“Orb”

A Two Legged Robot
Final Report

University of Florida

Department of Electrical Engineering
EEL 5666 - Intelligent Machines Design Laboratory

Date:	4/29/97
Student Name:	Purvesh Thakker
Instructor:	Keith L. Doty

Table of Contents

Abstract

For this robotics project I built a two-legged robot named Orb. Orb walks along until it sees an object. It then turns its head left and right to determine which direction is clear. It turns in that direction and keeps going. Orb uses parallel linkages for leg segments to keep from falling forwards and backwards, and a shifting weight to keep from falling to either side. Two Motorola 68HC11 microcontrollers run the robot. One acts as Orb's brain, while the other acts as a servo controller. The walk routines use humanly intuitive language with timing delays.

Executive Summary

For this Robotics project I built a two-legged robot named Orb. Orb walks along until it sees an object. It then turns its head left and right to determine which direction is clear. It turns in that direction and keeps going.

Mechanically Orb needs to worry about two types of balance, front-to-back balance and side-to-side balance. Charles Pitzer solved the front-to-back balance problem with his design for the legs. He used parallel linkages to keep the feet and body horizontal. With this mechanical design, the robot does not ever fall in the front-to-back direction. Bob Pitzer solved the side-to-side balance problem with his design for the robot body. He shifts the robot's center of gravity left and right by placing the batteries on a pendulum type arm. A servo then swings the weight left and right. A dome designed by Casey Barker covers Orb's body. The dome sits on a servo and can turn left and right.

Electronically, Orb has two 68HC11 microcontrollers and a Servomux board. One microcontroller acts as the robot's brain. It sends instructions to the second microcontroller which acts a servo controller. The servo controller sends encoded servo signals to the servomux board which decodes them. The servos plug directly into the servomux board. Orb also has four infrared sensors to detect walls, two in the dome and two in the feet.

Orb's software has a layered design that gets more abstract as it goes along. Changes in the lower layers ripple through the upper layers. This makes programming the robot to walk easy since commands are given in humanly intuitive language. I program walk routines by sending commands, delaying for a period of time, and sending more commands.

Introduction

For this robotics project I built a two-legged walking robot. Orb now has the ability to walk, turn, and see obstacles. If it detects an obstacle when walking along, Orb stops, looks left and right, and turns in the less cluttered direction. In this paper, I divide the work for this project into three categories: mechanics, electronics, and software. After a brief overview of these categories I will go into more detail with each.

Overview

Mechanically Orb has two types of balance that it must take into account, front-to-back balance and side-to-side balance. Charles Pitzer solved the front to back balance with his design for the legs, and Bob Pitzer solved the side to side balance with his design for the robot body. Orb also has a twelve inch dome designed by Casey Barker. The Pitzer brothers designed the entire robot on AutoCad and cut it out by machine using the T-Tech in the robotics lab. They built the robot out of light but strong plywood and designed it entirely of interlocking parts. Because of the interlocking parts, we did not need to use glue, and so can take apart the robot when needed.

Electronically, Orb has two 68HC11 microcontrollers. The first processor acts as the brain of the robot. Using inputs from four infrared sensors and the walk routines that I programmed, the first processor decides when to move each servo, where to move it, and how fast to move it. It then passes instructions to the second processor which controls the servos. The second processor time shares its five output compare lines to run up to twenty servos. The servomux board then decodes the output compare lines. The servos plug directly into the servomux board.

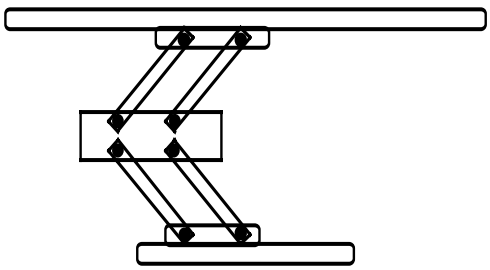
The software has a layered design that becomes more and more abstract. This allows me to program walk routines very easily with human language instead of numbers.

Any changes I make in lower layers ripple through the upper layers. The ultimate goal was to set up the walk routines so that moving the robot was just like moving a wheeled robot. I would only need to call a function that says go forward, or turn left or right.

