Anti-Roach Terrorism Act

University of Florida

Department of Electrical and Computer Engineering

Intelligent Machines Design Laboratory

Zer Liang

Instructor: Keith L. Doty

December 12, 1997

TABLE OF CONTENTS

ABSTRACT

Legged beings are naturally adaptable, able to move over almost any terrain. Legged robots, in theory, would be more flexible over various terrains than wheeled ones. The purpose of the Anti-Roach Terrorism Act is to crack down on the various roaches that have been terrorizing neighborhood homes. The enforcer of the act would be a six-legged walking intelligence agent, code-named RAID. The objective of this agent is to use legs as a mean of locomotion, infiltrate enemy compounds and exterminate the guilty parties.

RAID was designed to seek out roaches or rats and spray them. RAID demonstrated 3 behaviors at the end of its contract: obstacle avoidance, bump detection and object following.

EXECUTIVE SUMMARY

RAID is a six-legged robot designed to seek out targets like roaches and exterminate it. The most basic and important thing about a walker is the leg design. I chose the pentograph mechanism because it was simple yet sturdy and effective. The platform was made out of aluminum for its durability over wood and relative lightness. For the robot to walk effectively, a good walking gait must be programmed. For this I used Jenny Laine's code. The code allows calibration of the legs, walking gaits, speed and direction.

As far as its behaviors go, the robot must be able to move in its surrounding before it can seek out the target. To do this, obstacle avoidance is needed. 40 kHz IR emitters and receivers were used for this behavior. To avoid skinny or unseen obstacles, bump switches were used for collision detection so the robot knows when to turn the other way. For seeking out its target, CDS cells were used. CDS detects the amount of light in the surrounding. When the cells are pointed at a dark object, less light is detected by the sensors. However, since CDS cells detects the amount of light in the area, shadows also give object-like readings. To alleviate this problem, the resistance of the CDS cell pointed at the desired object was measured in normal light. A pull-up resistor of the same resistance was used to keeping the readings in the middle of the A/D range.

INTRODUCTION

For an autonomous agent, I wanted one that was not too common. Legged robots came to mind. My idea of designing a robot is actually building everything from scratch, not assembling something pre-designed, so the decision to build my own robot was made. Of course, the robot has to perform a certain task. Chasing roaches and rats sounded like a nice idea so RAID was envisioned. It would do basic obstacle avoidance, seek out preys and kill it by spraying it.

Mobile Platform

The initial design of the platform was made from ¼" plywood. However, with the discovery of a chipped edge, different materials were considered. Upon seeing an optical mouse pad with slight bluish tint, the decision was made that the platform would be aluminum (the mouse pad). The aluminum is strong yet light, weighing about the same as the previous wooden platform, and far more attractive. The platform was cut to size by a metal shearing machine and the servo slots were nibbled to approximate dimension and hand filed to the correct size. The platform dimensions are shown in figure 1.

