NOSY

José António Barros Vieira
Professor Keith L. Dothy
Professor Scott Jantz

University of Aveiro
Electronics and Telecommunication Engineering Dept.
Campo Santiago
3800 Aveiro

July 21th, 1995

Autonomous Mobile Robots

1. Abstract

In this project I constructed an autonomous mobile robot, called **nosy**. It starts moving random avoiding obstacles and looking for a dark line on the ground. When nosy finds the line it follows it, if it sees an obstacle avoids it and starts looking for the line again. Nosy follows the line until it gets in a garage and stays there. In the garage it could, for example recharges the batteries. When it hears a loud sound as "nooossyyy" it gets out from the garage and runs for all space avoiding obstacles with an algorithm that adapts the velocity with the proximity of the obstacles. It stays in this stage for one minute, on the end it makes two beeps to alert the change of behavior, the end of this time could be when it feels that the batteries are low and needed to be recharge. Then it starts looking for the line again and begins all these behaviors.

Nosy has, basically four different behaviors.

I present some results and conclusions of all behaviors, and nosy's perception of "his" world.

2. Introduction

This work introduces us in to the world of autonomous mobile robots.

In this project nosy avoids obstacles adapting the velocity, follows a dark line, but if it sees an obstacle avoids it and looks for the dark line again. When it enters the garage it stops and stays there. When nosy hears a sound it turns around and comes out from the garage and begins again these four different behaviors:

1º avoids obstacles adapting the velocity, to do this nosy has 3 infra-red sensors in front of the platform.
2º look for the line random and avoids obstacles, to do this nosy has 3 infra-red sensors in front and 3 like a nose.
3º follows dark line on the ground, to do this nosy has 3 infra-red sensors in front like a nose.
4º stops and waits in the garage until it hears a loud sound, to do this nosy has 1 infra-red sensors that are pointed to the ceiling and a microphone (sound sensors) pointing up.

3. Executive Summary

The four different behaviours have some particularities that I'm going to describe:

In the first behaviour nosy moves avoiding obstacles and adapting the velocity . There are three different zones safe, threshold and danger zone, depending in what zone nosy is the velocity will be bigger or smaller.

In the second behaviour I had some difficulties to start the following of the line. If the line appears orthogonal with the directional of the movement, nosy has difficulties to follow the line. This difficulty gets bigger when the velocity is higher. To solve these problems I reduce the velocity and if it sees the line orthogonal it turns to the right for example.

The third behaviour is follow the line. I use three infra-red sensors in line and near to the ground. When the line has a 90° angle nosy some time gets lost to avoid it I use the old_nose information to make the correction of direction, see code in Appendix A.

Finally on the fourth behaviours the problem is to choose the ideal microphones thresholds because the sensor feels any noise. One other problem is when and how frequent should I sample the sound sensor.

4. Integrated System

The nosy will acts as describe in the next Figure 1 :

