Toeing the Line

Experiments with Line-following Algorithms

Jonathan A. Gray
Grade 9

Contents

Toeing the Line

Experiments with Line-following Algorithms

Jonathan A. Gray

Abstract

Following lines is an essential part of robot navigation in a First LEGO League robotics competition. However, programming a robot to follow a line is not an easy task. This project compares several algorithms that robots can use for line following. I hypothesized that robots that were programmed to stay closer to the line and stop less for corrections would be faster and more accurate. For the experiment, a basic LEGO robot used six different algorithms to follow a winding black line, and the robot's speed and accuracy on each run were recorded. My hypothesis proved to be incorrect. The results showed that algorithms programmed to continuously search for the line were more accurate than those that tried not to lose the line. It was also found that the faster algorithms sacrificed accuracy for speed.

Introduction

First LEGO League is a worldwide competition in which teams build LEGO robots to complete a series of missions under a time limit. One of the biggest problems our team faced involved getting the robot from place to place quickly and accurately. There were lines on the field leading to various mission objectives, and we were allowed the use of two light sensors.

Following those lines sounds like a pretty easy task, right? Wrong! Our robot persisted in leaving the line. Our navigation failures inspired me to use this science project as an opportunity to improve our chances for next year.

Purpose

Which one of six specific algorithms will allow a robot with one or two light sensors to follow a winding black line on a white background most accurately and most quickly?

A light sensor outputs a number from 0 to 100 depending on how much light enters the lens. In this way, the light sensor can "see" a small area on the floor. When the sensor is over a black line, it reads a lower number than when it is over a white background. If it "sees" an area that is part black and part white (e.g. when it is over the very edge of the line), it registers an in-between, or "gray" value.

The six algorithms tested in this experiment use one or two light sensors. Algorithms 1-5 start centered over the line, with the back of the robot flush with the beginning of the line. Algorithm 6 begins on the side of the line, with the back light sensor flush with the beginning of the line. Algorithms 1, 3, and 6 follow the side of the line, and although they are described as following only the left side of the line, they are tested on both sides.

#	Name	Sensors	Description
1	FOLLOW GRAY	1 sensor in front	The robot follows the edge of the line: if the sensor detects "gray" the robot goes straight; if it detects black it turns towards the left, and if it detects white it turns towards the right.

#	Name	Sensors	Description
2	ONE INSIDE	1 sensor in front	The robot zigzags inside the line: it turns toward the left until it detects white, then turns toward the right until it detects black and then white, then left until black and white, and so on.
3	ONE BOUNCE	1 sensor in front	The robot bounces off the side of the line: it turns left until it detects white, then right until black, then left until white, and so on.
4	STRADDLE	2 sensors in front	The light sensors are positioned on either side of the line: the robot goes straight while both sensors detect white, left when the left sensor detects black, and right when the right sensor detects black.
5	TWO INSIDE	2 sensors in front	The light sensors are positioned side-by-side inside the line: while both sensors detect black, the robot goes straight; when the left light sensor detects white, the robot turns right, and when the right sensor detects white, the robot turns left.
6	TWO BOUNCE	2 sensors: one in front and one in back	The robot bounces off the side of the line, with one light sensor in front and one in back: it turns toward the right until the front sensor detects white, then turns left in place until the rear sensor detects black, then turns toward the right until the front sensor detects black, and so on.

Hypothesis

I predict that the algorithms will rank from fastest and most accurate to slowest and least accurate as follows:

Algorithm 4 – STRADDLE
Algorithm 1 – FOLLOW GRAY
Algorithm 5 – TWO INSIDE
Algorithm 2 – ONE INSIDE
Algorithm 3 – ONE BOUNCE
Algorithm 6 – TWO BOUNCE

These predictions are based on how close to the line the robot is programmed to stay and how much time it should spend correcting its route.

Materials

- 1x RoboLab 2.5 software running on a PC
- 1x LEGO infrared tower with cable
- 1x LEGO RCX 1.0
- 1x 12V AC/DC adapter
- 1x 125V extension cord
- 2x LEGO 9V motors with gear reduction
- 2x LEGO light sensors

- Assortment of non-electric LEGO parts included in Robotics Invention System 2.0
- 1x Robotics Invention System 2.0 Constructopedia
- 2x 22x28 in. pieces of white poster board
- ¾ in. black electrical tape
- Transparent packing tape
- 1x Digital stopwatch
- 1x 4x8 ft. flat table with fluorescent shop light hanging above
- 2x Halogen work lamps

Setup

Programming the robot:

1. In RoboLab, create a program for each algorithm; for algorithms that follow the side of the line, create two programs. (See Figures 1-9)
2. Download the first five programs to the RCX using the IR tower.

Building the robot:

1. Assemble the following using LEGO parts, according to the instructions in the Roverbot section of the Constructopedia (Robotics Invention System 2.0 Constructopedia, 10):
   - 1x Driving base (See Figure 10)
   - 4x Front wheel attachments (See Figure 11)
   - 2x Light sensor attachments
2. Attach the wheels to the driving base. (See Figure 12)
3. Attach one light sensor to the front of the robot using a yellow double-peg. (See Figure 13)
4. Set aside the other light sensor and the following additional pieces, which will be used to make necessary modifications for certain algorithms:
   - 1x Yellow double-peg
   - 2x Long black pegs
   - 2x Long pegs with axle connector at one end
   - 2x Half-peg/half-axles
   - 1x Axle (6 pegs long)
   - 4x 1x1 conical bricks

Setting up the test area:

1. Create a line in two parts using black electrical tape on poster board. (See Figures 14-15)
2. Tape the two sections together on the back with packing tape, making sure that the lines on the two boards line up. (See Figure 16)
3. Place the line on the table.
4. Place the halogen lamps on opposite sides of the table.
5. Turn on the halogen lamps and the shop light above the table.
6. Plug the extension cord into an outlet near the table.
7. Plug the AC/DC adapter into the robot, and plug the other end into the extension cord.

Procedure

1. Place the robot on the table so that the back of the robot is flush with the beginning of the line, and the light sensor is centered over the line.
2. Start the stopwatch and run Program 1 on the RCX at the same time.
3. Hold the power cord lightly above the robot so that the robot does not drive over it. Do not apply any resistance to the power cord.
4. Stop the stopwatch when the robot reaches the end of the line or leaves it, and record the time displayed in seconds.
5. Measure the distance the robot traveled along the line, and record that distance in inches.
6. Repeat Steps 1-5 with Program 2.
7. Repeat Steps 1-6.
8. Repeat Steps 1-5 with Program 3, four times.
9. Repeat Steps 1-5 with Program 4.
10. Repeat Steps 1-5 with Program 5.
11. Repeat Steps 9-10.
12. Download the final four programs to the RCX.
13. Remove the light sensor attachment from Roverbot.
14. Insert the half-peg/half-axles into the long pegs with axle connectors on one end. (See Figure 17)
15. Attach the two light sensor attachments to each other using the parts created in Step 14. (See Figure 18)
16. Attach the four 1x1 conical bricks to the front of Roverbot. (See Figure 19)
17. Attach the light sensors to the conical bricks. (See Figure 20)
18. Repeat Steps 1-5 with Program 1, four times.
19. Remove the light sensor attachment and conical bricks from the robot.
20. Separate the light sensor attachments so that they are in their original form (refer to Setup: Building the Robot: Step1).
21. Remove the right-angle piece and pegs from one side of one light sensor attachment.
22. Remove the same pieces from the opposite side of the other light sensor attachment.
23. Insert two light gray pegs in place of the ones taken out in Step 22.
24. Remove the 4-peg axles from both light sensor attachments.
25. Insert one 6-peg axle in place of the two 4-peg axles removed in Step 24, attaching the two light sensors to each other. (see Figure 21)
26. Attach the light sensors to the front of Roverbot with two long black pegs. (See Figures 22-23)
27. Repeat Steps 1-5 with Program 2, four times.
28. Remove the light sensor attachment and return the light sensors to their original, separate, attachments.
29. Using the yellow double-pegs, attach one light sensor to the front of Roverbot and one to the back. (See Figure 24)
30. Place the robot on the table so that the back of the robot is flush with the beginning of the line, and the light sensor is centered over the line.
31. Repeat Steps 2-5 with Program 3.
32. Repeat Steps 30-31 with Program 4.
33. Repeat Steps 30-32.

34. Calculate a "score" for accuracy in percent by dividing the distance traveled over the total distance of the line.
35. Calculate speed in inches per second by dividing distance traveled over time.
36. Calculate a "score" for speed in percent by dividing the speed over the highest speed encountered.
37. Calculate an overall "score" by averaging the accuracy "score" and the speed "score."

