**Post-Lab 5 Report       Alan Palayil**

ECE 100-02 Teammates: Wedge, Raymond

Prof. Oruklu Lab Date: 10/04/18

TA: Rafael Due Date: 10/11/18

**Problem Statement (Proposal)**

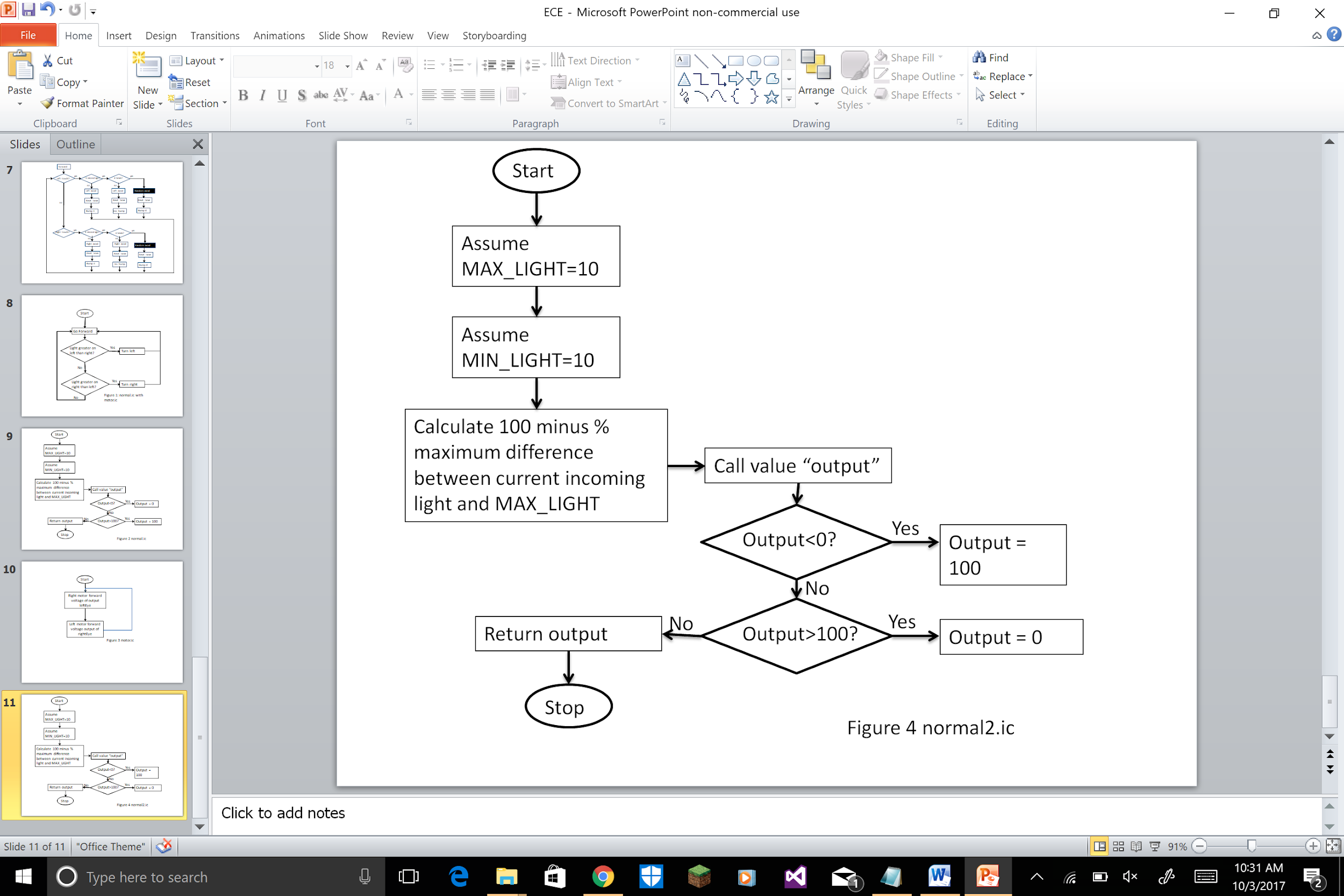
To program and design a robot which utilizes two light sensors to navigate through a maze which consists of turns up to 135 degrees while using Interactive C and HandyBoard.

