University of Florida
Department of Electrical and Computer Engineering

EEL 5666

Intelligent Machines Design Laboratory

Spring 2001

"Thief"

Uriel Rodriguez

April 25, 2001

TAs: Rand Chandler, Scott Nortman

Instructor: A. A. Arroyo

Abstract

The goal of my robot is to build a robot that will simulate a thief. It is not as bad as it sounds. In reality, it's just a robot that will find small objects and hide them as a practical joke.

Executive Summary

Thief was able to perform all of its tasks successfully. The robot travels at full speed while searching for an object. Once an object it's encountered, the robot slows down to half speed so that it can align itself with the object. A break beam is used to determine if the object is ready to be trapped. After the object is trapped, the robot will begin looking for a dark spot where it can drop off the object. While searching for an object, if the robot hears any loud noises, it will go and hide in a dark spot. It will remain there for a few seconds and then begin searching for the object again.

Introduction

While thinking of an idea for the robot that I wanted to build in the class, I came to the conclusion that helping and pleasing people is harder than to annoy them. Therefore, I decided to build a robot that will play practical jokes on people by finding small objects on the floor and hiding in a dark spot in the room. This being my first attempt at building a robot, I decided to choose a simpler robot so that I can have a good chance at building something that works.

Integrated System

The electronic part of the robot will mainly consist of a TJPRO11, which has an MC68HC11 with 32K of RAM. It will also use servos to drive the wheels and the mechanical arm, IR sensors for obstacle avoidance and finding objects, CdS photo resistors to find the dark parts of the room, and a microphone to detect noise.

Mobile Platform

The platform will be designed from scratch using the MIL T-Tech machine. It will be a rolling, indoor platform. It will have a moving arm in the front controlled by a servo to trap any objects that it might find.

Actuation

Two servos will be used to drive the wheels. The servos will be hacked to provide 360 degrees of rotation. The servos had to be precisely calibrated and small tweaking in software was required in order for the wheels to turn at the same rate.

A mini-servo was used to operate the arm used to trap the object. The servo had to be small because of the amount of room available. This servo was left unhacked because only 90 degrees of rotation were needed. The arm was constructed of a small piece of balsa wood and attached to the servo.

A problem was encountered after adding the servo to the platform. Before the servo was installed, the servos used to drive the wheels where controlled by the motorp function provided by IMDL. The mini-servo is controlled by the servo function. Apparently, the motorp and servo functions don't seem to work in harmony. When init_servotjp was executed, it interfered with motor routines. As a solution, the servo routines were used to control both the mini-servo and the hacked servos driving the wheels.

Sensors

Infrared. The IR sensors will be hacked so that it outputs an analog value. This value will reflect the distance between the object and the robot. Two IR sensors will be used for obstacle avoidance and placed on the top part of the robot. Two other IR sensors will be used to detect small objects and will be placed near the ground. An unhacked IR will be used as a break beam, so that the robot can determine if the object is ready to be trapped. This IR will remain unhacked for two reasons. The first reason is that you only need to know when the break beam is broken as opposed to how strong the signal it's receiving. The second reason is that I ran out of analog ports.

CdS Photo Resistors. Cadmium Sulfide photo resistors will be placed on top of the robot to determine the darker parts of the room. The photo resistors were implemented as a voltage divider network.

Microphone. A microphone will be place on top of the robot to detect loud sounds. The microphone will be connected in series with a 2.2KΩ resistor in order to create a voltage divider network. The output will then be run through an LM386 Op-amp to amplify the signal as illustrated in Figure 1.