Results

Algorithm 1 – FOLLOW GRAY:				
	Distance (in.)	*Time (s)*	*Accuracy (%)*	***Speed (in./s)***
Trial 1 (left)	106	22.57	83%	4.70
Trial 2 (right)	53	12.04	42%	4.40
Trial 3 (left)	106	22.59	83%	4.69
Trial 4 (right)	53	12.53	42%	4.23
Average	79.50	17.43	63%	4.56

Algorithm 2 – ONE INSIDE:				
	Distance (in.)	*Time (s)*	*Accuracy (%)*	***Speed (in./s)***
Trial 1	49	18.19	39%	2.69
Trial 2	92	31.56	72%	2.92
Trial 3	53	20.18	42%	2.63
Trial 4	92	32.28	72%	2.85
Average	71.50	25.55	56%	2.80

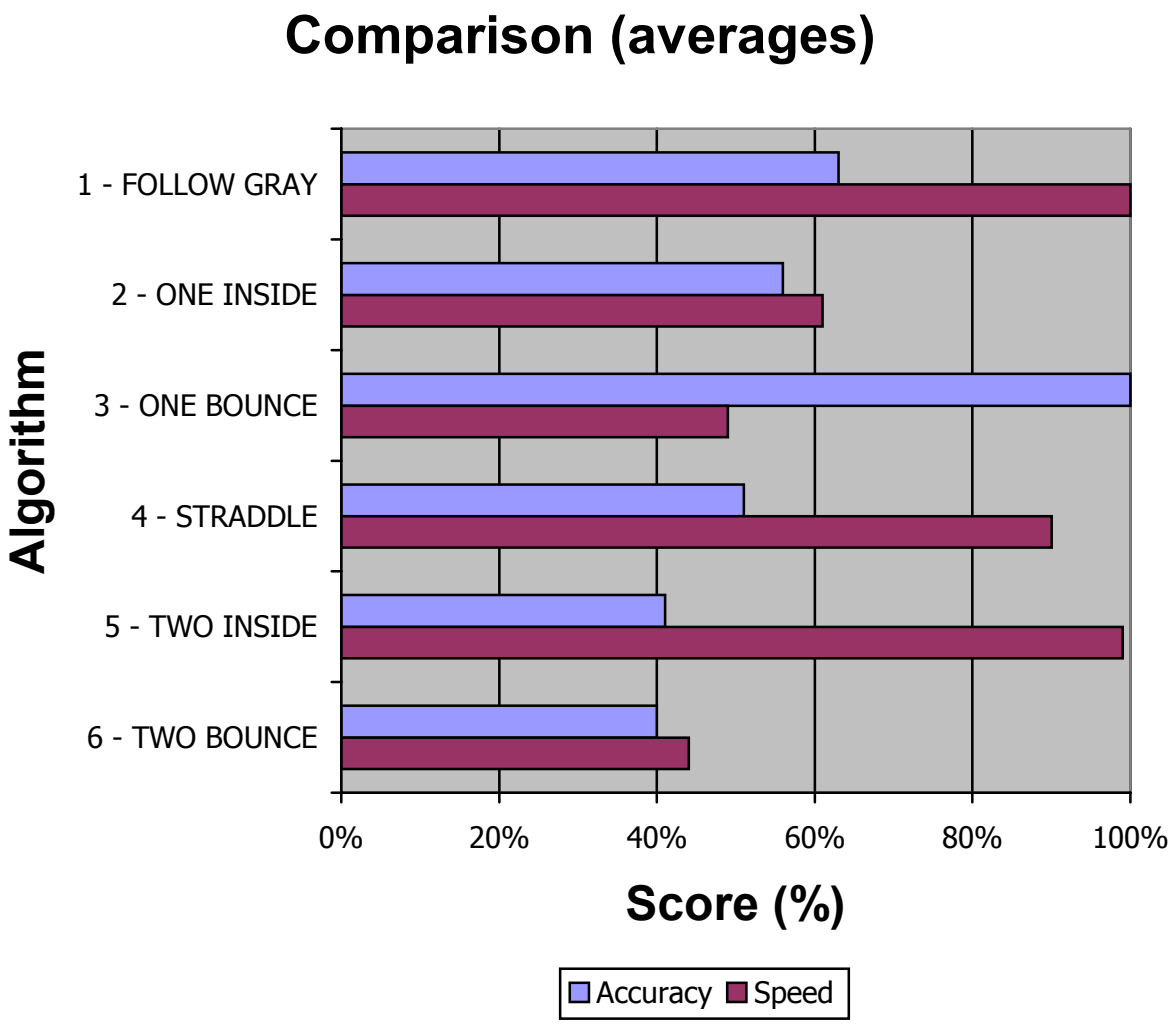
Algorithm 3 – ONE BOUNCE:				
	Distance (in.)	*Time (s)*	*Accuracy (%)*	***Speed (in./s)***
Trial 1 (left)	127	55.62	100%	2.28
Trial 2 (right)	127	58.19	100%	2.18
Trial 3 (left)	127	55.72	100%	2.28
Trial 4 (right)	127	58.72	100%	2.16
Average	127.00	57.06	100%	2.23

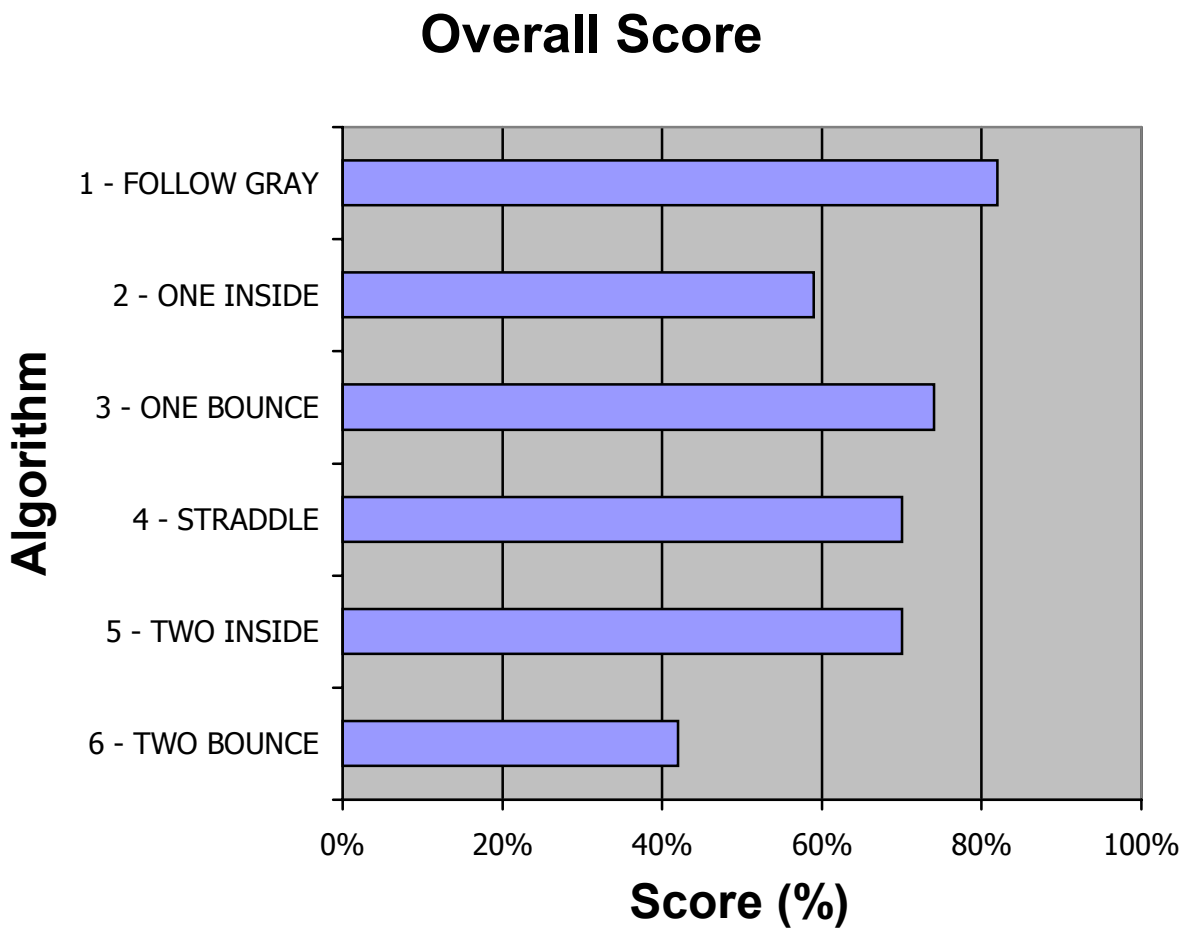
Algorithm 4 – STRADDLE:				
	Distance (in.)	*Time (s)*	*Accuracy (%)*	***Speed (in./s)***
Trial 1	51	11.65	40%	4.38
Trial 2	51	12.00	40%	4.25
Trial 3	51	11.66	40%	4.37
Trial 4	107	28.29	84%	3.78
Average	65.00	15.90	51%	4.09