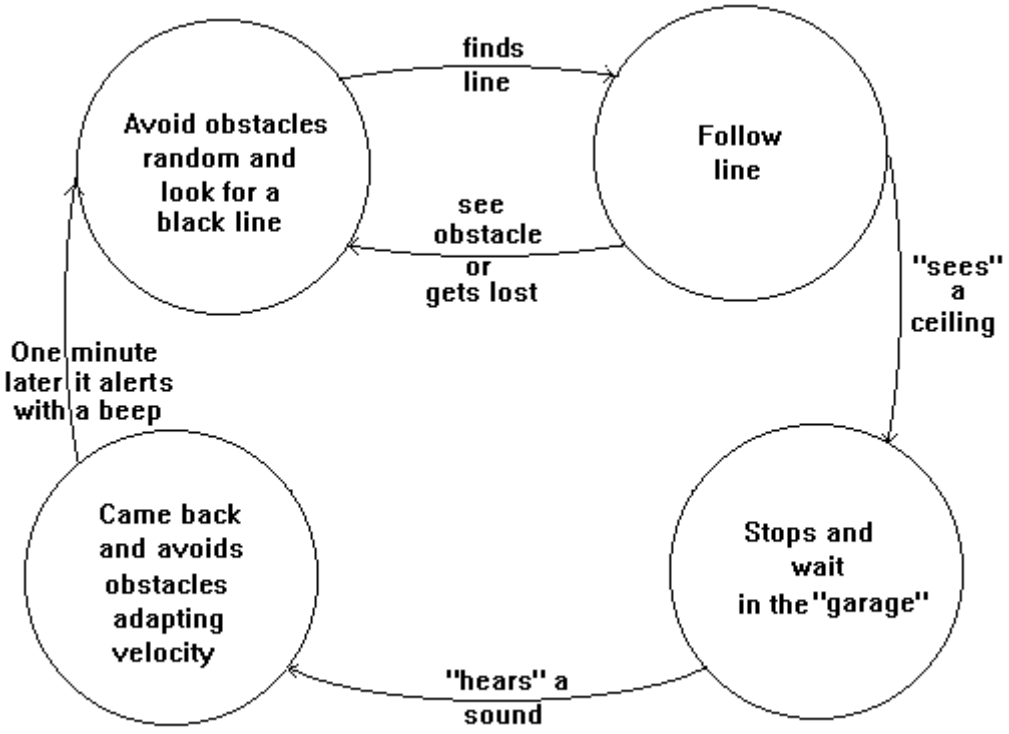

Figure 1 - Flow diagram.

The high level functions are describe in the Figure 2 as we can see there are four different behaviors depending on the exterior perception of the nosy by sensors.

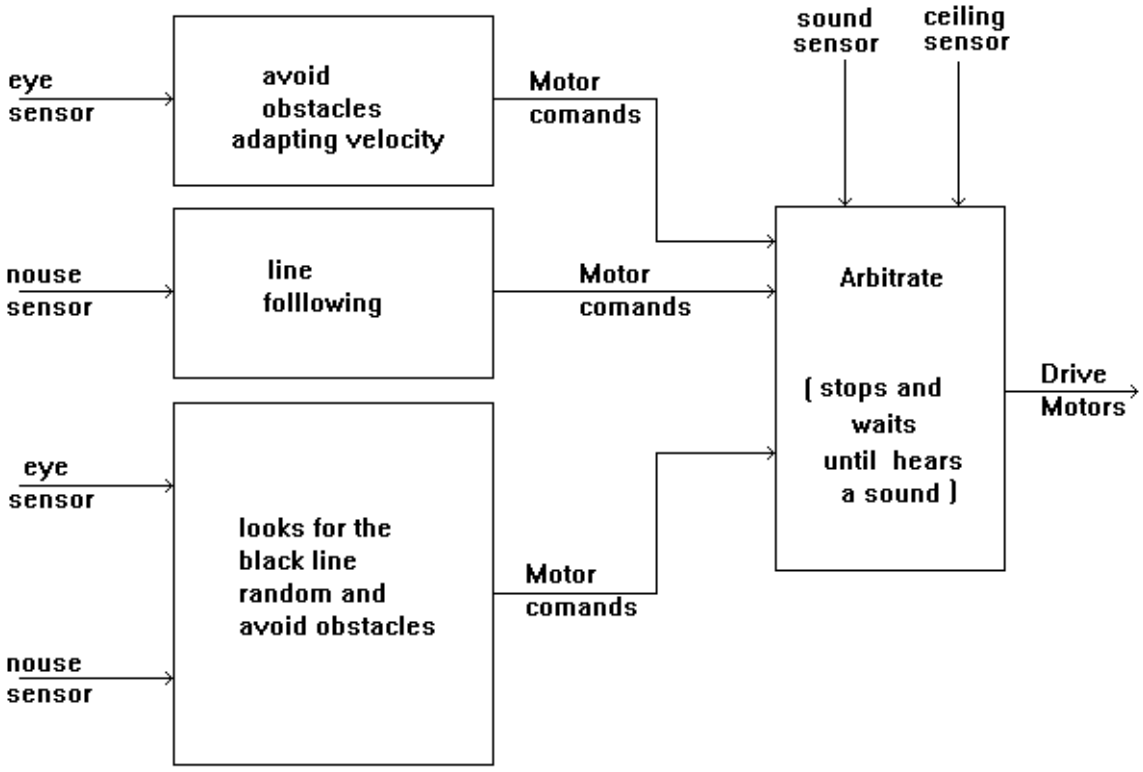

Figure 2 - Graphical representation of behavior program.

The code of this graphical representation is in the Appendix A "nosy9.c".

5. Mobile platform

Nosy platform has 7 infra-red sensors, 3 eyes, 3 noses and 1 ceiling detector, a microphone (sound sensor) and a piezo buzzer.

