Intelligent Machine Design Laboratory

Professor Keith L. Doty

Scarab : A Collector Robot

Brad Malemezian

University of Florida

Electrical and Computer Engineering

September 15, 1997

Table of Contents

Abstract

My robot, named Scarab becuase of it looks like a beatle, is an autonomous mobile agent whose purpose is to collect object scattered about the room and bring them to the brightest position in the room. It is controlled by the Motorola 68HC11E9 EVBU board equipped with 32K bits of expansion memory. The most crucial part of my robot's design is its front mounted claw and platform. This feature gives Scarab its object manipulation capabilities. I did all of the programming of Scarab's behaviors in Integrated C. The most important of these behaviors is the object detection and capture behavior which takes the readings from the claw-mounted sensors and uses the data to cause the two servos to close and lift the claw properly. Scarab's software integrates the sensory data into the various behaviors which perform motor actuations so that the robot can perfom its function as a collector robot.

Executive Summary

To create Scarab, I had to integrate the four subsystems consisting of the mobile platform and gripping claw, motor and servo actuation, diverse sensors, and several behaviors into one entity that accomplises the goal of moving objects scattered about the room to one common location.

The mobile platform subsystem is the physical structure of the robot upon which all of the other hardware is built. It includes the front mounted gripping claw which is the main hardware responsible for collecting objects.

The actuation subsystem includes control over both motors and servos. The motors are responsible for moving the robot from place to place, while the servos are responsible for lifting and closing the claw.

The sensor subsystem is responsible for gathering data about the environment in a form that the robot can use to make decisions. My sensor suite includes IR and break beam sensors, light sensors, and bump and limit switches.

The behavior subsystem is the system that defines the actions that the robot takes based upon the sensor inputs. My robot has obstacle avoidance, object detection and capture, light following, and object deposit behaviors.

Together, these four subsystems make up the robot I have named Scarab. With them all working properly, Scarab is an effective collector robot.

Introduction

In creating my robot, I wanted to create a self-controlled entity that could accomplish a given task. The task that I have chosen for Scarab is that of object collection and relocation. This is not an easy problem - in order to get a working robot, I found that I had to make many assumptions and limitations to the environment. Even with these limitations, I felt that I could make something that with more time and resources had the potential to be a useful invention. With that goal in mind, I proceeded into the many details involved in building a functioning robot.

Integrated System

The integrated system for my robot takes as input the data from the senors, processes this data, makes a decision as to what behavior to follow, and finally initiates the desired actuation in the servos and motors. The integrated system incorporates both the hardware and the software; it allows them to work together to produce the desired results. Since Scarab is a collector robot, its integrated system is designed in a way to facilitate the collection and moving of objects. Scarab's integrated system consists of a mobile platform with front mounted claw, both motor and servo actuation, a sensor suite which includes IR, light, and bump detectors, and several behaviors that allow it to be a collector robot.

Mobile Platform

Scarab's platform consists of a baseplate upon which the various components are attached and a front mounted claw. The baseplate is cut from a 1/8 inch aluminum sheet into a 10 inch diameter circular shape with several holes for mounting components. The design is based upon Novasoft's TALRIK design with one exception: I cut a rectangular piece out of the front of it to allow for the attachment of the claw. I designed the front mounted claw to allow it to grip small cylindrical objects between one and two inches in diameter. (I use toilet paper rolls in my experiments) I cut these claws out of a 1/16 inch aluminum sheet and built a claw platform to mount the claws, servo, and gears. This claw system gives my robot the ability to physically grasp an object of the correct size.

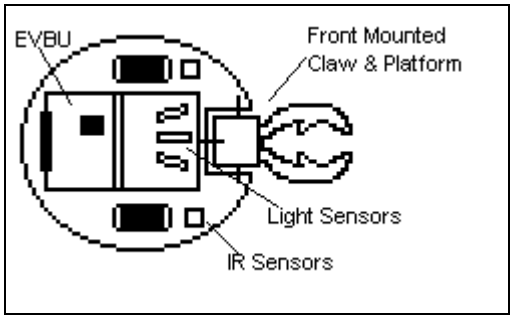

Figure 1 - Mobile Platform Design

Actuation

Since my robot has both motors (hacked servos) that drive the wheels to allow the robot to move and servos (unhacked) to control the claw movement, it has both motor and servo actuation routines.

