SPOT

by
Joel Beasley
12/4/95

EEL 5934
Professor Keith L. Doty

Table of Contents

Abstract

It is my objective to design Spot the robot to exhibit behaviors inspired by humans. To do this, Spot's actions and abilities need to be comparable to human actions and abilities. Table 1 below shows how Spot's capabilities compare to humans capabilities.

Human	Spot
Brain for memory and decisions	Expanded 68HC11 microprocessor for memory and decisions
Two legs and feet for motion	Two motors and wheels for motion
Two eyes for vision	Two infrared emitters/detectors for vision
Two pupils that sense light intensity	Two CdS cells to detect light intensity
Two hands to touch/feel objects	Two microswitch bump sensors to detect objects
Two ears for hear	Two microphones to detect sound
Mouth for communication	Piezo buzzer as an indicator
Inner ear for balance	Mercury switch to detect tilt
Food for energy	Rechargeable batteries for energy

Table 1. Human vs. Spot Abilities.

Introduction to Spot

To accomplish the objective Spot needs several things which are discussed in this paper. The first item that Spot requires is his Piggy Back Board which contains the electronics for his memory and sensors. Next, he needs an appropriate platform and method for moving himself. For this, Spot has a white, plastic octagon base with a clear, Plexiglas, hexadecagon top to guard against low hanging objects interfering with Spot's wires and components. Spot controls the movement of his platform with two motors and two wheels. To allow Spot to correctly control his platform he has several sensor groups -- infrared, cadmium sulfide, microswitch, microphone, and mercury. Spot also has the intelligence, provided to him by his

programming, to know when he is caught in situations or places that he does not want to be in.

68HC11 EVBU with Piggy Back Board (PBB)

Spot's Piggy Back Board interfaces with the Motorola 68HC11 using the P5 header of the EVBU. The 32k memory expansion, motor driver, and voltage regulation circuitry, designed by the Machine Intelligence Lab, are located on Spot's PBB. Along with this basic circuitry, Spot's PBB also contains some of the circuitry used by the 68HC11 to give Spot his life and personality. This circuitry consists of 2 LED indicators, 2 toggle switches, a piezo buzzer, a mercury switch, 8 A/D input plugs, four infrared emitter plugs, and various resistors.

The red and green LEDs are indicators as to what power mode Spot is in. The power mode is controlled by the red toggle switch, and the silver toggle switch controls Spot's program mode. Table 2 below summarizes Spot's LED indicators and Switch controls.

Requirements	Switch	Pointing to Spot's	Mode	LED(s) Lit
+5V regulated	red	left	standby	red only
+5V regulated	red	right	ready	red and green
+5V regulated, red to right	silver	left	download pcode	red and green
+5V regulated, red to right	silver	right	run IC	red and green

Table 2. Power and Program Modes.

In "standby" mode, Spot is only supplying +5V regulated power to his components which require backup power, but in "ready" mode he supplies power to all of his circuitry. To use PCBUG11 for setting up Spot's memory, he needs to be in "download pcode" mode. Once

the pcode has been successfully loaded to Spot's memory, he can be switched to "run IC" mode so Interactive C can be used for loading and executing his behavior programs.

The rest of the circuitry located on the PBB will be discussed with it's corresponding sensor description.

Plexiglas-Covered Octagon Platform

Spot's body is a 10" white plastic base with a clear .25" thick Plexiglas top secured about 2.5" above the base. This top protects Spot from overhanging objects that might try to interfere with his circuitry and wiring, and adds a great deal of rigidity to Spot's body. The base is cut in the shape of an octagon, but the top is an upgraded version of the original top, and it is cut in the shape of a hexadecagon (16 sides) (idea given to me by Daniell Sutton). These multisided shapes eliminate the protruding corners found on a square platform and allow spot to turn with ease and not have to worry about catching large corners on objects. Spot's platform is also unique in that his wheels are mounted an inch and a half inside the edge of the platform. By placing Spot's wheels on the inside of the platform he is able to maneuver more effectively by making very quick, sharp turns to avoid objects and out maneuver other robots.

