Oscar:

The Autonomous Mobile Garbage Collector

Final Report

Chris Beattie

December 9, 1996

University of Florida
Department of Electrical and Computer Engineering

EEL 5666
Intelligent Machines Design Laboratory
Instructor: Dr. Keith L. Doty
TA's: Scott Jantz
Casey Barker
Chris Gomez

Table of Contents

Abstract

The author investigates the requirements for collection of objects. This experiment will be conducted with the autonomous agent Oscar, designed by Chris Beattie. Oscar will use a various IR emitters, IR detectors, bump sensors, and limit switches to avoid objects and detect containers. Once sufficient space for lowering the arm is detected, the agent will deploy its claw and search for a “garbage can”. Upon locating a can, Oscar will grasp the can in its claw, dump the contents of the can into its collection bin, then replace the can. The behaviors will then be repeated.

Executive Summary

Oscar is designed on the Mekatronix ® Talrik body with several modifications. The basic body is a 10” diameter disc made out of 1/8” thick aluminum. Oscar also has an aluminum arm and wooden claw attached the front of the body for manipulating objects. An acrylic 10” x 5” x 5” collection bin is attached to the rear of the body, and a 1/4” acrylic polycarbon bumper rings the perimeter of the body. A Motorola M68HC11 microprocessor, in conjunction with two Novasoft ® expansion boards, the ME11, and MTSX01, control the robot.

Oscar’s motion is governed by two wheels mounted on a differential axis to provide a zero turn radius. Oscar implements four IR sensors arrayed in a “cross-eyed” pattern to implement object avoidance, supplemented by a suite of 10 microswitches used as a bump sensor. The claw has three IR sensors used to detect canisters for collection and a limit switch inside the claw to signal when a canister is in position.

Oscar demonstrates five behaviors: object avoidance, collision recovery, object detection, object manipulation, and object collection. The first two behaviors allow Oscar to find a clear space sufficient to lower the claw and arm assembly, after which an object for collection is sought. Through object detection, a container is found, then picked up, and the contents are dumped into it’s collection bin, simulating the function of a garbage collector.

Introduction

I have chosen to create an autonomous mobile garbage collector for my IMDL project. The realization of this agent could eventually be applied on a real-life scale, replacing costly human labor involved in collecting garbage. My agent is named Oscar, after the lovable Oscar the Grouch of Sesame Street fame. This name not only lends personality to Oscar, but increases his marketability by giving it a "cuteness" factor. Oscar's goal is to find simulated garbage cans, and empty their contents into his collection bin, while at the same time avoiding other objects. Oscar's goals and behaviors will be implemented through a suite of IR sensors and threshold switches working in conjunction with an arm and claw assembly. A Motorola M68HC11, Novasoft ME11, and Novasoft MTSX01, will be combined to provide Oscar's control. The behaviors will be detailed further later in the paper. This paper will also detail the platform, sensors, and actuation used to carry out these behaviors and reach his goals, as well as provide the algorithms used by Oscar.

Integrated System

Oscar's body consists of a 1/8" thick 10" diameter aluminum body based on Novasoft's Talrik robot body. A Motorola M68HC11, Novasoft ME11, and Novasoft MTSX01 combine to provide Oscar's control. This combination of three boards will hereto be referred to as my "control board". A suite of four IR sensors will be laid out on the front of the body to implement object avoidance. These sensors are supplemented by a series of ten mircoswitches used in combination with a flexible bumper, both placed around the perimeter of the robot to implement collision recovery. A separate suite of three IR sensors are placed on the end of the claw to implement object detection. Once an object is detected, a limit switch inside the claw signals the control board that the object is ready for manipulation, and then Oscar lifts the container and dumps its contents into his collection bin. The code for the control board is written in IC, and contains separate subroutines for object and collision recovery, object detection, and object manipulation and collection. Figure 1 shows an overall view of the integrated system.

