Enrique Bastante

Polyphemus

University of Florida
Department of Electrical and Computer Engineering

EEL 5666

Intelligent Machines Design Laboratory

Table of Contents

Abstract	3
Executive Summary	4
Introduction	5
Integrated Systems	6
Mobile Platform	7
Actuation	8
Sensors	8
Behaviors	14
Experimental Layout And Results	16
Conclusion	20
Documentation	21
Appendix A - Vendors	22
Appendix B - Code	23

Abstract

Polyphemus is an autonomous mobile robot featuring collision avoidance, collision detection, sound monitoring, line following, motion detection, and motion flow. To accomplish these behaviors, Polyphemus uses a total of 34 sensors including 30 CdS cells, two IR detectors, one microphone, and one bump switch.

Executive Summary

Polyphemus is controlled by a Motorola 68HC11 microcontroller along with an EBVU board and an ME11 daughter card. The ME11 is used to provide two motor drivers, two selection lines, and a 40 kHz signal.

Polyphemus features 34 sensors. Two IR detectors mounted on the front of the robot are used along with two IR LEDs (driven at 40 kHz by the ME11) to accomplish collision avoidance.

A bump sensor is used to detect collision in case the IR system fails. The sensor is also mounted on the front of the robot.

A microphone is used to scan the environment for noise levels. If a loud noise is detected, the robot freezes for a couple of second (‘scared behavior).

An array of 30 CdS cells is used for motion detection, motion flow and line following. The array is built on a PC board and its mounted on the front.

The analog port had to be expanded using 4051 multiplexers to accommodate all the sensors.

Introduction

The name of my robot is *Polyphemus*. The idea for the robot came to me when I run into an article by Kevin Ross on a Low Resolution Vision System. The idea is to mount this system on a mobile platform, so that the robot can obtain information from its environment using a visual system.

To this date, visual sensing is accomplished with the use of very expensive equipment like digital cameras and powerful microprocessors, and very elaborate software. Unfortunately, both the hardware and software needed for these systems are way beyond my abilities, my budget, and the scope of this course. For this reason, I thought it would be interesting to experiment with a low resolution vision system as the one described in the article, and find out how much visual processing can be done with this very simple setting.

When I first started this project I wanted my robot to use the vision system to do line following and basic shape recognition. Unfortunately, it became obvious with time that shape recognition was beyond the capabilities of this sensor. I did discover that the sensor could be used as a motion detection device, so I decided to implement this as part of my behaviors. Other goals were to use a microphone

to do noise level detection, and of course collision avoidance and collision detection.

Integrated System

A Motorola 68HC11 EBVU board coupled with an ME11 daughter board was used to control *Polyphemus*. Two IR sensors were mounted on top of the robot's body facing straight ahead. The sensors detect 40kHz modulated IR signals, which are produced by connecting two LEDs to the ME11 board. The LEDs are also mounted at the front of the robot. The IR sensors and emitters are used to measure distances between the robot and obstacles in its path. The measurements of the sensors are used to avoid obstacles, steering the platform in a different direction if there is an obstacle in its immediate path. A bump sensor was also mounted on the robot for emergency situations. If the IR sensors fail to detect an obstacle, the bump sensor detects the collision. After the collision, the robot backs up, turn on its own axis, and resume its forward motion.

A microphone system was mounted on the robot and connected to the analog port. It is used to track the sound level in the environment. If at any given time the sound level exceeds a specified threshold, the robot stops and freezes

as if scared. After a small delay, the robot resumes doing what it was previously doing.

For the vision system, cadmium-sulfide (CdS) cells were used. The array of CdS cells was placed on a small board, which was mounted at the front of the robot. The array consists of 30 cells in five by six array. The cells are read using the analog port, which was expanded to accommodate all the sensors.

Mobile Platform

The components of Polyphemus were mounted on the body of an old plastic toy car. The top of the body was discarded and only the bottom used. The body is about 11.5 inches long and about 4.5 inches wide. The wheels were taken out and replaced with standard Talrik wheels driven by two motors. The wheels were attached in the back, one on each side, using epoxy glue and hot glue. A caster wheel was used for stability. It was attached in the front using screws.

Holes were drilled for the screws of the EBVU/ME11 board. Holes were also drilled for the IR detectors and the array of cells, which were placed at the front of the body.

Actuation

The robot uses two motors to drive two wheels. The motors used are actually two hacked servos. The servos used were purchased from Scott Jantz. They are manufactured by Hitec and the model is HS-422 Deluxe. They have an output torque of 3.1kg-cm at 4.8V, and weight 45.5 g.

The motors are connected to the ME11 the two motor drivers on the ME11 board (J4 and J5). The ME11 uses Port A, Port D, and a 754410 to drive the motors.

The motors are along with wheels to move the robot around its environment.

Sensors

Polyphemus sensor system consists of infrared emitters and detectors, a bump switch, a microphone, and an array of cadmium-sulfide photocells (CdS). There is two infrared emitter and two detectors. They are used for collision avoidance. The bump switch is used for situations when the collision avoidance system fails to detect an obstacle and a collision takes place. The microphone will implement a fear behavior. If a loud noise is detected, the robot will stop all other activity and remain still for a period of time.

IR System

The IR system consists of two emitters and two detectors. For the emitters, LEDs purchased from Scott Jantz were used. The detectors, which were also purchased from Scott, are made by Sharp, model GP1U58Y.

The LEDs are mounted on the front of the robot, facing forward, using snap-in panel mount LED holders purchased at Radio Shack. Dark gray foam is used to seal the back of the LEDs so that they do not interfere with the detector's readings. The LEDs output a 40 kHz frequency modulated infrared signal. This signal is driven by the ME11 board's digital outputs.

