**Lab 4: Localization**

ECSE 211: Design Principle &Methods

Group 27

Student Name: Roger Zhang

Student ID: 2608051430

Student Name: Boyang Ma

Student ID: 260674928

Faculty of Engineering   
Department of Electrical, Computer and Software Engineering  
McGill University  
October 11th 2018

**Design Evaluation:**

**Workflow Overview**

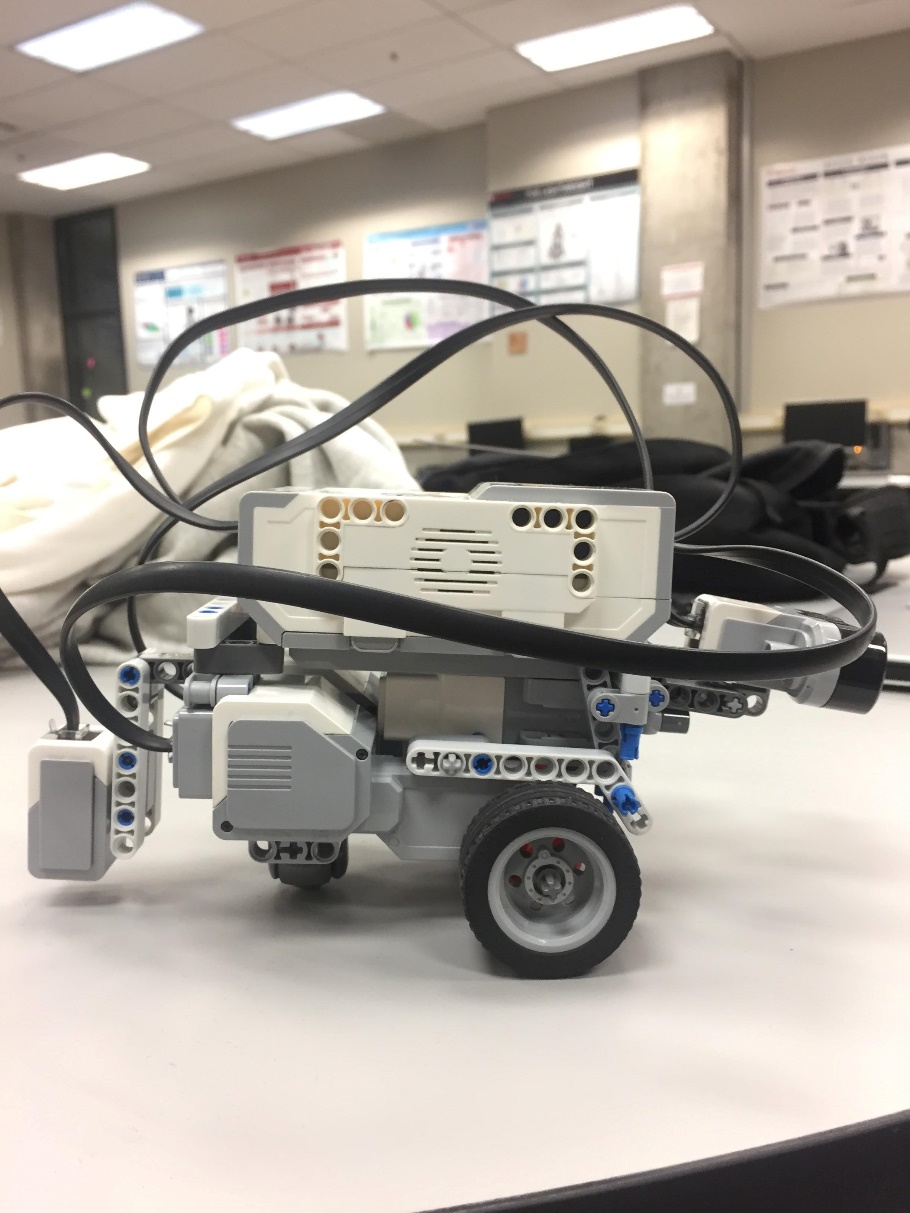
The workflow for this localization lab started with the redesign of the robot. The light localizer had to be added back and was put in front of the robot so that it could detect the lines before the robot passed them. The ultrasonic sensor and the rest of the components of the robot were kept at the same emplacement. The first software design task was to assemble all the necessary classes. Two major class had to be written from scratch: UltrasonicLocalizer and LightLocalizer. The rest of them were mostly recycled from previous labs. UltrasonicLocalizer had to be completed first since LightLocalizer relied on its output to proceed. Therefore, the software design for UltrasonicLocalizer was first completed. A fair bit of testing and data tuning followed to make sure it performed accordingly to lab requirements. LightLocalizer was then designed and more testing and data tuning were finally done.

