AMY PURKEY

INTELLIGENCE MACHINE DESIGN LABORATORY
SNARF – IMDL COMPETITION

SPRING 1998

April 25, 1998

TABLE OF CONTENTS

ABSTRACT

I designed an autonomous robot, Snarf. The robot was designed to enter in to the IMDL competition in the Spring semester of 1998. The robot will autonomously navigate around a playing field. It will pick up three different objects, six ounce can of tomato paste, small box of tea bags, and a one pound bag of rice. The robot will distinguish between these items and dump them into an appropriate bin.

The body was made out of one-quarter inch birch plywood. The scoop was made out of an aluminum rain diverter panel. I used two 40 oz-inch servos to power the wheels and two 40 oz-inch servos to power the scoop. I used a Talrik electronics kit. This kit included a MRC11 board, a MRSX01 board, and a MB2325 board. I used IR sensors for navigation. I also used bump switches, and limit switches for control. I also used a G2D05 sensor from Sharp electronics.

EXECUTIVE SUMMARY

I designed an autonomous robot for the IMDL Spring 1998 competition. The robot was called Snarf. The robot was designed to navigate a playing field finding three different objects and dumping them in their appropriate bin.

Snarf was designed on a square platform made of one-quarter inch birch plywood. The items were to be picked up by a rectangular scoop made of aluminum rain diverter. Snarf navigated by use of IR sensors. A G2D05 sensor made by sharp electronics detected items in the scoop.

I used a Talrik electronics kit in the robot. This kit included a MRC11 board, a MRSX01 board, and a MB2325 board. I used an interpreted language, IC to communicate through the communication board to the robot.

I accomplished navigating the arena by wall following, picking up objects and depositing them in the bins. It did not all come together perfect though.

INTRODUCTION

I am designing an autonomous robot to enter into the IMDL Spring 1998 competition. The competition consists of navigating around a confined arena, gaining control of three objects, and placing the objects in their appropriate bins.

I will use electronics from a Talrik electronics kit. The kit includes a MRC11 board, a MRSX01 board, and a MB2325 board. I will use the communication board to communicate with the robot through an interpreted language, IC.

I will design the robot using a square platform made out of birch plywood. The robot will gain control of the items through use of a scoop. The scoop will be made of aluminum rain diverter. I will use two wheels mounted on two servos to drive the robot. I will also use two additional servos to power the scoop.

I will be using four different types of sensors. I will be using limit switches to control the servo. I will be using IR sensors to navigate around the arena. I will be using bump switches to know when I am against a wall. I will also be a sensor made by Sharp, the GP2D02, to detect if I have something in the scoop.

The robot will navigate the arena by wall following in two different directions. The scoop will be raised by pivoting about an axle at approximately 120 degrees. It will also be lowered in the opposite direction.

INTEGREATED SYSTEM

I have designed an autonomous robot to enter into the IMDL Spring 1998 competition. The robot is powered by two 40 oz-inch servos. It is built on a square base and uses a scoop to lift objects. The scoop is powered by two 40 oz-inch servos.

The scoop is centered in the front of the robot and rotates about an axle approximately 9 inches above the ground. The scoop rotates 120 degrees from its horizontal position and dumps the objects over the body itself and into the bin.

The scoop will be stopped on both ends using limit switches. The limit switches give either a five-volt reading or a ground reading.

I am using IR sensors to detect my distance from the wall. I will use my initial IR readings to keep me that distance from the wall. I will perform a wall following behavior using these sensors.

I will detect if my robot has run into a wall by using bump switches. These bump switches react the same way as the limit switches. They give a five-volt reading if the switch has been hit and a ground reading if it has not been hit.

I will be using a Sharp GP2D05 sensor to detect if anything is on my scoop. This sensor uses IR but unlike the regular IR sensors this sensor can be limited to a specific distance from 10cm to 80cm.

MOBILE PLATFORM

My robot was built on a square platform using birch plywood. From the platform I cut out four-inch blocks from the back corners to mount my wheels. I used a square platform because I knew I wanted to wall follow and I thought that straight edges on the robot would make this task easiest. I also knew I was using a scoop to pick up my objects and thought a scoop would fit best on a squared front platform. I also added two castor wheels to the front of the platform for stability.

