```
/*******************************************************************************
**
Program for Advance Line following robot using ATmega32 mcu
Crystal Frequency - No external crystal used 1Mhz internal
Low Fuse Bit 0xE1
High fuse Bit 0x99
Compiler optimization --- o0
Circuit diagram and tutorial:-PlaywithRobots.com/make-it-form-scratch/advance-line-
follower-robot
NOTE:
instead of trying complete code at once, first test small functions like forward()
,left(),right(),line_follow(),step(),left_ninety() etc if they work code will work.
General logic:
Step function will make robot to go from one intersection to other intersection in
forward direction.
Left_ninety function will rotate the robot left till robot is again allinged on line
on an intersection

FEEL FREE TO CHANGE AND MODIFY!

Written by Abhishek
www.PlaywithRobots.com
*******************************************************************************
***/
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#define S1 (PINA&0x01)          // sensors
#define S2 (PINA&0x02)
#define S3 (PINA&0x04)
#define S4 (PINA&0x08)
#define FS (PINB&0x01)
#define N 1
#define S 2                     // directions
#define E 3
#define W 4
int bot_dir;                    // variable to store robot initial direction
int bot_x,bot_y;                 // variable to store robot initial cordinate
int grid_x , grid_y;            //  variable to store grid size
int block=0;                    // variable to store if block is detected or not

void forward(void)
{       PORTD&=0xF0;            // Forward function to move robot straing forward
        PORTD|=0x05;    }

void left(void)
{       PORTD&=0xF0;            // Left function to make robot turns left
        PORTD|=0x01;    }

void right(void)
{       PORTD&=0xF0;            // Right function to make robot turns right
        PORTD|=0x04;    }

void reverse(void)
{       PORTD&=0xF0;            // Reverse function to make robot move backward
        PORTD|=0x0A;    }

void stop(void)                 // Stop function to stop motion
{       PORTD&=0xF0;    }

void left_ninety(void)          // this function will make robot to rotate left on
the intersection
{ forward();                    // small delay to move robot forward before rotating
  _delay_ms(300);
  stop();
  PORTD&=0xF0;          // these values are for rotating robot left differentially
  PORTD|=0x09;
  _delay_ms(380);
  left();
  _delay_ms(100);
  while(S1==0);                         // while its sensors are not alligned rotate
  stop();               // stop after sensor allignment
  if(bot_dir==N)
        { bot_dir=W; return;}
  if(bot_dir==W)                        // change direction variable
        { bot_dir=S; return;}
  if(bot_dir==S)
        { bot_dir=E; return;}
  if(bot_dir==E)
        { bot_dir=N; return;}


}
void right_ninety(void) // this function will make robot to rotate right on the
intersection
{
forward();              // small delay to move robot forward before rotating
_delay_ms(300);
PORTD&=0xF0;
PORTD|=0x06;            // these values are for rotating robot right differentially
_delay_ms(380);
right();
_delay_ms(100);
while(S4==0)                            // while its sensors are not alligned rotate
        ;

stop();                 // stop after sensor allignment
if(bot_dir==N)
        { bot_dir=E; return;}
if(bot_dir==E)
        { bot_dir=S; return;}           // change direction variable
if(bot_dir==S)
        { bot_dir=W; return;}
if(bot_dir==W)
        { bot_dir=N; return;}
}


void line_follow(void)
{ if((~S2)&&(~S3))
forward();              // function for linefollowing
if((~S2)&&(S3))
left();
if((S2)&&(~S3))
right();
}

char step(void)
{ forward();            // small delay to avoid sensor fluctuation
_delay_ms(70);
while(S1||S4)
        {
        line_follow();
        if(FS)
                {block=1;
                        stop();         // follow line till intersection if block
detected return change block variable and exit
                return 1; }
        }
forward();
```

```
_delay_ms(70);
while(1)
        {if((S1==1)||(S4==1))           // move forward till the intersection is
crossed
        break;
        }
stop();                                 // stop just after intersection
if(bot_dir==N)
        bot_y++;
if(bot_dir==S)
        bot_y--;
if(bot_dir==E)                          // change robot corrdinate variable
        bot_x++;
if(bot_dir==W)
        bot_x--;
return 0;
}

void search(void)               // make the robot moves as shown in video in search
of block http://www.youtube.com/watch?v=GqAMgG-gc0c
{ bot_y=0;
bot_x=0;
while(1) {
 while(bot_y<grid_y)
        { if(step())
                return;
        }
 if((bot_x==grid_x)&&(bot_y==0))
        return;
 right_ninety();
 if(step())
        return;
 right_ninety();

 while(bot_y>0)
        { if(step())
                return;
        }
 if((bot_x==grid_x)&&(bot_y==0))
        return;

 left_ninety();
if(step())
        return;
 left_ninety();
}
}
over_turn()
{
PORTD&=0xF0;
PORTD|=0x06;
_delay_ms(900);
right();                        // over turn and changes direction
_delay_ms(100);
while(S4==0)
        ;
stop();
if(bot_dir==N)
        { bot_dir=S; return;}
if(bot_dir==E)
        { bot_dir=W; return;}
if(bot_dir==S)
        { bot_dir=N; return;}
if(bot_dir==W)
        { bot_dir=E; return;}
```

```
}

reach_home()                    // function to take robot to initial cooridnate
{               // many more condition exits according to robots direction of
approching block, write them if you want!
                step();
                bot_y--;
                if(bot_x!=0)
                        { left_ninety();
                                while(bot_x)
                                        step();
                        }
                if(bot_x==0)
                        { if( bot_dir==N)
                                { over_turn();
                                while(bot_y)
                                        step();
                                return;
                                }
                          if ( bot_dir==W)
                                { left_ninety();
                                while(bot_y)
                                        step();
                                return;
                                }
                        }

}
void main (void)
{
DDRB=0x00;  // making PORTB as input
DDRA=0x00;  // making PORTA as input
DDRD=0xFF;  // making PORTD as output
PORTD=0x30;  // setting Logic high on Enable pin of L293D
grid_x=3;       // setting maximun X coordinate of grid
grid_y=3;       // setting maximum Y coordinate of grid
bot_dir=1;      // Initial robot direction , 1= North
_delay_ms(1000);  // a small delay before code starts

search();   // function to search for robot
over_turn();    // make over turn
reach_home();   // return to initial coordinate
// line_follow();
while(1); // stuck in a infinite loop after task is completed
}
```