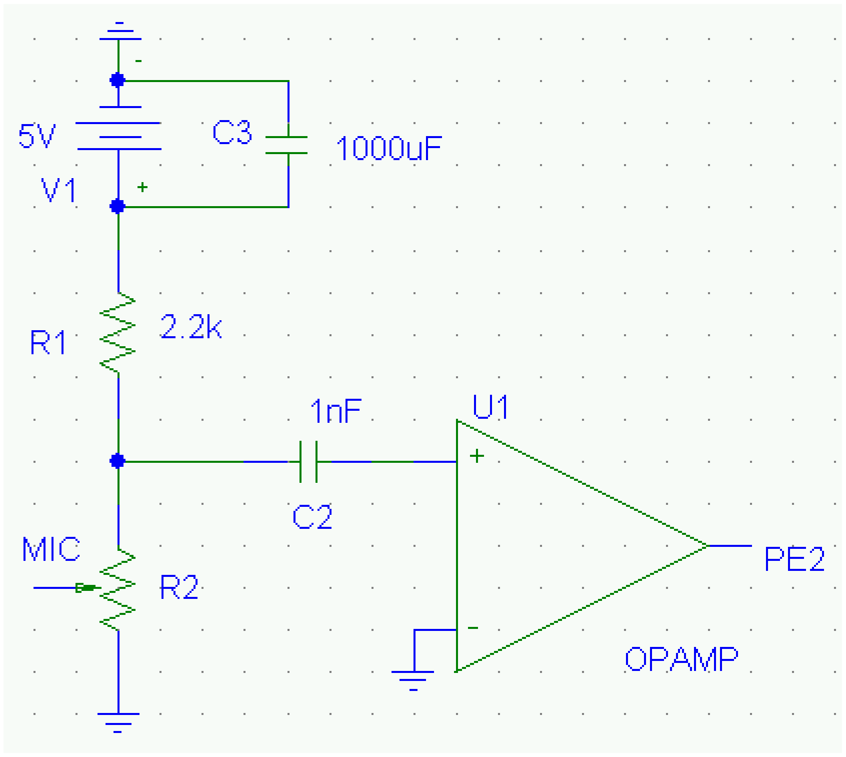

Figure 1.

To avoid having the noise produced by the servos interfere with the microphone, the microphone was placed a few inches above the rest of the platform, and thus, far away from the servos. The following data was obtained from the microphone circuit:

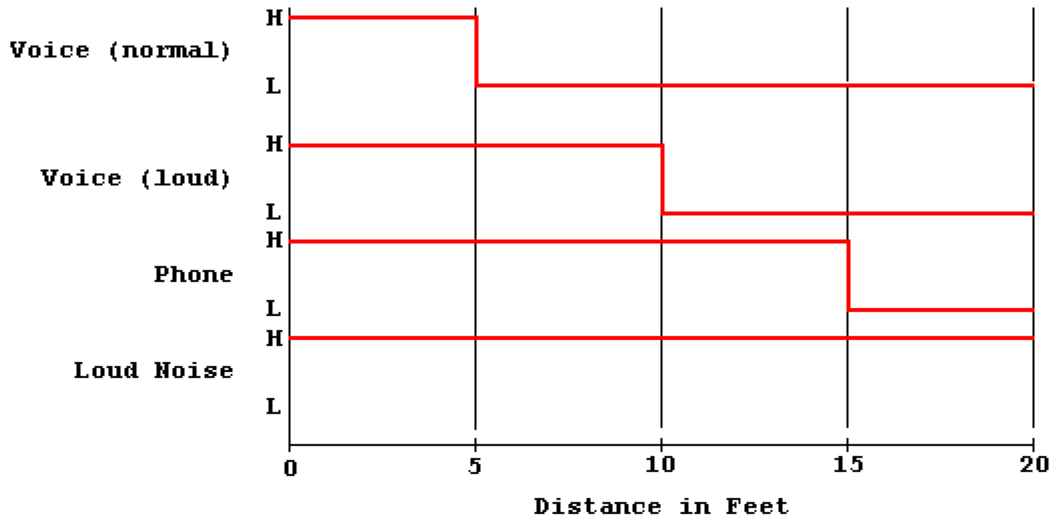

Table 1.

Bump Switches. Three bump switches will be placed in front of the robot to detect any collisions that could not be detected by the IR collision avoidance system. If a bump switch is activated, the robot will back up and then spin in a random direction.

