mbed ロボトレーサ解説

2013.7　葉山清輝（熊本高専）

１．概略

ニューテクノロジー振興財団の主催するマイクロマウス大会で開催されるロボトレース競技に合わせて，プロトタイピングツールの mbed（http://mbed.org/）を利用して設計したライントレースロボットです．２相ユニポーラステッピングモーター２個を走行用のモータとして利用します．ラインセンス用に６個，スタート・ゴールマーカーとコーナーマーカーのセンス用に２個の計８個の光センサを搭載しており，mbed のアナログポートよりライン上の位置を読み取ってモータを制御して走行します．mbed のソフトウェアの開発環境は PC にインストールする必要がなく，インターネットに接続された PC から mbed のサイトにアクセスすれば，ブラウザ上で開発環境を利用でき，mbed の基板と PC を USB で接続して bin ファイルをダウンロードし，リセットボタンを押すことで実できます．

図１は低価格で製作するために，秋月電子通商で販売されている安価なステッピングモータと取扱いが容易で安価な単３電池による構成例です．図２は大会参加を視野に入れて高トルクなステッピングモータと小型軽量なリチウムポリマーバッテリーを使った高速走行が可能な部品構成例で，高速走行時の安定性を増すためのジャイロセンサも搭載されています．

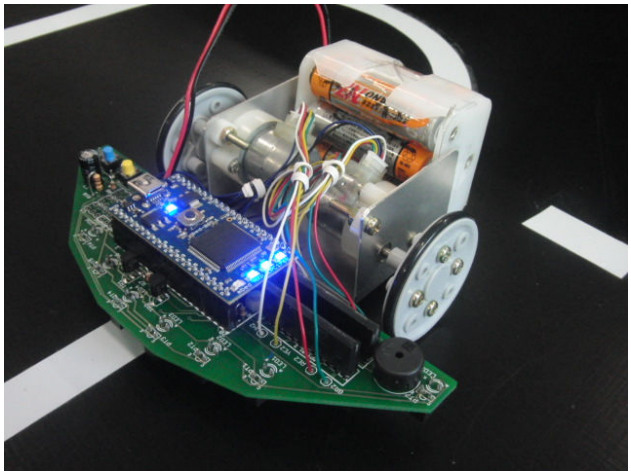

図１　安価な部品構成

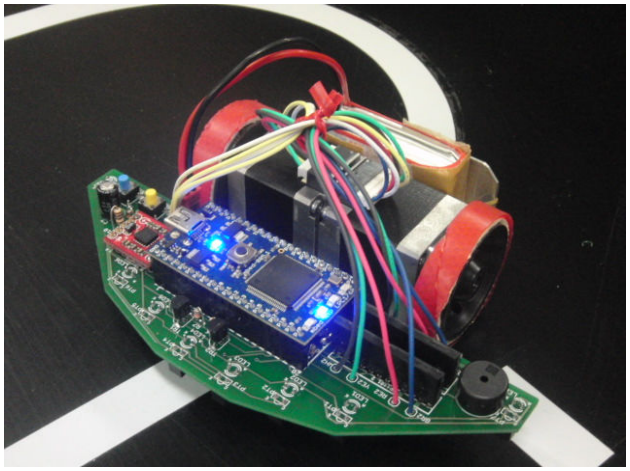

図２　高速走行可能な部品構成

２．基板と回路の説明

図３は回路の設計画面，図４は回路図です．

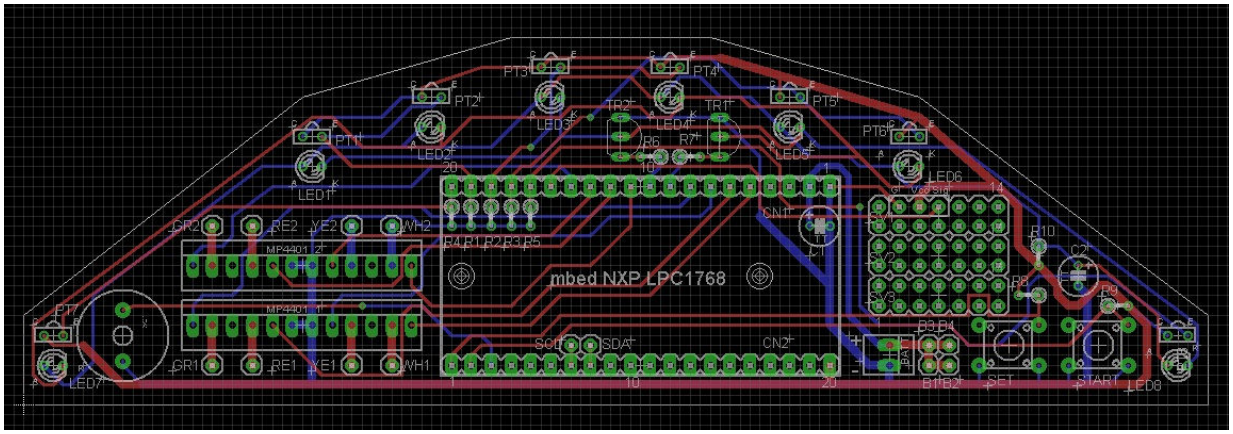

図３　回路の設計画面

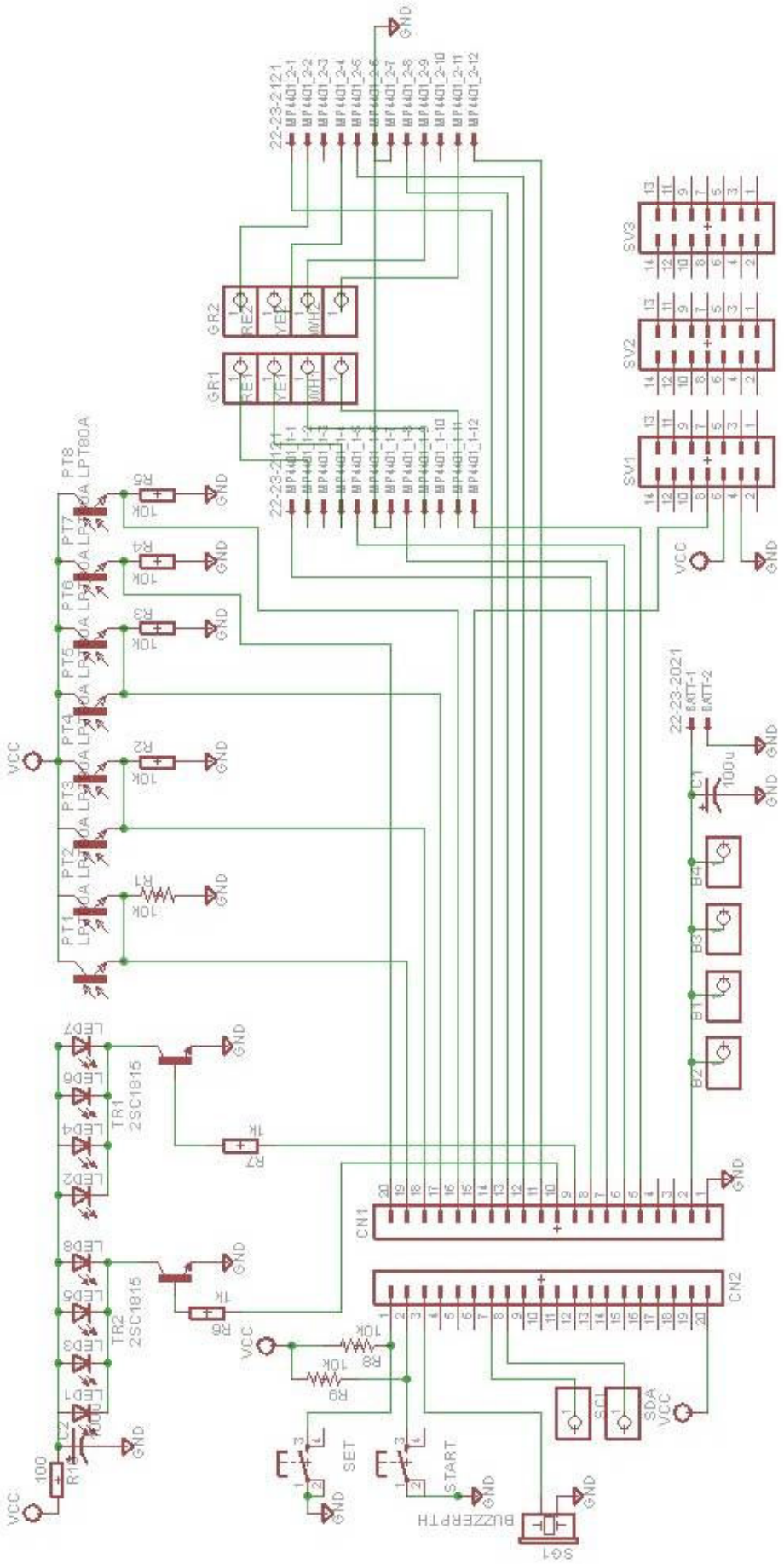
GND
VCC
SV1
SV2
SV3
GR1
GR2
RE1
YE1
WH1
RE2
YE2
WH2
CN1
CN2
TR1
TR2
2SC1815
SET
START
SDA
SCL
BUZZERPTH
SG1
BATT-1
BATT-2

