Intelligent Machines Design Lab
EEL5666

Final Report

"Shaggy"

Brian R. Harris
July 31, 1998

Table of Contents

A. Abstract

Successful completion of EEL5666 requires an autonomous robot that has integrated behaviors and sensors. Shaggy uses IR sensors and sonar to find and differentiate between golf balls and tennis balls. After collecting these objects, Shaggy will deposit them in a central location denoted by a sonar transmitting beacon.

B. Executive Summary

Successful completion of EEL5666 requires an autonomous robot that has integrated behaviors and sensors. Shaggy uses IR sensors and sonar to find and differentiate between golf balls and tennis balls. After collecting these objects, Shaggy will deposit them in a central location denoted by a sonar transmitting beacon.

Shaggy is based on the TalrikII platform. A collection bay is mounted on the front of the robot to trap balls. A gated arm, controlled by a servo, moves up and down to trap and release the balls.

Shaggy has a multilevel breakbeam sensor on the front of the collection bay. This sensor is based on IR detection and the two height levels of the of the IR sensors exploit the differences in height between a golf ball and tennis ball. When a ball is detected, the collection mechanism is triggered, and the type of ball is assessed.

The robot must take the balls to a beacon. The beacon is marked by a 40kHz sonar transmitter. The robot has a sonar receiver and uses the signal for direction and distance measurements to find the beacon. The sonar is based on the Maxim 266 filter chip.

C. Introduction

Successful completion of EEL5666 requires an autonomous robot that has integrated behaviors and sensors. Shaggy uses IR sensors and sonar to find and differentiate between golf balls and tennis balls. After collecting these objects, Shaggy will deposit them in a central location denoted by a sonar transmitting beacon.

Shaggy is based on the TalrikII platform. A collection bay is mounted on the front of the robot to trap balls. A gated arm, controlled by a servo, moves up and down to trap and release the balls. This gate implements the behavior of ball collection and deposition.

Shaggy has a multilevel breakbeam sensor on the front of the collection bay. This sensor is based on IR detection and the two height levels of the of the IR sensors exploit the differences in height between a golf ball and tennis ball. The LEDs are mounted across from the IR sensors. When the beam is broken, a ball has been detected. After a ball is detected, the collection mechanism is triggered, and the type of ball is assessed. This represents two more behaviors: ball detection and ball differentiation.

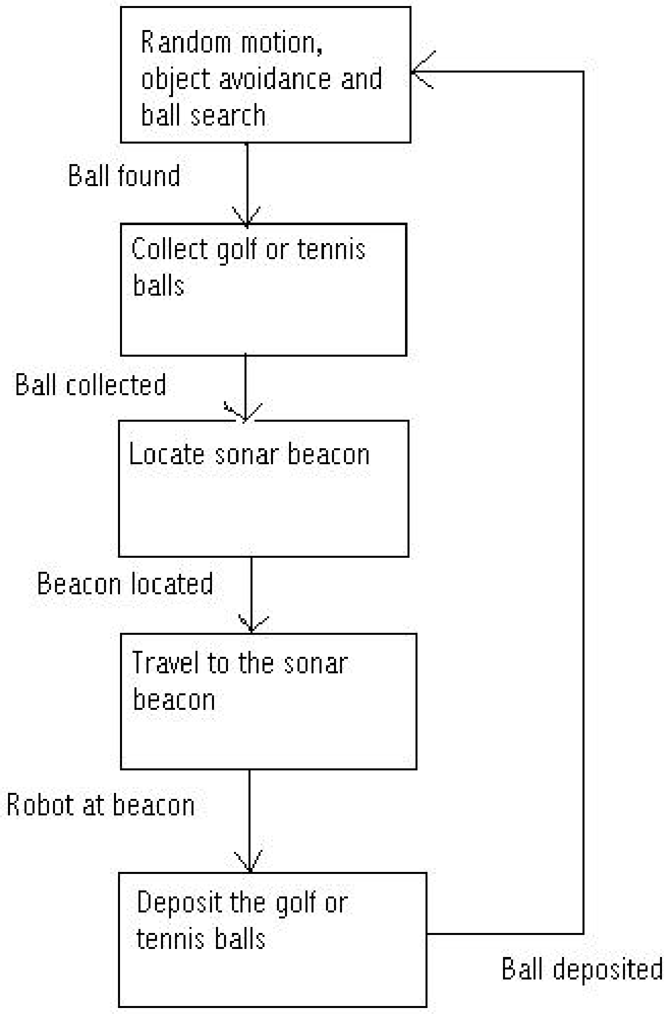
The robot must take the balls to a beacon. The beacon is marked by a 40kHz sonar transmitter. The robot has a sonar receiver and uses the signal for direction and distance measurements to find the beacon. The sonar is based on the Maxim 266 filter chip. Sonar works by sending and receiving sound at particular frequencies. A receiver can be designed by using a high quality audio filter. The sonar implements two more behaviors: beacon detection and beacon location.