The motor actuation routines are based upon input from three IR detectors, several bump sensors, and three light detectors. Based upon these inputs, the routine will tell the motors to cause the robot to turn left or right, go forward or backwards, or back left or

right. I have also implemented a motor control process in my program to allow constant acceleration. This routine keeps track of the current motor speeds and increments or decrements these speeds by a constant when the target and actual speeds are not equal. This process smoothes out the robot's motions and reduces the strain on the motor driver circuitry.

The servo actuation routines are responsible for controlling the claw in its two degrees of motion: opening/closing and lifting/ lowering. To open or close the claw, the robot turns the servo one degree every 30 milliseconds until it is to the desired angle or it detects that it has closed on an object. Since in my design the difference between an open and a closed clow is about 40 degrees, the process will take between one and two seconds. I use an idetical routine to lift or lower the claw, except there are not bump sensors - the servo just moves to a given position.

Sensors

My sensor suite consists of three infrared detecors, a break beam sensor, three visible light detectors, six bump sensors, and two limit switches. All of these sensors are connected in some way to an analog port to allow the robot to take easy and accurate measurements.

The infrared sensors I used on my robot are modified Sharp GP1U58Y IR detectors. I modified them so that they would output a analog signal rather than a digital one because the analog signal tells me more about the intensity of the detected IR and therefore the

distance to the obstacle. The IR detectors give Scarab a limited sight by measuring the intensity of reflected infrared light beams. This light is provided by the infrared light emmiting diodes mounted on the underside of the robot. The modified sensors output a voltage between 2 and 5 Volts. This sensor output is plugged directly into one of the EVBU's A/D ports and converts into a digital value ranging from 85 to 135. The IR detecting sensors provide crucial data for Scarab's obstacle avoidance behavior.

The break beam sensor consists of a hacked Sharp IR detector (as described above) and an IR LED mounted on the two opposing claws. This sensor operated very simply: if there is an object that breaks the beam (in the claw) the IR detector will have a low reading, otherwise it will have a high reading. This IR detector is also connected to an A/D port. This sensor is very simple but also very effective in detecting objects in the claw.

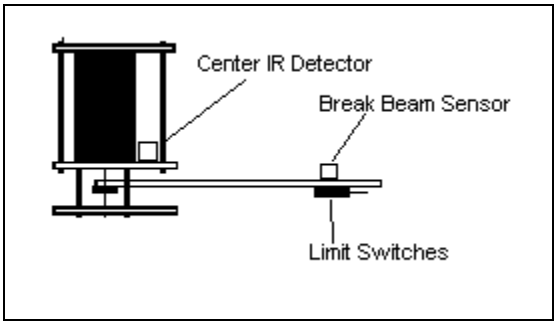

Figure 2 - Claw Mounted Sensors

The visible light sensors that I used are Radio Shack brand cadnium-sulfide cells. These sensors are basically resistors whose resistance depends upon the amount of visible light they detect. I wired each one in series with another resistor between +5V and ground as shown in figure 4. Since the robot will operate in lighted conditions which give

resistances between about 1K-ohms and 250K-ohms, I chose a series resistor with resistance of 14K-ohms, which is the geometric mean between the two limits and gives the highest data resolution. I mounted each light detector in a 1.5 inch piece of 1/4 inch diameter copper pipe and attached them on the front of the robot pointing left, forward, and right. With this harware intact, I am able to implement a easy light following behavior.