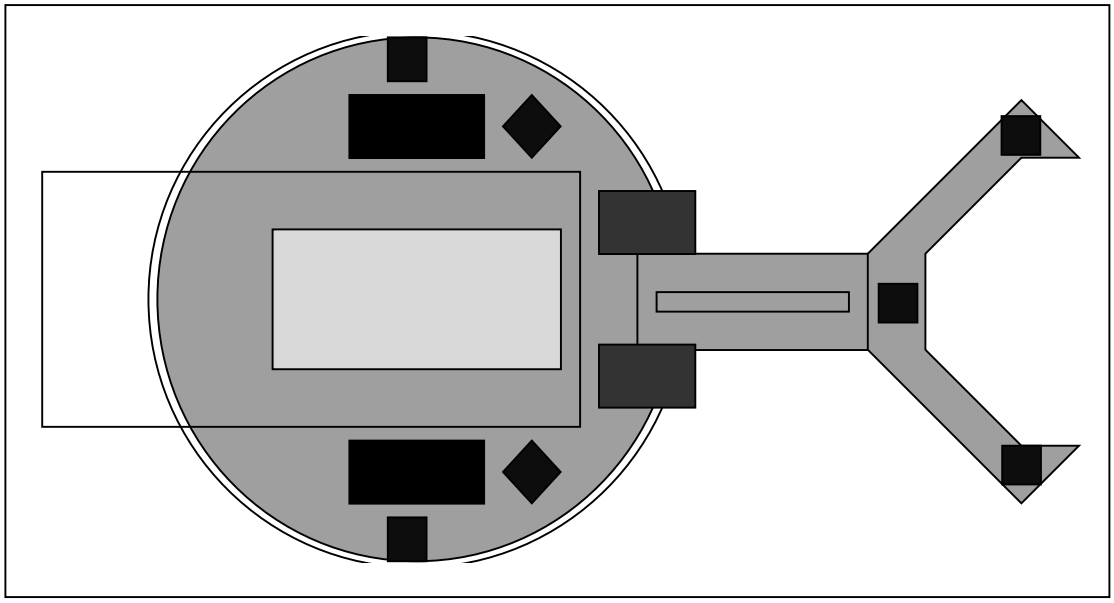

Figure 1: Overall view of Oscar

Mobile Platform

Oscar's platform consists of a basic Talrik body cut out of 1/8" aluminum. My claw and arm assembly are mounted to the front of the platform, and my collection bin is mounted to the rear, above the control board. I chose this design due the availability of the Talrik body, and its dynamic functionality. When I mounted the claw and arm to the front of the body, I had not properly compensated for its weight, and it cause my body to tip over, because the support caster is in the rear of the body. I corrected this by making the collection bin large enough to hang off the back of the body and counterbalance the weight of the claw. I had to do this because placing a separate caster under the front of the robot was not an option, since the height of the casters are greater than that of the wheels used for actuation. Figure 2 illustrates the Mobile Platform.

