**Post-Lab 8 Report Alan Palayil**

ECE 100-001 Teammates:Matthew V., Matthew G.

Prof. Oruklu Lab Date 11/01/19

TA: Rafael Due Date 11/08/19

**Problem Statement**

Program a HandyBug robot to play the game “Mint Shuffle X”. The robot needs to be programmed to push the first puck in its own corner and then to push off the other teams pucks to its corner for the robot to win. Trials with touch sensors and light sensors will be done to make the most efficient robot.

**Investigation/Research**

In Mint Shuffle X, two robots are placed in two sides of the board. The goal is to take the puck and push it one of two white corners. Once a puck is pushed into one of the white corners, it needs to find the other pucks and push them to score points.

Using multitasking this game can successfully accomplished. From Robotic Explorations multitasking is defined as “where the robot has several ‘behaviors’ running at the same time” (Martin 209). For the robot to use both the touch and light sensors successfully, remembering the purpose of each sensors places an important role. The touch sensors, if the robot bumps into an obstacle will register, then will turn and go forward in the opposite direction from the bump. The light sensors on the other hand can assist the robot by providing light readings; lower readings for brighter light and higher readings for darker light. Since the robot is utilizing both the types of sensors, it has more code to run.

The pucks according to the proposal are placed on tape. The robot will start in one of the corners. When the game starts, the robot should move forward in search of light. Once the tape readings are read, the robot should follow the white tape and eventually should hit a puck. Based on the impact from the puck, using touch sensors the robot should push the puck. The robot shouldn’t cross over as points will be deducted. Wandering function and periodic turn can be used from pg 212.

**Alternative Solutions**

int leftBound = 200; //Light readings

int rightBound = 150; //Light readings

int leftMotor= 0;

int rightMotor= 3;

int state = 0;

int neitherSide = 0;

int leftSide = 1;

int rightSide = 2;

int lineSide;

Int leftTouch=10;

int rightTouch=11;

int leftSense(){

if(analog(2)>leftBound){

return 0;

}

else{

return 1;

}

}

int rightSense(){

if(analog(3)>rightBound){

return 0;

}

else{

return 1;

}

}

void right()

{

bk(rightMotor);

fd(leftMotor);

}

void left()

{

bk(leftMotor);

fd(rightMotor);

}

void forward()

{

fd(leftMotor);

fd(rightMotor);

}

void seekLine1()

{

while(rightSense()==0&&leftSense()==0)

{

forward()

if (int digital(LEFT\_TOUCH)==1||digital (RIGHT\_TOUCH==1)){

left(1)}

}

}

void Line(){

forward();

if(leftSense() == 1 && rightSense() == 0){

lineSide = leftSide;

state = 2;

}else if(leftSense() == 0 && rightSense() == 1){

lineSide = rightSide;

state = 2;

}else if(leftSense() == 1 && rightSense() == 1){

lineSide = neitherSide;

}else if(leftSense() == 0 && rightSense() == 0){

state = 2;

}

}

void seekLine2()

{

foundLine = 0;

if(leftSense() == 1 || rightSense() == 1){

foundLine = 1;

state = 1;

}

if(foundLine == 0){

if(direction == 0){

left();

direction = 1;

}else if(direction == 1){

right();

direction = 0;

}

sleep(x/10.0);

stop();

if(leftSense() == 1 || rightSense() == 1){

foundLine = 1;

state = 1;

}else{

x++;

}

}

}

**Optimum Solution**

The optimum solution is the combination of functions such as seekLine1, Line, and seekLine2. The Line method can be modified during the lab section. When the robots finishes a line, it shouldn’t turn back and follow the line back. For this, the robot would need the touch sensors to determine whether or not the robot is pushing the puck. If the robot is pushing a puck, it should only give its attention to the touch sensors. This can be accomplished by adding a conditional statement to the Line algorithm. If the robot isn’t pushing any puck, it should execute the program like before which means the robot should continue to go forward for about a second until there is a line. This should work because of the gaps that are in the Mint Shuffle X game board. If the HandyBug is holding a puck and comes to an end of a line, it will continue to go forward for one second. If there is no line, then the seekLine method will be called to find a line. Overall, the physical body of the robot remains the same because this will result in the best outcome of all the pucks being found and moved. Not only this, the robot will be able to find the pucks in a timely manner due to the speed of the robot. The optimum solution remains the same where the robot will be hard coded for the first puck and then the robot will run through the other code to push the pucks to the other side, The only difference now is that instead of waiting a little to let the robot look for the pucks, the robot will start right away so it has time to find the pucks.