Mechanical Design

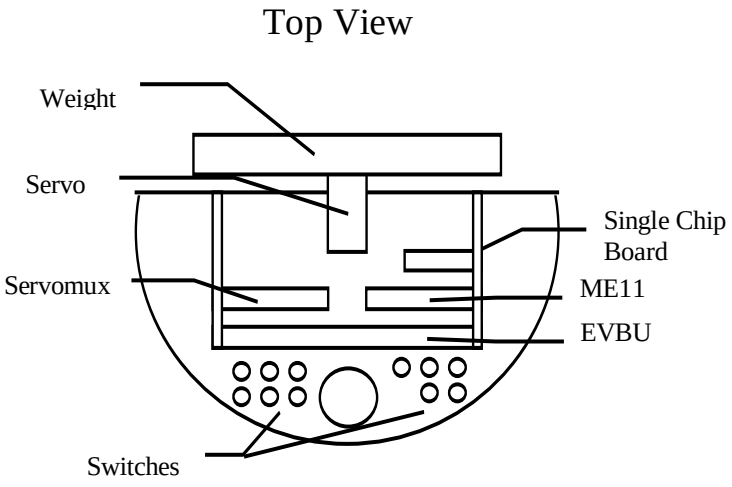
As stated earlier, Chuck solved the issue of front-back balance with his design for the legs. He used parallel linkages for each leg segment so that the feet and the body of the robot are always parallel to each other and the ground. As long as the feet are not moved too far forward, the robot does not fall in the front-back direction. This mechanical solution is much better than the software solution I originally wanted to use. That solution would have involved real time balancing using sensors. After seeing other projects that attempted real time balancing, I realize that it would be very difficult for a three degrees of freedom walker.

Side View



Bob Pitzer solved the side to side balance with a shifting weight. He mounted the batteries on a pendulum connected to a servo. The servo swings the battery to the left and right. Moving the weight shifts the center of gravity of the robot over one leg so that it can pick up the other leg. The size of the dome limits how far the weight shifts, and so the robot can almost stand on one leg, but not quite. Even though the robot does not quite balance statically, it falls slowly enough to pick up its opposite leg and put it back down in time. I had originally planned not to shift the center of gravity at all, and just take short quick steps. After seeing how difficult it is to walk with shifting the weight, I realize that my original plan would not have worked.

Electronics

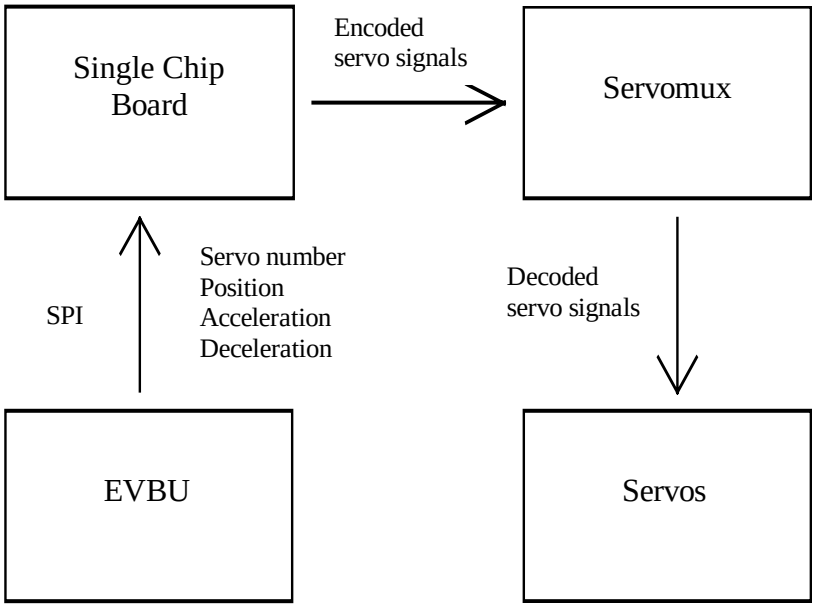


The brain processor runs IC and sends commands to the second processor over the SPI. The second processor runs assembly code and accelerates the servo up to some maximum velocity where it cruises and eventually decelerates down to its final position. The servo controller portion of the robot comprises my Senior Design Project. For this paper, the brain processor needs only call a function and pass certain variables as follows:

set_servo_pul(servo number, position, max velocity, acceleration, deceleration);

Servo number - the number of the servo to move
Position - the new position to go to
Max velocity - the maximum speed that the servo should move at
Acceleration - the rate at which the servo should accelerate
Deceleration - the rate at which the servo should decelerate

Because there are only five output compares on the 68HC11, the servo controller time shares the lines. I need some external hardware to decode the signals which the Servomux board does. The servos plug directly into this board.