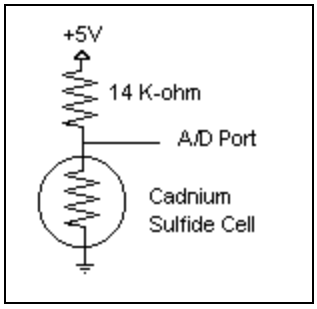

Figure 3 - CDS Schematic

The bump sensors are simple open or closed microswitches. When depressed, the switch is closed, otherwise it is open. To save A/D ports, I wired all of my bump sensors (and limit switches) through a network of resistors into one A/D port as shown in figure 4. Because the resistances are different for the each branch, I can tell which switch is depressed by the A/D reading. The bump sensors on my robot are necessary for those instances when the IR detectors do not see an object. When a bump is detected, my robot knows which seide the bump came from, so it knows which direction to go to escape.

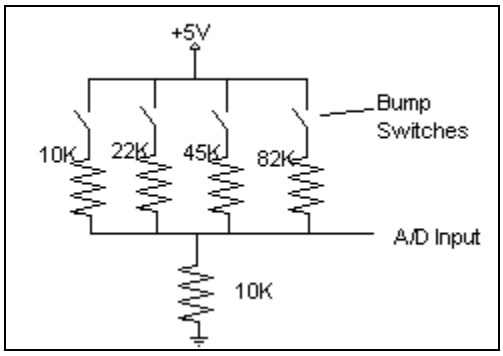

Figure 4 - Resistor Network Schematic

Th limit switches are similar to the bump sensors, but they are used for a different purpose. The limit switches on Scarab are mounted on opposing claws and are used to detect when an object has been grasped. The limit switches are connected to seperate branches of the resistor network mentioned above, so the robot knows whether or not both have been depressed. These sensors are a crucial part of the object recognition and grasping routines on my robot.

Behaviors

The behaviors that Scarab follows are obstacle avoidance, object detection and capture, light following, and object deposit. These four behaviors work together to accomplish the ultimate goal of collecting objects.

The obstacle avoidance behavior use data from the IR and bump sensors to determine the direction of movement for the robot. When the robot's IR sensors detect something that it considers an obstacle, it moves in a direction to hopefully avoid the object. If the bump sensors are depressed at any time, it will back up to free itself and continue on its way.

Object detection and capture is the behavior responsible for recognizing and picking up objects using the gripping claw. The robot detects a potential object when the center IR sensor sees something and the left and right IR sensors do not. The robot then moves forward until the break beam detects the object in the claw or the bump sensors are triggered. The latter condition signified a "miss" and results in an abort of the object retrieval. If the former is the case, the robot will close the claw until it is all the way closed or the limit switches are both depressed. The former once again signifies a miss and the retrieval is aborted, otherwise, the claw picks up the object and the behavior is complete.

