**Post-Lab 4 Report Alan Palayil**

ECE 100-001 Teammates: Wedge, Raymond

Prof. Oruklu Lab Date 9/27/17

TA: Rafael Due Date 10/4/17

**Problem Statement**

The aim of the lab was to create a robot which could follow the mask tape path using light sensors.

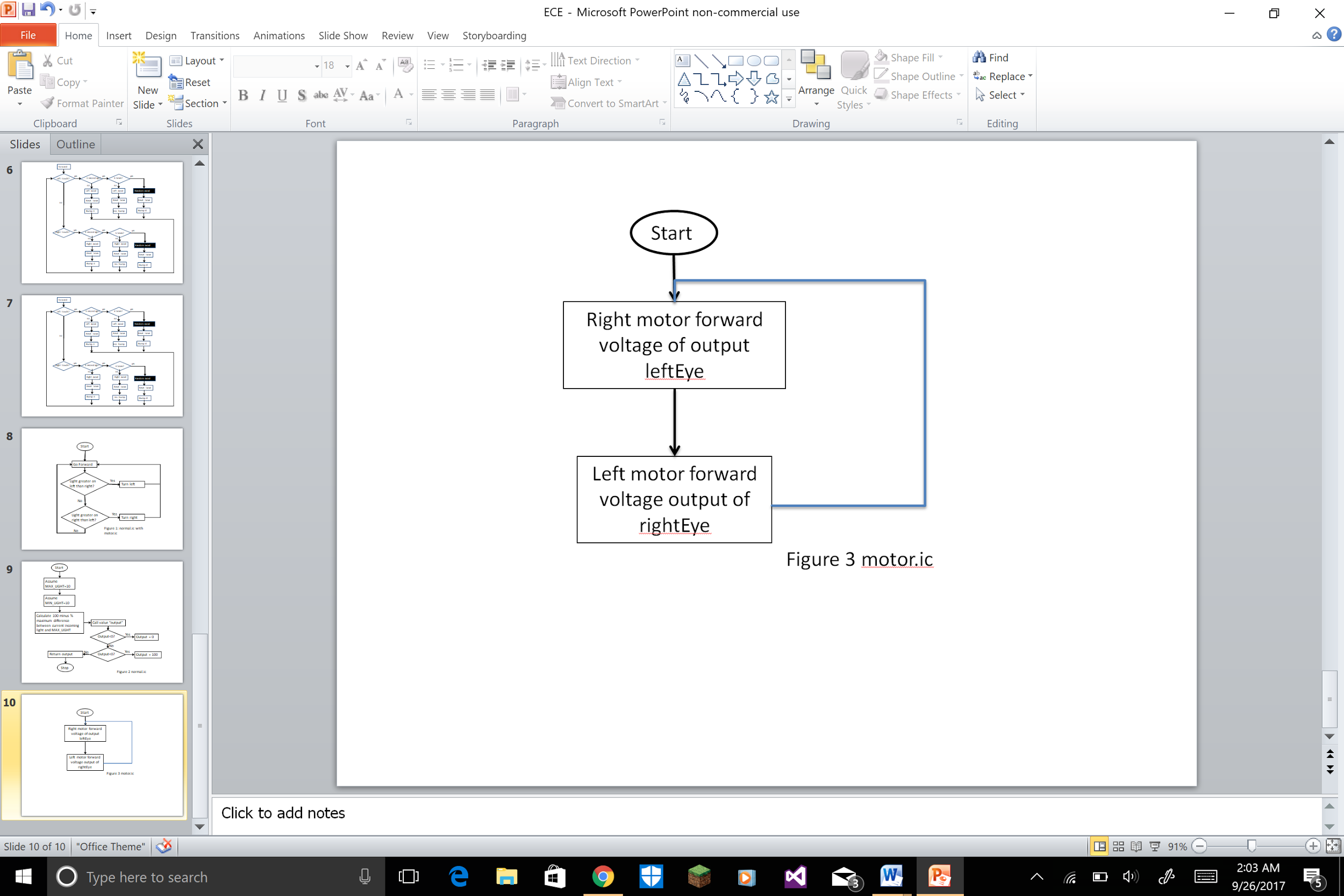
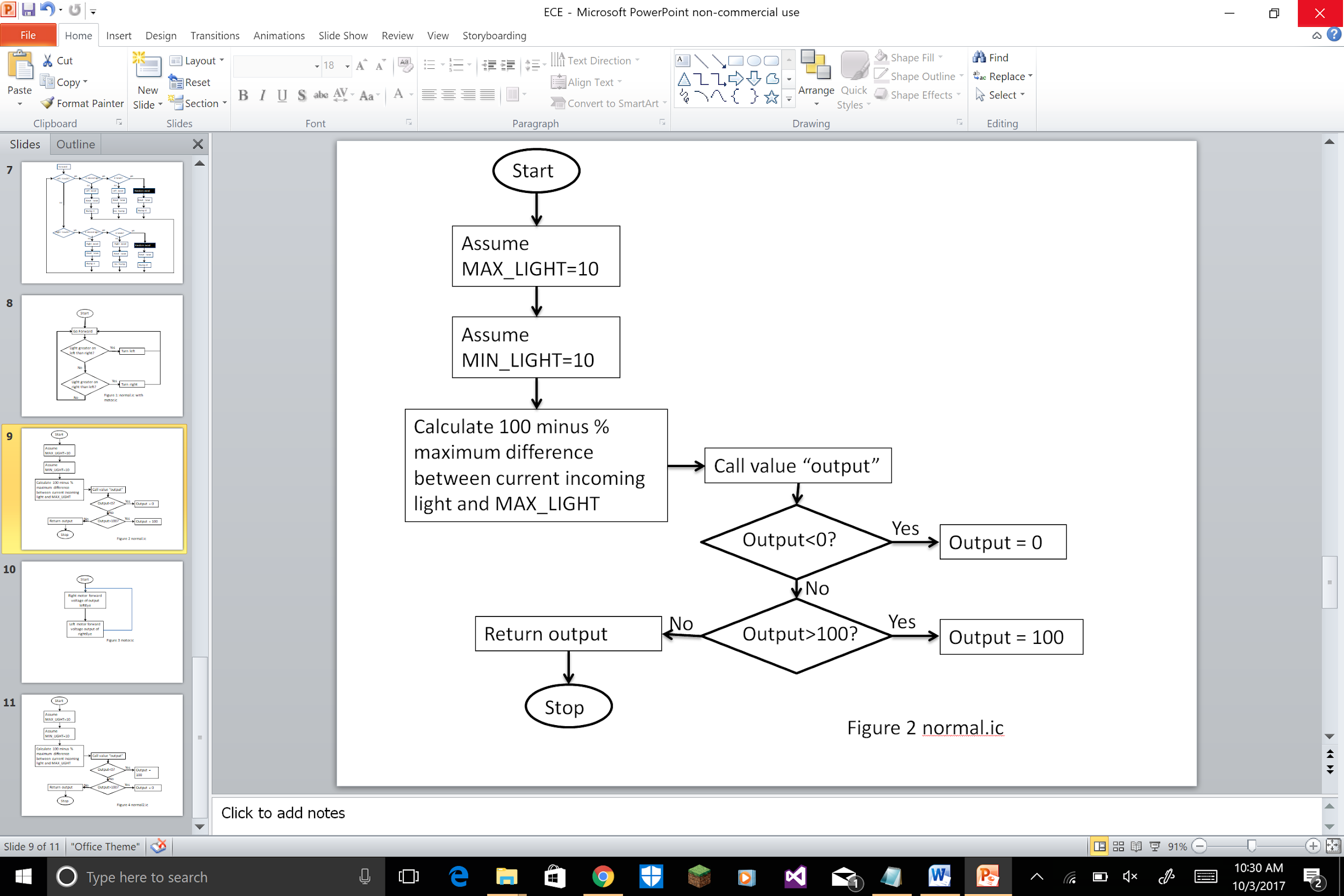