The Sharp IR detectors are mounted directly under the LEDs using tape to keep them secure. The Sharp detectors are sensitive to 40 kHz frequency modulated IR signals and are originally designed to produce a digital output signal. Since an analog output signal is needed for distance measurements, the detectors had to be hacked. The hack was accomplished following the directions in the web site's handouts.

The IR detectors are connected to PE1 and PE3.

CdS cells array

The array of CdS cells is used as a low-resolution vision system, and it is based on an article written by Kevin Ross. The array consists of 30 CdS cells, which were obtained at the IMDL lab, and was built on a component PC board bought at Radio Shack. The cells are arranged in a five by six array with about one fourth of an inch in between them. A representation of the array can be seen in *Figure 1*.

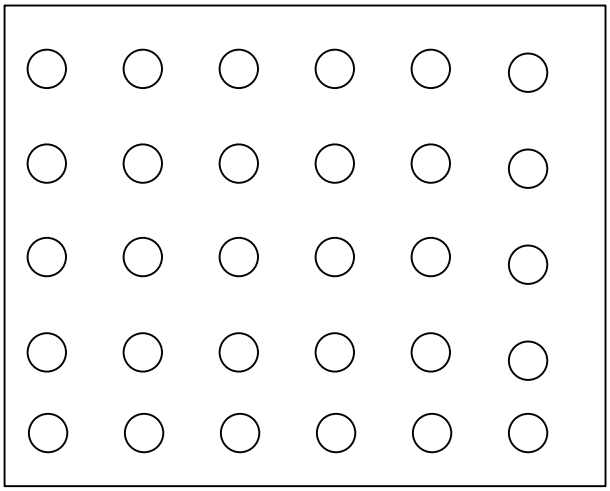


Figure 1

CdS are used to detect light intensity. Their internal resistance varies according to the intensity of the light in the environment. A higher level of light reduces the resistance in the cells, so it is an inverse light-resistance relation. To obtain a voltage proportionate to

the level of light, the cells must be connected to a 5V and to a resistor, which is connected to ground. The node between the cell and the resistor is then connected to the A/D converter. This configuration acts as a voltage divider, with the CdS cells behaving as a variable resistor. A diagram of a single CdS cell can be seen in *Figure 2.*

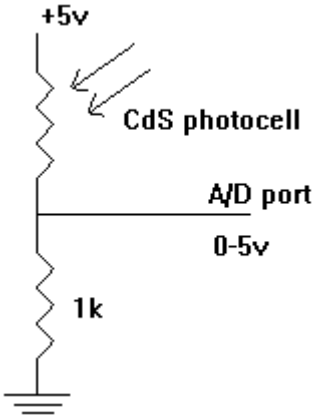


Figure 2

Since the 68HC11 has only eight A/D lines, and the array has 20 cells, some extra circuitry is needed. The A/D lines were multiplexed using four 74HC4051 multiplexers. The multiplexers provided 32 lines, more than enough for the array, and they only used four A/D lines (PE7, PE6, PE5, and PE4). For the select lines of the multiplexers, extra output lines were needed. Two extra output ports were created using 74HC574 flip-flops. The flip-flops were wired

to port C of the microcontroller and two select lines (Y0 and Y2) from the ME11 were used for selection. The design schematic can be seen in *Figure 3.*