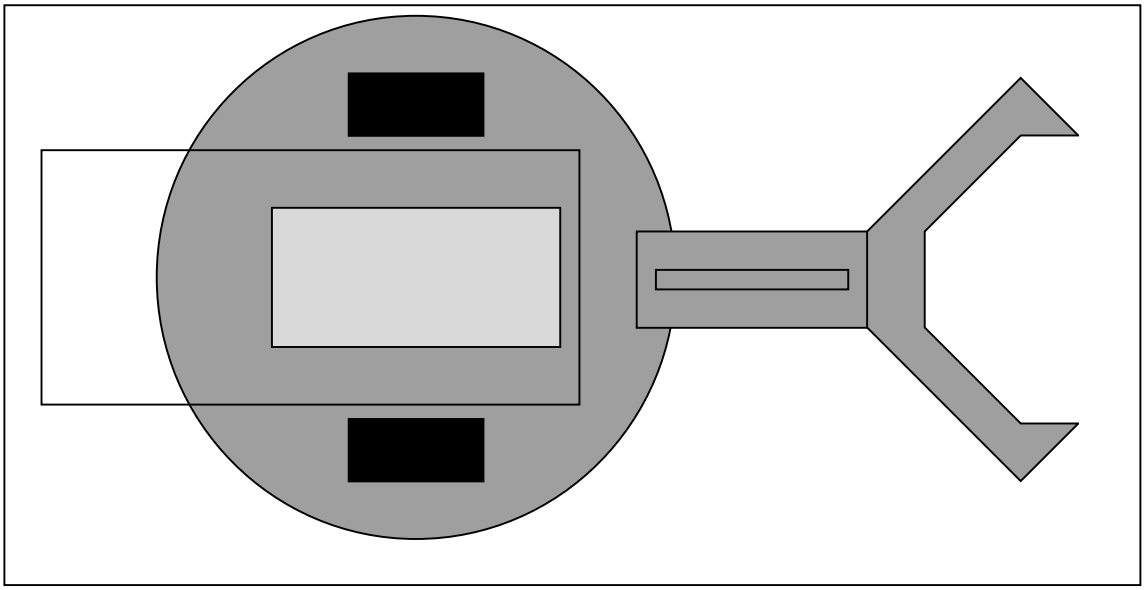

Figure 2: Platform, Control Board, Claw, Arm , Wheels, and Collection Bin

Actuation

Oscar has three types of actuators: his wheels, arm, and claw. Motor drivers on my control board operate the four servos connected to these devices. Two of the servos have been converted into DC motors via the standard MIL method provided by Scott Jantz. These two DC motors operate the wheels on the body in order to implement object avoidance, collision recovery, object collection, and basic movement. The arm is a seven inch long, one inch wide, aluminum bar. The arm is attached to the body by a hinge which allows it to be raised into a dumping position. The other end of the arm is connected to the claw. One servo has a bar connected to it which slides under the arm and implements the raising and lowering of the arm. A compression spring is attached to the collection bin to push the arm back past vertical so that it can be lowered by the servo

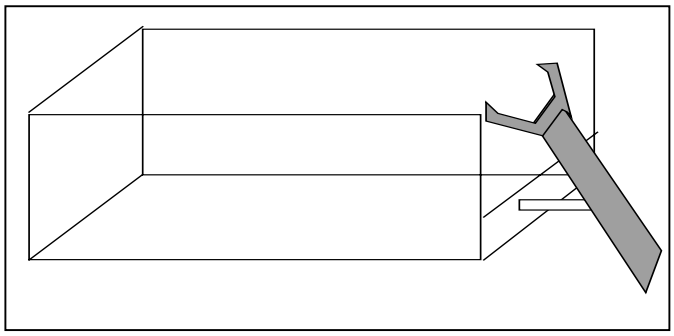

Figure 3: Arm, Claw, Collection Bin, and Spring

(see fig.3). When the arm is in its fully raised position, it will position the claw over the collection bin. The claw is based on the hand of Microbot's Minimover 5. The claw is closed by a single pull cable attached to a extension spring, which is pulled by the fourth servo. When the servo releases the extension spring, two torsion springs reopen the claw. The claw and arm, which I designed as my senior project under the guidance of Dr. Doty,

will be discussed in much further detail in my senior design paper, which is available from Dr. Doty.