The platform is circular that will facilitate the obstacles avoidance. Nosy has two wheels with two dc motors to control the movements and one other directional wheel. It has eight batteries of 1.2 volt and of course the EVBU board with some circuitry see Appendix B.

Nosy has a "nose" with three infra-red sensors near to the ground that are protected with a wire to crush with obstacles.

6. Actuation

Nosy has I have described has four different types of actuation.

I choose one minute to change from the avoid behaviour to the looking for the line behaviour because if I wait until the batteries are low it would be a long time. To detect the low charge of the batteries I could use an analog(-) entry to do these:

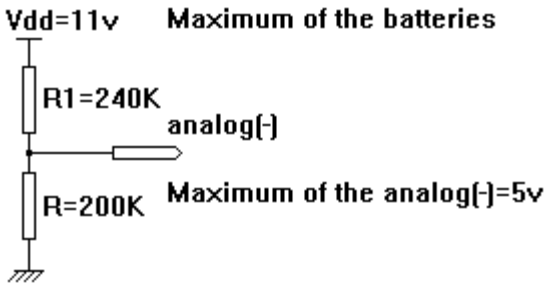

Figure 3 - Charge batteries detection.

If Vdd = 8v is the charge low, the analog(-) will have 3.64v and analog(-) would be the value of 185.

If analog(-) < 185

look_for_line_to_recharge();

I have a constant of compensation to the wheels because one wheel is slower ±15% than the other.

The sensors of the nose are in line and very directional to " smell " well the line on the ground. When it follows the line some times it makes small spikes, I think the reason is some reflections that the line has.

In the microphone I put high thresholds because it hears any sound in the room so, like these with a loud sound and while at least two seconds, nosy certainly responds.

7. Sensors

In the Table 1 is show the type of sensors, their functions, localization in the platform, the quantity.

Nosy's Sensors Suite

Sensor type	Function	Location	# Number
Infra-red IR	Proximity	In front	3
Infra-red IR	Line detection	In front nose	3
Infra-red IR	Ceiling detect.	Up in platform	1
Microphone	Sound detect.	Up in platform	1
Piezo buzzer	Change behav.	In platform	1

Table 1 - Sensors characteristics

7.1. Infra-red sensors

The infra-red sensors are a good help to avoid obstacles but we should control very well the thresholds because they have different reflections of white or black obstacles.

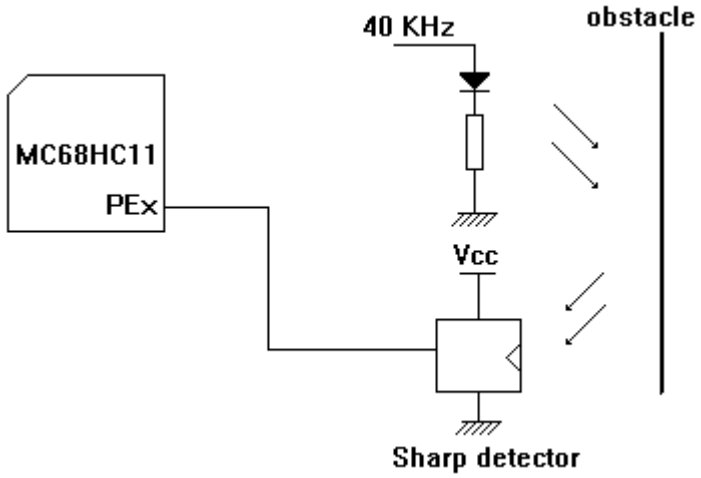

Figure 4 - Schematic of the emission and receiving infra-red sensors.

This difference is useful to me because I use precisely the infra-red to detect a black line in the ground.

The emission of the infra-red in the nose is made separately for each one because they are too close and they interfere on the neighbours.

7.2. Sound sensor

In my project I just use the microphone to call my robot but they are also use, for example to follow a specific sound in a room.

The schematic of the receiving and amplify the sound signal is in figure 5:

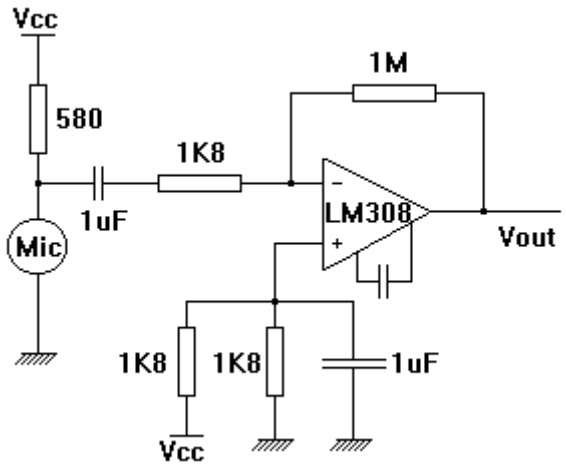

Figure 5 - A microphone with simple amplifier uses a LM308 op-amp.