Two Motor / Two Wheel Drive Movement

Spot's original wheels were two 2.75" Du-Bro model airplane wheels, but they have now been upgraded to 4" wagon wheels so that they extend below and above his platform. Spot's wheels are powered by two modified Futaba servo motors (suggested by Professor Doty). The servo motors are modified so that they will rotate 360° instead of just the 180° that

servos usually operate. This modification involved removing the electronics (circuit board and potentiometer) from the motor and cutting the stop off the drive gear, which essentially converted the servo into a gear-head motor. These Futaba motors at full speed provide Spot with about 45 wheel revolutions per minute and 42 oz/in of torque. By using two motors and two wheels, like a human's two legs and two feet, Spot is able to control his motion by simply reversing the speed of a wheel to turn in that direction. Also, by having just two wheels, Spot has a much quicker turning time and better maneuvering skills than a robot with four wheels.

Sensors

Two-Eyed Infrared (IR) Cross Vision

Spot uses his two Motorola IR emitters mounted on two Sharp IR detectors (model # GP1U5) to see objects so he can utilizes his quick maneuvering skills. The positioning of these two IR sensors is the important part of how Spot sees objects in front of him. One IR sensor is on the front left corner and the other is on the front right corner of the platform. However, instead of the IR sensors pointing straight forward, they are turned slightly inward giving Spot cross vision (a modified version of the Explorer Robot's Cross Eyes which uses four IR sensors). This cross vision allows Spot to see objects, even small ones, directly in front of him while, at the same time, see objects at his front sides since the IR sensors are only turned slightly. Remember, Spot only has two IR sensors, just as a human only has two eyes.

Two Microswitch Bumpers for Touch Detection

Spot is also built with two bumpers because his IR sensors sometimes are not able to see black objects and thin chair legs. His bumpers allow him to feel the object in stead of seeing

it. Spot's bumpers are each made up of a microswitch and a piece of coat hanger. The layout of

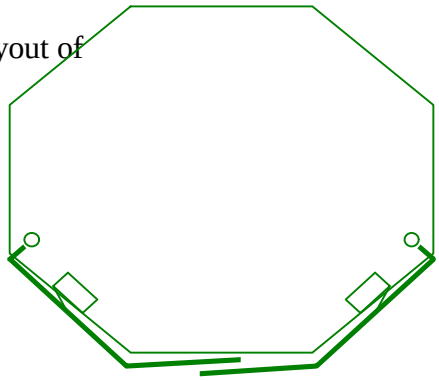

Figure 1. Bumper Layout.

the bumpers are shown to the left in Figure 1. When Spot runs into an object the coat hanger pieces will close the appropriate bump switch or switches if the object is directly in the center of Spot. The microswitches are wired with a resistor network so the bumpers only have to use one A/D port. Because of the resistor network, each switch or combination of switches causes a unique reading. Figure 2 below shows the resistor network used for the bumpers.