The scoop of my platform extended one-half of an inch outside of the robot. I did this because I wanted to keep my wheels from rubbing the wall while wall following but I still wanted to have my scoop close enough to the wall to pick up any thing close to the wall. When designing my scoop I made it about 18 inches tall. I did this so I could pull the scoop from the top and have it rotate about an axis approximately half way up and use leverage to avoid some of the problems of lifting the maximum three pounds of weight. The scoop rotates 120 degrees and the objects in the scoop slide down the back wall of the scoop and into the bin.

I ran into several problems with my platform design. The main problem I ran into was having the two castor wheels. Three points define a plane and with two wheels and two castor wheels I always had one wheel off the ground. My main problem here was the wheel that mainly stayed off the ground was one of my driving wheels. I eventually fixed this problem by moving the driving wheels up front and having only one castor wheel in the back. I tried having only one castor wheel up front but the platform was too unstable. I was also using driving wheels that had very little surface area touching the ground. Using a thicker wheel may have solved this problem.

I also ran into problems with my scoop being too large up front. It was difficult to turn corners while wall following because with the large scoop my robot was essentially blind up front. I solved this problem by wall following until the bump switches located on the front of the scoop were activated. Then I reversed the robot for a specific time and turned the corner and continued to wall follow. Also the scoop made it difficult to wall follow because since it extended about 15 inches past the wheels when I would wall follow and the front end would get too far from the wall the code said to turn back towards the wall. The angle the front was turned into the wall the scoop would sometimes hit before the front sensor read the appropriate value. I somewhat solved this problem by adjusting the speed in which the robot turned back towards the wall.

ACTUATION

I used four 40 oz-inch servos to actuate my robot. Two of the servos were used to drive the robot. The other two were used to drive the scoop. I mainly chose these servos because they were cheap. I felt that they would efficiently drive my robot. I also felt that if I used the correct pulley system that they would also have enough power to pick up the maximum weight of three pounds.

In servos controlling the driving wheels I used a Talrik hack. This allowed me to use the servos as motors. This hack is located in the TJ assembly manual. I did not need high-powered motors. I did learn that these particular servos bogged down in certain situations and I think motors with a little more torque should have been used.
For the motors I used mainly six different algorithms. I used an algorithm for both motors forward, both motors backwards, left motor forward, left motor backward, right motor forward, right motor backward. The code is shown in the appendix.

The servos controlling the scoop were hacked using the hacking technique out of the TJ manual. This allowed me to essentially use the servos as motors but run them off of the servo ports. With this hack I did not have control of the position of the servo. The servo rotated 360 degrees freely. With this hack I did have control of the speed of the servos. The center position of the servo is 90 degrees. If you activated it at an angle of less than 90 degrees it would rotate in one direction. If you activated it at an angle of greater than 90 degrees it would rotate in the opposite direction. The closer you chose to 90 degrees the slower the servo would turn. The farther you chose from 90 degrees the faster the servo would turn. This feature was needed in my robot design because I wanted to lift the objets as fast as possible. Although the scoop was returned to its starting position by its own weight. In this instance I needed the servos to turn slower so it would keep tension on the string while the scoop was falling.

The only problem I had with this approach was the servos had to be centered at 90 degrees to have accurate control of the speed and direction of the servos. Over short periods of time they would bounce and become centered at different points. This would cause one servo to rotate faster than the other or possibly in a different direction.

For the servo actuation I used two basic algorithms. I used one algorithm using a faster rotating servo command to raise the scoop. I also used a slower rotating servo command rotating the opposite direction to lower the servo. This code is located in the appendix.

SENSORS

IR Sensors

In this project I used four main sensors. The first and most abundant sensor I used was IR. I used seven of these sensor to wall follow and to detect when I was in front of the bins. The IR receivers were hacked using the IR hack given in the Talrik manual.
The code I wrote for the wall following was to initially read the values of my sensors, then if the new values of the sensors differed from the initial readings to move in the appropriate direction. The code I wrote for detecting the bins was I had one sensor on the bottom of the platform. When this sensor got to the center bin it would see nothing. So I had a piece of code that read when that sensor was below 90 and I knew it was then in front of the bin.

The biggest problem I had with this sensor was it is not consistent in its readings. Depending on different light settings you would get different readings. Also I found when you would get extremely close to the wall the receiver would not pick up any of the light the IR emitter was giving off. This problem was fixed by moving the emitter on top of the receiver.