Behaviors

Obstacle Avoidance. Obstacle avoidance will be implemented using the top two infrared sensors. Every time the robot is reset, it reads values from the IR sensors and uses this information to establish a threshold value used for deciding when an obstacle is close. If obstacle avoidance fails to recognize an obstacle, the bump sensors are used as a last resort. If the bump sensors are activated, the robot will back up and spin in a random direction and continue on it's way.

Search for Object. Searching for the object will consist of randomly walking around while until detecting an object near by while utilizing collision avoidance. It will use the two bottom IR sensors for finding near by objects. It will use the break beam to determine if it has found an object.

Trap Object. Once the robot determines that an object ready to be trapped, it will use the arm attached to the mini-servo to trap the object. Similarly, once it determines that it has taken the object to a dark spot, the arm will go to it's up position.

Find Dark Spot. The robot will need to find a dark spot in two occasions. The first being when it finds an object, and the second when it detects a noise, and it needs to hide. The two photo resistors will be used to locate a dark spot. These sensors also take initial values at reset to calibrated themselves and determine a threshold.

Listen. While the robot is searching for an object, it will "listen" with the microphone for any intruders. If any noise is detected, the robot will find the darkest spot in the room, and hide. In order to help minimize the problem of the servos creating enough noise to make the robot want to hide, the microphone was raised off the platform by a few inches. Also, this problem is also lessened in software by only reacting to large values reported by the microphone circuit.

Conclusion

Even with the unexpected setbacks, Thief did all of the things that it was designed to do. To me, IMDL was a very valuable class in which you learn how hardware and software interact. You also learn how to take an idea from conception to implementation. I believe that most of the useful knowledge that I learned from this class was from debugging since you don't know if the problem was in hardware or software, so you must double check everything.