**Figure 1: Detailed Workflow**

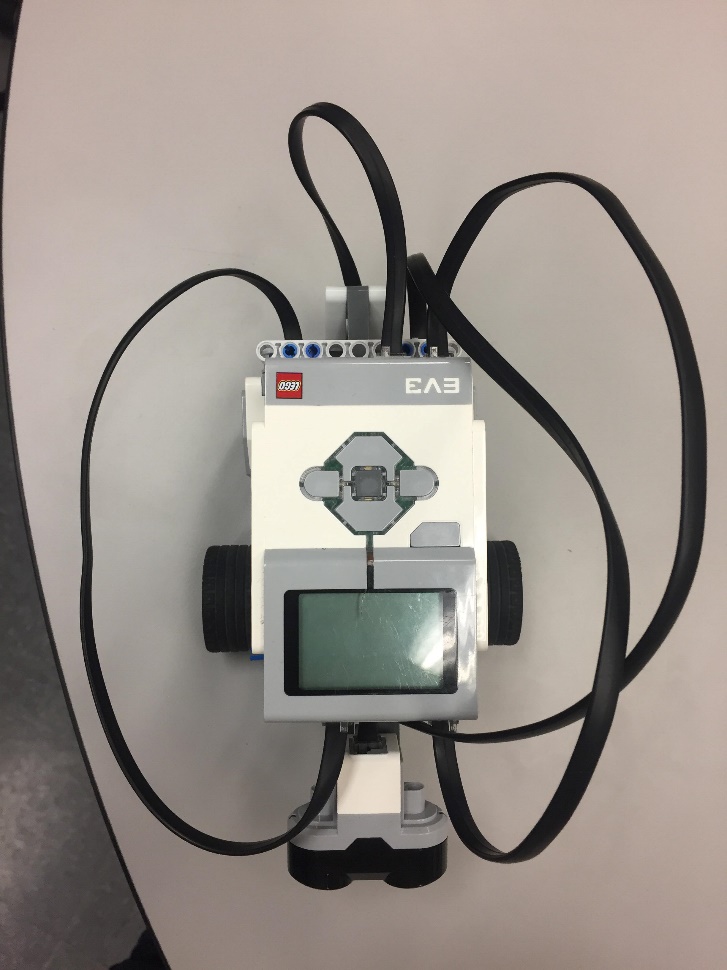
**Hardware Design**

The general structure was kept the same as the previous labs. The sole difference was the adding of the light sensor at the back of the robot. The brick was placed immediately above the two motors, which are parallel to each other to allow the robot to move linearly (Figure 2). The ultrasonic sensor was kept directly in front of the brick at a height that was judged best to detected obstacles in front of the robot.

Figure 2 also shows the light sensor at the back of the robot. It was located far from the center of rotation of the robot so that it could efficiently detect the positions of the grid lines for the LightLocalizer part of the lab.