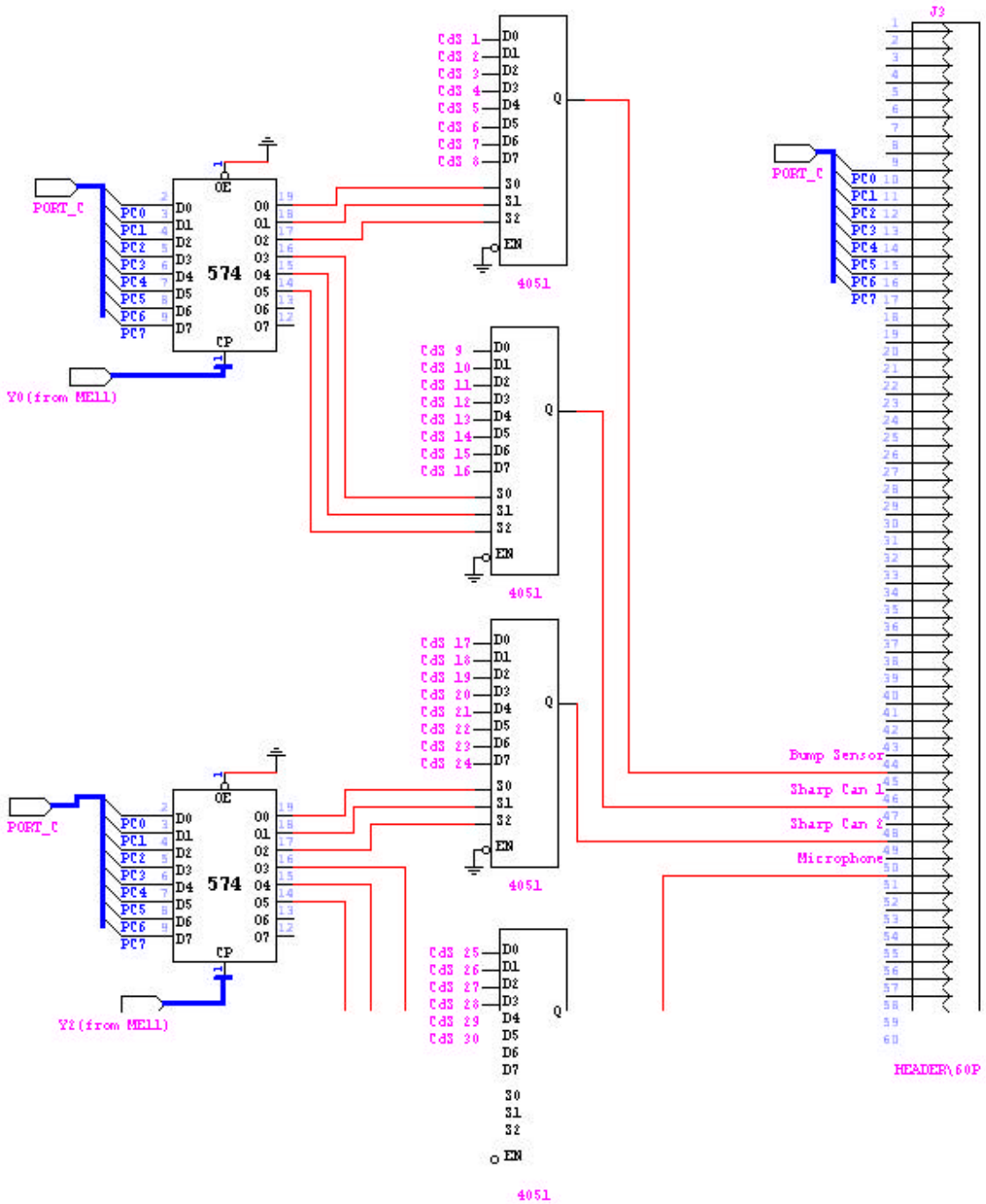


Figure 3

The cells I used had a resistance value of about 100 ohms when placed directly in front of light source and around 60 k-ohms when they were in almost complete darkness. The average value in a room with artificial light was 8 k-ohms. I chose to use resistors of 10 k-ohms value in the array circuit so that the average value would be around 2.5 volts, leaving plenty of room to swing in both directions.

Microphone

The microphone is be used to detect noise levels and trigger a scared behavior when a loud noise is detected. The microphone used is a PC Board microphone from Radio Shack will be used. The microphone needs an amplifier to produce a signal the microcontroller can use. The amplifier used is an LM386 and was bought at Electronics Plus. The capacitors and resistor were found in the IMDL lab. The circuit used is the one found in “Mobile Robots” (pg 143), and was built on a Dual Purpose PC Board bought at Radio Shack. The output line is connected to PE2.

Bump Switch

The bump switch sensor is located at the front of the robot. It was obtained at the IMDL lab. A plastic bumper is attached to the switch to cover more area. When the bump hits an obstacle, the switch sends a signal to the A/D port

(PE4). The robot reacts to this signal by backing up and going in a different direction.

Behaviors

Polyphemus has a total of five behaviors:

Collision Avoidance

Polyphemus tries to avoid objects by steering away from. When the readings from its IR detectors exceed a particular threshold (around 8 inches), the robot steers in a different direction to avoid the object.

Collision Detection

If collision avoidance fails and a collision does occur, Polyphemus goes into collision detection mode. The bump sensor signal goes from 0 to 255, alerting the robot that a collision has occurred. The robot then backs up and goes in a different direction.

Scared Behavior

The robot continuously scans the environment for the noise level. If it hears a loud noise (above the threshold) the robot gets scared, therefore it freezes. After a 2 second delay, the robot reassumes doing its previous task.

Motion Detection

For this behavior, the array of cells is divided into two sectors of 15 cells each: the right and the left sectors. If motion is detected in the right sector, the robot spins to the right. If motion is detected in the left sector, the robot spins in that direction.

Line Following

The robot follows a path marked by a black line on a white surface. This is possible because the color black does not reflect light as well as the color white. To follow a line, averaged readings from two cells in the center columns, a cell in the leftmost column, and a cell in the rightmost column are taken. The robot is then programmed to keep the lower value (the dark color) in the center of the array by steering in the correct direction.

Motion Flow

Polyphemus uses motion flow to determine if it is moving. To do this, the robot takes snap shots of the ground every 500 milliseconds using the array of CdS cells. It compares snap shots to determine if there is any difference between them. If the change is significant, the robot determines that it has moved. If there is little or no change, the robot tries again. If five consecutive snap shots are the same, the robot determines that there has been no motion; therefore, it must be stuck. In this case, the robot backs

up and goes in a different direction. Motion flow makes the robot makes it more difficult for the robot to get stuck, because even if the bumper fails, the robot can determine if it has gotten stuck.

Experimental Layouts and Results

IR System

The IR system was tested using an ICC11 routine I wrote. It continuously reads the IR detectors and prints them to the screen.

For the experiment, the robot was put on the ground and the motors were turned off so it would remain static. A hard cover book was used as an obstacle, and a measuring tape was used to measure the distance from the book to the IR detectors. The values of the detectors were recorded every half an inch beginning at 24 inches, all the way up to 0.5 inches.

This experiment was used to determine the threshold for collision avoidance. After examining the data, a threshold of 100 was picked.

Figure 4 shows the data from the left IR detector. *Figure 5* shows the data from the right IR detector. *Figure 6* shows both in the same graph.