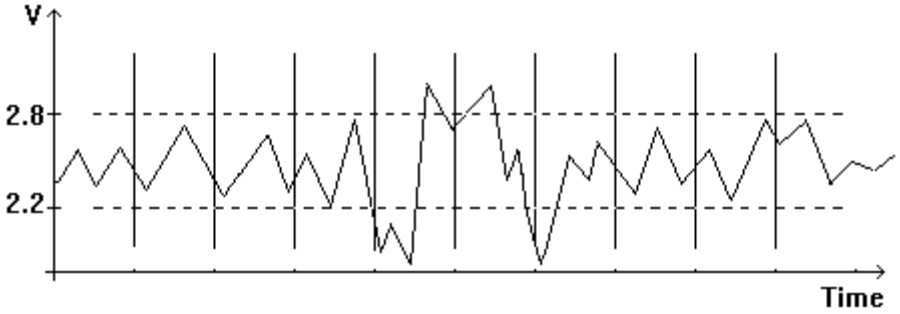

Figure 6 - Output signal from microphone.

The nosy is to take some action when it detects a loud sound, that is, when the signal from the microphone goes above the upper dashed line or below the lower dashed line. Each vertical bar represents a sample. Unless samples are taken at very frequent intervals, a sound of interest can easily be missed [7]. If the loud sound had at least two seconds nosy will respond to it.

7.3. Piezo buzzer

I use a piezo buzzer just to make the alert that the "batteries are low" and then nosy should look for line to recharge. The sound emitted are two beeps of one second.

The piezo buzzer isn't really a sensor, it's more an emitter.

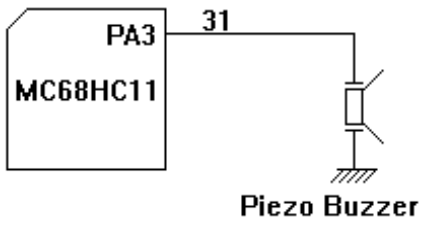

Figure 7 - Connection of the piezo buzzer to the MC68HC11 board [7].

8. Behaviours and Results

Nosy has four different behaviors three of then are simple processes and the fourth is made in the arbitrate process. See the code in Appendix A.

The first behavior is avoid obstacles with adapting speed. Here I have three different zones save, threshold and danger zone. I have a lot of functions that I call depending where nosy is and what sensor is closer to the obstacle. In the find line behavior there are some problems to find the line when it sees the line near or really orthogonal with his direction of movement and with relatively high speed it almost never see the line. In the following behavior the problems appear with high speed of following and angles in the line of 90°, to try to solve these I use the old nose information to correct the movements but it isn't really enough. Other problems are the spikes that it makes when following the line, those could be because of the algorithm or some reflections and irregularities that the line has. These spikes some time helps nosy readjusting the right trajectory.

When nosy is following the line if it sees an obstacle it avoid with a simple algorithm and look for the line random again.

When nosy gets in the garage it stops and 300 ms after it keeps reading the sound sensor, this time is to guarantee that it doesn't hear the noise of the wheels. If it hears a loud sound it gets out and starts these behaviors again.

The results I present here are some experimental numbers that I get in an area as show in Figure 8.

Figure 8 - Experimental area.

Nº of times that nosy cross the line without get it.	Nº of times that nosy lose the line.	Nº of times that nosy crash in an obstacle.	Nº of times that I call nosy to get out from the garage.
1	1	1	1
0	0	0	1
0	0	1	1
2	0	0	1
0	0	0	2
2	0	0	1
0	0	0	1
1	0	0	1
0	0	0	1
0	0	0	1
0	1	0	1
0	1	0	1
2	0	1	1

Table 2 - Some experimental results.

From the analise of the results, I can say that, with the speeds that I experiment nosy almost never lose the line and it always respond when I call him. I can say that in 50% of times that it crosses the line nosy gets it.