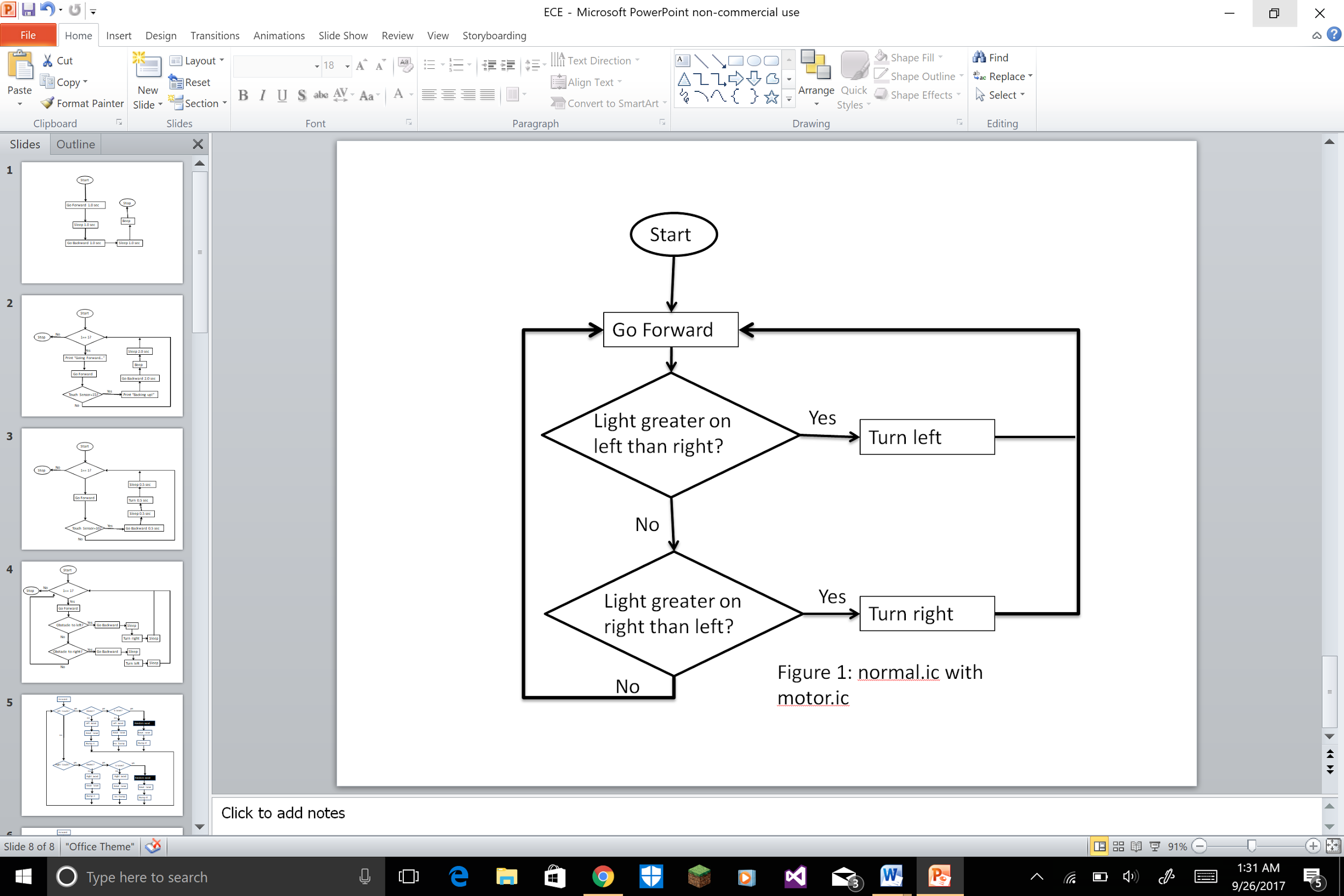
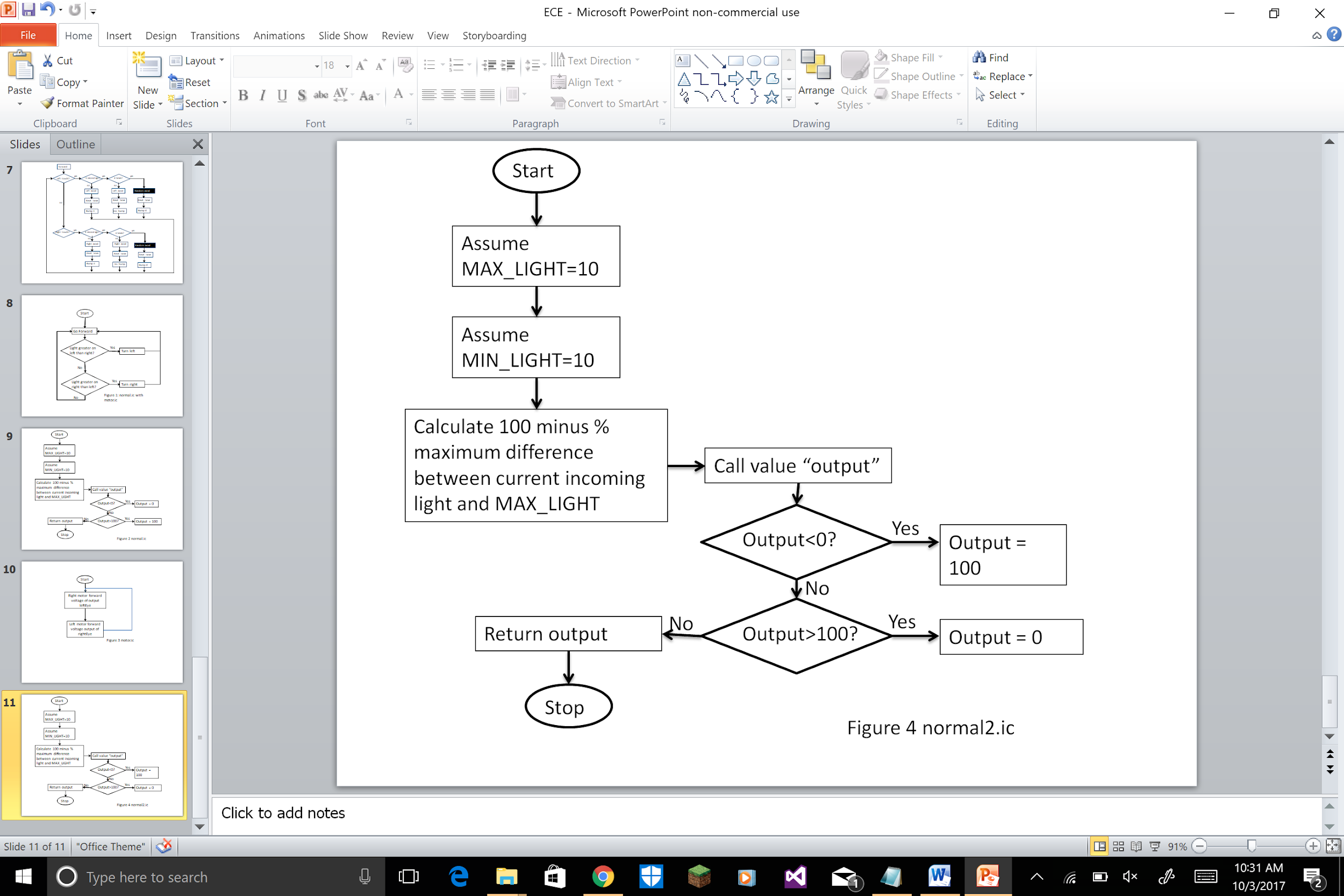
**Investigation/Research**