Appendices

```
#include <analog.h>
#include <clocktjp.h>
#include <isrdecl.h>
#include <vectors.h>
#include <stdio.h>
#include <hc11.h>

#define LEFT 2
#define RIGHT 0
#define BUMPER analog(0)
#define MIC analog(1)
#define BL_IR analog(2)
#define BR_IR analog(3)
#define TL_IR analog(5)
#define TR_IR analog(7)
#define L_PR analog(4)
#define R_PR analog(6)
#define BREAKBEAM PORTA
#define IR_ON *(unsigned char *) 0x7000=0xff
#define IR_OFF *(unsigned char *) 0x7000=0x00
#define th_t 8
#define th_b 10
#define th_light 20

int i, count, found, hidden;
int tlir, trir, blir, brir, tthresh, bthresh;
int find_bl, find_br, avoid_tl, avoid_tr;
int lpr, rpr, light_diff;
int bb, mic;
int new_r, new_l, speed_r, speed_l;

void search();
void down();
void up();
void read_sensors();
void init();
void spin();
void print_sensor_values();
void hide();
void listen();
void letgo();

void main()
{
  init();
  while (BUMPER < 120);
  while(1)
  {
    while (!found)
    {
      read_sensors();
              search();

              for (i=0; i < 12; i++)
              {
                listen();
                wait(5);
              }
            }
            hide();
            found = 0;
  }
}

/*** END MAIN ***/

void init()
{
  init_analog();
  init_servotjp();
  init_clocktjp();

  IR_ON;
  found = 0;
  wait(50);
  read_sensors();
  find_bl = blir + th_b;
  find_br = brir + th_b;
  avoid_tl = tlir + th_t;
  avoid_tr = trir + th_t;
  light_diff = rpr - lpr;

  up();

  wait(50);

}

void read_sensors()
{
    tlir = TL_IR;
    trir = TR_IR;
            blir = BL_IR;
            brir = BR_IR;
    bb = BREAKBEAM;
    rpr = R_PR;
    lpr = L_PR;
}

void listen()
{
  mic = MIC;
  if (mic > 175)
  {
    servo(0, 0);
    servo(2, 0);
    wait(2500);
    hide();
    wait(10000);
    /* backup and turn */
    servo(RIGHT, 1000);
    servo(LEFT, 4000);
    wait(800);
    spin();
  }
}

void search()
{
  /* check bumper */
  if ( (BUMPER>10) && (BUMPER<110) )
  {
    /* reverse */
    servo(RIGHT, 1000);
    servo(LEFT, 4000);
    wait(800);
    spin();
  }

  /* check obstacles */
  else if((tlir > avoid_tl) || (trir > avoid_tr))
  {
            if (tlir > (trir + 5))
            {
              /* turn right */
      servo(RIGHT, 3250);
    servo(LEFT, 2500);
          }
          else if (trir > (tlir + 5))
          {
            /* turn left */
    servo(RIGHT, 4000);
    servo(LEFT, 3000);
          }
          else
            spin();
}

/* look for object */
else if((brir > find_br) || (blir > find_bl))
{
  if (blir > (brir + 5))
          {
            /* turn left */
    servo(RIGHT, 4000);
    servo(LEFT, 3000);
          }
          else if (brir > (blir + 5))
          {
            /* turn right */
    servo(RIGHT, 3250);
    servo(LEFT, 2500);
          }
          else
          {
            /* slow down, go straight */
    servo(RIGHT, 3500);
    servo(LEFT, 2500);
          }
}

/* found object? */
else if (bb == 0)
{
  servo(RIGHT, 0);
  servo(LEFT, 0);
  down();
  for (i=0; i < 30000; i++);
  wait(200);
  found = 1;
}
```

```
  /* go straight */
  else
  {
     servo(RIGHT, 4000);
     servo(LEFT, 1000);
  }
}

void hide()
{
  hidden = 0;
  wait(100);
  while(!hidden)
  {
    read_sensors();

    /* at dark spot? */
    if ( rpr >= 236 && lpr >= 236 )
    {
      hidden = 1;
    }

    /* collision avoidance */
    else if ( (BUMPER>10) && (BUMPER<110) )
    {
      /* reverse */
      servo(RIGHT, 1000);
      servo(LEFT, 4000);
      wait(800);
      spin();
    }

    /* check obstacles */
    else if((tlir > 128) || (trir > 128))
    {
            if (tlir > (trir + 5))
            {
             /* turn right */
      servo(RIGHT, 3250);
      servo(LEFT, 2500);
            }
            else if (trir > (tlir + 5))
            {
             /* turn left */
      servo(RIGHT, 4000);
      servo(LEFT, 3000);
```

```
          }
          else
            spin();
    }


    /* find dark spot */
    else if ( (rpr - light_diff) > (lpr + 20) )
    {
      /* turn right */
      servo(RIGHT, 3250);
      servo(LEFT, 2500);
    }
    else if( lpr > (rpr - light_diff + 20) )
    {
      /* turn left */
      servo(RIGHT, 4000);
      servo(LEFT, 3000);
    }
    else
    {
      /* go straight */
      servo(RIGHT, 4000);
      servo(LEFT, 1000);
    }
  }
  servo(RIGHT, 0);
  servo(LEFT, 0);

  if (found)
  {
    letgo();
  }
}

void letgo()
{
  wait(100);
  up();
  for (i=0; i < 30000; i++);
  servo(RIGHT, 1000);
  servo(LEFT, 4000);
  wait(1500);
  spin();
}
```

```
void spin()
{
 unsigned rand;
 rand = TCNT;

 if (rand & 0x0001)
 {
  /* spin clockwise */
  servo(LEFT, 2000);
  servo(RIGHT, 2000);
 }
 else
 {
  /* spin counterclockwise */
  servo(LEFT, 4000);
  servo(RIGHT, 4000);
 }

 i = (rand % 1024);

 if (i>250)
  wait(i);
 else
  wait(250);
}

void down(void)
{
 servo(1, 3500);
}

void up(void)
{
 servo(1, 5500);
}

void print_sensor_values()
{
/*
  printf("\n\ nBOTTOM RIGHT IR Value: {%d}\n",brir);
  printf("BOTTOM LEFT IR Value: {%d}\n",blir);
  printf("TOP RIGHT IR Value: {%d}\n",trir);
  printf("TOP LEFT IR Value: {%d}\n",tlir);

  i = read_int();
*/}
```