ECSE 211: Design Principles and Methods

Lab 3: Navigation

Arianit Vavla : 260868601

Cheng Chen: 260775674

Group 20

October 2019

Section 1: Design Evaluation

Hardware Design

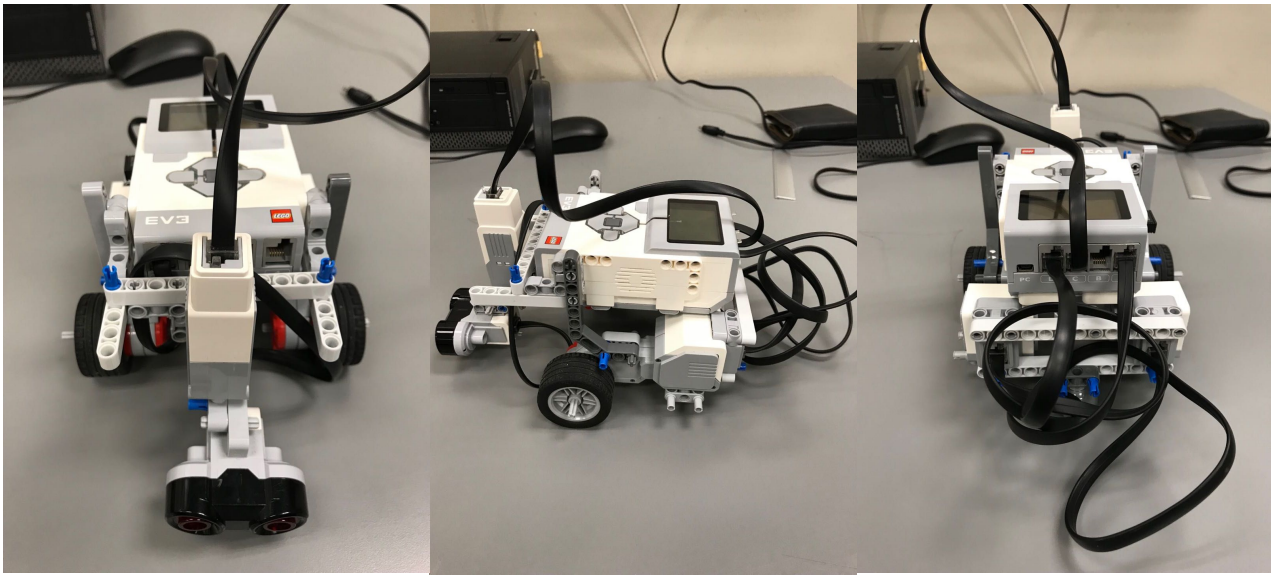

Figure 1. Robot

We designed the robot using similar structures from the wall follower lab. We fixed the motor to the sides of the EV3 controller and connected the motors to two tires in the front. On the back of the robot, we used the metal ball at the back of the robot in order to gain more stability. The ultrasonic sensor was also connected to another motor so that the sensor can scan at every angle since we do not know which direction the obstacle will be at.

For the first few tries, we found that the robot did not follow the path as we expected. The robot is not turning at the accurate angle. Instead, it deviated from the path. After a closer look at the robot, we found that the tire was too loose which results in slips while the robot was accelerating, decelerating or turning. After switching to a new tire, everything was working as expected. When we were testing the obstacle avoidance functionality, for the first few tries, we did not connect the ultrasonic sensor to a motor, which means the sensor was always pointing at one direction. While we were testing, we decided to use the ultrasonic

sensor to swipe the area by connecting to another motor since we do not know where the obstacle will be.

Software

First of all, we implemented the simple navigation algorithm. When the user starts the program, there will be a prompt saying that push left for simple navigation and push right for navigation with avoidance. The simple navigation will let the robot travel to a series of points as specified. We used the odometer we implemented for last lab for the robot to get its current coordinates. We also used the PController from previous lab. For the simple navigation part, the odometer will pass the coordinates to the navigation class. The turnTo method in the navigation class will process the coordinates and use it to calculate the distance to the next point and the angle of turning. The hardest part of this class is to get the robot to turn at the minimal angle. We implemented a method called getMinimalAngle to calculate the minimal angle. We found that there is a function called atan2 which gives us the angle if we know the x and y distance of it. Then we used geometry to calculate the minimal angle which will then be passed to turnTo method to make a proper turn.

