定
  - 入力1,出力は0
- プルダウン接続かプルダウン接続か？
  - RA1:プルダウン接続=>正論理
  - RA2:プルダウン接続=>負論理

TRISAの設定

ピン名	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
接続先	SW	LED	Reset	VR	H_Brd.	H_Brd.	SW	LED
入出力	入力	出力	-	入力	出力	出力	入力	出力
TRISA	1	0	0	1	0	0	1	0

センサーの入力

- センサーはいろいろな物がある
  - モータの回転角度：エンコーダー
  - モータの回転速度：タコジェネレータ
  - タッチセンサー，レーザー距離計，SW，etc
- センサーと何かを結び付けると制御システムが完成する
  - IoT：マイコンとセンサーと無線
  - ロボット：マイコンとセンサーとモータ
- 赤外線反射センサー(RPR220)
  - 赤外線が反射：白，吸収：黒
- 今回はインバータ(NOT)を入れているので難しい
  - 信号入力の安定用

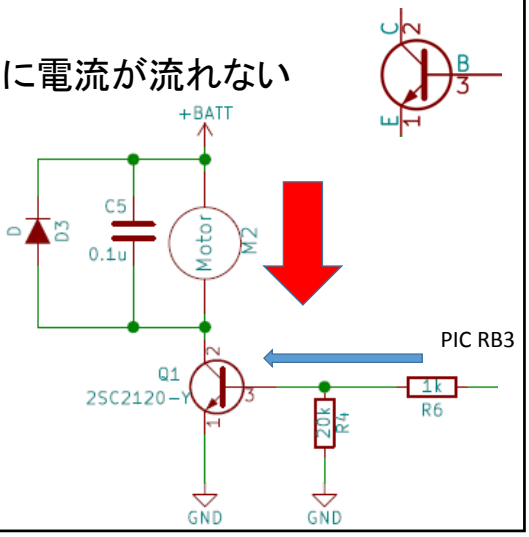
赤外線	センサー出力	74HC04入力	74HC04出力	LED D5	PIC RB4
反射	5V	5V(Hi)	0V(Low)	OFF	0
無反射	0V	0V(Low)	5V(Hi)	ON	1

モータのドライブ

- マイコンでモータを動かすには？
- 絶対やってはダメ！！！
  - モータを直接マイコンのPINに接続する
  - マイコンのPINに過電流が流れ，マイコンが故障する
- トランジスタ・FET・Hブリッジを使うこと！！
  - 基礎電子工学を思いだそう！！
  - トランジスタ(小型DCモータ)
    - 電流で制御する3端子の半導体素子
    - PNP,NPN型
    - 正転のみ
  - FET(中小型DCモータ)
    - 電圧で制御する3端子の半導体素子
    - 正転のみ
  - Hブリッジ(中小型DCモータ)
    - 4つのトランジスタを組み合わせた半導体IC
    - 正逆転，ブレーキが可能

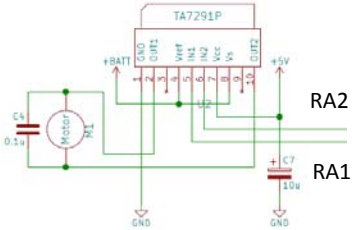
トランジスタによるモータドライブ

- RB3をONにするとベースに電流が流れる
- BATTからGNDに,モータを動作させる電流が流れる
  - モータが回転する
- RB3をOFFにするとベースに電流が流れない
  - モータがSTOP
- **フライフォイールダイオード**
  - **必須:電子回路の保護**
  - **アノードを+側**
  - **カソードを-側に**
- **ノイズ除去用コンデンサー**
  - **モータから発生するノイズを除去**
  - **必ず付ける事！！**

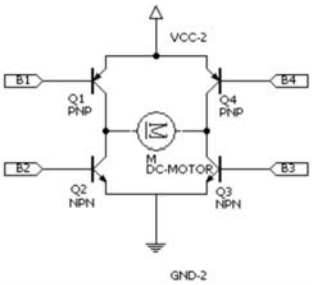


Hブリッジによるモータドライブ

- モータを正転逆転させる事が可能
  - 移動台車，ロボットアームでは必要
- 2つのPNP,2つのNPNのトランジスタを使っている
- TA7291Pでは，IN1とIN2で制御する
  - 電解コンデンサC7を忘れないこと.
    - モータが脈動します.