Limit Switches

The second sensor I used was the limit switch. This sensor was used to stop the servo. Since I hacked the servo I had no control over when they stopped because I could not stop them at a specific degree. So I used limit switches to do this. I had one switch at each maximum position of my scoop. The code I wrote for this sensor simply stopped the scoop when the limit switch was triggered. The only problem with this switch was that if it were not triggered in the right spot sometimes it would not be activated.

Bump Switch

The third switch I used was a bump switch. I used this sensor to tell if my robot had run into a wall. If the bump switch was triggered the robot was at a wall. The code for this sensor read the switches and if one was activated it performed the task related with that particular switch being activated. This switch had the same problem as the limit switch in that it was not always activated properly.

Sharp GP2D05

This sensor uses IR. It is given a pulse from the out1 pin through a voltage divider and sends IR pulses. This sensor was used to detect if something was on my bin. This sensor is superior to the general IR sensors in that you can fine-tune them to a specific distance by turning a screw that activates a potentiometer. This sensor ranges from 10 cm to 80 cm. The main problem with these sensors is you have to give them a pulse you create but it has to be within the limits of the specifications for the sensor. The sensors also blew out very quickly. I had to write code for the pulse given to the sensor and also code to tell if something was in my scoop.

All code and circuit diagrams for all sensors are given in the appendix.

BEHAVIORS

I had several behaviors in my program to complete the project. My main behaviors were wall following, finding the bins, raising the scoop, and lowering the scoop. I also had several other behaviors that integrated the different sensors.

Wall Following

In my program I needed to wall follow in both directions. I needed to wall follow to the left to be able to find the objects. I needed to wall follow to the right to make sure that after I picked up an object I did not run into any other ones.

Raising the Scoop

I used a simple algorithm to raise the scoop. This consisted of turning the servo at a quick speed in a clockwise direction. I used the limit switches above to tell when the scoop was at its maximum raised position. I ran into problems using the servos and the motors at the same time. The program IC does not like to run the servos at the same time it runs the motors. This will cause the motors to shake and not drive properly. If you do this you must use the sefvo_off() command to turn off the servos before turning on the motors.

Lowering the Scoop

I used basically the same algorithm to lower the scoop as I did to raise it. The difference being I activated the servos in a counter clockwise direction at a slower speed. I also used a limit switch at the opposite end to know when the scoop was at its maximum lowering point.

Finding the Bins

I found the bins by detecting when my sensors located under the platform read a value less than it normally read. This was accomplished by having the sensors low enough so that they would see under the center bin and not reflect any IR.

All code for all behaviors is given in the appendix.

EXPERIMENTAL LAYOUTS AND RESULTS

The main experiment I performed dealt with navigating around the arena. Wall following was the biggest one. I had to figure out a way to navigate around the arena and pick up the objects. I chose to do this by wall following. I had to use different speeds to move my robot away from the wall if it was too close. I had to also experiment with what speeds moved the robot back towards the wall if it was too far away. I also had to experiment with different speeds with respect to different values of IR readings.

I also had to experiment with the Sharp sensor. I had to find a pulse that would correspond with the sensor. I also had to experiment with the different distances I needed for detecting objects on my scoop.

CONCLUSION

In this project I accomplished many of my goals. I designed and built an autonomous robot. I came up with a plan to navigate the arena and score points and my plan worked. I used different sensors to accomplish different tasks like wall following and scoop limits. I developed algorithms such as wall following in two directions, lifting my scoop, lowering my scoop, and finding the bins. I learned a great deal about the boards I was using. I learned the limitations of certain aspects of the board such as having to mux out analog ports and using the cds cells. I also learned limitations of IC such as not being able to use the servos and motors at the same time. I also learned how different chips on the board worked and how different ports on the board worked. I also learned how processes work using IC and other aspects of coding.

Although I accomplished a lot in the project I did not accomplish everything I had hoped to accomplish. I had intended to detect which items were on my scoop by the amount of current flowing through the motors. I did not accomplish this task. I did accomplish a good wall following routine. However because of last minute design changes to my robot the routine I originally had did not work as good for my new design. I also failed to be able to pull everything together at one time. Each individual part of my robot worked but pulling it all together was the tricky part. I could pick up objects and put them in the bin but things did not always go as planned.