In addition to the four electronics boards there are four infrared sensors, two on the dome and two on the feet, 11 switches and sockets, and six servos. Orb uses five high torque servos (about 180 oz-in) and one regular servo (42 oz-in). The regular servo turns the dome while the high torque servos move the shifting weight and the legs.

Software

The first layer of the software allows me to move the joints of the robot with human language instead of numbers. To move a joint I simply call a function such as left_hip_forward(). The function then sends the appropriate variables to the servo controller. This way I do not have to think about the servo number, the forward position in clock pulses, or the speeds and accelerations for that movement. The next layer involves macro leg commands that move several joints at once. The next layer moves the legs in sequence with time delays. For example to take a right step forward, I may use the following sequence of commands.

1. Weight left wait 1/2 sec.
2. Right leg up wait 1/5 sec.
3. Right leg forward
   and left leg back wait 1/5 sec.

The next layer consists of coordinated steps and turns. It determines which leg is forward and steps the other, or moves the appropriate leg when turning. This way I only need to say step forward or turn left without wondering which leg to move. At this point controlling Orb is just like controlling a wheeled robot, and so the obstacle avoidance layer is very similar to a wheeled robot's obstacle avoidance program. It simply looks for a threshold value on the four infrared sensors. If any of the sensors sees a wall, the robot stops, looks left and right, and turns toward the direction that gives a lower reading on the two dome infrared sensors.

Conclusion

The robot walks well, turns well, and detects objects well, but with very few real world inputs. In the spirit of the class, it would be nice to add more sensors to the robot. One set of sensors that would be very useful in programming the robot to walk would be servo position feedback. The brain of the robot never knows exactly where the servos are at any given time because it sends commands to the servo controller. In fact, even the servo controller does not know where the servos are. It knows where the servos should be, but the servos are mechanical devices that take time to reach the positions that they are told to go to, especially when there is a load attached. I could take a reading off the potentiometers inside the servos. This reading could feed through the servo controller back to the robot brain, or it could go directly to the brain processor. With this new information I would not have to tweak all of the time delays when programming the robot

to walk. I could instead use the more intuitive approach of telling the robot to move a joint, wait until it reaches the position, and then perform the next movement.

In summary I learned a great deal this semester including both technical skills and just what takes to finish a project like this. Orb could never have worked this well without the mechanical expertise of Bob and Chuck Pitzer. UF is lucky to have a lab with the kind of cooperative environment found at MIL. I would highly recommend the class to anyone interested.