図４　回路図

mbed のアナログセンサは６ポートしかありませんので，ジャイロセンサ(CN1-p15)，スタート・ゴールマーカー(CN1-p16)，コーナーマーカー(CN1-p21)にそれぞれ１ポートずつ割り当てると，ラインセンサに使用できるのは３ポートしかありません．そこで，ラインセンサの幅をできるだけ広くとることが出来るようにフォトセンサを２個ずつ並列にしてアナログポートに割り当てました(CN1-p19 を PT１と PT2, CN1-p18 を PT3 と PT4, CN1-p17 を PT5 と PT6)．ラインの照明用に対応する LED（左は LED1,3,5,右は LED2,4,6）を左右で交互にパルス点灯してそれと同期してフォトトランジスタを読み取れば，左右のセンサの値を分離することができます．

ジャイロセンサは製品に切り替わりが早く，初心者と競技志向で共通に利用できるもの安価な物が見当たりませんでしたので，基板の右側にユニバーサル領域を設け，前部に電源，GND 及びジャイロ用のアナログポートを引き出してあります．また，I2C 接続のジャイロセンサも利用しやすいように，CN2-7 と８ピンのパッドを基板上に引き出してあります．

ステッピングモータの駆動回路は安価に構成するためにパワーFET モジュール（MP4401）にモータを直結して回すようにしました．モータの逆起電力に対する保護回路や，モータの電流制限回路は入っていませんので，場合によっては mbed ボードにダメージを与える可能性がありますが，試作品では今のところ問題なく動作しています．

そのほか，基板にはモード選択用のスイッチ（CN2-1），スタートスイッチ（CN2-2）と，走行中のマーカー検出を知らせるための圧電ブザー(SG1)を搭載しています．

３．基板の製作と組立

ロボトレース基板に部品をはんだ付けして組み立てます．部品表は表１に示す通りです．

表１　部品表

	品名	基板上の記号，備考
1	mbed マイコンボード　NXP　LPC1768	mbed　NXP　LPC1768
2	教育用ロボトレース基板	
3	カーボン抵抗（炭素皮膜抵抗）1／6W　1kΩ	R1,R2,R3,R4,R5,R6,R7
4	カーボン抵抗（炭素皮膜抵抗）1／6W　10kΩ	R8,R9
5	カーボン抵抗（炭素皮膜抵抗）1／4W　100Ω	R10
6	電解コンデンサー　220μF　25V	C1,C2
7	トランジスタ　2SC1815	TR1,TR2
8	パワーMOS－FETモジュール　MP4401	MP4401-1,MP4401-2
9	LED	LED1,LED2,LED3,LED4,LED5,LED6,LED7,LED8
10	フォトトランジスタ(LED とフォトトランジスタはフォトリフレクタを使用可，LBR-127HLD は極性入れ替えが必要)	PT1,PT2,PT3,PT4,PT5,PT6,PT7,PT8
11	圧電ブザ	SG1
12	タクトスイッチ	SET, START
13	ピンソケット（20P×２）	
14	バッテリースナップ（電池スナップ）プラスチック製	BATT
15	電池ボックス（単3X6・白・スナップ）	
16	ユニポーラステッピングモータ	SPG20-1362（秋月電子通商）など
17	自在タイヤφ40　２個入り	山崎教育システムより入手可能
18	スペーサ(10mm 程度)	
19	アルミ板材 A5サイズなど	適宜製作する
20	スベリ材	
21	（ジャイロセンサ，必要な場合）	SIG,Vcc,G(ユニバーサル領域)

LED とフォトトランジスタは個別の部品を用いても良いですが，LED の光が直接フォトトランジスタに入らないように側面をカバーしなければなりません．LED とフォトトランジスタペアになったフォトリフレクタを利用すると便利です． LBR-127HLD というフォトリフレクタが秋月電子通商から入手できますが，このままですとフォトトランジスタの極性が逆となり使えません．図５に示すように簡単に分解出来ますので，フォトトランジスタ（黒い方）の極性を入れ替えて使ってください．