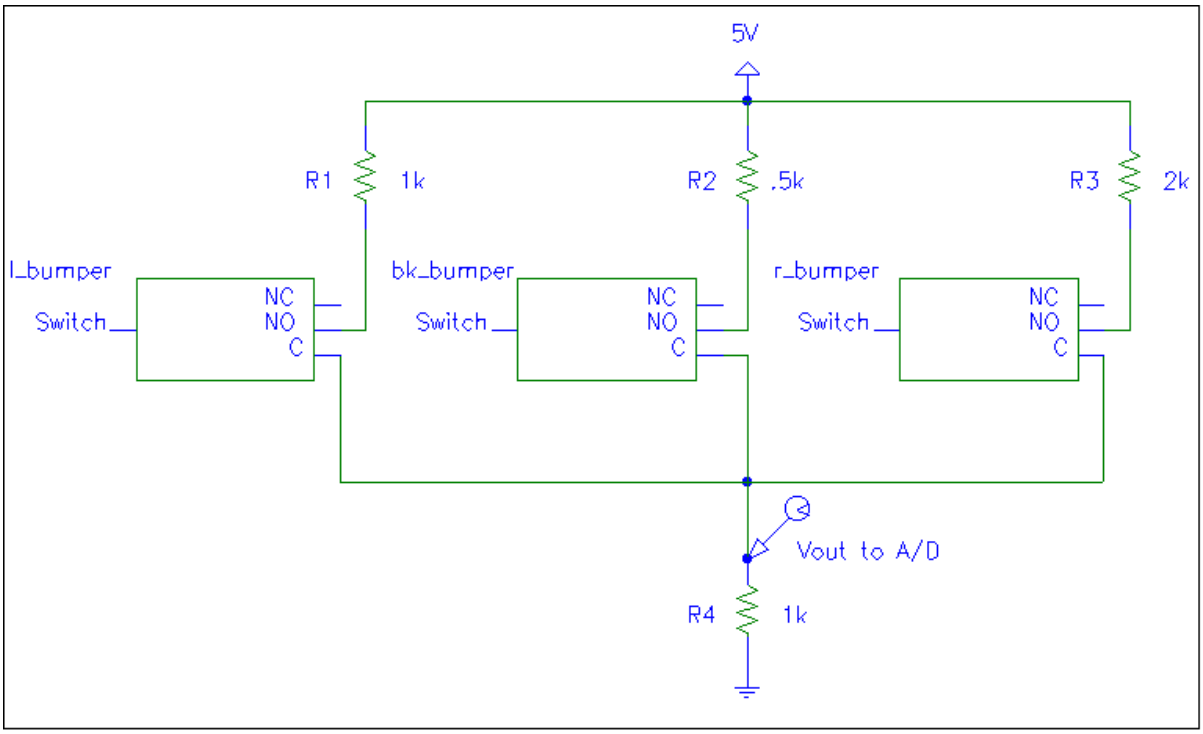

Figure 2. Bumper Switch and Resistor Circuit.

Figure 2 shows the circuit for three bump switches, but only two are used by Spot at this time. The third bump sensor is available if it is needed it in the future.

Internal Tachometers for Speed Detection

Spot's was going to be built with two tachometers internal to his motors so he could determine his speed accurately, however after they were tested with the A/D I determined that they were not going to provide much useful information because of their poor resolution. The design used is shown in Figure 3. This circuit was used to amplify the output of the tach motor and either add or subtract (depending on the direction of the wheels) the amplified value from 2.5 volts. This should have caused the A/D reading to vary, with the speed of the motors, around 2.5 V and not exceed the A/D boundaries. It did increase and decrease depending on the direction of the wheels, but the change was not linear and did not provide a stable reading for

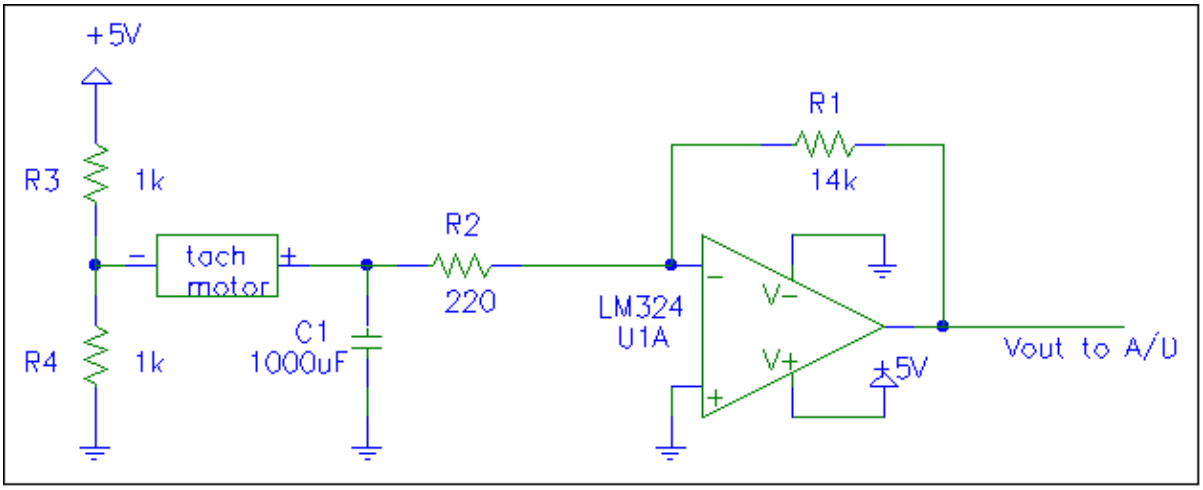

Figure 3. Tachometer Design.
(not implemented on Spot)

the A/D. Therefore the implementation of the internal tachometers was never completed and focus turned toward other sensors.