Sensors

I have implemented four different sensor suites on Oscar. The first is a series of four IR sensors arrayed in the classic cross-eyed pattern used on Talrik robots. These sensors detect 40 Khz modulated IR light and are used for object avoidance. Their alignment in a cross-eyed pattern allows for a complete covering of the area in front of the robot with no blind spots. The four object avoidance sensors are supplemented by a bump sensor, consisting of 10 mircoswitches dispersed around the perimeter of the body. These mircoswitches are surrounded by a 1/4" acrylic skirt which acts a bumper. This sensor is necessary to implement collision recovery, supplementing collision avoidance, because not all obstacles will be avoided. This can be attributed to the fact that IR light will not properly reflect off of shiny metal objects and will be absorbed by black objects. Whenever an object hits the skirt, it triggers a mircoswitch corresponding to the location of the collision. This tells the control board where the collision occurred and then the proper recovery algorithm is implemented. After the obstacle is cleared, normal object avoidance is resumed. The third sensor suite is an array of three columnated IR sensors operating at 40 Khz on the claw to implement object detection. One sensor is placed on each of the tips of the claw, and a third directly in the middle of the claw. When the claw is deployed, Oscar rotates in a complete circle looking for an object to pick up. When the two sensors on the tips of the claw do not detect anything, but the sensor in the middle of the claw does, Oscar knows that there is an object present that will fit in the claw and signals the control board. Oscar then goes into collection mode, where he attempts to close in on the object and guide it into his claw. Once the object is detected and positioned properly, the fourth sensor, a limit switch on the interior of the claw, signals

the control board that the object is in position to be picked up, and Oscar enters manipulation mode, and closes the claw. Then the object is inverted over the collection bin by the arm so that any contents of the container will be dumped into the collection bin. After shaking out all of the contents, Oscar replaces the container back on the ground and goes back into avoidance mode. One thing to be careful of when implementing a bumper is that you allow enough room for the mircoswitches, not only when determining the length of the acrylic strip to use, but also when accounting for any irregularities in the curvature of the strip due to the fasteners used to attach the skirt to your body. The placement of the sensors is shown in figure 4.