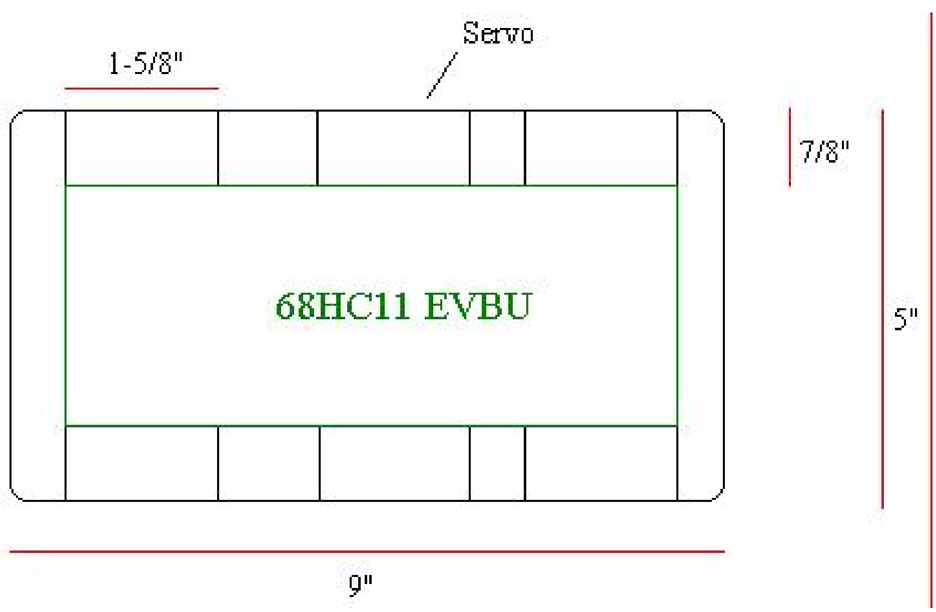

Figure 1. Platform dimensions

Actuation

Being a walking robot, legs are used as means of locomotion. The pentograph mechanism was chosen to be the design, as it is simple and reliable. The dimensions for the legs are as shown in figure 2.

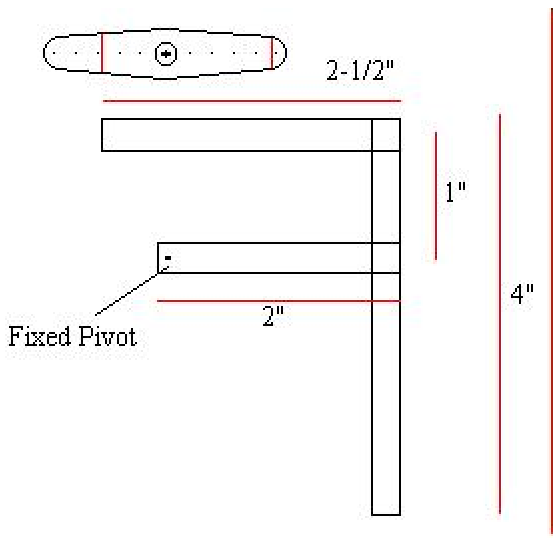

Figure 2. Leg dimensions

The basic static calculations for the dimensions shown above are as follows:

- The weight of the robot was approximately 45 oz.
- The servo has a torque rating of 42 oz/in.
- The weight on each leg using the tripod gate is approximately 45/3=15 oz.
- The weight able to be supported by each leg is F=T/L

$$F = \frac{42oz/in}{2in} = 21oz$$

As shown by the calculation, the robot is able to lift up its own weight.

The material for the legs was first chosen to be ¼” by 2-½” long metal bolts and 3/16” by 4” long threaded metal rod. However, due to the nature and size of the material, Ron Brown of the Aerospace Machines Laboratory suggested that the way to make the necessary cuts at the joints be by hand. The legs produced by this method were somewhat unstable, as the hand-eye coordination errors on the cuts proved to be costly. Also, due to the size of the material, 1/16” screws were used to connect the joints together, which tended to bend out of shape after some use. So a different material was chosen, ¼” aluminum block. The aluminum block is strong yet soft, making it very workable with a milling machine, which has an accuracy of 1/1000 of an inch if used properly. Also, the aluminum offered more overall surface area to work with for the same amount weight, thus allowing larger holes to be drilled without compromising the structure. More accurate cut, tighter fit and larger screw joints all contributed to a more stable leg design.

The swivel base, which joins the rotating and the lift servos, was made out of 3/64” aluminum sheet. The configuration is shown in figure 3.