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 in order 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.

**Alternative Solutions**

According to the research, the normal.ic is supposed to return a value based on the intensity of light recieved 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 in order 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. Motor program can take that value from normal.ic and set the torque levels of the motors. The main aim is, if the tape shifts towards the left, the right sensor will not be receiving any values which indicates that the sensor is not receiving any light. Correspondingly, the torque of the right motor will be sleep for the left motor to continue moving to make the turn. Using the second method by following the borders, is done just by changing the light variable to dark, and switching the output values. [The flowcharts are acquired from two sources: 1. Robot Explorations

2. Google Search Engine]

**Optimum Solution**

Using the light seeking method will be the optimum solution in which the light sensors will be seeking light ie.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.

**Construction/Implementation**

The team split the task between the teammates. We used the existing structure of the Handy Bug with major changes made to the front bumper, placing the light sensors vertically down. Using an idea from Appendix E design, we matched the left sensor to the right motor and vis versa. The major time of the lab was spend in trying the trial programs in order to check the sensors provided, measuring the value of the light reflected by the masking tape. The coding was compiled but, the normalized function was having a little issue and finding the issue took a little time to figure it out.

**Analysis & Testing**

The Handy Bug was only set to test the sensors and the to find the readings obtained by the sensors. The Handy Bug was able to move straight but when it was coming off the tape, it was not turning and continued moving straight. The team decided to fix the program and the problem during the next lab section.

**Final Evaluation**

Lab 5 will be used to check the codes with a little difference to get the robot to make the sharp 135° turns. The teams requires to work on different solutions to come up with a solution which can tackle this issue. The robot needs make the turns and still not lose time. Once that problem is solved, we can make tweaks to the program to perfect it. The only predictable drawbacks can be getting a faulty Handy Board.

**Questions**

1. **How do the light sensor reading change as the amount of light increases?**

With the increase in intensity of light, the value read by the sensor decreases.

1. **Do the two light sensors typically give the same reading, or do they vary by the angle toward the nearest light source?**

The two light sensors readings vary due to various reasons such as the angle from which light enters,etc.

1. **What are the maximum and minimum readings you can obtain? What are typical readings from just ambient light sources?**

The maximum and minimum values that can be obtained are 250 and 10 respectively. The typical reading from ambient light source should be in the range of 100 to 150.

1. **Do both sensors seem to provide the same reading for the same amount of light or are the readings shifted with respect to the other?**

Each sensor is different. The value varies in the units place.

**P. 85 Questions**

1. **Type in the normalize function and save it in the file normal.ic, replacing MAX\_LIGHT and MIN\_LIGHT constants with values determined from your prior experimentation with Handy Bug’s light sensors. What values will you use for the maximum and minimum light readings?**

Since values for maximum light can be obtained by placing the sensor in front of a light source and minimum value can be obtained by placing the sensor in the dark, the values vary but the average values for MIN\_LIGHT=190 and MAX\_LIGHT=20 should be kept the same.

1. **From the Interactive C command line, load the normal.ic function and experiment with the normalize function. For example, if the light sensor is plugged into analog input 0, try: while(1){printf(“normal=%d\n”, normalize(analog(0))));sleep(.1);}**

**Does normalize operate as advertised? What is the range of values that you are able to get it to produce?**

The maximum value was 250, and the minimum was 10. Normalize operated as printed.

1. **In the normal function, there are checks to test whether the computed output value is less than zero or greater than 100. How is it possible that either of these cases may occur?**

This situation can only occur if the light value is outside the predicted range.

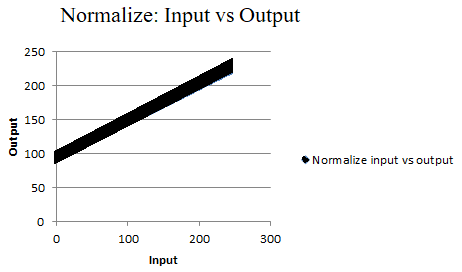
**How many normalize functions do you need for two sensors? Justify your answers.**

One is required as the normalize function assigns the variable for both sensors in the same program.

**Post Lab 4 Questions:**

**Given your experience, how has your solution strategy changed**

The strategy is to make the program to make sharper turns and to use the different point of views to tackle the programs normal.ic and motor.ic to work simultaneously and to let the robot make sharp turns.



**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**

Light sensing

int LEFT\_MOTOR = 0;

int RIGHT\_MOTOR = 3;

int leftEye = 3;

int rightEye = 2;

int normalize(int light)

{

int MAX\_LIGHT=10;

int MIN\_LIGHT=200;

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

if (output<50)

output=100;

if (output>100)

output=0;

return output;

}

void main()

{

while(1)

{

motor(LEFT\_MOTOR, normalize(rightEye);

motor(RIGHT\_MOTOR, normalize(analog((leftEye));

}

}

**The code which we plan to use is:**

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();

}

}

}