Two Microphone Array for Sound Detection

To hear sounds, Spot is built with two tiny condenser microphones. These microphones are located in a small project box from Radio Shack. They are each mounted in a piece of tubing that extends beyond the project box to provide some directionality to the microphones. The output of the microphones is amplified and read with two A/D ports. The amplification circuit is shown in Figure 5.

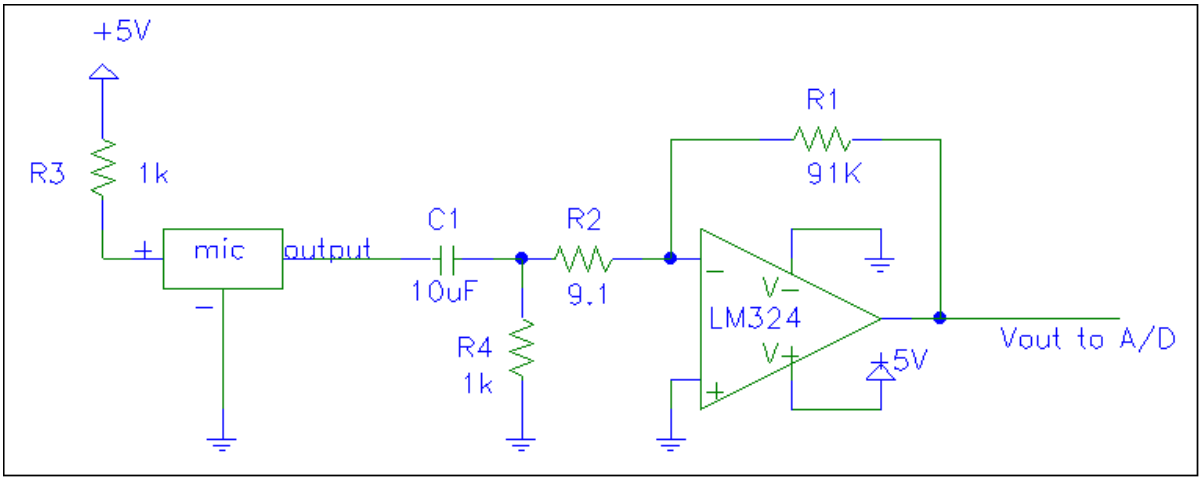

Figure 5. Microphone Amp Circuit.

This circuit takes the output from a microphone (about .4 mV for loud sound) and gives it a gain of about 10,000. Spot's two microphone sensors act as his ears so he can hear loud sounds and respond to them.

Two CdS Cells for Light Detection

In addition to responding to sound, Spot can also respond to light. For this capability, he has two Cadmium Sulfide (CdS) cell light sensors mounted in the front of the project box that contains his ears. With these CdS cells acting as the pupils of his eyes and the microphones acting as his ears, this project box acts as Spot's head. These CdS cells are wired in with

some resistors and voltage readings are taken with the A/D. These readings correspond to the intensity of light that Spot's CdS cells are exposed to. Spot can then use this knowledge to decision what to do. Figure 6 below show the circuit used for reading light intensity with CdS cells.

