Description of the mbed Robotracer

Kiyoteru Hayama
Kumamoto National College of Technology

1. Summary

The line follower robot called "robotracer" was designed using the prototyping tool of mbed (http://mbed.org/) corresponding to the "Robotrace" competition to be held in the micromouse competition in Japan. There are eight optical sensors, middle six sensors are used for the position on line, the others are used for detection of start and goal markers. And two unipolar stepper motors are used for running robot. The position of the robot on the line is detected by the six optical sensors, and then run by controlling the speed of the left and right motors. The installation of the development environment of mbed is not needed. The development environment can be used on Web browser. The mbed is recognized as a USB flash memory is connected to the USB port of the PC. After compilation the source code, you can download the binary object to the mbed. The binary object can run by the pressing reset button.

In order to make a robot easily for the purpose of education, custom-made printed circuit board (PCB) for robotracer is prepared. Figure 1 shows an example of the robot production by using inexpensive stepper motors and AA batteries for educational use. The PCB also can be used for competitive robot. The robotracer with high torque stepper motors and small Li-polymer battery are shown in Fig. 2. The gyro sensor for increasing stability during high-speed operation is also available.

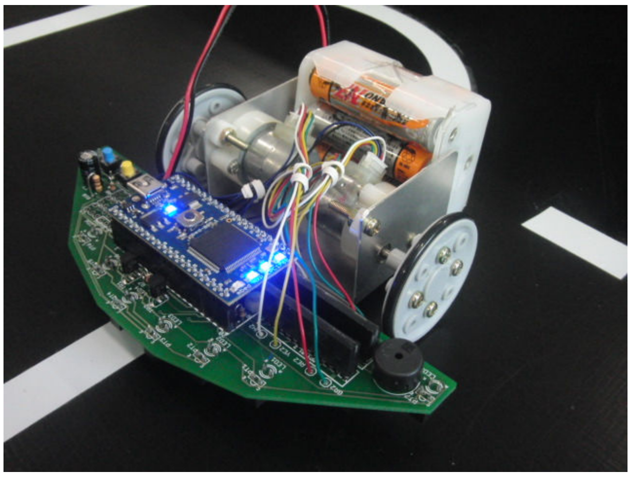


Fig. 1 Robotracer for educational use with inexpensive parts.

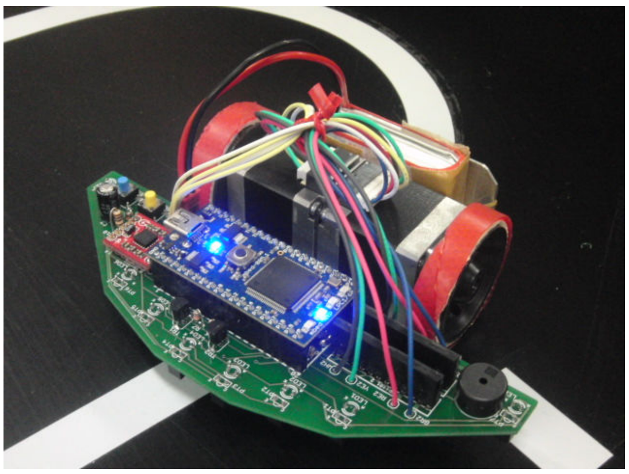


Fig. 2 Robotracer for competition.

2. Description of the PCB

Figure 3 and 4 shows circuit diagram of PCB for robotracer and circuit design of the PCB, respectively. There are only six analog input ports in mbed. So, there are only three ports for optical sensor for the positioning on the line (CN1-p17,p18,p19), excepted for start/goal marker detection ((CN1-p16), for corner marker detection (CN1-p20), and for gyro sensor(CN1-p15).