Finally, code must be written to integrate all these behaviors.

D. Integrated System

My robot will travel around an area and collect golf balls and tennis balls. The robot will then deposit these objects in a central location.

The robot uses a multi-level break beam sensor mounted at the end of the arms of the robot to detect golf and tennis balls. The robot has a collection bay and a movable arm that raises and lowers to collect and deposit the balls. Shaggy finds a central location to deposit its balls by using sonar. A receiver is mounted on the robot and a transmitter station marks the beacon and deposition area.


Random motion, object avoidance and ball search
Ball found
Collect golf or tennis balls
Ball collected
Locate sonar beacon
Beacon located
Travel to the sonar beacon
Robot at beacon
Deposit the golf or tennis balls
Ball deposited

This flowchart shows the simple operation of the system. The robot begins with random motion around a given area while looking for a ball. When it detects a ball, it collects the object. The robot then looks for a sonar beacon and travels toward the beacon. After finding the beacon the robot will deposit the ball and begin the process again. The only factor the chart does not reflect is object avoidance. This behavior must be integrated into the program, and it will be done with some difficulty. Several of the operations such as traveling to the sonar beacon and finding the direction of the sonar beacon must begin again if motor control shifts from that module to object avoidance.

E. Mobile Platform and Actuation

The robot is based on the Talrik^{II} platform. This platform provides a broad wheel base, and it is large enough to allow a collection bay to be mounted in the front of the robot.

Wheels

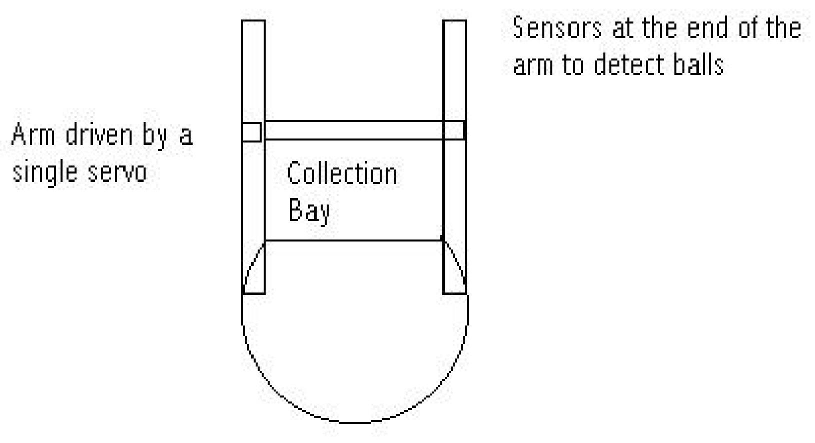
The robot uses two hacked servos to provide actuation for the wheels.

Collection Bay

The collection bay is a custom made design mounted on the front of the robot, and it is a substantial change to the basic circular platform. It consists of two arms extending from the wheel wells of the robot. Castors are mounted at the front of the arms to support the additional weight. A moving arm, made from a single piece of pine wood, is attached across the extensions. The arm forms the collection bay of the robot, and it opens and closes and collected and deposit balls. The arm is driven by the same servos that control the wheels of the robot. However, this is truly a servo; it is not hacked like the robot's motors.

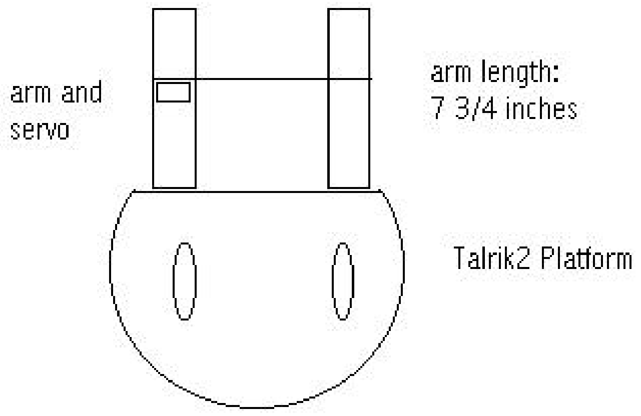
I considered several other design before implementing this option. I considered a brush the would constantly rotate and push the balls into the collection bay. This design would be difficult to implement because of the height difference between the golf and tennis balls. I would also be concerned about that assembly pushing the balls away from the robot. The robot would be operating on a hard, slick floor, and it would be difficult for the brush to grip the balls and suck them into the collection bay.