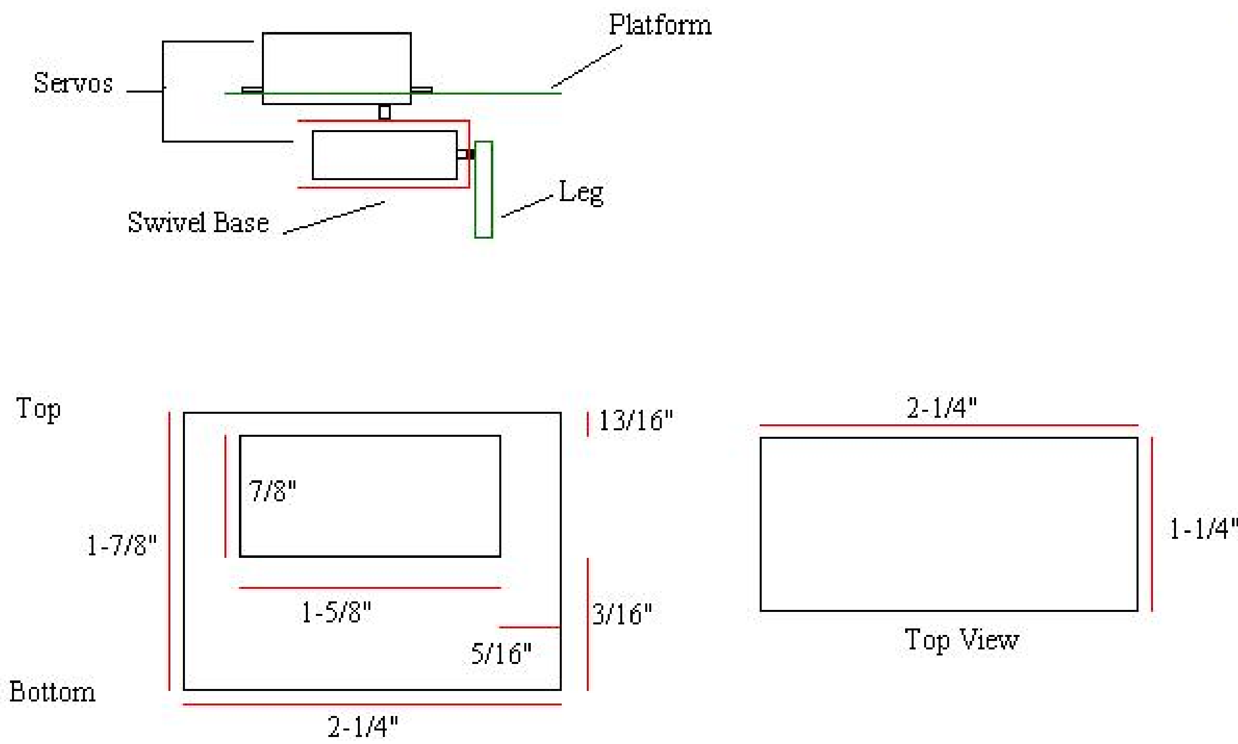

Figure 3. Swivel base configuration

The robot movement is controlled by Jenny Laine's code on a 68HC11 E9 EEPROM. With this walking algorithm, the robot is able to walk in four directions, left, right, forwards and backwards. Twenty five different speeds is also available with 01 being the fastest and 25 the slowest.

Sensors

IR Sensors

For the behaviors of obstacle avoidance, IR emitters and receivers were used. The IR emitters were modulated at 40kHz and the receivers were hacked 40 kHz Sharp IR receivers to provide analog readings. The IR emitters and receivers were mounted on the head of the robot at a slight angle outward. Readings in the lab environment were taken as shown in figure 1.

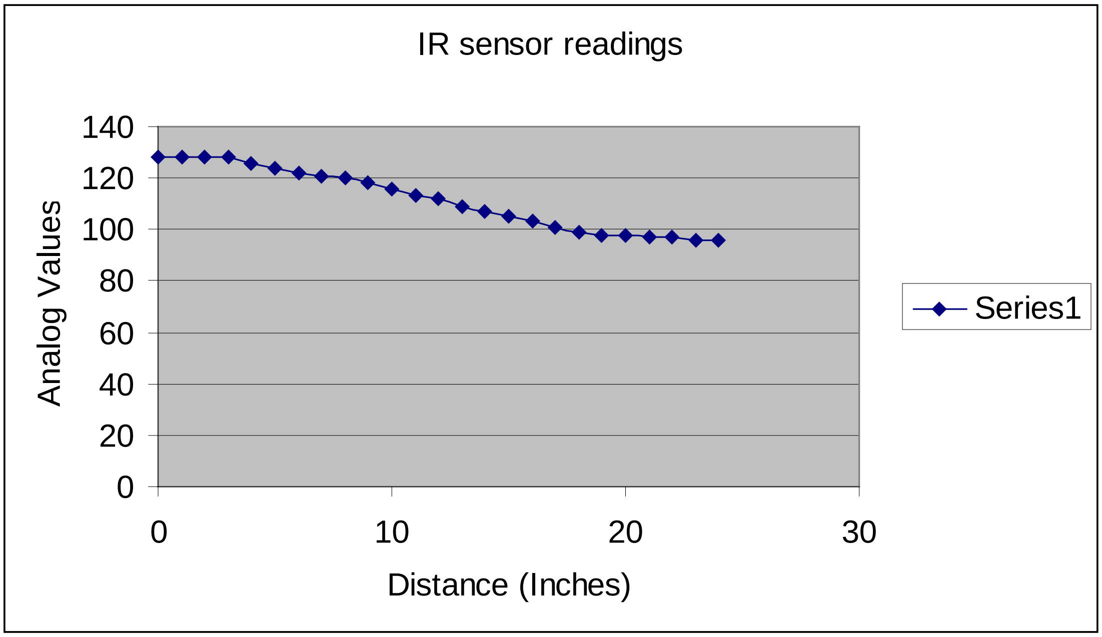

Figure 4. IR sensor readings.