I tested the same behaviors with a higher speed and the results are a little worst.

9. Conclusion

Nosy basically avoid obstacles with an adaptative algorithm and look random for a dark line and follow it until it gets in the garage to "recharge". When it hears a sound nosy comes out and starts these behaviors again.

The behavior of finding the line should be improved because if the line appears orthogonal with the direction of the movement it almost doesn't see the line.

The behavior of following the line should be improved because if I increment the speed it gets lost a lot of times, and I have problems too when the line has a 90° angle.

If I start the project now I would think in a more practical idea and I would make the algorithm more independent, without fixed thresholds and with good calibration of the

sensors. For example in the following algorithm I would do something like, when nosy feels no curves it increments the speed. I would experiment some different positions of the nose sensors that I think is very important.

The conclusion that I take in this first contact with machine perception is that adapting threshold, interference of different infra-red and calibration are very important to the world of the mobile robots.

10. Documentation

[1] Tracy L. Anderson and Max Donath, "Animal Behavior as a Paradigm for Developing Robot Autonomy", edited by Pattie Maes, MIT/Elsevier; pp 145-168.

[2] Randall D. Beer, Hillel J. Chiel, and Leon S. Sterling, "A Biological Perspective on Autonomous Agent Design", edited by Pattie Maes, MIT/Elsevier; pp 169-186.

[3] R. A. Brooks, "Elephants don't play chess" - extracted from "Designing Autonomous Agents", edited by Pattie Maes, MIT/Elsevier, pp3-15.

[4] John J, D'Azzo, Constantine H. Houpis, "Linear Control System Analysis and Design Conventional and Modern", McGraw-Hill International Editions, 3rd edition, pp 21, 505-541.

[5] Keith L. Doty and Akram Bou-Ghannam, "Controlling Situated Agent Behaviors with Consistent World Modeling and Reasoning".

[6] Keith L. Doty and Steven Louis Seed, "Autonomous Agent Map Construction in Unknown Enclosed Environments", MLC-COLT'94 Robot Learning Workshop, Rutgers, New Brunswick, N.J.

[7] Joseph L. Jones, Anita M. Flynn, "Mobile Robots Inspiration to Implementation", A.K.Peters, Ltd.

[8] K.S. Fu, R.C. Gonzalez, C.S.G. Lee, "Robotics Control, Sensing, Vision, and Intelligence", McGraw-Hill International Editions, pp. 267-293.

[9] Richard D. Klafter, Thomas A. Chmielewski, Michael Negin, "Robotic Enginnering An Integrated Approach", Prentice Hall, pp. 314-508, pp. 665-691.