Another design I considered was a gate in the front of the robot. The gate would be driven by a single servo and would raise and lower to collect and deposit the objects. This seemed feasible but difficult to implement because of all the moving parts. Specifically, the problem of the gate jamming in its rails was a concern. If this problem occurred it would be difficult to solve while the robot was operating and would essentially disable the robot from completing its tasks. Another problem was determining if the gate was open or closed. A piece of string would connect the servo to the gate. This would provide no sure way for exact operation of closing and opening the gate, and a malfunction would again disable the robot.


Sensors at the end of the arm to detect balls
Arm driven by a single servo
Collection Bay

This figure is my first design of the collection bay. The arms, mounted by the sides of the robot at the wheel bays, are approximately 14 inches long. This design looked great, but functionally it was horrible. The arms were too long and the castors, mounted at the very ends of the arms, put too much weight away from the center of the robot. The robot had trouble traveling straight. Also, when the robot rotated (as in the program to find the sonar beacon) its motion was very unwieldy. The wheels slipped, and the robot could not longer stop at a precise location. Also, the weight at the end of the arms made the robot tend to want to rotate at the ends of the arms instead of at the center of the circular body.

With these problems, my algorithm for finding and traveling to the sonar beacon no longer worked. The collection bay had to be redesigned.



My final collection bay design incorporates many of the lessons I learned from the previous attempt. The arms are mounted closer together, and at 7 ¾ inches they are almost half the length of the previous 14 inch arms. The new design places the castors approximately in the middle of the arms instead of at the end of the arms (as in the previous design).

The handling characteristics of this design are much, much better than the previous attempt. And, the handling characteristics of the robot with these arms are similar to the handling characteristics before the collection bay was added to the robot. My previously written sonar algorithms work well with this new collection bay, and the bay is still big enough to find and collect golf and tennis balls.

F. Sensors

The robot uses three distinct sensor systems. IR sensors provide simple object avoidance. A second system is a multilevel IR break beam sensor that is mounted in front of the collection bay. This system will detect golf balls and tennis balls for the robot to collect, and it will differentiate between the two types of objects. Finally, the robot will use a sonar sensor to take the balls to a central location. A sonar transmitter beacon will mark the central location. The robot will have a receiver that will allow it to find the station and travel to its location.

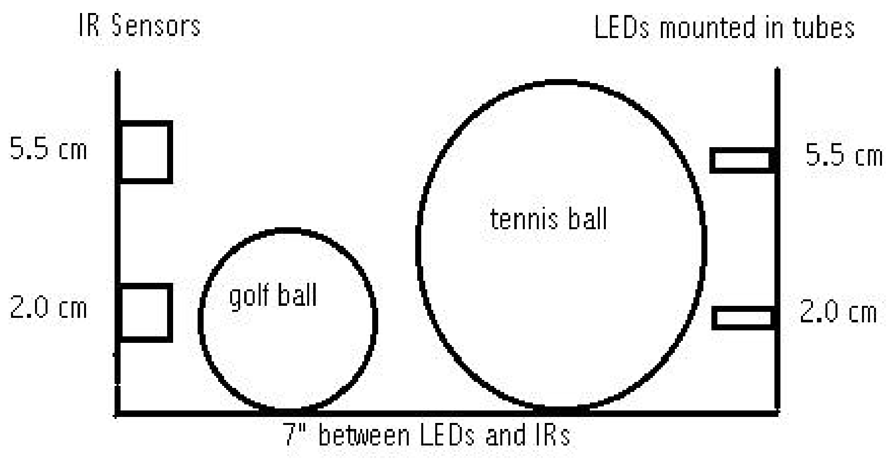
1. Object Avoidance

Object avoidance is simply based on Sharp IR sensors and LEDs oscillating at 40kHz.

2. IR Break Beam Sensor

System Overview and Diagram

My sensor system will use a two-level IR break beam to differentiate between golf balls and tennis balls. The IRs will be mounted at two different level and will be mounted across from their LEDs. The IR sensors will continuously receive a strong signal. When a ball rolls between the sensors the beam will be broken and the IRs will no longer receive a strong signal. A tennis ball will break both beams while a golf ball will only break the lower beam.



Sensor Positioning and Data

The sensors are positioned seven inches from the LEDs. This will be the width of the collection bay. The IRs sensors are at 2.0 cm and 5.5 cm heights. The LEDs are positioned directly across from the IR sensors.

The LEDs are mounted in cylinders to try to control the direction of the beam. To effectively implement this design the top IR must only receive a signal from the top LED and the bottom IR must only receive a signal from the bottom LED. I will discuss this problem more later in the report.

I took several data readings to determine the feasibility of this design. *The tennis ball and golf ball readings are with the balls positioned at random places between the sensors.* The IR sensors and the LEDs are mounted seven inches apart for all readings.

The IR sensors are connected to an A/D port (port E) on the MC68HC11. The IR sensors give values between 85 (no LED detection) and 130 (full strength LED detection).