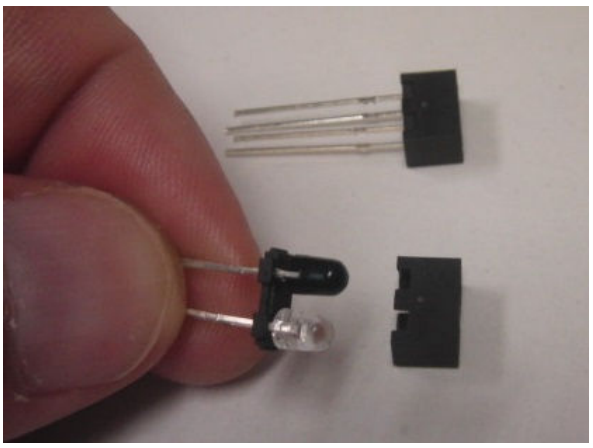

図５　フォトリフレクタを分解した様子
（黒い方がフォトトランジスタで極性の入れ替えが必要）

図６の組立後の写真を参考に部品を取り付けてください．フォトリフレクタは基板の裏面に取付け，それ以外の部品は基板の表面から取り付けます．トランジスタ，パワーMOSFET モジュール，電解コンデンサなど極性のある部品については回路図と基板のパターンの対応を考えながら方向を間違えないように取り付けてください．

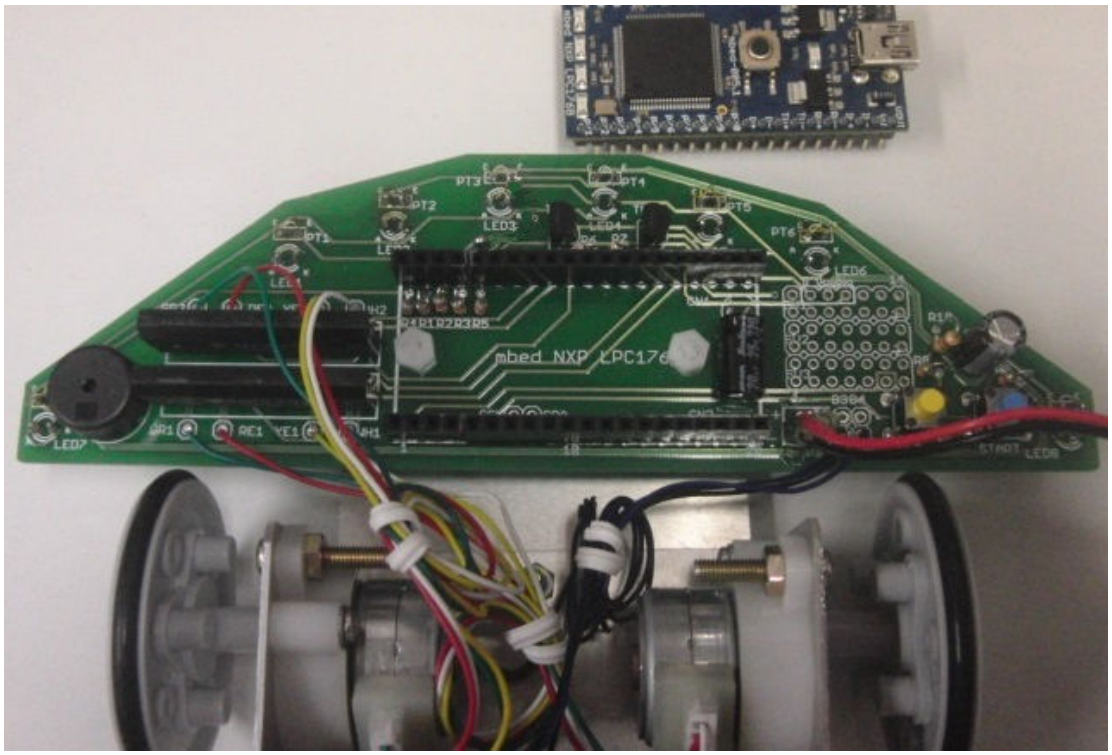

図６　組立後の写真

mbed 基板の下に収まる電解コンデンサは，足を折り曲げて横に倒してください．バッテリースナップは動かしていると断線しやすいので配線が終わった後に接着剤等で固定するとよいでしょう．

ステッピングモータの配線は，秋月電子通商から入手できる SPG20-1362 に付属の配線の色を元に表示しています．基板上の GR1 が右モータの緑色の配線です．そのほか，RE1 が赤色，YE1 が黄色，WH1 が白色の配線となります．GR2，RE2,YE2,WH2 には左モータの配線を接続してください．左右モータの黒と青の配線は電源に接続しますのでバッテリースナップ横の B1,B2,B3,B4 にまとめてハンダ付けしてください．

他社のモータを利用する場合は，GR，RE,YE,WH の順に励磁するとモータが回転するように取り付けてください（回転方向については，ソフトウェアで変えられます）．

低速で走行する場合にはジャイロセンサは不要ですが，高速走行を目指すなら適当なものを探して取り付けると良いでしょう．基板の右側にユニバーサル領域があり，VCC（3.3V），GND 及び Sig(信号線)が引き出してあります．