Algorithm 5 – TWO INSIDE:				
	Distance (in.)	*Time (s)*	*Accuracy (%)*	***Speed (in./s)***
Trial 1	53	11.40	42%	4.65
Trial 2	52	11.00	41%	4.73
Trial 3	51	11.94	40%	4.27
Trial 4	51	11.43	40%	4.46
Average	51.75	11.44	41%	4.52

Algorithm 6 – TWO BOUNCE:				
	Distance (in.)	*Time (s)*	*Accuracy (%)*	***Speed (in./s)***
Trial 1 (left)	49	21.22	39%	2.31
Trial 2 (right)	57	30.65	45%	1.86
Trial 3 (left)	49	24.25	39%	2.02
Trial 4 (right)	49	25.07	39%	1.95
Average	51.00	25.30	40%	2.02

Comparison (averages)				
Algorithm	**Accuracy (%)**	**Speed (in./s)**	**Speed (%)**	**Overall Score (%)**
1 - FOLLOW GRAY	63%	4.56	100%	82%
2 - ONE INSIDE	56%	2.80	61%	59%
3 - ONE BOUNCE	100%	2.23	49%	74%
4 - STRADDLE	51%	4.09	90%	70%
5 - TWO INSIDE	41%	4.52	99%	70%
6 - TWO BOUNCE	40%	2.02	44%	42%





Conclusions

My hypothesis was incorrect. The algorithms with one light sensor are more accurate than those with two. Perhaps this is a result of the way they follow the line. The algorithms with one light sensor continuously search for the line, so if they lose their way and get a little off track, they keep looking and eventually find the line again. When a robot with two light sensors overshoots the line on a sharp turn, it sees the other side of the line and, thinking it has turned successfully, straightens its course, bringing it away from the line and completely off track.

Most of the faster algorithms have compromised accuracy for speed. Algorithms 2 and 3 spend a lot of time correcting their paths, which makes them accurate, but slow. Algorithms 1, 4, and 5 spend more time driving straight, which explains their speed. Algorithm 6 is the only algorithm that turns in place, so it is very slow, and it turns back toward the line in a very wide arc, resulting in inaccuracy. Perhaps if the method for turning were changed (e.g. turning in place instead of forward turns) many of the algorithms would have been more accurate, but slower.

Further research and experimentation could be done on different methods of turning and different distances between two light sensors. The mechanical aspects of line-following robots might also be investigated.

Pictures

Figure 1: Algorithm 1: Left Side:

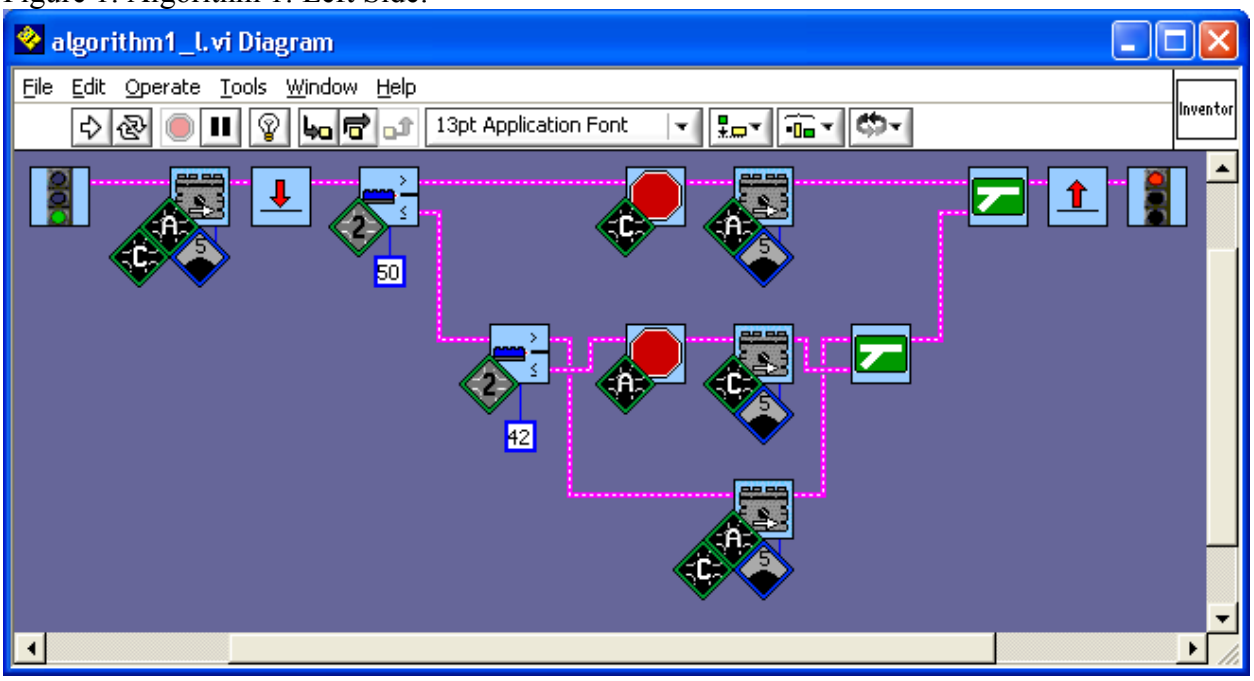


Figure 2: Algorithm 1: Right Side:

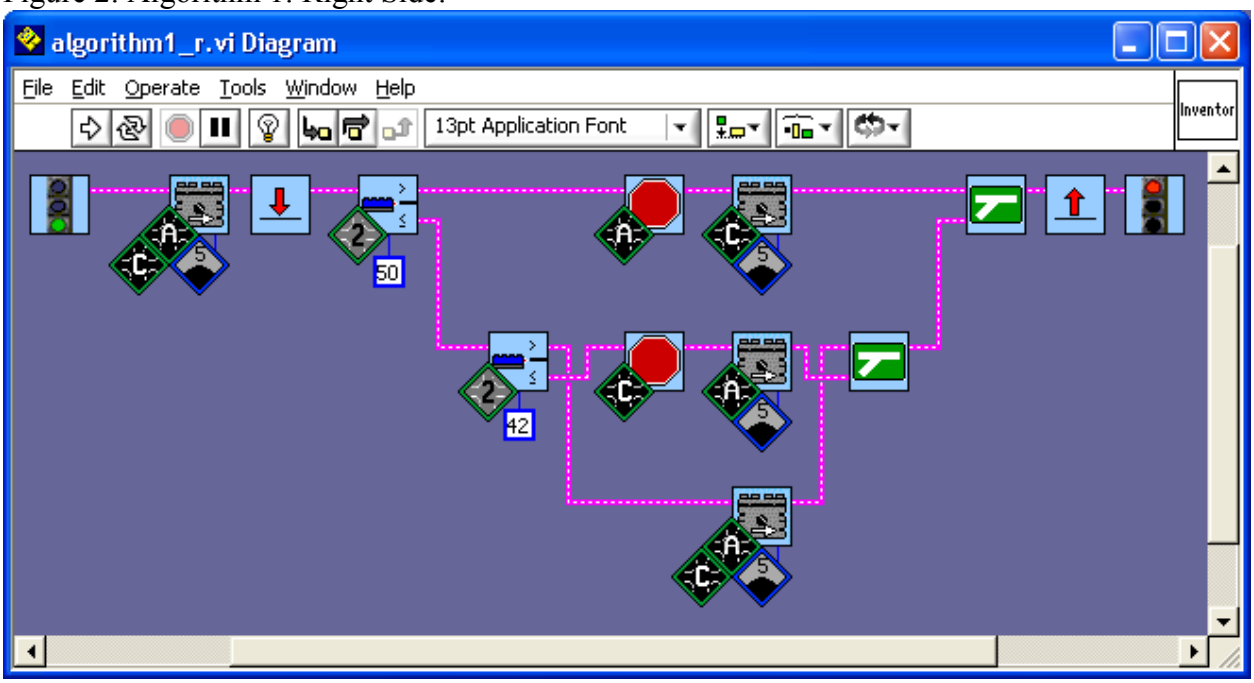


Figure 3: Algorithm 2:

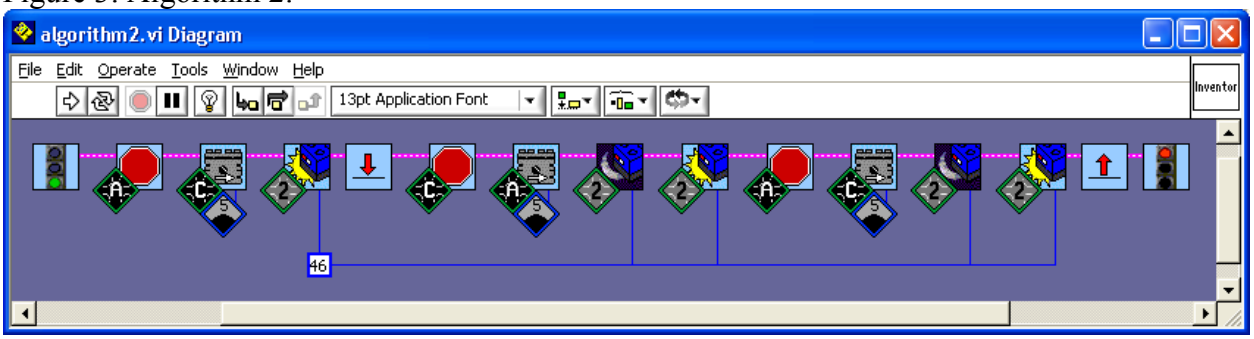


Figure 4: Algorithm 3: Left Side:

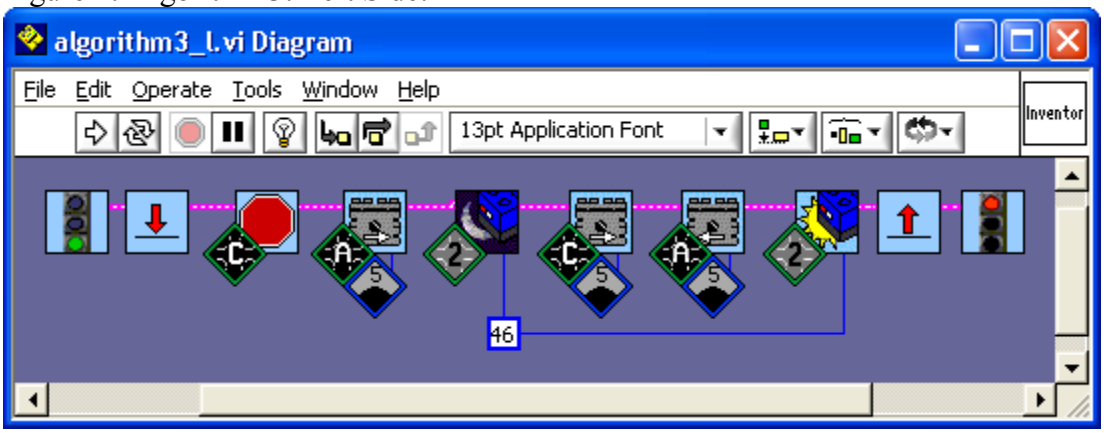


Figure 5: Algorithm 3: Right Side:

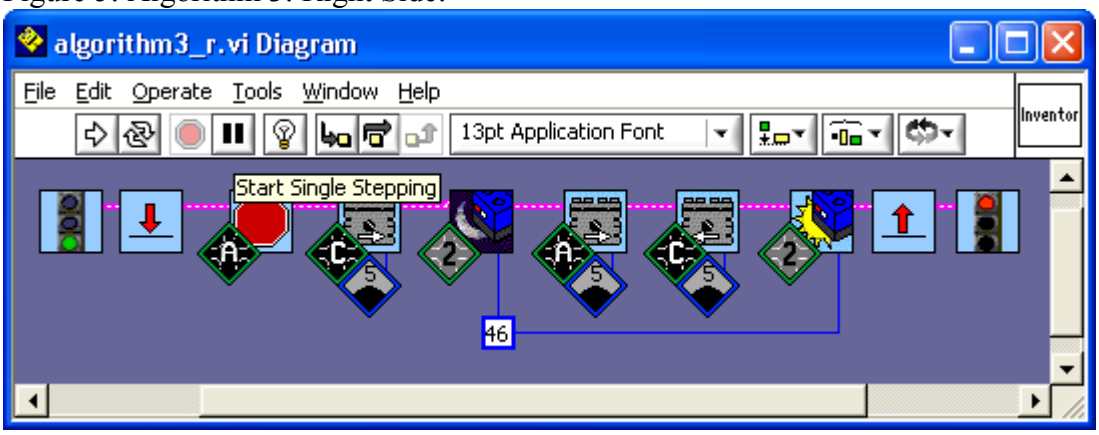


Figure 6: Algorithm 4:

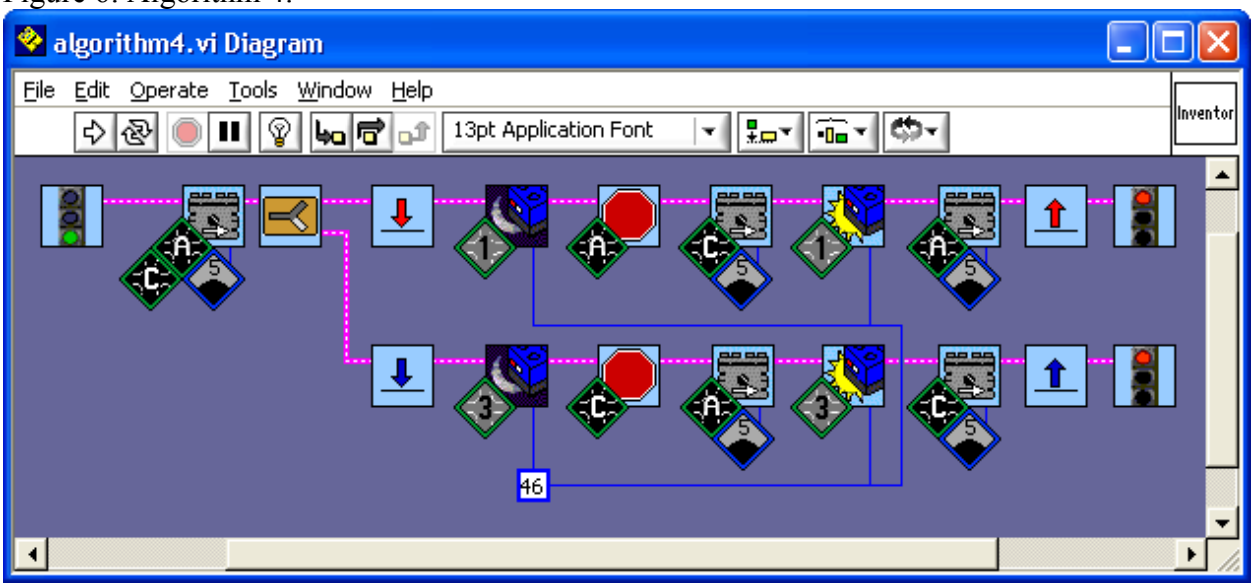


Figure 7: Algorithm 5:

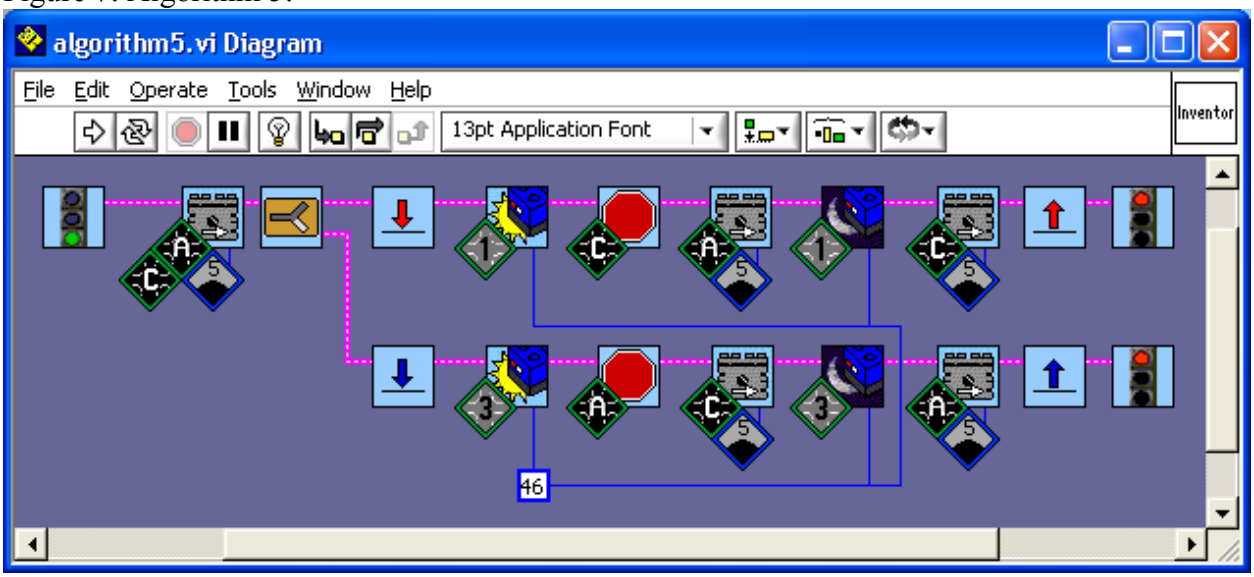


Figure 8: Algorithm 6: Left Side:

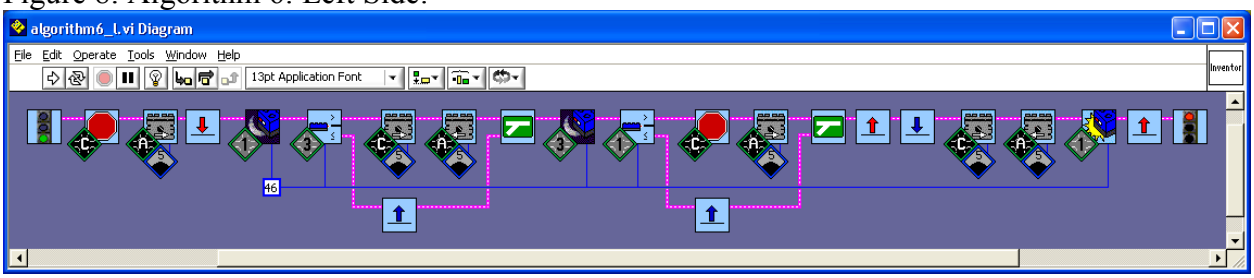


Figure 9: Algorithm 7: Right Side

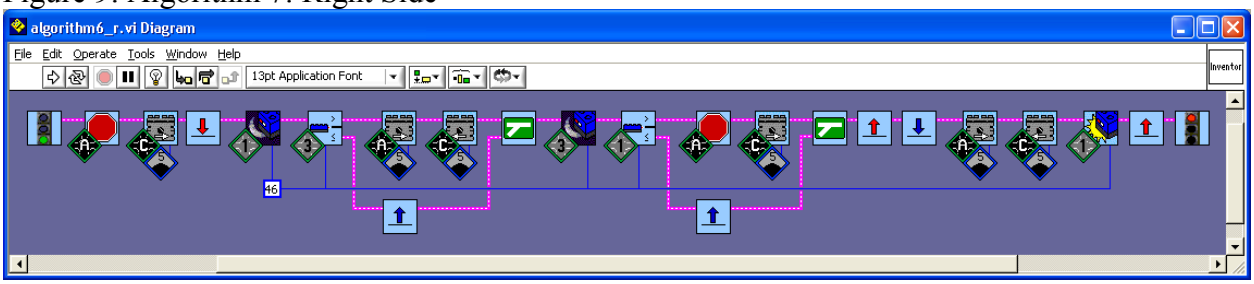


Figure 10: Roverbot Driving Base:

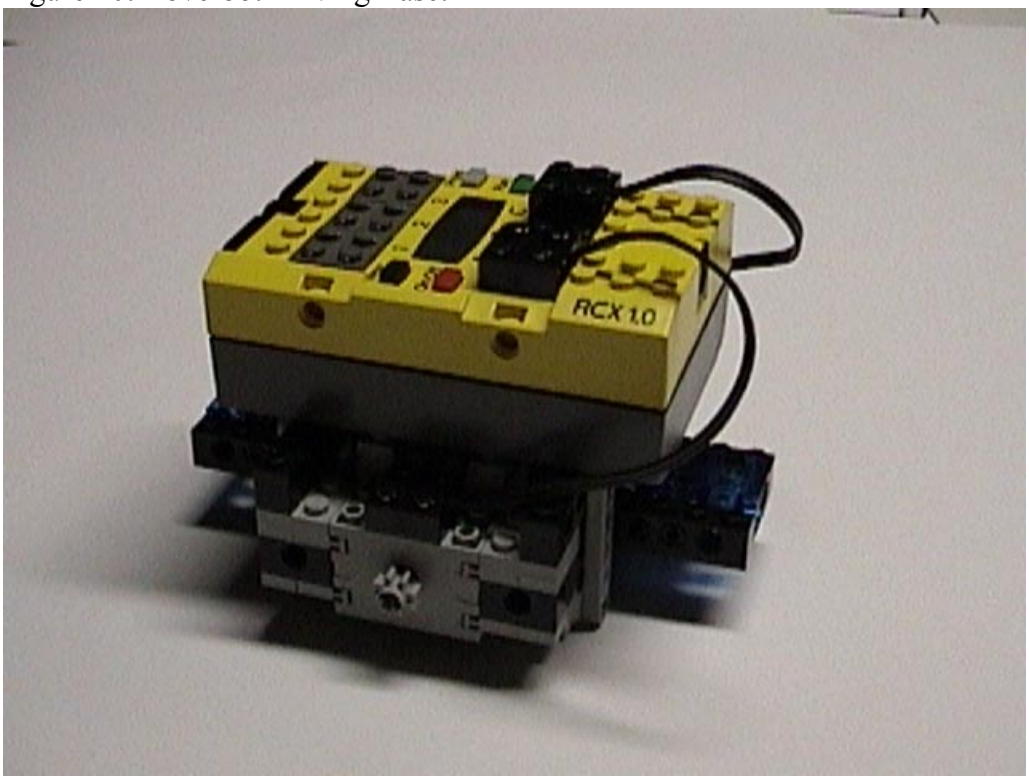


Figure 11: Roverbot Front Wheel Attachments:

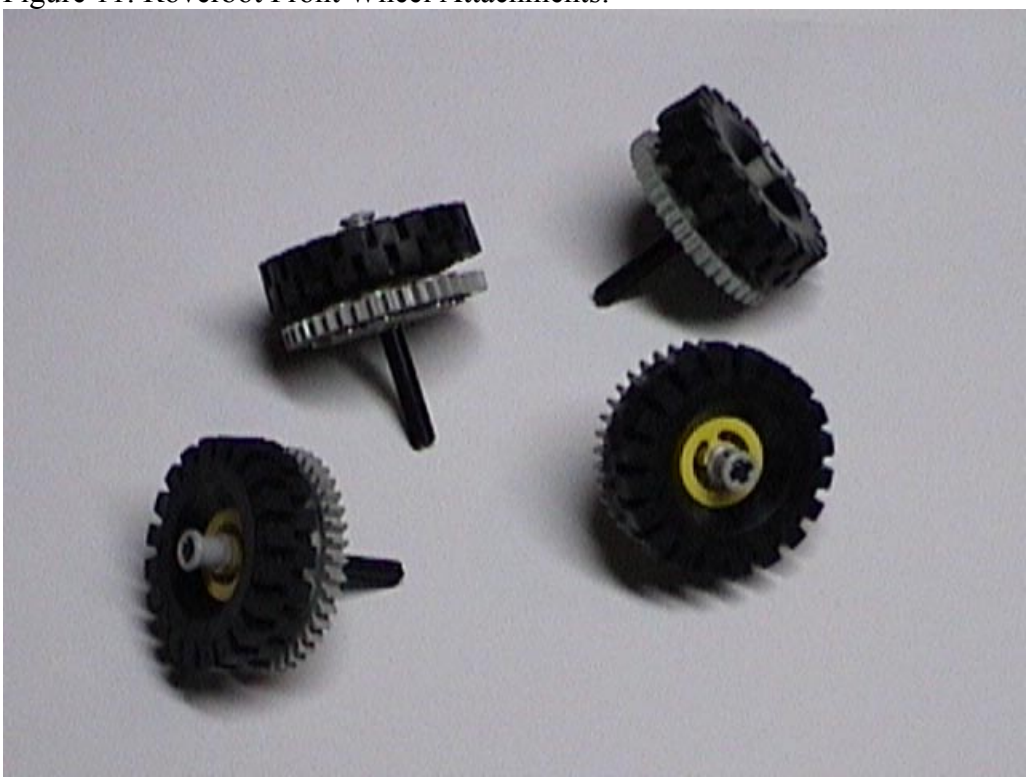


Figure 12: Roverbot with Wheels:

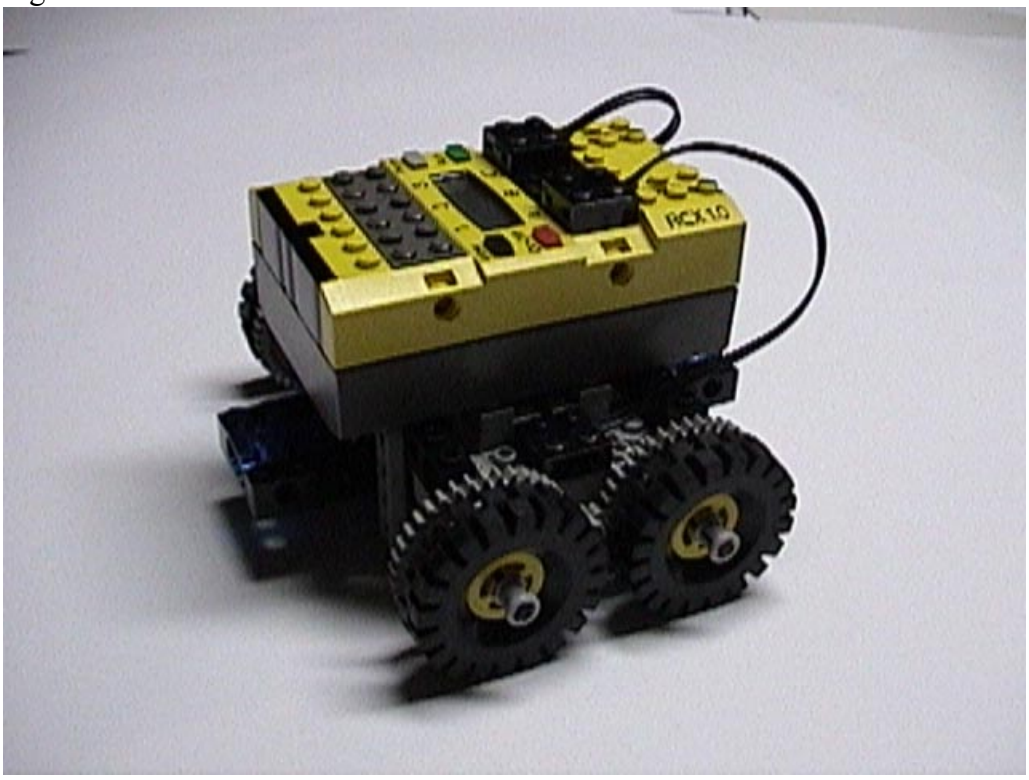


Figure 13: Roverbot with One Light Sensor:

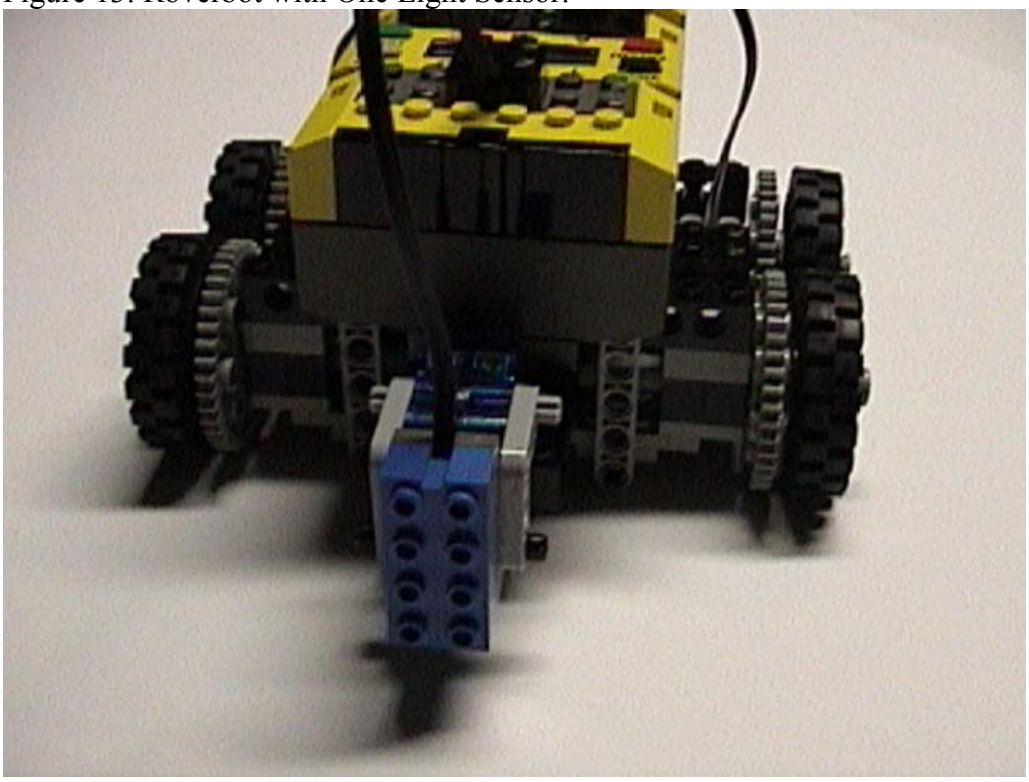


Figure 14: First Section of the Line:

Figure 15: Second Section of the Line:

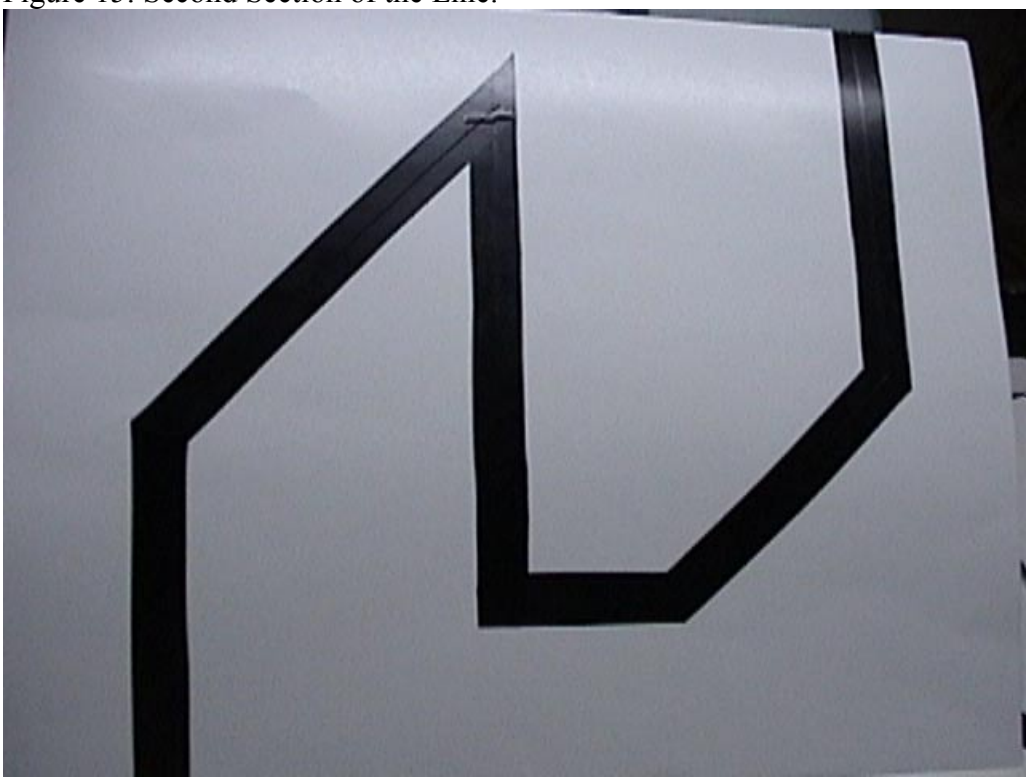


Figure 16: Entire Line:

Figure 17: Half-peg/half-axles Inserted Into Long Rods with Axle Connectors:

Figure 18: Light Sensors Attached to Each Other:

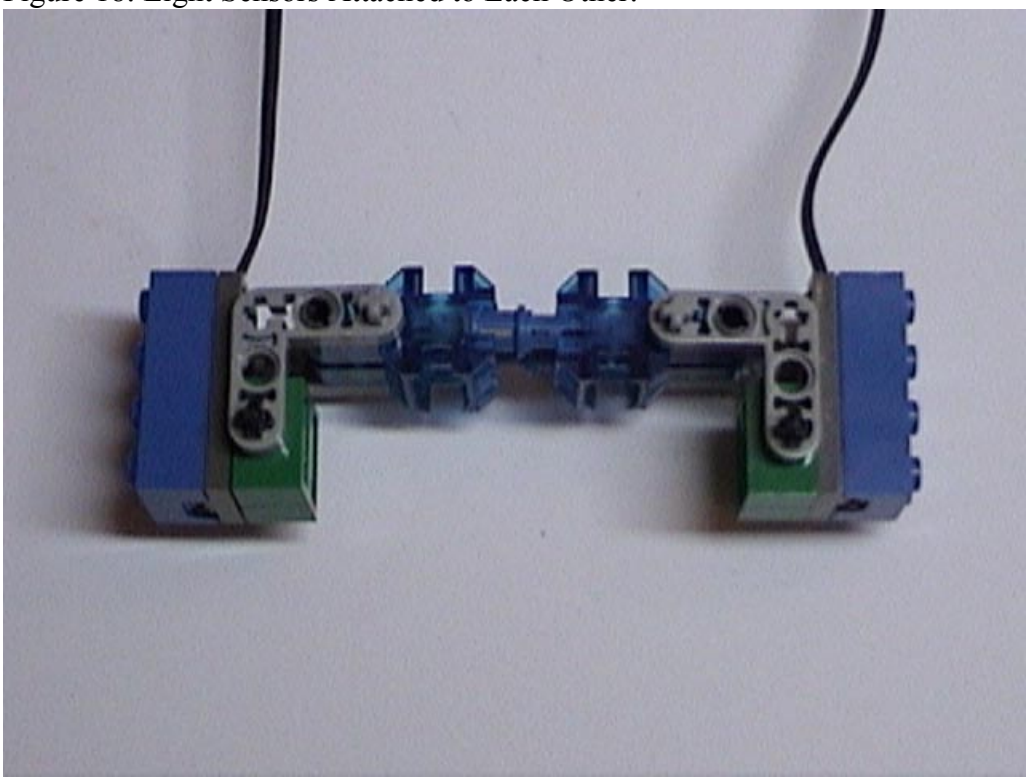


Figure 19: 1x1 Conical Pegs on Front of Roverbot:

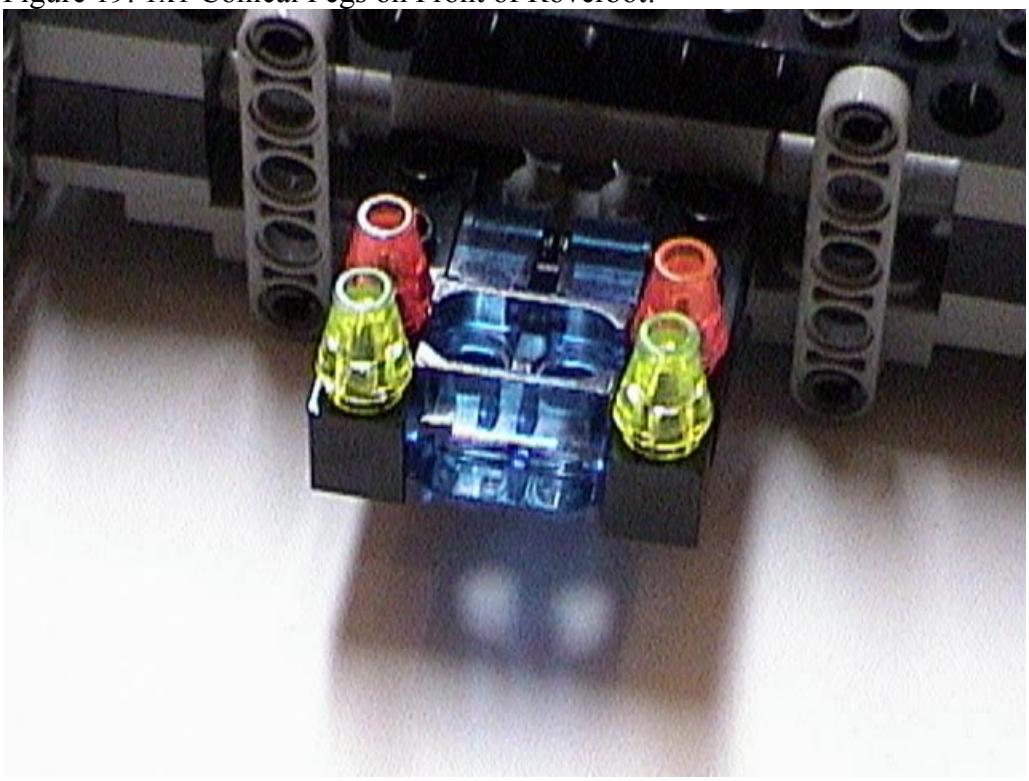


Figure 20: Roverbot, Configured to "Straddle" the Line:

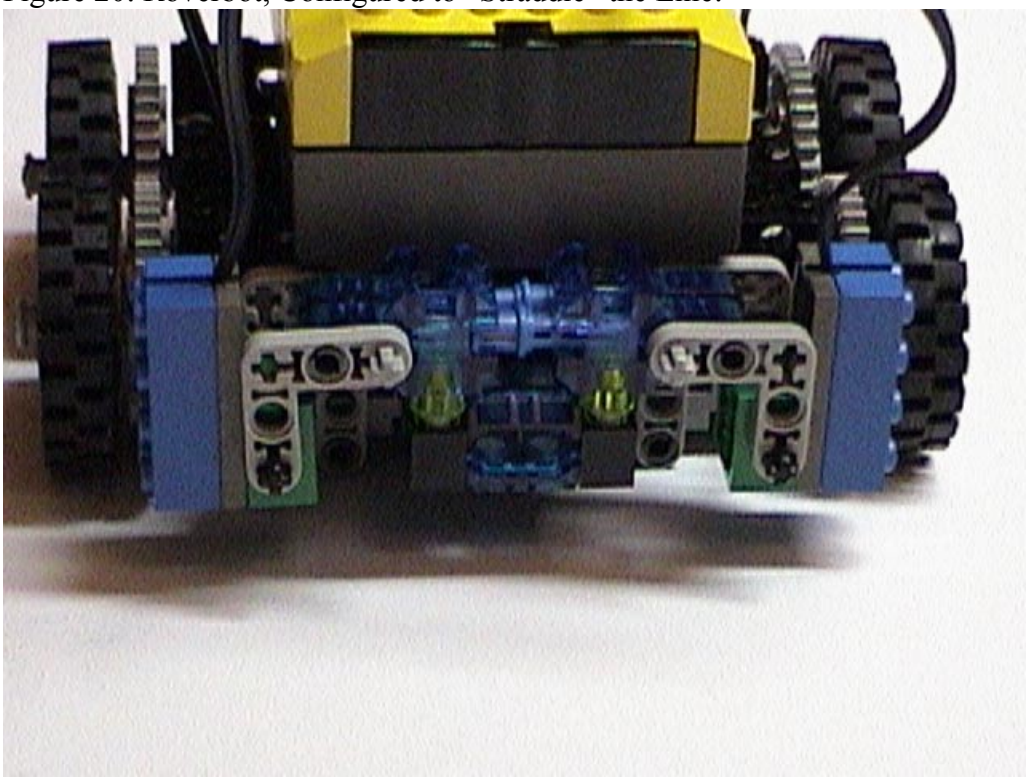


Figure 21: Attaching Two Light Sensors for Algorithm 5:

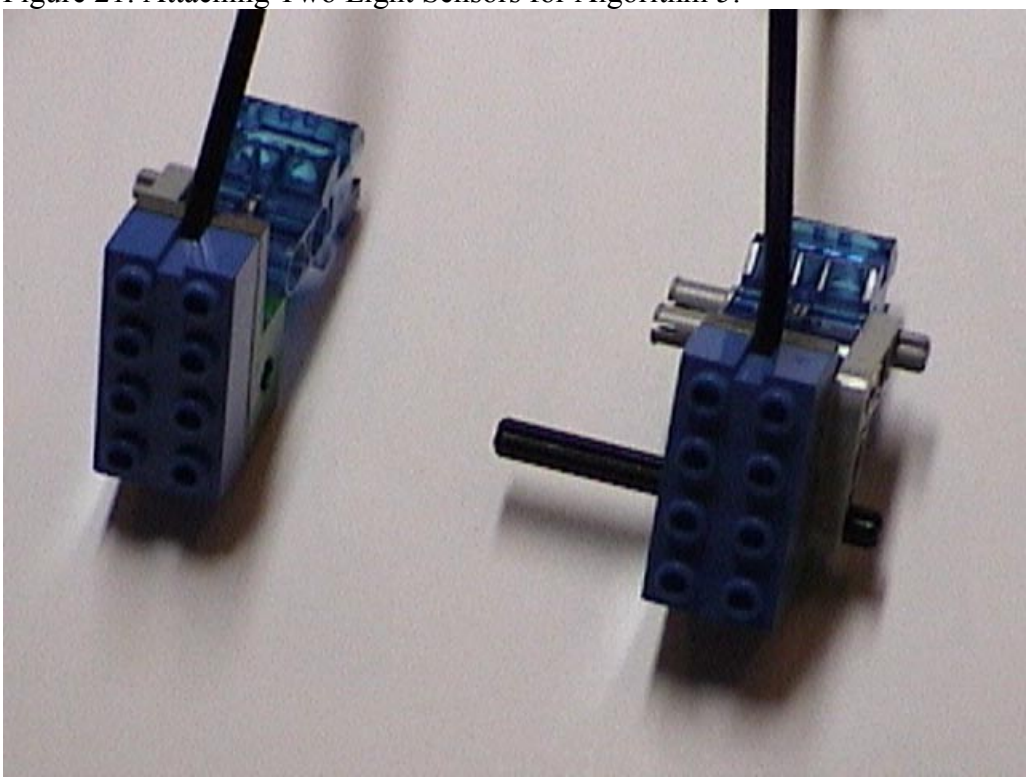


Figure 22: Attaching Long Black Pegs to the Light Sensors:

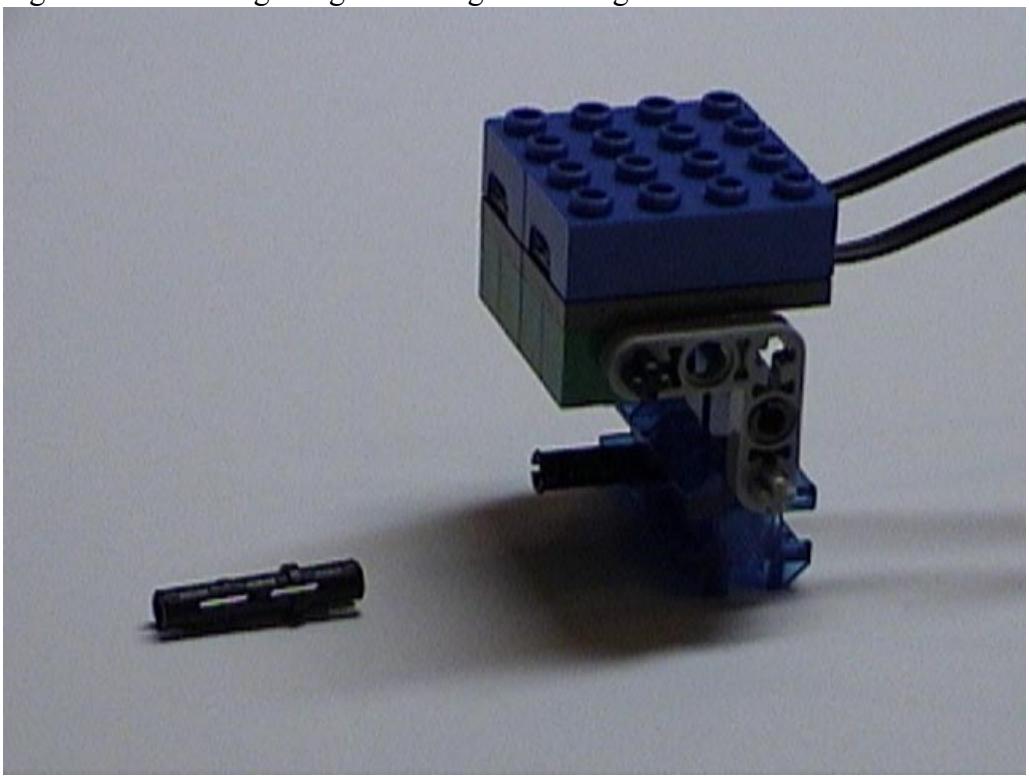


Figure 23: Light Sensors Configured for Algorithm 5:

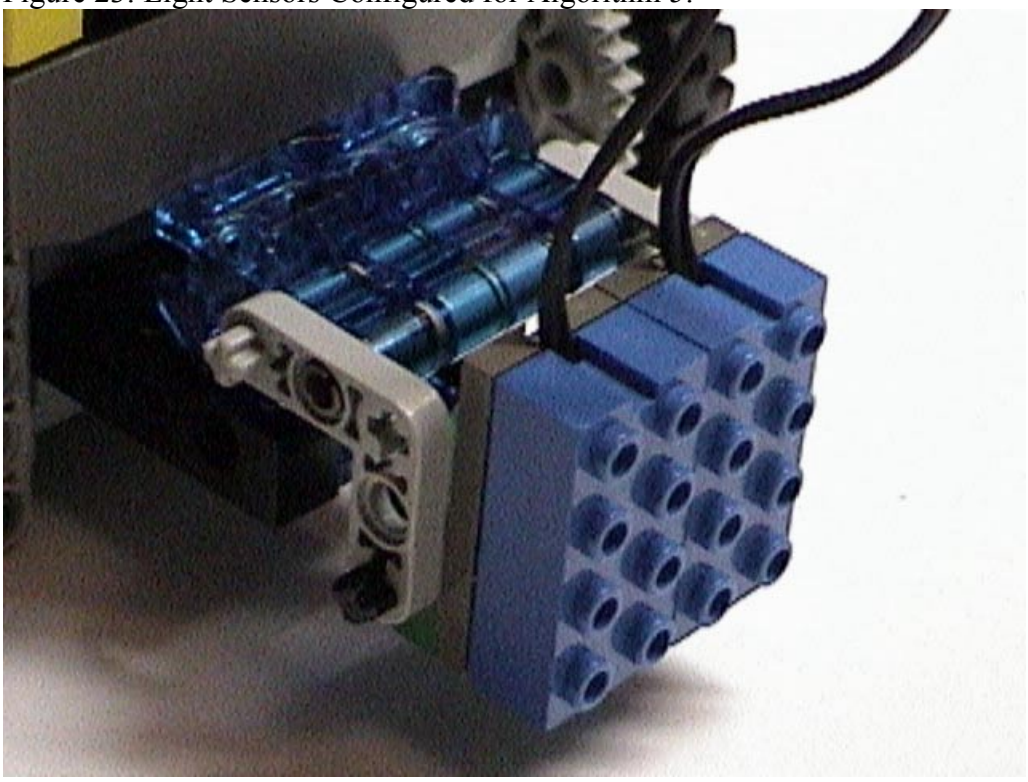
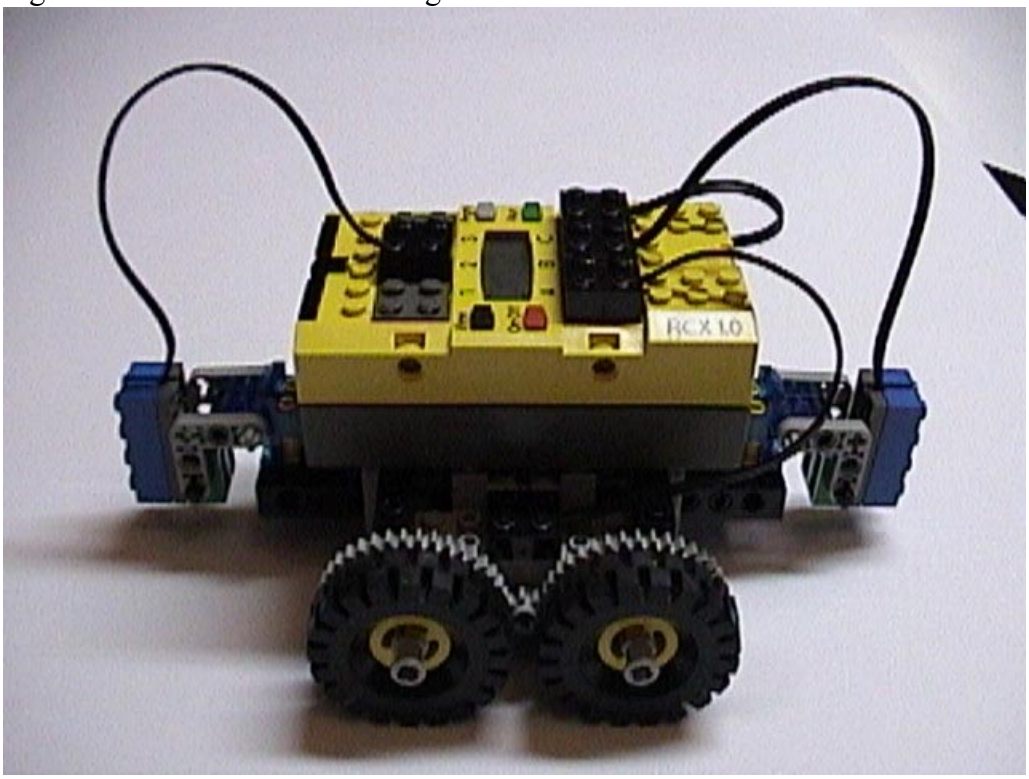


Figure 24: Roverbot with One Light Sensor in Front and One Behind:



Works Cited

Baum, Dave. Definitive Guide to LEGO MINDSTORMS: Second Edition. New York: Apress, 2003.

Erwin, Benjamin. Creative Projects with LEGO Mindstorms. New York: Addison, 2001.

Ferrari, Mario and Giulio Ferrari. Building Robots with LEGO MINDSTORMS: The ULTIMATE Tool for Mindstorms Maniacs. Rockland: Syngress, 2002.

Knudsen, Jonathan B. The Unofficial Guide to LEGO MINDSTORMS Robots. Cambridge: O'Reilly, 1999

Robotics Invention System 2.0 Constructopedia. LEGO, 2000.