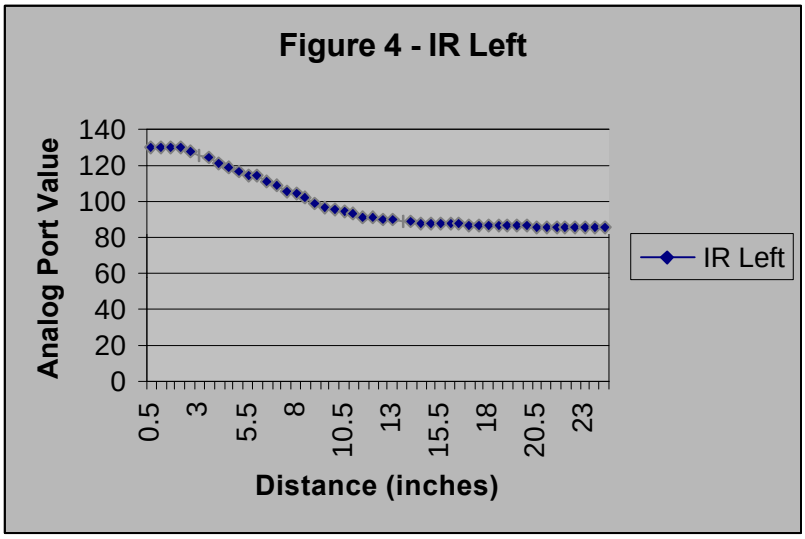
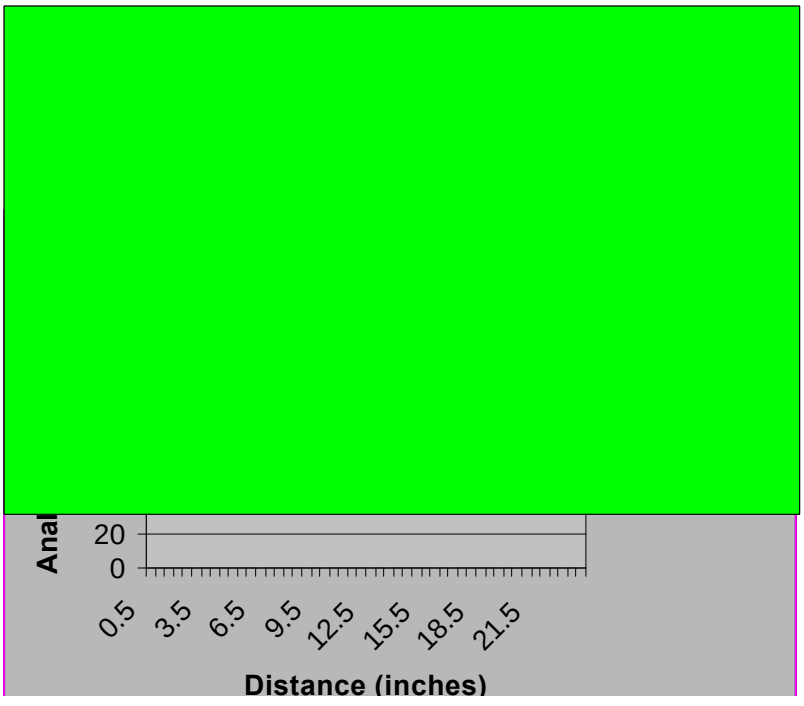


Figure 4 - IR Left
Analog Port Value
140
120
100
80
60
40
20
0
0.5
3
5.5
8
10.5
13
15.5
18
20.5
23
Distance (inches)
IR Left


Anal
20
0
0.5
3.5
6.5
9.5
12.5
15.5
18.5
21.5
Distance (inches)

CdS cells array

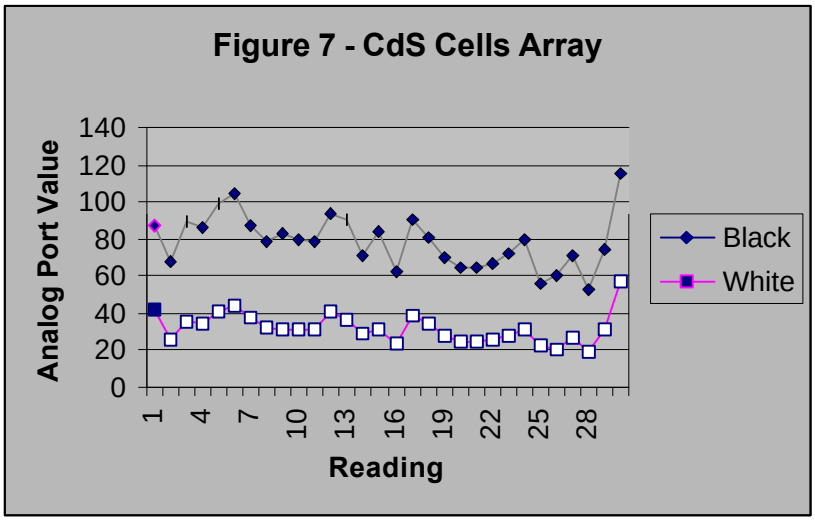
For this experiment, all 30 cells were read with both a white surface and a white surface. This experiment was done to observe the difference in readings between black and white. It can be see how the shadow of the board affects the array readings. *Table 1* shows the readings with a white surface, and *Table 2* the readings with a black surface. *Figure 7* shows the data in a graph.