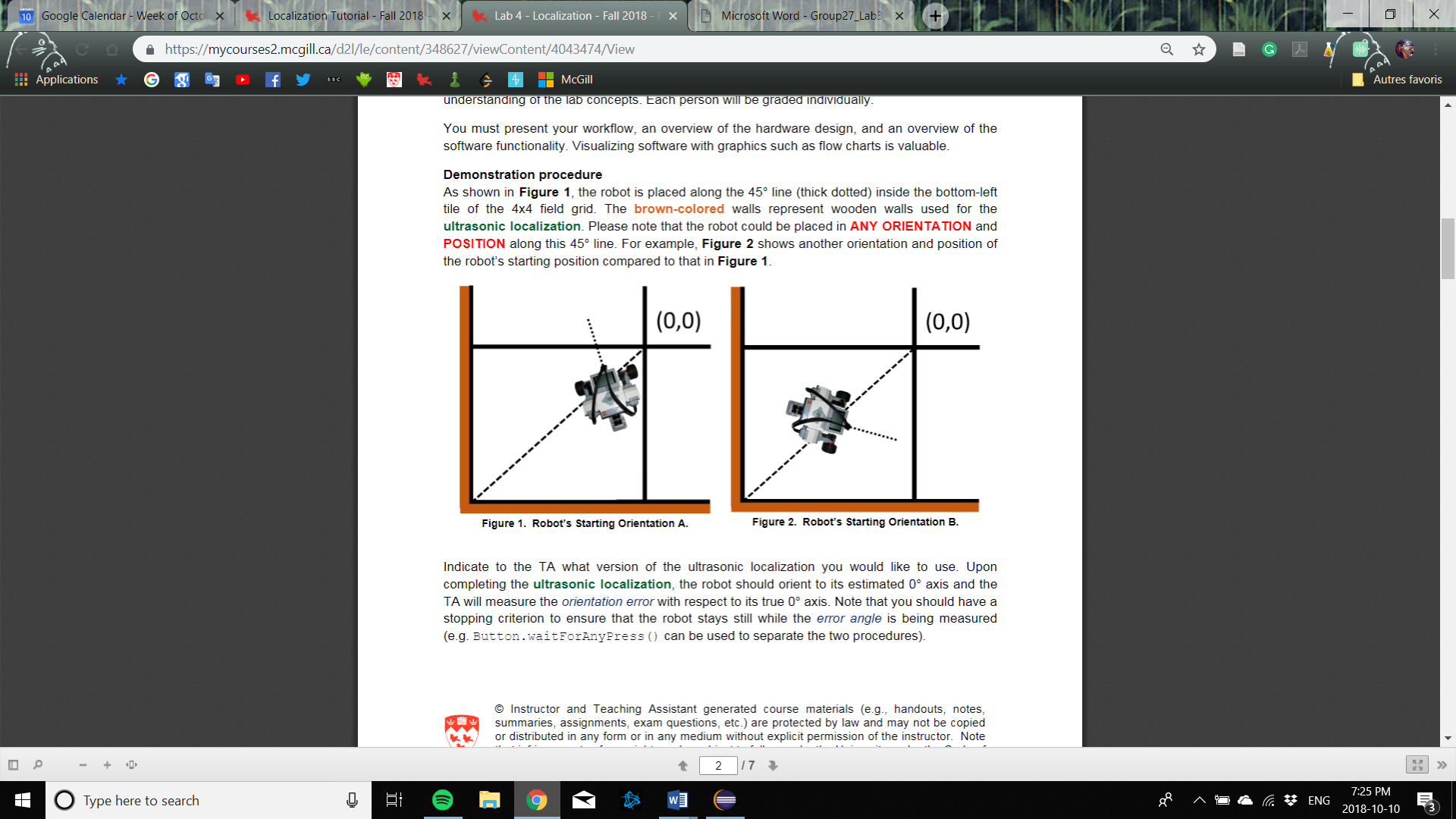
## **Figure 2: Side view of the robot**



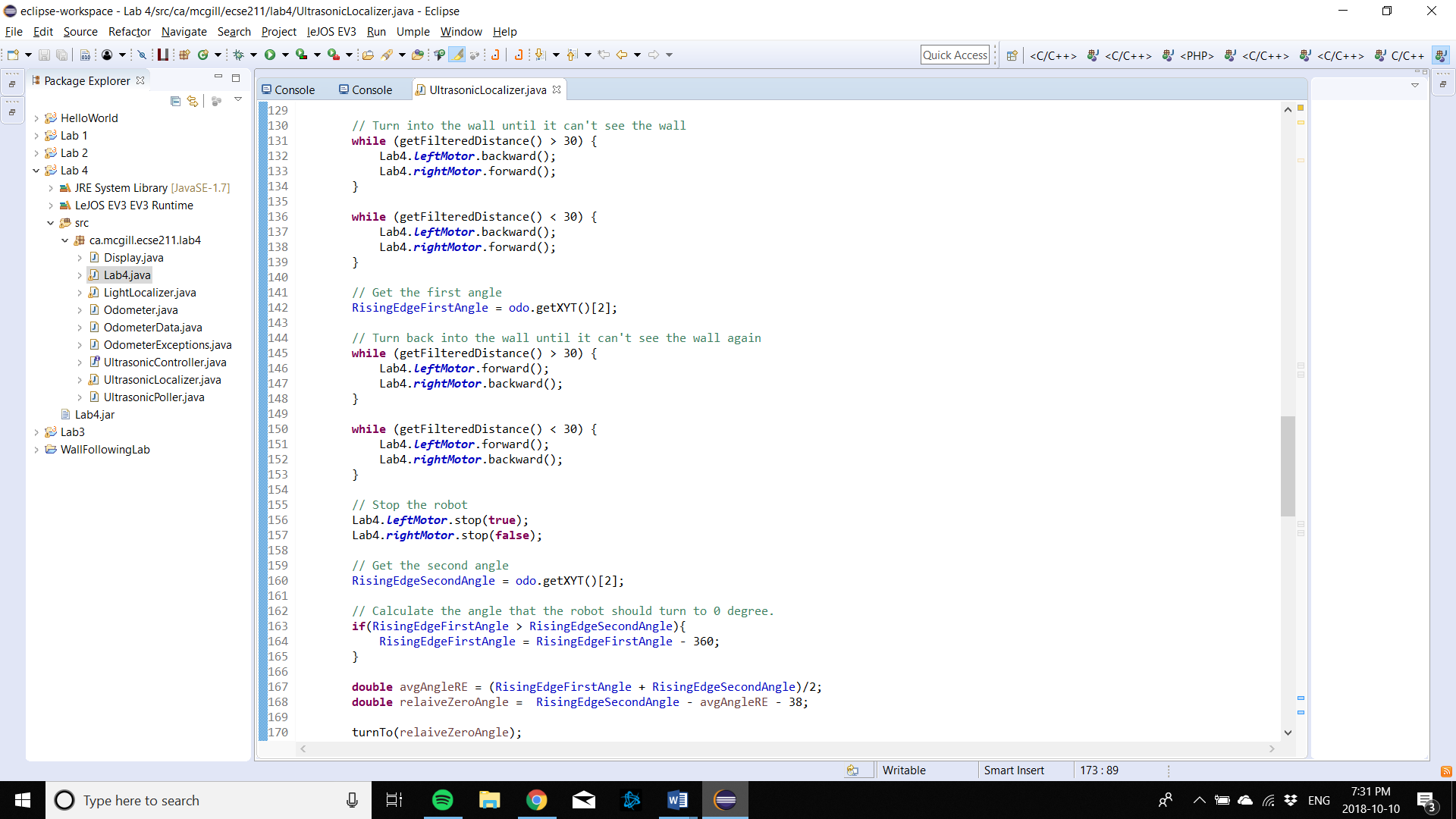
**Figure 3: Sky view of the robot**

## **Software Design**

The software design of the robot reuses 7 classes such as OdometerData and UltrasonicPoller which were used in previous labs. Two new classes were created: UltrasonicLocalizer and LightLocalizer. First, the purpose of the UltrasonicLocalizer class is to find the heading of the robot which is assumed to be unknown at the start of the lab. Knowing that the robot starts on the 45 line inside the bottom-left tile of a 4x4 field grid (Figure 4), the idea is to rotate the robot and use the data collected by the ultrasonic sensor to figure out the angle of heading. The class implements 2 methods to find the angle of heading: falling edge and rising edge. Rising edge, for example, starts by turning the robot counter-clockwise until the sensor records a distance lower than 30 cm. Then, it continues turning until it records a rising edge (distance bigger than 30 cm). It saves the angle at that moment as the first angle. It then starts turning clockwise until a second rising edge is recorded. The angle at that moment becomes the second angle. The average of the first and second angle can then be computed, which subtracted to the second angle and a constant found by tuning, allows to find how much the robot must turn to be at angle 0 (Figure 5). Falling edge implements the same principle but looks for falling edges instead of rising edges. After finding the 0 angle, the robot heads towards the point (0, 0) in preparation for the LightLocalizer tasks.