<u>IR Sensors at 5.5 cm and 2.0 cm with an LED at 5.5 cm</u>

IR sensor at 5.5 cm	with Tennis Ball	with Golf Ball
129	85	129
129	85	129
129	85	129
129	85	129
129	85	129
IR sensor at 2.0 cm	with Tennis Ball	with Golf Ball
126	86	118
125	86	125
125	87	87
125	88	88
126	88	87

Ideally, in this setting, the bottom IR sensor should always read 85 (no LED detection), and the top IR sensor will receive a full reading until blocked by a tennis ball.

Practically, the bottom IR sensor receives substantial interference from the top LED even though the LEDs are shielded in tubes. This accounts for the full readings when no tennis or golf ball is present and the sporadic reading when a golf ball is present. The variation in the bottom IR sensor when the golf ball is present is due to the positioning of the ball. When the golf ball is closer to the LED it is not high enough to block the light and the bottom IR receives a stronger reading. When the golf ball is closer to the IR sensors then it shields the bottom IR sensor from the LED and the sensor registers a lower value.

The top IR sensor provides the desired results. The sensor receives a full reading until blocked by a tennis ball. The position of the tennis ball in the beam doesn't affect the readings. The sensor is unaffected by the presence of a golf ball in the beam.

<u>IR Sensors at 5.5 cm and 2.0 cm with an LED at 2.0 cm</u>

IR sensor at 5.5 cm	with Tennis Ball	with Golf Ball
123	85	84
124	84	88
124	84	118
124	85	86
124	84	116
IR sensor at 2.0 cm	with Tennis Ball	with Golf Ball
130	86	86
130	88	86
130	89	89
130	88	89
130	86	88

In this set of readings we would expect the bottom IR sensor to receive a full reading until the beam is broken by a golf ball or tennis ball. The top IR should register a minimum reading at all times.

The bottom sensor received a maximum reading when no ball was in the beam. A golf ball or tennis ball broke the beam and the IR sensor received a minimum reading. The position of the ball within the beam did not effect the value of the readings. This was as desired.

The top sensor received interference from the bottom sensor. The sensor received a near maximum value with no ball in the beam. A tennis ball completely blocked the beam. A golf ball gave sporadic values. When the golf ball was toward the LED sensor it shielded the beam emanating from the LED and the top IR sensor received a small reading. When the golf ball was closer to the IR sensor the beam wasn't blocked and the top IR received a higher reading, similar to the initial readings when no ball was in the beam.

<u>IR Sensors at 5.5 cm and 2.0 cm and LEDs at 5.5 cm and 2.0 cm</u>

IR sensor at 5.5 cm	with Tennis Ball	with Golf Ball
129	86	129
129	86	129
129	86	129
129	86	129

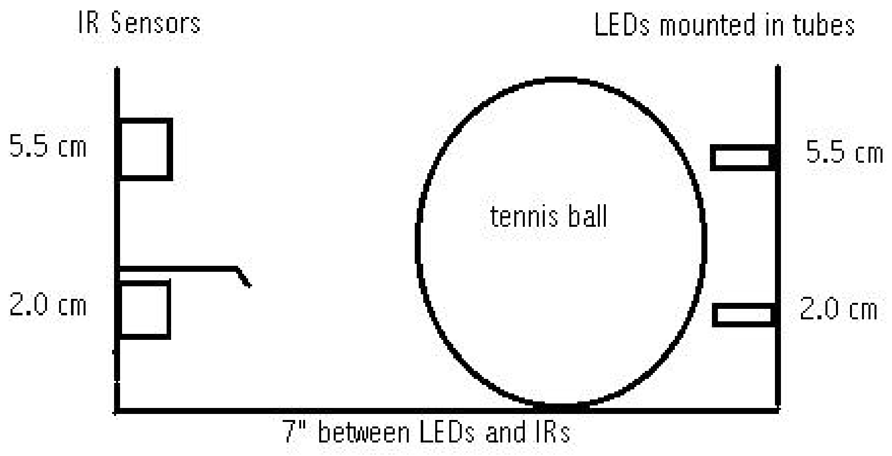
129	87	129
IR sensor at 2.0 cm	with Tennis Ball	with Golf Ball
129	87	112
130	87	115
130	93	90
130	90	89
130	89	111

This is the first trial with the whole system. The top sensor worked as expected. The top IR received a full reading with no balls in the beam. A golf ball did not break the beam and the tennis ball broke the beam without regard to its positioning.

The bottom sensor received a full reading with no ball in the beam. A tennis ball broke the beam with only a slight variation in readings. The golf ball, however, provided spurious results. The high readings are due to interference from the top LED. When the golf ball is close to the IR sensors it shields it from both LEDs and the bottom IR registers minimal values. When the golf ball is toward the LEDs then the top LED gives substantial interference to the bottom IR sensor.