The object avoidance algorithm is shown in the code in the appendix by comparing the left and right sensor readings to each other and the determined threshold values.

Bump Switches

For collision avoidance lever bump switches were used. Metal antennas were mounted onto the levers to provide an extension of the bump switch detection range. The code for collision avoidance is shown in the appendix.

Cadmium Sulfide Cells (CDS cells)

For roach detection, CDS cells were used. CDS cell detects darkness in its environment. With the right pull-up resistor value, dark object such as roaches can be detected in lighted areas. I used 22kΩ resistors for detecting a black mechanical mouse.

Behaviors

RAID demonstrated three behaviors, obstacle avoidance, collision detection, object following.

For obstacle avoidance, 2 IR emitters modulated at 40kHz and 2 Sharp 40kHz IR receivers were used. Two variables, near_ir_threshold and ir_threshold, were the threshold values used to determine its behavior. If the IR reading is greater than near_ir_threshold, the robot starts to veer away from the obstacle. Once the ir_threshold is reached, the robot turns away from the obstacle.

For collision detection, two lever bump switches were mounted on top of the bottom of the head. Iron hangers, acting as antennas, were used to extend the bump detection range of the bump switches. Similar behavior to the obstacle avoidance was used.

For object following, three CDS cells were mounted on top of the head. The cells were cumulated with about 1” heat shrink wrap. 22 kΩ pull-up resistors were used to put the normal condition in the middle of its detectable range. A black mechanical mouse was used as the object for the robot to follow. The mouse is detected by the robot in the range of 6-8 inches.

CONCLUSION

In designing this robot, I learned that metal work is very hard and tedious work. However, I did learn a lot about using power tools in machine labs.

If I can do this project over again, I would start by making the robot out of aluminum for it is light and durable. For the better part of the semester, my time was spent on designing and building the robot. However, with experience now, building a good platform would not take nearly as long therefore allowing more time for behavior enhancements. The design phase is a trial and error stage where prototypes are built to test out the design.

ACKNOWLEDGEMENTS

Thanks to Jenny Laine for her great walking code and also to Dave Novick for helping Jenny out with the configuring of the program for my robot configurations.

Thanks to Ron Brown, the machine lab operator at the Aerospace Engineering Building for letting me use the equipements necessary for building my robot.

Thanks to all the TA's and classmates for their advice and help.