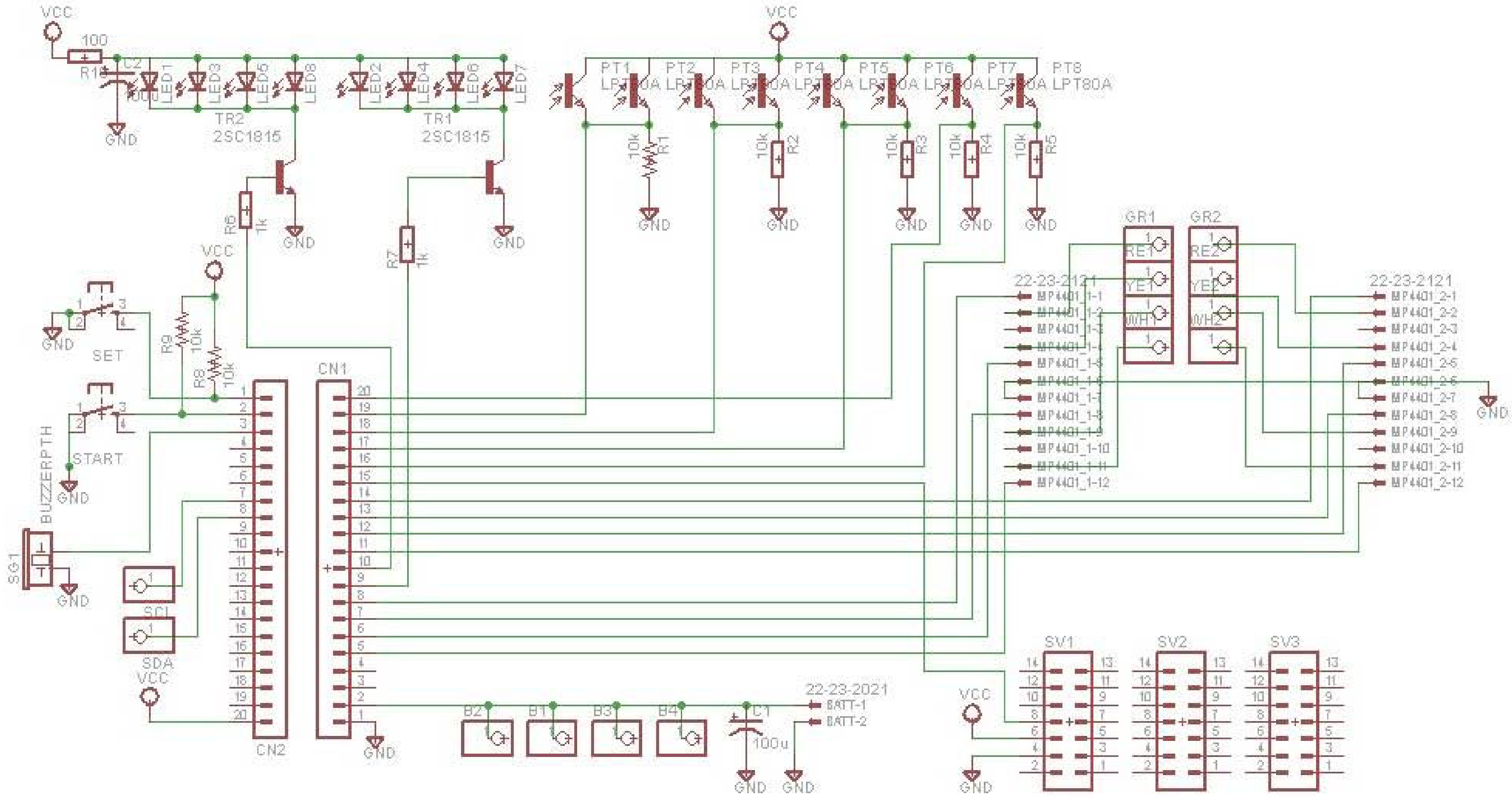


Figure 3 circuit diagram.