11. Appendices

11.1. Appendix A

```
/*      ================== nosy9.c ========================      */


/*      File name:    nosy9.c
        Programmer:   Jose Vieira
        Date:         July 20, 1995      */

/*      Globals         */


/*       IR             */

int     l_motor = 0,          /* motor id:  0 -> left, 1 -> right */
        r_motor = 1,

        l_speed = 0,          /* current motor speed */
        r_speed = 0,

        new_l_speed = 0,      /* desired motor speed */
        new_r_speed = 0;

int     s1 = 7,              /* motor speeds */
        s2 = 14,
        s3 = 21,
        s4 = 28,
        s5 = 35,
        s6 = 42,
        s7 = 49,
        s8 = 56,
        s9 = 63,
        s10= 70,
        s11= 77,
        s12= 84,
        s13= 91,
        s14= 100,

        direction_fb = 1,     /* direction of the robot in forward-backward
                              (1 -> forward, 0 -> stop, -1 -> backward) */
        direction_lr = 0,     /* direction of the robot in left-right
                              (-1 -> left, 0 -> straight, 1 -> right) */

        turn_time = 0,        /* time to turn about 45 degrees */
        motor_update_interval = 50;     /* time between motor speed
                                                        update */

int     l_eye ,
        r_eye ,
        c_eye ,
        eye_bra = 0,             /* ir sharp sensors initial values */
```

```
        eye_bla = 0,             /* (r -> right, l -> left, c -> center, */
        eye_c = c_eye=0,               /*  a -> angle,b -> back) */
        eye_ra = r_eye=0,
        eye_la = l_eye=0;

/*************************************************************************/
/* sensor thresholds                                                     */
/*************************************************************************/

int     front_threshold = 110,     /* threshold boundaries */
        right_threshold = 105,
        left_threshold = 105,

        front_danger = 120,       /* danger boundaries */
        right_danger = 120,
        left_danger = 120,

        front_safe = 95,          /* safe boundaries */
        right_safe = 90,
        left_safe = 90,

        open_threshold = 10;      /* differential front sensor reading
                                     required to stop turning when in a
                                                          trap */

int     eye_threshold=100,
        eye_threshold_1=115,    /*   obtacle avoid when finding the line */
        nose_threshold=125;

int     danger_threshold=124;

int     l_nose = 0,            /*       Variables to all sensors         */
        r_nose = 0,
        c_nose = 0,
        old_l_nose = 0,
        old_r_nose = 0,
        old_c_nose = 0,
        sound = 0,
        ceiling_d = 0;

int     min_th_sound = 122;     /*      Thresholds to the sound sensor  */
int     max_th_sound = 138;

int     ir_on_interval=50;
int     ir_off_interval=10;
float   fac_corr = 0.873;       /* Motor compensation   */
float   find_const = 0.25;      /* Reduction of speed when finding the line*/

float   line_follow_left = 0.0,
        line_follow_right = 0.0;

float   find_avoid_left = 0.0,
        find_avoid_right = 0.0;
```

```
/*      ====================================================         */

void wait(int m_second)
{
  long stop_time;
  stop_time = mseconds() + (long)m_second;
  while(stop_time > mseconds())
  defer();
}
/*      ====================================================       */

void sensor_module()
{
while(1)
        {
/*      ================== EYES AND CEILING ==================        */

        poke(0x4000,0b001000111);  /* setup front IR sensor array */
        wait(ir_on_interval);      /* wait for transients to die */
        eye_c=c_eye = analog(0);         /* make analog reading */
        eye_la=l_eye = analog(1);         /* make analog reading */
        eye_ra=r_eye = analog(2);         /* make analog reading */

        ceiling_d = analog(6);     /* make analog reading */
        poke(0x4000,0);            /* turn off */
        wait(ir_off_interval);

/*      ======================== NOSE ========================           */

        poke(0x4000,8);           /* setup front IR sensor array */
        wait(ir_on_interval);     /* wait for transients to die */
        c_nose = analog(3);       /* make analog reading */
        poke(0x4000,0);           /* turn off */

        wait(ir_off_interval);    /* wait for interference to die */

        poke(0x4000,16);          /* setup right angle IR sensor */
        wait(ir_on_interval);     /* wait for transients to die */
        l_nose = analog(4);       /* make analog reading */
        poke(0x4000,0);           /* turn off */

        wait(ir_off_interval);    /* wait for interference to die */

        poke(0x4000,32);          /* setup left angle IR sensor */
        wait(ir_on_interval);     /* wait for transients to die */
        r_nose = analog(5);       /* make analog reading */
        poke(0x4000,0);           /* turn off */

        wait(ir_off_interval);    /* wait for interference to die */

        }
}
```

```
/*      ====================================================          */

void motor_module()
{
while(1)
    {
    if (new_l_speed != l_speed)           /* increment left motor speed if
                                        necessary */
    {
    if ((new_l_speed * l_speed) < 0)    /* if change of direction is
                                      required then stop6 the motor */
            l_speed = 0;
    else