87	68	89	86	99	105
87	79	83	80	79	94
90	71	84	63	90	81
70	65	65	67	72	80
56	60	71	53	74	115

Table 1

42	26	36	34	41	44
38	32	31	31	31	41
37	29	31	24	39	34
28	25	25	26	28	31
23	21	27	19	31	57

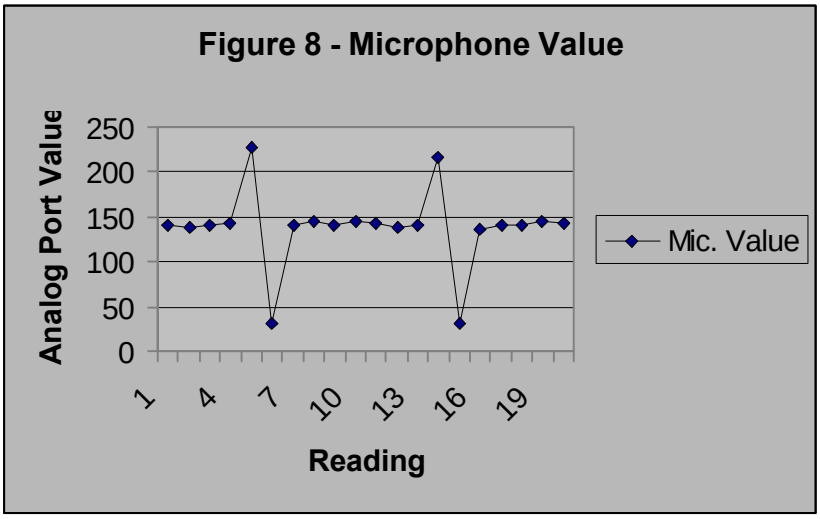
Table 2



Microphone

For this experiment, I took 20 readings from the analog port, and twice during the readings a short-loud whistle was introduced next to the microphone.

The purpose of this experiment was to determine a threshold value for the scared behavior. Figure 8 shows the data in a graph. Note that when the loud whistle is introduced, the value jumps and then drops dramatically due to saturation. After careful examination the threshold was picked to be 200.



Conclusion

Polyphemus proved to be a great learning experience. Considering this is my first experience building a robot, I am happy with what I accomplished.

The collision avoidance, collision detection, and microphone sensors were solid worked well. Unfortunately, I

encountered many problems with the array of CdS cells. The main problem I had was lightning. The board the array was built on was casting a shadow on the surface to be read. Many different lighting solution were tried, but all failed. The array did prove to be successful doing motion detection.

If I were to start over, I would focus on motion detection instead of shape recognition or any vision attempt. I would also make the array smaller, and maybe use a lens.

Documentation

This project would have not been possible without:

- Low Resolution Vision System. Kevin Ross. Seattle Robotics Society Website.
- Mekatronics manuals
- Mobile Robots. Jones, Seiger, Flynn. A.K. Peters Publishing. Natick, Massachusetts, 1999
- Mekatronics manuals
- Scott Jantz
- My classmates

Appendix A

Vendor Information

Electronics Plus
LM386 amplifier

Lowes
Caster Wheel

Mekatronics
316 NW 17th Street Suite A
Gainesville, FL 32603
Hitec HS-422 Deluxe Servos
LEDs
Sharp GP1U58Y
Wheels
Motor Driver 754410