Appendix

Orbmove.c

```
/*************************************************************************
****
        Constants
*************************************************************************
***/

int servo_save = 500;
int crouch_shift = 200;
int crouch_shift2 = 500;
int weight_lower = 0;           /* 500 for step2 */

int left_hip = 1;
int left_hip_center_pos = 3500;
int left_hip_up_pos = left_hip_center_pos - 1100;
int left_hip_max_pos = left_hip_center_pos + 1000;
int left_hip_min_pos = left_hip_up_pos + servo_save;
int left_hip_crouch_pos = left_hip_center_pos - crouch_shift;
int left_hip_crouch2_pos = left_hip_center_pos - crouch_shift2;

int right_hip = 2;
int right_hip_center_pos = 3650;
int right_hip_up_pos = right_hip_center_pos + 1100;
int right_hip_min_pos = right_hip_center_pos - 1000;
int right_hip_crouch_pos= right_hip_center_pos + crouch_shift;
int right_hip_crouch2_pos= right_hip_center_pos + crouch_shift2;
int right_hip_max_pos = right_hip_up_pos - servo_save;

int left_foot = 3;
int left_foot_center_pos = 3500;
int left_foot_up_pos = left_foot_center_pos + 1200;
int left_foot_min_pos = left_foot_center_pos - 650;
int left_foot_crouch_pos = left_foot_center_pos + crouch_shift;
int left_foot_crouch2_pos = left_foot_center_pos + crouch_shift2;
int left_foot_max_pos = left_foot_up_pos - servo_save;

int right_foot = 4;
int right_foot_center_pos = 3800;
int right_foot_up_pos = right_foot_center_pos - 1200;
int right_foot_max_pos = right_foot_center_pos + 650;
int right_foot_min_pos = right_foot_up_pos + servo_save;
```

```
int right_foot_crouch_pos = right_foot_center_pos - crouch_shift;
int right_foot_crouch2_pos = right_foot_center_pos - crouch_shift2;

int weight = 5;
int weight_center_pos = 3600;
int weight_min_pos = weight_center_pos - 1900 + weight_lower;
int weight_max_pos = weight_center_pos + 1700 - weight_lower;

int dome = 6;
int dome_center_pos = 2900;
int dome_min_pos = 1300;
int dome_max_pos = 4600;

int mvel=150;
int acc=40;
int dec=10;

int dome_mvel = 255;
int dome_acc = 255;
int dome_dec = 255;

int position;          /* 0 is left leg forward, 1 is right leg forward,
                            2 is feet together */
int walk_process_PID;

int walk_forward_speed;
int turn_left_speed;
int turn_right_speed;
int walk_backward_speed;

int turn_left_flag;
int turn_right_flag;
int walk_backward_flag;

/*************************************************************************
****
        Precise commands for each joint - Accepts positions in pulse widths.
        Send positive and negative pulse widths to move to a specific
        position from the center.
*************************************************************************
***/

void left_hip_place(int temp)
        {
        int pos;
```

```
        pos=left_hip_center_pos + temp;
        if (pos < left_hip_min_pos) {pos=left_hip_min_pos;}
        if (pos > left_hip_max_pos) {pos=left_hip_max_pos;}
        set_servo_pul(left_hip, pos, mvel, acc, dec);
        }

void right_hip_place(int temp)
        {
        int pos;

        pos=right_hip_center_pos - temp;
        if (pos < right_hip_min_pos) {pos=right_hip_min_pos;}
        if (pos > right_hip_max_pos) {pos=right_hip_max_pos;}
        set_servo_pul(right_hip, pos, mvel, acc, dec);
        }

void left_foot_place(int temp)
        {
        int pos;

        pos=left_foot_center_pos - temp;
        if (pos < left_foot_min_pos) {pos=left_foot_min_pos;}
        if (pos > left_foot_max_pos) {pos=left_foot_max_pos;}
        set_servo_pul(left_foot, pos, mvel, acc, dec);
        }

void right_foot_place(int temp)
        {
        int pos;

        pos=right_foot_center_pos + temp;
        if (pos < right_foot_min_pos) {pos=right_foot_min_pos;}
        if (pos > right_foot_max_pos) {pos=right_foot_max_pos;}
        set_servo_pul(right_foot, pos, mvel, acc, dec);
        }


void weight_place(int temp)
        {
        int pos;

        pos=weight_center_pos + temp;
        if (pos < weight_min_pos) {pos=weight_min_pos;}
        if (pos > weight_max_pos) {pos=weight_max_pos;}
```

```
        set_servo_pul(weight, pos, mvel, acc, dec);
        }

void dome_place(int temp)
        {
        int pos;

        pos=dome_center_pos + temp;
        if (pos < dome_max_pos) {pos=dome_max_pos;}
        if (pos > dome_min_pos) {pos=dome_min_pos;}
        set_servo_pul(dome, pos, mvel, acc, dec);
        }

/*************************************************************************
****
        Special positions for each joint
*************************************************************************
***/

void left_hip_forward() { set_servo_pul(left_hip, left_hip_max_pos, mvel, acc, dec);
}
void left_hip_center()  { set_servo_pul(left_hip, left_hip_center_pos, mvel, acc, dec);    }
void left_hip_crouch()  { set_servo_pul(left_hip, left_hip_crouch_pos, mvel, acc, dec);
}
void left_hip_crouch2() { set_servo_pul(left_hip, left_hip_crouch2_pos, mvel, acc, dec);
}
void left_hip_back()    { set_servo_pul(left_hip, left_hip_min_pos, mvel, acc, dec);      }
void left_hip_up()      { set_servo_pul(left_hip, left_hip_up_pos, mvel, acc, dec);       }

void right_hip_forward(){ set_servo_pul(right_hip, right_hip_min_pos, mvel, acc, dec);
}
void right_hip_center() { set_servo_pul(right_hip, right_hip_center_pos, mvel, acc, dec);
}
void right_hip_crouch() { set_servo_pul(right_hip, right_hip_crouch_pos, mvel, acc,
dec);   }
void right_hip_crouch2(){ set_servo_pul(right_hip, right_hip_crouch2_pos, mvel, acc,
dec);   }
void right_hip_back()   { set_servo_pul(right_hip, right_hip_max_pos, mvel, acc, dec);
}
void right_hip_up()     { set_servo_pul(right_hip, right_hip_up_pos, mvel, acc, dec);     }

void left_foot_forward(){ set_servo_pul(left_foot, left_foot_max_pos, mvel, acc, dec);
}
void left_foot_center() { set_servo_pul(left_foot, left_foot_center_pos, mvel, acc, dec);
}
```

```
void left_foot_crouch() { set_servo_pul(left_foot, left_foot_crouch_pos, mvel, acc, dec);
}
void left_foot_crouch2(){ set_servo_pul(left_foot, left_foot_crouch2_pos, mvel, acc,
dec);   }
