Intelligent Machine Design Laboratory

EEL 5934 (Robotics)

Professor: Keith L. Doty

THOR

Final Written Report

Chris Parman

December, 12 1996

Table of Contents:

Abstract.....2

Executive Summary……..2

Introduction.....3

Sensor configuration…….3

Platform configuration……..4

Preliminary behaviors……5

Cadmium Sulfide cells……..6

Robotic arm implementation…….7

Conclusion…….8

Documentation……...10

Appendix……..10

Abstract

The robot “THOR” is a Predator with a robotic arm. The robot will utilize predator behaviors and robotic arm behaviors. The predator behaviors will include bumper sensors and IR detection, along with these standard behaviors, “signal” jamming and the use of “kill” boxes will utilized in the robot design. I will be demonstrating the robot predator behaviors with along with a fellow students “prey” robot. The implementation is a robotic arm, this arm will pick up small-light objects then rotate no more than 90 degrees and place that object into a container. The basis of the robot design is a round wooden platform and is propelled by two wheels powered by two servo motors. The robot includes various infrared, contact and touch sensors. These sensors enable the robot to perform the behaviors mention above. The robot is controlled by a Motorola 68HC11 EVBU board with a NovaSoft 32k RAM expansion board.

Executive Summary

This project consists in the creation of a autonomous agent that displays specific programmed behaviors. These behaviors are programmed using Interactive C language interpreter and stored within 32k RAM of memory. The brains of the robot (autonomous agent) is a MC68HC11 microcontroler and ME11 expansion board. The expansion board provides motor control for two motors and a 8-bit analog/digital output. This expansion board also provides 32k RAM memory. The MC68HC11/ME11 expansion board assembly is mounted to a plywood base. Two motors, which drive two wheels, are mounted to the bottom of the plywood base. Three types of sensors are implemented within the autonomous agent. These three types are, IR emitters/detectors, micro switches as bumpers and Cadmium Sulfide cells as light detectors. In addition to these sensors, I have implemented a robotic arm. This robotic arm is used to pick up small, relatively light weight objects. All these sensors, in combination with the robotic arm, give the autonomous agent real

world functionality. Keeping this in mind, autonomous agents could be the next technology to enhance commerical and domestic environments well into the next century.

Introduction

This report is a preliminary documentation of the robotics project including sensor configuration, robot platform configuration, preliminary behaviors and future robotics implementation. The report starts out with an overview of the sensor configuration. Then a description of the robot platform, motor/wheel layout, power supply position and EVBU/Expansion board placement will be discussed. Following that description, a discussion of the preliminary behaviors are described in detail. Finally, a description of future implementations are going to be discussed in detail.

Sensor Configuration

The robot employs four IR sensors and IR emitters. These IR sensors are configured in such a way that there is almost no “null” area in front of the robot. This configuration is shown in figure 1. Together with the IR sensors, there are IR emitters that produce a 40-KHz signal. This signal is transmitted in various directions and deflects off objects in its path. This signal deflection is then captured by the IR sensor, thus determining the strength of that signal. This signal is then feed into the EVBU/expansion board where the signal is processed. This process is govern by a programming language, such a programming language is called “C” code. A interactive programming tool called “Interactive C” is used by various university autonomous robotic programs and autonomous robot hobbyist. This programming tool allows a C program to be written and downloaded into the EVBU/expansion board. After the has be downloaded, the robot should perform according to the programmed instructions. But it is not always the case, thus debugging and editing the program will be needed.

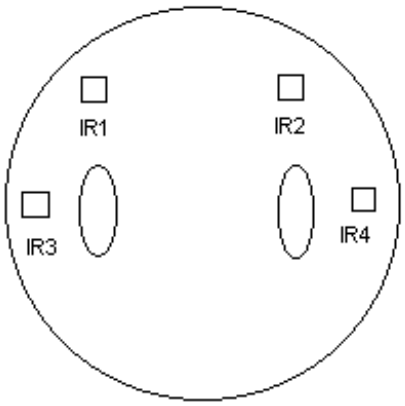

Figure 1.