IN1(5番ピン)	IN2(6番ピン)	OUT1(2番ピン)	OUT2(10番ピン)	モード
Low	Low	∞	∞	ストップ
High	Low	High	Low	正転
Low	High	Low	High	逆転
High	High	Low	Low	ブレーキ



B1	B2	B3	B4	モード
Low	Low	High	High	正転
High	High	Low	Low	逆転
High	Low	Low	High	停止

モータドライブ

- PINのON/OFFでモータは回せるが．．．．．
- 速度はどうやって制御するの?
- マイコンのPINを高速にONすれば速度が変わります

PWM制御をすればよい!!
3日目にやります！！

ライントレースロボットを作るには

- 今日までの回路を組み合わせればOK
- STEP1
  - 電源回路(3端子素子，パイロットLED)
- STEP2(基本回路)
  - マイコン
  - ISP端子
  - テスト用LED　2個
  - RS232C端子(デバッグ用)
- STEP3
  - 赤外センサー　x2
  - モータドライブ回路(トランジスタ) x2

回路図を参考に自分で方眼紙に回路図を作る
自作の回路図から実体配線図を書く

制御システム工学演習
-リファレンスボード編-

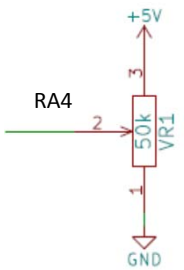
3/4

http://www2.kaiyodai.ac.jp/~jtahar0/posts/post.html

http://bit.ly/2x9HuL8

ADコンバータ

- ADCはアナログ(電圧)をデジタルの数値に変換する
- 今回は5Vの電圧をボリュームで分圧された値を計測する
  - 実際には，照度計等が
- 0[V]~5V[V]が以下の様に変換される
  - 10bit，1bitの重みは5/1024=4.88m[V]
  - 0V=>0x00(HEX)，00(DEC)
  - 5V=>0x3F，1023(DEC)
- ADコンバータ内部では10bitを
  - ADH(上位)=2bit
  - ADL(下位)=8bitで表現
  - AD=ADH<<8 + ADL
    - ADHを8bit右シフトする
  - Val[V]=AD*4.88/1000で電圧値がでる.
    - PICではPrintfで floatが印字出来ないので注意



タイマー割り込み

- タイマー割り込み，今回はTimer0を使う
- タイマー割り込みを使うと定期的な時間制御が可能
  - 自動制御ではサンプリングタイムと呼ぶ
  - サンプリングタイムが不安定だと制御が不安定
- 擬似的なマルチタスクが可能

Timer使用による疑似マルチタスク

サンプリング間隔(1mS) 等

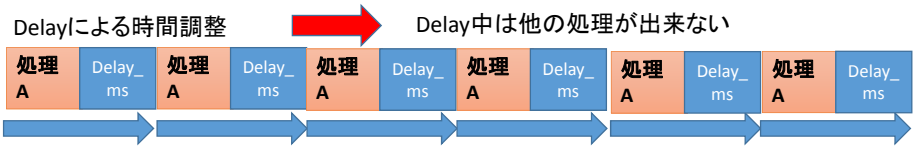
- 周期タスクをサンプリングタイムごとに呼び出し，残り時間でタスク2を実行する
- 正確なサンプリングが可能
- タスク2の処理時間<Ts-周期タスク時間である必要がある.
- オーバーすると暴走や中断が起こる

Delayを使って時間調整をしてはいけない理由

- 処理Aの時間が変動した場合delayで設定した時間がずれる
  - **サンプリングタイムが変動する**
  - Ts=100mSと設定したが．．．
  - 処理Aが見積もりでは75mSなので，delayを25mSに設定，Ts=75+25=100mS
    - 処理Aが50mSの時,Ts=50+25=75mS
    - 処理Aが80msの時，Ts=80+25=105mS