void left_foot_back()   { set_servo_pul(left_foot, left_foot_min_pos, mvel, acc, dec);      }
void left_foot_up()     { set_servo_pul(left_foot, left_foot_up_pos, mvel, acc, dec);      }

void right_foot_forward(){set_servo_pul(right_foot, right_foot_min_pos, mvel, acc, dec);
}
void right_foot_center(){ set_servo_pul(right_foot, right_foot_center_pos, mvel, acc,
dec);  }
void right_foot_crouch(){ set_servo_pul(right_foot, right_foot_crouch_pos, mvel, acc,
dec);  }
void right_foot_crouch2(){set_servo_pul(right_foot, right_foot_crouch2_pos, mvel, acc,
dec);  }
void right_foot_back()  { set_servo_pul(right_foot, right_foot_max_pos, mvel, acc, dec);
}
void right_foot_up()    { set_servo_pul(right_foot, right_foot_up_pos, mvel, acc, dec);
}

void weight_left()      { set_servo_pul(weight, weight_min_pos, mvel, acc, dec);        }
void weight_center()    { set_servo_pul(weight, weight_center_pos, mvel, acc, dec);     }
void weight_right()     { set_servo_pul(weight, weight_max_pos, mvel, acc, dec);        }

void dome_left()      { set_servo_pul(dome, dome_min_pos, dome_mvel, dome_acc,
dome_dec);        }
void dome_center()    { set_servo_pul(dome, dome_center_pos, dome_mvel, dome_acc,
dome_dec);     }
void dome_right()     { set_servo_pul(dome, dome_max_pos, dome_mvel, dome_acc,
dome_dec);        }

/*************************************************************************
****
        Special commands for an entire leg
*************************************************************************
***/

void left_leg_extend()  { left_hip_center();    left_foot_center();     }
void left_leg_crouch()  { left_hip_crouch();    left_foot_crouch();     }
void left_leg_crouch2() { left_hip_crouch2();   left_foot_crouch2();    }
void left_leg_contract(){ left_hip_up();        left_foot_up();         }

void left_leg_fforward(){ left_hip_forward();   left_foot_forward();    }
void left_leg_forward() { left_hip_forward();   left_foot_center();     }
void left_leg_back()    { left_hip_back();      left_foot_center();     }
```

```
void left_leg_fback()   { left_hip_back();      left_foot_back();       }


void right_leg_extend() { right_hip_center();   right_foot_center();    }
void right_leg_crouch() { right_hip_crouch();   right_foot_crouch();    }
void right_leg_crouch2(){ right_hip_crouch2();  right_foot_crouch2();   }
void right_leg_contract(){right_hip_up();       right_foot_up();        }

void right_leg_fforward(){right_hip_forward();  right_foot_forward();   }
void right_leg_forward(){ right_hip_forward();  right_foot_center();    }
void right_leg_back()   { right_hip_back();     right_foot_center();    }
void right_leg_fback()  { right_hip_back();     right_foot_back();      }

/*************************************************************************
****
        Simple step and twist routines
*************************************************************************
***/
void step_forward_left()
        {
weight_right();                 right_leg_crouch();     msleep(800L);
                left_leg_contract();    right_foot_center();    msleep(80L);
                                        right_leg_back();       msleep(100L);
                left_leg_crouch2();                             msleep(500L);
        position = 0;
        }
/*************************************************************************
****/
void step_forward_right()
        {
weight_left();  left_leg_crouch();                      msleep(800L);
                left_foot_center();     right_leg_contract();   msleep(80L);
                left_leg_back();                        msleep(100L);
                                        right_leg_crouch2();    msleep(500L);
        position = 1;
        }
/*************************************************************************
****/
void step_forward_left2()
        {
weight_left();                                  msleep(500L);
weight_right();                                 msleep(150L);
                left_leg_contract();    right_leg_extend();     msleep(150L);
                left_leg_crouch();      right_leg_back();       msleep(200L);
        position = 0;
```

```
        }
/*************************************************************************
****/
void step_forward_right2()
        {
weight_right();                                 msleep(500L);
weight_left();                                  msleep(150L);
                left_leg_extend();      right_leg_contract();   msleep(150L);
                left_leg_back();        right_leg_crouch();     msleep(200L);
        position = 1;
        }
/*************************************************************************
****/
void twist_left()
        {
                weight_left();
                left_hip_back(); left_foot_forward();
                right_hip_back(); right_foot_forward();
                msleep(800L);

/*              set_servo_pul(left_hip, left_hip_center_pos, 255, 255, 10);
                set_servo_pul(left_foot, left_foot_center_pos, 255, 255, 10);
*/
                set_servo_pul(right_hip, right_hip_center_pos, 255, 255, 10);
                set_servo_pul(right_foot, right_foot_center_pos, 255, 255, 10);
                msleep(100L);

                set_servo_pul(left_hip, left_hip_crouch_pos, 255, 255, 255);
                set_servo_pul(left_foot, left_foot_crouch_pos, 255, 255, 255);
                set_servo_pul(right_hip, right_hip_max_pos, 255, 255, 255);
                set_servo_pul(right_foot, right_foot_center_pos, 255, 255, 255);
                msleep(250L);

        position = 0;
        }
/*************************************************************************
****/
void twist_right()
        {
                weight_right();
                left_hip_back(); left_foot_forward();
                right_hip_back(); right_foot_forward();
                msleep(800L);

                set_servo_pul(left_hip, left_hip_center_pos, 255, 255, 5);
```

```
                set_servo_pul(left_foot, left_foot_center_pos, 255, 255, 5);
                set_servo_pul(right_hip, right_hip_center_pos, 255, 255, 5);