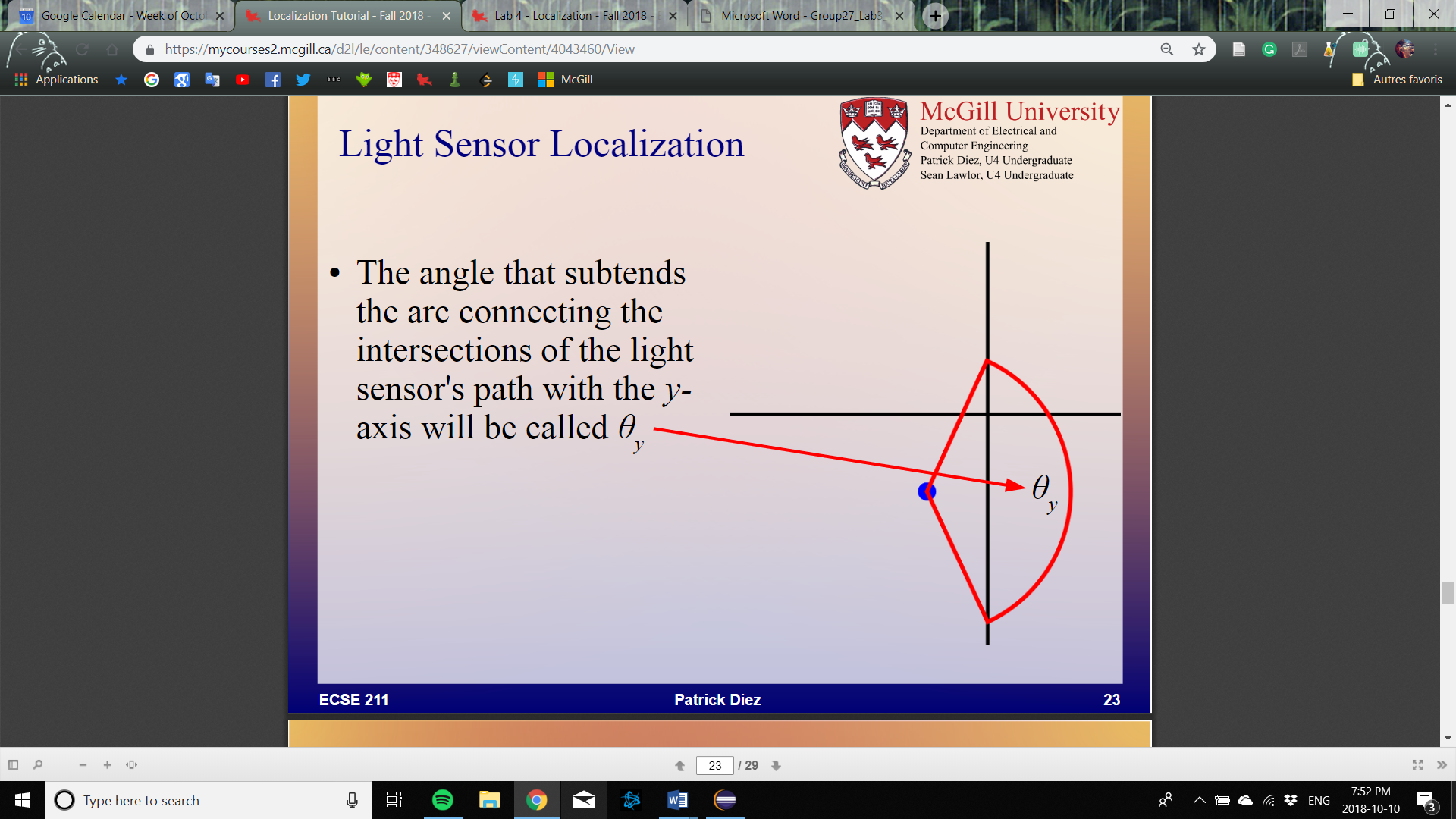


**Figure 4: Starting position of the robot**

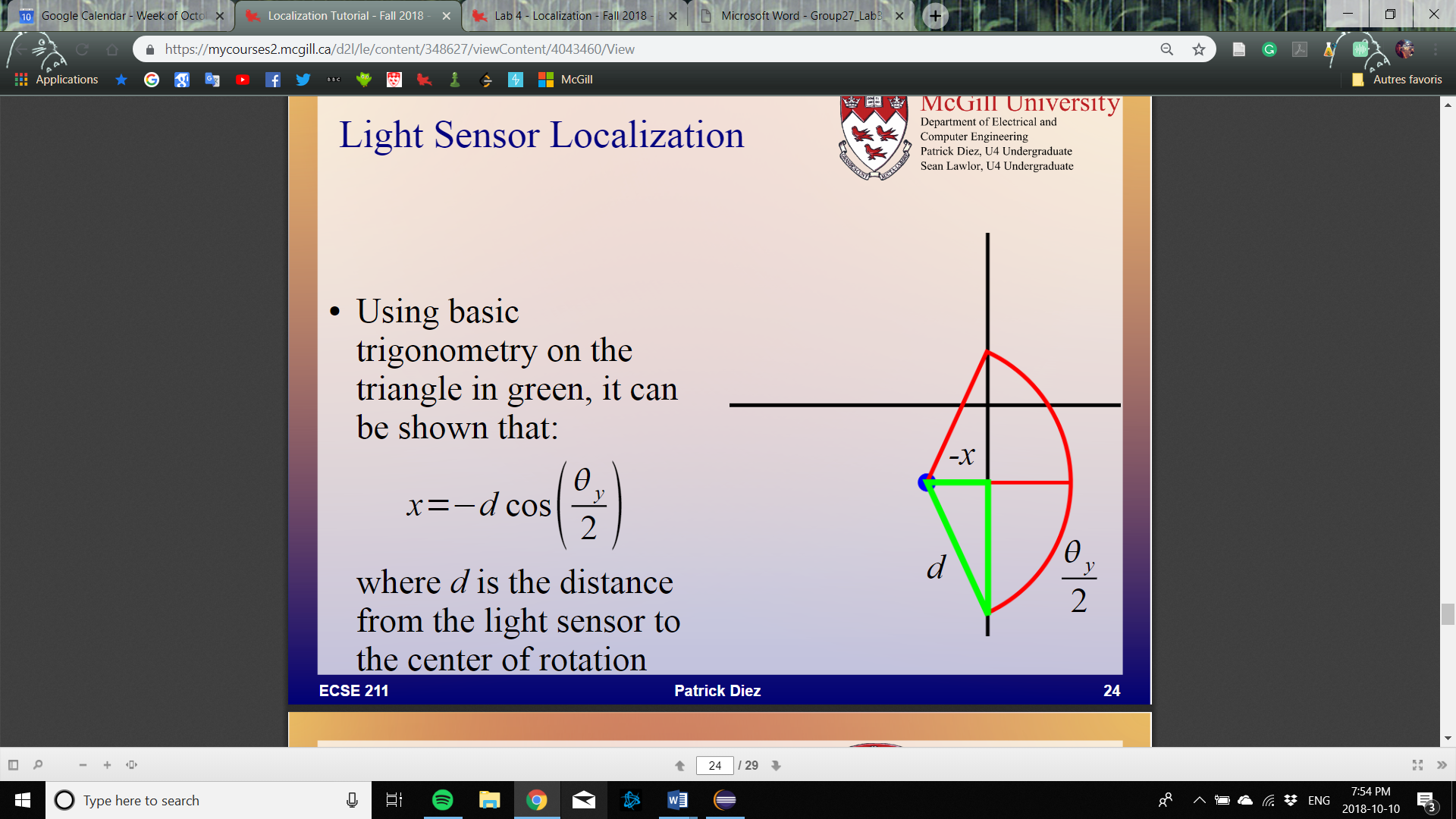


**Figure 5: Software design for rising edge**

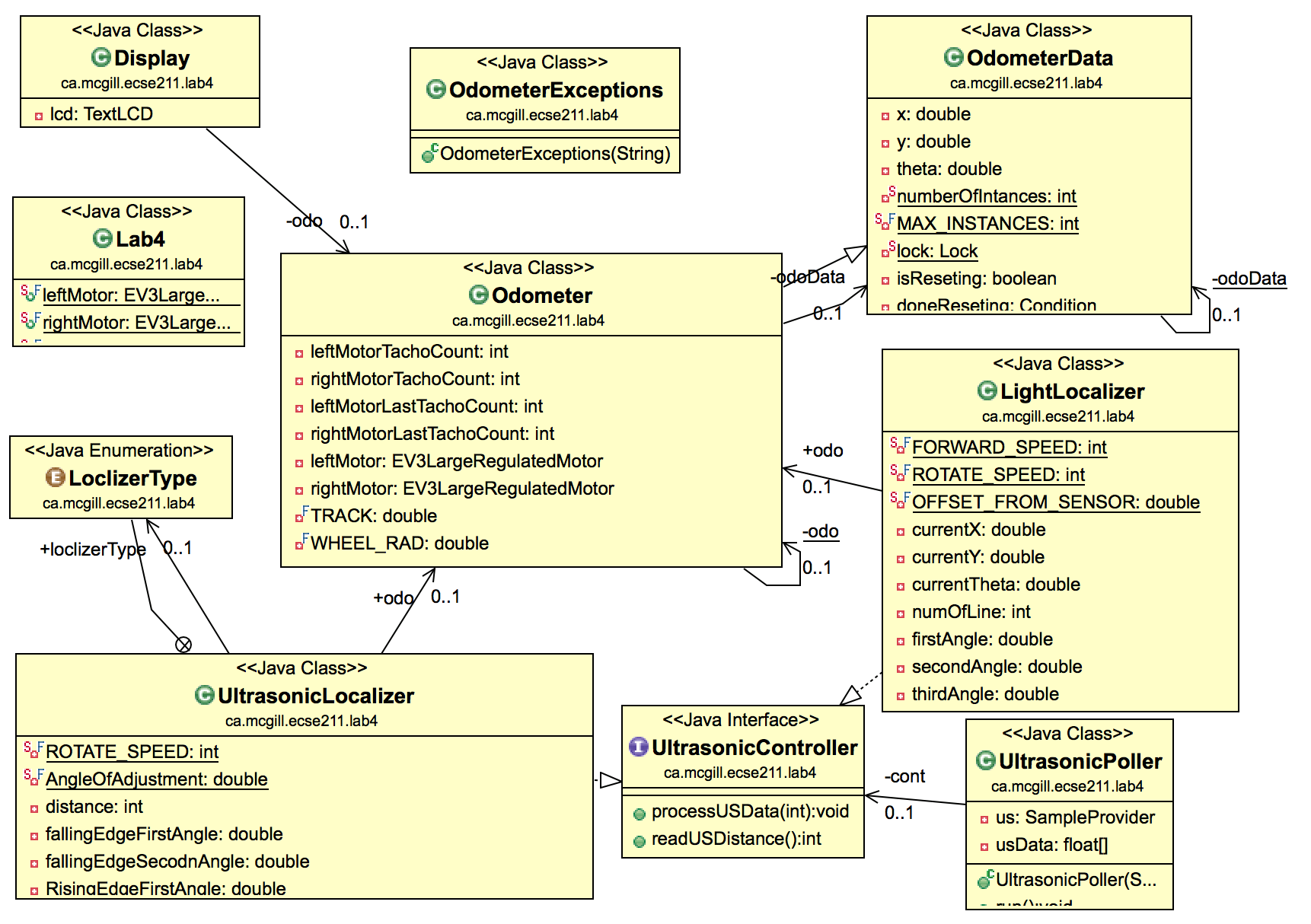
Secondly, once the robot has found its angle of heading, the job of the LightLocalizer class is to guide it to the point (0, 0). The robot already starts close to the origin because of the UltrasonicLocalizer class. Therefore, it starts executing a full 360 turn and saves the angle of heading every time the light sensor detects a black line. 4 angles should be saved, 2 for the x-axis and 2 for the y-axis. The angle that subtends the arc connecting the 2 points where the readings on the y-axis have been recorded (Figure 6) can then be divided by 2 and used with some trigonometry to find -x (Figure 7). The same is done for -y. Once -y and -x are found, the robot knows its precise location and can accurately travel to the point (0, 0).