Platform configuration

The robot base platform is made from “ply” wood, the thickness is ¼” and is cut in a circle. There are two wheel cut-outs within the base platform to accommodate the 2.75” diameter rubber wheels. The two wheel cut-outs were cut larger than the width of the rubber wheels to prevent “rubbing” of the wheels and (if needed in the future) making it easy the replace the wheels or change to a different diameter wheel. A picture of the base robot platform is show in figure 2.

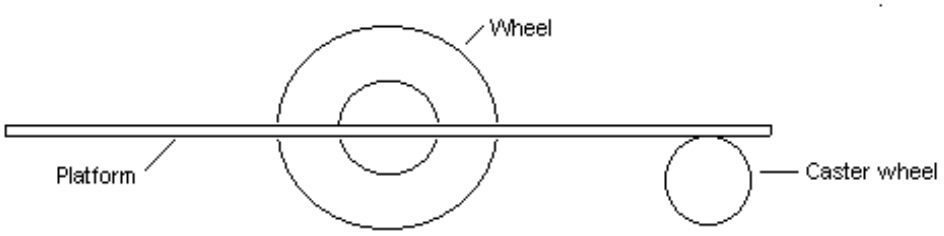

Figure 2.

In addition to the wheels, there is a single caster wheel at the rear of the robot base platform. This caster wheel is needed for stability for the robot in general and for the robotic arm I plan to implement. A battery pack (8-AAA batteries) are attached to the bottom of the base platform. This battery pack is the EVBU/expansion boards power supply.

Preliminary Behaviors

The preliminary behaviors of the robot are “object” avoidance and “bumper” detection. For the object avoidance IR sensors are used in conduction with IR emitters detect stationary objects or moving objects. There also used to detect differences in shades of color of objects, if needed. In cases, IR sensors can transmit information to a central system, ie,. computer terminal, collection station, another autonomous robot. For bumper detection “micro” switches are used. These micro switches are placed along the base (on the edge) robot platform. Along with the micro switches, a thin plastic strip is glued to the micro switch “plunger” to act as a bumper, similarly to a bumper on a car. The purpose of bumper detection is to detect objects that are not detected by the IR sensors, also adds to the character of the robot. A bumper layout is shown in figure 3.

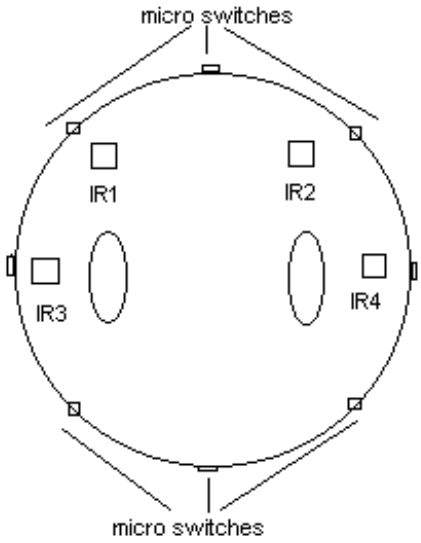

Figure 3.

Cadmium Sulfide cells

Photoresistors are very useful in distinguishing between light and dark areas, moving toward a luminous object and many other applications. For my robot three photoresistors well be implemented and strategically placed in order to take advantage of the software and detect light properly. Photoresistors are relatively easy to implement on the robot. The software is the brains behind the effectiveness of the photoresistor implementation. A "learned" algorithm will be used to give the photoresistors some learning behavior. This learning behavior will be used to remember differences between lighted objects found earlier in the photoresistor detect sequence and possibly make a match later in the detection sequence.

THEORY

The photoresistor circuit is comprised of a pull down resistor that is placed in a voltage divider configuration. The total resistance in this voltage divider circuit is.

$R_T = R + R_L$ (Please see the voltage divider circuit in figure 4)

According to Ohm's law, the current *I*, through the circuit is.

$I = V/R_T$

In order for the A/D converter in the MC68HC11 to measure a voltage, some current must flow into one or more pins (port PE0 to PE7) on the J4 header of the MC68HC11 board. The voltage divider output is given by.