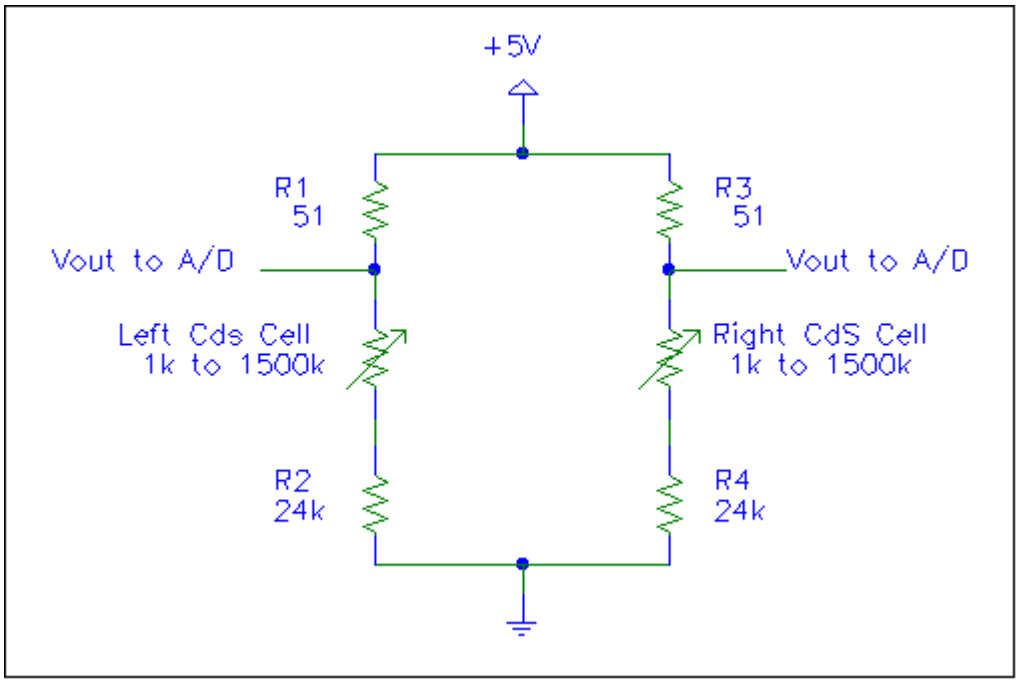

Figure 6. CdS Cell Circuit.

Mercury Switch Tilt Sensor

Spot has many sensors, however his mercury tilt switch is probably his most unique.. With this sensor Spot can determine when he is upside-down or right-side-up. The circuit used to take a reading from the tilt sensor is shown in Figure 7. In this circuit, the mercury switch is closed and connects power to the resistor when Spot is right-side-up, but if Spot gets turned over the mercury switch will no longer make contact and the A/D pin will be pulled to ground.

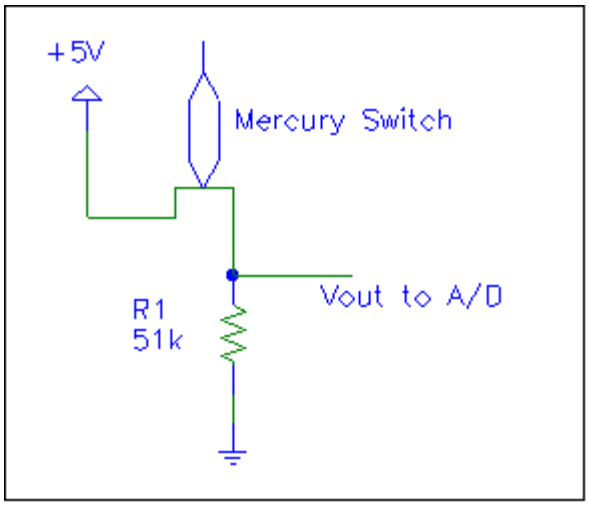

Figure 7. Mercury Tilt Switch Circuit.

BEHAVIORS

Collision Avoidance Behavior

Spot's collision avoidance behavior combines his maneuverability and IR cross vision to be able to avoid objects. Appendix A (Spot_17.c) shows Spot's current collision avoidance algorithm. If Spot sees, with his IR sensors (A/D pins 0 and 1), that he is approaching an object he will decrease the speed of his wheel that is opposite the object, which will cause him to slightly veer away from it. However, if he gets too close to an object, Spot will reverse the speed of the appropriate wheel so that he makes a very quick turn away from it until he no longer sees the object with his IR sensors, then he will continue forward. This method of using IR for collision avoidance works unless the object is black or if Spot becomes caught in a corner.

Object Detection Using Bumpers Behavior

Spot uses his two front bumpers to detect and locate objects that his IR sensors miss - black or very narrow objects. By reading his A/D pin 4 (analog(4)), Spot can tell when his is against an object. Also, since the A/D reading is different for all the bumper closed combinations, Spot will know the location of the object and which way to turn to get away from it. If Spot bumps an object, he stops his wheel on the side where the object is and reverses his other wheel. This causes spot to back up and turn away from the object at the same time. The algorithm for this object detection is shown in Appendix A.

Rip VanWinkle Behavior

Spot's napping behavior is controlled by the microphone array. If Spot hears a loud sound he will immediately stop and take a nap. He will stay where he is and nap forever until he is awakened by another loud noise.

Scared of the Dark Behavior

Spot uses his CdS cells to monitor the light level of the room. When his sensors tell him that the room is too dark, he starts looking for a brighter place to go. Spot acts as if he is afraid of the dark.