    if (new_l_speed > (l_speed + 7))  /* increase the speed of the robot
                                     gradually (max 7) to the desired
                                             speed */
           l_speed += 7;
           else
           l_speed = new_l_speed;          /* if the speed increas is less
                                     than 8 or if the desired speed is
                                       less, then set the current speed
                                            to the desired speed */
      }
    if (new_r_speed != r_speed)           /* increment right motor speed if
                                                        necessary */
         {
         if ((new_r_speed * r_speed) < 0)    /* if change of direction in
                                     required then stop the motor */
                r_speed = 0;
              else
        if (new_r_speed > (r_speed + 7))  /* increase the speed of the robot
                                        gradually (max 7) to the desired
                                                        speed */
                             r_speed += 7;
                    else
               r_speed = new_r_speed;    /* if the speed increas is less
                                      than 8 or if the desired speed is
                                     less, then set the current speed
                                             to the desired speed */
             }
    wait(motor_update_interval);          /* motor speeds are updated every
                                     motor_update_interval miliseconds */
       }
}

/*      ====================================================           */



void obstacle_avoidance_module()
{
int temp;

stop();
```

```
while(1)
        {
        if (eye_c >= front_threshold)         /* sense something in the
                                                     front ? */
                {
                temp = eye_c;                   /* temporary value of the
                                                   front sensor reading */
                if ((eye_ra >= right_threshold)   /* turn toward the open */
                        && (eye_ra > eye_la))
                        turn_right_slow();
                else
                        turn_left_slow();
                while (eye_c > (temp - open_threshold)) /* turn until the front
                                                   sensor reading falls by
                                                     the open_threshold */
                        defer();
                }
        else if ((eye_ra >= right_threshold)
                                    && (eye_ra > eye_la))
                                /* sense something to the left? */
                if (eye_ra > (eye_la + 5))        /* if the difference is
                                                   large, then arc more
                                                           sharply */
                        arc_right();
                else
                        slight_arc_right();
        else if (eye_la >= left_threshold)       /* sense something to the
                                                          right? */
                if (eye_la > (eye_ra + 5))         /* if the difference is
                                                   large, then arc more
                                                           sharply */
                        arc_left();
                else
                        slight_arc_left();
        else if ((eye_c >= front_safe) && (eye_ra >= right_safe)
                        && (eye_la >= left_safe))
                                        /* if all sensors are in
                                         a safe state, then ahead at
                                         full speed, else ahead at
                                         slow speed. */
                ahead_slow();
        else
                ahead_full();
        wait(50);                           /* update motor speed every
                                             50 miliseconds */
        }
}

/*      ====================================================           */

/* movement control functions */


void slight_arc_right()
```

```
        {
        new_l_speed = s7;
        new_r_speed = s3;
        }

void slight_arc_left()
        {
        new_l_speed = s3;
        new_r_speed = s7;
        }

void arc_right()
        {
        new_l_speed = s7;
        new_r_speed = s2;
        }

void arc_left()
        {
        new_l_speed = s2;
        new_r_speed = s7;
        }

void sharp_arc_right()
        {
        new_l_speed = s7;
        new_r_speed = 0;
        }

void sharp_arc_left()
        {
        new_l_speed = 0;
        new_r_speed = s7;
        }

void slow_stop()
        {
        new_l_speed = 0;
        new_r_speed = 0;
        }

void ahead_full()
        {
        new_l_speed = s14;
        new_r_speed = s14;
        }

void ahead_half()
        {
        new_l_speed = s7;
        new_r_speed = s7;
        }

void ahead_slow()
```

```
        {
        new_l_speed = s2;
        new_r_speed = s2;
        }

void turn_right()
        {
        new_l_speed = s3;
        new_r_speed = -s3;
        }

void turn_left()
        {
        new_l_speed = -s3;
        new_r_speed = s3;
        }

void turn_right_slow()
        {
        new_l_speed = s1 + 3;
        new_r_speed = -s2;
        }

void turn_left_slow()
        {
        if (l_speed > 0 || r_speed < 0)
        new_l_speed = -s2;
        new_r_speed = s1 + 3;
        }

void slight_back_arc_right()