APPENDIX

```
#include <hc11.h>
#include <mil.h>
#include <serial.h>
#include <analog.h>

#define TOC2_ISR object_spray
#pragma interrupt_handler object_spray

void delay();
void init_spray();
void ir_sensor();
void obstacle_avoid();
void object_follow();
void object_spray();
void arbitrate();
void read_sensor();

int left_ir, right_ir;
int left_cds, center_cds, right_cds;
int left_bump, right_bump;
int ir_threshold, near_ir_threshold, cds_threshold, near_cds_threshold;
int spray, turning_on, cyccnt;

void main()
{
   init_serial();
   init_analog();
/*    init_spray(); */

   ir_threshold=118;
   near_ir_threshold=110;
   cds_threshold=130;
   near_cds_threshold=125;

   spray=0, turning_on=0, cyccnt=0;

   while(1)
        {
          read_sensor();
     /*   ir_sensor();
          obstacle_avoid();
          object_follow();
```

```
            if(spray=1)
            {
              object_spray();
            }   */
            arbitrate();
          }
}

void read_sensor()
{
    left_ir=analog(0);
    right_ir=analog(1);
    left_cds=analog(2);
    center_cds=analog(3);
    right_cds=analog(4);
    left_bump=analog(5);
    right_bump=analog(6);
}

void ir_sensor()
{
    static int last_state;

    if(left_ir > ir_threshold && right_ir > ir_threshold)
    {
      write("04");
      if(left_ir > right_ir)
      {
          if(last_state != 1)
          {
            write("R");
            last_state=1;
            delay(); delay(); delay(); delay();
            write("02");
          }
      }
      else if(right_ir > left_ir)
      {
          if(last_state !=2)
          {
            write("L");
            last_state=2;
            delay(); delay(); delay(); delay();
            write("02");
          }
      }
```

```
  else if(right_ir == left_ir)
  {
      if(last_state !=3)
      {
        write("B");
        last_state=3;
        delay(); delay(); delay(); delay();

        if(left_ir > right_ir)
        {
          write("R");
          delay(); delay(); delay();
        }
        else if(right_ir > left_ir)
        {
          write("L");
          delay(); delay(); delay();
        }

        write("F");
        write("02");
      }
  }
}

else if(left_ir > near_ir_threshold && right_ir > near_ir_threshold)

{
  if(left_ir > right_ir)
  {
      if(last_state != 4)
      {
        write("R"); delay(); delay(); write("F"); delay(); delay();
        write("R"); delay(); delay(); write("F");
        last_state=4;
      }
  }
  else if(right_ir > left_ir)
  {
      if(last_state !=5)
      {
        write("L"); delay(); delay(); write("F"); delay(); delay();
        write("L"); delay(); delay(); write("F");
        last_state=5;
      }
  }
```

```
    }
    else
    {
      if(last_state !=6)
      {
          write("F");
          last_state=6;
      }
    }

}

void obstacle_avoid()
{
    if(left_bump < 127 || right_bump <127)
    {
      if(left_bump < 127 && right_bump < 127)
      {
          write("B");
          delay(); delay(); delay();
          write("F");
      }
      else if(left_bump <127)
      {
          write("B");
          delay();
          write("R");
      }
      else
      {
          write("B");
          delay();
          write("L");
      }
    }
    else ir_sensor();
}

void object_follow()
{
    if(left_cds > cds_threshold &&
       center_cds > cds_threshold &&
       right_cds > cds_threshold)
    {
      spray=1;
      write("01");
```

```
  if(left_cds > right_cds && left_cds > center_cds)
  {
     write("L");
  }

  else if(right_cds > left_cds && right_cds > center_cds)
  {
     write("R");
  }
}
else if (left_cds > cds_threshold ||
          center_cds > cds_threshold ||
          right_cds > cds_threshold )
{
  spray=1;
  cyccnt=0;
  turning_on=1;
  if (left_cds > center_cds &&
        left_cds > right_cds &&
        left_cds > cds_threshold)
  {
     spray=0;
     write("L");
  }
  else if(right_cds > center_cds &&
           right_cds > left_cds &&
           right_cds > cds_threshold)
  {
     spray=0;
     write("R");
  }
  else if(center_cds > right_cds &&
           center_cds > left_cds &&
           center_cds > cds_threshold)
  {
     write("F");
  }
}
else if(left_cds > near_cds_threshold ||
         center_cds > near_cds_threshold ||
         right_cds > near_cds_threshold)
{
  write("02");

  if(left_cds > right_cds && left_cds > center_cds)
```

```
      {
          write("L");
          delay(); delay();
      }

      else if(right_cds > left_cds && right_cds > center_cds)
      {
          write("R");
          delay(); delay();
      }

    }
    else
    {
      write("F");
    }

  write("02");
}


void arbitrate()
{
    if(left_ir > near_ir_threshold || right_ir > near_ir_threshold )
    {
      obstacle_avoid();
    }
    else
    {
      object_follow();
    }
}

void object_spray()
{

/* turn servo 45 degrees and return to starting position */

    if(spray && cyccnt < 10)
    {
          if(turning_on)
          {
             TCTL1 = 0x80;
             TOC2 = TOC2 + 3500;

             ++cyccnt;
```

```
        }
        else
        {
          TCTL1 = 0xC0;
          TOC2 = TOC2 + 36500;


        }
   }
   CLEAR_FLAG(TFLG1, 0x40);
}


void init_spray()
{
   TCTL1 = 0x80;
   SET_BIT(TMSK1, 0x40);  /* enable OC2 interrupt */
}


/*  approximately 1/4 second delay */

void delay()
{
  asm("ldx #45454");
  asm("here :");
  asm("dex");
  asm("nop");
  asm("nop");
  asm("bne here");
}
```