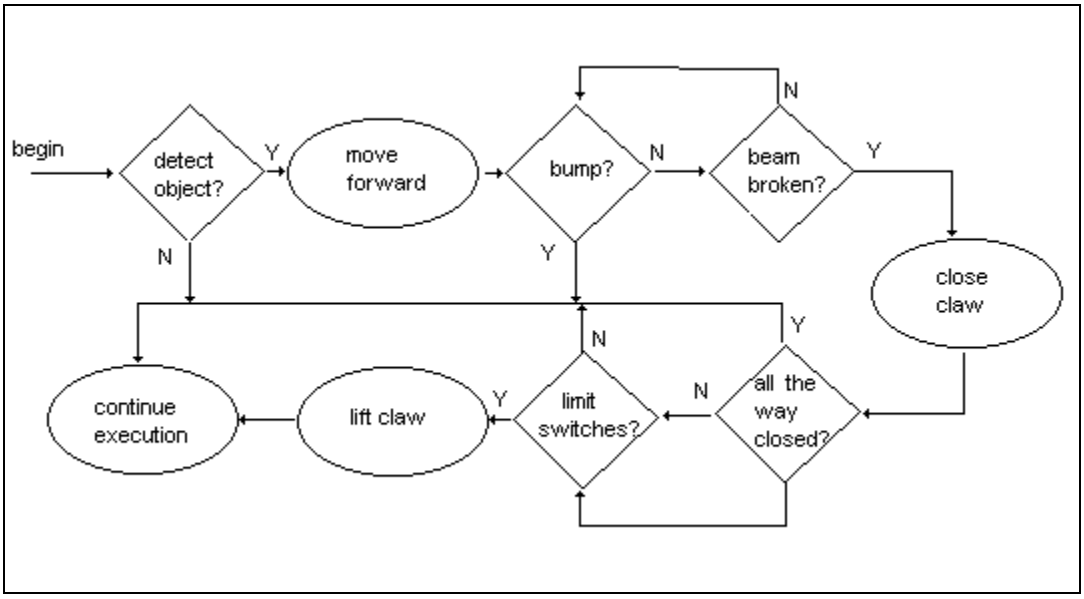

Figure 5 - Object Detection & Capture

The light following behavior allows Scarab to bring the object to the brightest location in the room. This behavior uses data from the left, right, and center light detectors and follows a simple algorithm to find the light source. If the light is brightest on the right side, turn right. If it is brightest on the left side, turn left. Otherwise, go forward. The

robots follows this algorithm until it reaches a place of sufficient brightness, when it proceeds to the object deposit behavior.

The object deposit behavior is simple. First, the robot lowers the claw and then it opens the claw. Next, it backs up and then turns right. With the object properly deposited, it begins wandering the room again until it finds another.

Conclusion

In just one semester, I feel that I have accomplished a great deal. I have designed and built a functioning robot. My robot's major weakness is its difficulty in correctly identifying objects to pick up and then grasping these objects. To overcome these limitations I need to add additional object detecting sensors and come up with some sort of object finding behavior. While my robot is not as complete as I would have liked it to be, with the given time and experience restraints, I can only call my finished product a success.