**Investigation/Research**

Robotic Explorations explains the coding of the light sensors for the robot to track the tape using the analog sensors. The two sensors are to be placed on the tape for it to follow the mask tape path. The working of light sensors and code is made up by constantly reading the values of the light sensors. The reading when the sensor is on the tape can be used as a reference when running the code. When the robot reaches a curve, the intensity of light in one sensor will be greater than the other. The robot should thus make the corresponding turns by slowing down one motor to stay on the path. The book contains reference programs such as "normal.ic". Using a combination of motor.ic from lab-2 and lab-3; and normal.ic we create a program which can trace and follow the path. "normal.ic" (p.84), assigns value to each sensor which indicates the amount of light that is falling on the sensor. When the right sensor value is greater than left sensor, then using motor.ic, we can adjust the robot's movement like the figure 3 in page 85 of the book.

**Alternate Solutions**

According to the research, the normal.ic is supposed to return a value based on the intensity of light received by the sensors. The program can have 3 different methods to trace and follow the path. One configuration is having one sensor on the tape, by doing this we can use the average value of both the sensors to keep the robot to follow the path. The other method is to have both the sensors outside the tape. The last method is to have both the sensors inside the tape. Using these three solutions we can configure the robot which will trace the path the fastest. The motor.ic program returns a value for it to turn. Using the figures provided in the book, we can formulate and combine the programs which can follow one of our three methods. By using the normalize function which assigns a value to each sensor. Using this procedure, we had issues with the program was responding when the light sensor was covered the corresponding motor would turn, but the speed of the robot was too fast for the robot to detect any turns. For this we tried to make new program to deal with this issue.



**Optimum Solution**

Using the light seeking method will be the optimum solution in which the light sensors will be seeking light i.e. they will be on the tape. So, if the right sensor is off the tape then the left motor will slow/stop down to turn back the robot to the path. The conditional output value should be changed. This process is done for sudden turns to take place more frequently and is an easy approach than the darkness seeking program. We have placed a dedicated turn function when the intensity of the light gets changed to a value greater than the coded-value.

**Milestone Report (Construction/Implementation)**

The problem persists where the robot still does not execute very sharp turns. The robot could turn for small turns but with the speed of the motor the robot was not able to sense sharp turns. The strategy proved faulty as when the robot went off the tape, the robot would start spinning. The robot design wasn’t an issue. The real issue could be from two things; first, the sensors readings difference caused a lot of issues. Second is the connection to HandyBoard and the HandyBoard itself. In order to fix the problems, the code needed to be changed and the most critical task is coding and any other complications with the code will put the project behind schedule.

The robot can make slight turns, the weight of the robot is also less in order to decrease the pull or load. The table below shows the milestones achieved till now.

|  |  |
| --- | --- |
| Speed of Robot | X |
| Slight Turns | X |
| Sharp Turns | - |
| Completing the course | - |

There were no significant changes to the construction of the robot.

**Final Evaluation:**

Lab 6 will be used to check the codes with a little difference to get the robot to make the sharp 135° turns. The teams require to work on different solutions to come up with a solution which can tackle this issue.

The team is still behind schedule as the robot still hasn’t completed the maze, and the only thing that can be done to improve our pace is testing different codes.

**References**

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

2. Oruklu, Erdal. 2018. ECE 100 Lecture Notes. Chicago: Illinois Institute of Technology, Electrical and Computer Engineering Department.