Follow Bright Light Behavior

Also by monitoring light intensity with his CdS cells, Spot can follow a bright light. When the light intensity read by the CdS cell A/D ports 2 and 3 gets above a specified value (when a light is pointed directly at Spot's CdS cells), Spot will try to follow the light. The algorithm for this is shown in Appendix A (Spot_17.c).

Fall and Live Behavior

Spot's mercury switch allows him to fall off of short (1 foot to 3 foot) objects and determine how he landed so he can keep going. Spot reads his A/D pin 7 to see if he is upside-down or right-side-up, then he can set his wheel speeds appropriately so he will move forward no matter which side he is on. Also, for this behavior to be accomplished Spot had to get a new, more rigid Plexiglas top.

Two Faced Behavior

Spot mercury switch also gives him the unique ability to have two personalities - a right-side-up personality and an upside-down personality. The difference between these personalities can vary depending on the programming. For example, Spot can be shy with his IR when he is right-side-up, but if he were to fall off of a ledge and land upside-down he could suddenly become fearless and bold with his IR sensors. Spot personality can be changed by setting his sensor limit or other variables to different values.

Escape From Corners Behavior

If Spot gets into a corner or tight area, his IR sensors and bump sensors could trick him into getting stuck, because when he sees something to his right, he will turn left, but then when he sees something to his left, he will turn right, hence getting trapped turning back and forth. Therefore, Spot was given the ability to determine when he is in a situation like this and free himself. This capability is found in one of Spot's previous programs (Spot_7.c). In that program Spot has several counts that he keeps track of: right_cnt, b_right_cnt, left_cnt,

b_left_cnt, cycle_cnt, cycle_cnt_max, bumps_per_cycle_max, and ir_per_cycle_max. If either the bump counts or the IR counts reaches its maximum within a specified time (cycle_cnt_max number of cycles), Spot realizes that he is in a corner or some sort of tight area. Then Spot will back up slightly to clear himself of any objects, turn 180°, start either the Charge song or the Looney Tunes theme playing (the song played is chosen randomly), and continue on his way moving forward.

Which Behavior? Piezo Indicator

Spot is also equipped with a piezo buzzer that is controlled using output compare on PA3. Spot uses his piezo buzzer as his voice to indicate what behavior he is executing. To distinguish between the behaviors he uses different tone frequencies.

Conclusion

Spot, the robot, is able to move around a room maneuvering his way through objects while responding to sound and light. He is able to do these things because of his many sensors that correspond to human abilities. Spot sees with his two IR sensors, feels with his two bumpers, listens with his two ears, gets an indication of light intensity with his two pupils, gets his balance with his inner ear, and communicates with his voice.

Spot may not be the greatest robot in the world, but he's my (almost human) baby and I like him.