Radio Shack
PC Component Board
PC Dual Purpose Board
PC Board Microphone

Appendix B

Code

```
/*************************************************************
**************
 * Title        avoid.c
*
 * Programmer  Enrique Bastante
     *
 * Date         July 24, 1999
*
 * Version     1    Collision Avoidance and Scared Behavior
                          *
 *                                              *
 *                                    *

*************************************************************
*************/

/***************************** Includes
*********************************/
#include <tjpbase.h>
#include <stdio.h>
#include <vectors.h>
#include <analog.h>
/************************ End of includes
*******************************/


void main(void)
/***************************** Main
*********************************/
{
  int ir0, ir1, micro, bump, rspeed, lspeed;

  init_motorme();
  init_clocktjp();
  init_analog();

while(1)
{

  IRE_ON;

  micro=analog(2);
  ir0=analog(3);
```

```
ir1=analog(1);
bump=analog(0);


if (micro > 200)
{
  rspeed = 0;
  lspeed = 0;
  motorme(RIGHT_MOTOR, rspeed);
  motorme(LEFT_MOTOR, lspeed);
  wait(2500);
}

if (bump > 200)
{
  rspeed = 0;
  lspeed = 0;
  motorme(RIGHT_MOTOR, rspeed);
  motorme(LEFT_MOTOR, lspeed);

  rspeed = -100;
  lspeed = -100;
  motorme(RIGHT_MOTOR, rspeed);
  motorme(LEFT_MOTOR, lspeed);
  wait(1500);

  rspeed = -100;
  lspeed = 0;
  motorme(RIGHT_MOTOR, rspeed);
  motorme(LEFT_MOTOR, lspeed);
  wait(1000);

  rspeed = 0;
  lspeed = 0;
  motorme(RIGHT_MOTOR, rspeed);
  motorme(LEFT_MOTOR, lspeed);

}

if (ir0 > 100)
  lspeed = -50;
else
  lspeed = 100;
if (ir1 > 100)
  rspeed = -50;
else
  rspeed = 100;
```

```
    motorme(RIGHT_MOTOR, rspeed);
    motorme(LEFT_MOTOR, lspeed);


  wait(50);
}


}
/***************************** End of Main
******************************/




/************************************************************
*************
 * Title        Motion.c
*
 * Programmer  Enrique Bastante
     *
 * Date         July 24, 1999
*
 * Version     1    Motion Detection
         *
 *                                                  *
 *                                  *

************************************************************
************/

/***************************** Includes
*********************************/
#include <tjpbase.h>
#include <stdio.h>
#include <vectors.h>
#include <analog.h>
/************************ End of includes
******************************/

#define CDS_01  *(unsigned char *)(0x4000) = 0x00
#define CDS_02  *(unsigned char *)(0x4000) = 0x01
#define CDS_03  *(unsigned char *)(0x4000) = 0x02
#define CDS_04  *(unsigned char *)(0x4000) = 0x03
```

```
#define CDS_05  *(unsigned char *)(0x4000) = 0x04
#define CDS_06  *(unsigned char *)(0x4000) = 0x05
#define CDS_07  *(unsigned char *)(0x4000) = 0x06
#define CDS_08  *(unsigned char *)(0x4000) = 0x07
#define CDS_09  *(unsigned char *)(0x4000) = 0x00
#define CDS_10  *(unsigned char *)(0x4000) = 0x08
#define CDS_11  *(unsigned char *)(0x4000) = 0x10
#define CDS_12  *(unsigned char *)(0x4000) = 0x18
#define CDS_13  *(unsigned char *)(0x4000) = 0x20
#define CDS_14  *(unsigned char *)(0x4000) = 0x28
#define CDS_15  *(unsigned char *)(0x4000) = 0x30
#define CDS_16  *(unsigned char *)(0x4000) = 0x38
#define CDS_17  *(unsigned char *)(0x5000) = 0x00
#define CDS_18  *(unsigned char *)(0x5000) = 0x01
#define CDS_19  *(unsigned char *)(0x5000) = 0x02
#define CDS_20  *(unsigned char *)(0x5000) = 0x03
#define CDS_21  *(unsigned char *)(0x5000) = 0x04
#define CDS_22  *(unsigned char *)(0x5000) = 0x05
#define CDS_23  *(unsigned char *)(0x5000) = 0x06
#define CDS_24  *(unsigned char *)(0x5000) = 0x07
#define CDS_25  *(unsigned char *)(0x5000) = 0x00
#define CDS_26  *(unsigned char *)(0x5000) = 0x08
#define CDS_27  *(unsigned char *)(0x5000) = 0x10
#define CDS_28  *(unsigned char *)(0x5000) = 0x18
#define CDS_29  *(unsigned char *)(0x5000) = 0x20
#define CDS_30  *(unsigned char *)(0x5000) = 0x28


void main(void)
/****************************** Main
*********************************/
{
  int i, left, right, cds[30], cds2[30], diff[30],
delta[30];
/* VT100 clear screen */
  char c1, clear[]= "\x1b\x5B\x32\x4A\x04";

/* VT100 position cursor at (x,y) = (3,12) command is
"\x1b[3;12H"*/
  char place[]= "\x1b[1;1H";  /*Home*/

  init_motorme();
  init_serial();
  init_clocktjp();
  init_analog();
```

```
while(1)
{

  for(i=0; i<30; i++)
    delta[i]=0;


  CDS_01;
  cds[0]=analog(4);

  CDS_02;
  cds[1]=analog(4);

  CDS_03;
  cds[2]=analog(4);

  CDS_04;
  cds[3]=analog(4);

  CDS_05;
  cds[4]=analog(4);

  CDS_06;
  cds[5]=analog(4);

  CDS_07;
  cds[6]=analog(4);

  CDS_08;
  cds[7]=analog(4);


  CDS_09;
  cds[8]=analog(5);

  CDS_10;
  cds[9]=analog(5);

  CDS_11;
  cds[10]=analog(5);

  CDS_12;
  cds[11]=analog(5);

  CDS_13;
  cds[12]=analog(5);
```

```
CDS_14;
cds[13]=analog(5);

CDS_15;
cds[14]=analog(5);

CDS_16;
cds[15]=analog(5);