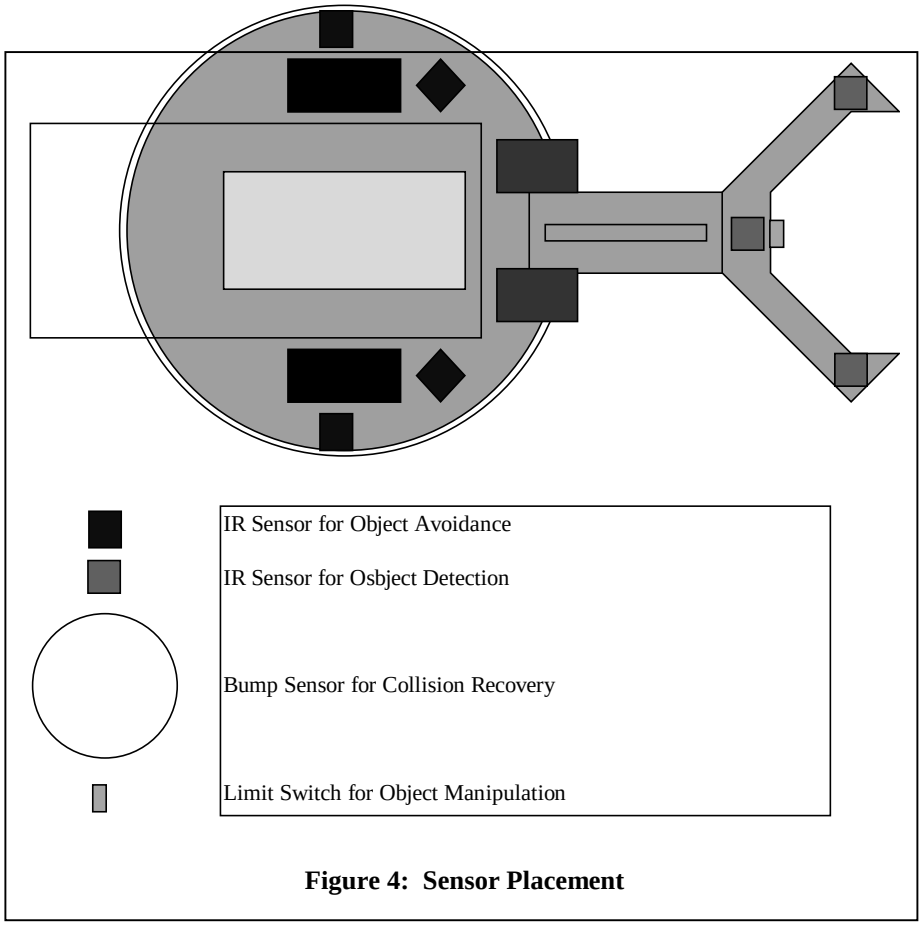

Figure 4: Sensor Placement

Behaviors

Oscar demonstrates five behaviors, the first of which is object avoidance. This behavior is implemented via the suite of four IR sensors discussed above. Whenever a reading is picked up on one of the sensors that is above the given threshold, Oscar thinks he has found an object to avoid. He tells the control board the location of the object, and then the control board implements a gradual turn to avoid the object. Should the IR sensors fail to detect an object, the supplementary bumper system will detect a collision, and Oscar will demonstrate his collision recovery behavior. Whenever a bump is detected by the bumper, the control board determines where the collision occurred and then responds appropriately by moving away from the object and turning to go off in a different direction. These two behaviors combine to allow Oscar to find sufficient space to deploy his claw and start his object detection behavior. Object detection consists of using the three IR sensors on his claw to find an object that will fit into his claw. Once Oscar determines that sufficient space exists, he lowers his claw and rotates 360° looking for an object as described above. Once an object has been found that is suitable for collection, Oscar moves into a position to allow for object manipulation. Oscar knows he is in position when the limit switch inside the claw is depressed, signaling the control board that an object is ready to be picked up. Oscar then implements his object manipulation behavior, closing the claw around the object. No sensors are needed to determine when the claw is fully closed because of the extension spring that is placed in line with the cable that closes the claw and the servo that pulls the cable. This feature is detailed further in my senior design paper, available from Dr. Doty. Once the object is secured for manipulation, the arm is raised, lifting the container and positioning it over

the collection bin. Oscar then enters his object collection behavior, lifting the container over the bin and shaking it in the true fashion of real garbage collectors, so that all of the contents of the container will be emptied out into the bin. After collecting the contents of the container, Oscar lowers the container back onto the ground, raises his claw and re-enters his object avoidance behavior. In this way Oscar is not like your typical garbage collector, as he attempts to leave the container in an upright position, unlike your average, real-life garbage man. However, after emptying several containers, Oscar grows tired of his mundane line of work and tends to be more like your typical garbage collector, dropping the containers a short distance to the ground, rudely leaving them lying prone on the ground. This is done for two reasons. First, to imitate real life and let Oscar display some of the grouchiness of his namesake, and second to prevent the repeated picking up of the same objects over and over again.

Experimental Layout

To test Oscar, I will place him in the "course" provided by the MIL. The course consists of an irregular polygon, approximately the shape of a 10' diameter circle, constructed out of white painted 2'x4's. Inside the arena, several random objects will be placed for Oscar to demonstrate his object avoidance and collision recovery behaviors. I will also insert several "garbage cans" into the playing field for Oscar to pick up and collect the contents of. The cans will consist of pieces of 2" diameter PVC pipe glued to cardboard bases. The layout is shown in figure 5.