After that, we implemented the obstacle avoidance algorithm. We created a boolean variable isNavigating. When the robot detects an obstacle, we make the isNavigating false and thus we enter the avoidance mode. We used the PController from previous lab. Once the robot does not detect the obstacle, isNavigating becomes false. The robot will go back to the original path using the odometer data. The above operations were accomplished by placing poller, navigation and avoidance in different threads. The following flow chart will show the software design of the lab.

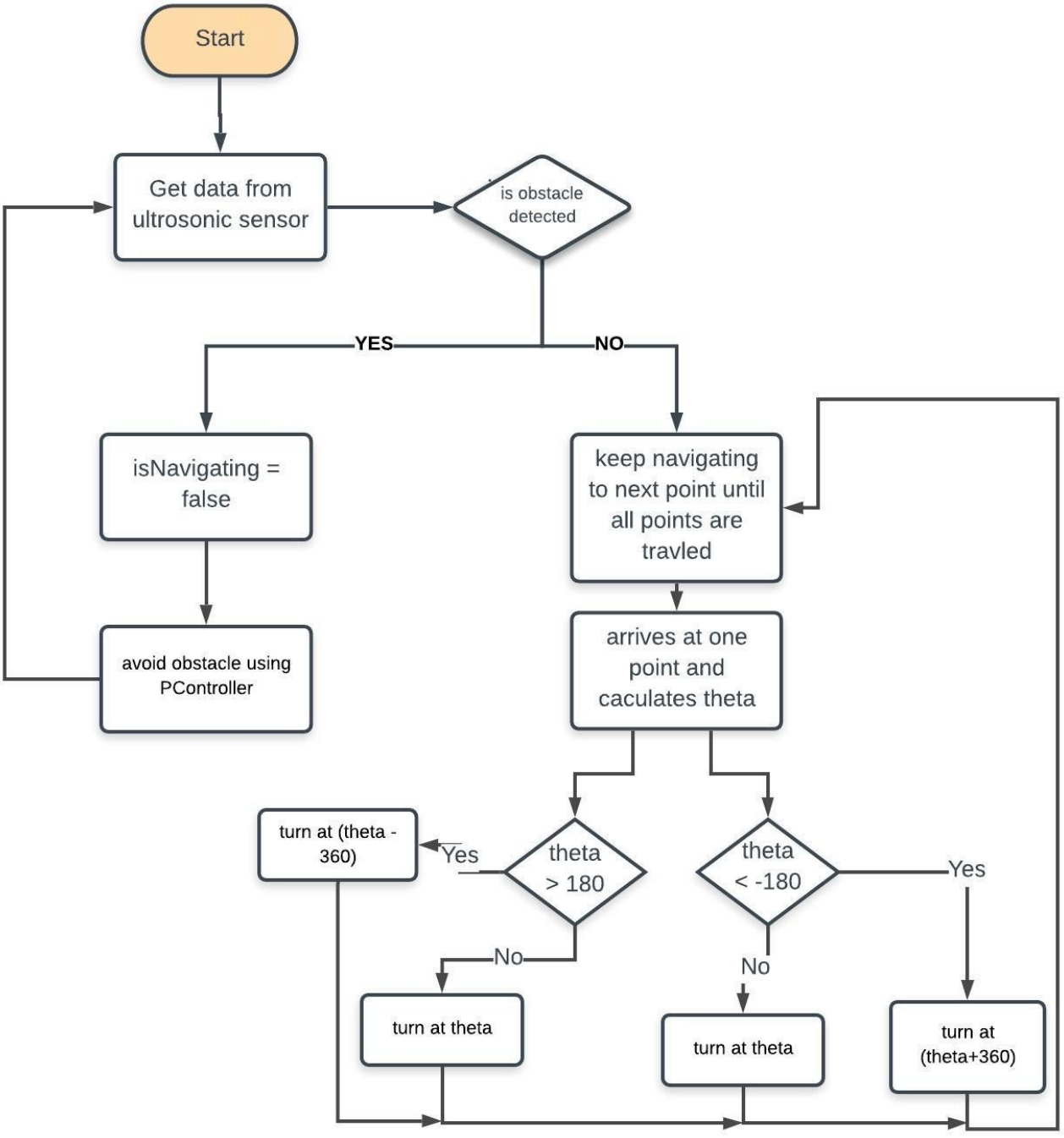

Figure 2. Flow chart

Section 2 and 3: Test Data and Analysis

To find the error, we used this formula:

$$\varepsilon = \sqrt{(X_D - X_F)^2 - (Y_D - Y_F)^2}$$

Where X_D and Y_D are the coordinates of the final destination of the robot and X_F and Y_F are the actual coordinates of where the robot is.

