NXT ROBOT Report

Mr. REAUME

TEJ3M

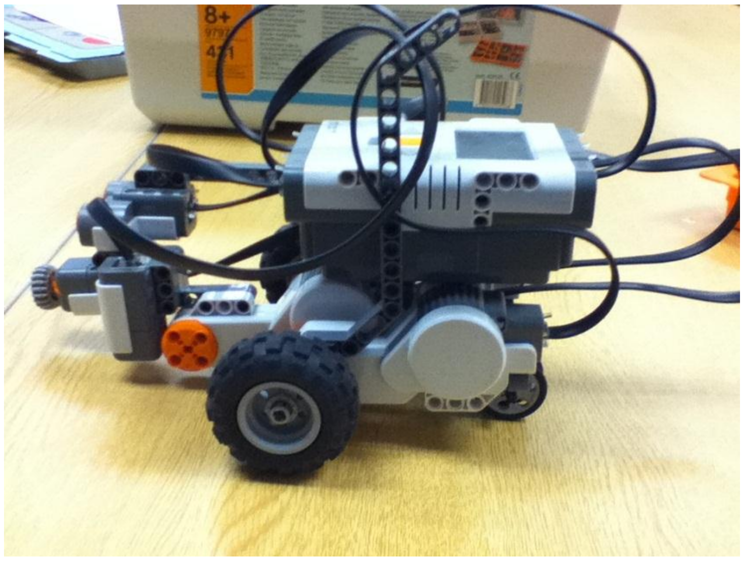
Completed By:

Omar & Teny

INDEX

(1) Identify the problem

- We started off researching a unique design to build our robot into, and after two days of building this robot we noticed that the shape and design this robot took prevented us from progressing any further. The robot was way higher than 6 inches, and sensors were crowded together creating an unbalanced robot and were not being used efficiently.



(2) Develop possible solutions

Brainstorming	Best Solution(s)
-Add more Lego pieces to fit the sensors -Use less sensors -Lower base of robot by eliminating the 3rd arm -Redesign robot	- Redesign the robot - Build on to robot

(3) Construct a prototype

Robot was completely redesigned with a lower base and sensors positioned effectively by following the given "Design Tutorial."

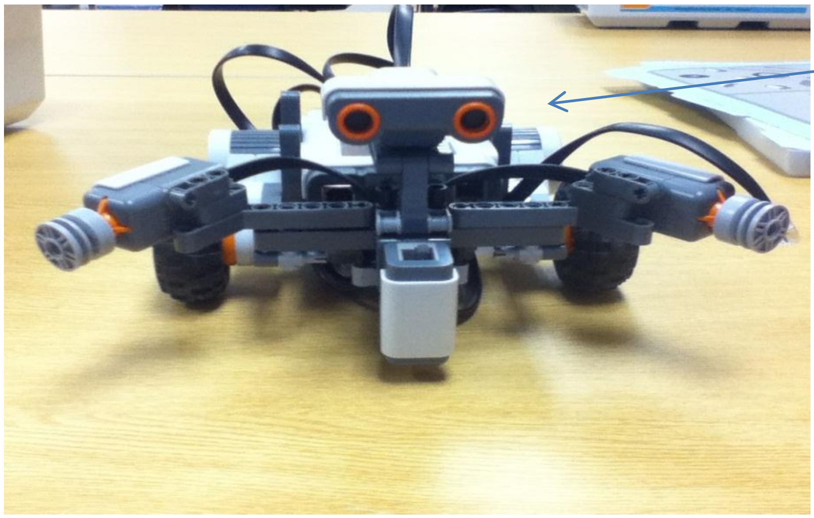
(4) Test and evaluate solution(s)

-Build on to Robot

- The idea of building on to the robot seemed like a good idea at first especially since we were very excited about our unique design from all the other students. But the facts were that the time to be creative and invent new ways was running out. It was more time consuming to invent new ways to make our previous design work than to redesign it completely, therefore we took the robot apart and applied the instructions given in the "Design Tutorial".