→ サンプリングタイムがずれる

- Delay中は他の処理が出来ない
  - 車でいうと障害物を発見しても止まれない．．．．



タイマー割り込みの使い方

- プリスケーラー(倍率)を決める
  - 256,128,64,32,16,8等が決定出来る．ここでは256
  - 割り込み間隔を求める(Ts)
- 減算カウンタcnt値を決定する
  - Ts時間経過すると**cnt=cnt-1**が実行される
- cnt=0となった時に**「周期タスクが実行される」**

```
void interrupt isrTMR0() // ISR:Interrupt Service Routine
{
   TMR0IF = 0;      // TMR0割込みをクリア
   cnt--;
   if(cnt == 0){
      cnt = CNT_DATA;       // 1秒カウンタ用変数を初期化
      PORTA ^= 0b01000001;   // RA0(LED1)とRA6(LED2)をビット反転
      printf("hello world¥n");
   }
}
```

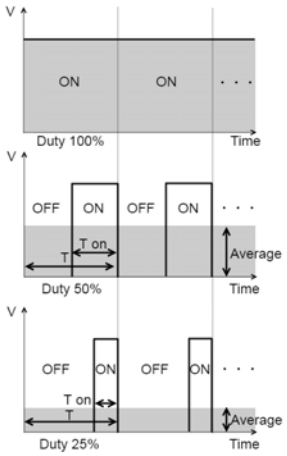
注意点
周期的に行いたい動作は
void interrupt isrTMR0
という特別な関数に書く
割り込み関数ですよという意味

$(1[s]/\mathrm{CLK}[Hz]) * 4 *$ プリスケーラ設定値 $* (255 - (TMR0$初期値$) - 1) =$ 割り込み間隔$[s]$

cnt=1/Ts

ライントレーサーでの割込の考え方

- センサーの感知とは別に速度を指示する**PWM**で行う
- PWM (Pulse Width Modulation（パルス幅変調）)制御
- パルスの幅を変えることで
  - エネルギーの量を制御する
  - Dutyと呼ばれる幅で制御
- 使用例
  - モータの速度制御
  - インバータ
    - 直流を交流に変換
  - LED照明の明るさを変化させる
  - ゲームのノイズ音源
  - 距離計測システム



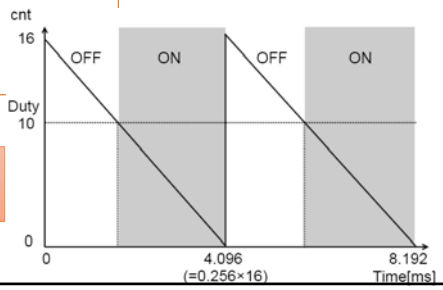
Timer0を使ったPWM制御

- 割り込み時間Tsをなるべく小さくする
  - 今回はプリスケーラーを2とする
    - Ts=0.256mS
  - カウンターをcnt=16とする
  - PWM周期はTs*cnt= 4.096 mS

$$(1/8000000) * 4 * 2 * \big(255 - (0 - 1)\big) = 2.56 * 10^{-4}[\mathrm{s}] = 0.256[\mathrm{ms}]$$

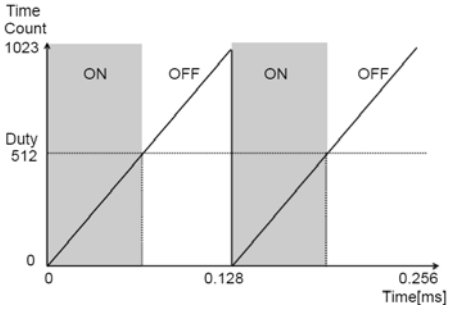
```
// タイマー0割り込み発生時に実行される関数
void interrupt isrTMR0() // ISR:Interrupt Service Routine
{
PWMの処理
}
```