**Appendix**

**The code we used:**

int LEFT\_MOTOR = 0;

int RIGHT\_MOTOR = 3;

int LEFT\_EYE = 3;

int RIGHT\_EYE = 2;

int MAX\_LIGHT;

int MIN\_LIGHT;

int ONTAPE;

void forward ()

{

motor (LEFT\_MOTOR, 40);

motor (RIGHT\_MOTOR, 40);

}

void turnRight ()

{

motor (LEFT\_MOTOR, 40);

motor (RIGHT\_MOTOR, -40);}

void turnLeft ()

{

motor (RIGHT\_MOTOR, 40);

motor (LEFT\_MOTOR, -40);}

float \_timer = seconds ();

void reset\_timer ()

{

\_timer = seconds ();}

float timer ()

{

return seconds () - \_timer;

return seconds () - \_timer;}

int normalize (int light)

{

int output = 100 - ((light - MAX\_LIGHT) \* 100) / (MIN\_LIGHT - MAX\_LIGHT);

if (output < 0)

output = 0;

if (output > 100)

output = 100;

return output;

}

void seek ()

{

motor (LEFT\_MOTOR, normalize (analog (RIGHT\_EYE)));

motor (RIGHT\_MOTOR, normalize (analog (LEFT\_EYE)));}

void turn ()

{

int avgLight = (analog (RIGHT\_EYE) + analog (LEFT\_EYE)) / 2;

while (avgLight > ONTAPE)

{

int right = analog (RIGHT\_EYE);

int left = analog (LEFT\_EYE);

if (right < left && timer () < 3.0)

{

turnRight ();

sleep (timer ());

}

Else

{

seek ();

}

if (right > left && timer () < 3.0)

{

turnLeft ();

sleep (timer ());

}

Else

{

seek ();

}

}

}

void main ()

{

while (1)

{

int avgLight = (analog (RIGHT\_EYE) + analog (LEFT\_EYE)) / 2;

if (avgLight <= ONTAPE)

{

forward ();

}

Else

{

reset\_timer ();

turn ();

}

}

}

**The we plan to use in the next Lab:**

int LEFT\_MOTOR = 3;

int RIGHT\_MOTOR = 0;

int LEFT\_EYE = 6;

int RIGHT\_EYE = 2;

int counterl = 0;

int counterr = 0;

int RIGHTDSTART = analog (RIGHT\_EYE);

int LEFTDSTART = analog (LEFT\_EYE);

void forward ()

{

fd (LEFT\_MOTOR);

fd (RIGHT\_MOTOR);

}

void backward ()

{

bk (LEFT\_MOTOR);

bk (RIGHT\_MOTOR);

}

void stopl ()

{

off (LEFT\_MOTOR);

}

void stopr ()

{

off (RIGHT\_MOTOR);

}

void stop ()

{

off (LEFT\_MOTOR);

off (RIGHT\_MOTOR);

}

void main ()

{

while (1)

{

if ((analog (RIGHT\_EYE) - RIGHTDSTART >= 0) && (analog (LEFT\_EYE) - LEFTDSTART < 0))

{

stopl ();

sleep (0.2);

motor (RIGHT\_MOTOR, 100);

sleep (0.5);

counterl++;

}

else if (counterl == 3)

{

motor (RIGHT\_MOTOR,100);

motor (LEFT\_MOTOR, -50);

counterl = 0;

}

if ((analog (RIGHT\_EYE) - RIGHTDSTART < 0) && (analog (LEFT\_EYE) - LEFTDSTART >= 0))

{

stopr ();

sleep (0.2);

motor (LEFT\_MOTOR, 100);

sleep (0.5);

counterr++;

}

else if (counterr == 3)

{

motor (LEFT\_MOTOR,100);

motor (RIGHT\_MOTOR, -50);

counterr = 0;

}

if ((analog (RIGHT\_EYE) - RIGHTDSTART < 0) && (analog (LEFT\_EYE) - LEFTDSTART < 0))

{

forward ();

}

if ((analog (RIGHT\_EYE) - RIGHTDSTART >= 0) && (analog (LEFT\_EYE) - LEFTDSTART >= 0))

{

stop ();

sleep (0.2);

backward ();

sleep (0.01);

}

}

}