Appendix A

```
/*
    THIS PROGRAM WAS:
     WRITTEN BY:          Joel D. Beasley
    LAST MODIFIED:        11/30/95
```

```
      COPYRIGHT:              1995

      IT PROVIDES "SPOT" WITH:

      2  WHEEL DRIVE MOTOR CONTROL
           (Big 4" Wheels)
      2 IR SENSOR, COLLISION AVOIDANCE BEHAVIOR
           (with gradual object dodging)
      2 BUMPER, OBJECT DETECTION
      2 MICROPHONE, SOUND SEEKING BEHAVIOR
           (Spot will turn in the direction of a loud sound)
      2 CdS CELL, LIGHT SEEKING BEHAVIOR
           (If Spot senses that the room he is in is
           too dark, he will try to find a brighter
           place to roam)
           or
           (If Spot detects a very bright light, like a
           flashlight, he will follow it)
      PIEZO BUZZER CONTROL
           (Spot uses various sounds to indicate what he
           is doing)
      ESCAPE FROM CORNERS BEHAVIOR
*/

/* VARIABLES */

/* MOTOR VARIABLES */
float speed_max;
float speed_slow;
float speed_zero;
float speed_offset;
int r_mot;
float r_speed;
int l_mot;
float l_speed;
/* SENSOR VARIABLES  */
int right_side_up;
int mercury_switch;
int ir_addr;
int ir_on;
int ir_off;
int r_ir_in;
int l_ir_in;
int ir_in_limit;
int ir_close;
int r_ir;
int l_ir;
int bump_read;
int bumpers;
int r_bumper;
int l_bumper;
int f_bumpers;
int bk_bumper;
int mic1;
int mic2;
int mic1level;
int mic2level;
int mic_level_limit;
int l_cds;
int r_cds;
int l_cds_in;
int r_cds_in;
int cds_upper_limit;
int cds_lower_limit;
/* DO VARIABLES */
int do_mic_read;
int do_sound_seek;
int do_ir;
int do_light_seek;
/* COUNT VARIABLES */
int right_cnt;
```

```
int b_right_cnt;
int left_cnt;
int b_left_cnt;
int cycle_cnt;
int cycle_cnt_max;
int bumps_per_cycle_max;
int ir_per_cycle_max;
/* TIME VARIABLES */
long ir_settling_time;
long deaf_time;
long sleep_time;
long back_time;
long a_180_time;
long b_turn_time;
long m_turn_time;
long cds_turn_time;
long no_time;
/* VARIABLE TO CHANGE ROBOT STATES */
persistent int state;
int go;
/* PROCESS ID VARIABLES */
int mr_id;

void main()
{      tone(16.0,.4);
       init_vars();
       poke(ir_addr,ir_on);
       mr_id = start_process (mic_read());
       while(go)
       {       if (state > 1)
               {       state = 0;
                       shut_down();
                       break;
               }
               upsidedown_test();
               if (do_sound_seek)
                       sound_seek();
               cds_read();
               if (do_light_seek)
                       light_seek();
               if (do_ir)
                       ir();
               bump();
               motors();
       }
       beep();
       beep();
       beep();
}

void init_vars()
{      speed_max = 100.0;
       speed_slow = 20.0;
       speed_zero = 0.0;
       speed_offset = 4.0;
       r_mot = 1;
       r_speed = speed_max;
       l_mot = 0;
       l_speed = speed_max;
       mercury_switch = 7;
       ir_addr = 0x7000;
       ir_on = 0x03;
       ir_off = 0x00;
       ir_in_limit = 125;
       ir_close = 109;
       r_ir = 1;
       l_ir = 0;
       bumpers = 4;
       r_bumper = 127;
       l_bumper = 169;
       f_bumpers = 192;
```

```
	bk_bumper = 100;
	mic1 = 5;
	mic2 = 6;
	mic_level_limit = 200;
	l_cds = 2;
	r_cds = 3;
	cds_upper_limit = 240;
	cds_lower_limit = 10;
	do_mic_read = 1;
	do_ir = 1;
	do_light_seek = 0;
	cycle_cnt_max = 300;
	bumps_per_cycle_max = 4;
	ir_per_cycle_max = 100;
	ir_settling_time = 2L;
	deaf_time = 500L;
	back_time = 20L;
	a_180_time = 600L;
	b_turn_time = 400L;
	m_turn_time = 300L;
	cds_turn_time = 200L;
	no_time = 0L;
	state = state + 1;
	go = 1;
}

void upsidedown_test()
{	right_side_up = analog(mercury_switch);
	if (right_side_up == 255)
	{	right_side_up = analog(mercury_switch);
		if (right_side_up == 255)
		{	right_side_up = analog(mercury_switch);
			if (right_side_up == 255)
			{	do_sound_seek = 1;
				speed_max = 100.0;
				speed_slow = 20.0;
			}
		}
	}
	else if (right_side_up == 0)
	{	right_side_up = analog(mercury_switch);
		if (right_side_up == 0)
		{	right_side_up = analog(mercury_switch);
			if(right_side_up == 0)
			{	do_sound_seek = 0;
				speed_max = -100.0;
				speed_slow = -20.0;
			}
		}
	}
}

void mic_read()
{	msleep(deaf_time);
	while (do_mic_read)
	{	mic1level = analog(mic1);
		mic2level = analog(mic2);
		if (mic1level>mic_level_limit && mic1level<220)
		{	do_mic_read = 0;
		}
		else if (mic2level>mic_level_limit && mic2level<220)
		{	do_mic_read = 0;
		}
	}
}

void sound_seek()
{	if (do_mic_read == 0)
	{	kill_process(mr_id);