使用するモーターとタイヤ・ホイール，バッテリーに合わせて機体を制作し，基板を取り付けてください．図７に示すように，基板はコース面から 15mm 程度（写真では 16.5mm）離し，センサはコースから 10mm 程度離して配置してください．図８のように機体を中央に置くとラインの両サイドの２個のセンサがそれぞれラインの約半分を検知するようになります．

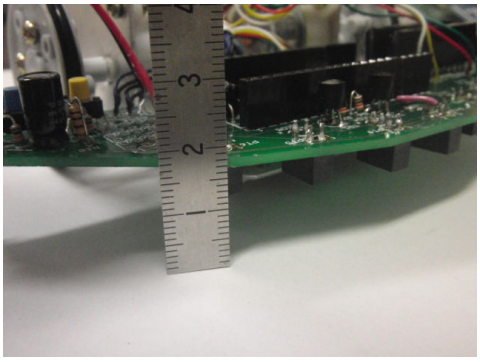

図７　基板の高さ

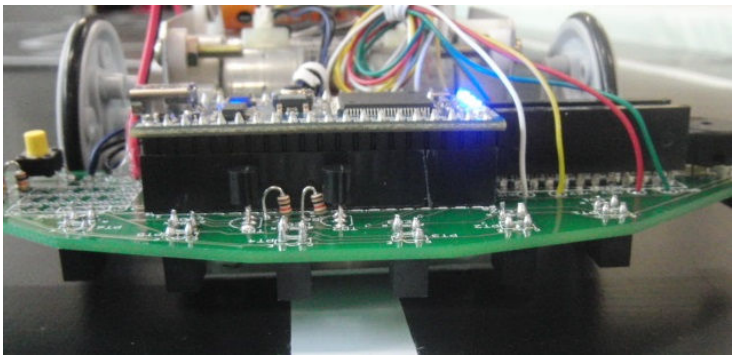

図８　ラインとセンサの関係

４．サンプルプログラムと操作方法について

mbed の使用方法は，書籍や Web サイトを参照してください．例えば，

mbed のホームページ　http://mbed.org/

じぇーけーそふとのこーなー　http://jksoft.cocolog-nifty.com/blog/mbed/

超お手軽マイコン mbed 入門，http://shop.cqpub.co.jp/book_guide/detail/17521/

などが参考になります．

サンプルプログラムは，　mbed のサイト内に公開している以下のリンクから入手出来ます．

https://mbed.org/users/hayama/code/mbedRobotracer_Edu/

zip 形式でダウンロードして，自分の mbed の workspace にインポートすると良いです．コンパイルして mbed に書き込んでください．

操作方法について説明します．

１）リセットボタンを押してプログラムを動作させます．

２）セットボタンで動作モードを選択します．動作モードは mbed ボード上の４個の LED に２進数で

表示されます.

３）スタートボタンで選択したモードで動作させます.

モード０：　センサチェック，シリアル通信でセンサの読みを PC に表示させます.

モード１：　５００ステップ前進

モード２：　５００ステップ右回転

モード３－７：　ライントレース走行

モード０のセンサチェックでは，USB 接続を利用したシリアル通信を行います．以下の URL を参考にドライバーをインストールしてください.

https://mbed.org/handbook/Windows-serial-configuration

ライントレース走行では，モードを進めると速度が上がりますがステッピングモータが脱調しやすくなります．スタート，ゴールマーカを読んで自動停止します.

大会に出場する場合は，タイムを縮めるために１回目の走行でコースを記憶して，２回目以降は加減速しながら高速走行しますが，サンプルプログラムにはこの機能は含まれていません．どのようにコースを記録し，高速走行させるかが大会の醍醐味ですので自分で作ってみてください.

以下，簡単にサンプルプログラムの説明をします.

最初に各種変数の定義と Ticker インスタンス（繰り返しタイマー割込み）の定義，バスの定義，アナログ入力とデジタル I/O の定義を行なっています.

バス定義した leds に値を与えることで LED による２進数表示ができます．ステッピングモータを回すには，motorR や motorL で定義したバスに励磁パターンを順次書き込みます.

アナログ入力はフォトセンサに５ポートでそのうち３ポートはフォトトランジスタを並列に接続して使います．ジャイロセンサに残りの１ポート使用します．例えば　AnalogIn pt12(p19)　というのは，p19 のアナログポートに，PT1 と PT2 を並列に接続して接続し光強度をアナログ値で読みます．この際，　DigitalOut ledCout(p9)と DigitalOut ledSGout(p10)　で定義した LED の点灯用の信号でどちらか一方の LED のみを点灯させ，並列に接続したフォトトランジスタの一方の信号を読み出します（例えば ledSGout を点灯した時に読み出されるのは PT1）.