An example of the calculation of the error for the first trial:

$$\sqrt{(91.44 - 90.04)^2 - (30.48 - 31.88)^2} = 1.98 \text{ cm}$$

Table 1: Data of Navigation Test

	X_F (cm)	Y_F (cm)	X_D (cm)	Y_D (cm)	*error value* ε (cm)
1	90.04	31.88	91.44	30.48	1.98
2	89.64	29.18	91.44	30.48	2.22
3	90.04	29.48	91.44	30.48	1.72
4	92.54	31.48	91.44	30.48	1.49
5	93.84	32.28	91.44	30.48	3.00
6	92.14	32.08	91.44	30.48	1.75
7	92.54	31.98	91.44	30.48	1.86
8	93.74	31.58	91.44	30.48	2.55
9	93.14	31.48	91.44	30.48	1.97
10	92.84	32.78	91.44	30.48	2.69

The equation to calculate the mean is:

$$\bar{\varepsilon} = \frac{\sum_{i=1}^{n} \varepsilon_i}{n}$$

Where n is the number of trials which is 10 in our case.

Example of calculation of the mean:

$$\frac{\sum_{i=1}^{10} \varepsilon_i}{10} = 2.12\,\text{cm}$$

The equation to calculate the standard deviation is:

$$\text{SD} = \sqrt{\frac{\sum_{i=1}^{n} (\varepsilon - \bar{\varepsilon})^2}{n}}$$

Where n is the number of trials which is 10 in our case.

Example of calculation of the standard deviation:

$$\sqrt{\frac{\sum_{i=1}^{10} (\varepsilon - \bar{\varepsilon})^2}{10}} = 0.48\,\text{cm}$$

Table 2: Mean and Standard Deviation of the Error Value

Mean (cm)	Standard Deviation (cm)
2.12	0.48

Section 4: Observations and Conclusions

Are the errors you observed due to the odometer or navigator? What are the main sources?

The errors we observed are due to the odometer because when it would turn for the third time, the robot would not have the correct angle, but the odometer showed that it was moving on the right angle, so it went off track. Another source that affected the error is the right tire that slips when the robot goes straight because sometimes, we could observe that the robot was drifting to the left when it was supposed to go straight when going to the first point without turning in the beginning, so the odometer shows the wrong location because of this drift and have an error for the next points until the last point. One of the reasons why this happens is because the ultrasonic sensor was placed slightly more to the left than the robot's center, so there was more weight on the left tire and slightly lifting the right one, which might have caused the slight slip of the right tire.

How accurately does the navigation controller move the robot to its destination?

The navigation controller moves the robot accurately because the error of the final destination and where the odometer says where it is at the end is small, between 0 and 2 mm, but it relied on the odometer values which is not accurate due to the reasons discussed above.

How quickly does it settle (i.e. stop oscillating) on its destination?

The robot stops moving as soon as the navigation says that it reached its destination according to the odometer value, which means there is no oscillation at the end.

How would increasing the speed of the robot affect the accuracy of your navigation?

Increasing the speed of the robot would decrease the accuracy of the navigation because the tires would slip even more and the odometer value will have a bigger error, plus the rotation of the robot would not be as accurate, which will shift the direction of the robot even more and will increase the error significantly.

Section 5: Further Improvements

One hardware improvement we could do is put the motors more to the side so the tires will have less stress on them because of the heavy brick that is directly above them. This would make the tires round and not squeeze them, so the radius of the wheels would be more accurate. We would also center the ultrasonic sensor so the right tire would slip less. As a software improvement, we could modify the value of the wheelbase because each time the robot turned to a new angle, it was not turning accurately towards its next point, but the odometer value showed that it was on the correct angle, and that is due to the wheelbase value because the formula to find the angle on the odometer is based on the value of the wheelbase.