Post-Lab 7 Report **Alan Palayil**

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

Prof. Oruklu Lab Date 10/26/19

TA: Rafael Due Date 11/01/19

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.

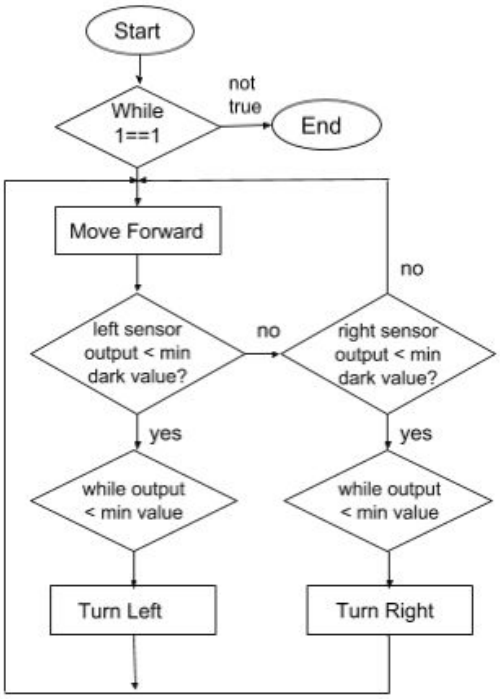
The pucks according to the proposal are placed on tape. The robot will start in one of the corners. When the game starts, the robot should move forward in search of light. Once the tape readings are read, the robot should follow the white tape and eventually should hit a puck. Based on the impact from 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

```
void groucho()

{

        while(1) {

                inside_corner();          /* position 1 to 2 */

                drive_to_top();           /* 2 to 3 */

                align_with_edge();        /* 3 to 4 */

                follow_edge_to_wall();          /* 4 to 5 */
```

```
        dump_ball_shuffle(); /* 5 to 6 to 7*/
        drive_to_bottom();      /* 7 to 8 */
        inside_corner();        /* 8 to 9 */
    }
}
```

[From Prof. Oruklu notes]



The first thing the robot needs to do is find the line. Once it detects the line, it can follow it and push a puck. The HandyBoard needs to be programmed simply to move around the Mint Shuffle X. Once a line is detected Method to keep one sensor on the tape and one off. That way it works as if the left sensor is the only one on the tape, the robot will continue to move forward. If they're both on the tape, the robot will turn right so the left sensor will be on the tape. If the right sensor is the only sensor on, then the robot will move right till the left sensor is on the tape. If neither the conditions are satisfied, another tape line seeking program should be called. This method will turn the HandyBug back and forth until a tape line is found.

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.

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

The Handy-Bug was constructed with the major design from the touch-sensor robot from Lab 1 with the light sensors both placed on the sides of the robot. Having the sensors close to the robot's body, it allowed the robot to follow the tape efficiently. The bumper was adapted to form an extended arm design to store the puck. The extenders were built outwards making the distance between them big. The gear ratio was changed to 1:8 [8-tooth gear placed on motor and 40-tooth gear placed on the axle] for a steady speed. Since the robot is steady, it would keep up and detect the tape.

The first 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 will be done during the next lab. For the next lab, the primary focus is the coding to make the complete the Mint Shuffle X game in the given time constraints.

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 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++;
                }
        }
}
```