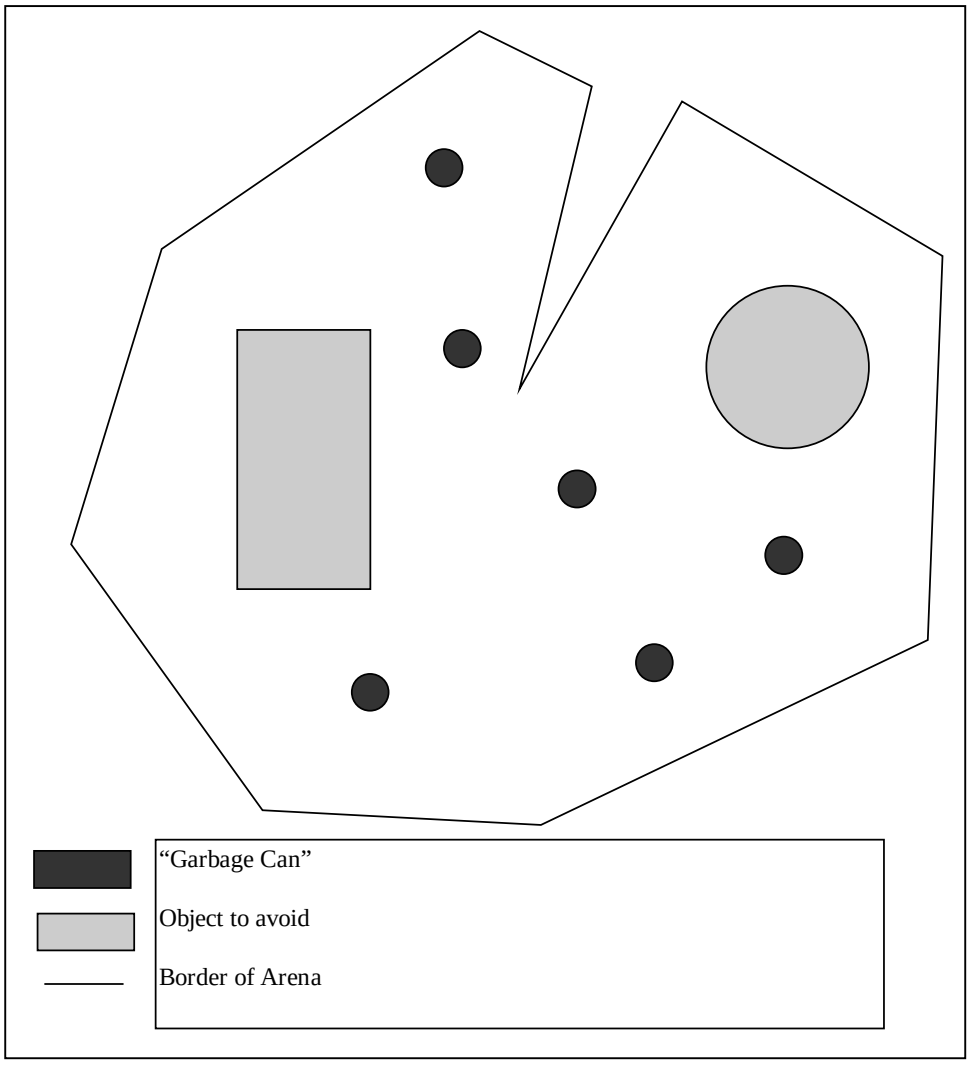

Figure 5: Experimental Layout

Conclusion

During the course of this semester, I have built a fully autonomous mobile agent than can achieve the goals I set out for him at the beginning of the semester. Oscar can avoid objects, recover from collisions, detect containers to be picked up, manipulate those containers, and collect the contents of those containers. He achieves all of these goals through a system of cleverly deployed simple sensors, including IR sensors, microswitches, and limit switches. I wrote all of the code in Interactive C, a C derivative, and Oscar displays signs of intelligence as he performs his behaviors while questing to complete his goals. Currently Oscar has hard set thresholds for object avoidance. I would like to modify this in the future so that he modifies those thresholds and calibrates them himself. The claw and arm assembly function well, although I would like to improve on the lifting system in the future. If I were to start the project over entirely, I would not try to tackle so much in one semester, but rather spread the work out over two semesters in order to do a more complete and thorough job.

References

[1] Joseph L. Jones and Anita M. Flynn, *Mobile Robots: Inspiration to Implementation*

K. Peters 1993

[2] *M68HC11 Reference Manual*

Motorola 1991

[3] *M68HC11E9 Technical Data Manual*

Motorola 1991

[4] Dr. Keith L. Doty, *Talrik, ME11, and MTSX01 Assembly Manuals*,

Novasoft - Mekatronix 1995

[5] Pattie Maes, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, MIT Press, Cambridge, MA, 1990.