PWM波形
Dutyをより小さい間OFF，OVERするとON



CCPモジュールを使ったPWM

- 最近のマイコンにはCCP(Compare/Capture/PWM)モジュール機能が入っている
- PIC16F88では1個入っている
  - PIC24ならば6個持っている
- タイマーをセットし，カウンターをセットすると自動で計算してくれる
- 条件を満たしている時はRA4をONとする
- 関数以下を参照の事
  - void initPWM1
  - void setPWM1Duty
- 基本ハードウエアが行う
  - ソフトウエアのタスクを考えなくてよい



たのしいおまけ？

- PWM機能を使うと音も出せる
- 音は，周波数を変化させた物だから．．．．
- モータの代わりにブザーを付けると音が鳴る
- ドはCで261.626Hz
  - 1オクターブを1/12等分した物が音名となる
  - 詳しく調べると平均律とかピタゴラス率とか面白い
- 昔のゲームはこうやって音を出していた
  - シンセサイザーのノイズ音源として使われている

- 移動台車だと愛知機械テクノシステム(株)のAGVが有名
  - 正式名称:Carry Bee
  - 通称：サザエさんカー

一見，楽しそうに見えるが．．

- AGVで荷物を運んでいる
- 逆に考えると．．．
  - **荷物を運ぶ作業員はいらなくなった**
- 何も考えずに日々過ごしている人は仕事が奪われる
  - 単純労働や法律守られている職業はいずれ．．．．
  - 機械に=>ロボット化
  - スキルの必要無い作業員=>期間工
  - 労働市場開放=>外国人労働者に
    - 外国人船員
    - 日本企業の外国企業化

→ 失業

技術者として食べて行くには，新しい技術にチャレンジし，スキルを磨くしかない

ライントレースロボットを作るには

- 今日までの回路を組み合わせればOK
- STEP1
  - 電源回路(3端子素子，パイロットLED)
- STEP2(基本回路)
  - マイコン
  - ISP端子
  - テスト用LED　2個
  - RS232C端子(デバッグ用)
- STEP3
  - 赤外センサー　x2
  - **モータドライブ回路(トランジスタ) x2**