**Milestone Report (Analysis/Testing, Construction/Implementation, Final Evaluation)**

The second lab session focused on the construction of the robot and the trial of 1 of the code. During the testing, the steered right instead to go forward, the code was modified to change the power output of the left motor in order for the robot to move straight. Testing of the hardware was done. The program had modifications made which included timer to START the robot, have a jump function to push the puck on the other side once the touch sensor detects the puck in the extender. The implementation of the program was also done. The primary focus is the coding to make the complete the Mint Shuffle X game in the given time constraints. The given code allowed the robot to start using the timer and push one puck to the white corner. For the second half, a predefined program is used which gets activated once the touch sensors sense the wall. Overall, the body of the robot is kept same for best results. The only difference now is that instead of waiting a little later to let the robot look for the pucks, the robot will start right away so it has time to find the pucks and retrieve them.

**References**

1. Martin, Fred G. 2001. Robotic Explorations: A Hands-On Introduction to Engineering. New Jersey: Prentice Hall.

2.Oruklu, Erdal. 2017. *ECE 100 Lecture Notes.* Chicago: Illinois Institute of Technology, Electrical and Computer Engineering Department.

**Appendix**

int LEFT\_MOTOR= 3;

int RIGHT\_MOTOR= 0;

int LEFT\_SENSOR = 15;

int RIGHT\_SENSOR = 12;

int bumpedBool = 0;

float \_timer;

void main() {

int i =0;

while(1){

if(start\_button()) {

reset\_timer();

while(timer()<15.0){

/\*if(bumpedBool == 0){

motor(LEFT\_MOTOR, 100);

motor(RIGHT\_MOTOR, 100);

//printf("timer=%f \n", timer());

//sleep(0.2);

printf("Left Sensor = %d RightSensor = %d", digital(LEFT\_SENSOR), digital(RIGHT\_SENSOR));

if(digital(LEFT\_SENSOR) || digital(RIGHT\_SENSOR)){

bumpedBool = 1;

}

}

\*/

printf("timer=%f \n", timer());

motor(LEFT\_MOTOR, 50); //forward 1st

motor(RIGHT\_MOTOR, 50);

sleep(2.2);

motor(LEFT\_MOTOR, -35); //back

motor(RIGHT\_MOTOR, -50);

sleep(2.9);

motor(LEFT\_MOTOR, -40); //turn

motor(RIGHT\_MOTOR, 40);

sleep(.95);

motor(LEFT\_MOTOR, 85); //forward 2nd

motor(RIGHT\_MOTOR, 100);

sleep(1.46);

motor(LEFT\_MOTOR, -70); //backwards

motor(RIGHT\_MOTOR, -100);

//sleep(2.0);//tape to back

sleep(1.44);

motor(LEFT\_MOTOR, 40); //turn

motor(RIGHT\_MOTOR, -40);

sleep(.05);

motor(LEFT\_MOTOR, 100); //forwards 3rd

motor(RIGHT\_MOTOR, 100);

sleep(1.8);

motor(LEFT\_MOTOR, 5); //curved turn

motor(RIGHT\_MOTOR, 100);

sleep(.65);

/\*while(turnCounter<10){

motor(LEFT\_MOTOR, 50); //forward

motor(RIGHT\_MOTOR, 50);

sleep(1.38);

motor(LEFT\_MOTOR, -31

}

\*/ break;

}

motor(LEFT\_MOTOR,0);

motor(RIGHT\_MOTOR,0);

break;

}

}

}

void reset\_timer() {

\_timer = seconds();

}

float timer() {

return (seconds() - \_timer);

}