CDS_17;
cds[16]=analog(6);

CDS_18;
cds[17]=analog(6);

CDS_19;
cds[18]=analog(6);

CDS_20;
cds[19]=analog(6);

CDS_21;
cds[20]=analog(6);

CDS_22;
cds[21]=analog(6);

CDS_23;
cds[22]=analog(6);

CDS_24;
cds[23]=analog(6);


CDS_25;
cds[24]=analog(7);

CDS_26;
cds[25]=analog(7);

CDS_27;
cds[26]=analog(7);

CDS_28;
cds[27]=analog(7);
```

```
CDS_29;
cds[28]=analog(7);

CDS_30;
cds[29]=analog(7);



for(i=0; i<5; i++)
{

  CDS_01;
  cds2[0]=analog(4);

  CDS_02;
  cds2[1]=analog(4);

  CDS_03;
  cds2[2]=analog(4);

  CDS_04;
  cds2[3]=analog(4);

  CDS_05;
  cds2[4]=analog(4);

  CDS_06;
  cds2[5]=analog(4);

  CDS_07;
  cds2[6]=analog(4);

  CDS_08;
  cds2[7]=analog(4);


  CDS_09;
  cds2[8]=analog(5);

  CDS_10;
  cds2[9]=analog(5);

  CDS_11;
  cds2[10]=analog(5);

  CDS_12;
```

```
cds2[11]=analog(5);

CDS_13;
cds2[12]=analog(5);

CDS_14;
cds2[13]=analog(5);

CDS_15;
cds2[14]=analog(5);

CDS_16;
cds2[15]=analog(5);


CDS_17;
cds2[16]=analog(6);

CDS_18;
cds2[17]=analog(6);

CDS_19;
cds2[18]=analog(6);

CDS_20;
cds2[19]=analog(6);

CDS_21;
cds2[20]=analog(6);

CDS_22;
cds2[21]=analog(6);

CDS_23;
cds2[22]=analog(6);

CDS_24;
cds2[23]=analog(6);


CDS_25;
cds2[24]=analog(7);

CDS_26;
cds2[25]=analog(7);

CDS_27;
```

```
  cds2[26]=analog(7);

  CDS_28;
  cds2[27]=analog(7);

  CDS_29;
  cds2[28]=analog(7);

  CDS_30;
  cds2[29]=analog(7);

  for(i=0; i<30; i++)
    diff[i] = cds[i] - cds2[i];

  for(i=0; i<30; i++)
    delta[i] = delta[i] + diff[i];

  wait(100);
}


printf("%s", clear);
printf("%s", place);

printf("\tTitle\tcelltest3.c\n"
"\tProgrammer\tEnrique M. Bastante\n"
"\tDate\t\tJuly 26, 1999\n"
"\tVersion\t\t1\n\n");


for(i=0; i<30; i++)
  if(delta[i] < 0)
    delta[i] = -delta[i];




right=0;
left=0;


for(i=0; i<30; i++)
{
  if(i!=0 && (i%3)==0)
    i = i+3;
  if(delta[i] > 2)
    left++;
```

```
  }

  for(i=0; i<30; i++)
  {
    if((i%3)==0)
      i = i+3;
    if(delta[i] > 2)
      right++;
  }



  printf(" Right is: %d \n", right);

  printf(" Left: %d \n", left);

  if(right - 1> left)
  {
    printf(" Turn right");
    motorme(RIGHT_MOTOR, 100);
    motorme(LEFT_MOTOR, -100);
    wait(1800);
    motorme(RIGHT_MOTOR, 0);
    motorme(LEFT_MOTOR, 0);
  }

  else if((left - 1) > right)
  {
    printf(" Turn left");
    motorme(RIGHT_MOTOR, -100);
    motorme(LEFT_MOTOR, 100);
    wait(1800);
    motorme(RIGHT_MOTOR, 0);
    motorme(LEFT_MOTOR, 0);
  }
  wait(300);



}


}
/***************************** End of Main
******************************/
```

```
/*************************************************************
**************
 * Title        Line.c
*
 * Programmer  Enrique Bastante
     *
 * Date         July 24, 1999
*
 * Version     1    Line Following
     *
 *                                                  *
 *                                    *

*************************************************************
*************/

/***************************** Includes
*********************************/
#include <tjpbase.h>
#include <stdio.h>
#include <vectors.h>
#include <analog.h>
/************************ End of includes
*******************************/

#define CDS_01  *(unsigned char *)(0x4000) = 0x00
#define CDS_02  *(unsigned char *)(0x4000) = 0x01
#define CDS_03  *(unsigned char *)(0x4000) = 0x02
#define CDS_04  *(unsigned char *)(0x4000) = 0x03
#define CDS_05  *(unsigned char *)(0x4000) = 0x04
#define CDS_06  *(unsigned char *)(0x4000) = 0x05



void main(void)
/***************************** Main
*********************************/
{
  int i, cdsr, cdsm1, cdsm2, cdsl, cdsm;


  init_motorme();
  init_serial();
  init_clocktjp();
  init_analog();
```

```
while(1)
{


  CDS_01;
  cdsl=analog(4);

  CDS_03;
  cdsm1=analog(4);

  CDS_04;
  cdsm2=analog(4);

  CDS_06;
  cdsr=analog(4);

  cdsm = (cdsm1 + cdsm2)/2;


  if((cdsr) < (cdsm) && (cdsr) < (cdsl))
  {
    motorme(RIGHT_MOTOR, 20);
    motorme(LEFT_MOTOR, 10);
  }

  else if(cdsl < (cdsr) && cdsl < (cdsm))
  {
    motorme(RIGHT_MOTOR, 10);
    motorme(LEFT_MOTOR, 20);
  }

  else
  {
    motorme(RIGHT_MOTOR, 20);
    motorme(LEFT_MOTOR, 20);
  }




}
```

```
}
/*************************** End of Main
******************************/
```

```
/*************************************************************
**************
 * Title        motion.c
*
 * Programmer  Enrique Bastante
     *
 * Date         July 24, 1999
*
 * Version     1                                          *
 *                  Motion Flow                             *
 *                                 *

*************************************************************
*************/

/*************************** Includes
*********************************/
#include <tjpbase.h>
#include <stdio.h>
#include <vectors.h>
#include <analog.h>
/************************ End of includes
*******************************/

#define CDS_01  *(unsigned char *)(0x4000) = 0x00
#define CDS_02  *(unsigned char *)(0x4000) = 0x01