→ Timer0を使ったPWMを使うと良い

→ 回路図を参考に自分で方眼紙に回路図を作る
自作の回路図から実体配線図を書く

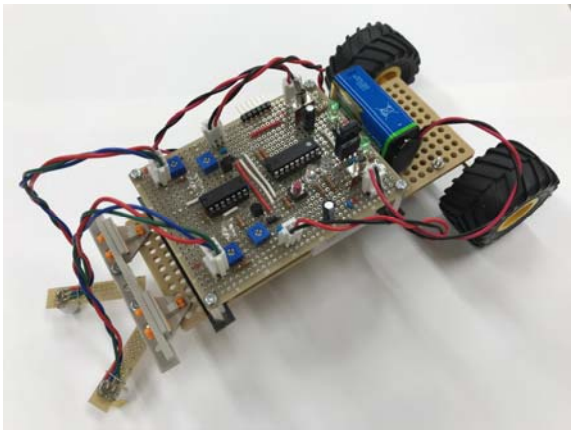
制御システム工学演習
-リファレンスボード編-

4/4

http://www2.kaiyodai.ac.jp/~jtahar0/posts/post.html

http://bit.ly/2x9HuL8

ライントレースの作成のポイント



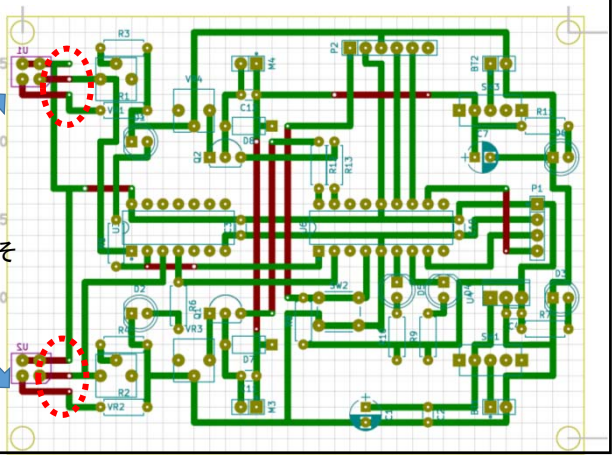
ラントレースロボット資料 Ⅰ

- ライントレースの本資料とプログラムは，
  【講義テキスト】ライントレーサー作成編にあります．
- **URL**
- **1)ライントレーサー回路図**
  - トランジスタを使ったライントレーサーの標準回路図
  - 基本回路図です．この通りに作成すると必ず動作します．
- **2)ライントレーサー部品表**
  - ライントレーサーの標準部品表
  - ライントレーサーを作成するための部品です．
  - Hブリッジは今回は使いません．
  - トランジスタを使ってください．
- **3)PICのPIN割り振り表**
  - ライントレーサーの標準PIN配置図
  - ライントレーサーのPIN配置表
  - です．PICと回路図の接続の確認
  - TRIS命令の設定

ライントレースロボット資料 Ⅱ

- **4)ライントレーサー実態配線図**
  - CADによる実態配線図
  - 表から，部品を刺していくこと
  - **緑のラインは裏の配線**です．スズメッキ線で配線してください
  - **赤のラインは被服線で表から配線**してください．

赤外線リフレクターのU1とU2
は裏から部品を配置して配線
ワイヤーで接続したい場合はそ
ばの3つの穴から配線

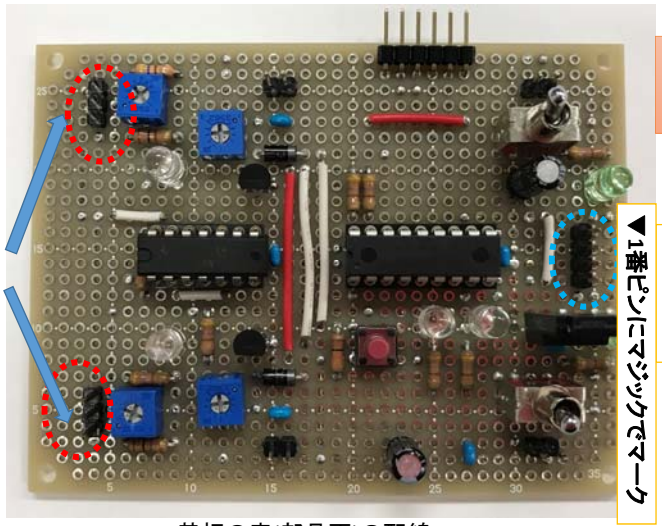


ライントレースロボット資料　III

- 実際のハンダ例
- ピンヘッダーでISP,rs232cを作る

ISP端子

▼1番ピンにマジックでマーク



両面スルーホール基板
穴の部分で裏と表が
つながっている

フォトリフレクター端子

▼1番ピンにマジックでマーク

Rs232c端子

基板の表(部品面)の配線

ライントレースロボット資料　IV

- スズメッキ線で配線

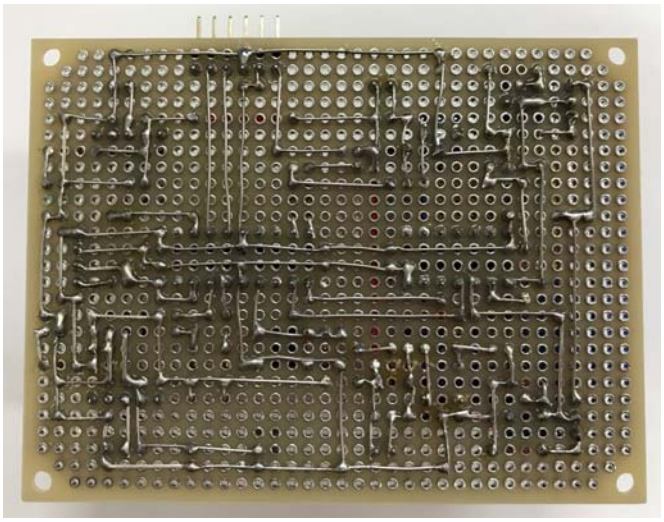
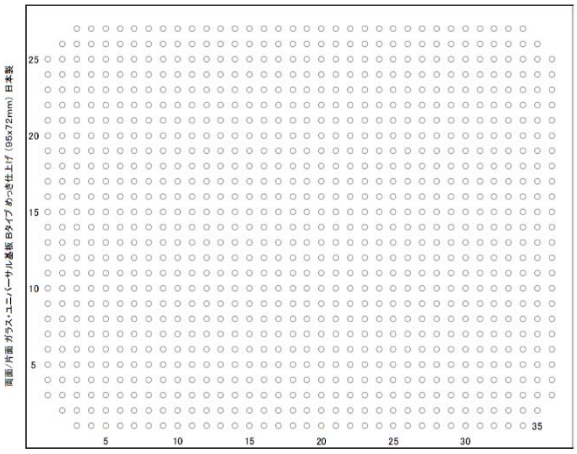