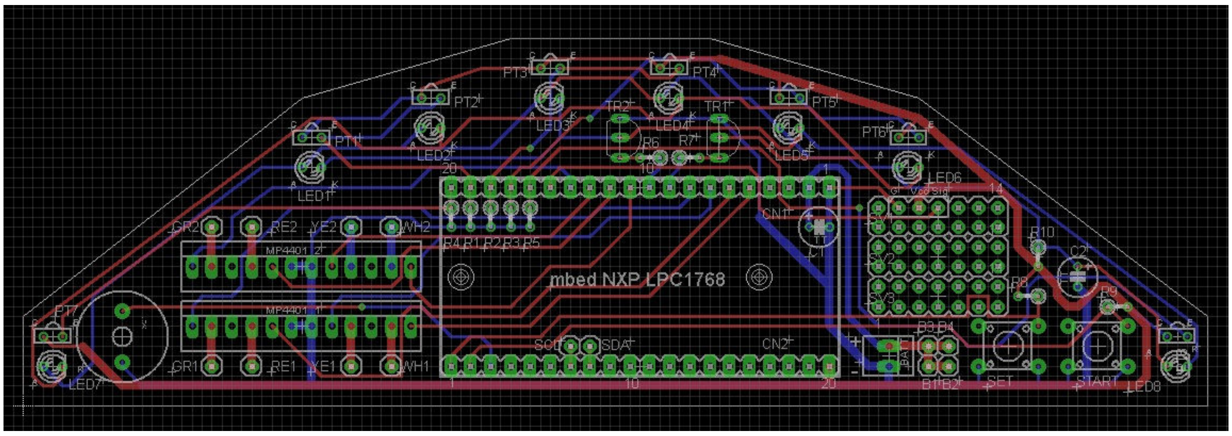


Fig. 4 Circuit design of the PCB

To widen the detection width of the line sensors, two optical sensors are connected parallel of each, six optical sensors are assigned to three analog ports. The parallel connected photo sensors are assigned to analog ports. Its detection are switched by the corresponding LED's light emission.

It was difficult to find inexpensive product of gyro sensor that can be used for both competition and educational. Therefore, a universal area is prepared for the right side of the PCB. The analog port, power supply and GND are arranged to the front part of the universal area. Moreover, two pins of CN2-7 and CN2-8 are pulled out on the PCB, assuming to use I2C communication devices.

In order to make motor driver circuit at low cost, exciting coils of stepper motor are directly connected to power FET module (MP4401), and the excitation pattern of the motor is directly controlled by four digital output port of mbed. The protection for the back electromotive force of the motor and current limiting of the motor are not included. Although there is a risk of giving a damage to mbed for the moment, but it works fine for now in the prototype robot.

In addition, it is equipped that mode select (CN2-1) and start (CN2-2) buttons, and a piezoelectric buzzer (CN2-3) to inform the marker detection.

3. Assembly of the robotracer

Table 1 shows electronic components for mbed robotracer. The circuit board of robotracer is made by soldering each component.

Table 1 electronic component for mbed robotracer

Item number	Description	Designators
1	mbed (NXP LPC1768)	mbed NXP LPC1768
2	PCB for robotracer	
3	1/6-watt Resistors, 1kΩ	R1,R2,R3,R4,R5,R6,R7
4	1/6-watt Resistors, 10kΩ	R8,R9
5	1/4-watt Resistors, 100Ω	R10
6	Electrolytic capacitors, 220μF	C1,C2
7	Transistors (2SC1815)	TR1,TR2

8	Power MOSFET modules (MP4401)	MP4401-1,MP4401-2
9	LEDs	LED1,LED2,LED3,LED4,LED5,LED6,LED7,LED8
10	Photo-reflectors (LBR-127HLD)	PT1,PT2,PT3,PT4,PT5,PT6,PT7,PT8
11	Piezoelectric buzzer	SG1
12	Tact switches	SET, START
13	Pin socket (20Px2)	
14	Battery snap	BATT
15	Battery case (AA x 6)	
16	Unipolar Stepper motor (SPG20-1362)	
17	Tire and wheels (φ40 x 2)	
18	Spacer (10mm x 2)	
19	Metal sheet for body making	
20	Skid	
21	(Gyro sensor, if needed)	SIG,Vcc,G (in universal area)

It is convenient to use a photo-reflector which became photo transistor paired with LED (Fig. 5). If not, the side of LED must be covered to avoid the LED light illuminate to the photo transistor directly. There is a difference of polarity by a product in the photo reflector. In the case of the photo-reflector (LBR-127HLD), the polarity of the photo transistor is reverse to the designed PCB. So the photo-reflector should be recomposed as shown in Fig. 5.