**Figure 6: Angle that subtends the arc connecting the 2 points where the readings on the y axis have been recorded**



**Figure 7: The relation between and -x**

****

**Figure 8: UML Class diagram (Using ObjectAID UML tool in eclipse)**

**Tuning Process**

The rising edge and falling edge software designs processed multiple ultrasonic sensor readings per second and relied on each of them being accurate. They were especially vulnerable to false negatives and false positives since a single wrong reading could jeopardize all the computations necessary to find the angle of heading. Therefore, a filter that filtered out false negative was implemented. It only filtered false negatives because the sensor outputs a maximum distance when it doesn’t read anything, which causes them to happen more often. Any distance over 50 cm would need to occur 3 times before being let through to the rising or falling edge methods.

**Test Data:**

The rising edge and falling edge localization systems where tested according to the following steps:

1. Place the robot in a tile corner along the 45 line, where two walls are present.
2. Choose a random orientation for the robot.
3. Run the ultrasonic localization routine using rising edge
4. Note the ultrasonic angle of the robot.
5. Compute the ultrasonic angle error using the ultrasonic angle and the expected value.
6. Continue the localization using the light sensor to center at (0, 0) and 0.
7. Note the final position and final angle of the robot.
8. Compute the Euclidean error distance error and final angle error.

**Test localization using rising edge**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ultrasonic angle error | Euclidean distance error (𝛜) | Final angle  error |
| Trials |  | cm |  |
| 1 | 2 | 0.5 | 5 |
| 2 | 2 | 0.2 | 13 |
| 3 | 9 | 1.220656 | 1 |
| 4 | 10 | 1.360147 | 9 |
| 5 | 7 | 0.72111 | 4 |
| 6 | 5 | 0.316228 | 3 |
| 7 | 9 | 1.526434 | 4 |
| 8 | 8 | 0.72111 | 3 |
| 9 | 8 | 0.761577 | 7 |
| 10 | 1 | 0.943398 | 6 |

**Table 1: Ultrasonic angle error, Euclidean error distance between the origin and the robot final position, and final angle error of the localization test with rising edge.**