<u>IR Sensors at 5.5 cm and 2.0 cm and LEDs at 5.5 cm and 2.0 cm with a 2.5 cm overhanging shield</u>

To get the sensors to work correctly, the bottom IR sensor must be shielded from the top LED. This is my first trial with a simple physical shield that extends approximately 2.5 cm out from the IR sensor like an awning.



IR sensor at 5.5 cm	with Tennis Ball	with Golf Ball
129	89	129
129	87	129
129	86	129
129	86	129
129	86	129
IR sensor at 2.0 cm	with Tennis Ball	with Golf Ball
130	86	114
130	87	110
130	88	93
130	87	95
130	87	116

(IR Sensors at 5.5 cm and 2.0 cm and LEDs at 5.5 cm and 2.0 cm
with a 2.5 cm overhanging shield)

I received similar results to the last test without the shield. The top LED still interfered with the bottom IR sensor when a golf ball was in the beam. My next trial increases the length of the shield.

<u>IR Sensors at 5.5 cm and 2.0 cm and LEDs at 5.5 cm and 2.0 cm
with a 3.5 cm overhanging shield</u>

IR sensor at 5.5 cm	with Tennis Ball	with Golf Ball
n/a	n/a	n/a
n/a	n/a	n/a
n/a	n/a	n/a
n/a	n/a	n/a
n/a	n/a	n/a
IR sensor at 2.0 cm	with Tennis Ball	with Golf Ball
130	n/a	91
130	n/a	91
130	n/a	89
130	n/a	92
130	n/a	96

In this final test I extended the length of the shield, and I retested the pertinent readings. I received the desired results. The bottom IR sensor still received a full reading when no ball was in the beam. When a golf ball was in the beam the shield adequately protected the bottom IR from interference from the top LED.

3. Sonar

My sonar design is based on Michael Apodaca's circuit from the Spring 1998 semester. The design is based on the Maxim 266 filter chip.

The output of the filter chip is a sine wave, and this is not suitable for inputting into the HC11's analog ports. However, by filtering out the DC component the chip will produce analog values between 115 (for no sonar detection) and 215 (for full strength detection). A simple circuit to strip the DC component can be made by using a diode, 47ohm resistor and a 220 microfarad capacitor in series. The sine wave is fed through the diode and the resulting DC signal is produced at the resistor and capacitor junction.

G. Behaviors

A successful robot has at least four behaviors, and my robot is no exception. Its first behavior is object avoidance; this is a common behavior among almost all our robots. The next behaviors are all associated with collecting and depositing golf and tennis balls. These behaviors are: 2) detect a golf ball or tennis ball, 3) collect the ball, 4) find the sonar beacon, 5) travel to the beacon and 6) deposit the ball.

Several interesting lessons were learned in implementing the seemingly simple ball collection behavior. The code for this is simply: stop, open the bay, drive forward and close the bay. This code does not work. Anytime the gate is open and the robot begins to drive forward the gate automatically closes, and after this process the robot occasionally loses motor control. This occurs because the twoservo.icb, twoservo.c and motor driver routines all use the TOC1 register. I wish the thank Kelly Krempin (and his robotics partner Charles Feddeman) for providing me with a solution to this problem. This solution has two parts. First, the ground to the servo must come from the regulated power supply on the board and not from the battery pack like I had previously wired the circuit. Second, after using the servo the TOC1 registers must be cleared with the poke(0x1016,0xff); and poke(0x1017,0xff); commands. This will restore motor control to the robot while regulating the ground will prevent the arm from coming shut while the robot is moving. The only part of the problem that has yet to be correct is that while the robot is moving with the servo out of the zero degrees position the motors will only operate at one hundred percent power.

H. Conclusion

The conclusion to my project, hopefully, will be a successful demonstration on July 31, 1998. Many of my design goals and behaviors have been successfully implemented. If I had more time to work on the project I would spend time development more robust software code. I would like to rework obstacle avoidance into all parts of the program, and I would like to develop an improved sonar array for receiving the signal from the transmitter beacon.

I. References

Doty, Keith L., *Talrik^{II} Assembly Manual*, Mekatronix^{TM}, Gainesville, FL, 1997.

Martin, Fred G., *The 6.270 Robot Builder's Guide*, Cambridge, Massachusetts, 1992.

MEKATRONIX^{TM}, *Assembly Manual Mekatronix^{TM} ME11 Expansion Board for the MC68HC11 EVBU*, MEKATRONIX^{TM}, Gainesville, FL, 1997.