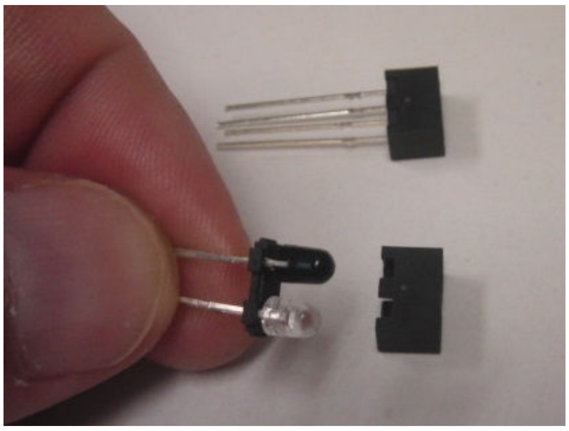


Fig. 5 Photo-reflector

Attach the parts to the PCB as a picture of figure 6. The photo-reflectors are mounted on the bottom side of the PCB, and other parts are mounted on the top side of PCB. Some of the parts, npn transistors, power MOSFETs, electrolytic capacitance, etc. have polarity, don't mistake the direction of the mount of the parts while checking the corresponding pattern of the board and the circuit diagram.

The electrolytic capacitor with bending legs is stored under the PCB. Marks for wiring of the stepper motor are based on the color of the wiring of stepper motor (SPG20-1362). The GR1 on the PCB is for the green wire, the RE1 is for the red wire, the YE1 is for the yellow wire, and the WH1 is for the white wire for the right stepper motor. The GR2, RE2, YE2, WH2 of each color wires are for the left stepper motor. The black and blue wires of both stepper motors are connected to power supply of the B1, B2, B3 and B4. If using the other kind of motors, attach the corresponding wires with same direction of the motor rotation (The direction of rotation can be also changed in software).

Gyro sensor is not necessary at low speed running. If you want make robot for high-speed running, it is preferable to find and install the appropriate sensor to the universal area. After soldering all parts, it is a good to fix the battery snap with an adhesive, because it is easy to break in use.

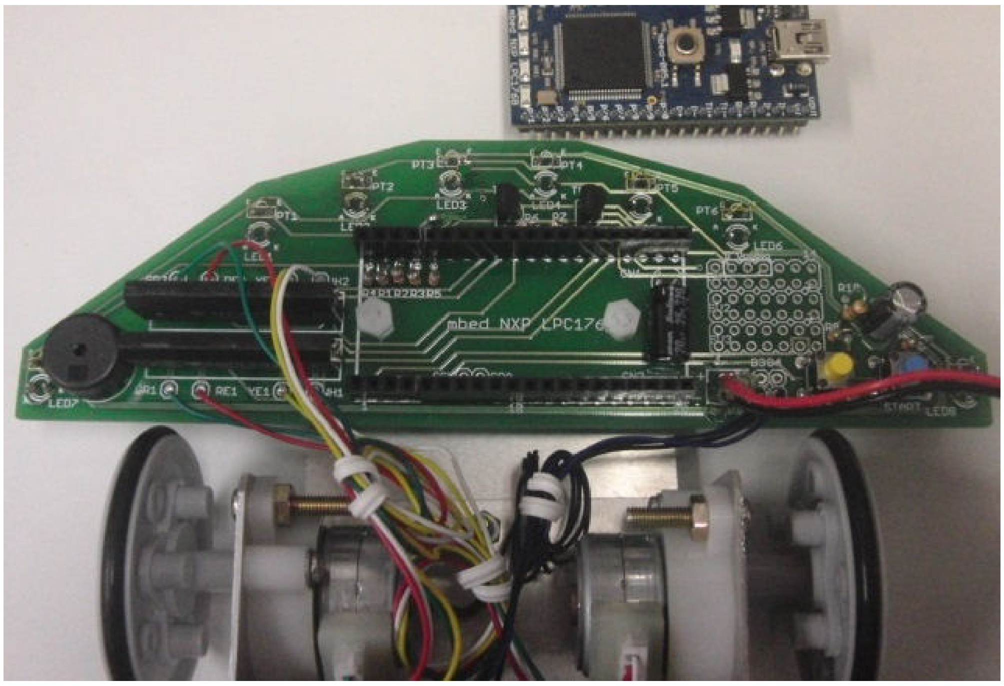
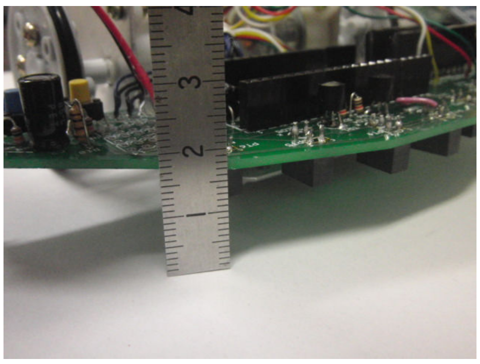