I would like to finish pulling everything together in my robot. I would like to get it to navigate the entire playing field and not just part of it. I would like to change my platform somewhat to make it better suit the needs of the contest. I would do this by using thicker wheels as well as finding a way to only have three wheels on the robot.

APPENDIX

```
/*motor routines*/

  int m=0xff;
  int v=peek(0xffb8);
  int q=(v|0x1f);

    void forward()
    {
     poke(0xffb8,m);
     motor(0,81.0);
     motor(1,100.0);
    }
    void motor1bk()
    {
     int m=(m&0xbf);
     poke(0xffb8,m);
     motor(1,100.0);
    }
    void motor0bk()
    {
     int m=(m&0x7f);
     poke(0xffb8,m);
     motor(0,100.0);
    }
    void motor1fd()
    {
     int m=(m|0x40);
     poke(0xffb8,m);
     motor(1,100.0);
    }
    void motor0fd()
    {
     int m=(m|0x80);
     poke(0xffb8,m);
     motor(0,100.0);
    }

    void bothback()
    {
     poke(0xffb8,0x3f);
     motor(0,100.0);
     motor(1,100.0);
    }




/*Scoop dump and return code*/


int pid;
int m =(0xff);
void scoop()
{
        int done = 0;
        servo_deg(180.0);
        servo_on();
        m=(m|0x1f);
        m=(m&0xea);
        while(!done)
        {
        poke(0xffb8,m);
        if (analog(0)>100)
        done = 1;
        }
        servo_off();
        msleep(6000L);
        servo_deg(70.0);
        servo_on();
        m=(m|0x1f);
        m=(m&0xeb);
        done = 0;
        while(!done)
        {
        poke(0xffb8,m);
        if (analog(0)>100)
        done = 1;
        }
        servo_off();
}

void main()
{
        pid=start_process (scoop());
}


/*Wall following routine*/

void main()
{
        int default_front = analog(3);
        int default_back = analog(6);
        int done = 0;
        int finish = 0;
        int exit = 0;
        while(!done)
        {
           int IR_front = analog(3);
           int IR_back = analog(6);
            m=(0x1f|m);
            m=(0xec&m);
            poke(0xffb8,m);
            if (analog(0)>100)
            {
              m=(0x1f|m);
              m=(0xed&m);
              poke(0xffb8,m);
              if (analog(0)>100);
              {
                m=(m&0x3f);
                poke(0xffb8,m);
                motor(0,100.0);
                motor(1,0.0);
                msleep(1000L);
                done = 1;
              }
            }
            else if (default_front - IR_front > 1)
              {
               motor(0,50.0);
               motor(1,100.0);
              }
            else if (IR_front < 90 )
              {
                motor(0,100.0);
                motor(1,60.0);
              }

            else if ((IR_front - default_front) > (IR_back -
default_back))
              {
               motor(0,100.0);
               motor(1,50.0);
              }
            else if ((IR_front - default_front) < (IR_back -
default_back))
              {
               motor(0,100.0);
               motor(1,100.0);
              }
            else
              {
               motor(0,100.0);
               motor(1,60.0);
              }
           }


int m = (0xff);
void main()
{
        int default_front = analog(7);
        int default_back = analog(5);
        while(1)
        {
           int IR_front = analog(7);
           int IR_back = analog(5);
            m=(0x1f|m);
            m=(0xec&m);
            poke(0xffb8,m);
            if (analog(0)>100)
            {
              m=(0x1f|m);
              m=(0xed&m);
              poke(0xffb8,m);
              if (analog(0)>100);
              {
                m=(m&0x3f);
                poke(0xffb8,m);
                motor(0,100.0);
                motor(1,100.0);
                msleep(2000L);
              }
        }
            else if (default_front - IR_front > 1)
              {
               motor(1,0.0);
               motor(0,100.0);
              }
            else if ((IR_front - default_front) > (IR_back -
default_back))
              {
               motor(1,100.0);
               motor(0,65.0);
              }
            else if ((IR_front - default_front) < (IR_back -
default_back))
              {
               motor(1,40.0);
               motor(0,100.0);
              }
            else
              {
               motor(0,100.0);
               motor(1,90.0);
              }
        }
}
```