Lab 05
CPE470
Team 6 Report

Luke A. Fraser, Jessica E. Smith

28th October, 2013

Introduction

In this Lab we designed and built a NXT robot that was able follow the spiral line in the LME lab room. Along with following the line the robot also needed to detect different colors on the table and either ignore or recognize depending on the portion of the challenge it was on. The robot was intnded to first ignore and continue over a green line and then when it detects a black line start following that line until it reaches a yellow square. The next stange consists of the team assisting the robot with sound to direct it towards a given sectino of the playing field. All of this was timed and the faster the robot was able to complete the given task the more points a team would receive.

Hardware & Design

The line following challenge required the use of more sensors than any previous project and inorder to deal with additional sensors we used the multiplexor given to us. This allowed the use of both light sensors, both ultra-sonic sensors, color sensor, and the sound sensor on a single NXT device. The multiplexer was different from the previous ways of communication with sensors. This proved to be problematic when trying to mix native BRICX code and the multiplexer header code, but after some frustration all the problems with the multiplexer were worked out.

Another frustrating challenge of the this project was attempting to attach every sensor that we needed to the robot. This proved to be difficult just because there were so many different pieces that need to be positioned in very close to each other. This did not allow for easy adjustments of the robot to be made. The most difficult part was getting the color sensor and light sensors in the right positions in order to acquire useable data to follow the line.

Discussion

The robot setup for this competition was not ideal. To improve the robot in the future I would attach the sensors differently. The sensors were to sensative to movement and had to be positioned very well in order for the system to work. This issue was most evident in the color sensor. The color sensor was positioned too high on our robot and was not able to get a good reading of the colors present on the mat. The sensor should have been placed closer to the ground in order to get a better reading of the colors. The color

sensor also should have been placed slightly in front of the light sensors so that at the end of the light the color sensor would read yellow before turning away from the line. In tests before the competition our robot would sometimes miss the yellow square at the end of the line because it was too far back on the robot. overall the hardware setup was not ideal.

Software Design

The software designed for this assignment followed a simple priniciple of going back and forth across the line moving little by little until the end of the line was found. This was not the method that we started with, but is what we found that produced the best results for the LME Lab room line mat. The reason we chose this method was to deal with the issue of corners on the mat. Our first attempts did not reliably follow the line on the table and new features were added to get the robot to make the turns.

Ultimately the piece of code that helped the most was how we handled the case of double black. The case in which both sensors read black. This senario would sometimes occur when the robot reached a corner in the line. In order to deal with this problem, because the each turn was gauranteed to be a right turn we decided that when a double black is encountered the robot should continue to torun right and not attempt to correct in anyway. This addition to the robot enabled us to remain on the line evan during sharp turns. It also helped with there were other lines interfearing with the active line the robot was suposed to be following. In the end the robot was succesfully able to follow the line in the competition with only minor issues.