        {
        new_l_speed = -s7;
        new_r_speed = -s2;
        }

void slight_back_arc_left()
        {
        new_l_speed = -s2;
        new_r_speed = -s7;
        }

void back_arc_right()
        {
        new_l_speed = -s7;
        new_r_speed = 0;
        }

void back_arc_left()
        {
        new_l_speed = 0;
        new_r_speed = -s7;
        }
```

```
void back_sharp_arc_right()
       {
       new_l_speed = -s7;
       new_r_speed = s1;
       }

void back_sharp_arc_left()
       {
       new_l_speed = s1;
       new_r_speed = -s7;
       }

void back_full()
       {
       new_l_speed = -s7;
       new_r_speed = -s7;
       }

void back_half()
       {
       new_l_speed = -s4;
       new_r_speed = -s4;
       }

void back_slow()
       {
       new_l_speed = -s1;
       new_r_speed = -s1;
       }


/*      ====================================================            */


void line_follow_behavior()
{
  while(1)
  {
    if ( l_nose < c_nose && l_nose < r_nose)
      {
       line_follow_left = 0.0;
       line_follow_right = 12.0;
      }
    else if ( r_nose < c_nose && r_nose < l_nose )
         {
          line_follow_left = 12.0;
          line_follow_right = 0.0;
         }
          else
           {
            line_follow_left = 12.0;
            line_follow_right = 12.0;
            }
    if ( r_nose > nose_threshold && l_nose > nose_threshold
```

```
    && c_nose > nose_threshold )
    {
     if ( old_l_nose < old_r_nose )
         {
          line_follow_left = 0.0;
          line_follow_right = 15.0;
          wait(30);
         }
          else
           {
            line_follow_left = 15.0;
            line_follow_right = 0.0;
            wait(30);
            }
         }

    old_l_nose = l_nose;
    old_r_nose = r_nose;
    old_c_nose = c_nose;

  }
}

/*      ====================================================            */

void find_avoid_behavior()
{
int time_rotation,
    dir_rotation;

  while(1)
  {
    time_rotation = (int)mseconds() & 0x0DFF;
    dir_rotation =  (int)mseconds() & 0x0001;

    if (( l_eye < eye_threshold) &&  (r_eye < eye_threshold))
     if ( c_eye < eye_threshold)
      {
      find_avoid_left = 60.0;
      find_avoid_right = 60.0;
      }
     else if ( c_eye < danger_threshold )
      {
      if ( dir_rotation == 1 )
        {
         find_avoid_left = -60.0;
         find_avoid_right = 60.0;
        }
        else
        {
         find_avoid_left = 60.0;
         find_avoid_right = -60.0;
        }
      }
```

```
      else
      {
      find_avoid_left = 0.0;
      find_avoid_right = 0.0;
      find_avoid_left = -20.0;
      find_avoid_right = -40.0;
      }
    else
     if (l_eye > eye_threshold)
     {
      find_avoid_left = 60.0;
      find_avoid_right = -60.0;
      wait(time_rotation);
     }
     else if (r_eye > eye_threshold)
       {
      find_avoid_left = -60.0;
      find_avoid_right = 60.0;
      wait(time_rotation);
       }
    if (( l_eye > danger_threshold) ||  (r_eye > danger_threshold) ||( c_eye >
danger_threshold) )
    {
      find_avoid_left = 0.0;
      find_avoid_right = 0.0;
      find_avoid_left = -40.0;
      find_avoid_right = -20.0;

    }

  }
}

/*      ====================================================          */

void turn_start()
{
int rand = (int)mseconds() & 0x0001;

if (rand == 1)    /* left */
       {
                /* turn back */
       motor(0,-30.0);
       motor(1,30.0);
       wait(1500);
                 /* start */
       motor(0,50.0);
       motor(1,50.0);
       wait(1500);
       }
else
       {
              /* turn back */
       motor(1,-30.0);
```

```
        motor(0,30.0);
        wait(1500);
                  /* start */
        motor(0,50.0);
        motor(1,50.0);
        wait(1500);
        }
}
/*      ====================================================        */

void behavior_arbitrate()
{
int  i=0,
     j=0,
     k=1;
 while(1)
  {
    if (ceiling_d > 100)
    {
    motor(0,0.0);
    motor(1,0.0);
    wait(300);
    sound = analog(7);
        if  (sound < min_th_sound ||  sound > max_th_sound)
            {
            k=0;
            turn_start();
            }
            else
            {
             motor(0,0.0);
             motor(1,0.0);
            }
    }
   else
   {
        if  ( k==1 )  {

        if( ( r_nose < nose_threshold || l_nose < nose_threshold
        || c_nose < nose_threshold ) && ( l_eye < eye_threshold_1)
        && ( r_eye < eye_threshold_1) && (c_eye < eye_threshold_1) )
        {
           motor(0,line_follow_left);  /*  function of follow line  */
           motor(1,fac_corr*line_follow_right);
        }
        else
        {
         motor(0,find_const*find_avoid_left);
         motor(1,find_const*fac_corr*find_avoid_right);
        }
        }
        else
        {
        if (k == 0)
```

```
        {
        motor(1,(float)l_speed);
        motor(0,fac_corr*(float)r_speed);
        i=i+1;
        if (i >= 2000)
        {
        k=1;
        tone(500.,1.0);
        tone(500.,1.0);        /* Time to avoid random +- 60 seconds with 3000*/
        i=0;
        }
        else k=0;
       }
       }

    }
  }
}
/*      ======================================================         */

void main()
 {
   start_process(sensor_module());
   start_process(motor_module());
   start_process(obstacle_avoidance_module());
   start_process(find_avoid_behavior());
   start_process(line_follow_behavior());
   start_process(behavior_arbitrate());
 }
```

11.2. Appendix B

/* ========== Circuits ========== */