**Test localization using falling edge**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ultrasonic angle error | Euclidean distance error (𝛜) | Final angle  error |
| Trials |  | cm |  |
| 1 | 9 | 1.746425 | 3 |
| 2 | 7 | 1.676305 | 3 |
| 3 | 2 | 1.140175 | 2 |
| 4 | 5 | 1.3 | 1 |
| 5 | 10 | 1.486607 | 6 |
| 6 | 2 | 1.044031 | 2 |
| 7 | 4 | 1.1 | 1 |
| 8 | 9 | 1.264911 | 1 |
| 9 | 2 | 0.72111 | 2 |
| 10 | 2 | 0.984886 | 2 |

**Table 2: Ultrasonic angle error, Euclidean error distance between the origin and the robot final position, and final angle error of the localization test with falling edge.**

**Test Analysis:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ultrasonic angle error | Euclidean error distance | Final angle error |
|  |  | cm |  |
| Mean | 6.1 | 0.827066 | 5.5 |
| Standard Deviation | 3.3483 | 0.438383 | 3.472111 |

**Table 3: Mean and standard deviation of the ultrasonic angle error, Euclidean**

**error distance and final angle error of the rising edge localization test.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ultrasonic angle error | Euclidean error distance | Final angle error |
|  |  | cm |  |
| Mean | 5.2 | 1.246445 | 2.3 |
| Standard Deviation | 3.29309 | 0.318634 | 1.494434 |

**Table 4: Mean and standard deviation of the ultrasonic angle error, Euclidean error distance and final angle error of the falling edge localization test.**

**Formulas and sample calculations:**

Euclidean error distance:

and : The real-world location of the center of the track of the robot.

and : Where the robot is supposed to be according to the requirements, in this case the origin (0, 0).

Using data from trial 1 of the rising edge localization test:

cm

Mean of the Euclidean error distances:

Using data from the rising edge localization test:

0.827066

Standard deviation of the Euclidean standard distances:

Using data from the rising edge localization test:

0.438383

The means and standard deviations of the ultrasonic angle error, and final angle error of both the rising edge and falling edge localization tests are computed in the same way.

**Observations and conclusions:**

According to the data collected during testing, localization using falling edge performed the best when it came to the angle error. Not only are the means of the ultrasonic angle error and the final angle error for the falling edge lower then rising edge, but the standard deviations are also lower. This indicates that falling edge was also more consistent at being accurate. However, when it comes to the Euclidean error distance, rising edge was on average 0.4 cm more accurate at placing the robot on the origin.

Since no angle correction was used after the initial computing of the angle, the ultrasonic angle had a big impact on the final angle. The accuracy of the angle of the end of each trial almost entirely relied on the accuracy of the angle found initially.

The most impactful factor for both rising edge and falling was probably the frequency and accuracy of the polling from the ultrasonic sensor. Since the methods relied on detecting a falling or rising edge at a precise distance, recording an angle one degree too late could mean a few degrees difference in the end. Furthermore, external unpredictable factors such as the slipping of the wheels and the battery level also affect the performance of the rising and falling edge methods.

Changing the light conditions could impact the light localization negatively. The light sensor differentiates colors by emitting a red light and measuring the amount of black reflected. A variation in the ambient light could tamper with the amount of reflected black and cause the sensor to miss the black lines.

**Further improvements:**

The accuracy of the ultrasonic angle relies heavily on detecting a rising or a falling edge at the right moment. Therefore, more sampling from the sensor would minimize its error. However, the sampling rate is a constant that cannot be modified. Slowing the rotation speed of the robot while sampling could then be a solution for getting more samplings for a same area. Another alternative would be to look for a precise distance before recording the angles. For example, instead of recording any rising edge which occurs when a distance over 30 is recorded, the software could make the robot change rotation direction until it records a distance between 30 and 34. That way, the angles used during the computations to find the ultrasonic angle would be more accurate.

Assuming the robot always starts in a corner, another form of localization would be to make it spin 360 and only record small distances, only the ones below 45 for example. That would filter out any false negatives and walls that are too far to be detected. Then, amongst the remaining values, it could be determined that the corner is represented by the local maximum. This way, the robot could find its angle of heading without having to look for rising or falling edges.

Looking at the average Euclidean error distance of around 1 cm for the light localization system, it can be determined that it is very accurate at finding the desired coordinates. Therefore, it could be used in combination with navigation. Every time the robot navigates to a new point, the light sensor localization system could make sure the robot is exactly on the point.