                set_servo_pul(right_foot, right_foot_center_pos, 255, 255, 5);
                msleep(50L);

                set_servo_pul(left_hip, left_hip_min_pos, 255, 255, 255);
                set_servo_pul(left_foot, left_foot_center_pos, 255, 255, 255);
                set_servo_pul(right_hip, right_hip_crouch_pos, 255, 255, 255);
                set_servo_pul(right_hip, right_foot_crouch_pos, 255, 255, 255);
                msleep(250L);

        position = 1;
        }
/*************************************************************************
****/
void short_step_forward_left()
        {
weight_right();                                 msleep(500L);
                left_leg_contract();                    msleep(125L);
                left_leg_crouch();      right_leg_crouch();     msleep(300L);
        position = 2;
        }
/*************************************************************************
****/
void short_step_forward_right()
        {
weight_left();                                  msleep(500L);
                                        right_leg_contract();   msleep(125L);
                left_leg_crouch();      right_leg_crouch();     msleep(300L);
        position = 2;
        }
/*************************************************************************
****/
void step_backward_left()
        {
weight_right();                                 msleep(500L);
                left_hip_up();                  msleep(100L);
                left_foot_center();                     msleep(50L);
                left_leg_back();        right_leg_crouch();     msleep(200L);
        position = 1;
        }
/*************************************************************************
****/
void step_backward_right()
        {
```

```
weight_left();                                  msleep(500L);
                                        right_hip_up();         msleep(100L);
                                        right_foot_center();    msleep(50L);
                left_leg_crouch();      right_leg_back();       msleep(200L);
        position = 0;
        }
/*************************************************************************
****/
void short_step_backward_left()
        {
weight_right(); left_leg_extend();                      msleep(500L);
                left_hip_up();                          msleep(100L);
                left_foot_center();                     msleep(50L);
                left_leg_back();        right_leg_back();     msleep(400L);
                left_leg_crouch();      right_leg_crouch();     msleep(200L);
        position = 2;
        }

/*************************************************************************
****/
void short_step_backward_right()
        {
weight_left();                  right_leg_extend();     msleep(500L);
                                        right_hip_up();         msleep(100L);
                                        right_foot_center();    msleep(50L);
                left_leg_back();        right_leg_back();       msleep(400L);
                left_leg_crouch();      right_leg_crouch();     msleep(200L);
        position = 2;
        }
/*************************************************************************
****/
void weight_swing_left()
        {
        weight_left();  msleep(350L);
        }
/*************************************************************************
****/
void weight_swing_center()
        {
        weight_center();        msleep(200L);
        }
/*************************************************************************
****/
void weight_swing_right()
        {
```

```
        weight_right();   msleep(350L);
        }
/*************************************************************************
****/
void dome_turn_left()
        {
        dome_left();    msleep(500L);
        }
/*************************************************************************
****/
void dome_turn_center()
        {
        dome_center();  msleep(300L);
        }
/*************************************************************************
****/

void dome_turn_right()
        {
        dome_right();   msleep(500L);
        }
/*************************************************************************
****
        Coordinated step and turn routines
*************************************************************************
***/
void step_forward()
        {
        if (position == 0 )     step_forward_right();
        else                    step_forward_left();
        }
/*************************************************************************
****/
void short_step_forward()
        {
        if (position == 0 )     short_step_forward_right();
        else                    short_step_forward_left();
        }
/*************************************************************************
****/
void step_left()
        {
        if (position == 0)
                {
                short_step_backward_left();
```

```
                msleep(500L);
                twist_left();
                }
        else
                {
                short_step_backward_right();
                msleep(500L);
                twist_left();
                }
        }
/*************************************************************************
****/
void step_right()
        {
        if (position == 0)
                {
                short_step_backward_left();
                msleep(500L);
                twist_right();
                }
        else
                {
                short_step_backward_right();
                msleep(500L);
                twist_right();
                }
        }
/*************************************************************************
****/
void step_backward()
        {
        if (position == 0 )     step_backward_left();
        else                    step_backward_right();
        }
/*************************************************************************
****/
void short_step_backward()
        {
        if (position == 0 )     short_step_backward_left();
        else                    short_step_backward_right();
        }
```