-Complete Redesign of Robot

- The robotic measurements finally met design constraints and ran smoothly. At this point sensors were not included (programmed to work) since we were focused more on the design than the programming.



Since the front of Robot was supported by the two Touch-Sensors. The Ultrasonic- Sensor was not needed in the front anymore and therefore positioned to the right side of the robot in order to make the programming much simpler.

(5) Redesign

-We lowered the base, positioned two touch sensors diagonally, positioned the light sensor facing the floor and anchored the ultrasonic sensor to the right side of the robot. To make the robot efficient we used as little weight as possible, so we got rid of the 3rd arm, gave it a small but strong front bumper to hold the sensors, and made sure it was under 6 inches in height.

(6) Conclusion

-After much time spent constructing a design for the robot we came to conclusion that this was by far the most efficient way one should build their robot. Although if a student believes they can build it fashionable yet efficient they should consider the amount of time they are given and the teacher's constraints before beginning.

(1) Identify the problem

-While preforming the line test the robot suddenly turns 180 degrees and goes in the other direction. The robot "checks" too many times while completing the test causing it to take a longer than it should to reach the end.

(2) Develop possible solutions

Brainstorming	Best Solution(s)
-Decrease speed -Decrease turning speed and "Wait" time -Increase Threshold	-Increase Threshold -Slow down the robot

(3) Construct a prototype

-No change to original design of robot. The sensor stayed in the same position.

(4) Test and evaluate solution(s)

-Threshold

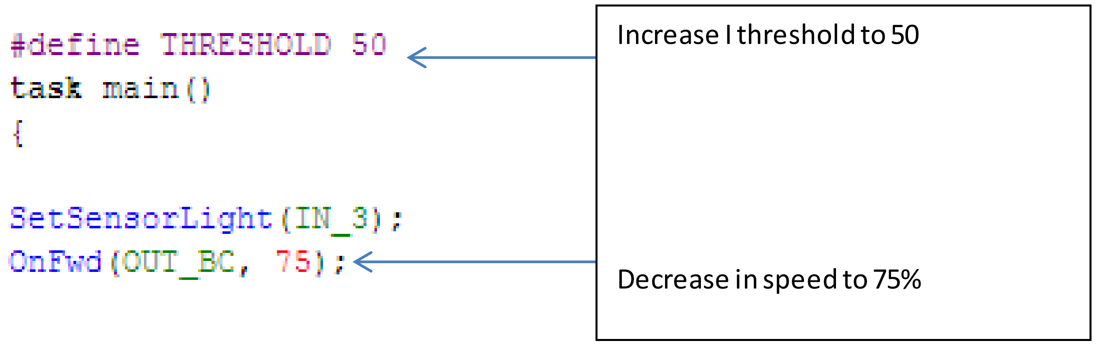
- The idea of increasing the threshold was to make it clear for the robot what we want it to follow. The threshold we started off with was at 30, but eventually we increased it bit by bit until it was easy and simple for the robot to understand it's mission.

-Decrease Robot Speed

- We thought that maybe the robot is getting off track due to intensity of speed. We slowed it down to 75 and surprisingly it preformed much better.

(5) Redesign

The program was modified by changing the speed of the robot to 75% and the threshold to 50.



```
#define THRESHOLD 50
task main()
{

SetSensorLight(IN_3);
OnFwd(OUT_BC, 75);
```

(6) Conclusion

This issue was resolved by adding both solutions into the program which gave us great results. The robot was able to complete the line test with no problems. It did turn around 180 degrees during the test although this could be due to a glitch in the program because it follows all the other commands it's given and none of them is a command to turn 180 degrees. The checking issue was still there but much less.

(1) Identify the problem

-There are two touch sensors positioned diagonally on the robot, so that if the robot hit's an obstacle it will back up and turn depending on which sensor was touched. Yet the right touch sensor does not seem to respond when it's touched.