Appendix

/* Program Code */

```
int LEFT, MIDDLE, RIGHT;
int CLAW, BUMP;
int RCDS, LCDS, MCDS;
int THRESHOLD=95;
int LEFT_MOTOR=0;
int RIGHT_MOTOR=1;
int R_TARGET, L_TARGET;
int R_SPEED, L_SPEED;
int FULLSPEED=100;
int HALFSPEED=50;
```

```
int HUNTSPEED=13;
int MOTOR_INC=5;
int OPENCLAW=95;
int CLOSEDCLAW=60;
int DOWNCLAW=40;
int UPCLAW=0;
int HAVEOBJECT;

void wait(int m_second) {
 long stop_time;
 stop_time = mseconds() + (long)m_second;
 while(stop_time > mseconds())
   defer();
}

void waste_time() {
  int x,y;

  for(x=0; x<3000; x++)
    y=x*2;
}

/* reads the IR dectors and stores into LEFT, MIDDLE, RIGHT, and CLAW */
/* reads the bump sensor network analog value and store into BUMP */
/* reads the cadnium sulfide cells and stores into LCDS, MCDS, and RCDS */
void read_sensors(){
  while(1){
    LEFT=analog(1);
    MIDDLE=analog(2);
    RIGHT=analog(0);
    CLAW=analog(3);
    BUMP=analog(7);
    LCDS=analog(4);
    MCDS=analog(5);
    RCDS=analog(6);
  defer();
  }
}


/* controls the servos using linear acceleration */
void drive_motors(){
 while(1) {
  if(R_SPEED<R_TARGET) {
    R_SPEED=R_SPEED+MOTOR_INC;
    if(R_SPEED>R_TARGET)
      R_SPEED=R_TARGET;
  }
  else if(R_SPEED>R_TARGET) {
    R_SPEED=R_SPEED-MOTOR_INC;
    if(R_SPEED<R_TARGET)
      R_SPEED=R_TARGET;
  }
```

```
  if(L_SPEED<L_TARGET) {
    L_SPEED=L_SPEED+MOTOR_INC;
    if(L_SPEED>L_TARGET)
      L_SPEED=L_TARGET;
  }
  else if(L_SPEED>L_TARGET) {
    L_SPEED=L_SPEED-MOTOR_INC;
    if(L_SPEED<L_TARGET)
      L_SPEED=L_TARGET;
  }

  motor(LEFT_MOTOR,(float) L_SPEED);
  motor(RIGHT_MOTOR,(float) R_SPEED);

  defer();
 }
}


/* direction routines - cause robot to go in specified direction */
void turn_right() {
  R_TARGET=-HALFSPEED;
  L_TARGET=HALFSPEED;
}

void turn_left() {
  R_TARGET=HALFSPEED;
  L_TARGET=-HALFSPEED;
}

void back_right(int immed) {
  R_TARGET=-HALFSPEED;
  L_TARGET=-FULLSPEED;
  if (immed) {
    R_SPEED=R_TARGET;
    L_SPEED=L_TARGET;
  }
}

void back_left(int immed) {
  R_TARGET=-FULLSPEED;
  L_TARGET=-HALFSPEED;
  if (immed) {
    R_SPEED=R_TARGET;
    L_SPEED=L_TARGET;
  }
}

void go_forward() {
  L_TARGET=FULLSPEED;
  R_TARGET=FULLSPEED;
}

void stopm() {
```

```
  L_TARGET=0;
  R_TARGET=0;
}

/* causes motors to go target speed */
void gotargspd() {
  while(L_SPEED!=L_TARGET || R_SPEED!=R_TARGET)
    defer();
}

/* routine to approach and pick up object using claw */
void get_object() {
  int CLAWANGLE=OPENCLAW;
  int LIFTANGLE=DOWNCLAW;

  stopm();
  gotargspd();
  poke(0x7000,1);
  servo_deg1((float) CLAWANGLE);
  servo_deg2((float) LIFTANGLE);
  wait(1000);

  L_TARGET=HUNTSPEED;
  R_TARGET=HUNTSPEED;
  L_SPEED=L_TARGET;
  R_SPEED=R_TARGET;

  while(CLAW>THRESHOLD && BUMP<2);

  if (BUMP>=2) {
    HAVEOBJECT=0;
    poke(0x7000,254);
  }
  else {
    stopm();
    L_SPEED=L_TARGET;
    R_SPEED=R_TARGET;

    wait(500);

    while((BUMP<70 || BUMP>73) && CLAWANGLE>CLOSEDCLAW-10) {
      CLAWANGLE=CLAWANGLE-1;
      servo_deg1((float) CLAWANGLE);
      wait(30);
    }

    if (CLAWANGLE == CLOSEDCLAW-10) {
      HAVEOBJECT=0;
      CLAWANGLE=OPENCLAW;
      servo_deg1((float) CLAWANGLE);
      poke(0x7000,254);
    }
    else {
      HAVEOBJECT=1;
```

```
      while(LIFTANGLE>UPCLAW) {
        LIFTANGLE=LIFTANGLE-1;
        servo_deg2((float) LIFTANGLE);
        wait(30);
      }
      poke(0x7000,250);
    }
  }
  wait(300);
}


/* calls appropriate direction routine based upon sensor data */
void navigate(){
  while (1){
    if (BUMP >= 84 && BUMP <= 87 ) {
      back_left(1);
      wait(1000);
    }
    else if (BUMP >= 127 && BUMP <= 130 ) {
      back_right(1);
      wait(1000);
    }
    else if (LEFT > THRESHOLD && RIGHT <= THRESHOLD)
      turn_right();
    else if (RIGHT > THRESHOLD && LEFT <= THRESHOLD)
      turn_left();
    else if (LEFT > THRESHOLD && RIGHT > THRESHOLD)
      back_left(0);
    else if (MIDDLE > THRESHOLD && HAVEOBJECT==0)
      get_object();
    else if (LCDS<MCDS && LCDS<RCDS && HAVEOBJECT==1)
      turn_left();
    else if (RCDS<MCDS && RCDS<LCDS && HAVEOBJECT==1)
      turn_right();
    else
      go_forward();

    defer();
  }
}

/* initializes variables and spawns processes */
void main(){
  stopm();
  L_SPEED=L_TARGET;
  R_SPEED=R_TARGET;
  HAVEOBJECT=0;
  poke(0x7000,254);
  servo_on();
  servo_deg1((float) OPENCLAW);
  servo_deg2((float) DOWNCLAW);

  waste_time();
```

```
  start_process(read_sensors());
  start_process(drive_motors());
  start_process(navigate());
}
```