Appendix A: Source Code

```
/*      Brian R. Harris                  */
/*      EEL5666                          */
/*      Intelligent Machine Design Lab  */
/*      Robot: Shaggy

/*      Final Code Development Module   */
/*      Created: 6/3/98                 */
/*      Last Modified: 7/30/98          */

/*
     SHAGGY is an autonomous robot that is build to collect
     golf balls and tennis balls and then to deposit them
     in a central location
*/

/*      This program must have the twoservo.icb and twoservo.c libraries
        loaded to run the servo commands
*/

/*
     In this program the robot travels randomly avoiding objects until
        it detects a ball.  Then the robot collects the ball.  Next,
        the robot will use sonar to find the home beacon and travel
        to it.

        Note:  Many of the sleep(); statements in this program are
        for demonstation and debugging purposes.  They are used to
        illustrate where in the code the program is currently
        executing.
*/

/*
        This program suffers from an abnormality in the IC software.
     The motor drivers and the servo program produce unexepcted results
     when they operate simultaneously.  I've been told this is
     because they both use OC1.  It is possible to correct part
     of this problem in software, and I will try to do just that.
*/

/*
     This code includes no obstacle avoidance while the robot is
     travelling to the sonar beacon.  The deposit ball routine is
     not implemented.  Also, the code is not continuously looped, and
     I think I will leave the robot performing only one cycle for
     demo purposes.  Making the change in the program for continuous
     operation is trivial.
*/

/*      HARDWARE INSTALLATION NOTES     */
/*      servo_deg2() = pin 31;          */


/*      GLOBAL VARIABLES AND CONSTANTS  */

/*      CONSTANTS       */
int     left_motor = 0;
int     rt_motor = 1;
int     beacon_thres1 = 135;
int     BEACON_FOUND = 205;

/*      INPUT PINS      */
int     left_ir_pin = 0;
int     rt_ir_pin = 1;
int     tennis_ir_pin = 2;
int     golf_ir_pin = 3;
int     sonar_pin = 7;


/*      VARIABLES       */
int     left_ir_val;
int     rt_ir_val;
int     tennis_ir_val;
int     golf_ir_val;
int     sonar_val;

int     acquired_beacon;
int     peakvalue;
int     i;
int     beaconinrange;
int     atbeacon;
int     initial;
int     tennistemp;
int     temp;
int     ball_collected = 0;
int     ball_found = 0;
int     obj_to_avoid = 0;


void main(void)
{
      /* Initialize the robot */
      robo_init();

      /* Start the robot moving forward */
      forward();

      /* A continuous loop for random motion until a
           ball is detected
      */
      while(ball_collected == 0)
      {
            /* This function will read the object avoidance IR sensors
                   and set the flag obj_to_avoid = 1 if it finds an object
                   the robot needs to avoid.
            */
            read_sensors();

            /* If a ball is found then collect the ball. */
            if(ball_found > 0)
            {
                  collect();
                  ball_collected = 1;
            }

            /* Otherwise just do obstacle avoidance. */
            if (obj_to_avoid > 0)
            {
                  object_avoid();
                  obj_to_avoid = 0;
            }
      }

        /* Obstacle avoidance stops here! */

        /*
      servo_off();
      sleep(2.0);
        */

        /* Find the beacon's direction */
      beacon_dir();
```

```
        /* Goto the beacon */
      sleep(3.0);
      gotobeacon();
}

void open(void)
{
      /* raise the gate */
      servo_on();
      sleep(0.3);
      servo_deg2(90.0);
      sleep(0.3);
/*        servo_off(); */

        /* Clear the OC1 register.  This is an attempted
           software fix of IC's problems when operating
           the servos and the motors together.
        */
/*        poke(0x1016,0x00);
        poke(0x1017,0x00);
*/
      return;
}

void close(void)
{
        /* Close the gate. */

        servo_on();
        sleep(0.3);
      servo_deg2(0.0);
        sleep(0.3);
      servo_off();

        /* Clear the OC1 register.  This is an attempted
           software fix to IC's problems when operating
           the servos and the motors
        */
        poke(0x1016,0x00);
        poke(0x1017,0x00);


      return;
}

void collect(void)
{
        /* Stop the motors if they are not already stopped. */
      mstop();
      sleep(3.0);

        /* Raise the gate. */
      open();
      sleep(3.0);

        /* Get the ball in the bay by moving forware for one
           second.  At this point IC does not have speed control
           of the motors.  The robot should go forward at full
           speed.  This is a bug in the IC libraries.
        */
      sforward();
        sleep(0.5);
      mstop();

        /* The ball is in the gate.  Now close the gate. */
      sleep(1.0);
      close();
        sleep(1.0);

}

void robo_init(void)
{
        /* Start IR LEDs */
      poke(0x7000,0xff);

        /* Turn off the ball detection IR sensors. */
        poke(0x6000,0x00);

        /* Close the gate (because it might be open). */
        close();

      return;
}

void sforward(void)