写真　ハンダ面の配線

ライントレースロボット資料　IV

- **5)ユニバーサル基板図**
  - ユニバーサル基板の図.
  - これに実態配線図を作成



ライントレース作成の手引き

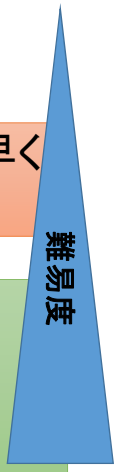
- まずは，基本のライントレース回路，プログラム
- 変な個性を出そうとしてセンサ数を増やしても
  - 苦労するだけでタイムは上がりません
- **生物の進化の過程を見ても．．．．**
  - **巨大化したり，代わった形状に進化した生物は絶滅.**

- 繁栄している生物は**頭脳を進化させる戦略**をとる
  - 人間:頭脳により最も繁栄した生物
  - 昆虫:単純だが高速な信号処理を実現している
  - **ライントレースロボットは昆虫に近い**
  - ネズミ:しっぽをタッチセンサとし壁伝いに移動する

ロボットはハードウエアではなく
ソフトウエアが勝敗を分ける

早いライントレーサを作るには？

- LV1：とにかく早く
  - カーブで曲がりきれずタイムロス
- LV2：カーブでは，ゆっくり曲がる．直線では早く
  - カーブが続くとタイムロス
- LV3:カーブと速度の関係を予想する
  - 台車の速度とカーブの関係式を知る必要
  - PIDコントロール
  - 制御理論



テクニック

- 1)必ず回路図とピン配置表を作成してください．
- 2)回路図から，実体配線図を作成してください．
  - 線は色分けするとよい
  - 実体配線図は表から書きます．
  - コピー機のミラーモードでコピーしてください．裏からの配線図になる
- 3)実体配線図をベースにして電源回路から作成
  - パイロットLED，SW，三端子レギュレータ
  - 5Vが出るか確認
- 4) PICとISP端子，テストLED，rs232c端子，リセット回路を配線
  - ISP端子とrs23cはピンヘッダー
  - **リセット回路を忘れる例が多いです．**
  - 「**16F88_linetrace_blink.X**」
  - 1HzでLEDの点滅とテラタームに「Hello World!」を確認してください．

テクニックII

- 5) ON/OFF制御:「**16F88_linetrace_r1.X**」
  - カーブの多い場所だとゆっくり動き，直線では早く動くが出来ない
  - 基板とベースを固定するのに，「田原研アクリルステイ」を使って固定
    - レーザーカッターで切り出し
- 6)DelayによるPWM:「**16F88_linetrace_r2.X**」
  - モータをdelayを使ってPWM制御する
  - 望ましいプログラムではない
  - カーブの多い場所だとゆっくり動き，直線では早く動くといった事が可
  - 一つのタイミングを変えると全てのタイミングが変化し安定しない
- 7)TimerによるPWM:「**16F88_linetrace_r3.X**」
  - Timer0を使ってPWM制御を行うプログラム
  - 最終目標
  - 調整する事で滑らかな動きが可
  - タイマーで時間管理
  - PWMの動作を変えても動作タイミングは代わらない