[6] Dr. Keith L. Doty, Scott Jantz, Charles Barker, Lee Rossey, and Chris Gomez, (countless invaluable verbal references)

Appendix B: Code and Algorithms

```
/* Motor control/analog input program
   by Chris Beattie for Oscar

   Sensors:      3       2
                     4            5
   Motors:     0          1
*/

/* Global Variable assignment */
int r_diag,r_side,l_diag,l_side;
int r_avoid=0;
int l_avoid=0;
int diag_thr=110;
int side_thr=90;

/* Takes a reading off of one of the IRDTs */
int irdt(int detector) {
  if ((detector >= 2) && (detector <= 7)) return(analog(detector));
  else if (detector = 8) { poke(0x4000,0x00); return(analog(0)); }
  else if (detector = 9) { poke(0x4000,0x01); return(analog(0)); }
  else if (detector =10) { poke(0x4000,0x02); return(analog(0)); }
  else if (detector =11) { poke(0x4000,0x03); return(analog(0)); }
  else if (detector =12) { poke(0x4000,0x04); return(analog(0)); }
  else if (detector =13) { poke(0x4000,0x05); return(analog(0)); }
  else if (detector =14) { poke(0x4000,0x06); return(analog(0)); }
  else return(0);
}

/* Takes a reading off of one of the CDSs */
int cds(int detector) {
  if (detector = 1) { poke(0x4000,0x07); return(analog(0)); }
  else if (detector = 2) { poke(0x4000,0x00); return(analog(1)); }
  else if (detector = 3) { poke(0x4000,0x08); return(analog(1)); }
  else if (detector = 4) { poke(0x4000,0x10); return(analog(1)); }
  else if (detector = 5) { poke(0x4000,0x18); return(analog(1)); }
  else if (detector = 6) { poke(0x4000,0x20); return(analog(1)); }
  else return(0);
}

/* Takes reading off of front bumper */
int f_bump() {
  poke(0x4000,0x30);
  return(analog(1));
}

/* Takes reading off of rear bumper */
int r_bump() {
  poke(0x4000,0x28);
  return(analog(1));
}
```

```
/* Takes readings off of the four object avoidance detectors */
int read() {
  int react;
  int high;
  r_diag=irdt(2)-diag_thr;
  l_diag=irdt(3)-diag_thr;
  l_side=irdt(4)-side_thr;
  r_side=irdt(5)-side_thr;
  high=0; react=0;
  if (r_diag >= 0) { high=r_diag; react=2; }
  if ((l_diag >= 0) && (l_diag > high)) { high=l_diag; react=3; }
  if ((l_side >= 0) && (l_side > high)) { high=l_side; react=4; }
  if ((r_side >= 0) && (r_side > high)) { high=r_side; react=5; }
  if ((r_diag >= -2) && (l_diag >= -2)) { high=0; react=1; }
  if ((r_side >= 0) && (l_side >= 0)) { high=0; react=1; }
  return(react);
}

/* Sets motor 0 to l_motor and motor 1 to r_motor */
void motor_adjust(int l_motor, int r_motor) {
  int i;
  int l_step=(l_motor-l_avoid)/10;
  int r_step=(r_motor-r_avoid)/10;
  for (i=0;i<10;i++) {
            l_avoid+=l_step;
            r_avoid+=r_step;
            motor(0,(float)l_avoid);
            motor(1,(float)r_avoid);
            sleep(0.05);
  }
}

/* Object avoidance routine */
void object_avoid() {
  int sensor=read();
  int back=0;
  while(sensor) {
            if (sensor == 1) { motor_adjust(-100,-100); back=1; }
            if ((sensor == 2) || (sensor == 4)) motor_adjust(100,-100);
            if ((sensor == 3) || (sensor == 5)) motor_adjust(-100,100);
            sensor=read();
  }
  if (back) { motor_adjust(-100,100); sleep(0.02); }
  back=0;
  motor_adjust(100,100);
}

void main() {
  motor_adjust(100,100);
  poke(0x7000,0xff);
  while(1) {
            object_avoid();
  }
```

```
}
```

Note: This code is incomplete and will be supplemented on December 12, 1996.