{
        /* Original intention: start the robot moving slowly.
           Current: Full speed ahead.
        */
        motor(left_motor,30.0);
        motor(rt_motor,30.0);

      return;
}

void mstop(void)
{
        /* Stop the robot. */
      motor(left_motor,0.0);
      motor(rt_motor,0.0);

      return;
}

void forward(void)
{
        /* Start the robot moving slowly at first to prevent the
           wheels from slipping because of the weight of the
           collection bay on the front of the robot.
        */
        motor(left_motor,40.0);
        motor(rt_motor,30.0);
      sleep(1.0);

        /* Now, let the robot move at its maximum speed. */
        motor(left_motor,80.0);
        motor(rt_motor,70.0);

      return;
}

void turn_left(void)
{
        /* Stop the motors. */
      motor(left_motor,0.0);
      motor(rt_motor,0.0);
      sleep(0.2);

        /* Turn for 1/2 second. */
      motor(left_motor,-35.0);
      motor(rt_motor,35.0);
      sleep(.5);
```

```
        /* Stop the motors. */
      motor(left_motor,0.0);
      motor(rt_motor,0.0);
      sleep(0.2);

      return;
}

void turn_right(void)
{
        /* Stop the motors. */
      motor(left_motor,0.0);
      motor(rt_motor,0.0);
      sleep(0.2);

        /* Turn for 1/2 second. */
      motor(left_motor,35.0);
      motor(rt_motor,-35.0);
      sleep(.5);

        /* Stop the motors. */
      motor(left_motor,0.0);
      motor(rt_motor,0.0);
      sleep(0.2);

      return;
}

void object_avoid(void)
{

                /* Look for a left object.  If the object is found
                   then turn right.
                */
            if( left_ir_val > 105 )
            {
                  turn_right();
                  sleep(.2);
                  forward();
            } /* end if */

                /* Else look for right object.  If the object is
                   found then turn right.
                */
            else if ( rt_ir_val > 105 )
            {
                  turn_left();
                  sleep(.2);
                  forward();
            } /* end else if */

            return;
}


void m_stop(void)
{
        /* This is another routine to stop the motors. */

        /* Stop the motors. */
      motor(left_motor,0.0);
      motor(rt_motor,0.0);
      sleep(1.0);

      return;
}
```

```
void rstop(void)
{
        /* This function stops the robot's motors when it is rotating.
           The robot will move slightly back to the right to account
           for imprecise stopping because of the weight and momentum
           of the robot when it rotates.
        */

        /* Give the motors a slight reverse. */
        motor(left_motor,10.0);
        motor(rt_motor,-10.0);
        sleep(0.5);

        /* Now, stop the motors rotating. */
      motor(left_motor,0.0);
      motor(rt_motor,0.0);
      sleep(0.5);
}

void    rotateleft(void)
{
        /* Start the robot turning to the left. */
      motor(left_motor,-55.0);
      motor(rt_motor,55.0);

      return;
}

void    slow_rotateleft(void)
{
        /* Start the robot turning to the left. */
      motor(left_motor,-35.0);
      motor(rt_motor,35.0);

      return;
}

void    findmax(void)
{
        /* Find the maximum sonar value in the robot's 360 degree
           rotation range.  The sonar reads analog values between
           approximately 115 and 210.
      */

      i = 0;
      peakvalue = 0;

        /* Rotate the robot. */
      rotateleft();

        /* While the robot is rotating take sonar readings and compare
           them against the peak value.

           The robot will spin for approximately
           160 x 0.025 seconds
        */
      while( i < 160 )
      {
                /* Take sonar readings. */
            readsonar();

            if( sonar_val > peakvalue )
            {
                  peakvalue = sonar_val;
            }
            sleep(.025);
            i = i + 1;
      }

        /* Stop the robot from rotating.  This function should
           probably be rstop(); instead of m_stop();
        */
      m_stop();
      sleep(3.0);

        /* If the sonar has a value approximately 10 points greater
           than its minimum input value a sonar beacon has been
            found.  Set the flag beaconinrange.
        */
      if( peakvalue > 125 )
      {
            beaconinrange = 1;
      }

}

void begin(void)
{
      /* The robot is already pointing at the beacon
           and this function allows the robot to initially
           travel toward the beacon for a period of time
      */

        /* If the robot has found the beacon then stop execution. */
      if ( peakvalue > ( BEACON_FOUND - 1 ) )
      {
            return;
      }

        /* If the robot is not close enough to the beacon then start
           moving toward the beacon.
        */
      forward();

        /* The robot will move toward the beacon for two seconds
           regardless of what the sonar value reads during this
           time.
        */
      if (peakvalue < 160)
      {
            sleep(2.0);
      }
      else if ( peakvalue < 180 )
      {
            sleep(2.0);
      }
      else
      {
            /* don't sleep at all */
      }
}

void readsonar()
{
        /* This program read the sonar receiver and returns a