		tone(40.0,.8);
		stop();
```

```
            do_mic_read = 1;
            mr_id = start_process(mic_read());
        }
}

void cds_read()
{       r_cds_in = analog(r_cds);
        l_cds_in = analog(l_cds);
        if (r_cds_in>cds_upper_limit || l_cds_in>cds_upper_limit)
        {       r_cds_in = analog(r_cds);
                l_cds_in = analog(l_cds);
                if (r_cds_in>cds_upper_limit || l_cds_in>cds_upper_limit)
                {       r_cds_in = analog(r_cds);
                        l_cds_in = analog(l_cds);
                        if (r_cds_in>cds_upper_limit || l_cds_in>cds_upper_limit)
                        {       tone(25.0,.2);
                                do_light_seek = 1;
                                do_ir = 0;
                        }
                }
        }
        else if (r_cds_in<cds_lower_limit || l_cds_in<cds_lower_limit)
        {       r_cds_in = analog(r_cds);
                l_cds_in = analog(l_cds);
                if (r_cds_in<cds_lower_limit || l_cds_in<cds_lower_limit)
                {       r_cds_in = analog(r_cds);
                        l_cds_in = analog(l_cds);
                        if (r_cds_in<cds_lower_limit || l_cds_in<cds_lower_limit)
                        {       tone(500.0,.2);
                                do_light_seek = 1;
                                do_ir = 0;
                        }
                }
        }
        else
        {       do_light_seek = 0;
                do_ir = 1;
        }
}

void light_seek()
{
        if (r_cds_in>l_cds_in)
        {       r_speed = -speed_max;
                l_speed = speed_max;
                sleep_time = cds_turn_time;
        }
        else if (r_cds_in<l_cds_in)
        {       r_speed = speed_max;
                l_speed = -speed_max;
                sleep_time = cds_turn_time;
        }
        else if (r_cds_in==l_cds_in)
        {       r_speed = speed_max;
                l_speed = speed_max;
                sleep_time = cds_turn_time;
        }
        motors();
}

void ir()
{       r_ir_in = analog(r_ir);
        l_ir_in = analog(l_ir);
        if (r_ir_in > ir_in_limit || l_ir_in > ir_in_limit)
        {       if (r_ir_in > l_ir_in)
                {       r_speed = speed_max;
                        l_speed = -speed_max;
                        sleep_time = no_time;
                        left_cnt = left_cnt + 1;
                }
                else if (r_ir_in < l_ir_in)
```

```
            {       r_speed = -speed_max;
                    l_speed = speed_max;
                    sleep_time = no_time;
                    right_cnt = right_cnt + 1;
            }
            else if (r_ir_in == l_ir_in)
            {       r_speed = speed_max;
                    l_speed = speed_max;
                    sleep_time = no_time;
            }
      }
      else if (r_ir_in > ir_close || l_ir_in > ir_close)
      {       if (r_ir_in > l_ir_in)
            {       r_speed = speed_max;
                    l_speed = speed_slow;
                    sleep_time = no_time;
            }
            else if (r_ir_in < l_ir_in)
            {       r_speed = speed_slow;
                    l_speed = speed_max;
                    sleep_time = no_time;
            }
            else if (r_ir_in == l_ir_in)
            {       r_speed = speed_slow;
                    l_speed = speed_slow;
                    sleep_time = no_time;
            }
      }
      else
      {       r_speed = speed_max;
            l_speed = speed_max;
            sleep_time = no_time;
      }
}

void bump()
{       bump_read = analog(bumpers);
      if (bump_read)
      {       if (bump_read == r_bumper)
            {       r_speed = speed_zero;
                    l_speed = -speed_max;
                    sleep_time = b_turn_time;
                    b_left_cnt = b_left_cnt + 1;
            }
            else if (bump_read == l_bumper)
            {       r_speed = -speed_max;
                    l_speed = speed_zero;
                    sleep_time = b_turn_time;
                    b_right_cnt = b_right_cnt + 1;
            }
            else if (bump_read == f_bumpers)
            {       r_speed = -speed_max;
                    l_speed = speed_zero;
                    sleep_time = b_turn_time;
                    b_right_cnt = b_right_cnt + 1;
            }
      }
}

void motors()
{       motor(r_mot,r_speed);
      motor(l_mot,l_speed);
      msleep(sleep_time);
}

void shut_down()
{       stop();
      go = 0;
      poke(ir_addr,ir_off);
      kill_process(mr_id);
}
```