$V_{PE()} = IR_L$

The resistance of the photoresistor falls as the light level increases. This means that the voltage at $PE_{()}$ (see note 1) decreases as a result of decreased resistance from the photoresistor. From this result, we can deduce that the resistance of the photoresistor is proportional to the voltage delivered to port $PE_{()}$. The 8-bit A/D converter in the MC68HC11 maps the variable voltage, $V_{PE()}$, into the range 0 to 255. A good compromise between sensitivity and range is achieved by adjusting R to a middle range exhibited by the photoresistor when exposed to direct light. Typically, photoresistors are made from cadmium sulfide (CdS) and are commonly found at Radio Shack and from various electronic component distributors.

Note 1. PE_0 refers to ports PE1 through PE7

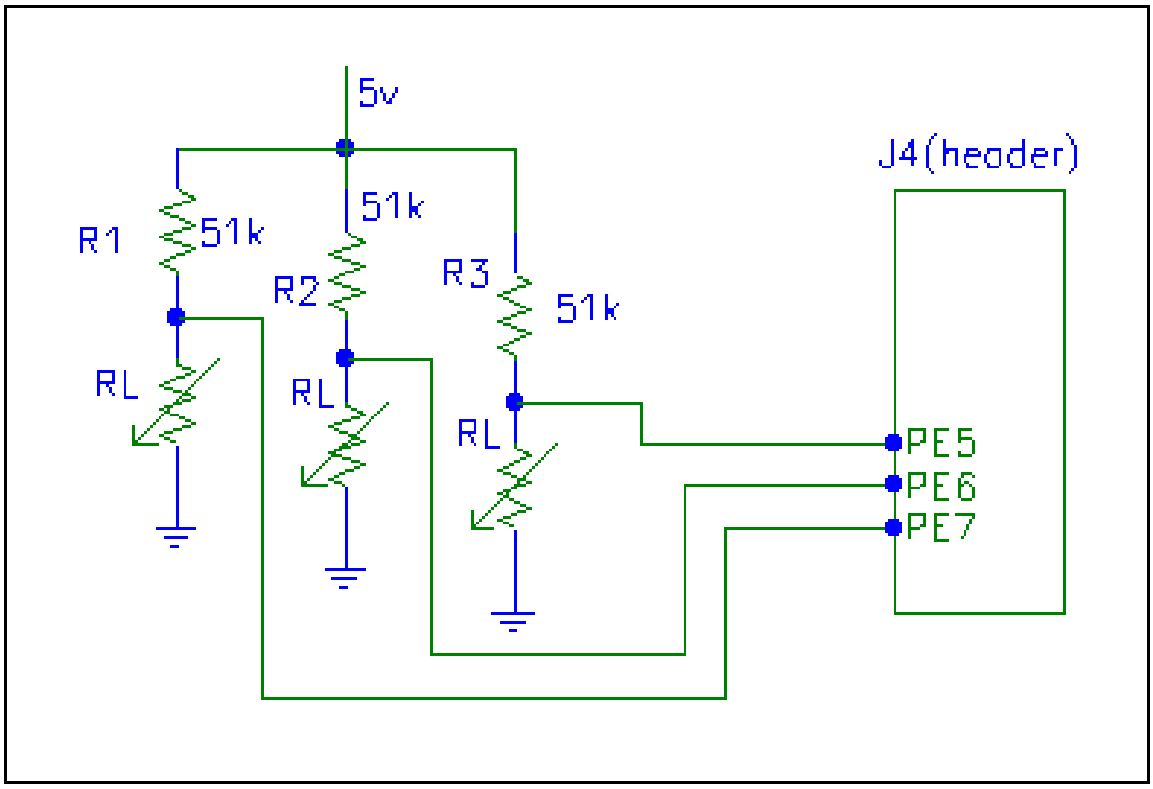

Figure 4. Voltage divider circuit

Robotic Arm Implementation

This robotic arm with have horizontal rotation. The amount of rotation will be from 0 degrees to 90 degrees counterclockwise. There will be one servo motor driving this robotic arm and one to control the gripper. The robotic arm will be made out of bass wood and plastic gears will be used to move the gripper claws. The purpose of this robotic arm is demonstrate the ability to pick up small light objects and deposit the objects into a remote container. The arm will detect objects within close proximity using IR detection. The IR detector and IR emitter is placed behind the gripper claws. In figure 5 shows the placement of such sensor.