(2) Develop possible solutions

Brainstorming	**Best Solution(s)**
-Replace Sensor -Reprogram	-Reprogram

(3) Construct a prototype

-No change to original design of robot. The sensor stayed in the same position.

(4) Test and evaluate solution(s)

-Reprogram

- The programming was checked over and over again until finally a very small but costly mistake was identified. The parenthesis used '{' after the *'if'* statement was in fact a regular bracket symbol instead '('.

(5) Redesign

The issue was resolved through modification to the programming.

(6) Conclusion

This issue may have occurred to many other students, and the only solution to it is to check the programming. This may sound easy but it truly becomes frustrating when you preform check after check with no results. So we decided to organize the program by lining up the *'while' and 'if'* 'statements and closing the parenthesis at the end of each command. We patiently went over the programming scrutinizing every letter typed, and lining things up to stay organized until we finally found a small mistaken bracket symbol for a parenthesis.

(1) Identify the problem

-When a touch sensor is touched we expected the ultrasonic to take part in deciding which way the robot should turn, but of course it didn't.

(2) Develop possible solutions

Brainstorming	**Best Solution(s)**
-Put an ultrasonic *'if'* statement inside a touch-sensor *'if'* statement -Close the touch *'if'* statement and make another ultrasonic if statement below it -Join the touch-sensors together with a || symbol and adding an ultrasonic command below it. -Add an *'else'* statement	-Add an *'else'* statement -Join the touch-sensors with a || and add an ultrasonic *'if'* statement below them.

(2) Construct a prototype

-At this point the Ultrasonic-Sensor was actually positioned facing the right side of the robot. This was due to lessen the amount of programming needed and make things simple. If a touch-sensor was touched, the ultrasonic sensor would check if there was a wall in front of it, (depending on the distance in the programming) if there was a wall, then the robot would turn left. If there wasn't the robot would turn right.

.

(4) Test and evaluate solution(s)

-Reprogram

- The ideas in the brainstorming above all belong under the category of programming. It was confusing which statement we had to use in order to make the robot carry out the function we wanted it to carry out. So after testing out all of them we finally settled on the *'else'* statement. This statement was very simple to use; after an *'if'* statement indicating a touch sensor was touched, you simply close the *'if'*statement, type the statement *'else'* below it and the function you would like the robot to carry out.

(5) Redesign

The issue was resolved through modification of the programming.

(6) Conclusion

In conclusion before a person goes into thinking how they want their robot to function , it is truly important to understand the programming behind the NXT robot and what each command means. This would not only save you time but a lot of frustrations. Eventually we figured out that the *'else'* statement was the right statement to carry out the function we intended for this robot. But if we had completely read the programming tutorial and understood it, it would have taken us very minimal time to complete this function.

(1) Identify the problem

- During the maze test our robot was only able to make it till the 3rd turn. Then it would either get stuck or turn back in the other direction.

(2) Develop possible solutions

Modifications Should be Made to Robot
-Add 3rd arm with ultrasonic sensor facing forward with the ability to check both ways. -Create an entire touch-sensor bumper -Increase threshold of ultrasonic-sensor

(3) Construct a prototype

-Before the maze test, the robot's design was slightly modified after running some tests. We had to lower the touch sensors and make them face forward at a 90 degree instead of diagonally. This resolution was to eliminate the robot from getting stuck when it hits an object directly in the middle.

.

(4) Solutions

-Redesign

- The robot need a 3rd arm and an ultrasonic –sensor that is able to check both ways before making a turn. Although this solution may require more programming it is the best way to ensure that our robot will make the turns with no difficulties. The robot also needs a wide and strong front bumper that is attacked to the touch sensors; this will prevent our robot from getting stuck when hitting a triangular shaped object in the middle.

(6) Conclusion

-Things could have gone better if we had applied the modifications above to the robot. The more tests we'd run the more we would find out about the robot. Instead of spending most of the time building we should have spent more time programming and testing it. This was truly a fun learning experience.