最後の main 関数の中の，timerS.attach_us(&Sensor, 2000)　と timerM.attach_us(&stepMotor, 400)　は，それぞれセンサとモータのタイマー割込みの間隔をマイクロ秒単位で設定しています.

センサ読み込みは 2ms 毎に Sensor 関数を呼び出して行います．モータは stepMotor 関数を 400 マイクロ秒単位で呼び出して回します.

Sensor 関数では，timS という変数を 0 と１に交互に変化させ，それに応じて，ラインセンサを半分ずつ交互に読み取っています．スタート・ゴールマーカーとコーナーマーカーも同様に交互に読み取っています.

読み取りの際には，LED 消灯時と LED 点灯時の差を取り，外乱の影響を除外しています.

sensD という変数には，ラインセンサ値を各ビットとしたディジタル値が記憶されます．ラインの有無を判断するしきい値はプログラム最初の define 文で定義される STH の値です．予め定義された sensArray 配列を使って，ラインの位置を PT３と４を中央として，－５（ラインが PT1 上）から５（ラインが PT6 上）までの値に変換して sensDout に記憶します．　この関数の中で，ブザーの ON-OFF も行なっています.

stepMotor 関数では，timR と timL をカウンタとして使いながら，ステッピングモータの励磁パターンの制御を行なっています．modeR と modeL がそれぞれ右モータと左モータの回転方向（または停止）を示します．patR と patL は励磁パターンを与えるインデックスで，励磁パターンは RMOTOR と LMOTOR 配列に定義してあります.

Main 関数内で変数の初期化を行った後，セットスイッチで動作モードの選択，スタートスイッチで選択したモードの実行を行います.

runTurn 関数は左右モータの回転方向，回転速度，ステップを指定して走行させるものです.

ライントレース走行は run 関数で制御されています．引数 n は速度，引数mは速度制御に用いる比例値です．最初に runTurn 関数で指定半分の速度で加速し，指定速度に切り替えます．ラインの位置を検出しながら走行し，ラインがトレーサの中央から外れると，ラインが中央に戻るように片側のモータを減速します．例えば，トレーサがラインから右に外れると左のモータを減速します．ラインから大きく外れるとより急な減速をかけるように制御します．

走行中は，スタート・ゴールマーカーおよびコーナーマーカーを検知してブザーを鳴らすようになっています．frun という変数を使ってスタートマーカーの通過，ゴールマーカーの通過，停止処理を順に行なっています．

ラインが交差した部分とマーカーの読み間違えを起こさないように，マーカーの有無の判断は検出した瞬間に行うのではなく，センサの積算値によってマーカーの幅程度の通過距離でマージンをとって判断を行なっています．マーカーの有無の判断に用いる積算値のしきい値が define 文の SGTH または CTH の値です．

コースの記録を行う場合には，マーカーを利用すると良いです．２回目で速度を上げて走る場合には，例えばマーカー間のコースの距離，曲率，角度，現在と次の区画との関係（次のコースで加速するのか，逆に減速が必要か）を利用して速度や制御パラメータを決定すると良いでしょう．