#define CDS_03  *(unsigned char *)(0x4000) = 0x02
#define CDS_04  *(unsigned char *)(0x4000) = 0x03
#define CDS_05  *(unsigned char *)(0x4000) = 0x04
#define CDS_06  *(unsigned char *)(0x4000) = 0x05
#define CDS_07  *(unsigned char *)(0x4000) = 0x06
#define CDS_08  *(unsigned char *)(0x4000) = 0x07
#define CDS_09  *(unsigned char *)(0x4000) = 0x00
#define CDS_10  *(unsigned char *)(0x4000) = 0x08
#define CDS_11  *(unsigned char *)(0x4000) = 0x10
#define CDS_12  *(unsigned char *)(0x4000) = 0x18
```

```
#define CDS_13  *(unsigned char *)(0x4000) = 0x20
#define CDS_14  *(unsigned char *)(0x4000) = 0x28
#define CDS_15  *(unsigned char *)(0x4000) = 0x30
#define CDS_16  *(unsigned char *)(0x4000) = 0x38
#define CDS_17  *(unsigned char *)(0x5000) = 0x00
#define CDS_18  *(unsigned char *)(0x5000) = 0x01
#define CDS_19  *(unsigned char *)(0x5000) = 0x02
#define CDS_20  *(unsigned char *)(0x5000) = 0x03
#define CDS_21  *(unsigned char *)(0x5000) = 0x04
#define CDS_22  *(unsigned char *)(0x5000) = 0x05
#define CDS_23  *(unsigned char *)(0x5000) = 0x06
#define CDS_24  *(unsigned char *)(0x5000) = 0x07
#define CDS_25  *(unsigned char *)(0x5000) = 0x00
#define CDS_26  *(unsigned char *)(0x5000) = 0x08
#define CDS_27  *(unsigned char *)(0x5000) = 0x10
#define CDS_28  *(unsigned char *)(0x5000) = 0x18
#define CDS_29  *(unsigned char *)(0x5000) = 0x20
#define CDS_30  *(unsigned char *)(0x5000) = 0x28


void main(void)
/****************************** Main
*********************************/
{
  int i, count, cds[30], cds2[30], diff, delta[30], rspeed,
lspeed;
/* VT100 clear screen */
  char c1, clear[]= "\x1b\x5B\x32\x4A\x04";

/* VT100 position cursor at (x,y) = (3,12) command is
"\x1b[3;12H"*/
  char place[]= "\x1b[1;1H";  /*Home*/

  init_motorme();
  init_serial();
  init_clocktjp();
  init_analog();

  count = 0;


  printf("%s", clear);
  printf("%s", place);

  printf("\tTitle\tMotion.c\n"
  "\tProgrammer\tEnrique M. Bastante\n"
  "\tDate\t\tJuly 26, 1999\n"
```

```
  "\tVersion\t\t1\n\n");

while(1)
{

  for(i=0; i<30; i++)
    delta[i]=0;


  CDS_01;
  cds[0]=analog(4);

  CDS_02;
  cds[1]=analog(4);

  CDS_03;
  cds[2]=analog(4);

  CDS_04;
  cds[3]=analog(4);

  CDS_05;
  cds[4]=analog(4);

  CDS_06;
  cds[5]=analog(4);

  CDS_07;
  cds[6]=analog(4);

  CDS_08;
  cds[7]=analog(4);


  CDS_09;
  cds[8]=analog(5);

  CDS_10;
  cds[9]=analog(5);

  CDS_11;
  cds[10]=analog(5);

  CDS_12;
  cds[11]=analog(5);

  CDS_13;
```

```
cds[12]=analog(5);

CDS_14;
cds[13]=analog(5);

CDS_15;
cds[14]=analog(5);

CDS_16;
cds[15]=analog(5);


CDS_17;
cds[16]=analog(6);

CDS_18;
cds[17]=analog(6);

CDS_19;
cds[18]=analog(6);

CDS_20;
cds[19]=analog(6);

CDS_21;
cds[20]=analog(6);

CDS_22;
cds[21]=analog(6);

CDS_23;
cds[22]=analog(6);

CDS_24;
cds[23]=analog(6);


CDS_25;
cds[24]=analog(7);

CDS_26;
cds[25]=analog(7);

CDS_27;
cds[26]=analog(7);

CDS_28;
```

```
cds[27]=analog(7);

CDS_29;
cds[28]=analog(7);

CDS_30;
cds[29]=analog(7);


  wait(500);

  CDS_01;
  cds2[0]=analog(4);

  CDS_02;
  cds2[1]=analog(4);

  CDS_03;
  cds2[2]=analog(4);

  CDS_04;
  cds2[3]=analog(4);

  CDS_05;
  cds2[4]=analog(4);

  CDS_06;
  cds2[5]=analog(4);

  CDS_07;
  cds2[6]=analog(4);

  CDS_08;
  cds2[7]=analog(4);


  CDS_09;
  cds2[8]=analog(5);

  CDS_10;
  cds2[9]=analog(5);

  CDS_11;
  cds2[10]=analog(5);

  CDS_12;
  cds2[11]=analog(5);
```

```
CDS_13;
cds2[12]=analog(5);

CDS_14;
cds2[13]=analog(5);

CDS_15;
cds2[14]=analog(5);

CDS_16;
cds2[15]=analog(5);


CDS_17;
cds2[16]=analog(6);

CDS_18;
cds2[17]=analog(6);

CDS_19;
cds2[18]=analog(6);

CDS_20;
cds2[19]=analog(6);

CDS_21;
cds2[20]=analog(6);

CDS_22;
cds2[21]=analog(6);

CDS_23;
cds2[22]=analog(6);

CDS_24;
cds2[23]=analog(6);


CDS_25;
cds2[24]=analog(7);

CDS_26;
cds2[25]=analog(7);

CDS_27;
cds2[26]=analog(7);
```

```
  CDS_28;
  cds2[27]=analog(7);

  CDS_29;
  cds2[28]=analog(7);

  CDS_30;
  cds2[29]=analog(7);

  for(i=0; i<30; i++)
    delta[i] = cds[i] - cds2[i];

for(i=0; i<30; i++)
  if(delta[i] < 0)
    delta[i] = -delta[i];

  diff = 0;

  for(i=0; i<30; i++)
    if(delta[i] > 1)
      diff = diff + 1;
  printf(" Diff is: %d \n", diff);

  if(diff < 5)
     count = count + 1;
  else
     count = 0;

  if(count > 3)
  {
     printf(" Stopped\n\n");

     rspeed = 0;
     lspeed = 0;
     motorme(RIGHT_MOTOR, rspeed);
     motorme(LEFT_MOTOR, lspeed);

     rspeed = -100;
     lspeed = -100;
     motorme(RIGHT_MOTOR, rspeed);
     motorme(LEFT_MOTOR, lspeed);
     wait(1500);

     rspeed = -100;
     lspeed = 0;
     motorme(RIGHT_MOTOR, rspeed);
```

```
        motorme(LEFT_MOTOR, lspeed);
        wait(1000);

        rspeed = 0;
        lspeed = 0;
        motorme(RIGHT_MOTOR, rspeed);
        motorme(LEFT_MOTOR, lspeed);

    }

    else
    { lspeed = 100;
      rspeed = 100;

      motorme(RIGHT_MOTOR, rspeed);
      motorme(LEFT_MOTOR, lspeed);
    }
}
}
/*************************** End of Main
*****************************/
```