Fig.6 Photo of after assembly of robotracer

The production body to the size of wheels, tires the battery to be used, you assemble the robot. As shown in figure 7, Height of the PCB and photo-reflectors from the course are about 15mm and 10mm, respectively. Place the center of robot on line as shown in Figure 8, middle of two sensors on the line is taken to detect about half of line.



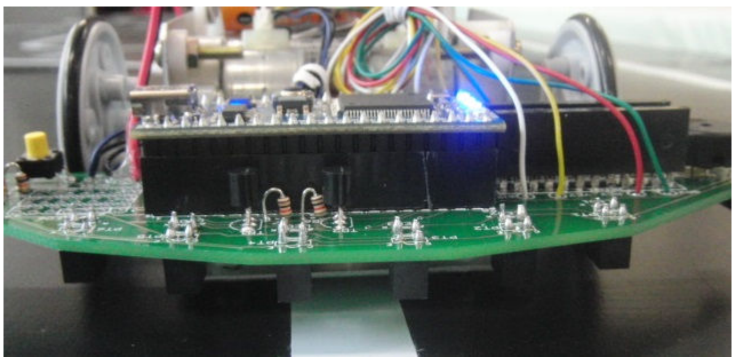


Fig. 7 Height of PCB and photo-reflectors. Fig. 8 Relationship of photo-reflector and line.

4. How to work with the sample program

For the usage of mbed, please refer to the related books and Web sites. URL of the mbed homepage is, http://mbed.org/
The sample program is available from the following link published by the author,

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

Download the sample program by a zip format, then you can import it your mbed workspace. Compile the sample program and download the binary object to the flash memory in mbed. How to work the robotracer is follows,

1) After pressing the reset button on the mbed board, the sample program is running.

2) Select the operation mode by pressing the set button. The operation mode is indicated by LED's on mbed in binary code.

3) Press start button, then the robotracer operates selected mode.

About the operation modes:

0: Check the sensor level, which is indicated LED's and sending by serial communication port.

1: Run forward for 500 steps of stepper motor.

2: Turn right for 500 steps.

3 to 7: Run following on line.

To use the serial communication port, the device driver should be installed on PC as shown in the following URL,

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

In the line following, mode 3 is the lowest speed, and the speed increases with mode number. However, the stepper motor is more likely to pull-out Increasing with the speed. So that you must chose suitable speed for running. The robotracer recognizes start and goal position by start/goal markers indicated at the side of course line. The robotracer stops automatically at the goal position.

In the competition, the configuration of the course is recorded in the memory for the first run, then, the robotracer runs faster using the course records in order to shorten the time records at the second run.