リスト１　サンプルプログラム

```
//*****************************************************************************
//
//      mbed Robotracer for education
//      (c) Kiyoteru Hayama(Kumamoto National College of Technology)
//
//*****************************************************************************
#include "mbed.h"

Serial pc(USBTX, USBRX);

// run parameters
#define STH      0.5                    // threshold value for digital photo sensor
#define SGTH     10                          // threshold value for start/goal marker
#define CTH      10                          // threshold value for corner marker

// pattern table for stepping motor
const unsigned char RMOTOR[]={0x09, 0x0C, 0x06, 0x03, 0x00};      // magnetization pattern for left motor
const unsigned char LMOTOR[]={0x03, 0x06, 0x0C, 0x09, 0x00};      // magnetization pattern for right motor

const int sensArray[64]={0,5,3,4,1,0,2,0,-1,0,0,0,0,0,0,0,-3,0,0,0,0,0,0,0,-2,0,0,0,0,0,0,0,-5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

unsigned char pmode=0;                          //   program mode

volatile float pt1B, pt2B, pt3B, pt4B, pt5B, pt6B, ptSGB, ptCB;               // sensor values during turn-off the LED
volatile float sens1, sens2, sens3, sens4, sens5, sens6, sensSG, sensC;         // sensor values
volatile int     sensD,sensDout=0;

volatile int markerSG, markerC, markerX;
unsigned char frun, fmarker;
```

```
volatile int spdR,spdL;
volatile unsigned char modeR=0, modeL=0;                // run forward both motor
volatile int stepR, stepL;                              // varilable for set step of motor
volatile unsigned char patR=0, patL=0;                  // index of motor pattern
volatile int cntR, cntL;                                // count of motor steps

volatile unsigned char timR=0, timL=0;                  // timer for motors
volatile unsigned char timS=0;                          // timer for sensors
volatile int bp=0;                                      // couter for beep

Ticker timerS;                  // defince interval timer
Ticker timerM;                  // defince interval timer

BusOut     leds( LED4, LED3, LED2, LED1 );      // for LED display
BusOut motorR(p5,   p6,   p7,   p8 );           // output for right motor
BusOut motorL(p11, p12, p13, p14 );             // output for left motor

AnalogIn pt12(p19);                             // front right sensor, analog input
AnalogIn pt34(p18);                             // front left sensor, analog input
AnalogIn pt56(p17);                             // right sensor, analog input
AnalogIn ptC(p20);                              // left sensor, analog input
AnalogIn ptSG(p16);                             // left sensor, analog input
AnalogIn gyro(p15);                             // for Gyro, analog input, reserved

DigitalIn setSw(p21);                           // set-switch, digital input
DigitalIn startSw(p22);                         // start-switch, digital input
DigitalOut ledCout(p9);                         // LED output signal for corner marker
DigitalOut ledSGout(p10);                       // LED output signal for start/goal marker
DigitalOut buzzer(p23);                         // buzzer out

//--------------------------------------------------------------------------------
// interrupt by us timerS
//--------------------------------------------------------------------------------
void Sensor() {
    // read sensors
    // 1st-step:measure background during LED-off, 2nd-step: measure reflecting light during LED-on.
sensor value is differnce of both.
    timS = !timS;
    if (timS==0){
        pt1B=pt12;                              // measure all background values
        pt3B=pt34;
        pt5B=pt56;
        ptSGB=ptSG;
        ledSGout=1;                             // LED-ON
        wait_us(50);                            // delay
        sens1=abs(pt12-pt1B);                   //"sabunn"
        sens3=abs(pt34-pt3B);
        sens5=abs(pt56-pt5B);
        sensSG=abs(ptSG-ptSGB);
        ledSGout=0;                             // LED-OFF
```

```
    } else{
        pt2B=pt12;                              // measure all background values
        pt4B=pt34;
        pt6B=pt56;
        ptCB=ptC;
        ledCout=1;                              // LED-ON
        wait_us(50);                            // delay
        sens2=abs(pt12-pt2B);
        sens4=abs(pt34-pt4B);
        sens6=abs(pt56-pt6B);
        sensC=abs(ptC-ptCB);
        ledCout=0;                              // LED-OFF
    }

    sensD=0;
    if (sens1>STH ) sensD |= 0x20; else sensD &= ~(0x20);
    if (sens2>STH ) sensD |= 0x10; else sensD &= ~(0x10);
    if (sens3>STH ) sensD |= 0x08; else sensD &= ~(0x08);
    if (sens4>STH ) sensD |= 0x04; else sensD &= ~(0x04);
    if (sens5>STH ) sensD |= 0x02; else sensD &= ~(0x02);
    if (sens6>STH ) sensD |= 0x01; else sensD &= ~(0x01);
    sensDout=sensArray[sensD];

    // corner and start/goal marker detection
    if (sensSG>STH) markerSG++; else if (markerSG>0) markerSG--;
    if (sensC>STH ) markerC++;   else if (markerC>0)   markerC--;
    // cross line detection
    if (markerSG>1 && markerC>1) markerX=1;                 // both marker
    if (markerX==1 && markerSG==0 && markerC==0) markerX=0; // ignore cross line

    // buzzer
    if (bp>0){                  // high beep
        bp--;
        if (buzzer==1) buzzer=0; else buzzer=1;      // alternate ON-OFF
    }
}

//--------------------------------------------------------------------------------
// interrupt by us timerR/L
// motor rotation,   mode = 0: free・・: forward・・: reverse・・: break