Discussion

The program written for this assignment was designed specifically for the line in the lab room. Because of this the robot will not perform well following an abitrary line. The system was designed for right turns only and had a preference to travel to the right when it was lost. A better system would need to be considered if the line was a of a different shape.

```
1
2 #include "HTSMUX-driver.h"
3
4 #define LIGHT_PORT_1 msensor_S3_1
5 #define LIGHT_PORT_2 msensor_S3_2
6 #define COLOR_PORT msensor_S3_3
7 #define SOUND_PORT IN_2
8
```

```
 9 mutex moveMutex;
10 bool findHome = false;
11 float A_SPEED = 1;
12 float B_SPEED = .9;
13 int BLACKL = 350;
14 int BLACKR = 400;
15 int threshold = 90;
16 bool lineFound = false;
17 int BLACKLINE = 4;
18 bool seenBlack = false;
19
20
21 /*////////////////////////
22       FIND LINE
23 *////////////////////////
24
25 task find_line(){
26   Acquire(moveMutex);
27   while(lineFound == false){
28     OnFwd(OUT_A, 60*A_SPEED);
29     OnFwd(OUT_B, 60*B_SPEED);
30     ClearScreen();
31     int color =  smuxSensorHTColorNum(COLOR_PORT);
32     NumOut(0, LCD_LINE4, color);
33     Wait(100);
34     if(color <= BLACKLINE && seenBlack == true){
35       lineFound = true;
36     }
37     if(color <= BLACKLINE && seenBlack == false){
38        seenBlack = true;
39     }
40
41   }
42   Release(moveMutex);
43 }
44
45 /*////////////////////////
46       LINE FOLLOW
47 OUR STRATEGY:
48   Keep turn towards the line
49 when you see the line. If
50 both see the line, do a sweeping
51 turn right
52 *////////////////////////
53
54
55 void moveLeftSideM(){
56     OnFwd(OUT_A,60*A_SPEED);
57     OnFwd(OUT_B,20*B_SPEED);
```

```
58      Wait(120);
59  }
60
61  void moveRightSideM(){
62      OnFwd(OUT_B,60*B_SPEED);
63      OnFwd(OUT_A,20*A_SPEED);
64      Wait(120);
65  }
66  void slowTurnRight(){
67      OnFwd(OUT_B,50*B_SPEED);
68      OnFwd(OUT_A,30*A_SPEED);
69      Wait(200);
70
71  }
72
73  void turnaround(){
74       OnFwd(OUT_B,90*B_SPEED);
75       OnFwd(OUT_A,-90*A_SPEED);
76       Wait(400);
77  }
78
79  void moveBothSideM(){
80     OnFwd(OUT_B, 100*B_SPEED);
81     OnFwd(OUT_A, 100*A_SPEED);
82     Wait(300);
83  }
84  // turn around quickly until a noise then go at full speed that way
85  task find_home(){
86       while(true){
87           if(findHome == true){
88                 // input soundvalue
89
90                 if(Sensor(SOUND_PORT) > threshold){//need varibles
91                    Acquire(moveMutex);
92                    moveBothSideM();
93                    Release(moveMutex);
94                 }
95                 else{
96                      Acquire(moveMutex);
97                      turnaround();
98                      Release(moveMutex);
99                 }
100          }
101      }
102 }
103
104 // Checking sensors for seeing the line
105
106 task check_line(){
```

```
107     bool turn = false;
108
109     while(findHome==false){
110         ClearScreen();
111         int right = smuxSensorLegoLightRaw(LIGHT_PORT_1);
112         NumOut(0, LCD_LINE1, right);
113         int left = smuxSensorLegoLightRaw(LIGHT_PORT_2);
114         NumOut(0, LCD_LINE2, left);
115
116         /*if(right > BLACKR && left > BLACKL){
117             Acquire(moveMutex);
118             slowTurnRight();
119             Release(moveMutex);
120         }
121         else if(right > BLACKL)
122         {
123            Acquire(moveMutex);
124            moveLeftSideM();
125            Release(moveMutex);
126         }
127         else if(left > BLACKR){
128            Acquire(moveMutex);
129            moveRightSideM();
130            Release(moveMutex);
131         }*/
132        if(right < BLACKR && left < BLACKL){
133            turn = true;
134        }
135         if(turn==true){
136             if(right < BLACKR){
137                 Acquire(moveMutex);
138                 moveRightSideM();
139                 Release(moveMutex);
140                 turn=false;
141             }
142         }
143         else{
144             if(left < BLACKL){
145                 Acquire(moveMutex);
146                 moveLeftSideM();
147                 Release(moveMutex);
148                 turn=true;
149             }
150
151         }
152         int colorVal = smuxSensorHTColorNum(COLOR_PORT);
153         if(colorVal > 4 && colorVal < 8){
154             findHome = true;
155             PlayTone(1500, 1000);
```

```
156             Wait(1000);
157             TextOut(0, LCD_LINE4, "FINDING HOME" );
158         }
159
160      }
161
162 }
163
164 task main(){
165     SetSensor(S3, SENSOR_LOWSPEED);
166     SetSensorSound(SOUND_PORT);
167     if (!HTSMUXscanPorts(S3)) {
168        // Scan failed, handle the error
169        TextOut(0, LCD_LINE1, "Scan failed!");
170        Wait(1000);
171     }
172     smuxSetSensorLegoLight(LIGHT_PORT_1, true);
173     smuxSetSensorLegoLight(LIGHT_PORT_2, true);
174
175     Precedes(find_line, check_line, find_home);
176
177 }
```