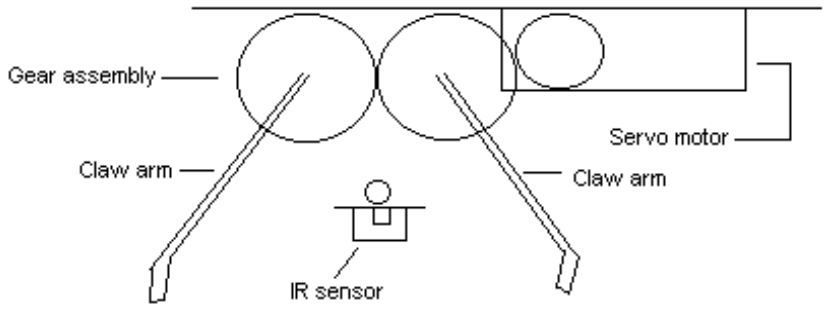

Figure 5. IR sensor placement.

The robotic arm is mounted to a plywood base (robotic arm plateform). This base is secured to four plywood posts that are mounted to the robot plywood base. A servo motor is fasten to the robotic arm platform and secured by two small pieces of plywood. The servo motor is fasten to the small pieces of plywood by screws. This servo motor provides rotational movement for the robotic arm. In figure 6, a diagram shows the implementation of such a servo motor.

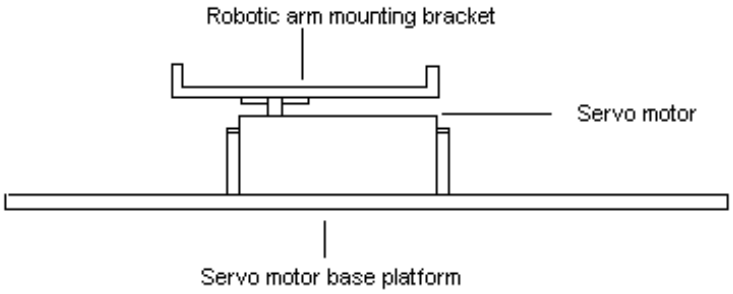

Figure 6. Servo motor base platform.

Conclusion

At this point in time Thor is able to perform object avoidance behavior, detecting light and pick up small objects using its robotic arm. Using only three types of sensors, (IR, contact sensors and light detection), the robot is able to implement a variety of behaviors. These behaviors are, object avoidance, collision avoidance and detection of light using Cadmium Sulfide cells. When these behaviors are combined it gives the robot a basic behavior characteristic and functionality. The robotic arm was implemented to give the robot some functional application. This particular application could be used in a factory environment where autonomous robots can be utilized in situations of part retrieval and placement. Such a situation would be in the manufacturing of a electronic or mechanical device, ie., VCR, TV set, various power tools, gear assemblies.