// right motor rotation
//--------------------------------------------------------------------------------
void stepMotor(){

    if (timR>0) timR--;                         //count down timR when timR=0 do next process
    if (timR==0) {
        timR=spdR;
        if (modeR==1) {if (patR < 3) patR++; else patR = 0; }
        if (modeR==2) {if (patR > 0) patR--; else patR = 3; }
        cntR++;                                         // count up right moter step
    }
```

```
    // left motor rotation
    if (timL>0) timL--;                                  //count down timL when timL=0 do next process
    if (timL==0) {
        timL=spdL;
        //modeL==1;
        if (modeL==1) {if (patL < 3) patL++; else patL = 0; }
        if (modeL==2) {if (patL > 0) patL--; else patL = 3; }
        cntL++;                                          // count up left moter step
    }

    if (modeR==0 || modeL==0) { patR=4; patL=4; }    // motor free when mode=0
    motorR= RMOTOR[patR];           // pattern output to right motor
    motorL= LMOTOR[patL];             // pattern output to left motor
}

// ------------------------------------------------
// beep
// ------------------------------------------------
void beep(int n){
    bp=n;       // set beep couter
}

//--------------------------------------------------------------------------------
// check sensor value using serial port
//--------------------------------------------------------------------------------
void check_sens(){
  while (1){
    pc.printf("sensC :"); pc.printf("%f¥n",sensC); //
    pc.printf("sensSG :"); pc.printf("%f¥n",sensSG); //
    pc.printf("sens1 :"); pc.printf("%f¥n",sens1); //
    pc.printf("sens2 :"); pc.printf("%f¥n",sens2); //
    pc.printf("sens3 :"); pc.printf("%f¥n",sens3); //
    pc.printf("sens4 :"); pc.printf("%f¥n",sens4); //
    pc.printf("sens5 :"); pc.printf("%f¥n",sens5); //
    pc.printf("sens6 :"); pc.printf("%f¥n",sens6); //
    pc.printf("sensD :"); pc.printf("%d¥n",sensD);
    pc.printf("sensDout :"); pc.printf("%d¥n",sensDout);
    wait (0.5);
  }
}

//--------------------------------------------------------------------------------
// break and release motors
//--------------------------------------------------------------------------------
void run_release(){
    modeR=0; modeL=0;                   // motor release
}
void run_break(){
  modeR=3; modeL=3;                     // mode 0 means break the motor
  wait(0.5);
```

```
   run_release();
}

//-----------------------------------------------------------------------------
// run and turn
// (mR,mL)=(1,1):forward, (2,1): turn right, (1,2): turn left, (2,2): Reverse
// spd: speed,
// nstep: number of step
//-----------------------------------------------------------------------------
void runTurn(int mR,int mL, int spd, int nstep ){
      modeR=mR;modeL=mL;
      spdR=spdL=spd;
      cntR=0; stepR=nstep;
      while (cntR<stepR);
}

//------------------------
// run
// n: run speed, m: factor of reduce speed
//------------------------
 void run(int n, int m){
      int cntGoal=0;      // couter for run after goal

      markerSG=0; markerC=0; markerX=0; fmarker=0;
      frun=0;
      runTurn(1,1,n*2,50);      // slow start

      while(startSw==1 ){

            spdR=spdL=n;
            if (sensDout>0){
                  spdR+=sensDout*m;
            } else {
                  spdL-=sensDout*m;
            }

            // corner marker check
            if (markerX==1) fmarker=0;
            if (markerX==0 && fmarker==0 && markerC>5) fmarker=1;
            if (markerX==0 && fmarker==1 && markerC==0){
                  fmarker=0; beep(50);
            }

            // start/goal marker check
            if (frun==0 && markerSG>SGTH) frun=1;   // start marker detect
            if (frun==1 && markerSG==0){              // start marker fix
                  frun=2; beep(100);
            }
            if (frun==2 && markerSG>SGTH) frun=3;   // goal marker detect
            if (frun==3 && markerX==1)   frun=2;   // ignor cross line
            if (frun==3 && markerSG==0){              // goal marker fix
```

```
            frun=4; beep(100);
            cntGoal=cntR;
        }
        if (frun==4 && cntR>(cntGoal+500)) break;
        wait(0.005);                                    // wait 5ms for control loop
    }
    run_break();
}

//-------------------------------------------------------------------------
//      main
//-------------------------------------------------------------------------
int main(){

    timerS.attach_us(&Sensor, 2000);      // set timer for sensor
    timerM.attach_us(&stepMotor, 400);    // set timer for motor

    while (1) {

        // initialize motor
        run_release();

        while (startSw==1) {                  // program mode selection
            if (setSw==0) {
                wait(0.01);
                beep(50);
                while (setSw==0);
                wait(0.01);
                pmode++;
                if (pmode>7) pmode=0;
            }
            leds=pmode;
        }
        leds=0; beep(50);
        wait(0.5);

        // go selected program
        switch(pmode){
            case  0: check_sens(); break;               // check sensors
            case  1: runTurn(1,1,15,500); break;    // run forward
            case  2: runTurn(2,1,15,500); break;    // turn right
            case  3: run(10,30);   break;               // trace run
            case  4: run(8,24);   break;
            case  5: run(6,18);   break;
            case  6: run(5,15);   break;
            case  7: run(4,12);   break;
        }
    }
}
```