The sample program is described as follows,

The definition of the various variables, ticker instances (timer interrupt), buses, digital I / O and analog inputs are done at first part of the program. The LED's are available for binary display by giving a value to the "leds" instance, which is bus definition. To rotate the stepper motor, sequentially gives excitation pattern to the bus defined as "motorR" and "motorL".
Three analog ports are assigned to parallel connected phototransistors, which are used for line sensor. For example, "AnalogIn pt12 (p19)" means the analog port of p19 is used to detect analog value of parallel connected phototransistor of PT1 and PT2. In this case, the analog port can be read the signal from the either phototransistor corresponding LED light emission, which are defined "DigitalOut" instance of "ledCout(p9)" and "ledSGout(p10)". (For example, "PT1" is detected when "ledSGout" is active.)

Duration of the called functions of timer interrupts are defined in the main function at the last part, such as "timerS.attach_us(&Sensor, 2000)" and "timerM.attach_us(&stepMotor, 400)". The “Sensor” and “stepMoter” functions for control of the sensor and stepper motors are called every 2000 and 400 micro seconds, respectively.

In the "Sensor" function, variable of "timS" are varied alternately 1 and 0, and the corresponding phototransistor for line sensor is detected alternately. The start and goal markers are also detected alternately. The detected analog value is subtract for the value when the LED light is on and off to eliminate the background noise. Threshold to determine the presence or absence of the line is the “STH” value defined at the first part of the program. The variable named “sensD” is stored with the digital value of each bit corresponding to the each optical sensor in line sensor. The position of a line is changed into the values from "-5" to "5" using pre-defined array of "sensArray", and the value is stored to "sensDout". The buzzer control is also done in this function.

In the “stepMotor” function, the excitation pattern of the stepper motor is controlled, using “timR” and “timL” as a counter. "modeR" and "modeL" indicate the state of the direction of rotation, stop and free of the right and left motors, respectively. The excitation pattern is defined as "RMOTOR" and "LMOTOR" arrays, and the “patR”and “patL”are the indexes for the arrays.

In the main function, after initialization of variables, the operation mode can be select by the set switch, and the execution of the selected mode is done by the start switch. “runTurn” function is to be run by specifying the direction of rotation of the left and right motor, rotation speed, and the motor steps.

The “run” function controls the running for line following. The argument n and m are the factor of reference speed and speed control, respectively. In the function, the robot run at half of reference speed using “runTurn” function initially, then, the robot run at reference speed with speed difference controlled by the position on the line. If the center of robot shifts from the line, one side of the motor becomes slower for return to center. For example, when the robot is running right of the line, the left motor is reduced.

During running for line following, the start/goal and corner markers are detected and generate a beep. Passage of start marker, detection the intersection, passage of goal marker and stop enough distance apart from goal position are recognized using the state variable of "frun". To avoid the misread of a marker and intersection, the determination of the presence of the marker is carried out by the integrated value of the sensor after passed the distance of line width. The threshold of the integrated value of the corner and start/goal markers are the "CTH" and "SGTH" defined at initial part of the program, respectively.

To record the course for the first run, it is better to do each time with the corner marker detection. For the second run in high-speed, it is good to use the course parameters to determine the control parameters and speed, such as the distance between markers, curvature, angle, etc.

Finally, the sample program of the mbed robotracer for education is shown in appendix 1.

Appendix 1

```
//**************************************************************************
//
//      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};     // excitation pattern for left motor
const unsigned char LMOTOR[]={0x03, 0x06, 0x0C, 0x09, 0x00};     // excitation 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,
0,0,0,0,0,0,0,0,-4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

unsigned char pmode=0;                                                        // operation mode
volatile float pt1B, pt2B, pt3B, pt4B, pt5B, pt6B, ptSGB, ptCB;              // background sensor values
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);                   // eliminate background
        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--;
    // detection of intersection
    if (markerSG>1 && markerC>1) markerX=1;                        // both marker
    if (markerX==1 && markerSG==0 && markerC==0) markerX=0;     // ignore intersection

    // 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    1: forward    2: reverse    3: 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];                               // excitation pattern output to right motor
      motorL= LMOTOR[patL];                               // excitation 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 intersection
            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;            // run for line following
            case  4: run(8,24);   break;
            case  5: run(6,18);   break;
            case  6: run(5,15);   break;
            case  7: run(4,12);   break;
        }
    }
}
```