Documentation

J. L. Jones and A. M. Flynn, "Mobile Robots: Inspiration to implementation" , A. K. Peters, ltd., Wellesley Ma, 1993.

F. G. Martin, "The 6.270 Robots Builder's Guide" , F. G. Martin, 1992.

Motrola, MC68HC11 EVBU User's Manual, Motrola, Inc., 1992

Appendix

```
/* Motor/object avoidance control/analog input program
   and robotic arm control routine.
   For Thor the Wonder Robot
   By: Chris Parman  December, 12 1996
   IMDL laboratory, Professor Keith L. Doty
*/


/* constants and global variables */
  int left_eye;
  int center_eye;
  int right_eye;
  int motor_flag=0;
  int RIGHT_MOTOR = 0;
  int LEFT_MOTOR = 1;
  float SLOW_SPEED = 25.0;
  float MODERATE_SPEED = 50.0;
```

```
  float FAST_SPEED = 75.0;
  float servo_close = 90.0;
  float servo_open = 30.0;
  float servo_rotate = 90.0;
  float servo_return = 0.0;
  int near_left;
  int near_right;
  int far_left;
  int far_right;
  int gripper_switch;
  int servo_IR;
  int thr1 = 100;
  int thr2 = 115;
  int thr3 = 125;

/* read the status of the object avoidance IR sensors */
  void read_IR_sensors()
  {
       while(1) {
       poke(0x7000,0xff);
       near_left = analog(0);
       near_right = analog(1);
       far_left = analog(2);
       far_right = analog(3);
       }
  }

/* reads the status of the cadmium sulfide cells */
```

```
void light_sensor()
{
  while(1) {
    poke(0x4000,0b00000000);
    left_eye = analog(4);
    poke(0x4000,0b00001000);
    center_eye = analog(5);
/*    poke(0x4000,0b00010000);
    right_eye = analog(7); */


  }
}

/* read the IR sensor and gripper switch */
/* on the robotic arm and returns a value */
servo_sensor_read()
{
        while(1) {
        poke(0x4000,0b00010000);
        gripper_switch = analog(6);
        servo_IR = analog(7);
        }
}


/* determines what direction to turn  */
/* base on light sensor status        */
```

```
void light()
{
  while(1) {
    if (left_eye < 100) {
          accelerate2(SLOW_SPEED);
 }

    else if (center_eye < 100) {
        accelerate1(SLOW_SPEED);
         }
    else if (right_eye < 100) {
          accelerate3(SLOW_SPEED);
          }
    else if (center_eye > 100){
          motor(0,25.0);
          motor(1,25.0);
      motor_flag = 0;
    }
  }
}
```

```
/* determines what direction to turn */
/* base on the status of the object  */
/* avoidance status                  */
 void obstacle_avoidance()
```

```
{
 while(1) {


  if (far_right > thr1)
    {
     accelerate2(MODERATE_SPEED);
     motor_flag=1;
     }
    if (far_left > thr1)
    {
     accelerate3(MODERATE_SPEED);
     motor_flag=1;
     }


  if (near_left > thr1)
     {
          accelerate1(FAST_SPEED);
          motor_flag=1;
     }
  if (near_right > thr1)
     {
          accelerate1(FAST_SPEED);
          motor_flag=1;
     }

  if ((near_left < thr1) & (near_right < thr1))
```

```
        {
            if (motor_flag == 1)
                accelerate(FAST_SPEED);
            else
            {       motor(RIGHT_MOTOR,MODERATE_SPEED);
                    motor(LEFT_MOTOR,MODERATE_SPEED);
            }
            motor_flag = 0;
        }
   }
}

/* servo control routine */
{
        while(1) {
        if(servo_IR > thr2 && servo_IR < thr3)
        accelerate_servo1(servo_close);
        wait(500);
        accelerate_servo2(servo_rotate);
        wait(500);
        accelerate_servo1(servo_open);
        wait(500);
        accelerate_servo2(servo_return);
        wait(500);
        }
}
```

```
/* loop acceleration routines for controlling */
/* speed at which the motors and servos move  */
void accelerate(float speed)
{
	float i, inc;
	inc = speed/10.0;

	for(i=inc;i<=speed;i+=inc)
	{
		motor(0,i);
		motor(1,i);
		sleep(0.2);
	}
}
void accelerate1(float speed1)
{
	float j, inc1;
	inc1 = speed1/10.0;
	for(j=inc1;j<=speed1;j+=inc1)
	{
	motor(0,-j);
	motor(1,-j);
	sleep(0.15);
	}
}
void accelerate2(float speed2)
{
	float k, inc2;
```

```
        inc2 = speed2/10.0;
        for(k=inc2;k<=speed2;k+=inc2)
        {
        motor(0,-k);
        motor(1,k);
        sleep(0.15);
        }
}
void accelerate3(float speed3)
{
        float x, inc3;
        inc3 = speed3/10.0;
        for(x=inc3;x<=speed3;x+=inc3)
        {
        motor(0,x);
        motor(1,-x);
        sleep(0.15);
        }
}
void accelerate_servo1(float deg_servo1)
{
        float a, inca;
        inca = deg_servo1/10.0;
        for(a=inca;a<=deg_servo1;a+=inca)
        {
        servo_deg1(a);
        }
}
```

```
void accelerate_servo2(float deg_servo2)
{
        float b, incb;
        incb = deg_servo2/10.0;
        for(b=incb;b<=deg_servo2;b+=incb)
        {
        servo_deg2(b);
        }
}
```

```
/* this section enables all sensing and behavior routines */
void main()
{
  sleep(5.0);
  start_process(read_IR_sensors());
  start_process(obstacle_avoidance());
  start_process(light_sensor());
  start_process(light());
  start_process(servo_sersor_read());
  }
```