           single value for a reading.  It, however, takes multiple
           readings and averages them to prevent anomolous readings
           from influencing the robot.
        */

        /* Initialize the variable. */
      sonar_val = 0;
```

```
          /* Sum the inputs. */
       sonar_val = analog(sonar_pin);
       sonar_val = sonar_val + analog(sonar_pin);
       sonar_val = sonar_val + analog(sonar_pin);
       sonar_val = sonar_val + analog(sonar_pin);

          /* Take the average. */
       sonar_val = sonar_val/4;

       return;
}

void gotobeacon(void)
{
          /* Initialize variables */
       atbeacon = 0;
       readsonar();
       peakvalue = sonar_val;

          /* Now work with the sensor input. */
       while( (atbeacon == 0) && (beaconinrange == 1) )
       {
               /* The beacon is found! */
               if (sonar_val > (BEACON_FOUND-1) )
               {
                       m_stop();
                       atbeacon = 1;
               }
               else
               {
                       /* Else the beacon is not found */

                       /* Travel toward beacon for a little bit */
                       begin();

                       /* sonar_val = analog(sonar_pin); */
                       readsonar();

                       if (sonar_val > peakvalue)
                       {
                               peakvalue = sonar_val;
                       }


                       /* Travel until the robot loses a strong sonar signal */
                       while(  (((sonar_val + 20) > peakvalue ) ||
                                (( sonar_val > 160 ) && ( sonar_val < BEACON_FOUND
))) &&
                                ( atbeacon == 0 )  )
                       {
                               /* Constantly take new sonar readings */
                               /* sonar_val = analog(sonar_pin); */
                               readsonar();
                               if (sonar_val > peakvalue)
                               {
                                       peakvalue = sonar_val;
                               }

                               /* The robot has found the beacon */
                               if(sonar_val >= BEACON_FOUND)
                               {
                                       atbeacon = 1;
                                       m_stop();
                               }
                       }
```

```
                if( atbeacon == 0 )
                {
                      /* Or the robot has lost the signal */

                      /* Stop */
                      m_stop();

                      /* Reacquire the beacon */
                      beacon_dir();

                      /* Set new initial setttings */
                      /* peakvalue = analog(sonar_pin); */
                      readsonar();
                      peakvalue = sonar_val;
                      atbeacon = 0;
                } /* end if */

                /* From this point the function begins the while loop
again */
          } /* end else */
     } /* end while */
} /* end gotobeacon() */

void beacon_dir(void)
{
     /* stop the robot's motors */
     m_stop();

     /* initialize variables */
     acquired_beacon = 0;
     beaconinrange = 0;

     findmax();

     /* find the direction of the beacon */
     if( beaconinrange == 1 )
     {
          sonar_val = analog(sonar_pin);

          /* Look for the beacon by rotating the robot */

          /* start robot rotating */
          slow_rotateleft();

          while( acquired_beacon == 0 )
          {
                /* take a sonar sensor reading */
                /* sonar_val = analog(sonar_pin); */
                readsonar();

                /* look for transmitter beacon */
                if( (sonar_val + 3) > peakvalue )
                {
                      rstop();
                      acquired_beacon = 1;
                } /* end if */
          } /* end while */
     } /* end if */
     else
     {
          /* do something else because the beacon hasn't been found */
     }

} /* end of beacon_dir */

void read_sensors(void)
{
	left_ir_val = analog(left_ir_pin);
	rt_ir_val = analog(rt_ir_pin);
	tennis_ir_val = analog(tennis_ir_pin);
	golf_ir_val = analog(golf_ir_pin);

	if(((left_ir_val > 105) || (rt_ir_val > 105)))
	{
		obj_to_avoid = 1;
	}

	if(golf_ir_val < 120)
	{
		ball_found = 1;
		    m_stop();

		    for(temp = 0; temp < 5; temp = temp + 1)
		    {
			    tennistemp = analog(tennis_ir_pin);

			    if(tennistemp < tennis_ir_val)
			    {
				    tennis_ir_val = tennistemp;
			    }

			    sleep(0.05);
		    }

		    /* A ball has been found.  Now differentiate between
		       a golf ball and a tennis ball.
		    */

		    /* If the tennis ball IR is also tripped then a tennis
		       ball has been found and light up both LEDs.  The
		       LEDs are connected on a MMIO at $6000.  Data lines
		       0 and 1 control the LEDs.
		    */
		    if(tennis_ir_val < 120)
		    {
			    poke(0x6000,0x03);
		    }

		    /* Else, the tennis ball sensor has not been tripped
		       and only one LED is lit to signify a golf ball.
		    */
		    else
		    {
			    poke(0x6000,0x01);
		    }

	}

	return;
}
```