Orb.c

```
/*****************************************************************************
****
        Variables and Constants
*****************************************************************************
***/

int left_dome_ir;
int right_dome_ir;
int left_foot_ir;
int right_foot_ir;

int left_dome_ir_adjust = 0;
int right_dome_ir_adjust = 0;
int left_foot_ir_adjust = 0;
int right_foot_ir_adjust = 0;

int foot_threshold = 120;
int dome_threshold = 120;

int left_glance;
int right_glance;

/*****************************************************************************
****
        Functions
*****************************************************************************
***/


void main()
        {
        servos_on();
        poke(0x7000,0xff);
        start_process(avoid_obstacles(),1);
        }


void read_sensors()
        {
        left_dome_ir = analog(0) - left_dome_ir_adjust;
        right_dome_ir = analog(1) - right_dome_ir_adjust;
        left_foot_ir = analog(2) - left_foot_ir_adjust;
        right_foot_ir = analog(3) - right_foot_ir_adjust;
        }
```

```
void avoid_obstacles()
	{
	dome_turn_center();
	while(1)
		{
		read_sensors();
		if (left_dome_ir > dome_threshold || right_dome_ir > dome_threshold
		|| left_foot_ir > foot_threshold || right_foot_ir > foot_threshold)
			{
			dome_turn_left();
			read_sensors();
			left_glance = (left_dome_ir + right_dome_ir)/2;
			dome_turn_right();
			msleep(300L);
			read_sensors();
			right_glance = (left_dome_ir + right_dome_ir)/2;
			dome_turn_center();

			if (left_glance > right_glance) {  short_step_forward();
twist_right(); }
			else {  short_step_forward(); twist_left(); }
			}
		else step_forward();
		}
	}
```